YASKAWA

VIPA SPEED7 Library

OPL_SP7-LIB | SW90MS0MA V10.017 | Manual

HB00 | OPL_SP7-LIB | SW90MS0MA V10.017 | en | 20-18

Block library - Simple Motion Control

YASKAWA Europe GmbH
Ohmstraße 4
91074 Herzogenaurach
Tel.: +49 9132 744 0
Fax: +49 9132 744 186
Email: info@yaskawa.eu.com
Internet: www.yaskawa.eu.com

Table of contents

1 General

1.1 Copyright © YASKAWA Europe GmbH

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 VIPA block library *'Simple Motion Control'*:

- 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.

Supplementary information and useful tips.

2 Overview

Block library *'Simple Motion Control'*

The block library can be found for download in the *'Service/Support'* area of www.vipa.com at *'Downloads ➔ VIPA Lib'* as *'Block library Simple Motion Control - SW90MS0MA'*. The library is available as packed zip file. As soon as you want to use these blocks you have to import them into your project.

> *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
SimpleMotion_S7_V0039.zip	■ Block library for Siemens SIMATIC Manager. ■ For use in VIPA CPUs or S7-300 CPUs from Siemens.
SimpleMotion_TIA_V0025.zip	■ Block library for Siemens TIA Portal V14/V15. ■ For use in VIPA CPUs or S7-300 CPUs from Siemens.
SimpleMotion_Movicon0007.zip	Symbol library for Movicon
Demo_S7_Movicon_V0023.zip	■ Demo project for Siemens SIMATIC Manager and Movicon. ■ For use in VIPA CPUs and TouchPanels or S7-300 CPUs from Siemens.
Demo_TIA_Movicon_V0018.zip	■ Demo project for Siemens TIA Portal V14 and Movicon. ■ For use in VIPA CPUs and TouchPanels or S7-300 CPUs from Siemens.

Properties

With the *Simple Motion Control Library* blocks, you can easily integrate drives into your applications without detailed knowledge. Here various drives and bus systems are supported. The PLCopen blocks enable you to implement simple drive tasks in your control system. This system offers the following features:

- Can be used in VIPA *SPEED7 Studio* and Siemens SIMATIC Manager
- Implementation of simple drive functions
  - Switch on or off
  - Speed setting
  - Relative or absolute positioning
  - Homing
  - Read and write parameters
  - Query of axis position and status
- Easy commissioning and diagnostics without detailed knowledge of the drives
- Support of various drives and field buses
- Visualization of individual axes
- Scalable by using PLCopen blocks

Structure

The *Simple Motion Control Library* is divided into the following groups:

- Axis Control
  - General blocks for controlling the drives.
- Sigma5 EtherCAT
  - Specific blocks for the use of *Sigma-5* drives, which are connected via EtherCAT.
- Sigma7 EtherCAT
  - Specific blocks for the use of *Sigma-7S* drives, which are connected via EtherCAT.
  - Specific blocks for the use of *Sigma-7W* drives, which are connected via EtherCAT.
- Sigma5+7 PROFINET
  - Specific blocks for the use of *Sigma-5* respectively *Sigma-7* drives, which are connected via PROFINET.
- Sigma5+7 PulseTrain
  - Specific block for the use of *Sigma-5* respectively *Sigma-7* drives, which are connected via Pulse Train.
- V1000 PWM
  - Specific block for the use of *V1000* inverter drives, which are connected via PWM.
- V1000 Modbus RTU
  - Specific blocks for the use of *V1000* inverter drives, which are connected via Modbus RTU.
- Inverter EtherCAT
  - Specific block for the use of inverter drives, which are connected via EtherCAT.

3 Usage *Sigma-5/7* EtherCAT

3.1 Usage *Sigma-5* EtherCAT

3.1.1 Overview

Precondition

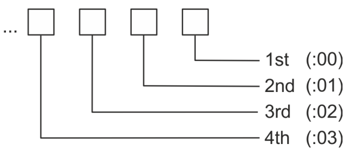
- SPEED7 Studio from V1.6.1
  or
- Siemens SIMATIC Manager from V 5.5, SP2 & *SPEED7 EtherCAT Manager* & *Simple Motion Control Library*
- CPU with EtherCAT master, e.g. CPU 015-CEFNR00
- *Sigma-5* drive with EtherCAT option card

Steps of configuration

1. Set the parameters on the drive
   - The setting of the parameters happens by means of the software tool *Sigma Win+*.
2. Hardware configuration in VIPA *SPEED7 Studio* or Siemens SIMATIC Manager
   - Configuring a CPU with EtherCAT master functionality.
   - Configuration of a *Sigma-5* EtherCAT drive.
   - Configuring the EtherCAT connection via *SPEED7 EtherCAT Manager*.
3. Programming in VIPA *SPEED7 Studio* or Siemens SIMATIC Manager
   - Connecting the *Init* block to configure the axis.
   - Connecting the *Kernel* block to communicate with the axis.
   - Connecting the blocks for the motion sequences.

3.1.2 Set the parameters on the drive

Parameter digits



CAUTION!
Before the commissioning, you have to adapt your drive to your application with the *Sigma Win+* software tool! More may be found in the manual of your drive.

The following parameters must be set via *Sigma Win+* to match the *Simple Motion Control Library*:

Sigma-5 (20bit encoder)

Servopack Parameter	Address:digit	Name	Value
Pn205	(2205h)	Multiturn Limit Setting	65535
Pn20E	(220Eh)	Electronic Gear Ratio (Numerator)	1
Pn210	(2210h)	Electronic Gear Ratio (Denominator)	1
PnB02	(2701h:01)	Position User Unit (Numerator)	1
PnB04	(2701h:02)	Position User Unit (Denominator)	1
PnB06	(2702h:01)	Velocity User Unit (Numerator)	1
PnB08	(2702h:02)	Velocity User Unit (Denominator)	1

Servopack Parameter	Address:digit	Name	Value
PnB0A	(2703h:01)	Acceleration User Unit (Numerator)	1
PnB0C	(2703h:02)	Acceleration User Unit (Denominator)	1

Please note that you have to enable the corresponding direction of your axis in accordance to your requirements. For this use the parameters Pn50A (P-OT) respectively Pn50B (N-OT) in Sigma Win+.

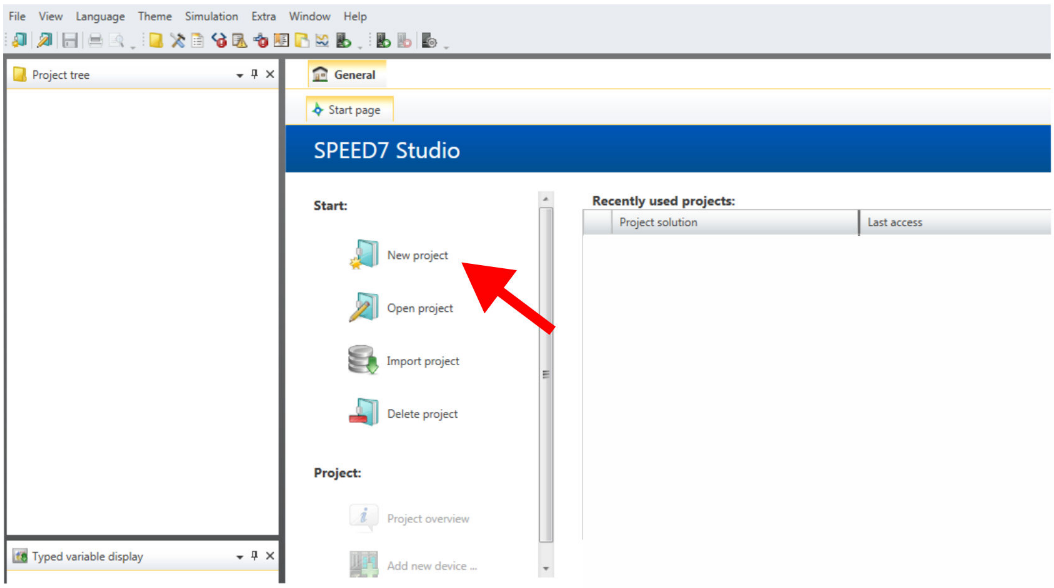
3.1.3 Usage in VIPA *SPEED7 Studio*

3.1.3.1 Hardware configuration

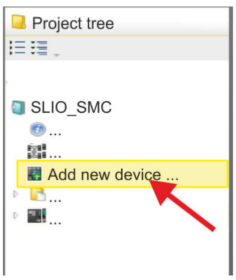
Add CPU in the project

Please use for configuration the *SPEED7 Studio* V1.6.1 and up.

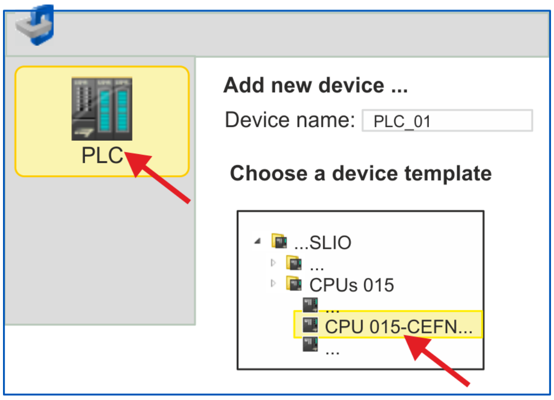
1. Start the *SPEED7 Studio*.



2. Create a new project at the start page with *'New project'*.
   ⇨ A new project is created and the view *'Devices and networking'* is shown.



3. Click in the *Project tree* at *'Add new device ...'*.



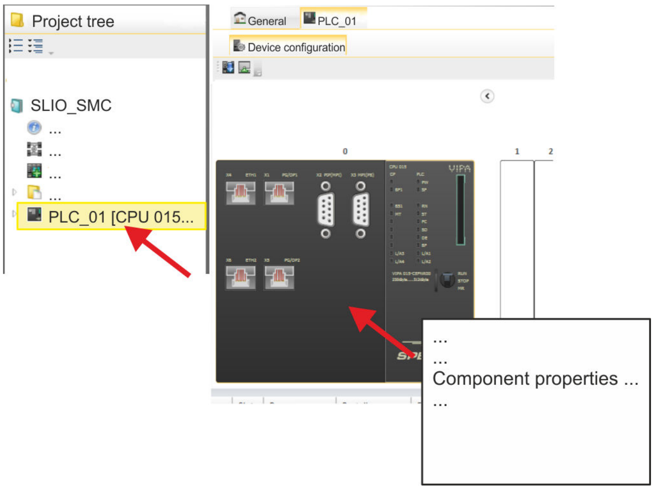
⇨ A dialog for device selection opens.

4. Select from the *'Device templates'* a CPU with EtherCAT master functions such as CPU 015-CEFNR00 and click at [OK].

   ⇨ The CPU is inserted in *'Devices and networking'* and the *'Device configuration'* is opened.

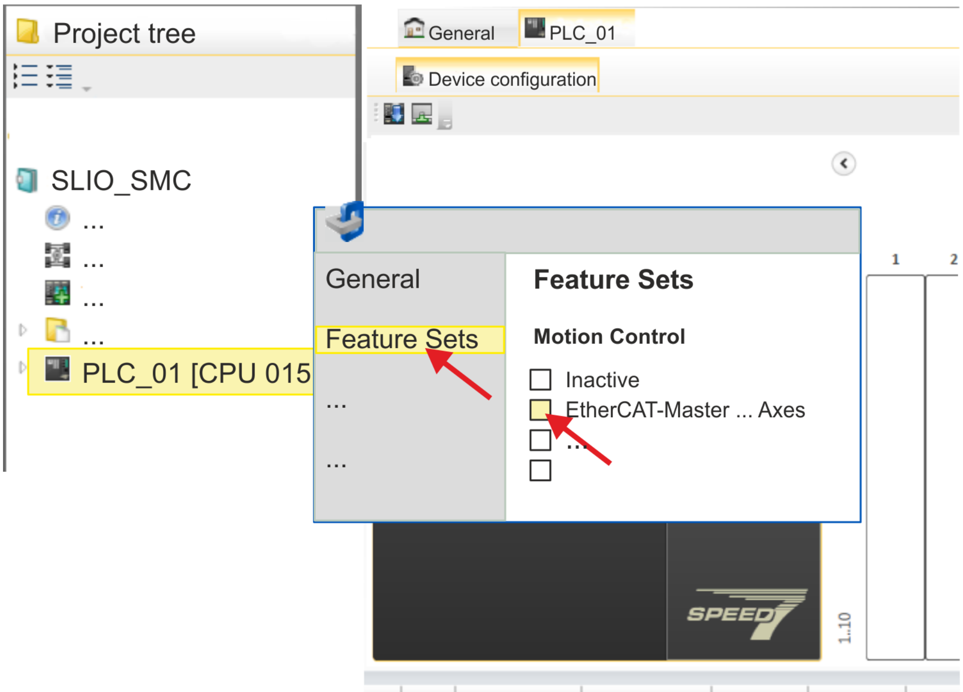
Activate motion control functions

If the EtherCAT master functionality is not yet activated on your CPU, the activation takes place as follows:



1. Click at the CPU in the *'Device configuration'* and select *'Context menu ➔ Components properties'*.

   ⇨ The properties dialog of the CPU is opened.



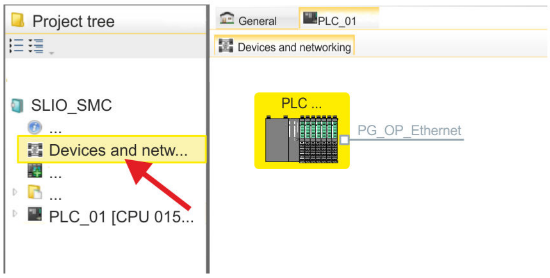
2. Click at *'Feature Sets'* and activate at *'Motion Control'* the parameter *'EtherCAT-Master... Axes'*. The number of axes is not relevant in this example.

3. Confirm your input with [OK].
   ⇨ The motion control functions are now available in your project.

> **CAUTION!**
> Please note due to the system, with every change to the feature set settings, the EtherCAT field bus system and its motion control configuration will be deleted from your project!

Configuration of Ethernet PG/OP channel

1. Click in the *Project tree* at *'Devices and networking'*.
   ⇨ You will get a graphical object view of your CPU.



2. Click at the network *'PG_OP_Ethernet'*.
3. Select *'Context menu ➔ Interface properties'*.
   ⇨ A dialog window opens. Here you can enter the IP address data for your Ethernet PG/OP channel. You get valid IP address parameters from your system administrator.
4. Confirm with [OK].
   ⇨ The IP address data are stored in your project listed in *'Devices and networking'* at *'Local components'*.

   After transferring your project your CPU can be accessed via Ethernet PG/OP channel with the set IP address data.

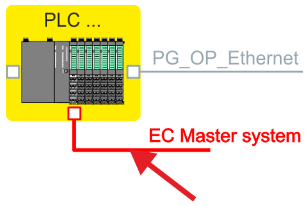
Installing the ESI file

For the *Sigma-5* EtherCAT drive can be configured in the *SPEED7 EtherCAT Manager*, the corresponding ESI file must be installed. Usually, the *SPEED7 Studio* is delivered with current ESI files and you can skip this part. If your ESI file is not up-to date, you will find the latest ESI file for the *Sigma-5* EtherCAT drive under www.yaskawa.eu.com at *'Service ➔ Drives & Motion Software'*.

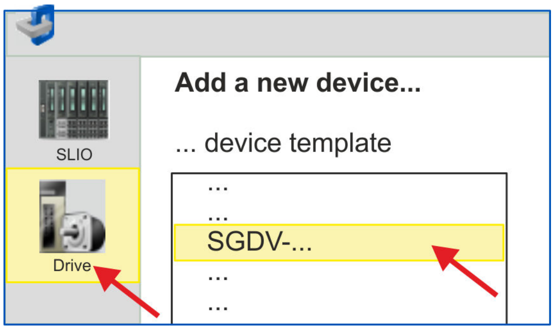
1. Download the according ESI file for your drive. Unzip this if necessary.
2. Navigate to your *SPEED7 Studio*.
3. Open the corresponding dialog window by clicking on *'Extras ➔ Install device description (EtherCAT - ESI)'*.
4. Under *'Source path'*, specify the ESI file and install it with [Install].
   ⇨ The devices of the ESI file are now available.

Add a *Sigma-5* drive

1. Click in the Project tree at *'Devices and networking'*.
2. Click here at *'EC-Mastersystem'* and select *'Context menu → Add new device'*.



⇨ The device template for selecting an EtherCAT device opens.

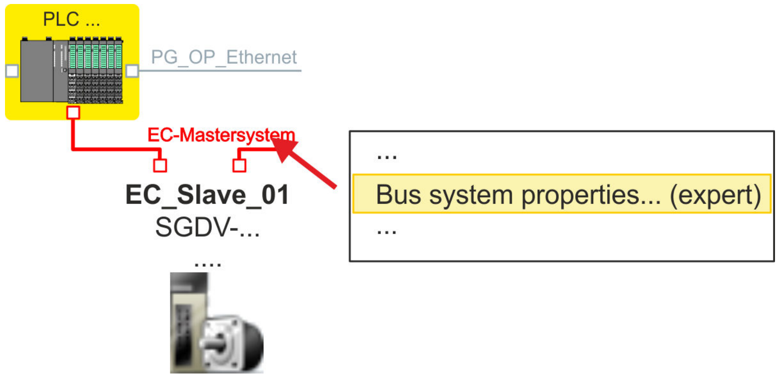


3. Select your *Sigma-5* drive:
   - SGDV-xxxxE5...
   - SGDV-xxxxE1...

   Confirm with [OK]. If your drive does not exist, you must install the corresponding ESI file as described above.

⇨ The *Sigma-5* drive is connected to your EC-Mastersystem.

Configure *Sigma-5* drive

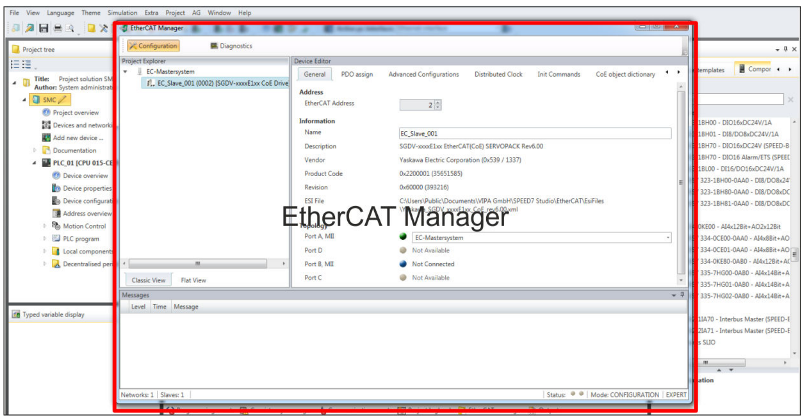


1. Click here at *'EC-Mastersystem'* and select *'Context menu ➔ Bus system properties (expert)'*.

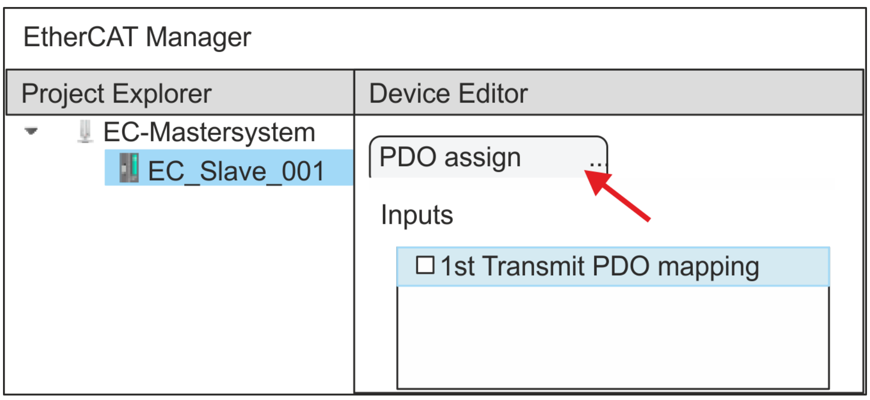
> *You can only edit PDOs in 'Expert mode'! Otherwise, the buttons are hidden.*

⇨ The *SPEED7 EtherCAT Manager* opens. Here you can configure the EtherCAT communication to your *Sigma-5* drive.

More information about the usage of the *SPEED7 EtherCAT Manager* may be found in the online help of the *SPEED7 Studio*.



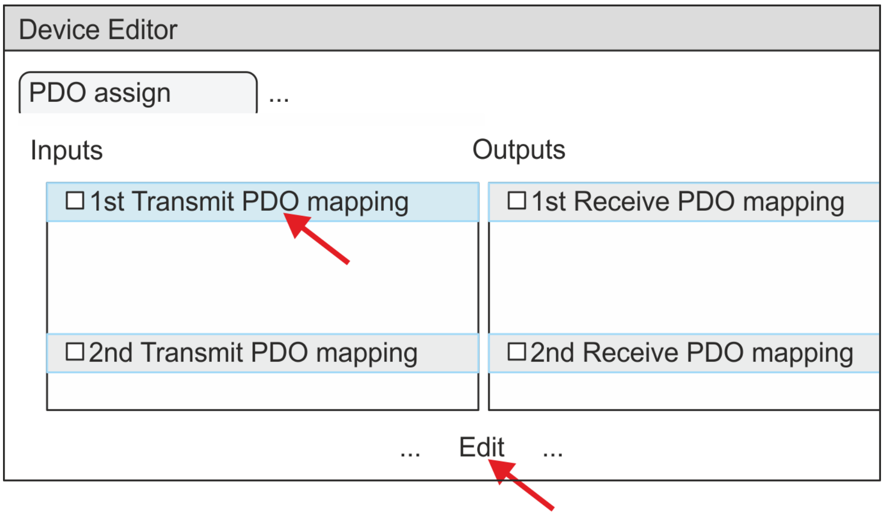
2. Click on the slave in the *SPEED7 EtherCAT Manager* and select the *'PDO assign'* tab in the *'Device editor'*.



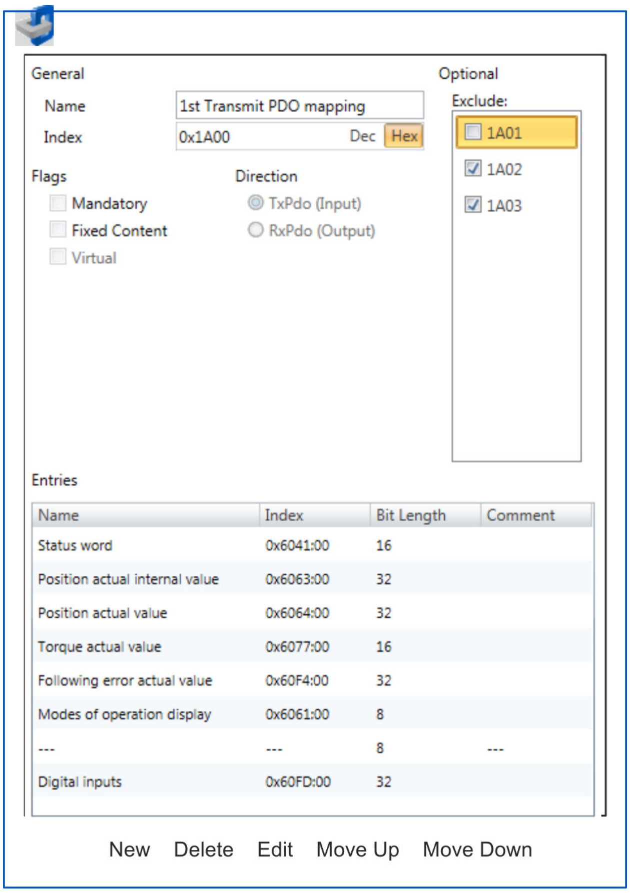
⇨ This dialog shows a list of the PDOs.

3. By selecting the appropriate mapping, you can edit the PDOs with [Edit]. Select the mapping *'1st Transmit PDO mapping'* and click at [Edit].

> *Please note that some PDOs can not be edited because of the default settings. By de-activating already activated PDOs, you can release the processing of locked PDOs.*



⇨ The dialog *'Edit PDO'* is opened. Please check the PDO settings listed here and adjust them if necessary. Please also take into account the order of the *'Entries'* and add them accordingly.



The following functions are available for editing the *'Entries'*:

- New
  – Here you can create a new entry in a dialog by selecting the corresponding entry from the *'CoE object dictionary'* and making your settings. The entry is accepted with [OK] and is listed in the list of entries.
- Delete
  – This allows you to delete a selected entry.
- Edit
  – This allows you to edit the general data of an entry.
- Move Up/Down
  – This allows you to move the selected entry up or down in the list.

4. Perform the following settings:

Inputs: 1st Transmit PDO 0x1A00

- General
  – Name: 1st Transmit PDO mapping
  – Index: 0x1A00
- Flags
  – Everything de-activated
- Direction
  – TxPdo (Input): activated
- Exclude
  Please note these settings, otherwise the PDO mappings can not be activated at the same time!
  – 1A01: de-activated
- Entries

Name	Index	Bit length
Status word	0x6041:00	16bit
Position actual internal value	0x6063:00	32bit
Position actual value	0x6064:00	32bit
Torque actual value	0x6077:00	16bit
Following error actual value	0x60F4:00	32bit
Modes of operation display	0x6061:00	8bit
---	---	8bit
Digital inputs	0x60FD:00	32bit

Close the dialog *'Edit PDO'* with [OK].

5. Select the mapping *'2nd Transmit PDO mapping'* and click at [Edit]. Perform the following settings:

Inputs: 2nd Transmit PDO 0x1A01

- General
  - Name: 2nd Transmit PDO mapping
  - Index: 0x1A01
- Flags
  - Everything de-activated
- Direction
  - TxPdo (Input): activated
- Exclude
  Please note these settings, otherwise the PDO mappings can not be activated at the same time!
  - 1A00: de-activated
  - 1A02: de-activated
  - 1A03: de-activated
- Entries

Name	Index	Bit length
Touch probe status	0x60B9:00	16bit
Touch probe 1 position value	0x60BA:00	32bit
Touch probe 2 position value	0x60BC:00	32bit
Velocity actual value	0x606C:00	32bit

Close the dialog *'Edit PDO'* with [OK].

6. Select the mapping *'1st Receive PDO mapping'* and click at [Edit]. Perform the following settings:

Outputs: 1st Receive PDO 0x1600

- General
  - Name: 1st Receive PDO mapping
  - Index: 0x1600
- Flags
  - Everything de-activated
- Direction
  - RxPdo (Output): activated
- Exclude
  Please note these settings, otherwise the PDO mappings can not be activated at the same time!
  - 1601: de-activated
  - 1602: de-activated
  - 1603: de-activated
- Entries

Name	Index	Bit length
Control word	0x6040:00	16bit
Target position	0x607A:00	32bit
Target velocity	0x60FF:00	32bit
Modes of operation	0x6060:00	8bit
---	---	8bit
Touch probe function	0x60B8:00	16bit

Close the dialog *'Edit PDO'* with [OK].

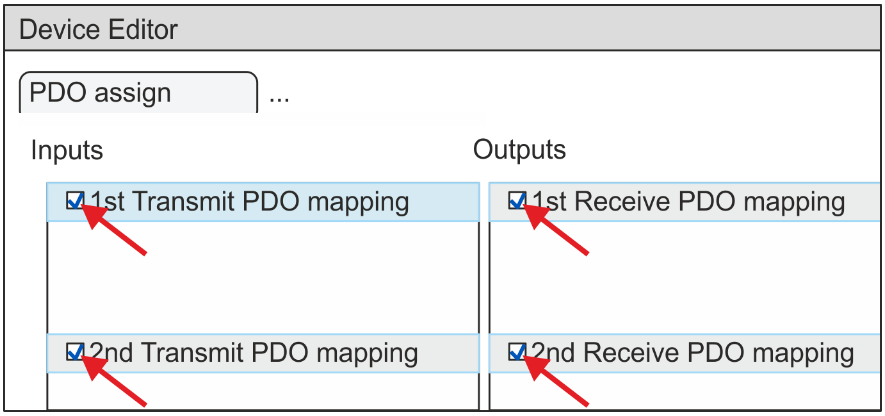
7. Select the mapping *'2nd ReceivePDO mapping'* and click at [Edit]. Perform the following settings:

Outputs: 2nd Receive PDO 0x1601

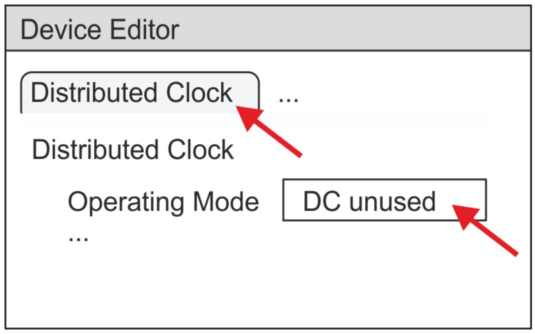
- General
  - Name: 2nd Receive PDO mapping
  - Index: 0x1601
- Flags
  - Everything de-activated
- Direction
  - RxPdo (Output): activated
- Exclude
  Please note these settings, otherwise the PDO mappings can not be activated at the same time!
  - 1600: de-activated
  - 1602: activated
  - 1603: activated
- Entries
  - Profile velocity: 0x6081:00 → 32 Bit
  - Profile acceleration: 0x6083:00 → 32 Bit
  - Profile deceleration: 0x6084:00 → 32 Bit

Close the dialog *'Edit PDO'* with [OK].

8. In PDO assignment, activate the PDOs 1 and 2 for the inputs and outputs. All subsequent PDOs must remain de-activated. If this is not possible, please check the respective PDO parameter *'Exclude'*.

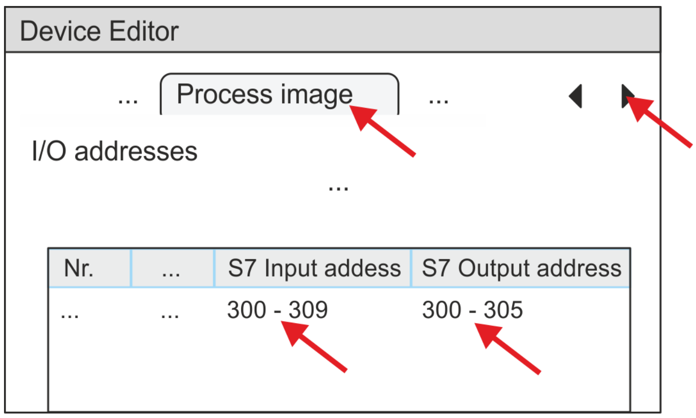


9. In the *'Device Editor'* of the *SPEED7 EtherCAT Manager*, select the *'Distributed clocks'* tab and set *'DC unused'* as *'Operating mode'*.



10. Select the *'Process image'* tab via the arrow key in the *'Device editor'* and note for the parameter of the block FB 871 - VMC_InitSigma5_EC the following PDO.

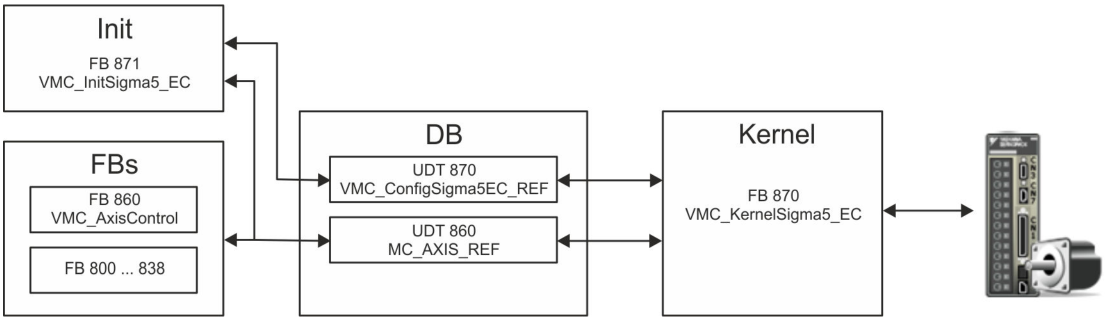
- *'S7 Input address'* → *'InputsStartAddressPDO'*
- *'S7 Output address'* → *'OutputsStartAddressPDO'*



11. By closing the dialog of the *SPEED7 EtherCAT Manager* with [X] the configuration is taken to the *SPEED7 Studio*.

3.1.3.2 User program

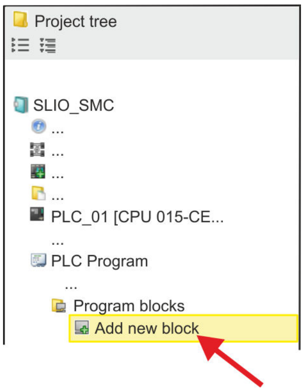
3.1.3.2.1 Program structure



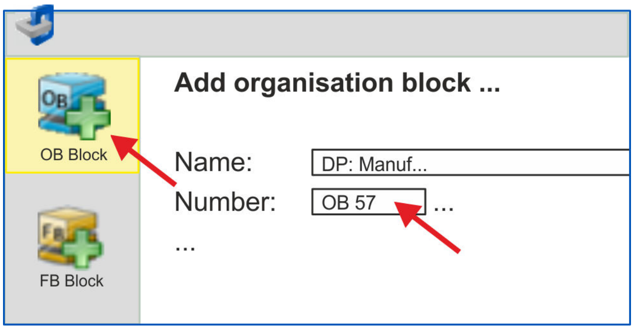
- DB
  A data block (axis DB) for configuration and status data must be created for each axis of a drive. The data block consists of the following data structures:
  – UDT 870 - *VMC_Config*Sigma5EC_REF
    The data structure describes the structure of the configuration of the drive.
    Specific data structure for *Sigma-5* EtherCAT.
  – UDT 860 - *MC_AXIS_REF*
    The data structure describes the structure of the parameters and status information of drives.
    General data structure for all drives and bus systems.
- FB 871 - *VMC_Init*Sigma5_EC
  – The *Init*t block is used to configure an axis.
  – Specific block for *Sigma-5* EtherCAT.
  – The configuration data for the initialization must be stored in the *axis DB*.
- FB 870 - *VMC_Kernel*Sigma5_EC
  – The *Kernel* block communicates with the drive via the appropriate bus system, processes the user requests and returns status messages.
  – Specific block for *Sigma-5* EtherCAT.
  – The exchange of the data takes place by means of the *axis DB*.
- FB 860 - *VMC_AxisControl*
  – General block for all drives and bus systems.
  – Supports simple motion commands and returns all relevant status messages.
  – The exchange of the data takes place by means of the *axis DB*.
  – For motion control and status query, via the instance data of the block you can link a visualization.
  – In addition to the FB 860 - *VMC_AxisControl*, *PLCopen* blocks can be used.
- FB 800 ... FB 838 - *PLCopen*
  – The PLCopen blocks are used to program motion sequences and status queries.
  – General blocks for all drives and bus systems.

3.1.3.2.2 Programming

Copy blocks into project

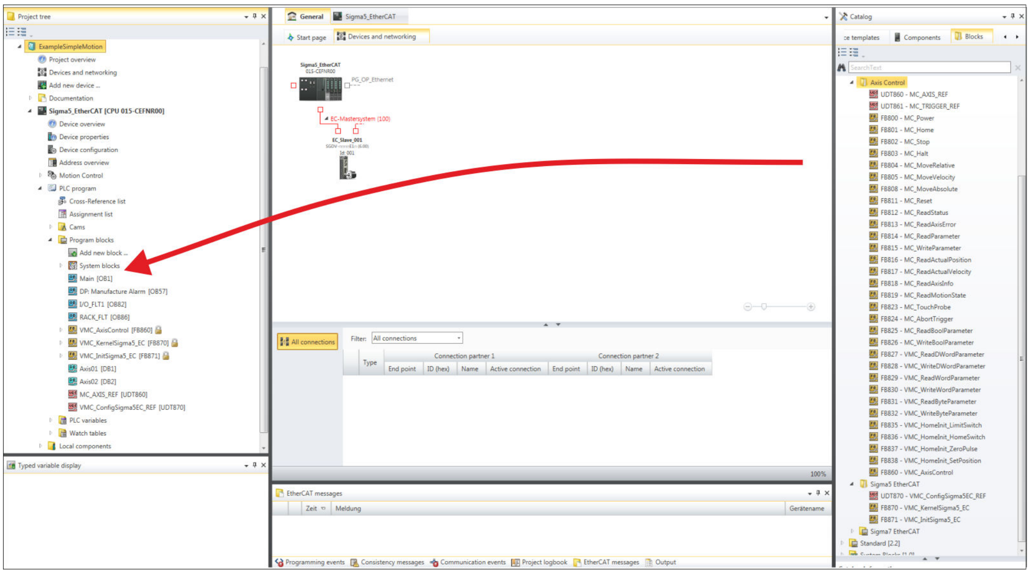


1. Click in the *Project tree* within the CPU at *'PLC program'*, *'Program blocks'* at *'Add New block'*.



⇨ The dialog *'Add block'* is opened.

2. Select the block type *'OB block'* and add OB 57, OB 82 and OB 86 to your project.



3. In the *'Catalog'*, open the *'Simple Motion Control'* library at *'Blocks'* and drag and drop the following blocks into *'Program blocks'* of the *Project tree*:
   - *Sigma-5* EtherCAT:
     – UDT 870 - VMC_ConfigSigma5EC_REF
     – FB 870 - VMC_KernelSigma5_EC
     – FB 871 - VMC_InitSigma5_EC
   - Axis Control
     – UDT 860 - MC_AXIS_REF
     – Blocks for your movement sequences

Create axis DB

1. Add a new DB as your *axis DB* to your project. Click in the *Project tree* within the CPU at *'PLC program'*, *'Program blocks'* at *'Add New block'*, select the block type *'DB block'* and assign the name "Axis01" to it. The DB number can freely be selected such as DB 10.

⇨ The block is created and opened.

2. ■ In "Axis01", create the variable "Config" of type UDT 870. These are specific axis configuration data.
■ In "Axis01", create the variable "Axis" of type UDT 860. During operation, all operating data of the axis are stored here.

Axis01 [DB10]
Data block structure

	Adr...	Name	Data type	...
	...	Config	UDT	[870]
	...	Axis	UDT	[860]

OB 1

Configuration of the axis

Open OB 1 and program the following FB calls with associated DBs:

FB 871 - VMC_InitSigma5_EC, DB 871 *Chap. 3.1.5.3 'FB 871 - VMC_InitSigma5_EC - Sigma-5 EtherCAT initialization' page 45*

At *InputsStartAddressPDO* respectively *OutputsStartAddressPDO*, enter the address from the *SPEED7 EtherCAT Manager*. *22*

```
⇨ CALL  "VMC_InitSigma5_EC" , "DI_InitSgm5ETC01"
  Enable                :="InitS5EC1_Enable"
  LogicalAddress        :=300
  InputsStartAddressPDO :=300(EtherCAT-Man.:S7 Input address)
  OutputsStartAddressPDO:=300(EtherCAT-Man.:S7 Output
  address)
  EncoderType           :=1
  EncoderResolutionBits :=20
  FactorPosition        :=1.048576e+006
  FactorVelocity        :=1.048576e+006
  FactorAcceleration    :=1.048576e+002
  OffsetPosition        :=0.000000e+000
  MaxVelocityApp        :=5.000000e+001
  MaxAccelerationApp    :=1.000000e+002
  MaxDecelerationApp    :=1.000000e+002
  MaxVelocityDrive      :=6.000000e+001
  MaxAccelerationDrive  :=1.500000e+002
  MaxDecelerationDrive  :=1.500000e+002
  MaxPosition           :=1.048500e+003
  MinPosition           :=-1.048514e+003
  EnableMaxPosition     :=TRUE
  EnableMinPosition     :=TRUE
  MinUserPosition       :="InitS5EC1_MinUserPos"
  MaxUserPosition       :="InitS5EC1_MaxUserPos"
  Valid                 :="InitS5EC1_Valid"
  Error                 :="InitS5EC1_Error"
  ErrorID               :="InitS5EC1_ErrorID"
  Config                :="Axis01".Config
  Axis                  :="Axis01".Axis
```

Connecting the Kernel for the axis

The *Kernel* processes the user commands and passes them appropriately processed on to the drive via the respective bus system.

FB 870 - VMC_KernelSigma5_EC, DB 870 *Chap. 3.1.5.2 'FB 870 - VMC_KernelSigma5_EC - Sigma-5 EtherCAT Kernel' page 45*

```
⇨      CALL  "VMC_KernelSigma5_EC" , "DI_KernelSgm5ETC01"
        Init  :="KernelS5EC1_Init"
        Config:="Axis01".Config
        Axis  :="Axis01".Axis
```

Connecting the block for motion sequences

For simplicity, the connection of the FB 860 - VMC_AxisControl is to be shown here. This universal block supports simple motion commands and returns status messages. The inputs and outputs can be individually connected. Please specify the reference to the corresponding axis data at *'Axis'* in the *axis DB*.

1. FB 860 - VMC_AxisControl, DB 860 *Chap. 9.2.2 'FB 860 - VMC_AxisControl - Control block axis control' page 363*

   ⇨
```
CALL  "VMC_AxisControl" , "DI_AxisControl01"
 AxisEnable        :="AxCtrl1_AxisEnable"
 AxisReset         :="AxCtrl1_AxisReset"
 HomeExecute       :="AxCtrl1_HomeExecute"
 HomePosition      :="AxCtrl1_HomePosition"
 StopExecute       :="AxCtrl1_StopExecute"
 MvVelocityExecute:="AxCtrl1_MvVelExecute"
 MvRelativeExecute:="AxCtrl1_MvRelExecute"
 MvAbsoluteExecute:="AxCtrl1_MvAbsExecute"
 PositionDistance :="AxCtrl1_PositionDistance"
 Velocity          :="AxCtrl1_Velocity"
 Acceleration      :="AxCtrl1_Acceleration"
 Deceleration      :="AxCtrl1_Deceleration"
 JogPositive       :="AxCtrl1_JogPositive"
 JogNegative       :="AxCtrl1_JogNegative"
 JogVelocity       :="AxCtrl1_JogVelocity"
 JogAcceleration   :="AxCtrl1_JogAcceleration"
 JogDeceleration   :="AxCtrl1_JogDeceleration"
 AxisReady         :="AxCtrl1_AxisReady"
 AxisEnabled       :="AxCtrl1_AxisEnabled"
 AxisError         :="AxCtrl1_AxisError"
 AxisErrorID       :="AxCtrl1_AxisErrorID"
 DriveWarning      :="AxCtrl1_DriveWarning"
 DriveError        :="AxCtrl1_DriveError"
 DriveErrorID      :="AxCtrl1_DriveErrorID"
 IsHomed           :="AxCtrl1_IsHomed"
 ModeOfOperation   :="AxCtrl1_ModeOfOperation"
 PLCopenState      :="AxCtrl1_PLCopenState"
 ActualPosition    :="AxCtrl1_ActualPosition"
 ActualVelocity    :="AxCtrl1_ActualVelocity"
 CmdDone           :="AxCtrl1_CmdDone"
 CmdBusy           :="AxCtrl1_CmdBusy"
 CmdAborted        :="AxCtrl1_CmdAborted"
 CmdError          :="AxCtrl1_CmdError"
 CmdErrorID        :="AxCtrl1_CmdErrorID"
 DirectionPositive:="AxCtrl1_DirectionPos"
 DirectionNegative:="AxCtrl1_DirectionNeg"
 SWLimitMinActive :="AxCtrl1_SWLimitMinActive"
 SWLimitMaxActive :="AxCtrl1_SWLimitMaxActive"
 HWLimitMinActive :="AxCtrl1_HWLimitMinActive"
 HWLimitMaxActive :="AxCtrl1_HWLimitMaxActive"
 Axis              :="Axis01".Axis
```

> *For complex motion tasks, you can use the PLCopen blocks. Please specify the reference to the corresponding axis data at Axis in the axis DB.*

Your project now includes the following blocks:

- OB 1 - Main
- OB 57 - DP Manufacturer Alarm
- OB 82 - I/O_FLT1
- OB 86 - Rack_FLT
- FB 860 - VMC_AxisControl with instance DB

- FB 870 - VMC_KernelSigma5_EC with instance DB
- FB 871 - VMC_InitSigma5_EC with instance DB
- UDT 860 - MC_Axis_REF
- UDT 870 - VMC_ConfigSigma5EC_REF

Sequence of operations

1. Select *'Project → Compile all'* and transfer the project into your CPU.

   You can find more information on the transfer of your project in the online help of the *SPEED7 Studio*.

   ⇨ 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. To do this, call the *Init* block FB 871 - VMC_InitSigma5_EC with *Enable* = TRUE.

   ⇨ The output *Valid* returns TRUE. In the event of a fault, you can determine the error by evaluating the *ErrorID*.

   You have to call the *Init* block again if you load a new axis DB or you have changed parameters on the *Init* block.

   Do not continue until the Init block does not report any errors!

3. Ensure that the *Kernel* block FB 870 - VMC_KernelSigma5_EC is cyclically called. In this way, control signals are transmitted to the drive and status messages are reported.
4. Program your application with the FB 860 - VMC_AxisControl or with the PLCopen blocks.

Controlling the drive via HMI

There is the possibility to control your drive via HMI. For this, a predefined symbol library is available for Movicon to access the VMC_AxisControl function block. *Chap. 10 'Controlling the drive via HMI' page 432*

3.1.4 Usage in Siemens SIMATIC Manager

3.1.4.1 Precondition

Overview

- Please use for configuration the Siemens SIMATIC Manager V 5.5 SP2 and up.
- 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 CPU'* is to be installed in the hardware catalog by means of the GSDML.
- The configuration of the EtherCAT masters happens in the Siemens SIMATIC Manager by means of a virtual PROFINET IO device *'EtherCAT network'*. The *'EtherCAT network'* is to be installed in the hardware catalog by means of the GSDML.
- The *'EtherCAT network'* can be configured with the VIPA Tool *SPEED7 EtherCAT Manager*.
- For the configuration of the drive in the *SPEED7 EtherCAT Manager* the installation of the according ESI file is necessary.

Installing the IO device *'VIPA SLIO System'*

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'*.

Installing the IO device EtherCAT network

The installation of the PROFINET IO devices *'EtherCAT Network'* happens in the hardware catalog with the following approach:

1. Go to the service area of www.vipa.com
2. Load from the download area at *'Config files ➔ EtherCAT'* the GSDML file for your EtherCAT master.
3. Extract the files 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 *'EtherCAT Network'* can be found at *'PROFINET IO ➔ Additional field devices ➔ I/O ➔ VIPA EtherCAT System'*.

Installing the *SPEED7 EtherCAT Manager*

The configuration of the PROFINET IO device *'EtherCAT Network'* happens by means of the VIPA *SPEED7 EtherCAT Manager*. This may be found in the service area of www.vipa.com at *'Service/Support ➔ Downloads ➔ Software'*.

The installation happens with the following proceeding:

1. Close the Siemens SIMATIC Manager.
2. Go to the service area of www.vipa.com
3. Load the *SPEED7 EtherCAT Manager* and unzip it on your PC.
4. For installation start the file EtherCATManager_v... .exe.
5. Select the language for the installation.
6. Accept the licensing agreement.
7. Select the installation directory and start the installation.
8. After installation you have to reboot your PC.

   ⇨ The *SPEED7 EtherCAT Manager* is installed and can now be called via the context menu of the Siemens SIMATIC Manager.

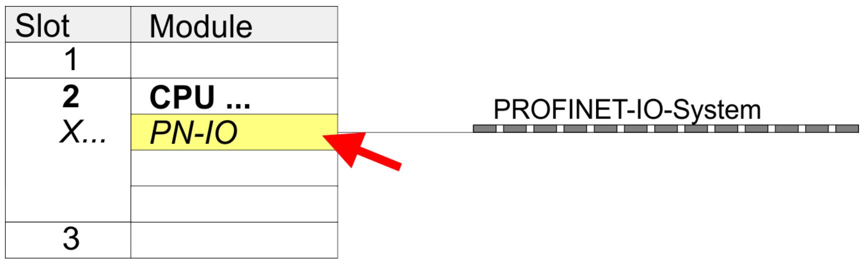
3.1.4.2 Hardware configuration

Configuring the 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	

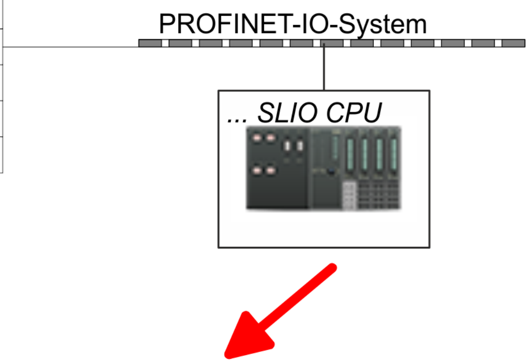
To be compatible with the Siemens SIMATIC Manager the following steps should be executed:

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 the CPU 315-2 PN/DP (315-2EH14 V3.2).
4. The integrated PROFIBUS DP master (jack X3) is to be configured and connected via the sub module *'X1 MPI/DP'*.
5. The integrated EtherCAT master is to be configured via the sub module *'X2 PN-IO'* as a virtual PROFINET network.
6. Click at the sub module *'PN-IO'* of the CPU.
7. Select *'Context menu ➔ Insert PROFINET IO System'*.



8. Create with [New] a new sub net and assign valid address data
9. Click at the sub module *'PN-IO'* of the CPU and open with *'Context menu ➔ Properties'* the properties dialog.
10. Enter at *'General'* a *'Device name'*. The device name must be unique at the Ethernet subnet.

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



Slot	Module	Order number	
0	***... SLIO CPU ...***	***015-...***	
X2	*015-...*		
1			
2			
3			
...			

11. Navigate in the hardware catalog to the directory *'PROFINET IO ➔ Additional field devices ➔ I/O ➔ VIPA SLIO System'* and connect the IO device *'015-CFFNR00 CPU'* to your PROFINET system.

⇨ In the Device 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

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

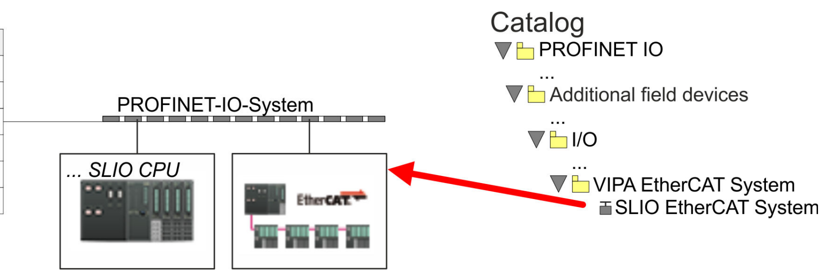
1. Place for the Ethernet PG/OP channel at slot 4 the Siemens CP 343-1 (SIMATIC 300 \ CP 300 \ Industrial Ethernet \CP 343-1 \ 6GK7 343-1EX30 0XE0 V3.0).

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!

Insert *'EtherCAT network'*

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

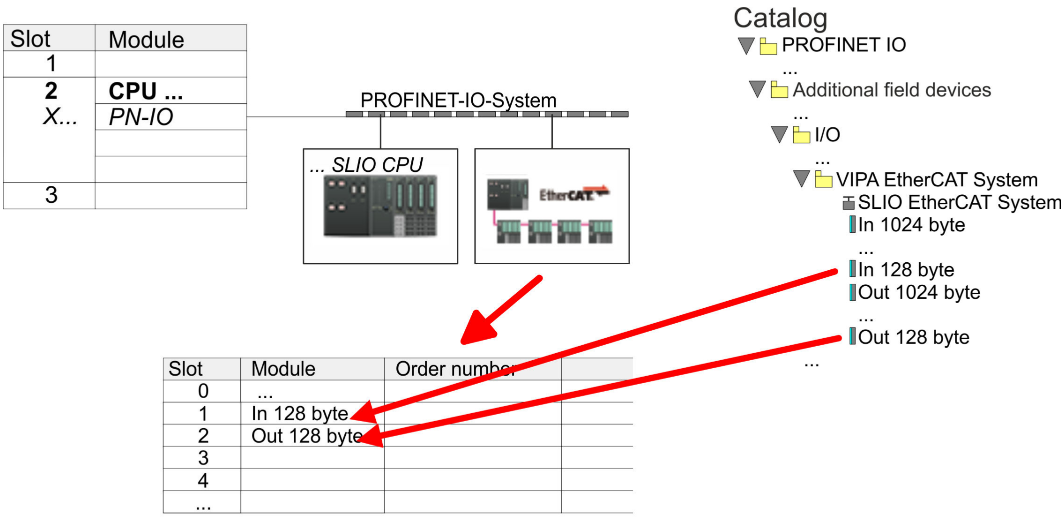


1. Navigate in the hardware catalog to the directory *'PROFINET IO ➔ Additional field devices ➔ I/O ➔ VIPA EtherCAT System'* and connect the IO device *'SLIO EtherCAT System'* to your PROFINET system.

2. Click at the inserted IO device *'EtherCAT Network'* and define the areas for in and output by drag and dropping the according *'Out'* or *'In'* area to a slot.

   Create the following areas:

   - In 128byte
   - Out 128byte

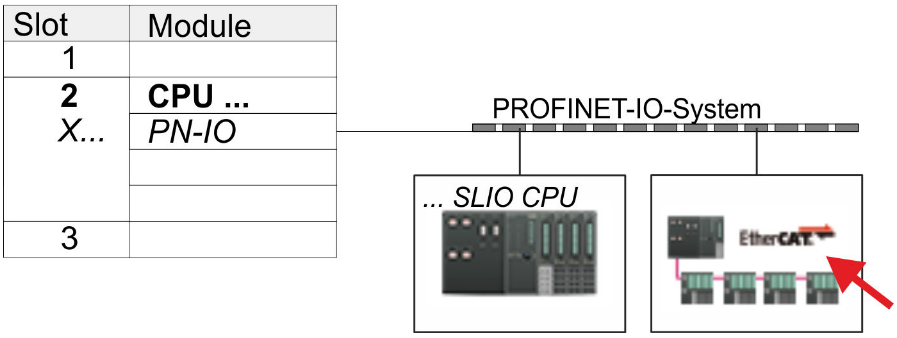


3. Select *'Station → Save and compile'*

***Sigma-5* Configure EtherCAT drive**

The drive is configured in the *SPEED7 EtherCAT Manager*.

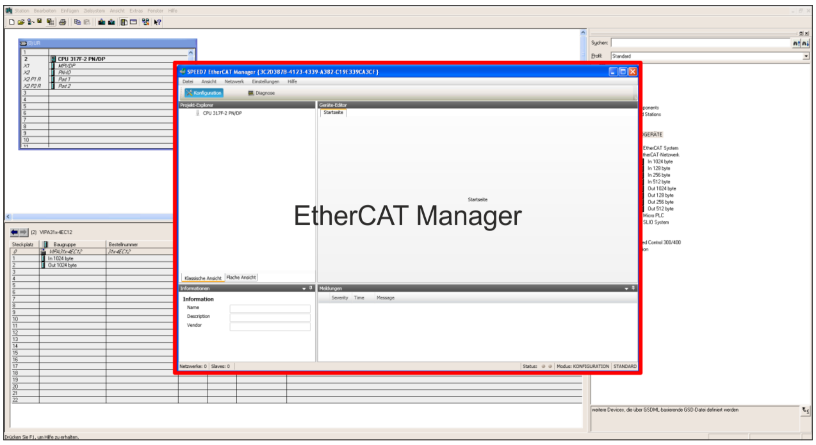
Before calling the SPEED7 EtherCAT Manager you have always to save your project with 'Station ➔ Save and compile'.



1. Click at an inserted IO device *'EtherCAT Network'* and select *'Context menu ➔ Start Device-Tool ➔ SPEED7 EtherCAT Manager'*.

   ⇨ The *SPEED7 EtherCAT Manager* opens. Here you can configure the EtherCAT communication to your *Sigma-5* drive.

   More information about the usage of the *SPEED7 EtherCAT Manager* may be found in the according manual or online help.



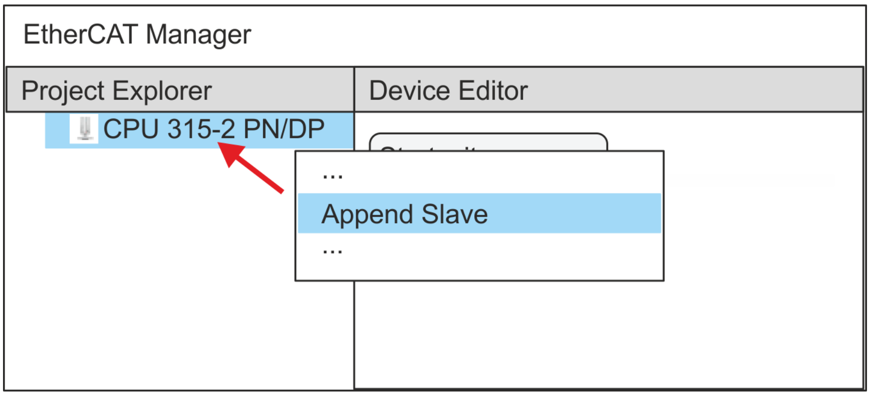
3. For the *Sigma-5* EtherCAT drive to be configured in the *SPEED7 EtherCAT Manager*, the corresponding ESI file must be installed. The ESI file for the *Sigma-5* EtherCAT drive can be found under www.yaskawa.eu.com at *'Service ➔ Drives & Motion Software'*. Download the according ESI file for your drive. Unzip this if necessary.

4. Open in the *SPEED7 EtherCAT Manager* via *'File ➔ ESI Manager'* the dialogue window *'ESI Manager'*.

5. In the *'ESI Manager'* click at [Add File] and select your ESI file. With [Open], the ESI file is installed in the *SPEED7 EtherCAT Manager*.

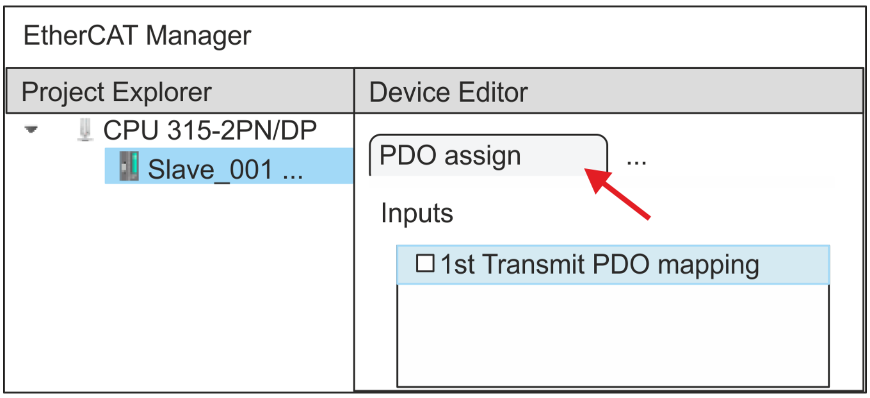
6. Close the *'ESI Manager'*.

   ⇨ Your *Sigma-5* EtherCAT drive is now available for configuration.

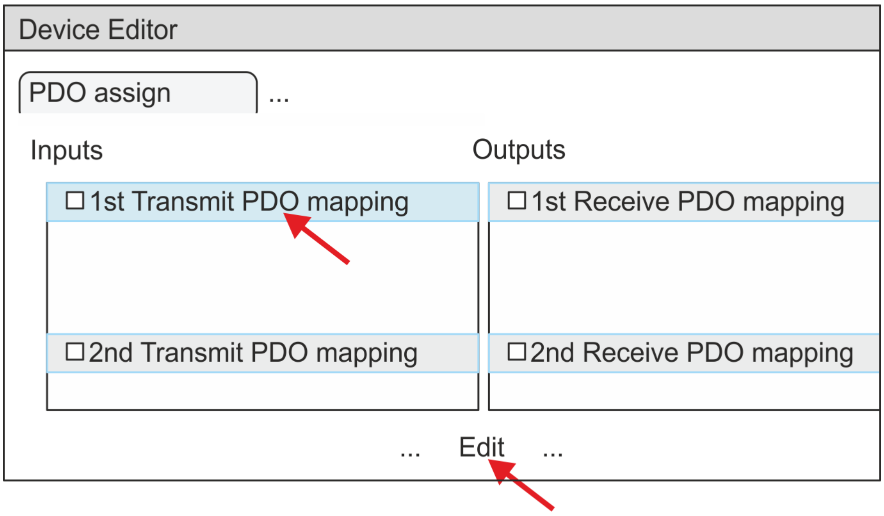


7. In the EtherCAT Manager, click on your CPU and open via *'Context menu ➔ Append Slave'* the dialog box for adding an EtherCAT slave.
   ⇨ The dialog window for selecting an EtherCAT slave is opened.
8. Select your *Sigma-5* EtherCAT drive and confirm your selection with [OK].
   ⇨ The *Sigma-5* EtherCAT drive is connected to the master and can now be configured.
9. *You can only edit PDOs in 'Expert mode'! Otherwise, the buttons are hidden. By activating the 'Expert mode' you can switch to advanced setting.*

   By activating *'View ➔ Expert'* you can switch to the *Expert mode*.
10. Click on the *Sigma-5* EtherCAT Slave in the *SPEED7 EtherCAT Manager* and select the *'PDO assign'* tab in the *'Device editor'*.

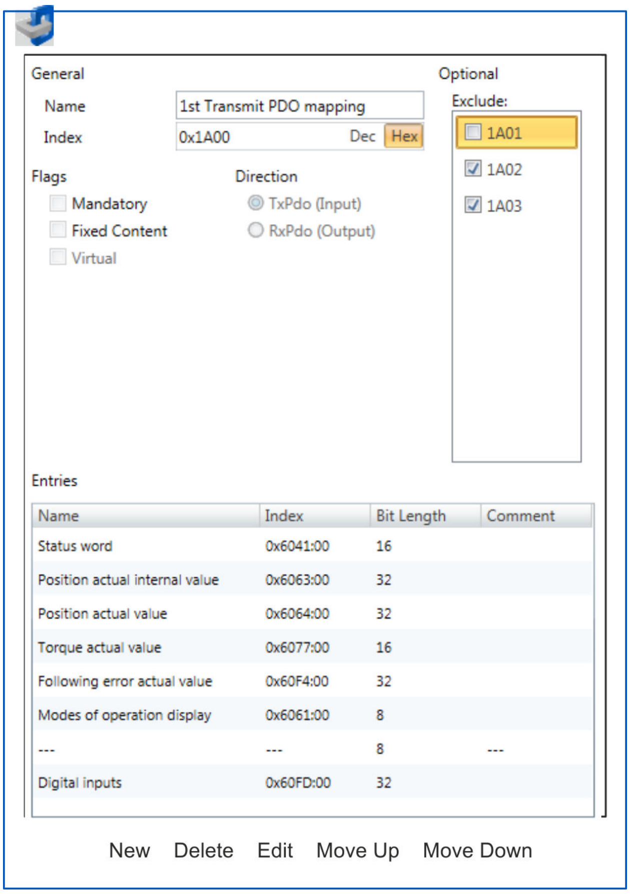


⇨ This dialog shows a list of the PDOs.



11. By selecting the appropriate PDO mapping, you can edit the PDOs with [Edit]. Select the mapping *'1st Transmit PDO mapping'* and click at [Edit].

> *Please note that some PDOs can not be edited because of the default settings. By de-activating already activated PDOs, you can release the processing of locked PDOs.*



⇨ The dialog *'Edit PDO'* is opened. Please check the PDO settings listed here and adjust them if necessary. Please also take into account the order of the *'Entries'* and add them accordingly.

The following functions are available for editing the *'Entries'*:

- New
  - Here you can create a new entry in a dialog by selecting the corresponding entry from the *'CoE object dictionary'* and making your settings. The entry is accepted with [OK] and is listed in the list of entries.
- Delete
  - This allows you to delete a selected entry.
- Edit
  - This allows you to edit the general data of an entry.
- Move Up/Down
  - This allows you to move the selected entry up or down in the list.

12. Perform the following settings:

Inputs: 1st Transmit PDO 0x1A00

- General
  - Name: 1st Transmit PDO mapping
  - Index: 0x1A00
- Flags
  - Everything de-activated
- Direction
  - TxPdo (Input): activated
- Exclude
  Please note these settings, otherwise the PDO mappings can not be activated at the same time!
  - 1A01: de-activated
- Entries

Name	Index	Bit length
Status word	0x6041:00	16bit
Position actual internal value	0x6063:00	32bit
Position actual value	0x6064:00	32bit
Torque actual value	0x6077:00	16bit
Following error actual value	0x60F4:00	32bit
Modes of operation display	0x6061:00	8bit
---	---	8bit
Digital inputs	0x60FD:00	32bit

Close the dialog *'Edit PDO'* with [OK].

13. Select the mapping *'2nd Transmit PDO mapping'* and click at [Edit]. Perform the following settings:

Inputs: 2nd Transmit PDO 0x1A01

- General
  – Name: 2nd Transmit PDO mapping
  – Index: 0x1A01
- Flags
  – Everything de-activated
- Direction
  – TxPdo (Input): activated
- Exclude
  Please note these settings, otherwise the PDO mappings can not be activated at the same time!
  – 1A00: de-activated
  – 1A02: de-activated
  – 1A03: de-activated
- Entries

Name	Index	Bit length
Touch probe status	0x60B9:00	16bit
Touch probe 1 position value	0x60BA:00	32bit
Touch probe 2 position value	0x60BC:00	32bit
Velocity actual value	0x606C:00	32bit

Close the dialog *'Edit PDO'* with [OK].

14. Select the mapping *'1st Receive PDO mapping'* and click at [Edit]. Perform the following settings:

Outputs: 1st Receive PDO 0x1600

- General
  - Name: 1st Receive PDO mapping
  - Index: 0x1600
- Flags
  - Everything de-activated
- Direction
  - RxPdo (Output): activated
- Exclude
  Please note these settings, otherwise the PDO mappings can not be activated at the same time!
  - 1601: de-activated
  - 1602: de-activated
  - 1603: de-activated
- Entries

Name	Index	Bit length
Control word	0x6040:00	16bit
Target position	0x607A:00	32bit
Target velocity	0x60FF:00	32bit
Modes of operation	0x6060:00	8bit
---	---	8bit
Touch probe function	0x60B8:00	16bit

Close the dialog *'Edit PDO'* with [OK].

15. Select the mapping *'2nd Receive PDO mapping'* and click at [Edit]. Perform the following settings:

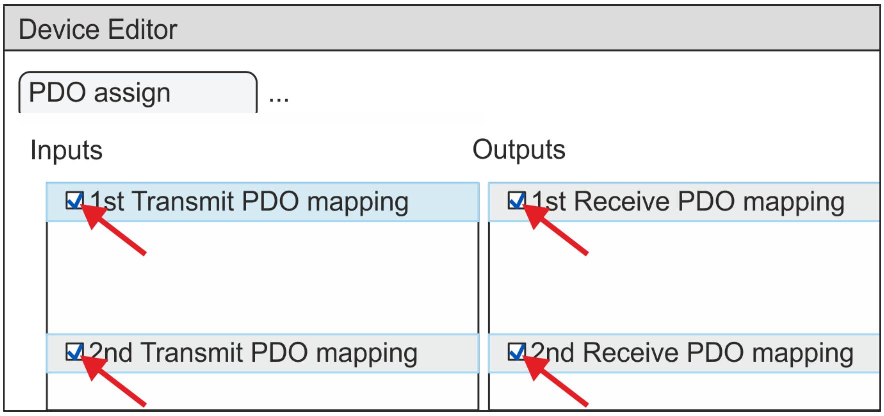
Outputs: 2nd Receive PDO 0x1601

- General
  - Name: 2nd Receive PDO mapping
  - Index: 0x1601
- Flags
  - Everything de-activated
- Direction
  - RxPdo (Output): activated
- Exclude
  Please note these settings, otherwise the PDO mappings can not be activated at the same time!
  - 1600: de-activated
  - 1602: activated
  - 1603: activated
- Entries

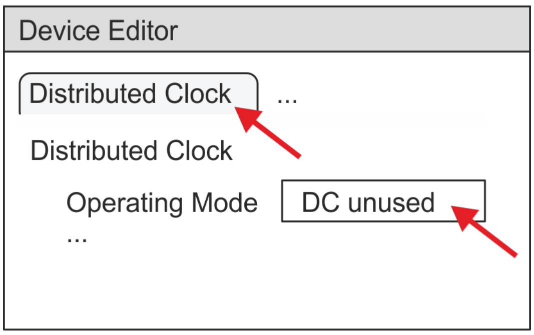
Name	Index	Bit length
Profile velocity	0x6081:00	32bit
Profile acceleration	0x6083:00	32bit
Profile deceleration	0x6084:00	32bit

Close the dialog *'Edit PDO'* with [OK].

16. In PDO assignment, activate the PDOs 1 and 2 for the inputs and outputs. All subsequent PDOs must remain de-activated. If this is not possible, please check the respective PDO parameter *'Exclude'*.

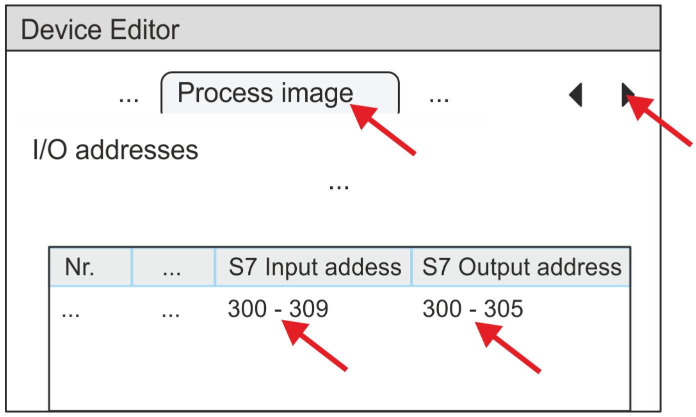


17. In the *'Device Editor'* of the *SPEED7 EtherCAT Manager*, select the *'Distributed clocks'* tab and set *'DC unused'* as *'Operating mode'*.



18. Select the *'Process image'* tab via the arrow key in the *'Device editor'* and note for the parameter of the block FB 871 - VMC_InitSigma5_EC the following PDO.

- *'S7 Input address'* → *'InputsStartAddressPDO'*
- *'S7 Output address'* → *'OutputsStartAddressPDO'*

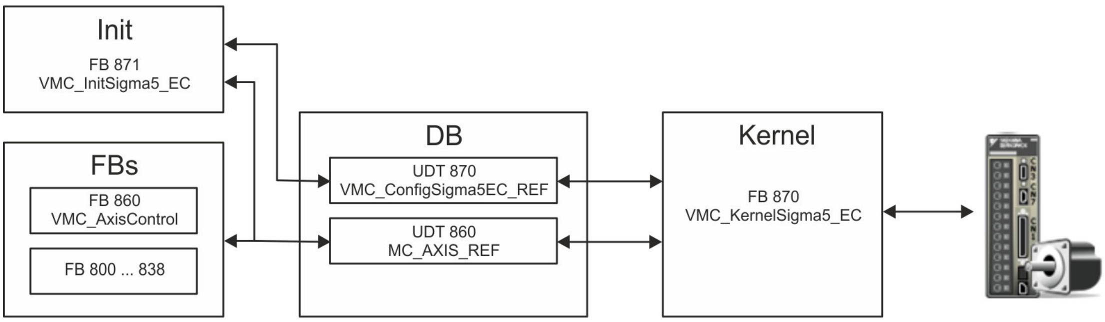


19. By closing the *SPEED7 EtherCAT Manager* with [X] the configuration is taken to the project. You can always edit your EtherCAT configuration in the *SPEED7 EtherCAT Manager*, since the configuration is stored in your project.

20. Save and compile your configuration

3.1.4.3 User program

3.1.4.3.1 Program structure



- DB
  A data block (axis DB) for configuration and status data must be created for each axis of a drive. The data block consists of the following data structures:
  – UDT 870 - *VMC_Config*Sigma5EC_REF
    The data structure describes the structure of the configuration of the drive.
    Specific data structure for *Sigma-5* EtherCAT.
  – UDT 860 - *MC_AXIS_REF*
    The data structure describes the structure of the parameters and status information of drives.
    General data structure for all drives and bus systems.
- FB 871 - *VMC_Init*Sigma5_EC
  – The *Init*t block is used to configure an axis.
  – Specific block for *Sigma-5* EtherCAT.
  – The configuration data for the initialization must be stored in the *axis DB*.

- FB 870 - *VMC_Kernel*Sigma5_EC
  - The *Kernel* block communicates with the drive via the appropriate bus system, processes the user requests and returns status messages.
  - Specific block for *Sigma-5* EtherCAT.
  - The exchange of the data takes place by means of the *axis DB*.
- FB 860 - *VMC_AxisControl*
  - General block for all drives and bus systems.
  - Supports simple motion commands and returns all relevant status messages.
  - The exchange of the data takes place by means of the *axis DB*.
  - For motion control and status query, via the instance data of the block you can link a visualization.
  - In addition to the FB 860 - *VMC_AxisControl*, *PLCopen* blocks can be used.
- FB 800 ... FB 838 - *PLCopen*
  - The PLCopen blocks are used to program motion sequences and status queries.
  - General blocks for all drives and bus systems.

3.1.4.3.2 Programming

Include library

1. Go to the service area of www.vipa.com.
2. Download the *Simple Motion Control* library from the download area at *'VIPA Lib'*.
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-5* EtherCAT:
  - UDT 870 - VMC_ConfigSigma5EC_REF
  - FB 870 - VMC_KernelSigma5_EC
  - FB 871 - VMC_InitSigma5_EC
- Axis Control
  - UDT 860 - MC_AXIS_REF
  - Blocks for your movement sequences

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.

Create axis DB

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

   Specify the following parameters:

   - Name and type
     - The DB no. as *'Name'* can freely be chosen, such as DB 10.
     - Set *'Shared DB'* as the *'Type'*.
   - Symbolic name
     - Specify "Axis01".

   Confirm your input with [OK].

   ⇨ The block is created.

2. Open DB 10 "Axis01" by double-click.

   - In "Axis01", create the variable "Config" of type UDT 870. These are specific axis configuration data.
   - In "Axis01", create the variable "Axis" of type UDT 860. During operation, all operating data of the axis are stored here.

DB10

Address	Name	Typ	...
		Struct	
...	Config	"VMC_ConfigSigma5EC_REF"	
...	Axis	"MC_AXIS_REF	
...		END_STRUCT	

OB 1

Configuration of the axis

Open OB 1 and program the following FB calls with associated DBs:

FB 871 - VMC_InitSigma5_EC, DB 871 *Chap. 3.1.5.3 'FB 871 - VMC_InitSigma5_EC - Sigma-5 EtherCAT initialization' page 45*

At *InputsStartAddressPDO* respectively *OutputsStartAddressPDO*, enter the address from the *SPEED7 EtherCAT Manager*. *38*

```
⇨ CALL  "VMC_InitSigma5_EC" , "DI_InitSgm5ETC01"
  Enable                 :="InitS5EC1_Enable"
  LogicalAddress         :=300
  InputsStartAddressPDO :=300 (EtherCAT-Man.: S7 Input
  address)
  OutputsStartAddressPDO:=300 (EtherCAT-Man.: S7 Output
  address)
  EncoderType            :=1
  EncoderResolutionBits :=20
  FactorPosition         :=1.048576e+006
  FactorVelocity         :=1.048576e+006
  FactorAcceleration     :=1.048576e+002
  OffsetPosition         :=0.000000e+000
  MaxVelocityApp         :=5.000000e+001
  MaxAccelerationApp     :=1.000000e+002
  MaxDecelerationApp     :=1.000000e+002
  MaxVelocityDrive       :=6.000000e+001
  MaxAccelerationDrive   :=1.500000e+002
  MaxDecelerationDrive   :=1.500000e+002
  MaxPosition            :=1.048500e+003
  MinPosition            :=-1.048514e+003
  EnableMaxPosition      :=TRUE
  EnableMinPosition      :=TRUE
  MinUserPosition        :="InitS5EC1_MinUserPos"
  MaxUserPosition        :="InitS5EC1_MaxUserPos"
  Valid                  :="InitS5EC1_Valid"
  Error                  :="InitS5EC1_Error"
  ErrorID                :="InitS5EC1_ErrorID"
  Config                 :="Axis01".Config
  Axis                   :="Axis01".Axis
```

Connecting the Kernel for the axis

The *Kernel* processes the user commands and passes them appropriately processed on to the drive via the respective bus system.

FB 870 - VMC_KernelSigma5_EC, DB 870 *Chap. 3.1.5.2 'FB 870 - VMC_KernelSigma5_EC - Sigma-5 EtherCAT Kernel' page 45*

```
⇨       CALL  "VMC_KernelSigma5_EC" , "DI_KernelSgm5ETC01"
         Init  :="KernelS5EC1_Init"
         Config:="Axis01".Config
         Axis  :="Axis01".Axis
```

Connecting the block for motion sequences

For simplicity, the connection of the FB 860 - VMC_AxisControl is to be shown here. This universal block supports simple motion commands and returns status messages. The inputs and outputs can be individually connected. Please specify the reference to the corresponding axis data at *'Axis'* in the *axis DB*.

FB 860 - VMC_AxisControl, DB 860 *Chap. 9.2.2 'FB 860 - VMC_AxisControl - Control block axis control' page 363*

⇨

```
CALL  "VMC_AxisControl" , "DI_AxisControl01"
 AxisEnable        :="AxCtrl1_AxisEnable"
 AxisReset         :="AxCtrl1_AxisReset"
 HomeExecute       :="AxCtrl1_HomeExecute"
 HomePosition      :="AxCtrl1_HomePosition"
 StopExecute       :="AxCtrl1_StopExecute"
 MvVelocityExecute:="AxCtrl1_MvVelExecute"
 MvRelativeExecute:="AxCtrl1_MvRelExecute"
 MvAbsoluteExecute:="AxCtrl1_MvAbsExecute"
 PositionDistance :="AxCtrl1_PositionDistance"
 Velocity          :="AxCtrl1_Velocity"
 Acceleration      :="AxCtrl1_Acceleration"
 Deceleration      :="AxCtrl1_Deceleration"
 JogPositive       :="AxCtrl1_JogPositive"
 JogNegative       :="AxCtrl1_JogNegative"
 JogVelocity       :="AxCtrl1_JogVelocity"
 JogAcceleration   :="AxCtrl1_JogAcceleration"
 JogDeceleration   :="AxCtrl1_JogDeceleration"
 AxisReady         :="AxCtrl1_AxisReady"
 AxisEnabled       :="AxCtrl1_AxisEnabled"
 AxisError         :="AxCtrl1_AxisError"
 AxisErrorID       :="AxCtrl1_AxisErrorID"
 DriveWarning      :="AxCtrl1_DriveWarning"
 DriveError        :="AxCtrl1_DriveError"
 DriveErrorID      :="AxCtrl1_DriveErrorID"
 IsHomed           :="AxCtrl1_IsHomed"
 ModeOfOperation   :="AxCtrl1_ModeOfOperation"
 PLCopenState      :="AxCtrl1_PLCopenState"
 ActualPosition    :="AxCtrl1_ActualPosition"
 ActualVelocity    :="AxCtrl1_ActualVelocity"
 CmdDone           :="AxCtrl1_CmdDone"
 CmdBusy           :="AxCtrl1_CmdBusy"
 CmdAborted        :="AxCtrl1_CmdAborted"
 CmdError          :="AxCtrl1_CmdError"
 CmdErrorID        :="AxCtrl1_CmdErrorID"
 DirectionPositive:="AxCtrl1_DirectionPos"
 DirectionNegative:="AxCtrl1_DirectionNeg"
 SWLimitMinActive :="AxCtrl1_SWLimitMinActive"
 SWLimitMaxActive :="AxCtrl1_SWLimitMaxActive"
 HWLimitMinActive :="AxCtrl1_HWLimitMinActive"
 HWLimitMaxActive :="AxCtrl1_HWLimitMaxActive"
 Axis              :="Axis01".Axis
```

For complex motion tasks, you can use the PLCopen blocks. Please specify the reference to the corresponding axis data at Axis in the axis DB.

Your project now includes the following blocks:

- OB 1 - Main
- OB 57 - DP Manufacturer Alarm
- OB 82 - I/O_FLT1
- OB 86 - Rack_FLT
- FB 860 - VMC_AxisControl with instance DB

- FB 870 - VMC_KernelSigma5_EC with instance DB
- FB 871 - VMC_InitSigma5_EC with instance DB
- UDT 860 - MC_Axis_REF
- UDT 870 - VMC_ConfigSigma5EC_REF

Sequence of operations

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

   The transfer can only be done by the Siemens SIMATIC Manager - not hardware configurator!

   Since slave and module parameters are transmitted by means of SDO respectively SDO Init command, the configuration remains active, until a power cycle is performed or new parameters for the same SDO objects are transferred.

   With an overall reset the slave and module parameters are not reset!

   ⇨ 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. To do this, call the *Init* block FB 871 - VMC_InitSigma5_EC with *Enable* = TRUE.

   ⇨ The output *Valid* returns TRUE. In the event of a fault, you can determine the error by evaluating the *ErrorID*.

   You have to call the *Init* block again if you load a new axis DB or you have changed parameters on the *Init* block.

   Do not continue until the Init block does not report any errors!

3. Ensure that the *Kernel* block FB 870 - VMC_KernelSigma5_EC is cyclically called. In this way, control signals are transmitted to the drive and status messages are reported.
4. Program your application with the FB 860 - VMC_AxisControl or with the PLCopen blocks.

Controlling the drive via HMI

There is the possibility to control your drive via HMI. For this, a predefined symbol library is available for Movicon to access the VMC_AxisControl function block. *Chap. 10 'Controlling the drive via HMI' page 432*

3.1.4.4 Copy project

Proceeding

In the example, the station *'Source'* is copied and saved as *'Target '*.

1. Open the hardware configuration of the *'Source'* CPU and start the *SPEED7 EtherCAT Manager*.
2. In the *SPEED7 EtherCAT Manager*, via *'File → Save as'* save the configuration in your working directory.

3. Close the *SPEED7 EtherCAT Manager* and the hardware configurator.
4. Copy the station *'Source'* with Ctrl + C and paste it as *'Target'* into your project with Ctrl + V.
5. Select the *'Blocks'* directory of the *'Target'* CPU and delete the *'System data'*.
6. Open the hardware configuration of the *'Target'* CPU. Adapt the IP address data or re-network the CPU or the CP again.

> *Before calling the SPEED7 EtherCAT Manager you have always to save your project with 'Station → Save and compile'.*

7. Safe your project with *'Station → Safe and compile'*.
8. Open the *SPEED7 EtherCAT Manager*.
9. Use *'File → Open'* to load the configuration from your working directory.
10. Close the *SPEED7 EtherCAT Manager*.
11. Save and compile your configuration.

3.1.5 Drive specific blocks

The PLCopen blocks for axis control can be found here: *Chap. 9 'Blocks for axis control' page 361*

3.1.5.1 UDT 870 - VMC_ConfigSigma5EC_REF - *Sigma-5* EtherCAT Data structure axis configuration

This is a user-defined data structure that contains information about the configuration data. The UDT is specially adapted to the use of a *Sigma-5* drive, which is connected via EtherCAT.

3.1.5.2 FB 870 - VMC_KernelSigma5_EC - *Sigma-5* EtherCAT Kernel

Description

This block converts the drive commands for a *Sigma-5* axis via EtherCAT and communicates with the drive. For each *Sigma-5* axis, an instance of this FB is to be cyclically called.

Please note that this module calls the SFB 238 internally.

In the SPEED7 Studio, this module is automatically inserted into your project.

In Siemens SIMATIC Manager, you have to copy the SFB 238 from the Motion Control Library into your project.

Parameter	Declaration	Data type	Description
Init	INPUT	BOOL	The block is internally reset with an edge 0-1. Existing motion commands are aborted and the block is initialized.
Config	IN_OUT	UDT870	Data structure for transferring axis-dependent configuration data to the *AxisKernel*.
Axis	IN_OUT	MC_AXIS_REF	Data structure for transferring axis-dependent information to the *AxisKernel* and PLCopen blocks.

3.1.5.3 FB 871 - VMC_InitSigma5_EC - *Sigma-5* EtherCAT initialization

Description

This block is used to configure the axis. The module is specially adapted to the use of a *Sigma-5* drive, which is connected via EtherCAT.

Parameter	Declaration	Data type	Description
Enable	INPUT	BOOL	Release of initialization
Logical address	INPUT	INT	Start address of the PDO input data
InputsStartAddressPDO	INPUT	INT	Start address of the input PDOs
OutputsStartAddressPDO	INPUT	INT	Start address of the output PDOs

Parameter	Declaration	Data type	Description
EncoderType	INPUT	INT	Encoder type ■ 1: Absolute encoder ■ 2: Incremental encoder
EncoderResolutionBits	INPUT	INT	Number of bits corresponding to one encoder revolution. Default: 20
FactorPosition	INPUT	REAL	Factor for converting the position of user units [u] into drive units [increments] and back. It's valid: $p_{[increments]} = p_{[u]}$ x *FactorPosition* Please consider the factor which can be specified on the drive via the objects 0x2701: 1 and 0x2701: 2. This should be 1.
Velocity Factor	INPUT	REAL	Factor for converting the speed of user units [u/s] into drive units [increments/s] and back. It's valid: $v_{[increments/s]} = v_{[u/s]}$ x *FactorVelocity* Please also take into account the factor which you can specify on the drive via objects 0x2702: 1 and 0x2702: 2. This should be 1.
FactorAcceleration	INPUT	REAL	Factor to convert the acceleration of user units [u/s^2] in drive units [10^{-4} x increments/s^2] and back. It's valid: 10^{-4} x $a_{[increments/s^2]} = a_{[u/s^2]}$ x *FactorAcceleration* Please also take into account the factor which you can specify on the drive via objects 0x2703: 1 and 0x2703: 2. This should be 1.
OffsetPosition	INPUT	REAL	Offset for the zero position [u].
MaxVelocityApp	INPUT	REAL	Maximum application speed [u/s]. The command inputs are checked to the maximum value before execution.
MaxAccelerationApp	INPUT	REAL	Maximum acceleration of the application [u/s^2]. The command inputs are checked to the maximum value before execution.
MaxDecelerationApp	INPUT	REAL	Maximum application deceleration [u/s^2]. The command inputs are checked to the maximum value before execution.
MaxPosition	INPUT	REAL	Maximum position for monitoring the software limits [u].
MinPosition	INPUT	REAL	Minimum position for monitoring the software limits [u].
EnableMaxPosition	INPUT	BOOL	Monitoring maximum position ■ TRUE: Activates the monitoring of the maximum position.
EnableMinPosition	INPUT	BOOL	Monitoring minimum position ■ TRUE: Activation of the monitoring of the minimum position.
MinUserPosition	OUTPUT	REAL	Minimum user position based on the minimum encoder value of 0x80000000 and the *FactorPosition* [u].

Parameter	Declaration	Data type	Description
MaxUserPosition	OUTPUT	REAL	Maximum user position based on the maximum encoder value of 0x7FFFFFFF and the *FactorPosition* [u].
Valid	OUTPUT	BOOL	Initialization ■ TRUE: Initialization is valid.
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. 12 'ErrorID - Additional error information' page 457*
Config	IN_OUT	UDT870	Data structure for transferring axis-dependent configuration data to the *AxisKernel*.
Axis	IN_OUT	MC_AXIS_REF	Data structure for transferring axis-dependent information to the *AxisKernel* and PLCopen blocks.

3.2 Usage *Sigma-7S* EtherCAT

3.2.1 Overview

Usage of the double-axis drive *Chap. 3.3 'Usage Sigma-7W EtherCAT' page 85*

Precondition

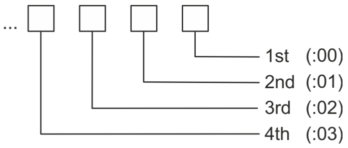
- SPEED7 Studio from V1.6.1
  or
- Siemens SIMATIC Manager from V 5.5, SP2 & *SPEED7 EtherCAT Manager* & *Simple Motion Control Library*
- CPU with EtherCAT master, e.g. CPU 015-CEFNR00
- *Sigma-7S* drive with EtherCAT option card

Steps of configuration

1. Set the parameters on the drive
   - The setting of the parameters happens by means of the software tool *Sigma Win+*.
2. Hardware configuration in VIPA *SPEED7 Studio* or Siemens SIMATIC Manager
   - Configuring a CPU with EtherCAT master functionality.
   - Configuration of a *Sigma-7S* EtherCAT drive.
   - Configuring the EtherCAT connection via *SPEED7 EtherCAT Manager*.
3. Programming in VIPA *SPEED7 Studio* or Siemens SIMATIC Manager
   - Connecting the *Init* block to configure the axis.
   - Connecting the *Kernel* block to communicate with the axis.
   - Connecting the blocks for the motion sequences.

3.2.2 Set the parameters on the drive

Parameter digits



CAUTION!
Before the commissioning, you have to adapt your drive to your application with the *Sigma Win+* software tool! More may be found in the manual of your drive.

The following parameters must be set via *Sigma Win+* to match the *Simple Motion Control Library*:

Sigma-7S (24bit encoder)

Servopack Parameter	Address:digit	Name	Value
Pn205	(2205h)	Multiturn Limit Setting	65535
Pn20E	(220Eh)	ElectronicGear Ratio (Numerator)	16
Pn210	(2210h)	Electronic Gear Ratio (Denominator)	1
PnB02	(2701h:01)	Position User Unit (Numerator)	1
PnB04	(2701h:02)	Position User Unit (Denominator)	1
PnB06	(2702h:01)	Velocity User Unit (Numerator)	1
PnB08	(2702h:02)	Velocity User Unit (Denominator)	1
PnB0A	(2703h:01)	Acceleration User Unit (Numerator)	1
PnB0C	(2703h:02)	Acceleration User Unit (Denominator)	1

Please note that you have to enable the corresponding direction of your axis in accordance to your requirements. For this use the parameters Pn50A (P-OT) respectively Pn50B (N-OT) in Sigma Win+.

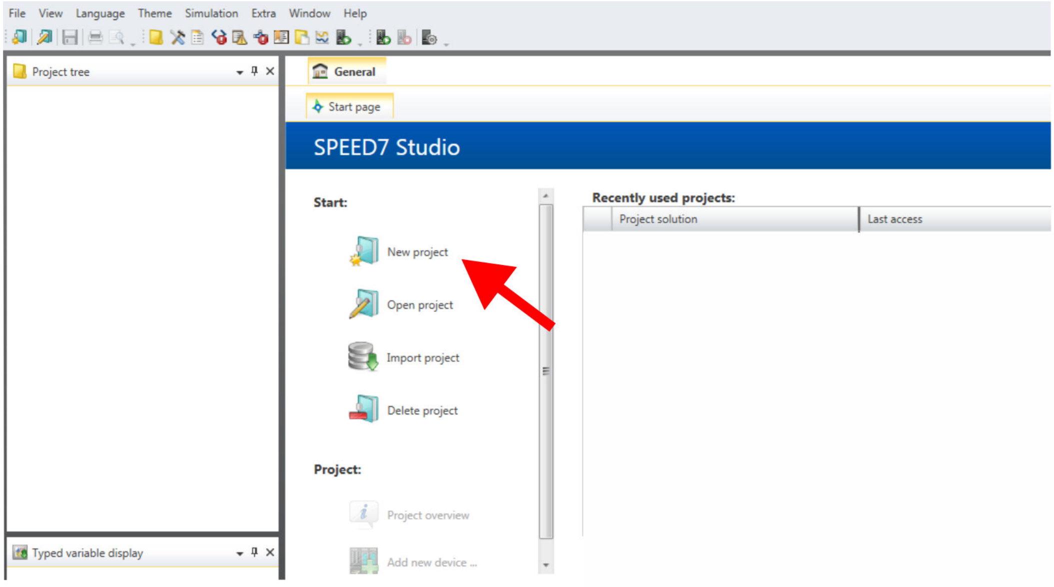
3.2.3 Usage in VIPA *SPEED7 Studio*

3.2.3.1 Hardware configuration

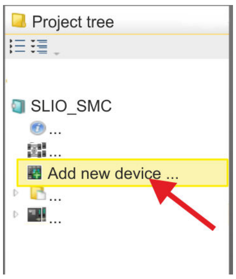
Add CPU in the project

Please use for configuration the *SPEED7 Studio* V1.6.1 and up.

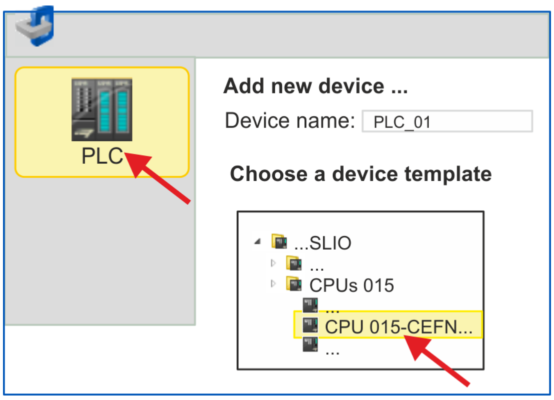
1. Start the *SPEED7 Studio*.



2. Create a new project at the start page with *'New project'*.
   ⇨ A new project is created and the view *'Devices and networking'* is shown.



3. Click in the *Project tree* at *'Add new device ...'*.

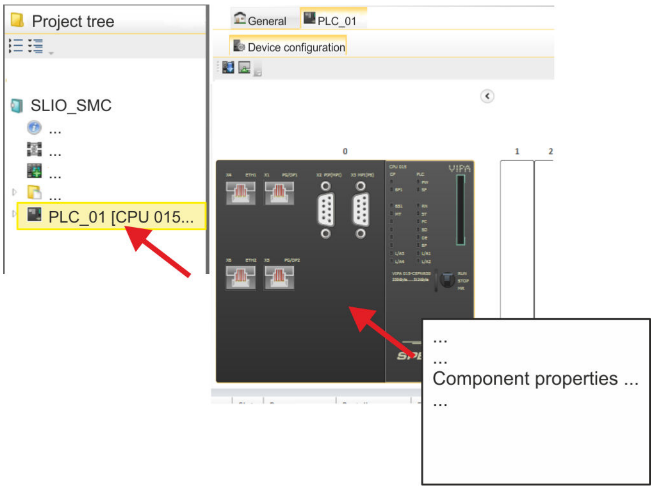


   ⇨ A dialog for device selection opens.

4. Select from the *'Device templates'* a CPU with EtherCAT master functions such as CPU 015-CEFNR00 and click at [OK].
   ⇨ The CPU is inserted in *'Devices and networking'* and the *'Device configuration'* is opened.

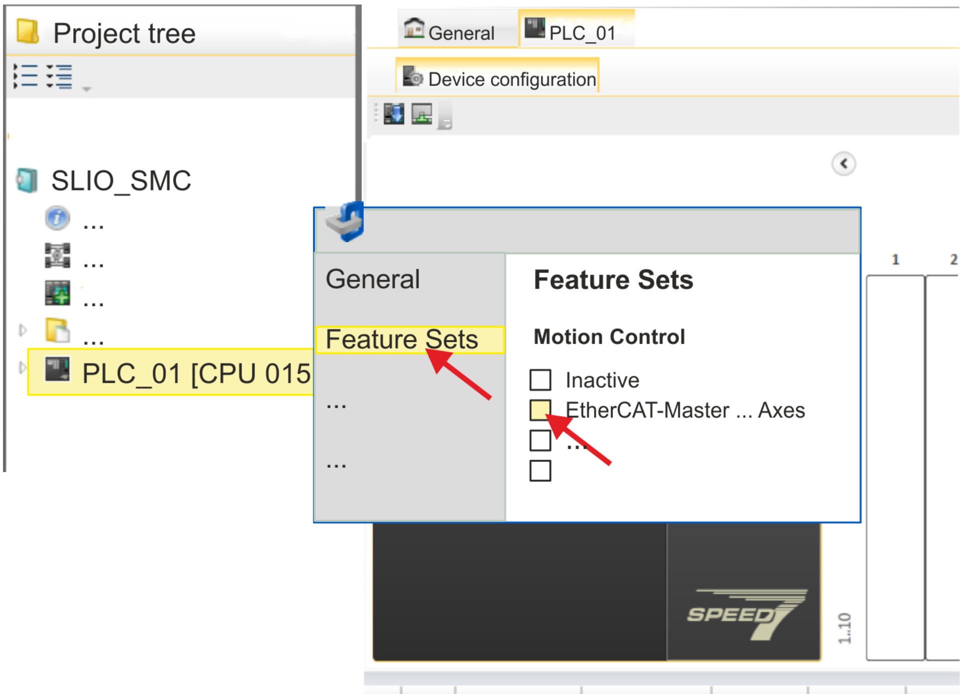
Activate motion control functions

If the EtherCAT master functionality is not yet activated on your CPU, the activation takes place as follows:



1. Click at the CPU in the *'Device configuration'* and select *'Context menu ➔ Components properties'*.

   ⇨ The properties dialog of the CPU is opened.



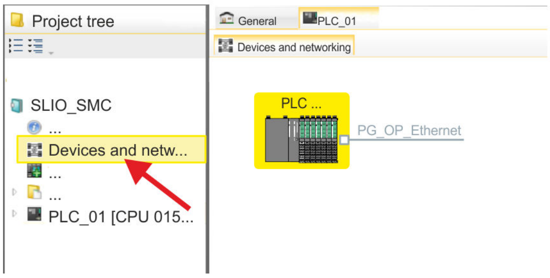
2. Click at *'Feature Sets'* and activate at *'Motion Control'* the parameter *'EtherCAT-Master... Axes'*. The number of axes is not relevant in this example.
3. Confirm your input with [OK].

   ⇨ The motion control functions are now available in your project.

> **CAUTION!**
> Please note due to the system, with every change to the feature set settings, the EtherCAT field bus system and its motion control configuration will be deleted from your project!

Configuration of Ethernet PG/OP channel

1. Click in the *Project tree* at *'Devices and networking'*.
   ⇨ You will get a graphical object view of your CPU.



2. Click at the network *'PG_OP_Ethernet'*.
3. Select *'Context menu ➔ Interface properties'*.
   ⇨ A dialog window opens. Here you can enter the IP address data for your Ethernet PG/OP channel. You get valid IP address parameters from your system administrator.
4. Confirm with [OK].
   ⇨ The IP address data are stored in your project listed in *'Devices and networking'* at *'Local components'*.

   After transferring your project your CPU can be accessed via Ethernet PG/OP channel with the set IP address data.

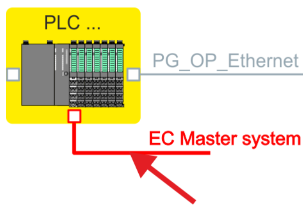
Installing the ESI file

For the *Sigma-7* EtherCAT drive can be configured in the *SPEED7 EtherCAT Manager*, the corresponding ESI file must be installed. Usually, the *SPEED7 Studio* is delivered with current ESI files and you can skip this part. If your ESI file is not up-to date, you will find the latest ESI file for the *Sigma-7* EtherCAT drive under *www.yaskawa.eu.com* at *'Service ➔ Drives & Motion Software'*.

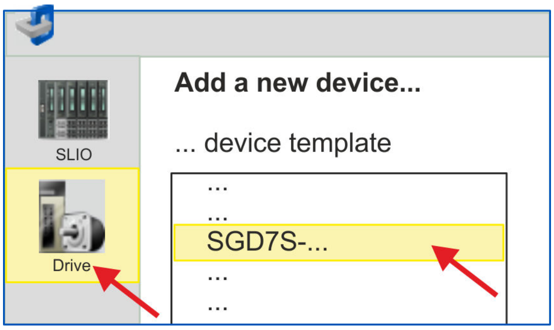
1. Download the according ESI file for your drive. Unzip this if necessary.
2. Navigate to your *SPEED7 Studio*.
3. Open the corresponding dialog window by clicking on *'Extra ➔ Install device description (EtherCAT - ESI)'*.
4. Under *'Source path'*, specify the ESI file and install it with [Install].
   ⇨ The devices of the ESI file are now available.

Add a *Sigma-7S* single axis drive

1. Click in the Project tree at *'Devices and networking'*.
2. Click here at *'EC-Mastersystem'* and select *'Context menu ➔ Add new device'*.



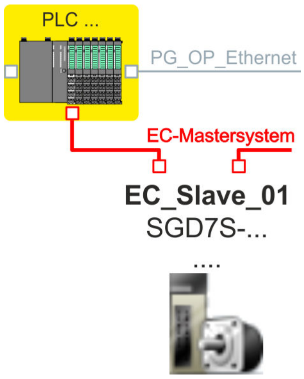
   ⇨ The device template for selecting an EtherCAT device opens.



3. Select your *Sigma-7* drive:

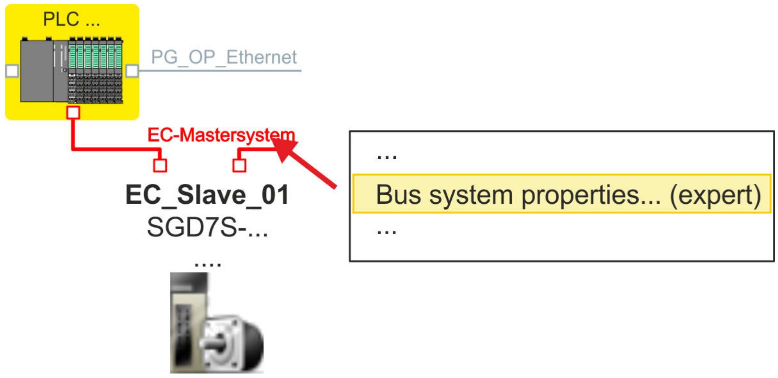
- SGD7S-xxxAA0...
- SGD7S-xxxDA0...
- SGD7S-xxxxA0...

Confirm with [OK]. If your drive does not exist, you must install the corresponding ESI file as described above.



⇨ The *Sigma-7* drive is connected to your EC-Mastersystem.

Configure *Sigma-7S* single axis drive

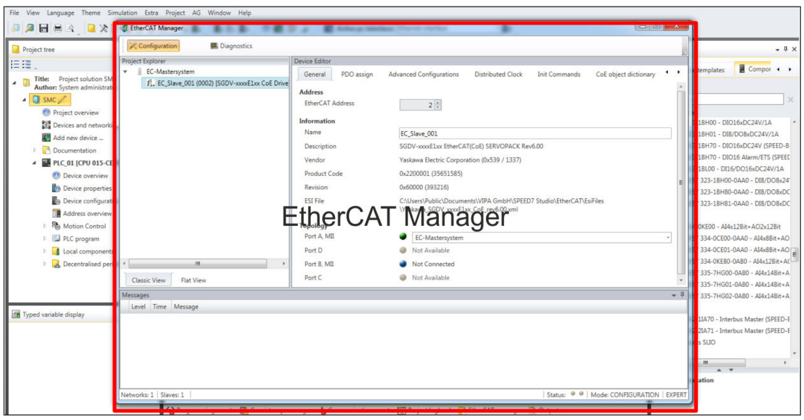


1. Click here at *'EC-Mastersystem'* and select *'Context menu ➔ Bus system properties (expert)'*.

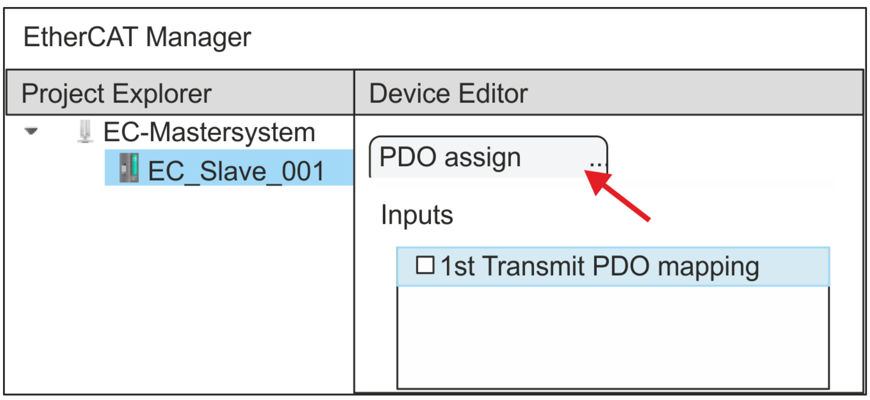
> *You can only edit PDOs in 'Expert mode'! Otherwise, the buttons are hidden.*

⇨ The *SPEED7 EtherCAT Manager* opens. Here you can configure the EtherCAT communication to your *Sigma-7* drive.

More information about the usage of the *SPEED7 EtherCAT Manager* may be found in the online help of the *SPEED7 Studio*.



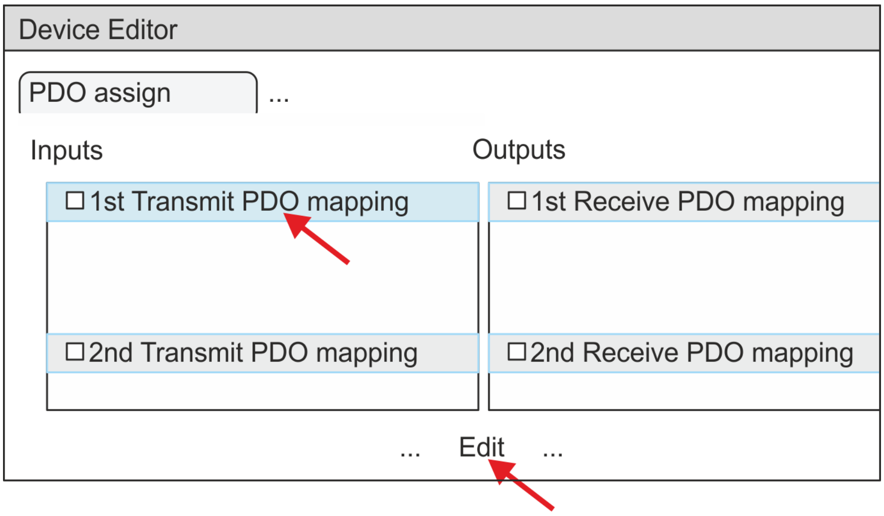
2. Click on the slave in the *SPEED7 EtherCAT Manager* and select the *'PDO assign'* tab in the *'Device editor'*.



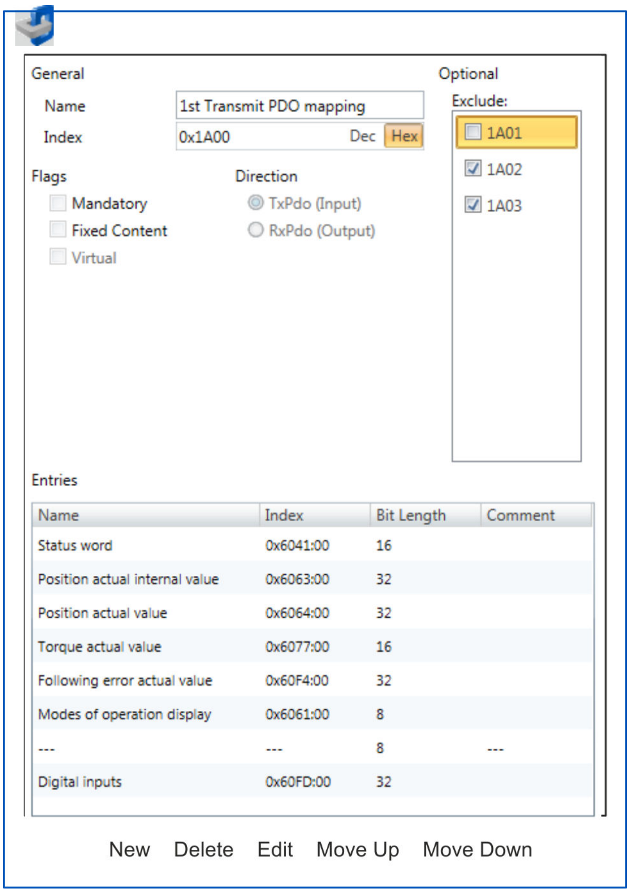
⇨ This dialog shows a list of the PDOs.

3. By selecting the appropriate mapping, you can edit the PDOs with [Edit]. Select the mapping *'1st Transmit PDO mapping'* and click at [Edit].

> *Please note that some PDOs can not be edited because of the default settings. By de-activating already activated PDOs, you can release the processing of locked PDOs.*



⇨ The dialog *'Edit PDO'* is opened. Please check the PDO settings listed here and adjust them if necessary. Please also take into account the order of the *'Entries'* and add them accordingly.



The following functions are available for editing the *'Entries'*:

- New
  – Here you can create a new entry in a dialog by selecting the corresponding entry from the *'CoE object dictionary'* and making your settings. The entry is accepted with [OK] and is listed in the list of entries.
- Delete
  – This allows you to delete a selected entry.
- Edit
  – This allows you to edit the general data of an entry.
- Move Up/Down
  – This allows you to move the selected entry up or down in the list.

4. Perform the following settings:

Inputs: 1st Transmit PDO 0x1A00

- General
  – Name: 1st Transmit PDO mapping
  – Index: 0x1A00
- Flags
  – Everything de-activated
- Direction
  – TxPdo (Input): activated
- Exclude
  Please note these settings, otherwise the PDO mappings can not be activated at the same time!
  – 1A01: de-activated
- Entries

Name	Index	Bit length
Status word	0x6041:00	16bit
Position actual internal value	0x6063:00	32bit
Position actual value	0x6064:00	32bit
Torque actual value	0x6077:00	16bit
Following error actual value	0x60F4:00	32bit
Modes of operation display	0x6061:00	8bit
---	---	8bit
Digital inputs	0x60FD:00	32bit

Close the dialog *'Edit PDO'* with [OK].

5. Select the mapping *'2nd Transmit PDO mapping'* and click at [Edit]. Perform the following settings:

Inputs: 2nd Transmit PDO 0x1A01

- General
  - Name: 2nd Transmit PDO mapping
  - Index: 0x1A01
- Flags
  - Everything de-activated
- Direction
  - TxPdo (Input): activated
- Exclude

  Please note these settings, otherwise the PDO mappings can not be activated at the same time!
  - 1A00: de-activated
  - 1A02: de-activated
  - 1A03: de-activated
- Entries

Name	Index	Bit length
Touch probe status	0x60B9:00	16bit
Touch probe 1 position value	0x60BA:00	32bit
Touch probe 2 position value	0x60BC:00	32bit
Velocity actual value	0x606C:00	32bit

Close the dialog *'Edit PDO'* with [OK].

6. Select the mapping *'1st Receive PDO mapping'* and click at [Edit]. Perform the following settings:

Outputs: 1st Receive PDO 0x1600

- General
  - Name: 1st Receive PDO mapping
  - Index: 0x1600
- Flags
  - Everything de-activated
- Direction
  - RxPdo (Output): activated
- Exclude

  Please note these settings, otherwise the PDO mappings can not be activated at the same time!
  - 1601: de-activated
  - 1602: de-activated
  - 1603: de-activated
- Entries

Name	Index	Bit length
Control word	0x6040:00	16bit
Target position	0x607A:00	32bit
Target velocity	0x60FF:00	32bit
Modes of operation	0x6060:00	8bit
---	---	8bit
Touch probe function	0x60B8:00	16bit

Close the dialog *'Edit PDO'* with [OK].

7. Select the mapping *'2nd Receive PDO mapping'* and click at [Edit]. Perform the following settings:

Outputs: 2nd Receive PDO 0x1601

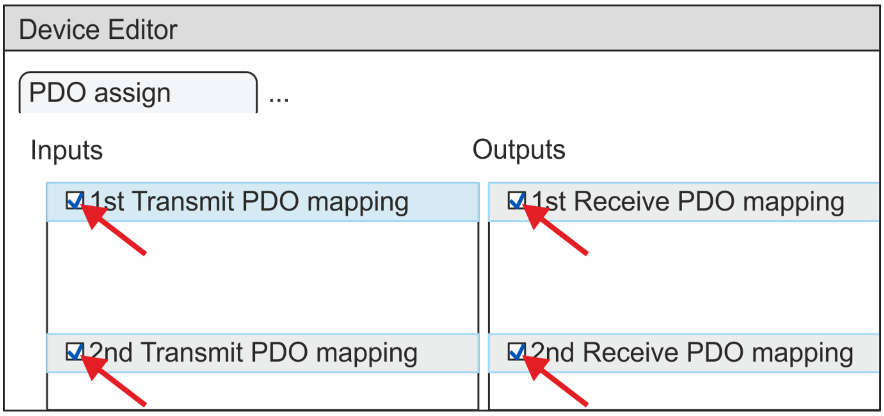
- General
  - Name: 2nd Receive PDO mapping
  - Index: 0x1601
- Flags
  - Everything de-activated
- Direction
  - RxPdo (Output): activated
- Exclude

  Please note these settings, otherwise the PDO mappings can not be activated at the same time!
  - 1600: de-activated
  - 1602: activated
  - 1603: activated
- Entries

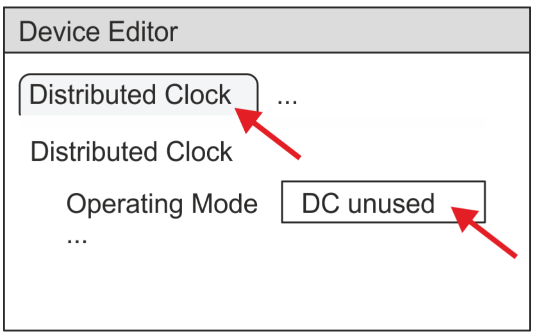
Name	Index	Bit length
Profile velocity	0x6081:00	32Bit
Profile acceleration	0x6083:00	32Bit
Profile deceleration	0x6084:00	32Bit

Close the dialog *'Edit PDO'* with [OK].

8. In PDO assignment, activate the PDOs 1 and 2 for the inputs and outputs. All subsequent PDOs must remain de-activated. If this is not possible, please check the respective PDO parameter *'Exclude'*.

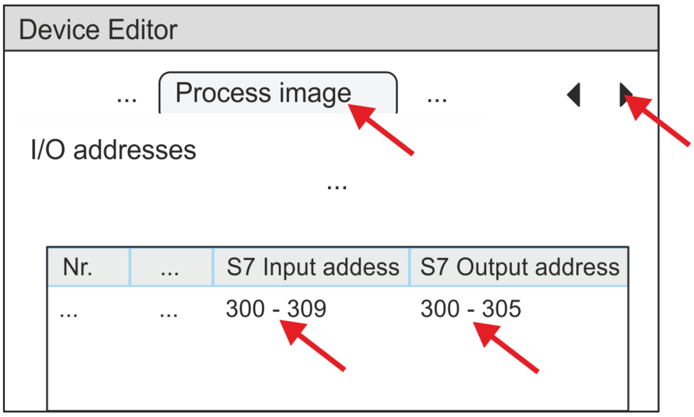


9. In the *'Device Editor'* of the *SPEED7 EtherCAT Manager*, select the *'Distributed clocks'* tab and set *'DC unused'* as *'Operating mode'*.

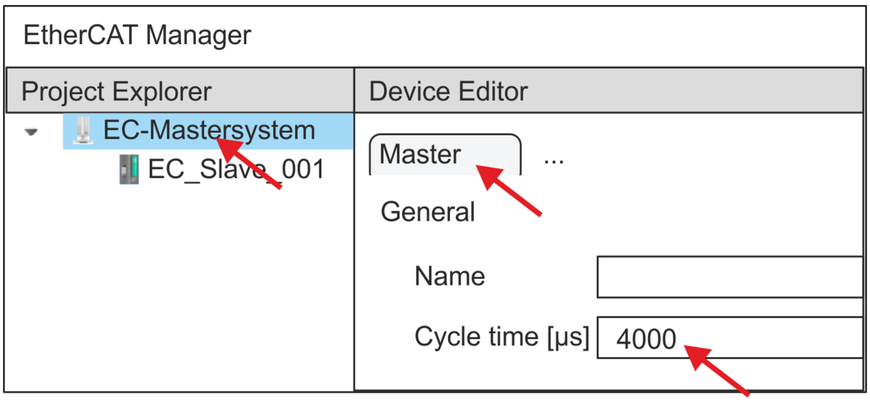


10. Select the *'Process image'* tab via the arrow key in the *'Device editor'* and note for the parameter of the block FB 873 - VMC_InitSigma7S_EC the following PDO.

- *'S7 Input address'* → *'InputsStartAddressPDO'*
- *'S7 Output address'* → *'OutputsStartAddressPDO'*



11. Click on *'EC-Mastersystem'* in the *SPEED7 EtherCAT Manager* and select the *'Master'* tab in the *'Device editor'*.

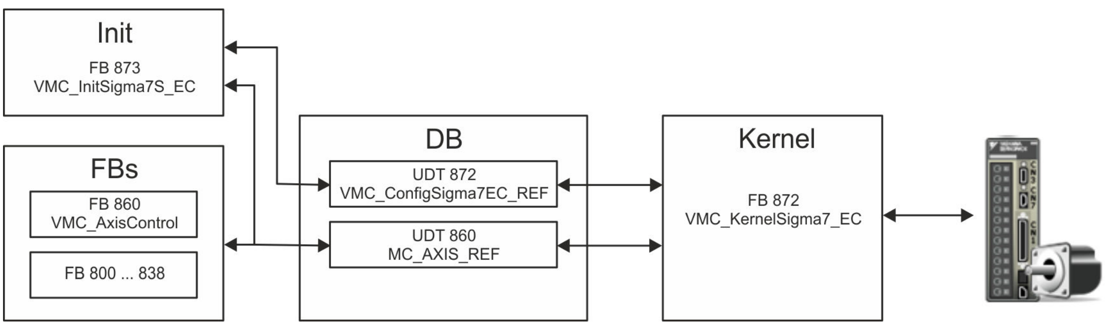


⇨ Set a cycle time of at least 4ms for Sigma-7S (400V) drives (SGD7S-xxxDA0 ... and SGD7S-xxxxA0 ...). Otherwise, leave the value at 1ms.

12. By closing the dialog of the *SPEED7 EtherCAT Manager* with [X] the configuration is taken to the *SPEED7 Studio*.

3.2.3.2 User program

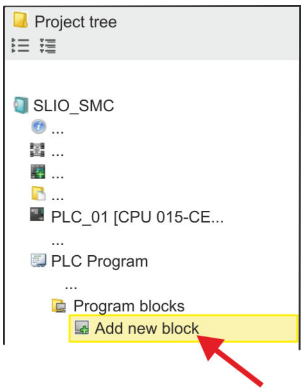
3.2.3.2.1 Program structure



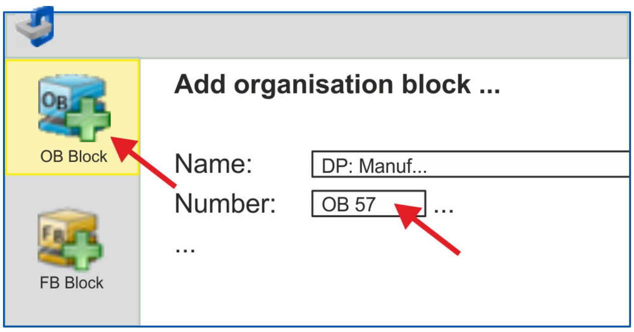
- DB
  A data block (axis DB) for configuration and status data must be created for each axis of a drive. The data block consists of the following data structures:
  – UDT 872 - *VMC_Config*Sigma7EC_REF
    The data structure describes the structure of the configuration of the drive.
    Specific data structure for *Sigma-7* EtherCAT.
  – UDT 860 - *MC_AXIS_REF*
    The data structure describes the structure of the parameters and status information of drives.
    General data structure for all drives and bus systems.
- FB 873 - *VMC_Init*Sigma7S_EC
  – The *Init*t block is used to configure an axis.
  – Specific block for *Sigma-7S* EtherCAT.
  – The configuration data for the initialization must be stored in the *axis DB*.
- FB 872 - *VMC_Kernel*Sigma7_EC
  – The *Kernel* block communicates with the drive via the appropriate bus system, processes the user requests and returns status messages.
  – Specific block for *Sigma-7* EtherCAT.
  – The exchange of the data takes place by means of the *axis DB*.
- FB 860 - *VMC_AxisControl*
  – General block for all drives and bus systems.
  – Supports simple motion commands and returns all relevant status messages.
  – The exchange of the data takes place by means of the *axis DB*.
  – For motion control and status query, via the instance data of the block you can link a visualization.
  – In addition to the FB 860 - *VMC_AxisControl*, *PLCopen* blocks can be used.
- FB 800 ... FB 838 - *PLCopen*
  – The PLCopen blocks are used to program motion sequences and status queries.
  – General blocks for all drives and bus systems.

3.2.3.2.2 Programming

Copy blocks into project

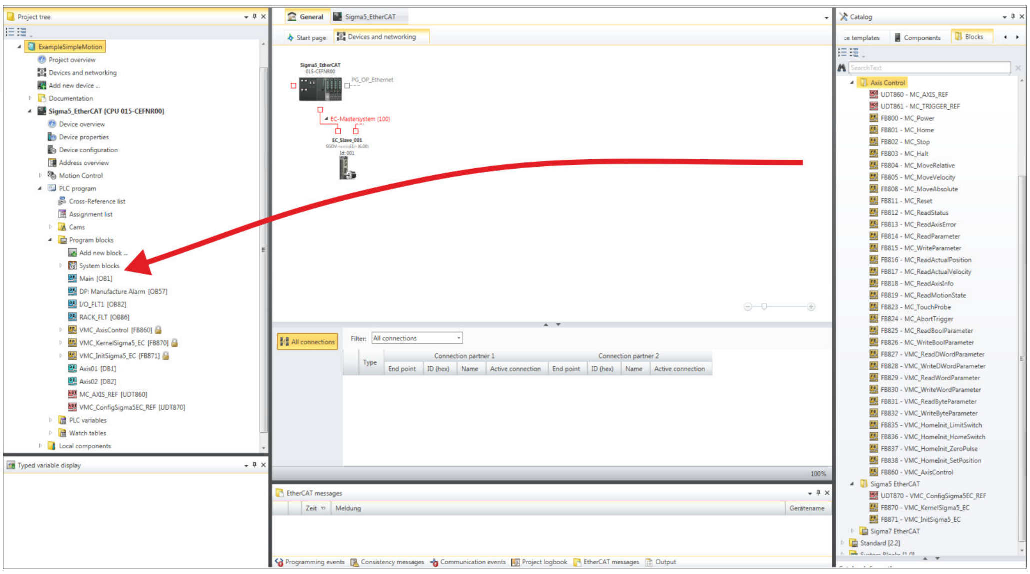


1. Click in the *Project tree* within the CPU at *'PLC program'*, *'Program blocks'* at *'Add New block'*.



⇨ The dialog *'Add block'* is opened.

2. Select the block type *'OB block'* and add one after the other OB 57, OB 82 and OB 86 to your project.



3. In the *'Catalog'*, open the *'Simple Motion Control'* library at *'Blocks'* and drag and drop the following blocks into *'Program blocks'* of the *Project tree*:
   - *Sigma-7* EtherCAT:
     – UDT 872 - VMC_ConfigSigma7EC_REF
     – FB 872 - VMC_KernelSigma7_EC
     – FB 873 - VMC_InitSigma7S_EC
   - Axis Control
     – UDT 860 - MC_AXIS_REF
     – Blocks for your movement sequences

Create axis DB

1. Add a new DB as your *axis DB* to your project. Click in the *Project tree* within the CPU at *'PLC program'*, *'Program blocks'* at *'Add New block'*, select the block type *'DB block'* and assign the name "Axis01" to it. The DB number can freely be selected such as DB10.

   ⇨ The block is created and opened.

2. - In "Axis01", create the variable "Config" of type UDT 872. These are specific axis configuration data.
   - In "Axis01", create the variable "Axis" of type UDT 860. During operation, all operating data of the axis are stored here.

Axis01 [DB10]

Data block structure

	Adr...	Name	Data type	...
	...	Config	UDT	[872]
	...	Axis	UDT	[860]

OB 1

Configuration of the axis

Open OB 1 and program the following FB calls with associated DBs:

FB 873 - VMC_InitSigma7S_EC, DB 873 *Chap. 3.2.5.3 'FB 873 - VMC_InitSigma7S_EC - Sigma-7S EtherCAT Initialization' page 83*

At *InputsStartAddressPDO* respectively *OutputsStartAddressPDO*, enter the address from the *SPEED7 EtherCAT Manager*. 59

```
CALL  "VMC_InitSigma7S_EC" , "DI_InitSgm7SETC01"
Enable                :="InitS7SEC1_Enable"
LogicalAddress        :=300
InputsStartAddressPDO :=300 (EtherCAT-Man.: S7 Input
address)
OutputsStartAddressPDO:=300 (EtherCAT-Man.: S7 Output
address)
EncoderType           :=1
EncoderResolutionBits :=20
FactorPosition        :=1.048576e+006
FactorVelocity        :=1.048576e+006
FactorAcceleration    :=1.048576e+002
OffsetPosition        :=0.000000e+000
MaxVelocityApp        :=5.000000e+001
MaxAccelerationApp    :=1.000000e+002
MaxDecelerationApp    :=1.000000e+002
MaxVelocityDrive      :=6.000000e+001
MaxAccelerationDrive  :=1.500000e+002
MaxDecelerationDrive  :=1.500000e+002
MaxPosition           :=1.048500e+003
MinPosition           :=-1.048514e+003
EnableMaxPosition     :=TRUE
EnableMinPosition     :=TRUE
MinUserPosition       :="InitS7SEC1_MinUserPos"
MaxUserPosition       :="InitS7SEC1_MaxUserPos"
Valid                 :="InitS7SEC1_Valid"
Error                 :="InitS7SEC1_Error"
ErrorID               :="InitS7SEC1_ErrorID"
Config                :="Axis01".Config
Axis                  :="Axis01".Axis
```

Connecting the Kernel for the axis

The *Kernel* processes the user commands and passes them appropriately processed on to the drive via the respective bus system.

FB 872 - VMC_KernelSigma7_EC, DB 872 *Chap. 3.2.5.2 'FB 872 - VMC_KernelSigma7_EC - Sigma-7 EtherCAT Kernel' page 83*

```
      CALL  "VMC_KernelSigma7_EC" , "DI_KernelSgm5ETC01"
       Init  :="KernelS7SEC1_Init"
       Config:="Axis01".Config
       Axis  :="Axis01".Axis
```

Connecting the block for motion sequences

For simplicity, the connection of the FB 860 - VMC_AxisControl is to be shown here. This universal block supports simple motion commands and returns status messages. The inputs and outputs can be individually connected. Please specify the reference to the corresponding axis data at *'Axis'* in the *axis DB*.

FB 860 - VMC_AxisControl, DB 860 *Chap. 9.2.2 'FB 860 - VMC_AxisControl - Control block axis control' page 363*

⇨

```
CALL  "VMC_AxisControl" , "DI_AxisControl01"
 AxisEnable        :="AxCtrl1_AxisEnable"
 AxisReset         :="AxCtrl1_AxisReset"
 HomeExecute       :="AxCtrl1_HomeExecute"
 HomePosition      :="AxCtrl1_HomePosition"
 StopExecute       :="AxCtrl1_StopExecute"
 MvVelocityExecute:="AxCtrl1_MvVelExecute"
 MvRelativeExecute:="AxCtrl1_MvRelExecute"
 MvAbsoluteExecute:="AxCtrl1_MvAbsExecute"
 PositionDistance :="AxCtrl1_PositionDistance"
 Velocity          :="AxCtrl1_Velocity"
 Acceleration      :="AxCtrl1_Acceleration"
 Deceleration      :="AxCtrl1_Deceleration"
 JogPositive       :="AxCtrl1_JogPositive"
 JogNegative       :="AxCtrl1_JogNegative"
 JogVelocity       :="AxCtrl1_JogVelocity"
 JogAcceleration   :="AxCtrl1_JogAcceleration"
 JogDeceleration   :="AxCtrl1_JogDeceleration"
 AxisReady         :="AxCtrl1_AxisReady"
 AxisEnabled       :="AxCtrl1_AxisEnabled"
 AxisError         :="AxCtrl1_AxisError"
 AxisErrorID       :="AxCtrl1_AxisErrorID"
 DriveWarning      :="AxCtrl1_DriveWarning"
 DriveError        :="AxCtrl1_DriveError"
 DriveErrorID      :="AxCtrl1_DriveErrorID"
 IsHomed           :="AxCtrl1_IsHomed"
 ModeOfOperation   :="AxCtrl1_ModeOfOperation"
 PLCopenState      :="AxCtrl1_PLCopenState"
 ActualPosition    :="AxCtrl1_ActualPosition"
 ActualVelocity    :="AxCtrl1_ActualVelocity"
 CmdDone           :="AxCtrl1_CmdDone"
 CmdBusy           :="AxCtrl1_CmdBusy"
 CmdAborted        :="AxCtrl1_CmdAborted"
 CmdError          :="AxCtrl1_CmdError"
 CmdErrorID        :="AxCtrl1_CmdErrorID"
 DirectionPositive:="AxCtrl1_DirectionPos"
 DirectionNegative:="AxCtrl1_DirectionNeg"
 SWLimitMinActive :="AxCtrl1_SWLimitMinActive"
 SWLimitMaxActive :="AxCtrl1_SWLimitMaxActive"
 HWLimitMinActive :="AxCtrl1_HWLimitMinActive"
 HWLimitMaxActive :="AxCtrl1_HWLimitMaxActive"
 Axis              :="Axis01".Axis
```

For complex motion tasks, you can use the PLCopen blocks. Please specify the reference to the corresponding axis data at Axis in the axis DB.

Your project now includes the following blocks:

- OB 1 - Main
- OB 57 - DP Manufacturer Alarm
- OB 82 - I/O_FLT1
- OB 86 - Rack_FLT
- FB 860 - VMC_AxisControl with instance DB

- FB 872 - VMC_KernelSigma7_EC with instance DB
- FB 873 - VMC_InitSigma7S_EC with instance DB
- UDT 860 - MC_Axis_REF
- UDT 872 - VMC_ConfigSigma7EC_REF

Sequence of operations

1. Select *'Project* ➔ *Compile all'* and transfer the project into your CPU.

   You can find more information on the transfer of your project in the online help of the *SPEED7 Studio*.

   ⇨ 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. To do this, call the *Init* block FB 873 - VMC_InitSigma7S_EC with *Enable* = TRUE.

   ⇨ The output *Valid* returns TRUE. In the event of a fault, you can determine the error by evaluating the *ErrorID*.

   You have to call the *Init* block again if you load a new axis DB or you have changed parameters on the *Init* block.

   > *Do not continue until the Init block does not report any errors!*

3. Ensure that the *Kernel* block FB 872 - VMC_KernelSigma7_EC is called cyclically. In this way, control signals are transmitted to the drive and status messages are reported.
4. Program your application with the FB 860 - VMC_AxisControl or with the PLCopen blocks.

Controlling the drive via HMI

There is the possibility to control your drive via HMI. For this, a predefined symbol library is available for Movicon to access the VMC_AxisControl function block. *Chap. 10 'Controlling the drive via HMI' page 432*

3.2.4 Usage in Siemens SIMATIC Manager

3.2.4.1 Precondition

Overview

- Please use for configuration the Siemens SIMATIC Manager V 5.5 SP2 and up.
- 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 CPU'* is to be installed in the hardware catalog by means of the GSDML.
- The configuration of the EtherCAT masters happens in the Siemens SIMATIC Manager by means of a virtual PROFINET IO device *'EtherCAT network'*. The *'EtherCAT network'* is to be installed in the hardware catalog by means of the GSDML.
- The *'EtherCAT network'* can be configured with the VIPA Tool *SPEED7 EtherCAT Manager*.
- For the configuration of the drive in the *SPEED7 EtherCAT Manager* the installation of the according ESI file is necessary.

Installing the IO device *'VIPA SLIO System'*

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'*.

Installing the IO device EtherCAT network

The installation of the PROFINET IO devices *'EtherCAT Network'* happens in the hardware catalog with the following approach:

1. Go to the service area of www.vipa.com
2. Load from the download area at *'Config files ➔ EtherCAT'* the GSDML file for your EtherCAT master.
3. Extract the files 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 *'EtherCAT Network'* can be found at *'PROFINET IO ➔ Additional field devices ➔ I/O ➔ VIPA EtherCAT System'*.

Installing the *SPEED7 EtherCAT Manager*

The configuration of the PROFINET IO device *'EtherCAT Network'* happens by means of the VIPA *SPEED7 EtherCAT Manager*. This may be found in the service area of www.vipa.com at *'Service/Support ➔ Downloads ➔ Software'*.

The installation happens with the following proceeding:

1. Close the Siemens SIMATIC Manager.
2. Go to the service area of www.vipa.com
3. Load the *SPEED7 EtherCAT Manager* and unzip it on your PC.
4. For installation start the file EtherCATManager_v... .exe.
5. Select the language for the installation.
6. Accept the licensing agreement.
7. Select the installation directory and start the installation.
8. After installation you have to reboot your PC.

   ⇨ The *SPEED7 EtherCAT Manager* is installed and can now be called via the context menu of the Siemens SIMATIC Manager.

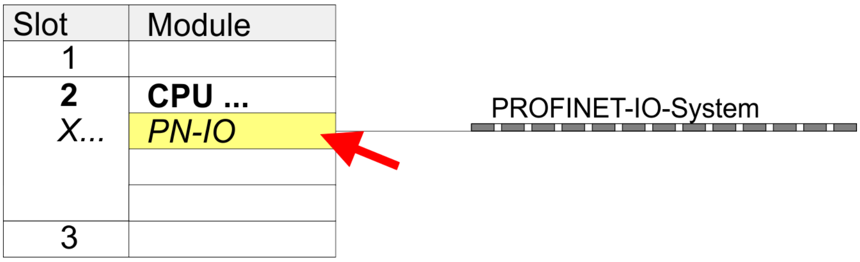
3.2.4.2 Hardware configuration

Configuring the CPU in the project

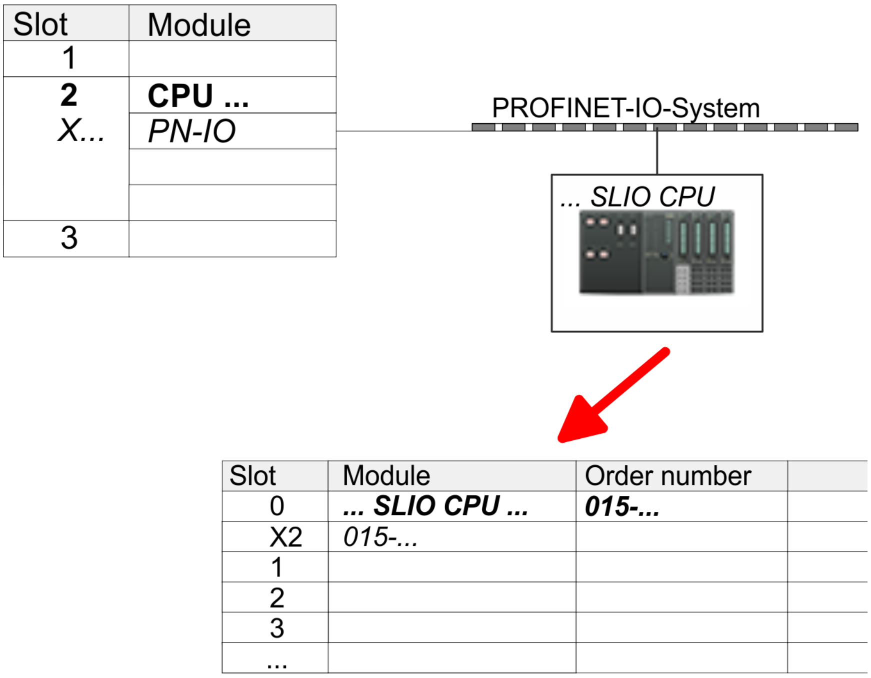
Slot	Module
1	
2	**CPU 315-2 PN/DP**
X1	*MPI/DP*
X2	*PN-IO*
X2...	*Port 1*
X2...	*Port 2*
3	

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 the CPU 315-2 PN/DP (315-2EH14 V3.2).
4. The integrated PROFIBUS DP master (jack X3) is to be configured and connected via the sub module *'X1 MPI/DP'*.
5. The integrated EtherCAT master is to be configured via the sub module *'X2 PN-IO'* as a virtual PROFINET network.
6. Click at the sub module *'PN-IO'* of the CPU.
7. Select *'Context menu ➔ Insert PROFINET IO System'*.



8. Create with [New] a new sub net and assign valid address data
9. Click at the sub module *'PN-IO'* of the CPU and open with *'Context menu ➔ Properties'* the properties dialog.
10. Enter at *'General'* a *'Device name'*. The device name must be unique at the Ethernet subnet.



11. Navigate in the hardware catalog to the directory *'PROFINET IO ➔ Additional field devices ➔ I/O ➔ VIPA SLIO System'* and connect the IO device *'015-CFFNR00 CPU'* to your PROFINET system.

⇨ In the Device 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

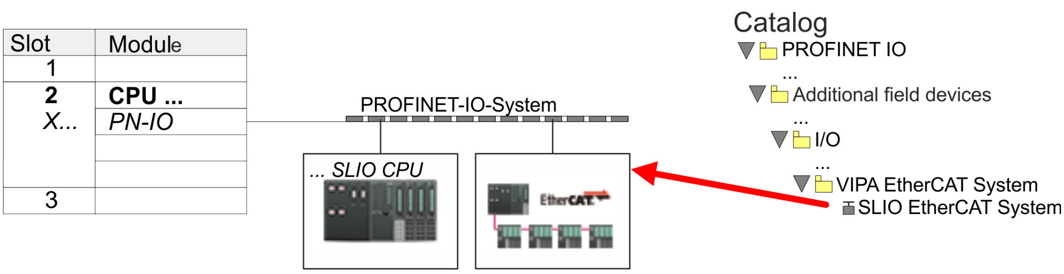
Slot	Module
1	
2	**CPU ...**
X...	*PN-IO*
3	
4	343-1EX30
5	
...	

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!

Insert *'EtherCAT network'*

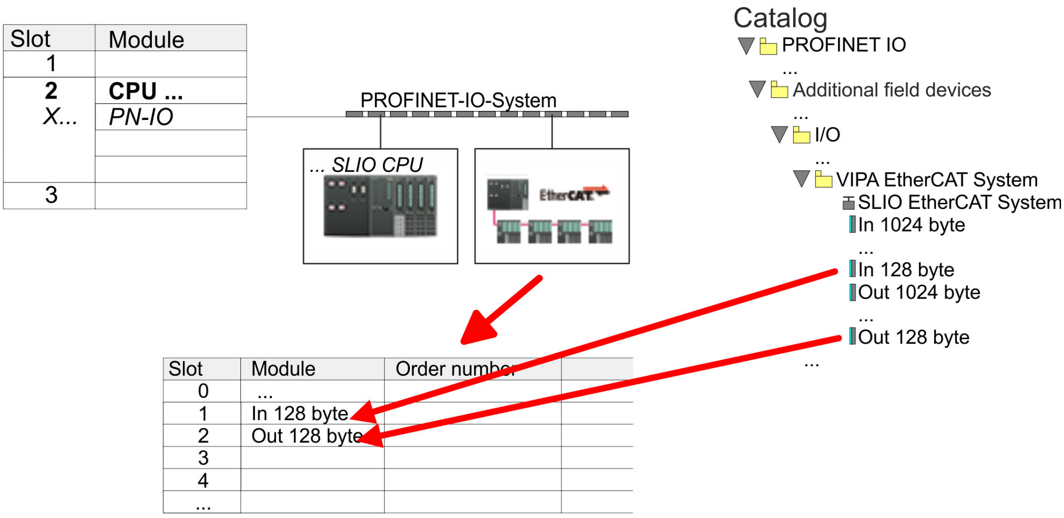


1. Navigate in the hardware catalog to the directory *'PROFINET IO ➔ Additional field devices ➔ I/O ➔ VIPA EtherCAT System'* and connect the IO device *'SLIO EtherCAT System'* to your PROFINET system.

2. Click at the inserted IO device *'EtherCAT Network'* and define the areas for in and output by drag and dropping the according *'Out'* or *'In'* area to a slot.

   Create the following areas:

   - In 128byte
   - Out 128byte

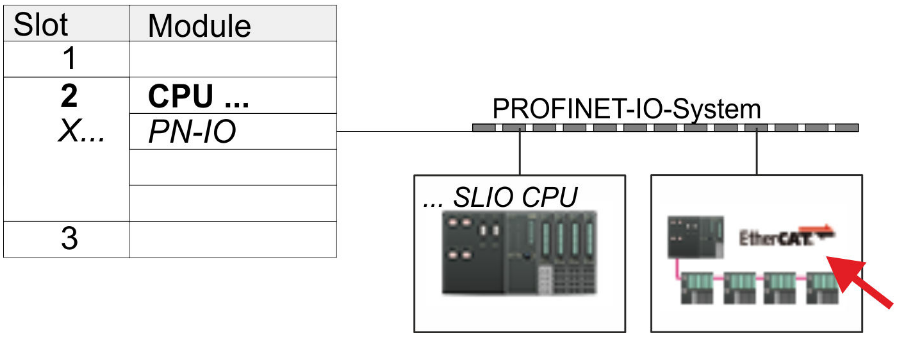


3. Select *'Station → Save and compile'*

Sigma-7S Configure EtherCAT drive

The drive is configured in the *SPEED7 EtherCAT Manager*.

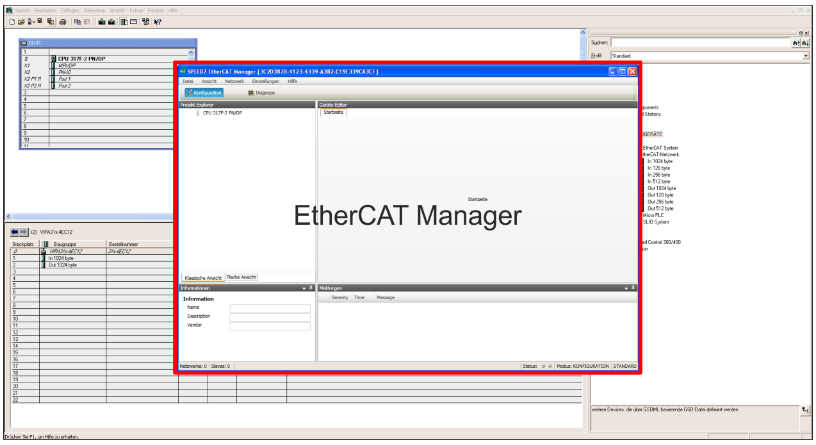
Before calling the SPEED7 EtherCAT Manager you have always to save your project with ‘Station ➔ Save and compile’.



1. Click at an inserted IO device *‘EtherCAT Network’* and select *‘Context menu ➔ Start Device-Tool ➔ SPEED7 EtherCAT Manager’*.

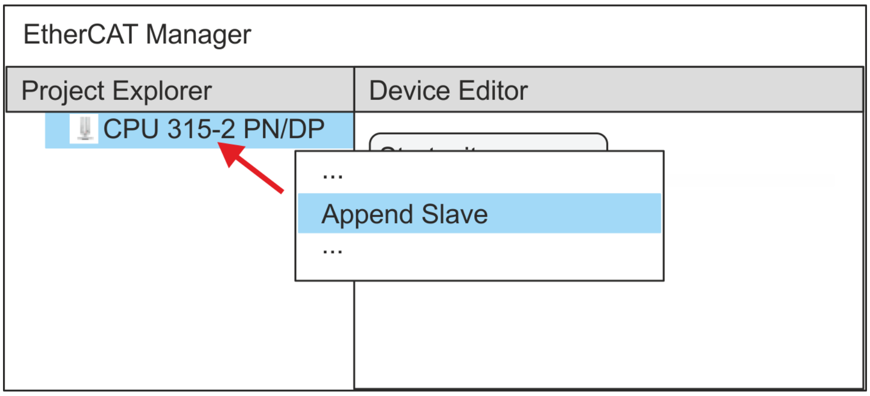
   ⇨ The *SPEED7 EtherCAT Manager* opens. Here you can configure the EtherCAT communication to your *Sigma-7S* drive.

   More information about the usage of the *SPEED7 EtherCAT Manager* may be found in the according manual or online help.



3. For the *Sigma-7S* EtherCAT drive to be configured in the *SPEED7 EtherCAT Manager*, the corresponding ESI file must be installed. The ESI file for the *Sigma-7S* EtherCAT drive can be found under www.yaskawa.eu.com at *‘Service ➔ Drives & Motion Software’*. Download the according ESI file for your drive. Unzip this if necessary.
4. Open in the *SPEED7 EtherCAT Manager* via *‘File ➔ ESI Manager’* the dialogue window *‘ESI Manager’*.
5. In the *‘ESI Manager’* click at [Add File] and select your ESI file. With [Open], the ESI file is installed in the *SPEED7 EtherCAT Manager*.
6. Close the *‘ESI Manager’*.

   ⇨ Your *Sigma-7S* EtherCAT drive is now available for configuration.



7. In the EtherCAT Manager, click on your CPU and open via *'Context menu → Append Slave'* the dialog box for adding an EtherCAT slave.

   ⇨ The dialog window for selecting an EtherCAT slave is opened.

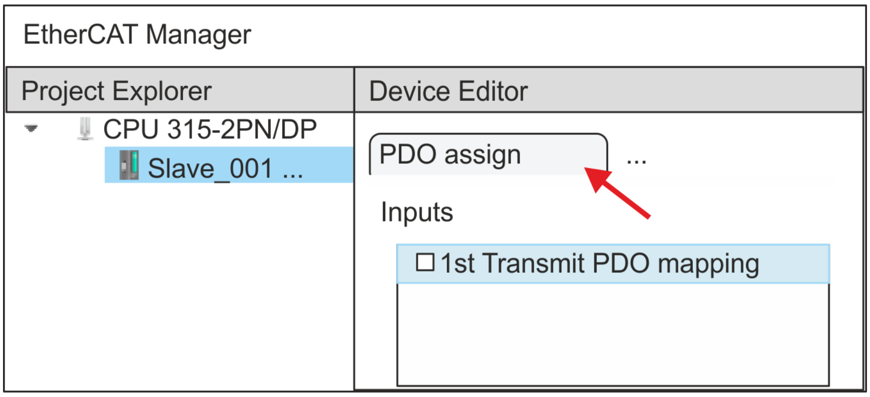
8. Select your *Sigma-7S* EtherCAT drive and confirm your selection with [OK].

   ⇨ The *Sigma-7S* EtherCAT drive is connected to the master and can now be configured.

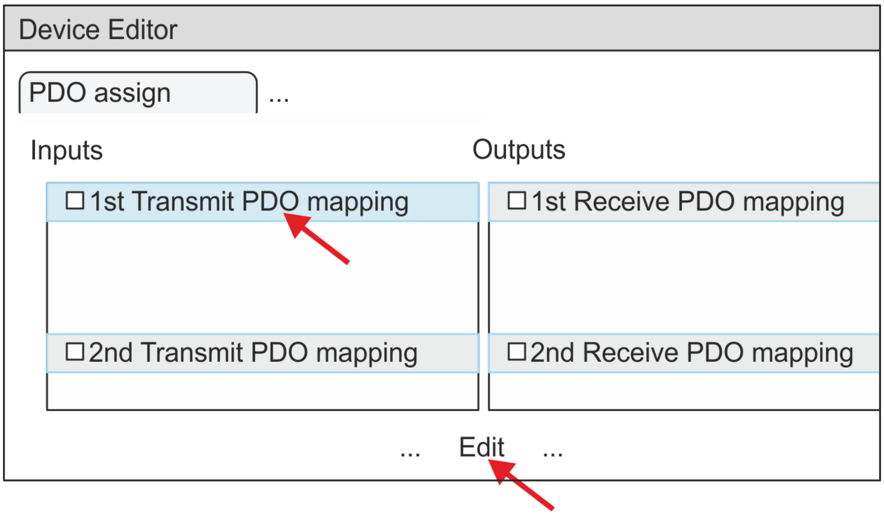
9. *You can only edit PDOs in 'Expert mode'! Otherwise, the buttons are hidden. By activating the 'Expert mode' you can switch to advanced setting.*

   By activating *'View → Expert'* you can switch to the *Expert mode*.

10. Click on the *Sigma-7S* EtherCAT Slave in the *SPEED7 EtherCAT Manager* and select the *'PDO assign'* tab in the *'Device editor'*.

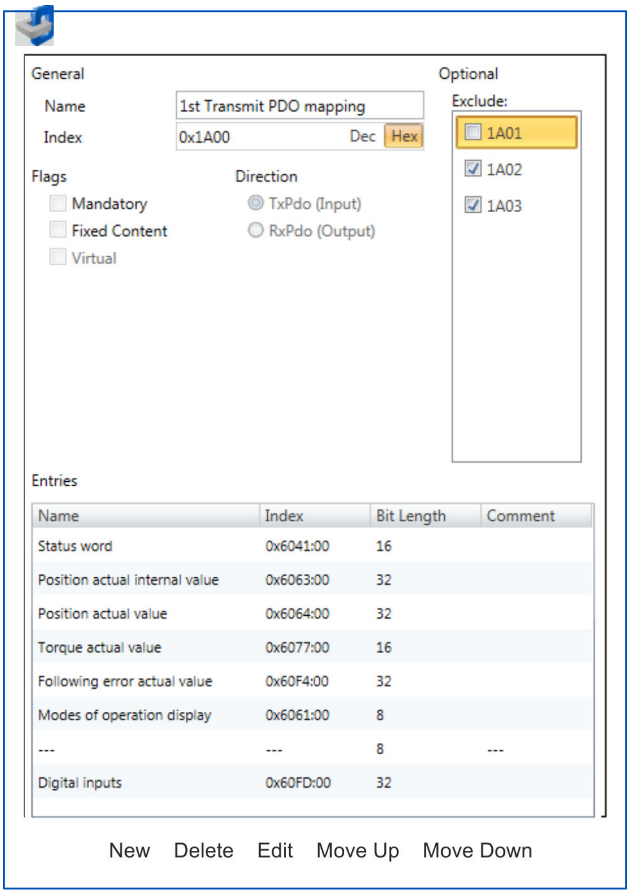


⇨ This dialog shows a list of the PDOs.



11. By selecting the appropriate PDO mapping, you can edit the PDOs with [Edit]. Select the mapping *'1st Transmit PDO mapping'* and click at [Edit].

> *Please note that some PDOs can not be edited because of the default settings. By de-activating already activated PDOs, you can release the processing of locked PDOs.*



⇨ The dialog *'Edit PDO'* is opened. Please check the PDO settings listed here and adjust them if necessary. Please also take into account the order of the *'Entries'* and add them accordingly.

The following functions are available for editing the *'Entries'*:

- New
  – Here you can create a new entry in a dialog by selecting the corresponding entry from the *'CoE object dictionary'* and making your settings. The entry is accepted with [OK] and is listed in the list of entries.
- Delete
  – This allows you to delete a selected entry.
- Edit
  – This allows you to edit the general data of an entry.
- Move Up/Down
  – This allows you to move the selected entry up or down in the list.

12. Perform the following settings:

Inputs: 1st Transmit PDO 0x1A00

- General
  – Name: 1st Transmit PDO mapping
  – Index: 0x1A00
- Flags
  – Everything de-activated
- Direction
  – TxPdo (Input): activated
- Exclude
  Please note these settings, otherwise the PDO mappings can not be activated at the same time!
  – 1A01: de-activated
- Entries

Name	Index	Bit length
Status word	0x6041:00	16bit
Position actual internal value	0x6063:00	32bit
Position actual value	0x6064:00	32bit
Torque actual value	0x6077:00	16bit
Following error actual value	0x60F4:00	32bit
Modes of operation display	0x6061:00	8bit
---	---	8bit
Digital inputs	0x60FD:00	32bit

Close the dialog *'Edit PDO'* with [OK].

13. Select the mapping *'2nd Transmit PDO mapping'* and click at [Edit]. Perform the following settings:

Inputs: 2nd Transmit PDO 0x1A01

- General
  - Name: 2nd Transmit PDO mapping
  - Index: 0x1A01
- Flags
  - Everything de-activated
- Direction
  - TxPdo (Input): activated
- Exclude
  Please note these settings, otherwise the PDO mappings can not be activated at the same time!
  - 1A00: de-activated
  - 1A02: de-activated
  - 1A03: de-activated
- Entries

Name	Index	Bit length
Touch probe status	0x60B9:00	16bit
Touch probe 1 position value	0x60BA:00	32bit
Touch probe 2 position value	0x60BC:00	32bit
Velocity actual value	0x606C:00	32bit

Close the dialog *'Edit PDO'* with [OK].

14. Select the mapping *'1st Receive PDO mapping'* and click at [Edit]. Perform the following settings:

Outputs: 1st Receive PDO 0x1600

- General
  – Name: 1st Receive PDO mapping
  – Index: 0x1600
- Flags
  – Everything de-activated
- Direction
  – RxPdo (Output): activated
- Exclude
  Please note these settings, otherwise the PDO mappings can not be activated at the same time!
  – 1601: de-activated
  – 1602: de-activated
  – 1603: de-activated
- Entries

Name	Index	Bit length
Control word	0x6040:00	16bit
Target position	0x607A:00	32bit
Target velocity	0x60FF:00	32bit
Modes of operation	0x6060:00	8bit
---	---	8bit
Touch probe function	0x60B8:00	16bit

Close the dialog *'Edit PDO'* with [OK].

15. Select the mapping *'2nd Receive PDO mapping'* and click at [Edit]. Perform the following settings:

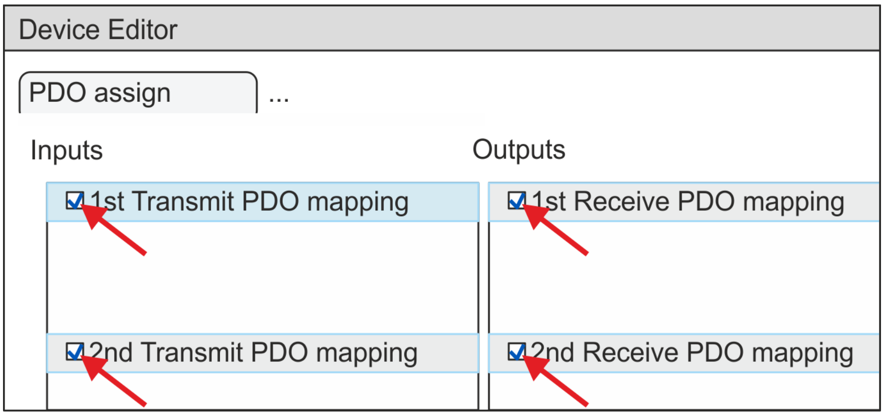
Outputs: 2nd Receive PDO 0x1601

- General
  - Name: 2nd Receive PDO mapping
  - Index: 0x1601
- Flags
  - Everything de-activated
- Direction
  - RxPdo (Output): activated
- Exclude
  Please note these settings, otherwise the PDO mappings can not be activated at the same time!
  - 1600: de-activated
  - 1602: activated
  - 1603: activated
- Entries

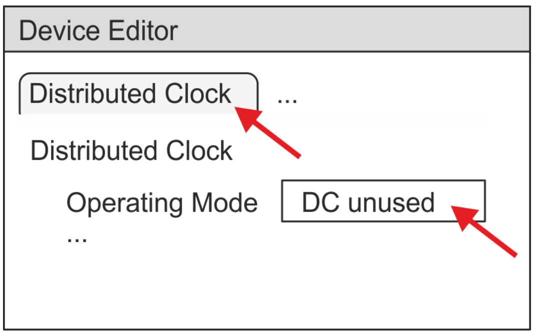
Name	Index	Bit length
Profile velocity	0x6081:00	32bit
Profile acceleration	0x6083:00	32bit
Profile deceleration	0x6084:00	32bit

Close the dialog *'Edit PDO'* with [OK].

16. In PDO assignment, activate the PDOs 1 and 2 for the inputs and outputs. All subsequent PDOs must remain de-activated. If this is not possible, please check the respective PDO parameter *'Exclude'*.

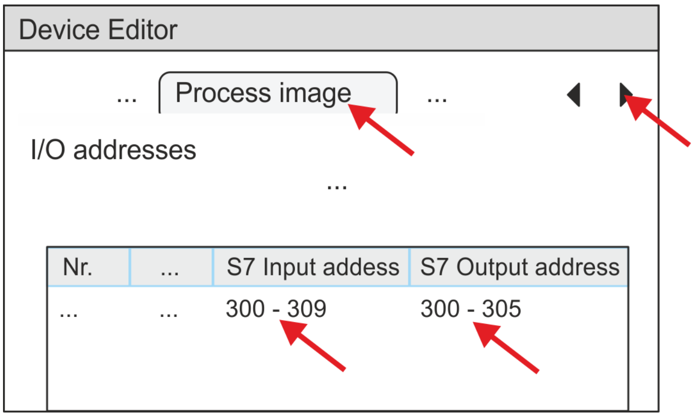


17. In the *'Device Editor'* of the *SPEED7 EtherCAT Manager*, select the *'Distributed clocks'* tab and set *'DC unused'* as *'Operating mode'*.

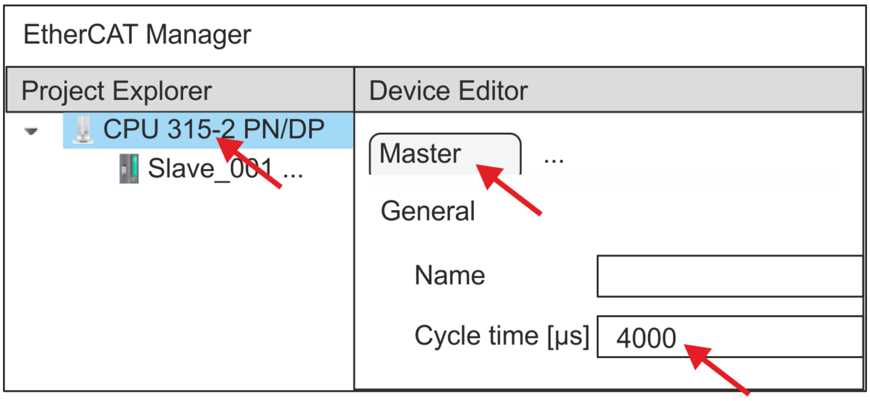


18. Select the *'Process image'* tab via the arrow key in the *'Device editor'* and note for the parameter of the block FB 873 - VMC_InitSigma7S_EC the following PDO.

- *'S7 Input address'* → *'InputsStartAddressPDO'*
- *'S7 Output address'* → *'OutputsStartAddressPDO'*



19. Click on your CPU in the *SPEED7 EtherCAT Manager* and select the *'Master'* tab in the *'Device editor'*.



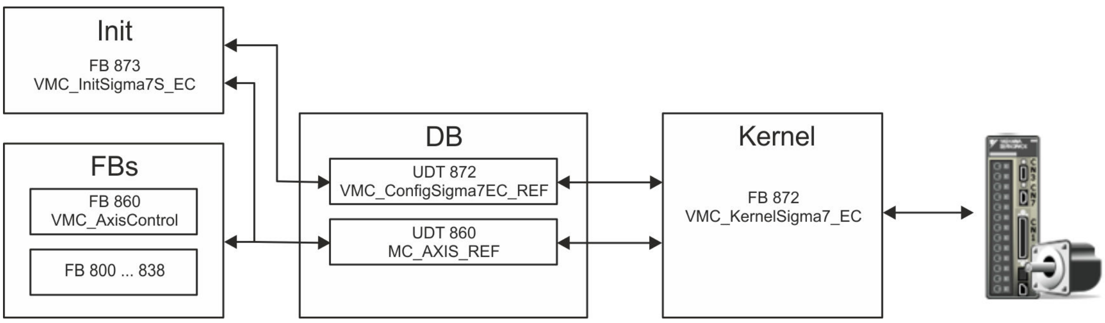
⇨ Set a cycle time of at least 4ms for Sigma-7S (400V) drives (SGD7S-xxxDA0 ... and SGD7S-xxxxA0 ...). Otherwise, leave the value at 1ms.

20. By closing the *SPEED7 EtherCAT Manager* with [X] the configuration is taken to the project. You can always edit your EtherCAT configuration in the *SPEED7 EtherCAT Manager*, since the configuration is stored in your project.

21. Save and compile your configuration.

3.2.4.3 User program

3.2.4.3.1 Program structure



- DB
  A data block (axis DB) for configuration and status data must be created for each axis of a drive. The data block consists of the following data structures:
  – UDT 872 - *VMC_Config*Sigma7EC_REF
    The data structure describes the structure of the configuration of the drive.
    Specific data structure for *Sigma-7* EtherCAT.
  – UDT 860 - *MC_AXIS_REF*
    The data structure describes the structure of the parameters and status information of drives.
    General data structure for all drives and bus systems.
- FB 873 - *VMC_Init*Sigma7S_EC
  – The *Init*t block is used to configure an axis.
  – Specific block for *Sigma-7S* EtherCAT.
  – The configuration data for the initialization must be stored in the *axis DB*.
- FB 872 - *VMC_Kernel*Sigma7_EC
  – The *Kernel* block communicates with the drive via the appropriate bus system, processes the user requests and returns status messages.
  – Specific block for *Sigma-7* EtherCAT.
  – The exchange of the data takes place by means of the *axis DB*.
- FB 860 - *VMC_AxisControl*
  – General block for all drives and bus systems.
  – Supports simple motion commands and returns all relevant status messages.
  – The exchange of the data takes place by means of the *axis DB*.
  – For motion control and status query, via the instance data of the block you can link a visualization.
  – In addition to the FB 860 - *VMC_AxisControl*, *PLCopen* blocks can be used.
- FB 800 ... FB 838 - *PLCopen*
  – The PLCopen blocks are used to program motion sequences and status queries.
  – General blocks for all drives and bus systems.

3.2.4.3.2 Programming

Include library

1. Go to the service area of www.vipa.com.
2. Download the *Simple Motion Control* library from the download area at *'VIPA Lib'*.
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-7S* EtherCAT:
    – UDT 872 - VMC_ConfigSigma7EC_REF
    – FB 872 - VMC_KernelSigma7_EC
    – FB 873 - VMC_InitSigma7S_EC
  - Axis Control
    – UDT 860 - MC_AXIS_REF
    – Blocks for your movement sequences

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.

Create axis DB

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

   Specify the following parameters:

   - Name and type
     – The DB no. as *'Name'* can freely be chosen, such as DB10.
     – Set *'Shared DB'* as the *'Type'*.
   - Symbolic name
     – Specify "Axis01".

   Confirm your input with [OK].

   ⇨ The block is created.

2. Open DB10 "Axis01" by double-click.

   - In "Axis01", create the variable "Config" of type UDT 872. These are specific axis configuration data.
   - In "Axis01", create the variable "Axis" of type UDT 860. During operation, all operating data of the axis are stored here.

DB10

Address	Name	Type	...
		Struct	
...	Config	"VMC_ConfigSigma7EC_REF"	
...	Axis	"MC_AXIS_REF	
...		END_STRUCT	

OB 1

Configuration of the axis

Open OB 1 and program the following FB calls with associated DBs:

FB 873 - VMC_InitSigma7S_EC, DB 873 *Chap. 3.2.5.3 'FB 873 - VMC_Init-Sigma7S_EC - Sigma-7S EtherCAT Initialization' page 83*

At *InputsStartAddressPDO* respectively *OutputsStartAddressPDO*, enter the address from the *SPEED7 EtherCAT Manager*. *76*

```
CALL  "VMC_InitSigma7S_EC" , "DI_InitSgm7SETC01"
Enable                 :="InitS7SEC1_Enable"
LogicalAddress         :=300
InputsStartAddressPDO  :=300(EtherCAT-Man:S7 Input address)
OutputsStartAddressPDO:=300(EtherCAT-Man:S7 Output address)
EncoderType            :=1
EncoderResolutionBits  :=20
FactorPosition         :=1.048576e+006
FactorVelocity         :=1.048576e+006
FactorAcceleration     :=1.048576e+002
OffsetPosition         :=0.000000e+000
MaxVelocityApp         :=5.000000e+001
MaxAccelerationApp     :=1.000000e+002
MaxDecelerationApp     :=1.000000e+002
MaxVelocityDrive       :=6.000000e+001
MaxAccelerationDrive   :=1.500000e+002
MaxDecelerationDrive   :=1.500000e+002
MaxPosition            :=1.048500e+003
MinPosition            :=-1.048514e+003
EnableMaxPosition      :=TRUE
EnableMinPosition      :=TRUE
MinUserPosition        :="InitS5EC1_MinUserPos"
MaxUserPosition        :="InitS5EC1_MaxUserPos"
Valid                  :="InitS5EC1_Valid"
Error                  :="InitS5EC1_Error"
ErrorID                :="InitS5EC1_ErrorID"
Config                 :="Axis01".Config
Axis                   :="Axis01".Axis
```

Connecting the Kernel for the axis

The *Kernel* processes the user commands and passes them appropriately processed on to the drive via the respective bus system.

FB 872 - VMC_KernelSigma7_EC, DB 872 *Chap. 3.2.5.2 'FB 872 - VMC_Kernel-Sigma7_EC - Sigma-7 EtherCAT Kernel' page 83*

```
      CALL  "VMC_KernelSigma7_EC" , "DI_KernelSgm7ETC01"
       Init  :="KernelS7EC1_Init"
       Config:="Axis01".Config
       Axis  :="Axis01".Axis
```

Connecting the block for motion sequences

For simplicity, the connection of the FB 860 - VMC_AxisControl is to be shown here. This universal block supports simple motion commands and returns status messages. The inputs and outputs can be individually connected. Please specify the reference to the corresponding axis data at *'Axis'* in the *axis DB*.

FB 860 - VMC_AxisControl, DB 860 *Chap. 9.2.2 'FB 860 - VMC_AxisControl - Control block axis control' page 363*

```
CALL  "VMC_AxisControl" , "DI_AxisControl01"
 AxisEnable        :="AxCtrl1_AxisEnable"
 AxisReset         :="AxCtrl1_AxisReset"
 HomeExecute       :="AxCtrl1_HomeExecute"
 HomePosition      :="AxCtrl1_HomePosition"
 StopExecute       :="AxCtrl1_StopExecute"
 MvVelocityExecute:="AxCtrl1_MvVelExecute"
 MvRelativeExecute:="AxCtrl1_MvRelExecute"
 MvAbsoluteExecute:="AxCtrl1_MvAbsExecute"
 PositionDistance :="AxCtrl1_PositionDistance"
 Velocity          :="AxCtrl1_Velocity"
 Acceleration      :="AxCtrl1_Acceleration"
 Deceleration      :="AxCtrl1_Deceleration"
 JogPositive       :="AxCtrl1_JogPositive"
 JogNegative       :="AxCtrl1_JogNegative"
 JogVelocity       :="AxCtrl1_JogVelocity"
 JogAcceleration   :="AxCtrl1_JogAcceleration"
 JogDeceleration   :="AxCtrl1_JogDeceleration"
 AxisReady         :="AxCtrl1_AxisReady"
 AxisEnabled       :="AxCtrl1_AxisEnabled"
 AxisError         :="AxCtrl1_AxisError"
 AxisErrorID       :="AxCtrl1_AxisErrorID"
 DriveWarning      :="AxCtrl1_DriveWarning"
 DriveError        :="AxCtrl1_DriveError"
 DriveErrorID      :="AxCtrl1_DriveErrorID"
 IsHomed           :="AxCtrl1_IsHomed"
 ModeOfOperation   :="AxCtrl1_ModeOfOperation"
 PLCopenState      :="AxCtrl1_PLCopenState"
 ActualPosition    :="AxCtrl1_ActualPosition"
 ActualVelocity    :="AxCtrl1_ActualVelocity"
 CmdDone           :="AxCtrl1_CmdDone"
 CmdBusy           :="AxCtrl1_CmdBusy"
 CmdAborted        :="AxCtrl1_CmdAborted"
 CmdError          :="AxCtrl1_CmdError"
 CmdErrorID        :="AxCtrl1_CmdErrorID"
 DirectionPositive:="AxCtrl1_DirectionPos"
 DirectionNegative:="AxCtrl1_DirectionNeg"
 SWLimitMinActive :="AxCtrl1_SWLimitMinActive"
 SWLimitMaxActive :="AxCtrl1_SWLimitMaxActive"
 HWLimitMinActive :="AxCtrl1_HWLimitMinActive"
 HWLimitMaxActive :="AxCtrl1_HWLimitMaxActive"
 Axis              :="Axis01".Axis
```

For complex motion tasks, you can use the PLCopen blocks. Please specify the reference to the corresponding axis data at Axis in the axis DB.

Your project now includes the following blocks:

- OB 1 - Main
- OB 57 - DP Manufacturer Alarm
- OB 82 - I/O_FLT1
- OB 86 - Rack_FLT
- FB 860 - VMC_AxisControl with instance DB

- FB 872 - VMC_KernelSigma7_EC with instance DB
- FB 873 - VMC_InitSigma7S_EC with instance DB
- UDT 860 - MC_Axis_REF
- UDT 872 - VMC_ConfigSigma7EC_REF

Sequence of operations

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

   The transfer can only be done by the Siemens SIMATIC Manager - not hardware configurator!

   Since slave and module parameters are transmitted by means of SDO respectively SDO Init command, the configuration remains active, until a power cycle is performed or new parameters for the same SDO objects are transferred.

   With an overall reset the slave and module parameters are not reset!

   ⇨ 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. To do this, call the *Init* block FB 873 - VMC_InitSigma7S_EC with *Enable* = TRUE.

   ⇨ The output *Valid* returns TRUE. In the event of a fault, you can determine the error by evaluating the *ErrorID*.

   You have to call the *Init* block again if you load a new axis DB or you have changed parameters on the *Init* block.

   Do not continue until the Init block does not report any errors!

3. Ensure that the *Kernel* block FB 872 - VMC_KernelSigma7_EC is called cyclically. In this way, control signals are transmitted to the drive and status messages are reported.
4. Program your application with the FB 860 - VMC_AxisControl or with the PLCopen blocks.

Controlling the drive via HMI

There is the possibility to control your drive via HMI. For this, a predefined symbol library is available for Movicon to access the VMC_AxisControl function block. *Chap. 10 'Controlling the drive via HMI' page 432*

3.2.4.4 Copy project

Proceeding

In the example, the station *'Source'* is copied and saved as *'Target '*.

1. Open the hardware configuration of the *'Source'* CPU and start the *SPEED7 EtherCAT Manager*.
2. In the *SPEED7 EtherCAT Manager*, via *'File → Save as'* save the configuration in your working directory.

3. Close the *SPEED7 EtherCAT Manager* and the hardware configurator.
4. Copy the station *'Source'* with Ctrl + C and paste it as *'Target'* into your project with Ctrl + V.
5. Select the *'Blocks'* directory of the *'Target'* CPU and delete the *'System data'*.
6. Open the hardware configuration of the *'Target'* CPU. Adapt the IP address data or re-network the CPU or the CP again.

> *Before calling the SPEED7 EtherCAT Manager you have always to save your project with 'Station ➔ Save and compile'.*

7. Safe your project with *'Station ➔ Safe and compile'*.
8. Open the *SPEED7 EtherCAT Manager*.
9. Use *'File ➔ Open'* to load the configuration from your working directory.
10. Close the *SPEED7 EtherCAT Manager*.
11. Save and compile your configuration.

3.2.5 Drive specific blocks

The PLCopen blocks for axis control can be found here: *Chap. 9 'Blocks for axis control' page 361*

3.2.5.1 UDT 872 - VMC_ConfigSigma7EC_REF - *Sigma-7* EtherCAT Data structure axis configuration

This is a user-defined data structure that contains information about the configuration data. The UDT is specially adapted to the use of a *Sigma-7* drive, which is connected via EtherCAT.

3.2.5.2 FB 872 - VMC_KernelSigma7_EC - *Sigma-7* EtherCAT Kernel

Description

This block converts the drive commands for a *Sigma-7* axis via EtherCAT and communicates with the drive. For each *Sigma-7* axis, an instance of this FB is to be cyclically called.

Please note that this module calls the SFB 238 internally.

In the SPEED7 Studio, this module is automatically inserted into your project.

In Siemens SIMATIC Manager, you have to copy the SFB 238 from the Motion Control Library into your project.

Parameter	Declaration	Data type	Description
Init	INPUT	BOOL	The block is internally reset with an edge 0-1. Existing motion commands are aborted and the block is initialized.
Config	IN_OUT	UDT872	Data structure for transferring axis-dependent configuration data to the *AxisKernel*.
Axis	IN_OUT	MC_AXIS_REF	Data structure for transferring axis-dependent information to the *AxisKernel* and PLCopen blocks.

3.2.5.3 FB 873 - VMC_InitSigma7S_EC - *Sigma-7S* EtherCAT Initialization

Description

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

Parameter	Declaration	Data type	Description
Enable	INPUT	BOOL	Release of initialization
Logical address	INPUT	INT	Start address of the PDO input data
InputsStartAddressPDO	INPUT	INT	Start address of the input PDOs
OutputsStartAddressPDO	INPUT	INT	Start address of the output PDOs

Parameter	Declaration	Data type	Description
EncoderType	INPUT	INT	Encoder type ■ 1: Absolute encoder ■ 2: Incremental encoder
EncoderResolutionBits	INPUT	INT	Number of bits corresponding to one encoder revolution. Default: 20
FactorPosition	INPUT	REAL	Factor for converting the position of user units [u] into drive units [increments] and back. It's valid: $p_{[increments]} = p_{[u]}$ x *FactorPosition* Please consider the factor which can be specified on the drive via the objects 0x2701: 1 and 0x2701: 2. This should be 1.
Velocity Factor	INPUT	REAL	Factor for converting the speed of user units [u/s] into drive units [increments/s] and back. It's valid: $v_{[increments/s]} = v_{[u/s]}$ x *FactorVelocity* Please also take into account the factor which you can specify on the drive via objects 0x2702: 1 and 0x2702: 2. This should be 1.
FactorAcceleration	INPUT	REAL	Factor to convert the acceleration of user units [u/s^2] in drive units [10^{-4} x increments/s^2] and back. It's valid: 10^{-4} x $a_{[increments/s^2]} = a_{[u/s^2]}$ x *FactorAcceleration* Please also take into account the factor which you can specify on the drive via objects 0x2703: 1 and 0x2703: 2. This should be 1.
OffsetPosition	INPUT	REAL	Offset for the zero position [u].
MaxVelocityApp	INPUT	REAL	Maximum application speed [u/s]. The command inputs are checked to the maximum value before execution.
MaxAccelerationApp	INPUT	REAL	Maximum acceleration of application [u/s^2]. The command inputs are checked to the maximum value before execution.
MaxDecelerationApp	INPUT	REAL	Maximum application delay [u/s^2]. The command inputs are checked to the maximum value before execution.
MaxPosition	INPUT	REAL	Maximum position for monitoring the software limits [u].
MinPosition	INPUT	REAL	Minimum position for monitoring the software limits [u].
EnableMaxPosition	INPUT	BOOL	Monitoring maximum position ■ TRUE: Activates the monitoring of the maximum position.
EnableMinPosition	INPUT	BOOL	Monitoring minimum position ■ TRUE: Activation of the monitoring of the minimum position.
MinUserPosition	OUTPUT	REAL	Minimum user position based on the minimum encoder value of 0x80000000 and the *FactorPosition* [u].

Parameter	Declaration	Data type	Description
MaxUserPosition	OUTPUT	REAL	Maximum user position based on the maximum encoder value of 0x7FFFFFFF and the *FactorPosition* [u].
Valid	OUTPUT	BOOL	Initialization ■ TRUE: Initialization is valid.
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. 12 'ErrorID - Additional error information' page 457*
Config	IN_OUT	UDT872	Data structure for transferring axis-dependent configuration data to the *AxisKernel*.
Axis	IN_OUT	MC_AXIS_REF	Data structure for transferring axis-dependent information to the *AxisKernel* and PLCopen blocks.

3.3 Usage *Sigma-7W* EtherCAT

3.3.1 Overview

Usage of the single-axis drive *Chap. 3.2 'Usage Sigma-7S EtherCAT' page 47*

Precondition

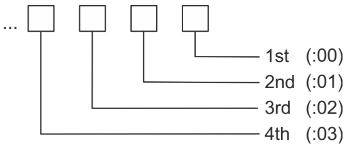
- SPEED7 Studio from V1.6.1
  or
- Siemens SIMATIC Manager from V 5.5, SP2 & *SPEED7 EtherCAT Manager* & *Simple Motion Control Library*
- CPU with EtherCAT master, e.g. CPU 015-CEFNR00
- *Sigma-7W* Double-axis drive with EtherCAT option card

Steps of configuration

1. Set the parameters on the drive
   - The setting of the parameters happens by means of the software tool *Sigma Win+*.
2. Hardware configuration in VIPA *SPEED7 Studio* or Siemens SIMATIC Manager
   - Configuring a CPU with EtherCAT master functionality
   - Configuration of the *Sigma-7W* EtherCAT double axes.
   - Configuring the EtherCAT connection via *SPEED7 EtherCAT Manager*
3. Programming in VIPA *SPEED7 Studio* or Siemens SIMATIC Manager
   - *Init* block for the configuration of the double axes.
   - *Kernel* block for communication with one axis each.
   - Connecting the blocks for motion sequences.

3.3.2 Set the parameters on the drive

Parameter digits



CAUTION!
Before the commissioning, you have to adapt your drive to your application with the *Sigma Win+* software tool! More may be found in the manual of your drive.

The following parameters must be set via *Sigma Win+* to match the *Simple Motion Control Library*:

Axis 1 - Module 1 (24bit encoder)

Servopack Parameter	Address:digit	Name	Value
Pn205	(2205h)	Multiturn Limit Setting	65535
Pn20E	(220Eh)	Electronic Gear Ratio (Numerator)	16
Pn210	(2210h)	Electronic Gear Ratio (Denominator)	1
PnB02	(2701h:01)	Position User Unit (Numerator)	1
PnB04	(2701h:02)	Position User Unit (Denominator)	1
PnB06	(2702h:01)	Velocity User Unit (Numerator)	1
PnB08	(2702h:02)	Velocity User Unit (Denominator)	1
PnB0A	(2703h:01)	Acceleration User Unit (Numerator)	1
PnB0C	(2703h:02)	Acceleration User Unit (Denominator)	1

Axis 2 - Module 2 (24Bit Encoder)

Servopack Parameter	Address:digit	Name	Value
Pn205	(2A05h)	Multiturn Limit Setting	65535
Pn20E	(2A0Eh)	Electronic Gear Ratio (Numerator)	16
Pn210	(2A10h)	Electronic Gear Ratio (Denominator)	1
PnB02	(2F01h:01)	Position User Unit (Numerator)	1
PnB04	(2F01h:02)	Position User Unit (Denominator)	1
PnB06	(2F02h:01)	Velocity User Unit (Numerator)	1
PnB08	(2F02h:02)	Velocity User Unit (Denominator)	1
PnB0A	(2F03h:01)	Acceleration User Unit (Numerator)	1
PnB0C	(2F03h:02)	Acceleration User Unit (Denominator)	1

Please note that you have to enable the corresponding direction of your axis in accordance to your requirements. For this use the parameters Pn50A (P-OT) respectively Pn50B (N-OT) in Sigma Win+.

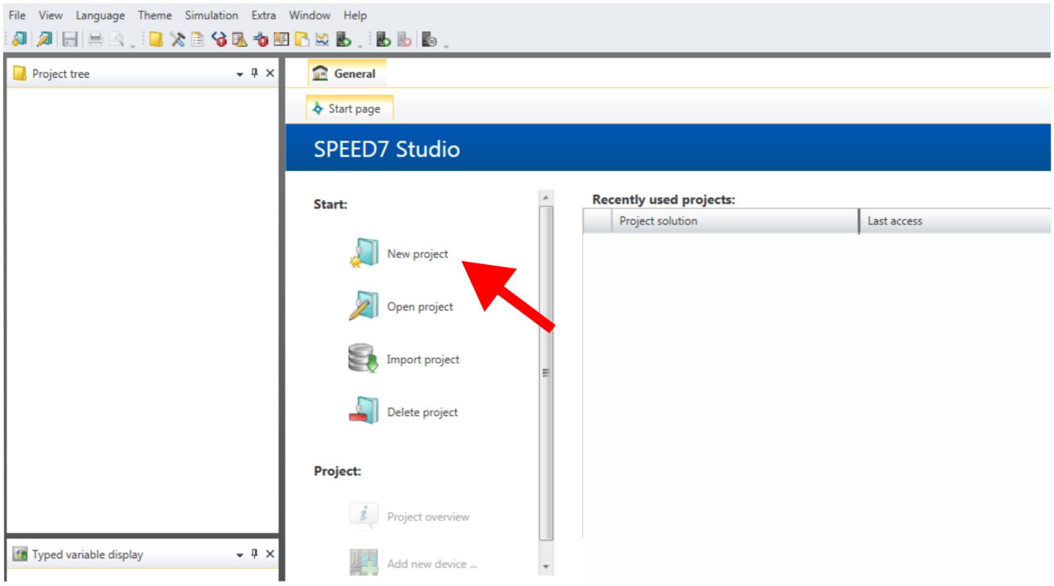
3.3.3 Usage in VIPA *SPEED7 Studio*

3.3.3.1 Hardware configuration

Add CPU in the project

Please use for configuration the *SPEED7 Studio* V1.6.1 and up.

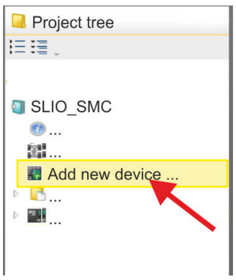
1. Start the *SPEED7 Studio*.

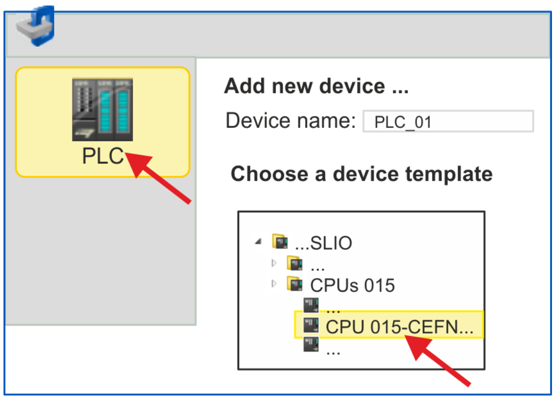


2. Create a new project at the start page with *'New project'*.

   ⇨ A new project is created and the view *'Devices and networking'* is shown.

3. Click in the *Project tree* at *'Add new device ...'*.





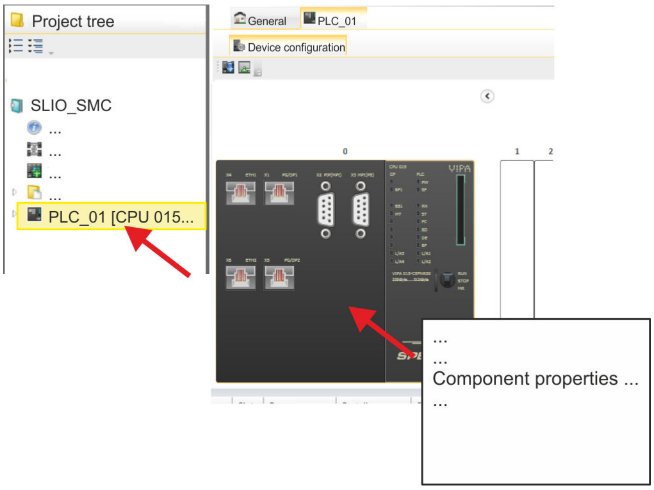
   ⇨ A dialog for device selection opens.

4. Select from the *'Device templates'* a CPU with EtherCAT master functions such as CPU 015-CEFNR00 and click at [OK].

   ⇨ The CPU is inserted in *'Devices and networking'* and the *'Device configuration'* is opened.

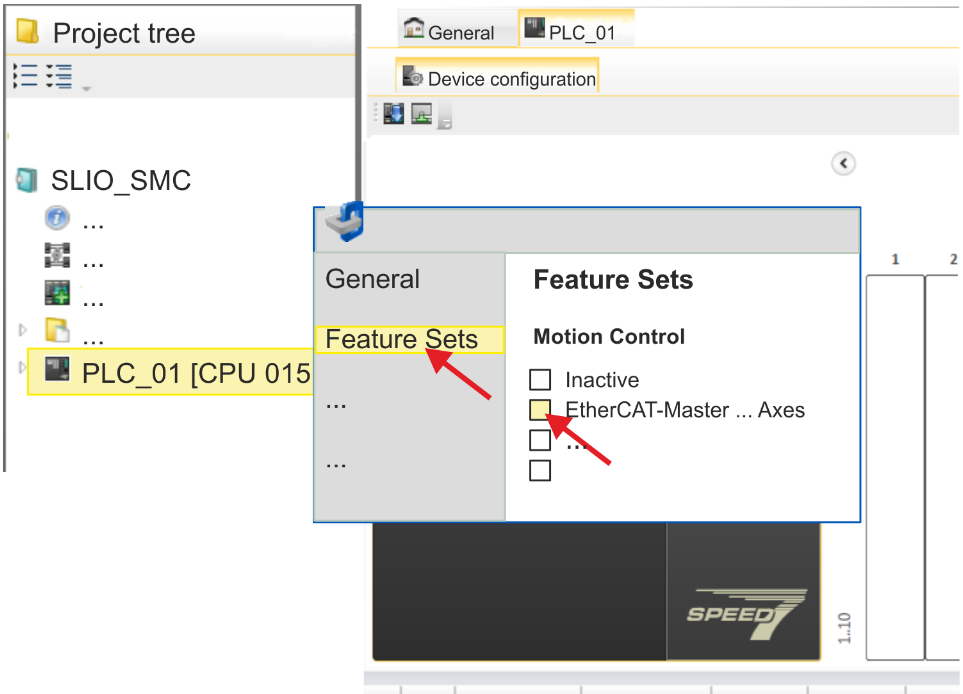
Activate motion control functions

If the EtherCAT master functionality is not yet activated on your CPU, the activation takes place as follows:



1. Click at the CPU in the *'Device configuration'* and select *'Context menu ➔ Components properties'*.

   ⇨ The properties dialog of the CPU is opened.



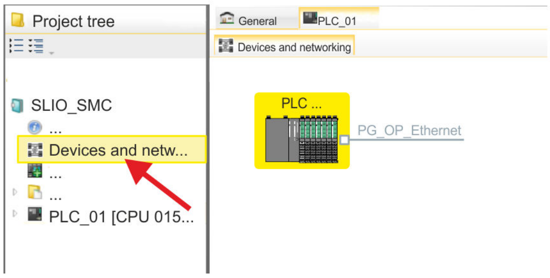
2. Click at *'Feature Sets'* and activate at *'Motion Control'* the parameter *'EtherCAT-Master... Axes'*. The number of axes is not relevant in this example.
3. Confirm your input with [OK].

   ⇨ The motion control functions are now available in your project.

> **CAUTION!**
> Please note due to the system, with every change to the feature set settings, the EtherCAT field bus system and its motion control configuration will be deleted from your project!

Configuration of Ethernet PG/OP channel

1. Click in the *Project tree* at *'Devices and networking'*.
   ⇨ You will get a graphical object view of your CPU.



2. Click at the network *'PG_OP_Ethernet'*.
3. Select *'Context menu ➔ Interface properties'*.
   ⇨ A dialog window opens. Here you can enter the IP address data for your Ethernet PG/OP channel. You get valid IP address parameters from your system administrator.
4. Confirm with [OK].
   ⇨ The IP address data are stored in your project listed in *'Devices and networking'* at *'Local components'*.

   After transferring your project your CPU can be accessed via Ethernet PG/OP channel with the set IP address data.

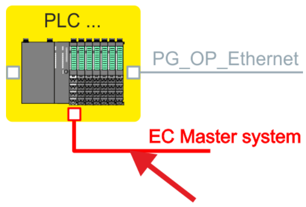
Installing the ESI file

For the *Sigma-7* EtherCAT drive can be configured in the *SPEED7 EtherCAT Manager*, the corresponding ESI file must be installed. Usually, the *SPEED7 Studio* is delivered with current ESI files and you can skip this part. If your ESI file is not up-to date, you will find the latest ESI file for the *Sigma-7* EtherCAT drive under www.yaskawa.eu.com at *'Service ➔ Drives & Motion Software'*.

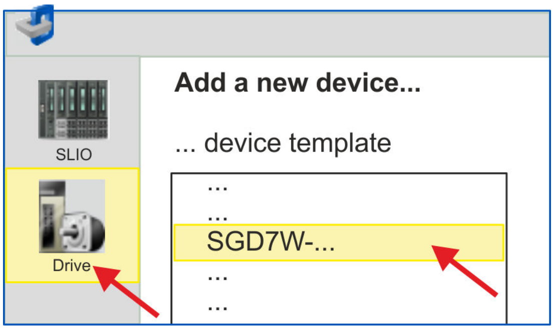
1. Download the according ESI file for your drive. Unzip this if necessary.
2. Navigate to your *SPEED7 Studio*.
3. Open the corresponding dialog window by clicking on *'Extra ➔ Install device description (EtherCAT - ESI)'*.
4. Under *'Source path'*, specify the ESI file and install it with [Install].
   ⇨ The devices of the ESI file are now available.

***Sigma-7W* add a double-axis drive**

1. Click in the Project tree at *'Devices and networking'*.
2. Click here at *'EC-Mastersystem'* and select *'Context menu ➔ Add new device'*.

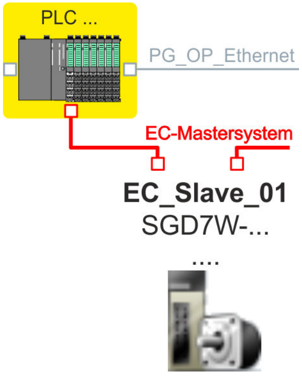


   ⇨ The device template for selecting an EtherCAT device opens.



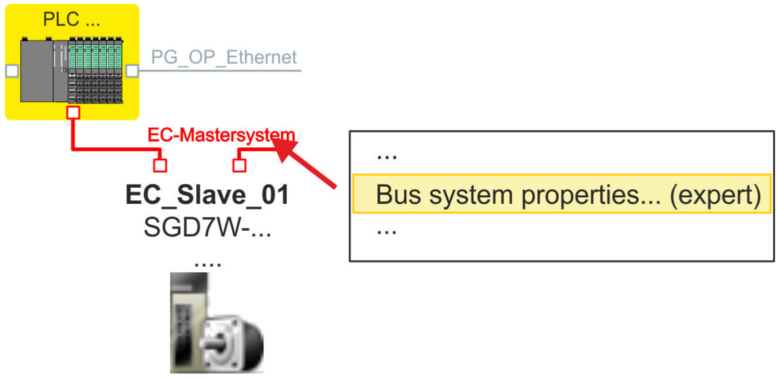
3. Select your *Sigma-7W* double-axis drive:
   - SGD7W-xxxxA0 ...

   Confirm your input with [OK]. If your drive does not exist, you must install the corresponding ESI file as described above.



⇨ The *Sigma-7W* double-axis drive is connected to your EC master system.

Configure *Sigma-7W* double-axis drive

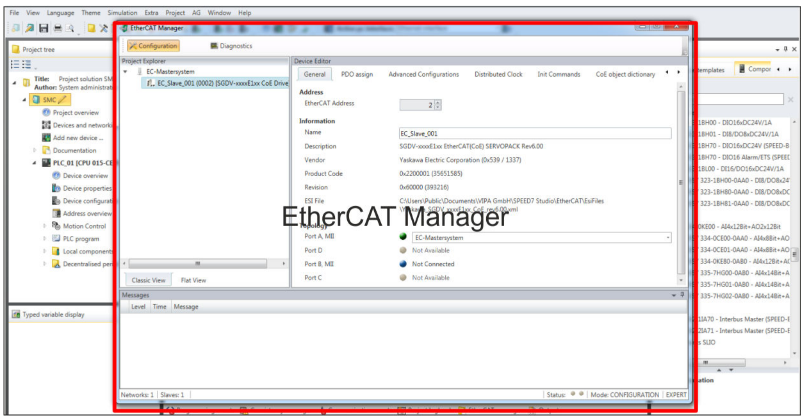


1. Click here at *'EC-Mastersystem'* and select *'Context menu ➔ Bus system properties (expert)'*.

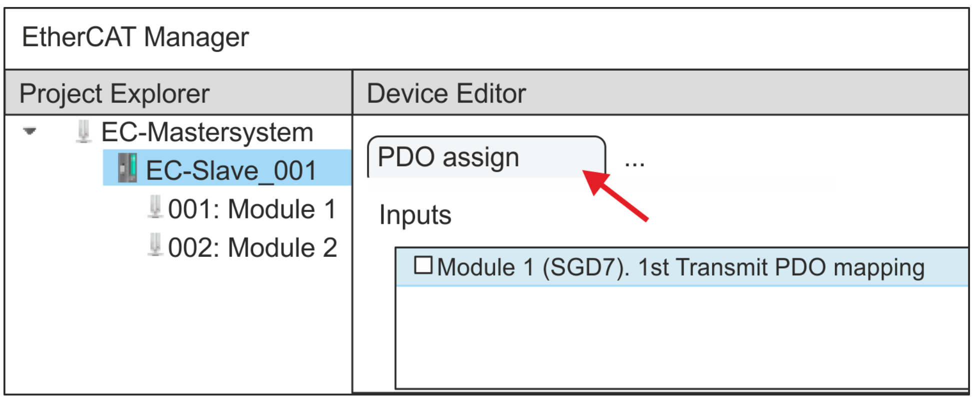
> *You can only edit PDOs in 'Expert mode'! Otherwise, the buttons are hidden.*

⇨ The *SPEED7 EtherCAT Manager* opens. Here you can configure the EtherCAT communication to your *Sigma-7W* double-axis drive.

More information about the usage of the *SPEED7 EtherCAT Manager* may be found in the online help of the *SPEED7 Studio*.



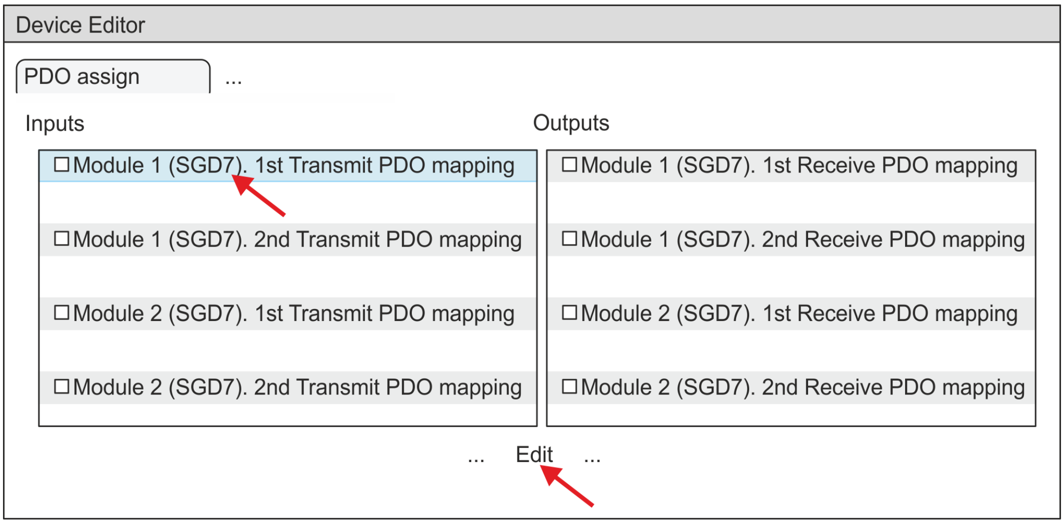
2. Click on the slave in the *SPEED7 EtherCAT Manager* and select the *'PDO assign'* tab in the *'Device editor'*.



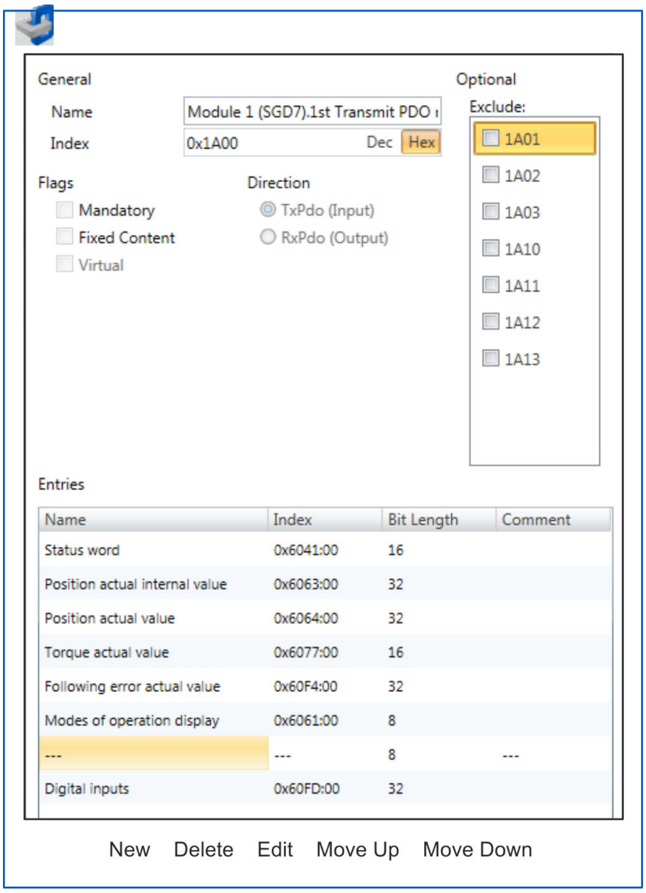
⇨ This dialogue shows a list of the PDOs for *'Module 1'* (axis 1) and *'Module 2'* (axis 2).

3. By selecting the appropriate mapping, you can edit the PDOs with [Edit]. Select the mapping *'Module 1 (SGD7). 1st Transmit PDO mapping'* and click at [Edit].

> *Please note that some PDOs can not be edited because of the default settings. By de-activating already activated PDOs, you can release the processing of locked PDOs.*



⇨ The dialog *'Edit PDO'* is opened. Please check the PDO settings listed here and adjust them if necessary. Please also take into account the order of the *'Entries'* and add them accordingly.



The following functions are available for editing the *'Entries'*:

- New
  – Here you can create a new entry in a dialog by selecting the corresponding entry from the *'CoE object dictionary'* and making your settings. The entry is accepted with [OK] and is listed in the list of entries.
- Delete
  – This allows you to delete a selected entry.
- Edit
  – This allows you to edit the general data of an entry.
- Move Up/Down
  – This allows you to move the selected entry up or down in the list.

4. Perform the following settings for the Transmit PDOs:

Inputs: 1st Transmit PDO

Module 1 (SGD7). 1st Transmit PDO mapping	Module 2 (SGD7). 1st Transmit PDO mapping
Name: Module 1 (SGD7). 1st Transmit PDO mapping	Name: Module 2 (SGD7). 1st Transmit PDO mapping
Index: 0x1A00	Index: 0x1A10
Flags: Everything de-activated	
Direction TxPdo (Input): activated	
Exclude: 1A01: de-activated	1A11: de-activated
Please note these settings, otherwise the PDO mappings can not be activated at the same time!	

Entries	Module 1 (axis 1)	Module 2 (axis 2)	Bit length
Name	**Index**	**Index**	
Status word	0x6041:00	0x6841: 00	16bit
Position actual internal value	0x6063:00	0x6863:00	32bit
Position actual value	0x6064:00	0x6864:00	32bit
Torque actual value	0x6077:00	0x6877:00	16bit
Following error actual value	0x60F4:00	0x68F4:00	32bit
Modes of operation display	0x6061:00	0x6861:00	8bit
---	---	---	8bit
Digital inputs	0x60FD:00	0x68FD:00	32bit

Inputs: 2nd Transmit PDO

Module 1 (SGD7). 2nd Transmit PDO mapping	Module 2 (SGD7). 2nd Transmit PDO mapping
Name: Module 1 (SGD7). 2nd Transmit PDO mapping	Name: Module 2 (SGD7). 2nd Transmit PDO mapping
Index: 0x1A01	Index: 0x1A11
Flags: Everything de-activated	
Direction TxPdo (Input): activated	
Exclude: 1A00, 1A02, 1A03: de-activated	1A10, 1A12, 1A13: de-activated
Please note these settings, otherwise the PDO mappings can not be activated at the same time!	

Entries	Module 1 (axis 1)	Module 2 (axis 2)	Bit length
Name	**Index**	**Index**	
Touch probe status	0x60B9:00	0x68B9:00	16bit
Touch probe 1 position value	0x60BA:00	0x68BA:00	32bit
Touch probe 2 position value	0x60BC:00	0x68BC:00	32bit
Velocity actual value	0x606C:00	0x686C:00	32bit

5. Perform the following settings for the Receive PDOs:

Outputs: 1st Receive PDO

Module 1 (SGD7). 1st Receive PDO	Module 2 (SGD7). 1st Receive PDO
Name: Module 1 (SGD7). 1st Receive PDO mapping	Name: Module 2 (SGD7). 1st Receive PDO mapping
Index: 0x1600	Index: 0x1610
Flags: Everything de-activated	
Direction RxPdo (Output): activated	
Exclude: 1601, 1602, 1603: de-activated	1611, 1612, 1613: de-activated
Please note these settings, otherwise the PDO mappings can not be activated at the same time!	

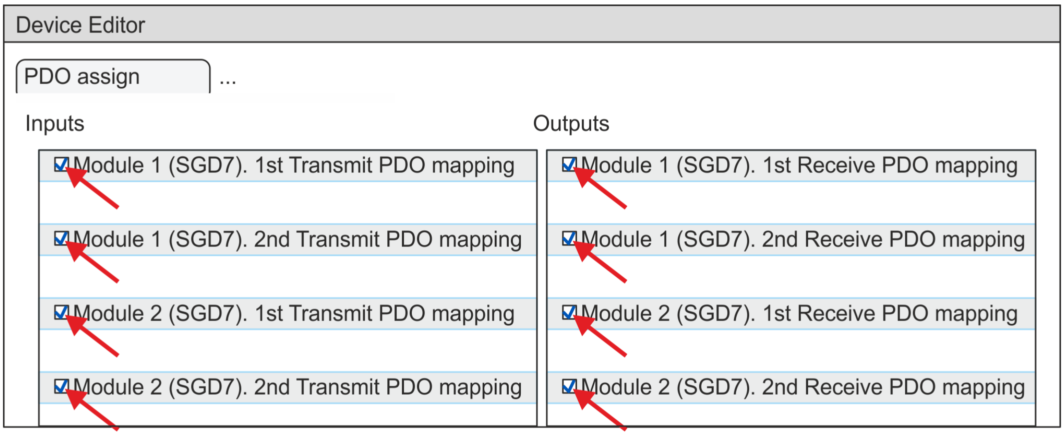
Entries	Module 1 (axis 1)	Module 2 (axis 2)	Bit length
Name	**Index**	**Index**	
Control word	0x6040:00	0x6840: 00	16bit
Target position	0x607A:00	0x687A: 00	32bit
Target velocity	0x60FF:00	0x68FF: 00	32bit
Modes of operation	0x6060:00	0x6860: 00	8bit
---	---	---	8bit
Touch probe function	0x60B8:00	0x68B8: 00	16bit

Outputs: 2nd Receive PDO

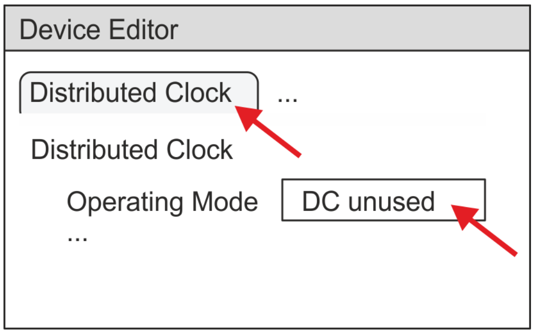
Module 1 (SGD7). 2nd Receive PDO	Module 2 (SGD7). 2nd Receive PDO
Name: Module 1 (SGD7). 2nd Receive PDO mapping	Name: Module 2 (SGD7). 2nd Receive PDO mapping
Index: 0x1601	Index: 0x1611
Flags: Everything de-activated	
Direction RxPdo (Output): activated	
Exclude: 1600, 1602, 1603: de-activated	1610, 1612, 1613: de-activated
Please note these settings, otherwise the PDO mappings can not be activated at the same time!	

Entries	Module 1 (axis 1)	Module 2 (axis 2)	Bit length
Name	**Index**	**Index**	
Profile velocity	0x6081:00	0x6881: 00	32bit
Profile acceleration	0x6083:00	0x6883: 00	32bit
Profile deceleration	0x6084:00	0x6884: 00	32bit

6. For *'Module 1'* and *'Module 2'* in PDO assignment, activate the PDOs 1 and 2 for the inputs and outputs. All subsequent PDOs must remain de-activated. If this is not possible, please check the respective PDO parameter *'Exclude'*.

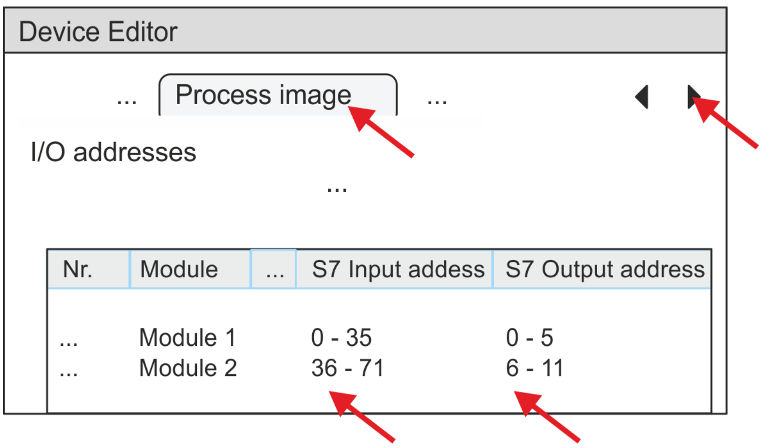


7. In the *'Device Editor'* of the *SPEED7 EtherCAT Manager*, select the *'Distributed clocks'* tab and set *'DC unused'* as *'Operating mode'*.

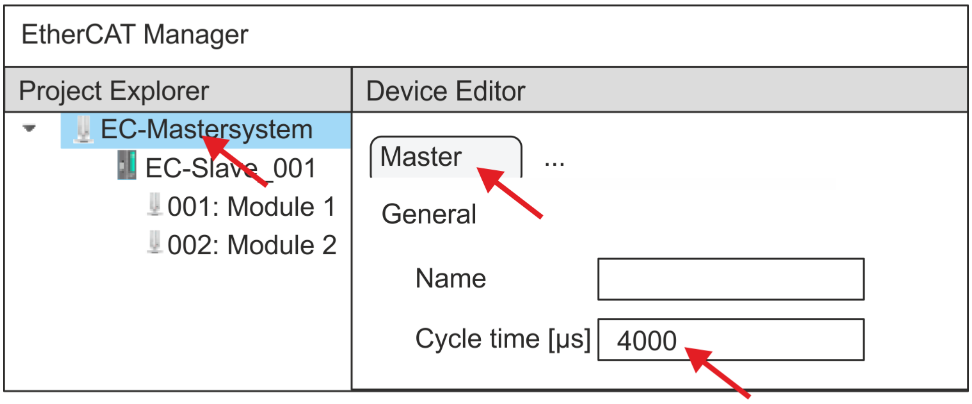


8. Select the *'Process image'* tab in the *'device editor'* using the arrow key and note the following PDO start addresses for the parameters of the block FB 874 - VMC_InitSigma7W_EC:

- Module 1: *'S7 Input address'* → *'M1_PdoInputs'* (here 0)
- Module 2: *'S7 Input address'* → *'M2_PdoInputs'* (here 36)
- Module 1: *'S7 Output address'* → *'M1_PdoOutputs'* (here 0)
- Module 2: *'S7 Output address'* → *'M2_PdoOutputs'* (here 36)



9. Click on *'EC-Mastersystem'* in the *SPEED7 EtherCAT Manager* and select the *'Master'* tab in the *'Device editor'*.

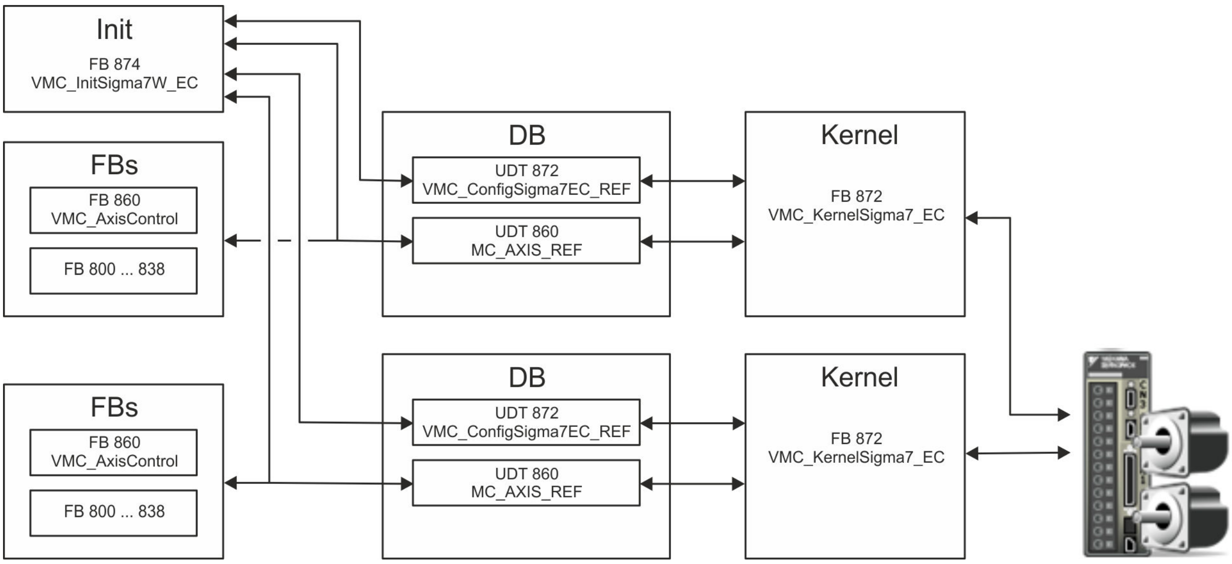


⇨ Set a cycle time of at least 4ms for Sigma-7W (400V) drives.

10. By closing the dialog of the *SPEED7 EtherCAT Manager* with [X] the configuration is taken to the *SPEED7 Studio*.

3.3.3.2 User program

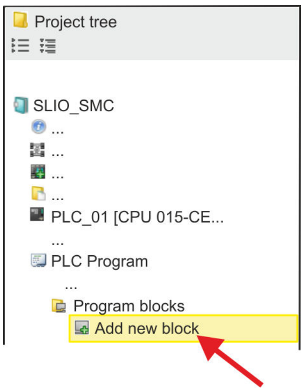
3.3.3.2.1 Program structure



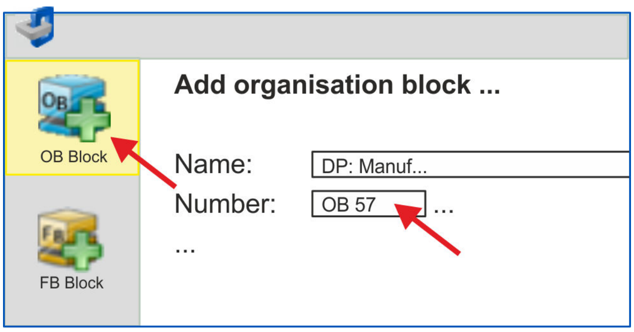
- DB
  A data block (axis DB) for configuration and status data must be created for each axis of a drive. The data block consists of the following data structures:
  – UDT 872 - *VMC_Config*Sigma7EC_REF
    The data structure describes the structure of the configuration of the drive.
    Specific data structure for *Sigma-7* EtherCAT.
  – UDT 860 - *MC_AXIS_REF*
    The data structure describes the structure of the parameters and status information of drives.
    General data structure for all drives and bus systems.
- FB 874 - *VMC_Init*Sigma7W_EC
  – The *Init*t block is used to configure the double-axis drive.
  – Specific block for *Sigma-7W* EtherCAT.
  – The configuration data for the initialization must be stored in the *axis DB*.
- FB 872 - *VMC_Kernel*Sigma7_EC
  – The *Kernel* block communicates with the drive via the appropriate bus system, processes the user requests and returns status messages.
  – The FB 872 - *VMC_Kernel*Sigma7_EC must be called for each axis.
  – Specific block for *Sigma-7* EtherCAT.
  – The exchange of the data takes place by means of the *axis DB*.
- FB 860 - *VMC_AxisControl*
  – General block for all drives and bus systems.
  – The FB 860 - *VMC_AxisControl* must be called for each axis.
  – Supports simple motion commands and returns all relevant status messages.
  – The exchange of the data takes place by means of the *axis DB*.
  – For motion control and status query, via the instance data of the block you can link a visualization.
  – In addition to the FB 860 - *VMC_AxisControl*, *PLCopen* blocks can be used.
- FB 800 ... FB 838 - PLCopen
  – The PLCopen blocks are used to program motion sequences and status queries.
  – The PLCopen blocks must be called for each axis.

3.3.3.2.2 Programming

Copy blocks into project

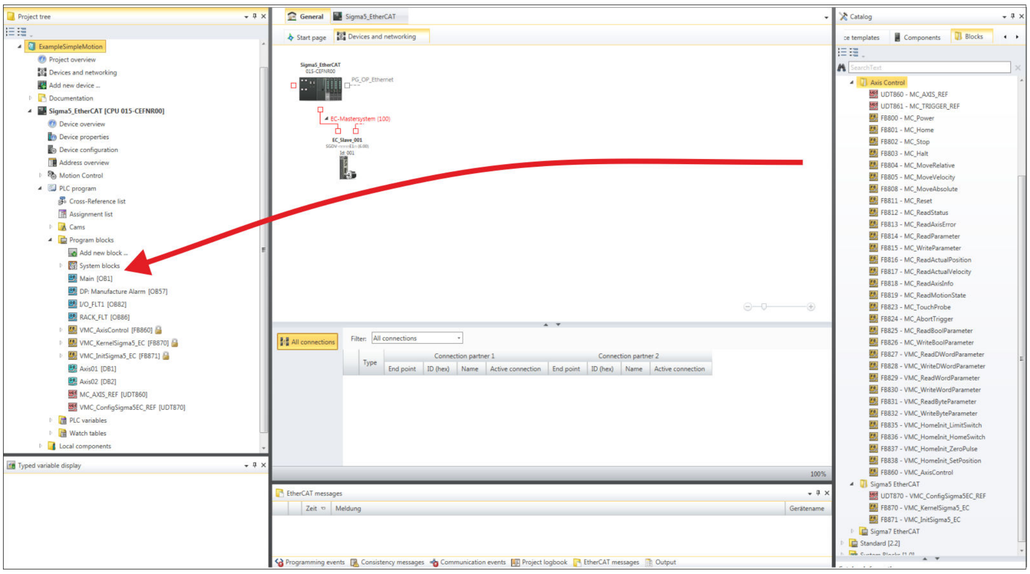


1. Click in the *Project tree* within the CPU at *'PLC program'*, *'Program blocks'* at *'Add New block'*.



⇨ The dialog *'Add block'* is opened.

2. Select the block type *'OB block'* and add one after the other OB 57, OB 82 and OB 86 to your project.



3. In the *'Catalog'*, open the *'Simple Motion Control'* library at *'Blocks'* and drag and drop the following blocks into *'Program blocks'* of the *Project tree*:

- *Sigma-7* EtherCAT:
  - UDT 872 - VMC_ConfigSigma7EC_REF
  - FB 872 - VMC_KernelSigma7_EC
  - FB 874 - VMC_InitSigma7W_EC
- Axis Control
  - UDT 860 - MC_AXIS_REF
  - Blocks for your movement sequences

Create axis DB for *'Module 1'*

1. Add a new DB as your *axis DB* to your project. Click in the *Project tree* within the CPU at *'PLC program'*, *'Program blocks'* at *'Add New block'*, select the block type *'DB block'* and assign the name "Axis01" to it. The DB number can freely be selected such as DB 10.

⇨ The block is created and opened.

2.
- In "Axis01", create the variable "Config" of type UDT 872. These are specific axis configuration data.
- In "Axis01", create the variable "Axis" of type UDT 860. During operation, all operating data of the axis are stored here.

Axis01 [DB10]

Data block structure

	Addr...	Name	Data type	...
	...	Config	UDT	[872]
	...	Axis	UDT	[860]

Create axis DB for *'Module 2'*

1. Add another DB as your *axis DB* to your project and assign it the name "Axis02". The DB number can freely be selected such as DB 11.

⇨ The block is created and opened.

2.
- In "Axis02", create the variable "Config" of type UDT 872. These are specific axis configuration data.
- In "Axis02", create the variable "Axis" of type UDT 860. During operation, all operating data of the axis are stored here.

Axis02 [DB11]
Data block structure

		Addr...	Name	Data type	...
		...	Config	UDT	[872]
		...	Axis	UDT	[860]

OB 1

Configuration of the double-axis

Open OB 1 and program the following FB calls with associated DBs:

FB 874 - VMC_InitSigma7W_EC, DB 874 *Chap. 3.3.5.3 'FB 874 - VMC_InitSigma7W_EC - Sigma-7W EtherCAT Initialization' page 124*

At *M1/M2_PdoInputs* respectively *M1/M2_PdoOutputs*, enter the address from the *SPEED7 EtherCAT Manager* for the according axis. *97*

```
⇨ CALL  "VMC_InitSigma7W_EC" , "DI_InitSgm7WETC01"
  Enable                      :=TRUE
  LogicalAddress              :=0
  M1_PdoInputs                :=0 (EtherCAT-Manager
                                   Module1: S7 Input address)

  M1_PdoOutputs               :=0 (EtherCAT-Manager
                                   Module1: S7 Output address)
  M1_EncoderType              :=2
  M1_EncoderResolutionBits    :=20
  M1_FactorPosition           :=1.048576e+006
  M1_FactorVelocity           :=1.048576e+006
  M1_FactorAcceleration       :=1.048576e+002
  M1_OffsetPosition           :=0.000000e+000
  M1_MaxVelocityApp           :=5.000000e+001
  M1_MaxAccelerationApp       :=1.000000e+002
  M1_MaxDecelerationApp       :=1.000000e+002
  M1_MaxVelocityDrive         :=6.000000e+001
  M1_MaxAccelerationDrive     :=1.500000e+002
  M1_MaxDecelerationDrive     :=1.500000e+002
  M1_MaxPosition              :=1.048500e+003
  M1_MinPosition              :=-1.048514e+003
  M1_EnableMaxPosition        :=TRUE
  M1_EnableMinPosition        :=TRUE
  M2_PdoInputs                :=36 (EtherCAT-Manager
                                   Module2: S7 Input address)

  M2_PdoOutputs               :=36 (EtherCAT-Manager
                                   Module2: S7 Output address)
  M2_EncoderType              :=2
  M2_EncoderResolutionBits    :=20
  M2_FactorPosition           :=1.048576e+006
  M2_FactorVelocity           :=1.048576e+006
  M2_FactorAcceleration       :=1.048576e+002
  M2_OffsetPosition           :=0.000000e+000
  M2_MaxVelocityApp           :=5.000000e+001
  M2_MaxAccelerationApp       :=1.000000e+002
  M2_MaxDecelerationApp       :=1.000000e+002
  M2_MaxVelocityDrive         :=6.000000e+001
  M2_MaxAccelerationDrive     :=1.500000e+002
  M2_MaxDecelerationDrive     :=1.500000e+002
  M2_MaxPosition              :=1.048500e+003
  M2_MinPosition              :=-1.048514e+003
  M2_EnableMaxPosition        :=TRUE
  M2_EnableMinPosition        :=TRUE
  M1_MinUserPosition          :=-1000.0
  M1_MaxUserPosition          :=1000.0
  M2_MinUserPosition          :=-1000.0
  M2_MaxUserPosition          :=1000.0
  Valid                       :="InitS7WEC1_Valid"
  Error                       :="InitS7WEC1_Error"
```

```
ErrorID                  :="InitS7WEC1_ErrorID"
M1_Config                :="Axis01".Config
M1_Axis                  :="Axis01".Axis
M2_Config                :="Axis02".Config
M2_Axis                  :="Axis02".Axis
```

Connecting the kernel for the respective axis

The *Kernel* processes the user commands and passes them appropriately processed on to the drive via the respective bus system.

FB 872 - VMC_KernelSigma7_EC, DB 872 for axis 1

FB 872 - VMC_KernelSigma7_EC, DB 1872 for axis 2 *Chap. 3.2.5.2 'FB 872 - VMC_KernelSigma7_EC - Sigma-7 EtherCAT Kernel' page 83*

⇨

```
CALL  "VMC_KernelSigma7_EC" , DB 872
 Init  :="KernelS7WEC1_Init"
 Config:="Axis01".Config
 Axis  :="Axis01".Axis

CALL  "VMC_KernelSigma7_EC" , DB 1872
 Init  :="KernelS7WEC2_Init"
 Config:="Axis02".Config
 Axis  :="Axis02".Axis
```

Connecting the block for motion sequences

For simplicity, the connection of the FB 860 - VMC_AxisControl is to be shown here. This universal block supports simple motion commands and returns status messages. The inputs and outputs can be individually connected. Please specify the reference to the corresponding axis data at *'Axis'* in the *axis DB*.

FB 860 - VMC_AxisControl, DB 860 ↳ *Chap. 9.2.2 'FB 860 - VMC_AxisControl - Control block axis control' page 363*

⇨
```
CALL  "VMC_AxisControl" , "DI_AxisControl01"
 AxisEnable        :="AxCtrl1_AxisEnable"
 AxisReset         :="AxCtrl1_AxisReset"
 HomeExecute       :="AxCtrl1_HomeExecute"
 HomePosition      :="AxCtrl1_HomePosition"
 StopExecute       :="AxCtrl1_StopExecute"
 MvVelocityExecute:="AxCtrl1_MvVelExecute"
 MvRelativeExecute:="AxCtrl1_MvRelExecute"
 MvAbsoluteExecute:="AxCtrl1_MvAbsExecute"
 PositionDistance :="AxCtrl1_PositionDistance"
 Velocity          :="AxCtrl1_Velocity"
 Acceleration      :="AxCtrl1_Acceleration"
 Deceleration      :="AxCtrl1_Deceleration"
 JogPositive       :="AxCtrl1_JogPositive"
 JogNegative       :="AxCtrl1_JogNegative"
 JogVelocity       :="AxCtrl1_JogVelocity"
 JogAcceleration   :="AxCtrl1_JogAcceleration"
 JogDeceleration   :="AxCtrl1_JogDeceleration"
 AxisReady         :="AxCtrl1_AxisReady"
 AxisEnabled       :="AxCtrl1_AxisEnabled"
 AxisError         :="AxCtrl1_AxisError"
 AxisErrorID       :="AxCtrl1_AxisErrorID"
 DriveWarning      :="AxCtrl1_DriveWarning"
 DriveError        :="AxCtrl1_DriveError"
 DriveErrorID      :="AxCtrl1_DriveErrorID"
 IsHomed           :="AxCtrl1_IsHomed"
 ModeOfOperation   :="AxCtrl1_ModeOfOperation"
 PLCopenState      :="AxCtrl1_PLCopenState"
 ActualPosition    :="AxCtrl1_ActualPosition"
 ActualVelocity    :="AxCtrl1_ActualVelocity"
 CmdDone           :="AxCtrl1_CmdDone"
 CmdBusy           :="AxCtrl1_CmdBusy"
 CmdAborted        :="AxCtrl1_CmdAborted"
 CmdError          :="AxCtrl1_CmdError"
 CmdErrorID        :="AxCtrl1_CmdErrorID"
 DirectionPositive:="AxCtrl1_DirectionPos"
 DirectionNegative:="AxCtrl1_DirectionNeg"
 SWLimitMinActive :="AxCtrl1_SWLimitMinActive"
 SWLimitMaxActive :="AxCtrl1_SWLimitMaxActive"
 HWLimitMinActive :="AxCtrl1_HWLimitMinActive"
 HWLimitMaxActive :="AxCtrl1_HWLimitMaxActive"
 Axis              :="Axis...".Axis
```

At *Axis*, enter "Axis01" for axis 1 and "Axis02" for axis 2.

> *For complex motion tasks, you can use the PLCopen blocks. Here you must also specify the reference to the corresponding axis data at Axis in the axis DB.*

Your project now includes the following blocks:

- OB 1 - Main
- OB 57 - DP Manufacturer Alarm
- OB 82 - I/O_FLT1
- OB 86 - Rack_FLT

- FB 860 - VMC_AxisControl with instance DB
- FB 872 - VMC_KernelSigma7_EC with instance DB
- FB 874 - VMC_InitSigma7W_EC with instance DB
- UDT 860 - MC_Axis_REF
- UDT 872 - VMC_ConfigSigma7EC_REF

Sequence of operations

1. Select *'Project ➔ Compile all'* and transfer the project into your CPU.

   You can find more information on the transfer of your project in the online help of the *SPEED7 Studio*.

   ⇨ You can take your application into operation now.

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

2. Before the double-axis drive can be controlled, it must be initialized. To do this, call the *Init* block FB 874 - VMC_InitSigma7W_EC with *Enable* = TRUE.

   ⇨ The output *Valid* returns TRUE. In the event of a fault, you can determine the error by evaluating the *ErrorID*.

   You have to call the *Init* block again if you load a new axis DB or you have changed parameters on the *Init* block.

   Do not continue until the Init block does not report any errors!

3. Ensure that the *Kernel* block FB 872 - VMC_KernelSigma7_EC is called cyclically for each axis. In this way, control signals are transmitted to the drive and status messages are reported.

4. Program your application with the FB 860 - VMC_AxisControl or with the PLCopen blocks for each axis.

Controlling the drive via HMI

There is the possibility to control your drive via HMI. For this, a predefined symbol library is available for Movicon to access the VMC_AxisControl function block. *Chap. 10 'Controlling the drive via HMI' page 432*

3.3.4 Usage in Siemens SIMATIC Manager

3.3.4.1 Precondition

Overview

- Please use for configuration the Siemens SIMATIC Manager V 5.5 SP2 and up.
- 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 CPU'* is to be installed in the hardware catalog by means of the GSDML.
- The configuration of the EtherCAT masters happens in the Siemens SIMATIC Manager by means of a virtual PROFINET IO device *'EtherCAT network'*. The *'EtherCAT network'* is to be installed in the hardware catalog by means of the GSDML.
- The *'EtherCAT network'* can be configured with the VIPA Tool *SPEED7 EtherCAT Manager*.
- For the configuration of the drive in the *SPEED7 EtherCAT Manager* the installation of the according ESI file is necessary.

Installing the IO device *'VIPA SLIO System'*

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'*.

Installing the IO device EtherCAT network

The installation of the PROFINET IO devices *'EtherCAT Network'* happens in the hardware catalog with the following approach:

1. Go to the service area of www.vipa.com
2. Load from the download area at *'Config files ➔ EtherCAT'* the GSDML file for your EtherCAT master.
3. Extract the files 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 *'EtherCAT Network'* can be found at *'PROFINET IO ➔ Additional field devices ➔ I/O ➔ VIPA EtherCAT System'*.

Installing the *SPEED7 EtherCAT Manager*

The configuration of the PROFINET IO device *'EtherCAT Network'* happens by means of the VIPA *SPEED7 EtherCAT Manager*. This may be found in the service area of www.vipa.com at *'Service/Support ➔ Downloads ➔ Software'*.

The installation happens with the following proceeding:

1. Close the Siemens SIMATIC Manager.
2. Go to the service area of www.vipa.com
3. Load the *SPEED7 EtherCAT Manager* and unzip it on your PC.
4. For installation start the file EtherCATManager_v... .exe.
5. Select the language for the installation.
6. Accept the licensing agreement.
7. Select the installation directory and start the installation.
8. After installation you have to reboot your PC.

   ⇨ The *SPEED7 EtherCAT Manager* is installed and can now be called via the context menu of the Siemens SIMATIC Manager.

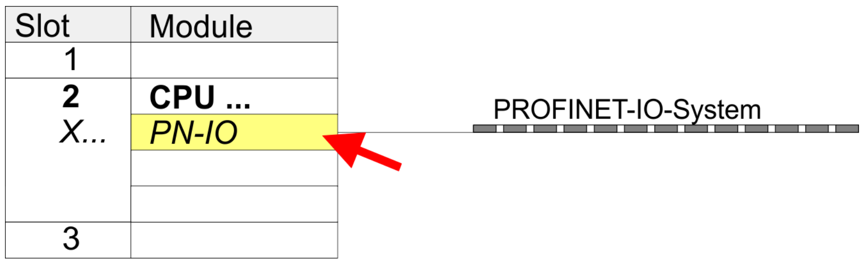
3.3.4.2 Hardware configuration

Configuring the CPU in the project

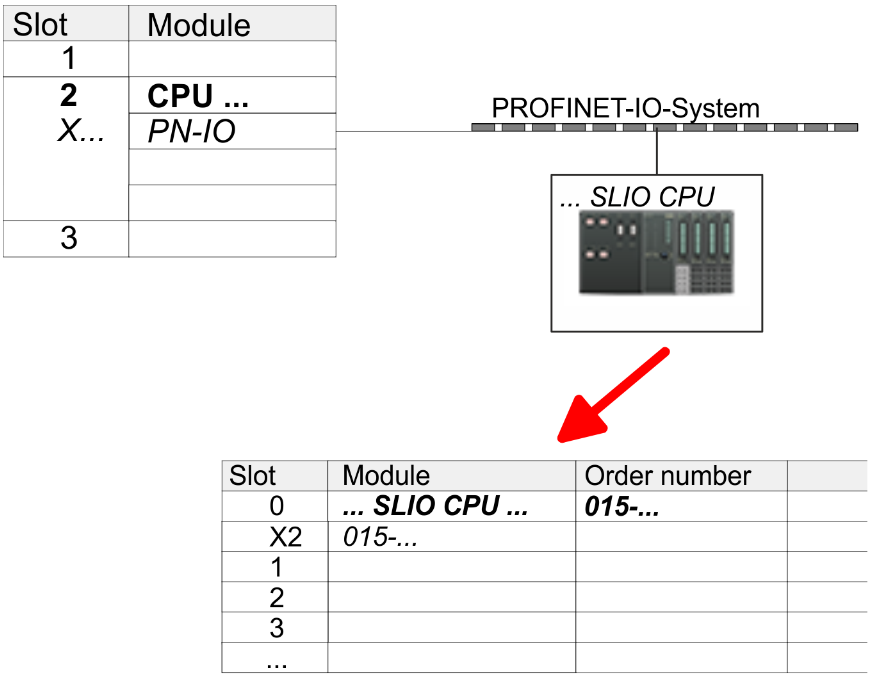
Slot	Module
1	
2	**CPU 315-2 PN/DP**
X1	*MPI/DP*
X2	*PN-IO*
X2...	*Port 1*
X2...	*Port 2*
3	

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 the CPU 315-2 PN/DP (315-2EH14 V3.2).
4. The integrated PROFIBUS DP master (jack X3) is to be configured and connected via the sub module *'X1 MPI/DP'*.
5. The integrated EtherCAT master is to be configured via the sub module *'X2 PN-IO'* as a virtual PROFINET network.
6. Click at the sub module *'PN-IO'* of the CPU.
7. Select *'Context menu ➔ Insert PROFINET IO System'*.



8. Create with [New] a new sub net and assign valid address data
9. Click at the sub module *'PN-IO'* of the CPU and open with *'Context menu ➔ Properties'* the properties dialog.
10. Enter at *'General'* a *'Device name'*. The device name must be unique at the Ethernet subnet.



11. Navigate in the hardware catalog to the directory *'PROFINET IO ➔ Additional field devices ➔ I/O ➔ VIPA SLIO System'* and connect the IO device *'015-CFFNR00 CPU'* to your PROFINET system.

⇨ In the Device 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

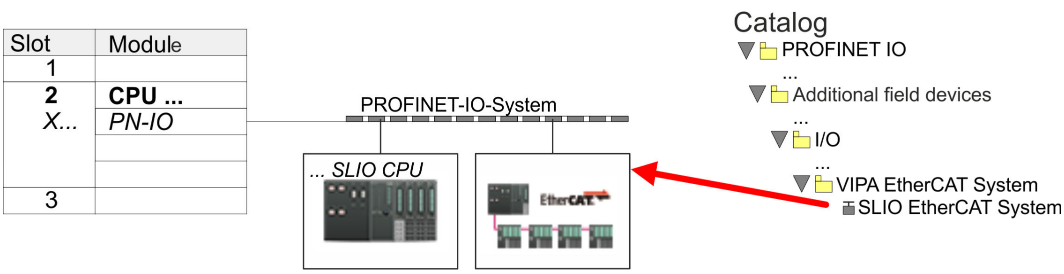
Slot	Module
1	
2	**CPU ...**
X...	*PN-IO*
3	
4	343-1EX30
5	
...	

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!

Insert *'EtherCAT network'*

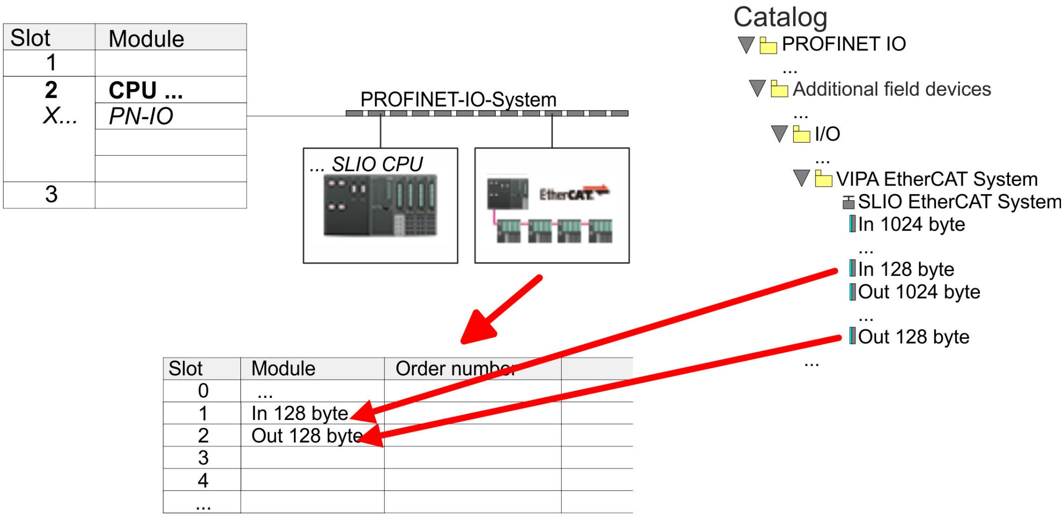


1. Navigate in the hardware catalog to the directory *'PROFINET IO ➔ Additional field devices ➔ I/O ➔ VIPA EtherCAT System'* and connect the IO device *'SLIO EtherCAT System'* to your PROFINET system.

2. Click at the inserted IO device *'EtherCAT Network'* and define the areas for in and output by drag and dropping the according *'Out'* or *'In'* area to a slot.

   Create the following areas:

   - In 128byte
   - Out 128byte

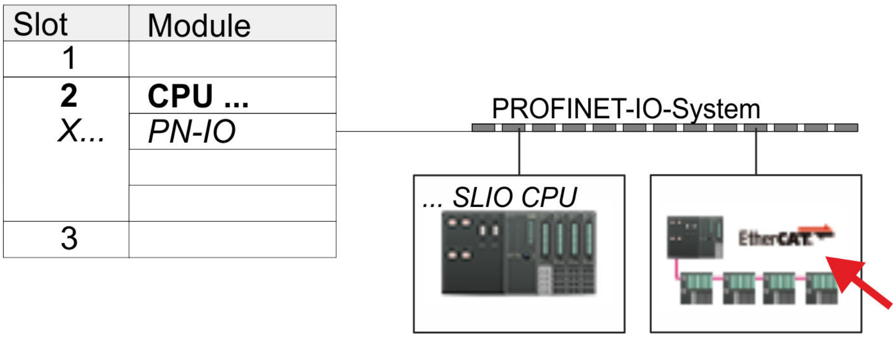


3. Select *'Station → Save and compile'*

Configure *Sigma-7W* EtherCAT double-axis drive

The double-axis drive is configured in the *SPEED7 EtherCAT Manager*.

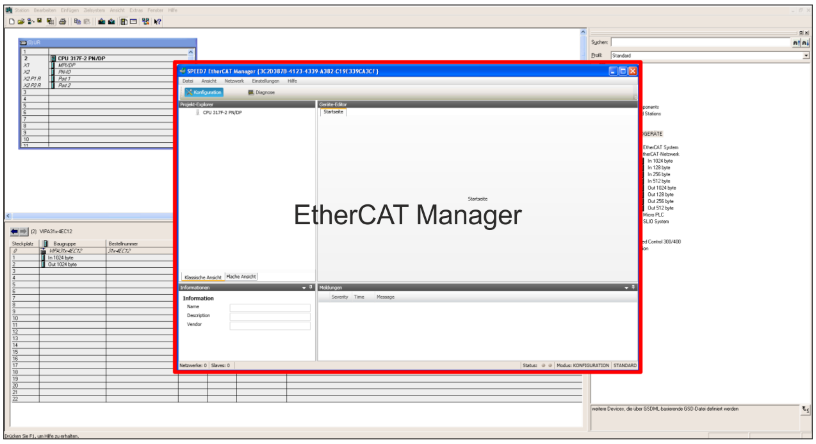
Before calling the SPEED7 EtherCAT Manager you have always to save your project with 'Station ➔ Save and compile'.



1. Click at an inserted IO device *'EtherCAT Network'* and select *'Context menu ➔ Start Device-Tool ➔ SPEED7 EtherCAT Manager'*.

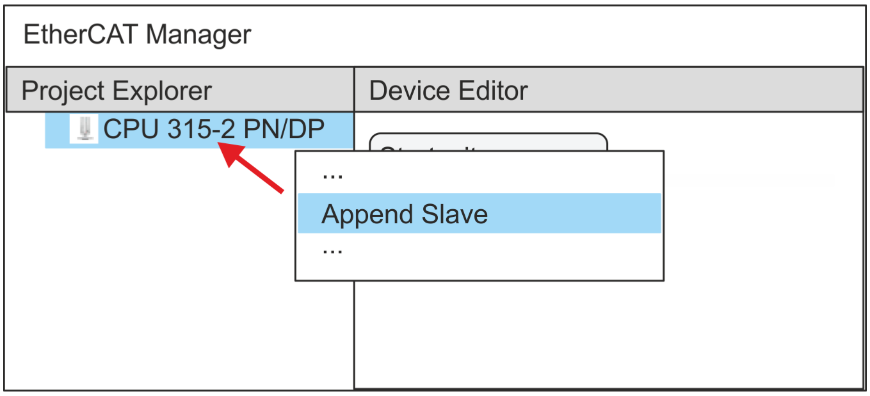
   ⇨ The *SPEED7 EtherCAT Manager* opens. Here you can configure the EtherCAT communication to your *Sigma-7W* EtherCAT double-axis drive.

   More information about the usage of the *SPEED7 EtherCAT Manager* may be found in the according manual or online help.



3. For the *Sigma-7W* EtherCAT drive to be configured in the *SPEED7 EtherCAT Manager*, the corresponding ESI file must be installed. The ESI file for the *Sigma-7W* EtherCAT double-axis drive can be found under www.yaskawa.eu.com at *'Service ➔ Drives & Motion Software'*. Download the according ESI file for your drive. Unzip this if necessary.
4. Open in the *SPEED7 EtherCAT Manager* via *'File ➔ ESI Manager'* the dialogue window *'ESI Manager'*.
5. In the *'ESI Manager'* click at [Add File] and select your ESI file. With [Open], the ESI file is installed in the *SPEED7 EtherCAT Manager*.
6. Close the *'ESI Manager'*.

   ⇨ Your *Sigma-7W* EtherCAT double-axis drive is now available for configuration.



7. In the EtherCAT Manager, click on your CPU and open via *'Context menu → Append Slave'* the dialog box for adding an EtherCAT slave.

   ⇨ The dialog window for selecting an EtherCAT slave is opened.

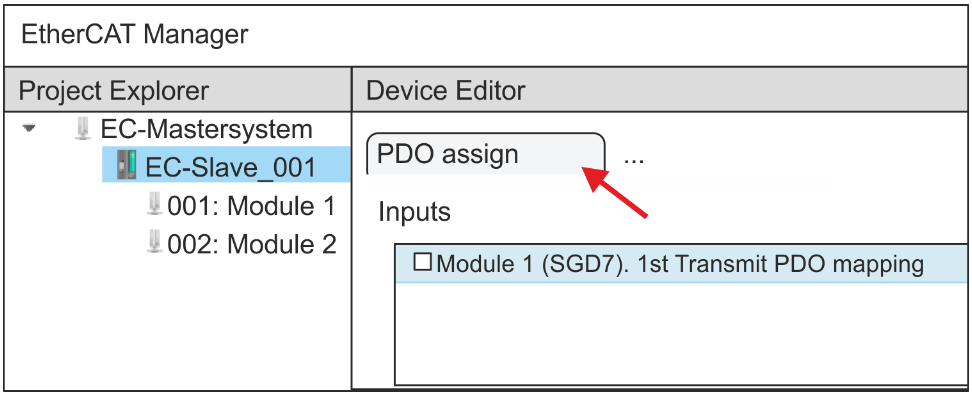
8. Select your *Sigma-7W* EtherCAT double-axis drive and confirm your selection with [OK].

   ⇨ The *Sigma-7W* EtherCAT double-axis drive is connected to the master and can now be configured.

9. *You can only edit PDOs in 'Expert mode'! Otherwise, the buttons are hidden. By activating the 'Expert mode' you can switch to advanced setting.*

   By activating *'View → Expert'* you can switch to the *Expert mode*.

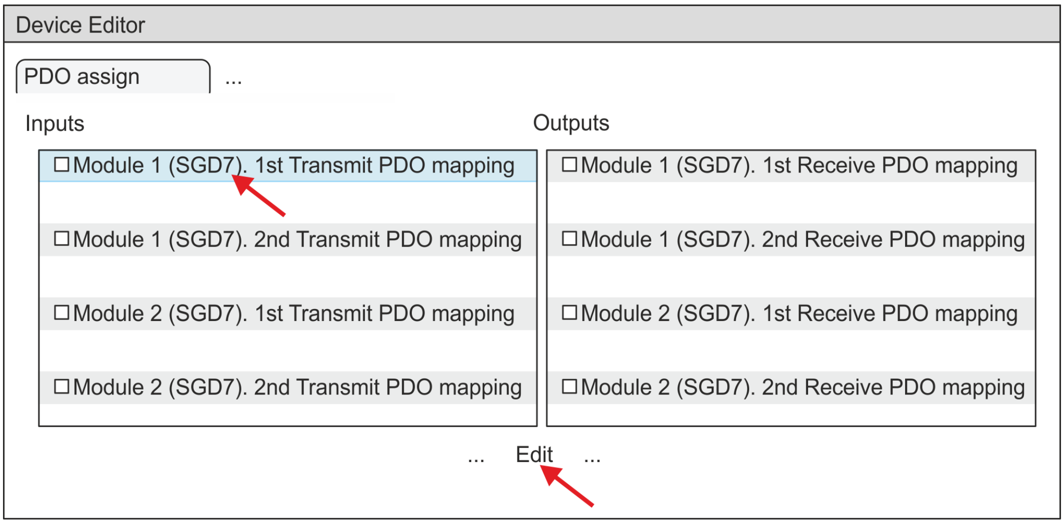
10. Click on the *Sigma-7W* EtherCAT Slave in the *SPEED7 EtherCAT Manager* and select the *'PDO assign'* tab in the *'Device editor'*.



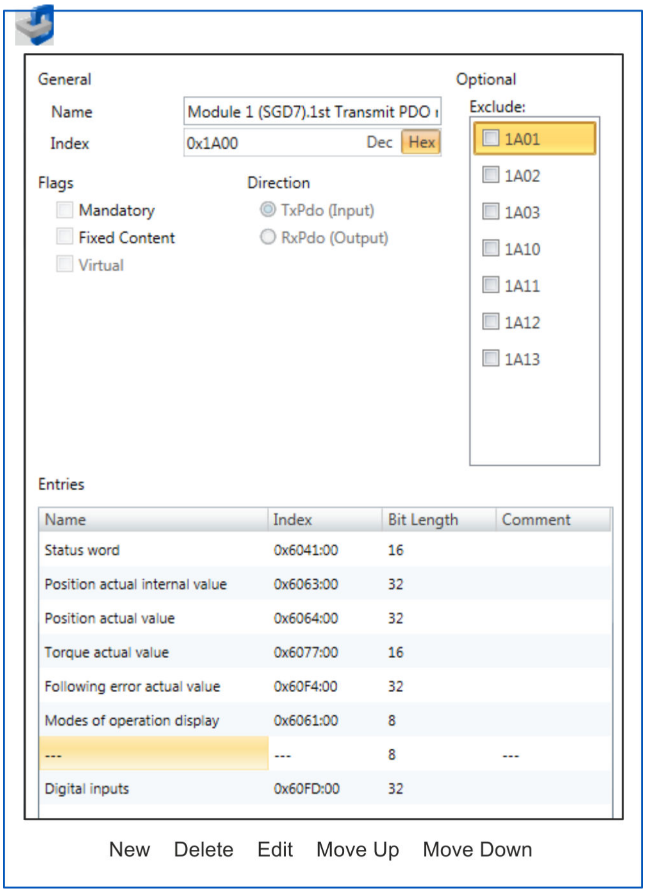
   ⇨ This dialogue shows a list of the PDOs.

11. By selecting the appropriate mapping, you can edit the PDOs with [Edit]. Select the mapping *'Module 1 (SGD7). 1st Transmit PDO mapping'* and click at [Edit].

> *Please note that some PDOs can not be edited because of the default settings. By de-activating already activated PDOs, you can release the processing of locked PDOs.*



⇨ The dialog *'Edit PDO'* is opened. Please check the PDO settings listed here and adjust them if necessary. Please also take into account the order of the *'Entries'* and add them accordingly.



The following functions are available for editing the *'Entries'*:

- New
  – Here you can create a new entry in a dialog by selecting the corresponding entry from the *'CoE object dictionary'* and making your settings. The entry is accepted with [OK] and is listed in the list of entries.
- Delete
  – This allows you to delete a selected entry.
- Edit
  – This allows you to edit the general data of an entry.
- Move Up/Down
  – This allows you to move the selected entry up or down in the list.

12. Perform the following settings for the Transmit PDOs:

Inputs: 1st Transmit PDO

Module 1 (SGD7). 1st Transmit PDO mapping	Module 2 (SGD7). 1st Transmit PDO mapping
Name: Module 1 (SGD7). 1st Transmit PDO mapping	Name: Module 2 (SGD7). 1st Transmit PDO mapping
Index: 0x1A00	Index: 0x1A10
Flags: Everything de-activated	
Direction TxPdo (Input): activated	
Exclude: 1A01: de-activated	1A11: de-activated
Please note these settings, otherwise the PDO mappings can not be activated at the same time!	

Entries	Module 1 (axis 1)	Module 2 (axis 2)	Bit length
Name	**Index**	**Index**	
Status word	0x6041:00	0x6841: 00	16bit
Position actual internal value	0x6063:00	0x6863:00	32bit
Position actual value	0x6064:00	0x6864:00	32bit
Torque actual value	0x6077:00	0x6877:00	16bit
Following error actual value	0x60F4:00	0x68F4:00	32bit
Modes of operation display	0x6061:00	0x6861:00	8bit
---	---	---	8bit
Digital inputs	0x60FD:00	0x68FD:00	32bit

Inputs: 2nd Transmit PDO

Module 1 (SGD7). 2nd Transmit PDO mapping	Module 2 (SGD7). 2nd Transmit PDO mapping
Name: Module 1 (SGD7). 2nd Transmit PDO mapping	Name: Module 2 (SGD7). 2nd Transmit PDO mapping
Index: 0x1A01	Index: 0x1A11
Flags: Everything de-activated	
Direction TxPdo (Input): activated	
Exclude: 1A00, 1A02, 1A03: de-activated	1A10, 1A12, 1A13: de-activated
Please note these settings, otherwise the PDO mappings can not be activated at the same time!	

Entries	Module 1 (axis 1)	Module 2 (axis 2)	Bit length
Name	**Index**	**Index**	
Touch probe status	0x60B9:00	0x68B9:00	16bit
Touch probe 1 position value	0x60BA:00	0x68BA:00	32bit
Touch probe 2 position value	0x60BC:00	0x68BC:00	32bit
Velocity actual value	0x606C:00	0x686C:00	32bit

13. Perform the following settings for the Receive PDOs:

Outputs: 1st Receive PDO

Module 1 (SGD7). 1st Receive PDO	Module 2 (SGD7). 1st Receive PDO
Name: Module 1 (SGD7). 1st Receive PDO mapping	Name: Module 2 (SGD7). 1st Receive PDO mapping
Index: 0x1600	Index: 0x1610
Flags: Everything de-activated	
Direction RxPdo (Output): activated	
Exclude: 1601, 1602, 1603: de-activated	1611, 1612, 1613: de-activated
Please note these settings, otherwise the PDO mappings can not be activated at the same time!	

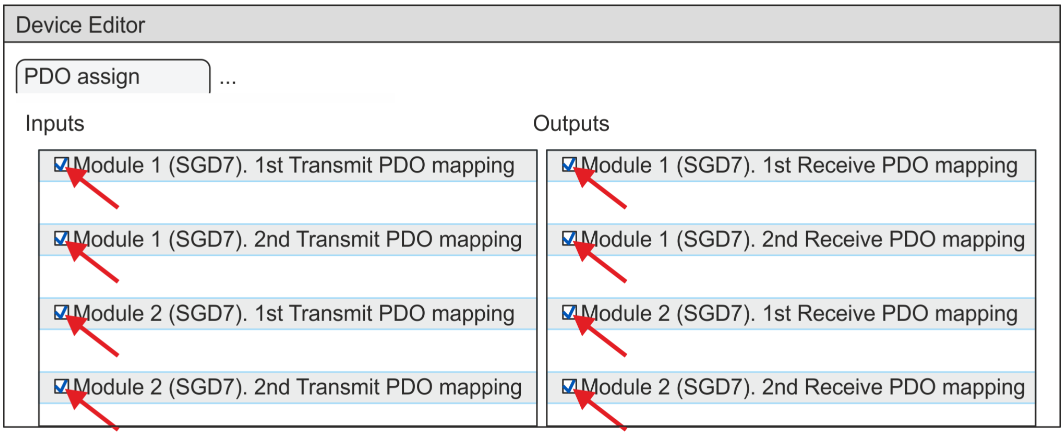
Entries	Module 1 (axis 1)	Module 2 (axis 2)	Bit length
Name	**Index**	**Index**	
Control word	0x6040:00	0x6840: 00	16bit
Target position	0x607A:00	0x687A: 00	32bit
Target velocity	0x60FF:00	0x68FF: 00	32bit
Modes of operation	0x6060:00	0x6860:00	8bit
---	---	---	8bit
Touch probe function	0x60B8:00	0x68B8: 00	16bit

Outputs: 2nd Receive PDO

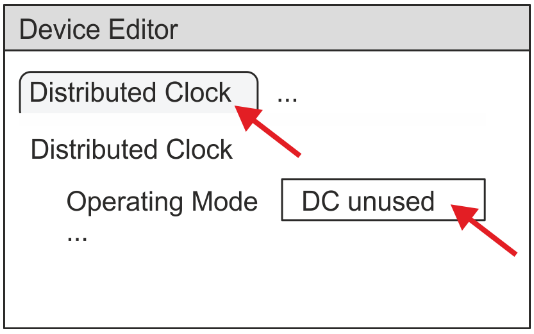
Module 1 (SGD7). 2nd Receive PDO	Module 2 (SGD7). 2nd Receive PDO
Name: Module 1 (SGD7). 2nd Receive PDO mapping	Name: Module 2 (SGD7). 2nd Receive PDO mapping
Index: 0x1601	Index: 0x1611
Flags: Everything de-activated	
Direction RxPdo (Output): activated	
Exclude: 1600, 1602, 1603: de-activated	1610, 1612, 1613: de-activated
Please note these settings, otherwise the PDO mappings can not be activated at the same time!	

Entries	Module 1 (axis 1)	Module 2 (axis 2)	Bit length
Name	**Index**	**Index**	
Profile velocity	0x6081:00	0x6881:00	32bit
Profile acceleration	0x6083:00	0x6883:00	32bit
Profile deceleration	0x6084:00	0x6884:00	32bit

14. For *'Module 1'* and *'Module 2'* in PDO assignment, activate the PDOs 1 and 2 for the inputs and outputs. All subsequent PDOs must remain de-activated. If this is not possible, please check the respective PDO parameter *'Exclude'*.

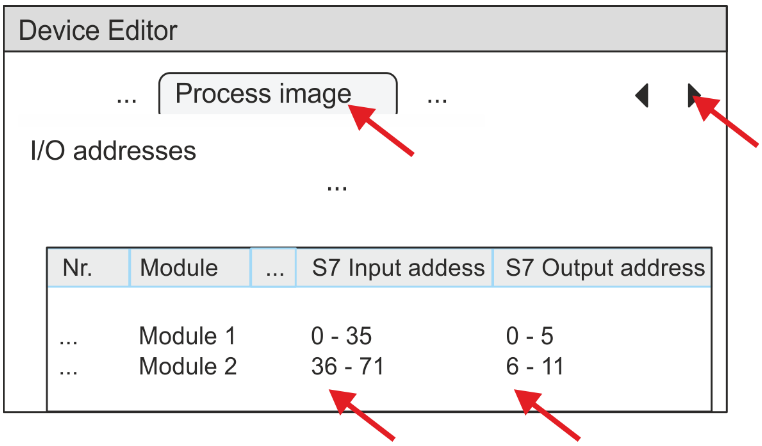


15. In the *'Device Editor'* of the *SPEED7 EtherCAT Manager*, select the *'Distributed clocks'* tab and set *'DC unused'* as *'Operating mode'*.

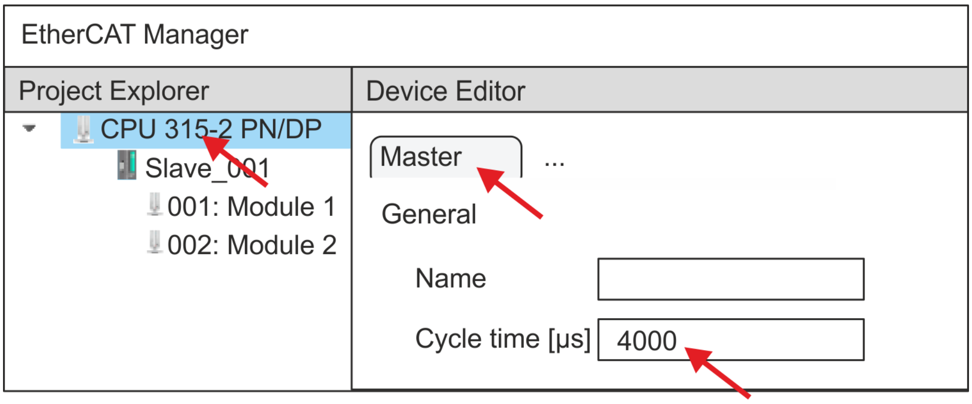


16. Select the *'Process image'* tab in the *'device editor'* using the arrow key and note the following PDO start addresses for the parameters of the block FB 874 - VMC_InitSigma7W_EC:

- Module 1: *'S7 Input address'* → *'M1_PdoInputs'* (here 0)
- Module 2: *'S7 Input address'* → *'M2_PdoInputs'* (here 36)
- Module 1: *'S7 Output address'* → *'M1_PdoOutputs'* (here 0)
- Module 2: *'S7 Output address'* → *'M2_PdoOutputs'* (here 36)



17. Click on your CPU in the *SPEED7 EtherCAT Manager* and select the *'Master'* tab in the *'Device editor'*.



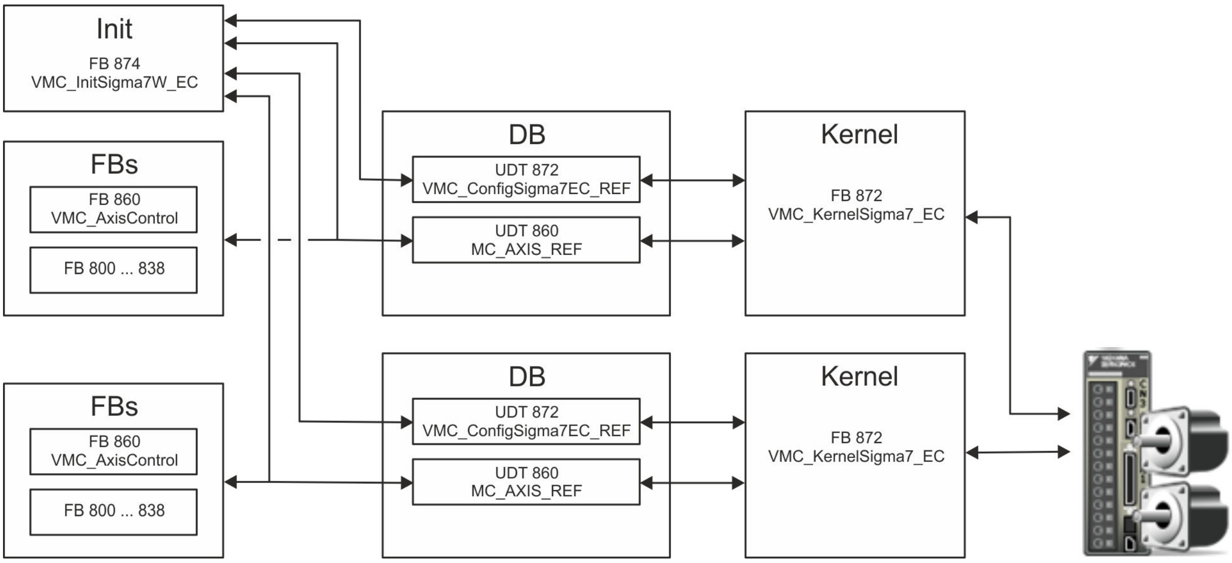
⇨ Set a cycle time of at least 4ms for Sigma-7W (400V) drives.

18. By closing the *SPEED7 EtherCAT Manager* the EtherCAT configuration is taken to the project. You can always edit your EtherCAT configuration in the *SPEED7 EtherCAT Manager*, since the configuration is stored in your project.

19. Save and compile your configuration.

3.3.4.3 User program

3.3.4.3.1 Program structure



- DB
  A data block (axis DB) for configuration and status data must be created for each axis of a drive. The data block consists of the following data structures:
  – UDT 872 - *VMC_Config*Sigma7EC_REF
    The data structure describes the structure of the configuration of the drive.
    Specific data structure for *Sigma-7* EtherCAT.
  – UDT 860 - *MC_AXIS_REF*
    The data structure describes the structure of the parameters and status information of drives.
    General data structure for all drives and bus systems.
- FB 874 - *VMC_Init*Sigma7W_EC
  – The *Init*t block is used to configure the double-axis drive.
  – Specific block for *Sigma-7W* EtherCAT.
  – The configuration data for the initialization must be stored in the *axis DB*.
- FB 872 - *VMC_Kernel*Sigma7_EC
  – The *Kernel* block communicates with the drive via the appropriate bus system, processes the user requests and returns status messages.
  – The FB 872 - *VMC_Kernel*Sigma7_EC must be called for each axis.
  – Specific block for *Sigma-7* EtherCAT.
  – The exchange of the data takes place by means of the *axis DB*.
- FB 860 - *VMC_AxisControl*
  – General block for all drives and bus systems.
  – The FB 860 - *VMC_AxisControl* must be called for each axis.
  – Supports simple motion commands and returns all relevant status messages.
  – The exchange of the data takes place by means of the *axis DB*.
  – For motion control and status query, via the instance data of the block you can link a visualization.
  – In addition to the FB 860 - *VMC_AxisControl*, *PLCopen* blocks can be used.
- FB 800 ... FB 838 - PLCopen
  – The PLCopen blocks are used to program motion sequences and status queries.
  – The PLCopen blocks must be called for each axis.

3.3.4.3.2 Programming

Include library

1. Go to the service area of www.vipa.com.
2. Download the *Simple Motion Control* library from the download area at *'VIPA Lib'*.
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-7W* EtherCAT:
  – UDT 872 - VMC_ConfigSigma7EC_REF
  – FB 872 - VMC_KernelSigma7_EC
  – FB 874 - VMC_InitSigma7W_EC
- Axis Control
  – UDT 860 - MC_AXIS_REF
  – Blocks for your movement sequences

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.

Create axis DB for *'Module 1'*

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

   Specify the following parameters:

   - Name and type
     – The DB no. as *'Name'* can freely be chosen, such as DB 10.
     – Set *'Shared DB'* as the *'Type'*.
   - Symbolic name
     – Specify "Axis01".

   Confirm your input with [OK].

   ⇨ The block is created.

2. Open DB 10 "Axis01" by double-click.

   - In "Axis01", create the variable "Config" of type UDT 872. These are specific axis configuration data.
   - In "Axis01", create the variable "Axis" of type UDT 860. During operation, all operating data of the axis are stored here.

   DB10

Address	Name	Type	...
		Struct	
...	Config	"VMC_ConfigSigma7EC_REF"	
...	Axis	"MC_AXIS_REF	
...		END_STRUCT	

Create axis DB for *'Module 2'*

1. Add another DB as your *axis DB* to your project and assign it the name "Axis02". The DB number can freely be selected such as DB11.

   ⇨ The block is created.

2. Open DB 11 "Axis02" by double-click.

   - In "Axis02", create the variable "Config" of type UDT 872. These are specific axis configuration data.
   - In "Axis02", create the variable "Axis" of type UDT 860. During operation, all operating data of the axis are stored here.

   DB 11

Address	Name	Type	...
		Struct	
...	Config	"VMC_ConfigSigma7EC_REF"	
...	Axis	"MC_AXIS_REF	
...		END_STRUCT	

OB 1

Configuration of the double-axis

Open OB 1 and program the following FB calls with associated DBs:

FB 874 - VMC_InitSigma7W_EC, DB 874 *Chap. 3.3.5.3 'FB 874 - VMC_Init-Sigma7W_EC - Sigma-7W EtherCAT Initialization' page 124*

At *M1/M2_PdoInputs* respectively *M1/M2_PdoOutputs*, enter the address from the *SPEED7 EtherCAT Manager* for the according axis. *116*

```
CALL  "VMC_InitSigma7W_EC" , "DI_InitSgm7WETC01"
Enable                    :=TRUE
LogicalAddress            :=0
M1_PdoInputs              :=0 (EtherCAT-Manager
                               Module1: S7 Input address)

M1_PdoOutputs             :=0 (EtherCAT-Manager
                               Module1: S7 Output address)
M1_EncoderType            :=2
M1_EncoderResolutionBits  :=20
M1_FactorPosition         :=1.048576e+006
M1_FactorVelocity         :=1.048576e+006
M1_FactorAcceleration     :=1.048576e+002
M1_OffsetPosition         :=0.000000e+000
M1_MaxVelocityApp         :=5.000000e+001
M1_MaxAccelerationApp     :=1.000000e+002
M1_MaxDecelerationApp     :=1.000000e+002
M1_MaxVelocityDrive       :=6.000000e+001
M1_MaxAccelerationDrive   :=1.500000e+002
M1_MaxDecelerationDrive   :=1.500000e+002
M1_MaxPosition            :=1.048500e+003
M1_MinPosition            :=-1.048514e+003
M1_EnableMaxPosition      :=TRUE
M1_EnableMinPosition      :=TRUE
M2_PdoInputs              :=36 (EtherCAT-Manager
                               Module2: S7 Input address)

M2_PdoOutputs             :=36 (EtherCAT-Manager
                               Module2: S7 Output address)
M2_EncoderType            :=2
M2_EncoderResolutionBits  :=20
M2_FactorPosition         :=1.048576e+006
M2_FactorVelocity         :=1.048576e+006
M2_FactorAcceleration     :=1.048576e+002
M2_OffsetPosition         :=0.000000e+000
M2_MaxVelocityApp         :=5.000000e+001
M2_MaxAccelerationApp     :=1.000000e+002
M2_MaxDecelerationApp     :=1.000000e+002
M2_MaxVelocityDrive       :=6.000000e+001
M2_MaxAccelerationDrive   :=1.500000e+002
M2_MaxDecelerationDrive   :=1.500000e+002
M2_MaxPosition            :=1.048500e+003
M2_MinPosition            :=-1.048514e+003
M2_EnableMaxPosition      :=TRUE
M2_EnableMinPosition      :=TRUE
M1_MinUserPosition        :=-1000.0
M1_MaxUserPosition        :=1000.0
M2_MinUserPosition        :=-1000.0
M2_MaxUserPosition        :=1000.0
Valid                     :="InitS7WEC1_Valid"
Error                     :="InitS7WEC1_Error"
```

```
ErrorID                  :="InitS7WEC1_ErrorID"
M1_Config                :="Axis01".Config
M1_Axis                  :="Axis01".Axis
M2_Config                :="Axis02".Config
M2_Axis                  :="Axis02".Axis
```

Connecting the kernel for the respective axis

The *Kernel* processes the user commands and passes them appropriately processed on to the drive via the respective bus system.

FB 872 - VMC_KernelSigma7_EC, DB 872 for axis 1

FB 872 - VMC_KernelSigma7_EC, DB 1872 for axis 2 ↪ *Chap. 3.2.5.2 ‘FB 872 - VMC_KernelSigma7_EC - Sigma-7 EtherCAT Kernel’ page 83*

⇨

```
CALL  "VMC_KernelSigma7_EC" , DB 872
 Init  :="KernelS7WEC1_Init"
 Config:="Axis01".Config
 Axis  :="Axis01".Axis

CALL  "VMC_KernelSigma7_EC" , DB 1872
 Init  :="KernelS7WEC2_Init"
 Config:="Axis02".Config
 Axis  :="Axis02".Axis
```

Connecting the block for motion sequences

For simplicity, the connection of the FB 860 - VMC_AxisControl is to be shown here. This universal block supports simple motion commands and returns status messages. The inputs and outputs can be individually connected. Please specify the reference to the corresponding axis data at *'Axis'* in the *axis DB*.

→ FB 860 - VMC_AxisControl, DB 860 *Chap. 9.2.2 'FB 860 - VMC_AxisControl - Control block axis control' page 363*

⇨
```
       CALL  "VMC_AxisControl" , "DI_AxisControl01"
        AxisEnable        :="AxCtrl1_AxisEnable"
        AxisReset         :="AxCtrl1_AxisReset"
        HomeExecute       :="AxCtrl1_HomeExecute"
        HomePosition      :="AxCtrl1_HomePosition"
        StopExecute       :="AxCtrl1_StopExecute"
        MvVelocityExecute:="AxCtrl1_MvVelExecute"
        MvRelativeExecute:="AxCtrl1_MvRelExecute"
        MvAbsoluteExecute:="AxCtrl1_MvAbsExecute"
        PositionDistance :="AxCtrl1_PositionDistance"
        Velocity          :="AxCtrl1_Velocity"
        Acceleration      :="AxCtrl1_Acceleration"
        Deceleration      :="AxCtrl1_Deceleration"
        JogPositive       :="AxCtrl1_JogPositive"
        JogNegative       :="AxCtrl1_JogNegative"
        JogVelocity       :="AxCtrl1_JogVelocity"
        JogAcceleration   :="AxCtrl1_JogAcceleration"
        JogDeceleration   :="AxCtrl1_JogDeceleration"
        AxisReady         :="AxCtrl1_AxisReady"
        AxisEnabled       :="AxCtrl1_AxisEnabled"
        AxisError         :="AxCtrl1_AxisError"
        AxisErrorID       :="AxCtrl1_AxisErrorID"
        DriveWarning      :="AxCtrl1_DriveWarning"
        DriveError        :="AxCtrl1_DriveError"
        DriveErrorID      :="AxCtrl1_DriveErrorID"
        IsHomed           :="AxCtrl1_IsHomed"
        ModeOfOperation   :="AxCtrl1_ModeOfOperation"
        PLCopenState      :="AxCtrl1_PLCopenState"
        ActualPosition    :="AxCtrl1_ActualPosition"
        ActualVelocity    :="AxCtrl1_ActualVelocity"
        CmdDone           :="AxCtrl1_CmdDone"
        CmdBusy           :="AxCtrl1_CmdBusy"
        CmdAborted        :="AxCtrl1_CmdAborted"
        CmdError          :="AxCtrl1_CmdError"
        CmdErrorID        :="AxCtrl1_CmdErrorID"
        DirectionPositive:="AxCtrl1_DirectionPos"
        DirectionNegative:="AxCtrl1_DirectionNeg"
        SWLimitMinActive :="AxCtrl1_SWLimitMinActive"
        SWLimitMaxActive :="AxCtrl1_SWLimitMaxActive"
        HWLimitMinActive :="AxCtrl1_HWLimitMinActive"
        HWLimitMaxActive :="AxCtrl1_HWLimitMaxActive"
        Axis              :="Axis...".Axis
```

At *Axis*, enter "Axis01" for axis 1 and "Axis02" for axis 2.

> *For complex motion tasks, you can use the PLCopen blocks. Here you must also specify the reference to the corresponding axis data at Axis in the axis DB.*

Your project now includes the following blocks:

- OB 1 - Main
- OB 57 - DP Manufacturer Alarm
- OB 82 - I/O_FLT1
- OB 86 - Rack_FLT

- FB 860 - VMC_AxisControl with instance DB
- FB 872 - VMC_KernelSigma7_EC with instance DB
- FB 874 - VMC_InitSigma7W_EC with instance DB
- UDT 860 - MC_Axis_REF
- UDT 872 - VMC_ConfigSigma7EC_REF

Sequence of operations

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

   The transfer can only be done by the Siemens SIMATIC Manager - not hardware configurator!

   > *Since slave and module parameters are transmitted by means of SDO respectively SDO Init command, the configuration remains active, until a power cycle is performed or new parameters for the same SDO objects are transferred.*
   >
   > ***With an overall reset the slave and module parameters are not reset!***

   ⇨ You can take your application into operation now.

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

2. Before the double-axis drive can be controlled, it must be initialized. To do this, call the *Init* block FB 874 - VMC_InitSigma7W_EC with *Enable* = TRUE.

   ⇨ The output *Valid* returns TRUE. In the event of a fault, you can determine the error by evaluating the *ErrorID*.

   You have to call the *Init* block again if you load a new axis DB or you have changed parameters on the *Init* block.

   > *Do not continue until the Init block does not report any errors!*

3. Ensure that the *Kernel* block FB 872 - VMC_KernelSigma7_EC is called cyclically for each axis. In this way, control signals are transmitted to the drive and status messages are reported.
4. Program your application with the FB 860 - VMC_AxisControl or with the PLCopen blocks for each axis.

Controlling the drive via HMI

There is the possibility to control your drive via HMI. For this, a predefined symbol library is available for Movicon to access the VMC_AxisControl function block. *Chap. 10 'Controlling the drive via HMI' page 432*

3.3.4.4 Copy project

Proceeding

In the example, the station *'Source'* is copied and saved as *'Target '*.

1. Open the hardware configuration of the *'Source'* CPU and start the *SPEED7 EtherCAT Manager*.
2. In the *SPEED7 EtherCAT Manager*, via *'File ➔ Save as'* save the configuration in your working directory.
3. Close the *SPEED7 EtherCAT Manager* and the hardware configurator.
4. Copy the station *'Source'* with Ctrl + C and paste it as *'Target'* into your project with Ctrl + V.
5. Select the *'Blocks'* directory of the *'Target'* CPU and delete the *'System data'*.
6. Open the hardware configuration of the *'Target'* CPU. Adapt the IP address data or re-network the CPU or the CP again.

> *Before calling the SPEED7 EtherCAT Manager you have always to save your project with 'Station ➔ Save and compile'.*

7. Safe your project with *'Station ➔ Safe and compile'*.
8. Open the *SPEED7 EtherCAT Manager*.
9. Use *'File ➔ Open'* to load the configuration from your working directory.
10. Close the *SPEED7 EtherCAT Manager*.
11. Save and compile your configuration.

3.3.5 Drive specific blocks

The PLCopen blocks for axis control can be found here: *Chap. 9 'Blocks for axis control' page 361*

3.3.5.1 UDT 872 - VMC_ConfigSigma7EC_REF - *Sigma-7* EtherCAT Data structure axis configuration

This is a user-defined data structure that contains information about the configuration data. The UDT is specially adapted to the use of a *Sigma-7* drive, which is connected via EtherCAT.

3.3.5.2 FB 872 - VMC_KernelSigma7_EC - *Sigma-7* EtherCAT Kernel

Description

This block converts the drive commands for a *Sigma-7* axis via EtherCAT and communicates with the drive. For each *Sigma-7* axis, an instance of this FB is to be cyclically called.

Please note that this module calls the SFB 238 internally.

In the SPEED7 Studio, this module is automatically inserted into your project.

In Siemens SIMATIC Manager, you have to copy the SFB 238 from the Motion Control Library into your project.

Parameter	Declaration	Data type	Description
Init	INPUT	BOOL	The block is internally reset with an edge 0-1. Existing motion commands are aborted and the block is initialized.
Config	IN_OUT	UDT872	Data structure for transferring axis-dependent configuration data to the *AxisKernel*.
Axis	IN_OUT	MC_AXIS_REF	Data structure for transferring axis-dependent information to the *AxisKernel* and PLCopen blocks.

3.3.5.3 FB 874 - VMC_InitSigma7W_EC - *Sigma-7W* EtherCAT Initialization

Description

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

Parameter	Declaration	Data type	Description
Enable	INPUT	BOOL	Release of initialization
LogicalAddress	INPUT	INT	Start address of the PDO input data
M1_PdoInputs	INPUT	INT	Start address of the input PDOs for axis 1
M1_PdoOutputs	INPUT	INT	Start address of the output PDOs for axis 1

Parameter	Declaration	Data type	Description
M1_EncoderType	INPUT	INT	Encoder type of axis 1 ■ 1: Absolute encoder ■ 2: Incremental encoder
M1_EncoderResolutionBits	INPUT	INT	Number of bits corresponding to one encoder revolution of axis 1. Default: 20
M1_FactorPosition	INPUT	REAL	Factor for converting the position of user units [u] into drive units [increments] and back of axis 1. It's valid: $p_{[increments]} = p_{[u]}$ x *FactorPosition* Please consider the factor which can be specified on the drive via the objects 0x2701: 1 and 0x2701: 2. This should be 1.
M1_FactorVelocity	INPUT	REAL	Factor for converting the speed of user units [u/s] into drive units [increments/s] and back of axis 1. It's valid: $v_{[increments/s]} = v_{[u/s]}$ x *FactorVelocity* Please also take into account the factor which you can specify on the drive via objects 0x2702: 1 and 0x2702: 2. This should be 1.
M1_FactorAcceleration	INPUT	REAL	Factor to convert the acceleration of user units [u/s^2] in drive units [10^{-4} x increments/s^2] and back of axis 1. It's valid: 10^{-4} x $a_{[increments/s^2]} = a_{[u/s^2]}$ x *FactorAcceleration* Please also take into account the factor which you can specify on the drive via objects 0x2703: 1 and 0x2703: 2. This should be 1.
M1_OffsetPosition	INPUT	REAL	Offset for the zero position of axis 1 [u].
M1_MaxVelocityApp	INPUT	REAL	Maximum application speed of axis 1 [u/s]. The command inputs are checked to the maximum value before execution.
M1_MaxAccelerationApp	INPUT	REAL	Maximum acceleration of application of axis 1 [u/s^2]. The command inputs are checked to the maximum value before execution.
M1_MaxDecelerationApp	INPUT	REAL	Maximum acceleration of application of axis 1 [u/s^2]. The command inputs are checked to the maximum value before execution.
M1_MaxPosition	INPUT	REAL	Maximum position for monitoring the software limits of axis 1 [u].
M1_MinPosition	INPUT	REAL	Minimum position for monitoring the software limits of axis 1 [u].
M1_EnableMaxPosition	INPUT	BOOL	Monitoring maximum position of axis 1 ■ TRUE: Activates the monitoring of the maximum position.
M1_EnableMinPosition	INPUT	BOOL	Monitoring minimum position of axis 1 ■ TRUE: Activation of the monitoring of the minimum position.
M2_PdoInputs	INPUT	INT	Start address of the input PDOs for axis 2

Parameter	Declaration	Data type	Description
M2_PdoOutputs	INPUT	INT	Start address of the output PDOs for axis 2
M2_EncoderType	INPUT	INT	Encoder type of axis 2 ■ 1: Absolute encoder ■ 2: Incremental encoder
M2_EncoderResolutionBits	INPUT	INT	Number of bits corresponding to one encoder revolution of axis 2. Default: 20
M2_FactorPosition	INPUT	REAL	Factor for converting the position of user units [u] into drive units [increments] and back of axis 2. It's valid: $p_{[increments]} = p_{[u]}$ x *FactorPosition* Please consider the factor which can be specified on the drive via the objects 0x2701: 1 and 0x2701: 2. This should be 1.
M2_FactorVelocity	INPUT	REAL	Factor for converting the speed of user units [u/s] into drive units [increments/s] and back of axis 2. It's valid: $v_{[increments/s]} = v_{[u/s]}$ x *FactorVelocity* Please also take into account the factor which you can specify on the drive via objects 0x2702: 1 and 0x2702: 2. This should be 1.
M2_FactorAcceleration	INPUT	REAL	Factor to convert the acceleration of user units [u/s^2] in drive units [10^{-4} x increments/s^2] and back of axis 2. It's valid: 10^{-4} x $a_{[increments/s^2]} = a_{[u/s^2]}$ x *FactorAcceleration* Please also take into account the factor which you can specify on the drive via objects 0x2703: 1 and 0x2703: 2. This should be 1.
M2_OffsetPosition	INPUT	REAL	Offset for the zero position of axis 2 [u].
M2_MaxVelocityApp	INPUT	REAL	Maximum application speed of axis 2 [u/s]. The command inputs are checked to the maximum value before execution.
M2_MaxAccelerationApp	INPUT	REAL	Maximum acceleration of application of axis 2 [u/s^2]. The command inputs are checked to the maximum value before execution.
M2_MaxDecelerationApp	INPUT	REAL	Maximum acceleration of application of axis 2 [u/s^2]. The command inputs are checked to the maximum value before execution.
M2_MaxPosition	INPUT	REAL	Maximum position for monitoring the software limits of axis 2 [u].
M2_MinPosition	INPUT	REAL	Minimum position for monitoring the software limits of axis 2 [u].
M2_EnableMaxPosition	INPUT	BOOL	Monitoring maximum position of axis 2 ■ TRUE: Activates the monitoring of the maximum position.
M2_EnableMinPosition	INPUT	BOOL	Monitoring minimum position of axis 2 ■ TRUE: Activation of the monitoring of the minimum position.

Parameter	Declaration	Data type	Description
M1_MinUserPosition	OUTPUT	REAL	Minimum user position for axis 1 based on the minimum encoder value of 0x80000000 and the *FactorPosition* [u].
M1_MaxUserPosition	OUTPUT	REAL	Maximum user position for axis 1 based on the maximum encoder value of 0x7FFFFFFF and the *FactorPosition* [u].
M2_MinUserPosition	OUTPUT	REAL	Minimum user position for axis 2 based on the minimum encoder value of 0x80000000 and the *FactorPosition* [u].
M2_MaxUserPosition	OUTPUT	REAL	Maximum user position for axis 2 based on the maximum encoder value of 0x7FFFFFFF and the *FactorPosition* [u].
Valid	OUTPUT	BOOL	Initialization ■ TRUE: Initialization is valid.
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. 12 'ErrorID - Additional error information' page 457*
M1_Config	IN_OUT	UDT872	Data structure for transferring axis-dependent configuration data to the *AxisKernel* for axis 1.
M1_Axis	IN_OUT	MC_AXIS_REF	Data structure for transferring axis-dependent information to the *AxisKernel* and PLCopen blocks for axis 1.
M2_Config	IN_OUT	UDT872	Data structure for transferring axis-dependent configuration data to the *AxisKernel* for axis 2.
M2_Axis	IN_OUT	MC_AXIS_REF	Data structure for transferring axis-dependent information to the *AxisKernel* and PLCopen blocks for axis 2.

4 Usage *Sigma-5/7* PROFINET

4.1 Usage *Sigma-5* PROFINET

4.1.1 Overview

Precondition

- SPEED7 Studio from V1.8
  or
- Siemens SIMATIC Manager from V 5.5, SP2 respectively TIA Portal V 14 & *Simple Motion Control Library*
- CPU with PROFINET IO controller, such as CPU 015-CEFPR01
- *Sigma-5* drive with PROFINET option card

Steps of configuration

1. Set parameters on the drive using the rotary switch of the *Sigma-5* option card.
2. Hardware configuration in the VIPA *SPEED7 Studio*, Siemens SIMATIC Manager or TIA Portal.
   - Configuring a CPU with PROFINET IO controller.
   - Configuring a *Sigma-5* PROFINET drive.
3. Programming in the VIPA *SPEED7 Studio*, Siemens SIMATIC Manager or TIA Portal.
   - 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.

4.1.2 Set the parameters on the drive

Parameter Sigma-5

Before initial commissioning, you have to set the PROFINET option card of the Sigma-5 drive to *'Telegram 100 (all OP modes)'*. For this there is a rotary switch *'S12'* on the front of the option card. Turn it to position *'E'*. Further settings are not required for PROFINET communication.

> *Please note that you have to enable the corresponding direction of your axis in accordance to your requirements. For this use the parameters Pn50A (P-OT) respectively Pn50B (N-OT) in Sigma Win+.*

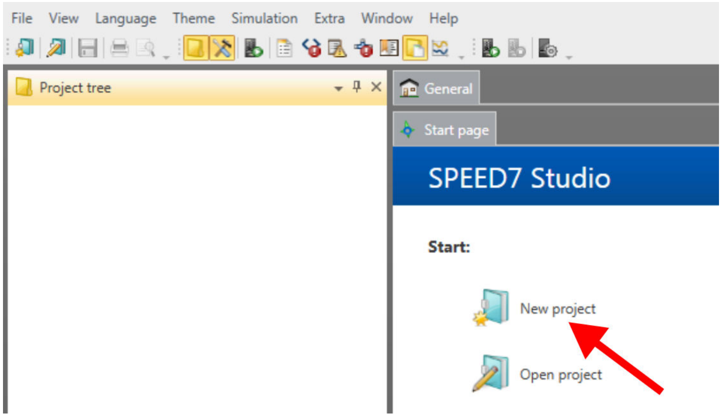
4.1.3 Usage in VIPA *SPEED7 Studio*

4.1.3.1 Hardware configuration System MICRO

Add CPU in the project

Please use the *SPEED7 Studio* V1.8 and up for the configuration.

1. Start the *SPEED7 Studio*.

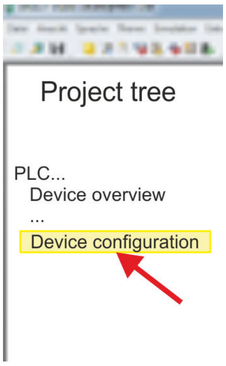


2. Create a new project at the start page with *'New project'* and assign a *'Project name'*.

   ⇨ A new project is created and the view *'Devices and networking'* is shown.

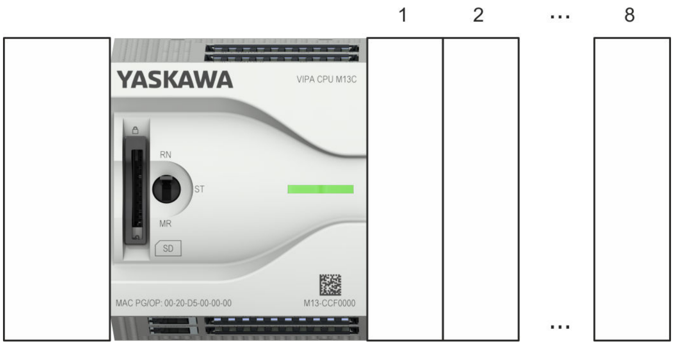
3. Click in the *Project tree* at *'Add new device ...'*.

   ⇨ A dialog for device selection opens.



4. Select from the *'Device templates'* the System MICRO CPU M13-CCF0000 V2.4.... and click at [OK].

   ⇨ The CPU is inserted in *'Devices and networking'* and the *'Device configuration'* is opened.



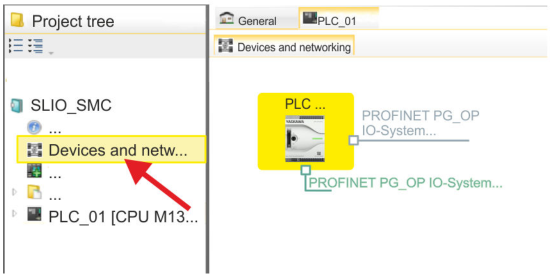
Device configuration

Slot	Module	...	...	...	...
0	CPU M13-CCF0000				
-X2	MPI interface				
-X3	PROFINET PG_OP IO-System				
...	...			...	

Configuration of Ethernet PG/OP channel

1. Click in the *Project tree* at *'Devices and networking'*.

   ⇨ You will get a graphical object view of your CPU. Here both interfaces of the PROFINET respectively Ethernet PG / OP channel switch are listed with identical name.



2. Click at one of the network *'PROFINET PG_OP_Ethernet IO-System ...'*.
3. Select *'Context menu ➔ Interface properties'*.

   ⇨ A dialog window opens. Here you can enter the IP address data for your Ethernet PG/OP channel. You get valid IP address parameters from your system administrator.
4. Confirm with [OK].

   ⇨ The IP address data are stored in your project listed in *'Devices and networking'* at *'Local components'*.

   After transferring your project your CPU can be accessed via Ethernet PG/OP channel with the set IP address data.

Installing the GSDML file

For the *Sigma-5* PROFINET drive can be configured in the *SPEED7 Studio*, the corresponding GSDML file must be installed. Usually, the *SPEED7 Studio* is delivered with current GSDML files and you can skip this part. If your GSDML file is not up-to date, you will find the latest GSDML file for the *Sigma-5* PROFINET drive under www.yaskawa.eu.com at *'Service ➔ Drives & Motion Software'*.

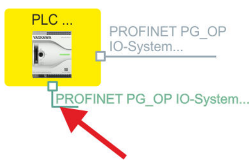
1. Download the according GSDML file for your drive. Unzip this if necessary.
2. Navigate to your *SPEED7 Studio*.
3. Open the corresponding dialog window by clicking on *'Extras ➔ Install device description (PROFINET - GSDML)'*.
4. Under *'Source path'*, specify the GSDML file and install it with [Install].

   ⇨ The devices of the GSDML file are now available.

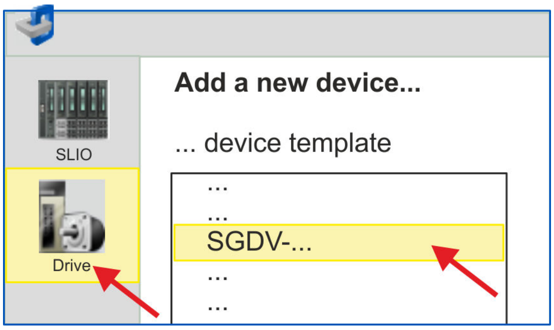
Add a *Sigma-5* drive

During configuration a *Sigma-5* PROFINET IO device must be configured for each axis.

1. Click in the Project tree at *'Devices and networking'*.
2. Click here at *'PROFINET PG_OP_Ethernet IO-System ...'* and select *'Context menu ➔ Add new device'*.



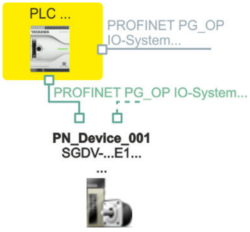
   ⇨ The device template for selecting PROFINET device opens.



3. Select your *Sigma-5* drive:

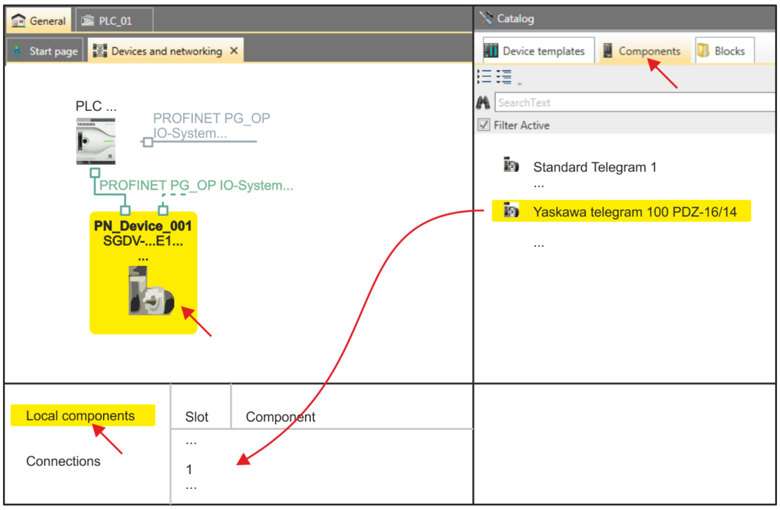
- SGDV-xxxxE1...

Confirm your input with [OK]. If your drive does not exist, you must install the corresponding GSDML file as described above.



⇨ The *Sigma-5* drive is connected to your PROFINET IO controller.

4. Click on the *Sigma-5* drive.



5. At *'Catalog'* select the *'Components'* tab.

⇨ The telegrams for the *Sigma-5* drive are listed.

6. Select *'Yaskawa telegram 100 PZD...'* drag&drop it to *'Slot 1'* of *'Local components'*.

⇨ Telegram 100 is inserted with the corresponding subgroups.

The connection between the axes in the hardware configuration and your user program is made by specifying the following module properties in the call parameters of FB 891 - VMC InitSigma_PN:

- *Module properties ‘Parameter Access Point’: Diagnostic address of slot 1 of the slot overview*
  - *FB 891 - VMC InitSigma_PN: ParaAccessPointAddress: Setting of the diagnostic address of slot 1 of the slot overview.*
- *Module properties ‘YASKAWA Telegram PZD...’: Respective start address of the input/output address range.*
  - *FB 891 - VMC InitSigma_PN: ‘InputsStartAddress’: Setting of the start address of the input address range.*
  - *FB 891 - VMC InitSigma_PN: ‘OutputsStartAddress’: Setting of the start address of the output address range.*
  - *FB 891 - VMC InitSigma_PN: ‘LogicalAddress’: Setting of the of the smaller value of the start addresses of the input/output address range.*

- User program ⇨ *139*
- FB 891 - VMC InitSigma_PN ⇨ *217*

Example hardware configuration

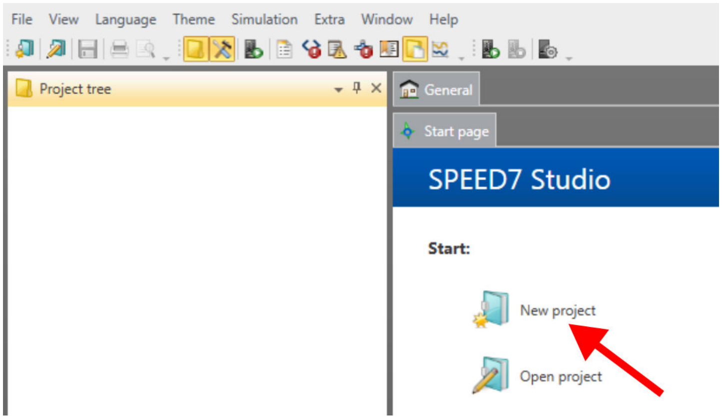
Slot	Component	...	I-Adr.	O-Adr.	Diagnostic address
0	SGDV-OCB03A		2045		2045
X1	PN-IO		2039		2039
X1 P1	Port 1		2038		2038
X1 P2	Port 2		2037		2037
1	DO with YASKAWA telegr.100, PZD-16/14		2036		2036
1.1	Parameter Access Point		2036		2036
1.2	YASKAWA telegram, PZD-16/14		0-27	0-31	2036

4.1.3.2 Hardware configuration System SLIO

Add CPU in the project

Please use the *SPEED7 Studio* V1.8 and up for the configuration.

1. Start the *SPEED7 Studio*.

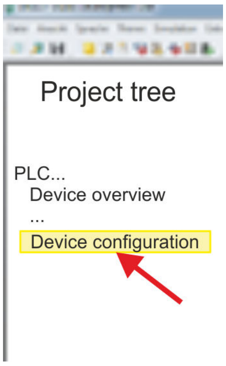


2. Create a new project at the start page with *'New project'* and assign a *'Project name'*.

   ⇨ A new project is created and the view *'Devices and networking'* is shown.

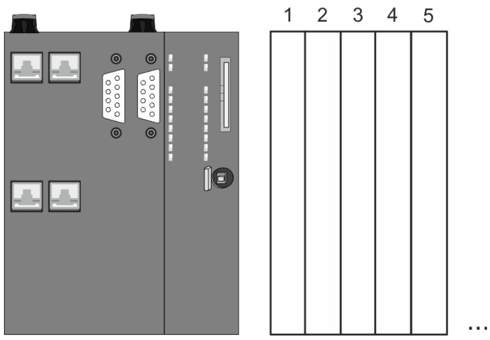
3. Click in the *Project tree* at *'Add new device ...'*.

   ⇨ A dialog for device selection opens.



4. Select from the *'Device templates'* your PROFINET CPU e.g..CPU 015-CEFPR01 and click at [OK].

   ⇨ The CPU is inserted in *'Devices and networking'* and the *'Device configuration'* is opened.

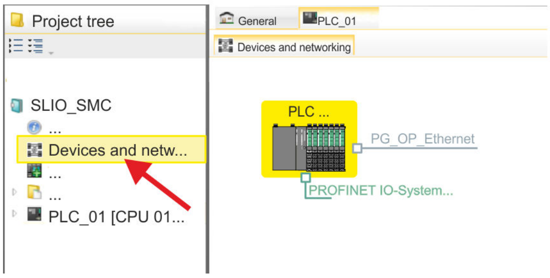


Device configuration

Slot	Module	...	...	...	...
0	CPU 015-CEFPR01				
-X1	PG_OP_Ethernet				
-X3	MPI interface				
-X4	PROFINET-IO-System				
...	...			...	

Configuration of Ethernet PG/OP channel

1. Click in the *Project tree* at *'Devices and networking'*.
   ⇨ You will get a graphical object view of your CPU.



2. Click at the network *'PG_OP_Ethernet'*.
3. Select *'Context menu ➔ Interface properties'*.
   ⇨ A dialog window opens. Here you can enter the IP address data for your Ethernet PG/OP channel. You get valid IP address parameters from your system administrator.
4. Confirm with [OK].
   ⇨ The IP address data are stored in your project listed in *'Devices and networking'* at *'Local components'*.

   After transferring your project your CPU can be accessed via Ethernet PG/OP channel with the set IP address data.

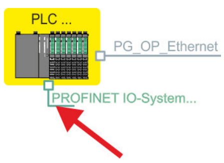
Installing the GSDML file

For the *Sigma-5* PROFINET drive can be configured in the *SPEED7 Studio*, the corresponding GSDML file must be installed. Usually, the *SPEED7 Studio* is delivered with current GSDML files and you can skip this part. If your GSDML file is not up-to date, you will find the latest GSDML file for the *Sigma-5* PROFINET drive under *www.yaskawa.eu.com* at *'Service ➔ Drives & Motion Software'*.

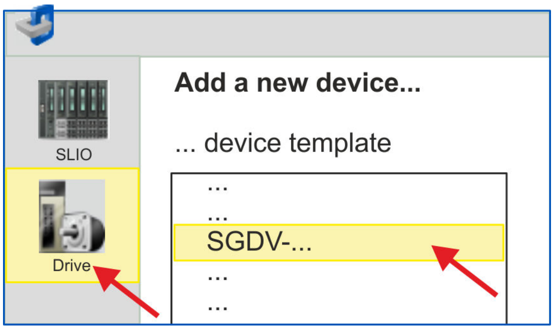
1. Download the according GSDML file for your drive. Unzip this if necessary.
2. Navigate to your *SPEED7 Studio*.
3. Open the corresponding dialog window by clicking on *'Extras ➔ Install device description (PROFINET - GSDML)'*.
4. Under *'Source path'*, specify the GSDML file and install it with [Install].
   ⇨ The devices of the GSDML file are now available.

Add a *Sigma-5* drive

1. Click in the Project tree at *'Devices and networking'*.
2. Click here at *'PROFINET IO-System ...'* and select *'Context menu ➔ Add new device'*.

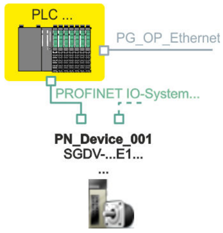


   ⇨ The device template for selecting PROFINET device opens.



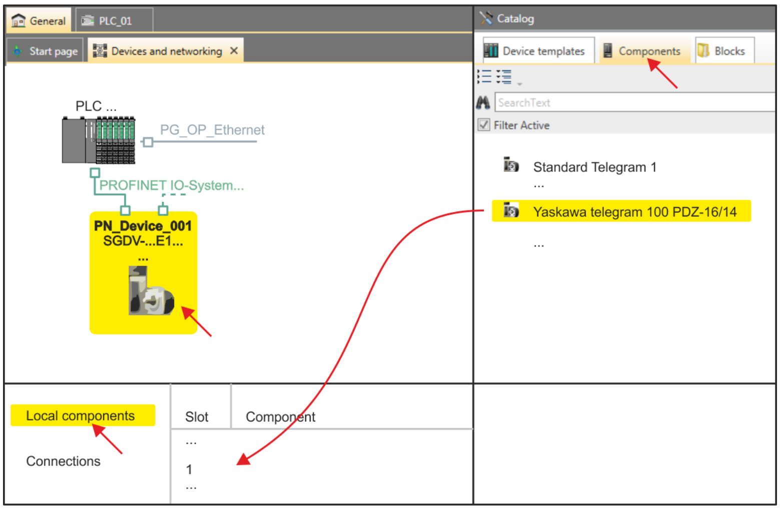
3. Select your *Sigma-5* drive:
   - SGDV-xxxxE1...

   Confirm your input with [OK]. If your drive does not exist, you must install the corresponding GSDML file as described above.



   ⇨ The *Sigma-5* drive is connected to your PROFINET IO controller.

4. Click on the *Sigma-5* drive



5. At *'Catalog'* select the *'Components'* tab.

   ⇨ The telegrams for the *Sigma-5* drive are listed.

6. Select *'Yaskawa telegram 100 PZD...'* drag&drop it to *'Slot 1'* of *'Local components'*.

   ⇨ Telegram 100 is inserted with the corresponding subgroups.

The connection between the axes in the hardware configuration and your user program is made by specifying the following module properties in the call parameters of FB 891 - VMC InitSigma_PN:

- *Module properties 'Parameter Access Point': Diagnostic address of slot 1 of the slot overview*
  - *FB 891 - VMC InitSigma_PN: ParaAccessPointAddress: Setting of the diagnostic address of slot 1 of the slot overview.*
- *Module properties 'YASKAWA Telegram PZD...': Respective start address of the input/output address range.*
  - *FB 891 - VMC InitSigma_PN: 'InputsStartAddress': Setting of the start address of the input address range.*
  - *FB 891 - VMC InitSigma_PN: 'OutputsStartAddress': Setting of the start address of the output address range.*
  - *FB 891 - VMC InitSigma_PN: 'LogicalAddress': Setting of the of the smaller value of the start addresses of the input/output address range.*

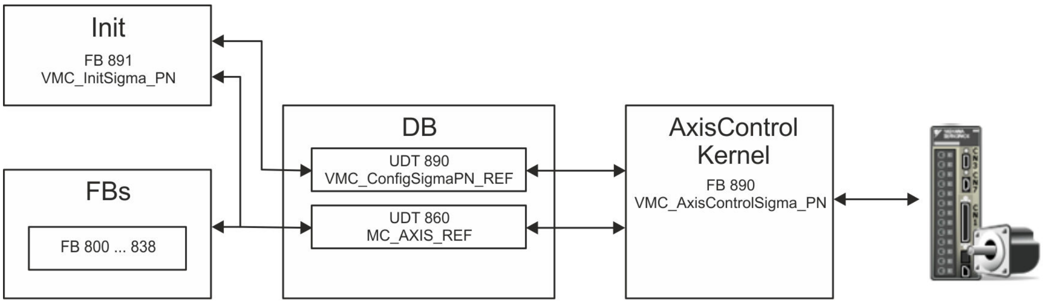
- User program ➔ *139*
- FB 891 - VMC InitSigma_PN ➔ *217*

Example hardware configuration

Slot	Component	...	I-Adr.	O-Adr.	Diagnostic address
0	SGDV-OCB03A		2045		2045
X1	PN-IO		2039		2039
X1 P1	Port 1		2038		2038
X1 P2	Port 2		2037		2037
1	DO with YASKAWA telegr.100, PZD-16/14		2036		2036
1.1	Parameter Access Point		2036		2036
1.2	YASKAWA telegram, PZD-16/14		0-27	0-31	2036

4.1.3.3 User program

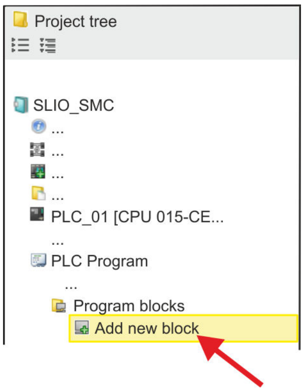
4.1.3.3.1 Program structure



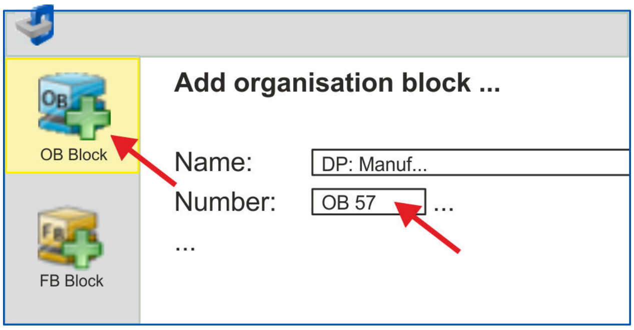
- DB
  A data block (axis DB) for configuration and status data must be created for each axis of a drive. The data block consists of the following data structures:
  – UDT 890 - *VMC_Config*SigmaPN_REF
    The data structure describes the structure of the configuration of the drive.
    Specific data structure for *Sigma-5/7* PROFINET.
  – UDT 860 - *MC_AXIS_REF*
    The data structure describes the structure of the parameters and status information of drives.
    General data structure for all drives and bus systems.
- FB 891 - *VMC_Init*Sigma_PN
  – The *Init*t block is used to configure an axis.
  – Specific block for *Sigma-5/7* PROFINET.
  – The configuration data for the initialization must be stored in the *axis DB*.
- FB 890 - *VMC_AxisControl*Sigma_PN
  – Specific block for *Sigma-5/7* PROFINET.
  – This block is a combination of *Kernel* and *AxisControl* and communicates with the drive via PROFINET, processes the user requests and returns status messages.
  – This block supports simple motion commands and returns all relevant status messages.
  – The exchange of the data takes place by means of the *axis DB*.
  – For motion control and status query, via the instance data of the block you can link a visualization.
  – In addition to the FB 890 - *VMC_AxisControl*Sigma_PN, *PLCopen* blocks can be used.
- FB 800 ... FB 838 - *PLCopen*
  – The PLCopen blocks are used to program motion sequences and status queries.
  – General blocks for all drives and bus systems.

4.1.3.3.2 Programming

Create interrupt OBs



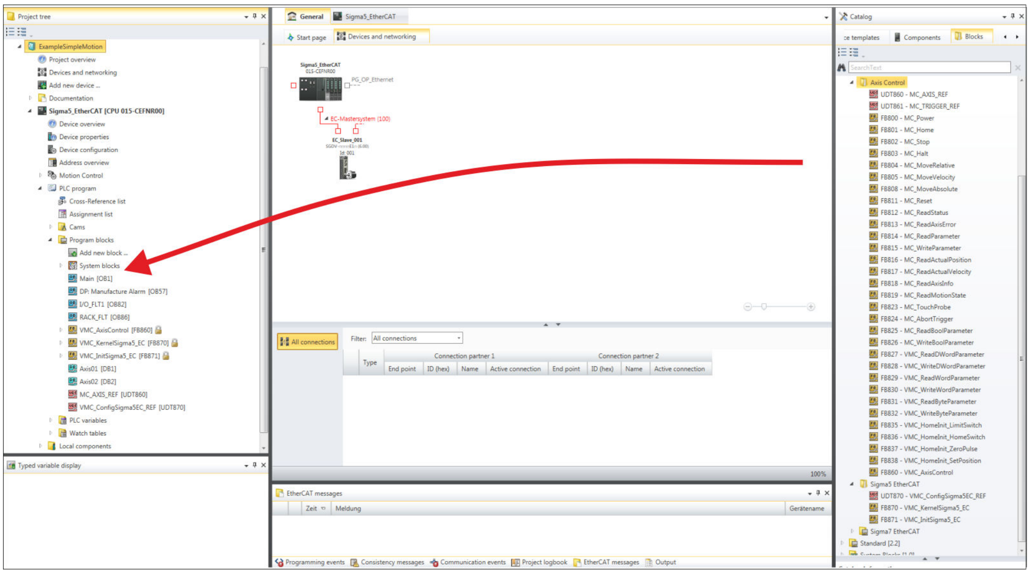
1. Click in the *Project tree* within the CPU at *'PLC program'*, *'Program blocks'* at *'Add New block'*.



⇨ The dialog *'Add block'* is opened.

2. Select the block type *'OB block'* and add one after the other OB 57, OB 82 and OB 86 to your project.

Copy blocks into project



In the *'Catalog'*, open the *'Simple Motion Control'* library at *'Blocks'* and drag and drop the following blocks into *'Program blocks'* of the *Project tree*:

- *Sigma* PROFINET:
  - UDT 890 - VMC_ConfigSigmaPN_REF *Chap. 4.3.1 'UDT 890 - VMC_ConfigSigmaPN_REF - Sigma-5/7 PROFINET Data structure axis configuration' page 213*
  - FB 890 - VMC_AxisControlSigma_PN *Chap. 4.3.2 'FB 890 - VMC_Axis-ControlSigma_PN - control block axis control for Sigma-5/7 PROFINET' page 213*
  - FB 891 - VMC_InitSigma_PN *Chap. 4.3.3 'FB 891 - VMC_InitSigma_PN - Sigma-5/7 PROFINET initialization' page 217*
- Axis control
  - UDT 860 - MC_AXIS_REF *Chap. 9.2.1 'UDT 860 - MC_AXIS_REF - Data structure axis data' page 363*
  - FB 860 - VMC_AxisControl *Chap. 9.2.2 'FB 860 - VMC_AxisControl - Control block axis control' page 363*

Create axis DB

1. Add a new DB as your *axis DB* to your project. Click in the *Project tree* within the CPU at *'PLC program'*, *'Program blocks'* at *'Add New block'*, select the block type *'DB block'* and assign the name "Axis01" to it. The DB number can freely be selected such as DB 10.

   ⇨ The block is created and opened.

2. - In "Axis01", create the variable "Config" of type UDT 890. These are specific axis configuration data.
   - In "Axis01", create the variable "Axis" of type UDT 860. During operation, all operating data of the axis are stored here.

Axis01 [DB10]

Data block structure

		Adr...	Name	Data type	...
		...	Config	UDT	[890]
		...	Axis	UDT	[860]

OB 1 - configuration of the axes

Open OB 1 and program the following FB calls with associated DBs:

FB 891 - VMC_InitSigma_PN, DB 891

The connection between the axes in the hardware configuration and your user program is made by specifying the following module properties in the call parameters of FB 891 - VMC InitSigma_PN:

- *Module properties 'Parameter Access Point': Diagnostic address of slot 1 of the slot overview*
  - *FB 891 - VMC InitSigma_PN: ParaAccessPointAddress:*
    Setting of the diagnostic address of slot 1 of the slot overview.
- *Module properties 'YASKAWA Telegram PZD...':*
  Respective start address of the input/output address range.
  - *FB 891 - VMC InitSigma_PN: 'InputsStartAddress':*
    Setting of the start address of the input address range.
  - *FB 891 - VMC InitSigma_PN: 'OutputsStartAddress':*
    Setting of the start address of the output address range.
  - *FB 891 - VMC InitSigma_PN: 'LogicalAddress':*
    Setting of the of the smaller value of the start addresses of the input/output address range.

- Hardware configuration ➪ *129*
- FB 891 - VMC InitSigma_PN ➪ *217*

Example hardware configuration

Slot	Component	...	I-Adr.	O-Adr.	Diagnostic address
0	SGDV-OCB03A		2045		2045
X1	PN-IO		2039		2039
X1 P1	Port 1		2038		2038
X1 P2	Port 2		2037		2037
1	DO with YASKAWA telegr.100, PZD-16/14		2036		2036
1.1	Parameter Access Point		2036		2036
1.2	YASKAWA telegram, PZD-16/14		0-27	0-31	2036

Example call

```
CALL  "VMC_InitSigma_PN" , "VMC_InitSigma_PN_1"
Enable                     :="InitS5PN1_Enable"
LogicalAddress             :=0 //HW-Konfig: Smallest IO addr.
ParaAccessPointAdress      :=2036 //HW-Konfig: Diag addr.
InputsStartAddress         :=0 //HW-Konfig: Telegr.100 start I addr.
OutputsStartAddress        :=0 //HW-Konfig: Telegr. 100 start O addr.
EncoderType                :=1
EncoderResolutionBits      :=20
FactorPosition             :=1.048576e+006
FactorVelocity             :=1.048576e+006
FactorAcceleration         :=1.048576e+006
OffsetPosition             :=0.000000e+000
MaxVelocityApp             :=5.000000e+001
MaxAccelerationApp         :=1.000000e+002
MaxDecelerationApp         :=1.000000e+002
MaxVelocityDrive           :=6.000000e+001
MaxPosition                :=1.048500e+003
MinPosition                :=-1.048514e+003
EnableMaxPosition          :=TRUE
EnableMinPosition          :=TRUE
MinUserPosition            :="InitS5PN1_MinUserPos"
MaxUserPosition            :="InitS5PN1_MaxUserPos"
Valid                      :="InitS5PN1_Valid"
Error                      :="InitS5PN1_Error"
ErrorID                    :="InitS5PN1_ErrorID"
Config                     :="Axis01".Config
Axis                       :="Axis01".Axis
```

Connecting the AxisControl

FB 890 - VMC_AxisControlSigma_PN, DB 890 ✎ *Chap. 4.3.2 'FB 890 - VMC_AxisControlSigma_PN - control block axis control for Sigma-5/7 PROFINET' page 213*

This block processes the user commands and passes them appropriately processed on to the drive via PROFINET.

```
CALL  "VMC_AxisControlSigma_PN" , "DI_AxisControlSigmaPN01"
 AxisEnable        :="AxCtrl1_AxisEnable"
 AxisReset         :="AxCtrl1_AxisReset"
 HomeExecute       :="AxCtrl1_HomeExecute"
 HomePosition      :="AxCtrl1_HomePosition"
 StopExecute       :="AxCtrl1_StopExecute"
 MvVelocityExecute:="AxCtrl1_MvVelExecute"
 MvRelativeExecute:="AxCtrl1_MvRelExecute"
 MvAbsoluteExecute:="AxCtrl1_MvAbsExecute"
 PositionDistance  :="AxCtrl1_PositionDistance"
 Direction         :="AxCtrl1_Direction"
```

```
Velocity           :="AxCtrl1_Velocity"
Acceleration       :="AxCtrl1_Acceleration"
Deceleration       :="AxCtrl1_Deceleration"
JogPositive        :="AxCtrl1_JogPositive"
JogNegative        :="AxCtrl1_JogNegative"
JogVelocity        :="AxCtrl1_JogVelocity"
JogAcceleration    :="AxCtrl1_JogAcceleration"
JogDeceleration    :="AxCtrl1_JogDeceleration"
AxisReady          :="AxCtrl1_AxisReady"
AxisEnabled        :="AxCtrl1_AxisEnabled"
AxisError          :="AxCtrl1_AxisError"
AxisErrorID        :="AxCtrl1_AxisErrorID"
DriveWarning       :="AxCtrl1_DriveWarning"
DriveError         :="AxCtrl1_DriveError"
DriveErrorID       :="AxCtrl1_DriveErrorID"
IsHomed            :="AxCtrl1_IsHomed"
ModeOfOperation    :="AxCtrl1_ModeOfOperation"
PLCopenState       :="AxCtrl1_PLCopenState"
ActualPosition     :="AxCtrl1_ActualPosition"
ActualVelocity     :="AxCtrl1_ActualVelocity"
CmdDone            :="AxCtrl1_CmdDone"
CmdBusy            :="AxCtrl1_CmdBusy"
CmdAborted         :="AxCtrl1_CmdAborted"
CmdError           :="AxCtrl1_CmdError"
CmdErrorID         :="AxCtrl1_CmdErrorID"
DirectionPositive:="AxCtrl1_DirectionPos"
DirectionNegative:="AxCtrl1_DirectionNeg"
SWLimitMinActive :="AxCtrl1_SWLimitMinActive"
SWLimitMaxActive :="AxCtrl1_SWLimitMaxActive"
HWLimitMinActive :="AxCtrl1_HWLimitMinActive"
HWLimitMaxActive :="AxCtrl1_HWLimitMaxActive"
Axis               :="Axis01".Axis
```

For complex motion tasks, you can use the PLCopen blocks. Please specify the reference to the corresponding axis data at Axis in the axis DB.

Your project now includes the following blocks:

- OB 1 - Main
- OB 57 - DP Manufacturer Alarm
- OB 82 - I/O_FLT1
- OB 86 - Rack_FLT
- FB 890 - VMC_AxisControlSigma_PN with instance DB
- FB 891 - VMC_InitSigma_PN with instance DB
- UDT 860 - MC_Axis_REF
- UDT 890 - VMC_ConfigSigmaPN_REF

Sequence of operations

1. Select *'Project → Compile all'* and transfer the project into your 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. To do this, call the *Init* block FB 891 - VMC_InitSigma_PN with *Enable* = TRUE.
   ⇨ The output *Valid* returns TRUE. In the event of a fault, you can determine the error by evaluating the *ErrorID*.

   You have to call the *Init* block again if you load a new axis DB or you have changed parameters on the *Init* block.

   > *Do not continue until the Init block does not report any errors!*

3. Program your application with the FB 890 - VMC_AxisControlSigma_PN or with the PLCopen blocks.

4.1.4 Usage in Siemens SIMATIC Manager

4.1.4.1 Hardware configuration System MICRO respectively SLIO

Precondition

Overview

- Please use for configuration the Siemens SIMATIC Manager V5.5 SP2 and up.
- The configuration of the VIPA System MICRO respectively SLIO CPU happens in the Siemens SIMATIC Manager by means of a virtual PROFINET IO device.
  The PROFINET IO device is to be installed in the hardware catalog by means of a GSDML.
- For the PROFINET drive can be configured in the Siemens SIMATIC Manager, the corresponding GSDML file must be installed.

Install GSDML file for System MICRO respectively SLIO

The installation of the PROFINET IO device 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 System MICRO or SLIO 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'*.

   From YASKAWA there are the following PROFINET IO devices:
   - System MICRO: *'VIPA Micro PLC'*
   - System SLIO: *'VIPA System SLIO'*

Install GSDML file for *Sigma-5* PROFINET drive

The GSDML file for the *Sigma-5* PROFINET drive can be found at www.yaskawa.eu.com under *'Service ➔ Drives & Motion Software'*.

Please use the following GSDML:

- GSDML-V2.3-Yaskawa-SGDV-OCB03A-20140228.xml

The installation happens with the following proceeding:

1. Download the according GSDML file for your drive.
2. Extract the file into your working directory.
3. Start the Siemens hardware configurator.
4. Close all the projects.
5. Select *'Options ➔ Install new GSD file'*.
6. Navigate to your working directory and install the according GSDML file.
   ⇨ After the installation the PROFINET IO device for the *Sigma-5* drive can be found at *'PROFINET IO ➔ Additional field devices ➔ Drives ➔ YASKAWA Drives'*.

Add CPU in the project

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

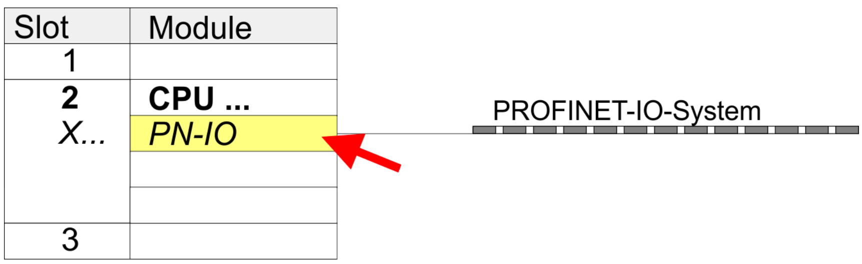
1. Start the Siemens hardware configurator with a new project.
2. Insert a profile rail from the hardware catalog.
3. Depending on the VIPA CPU used, place the following CPU from Siemens at *'Slot'* number 2:

VIPA CPU	to be configured as SIMATIC S7-300> ...
M13-CCF0000 from V2.4.12	CPU 314C-2 PN/DP (6ES7 314-6EH04-0AB0 V3.3)
013-CCF0R00 from V2.4.12	CPU 314C-2 PN/DP (6ES7 314-6EH04-0AB0 V3.3)
014-CEF0R01 from V2.4.12	CPU 315-2 PN/DP (6ES7 315-2EH14-0AB0 V3.2)
015-CEFNR00 from V2.4.16	CPU 315-2 PN/DP (6ES7 315-2EH14-0AB0 V3.2)
015-CEFPR01 from V2.4.12	CPU 315-2 PN/DP (6ES7 315-2EH14-0AB0 V3.2)
017-CEFPR00 from V2.4.12	CPU 317-2PN/DP (6ES7 317-2EK14-0AB0 V3.2)

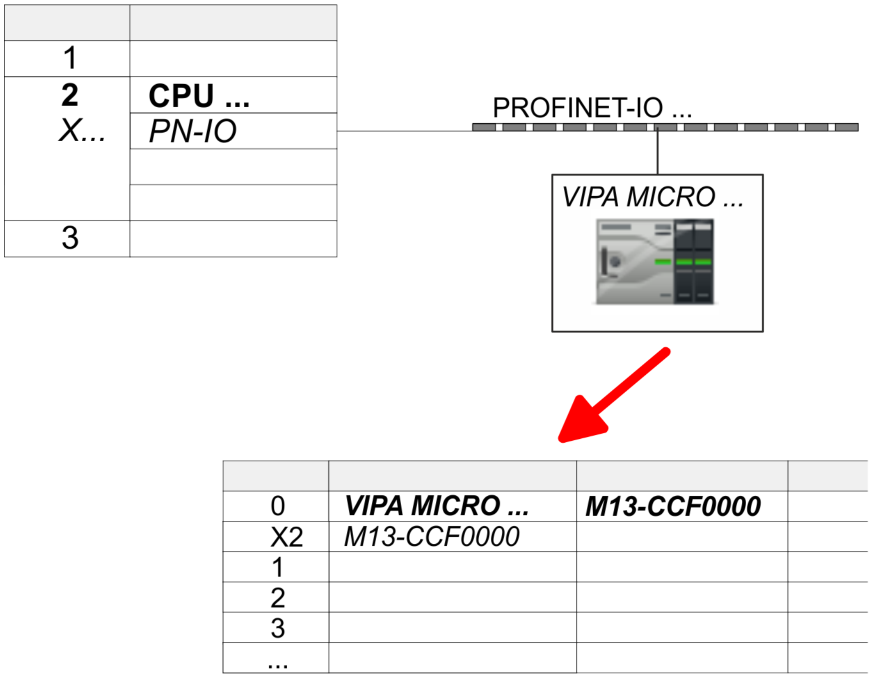
⇨ The CPU is inserted at the profile rail, such as the CPU 314C-2 PN/DP for System MICRO.

Connection CPU as PROFINET IO device

1. Click at the sub module *'PN-IO'* of the CPU.
2. Select *'Context menu ➔ Insert PROFINET IO System'*.



3. Create with [New] a new sub net and assign valid address data
4. Click at the sub module *'PN-IO'* of the CPU and open with *'Context menu ➔ Properties'* the properties dialog.
5. Enter at *'General'* a *'Device name'*. The device name must be unique at the Ethernet subnet.



6. Navigate in the hardware catalog to the directory *'PROFINET IO ➔ Additional field devices ➔ I/O'* and connect e.g. for the System MICRO the IO device *'M13-CCF0000'* to your PROFINET system.

   From YASKAWA there are the following PROFINET IO devices:

   - System MICRO: *'VIPA Micro PLC'*
   - System SLIO: *'VIPA System SLIO'*

   ⇨ In the Device overview of the PROFINET IO device *'VIPA MICRO PLC'* the CPU is already placed at slot 0.

Configuration of Ethernet PG/OP channel

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!

***Sigma-5* Insert and configure PROFINET drive**

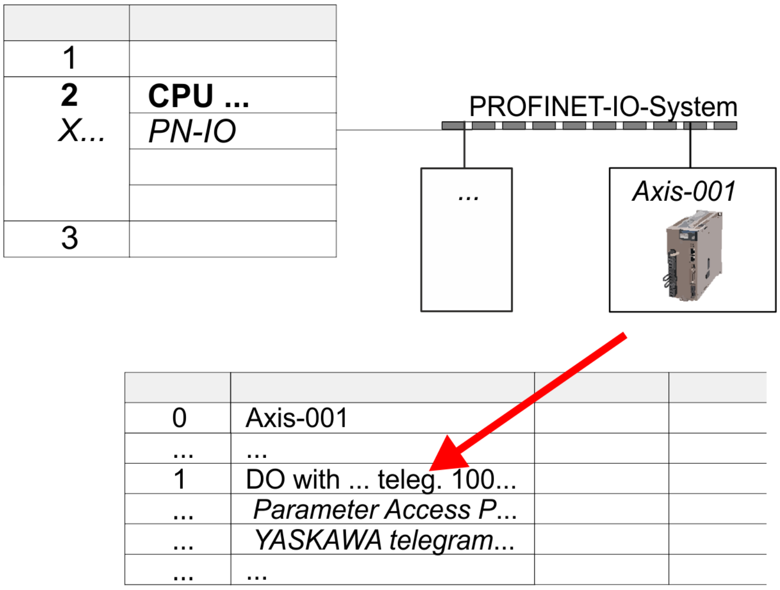
During configuration a *Sigma-5* PROFINET IO device must be configured for each axis.

1. Select your *Sigma-5* PROFINET drive *'SGDV-xxxxE1...'* from the hardware catalog and drag it onto the *'PROFINET-IO-System'*.

   ⇨ The *Sigma-5* PROFINET drive is connected to the IO controller and can now be configured.

2. Click at the *Sigma-5* IO device and open with *'Context menu ➔ Properties'* the properties dialog.
3. Assign a suitable *'Device name'* such as Axis-001.

4. Confirm your input with [OK].



5. In the hardware catalog, expand the *Sigma-5* PROFINET drive *'SGDV-xxxxE1...'* to show its components and drag&drop the component *'DO with YASKAWA telegr. 100...'* to slot 1 of the *Sigma-5* PROFINET drive.

   ⇨ Telegram 100 is inserted with the corresponding subgroups.

The connection between the axes in the hardware configuration and your user program is made by specifying the following module properties in the call parameters of FB 891 - VMC InitSigma_PN:

– *Module properties 'Parameter Access Point': Diagnostic address of slot 1 of the slot overview*
  – *FB 891 - VMC InitSigma_PN: ParaAccessPointAddress:*
    Setting of the diagnostic address of slot 1 of the slot overview.
– *Module properties 'YASKAWA Telegram PZD...':*
  Respective start address of the input/output address range.
  – *FB 891 - VMC InitSigma_PN: 'InputsStartAddress':*
    Setting of the start address of the input address range.
  – *FB 891 - VMC InitSigma_PN: 'OutputsStartAddress':*
    Setting of the start address of the output address range.
  – *FB 891 - VMC InitSigma_PN: 'LogicalAddress':*
    Setting of the of the smaller value of the start addresses of the input/output address range.

■ User program ➮ *152*
■ FB 891 - VMC InitSigma_PN ➮ *217*

Example hardware configuration

Slot	Module	...	I Addr.	Q Addr.	Diagnostic address
0	SGDV-OCB03A				2037
X1	PN-IO				2036
X1 P1	Port 1				2035
X1 P2	Port 2				2034
1	DO with YASKAWA telegr.100, PZD-16/14				2033
1.1	Parameter Access Point				2033
1.2	YASKAWA telegram, PZD-16/14		284-311	288-319	

4.1.4.2 Hardware configuration System 300S

Precondition

- Please use for configuration the Siemens SIMATIC Manager V5.5 SP2 and up.
- For the PROFINET drive can be configured in the Siemens SIMATIC Manager, the corresponding GSDML file must be installed.
- The blocks can be used with the following CPUs:
  - System 300S CPU 315-4PN43
  - System 300S CPU 315-4PN23
  - System 300S CPU 317-4PN23
- The configuration of the System 300S PROFINET CPU takes place in the Siemens SIMATIC Manager as a corresponding Siemens CPU.
  - The CPUs 315-4PNxx are to be configured as Siemens CPU 315-2 PN/DP (6ES7 315-2EH14-0AB0 V3.2).
  - The CPU 317-4PN23 is to be configured as Siemens CPU 317-2 PN/DP (6ES7 317-2EK14-0AB0 V3.2).

Install GSDML file for *Sigma-5* PROFINET drive

The GSDML file for the *Sigma-5* PROFINET drive can be found at www.yaskawa.eu.com under *'Service ➔ Drives & Motion Software'*.

Please use the following GSDML:

- GSDML-V2.3-Yaskawa-SGDV-OCB03A-20140228.xml

The installation happens with the following proceeding:

1. Download the according GSDML file for your drive.
2. Extract the file into your working directory.
3. Start the Siemens hardware configurator.
4. Close all the projects.
5. Select *'Options ➔ Install new GSD file'*.
6. Navigate to your working directory and install the according GSDML file.

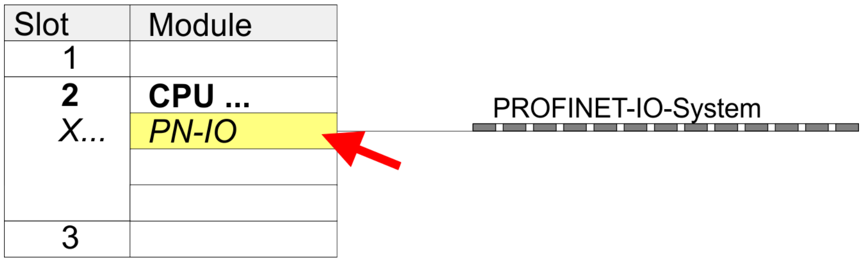
   ⇨ After the installation the PROFINET IO device for the *Sigma-5* drive can be found at *'PROFINET IO ➔ Additional field devices ➔ Drives ➔ YASKAWA Drives'*.

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. Select *'Context menu ➔ Insert PROFINET IO System'*.



6. Create with [New] a new sub net.
7. Click at the sub module *'PN-IO'* of the CPU and open with *'Context menu ➔ Properties'* the properties dialog.
8. Enter at *'General'* a *'Device name'*. The device name must be unique at the Ethernet subnet.

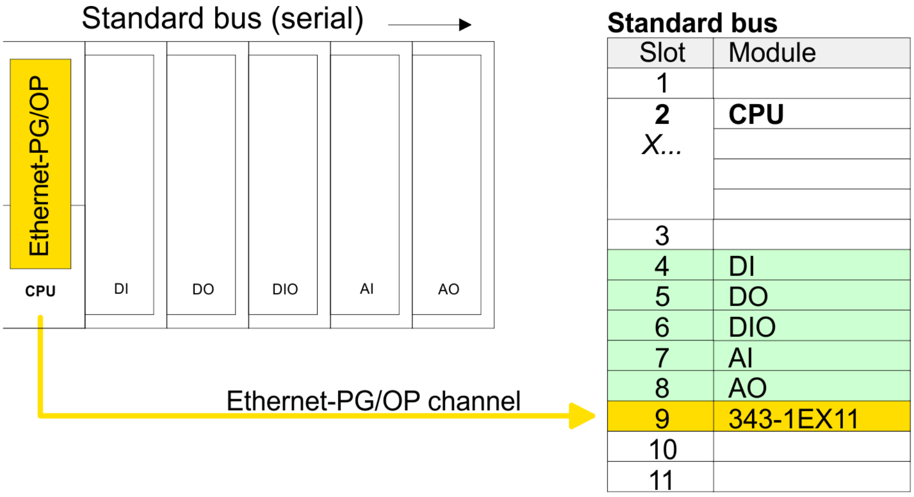
Configuration of Ethernet PG/OP channel

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.



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

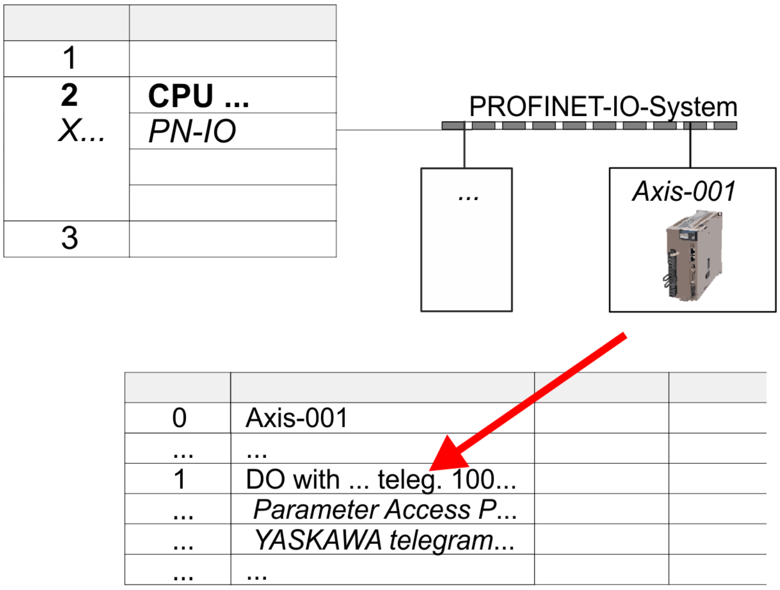
Sigma-5 Insert and configure PROFINET drive

During configuration a *Sigma-5* PROFINET IO device must be configured for each axis.

1. Select your *Sigma-5* PROFINET drive *'SGDV-xxxxE1...'* from the hardware catalog and drag it onto the *'PROFINET-IO-System'*.

   ⇨ The *Sigma-5* PROFINET drive is connected to the IO controller and can now be configured.

2. Click at the *Sigma-5* IO device and open with *'Context menu → Properties'* the properties dialog.
3. Assign a suitable *'Device name'* such as Axis-001.
4. Confirm your input with [OK].



5. In the hardware catalog, expand the *Sigma-5* PROFINET drive *'SGDV-xxxxE1...'* to show its components and drag&drop the component *'DO with YASKAWA telegr. 100...'* to slot 1 of the *Sigma-5* PROFINET drive.

⇨ Telegram 100 is inserted with the corresponding subgroups.

The connection between the axes in the hardware configuration and your user program is made by specifying the following module properties in the call parameters of FB 891 - VMC InitSigma_PN:

- *Module properties 'Parameter Access Point': Diagnostic address of slot 1 of the slot overview*
  - *FB 891 - VMC InitSigma_PN: ParaAccessPointAddress:*
    Setting of the diagnostic address of slot 1 of the slot overview.
- *Module properties 'YASKAWA Telegram PZD...':*
  Respective start address of the input/output address range.
  - *FB 891 - VMC InitSigma_PN: 'InputsStartAddress':*
    Setting of the start address of the input address range.
  - *FB 891 - VMC InitSigma_PN: 'OutputsStartAddress':*
    Setting of the start address of the output address range.
  - *FB 891 - VMC InitSigma_PN: 'LogicalAddress':*
    Setting of the of the smaller value of the start addresses of the input/output address range.

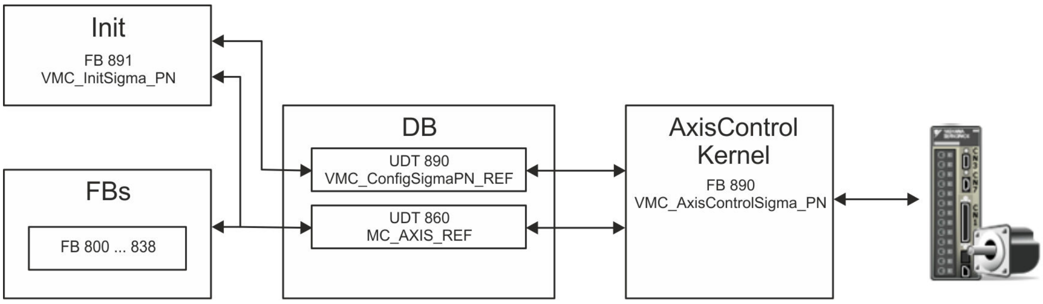
- User program ⇨ *152*
- FB 891 - VMC InitSigma_PN ⇨ *217*

Example hardware configuration

Slot	Module	...	I Addr.	Q Addr.	Diagnostic address
0	SGDV-OCB03A				2037
X1	PN-IO				2036
X1 P1	Port 1				2035
X1 P2	Port 2				2034
1	DO with YASKAWA telegr.100, PZD-16/14				2033
1.1	Parameter Access Point				2033
1.2	YASKAWA telegram, PZD-16/14		284-311	288-319	

4.1.4.3 User program

4.1.4.3.1 Program structure



- DB
  A data block (axis DB) for configuration and status data must be created for each axis of a drive. The data block consists of the following data structures:
  – UDT 890 - *VMC_Config*SigmaPN_REF
    The data structure describes the structure of the configuration of the drive.
    Specific data structure for *Sigma-5/7* PROFINET.
  – UDT 860 - *MC_AXIS_REF*
    The data structure describes the structure of the parameters and status information of drives.
    General data structure for all drives and bus systems.
- FB 891 - *VMC_Init*Sigma_PN
  – The *Init*t block is used to configure an axis.
  – Specific block for *Sigma-5/7* PROFINET.
  – The configuration data for the initialization must be stored in the *axis DB*.
- FB 890 - *VMC_AxisControl*Sigma_PN
  – Specific block for *Sigma-5/7* PROFINET.
  – This block is a combination of *Kernel* and *AxisControl* and communicates with the drive via PROFINET, processes the user requests and returns status messages.
  – This block supports simple motion commands and returns all relevant status messages.
  – The exchange of the data takes place by means of the *axis DB*.
  – For motion control and status query, via the instance data of the block you can link a visualization.
  – In addition to the FB 890 - *VMC_AxisControl*Sigma_PN, *PLCopen* blocks can be used.
- FB 800 ... FB 838 - *PLCopen*
  – The PLCopen blocks are used to program motion sequences and status queries.
  – General blocks for all drives and bus systems.

4.1.4.3.2 Programming

Include library

1. Go to the service area of www.vipa.com.
2. Download the *Simple Motion Control* library from the download area at *'VIPA Lib'*.
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].

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.

Copy blocks into project

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

- *Sigma* PROFINET:
  – UDT 890 - VMC_ConfigSigmaPN_REF *Chap. 4.3.1 'UDT 890 - VMC_ConfigSigmaPN_REF - Sigma-5/7 PROFINET Data structure axis configuration' page 213*
  – FB 890 - VMC_AxisControlSigma_PN *Chap. 4.3.2 'FB 890 - VMC_AxisControlSigma_PN - control block axis control for Sigma-5/7 PROFINET' page 213*
  – FB 891 - VMC_InitSigma_PN *Chap. 4.3.3 'FB 891 - VMC_InitSigma_PN - Sigma-5/7 PROFINET initialization' page 217*
- Axis control
  – UDT 860 - MC_AXIS_REF *Chap. 9.2.1 'UDT 860 - MC_AXIS_REF - Data structure axis data' page 363*
  – FB 860 - VMC_AxisControl *Chap. 9.2.2 'FB 860 - VMC_AxisControl - Control block axis control' page 363*

Create axis DB

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

   Specify the following parameters:
   - Name and type
     – The DB no. as *'Name'* can freely be chosen, such as DB10.
     – Set *'Shared DB'* as the *'Type'*.
   - Symbolic name
     – Specify "Axis01".

   Confirm your input with [OK].

   ⇨ The block is created.
2. Open DB10 "Axis01" by double-click.
   - In "Axis01", create the variable "Config" of type UDT 890. These are specific axis configuration data.
   - In "Axis01", create the variable "Axis" of type UDT 860. During operation, all operating data of the axis are stored here.

DB10

Address	Name	Type	...
		Struct	
...	Config	"VMC_ConfigSigmaPN_REF"	
...	Axis	"MC_AXIS_REF	
...		END_STRUCT	

OB 1 - configuration of the axes

Open OB 1 and program the following FB calls with associated DBs:

FB 891 - VMC_InitSigma_PN, DB 891

The connection between the axes in the hardware configuration and your user program is made by specifying the following module properties in the call parameters of FB 891 - VMC InitSigma_PN:

- *Module properties 'Parameter Access Point': Diagnostic address of slot 1 of the slot overview*
  - *FB 891 - VMC InitSigma_PN: ParaAccessPointAddress: Setting of the diagnostic address of slot 1 of the slot overview.*
- *Module properties 'YASKAWA Telegram PZD...': Respective start address of the input/output address range.*
  - *FB 891 - VMC InitSigma_PN: 'InputsStartAddress': Setting of the start address of the input address range.*
  - *FB 891 - VMC InitSigma_PN: 'OutputsStartAddress': Setting of the start address of the output address range.*
  - *FB 891 - VMC InitSigma_PN: 'LogicalAddress': Setting of the of the smaller value of the start addresses of the input/output address range.*

- Hardware configuration ↳ *142*
- FB 891 - VMC InitSigma_PN ↳ *217*

Example hardware configuration

Slot	Module	...	I Addr.	Q Addr.	Diagnostic address
0	SGDV-OCB03A				2037
X1	PN-IO				2036
X1 P1	Port 1				2035
X1 P2	Port 2				2034
1	DO with YASKAWA telegr.100, PZD-16/14				2033
1.1	Parameter Access Point				2033
1.2	YASKAWA telegram, PZD-16/14		284-311	288-319	

Example call

```
CALL  "VMC_InitSigma_PN" , "VMC_InitSigma_PN_1"
Enable                    :="InitS5PN1_Enable"
LogicalAddress            :=284 //HW-Konfig: Smallest IO addr.
ParaAccessPointAdress     :=2033 //HW-Konfig: Diag addr.
InputsStartAddress        :=284 //HW-Konfig: Telegr.100 start I addr.
OutputsStartAddress       :=288 //HW-Konfig: Telegr. 100 start O addr.
EncoderType               :=1
EncoderResolutionBits     :=20
FactorPosition            :=1.048576e+006
FactorVelocity            :=1.048576e+006
FactorAcceleration        :=1.048576e+006
OffsetPosition            :=0.000000e+000
MaxVelocityApp            :=5.000000e+001
MaxAccelerationApp        :=1.000000e+002
MaxDecelerationApp        :=1.000000e+002
MaxVelocityDrive          :=6.000000e+001
MaxPosition               :=1.048500e+003
MinPosition               :=-1.048514e+003
EnableMaxPosition         :=TRUE
EnableMinPosition         :=TRUE
```

```
MinUserPosition       :="InitS5PN1_MinUserPos"
MaxUserPosition       :="InitS5PN1_MaxUserPos"
Valid                 :="InitS5PN1_Valid"
Error                 :="InitS5PN1_Error"
ErrorID               :="InitS5PN1_ErrorID"
Config                :="Axis01".Config
Axis                  :="Axis01".Axis
```

Connecting the AxisControl

FB 890 - VMC_AxisControlSigma_PN, DB 890 *Chap. 4.3.2 'FB 890 - VMC_AxisControlSigma_PN - control block axis control for Sigma-5/7 PROFINET' page 213*

This block processes the user commands and passes them appropriately processed on to the drive via PROFINET.

```
CALL  "VMC_AxisControlSigma_PN" , "DI_AxisControlSigmaPN01"
 AxisEnable        :="AxCtrl1_AxisEnable"
 AxisReset         :="AxCtrl1_AxisReset"
 HomeExecute       :="AxCtrl1_HomeExecute"
 HomePosition      :="AxCtrl1_HomePosition"
 StopExecute       :="AxCtrl1_StopExecute"
 MvVelocityExecute:="AxCtrl1_MvVelExecute"
 MvRelativeExecute:="AxCtrl1_MvRelExecute"
 MvAbsoluteExecute:="AxCtrl1_MvAbsExecute"
 PositionDistance :="AxCtrl1_PositionDistance"
 Direction         :="AxCtrl1_Direction"
 Velocity          :="AxCtrl1_Velocity"
 Acceleration      :="AxCtrl1_Acceleration"
 Deceleration      :="AxCtrl1_Deceleration"
 JogPositive       :="AxCtrl1_JogPositive"
 JogNegative       :="AxCtrl1_JogNegative"
 JogVelocity       :="AxCtrl1_JogVelocity"
 JogAcceleration   :="AxCtrl1_JogAcceleration"
 JogDeceleration   :="AxCtrl1_JogDeceleration"
 AxisReady         :="AxCtrl1_AxisReady"
 AxisEnabled       :="AxCtrl1_AxisEnabled"
 AxisError         :="AxCtrl1_AxisError"
 AxisErrorID       :="AxCtrl1_AxisErrorID"
 DriveWarning      :="AxCtrl1_DriveWarning"
 DriveError        :="AxCtrl1_DriveError"
 DriveErrorID      :="AxCtrl1_DriveErrorID"
 IsHomed           :="AxCtrl1_IsHomed"
 ModeOfOperation   :="AxCtrl1_ModeOfOperation"
 PLCopenState      :="AxCtrl1_PLCopenState"
 ActualPosition    :="AxCtrl1_ActualPosition"
 ActualVelocity    :="AxCtrl1_ActualVelocity"
 CmdDone           :="AxCtrl1_CmdDone"
 CmdBusy           :="AxCtrl1_CmdBusy"
 CmdAborted        :="AxCtrl1_CmdAborted"
 CmdError          :="AxCtrl1_CmdError"
 CmdErrorID        :="AxCtrl1_CmdErrorID"
 DirectionPositive:="AxCtrl1_DirectionPos"
 DirectionNegative:="AxCtrl1_DirectionNeg"
 SWLimitMinActive  :="AxCtrl1_SWLimitMinActive"
 SWLimitMaxActive  :="AxCtrl1_SWLimitMaxActive"
 HWLimitMinActive  :="AxCtrl1_HWLimitMinActive"
 HWLimitMaxActive  :="AxCtrl1_HWLimitMaxActive"
 Axis              :="Axis01".Axis
```

For complex motion tasks, you can use the PLCopen blocks. Please specify the reference to the corresponding axis data at Axis in the axis DB.

Your project now includes the following blocks:

- OB 1 - Main
- OB 57 - DP Manufacturer Alarm
- OB 82 - I/O_FLT1
- OB 86 - Rack_FLT
- FB 890 - VMC_AxisControlSigma_PN with instance DB
- FB 891 - VMC_InitSigma_PN with instance DB
- UDT 860 - MC_Axis_REF
- UDT 890 - VMC_ConfigSigmaPN_REF

Sequence of operations

1. Select *'Project ➔ Compile all'* and transfer the project into your 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. To do this, call the *Init* block FB 891 - VMC_InitSigma_PN with *Enable* = TRUE.
   ⇨ The output *Valid* returns TRUE. In the event of a fault, you can determine the error by evaluating the *ErrorID*.

   You have to call the *Init* block again if you load a new axis DB or you have changed parameters on the *Init* block.

   Do not continue until the Init block does not report any errors!

3. Program your application with the FB 890 - VMC_AxisControlSigma_PN or with the PLCopen blocks.

4.1.5 Usage in Siemens TIA-Portal

4.1.5.1 Hardware configuration System MICRO respectively SLIO

Precondition

Overview

- Please use the Siemens TIA Portal from V.14 for the configuration.
- The configuration of the VIPA System MICRO respectively SLIO happens in the Siemens TIA Portal by means of a virtual PROFINET IO device.
  The PROFINET IO device is to be installed in the hardware catalog by means of a GSDML.
- For the PROFINET drive can be configured in the Siemens TIA Portal, the corresponding GSDML file must be installed.

Install GSDML file for System MICRO respectively SLIO

The installation of the PROFINET IO device 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 System MICRO or SLIO 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 ➔ I/O ➔ VIPA ...'*.

   From YASKAWA there are the following PROFINET IO devices:

   - System MICRO: *'VIPA Micro PLC'*
   - System SLIO: *'VIPA System SLIO'*

> *Thus, the VIPA components can be shown, you have to deactivate the ' Filter' of the hardware catalog.*

Install GSDML file for *Sigma-5* PROFINET drive

The GSDML file for the *Sigma-5* PROFINET drive can be found at www.yaskawa.eu.com under *'Service ➔ Drives & Motion Software'*.

Please use the following GSDML:

- GSDML-V2.3-Yaskawa-SGDV-OCB03A-20140228.xml

The installation happens with the following proceeding:

1. Download the according GSDML file for your drive.
2. Extract the file into your working directory.
3. Start the Siemens TIA Portal.
4. Close all the projects.
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 PROFINET IO device for the *Sigma-5* drive can be found at *'Additional field devices ➔ PROFINET IO ➔ Drives ➔ Yaskawa ...'*.

Add CPU in the project

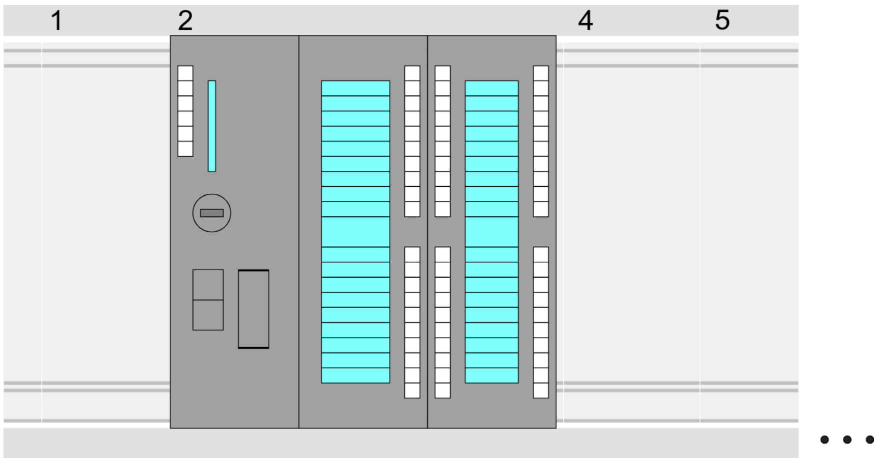
To be compatible with the Siemens SIMATIC TIA Portal the following steps should be executed:

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. Depending on the VIPA CPU used, select the following CPU from Siemens:

VIPA CPU	to configure as SIMATIC S7-300 > ...
M13-CCF0000 from V2.4.12	CPU 314C-2 PN/DP (6ES7 314-6EH04-0AB0 V3.3)
013-CCF0R00 from V2.4.12	CPU 314C-2 PN/DP (6ES7 314-6EH04-0AB0 V3.3)
014-CEF0R01 from V2.4.12	CPU 315-2 PN/DP (6ES7 315-2EH14-0AB0 V3.2)
015-CEFNR00 from V2.4.16	CPU 315-2 PN/DP (6ES7 315-2EH14-0AB0 V3.2)
015-CEFPR01 from V2.4.12	CPU 315-2 PN/DP (6ES7 315-2EH14-0AB0 V3.2)
017-CEFPR00 from V2.4.12	CPU 317-2PN/DP (6ES7 317-2EK14-0AB0 V3.2)

⇨ The CPU is inserted with a profile rail, such as the CPU 314C-2 PN/DP for System MICRO.



Device overview:

Module	...	Slot	...	Type	...
PLC...		2		CPU 314C-2PN/DP	
MPI interface...		2 X1		MPI/DP interface	
PROFINET interface...		2 X2		PROFINET interface	
DI24/DO16...		2 5		DI24/DO16	
AI5/AO2...		2 6		AI5/AO2	
Count...		2 7		Count	
...					

Connection CPU as PROFINET IO device

1. Switch in the *Project area* to *'Network view'*.

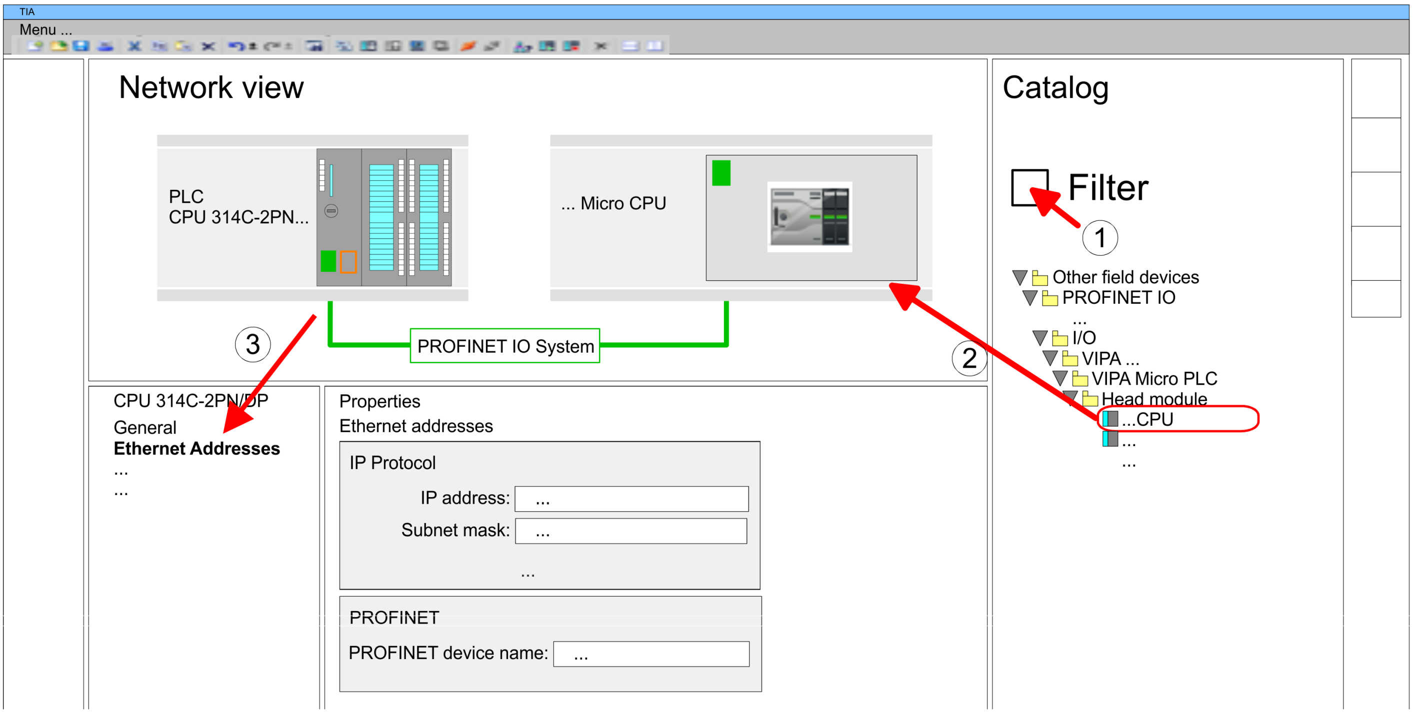
2. Navigate in the hardware catalog to *'Other field devices ➔ PROFINET IO ➔ I/O ➔ VIPA ...'* and connect the slave system to the CPU by dragging&dropping it from the hardware catalog to the *Network view* and connecting it via PROFINET to the CPU.

From YASKAWA there are the following PROFINET IO devices:

- System MICRO: *'VIPA Micro PLC'*
- System SLIO: *'VIPA System SLIO'*

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.



5. Select in the *Network view* the IO device such as *'VIPA MICRO PLC'* and switch to the *Device overview*.

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

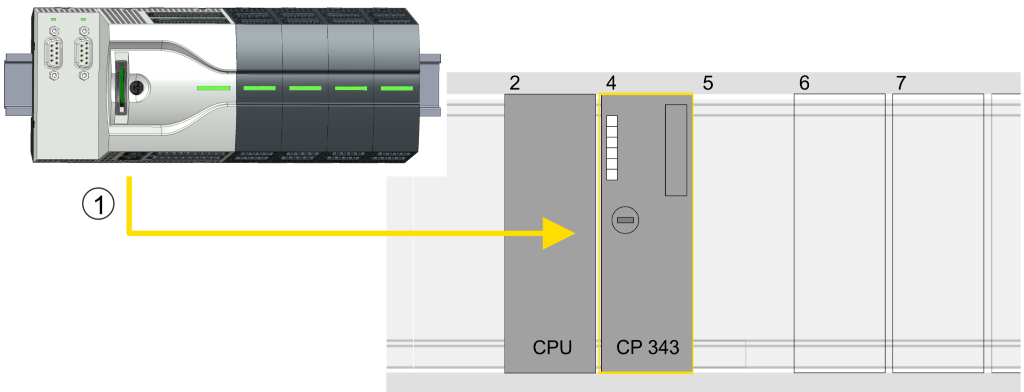
Configuration of Ethernet PG/OP channel

So that you may online access the according Ethernet interface, you have to assign IP address parameters by means of the "initialization". Please consider to use the same IP address data in your project for the 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.

1. As Ethernet PG/OP channel place at slot 4 of the Siemens system the Siemens 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 from the initialization.
3. Assign the CP to a *'Subnet'*. The IP address data are not accepted without assignment!
4. Transfer your project to your CPU.

   ⇨ The IP address data are stored in your current project. In the following this is shown exemplary on the System MICRO.



(1) Ethernet PG/OP channel

Device overview

Module	...	Slot	...	Type	...
PLC ...		2		CPU 314C-2PN/DP	
MPI/DP interface		2 X1		MPI/DP interface	
PROFINET interface		2 X2		PROFINET interface	
...		...		...	
CP 343-1		4		CP 343-1	
...		...		...	

Insert and configure *Sigma-5* PROFINET drive

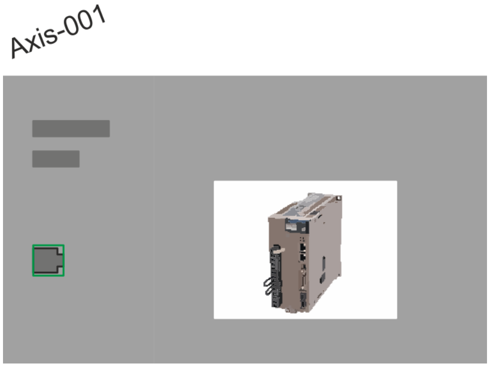
During configuration a *Sigma-5* PROFINET IO device must be configured for each axis.

1. Select your *Sigma-5* PROFINET drive *'SGDV-0CB...'* from the hardware catalog at *'Additional field devices ➔ PROFINET IO ➔ Drives ➔ Yaskawa ...'* and drag it onto the *'PROFINET-IO-System'*.

   ⇨ The *Sigma-5* PROFINET drive is connected to the IO controller and can now be configured.

2. Click at the *Sigma-5* IO device and open with *'Context menu ➔ Device configuration'* the *'Device overview'*.

3. Assign a suitable *'Device name'* such as Axis-001.



4. **Device overview**

Module	...	Slot	...	Type	...
Axis-001		0		SGDV-0CB03A	
PN-IO		0 X1		SGDV-0CB03A	
DO w/ Yaskawa telegr.100,PZD...		1		DO w/ Yaskawa telegr.100,PZD-16/14	
Parameter Access Point		1 1		Parameter Access Point	
Yaskawa telegram, PZD-16/14		1 2		Yaskawa telegram, PZD-16/14	
...		...		...	

In the hardware catalog, expand the *Sigma-5* PROFINET drive *'SGDV-0CB...'* to show its components and drag the component *'DO w/ YASKAWA telegr. 100...'* to *'Slot 1'* of the *Sigma-5* PROFINET drive.

⇨ Telegram 100 is inserted with the corresponding subgroups.

The connection between the axes in the hardware configuration and your user program is made by specifying the following module properties in the call parameters of FB 891 - VMC InitSigma_PN:

- *Module properties 'Parameter Access Point': Diagnostic address of slot 1 of the slot overview*
  - *FB 891 - VMC InitSigma_PN: ParaAccessPointAddress:*
    Setting of the diagnostic address of slot 1 of the slot overview.
- *Module properties 'YASKAWA Telegram PZD...':*
  Respective start address of the input/output address range.
  - *FB 891 - VMC InitSigma_PN: 'InputsStartAddress':*
    Setting of the start address of the input address range.
  - *FB 891 - VMC InitSigma_PN: 'OutputsStartAddress':*
    Setting of the start address of the output address range.
  - *FB 891 - VMC InitSigma_PN: 'LogicalAddress':*
    Setting of the of the smaller value of the start addresses of the input/output address range.

- User program 🖎 *166*
- FB 891 - VMC InitSigma_PN 🖎 *217*

Example hardware configuration

Slot	Component	...	I-Adr.	O-Adr.	Diagnostic address
0	SGDV-OCB03A				2037
X1	PN-IO				2036
X1 P1	Port 1				2035
X1 P2	Port 2				2034
1	DO with YASKAWA telegr.100, PZD-16/14				2033
1.1	Parameter Access Point				2033
1.2	YASKAWA telegram, PZD-16/14		284-311	288-319	

4.1.5.2 Hardware configuration System 300S

Precondition

Overview

- Please use the Siemens TIA Portal from V.14 for the configuration.
- For the PROFINET drive can be configured in the Siemens TIA Portal, the corresponding GSDML file must be installed.
- The blocks can be used with the following CPUs:
  – System 300S CPU 315-4PN43
  – System 300S CPU 315-4PN23
  – System 300S CPU 317-4PN23
- The configuration of the System 300S PROFINET CPU takes place in the Siemens TIA Portal as a corresponding Siemens CPU.
  – The CPUs 315-4PNxx are to be configured as Siemens CPU 315-2 PN/DP (6ES7 315-2EH14-0AB0 V3.2).
  – The CPU 317-4PN23 is to be configured as Siemens CPU 317-2 PN/DP (6ES7 317-2EK14-0AB0 V3.2).

Install GSDML file for *Sigma-5* PROFINET drive

The GSDML file for the *Sigma-5* PROFINET drive can be found at www.yaskawa.eu.com under *'Service ➔ Drives & Motion Software'*.

Please use the following GSDML:

- GSDML-V2.3-Yaskawa-SGDV-OCB03A-20140228.xml

The installation happens with the following proceeding:

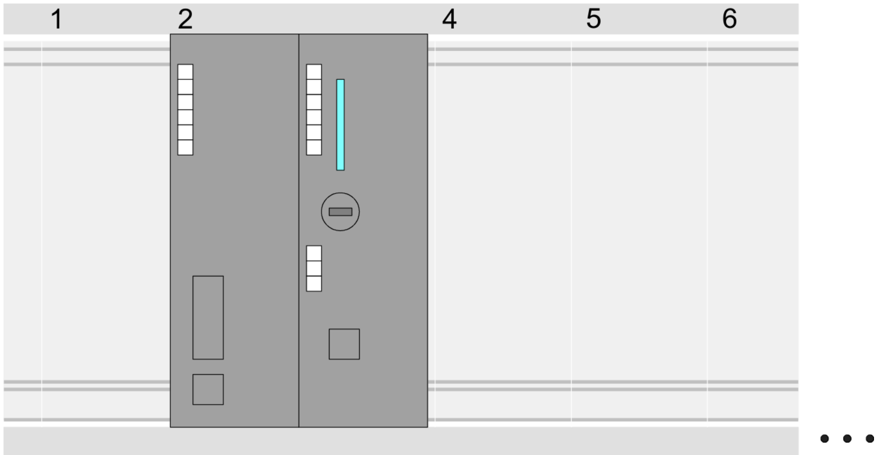
1. Download the according GSDML file for your drive.
2. Extract the file into your working directory.
3. Start the Siemens TIA Portal.
4. Close all the projects.
5. Select *'Options ➔ Install general station description file (GSD)'*.
6. Navigate to your working directory and install the according GSDML file.
   ⇨ After the installation the PROFINET IO device for the *Sigma-5* drive can be found at *'Additional field devices ➔ PROFINET IO ➔ Drives ➔ Yaskawa ...'*.

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. Depending on the VIPA CPU used, select the following CPU from Siemens:
   - The CPUs 315-4PNxx are to be configured as Siemens CPU 315-2 PN/DP (6ES7 315-2EH14-0AB0 V3.2).
   - The CPU 317-4PN23 is to be configured as Siemens CPU 317-2 PN/DP (6ES7 317-2EK14-0AB0 V3.2).

   ⇨ The CPU is inserted with a profile rail, such as the CPU 314C-2 PN/DP for VIPA CPU 315-4PN23.



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	
...		...		...	

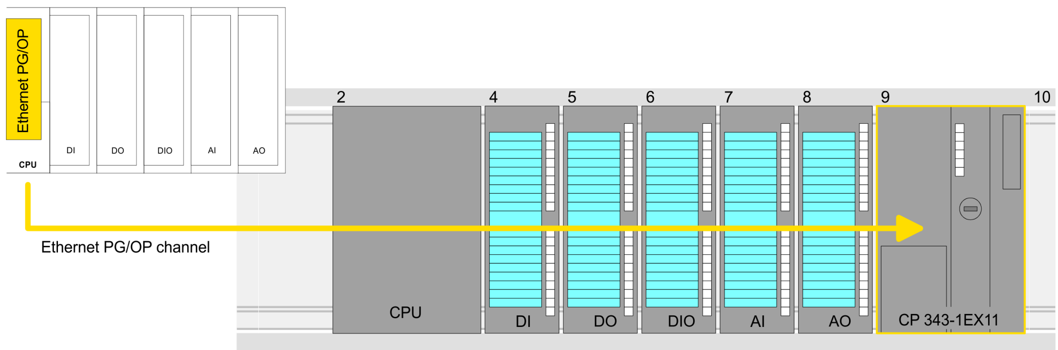
Configuration of Ethernet PG/OP channel

So that you may online access the according Ethernet interface, you have to assign IP address parameters by means of the "initialization". Please consider to use the same IP address data in your project for the 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.

1. For the Ethernet PG/OP channel, always configure a Siemens CP 343-1 (6GK7 343-1EX11 0XE0) as the last module after the inserted System 300 modules.
2. 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.
3. Assign the CP to a *'Subnet'*. The IP address data are not accepted without assignment!
4. Transfer your project to your CPU.

   ⇨ The IP address data are stored in your current project. As an example, this is shown below on the CPU 315-4PN23.



Device overview

Module	...	Slot	...	Type	...
PLC...		2		CPU 315-2PN/DP	
...		...		...	
DI...		4		DI...	
DO...		5		DO...	
DIO...		6		DIO...	
AI...		7		AI...	
AO...		8		AO...	
CP 343-1		9		CP 343-1	

Insert and configure *Sigma-5* PROFINET drive

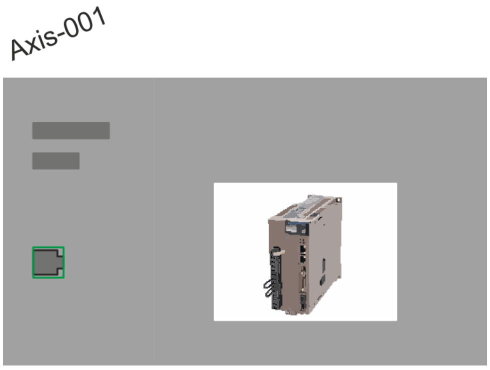
During configuration a *Sigma-5* PROFINET IO device must be configured for each axis.

1. Select your *Sigma-5* PROFINET drive *'SGDV-0CB...'* from the hardware catalog at *'Additional field devices ➔ PROFINET IO ➔ Drives ➔ Yaskawa ...'* and drag it onto the *'PROFINET-IO-System'*.

   ⇨ The *Sigma-5* PROFINET drive is connected to the IO controller and can now be configured.

2. Click at the *Sigma-5* IO device and open with *'Context menu ➔ Device configuration'* the *'Device overview'*.

3. Assign a suitable *'Device name'* such as Axis-001.



4. **Device overview**

Module	...	Slot	...	Type	...
Axis-001		0		SGDV-0CB03A	
PN-IO		0 X1		SGDV-0CB03A	
DO w/ Yaskawa telegr.100,PZD...		1		DO w/ Yaskawa telegr.100,PZD-16/14	
Parameter Access Point		1 1		Parameter Access Point	
Yaskawa telegram, PZD-16/14		1 2		Yaskawa telegram, PZD-16/14	
...		...		...	

In the hardware catalog, expand the *Sigma-5* PROFINET drive *'SGDV-0CB...'* to show its components and drag the component *'DO w/ YASKAWA telegr. 100...'* to *'Slot 1'* of the *Sigma-5* PROFINET drive.

⇨ Telegram 100 is inserted with the corresponding subgroups.

The connection between the axes in the hardware configuration and your user program is made by specifying the following module properties in the call parameters of FB 891 - VMC InitSigma_PN:

- *Module properties 'Parameter Access Point': Diagnostic address of slot 1 of the slot overview*
  - *FB 891 - VMC InitSigma_PN: ParaAccessPointAddress: Setting of the diagnostic address of slot 1 of the slot overview.*
- *Module properties 'YASKAWA Telegram PZD...': Respective start address of the input/output address range.*
  - *FB 891 - VMC InitSigma_PN: 'InputsStartAddress': Setting of the start address of the input address range.*
  - *FB 891 - VMC InitSigma_PN: 'OutputsStartAddress': Setting of the start address of the output address range.*
  - *FB 891 - VMC InitSigma_PN: 'LogicalAddress': Setting of the of the smaller value of the start addresses of the input/output address range.*

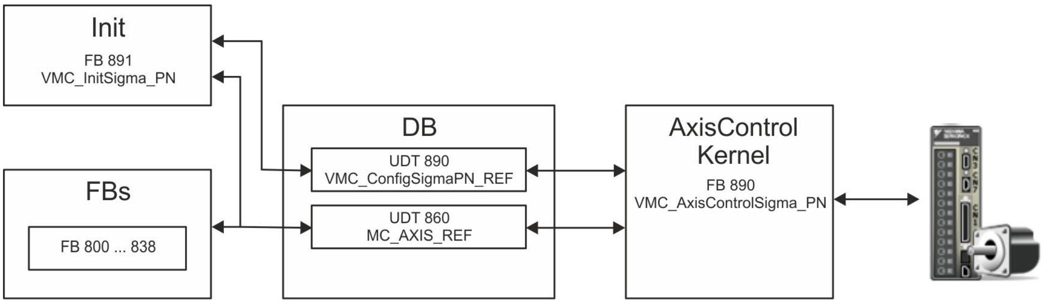
- User program ➪ *166*
- FB 891 - VMC InitSigma_PN ➪ *217*

Example hardware configuration

Slot	Component	...	I-Adr.	O-Adr.	Diagnostic address
0	SGDV-OCB03A				2037
X1	PN-IO				2036
X1 P1	Port 1				2035
X1 P2	Port 2				2034
1	DO with YASKAWA telegr.100, PZD-16/14				2033
1.1	Parameter Access Point				2033
1.2	YASKAWA telegram, PZD-16/14		284-311	288-319	

4.1.5.3 User program

4.1.5.3.1 Program structure



- DB
  A data block (axis DB) for configuration and status data must be created for each axis of a drive. The data block consists of the following data structures:
  – UDT 890 - *VMC_Config*SigmaPN_REF
    The data structure describes the structure of the configuration of the drive.
    Specific data structure for *Sigma-5/7* PROFINET.
  – UDT 860 - *MC_AXIS_REF*
    The data structure describes the structure of the parameters and status information of drives.
    General data structure for all drives and bus systems.
- FB 891 - *VMC_Init*Sigma_PN
  – The *Init*t block is used to configure an axis.
  – Specific block for *Sigma-5/7* PROFINET.
  – The configuration data for the initialization must be stored in the *axis DB*.
- FB 890 - *VMC_AxisControl*Sigma_PN
  – Specific block for *Sigma-5/7* PROFINET.
  – This block is a combination of *Kernel* and *AxisControl* and communicates with the drive via PROFINET, processes the user requests and returns status messages.
  – This block supports simple motion commands and returns all relevant status messages.
  – The exchange of the data takes place by means of the *axis DB*.
  – For motion control and status query, via the instance data of the block you can link a visualization.
  – In addition to the FB 890 - *VMC_AxisControl*Sigma_PN, *PLCopen* blocks can be used.
- FB 800 ... FB 838 - *PLCopen*
  – The PLCopen blocks are used to program motion sequences and status queries.
  – General blocks for all drives and bus systems.

4.1.5.3.2 Programming

Include library

1. Go to the service area of www.vipa.com.
2. Download the *Simple Motion Control* library from the download area at *‘VIPA Lib’*.
   The library is available as packed zip file for the corresponding TIA Portal version.
3. Start your un-zip application with a double click on the file ...TIA_V*xx*.zip and copy all the files and folders in a work directory for the Siemens TIA Portal.
4. Switch to the *Project view* of the Siemens TIA Portal.
5. Choose "Libraries" from the task cards on the right side.
6. Click at "Global library".

7. Click on the free area inside the *'Global Library'* and select *'Context menu ➔ Retrieve library'*.

8. Navigate to your work directory and load the file ...Simple Motion.zal*xx*.

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.

Copy blocks into project

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

- *Sigma* PROFINET:
  – UDT 890 - VMC_ConfigSigmaPN_REF ↬ *Chap. 4.3.1 'UDT 890 - VMC_ConfigSigmaPN_REF - Sigma-5/7 PROFINET Data structure axis configuration' page 213*
  – FB 890 - VMC_AxisControlSigma_PN ↬ *Chap. 4.3.2 'FB 890 - VMC_AxisControlSigma_PN - control block axis control for Sigma-5/7 PROFINET' page 213*
  – FB 891 - VMC_InitSigma_PN ↬ *Chap. 4.3.3 'FB 891 - VMC_InitSigma_PN - Sigma-5/7 PROFINET initialization' page 217*
- Axis control
  – UDT 860 - MC_AXIS_REF ↬ *Chap. 9.2.1 'UDT 860 - MC_AXIS_REF - Data structure axis data' page 363*
  – FB 860 - VMC_AxisControl ↬ *Chap. 9.2.2 'FB 860 - VMC_AxisControl - Control block axis control' page 363*

Create axis DB

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 assign it the name "Axis01". The DB number can freely be selected such as DB 10. Specify DB 10 and create this as a global DB with [OK].
   ⇨ The block is created and opened.
3. In "Axis01" create the following variables:
   - *'Config'* of Type UDT 890 - VMC_ConfigSigmaPN_REF.
     These are specific axis configuration data.
   - *'Config'* of Type UDT 860 - MC_AXIS_REF.
     During operation, all operating data of the axis are stored here.

OB 1 - configuration of the axes

Open OB 1 and program the following FB calls with associated DBs:

FB 891 - VMC_InitSigma_PN, DB 891

The connection between the axes in the hardware configuration and your user program is made by specifying the following module properties in the call parameters of FB 891 - VMC InitSigma_PN:

- *Module properties ‘Parameter Access Point’: Diagnostic address of slot 1 of the slot overview*
  - *FB 891 - VMC InitSigma_PN: ParaAccessPointAddress: Setting of the diagnostic address of slot 1 of the slot overview.*
- *Module properties ‘YASKAWA Telegram PZD...’: Respective start address of the input/output address range.*
  - *FB 891 - VMC InitSigma_PN: ‘InputsStartAddress’: Setting of the start address of the input address range.*
  - *FB 891 - VMC InitSigma_PN: ‘OutputsStartAddress’: Setting of the start address of the output address range.*
  - *FB 891 - VMC InitSigma_PN: ‘LogicalAddress’: Setting of the of the smaller value of the start addresses of the input/output address range.*

- Hardware configuration ↪ *155*
- FB 891 - VMC InitSigma_PN ↪ *217*

Example hardware configuration

Slot	Component	...	I-Adr.	O-Adr.	Diagnostic address
0	SGDV-OCB03A				2037
X1	PN-IO				2036
X1 P1	Port 1				2035
X1 P2	Port 2				2034
1	DO with YASKAWA telegr.100, PZD-16/14				2033
1.1	Parameter Access Point				2033
1.2	YASKAWA telegram, PZD-16/14		284-311	288-319	

Example call

```
CALL  "VMC_InitSigma_PN" , "VMC_InitSigma_PN_1"
Enable                     :="InitS5PN1_Enable"
LogicalAddress             :=284 //HW-Konfig: Smallest IO addr.
ParaAccessPointAdress      :=2033 //HW-Konfig: Diag addr.
InputsStartAddress         :=284 //HW-Konfig: Telegr.100 start I addr.
OutputsStartAddress        :=288 //HW-Konfig: Telegr. 100 start O addr.
EncoderType                :=1
EncoderResolutionBits      :=20
FactorPosition             :=1.048576e+006
FactorVelocity             :=1.048576e+006
FactorAcceleration         :=1.048576e+006
OffsetPosition             :=0.000000e+000
MaxVelocityApp             :=5.000000e+001
MaxAccelerationApp         :=1.000000e+002
MaxDecelerationApp         :=1.000000e+002
MaxVelocityDrive           :=6.000000e+001
MaxPosition                :=1.048500e+003
MinPosition                :=-1.048514e+003
EnableMaxPosition          :=TRUE
EnableMinPosition          :=TRUE
```

```
MinUserPosition       :="InitS5PN1_MinUserPos"
MaxUserPosition       :="InitS5PN1_MaxUserPos"
Valid                 :="InitS5PN1_Valid"
Error                 :="InitS5PN1_Error"
ErrorID               :="InitS5PN1_ErrorID"
Config                :="Axis01".Config
Axis                  :="Axis01".Axis
```

Connecting the AxisControl

FB 890 - VMC_AxisControlSigma_PN, DB 890 *Chap. 4.3.2 'FB 890 - VMC_AxisControlSigma_PN - control block axis control for Sigma-5/7 PROFINET' page 213*

This block processes the user commands and passes them appropriately processed on to the drive via PROFINET.

```
CALL  "VMC_AxisControlSigma_PN" , "DI_AxisControlSigmaPN01"
 AxisEnable        :="AxCtrl1_AxisEnable"
 AxisReset         :="AxCtrl1_AxisReset"
 HomeExecute       :="AxCtrl1_HomeExecute"
 HomePosition      :="AxCtrl1_HomePosition"
 StopExecute       :="AxCtrl1_StopExecute"
 MvVelocityExecute:="AxCtrl1_MvVelExecute"
 MvRelativeExecute:="AxCtrl1_MvRelExecute"
 MvAbsoluteExecute:="AxCtrl1_MvAbsExecute"
 PositionDistance :="AxCtrl1_PositionDistance"
 Direction         :="AxCtrl1_Direction"
 Velocity          :="AxCtrl1_Velocity"
 Acceleration      :="AxCtrl1_Acceleration"
 Deceleration      :="AxCtrl1_Deceleration"
 JogPositive       :="AxCtrl1_JogPositive"
 JogNegative       :="AxCtrl1_JogNegative"
 JogVelocity       :="AxCtrl1_JogVelocity"
 JogAcceleration   :="AxCtrl1_JogAcceleration"
 JogDeceleration   :="AxCtrl1_JogDeceleration"
 AxisReady         :="AxCtrl1_AxisReady"
 AxisEnabled       :="AxCtrl1_AxisEnabled"
 AxisError         :="AxCtrl1_AxisError"
 AxisErrorID       :="AxCtrl1_AxisErrorID"
 DriveWarning      :="AxCtrl1_DriveWarning"
 DriveError        :="AxCtrl1_DriveError"
 DriveErrorID      :="AxCtrl1_DriveErrorID"
 IsHomed           :="AxCtrl1_IsHomed"
 ModeOfOperation   :="AxCtrl1_ModeOfOperation"
 PLCopenState      :="AxCtrl1_PLCopenState"
 ActualPosition    :="AxCtrl1_ActualPosition"
 ActualVelocity    :="AxCtrl1_ActualVelocity"
 CmdDone           :="AxCtrl1_CmdDone"
 CmdBusy           :="AxCtrl1_CmdBusy"
 CmdAborted        :="AxCtrl1_CmdAborted"
 CmdError          :="AxCtrl1_CmdError"
 CmdErrorID        :="AxCtrl1_CmdErrorID"
 DirectionPositive:="AxCtrl1_DirectionPos"
 DirectionNegative:="AxCtrl1_DirectionNeg"
 SWLimitMinActive :="AxCtrl1_SWLimitMinActive"
 SWLimitMaxActive :="AxCtrl1_SWLimitMaxActive"
 HWLimitMinActive :="AxCtrl1_HWLimitMinActive"
 HWLimitMaxActive :="AxCtrl1_HWLimitMaxActive"
 Axis              :="Axis01".Axis
```

For complex motion tasks, you can use the PLCopen blocks. Please specify the reference to the corresponding axis data at Axis in the axis DB.

Your project now includes the following blocks:

- OB 1 - Main
- OB 57 - DP Manufacturer Alarm
- OB 82 - I/O_FLT1
- OB 86 - Rack_FLT
- FB 890 - VMC_AxisControlSigma_PN with instance DB
- FB 891 - VMC_InitSigma_PN with instance DB
- UDT 860 - MC_Axis_REF
- UDT 890 - VMC_ConfigSigmaPN_REF

Sequence of operations

1. Select *'Project ➔ Compile all'* and transfer the project into your 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. To do this, call the *Init* block FB 891 - VMC_InitSigma_PN with *Enable* = TRUE.

   ⇨ The output *Valid* returns TRUE. In the event of a fault, you can determine the error by evaluating the *ErrorID*.

   You have to call the *Init* block again if you load a new axis DB or you have changed parameters on the *Init* block.

   > *Do not continue until the Init block does not report any errors!*

3. Program your application with the FB 890 - VMC_AxisControlSigma_PN or with the PLCopen blocks.

4.2 Usage *Sigma-7* PROFINET

4.2.1 Overview

Precondition

- SPEED7 Studio from V1.8
  or
- Siemens SIMATIC Manager from V 5.5, SP2 respectively TIA Portal V 14 & *Simple Motion Control Library*
- CPU with PROFINET functionality, such as CPU 015-CEFPR01
- *Sigma-7* drive with PROFINET connection

Steps of configuration

1. Setting parameters on the drive
   - The setting of the parameters happens by means of the software tool *Sigma Win+*.
2. Hardware configuration in the VIPA *SPEED7 Studio*, Siemens SIMATIC Manager or TIA Portal.
   - Configuring a CPU with PROFINET functionality.
   - Configuring a *Sigma-7* PROFINET drive.
3. Programming in the VIPA *SPEED7 Studio*, Siemens SIMATIC Manager or TIA Portal.
   - 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.

4.2.2 Set the parameters on the drive

Parameter Sigma-7

CAUTION!
Before the commissioning, you have to adapt your drive to your application with the *Sigma Win+* software tool! More may be found in the manual of your drive.

The following parameter must be set via *Sigma Win+* to match the *Simple Motion Control Library*:

Sigma-7 (24bit encoder)

Servopack Parameter	Address	Name	Value
PnB32	606Dh	Velocity Window	1000 Velocity units
PnB34	606Eh	Velocity Window Time	50 ms
PnC20	0922h	Telegram Selection (100: General Telegram: All OP modes)	100

Please note that you have to enable the corresponding direction of your axis in accordance to your requirements. For this use the parameters Pn50A (P-OT) respectively Pn50B (N-OT) in Sigma Win+.

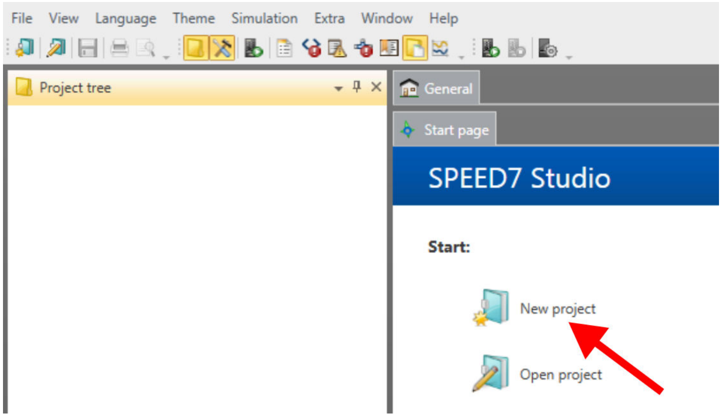
4.2.3 Usage in VIPA *SPEED7 Studio*

4.2.3.1 Hardware configuration System MICRO

Add CPU in the project

Please use the *SPEED7 Studio* V1.8 and up for the configuration.

1. Start the *SPEED7 Studio*.

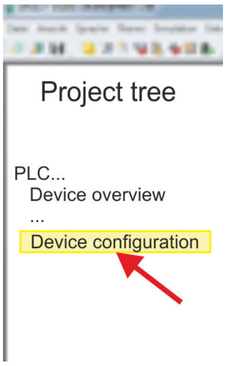


2. Create a new project at the start page with *'New project'* and assign a *'Project name'*.

   ⇨ A new project is created and the view *'Devices and networking'* is shown.

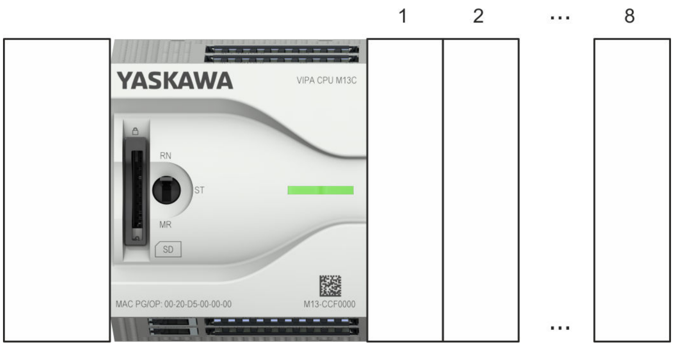
3. Click in the *Project tree* at *'Add new device ...'*.

   ⇨ A dialog for device selection opens.



4. Select from the *'Device templates'* the System MICRO CPU M13-CCF0000 V2.4.... and click at [OK].

   ⇨ The CPU is inserted in *'Devices and networking'* and the *'Device configuration'* is opened.



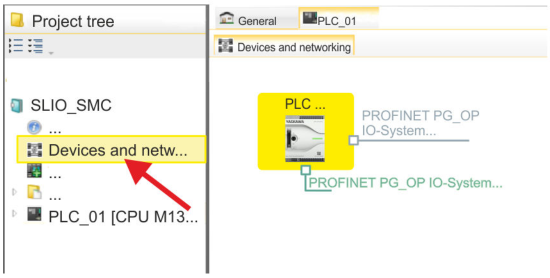
Device configuration

Slot	Module	...	...	...	...
0	CPU M13-CCF0000				
-X2	MPI interface				
-X3	PROFINET PG_OP IO-System				
...	...			...	

Configuration of Ethernet PG/OP channel

1. Click in the *Project tree* at *'Devices and networking'*.

   ⇨ You will get a graphical object view of your CPU. Here both interfaces of the PROFINET respectively Ethernet PG / OP channel switch are listed with identical name.



2. Click at one of the network *'PROFINET PG_OP_Ethernet IO-System ...'*.
3. Select *'Context menu ➔ Interface properties'*.

   ⇨ A dialog window opens. Here you can enter the IP address data for your Ethernet PG/OP channel. You get valid IP address parameters from your system administrator.

4. Confirm with [OK].

   ⇨ The IP address data are stored in your project listed in *'Devices and networking'* at *'Local components'*.

   After transferring your project your CPU can be accessed via Ethernet PG/OP channel with the set IP address data.

Installing the GSDML file

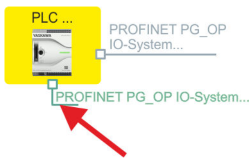
For the *Sigma-7* PROFINET drive can be configured in the *SPEED7 Studio*, the corresponding GSDML file must be installed. Usually, the *SPEED7 Studio* is delivered with current GSDML files and you can skip this part. If your GSDML file is not up-to date, you will find the latest GSDML file for the *Sigma-7* PROFINET drive under www.yaskawa.eu.com at *'Service ➔ Drives & Motion Software'*.

1. Download the according GSDML file for your drive. Unzip this if necessary.
2. Navigate to your *SPEED7 Studio*.
3. Open the corresponding dialog window by clicking on *'Extras ➔ Install device description (PROFINET - GSDML)'*.
4. Under *'Source path'*, specify the GSDML file and install it with [Install].

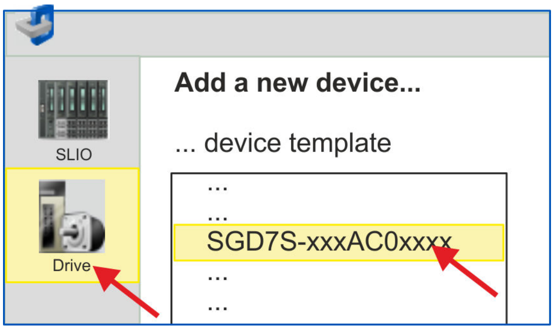
   ⇨ The devices of the GSDML file are now available.

Add a *Sigma-7* drive

1. Click in the Project tree at *'Devices and networking'*.
2. Click here at *'PROFINET PG_OP_Ethernet IO-System ...'* and select *'Context menu ➔ Add new device'*.



   ⇨ The device template for selecting PROFINET device opens.



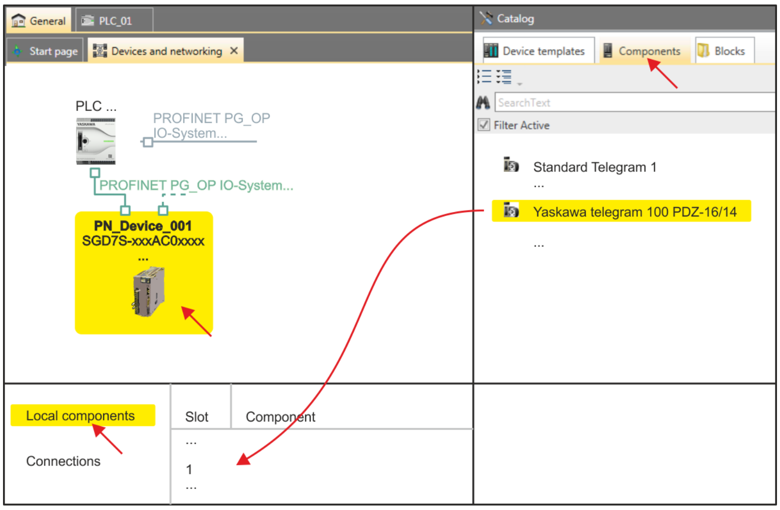
3. Select your *Sigma-7* drive:

- SGD7S-xxxAC0xxxx

Confirm your input with [OK]. If your drive does not exist, you must install the corresponding GSDML file as described above.

⇨ The *Sigma-7* drive is connected to your PROFINET IO controller.

4. Click on the *Sigma-7* drive.



5. At *'Catalog'* select the *'Components'* tab.

⇨ The telegrams for the *Sigma-7* drive are listed.

6. Select *'Yaskawa telegram 100 PZD...'* drag&drop it to *'Slot 1'* of *'Local components'*.

   ⇨ Telegram 100 is inserted with the corresponding subgroups.

> *The connection between the axes in the hardware configuration and your user program is made by specifying the following module properties in the call parameters of FB 891 - VMC Init-Sigma_PN:*
>
> – *Module properties 'Parameter Access Point': Diagnostic address of slot 1 of the slot overview*
>   – *FB 891 - VMC InitSigma_PN: ParaAccessPointAddress: Setting of the diagnostic address of slot 1 of the slot overview.*
>
> – *Module properties 'YASKAWA Telegram PZD...': Respective start address of the input/output address range.*
>   – *FB 891 - VMC InitSigma_PN: 'InputsStartAddress': Setting of the start address of the input address range.*
>   – *FB 891 - VMC InitSigma_PN: 'OutputsStartAddress': Setting of the start address of the output address range.*
>   – *FB 891 - VMC InitSigma_PN: 'LogicalAddress': Setting of the of the smaller value of the start addresses of the input/output address range.*

- User program ⇨ *182*
- FB 891 - VMC InitSigma_PN ⇨ *217*

Example hardware configuration

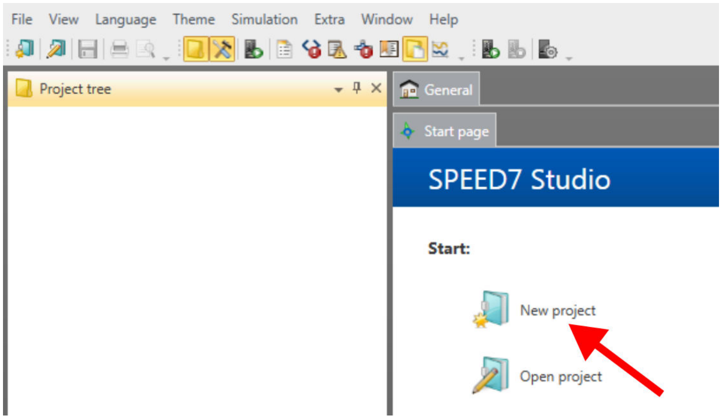
Slot	Component	...	I-Adr.	O-Adr.	Diagnostic address
0	SGD7S-xxxAC0xxxx		2035		2035
X1	PN-IO		2034		2034
X1 P1	Port 1		2033		2033
X1 P2	Port 2		2032		2032
1	DO with YASKAWA telegr.100, PZD-16/14		2044		2044
1.1	Parameter Access Point		2044		2044
1.2	YASKAWA telegram, PZD-16/14		28-55	32-63	2044

4.2.3.2 Hardware configuration System SLIO

Add CPU in the project

Please use the *SPEED7 Studio* V1.8 and up for the configuration.

1. Start the *SPEED7 Studio*.

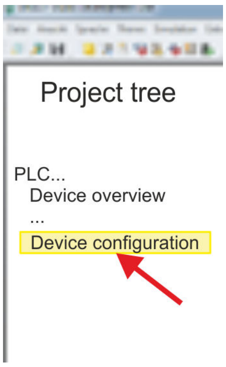


2. Create a new project at the start page with *'New project'* and assign a *'Project name'*.

   ⇨ A new project is created and the view *'Devices and networking'* is shown.

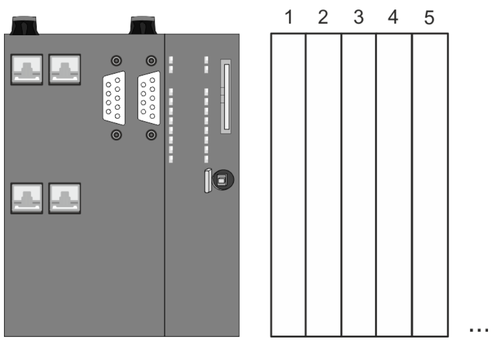
3. Click in the *Project tree* at *'Add new device ...'*.

   ⇨ A dialog for device selection opens.



4. Select from the *'Device templates'* your PROFINET CPU e.g..CPU 015-CEFPR01 and click at [OK].

   ⇨ The CPU is inserted in *'Devices and networking'* and the *'Device configuration'* is opened.

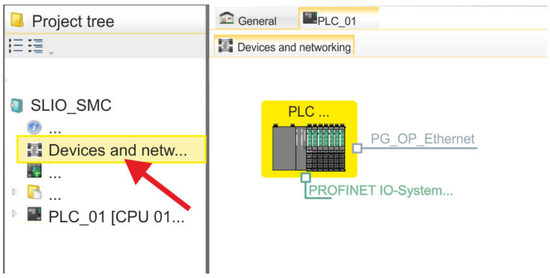


Device configuration

Slot	Module	...	...	...	...
0	CPU 015-CEFPR01				
-X1	PG_OP_Ethernet				
-X3	MPI interface				
-X4	PROFINET-IO-System				
...	...			...	

Configuration of Ethernet PG/OP channel

1. Click in the *Project tree* at *'Devices and networking'*.
   ⇨ You will get a graphical object view of your CPU.



2. Click at the network *'PG_OP_Ethernet'*.
3. Select *'Context menu ➔ Interface properties'*.
   ⇨ A dialog window opens. Here you can enter the IP address data for your Ethernet PG/OP channel. You get valid IP address parameters from your system administrator.
4. Confirm with [OK].
   ⇨ The IP address data are stored in your project listed in *'Devices and networking'* at *'Local components'*.

   After transferring your project your CPU can be accessed via Ethernet PG/OP channel with the set IP address data.

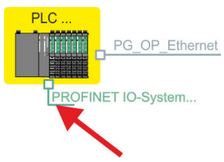
Installing the GSDML file

For the *Sigma-7* PROFINET drive can be configured in the *SPEED7 Studio*, the corresponding GSDML file must be installed. Usually, the *SPEED7 Studio* is delivered with current GSDML files and you can skip this part. If your GSDML file is not up-to date, you will find the latest GSDML file for the *Sigma-7* PROFINET drive under www.yaskawa.eu.com at *'Service ➔ Drives & Motion Software'*.

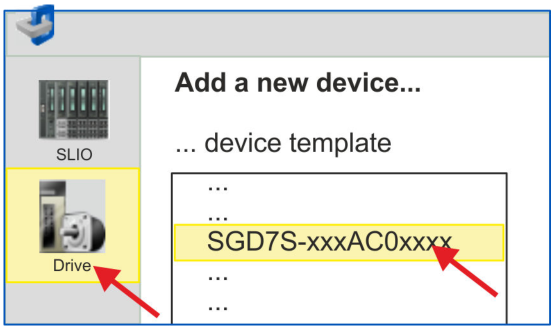
1. Download the according GSDML file for your drive. Unzip this if necessary.
2. Navigate to your *SPEED7 Studio*.
3. Open the corresponding dialog window by clicking on *'Extras ➔ Install device description (PROFINET - GSDML)'*.
4. Under *'Source path'*, specify the GSDML file and install it with [Install].
   ⇨ The devices of the GSDML file are now available.

Add a *Sigma-7* drive

1. Click in the Project tree at *'Devices and networking'*.
2. Click here at *'PROFINET IO-System ...'* and select *'Context menu ➔ Add new device'*.



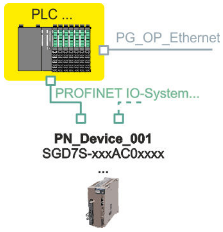
   ⇨ The device template for selecting PROFINET device opens.



3. Select your *Sigma-7* drive:

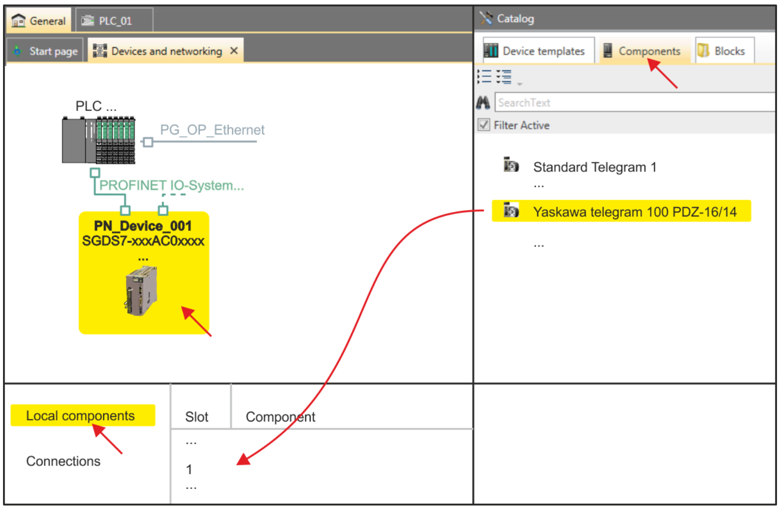
- SGDS7-xxxAC0xxxx

Confirm your input with [OK]. If your drive does not exist, you must install the corresponding GSDML file as described above.



⇨ The *Sigma-7* drive is connected to your PROFINET IO controller.

4. Click on the *Sigma-7* drive



5. At *'Catalog'* select the *'Components'* tab.

⇨ The telegrams for the *Sigma-7* drive are listed.

6. Select *'Yaskawa telegram 100 PZD...'* drag&drop it to *'Slot 1'* of *'Local components'*.

   ⇨ Telegram 100 is inserted with the corresponding subgroups.

> *The connection between the axes in the hardware configuration and your user program is made by specifying the following module properties in the call parameters of FB 891 - VMC Init-Sigma_PN:*
>
> – *Module properties 'Parameter Access Point': Diagnostic address of slot 1 of the slot overview*
>   – *FB 891 - VMC InitSigma_PN: ParaAccessPointAddress: Setting of the diagnostic address of slot 1 of the slot overview.*
>
> – *Module properties 'YASKAWA Telegram PZD...': Respective start address of the input/output address range.*
>   – *FB 891 - VMC InitSigma_PN: 'InputsStartAddress': Setting of the start address of the input address range.*
>   – *FB 891 - VMC InitSigma_PN: 'OutputsStartAddress': Setting of the start address of the output address range.*
>   – *FB 891 - VMC InitSigma_PN: 'LogicalAddress': Setting of the of the smaller value of the start addresses of the input/output address range.*

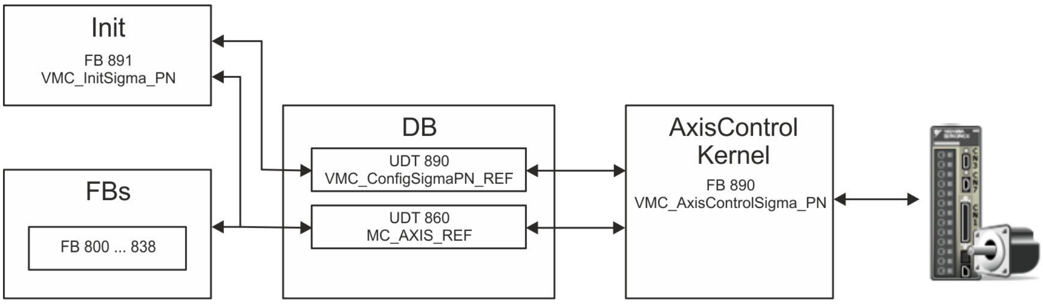
- User program ⇨ *182*
- FB 891 - VMC InitSigma_PN ⇨ *217*

Example hardware configuration

Slot	Component	...	I-Adr.	O-Adr.	Diagnostic address
0	SGD7S-xxxAC0xxxx		2035		2035
X1	PN-IO		2034		2034
X1 P1	Port 1		2033		2033
X1 P2	Port 2		2032		2032
1	DO with YASKAWA telegr.100, PZD-16/14		2044		2044
1.1	Parameter Access Point		2044		2044
1.2	YASKAWA telegram, PZD-16/14		28-55	32-63	2044

4.2.3.3 User program

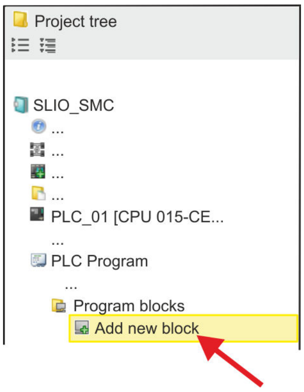
4.2.3.3.1 Program structure



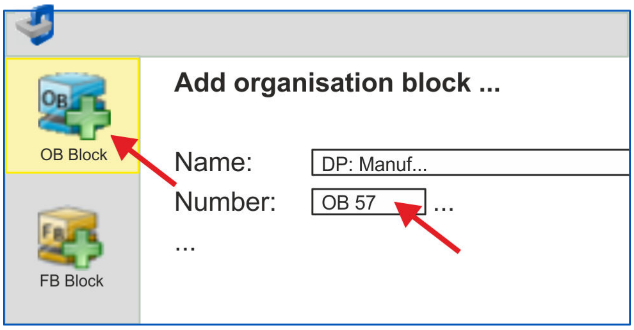
- DB
  A data block (axis DB) for configuration and status data must be created for each axis of a drive. The data block consists of the following data structures:
  – UDT 890 - *VMC_Config*SigmaPN_REF
    The data structure describes the structure of the configuration of the drive.
    Specific data structure for *Sigma-5/7* PROFINET.
  – UDT 860 - *MC_AXIS_REF*
    The data structure describes the structure of the parameters and status information of drives.
    General data structure for all drives and bus systems.
- FB 891 - *VMC_Init*Sigma_PN
  – The *Init*t block is used to configure an axis.
  – Specific block for *Sigma-5/7* PROFINET.
  – The configuration data for the initialization must be stored in the *axis DB*.
- FB 890 - *VMC_AxisControl*Sigma_PN
  – Specific block for *Sigma-5/7* PROFINET.
  – This block is a combination of *Kernel* and *AxisControl* and communicates with the drive via PROFINET, processes the user requests and returns status messages.
  – This block supports simple motion commands and returns all relevant status messages.
  – The exchange of the data takes place by means of the *axis DB*.
  – For motion control and status query, via the instance data of the block you can link a visualization.
  – In addition to the FB 890 - *VMC_AxisControl*Sigma_PN, *PLCopen* blocks can be used.
- FB 800 ... FB 838 - *PLCopen*
  – The PLCopen blocks are used to program motion sequences and status queries.
  – General blocks for all drives and bus systems.

4.2.3.3.2 Programming

Create interrupt OBs



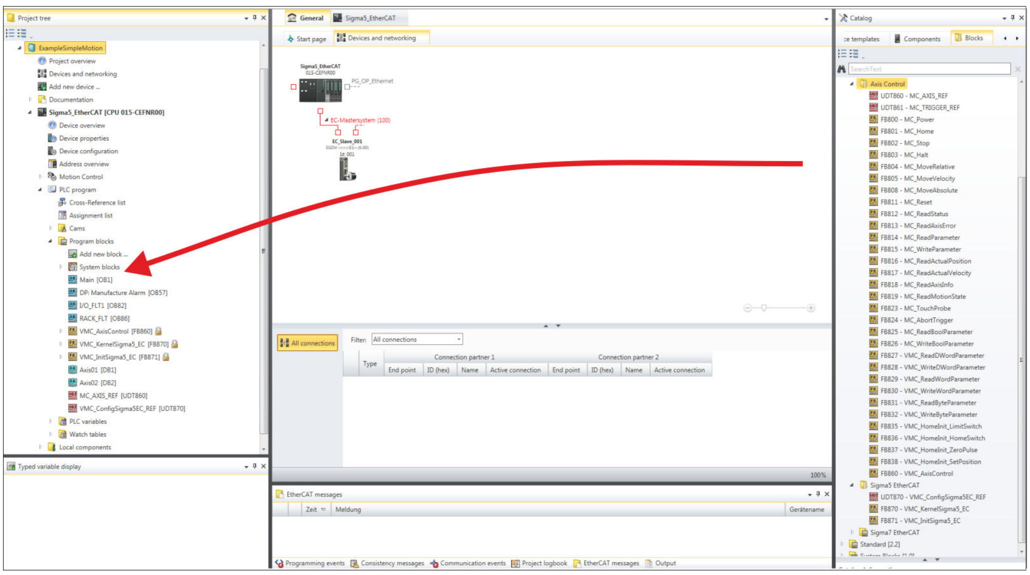
1. Click in the *Project tree* within the CPU at *'PLC program'*, *'Program blocks'* at *'Add New block'*.



⇨ The dialog *'Add block'* is opened.

2. Select the block type *'OB block'* and add one after the other OB 57, OB 82 and OB 86 to your project.

Copy blocks into project



In the *'Catalog'*, open the *'Simple Motion Control'* library at *'Blocks'* and drag and drop the following blocks into *'Program blocks'* of the *Project tree*:

- *Sigma* PROFINET:
  - UDT 890 - VMC_ConfigSigmaPN_REF *Chap. 4.3.1 'UDT 890 - VMC_ConfigSigmaPN_REF - Sigma-5/7 PROFINET Data structure axis configuration' page 213*
  - FB 890 - VMC_AxisControlSigma_PN *Chap. 4.3.2 'FB 890 - VMC_AxisControlSigma_PN - control block axis control for Sigma-5/7 PROFINET' page 213*
  - FB 891 - VMC_InitSigma_PN *Chap. 4.3.3 'FB 891 - VMC_InitSigma_PN - Sigma-5/7 PROFINET initialization' page 217*
- Axis control
  - UDT 860 - MC_AXIS_REF *Chap. 9.2.1 'UDT 860 - MC_AXIS_REF - Data structure axis data' page 363*
  - FB 860 - VMC_AxisControl *Chap. 9.2.2 'FB 860 - VMC_AxisControl - Control block axis control' page 363*

Create axis DB

1. Add a new DB as your *axis DB* to your project. Click in the *Project tree* within the CPU at *'PLC program'*, *'Program blocks'* at *'Add New block'*, select the block type *'DB block'* and assign the name "Axis01" to it. The DB number can freely be selected such as DB 10.

   ⇨ The block is created and opened.

2. - In "Axis01", create the variable "Config" of type UDT 890. These are specific axis configuration data.
   - In "Axis01", create the variable "Axis" of type UDT 860. During operation, all operating data of the axis are stored here.

Axis01 [DB10]

Data block structure

		Adr...	Name	Data type	...
		...	Config	UDT	[890]
		...	Axis	UDT	[860]

OB 1 - configuration of the axes

Open OB 1 and program the following FB calls with associated DBs:

FB 891 - VMC_InitSigma_PN, DB 891

The connection between the axes in the hardware configuration and your user program is made by specifying the following module properties in the call parameters of FB 891 - VMC InitSigma_PN:

- *Module properties 'Parameter Access Point': Diagnostic address of slot 1 of the slot overview*
  - *FB 891 - VMC InitSigma_PN: ParaAccessPointAddress:*
    Setting of the diagnostic address of slot 1 of the slot overview.
- *Module properties 'YASKAWA Telegram PZD...':*
  Respective start address of the input/output address range.
  - *FB 891 - VMC InitSigma_PN: 'InputsStartAddress':*
    Setting of the start address of the input address range.
  - *FB 891 - VMC InitSigma_PN: 'OutputsStartAddress':*
    Setting of the start address of the output address range.
  - *FB 891 - VMC InitSigma_PN: 'LogicalAddress':*
    Setting of the of the smaller value of the start addresses of the input/output address range.

- Hardware configuration ⇨ *172*
- FB 891 - VMC InitSigma_PN ⇨ *217*

Example hardware configuration

Slot	Component	...	I-Adr.	O-Adr.	Diagnostic address
0	SGD7S-xxxAC0xxxx		2035		2035
X1	PN-IO		2034		2034
X1 P1	Port 1		2033		2033
X1 P2	Port 2		2032		2032
1	DO with YASKAWA telegr.100, PZD-16/14		2044		2044
1.1	Parameter Access Point		2044		2044
1.2	YASKAWA telegram, PZD-16/14		28-55	32-63	2044

Example call

```
CALL  "VMC_InitSigma_PN" , "VMC_InitSigma_PN_1"
Enable                     :="InitS7PN1_Enable"
LogicalAddress             :=28 //HW-Konfig: Smallest IO addr.
ParaAccessPointAdress      :=2044 //HW-Konfig: Diag addr.
InputsStartAddress         :=28 //HW-Konfig: Telegr.100 start I addr.
OutputsStartAddress        :=32 //HW-Konfig: Telegr. 100 start O addr.
EncoderType                :=1
EncoderResolutionBits      :=20
FactorPosition             :=1.048576e+006
FactorVelocity             :=1.048576e+006
FactorAcceleration         :=1.048576e+006
OffsetPosition             :=0.000000e+000
MaxVelocityApp             :=5.000000e+001
MaxAccelerationApp         :=1.000000e+002
MaxDecelerationApp         :=1.000000e+002
MaxVelocityDrive           :=6.000000e+001
MaxPosition                :=1.048500e+003
MinPosition                :=-1.048514e+003
EnableMaxPosition          :=TRUE
EnableMinPosition          :=TRUE
MinUserPosition            :="InitS7PN1_MinUserPos"
MaxUserPosition            :="InitS7PN1_MaxUserPos"
Valid                      :="InitS7PN1_Valid"
Error                      :="InitS7PN1_Error"
ErrorID                    :="InitS7PN1_ErrorID"
Config                     :="Axis01".Config
Axis                       :="Axis01".Axis
```

Connecting the AxisControl

FB 890 - VMC_AxisControlSigma_PN, DB 890 *Chap. 4.3.2 'FB 890 - VMC_AxisControlSigma_PN - control block axis control for Sigma-5/7 PROFINET' page 213*

This block processes the user commands and passes them appropriately processed on to the drive via PROFINET.

```
CALL  "VMC_AxisControlSigma_PN" , "DI_AxisControlSigmaPN01"
 AxisEnable        :="AxCtrl1_AxisEnable"
 AxisReset         :="AxCtrl1_AxisReset"
 HomeExecute       :="AxCtrl1_HomeExecute"
 HomePosition      :="AxCtrl1_HomePosition"
 StopExecute       :="AxCtrl1_StopExecute"
 MvVelocityExecute:="AxCtrl1_MvVelExecute"
 MvRelativeExecute:="AxCtrl1_MvRelExecute"
 MvAbsoluteExecute:="AxCtrl1_MvAbsExecute"
 PositionDistance  :="AxCtrl1_PositionDistance"
 Direction         :="AxCtrl1_Direction"
```

```
Velocity          :="AxCtrl1_Velocity"
Acceleration      :="AxCtrl1_Acceleration"
Deceleration      :="AxCtrl1_Deceleration"
JogPositive       :="AxCtrl1_JogPositive"
JogNegative       :="AxCtrl1_JogNegative"
JogVelocity       :="AxCtrl1_JogVelocity"
JogAcceleration   :="AxCtrl1_JogAcceleration"
JogDeceleration   :="AxCtrl1_JogDeceleration"
AxisReady         :="AxCtrl1_AxisReady"
AxisEnabled       :="AxCtrl1_AxisEnabled"
AxisError         :="AxCtrl1_AxisError"
AxisErrorID       :="AxCtrl1_AxisErrorID"
DriveWarning      :="AxCtrl1_DriveWarning"
DriveError        :="AxCtrl1_DriveError"
DriveErrorID      :="AxCtrl1_DriveErrorID"
IsHomed           :="AxCtrl1_IsHomed"
ModeOfOperation   :="AxCtrl1_ModeOfOperation"
PLCopenState      :="AxCtrl1_PLCopenState"
ActualPosition    :="AxCtrl1_ActualPosition"
ActualVelocity    :="AxCtrl1_ActualVelocity"
CmdDone           :="AxCtrl1_CmdDone"
CmdBusy           :="AxCtrl1_CmdBusy"
CmdAborted        :="AxCtrl1_CmdAborted"
CmdError          :="AxCtrl1_CmdError"
CmdErrorID        :="AxCtrl1_CmdErrorID"
DirectionPositive:="AxCtrl1_DirectionPos"
DirectionNegative:="AxCtrl1_DirectionNeg"
SWLimitMinActive :="AxCtrl1_SWLimitMinActive"
SWLimitMaxActive :="AxCtrl1_SWLimitMaxActive"
HWLimitMinActive :="AxCtrl1_HWLimitMinActive"
HWLimitMaxActive :="AxCtrl1_HWLimitMaxActive"
Axis              :="Axis01".Axis
```

For complex motion tasks, you can use the PLCopen blocks. Please specify the reference to the corresponding axis data at Axis in the axis DB.

Your project now includes the following blocks:

- OB 1 - Main
- OB 57 - DP Manufacturer Alarm
- OB 82 - I/O_FLT1
- OB 86 - Rack_FLT
- FB 890 - VMC_AxisControlSigma_PN with instance DB
- FB 891 - VMC_InitSigma_PN with instance DB
- UDT 860 - MC_Axis_REF
- UDT 890 - VMC_ConfigSigmaPN_REF

Sequence of operations

1. Select *'Project ➔ Compile all'* and transfer the project into your 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. To do this, call the *Init* block FB 891 - VMC_InitSigma_PN with *Enable* = TRUE.
   ⇨ The output *Valid* returns TRUE. In the event of a fault, you can determine the error by evaluating the *ErrorID*.

   You have to call the *Init* block again if you load a new axis DB or you have changed parameters on the *Init* block.

   > *Do not continue until the Init block does not report any errors!*

3. Program your application with the FB 890 - VMC_AxisControlSigma_PN or with the PLCopen blocks.

4.2.4 Usage in Siemens SIMATIC Manager

4.2.4.1 Hardware configuration System MICRO respectively SLIO

Precondition

Overview

- Please use for configuration the Siemens SIMATIC Manager V5.5 SP2 and up.
- The configuration of the VIPA System MICRO respectively SLIO CPU happens in the Siemens SIMATIC Manager by means of a virtual PROFINET IO device.
  The PROFINET IO device is to be installed in the hardware catalog by means of a GSDML.
- For the PROFINET drive can be configured in the Siemens SIMATIC Manager, the corresponding GSDML file must be installed.

Install GSDML file for System MICRO respectively SLIO

The installation of the PROFINET IO device 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 System MICRO or SLIO 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'*.

   From YASKAWA there are the following PROFINET IO devices:

   - System MICRO: *'VIPA Micro PLC'*
   - System SLIO: *'VIPA System SLIO'*

Install GSDML file for *Sigma-7* PROFINET drive

The GSDML file for the *Sigma-7* PROFINET drive can be found at www.yaskawa.eu.com under *'Service ➔ Drives & Motion Software'*.

Please use the following GSDML:

- GSDML-V2.33-Yaskawa-SGD7S-xxxAC0xxxx-20170914.xml

The installation happens with the following proceeding:

1. Download the according GSDML file for your drive.
2. Extract the file into your working directory.
3. Start the Siemens hardware configurator.
4. Close all the projects.
5. Select *'Options ➔ Install new GSD file'*.
6. Navigate to your working directory and install the according GSDML file.
   ⇨ After the installation the PROFINET IO device for the *Sigma-7* drive at *'PROFINET IO ➔ Additional field devices ➔ Drives ➔ YASKAWA Drives'*.

Add CPU in the project

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

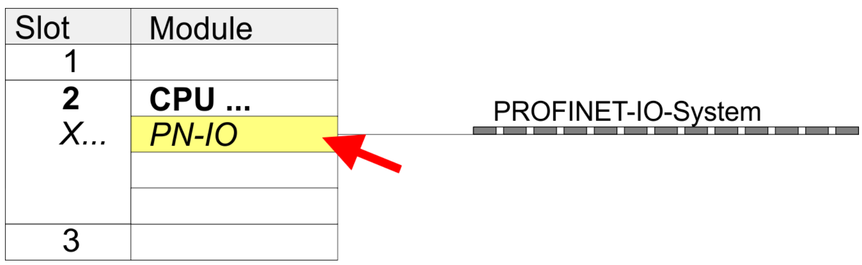
1. Start the Siemens hardware configurator with a new project.
2. Insert a profile rail from the hardware catalog.
3. Depending on the VIPA CPU used, place the following CPU from Siemens at *'Slot'* number 2:

VIPA CPU	to be configured as SIMATIC S7-300> ...
M13-CCF0000 from V2.4.12	CPU 314C-2 PN/DP (6ES7 314-6EH04-0AB0 V3.3)
013-CCF0R00 from V2.4.12	CPU 314C-2 PN/DP (6ES7 314-6EH04-0AB0 V3.3)
014-CEF0R01 from V2.4.12	CPU 315-2 PN/DP (6ES7 315-2EH14-0AB0 V3.2)
015-CEFNR00 from V2.4.16	CPU 315-2 PN/DP (6ES7 315-2EH14-0AB0 V3.2)
015-CEFPR01 from V2.4.12	CPU 315-2 PN/DP (6ES7 315-2EH14-0AB0 V3.2)
017-CEFPR00 from V2.4.12	CPU 317-2PN/DP (6ES7 317-2EK14-0AB0 V3.2)

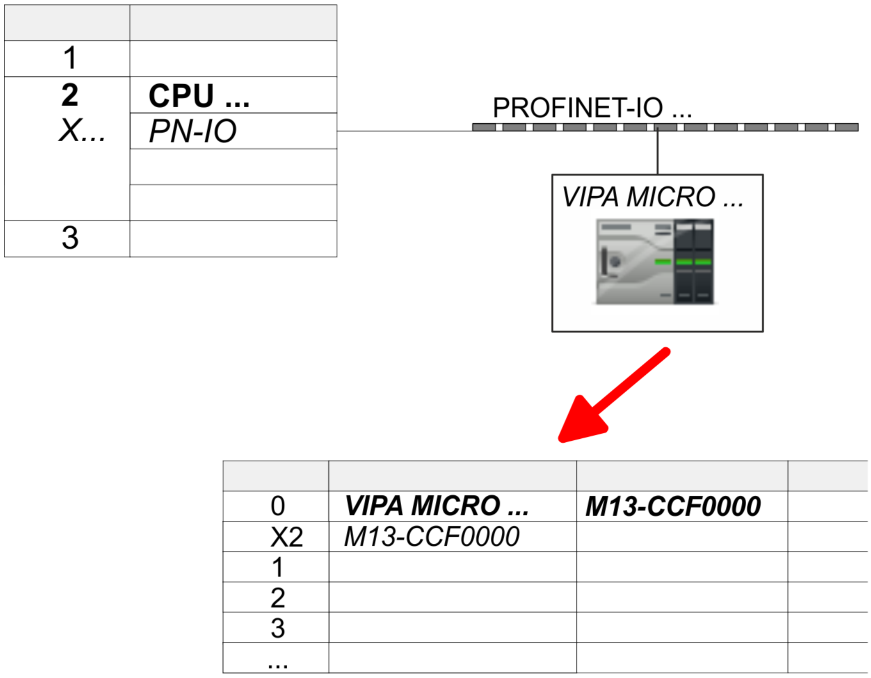
⇨ The CPU is inserted at the profile rail, such as the CPU 314C-2 PN/DP for System MICRO.

Connection CPU as PROFINET IO device

1. Click at the sub module *'PN-IO'* of the CPU.
2. Select *'Context menu ➔ Insert PROFINET IO System'*.



3. Create with [New] a new sub net and assign valid address data
4. Click at the sub module *'PN-IO'* of the CPU and open with *'Context menu ➔ Properties'* the properties dialog.
5. Enter at *'General'* a *'Device name'*. The device name must be unique at the Ethernet subnet.



6. Navigate in the hardware catalog to the directory *'PROFINET IO → Additional field devices → I/O'* and connect e.g. for the System MICRO the IO device *'M13-CCF0000'* to your PROFINET system.

   From YASKAWA there are the following PROFINET IO devices:

   - System MICRO: *'VIPA Micro PLC'*
   - System SLIO: *'VIPA System SLIO'*

   ⇨ In the Device overview of the PROFINET IO device *'VIPA MICRO PLC'* the CPU is already placed at slot 0.

Configuration of Ethernet PG/OP channel

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!

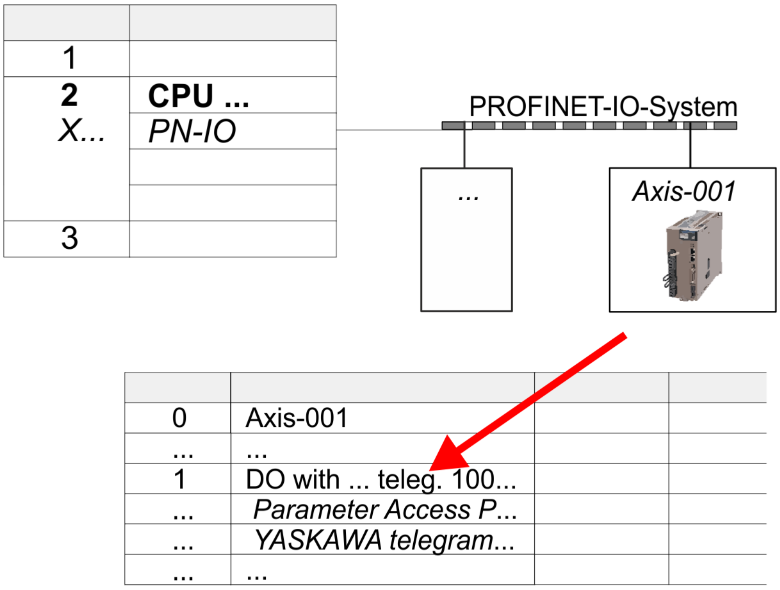
Insert and configure *Sigma-7* PROFINET drive

During configuration a *Sigma-7* PROFINET IO device must be configured for each axis.

1. Select your *Sigma-7* PROFINET drive *'SGD7S-xxxAC0xxxx'* 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. Click at the *Sigma-7* IO device and open with *'Context menu → Properties'* the properties dialog.
3. Assign a suitable *'Device name'* such as Axis-001.

4. Confirm your input with [OK].



5. In the hardware catalog, expand the *Sigma-7* PROFINET drive *'SGD7S-xxxAC0xxxx'* to show its components and drag&drop the component *'DO with YASKAWA telegr. 100...'* to slot 1 of the *Sigma-7* PROFINET drive.

   ⇨ Telegram 100 is inserted with the corresponding subgroups.

The connection between the axes in the hardware configuration and your user program is made by specifying the following module properties in the call parameters of FB 891 - VMC InitSigma_PN:

- *Module properties 'Parameter Access Point': Diagnostic address of slot 1 of the slot overview*
  - *FB 891 - VMC InitSigma_PN: ParaAccessPointAddress:*
    Setting of the diagnostic address of slot 1 of the slot overview.
- *Module properties 'YASKAWA Telegram PZD...':*
  Respective start address of the input/output address range.
  - *FB 891 - VMC InitSigma_PN: 'InputsStartAddress':*
    Setting of the start address of the input address range.
  - *FB 891 - VMC InitSigma_PN: 'OutputsStartAddress':*
    Setting of the start address of the output address range.
  - *FB 891 - VMC InitSigma_PN: 'LogicalAddress':*
    Setting of the of the smaller value of the start addresses of the input/output address range.

- User program ⇨ *195*
- FB 891 - VMC InitSigma_PN ⇨ *217*

Example hardware configuration

Slot	Component	...	I-Adr.	O-Adr.	Diagnostic address
0	SGD7S-xxxAC0xxxx				2035
X1	PN-IO				2034
X1 P1	Port 1				2033
X1 P2	Port 2				2032
1	DO with YASKAWA telegr.100, PZD-16/14				2044
1.1	Parameter Access Point				2044
1.2	YASKAWA telegram, PZD-16/14		28-55	32-63	

4.2.4.2 Hardware configuration System 300S

Precondition

- Please use for configuration the Siemens SIMATIC Manager V5.5 SP2 and up.
- For the PROFINET drive can be configured in the Siemens SIMATIC Manager, the corresponding GSDML file must be installed.
- The blocks can be used with the following CPUs:
  - System 300S CPU 315-4PN43
  - System 300S CPU 315-4PN23
  - System 300S CPU 317-4PN23
- The configuration of the System 300S PROFINET CPU takes place in the Siemens SIMATIC Manager as a corresponding Siemens CPU.
  - The CPUs 315-4PNxx are to be configured as Siemens CPU 315-2 PN/DP (6ES7 315-2EH14-0AB0 V3.2).
  - The CPU 317-4PN23 is to be configured as Siemens CPU 317-2 PN/DP (6ES7 317-2EK14-0AB0 V3.2).

Install GSDML file for *Sigma-7* PROFINET drive

The GSDML file for the *Sigma-7* PROFINET drive can be found at www.yaskawa.eu.com under *'Service ➔ Drives & Motion Software'*.

Please use the following GSDML:

- GSDML-V2.33-Yaskawa-SGD7S-xxxAC0xxxx-20170914.xml

The installation happens with the following proceeding:

1. Download the according GSDML file for your drive.
2. Extract the file into your working directory.
3. Start the Siemens hardware configurator.
4. Close all the projects.
5. Select *'Options ➔ Install new GSD file'*.
6. Navigate to your working directory and install the according GSDML file.

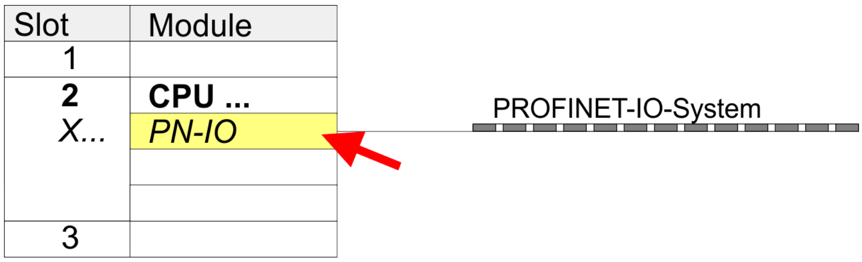
   ⇨ After the installation the PROFINET IO device for the *Sigma-7* drive at *'PROFINET IO ➔ Additional field devices ➔ Drives ➔ YASKAWA Drives'*.

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. Select *'Context menu ➔ Insert PROFINET IO System'*.



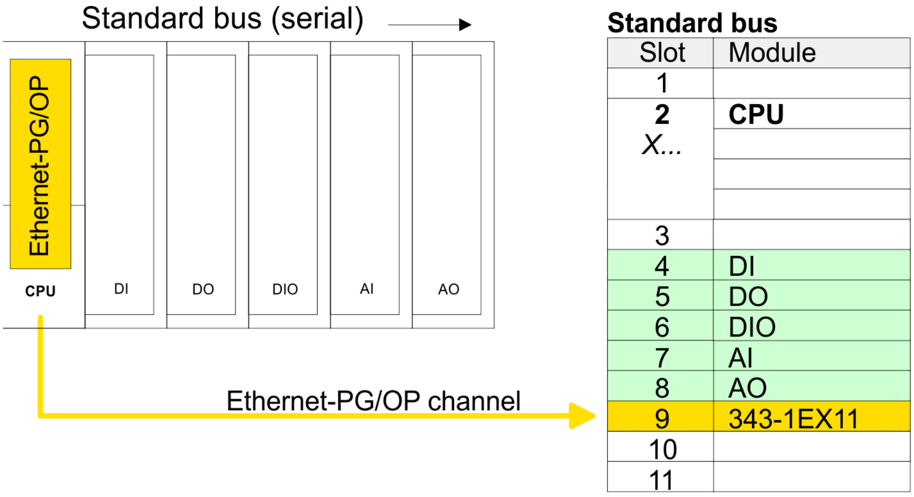
6. Create with [New] a new sub net.
7. Click at the sub module *'PN-IO'* of the CPU and open with *'Context menu ➔ Properties'* the properties dialog.
8. Enter at *'General'* a *'Device name'*. The device name must be unique at the Ethernet subnet.

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 <u>always</u> 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.

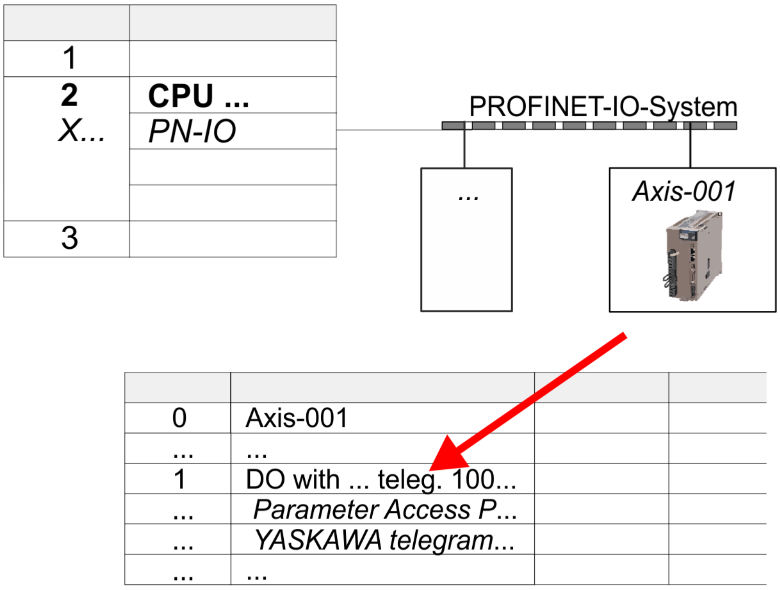


More information about the initialization and the usage of the Ethernet PG/OP channel can be found in the manual of the CPU.

Insert and configure *Sigma-7* PROFINET drive

During configuration a *Sigma-7* PROFINET IO device must be configured for each axis.

1. Select your *Sigma-7* PROFINET drive *'SGD7S-xxxAC0xxxx'* 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. Click at the *Sigma-7* IO device and open with *'Context menu ➔ Properties'* the properties dialog.
3. Assign a suitable *'Device name'* such as Axis-001.
4. Confirm your input with [OK].



5. In the hardware catalog, expand the *Sigma-7* PROFINET drive *'SGD7S-xxxAC0xxxx'* to show its components and drag&drop the component *'DO with YASKAWA telegr. 100...'* to slot 1 of the *Sigma-7* PROFINET drive.

   ⇨ Telegram 100 is inserted with the corresponding subgroups.

The connection between the axes in the hardware configuration and your user program is made by specifying the following module properties in the call parameters of FB 891 - VMC InitSigma_PN:

- *Module properties 'Parameter Access Point': Diagnostic address of slot 1 of the slot overview*
  - *FB 891 - VMC InitSigma_PN: ParaAccessPointAddress:*
    Setting of the diagnostic address of slot 1 of the slot overview.
- *Module properties 'YASKAWA Telegram PZD...':*
  Respective start address of the input/output address range.
  - *FB 891 - VMC InitSigma_PN: 'InputsStartAddress':*
    Setting of the start address of the input address range.
  - *FB 891 - VMC InitSigma_PN: 'OutputsStartAddress':*
    Setting of the start address of the output address range.
  - *FB 891 - VMC InitSigma_PN: 'LogicalAddress':*
    Setting of the of the smaller value of the start addresses of the input/output address range.

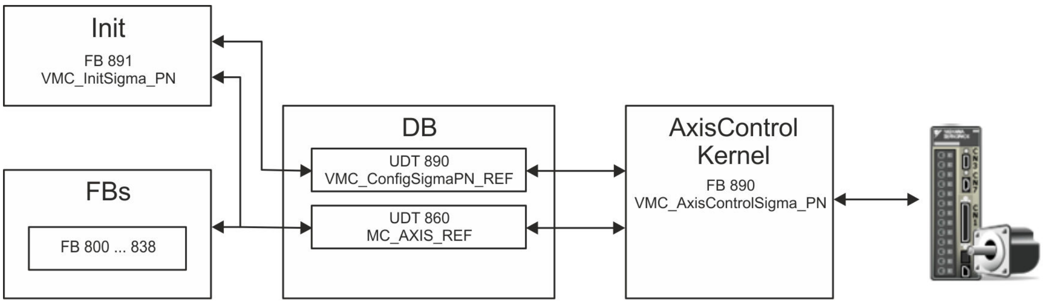
- User program ⇨ *195*
- FB 891 - VMC InitSigma_PN ⇨ *217*

Example hardware configuration

Slot	Component	...	I-Adr.	O-Adr.	Diagnostic address
0	SGD7S-xxxAC0xxxx				2035
X1	PN-IO				2034
X1 P1	Port 1				2033
X1 P2	Port 2				2032
1	DO with YASKAWA telegr.100, PZD-16/14				2044
1.1	Parameter Access Point				2044
1.2	YASKAWA telegram, PZD-16/14		28-55	32-63	

4.2.4.3 User program

4.2.4.3.1 Program structure



- DB
  A data block (axis DB) for configuration and status data must be created for each axis of a drive. The data block consists of the following data structures:
  – UDT 890 - *VMC_Config*SigmaPN_REF
    The data structure describes the structure of the configuration of the drive.
    Specific data structure for *Sigma-5/7* PROFINET.
  – UDT 860 - *MC_AXIS_REF*
    The data structure describes the structure of the parameters and status information of drives.
    General data structure for all drives and bus systems.
- FB 891 - *VMC_Init*Sigma_PN
  – The *Init*t block is used to configure an axis.
  – Specific block for *Sigma-5/7* PROFINET.
  – The configuration data for the initialization must be stored in the *axis DB*.

- FB 890 - *VMC_AxisControl*Sigma_PN
  – Specific block for *Sigma-5/7* PROFINET.
  – This block is a combination of *Kernel* and *AxisControl* and communicates with the drive via PROFINET, processes the user requests and returns status messages.
  – This block supports simple motion commands and returns all relevant status messages.
  – The exchange of the data takes place by means of the *axis DB*.
  – For motion control and status query, via the instance data of the block you can link a visualization.
  – In addition to the FB 890 - *VMC_AxisControl*Sigma_PN, *PLCopen* blocks can be used.
- FB 800 ... FB 838 - *PLCopen*
  – The PLCopen blocks are used to program motion sequences and status queries.
  – General blocks for all drives and bus systems.

4.2.4.3.2 Programming

Include library

1. Go to the service area of www.vipa.com.
2. Download the *Simple Motion Control* library from the download area at *'VIPA Lib'*.
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].

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.

Copy blocks into project

▶ Open the library after unzipping and drag and drop the following blocks into *'Blocks'* of your project:
- *Sigma* PROFINET:
  – UDT 890 - VMC_ConfigSigmaPN_REF ↪ *Chap. 4.3.1 'UDT 890 - VMC_ConfigSigmaPN_REF - Sigma-5/7 PROFINET Data structure axis configuration' page 213*
  – FB 890 - VMC_AxisControlSigma_PN ↪ *Chap. 4.3.2 'FB 890 - VMC_AxisControlSigma_PN - control block axis control for Sigma-5/7 PROFINET' page 213*
  – FB 891 - VMC_InitSigma_PN ↪ *Chap. 4.3.3 'FB 891 - VMC_InitSigma_PN - Sigma-5/7 PROFINET initialization' page 217*
- Axis control
  – UDT 860 - MC_AXIS_REF ↪ *Chap. 9.2.1 'UDT 860 - MC_AXIS_REF - Data structure axis data' page 363*
  – FB 860 - VMC_AxisControl ↪ *Chap. 9.2.2 'FB 860 - VMC_AxisControl - Control block axis control' page 363*

Create axis DB

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

   Specify the following parameters:

   - Name and type
     – The DB no. as *'Name'* can freely be chosen, such as DB10.
     – Set *'Shared DB'* as the *'Type'*.
   - Symbolic name
     – Specify "Axis01".

   Confirm your input with [OK].

   ⇨ The block is created.

2. Open DB10 "Axis01" by double-click.

   - In "Axis01", create the variable "Config" of type UDT 890. These are specific axis configuration data.
   - In "Axis01", create the variable "Axis" of type UDT 860. During operation, all operating data of the axis are stored here.

   DB10

Address	Name	Type	...
		Struct	
...	Config	"VMC_ConfigSigmaPN_REF"	
...	Axis	"MC_AXIS_REF	
...		END_STRUCT	

OB 1 - configuration of the axes

Open OB 1 and program the following FB calls with associated DBs:

FB 891 - VMC_InitSigma_PN, DB 891

The connection between the axes in the hardware configuration and your user program is made by specifying the following module properties in the call parameters of FB 891 - VMC InitSigma_PN:

– *Module properties 'Parameter Access Point': Diagnostic address of slot 1 of the slot overview*
  – *FB 891 - VMC InitSigma_PN: ParaAccessPointAddress: Setting of the diagnostic address of slot 1 of the slot overview.*
– *Module properties 'YASKAWA Telegram PZD...': Respective start address of the input/output address range.*
  – *FB 891 - VMC InitSigma_PN: 'InputsStartAddress': Setting of the start address of the input address range.*
  – *FB 891 - VMC InitSigma_PN: 'OutputsStartAddress': Setting of the start address of the output address range.*
  – *FB 891 - VMC InitSigma_PN: 'LogicalAddress': Setting of the of the smaller value of the start addresses of the input/output address range.*

- Hardware configuration ↳ *185*
- FB 891 - VMC InitSigma_PN ↳ *217*

Example hardware configuration

Slot	Component	...	I-Adr.	O-Adr.	Diagnostic address
0	SGD7S-xxxAC0xxxx				2035
X1	PN-IO				2034
X1 P1	Port 1				2033
X1 P2	Port 2				2032
1	DO with YASKAWA telegr.100, PZD-16/14				2044
1.1	Parameter Access Point				2044
1.2	YASKAWA telegram, PZD-16/14		28-55	32-63	

Example call

```
CALL  "VMC_InitSigma_PN" , "VMC_InitSigma_PN_1"
Enable                     :="InitS7PN1_Enable"
LogicalAddress             :=28 //HW-Konfig: Smallest IO addr.
ParaAccessPointAdress      :=2044 //HW-Konfig: Diag addr.
InputsStartAddress         :=28 //HW-Konfig: Telegr.100 start I addr.
OutputsStartAddress        :=32 //HW-Konfig: Telegr. 100 start O addr.
EncoderType                :=1
EncoderResolutionBits      :=20
FactorPosition             :=1.048576e+006
FactorVelocity             :=1.048576e+006
FactorAcceleration         :=1.048576e+006
OffsetPosition             :=0.000000e+000
MaxVelocityApp             :=5.000000e+001
MaxAccelerationApp         :=1.000000e+002
MaxDecelerationApp         :=1.000000e+002
MaxVelocityDrive           :=6.000000e+001
MaxPosition                :=1.048500e+003
MinPosition                :=-1.048514e+003
EnableMaxPosition          :=TRUE
EnableMinPosition          :=TRUE
MinUserPosition            :="InitS7PN1_MinUserPos"
MaxUserPosition            :="InitS7PN1_MaxUserPos"
Valid                      :="InitS7PN1_Valid"
Error                      :="InitS7PN1_Error"
ErrorID                    :="InitS7PN1_ErrorID"
Config                     :="Axis01".Config
Axis                       :="Axis01".Axis
```

Connecting the AxisControl

FB 890 - VMC_AxisControlSigma_PN, DB 890 *Chap. 4.3.2 'FB 890 - VMC_AxisControlSigma_PN - control block axis control for Sigma-5/7 PROFINET' page 213*

This block processes the user commands and passes them appropriately processed on to the drive via PROFINET.

```
CALL  "VMC_AxisControlSigma_PN" , "DI_AxisControlSigmaPN01"
 AxisEnable        :="AxCtrl1_AxisEnable"
 AxisReset         :="AxCtrl1_AxisReset"
 HomeExecute       :="AxCtrl1_HomeExecute"
 HomePosition      :="AxCtrl1_HomePosition"
 StopExecute       :="AxCtrl1_StopExecute"
 MvVelocityExecute:="AxCtrl1_MvVelExecute"
 MvRelativeExecute:="AxCtrl1_MvRelExecute"
 MvAbsoluteExecute:="AxCtrl1_MvAbsExecute"
 PositionDistance  :="AxCtrl1_PositionDistance"
 Direction         :="AxCtrl1_Direction"
```

```
Velocity          :="AxCtrl1_Velocity"
Acceleration      :="AxCtrl1_Acceleration"
Deceleration      :="AxCtrl1_Deceleration"
JogPositive       :="AxCtrl1_JogPositive"
JogNegative       :="AxCtrl1_JogNegative"
JogVelocity       :="AxCtrl1_JogVelocity"
JogAcceleration   :="AxCtrl1_JogAcceleration"
JogDeceleration   :="AxCtrl1_JogDeceleration"
AxisReady         :="AxCtrl1_AxisReady"
AxisEnabled       :="AxCtrl1_AxisEnabled"
AxisError         :="AxCtrl1_AxisError"
AxisErrorID       :="AxCtrl1_AxisErrorID"
DriveWarning      :="AxCtrl1_DriveWarning"
DriveError        :="AxCtrl1_DriveError"
DriveErrorID      :="AxCtrl1_DriveErrorID"
IsHomed           :="AxCtrl1_IsHomed"
ModeOfOperation   :="AxCtrl1_ModeOfOperation"
PLCopenState      :="AxCtrl1_PLCopenState"
ActualPosition    :="AxCtrl1_ActualPosition"
ActualVelocity    :="AxCtrl1_ActualVelocity"
CmdDone           :="AxCtrl1_CmdDone"
CmdBusy           :="AxCtrl1_CmdBusy"
CmdAborted        :="AxCtrl1_CmdAborted"
CmdError          :="AxCtrl1_CmdError"
CmdErrorID        :="AxCtrl1_CmdErrorID"
DirectionPositive:="AxCtrl1_DirectionPos"
DirectionNegative:="AxCtrl1_DirectionNeg"
SWLimitMinActive :="AxCtrl1_SWLimitMinActive"
SWLimitMaxActive :="AxCtrl1_SWLimitMaxActive"
HWLimitMinActive :="AxCtrl1_HWLimitMinActive"
HWLimitMaxActive :="AxCtrl1_HWLimitMaxActive"
Axis              :="Axis01".Axis
```

For complex motion tasks, you can use the PLCopen blocks. Please specify the reference to the corresponding axis data at Axis in the axis DB.

Your project now includes the following blocks:

- OB 1 - Main
- OB 57 - DP Manufacturer Alarm
- OB 82 - I/O_FLT1
- OB 86 - Rack_FLT
- FB 890 - VMC_AxisControlSigma_PN with instance DB
- FB 891 - VMC_InitSigma_PN with instance DB
- UDT 860 - MC_Axis_REF
- UDT 890 - VMC_ConfigSigmaPN_REF

Sequence of operations

1. Select *'Project* ➔ *Compile all'* and transfer the project into your 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. To do this, call the *Init* block FB 891 - VMC_InitSigma_PN with *Enable* = TRUE.
   ⇨ The output *Valid* returns TRUE. In the event of a fault, you can determine the error by evaluating the *ErrorID*.

   You have to call the *Init* block again if you load a new axis DB or you have changed parameters on the *Init* block.

   > *Do not continue until the Init block does not report any errors!*

3. Program your application with the FB 890 - VMC_AxisControlSigma_PN or with the PLCopen blocks.

4.2.5 Usage in Siemens TIA-Portal

4.2.5.1 Hardware configuration System MICRO respectively SLIO

Precondition

Overview

- Please use the Siemens TIA Portal from V.14 for the configuration.
- The configuration of the VIPA System MICRO respectively SLIO happens in the Siemens TIA Portal by means of a virtual PROFINET IO device.
  The PROFINET IO device is to be installed in the hardware catalog by means of a GSDML.
- For the PROFINET drive can be configured in the Siemens TIA Portal, the corresponding GSDML file must be installed.

Install GSDML file for System MICRO respectively SLIO

The installation of the PROFINET IO device 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 System MICRO or SLIO 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 ➔ I/O ➔ VIPA ...'*.

   From YASKAWA there are the following PROFINET IO devices:
   - System MICRO: *'VIPA Micro PLC'*
   - System SLIO: *'VIPA System SLIO'*

Thus, the VIPA components can be shown, you have to deactivate the ' Filter' of the hardware catalog.

Install GSDML file for *Sigma-7* PROFINET drive

The GSDML file for the *Sigma-7* PROFINET drive can be found at www.yaskawa.eu.com under *'Service ➔ Drives & Motion Software'*.

Please use the following GSDML:

- GSDML-V2.33-Yaskawa-SGD7S-xxxAC0xxxx-20170914.xml

The installation happens with the following proceeding:

1. Download the according GSDML file for your drive.
2. Extract the file into your working directory.
3. Start the Siemens TIA Portal.
4. Close all the projects.
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 PROFINET IO device for the *Sigma-7* drive can be found at *'Additional field devices ➔ PROFINET IO ➔ Drives ➔ Yaskawa ...'*.

Add CPU in the project

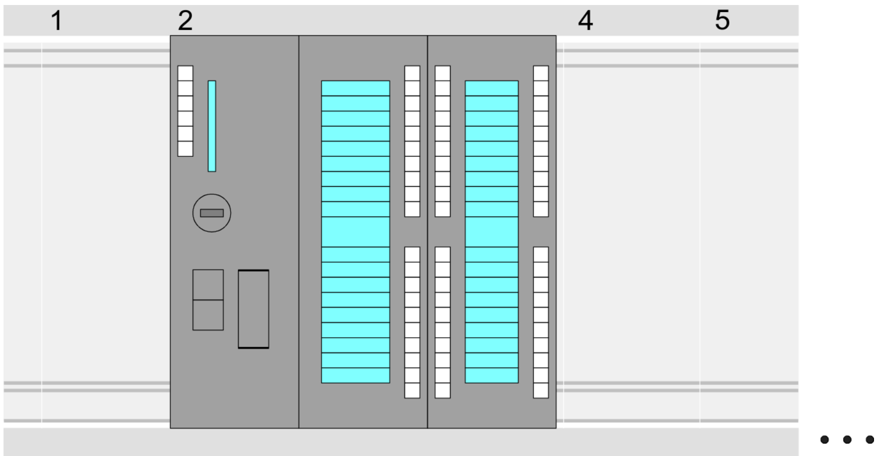
To be compatible with the Siemens SIMATIC TIA Portal the following steps should be executed:

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. Depending on the VIPA CPU used, select the following CPU from Siemens:

VIPA CPU	to configure as SIMATIC S7-300 > ...
M13-CCF0000 from V2.4.12	CPU 314C-2 PN/DP (6ES7 314-6EH04-0AB0 V3.3)
013-CCF0R00 from V2.4.12	CPU 314C-2 PN/DP (6ES7 314-6EH04-0AB0 V3.3)
014-CEF0R01 from V2.4.12	CPU 315-2 PN/DP (6ES7 315-2EH14-0AB0 V3.2)
015-CEFNR00 from V2.4.16	CPU 315-2 PN/DP (6ES7 315-2EH14-0AB0 V3.2)
015-CEFPR01 from V2.4.12	CPU 315-2 PN/DP (6ES7 315-2EH14-0AB0 V3.2)
017-CEFPR00 from V2.4.12	CPU 317-2PN/DP (6ES7 317-2EK14-0AB0 V3.2)

⇨ The CPU is inserted with a profile rail, such as the CPU 314C-2 PN/DP for System MICRO.



Device overview:

Module	...	Slot	...	Type	...
PLC...		2		CPU 314C-2PN/DP	
MPI interface...		2 X1		MPI/DP interface	
PROFINET interface...		2 X2		PROFINET interface	
DI24/DO16...		2 5		DI24/DO16	
AI5/AO2...		2 6		AI5/AO2	
Count...		2 7		Count	
...					

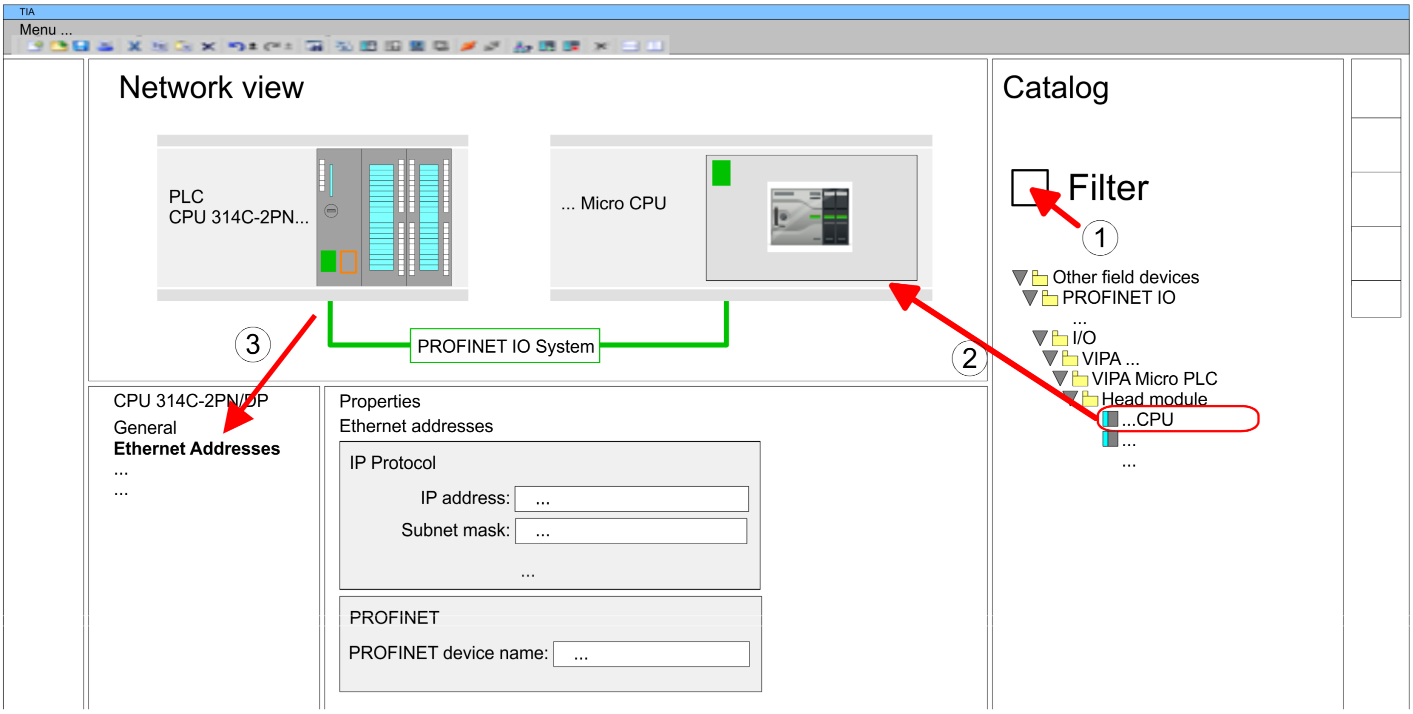
Connection CPU as PROFINET IO device

1. Switch in the *Project area* to *'Network view'*.
2. Navigate in the hardware catalog to *'Other field devices → PROFINET IO → I/O → VIPA ...'* 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.

   From YASKAWA there are the following PROFINET IO devices:

   - System MICRO: *'VIPA Micro PLC'*
   - System SLIO: *'VIPA System SLIO'*
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.



5. Select in the *Network view* the IO device such as *'VIPA MICRO PLC'* and switch to the *Device overview*.

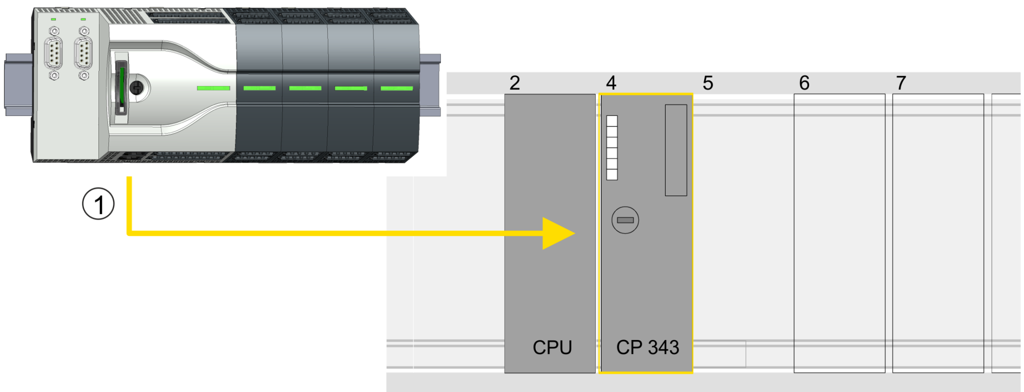
⇨ In the *Device overview* of the PROFINET IO device *'VIPA MICRO PLC'* the CPU is already placed at slot 0. From slot 1 you can place your System MICRO respectively SLIO modules.

Configuration of Ethernet PG/OP channel

So that you may online access the according Ethernet interface, you have to assign IP address parameters by means of the "initialization". Please consider to use the same IP address data in your project for the 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.*

1. As Ethernet PG/OP channel place at slot 4 of the Siemens system the Siemens 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 from the initialization.
3. Assign the CP to a *'Subnet'*. The IP address data are not accepted without assignment!
4. Transfer your project to your CPU.
   ⇨ The IP address data are stored in your current project. In the following this is shown exemplary on the System MICRO.



(1) Ethernet PG/OP channel

Device overview

Module	...	Slot	...	Type	...
PLC ...		2		CPU 314C-2PN/DP	
MPI/DP interface		2 X1		MPI/DP interface	
PROFINET interface		2 X2		PROFINET interface	
...		...		...	
CP 343-1		4		CP 343-1	
...		...		...	

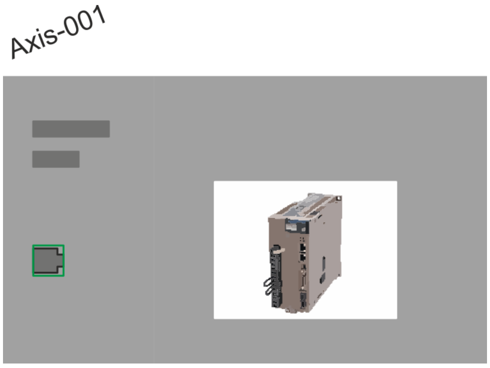
Insert and configure *Sigma-7* PROFINET drive

During configuration a *Sigma-7* PROFINET IO device must be configured for each axis.

1. Select your *Sigma-7* PROFINET drive *'SGD7S-xxxAC0xxxx'* from the hardware catalog at *'Additional field devices ➔ PROFINET IO ➔ Drives ➔ Yaskawa ...'* 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. Click at the *Sigma-7* IO device and open with *'Context menu ➔ Device configuration'* the *'Device overview'*.

3. Assign a suitable *'Device name'* such as Axis-001.



4. **Device overview**

Module	...	Slot	...	Type	...
Axis-001		0		SGD7S-xxxAC0xxxx	
PN-IO		0 X1		SGD7S-xxxAC0xxxx	
DO w/ Yaskawa telegr.100,PZD...		1		DO w/ Yaskawa telegr.100,PZD-16/14	
Parameter Access Point		1 1		Parameter Access Point	
Yaskawa telegram, PZD-16/14		1 2		Yaskawa telegram, PZD-16/14	
...		...		...	

In the hardware catalog, expand the *Sigma-7* PROFINET drive *'SGD7S-xxxAC0xxxx'* to show its components and drag the component *'DO w/ YASKAWA telegr. 100...'* to *'Slot 1'* of the *Sigma-7* PROFINET drive.

⇨ Telegram 100 is inserted with the corresponding subgroups.

The connection between the axes in the hardware configuration and your user program is made by specifying the following module properties in the call parameters of FB 891 - VMC InitSigma_PN:

- *Module properties 'Parameter Access Point': Diagnostic address of slot 1 of the slot overview*
  - *FB 891 - VMC InitSigma_PN: ParaAccessPointAddress:*
    Setting of the diagnostic address of slot 1 of the slot overview.
- *Module properties 'YASKAWA Telegram PZD...':*
  Respective start address of the input/output address range.
  - *FB 891 - VMC InitSigma_PN: 'InputsStartAddress':*
    Setting of the start address of the input address range.
  - *FB 891 - VMC InitSigma_PN: 'OutputsStartAddress':*
    Setting of the start address of the output address range.
  - *FB 891 - VMC InitSigma_PN: 'LogicalAddress':*
    Setting of the of the smaller value of the start addresses of the input/output address range.

- User program ⇨ *209*
- FB 891 - VMC InitSigma_PN ⇨ *217*

Example hardware configuration

Slot	Component	...	I-Adr.	O-Adr.	Diagnostic address
0	SGD7S-xxxAC0xxxx				2035
X1	PN-IO				2034
X1 P1	Port 1				2033
X1 P2	Port 2				2032
1	DO with YASKAWA telegr.100, PZD-16/14				2044
1.1	Parameter Access Point				2044
1.2	YASKAWA telegram, PZD-16/14		28-55	32-63	

4.2.5.2 Hardware configuration System 300S

Precondition

Overview

- Please use the Siemens TIA Portal from V.14 for the configuration.
- For the PROFINET drive can be configured in the Siemens TIA Portal, the corresponding GSDML file must be installed.
- The blocks can be used with the following CPUs:
  - System 300S CPU 315-4PN43
  - System 300S CPU 315-4PN23
  - System 300S CPU 317-4PN23
- The configuration of the System 300S PROFINET CPU takes place in the Siemens TIA Portal as a corresponding Siemens CPU.
  - The CPUs 315-4PNxx are to be configured as Siemens CPU 315-2 PN/DP (6ES7 315-2EH14-0AB0 V3.2).
  - The CPU 317-4PN23 is to be configured as Siemens CPU 317-2 PN/DP (6ES7 317-2EK14-0AB0 V3.2).

Install GSDML file for *Sigma-7* PROFINET drive

The GSDML file for the *Sigma-7* PROFINET drive can be found at www.yaskawa.eu.com under *'Service ➔ Drives & Motion Software'*.

Please use the following GSDML:

- GSDML-V2.33-Yaskawa-SGD7S-xxxAC0xxxx-20170914.xml

The installation happens with the following proceeding:

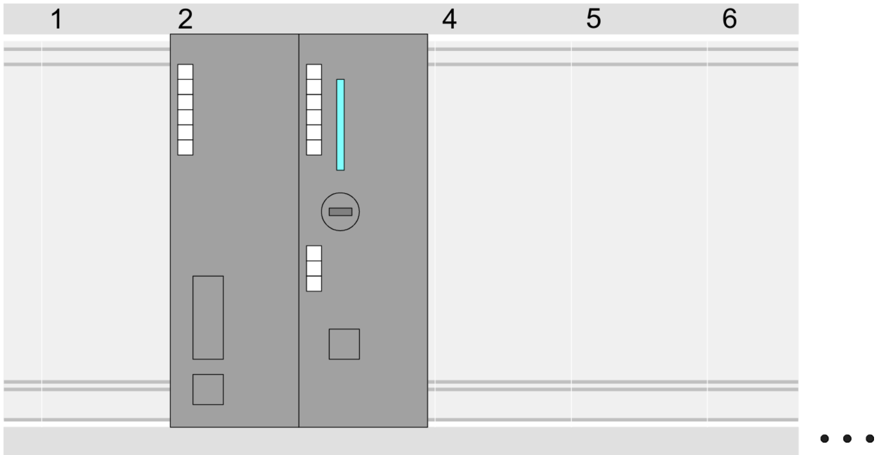
1. Download the according GSDML file for your drive.
2. Extract the file into your working directory.
3. Start the Siemens TIA Portal.
4. Close all the projects.
5. Select *'Options ➔ Install general station description file (GSD)'*.
6. Navigate to your working directory and install the according GSDML file.
   ⇨ After the installation the PROFINET IO device for the *Sigma-7* drive can be found at *'Additional field devices ➔ PROFINET IO ➔ Drives ➔ Yaskawa ...'*.

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. Depending on the VIPA CPU used, select the following CPU from Siemens:
   - The CPUs 315-4PNxx are to be configured as Siemens CPU 315-2 PN/DP (6ES7 315-2EH14-0AB0 V3.2).
   - The CPU 317-4PN23 is to be configured as Siemens CPU 317-2 PN/DP (6ES7 317-2EK14-0AB0 V3.2).

   ⇨ The CPU is inserted with a profile rail, such as the CPU 314C-2 PN/DP for VIPA CPU 315-4PN23.



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	
...		...		...	

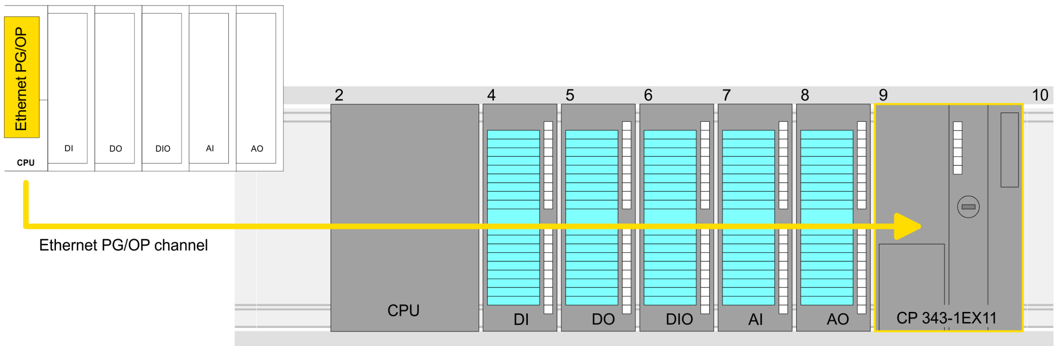
Configuration of Ethernet PG/OP channel

So that you may online access the according Ethernet interface, you have to assign IP address parameters by means of the "initialization". Please consider to use the same IP address data in your project for the 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.*

1. For the Ethernet PG/OP channel, always configure a Siemens CP 343-1 (6GK7 343-1EX11 0XE0) as the last module after the inserted System 300 modules.
2. 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.
3. Assign the CP to a *'Subnet'*. The IP address data are not accepted without assignment!
4. Transfer your project to your CPU.

   ⇨ The IP address data are stored in your current project. As an example, this is shown below on the CPU 315-4PN23.



Device overview

Module	...	Slot	...	Type	...
PLC...		2		CPU 315-2PN/DP	
...		...		...	
DI...		4		DI...	
DO...		5		DO...	
DIO...		6		DIO...	
AI...		7		AI...	
AO...		8		AO...	
CP 343-1		9		CP 343-1	

Insert and configure *Sigma-7* PROFINET drive

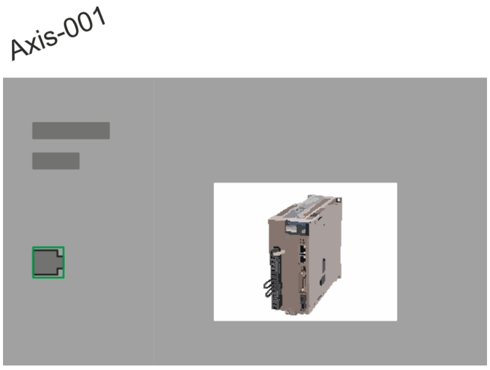
During configuration a *Sigma-7* PROFINET IO device must be configured for each axis.

1. Select your *Sigma-7* PROFINET drive *'SGD7S-xxxAC0xxxx'* from the hardware catalog at *'Additional field devices ➔ PROFINET IO ➔ Drives ➔ Yaskawa ...'* 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. Click at the *Sigma-7* IO device and open with *'Context menu ➔ Device configuration'* the *'Device overview'*.

3. Assign a suitable *'Device name'* such as Axis-001.



4. **Device overview**

Module	...	Slot	...	Type	...
Axis-001		0		SGD7S-xxxAC0xxxx	
PN-IO		0 X1		SGD7S-xxxAC0xxxx	
DO w/ Yaskawa telegr.100,PZD...		1		DO w/ Yaskawa telegr.100,PZD-16/14	
Parameter Access Point		1 1		Parameter Access Point	
Yaskawa telegram, PZD-16/14		1 2		Yaskawa telegram, PZD-16/14	
...		...		...	

In the hardware catalog, expand the *Sigma-7* PROFINET drive *'SGD7S-xxxAC0xxxx'* to show its components and drag the component *'DO w/ YASKAWA telegr. 100...'* to *'Slot 1'* of the *Sigma-7* PROFINET drive.

⇨ Telegram 100 is inserted with the corresponding subgroups.

The connection between the axes in the hardware configuration and your user program is made by specifying the following module properties in the call parameters of FB 891 - VMC InitSigma_PN:

- *Module properties 'Parameter Access Point': Diagnostic address of slot 1 of the slot overview*
  - *FB 891 - VMC InitSigma_PN: ParaAccessPointAddress:*
    Setting of the diagnostic address of slot 1 of the slot overview.
- *Module properties 'YASKAWA Telegram PZD...':*
  Respective start address of the input/output address range.
  - *FB 891 - VMC InitSigma_PN: 'InputsStartAddress':*
    Setting of the start address of the input address range.
  - *FB 891 - VMC InitSigma_PN: 'OutputsStartAddress':*
    Setting of the start address of the output address range.
  - *FB 891 - VMC InitSigma_PN: 'LogicalAddress':*
    Setting of the of the smaller value of the start addresses of the input/output address range.

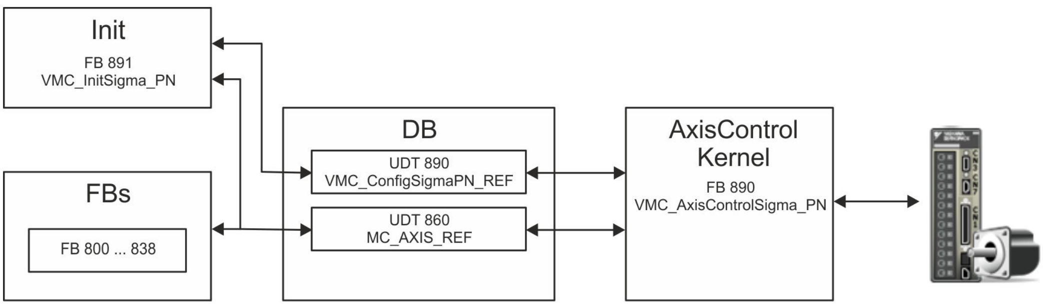
- User program ⇨ *209*
- FB 891 - VMC InitSigma_PN ⇨ *217*

Example hardware configuration

Slot	Component	...	I-Adr.	O-Adr.	Diagnostic address
0	SGD7S-xxxAC0xxxx				2035
X1	PN-IO				2034
X1 P1	Port 1				2033
X1 P2	Port 2				2032
1	DO with YASKAWA telegr.100, PZD-16/14				2044
1.1	Parameter Access Point				2044
1.2	YASKAWA telegram, PZD-16/14		28-55	32-63	

4.2.5.3 User program

4.2.5.3.1 Program structure



- DB
  A data block (axis DB) for configuration and status data must be created for each axis of a drive. The data block consists of the following data structures:
  – UDT 890 - *VMC_Config*SigmaPN_REF
    The data structure describes the structure of the configuration of the drive.
    Specific data structure for *Sigma-5/7* PROFINET.
  – UDT 860 - *MC_AXIS_REF*
    The data structure describes the structure of the parameters and status information of drives.
    General data structure for all drives and bus systems.
- FB 891 - *VMC_Init*Sigma_PN
  – The *Init*t block is used to configure an axis.
  – Specific block for *Sigma-5/7* PROFINET.
  – The configuration data for the initialization must be stored in the *axis DB*.
- FB 890 - *VMC_AxisControl*Sigma_PN
  – Specific block for *Sigma-5/7* PROFINET.
  – This block is a combination of *Kernel* and *AxisControl* and communicates with the drive via PROFINET, processes the user requests and returns status messages.
  – This block supports simple motion commands and returns all relevant status messages.
  – The exchange of the data takes place by means of the *axis DB*.
  – For motion control and status query, via the instance data of the block you can link a visualization.
  – In addition to the FB 890 - *VMC_AxisControl*Sigma_PN, *PLCopen* blocks can be used.
- FB 800 ... FB 838 - *PLCopen*
  – The PLCopen blocks are used to program motion sequences and status queries.
  – General blocks for all drives and bus systems.

4.2.5.3.2 Programming

Include library

1. Go to the service area of www.vipa.com.
2. Download the *Simple Motion Control* library from the download area at *'VIPA Lib'*.
   The library is available as packed zip file for the corresponding TIA Portal version.
3. Start your un-zip application with a double click on the file ...TIA_V*xx*.zip and copy all the files and folders in a work directory for the Siemens TIA Portal.
4. Switch to the *Project view* of the Siemens TIA Portal.
5. Choose "Libraries" from the task cards on the right side.
6. Click at "Global library".
7. Click on the free area inside the *'Global Library'* and select *'Context menu ➔ Retrieve library'*.
8. Navigate to your work directory and load the file ...Simple Motion.zal*xx*.

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.

Copy blocks into project

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

- *Sigma* PROFINET:
  – UDT 890 - VMC_ConfigSigmaPN_REF *Chap. 4.3.1 'UDT 890 - VMC_ConfigSigmaPN_REF - Sigma-5/7 PROFINET Data structure axis configuration' page 213*
  – FB 890 - VMC_AxisControlSigma_PN *Chap. 4.3.2 'FB 890 - VMC_AxisControlSigma_PN - control block axis control for Sigma-5/7 PROFINET' page 213*
  – FB 891 - VMC_InitSigma_PN *Chap. 4.3.3 'FB 891 - VMC_InitSigma_PN - Sigma-5/7 PROFINET initialization' page 217*
- Axis control
  – UDT 860 - MC_AXIS_REF *Chap. 9.2.1 'UDT 860 - MC_AXIS_REF - Data structure axis data' page 363*
  – FB 860 - VMC_AxisControl *Chap. 9.2.2 'FB 860 - VMC_AxisControl - Control block axis control' page 363*

Create axis DB

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 assign it the name "Axis01". The DB number can freely be selected such as DB 10. Specify DB 10 and create this as a global DB with [OK].
   ⇨ The block is created and opened.
3. In "Axis01" create the following variables:
   - *'Config'* of Type UDT 890 - VMC_ConfigSigmaPN_REF.
     These are specific axis configuration data.
   - *'Config'* of Type UDT 860 - MC_AXIS_REF.
     During operation, all operating data of the axis are stored here.

OB 1 - configuration of the axes

Open OB 1 and program the following FB calls with associated DBs:

FB 891 - VMC_InitSigma_PN, DB 891

The connection between the axes in the hardware configuration and your user program is made by specifying the following module properties in the call parameters of FB 891 - VMC InitSigma_PN:

– *Module properties 'Parameter Access Point': Diagnostic address of slot 1 of the slot overview*
  – *FB 891 - VMC InitSigma_PN: ParaAccessPointAddress:*
    Setting of the diagnostic address of slot 1 of the slot overview.
– *Module properties 'YASKAWA Telegram PZD...':*
  Respective start address of the input/output address range.
  – *FB 891 - VMC InitSigma_PN: 'InputsStartAddress':*
    Setting of the start address of the input address range.
  – *FB 891 - VMC InitSigma_PN: 'OutputsStartAddress':*
    Setting of the start address of the output address range.
  – *FB 891 - VMC InitSigma_PN: 'LogicalAddress':*
    Setting of the of the smaller value of the start addresses of the input/output address range.

- Hardware configuration *198*
- FB 891 - VMC InitSigma_PN *217*

Example hardware configuration

Slot	Component	...	I-Adr.	O-Adr.	Diagnostic address
0	SGD7S-xxxAC0xxxx				2035
X1	PN-IO				2034
X1 P1	Port 1				2033
X1 P2	Port 2				2032
1	DO with YASKAWA telegr.100, PZD-16/14				2044
1.1	Parameter Access Point				2044
1.2	YASKAWA telegram, PZD-16/14		28-55	32-63	

Example call

```
CALL  "VMC_InitSigma_PN" , "VMC_InitSigma_PN_1"
Enable                    :="InitS7PN1_Enable"
LogicalAddress            :=28 //HW-Konfig: Smallest IO addr.
ParaAccessPointAdress     :=2044 //HW-Konfig: Diag addr.
InputsStartAddress        :=28 //HW-Konfig: Telegr.100 start I addr.
OutputsStartAddress       :=32 //HW-Konfig: Telegr. 100 start O addr.
EncoderType               :=1
EncoderResolutionBits     :=20
FactorPosition            :=1.048576e+006
FactorVelocity            :=1.048576e+006
FactorAcceleration        :=1.048576e+006
OffsetPosition            :=0.000000e+000
MaxVelocityApp            :=5.000000e+001
MaxAccelerationApp        :=1.000000e+002
MaxDecelerationApp        :=1.000000e+002
MaxVelocityDrive          :=6.000000e+001
MaxPosition               :=1.048500e+003
MinPosition               :=-1.048514e+003
EnableMaxPosition         :=TRUE
EnableMinPosition         :=TRUE
MinUserPosition           :="InitS7PN1_MinUserPos"
MaxUserPosition           :="InitS7PN1_MaxUserPos"
Valid                     :="InitS7PN1_Valid"
Error                     :="InitS7PN1_Error"
ErrorID                   :="InitS7PN1_ErrorID"
Config                    :="Axis01".Config
Axis                      :="Axis01".Axis
```

Connecting the AxisControl

FB 890 - VMC_AxisControlSigma_PN, DB 890 ➪ *Chap. 4.3.2 'FB 890 - VMC_AxisControlSigma_PN - control block axis control for Sigma-5/7 PROFINET' page 213*

This block processes the user commands and passes them appropriately processed on to the drive via PROFINET.

```
CALL  "VMC_AxisControlSigma_PN" , "DI_AxisControlSigmaPN01"
 AxisEnable        :="AxCtrl1_AxisEnable"
 AxisReset         :="AxCtrl1_AxisReset"
 HomeExecute       :="AxCtrl1_HomeExecute"
 HomePosition      :="AxCtrl1_HomePosition"
 StopExecute       :="AxCtrl1_StopExecute"
 MvVelocityExecute:="AxCtrl1_MvVelExecute"
 MvRelativeExecute:="AxCtrl1_MvRelExecute"
 MvAbsoluteExecute:="AxCtrl1_MvAbsExecute"
 PositionDistance  :="AxCtrl1_PositionDistance"
 Direction         :="AxCtrl1_Direction"
```

```
Velocity          :="AxCtrl1_Velocity"
Acceleration      :="AxCtrl1_Acceleration"
Deceleration      :="AxCtrl1_Deceleration"
JogPositive       :="AxCtrl1_JogPositive"
JogNegative       :="AxCtrl1_JogNegative"
JogVelocity       :="AxCtrl1_JogVelocity"
JogAcceleration   :="AxCtrl1_JogAcceleration"
JogDeceleration   :="AxCtrl1_JogDeceleration"
AxisReady         :="AxCtrl1_AxisReady"
AxisEnabled       :="AxCtrl1_AxisEnabled"
AxisError         :="AxCtrl1_AxisError"
AxisErrorID       :="AxCtrl1_AxisErrorID"
DriveWarning      :="AxCtrl1_DriveWarning"
DriveError        :="AxCtrl1_DriveError"
DriveErrorID      :="AxCtrl1_DriveErrorID"
IsHomed           :="AxCtrl1_IsHomed"
ModeOfOperation   :="AxCtrl1_ModeOfOperation"
PLCopenState      :="AxCtrl1_PLCopenState"
ActualPosition    :="AxCtrl1_ActualPosition"
ActualVelocity    :="AxCtrl1_ActualVelocity"
CmdDone           :="AxCtrl1_CmdDone"
CmdBusy           :="AxCtrl1_CmdBusy"
CmdAborted        :="AxCtrl1_CmdAborted"
CmdError          :="AxCtrl1_CmdError"
CmdErrorID        :="AxCtrl1_CmdErrorID"
DirectionPositive:="AxCtrl1_DirectionPos"
DirectionNegative:="AxCtrl1_DirectionNeg"
SWLimitMinActive  :="AxCtrl1_SWLimitMinActive"
SWLimitMaxActive  :="AxCtrl1_SWLimitMaxActive"
HWLimitMinActive  :="AxCtrl1_HWLimitMinActive"
HWLimitMaxActive  :="AxCtrl1_HWLimitMaxActive"
Axis              :="Axis01".Axis
```

For complex motion tasks, you can use the PLCopen blocks. Please specify the reference to the corresponding axis data at Axis in the axis DB.

Your project now includes the following blocks:

- OB 1 - Main
- OB 57 - DP Manufacturer Alarm
- OB 82 - I/O_FLT1
- OB 86 - Rack_FLT
- FB 890 - VMC_AxisControlSigma_PN with instance DB
- FB 891 - VMC_InitSigma_PN with instance DB
- UDT 860 - MC_Axis_REF
- UDT 890 - VMC_ConfigSigmaPN_REF

Sequence of operations

1. Select *'Project* ➔ *Compile all'* and transfer the project into your 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. To do this, call the *Init* block FB 891 - VMC_InitSigma_PN with *Enable* = TRUE.

   ⇨ The output *Valid* returns TRUE. In the event of a fault, you can determine the error by evaluating the *ErrorID*.

   You have to call the *Init* block again if you load a new axis DB or you have changed parameters on the *Init* block.

   > *Do not continue until the Init block does not report any errors!*

3. Program your application with the FB 890 - VMC_AxisControlSigma_PN or with the PLCopen blocks.

4.3 Drive specific blocks

> *The PLCopen blocks for axis control can be found here:* *Chap. 9 'Blocks for axis control' page 361*

4.3.1 UDT 890 - VMC_ConfigSigmaPN_REF - *Sigma-5/7* PROFINET Data structure axis configuration

This is a user-defined data structure that contains information about the configuration data. The UDT is specially adapted to the use of a *Sigma-5/7* drive, which is connected via PROFINET.

4.3.2 FB 890 - VMC_AxisControlSigma_PN - control block axis control for Sigma-5/7 PROFINET

Description

The FB *VMC_AxisControlSigma_PN* is a combination of a *Kernel* for Sigma-5/7 axes for PROFINET and an *Axis_Control* for controlling the motion control functions. With the FB *VMC_AxisControlSigma_PN* you can control the connected axis. You can check the status of the drive, turn the drive on or off, or execute various motion commands.

> *The VMC_AxisControlSigma_PN block should never be used simultaneously with the PLCopen block MC_Power. Since the VMC_AxisControlSigma_PN contains functionalities of the MC_Power and the latest command from the Kernel is always executed, this can lead to a faulty behavior of the drive.*

> *Please note that an attempt to abort a movement e.g. by homing, the status of the current movement request can no longer be determined via CmdDone or CmdBusy. Here the evaluation of the current movement should be done via the current position or velocity and the PLCopen status.*

If a running MoveVelocity job is aborted by a new MoveRelative or Move-Absolute job, the corresponding drive is stopped and then the new move job is executed.

Parameter

Parameter	Declaration	Data type	Description
AxisEnable	INPUT	BOOL	■ Enable/disable axis – TRUE: The axis is enabled. – FALSE: The axis is disabled.
AxisReset	INPUT	BOOL	■ Reset axis – Edge 0-1: Axis reset is performed.
HomeExecute	INPUT	BOOL	■ Homing – Edge 0-1: Homing is started.
HomePosition	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 application unit.
StopExecute	INPUT	BOOL	■ Stop axis – Edge 0-1: Stopping of the axis is started.
MvVelocityExecute	INPUT	BOOL	■ Start moving the axis – Edge 0-1: The axis is accelerated / decelerated to the speed specified.
MvRelativeExecute	INPUT	BOOL	■ Start moving the axis – Edge 0-1: The relative positioning of the axis is started.
MvAbsoluteExecute	INPUT	BOOL	■ Start moving the axis – Edge 0-1: The absolute positioning of the axis is started.
Direction *	INPUT	BYTE	Mode for absolute positioning: ■ 0: shortest distance ■ 1: positive direction ■ 2: negative direction ■ 3: current direction
PositionDistance	INPUT	REAL	Absolute position or relative distance depending on the command in [user units].
Velocity	INPUT	REAL	Velocity setting (signed value) in [user units / s].
Acceleration	INPUT	REAL	Acceleration in [user units / s^2].
Deceleration	INPUT	REAL	Deceleration in [user units / s^2].
JogPositive	INPUT	BOOL	■ Drive axis with constant velocity in positive direction – Edge 0-1: Drive axis with constant velocity is started. – Edge 1-0: The axis is stopped.
JogNegative	INPUT	BOOL	■ Drive axis with constant velocity in negative direction – Edge 0-1: Drive axis with constant velocity is started. – Edge 1-0: The axis is stopped.
JogVelocity	INPUT	REAL	Speed setting for jogging (positive value) in [user units / s].

Parameter	Declaration	Data type	Description
JogAcceleration	INPUT	REAL	Acceleration in [user units / s^2].
JogDeceleration	INPUT	REAL	Delay for jogging in [user units / s^2].
KernelInitReset	INPUT	BOOL	Reset the kernel functions. Caution, running commands are aborted!
AxisReady	OUTPUT	BOOL	■ AxisReady – TRUE: The axis is ready to switch on. – FALSE: The axis is not ready to switch on. → Check and fix AxisError (see *AxisErrorID*). → Check and fix DriveError (see *DriveErrorID*). → Check initialization FB (input and output addresses or diagnostics address?)
AxisEnabled	OUTPUT	BOOL	■ Status axis – TRUE: Axis is switched on and accepts motion commands. – FALSE: Axis is not switched on and does not accepts motion commands.
AxisError	OUTPUT	BOOL	■ Motion axis error – TRUE: An error has occurred. Additional error information can be found in the parameter *AxisErrorID*. → The axis is disabled.
AxisErrorID	OUTPUT	WORD	Additional error information *Chap. 12 'ErrorID - Additional error information' page 457*
DriveWarning	OUTPUT	BOOL	■ Warning – TRUE: There is a warning on the drive. Additional information can be found in the manufacturer's manual.
DriveError	OUTPUT	BOOL	■ Error on the drive – TRUE: An error has occurred. Additional error information can be found in the parameter *DriveErrorID*. → The axis is disabled.
DriveErrorID	OUTPUT	WORD	■ Error – TRUE: There is an error on the drive. Additional information can be found in the manufacturer's manual.
IsHomed	OUTPUT	BOOL	■ Information axis: homed – TRUE: The axis is homed.

Parameter	Declaration	Data type	Description
ModeOfOperation	OUTPUT	INT	Drive-specific mode. For further information see drive manual. Example *Sigma-5*: 0: No mode changed/no mode assigned 1: Profile Position mode 2: Reserved (keep last mode) 3: Profile Velocity mode 4: Torque Profile mode 6: Homing mode 7: Interpolated Position mode 8: Cyclic Sync Position mode 9: Cyclic Sync Velocity mode 10: Cyclic Sync Torque mode Other Reserved (keep last mode)
PLCopenState	OUTPUT	INT	Current PLCopenState: 1: Disabled 2: Standstill 3: Homing 4: Discrete Motion 5: Continuous Motion 7: Stopping 8: Errorstop
ActualPosition	OUTPUT	REAL	Position of the axis in [user unit].
ActualVelocity	OUTPUT	REAL	Velocity of the axis in [user unit / s]
CmdDone	OUTPUT	BOOL	■ Status – TRUE: Job ended without error.
CmdBusy	OUTPUT	BOOL	■ Status – TRUE: Job is running.
CmdAborted	OUTPUT	BOOL	■ Status – TRUE: The job was aborted during processing by another job. If *Mv...Execute* is already FALSE before the command is interrupted, *CmdAborted* is set to TRUE for one cycle only.
CmdError	OUTPUT	BOOL	■ Status – TRUE: An error has occurred. Additional error information can be found in the parameter *CmdErrorID*.
CmdErrorID	OUTPUT	WORD	Additional error information *Chap. 12 'ErrorID - Additional error information' page 457*
DirectionPositive	OUTPUT	BOOL	■ Status motion job: Position increasing – TRUE: The position of the axis is increasing

Parameter	Declaration	Data type	Description
DirectionNegative	OUTPUT	BOOL	■ Status motion job: Position decreasing – TRUE: The position of the axis is decreasing
SWLimitMinActive	OUTPUT	BOOL	■ Software limit switch – TRUE: Software Limit switch Minimum active (Minimum position in negative direction exceeded).
SWLimitMaxActive	OUTPUT	BOOL	■ Software limit switch – TRUE: Software limit switch Maximum active (Max-imum position in positive direction exceeded).
HWLimitMinActive	OUTPUT	BOOL	■ Hardware limit switch – TRUE: Negative hardware limit switch active on the drive (NOT- Negative Overtravel).
HWLimitMaxActive	OUTPUT	BOOL	■ Hardware limit switch – TRUE: Positive hardware limit switch active on the drive (POT- Positive Overtravel).
Config	IN_OUT	VMC_Config-SigmaPN_REF	Reference to the configuration of the axis.
Axis	IN_OUT	MC_AXIS_REF	Reference to the axis.
*) This parameter is currently not supported! It is always taken the shortest way. The test is carried out on values from 0 to 3.			

4.3.3 FB 891 - VMC_InitSigma_PN - *Sigma-5/7* PROFINET initialization

Description

This block is used to configure the axis. The module is specially adapted to the use of a *Sigma-5/7* drive, which is connected via PROFINET.

Parameter

Parameter	Declaration	Data type	Description
Enable	INPUT	BOOL	Release of initialization
Logical address	INPUT	INT	Smallest address of the input/output address range of the hardware configuration of the axis.
ParaAccessPointAdress	INPUT	INT	Diagnostic address of slot 1 of the hardware configuration of the axis.
InputsStartAddress	INPUT	INT	Start address of the input address range of the hardware configuration of the axis.
OutputsStartAddress	INPUT	INT	Start address of the output address range of the hardware configuration of the axis.
EncoderType	INPUT	INT	Encoder type ■ 1: Absolute encoder ■ 2: Incremental encoder
EncoderResolutionBits	INPUT	INT	Number of bits corresponding to one encoder revolution. Default: 20
FactorPosition	INPUT	REAL	Factor for converting the position of user units [u] into drive units [increments] and back. It's valid: $p_{[increments]} = p_{[u]}$ x *FactorPosition* Please consider the factor which can be specified on the drive via the objects 0x2301: 1 and 0x2301: 2. This should be 1.
Velocity Factor	INPUT	REAL	Factor for converting the speed of user units [u/s] into drive units [increments/s] and back. It's valid: $v_{[increments/s]} = v_{[u/s]}$ x *FactorVelocity* Please also take into account the factor which you can specify on the drive via objects 0x2302: 1 and 0x2302: 2. This should be 1.
FactorAcceleration	INPUT	REAL	Factor to convert the acceleration of user units [u/s^2] in drive units [10^{-4} x increments/s^2] and back. It's valid: 10^{-4} x $a_{[increments/s^2]} = a_{[u/s^2]}$ x *FactorAcceleration* Please also take into account the factor which you can specify on the drive via objects 0x2303: 1 and 0x2303: 2. This should be 1.
OffsetPosition	INPUT	REAL	Offset for the zero position [u].
MaxVelocityApp	INPUT	REAL	Maximum application speed [u/s]. The command inputs are checked to the maximum value before execution.
MaxAccelerationApp	INPUT	REAL	Maximum acceleration of the application [u/s^2]. The command inputs are checked to the maximum value before execution.
MaxDecelerationApp	INPUT	REAL	Maximum application deceleration [u/s^2]. The command inputs are checked to the maximum value before execution.
MaxPosition	INPUT	REAL	Maximum position for monitoring the software limits [u].

Parameter	Declaration	Data type	Description
MinPosition	INPUT	REAL	Minimum position for monitoring the software limits [u].
EnableMaxPosition	INPUT	BOOL	Monitoring maximum position ■ TRUE: Activates the monitoring of the maximum position.
EnableMinPosition	INPUT	BOOL	Monitoring minimum position ■ TRUE: Activation of the monitoring of the minimum position.
MinUserPosition	OUTPUT	REAL	Minimum user position based on the minimum encoder value of 0x80000000 and the *FactorPosition* [u].
MaxUserPosition	OUTPUT	REAL	Maximum user position based on the maximum encoder value of 0x7FFFFFFF and the *FactorPosition* [u].
Valid	OUTPUT	BOOL	Initialization ■ TRUE: Initialization is valid.
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. 12 'ErrorID - Additional error information' page 457*
Config	IN_OUT	VMC_Config-SigmaPN_REF	Data structure for transferring axis-dependent configuration data to the *AxisKernel*.
Axis	IN_OUT	MC_AXIS_REF	Data structure for transferring axis-dependent information to the *AxisKernel* and PLCopen blocks.

5 Usage *Sigma-5/7* Pulse Train

5.1 Overview

Precondition

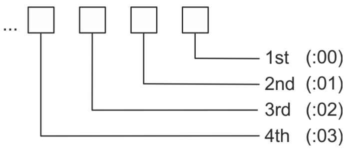
- SPEED7 Studio from V1.7
  or
- Siemens SIMATIC Manager from V 5.5, SP2 & *Simple Motion Control Library*
  or
- Siemens TIA Portal V 14 & *Simple Motion Control Library*
- System MICRO or System SLIO CPU with Pulse Train output, such as CPU M13-CCF0000 or CPU 013-CCF0R00.
- *Sigma-5*- respectively *Sigma-7* drive with Pulse Train option card

Steps of configuration

1. Setting parameters on the drive
   - The setting of the parameters happens by means of the software tool *Sigma Win+*.
2. Hardware configuration in the VIPA *SPEED7 Studio*, Siemens SIMATIC Manager or Siemens TIA Portal.
   - Configuring the CPU.
3. Programming in the VIPA *SPEED7 Studio*, Siemens SIMATIC Manager or Siemens TIA Portal.
   - *VMC_AxisControl_PT* block for configuration and communication with the axis, which is connected via Pulse Train.

5.2 Set the parameters on the drive

Parameter digits



CAUTION!
Before the commissioning, you have to adapt your drive to your application with the *Sigma Win+* software tool! More may be found in the manual of your drive.

The following table shows all parameters which do not correspond to the default values. The following parameters must be set via *Sigma Win+* to match the *Simple Motion Control Library*:

Sigma-5/7

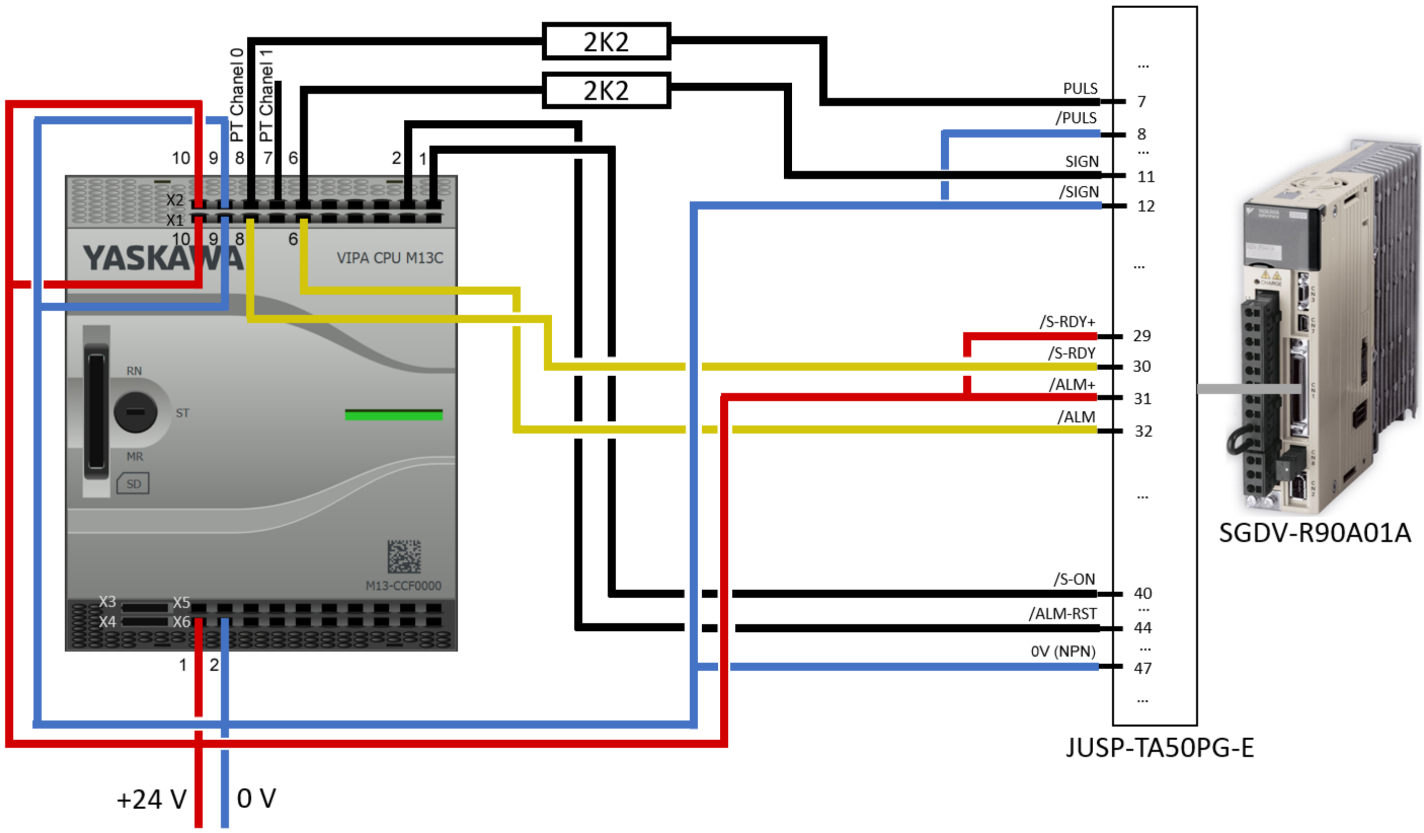
Servopack Parameter	Address:digit	Name	Value
Pn000	(2000h:01)	Basic Function Selection Switch 0	1: Position control (pulse train reference)
Pn002	(2002h:02)	Application Function Select Switch 2	1: Uses absolute encoder as incremental encoder
Pn200	(2200h:03)	Position Control Reference From Selection Switch	1: Uses reference input filter for open collector signal
Pn20E	(220Eh)	Electronic Gear Ratio (Numerator)	1024
Pn216	(2216h)	Position Reference Acceleration / Deceleration Time Constant	0
Pn217	(2217h)	Average Movement Time of Position Reference	0

Servopack Parameter	Address:digit	Name	Value
Pn50A	(250Ah:02)	/P-CON Signal Mapping	8: Sets signal off
Pn50A	(250Ah:03)	P-OT Signal Mapping	8: Forward run allowed
Pn50B	(250Bh:00)	N-OT Signal Mapping	8: Reverse run allowed
Pn50B	(250Bh:02)	/P-CL Signal Mapping	8: Sets signal off
Pn50B	(250Bh:03)	/N-CL Signal Mapping	8: Sets signal off

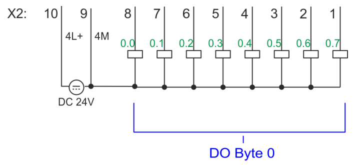
5.3 Wiring

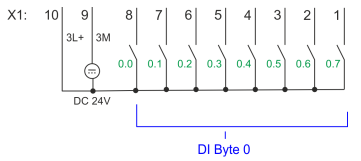
Sample application

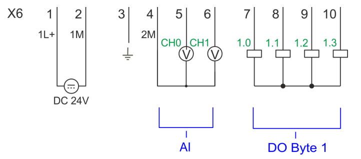
The following figure shows the connection of a Sigma-5 servo drive via Pulse Train to a system MICRO CPU M13C. In this example the pulse train channel 0 (X2 - pin 8) is connected. Please use X2 pin 7 to connect to channel 1.



Wiring







X2	Func-tion	Type	LED green red	Description
1	DO 0.7	O		Digital output DO 7
2	DO 0.6	O		Digital output DO 6
6	DO 0.2	O		Digital output DO 2
7	DO 0.1	O		Pulse Train Channel 1
8	DO 0.0	O		Pulse Train Channel 0
9	0 V	I		4M: GND for Pulse Train LED is on when there is an error, overload or short circuit at the outputs
10	DC 24V	I		4L+: DC 24V power supply for Pulse Train

X1	Func-tion	Type	LED green	Description
6	DI 0.2	I		Digital input DI 2
8	DI 0.0	I		Digital input DI 0
9	0 V	I		3M: GND power section supply for on-board DI
10	DC 24V	I		3L+: DC 24V power section supply for on-board DI

X6	Func-tion	Type	LED green	Description
1	Sys DC 24V	I		1L+: DC 24V for electronic section supply
2	Sys 0V	I		1M: GND for electronic section supply

5.4 Usage in VIPA *SPEED7 Studio*

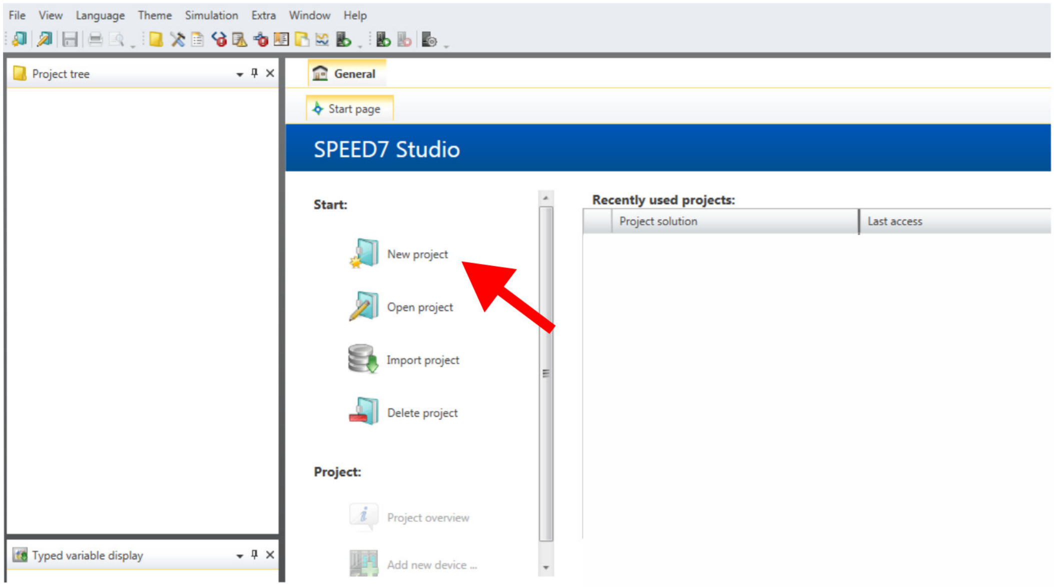
5.4.1 Hardware configuration

Add CPU in the project

Please use the *SPEED7 Studio* V1.7 and up for the configuration.

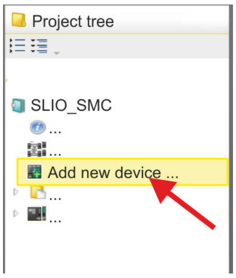
If you are using a channel other than channel 0, you must adapt it in the hardware configuration and in your user program.

1. Start the *SPEED7 Studio*.

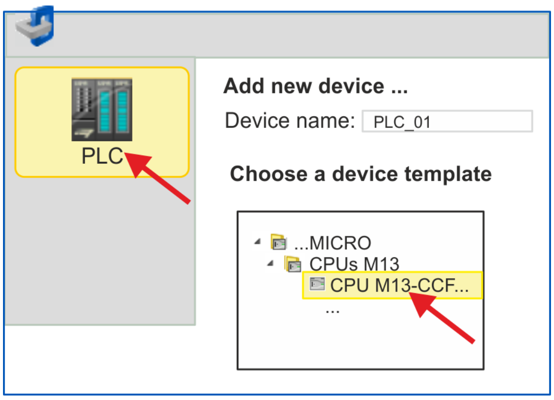


2. Create a new project at the start page with *'New project'* and assign a *'Project name'*.

   ⇨ A new project is created and the view *'Devices and networking'* is shown.



3. Click in the *Project tree* at *'Add new device ...'*.



   ⇨ A dialog for device selection opens.

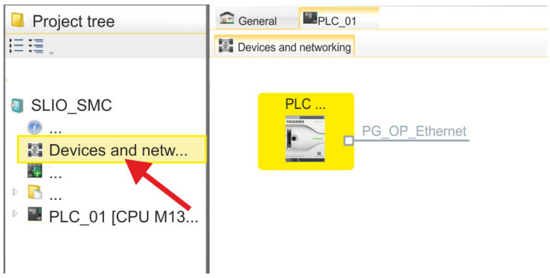
4. Select from the *'Device templates'* your CPU with Pulse Train functionality like the System MICRO CPU M13-CCF0000 and click at [OK].

   ⇨ The CPU is inserted in *'Devices and networking'* and the *'Device configuration'* is opened.

Configuration of Ethernet PG/OP channel

1. Click in the *Project tree* at *'Devices and networking'*.

   ⇨ You will get a graphical object view of your CPU.



2. Click at the network *'PG_OP_Ethernet'*.
3. Select *'Context menu ➔ Interface properties'*.

   ⇨ A dialog window opens. Here you can enter the IP address data for your Ethernet PG/OP channel. You get valid IP address parameters from your system administrator.
4. Confirm with [OK].

   ⇨ The IP address data are stored in your project listed in *'Devices and networking'* at *'Local components'*.

   After transferring your project your CPU can be accessed via Ethernet PG/OP channel with the set IP address data.

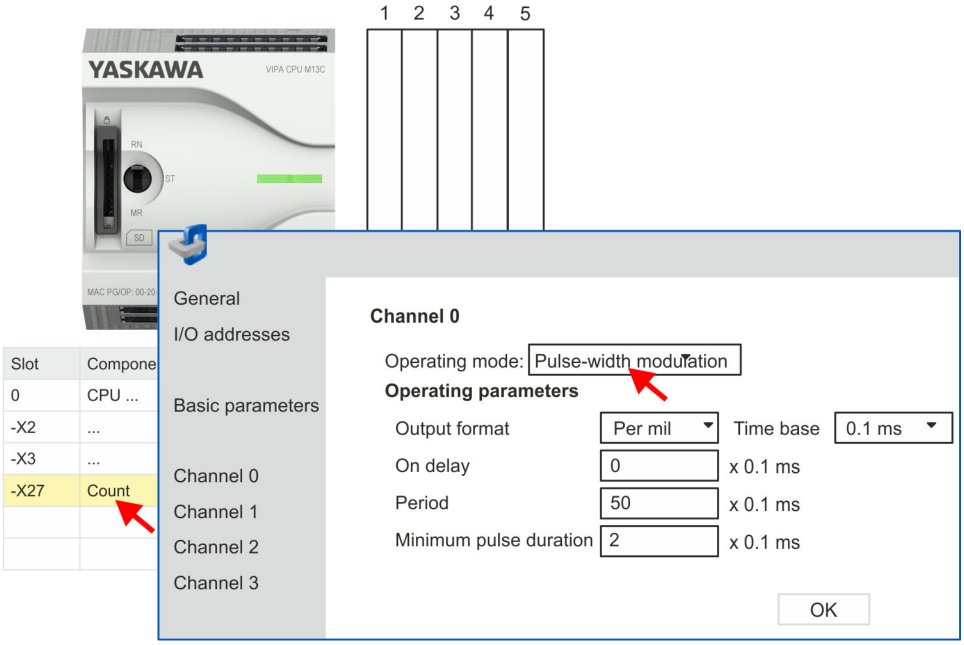
Switch I/O periphery to Pulse Train

For parametrization of the I/O periphery and the *technological functions* the corresponding sub modules of the CPU are to be used. For pulse train output, the sub module count must be switched to *'Pulse-width modulation'*.

1. Click in the *Project tree* at *'PLC... > Device configuration'*.
2. Click in the *'Device configuration'* at *'-X27 Count'* and select *'Context menu ➔ Components properties'*.

   ⇨ The properties dialog is opened.
3. For example, select *'channel 0'* and select the function *'Pulse-width modulation'* as *'Operating mode'*.

4. The operating parameters required for Pulse Train are internally adapted to the corresponding values. Leave all values unchanged.

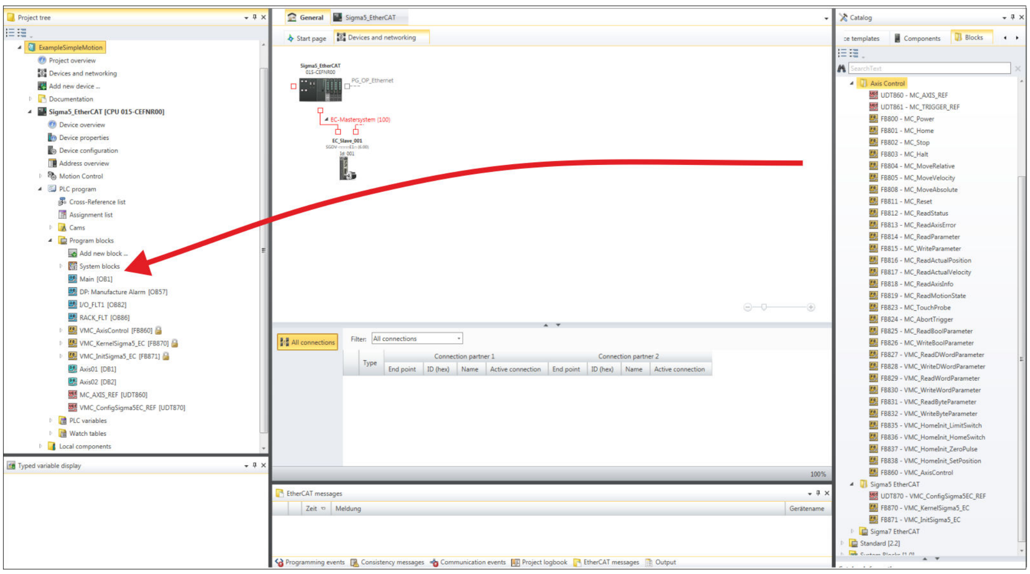


5. Close the dialog with [OK].

6. Select *'Project ➔ Compile all'*.

5.4.2 User program

Copy block to project



In the *'Catalog'*, open the *'Simple Motion Control'* library at *'Blocks'* and drag and drop the following blocks into *'Program blocks'* of the *Project tree*:

- *Sigma5+7* Pulse Train
  - FB 875 - VMC_AxisControl_PT *Chap. 5.7.1 'FB 875 - VMC_AxisControl_PT - Axis control via Pulse Train' page 238*

OB 1

Configuration of the axis

If you are using a channel other than channel 0, you must adapt it in the hardware configuration and in your user program.

1. Open in the *Project tree* within the CPU at *'PLC program'*, *'Programming blocks'* the OB 1 and program the Call FB 875, DB 875.
   ⇨ The dialog *'Add instance data block'* opens.
2. Set the number for the instance data block, if not already done, and close the dialog with [OK].
   ⇨ The block call is created and the parameters are listed
3. Assign the following parameters for the sample project. In particular, consider the two conversion factors *FactorPosition* and *FactorVelocity*:
   ⇨
```
CALL  FB    "VMC_AxisControl_PT" , "DI_AxisControl_PT"
         S_ChannelNumberPWM      := 0
         S_Ready                 := E 136.0
         S_Alarm                 := E 136.2
         FactorPosition          := 1024.0
         FactorVelocity          := 976.5625
         AxisEnable              := M 100.1
         AxisReset               := M 100.2
         StopExecute             := M 100.3
         MvVelocityExecute       := M 100.4
         MvRelativeExecute       := M 100.5
         JogPositive             := M 100.6
         JogNegative             := M 100.7
         PositionDistance        := MD 102
         Velocity                := MD 106
         S_On                    := A 136.7
         S_Direction             := A 136.2
         S_AlarmReset            := A 136.6
         MinUserDistance         := MD 110
         MaxUserDistance         := MD 114
         MinUserVelocity         := MD 118
         MaxUserVelocity         := MD 122
         AxisReady               := M 101.3
         AxisEnabled             := M 101.4
         AxisError               := M 101.5
         AxisErrorID             := MW 126
         DriveError              := M 101.6
         CmdActive               := MB 128
         CmdDone                 := M 130.0
         CmdBusy                 := M 130.1
         CmdAborted              := M 130.2
         CmdError                := M 130.3
         CmdErrorID              := MW 132
```

   The addresses of *S_Ready* and *S_Alarm* are derived from the addresses of the inputs which are connected to the drive's digital outputs. These can be determined via the sub module *'-X25 DI/DIO'* of the CPU.

   The addresses of *S_On*, *S_Direction* and *S_AlarmReset* are obtained from the addresses of the outputs which are connected to the digital inputs of the drive. These can be determined via the sub module *'-X25 DI/DIO'* of the CPU.

Sequence of operations

1. Select *'Project ➔ Compile all'* and transfer the project into your CPU.

   You can find more information on the transfer of your project in the online help of the *SPEED7 Studio*.

   ⇨ You can take your application into operation now.

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

2. Bring your CPU into RUN and turn on your drive.

   ⇨ The FB 875 - VMC_AxisControl_PT is executed cyclically.

3. As soon as *AxisReady* = TRUE, you can use *AxisEnable* to enable the drive.
4. You now have the possibility to control your drive via its parameters and to check its status. *Chap. 5.7.1 'FB 875 - VMC_AxisControl_PT - Axis control via Pulse Train' page 238*

Controlling the drive via HMI

There is the possibility to control your drive via an HMI. For this purpose, a predefined symbol library is available for Movicon to access the VMC_AxisControl_PT function module. *Chap. 10 'Controlling the drive via HMI' page 432*

5.5 Usage in Siemens SIMATIC Manager

5.5.1 Precondition

Overview

- Please use for configuration the Siemens SIMATIC Manager V 5.5 SP2 and up.
- The configuration of the VIPA CPU with Pulse Train functionality happens in the Siemens SIMATIC Manager by means of a virtual PROFINET IO device.
- The PROFINET IO Device is to be installed in the hardware catalog by means of a GSDML.

Installing the VIPA IO device

The installation of the PROFINET VIPA IO device 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 according PROFINET IO device can be found at *'PROFINET IO ➔ Additional field devices ➔ I/O ➔ VIPA ...'*.

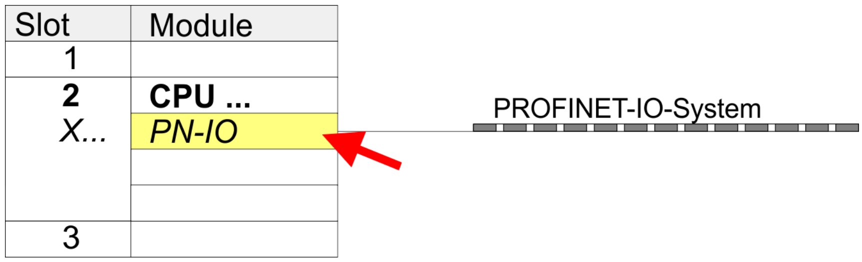
5.5.2 Hardware configuration

Add CPU in the project

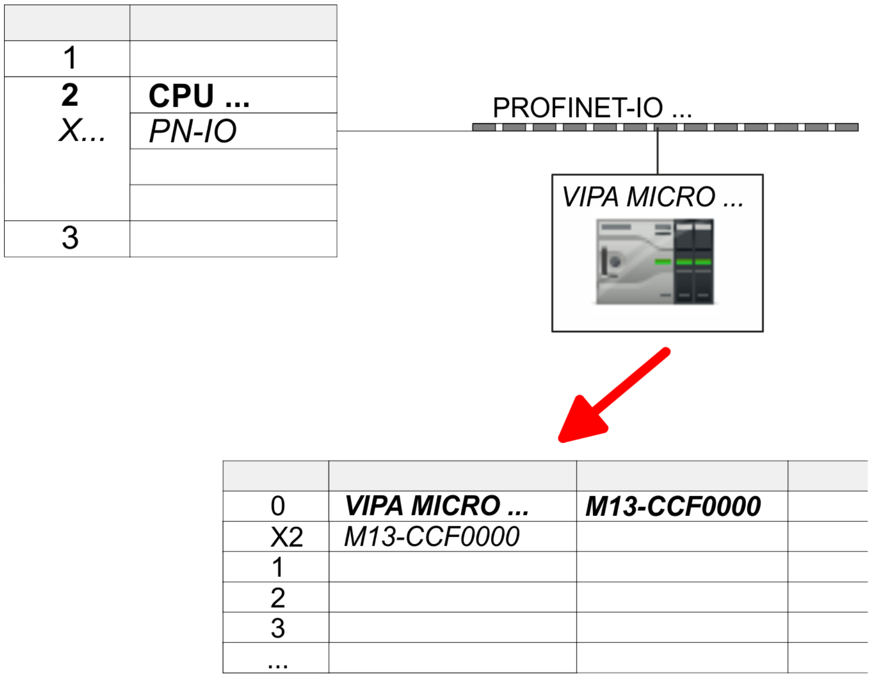
Slot	Module
1	
2	**CPU 314C-2PN/DP**
X1	*MPI/DP*
X2	*PN-IO*
X2...	*Port 1*
X2...	*Port 2*
...	...
3	

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 314C-2 PN/DP (314-6EH04-0AB0 V3.3).
4. Click at the sub module *'PN-IO'* of the CPU.
5. Select *'Context menu ➔ Insert PROFINET IO System'*.



6. Create with [New] a new sub net and assign valid address data.
7. Click at the sub module *'PN-IO'* of the CPU and open with *'Context menu ➔ Properties'* the properties dialog.
8. Enter at *'General'* a *'Device name'*. The device name must be unique at the Ethernet subnet.



9. Navigate in the hardware catalog to the directory *'PROFINET IO ➔ Additional field devices ➔ I/O ➔ VIPA ...'* and connect e.g. for the System MICRO the IO device *'M13-CCF0000'* to your PROFINET system.

⇨ In the *Device overview* of the PROFINET IO device *'VIPA MICRO PLC'* the CPU is already placed at slot 0.

Configuration of Ethernet PG/OP channel

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!

Switch I/O periphery to Pulse Train

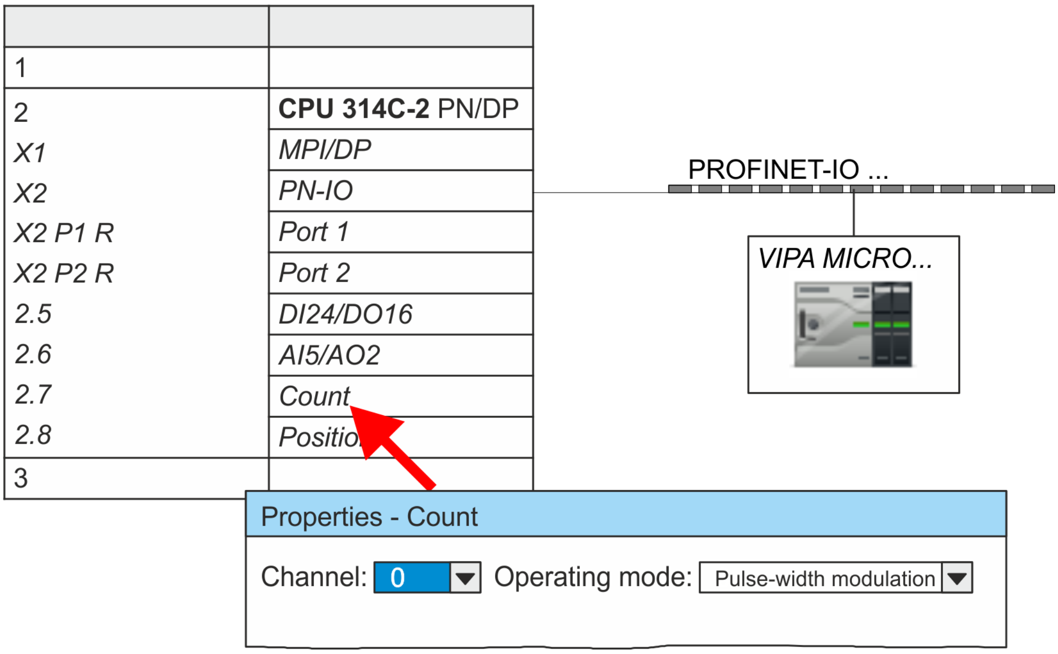
For parametrization of the input/output periphery and the *technological functions* the corresponding sub modules of the Siemens CPU 314C-2 PN/DP (314-6EH04-0AB0 V3.3) is to be used. For pulse train output, the sub module count must be switched to *'Pulse-width modulation'*. If you are using a channel other than channel 0, you must adapt it in the hardware configuration and in your user program.

1. Double-click the counter sub module of the CPU 314C-2 PN/DP.

⇨ The dialog *'Properties'* is opened.

2. For example, select *'channel 0'* and select the function *'Pulse-width modulation'* as *'Operating mode'*.

3. Leave all values unchanged.



4. Close the dialog with [OK].
5. Select *'Station → Save and compile'*.
6. Close the hardware configurator.

5.5.3 User program

Include library

1. Go to the service area of www.vipa.com.
2. Download the *Simple Motion Control* library from the download area at *'VIPA Lib'*.
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:
  - *Sigma5+7* Pulse Train
    - FB 875 - VMC_AxisControl_PT *Chap. 5.7.1 'FB 875 - VMC_AxisControl_PT - Axis control via Pulse Train' page 238*

OB 1

Configuration of the axis

1. Open the OB 1 and program the Call FB 875, DB 875.
   ⇨ The block call is created and the parameters are listed.

2. Assign the following parameters for the sample project. In particular, consider the two conversion factors *FactorPosition* and *FactorVelocity*:

⇨
```
CALL  FB    "VMC_AxisControl_PT" , "DI_AxisControl_PT"
         S_ChannelNumberPWM      := 0
         S_Ready                 := E 136.0
         S_Alarm                 := E 136.2
         FactorPosition          := 1024.0
         FactorVelocity          := 976.5625
         AxisEnable              := M 100.1
         AxisReset               := M 100.2
         StopExecute             := M 100.3
         MvVelocityExecute       := M 100.4
         MvRelativeExecute       := M 100.5
         JogPositive             := M 100.6
         JogNegative             := M 100.7
         PositionDistance        := MD 102
         Velocity                := MD 106
         S_On                    := A 136.7
         S_Direction             := A 136.2
         S_AlarmReset            := A 136.6
         MinUserDistance         := MD 110
         MaxUserDistance         := MD 114
         MinUserVelocity         := MD 118
         MaxUserVelocity         := MD 122
         AxisReady               := M 101.3
         AxisEnabled             := M 101.4
         AxisError               := M 101.5
         AxisErrorID             := MW 126
         DriveError              := M 101.6
         CmdActive               := MB 128
         CmdDone                 := M 130.0
         CmdBusy                 := M 130.1
         CmdAborted              := M 130.2
         CmdError                := M 130.3
         CmdErrorID              := MW 132
```

The addresses of *S_Ready* and *S_Alarm* are derived from the addresses of the inputs which are connected to the drive's digital outputs. These can be determined via the sub module *'DI24/DO16'* of the CPU.

The addresses of *S_On*, *S_Direction* and *S_AlarmReset* are obtained from the addresses of the outputs which are connected to the digital inputs of the drive. These can be determined via the sub module *'DI24/DO16'* of the CPU.

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. Bring your CPU into RUN and turn on your drive.

⇨ The FB 875 - VMC_AxisControl_PT is executed cyclically.

3. As soon as *AxisReady* = TRUE, you can use *AxisEnable* to enable the drive.

4. You now have the possibility to control your drive via its parameters and to check its status. *Chap. 5.7.1 'FB 875 - VMC_AxisControl_PT - Axis control via Pulse Train' page 238*

Controlling the drive via HMI

There is the possibility to control your drive via an HMI. For this purpose, a predefined symbol library is available for Movicon to access the VMC_AxisControl_PT function module. *Chap. 10 'Controlling the drive via HMI' page 432*

5.6 Usage in Siemens TIA Portal

5.6.1 Precondition

Overview

- Please use the Siemens TIA Portal V 14 and up for the configuration.
- The configuration of the VIPA CPU with Pulse Train functionality happens in the Siemens TIA Portal by means of a virtual PROFINET IO device.
- The PROFINET IO Device is to be installed in the hardware catalog by means of a GSDML.

Installing the VIPA IO device

The installation of the PROFINET VIPA IO device happens in the hardware catalog with the following approach:

1. Go to the service area of www.vipa.com.
2. Download the according file for your system - here System MICRO 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 ... > VIPA MICRO PLC*.

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

5.6.2 Hardware configuration

Add CPU in the project

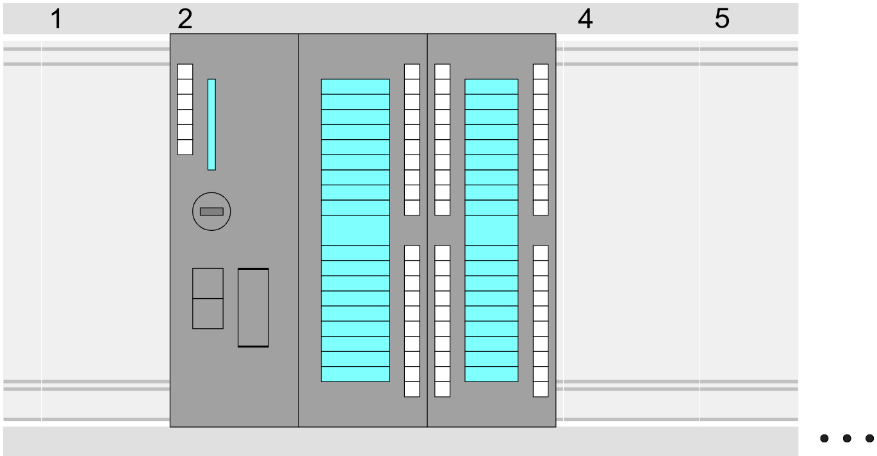
To be compatible with the Siemens SIMATIC TIA Portal the following steps should be executed:

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. Select the following CPU in the input dialog:

   SIMATIC S7-300 > CPU 314C-2 PN/DP (314-6EH04-0AB0 V3.3)

   ⇨ The CPU is inserted with a profile rail.

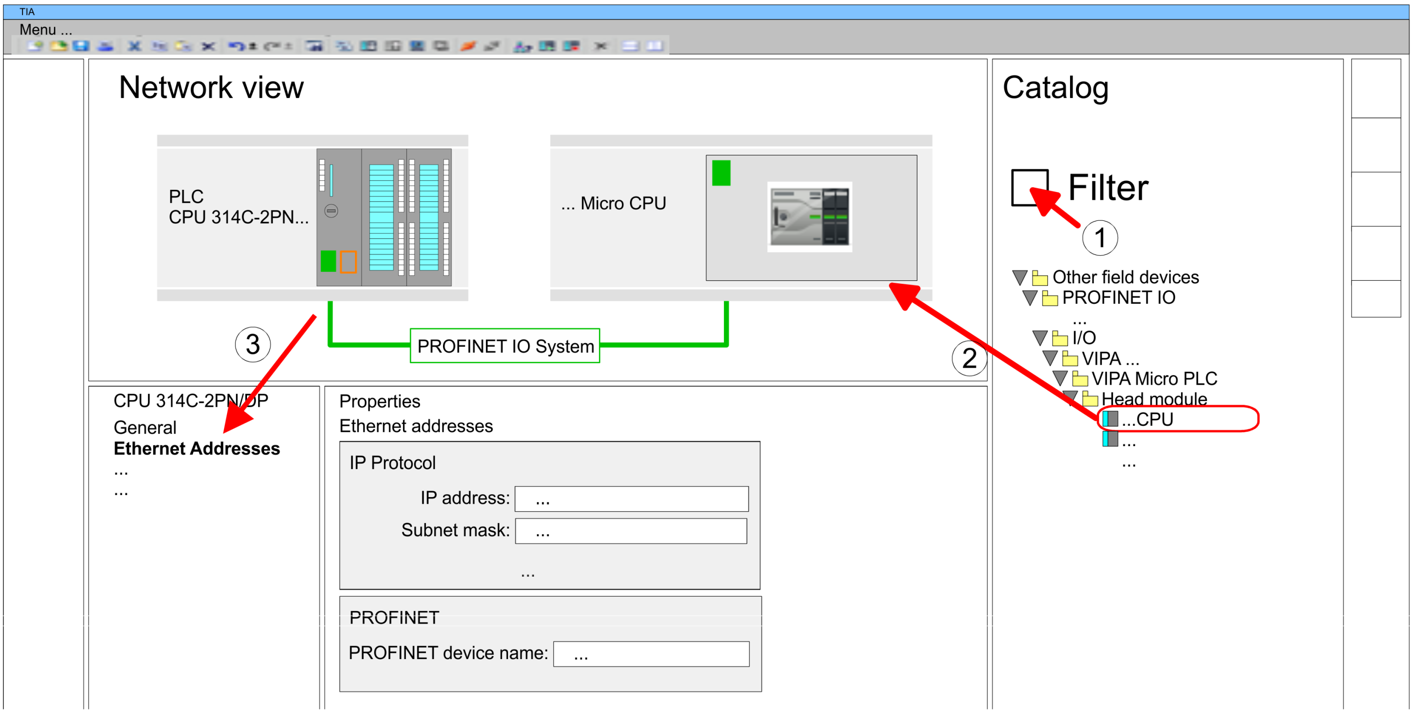


Device overview:

Module	...	Slot	...	Type	...
PLC...		2		CPU 314C-2PN/DP	
MPI interface...		2 X1		MPI/DP interface	
PROFINET interface...		2 X2		PROFINET interface	
DI24/DO16...		2 5		DI24/DO16	
AI5/AO2...		2 6		AI5/AO2	
Count...		2 7		Count	
...					

Connection CPU as PROFINET IO device

1. Switch in the *Project area* to *'Network view'*.
2. After installing the GSDML the IO device for the SLIO CPU may be found in the hardware catalog at *Other field devices > PROFINET > IO > VIPA ... > VIPA MICRO PLC*. 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 at 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.

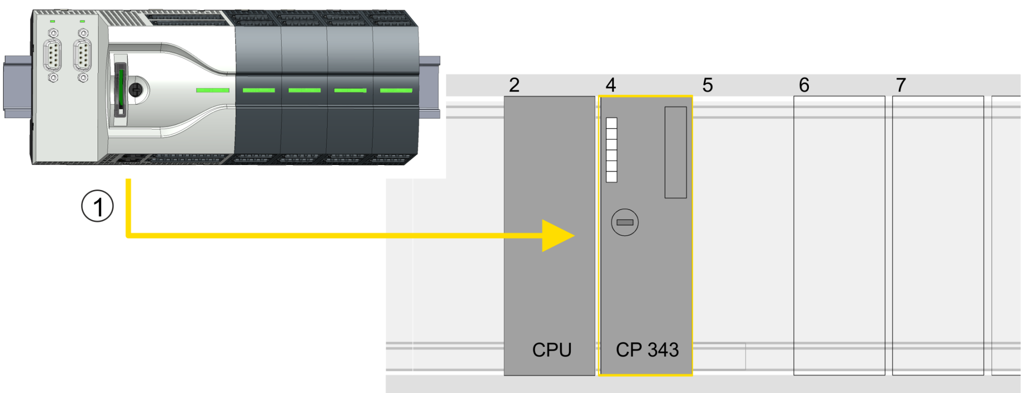


5. Select in the *Network view* the IO device *'VIPA MICRO PLC'* and switch to the *Device overview*.

   ⇨ In the *Device overview* of the PROFINET IO device *'VIPA MICRO PLC'* the CPU is already placed at slot 0.

Configuration of Ethernet PG/OP channel

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. You get valid IP address parameters from your system administrator.



1 Ethernet PG/OP channel

Device overview

Module	...	Slot	...	Type	...
PLC ...		2		CPU 314C-2PN/DP	

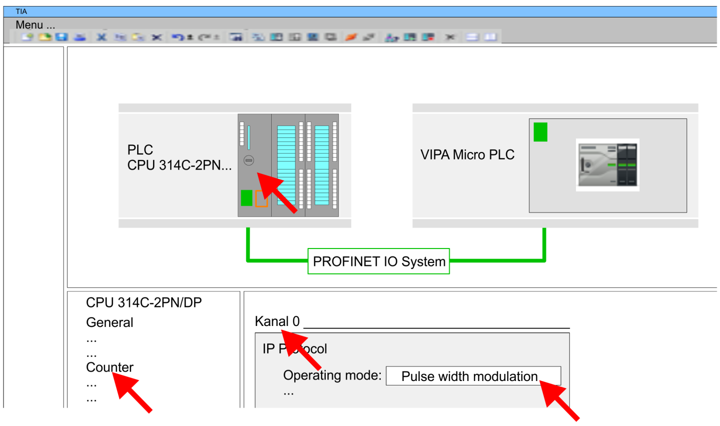
MPI/DP interface		2 X1		MPI/DP interface	
PROFINET interface		2 X2		PROFINET interface	
...		...		...	
CP 343-1		4		CP 343-1	
...		...		...	

Switch I/O periphery to Pulse Train

For parametrization of the input/output periphery and the *technological functions* the corresponding sub modules of the Siemens CPU 314C-2 PN/DP (314-6EH04-0AB0 V3.3) is to be used. For pulse train output, the sub module count must be switched to *'Pulse-width modulation'*. If you are using a channel other than channel 0, you must adapt it in the hardware configuration and in your user program.

1. Double-click the counter sub module of the CPU 314C-2 PN/DP.

   ⇨ The dialog *'Properties'* is opened.
2. For example, select *'channel 0'* and select the function *'Pulse-width modulation'* as *'Operating mode'*.
3. Leave all values unchanged.



4. Click at the CPU and select *'Context menu ➔ Compile ➔ All'*.

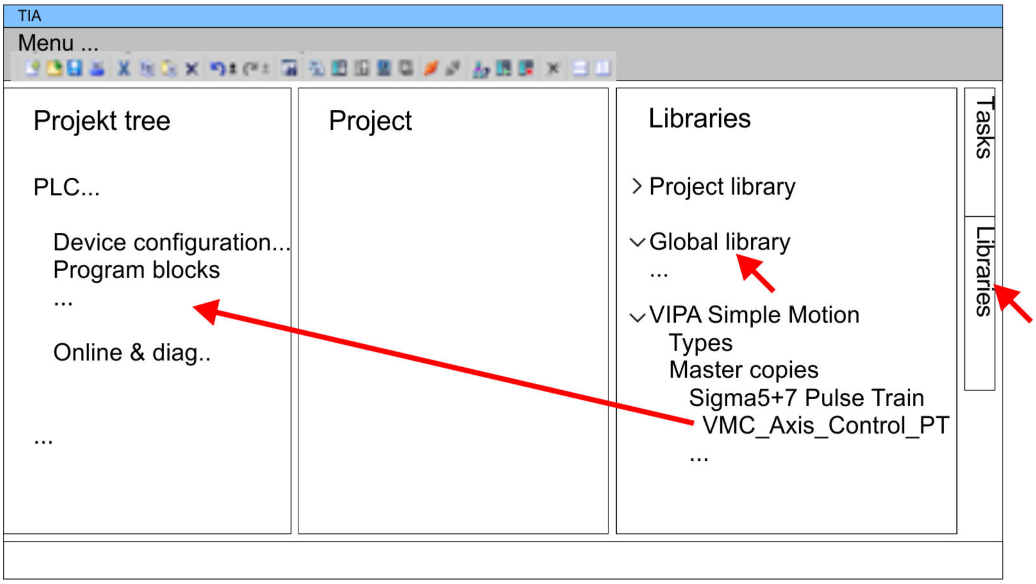
5.6.3 User program

Include library

1. Go to the service area of www.vipa.com.
2. Download the *Simple Motion Control* library from the download area at *'VIPA Lib'*.

   The library is available as packed zip file for the corresponding TIA Portal version.
3. Start your un-zip application with a double click on the file ...TIA_V*xx*.zip and copy all the files and folders in a work directory for the Siemens TIA Portal.
4. Switch to the *Project view* of the Siemens TIA Portal.
5. Choose "Libraries" from the task cards on the right side.
6. Click at "Global library".
7. Click on the free area inside the *'Global Library'* and select *'Context menu ➔ Retrieve library'*.
8. Navigate to your work directory and load the file ...Simple Motion.zal*xx*.

Copy blocks into project



Copy the following block from the library into the "Program blocks" of the *Project tree* of your project.

- *Sigma5+7* Pulse Train
  – FB 875 - VMC_AxisControl_PT ➜ *Chap. 5.7.1 'FB 875 - VMC_AxisControl_PT - Axis control via Pulse Train' page 238*

OB 1

Configuration of the axis

1. Open in the *Project tree* within the CPU at *'Programming blocks'* the OB 1 and program the Call FB 875, DB 875.
   ⇨ The dialog *'Add instance data block'* opens.
2. Set the number for the instance data block, if not already done, and close the dialog with [OK].
   ⇨ The block call is created and the parameters are listed

3. Assign the following parameters for the sample project. In particular, consider the two conversion factors *FactorPosition* and *FactorVelocity*:

⇨
```
CALL  FB    "VMC_AxisControl_PT" , "DI_AxisControl_PT"
         S_ChannelNumberPWM      := 0
         S_Ready                 := E 136.0
         S_Alarm                 := E 136.2
         FactorPosition          := 1024.0
         FactorVelocity          := 976.5625
         AxisEnable              := M 100.1
         AxisReset               := M 100.2
         StopExecute             := M 100.3
         MvVelocityExecute       := M 100.4
         MvRelativeExecute       := M 100.5
         JogPositive             := M 100.6
         JogNegative             := M 100.7
         PositionDistance        := MD 102
         Velocity                := MD 106
         S_On                    := A 136.7
         S_Direction             := A 136.2
         S_AlarmReset            := A 136.6
         MinUserDistance         := MD 110
         MaxUserDistance         := MD 114
         MinUserVelocity         := MD 118
         MaxUserVelocity         := MD 122
         AxisReady               := M 101.3
         AxisEnabled             := M 101.4
         AxisError               := M 101.5
         AxisErrorID             := MW 126
         DriveError              := M 101.6
         CmdActive               := MB 128
         CmdDone                 := M 130.0
         CmdBusy                 := M 130.1
         CmdAborted              := M 130.2
         CmdError                := M 130.3
         CmdErrorID              := MW 132
```

The addresses of *S_Ready* and *S_Alarm* are derived from the addresses of the inputs which are connected to the drive's digital outputs. These can be determined via the sub module *'DI24/DO16'* of the CPU.

The addresses of *S_On*, *S_Direction* and *S_AlarmReset* are obtained from the addresses of the outputs which are connected to the digital inputs of the drive. These can be determined via the sub module *'DI24/DO16'* of the CPU.

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. Bring your CPU into RUN and turn on your drive.

   ⇨ The FB 875 - VMC_AxisControl_PT is executed cyclically.

3. As soon as *AxisReady* = TRUE, you can use *AxisEnable* to enable the drive.

4. You now have the possibility to control your drive via its parameters and to check its status. *Chap. 5.7.1 'FB 875 - VMC_AxisControl_PT - Axis control via Pulse Train' page 238*

Controlling the drive via HMI

There is the possibility to control your drive via an HMI. For this purpose, a predefined symbol library is available for Movicon to access the VMC_AxisControl_PT function module. *Chap. 10 'Controlling the drive via HMI' page 432*

5.7 Drive specific block

5.7.1 FB 875 - VMC_AxisControl_PT - Axis control via Pulse Train

5.7.1.1 Description

With the FB *VMC_AxisControl_PT* you can control axis, which are connected via Pulse Train. You can check the status of the drive, turn the drive on or off, or execute various motion commands. A separate memory area is located in the instance data of the block. You can control your axis by means of an HMI. *Chap. 10 'Controlling the drive via HMI' page 432*

The control of a pulse train drive happens exclusively with the FB 875 VMC_AxisControl_PT. PLCopen blocks are not supported!

Parameter

Parameter	Declaration	Data type	Description
S_Channel-NumberPWM	INPUT	INT	Channel number of the PWM output, which is used for the control of the Pulse Train input of the servo (signal PULS).
S_Ready	INPUT	BOOL	■ Digital input for connecting the S_Ready signal (S-RDY) – TRUE: Servo is ready for the S_On signal.
S_Alarm	INPUT	BOOL	■ Digital input for connecting the S_Alarm signal (ALM) – FALSE if the servo has detected an error.
FactorPosition	INPUT	REAL	Factor for converting the position of user units into drive units (increments) and back. *'FactorPosition' page 241*
FactorVelocity	INPUT	REAL	Factor for converting the velocity of user units into drive units (increments) and back. *'FactorVelocity' page 242*
AxisEnable	INPUT	BOOL	■ Enable/disable axis – TRUE: The axis is enabled. – FALSE: The axis is disabled.
AxisReset	INPUT	BOOL	■ Reset axis – Edge 0-1: Axis reset is performed. – The status of a reset, started with *AxisReset*, is not indicated at the outputs *CmdActive*, *CmdDone*, *CmdBusy*, *CmdAborted*, *CmdError* and *CmdErrorID*.
StopExecute	INPUT	BOOL	■ Stop axis – Edge 0-1: Stopping of the axis is started. Note: StopExecute = 1: No other command can be started!

Parameter	Declaration	Data type	Description
MvVelocityExecute	INPUT	BOOL	■ Start moving the axis – Edge 0-1: The axis is accelerated / decelerated to the speed specified.
MvRelativeExecute	INPUT	BOOL	■ Start moving the axis – Edge 0-1: The relative positioning of the axis is started.
JogPositive	INPUT	BOOL	Jog operation positive ■ Drive axis with constant velocity in positive direction – Edge 0-1: Drive axis with constant velocity is started. – Edge 1-0: The axis is stopped.
JogNegative	INPUT	BOOL	Jog operation negative ■ Drive axis with constant velocity in negative direction – Edge 0-1: Drive axis with constant velocity is started. – Edge 1-0: The axis is stopped.
PositionDistance	INPUT	REAL	Absolute position or relative distance for *MvRelativeExecute* in [user units].
Velocity	INPUT	REAL	Velocity setting (signed value) in [user units / s].
S_ON	OUTPUT	BOOL	■ Digital output for controlling the S_On signal (S-ON) – TRUE: turns on the servo. – TRUE: turns off the servo.
S_Direction	OUTPUT	BOOL	■ Digital output for controlling the S_Direction signal (SIGN) – TRUE: Presetting of the direction of rotation positive direction for the servo. – FALSE: Presetting of the direction of rotation negative direction for the servo.
S_AlarmReset	OUTPUT	BOOL	■ Digital output for controlling the S_AlarmReset signal (ALM-RST) – TRUE: Alarms are reset in the servo. – FALSE: Alarms in the servo remain.
MinUserDistance	OUTPUT	REAL	Minimum drive distance (1 increment) of the servo [user units].
MinUserDistance	OUTPUT	REAL	Maximum drive distance (8388607 increments = maximum number of pulses of the PWM output) of the servo [user units].
MinUserVelocity	OUTPUT	REAL	Minimum speed (period duration = 65535µs = maximum period of the PWM output) of the servo [user units].
MinUserVelocity	OUTPUT	REAL	Maximum speed (period duration = 20µs = minimum period duration of the PWM output) of the servo [user units].
AxisReady	OUTPUT	BOOL	■ AxisReady – TRUE: The axis is ready to switch on. – FALSE: The axis is not ready to switch on. → Check and fix *AxisError* (see *AxisErrorID*). → Check and fix *DriveError*.

Parameter	Declaration	Data type	Description
AxisEnabled	OUTPUT	BOOL	■ Status axis – TRUE: Axis is switched on and accepts motion commands. – FALSE: Axis is not switched on and does not accepts motion commands. ■ Conditions for *AxisEnabled* = TRUE – *AxisEnable* = TRUE – *S_Ready* = TRUE – *S_Alarm* = TRUE
AxisError	OUTPUT	BOOL	■ Motion axis error – TRUE: An error has occurred. Additional error information can be found in the parameter *AxisErrorID*. → The axis is locked (*S_On* = FALSE and *AxisEnabled* = FALSE). Command is not executed.
AxisErrorID	OUTPUT	WORD	Additional error information ↳ *Chap. 12 'ErrorID - Additional error information' page 457*
DriveError	OUTPUT	BOOL	■ Error on the drive – TRUE: An error has occurred. – → The axis is disabled.
CmdActive	OUTPUT	BYTE	■ Command – 0: no Cmd active – 1: STOP – 2: MvVelocity – 3: MvRelative – 4: JogPos – 5: JogNeg
CmdDone	OUTPUT	BOOL	■ Status Done – TRUE: Job ended without error.
CmdBusy	OUTPUT	BOOL	■ Status busy – TRUE: Job is running.
CmdAborted	OUTPUT	BOOL	■ Status Aborted – TRUE: The job was aborted during processing by another job. Note: *CmdAborted* is reset when a Cmd is started
CmdError	OUTPUT	BOOL	■ Status Error – TRUE: An error has occurred. The axis is disabled Additional error information can be found in the parameter *CmdErrorID*.
CmdErrorID	OUTPUT	WORD	Additional error information ↳ *Chap. 12 'ErrorID - Additional error information' page 457*

5.7.1.2 Conversion factors

FactorPosition

The calculation of FactorPosition is only valid if servo parameter Reference Pulse Multiplier (Pn218) = 1.

$$FactorPosition = \frac{Resolution}{Numerator} \cdot Denominator$$

FactorPosition - Factor for converting the position of user units into drive units (increments) and back.

Resolution - Number of increments per user unit

Numerator - Numerator: Electronic Gear Ratio (Pn20E) of the servo parameter

Denominator - Denominator: Electronic Gear Ratio (Pn210) of the servo parameter

Example User unit for position = 1 revolution

FactorPosition - Factor for converting the position of user units into drive units (increments) and back.

Resolution - Number of increments per user unit

Resolution = 2^{20} = 1048576

Numerator - Numerator: Electronic Gear Ratio (Pn20E) of the servo parameter

Numerator = 1024

Denominator - Denominator: Electronic Gear Ratio (Pn210) of the servo parameter

Denominator = 1

$$FactorPosition = \frac{Resolution}{Numerator} \cdot Denominator$$

$$FactorPosition = \frac{1048576}{1024} \cdot 1 = 1024$$

Example minimum distance

MinPos - Minimum distance in rotations

Resolution - Number of increments per user unit

Resolution = 2^{20} = 1048576

Numerator - Numerator: Electronic Gear Ratio (Pn20E) of the servo parameter

Numerator = 1024

Period - Minimum period

Period = 1

$$MinPos = Numerator \cdot \frac{Period}{Resolutioon}$$

$$MinPos = 1024 \cdot \frac{1}{1048576} = \frac{1}{1024}$$

Example maximum distance

MaxPos - Maximum distance in revolutions

Resolution - Number of increments per user unit

Resolution = 2^{20} = 1048576

Numerator - Numerator: Electronic Gear Ratio (Pn20E) of the servo parameter

Numerator = 1024

Period - Maximum period

Period = 8388607

$$MaxPos = Numerator \cdot \frac{Period}{Resolution}$$

$$MaxPos = 1024 \cdot \frac{8388607}{1048576} = 8192$$

FactorVelocity

> *The calculation of FactorVelocity is only valid if servo parameter Reference Pulse Multiplier (Pn218) = 1.*

$$FactorVelocity = Time \cdot \frac{\frac{Numerator}{Denominator}}{Resolution}$$

Time - Time for 1 revolution in µs

Numerator - Numerator: Electronic Gear Ratio (Pn20E) of the servo parameter

Denominator - Denominator: Electronic Gear Ratio (Pn210) of the servo parameter

Resolution - Number of increments per user unit

Example User unit for velocity = revolution/min

FactorVelocity - Factor for converting of user units into drive units (increments) and back.

Time - Time for 1 revolution in µs

Time = 1min = $60 \cdot 10^6$µs

Numerator - Numerator: Electronic Gear Ratio (Pn20E) of the servo parameter

Numerator = 1024

Denominator - Denominator: Electronic Gear Ratio (Pn210) of the servo parameter

Denominator = 1

Resolution - Number of increments per user unit

Resolution = 2^{20} = 1048576

$$FactorVelocity = Time \cdot \frac{\frac{Numerator}{Denominator}}{Resolution}$$

$$FactorVelocity = 60 \cdot 10^6 \frac{\frac{1024}{1}}{1048576} = \frac{60 \cdot 10^6}{1024} = 58593{,}75$$

Example User unit for velocity = revolution/s

FactorVelocity - Factor for converting of user units into drive units (increments) and back.

Time - Time for 1 revolution in µs

Time = 1s = 10^6µs

Numerator - Numerator: Electronic Gear Ratio (Pn20E) of the servo parameter

Numerator = 1024

Denominator - Denominator: Electronic Gear Ratio (Pn210) of the servo parameter

Denominator = 1

Resolution - Number of increments per user unit

Resolution = 2^{20} = 1048576

$$FactorVelocity = Time \cdot \frac{\frac{Numerator}{Denominator}}{Resolution}$$

$$FactorVelocity = 10^6 \frac{\frac{1024}{1}}{1048576} = \frac{10^6}{1024} = 976{,}5625$$

Minimum velocity for revolutions/min

MinVel - Minimum velocity in revolutions/min

FactorVelocity - Factor for converting of user units into drive units (increments) and back.

$$MinVel = \frac{FactorVelocity}{65535} = \frac{58593{,}75}{65535} = 0{,}89$$

Maximum velocity for revolutions/min

MaxVel - Maximum velocity in revolutions/min

FactorVelocity - Factor for converting of user units into drive units (increments) and back.

$$MaxVel = \frac{FactorVelocity}{20} = \frac{58593{,}75}{20} = 2929{,}69$$

5.7.1.3 Functionality

Switch the drive on or off

- The *AxisEnable* input is used to switch an axis on or off.
- Switching on is only possible if *AxisReady* = TRUE, i.e. the axis is ready to switch on.
- As soon as the axis is switched on, this is indicated by the status information *AxisEnabled*.
- If the axis has an error, this is indicated by the status information *AxisError*. For more information refer to *AxisErrorID*.

Please note that you always have to call the block within OB 1, otherwise you will get the error message 0x8317.

Behavior of the outputs *CmdActive*, *CmdDone* and *CmdBusy*

The command processing can be divided into 3 phases. Depending on the operating mode, the outputs *CmdActive*, *CmdDone* and *CmdBusy* show the following behavior within these phases:

Velocity control with *Velocity* <> 0

- Phase 1: The command is started with edge 0-1 at *MvVelocityExecute*.
  – *CmdActive* = 2, *CmdDone* = FALSE, *CmdBusy* = TRUE
- Phase 2: The preset velocity was reached, *MvVelocityExecute* = TRUE
  – Command is still running.
  – *CmdActive* = 2, *CmdDone* = TRUE, *CmdBusy* = FALSE
- Phase 3: *MvVelocityExecute* = FALSE
  – Command is still running.
  – *CmdActive* = 2, *CmdDone* = FALSE, *CmdBusy* = FALSE

Velocity control with *Velocity* = 0

- Phase 1: The command is started with edge 0-1 at *MvVelocityExecute*.
  - *CmdActive* = 2, *CmdDone* = FALSE, *CmdBusy* = TRUE
- Phase 2: The velocity 0 was reached, *MvVelocityExecute* = TRUE
  - Axis stands still and is ready for further commands.
  - *CmdActive* = 0, *CmdDone* = TRUE, *CmdBusy* = FALSE
- Phase 3: *MvVelocityExecute* = FALSE
  - Axis stands still and is ready for further commands.
  - *CmdActive* = 0, *CmdDone* = FALSE, *CmdBusy* = FALSE

Stop axis

- Phase 1: The command is started with edge 0-1 at *StopExecute*.
  - *CmdActive* = 1, *CmdDone* = FALSE, *CmdBusy* = TRUE
- Phase 2: The velocity 0 was reached, *StopExecute* = TRUE
  - Axis stands still and stop command prevents the execution of further commands.
  - *CmdActive* = 1, *CmdDone* = TRUE, *CmdBusy* = FALSE
- Phase 3: *StopExecute* = FALSE
  - Axis stands still and is ready for further commands.
  - *CmdActive* = 0, *CmdDone* = FALSE, *CmdBusy* = FALSE

Relative positioning

- Phase 1: The command is started with edge 0-1 at *MvRelativeExecute*.
  - *CmdActive* = 3, *CmdDone* = FALSE, *CmdBusy* = TRUE
- Phase 2: The position target was reached, *MvRelativeExecute* = TRUE
  - No command active
  - *CmdActive* = 0, *CmdDone* = TRUE, *CmdBusy* = FALSE
- Phase 3: *MvRelativeExecute* = FALSE
  - *CmdActive* = 0, *CmdDone* = FALSE, *CmdBusy* = FALSE

Jog mode

- Phase 1: The command is started with edge 0-1 at *JogPositive* respectively *JogNegative*.
  - *CmdActive* = 4 respectively 5, *CmdDone* = FALSE, *CmdBusy* = TRUE
- Phase 2: The preset velocity was reached, *JogPositive* = TRUE respectively *JogNegative* = TRUE.
  - Command is still active, axis is only stopped with *JogPositive* = FALSE respectively *JogNegative* = FALSE.
  - *CmdActive* = 4 respectively 5, *CmdDone* = TRUE, *CmdBusy* = FALSE
- Phase 3: *JogPositive* = FALSE respectively *JogNegative* = FALSE
  - Axis stands still and is ready for further commands.
  - *CmdActive* = 0, *CmdDone* = FALSE, *CmdBusy* = FALSE

Acknowledge drive errors

- With *AxisReset* you can acknowledge errors on the drive.
- Errors are reported via *DriveError*.

Stop axis - MC_STOP

- You can stop an axis in motion by setting *StopExecute*.
- As long as *StopExecute* is set, no further pulses are generated and all commands are blocked.

Velocity mode - MC_MoveVelocity

- Precondition: The drive is switched on and *AxisReady* = TRUE.
- With *MvVelocityExecute*, you can bring the axis to rotate with constant velocity.
- You specify the velocity via *Velocity*.
- By setting 0, the axis stops as well as with *StopExecute*.

- The direction of rotation is determined by the sign of *Velocity*.
- The *Velocity* value can be 0 or *MinUserVelocity* ≤ *Velocity* ≤ *MaxUserVelocity*.

> *Due to the system the current velocity may deviate from the setpoint velocity. The deviation MaxVelError increases with increasing velocity and can be determined with the following formula.*

$$MaxVelError = \frac{FactorVelocity}{20} - \frac{FactorVelocity}{21}$$

Relative positioning - MC_MoveRelative

- Precondition: The drive is switched on and *AxisReady* = TRUE.
- The relative positioning happens by *MvRelativeExecute*.
- You can specify the distance in user units via *PositionDistance*.
- The direction of rotation is determined by the sign of *PositionDistance*.
- You specify the velocity via *Velocity*.
- By setting *StopExecute*, you can stop a running command.

Jog mode

- Precondition: The drive is switched on and *AxisReady* = TRUE.
- With an edge 0-1 at *JogPositive* or *JogNegative*, you can control your drive in jog mode. In this case, a jogging command is executed in the corresponding direction of rotation.
- You specify the velocity via *Velocity*. The sign is not relevant.
- With an edge 1-0 at *JogPositive* or *JogNegative* respectively by setting *StopExecute* the axis is stopped.

> *Please note that you receive an error message (0x8003) in jog mode at Velocity = 0!*

6 Usage inverter drive via PWM

6.1 Overview

Precondition

- SPEED7 Studio from V1.7.1
  or
- Siemens SIMATIC Manager from V 5.5, SP2 & *Simple Motion Control Library*
  or
- Siemens TIA Portal V 14 & *Simple Motion Control Library*
- System MICRO or System SLIO CPU with PWM output, such as CPU M13-CCF0000 or CPU 013-CCF0R00.
- Inverter drive with PWM input e.g. *V1000*.

Steps of configuration

1. Setting parameters on the inverter drive
   - The setting of the parameters happens by means of the software tool *Drive Wizard+*.
2. Hardware configuration in the VIPA *SPEED7 Studio*, Siemens SIMATIC Manager or Siemens TIA Portal.
   - Configuring the CPU.
3. Programming in the VIPA *SPEED7 Studio*, Siemens SIMATIC Manager or Siemens TIA Portal.
   - *VMC_AxisControlV1000PWM* block for configuration and communication with the axis, which is connected via PWM.

6.2 Set the parameters on the inverter drive

CAUTION!
Before the commissioning, you have to adapt your inverter drive to your application with the *Drive Wizard+* software tool! More may be found in the manual of your drive.

The following table shows all parameters, which do not correspond to the default values. The following parameters must be set via *Drive Wizard+* to match the *Simple Motion Control Library*. This is followed by a table with parameters, which can be adapted as a function of the application.

No.	Parameters that differ from the standard	Setting for *Simple Motion Control Library*
B1-01	Reference selection	■ 4: Pulse train input
B1-02	Operation method selection	■ 1: Control circuit terminal
H1-01	Terminal S1 function selection	■ 0040: Forward Run Command
H1-02	Terminal S2 function selection	■ 0041: Reverse Run Command
H2-01	Terminal MA/MB-MC selection	■ 000E: Fault
H2-02	P1 terminal selection	■ 0006
H6-01	Pulse train input function selection	■ 0: Frequency reference
H6-02	Pulse train input scaling	■ 20000Hz
H6-03	Pulse train input gain	■ 100.0%

No.	Parameters that differ from the standard	Setting for *Simple Motion Control Library*
H6-04	Pulse train input bias	■ 0.0%
H6-05	Pulse train input filter time	■ 0.10s
H6-06	Pulse train monitor selection	■ 102: Output frequency
H6-07	Pulse train monitor scaling	■ 20000Hz

No.	Parameters depending on the application	Example
C1-01	Acceleration time 1	■ 10.00s
C1-02	Deceleration time 1	■ 10.00s
C1-10	Accel/Decel time setting unit	■ 0: 0.01- second units
C1-11	Accel/Decel switching frequency	■ 0.0Hz
O1-02	Monitor selection after power up	■ 1: Frequency reference
O1-03	Display scaling	■ 2: min-1 unit

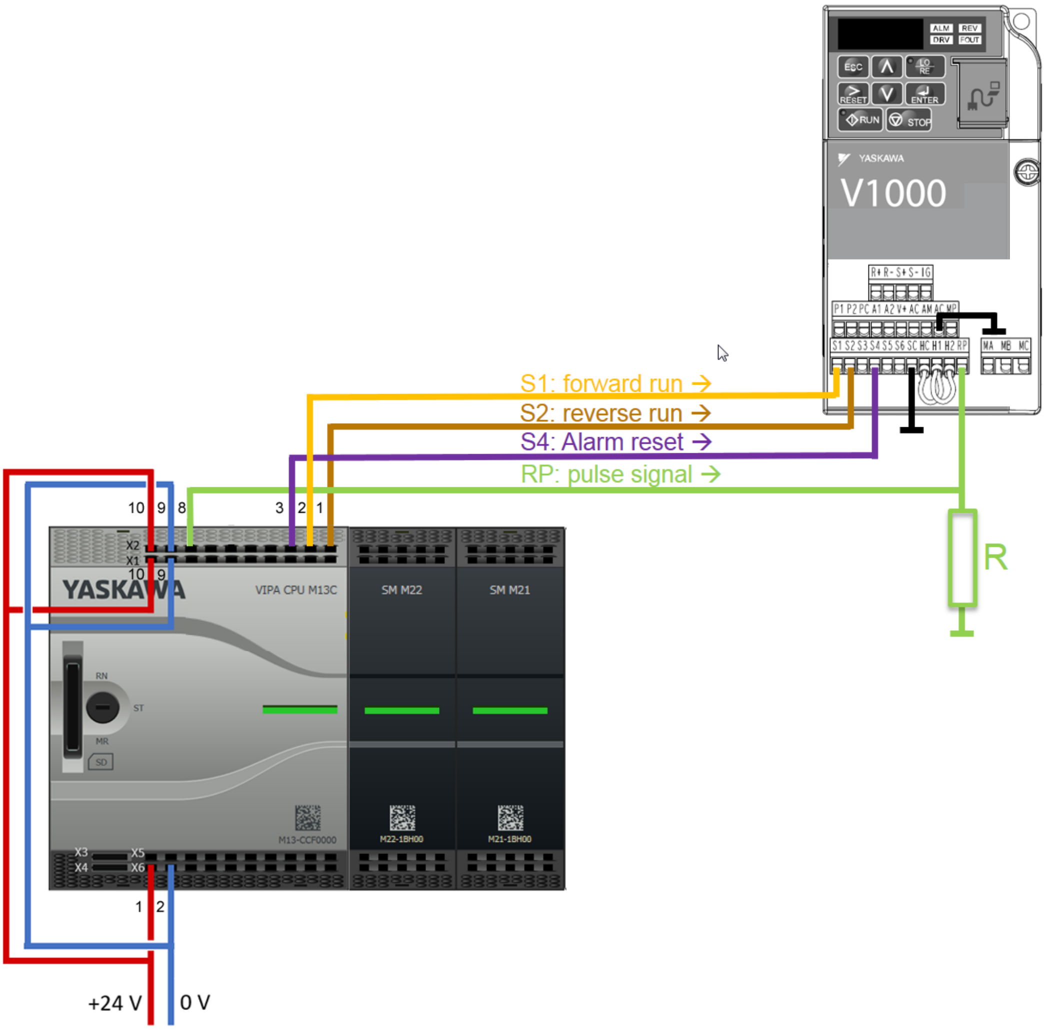
For all settings to be accepted, you must restart the inverter drive after parametrization!

6.3 Wiring

6.3.1 Connecting the *V1000* inputs

Sample application

The following figure shows an example application for connecting the inputs of a V1000 inverter drive via PWM to a System MICRO CPU M13C. In this example the PWM channel 0 (X2 - pin 8) is connected. Please use X2 - pin 7 to connect to channel 1.

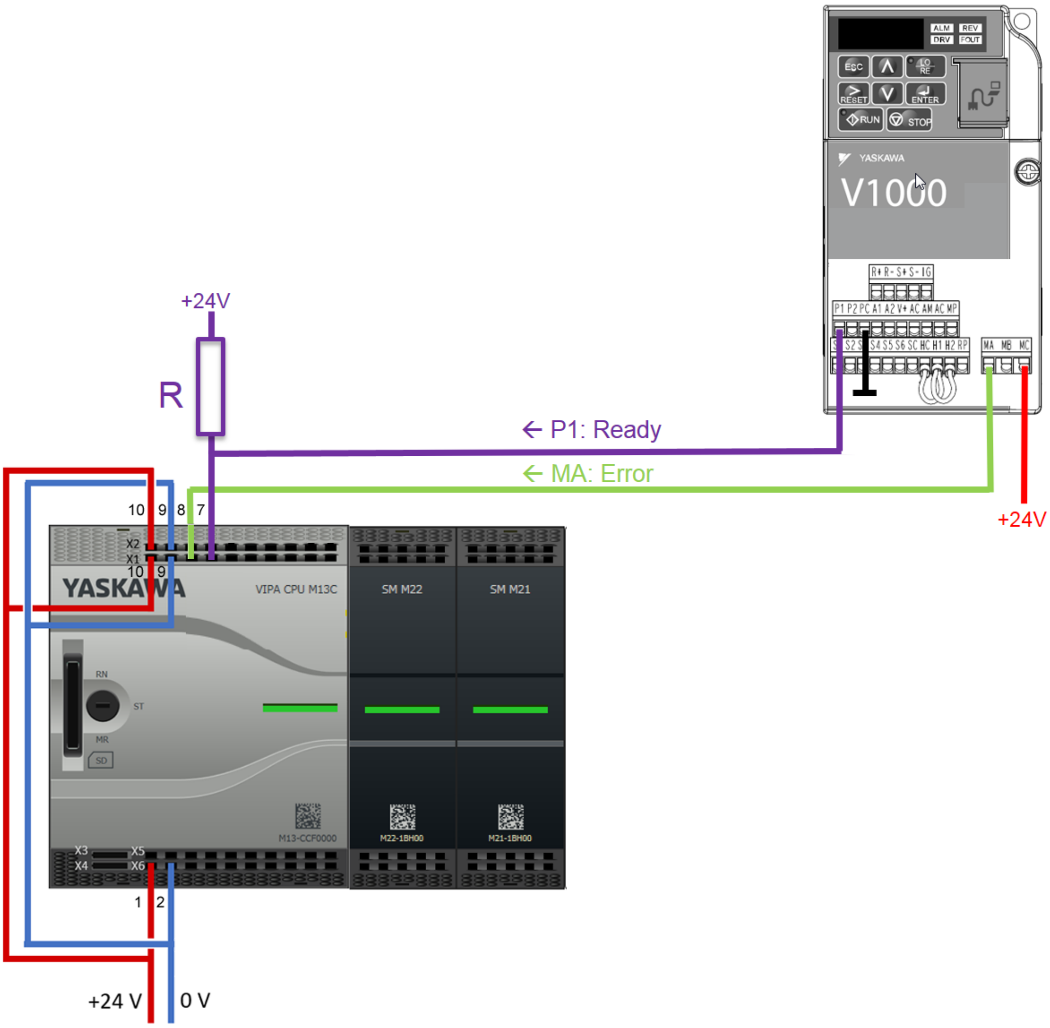


R Resistor
Value: max. 470Ω
Power dissipation: min. 0.6W
Resistance example: Metal film resistor 0207 wired with 0.6W power dissipation
Cable length max. 20m

6.3.2 Connecting the *V1000* outputs

Sample application

The following figure shows an example application for connecting the outputs of a V1000 inverter drive to a System MICRO CPU M13C.



R Resistor
Value: 4.7kΩ
Power dissipation: min. 0.25W
Resistance example: Carbon film resistor 0207 wired with 0.25W power dissipation

6.4 Usage in VIPA *SPEED7 Studio*

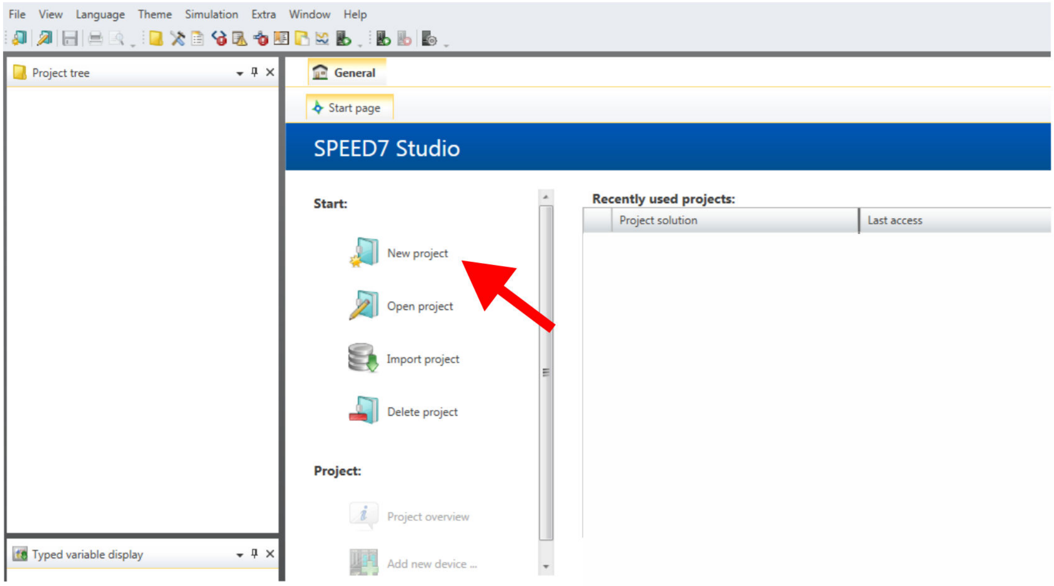
6.4.1 Hardware configuration

Add CPU in the project

Please use the *SPEED7 Studio* V1.7.1 and up for the configuration.

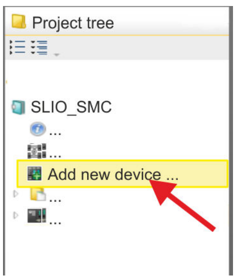
If you are using a channel other than channel 0, you must adapt it in the hardware configuration and in your user program.

1. Start the *SPEED7 Studio*.

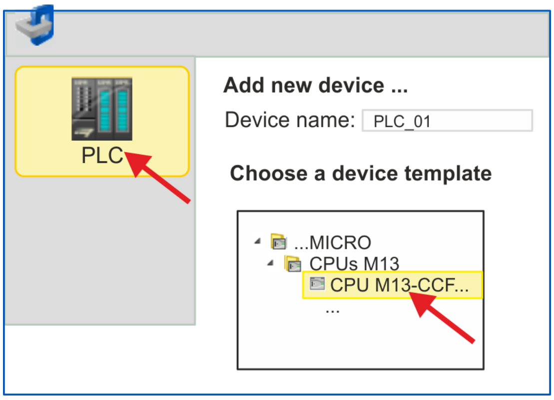


2. Create a new project at the start page with *'New project'* and assign a *'Project name'*.

⇨ A new project is created and the view *'Devices and networking'* is shown.



3. Click in the *Project tree* at *'Add new device ...'*.



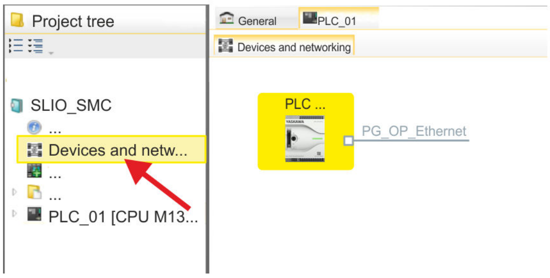
⇨ A dialog for device selection opens.

4. Select from the *'Device templates'* your CPU with PWM functionality like the System MICRO CPU M13-CCF0000 and click at [OK].

⇨ The CPU is inserted in *'Devices and networking'* and the *'Device configuration'* is opened.

Configuration of Ethernet PG/OP channel

1. Click in the *Project tree* at *'Devices and networking'*.
   ⇨ You will get a graphical object view of your CPU.



2. Click at the network *'PG_OP_Ethernet'*.
3. Select *'Context menu ➔ Interface properties'*.
   ⇨ A dialog window opens. Here you can enter the IP address data for your Ethernet PG/OP channel. You get valid IP address parameters from your system administrator.
4. Confirm with [OK].
   ⇨ The IP address data are stored in your project listed in *'Devices and networking'* at *'Local components'*.

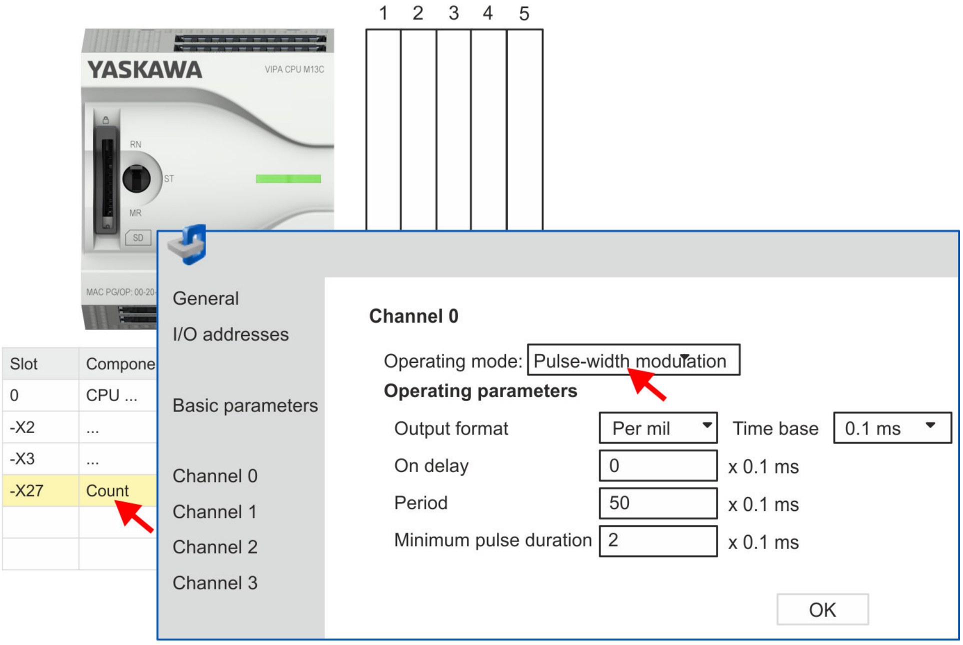
   After transferring your project your CPU can be accessed via Ethernet PG/OP channel with the set IP address data.

Switch I/O periphery to PWM

For parametrization of the I/O periphery and the *technological functions* the corresponding sub modules of the CPU are to be used. For PWM output, the sub module count must be switched to *'Pulse-width modulation'*.

1. Click in the *Project tree* at *'PLC... > Device configuration'*.
2. Click in the *'Device configuration'* at *'-X27 Count'* and select *'Context menu ➔ Components properties'*.
   ⇨ The properties dialog is opened.
3. For example, select *'channel 0'* and select the function *'Pulse-width modulation'* as *'Operating mode'*.

4. The operating parameters required for PWM are internally adapted to the corresponding values. Leave all values unchanged.

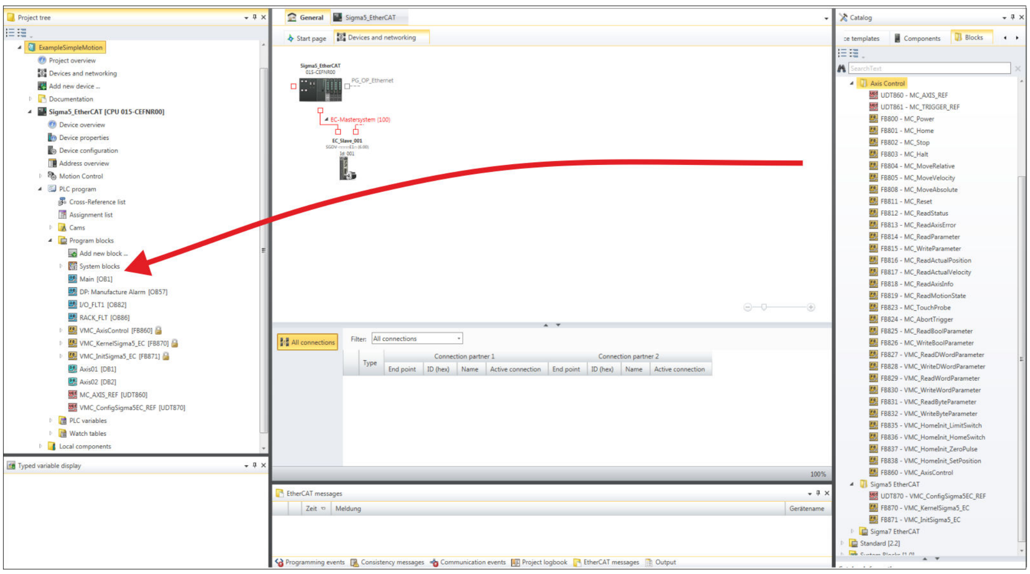


5. Close the dialog with [OK].

6. Select *'Project ➔ Compile all'*.

6.4.2 User program

Copy block to project



In the *'Catalog'*, open the *'Simple Motion Control'* library at *'Blocks'* and drag and drop the following blocks into *'Program blocks'* of the *Project tree*:

- V1000 PWM
  - FB885 – VMC_AxisControlV1000PWM *Chap. 6.7.1 'FB 885 - VMC_AxisControlV1000_PWM - Axis control over PWM' page 266*

OB 1

Configuration of the axis

If you are using a channel other than channel 0, you must adapt it in the hardware configuration and in your user program.

1. Open in *'Project tree ➔ ...CPU... ➔ PLC program ➔ Program blocks'* the OB 1 and program the Call FB 885, DB 885.

   ⇨ The dialog *'Add instance data block'* opens.

2. Set the number for the instance data block, if not already done, and close the dialog with [OK].

   ⇨ The block call is created and the parameters are listed.

3. Assign the following parameters for the sample project.

   ⇨
```
CALL  FB  "VMC_AxisControlV1000PWM" ,
"VMC_AxisCtrlV1000PWM_885"
       I_ChannelNumberPWM   :="Ax1_I_ChannelNumberPWM"
       I_MA_Alarm           :="Ax1_MA_Alarm"
       I_P1_Ready           :="I_P1_Ready"
       MaxVelocityDrive     :=1.000000e+002
       AxisEnable           :="Ax1_AxisEnable"
       AxisReset            :="Ax1_AxisReset"
       StopExecute          :="Ax1_StopExecute"
       MvVelocityExecute    :="Ax1_MvVelExecute"
       JogPositive          :="Ax1_JogPositive"
       JogNegative          :="Ax1_JogNegative"
       Velocity             :="Ax1_Velocity"
       I_S1_ForwardRun      :="Ax1_S1_ForwardRun"
       I_S2_ReverseRun      :="Ax1_S2_ReverseRun"
       I_S4_AlarmReset      :="Ax1_S4_AlarmReset"
       MinUserVelocity      :="Ax1_MinUserVelocity"
       MaxUserVelocity      :="Ax1_MaxUserVelocity"
       AxisReady            :="Ax1_AxisReady"
       AxisEnabled          :="Ax1_AxisEnabled"
       AxisError            :="Ax1_AxisError"
       AxisErrorID          :="Ax1_AxisErrorID"
       DriveError           :="Ax1_DriveError"
       CmdActive            :="Ax1_CmdActive"
       CmdDone              :="Ax1_CmdDone"
       CmdBusy              :="Ax1_CmdBusy"
       CmdAborted           :="Ax1_CmdAborted"
       CmdError             :="Ax1_CmdError"
       CmdErrorID           :="Ax1_CmdErrorID"
```

   The addresses of *I_P1_Ready* and *I_MA_Alarm* are derived from the addresses of the inputs which are connected to the digital outputs of the drive. These can be determined via the sub module *'-X25 DI/DIO'* of the CPU.

   The addresses of *I_S1_ForwardRun*, *I_S2_ReverseRun* and *I_S4_AlarmReset* are obtained from the addresses of the outputs which are connected to the digital inputs of the drive. These can be determined via the sub module *'-X25 DI/DIO'* of the CPU.

Sequence of operations

1. Select *'Project ➔ Compile all'* and transfer the project into your CPU.

   You can find more information on the transfer of your project in the online help of the *SPEED7 Studio*.

   ⇨ You can take your application into operation now.

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

2. Bring your CPU into RUN and turn on your drive.

   ⇨ The FB 885 - VMC_AxisControlV1000PWM is executed cyclically.

3. As soon as *AxisReady* = TRUE, you can use *AxisEnable* to enable the drive.
4. You now have the possibility to control your drive via its parameters and to check its status. *Chap. 6.7.1 'FB 885 - VMC_AxisControlV1000_PWM - Axis control over PWM' page 266*

6.5 Usage in Siemens SIMATIC Manager

6.5.1 Precondition

Overview

- Please use for configuration the Siemens SIMATIC Manager V 5.5 SP2 and up.
- The configuration of the VIPA CPU with PWM functionality happens in the Siemens SIMATIC Manager by means of a virtual PROFINET IO device.
- The PROFINET IO Device is to be installed in the hardware catalog by means of a GSDML.

Installing the VIPA IO device

The installation of the PROFINET VIPA IO device 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 according PROFINET IO device can be found at *'PROFINET IO ➔ Additional field devices ➔ I/O ➔ VIPA ...'*.

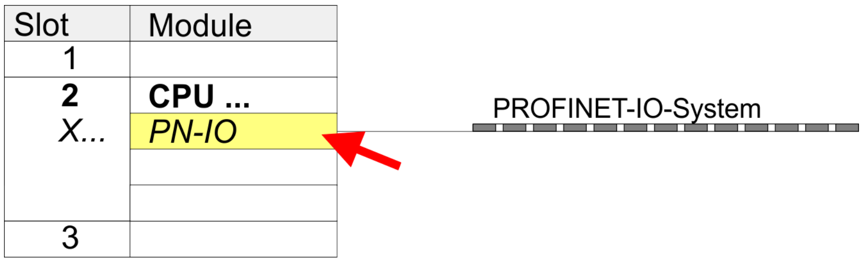
6.5.2 Hardware configuration

Add CPU in the project

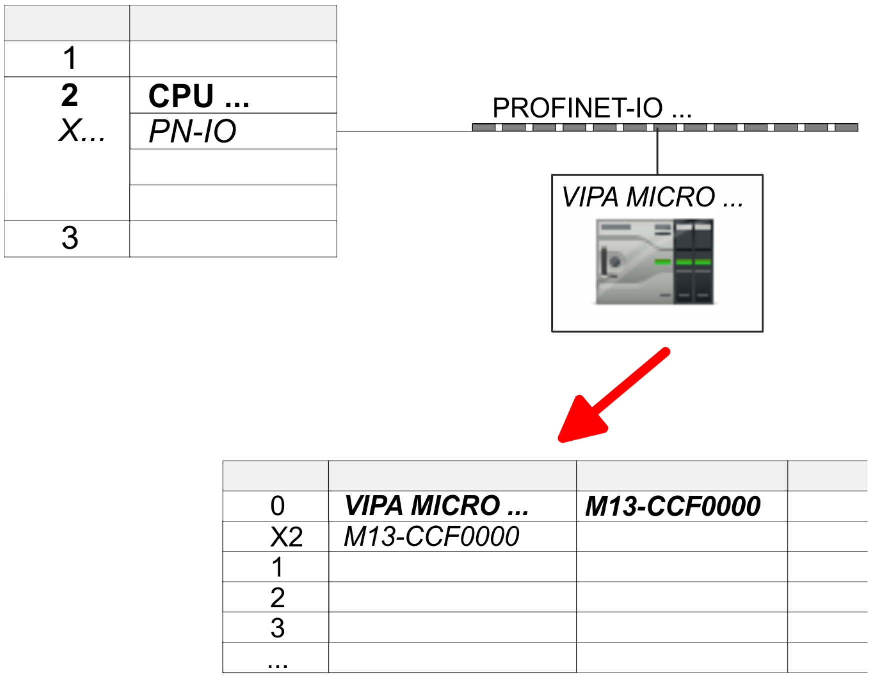
Slot	Module
1	
2	**CPU 314C-2PN/DP**
X1	*MPI/DP*
X2	*PN-IO*
X2...	*Port 1*
X2...	*Port 2*
...	...
3	

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 314C-2 PN/DP (314-6EH04-0AB0 V3.3).
4. Click at the sub module *'PN-IO'* of the CPU.
5. Select *'Context menu → Insert PROFINET IO System'*.



6. Create with [New] a new sub net and assign valid address data.
7. Click at the sub module *'PN-IO'* of the CPU and open with *'Context menu → Properties'* the properties dialog.
8. Enter at *'General'* a *'Device name'*. The device name must be unique at the Ethernet subnet.



9. Navigate in the hardware catalog to the directory *'PROFINET IO ➔ Additional field devices ➔ I/O ➔ VIPA ...'* and connect e.g. for the System MICRO the IO device *'M13-CCF0000'* to your PROFINET system.

⇨ In the *Device overview* of the PROFINET IO device *'VIPA MICRO PLC'* the CPU is already placed at slot 0.

Configuration of Ethernet PG/OP channel

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!

Switch I/O periphery to PWM

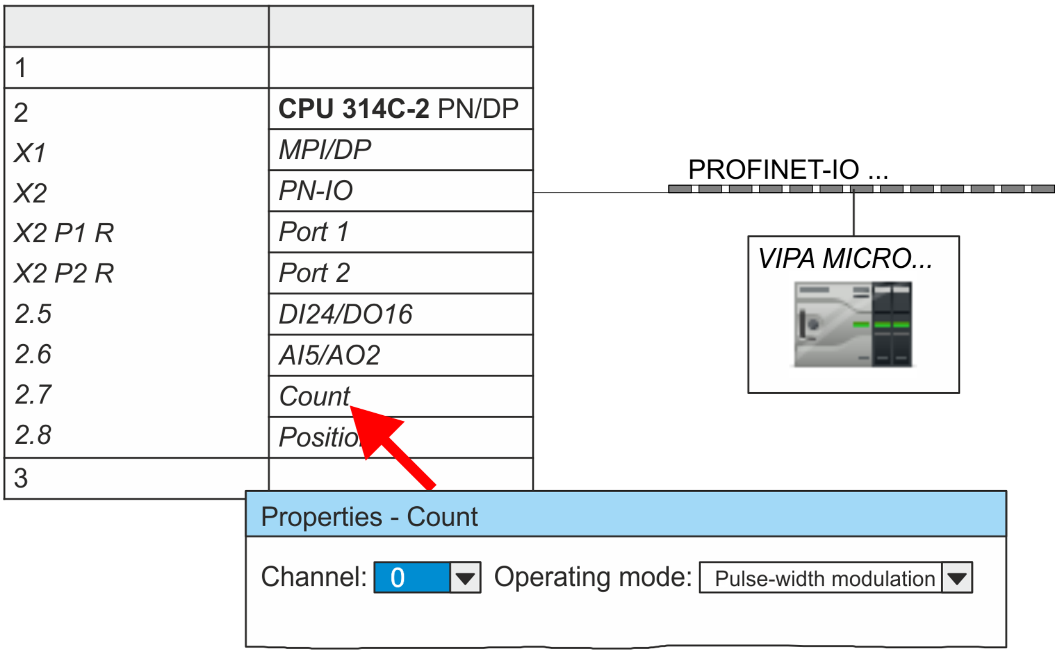
For parametrization of the input/output periphery and the *technological functions* the corresponding sub modules of the Siemens CPU 314C-2 PN/DP (314-6EH04-0AB0 V3.3) is to be used. For PWM output, the sub module count must be switched to *'Pulse-width modulation'*. If you are using a channel other than channel 0, you must adapt it in the hardware configuration and in your user program.

1. Double-click the counter sub module of the CPU 314C-2 PN/DP.

⇨ The dialog *'Properties'* is opened.

2. For example, select *'channel 0'* and select the function *'Pulse-width modulation'* as *'Operating mode'*.

3. Leave all values unchanged.



4. Close the dialog with [OK].
5. Select *'Station ➔ Save and compile'*.
6. Close the hardware configurator.

6.5.3 User program

Include library

1. Go to the service area of www.vipa.com.
2. Download the *Simple Motion Control* library from the download area at *'VIPA Lib'*.
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:
  - V1000 PWM
    - FB885 – VMC_AxisControlV1000PWM *Chap. 6.7.1 'FB 885 - VMC_AxisControlV1000_PWM - Axis control over PWM' page 266*

OB 1

Configuration of the axis

If you are using a channel other than channel 0, you must adapt it in the hardware configuration and in your user program.

1. Open in the *Project tree* within the CPU at *'PLC program'*, *'Programming blocks'* the OB 1 and program the Call FB 885, DB 885.
   ⇨ The dialog *'Add instance data block'* opens.
2. Set the number for the instance data block, if not already done, and close the dialog with [OK].
   ⇨ The block call is created and the parameters are listed

3. Assign the following parameters for the sample project:

⇨
```
CALL  FB   "VMC_AxisControlV1000PWM" ,
"VMC_AxisCtrlV1000PWM_885"
       I_ChannelNumberPWM  :="Ax1_I_ChannelNumberPWM"
       I_MA_Alarm          :="Ax1_MA_Alarm"
       I_P1_Ready          :="I_P1_Ready"
       MaxVelocityDrive    :=1.000000e+002
       AxisEnable          :="Ax1_AxisEnable"
       AxisReset           :="Ax1_AxisReset"
       StopExecute         :="Ax1_StopExecute"
       MvVelocityExecute   :="Ax1_MvVelExecute"
       JogPositive         :="Ax1_JogPositive"
       JogNegative         :="Ax1_JogNegative"
       Velocity            :="Ax1_Velocity"
       I_S1_ForwardRun     :="Ax1_S1_ForwardRun"
       I_S2_ReverseRun     :="Ax1_S2_ReverseRun"
       I_S4_AlarmReset     :="Ax1_S4_AlarmReset"
       MinUserVelocity     :="Ax1_MinUserVelocity"
       MaxUserVelocity     :="Ax1_MaxUserVelocity"
       AxisReady           :="Ax1_AxisReady"
       AxisEnabled         :="Ax1_AxisEnabled"
       AxisError           :="Ax1_AxisError"
       AxisErrorID         :="Ax1_AxisErrorID"
       DriveError          :="Ax1_DriveError"
       CmdActive           :="Ax1_CmdActive"
       CmdDone             :="Ax1_CmdDone"
       CmdBusy             :="Ax1_CmdBusy"
       CmdAborted          :="Ax1_CmdAborted"
       CmdError            :="Ax1_CmdError"
       CmdErrorID          :="Ax1_CmdErrorID"
```

The addresses of *I_P1_Ready* and *I_MA_Alarm* are derived from the addresses of the inputs which are connected to the digital outputs of the drive. These can be determined via the sub module *'-X25 DI/DIO'* of the CPU.

The addresses of *I_S1_ForwardRun*, *I_S2_ReverseRun* and *I_S4_AlarmReset* are obtained from the addresses of the outputs which are connected to the digital inputs of the drive. These can be determined via the sub module *'-X25 DI/DIO'* of the CPU.

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. Bring your CPU into RUN and turn on your drive.

⇨ The FB 885 - VMC_AxisControlV1000PWM is executed cyclically.

3. As soon as *AxisReady* = TRUE, you can use *AxisEnable* to enable the drive.

4. You now have the possibility to control your drive via its parameters and to check its status. *Chap. 6.7.1 'FB 885 - VMC_AxisControlV1000_PWM - Axis control over PWM' page 266*

6.6 Usage in Siemens TIA Portal

6.6.1 Precondition

Overview

- Please use the Siemens TIA Portal V 14 and up for the configuration.
- The configuration of the VIPA CPU with PWM functionality happens in the Siemens TIA Portal by means of a virtual PROFINET IO device.
- The PROFINET IO Device is to be installed in the hardware catalog by means of a GSDML.

Installing the VIPA IO device

The installation of the PROFINET VIPA IO device happens in the hardware catalog with the following approach:

1. Go to the service area of www.vipa.com.
2. Download the according file for your system - here System MICRO 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 ... > VIPA MICRO PLC*.

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

6.6.2 Hardware configuration

Add CPU in the project

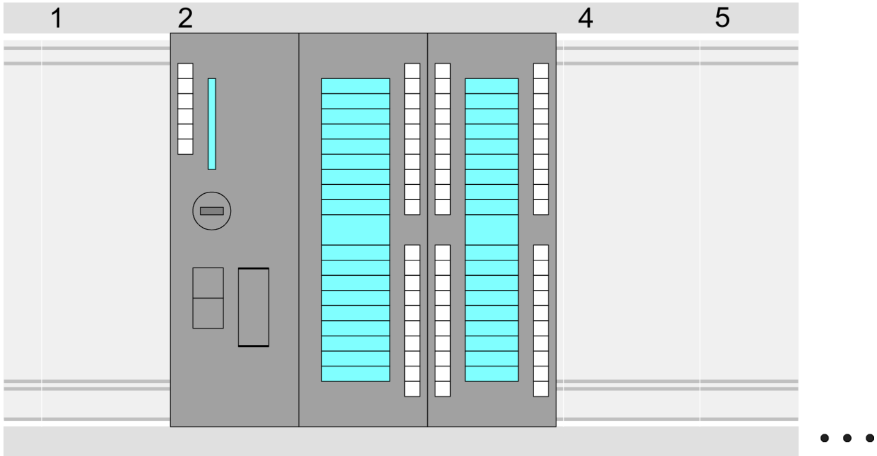
To be compatible with the Siemens SIMATIC TIA Portal the following steps should be executed:

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. Select the following CPU in the input dialog:

   SIMATIC S7-300 > CPU 314C-2 PN/DP (314-6EH04-0AB0 V3.3)

   ⇨ The CPU is inserted with a profile rail.

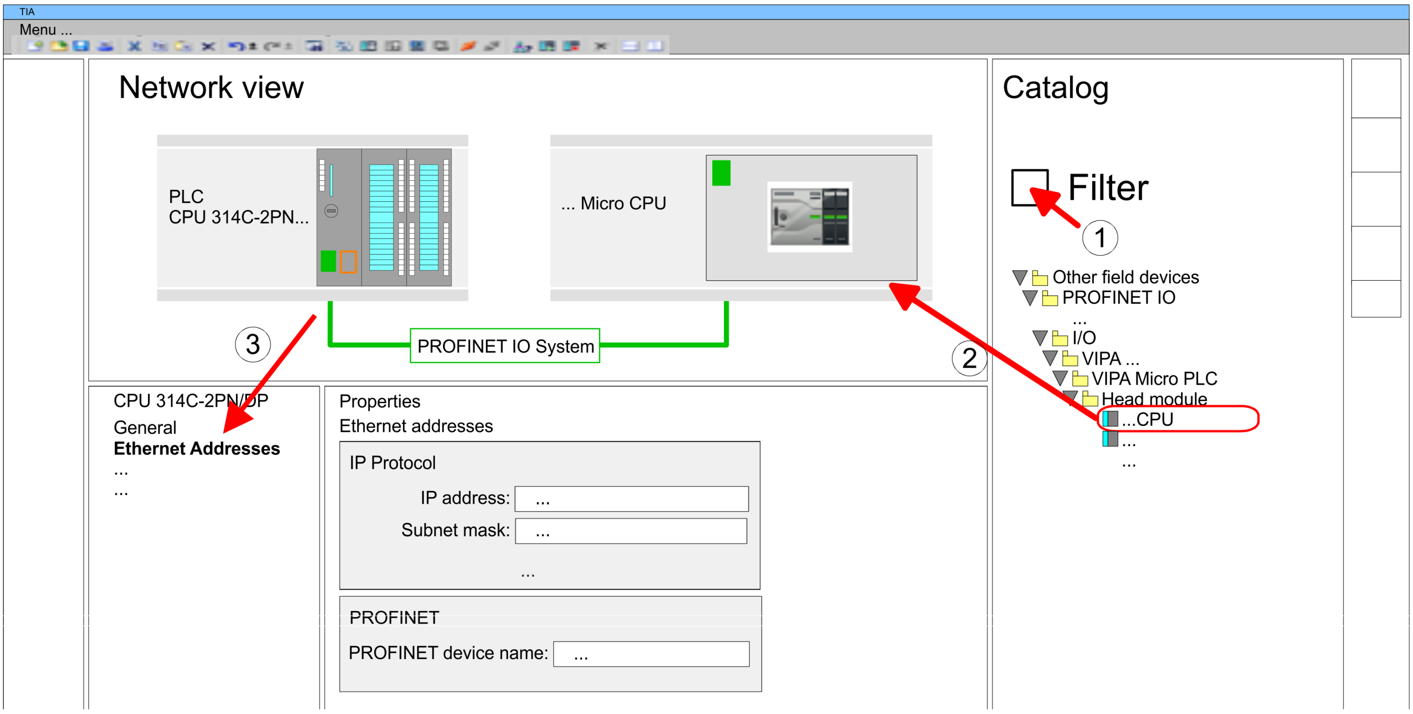


Device overview:

Module	...	Slot	...	Type	...
PLC...		2		CPU 314C-2PN/DP	
MPI interface...		2 X1		MPI/DP interface	
PROFINET inter-face...		2 X2		PROFINET interface	
DI24/DO16...		2 5		DI24/DO16	
AI5/AO2...		2 6		AI5/AO2	
Count...		2 7		Count	
...					

Connection CPU as PROFINET IO device

1. Switch in the *Project area* to *'Network view'*.
2. After installing the GSDML the IO device for the SLIO CPU may be found in the hardware catalog at *Other field devices > PROFINET > IO > VIPA ... > VIPA MICRO PLC*. 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 at 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.

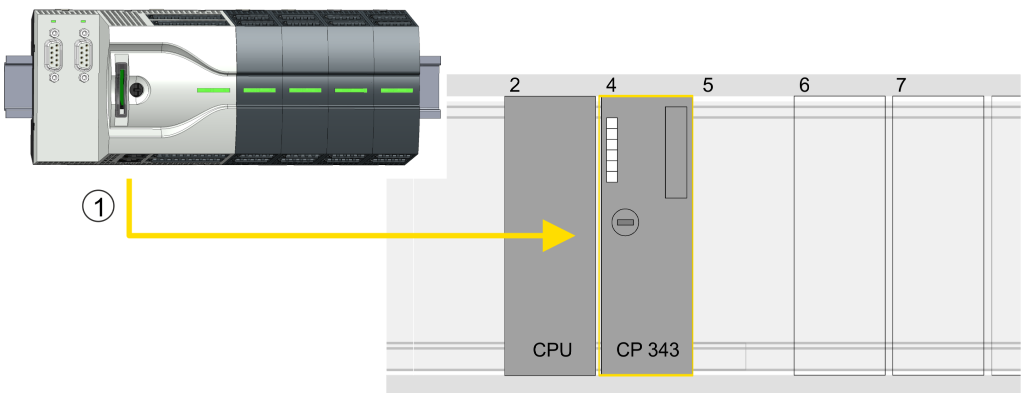


5. Select in the *Network view* the IO device *'VIPA MICRO PLC'* and switch to the *Device overview*.

⇨ In the *Device overview* of the PROFINET IO device *'VIPA MICRO PLC'* the CPU is already placed at slot 0.

Configuration of Ethernet PG/OP channel

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. You get valid IP address parameters from your system administrator.



1 Ethernet PG/OP channel

Device overview

Module	...	Slot	...	Type	...
PLC ...		2		CPU 314C-2PN/DP	

MPI/DP interface		2 X1		MPI/DP interface	
PROFINET inter-face		2 X2		PROFINET interface	
...		...		...	
CP 343-1		4		CP 343-1	
...		...		...	

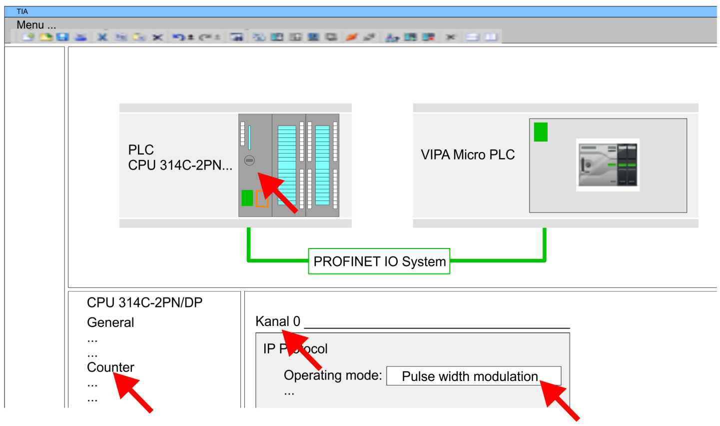
Switch I/O periphery to PWM

For parametrization of the input/output periphery and the *technological functions* the corresponding sub modules of the Siemens CPU 314C-2 PN/DP (314-6EH04-0AB0 V3.3) is to be used. For PWM output, the sub module count must be switched to *'Pulse-width modulation'*. If you are using a channel other than channel 0, you must adapt it in the hardware configuration and in your user program.

1. Double-click the counter sub module of the CPU 314C-2 PN/DP.

   ⇨ The dialog *'Properties'* is opened.

2. For example, select *'channel 0'* and select the function *'Pulse-width modulation'* as *'Operating mode'*.
3. Leave all values unchanged.



4. Click at the CPU and select *'Context menu ➔ Compile ➔ All'*.

6.6.3 User program

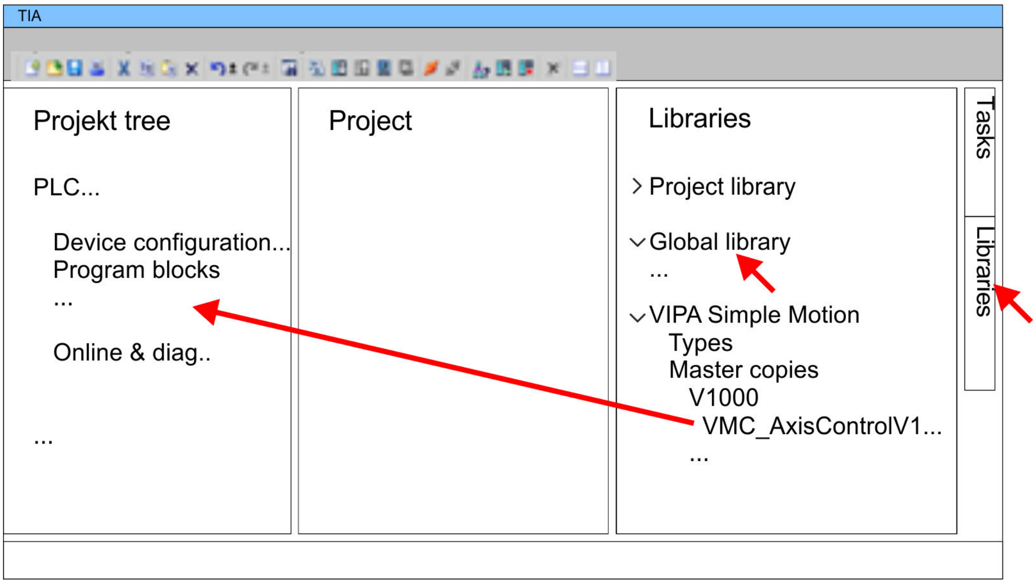
Include library

1. Go to the service area of www.vipa.com.
2. Download the *Simple Motion Control* library from the download area at *'VIPA Lib'*.

   The library is available as packed zip file for the corresponding TIA Portal version.

3. Start your un-zip application with a double click on the file ...TIA_V*xx*.zip and copy all the files and folders in a work directory for the Siemens TIA Portal.
4. Switch to the *Project view* of the Siemens TIA Portal.
5. Choose "Libraries" from the task cards on the right side.
6. Click at "Global library".
7. Click on the free area inside the *'Global Library'* and select *'Context menu ➔ Retrieve library'*.
8. Navigate to your work directory and load the file ...Simple Motion.zal*xx*.

Copy blocks into project



→ Copy the following block from the library into the "Program blocks" of the *Project tree* of your project.

- V1000 PWM
  - FB885 – VMC_AxisControlV1000PWM *Chap. 6.7.1 'FB 885 - VMC_AxisControlV1000_PWM - Axis control over PWM' page 266*

OB 1

Configuration of the axis

If you are using a channel other than channel 0, you must adapt it in the hardware configuration and in your user program.

1. Open in the *Project tree* within the CPU at *'Programming blocks'* the OB 1 and program the Call FB 885, DB 885.

   ⇨ The dialog *'Add instance data block'* opens.

2. Set the number for the instance data block, if not already done, and close the dialog with [OK].

   ⇨ The block call is created and the parameters are listed

3. Assign the following parameters for the sample project:

⇨
```
CALL  FB  "VMC_AxisControlV1000PWM" ,
"VMC_AxisCtrlV1000PWM_885"
       I_ChannelNumberPWM  :="Ax1_I_ChannelNumberPWM"
       I_MA_Alarm          :="Ax1_MA_Alarm"
       I_P1_Ready          :="I_P1_Ready"
       MaxVelocityDrive    :=1.000000e+002
       AxisEnable          :="Ax1_AxisEnable"
       AxisReset           :="Ax1_AxisReset"
       StopExecute         :="Ax1_StopExecute"
       MvVelocityExecute   :="Ax1_MvVelExecute"
       JogPositive         :="Ax1_JogPositive"
       JogNegative         :="Ax1_JogNegative"
       Velocity            :="Ax1_Velocity"
       I_S1_ForwardRun     :="Ax1_S1_ForwardRun"
       I_S2_ReverseRun     :="Ax1_S2_ReverseRun"
       I_S4_AlarmReset     :="Ax1_S4_AlarmReset"
       MinUserVelocity     :="Ax1_MinUserVelocity"
       MaxUserVelocity     :="Ax1_MaxUserVelocity"
       AxisReady           :="Ax1_AxisReady"
       AxisEnabled         :="Ax1_AxisEnabled"
       AxisError           :="Ax1_AxisError"
       AxisErrorID         :="Ax1_AxisErrorID"
       DriveError          :="Ax1_DriveError"
       CmdActive           :="Ax1_CmdActive"
       CmdDone             :="Ax1_CmdDone"
       CmdBusy             :="Ax1_CmdBusy"
       CmdAborted          :="Ax1_CmdAborted"
       CmdError            :="Ax1_CmdError"
       CmdErrorID          :="Ax1_CmdErrorID"
```

The addresses of *I_P1_Ready* and *I_MA_Alarm* are derived from the addresses of the inputs which are connected to the digital outputs of the drive. These can be determined via the sub module *'-X25 DI/DIO'* of the CPU.

The addresses of *I_S1_ForwardRun*, *I_S2_ReverseRun* and *I_S4_AlarmReset* are obtained from the addresses of the outputs which are connected to the digital inputs of the drive. These can be determined via the sub module *'-X25 DI/DIO'* of the CPU.

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. Bring your CPU into RUN and turn on your drive.

   ⇨ The FB 875 - VMC_AxisControl_PT is executed cyclically.

3. As soon as *AxisReady* = TRUE, you can use *AxisEnable* to enable the drive.

4. You now have the possibility to control your drive via its parameters and to check its status. *Chap. 6.7.1 'FB 885 - VMC_AxisControlV1000_PWM - Axis control over PWM' page 266*

6.7 Drive specific block

6.7.1 FB 885 - VMC_AxisControlV1000_PWM - Axis control over PWM

6.7.1.1 Description

With the FB *VMC_AxisControlV1000_PWM* you can control an inverter drive, which is connected via PWM and check its status.

Parameter

Parameter	Declaration	Data type	Description
I_Channel-NumberPWM	INPUT	INT	Channel number of the PWM output used to drive the PWM input of the inverter drive.
I_MA_Alarm	INPUT	BOOL	■ Digital input for connecting the *I_MA_Alarm* signal (MA) – TRUE: The inverter drive has detected an error.
I_P1_Ready	INPUT	BOOL	■ Digital input for connecting the *I_P1_Ready* signal – FALSE: The inverter drive is ready.
MaxVelocity-Drive	INPUT	REAL	■ Maximum speed of the inverter drive [user units]. *Chap. 6.7.1.2 'Calculating' page 268*
AxisEnable	INPUT	BOOL	■ Enable/disable axis – This parameter is used for block-internal release and has no influence on the inverter drive. – TRUE: The axis is enabled. – FALSE: The axis is disabled.
AxisReset	INPUT	BOOL	■ Reset axis – Edge 0-1: Axis reset is performed. – The status of a reset, started with *AxisReset*, is not indicated at the outputs *CmdActive*, *CmdDone*, *CmdBusy*, *CmdAborted*, *CmdError* and *CmdErrorID*.
StopExecute	INPUT	BOOL	■ Stop axis – Edge 0-1: Stopping of the axis is started. Note: *StopExecute* = 1: No other command can be started!
MvVelocityExe-cute	INPUT	BOOL	■ Start moving the axis – Edge 0-1: The axis is accelerated/decelerated to the speed specified.
JogPositive	INPUT	BOOL	Jog operation positive ■ Drive axis with constant velocity in positive direction – Edge 0-1: Drive axis with constant velocity is started. – Edge 1-0: The axis is stopped.
JogNegative	INPUT	BOOL	Jog operation negative ■ Drive axis with constant velocity in negative direction – Edge 0-1: Drive axis with constant velocity is started. – Edge 1-0: The axis is stopped.
Velocity	INPUT	REAL	Velocity setting (signed value) in [user units / s]. Note: *JogPositive* and *JogNegative* use the absolute value of the speed.
I_S1_Forwar-dRun	OUTPUT	BOOL	■ Digital output for controlling the inverter drive signal S1 – TRUE: Enables the inverter drive in positive direction.

Parameter	Declaration	Data type	Description
I_S2_Rever-seRun	OUTPUT	BOOL	■ Digital output for controlling the inverter drive signal S2 – TRUE: Enables the inverter drive in negative direction.
I_S4_Alarm-Reset	OUTPUT	BOOL	■ Digital output for controlling the inverter drive signal S4 – TRUE: Alarm messages are reset in the inverter drive. – FALSE: Alarm messages in the inverter drive remain.
MinUserVe-locity	OUTPUT	REAL	Minimum speed (period duration = 65535µs = maximum period of the PWM output) of the inverter drive [user units].
MinUserVe-locity	OUTPUT	REAL	Maximum speed at a maximum frequency of 20kHz of the inverter drive [user units].
AxisReady	OUTPUT	BOOL	■ AxisReady – TRUE: The axis is ready to switch on. – FALSE: The axis is not ready to switch on. → Check and fix *AxisError* (see *AxisErrorID*). → Check and fix *DriveError* (see DriveErrorID).
AxisEnabled	OUTPUT	BOOL	■ Status axis – TRUE: Axis is switched on and accepts motion commands. – FALSE: Axis is not switched on and does not accepts motion commands.
AxisError	OUTPUT	BOOL	■ Error on axis – TRUE: An error has occurred. Additional error information can be found in the parameter *AxisErrorID*. → The axis is locked (*S_On* = FALSE and *AxisEnabled* = FALSE). Command is not executed.
AxisErrorID	OUTPUT	WORD	Additional error information *Chap. 12 'ErrorID - Additional error information' page 457*
DriveError	OUTPUT	BOOL	■ Error on the inverter drive – TRUE: An error has occurred. → The axis is disabled.
CmdActive	OUTPUT	BYTE	■ Command – 0: no Cmd active – 1: STOP – 2: MvVelocity – 4: JogPos – 5: JogNeg
CmdDone	OUTPUT	BOOL	■ Status Done – TRUE: Job ended without error.
CmdBusy	OUTPUT	BOOL	■ Status Busy – TRUE: Job is running.
CmdAborted	OUTPUT	BOOL	■ Status Aborted – TRUE: The job was aborted during processing by another job. Note: *CmdAborted* is reset when a Cmd is started

Parameter	Declaration	Data type	Description
CmdError	OUTPUT	BOOL	■ Status Error – TRUE: An error has occurred. The axis is disabled Additional error information can be found in the parameter *CmdErrorID*.
CmdErrorID	OUTPUT	WORD	Additional error information *Chap. 12 'ErrorID - Additional error information' page 457*

CAUTION!
Please note that the block does not recognize a CPU restart. To prevent the axis from starting unintentionally during a CPU restart, the values at the inputs *AxisEnable*, *JogPositive* and *JogNegative* should be set to FALSE using the startup OB, eg OB 100!

6.7.1.2 Calculating

MaxVelocityDrive

$$n = 2 \cdot 60 \cdot \frac{f max,out}{poles} \frac{1}{min}$$

This value is used to normalize the input value *Velocity*.

$f_{max, out}$ - Maximum frequency (parameter E1-04)

poles - Number of motor poles (parameter E5-04)

n - Maximum speed of the inverter drive [user units] such as 1000.0 % or 3000.0 rotations/min.

6.7.1.3 Functionality

Switch the axis on or off

- The *AxisEnable* input is used to switch an axis on or off.
- Switching on is only possible if *AxisReady* = TRUE, i.e. the axis is ready to switch on.
- As soon as the axis is switched on, this is indicated by the status information *AxisEnabled*.
- If the axis has an error, this is indicated by the status information *AxisError*. For more information refer to *AxisErrorID*.

Acknowledge axis error

- With *AxisReset* you can acknowledge axis errors.
- Errors are reported via *DriveError*.

Stop axis

- You can stop an axis in motion by setting *StopExecute*.
- As long as *StopExecute* is set, no further pulses are generated and all commands are blocked.

Velocity mode

- Precondition: The axis is switched on and *AxisReady* = TRUE.
- With *MvVelocityExecute*, you can bring the axis to rotate with constant velocity.
- You specify the velocity via *Velocity*.
- By setting 0, the axis stops as well as with *StopExecute*.
- The direction of rotation is determined by the sign of *Velocity*.
- The *Velocity* value can be 0 or *MinUserVelocity* ≤ *Velocity* ≤ *MaxUserVelocity*.

Jog mode

- Precondition: The axis is switched on and *AxisReady* = TRUE.
- With an edge 0-1 at *JogPositive* or *JogNegative*, you can control your drive in jog mode. In this case, a jogging command is executed in the corresponding direction of rotation.
- You specify the velocity via *Velocity*. The sign is not relevant.
- With an edge 1-0 at *JogPositive* or *JogNegative* respectively by setting *StopExecute* the axis is stopped.

7 Usage inverter drive via Modbus RTU

7.1 Overview

Precondition

- SPEED7 Studio from V1.7.1
  or
- Siemens SIMATIC Manager from V 5.5, SP2 & *Simple Motion Control Library*
  or
- Siemens TIA Portal V 14 & *Simple Motion Control Library*
- System MICRO or System SLIO CPU with serial interface such as CPU M13-CCF0000 or CPU 013-CCF0R00.
- *V1000* inverter drive with serial interface and associated motor

Steps of configuration

1. Set the parameters on the inverter drive
   - The setting of the parameters happens by means of the software tool *Drive Wizard+*.
2. Hardware configuration in the VIPA *SPEED7 Studio*, Siemens SIMATIC Manager or Siemens TIA Portal.
   - Configuring the CPU.
3. Programming in the VIPA *SPEED7 Studio*, Siemens SIMATIC Manager or Siemens TIA Portal.
   - Connect the block for serial communication.
   - Connect the block for each Modbus slave.
   - Connect the block for the communication data of all Modbus slaves.
   - Connect the block for the communication manager.
   - Connect the block for initializing the inverter drive.
   - Connecting the blocks for motion sequences.

7.2 Set the parameters on the inverter drive

CAUTION!
Before the commissioning, you have to adapt your inverter drive to your application with the *Drive Wizard+* software tool! More may be found in the manual of your inverter drive.

The following table shows all parameters which do not correspond to the default values. The following parameters must be set via *Drive Wizard+* to match the *Simple Motion Control Library*.

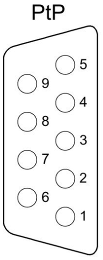
No.	Designation	Range of values	Setting for *Simple Motion Control Library*
H5-01	Slave address inverter drive	00h, 20h	By default, the slave address is set to 1Fh. Please note that addresses in the network must not be assigned more than once!
H5-02	Communication speed MEMOBUS/ Modbus	0, 1, 2, ..., 8	■ 3: 9600bit/s
H5-03	Transmission parity MEMOBUS/Modbus	0, 1, 2	■ 0: no parity

No.	Designation	Range of values	Setting for *Simple Motion Control Library*
H5-04	Stop method after communication error (CE error)	0, 1, 2, 3	■ 3: Operation continues with alarm
H5-05	Stop method after communication error (CE error)	0, 1	■ 1: Activated - If the connection is aborted for longer than 2s (adjustable via *H2-09*), a CE error is triggered.
H5-06	Waiting time between receiving and sending data from the inverter drive	5 ... 65ms	■ 5ms
H5-07	Request to send (RTS) control	0, 1	■ 1: Activated - RTS is activated only when sending (RS485 or RS422 and *multi-drop*)
H5-09	Time after which a communication error (CE error) is detected.	0,0 ... 10,0s	■ 2s
H5-10	Step size (resolution) for the MEM-OBUS/Modbus register 0025h	0, 1	By default, the resolution is set to 0.1V increments (0). ■ 0: 0.1V increments ■ 1: 1V increments
H5-11	ENTER function for connections	0, 1	■ 1: Enter command not required
H5-12	Selection start command method	0, 1	■ 1: Run/Stop
B1-01	Input source frequency setpoint 1	0, 1, 2, 3, 4	■ 2: MEMOBUS/Modbus communication
B1-02	Input source start command 1	0, 1, 2, 3	■ 2: MEMOBUS/Modbus communication
B1-15	Input source frequency setpoint 2	0, 1, 2, 3, 4	■ 2: MEMOBUS/Modbus communication
B1-16	Input source start command 2	0, 1, 2, 3	■ 2: MEMOBUS/Modbus communication

For all settings to be accepted, you must restart the inverter drive after parametrization!

7.3 Wiring

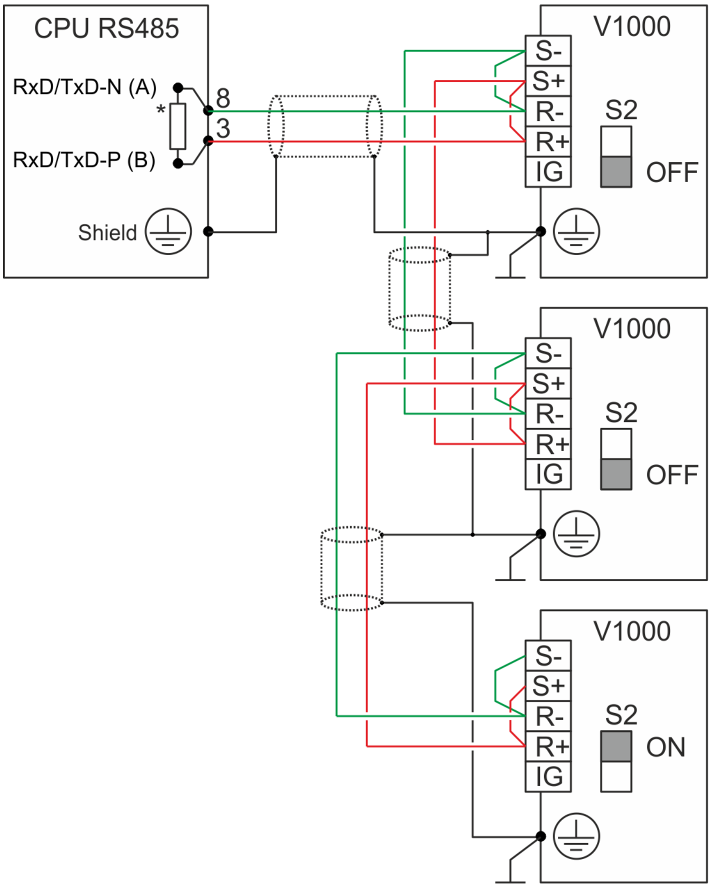
RS485 cabling



1. n.c.
2. M24V
3. RxD/TxD-P (line B)
4. RTS
5. M5V
6. P5V
7. P24V
8. RxD/TxD-N (line A)
9. n.c.

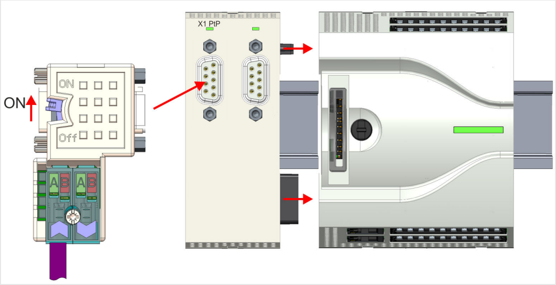
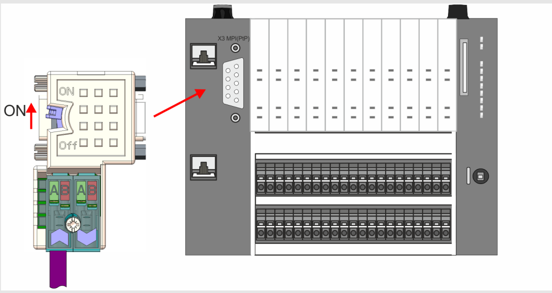
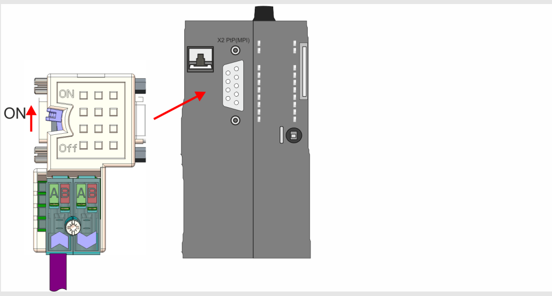
The following figure shows the connection of *V1000* inverter drives via RS485. Here the individual inverter drives are connected via PROFIBUS cables and connected to the CPU via a PROFIBUS connector to the PtP interface (**P**oint-**t**o-**P**oint).

- A maximum of 8 inverter drives can be connected via Modbus RTU.
- For all connected inverter drives, parameter H5-07 must be set to 1.
- The serial line must be terminated at its end with a terminator. To activate it, you must set switch S2 to *'ON'* on the corresponding inverter drive.

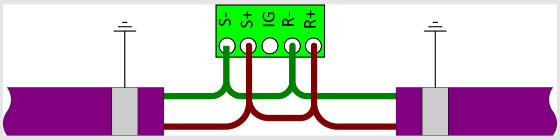
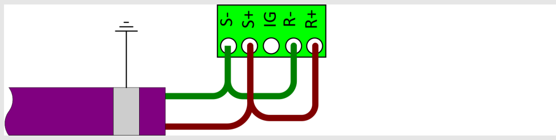
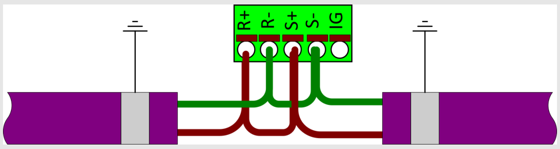
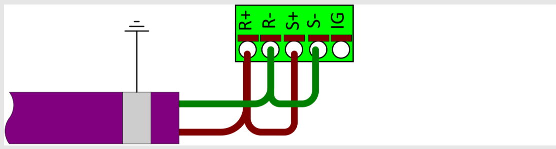
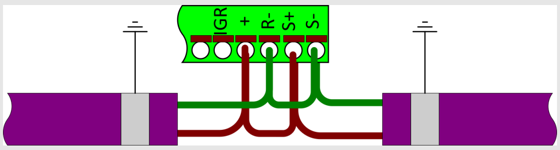
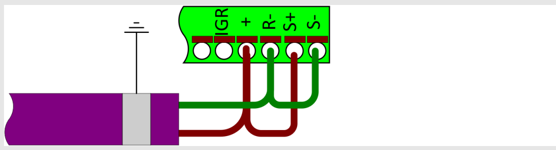
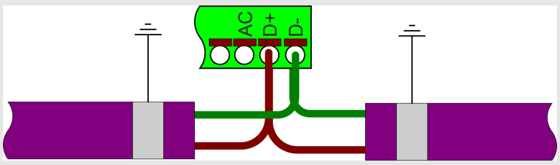
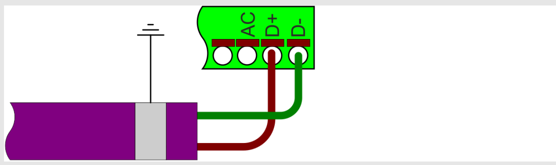


- **) For a trouble-free data traffic, use a terminating resistor of approx. 120Ω at the CPU, such as the VIPA PROFIBUS connector.*
- *Never connect the cable shield and the M5V (pin 5) together, due to the compensation currents the interfaces could be destroyed!*

Connection of the CPU

CPU	Connection	Comment
MICRO CPU M13C		■ PtP communication requires the optional EM M09 extension module. ■ The extension module provides interface X1: PtP (RS422/485) with fixed pin assignment. ■ For connection to the CPU, use a VIPA PROFIBUS connector. ■ Activate the terminating resistor on the PROFIBUS connector. ■ After switching on the power supply and a short start-up time, the CPU is ready for the PtP communication.
System SLIO CPU 013C		■ The CPU has the interface X3 MPI(PtP) with a fix pinout. ■ For connection to the CPU, use a VIPA PROFIBUS connector. ■ Activate the terminating resistor on the PROFIBUS connector. ■ After switching on the power supply and a short start-up time or after an overall reset, the interface has MPI functionality. You can activate the PtP functionality via the hardware configuration. *Chap. 7.4 'Usage in VIPA SPEED7 Studio' page 275* *Chap. 7.5 'Usage in Siemens SIMATIC Manager' page 289* *Chap. 7.6 'Usage in Siemens TIA Portal' page 304*
System SLIO CPU 014 ... 017		■ The CPU has the interface X2 PtP(MPI) which is per default set to PtP communication (point to point). ■ For connection to the CPU, use a VIPA PROFIBUS connector. ■ Activate the terminating resistor on the PROFIBUS connector. ■ After switching on the power supply and a short start-up time, the CPU is ready for the PtP communication.

Connection of the YASKAWA inverter drives

FU	Connection continuous	Connection termination
J1000		
V1000		
A1000		
GA700		

More can be found in the according manual.

7.4 Usage in VIPA *SPEED7 Studio*

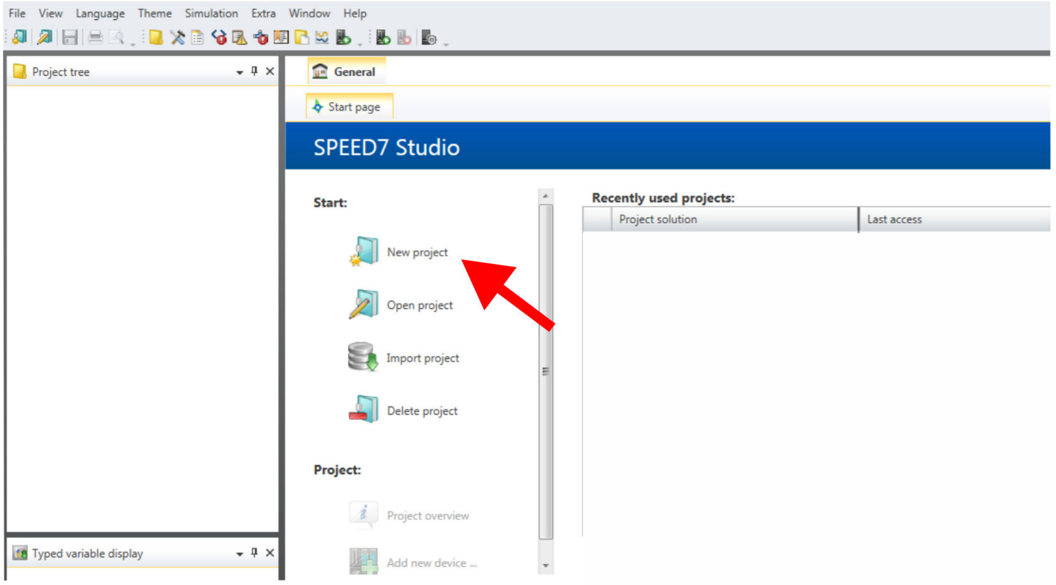
7.4.1 Hardware configuration

7.4.1.1 Hardware configuration System MICRO

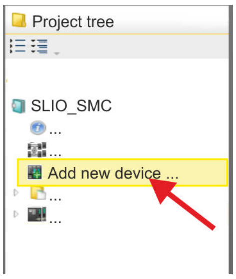
Add CPU in the project

Please use the *SPEED7 Studio* V1.7.1 and up for the configuration.

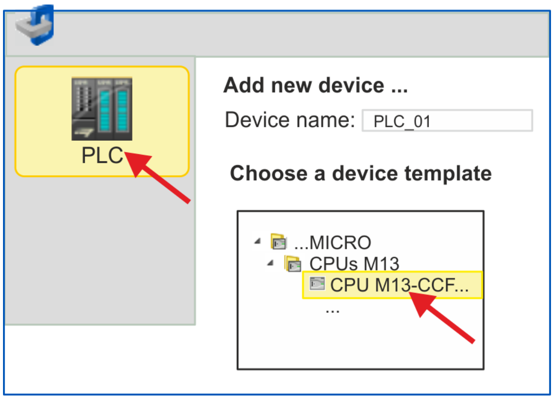
1. Start the *SPEED7 Studio*.



2. Create a new project at the start page with *'New project'* and assign a *'Project name'*.
   ⇨ A new project is created and the view *'Devices and networking'* is shown.



3. Click in the *Project tree* at *'Add new device ...'*.

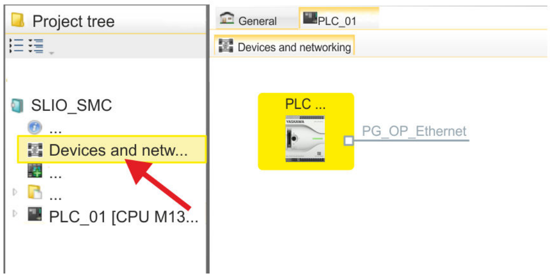


   ⇨ A dialog for device selection opens.

4. Select from the *'Device templates'* your System MICRO CPU M13-CCF0000 and click at [OK].
   ⇨ The CPU is inserted in *'Devices and networking'* and the *'Device configuration'* is opened.

Configuration of Ethernet PG/OP channel

1. Click in the *Project tree* at *'Devices and networking'*.
   ⇨ You will get a graphical object view of your CPU.

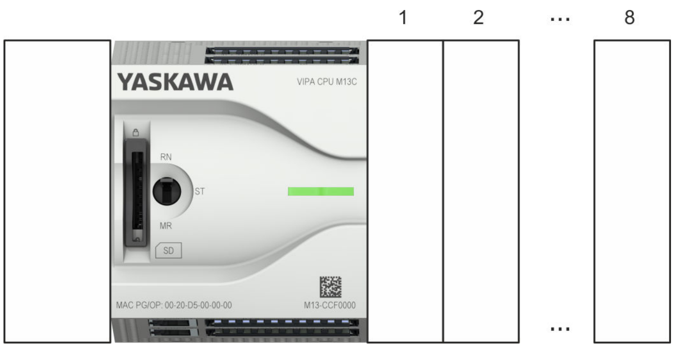


2. Click at the network *'PG_OP_Ethernet'*.
3. Select *'Context menu ➔ Interface properties'*.
   ⇨ A dialog window opens. Here you can enter the IP address data for your Ethernet PG/OP channel. You get valid IP address parameters from your system administrator.
4. Confirm with [OK].
   ⇨ The IP address data are stored in your project listed in *'Devices and networking'* at *'Local components'*.

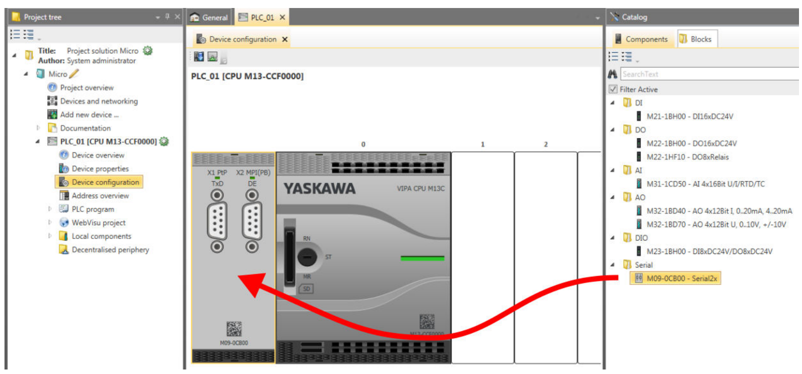
   After transferring your project your CPU can be accessed via Ethernet PG/OP channel with the set IP address data.

Enable PtP functionality

1. Click in the *Project tree* at *'PLC..CPU M13... ➔ Device configuration'*.
   ⇨ The *'Device configuration'* opens.



2. In the *'Catalog'* at *'Components'*, open the *'Serial'* collection and drag and drop the serial module *'M09-0CB00 - Serial2x'* to the left slot of the CPU. By default, the interface X1 is set to PtP functionality.

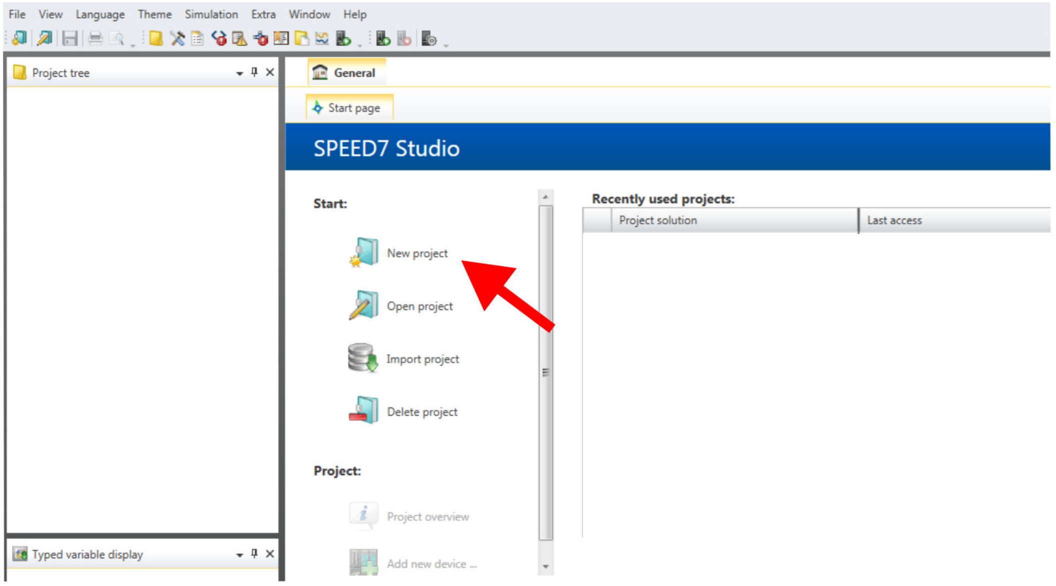


7.4.1.2 Hardware configuration System SLIO CPU 013C

Add CPU in the project

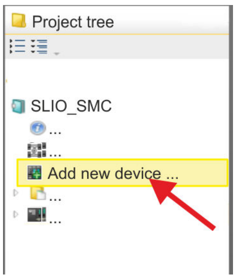
Please use the *SPEED7 Studio* V1.7.1 and up for the configuration.

1. Start the *SPEED7 Studio*.

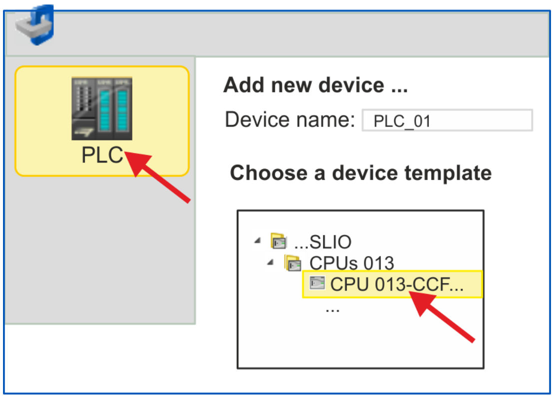


2. Create a new project at the start page with *'New project'* and assign a *'Project name'*.

   ⇨ A new project is created and the view *'Devices and networking'* is shown.



3. Click in the *Project tree* at *'Add new device ...'*.



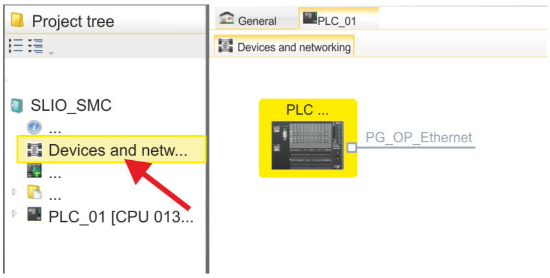
   ⇨ A dialog for device selection opens.

4. Select from the *'Device templates'* your System SLIO CPU 013-CCF0R00 and click at [OK].

   ⇨ The CPU is inserted in *'Devices and networking'* and the *'Device configuration'* is opened.

Configuration of Ethernet PG/OP channel

1. Click in the *Project tree* at *'Devices and networking'*.
   ⇨ You will get a graphical object view of your CPU.

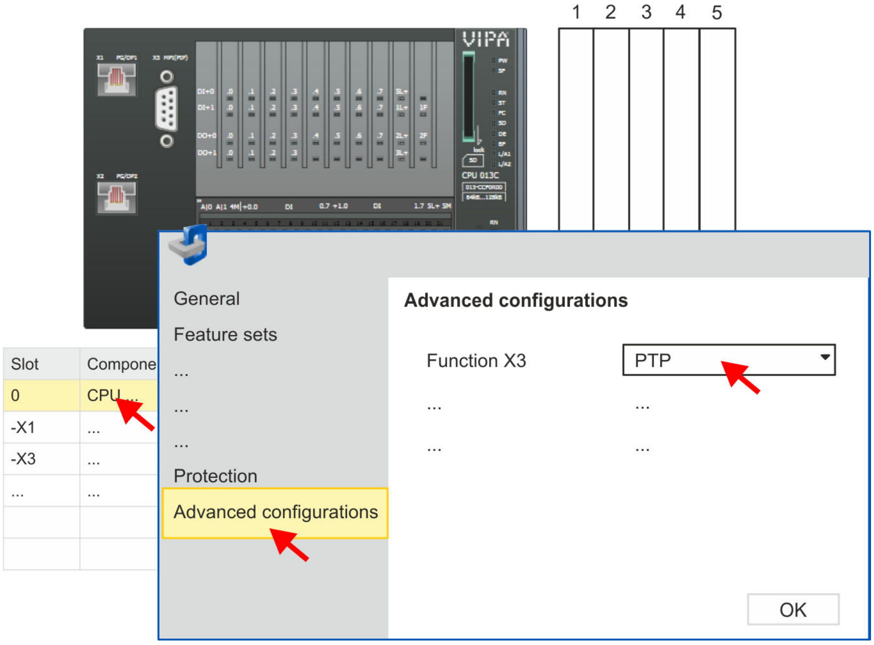


2. Click at the network *'PG_OP_Ethernet'*.
3. Select *'Context menu ➔ Interface properties'*.
   ⇨ A dialog window opens. Here you can enter the IP address data for your Ethernet PG/OP channel. You get valid IP address parameters from your system administrator.
4. Confirm with [OK].
   ⇨ The IP address data are stored in your project listed in *'Devices and networking'* at *'Local components'*.

   After transferring your project your CPU can be accessed via Ethernet PG/OP channel with the set IP address data.

Enable PtP functionality

1. Click in the *Project tree* at *'PLC... > Device configuration'*.
2. Click in the *'Device configuration'* at *'0 CPU 013...'* and select *'Context menu ➔ Components properties'*.
   ⇨ The properties dialog is opened.



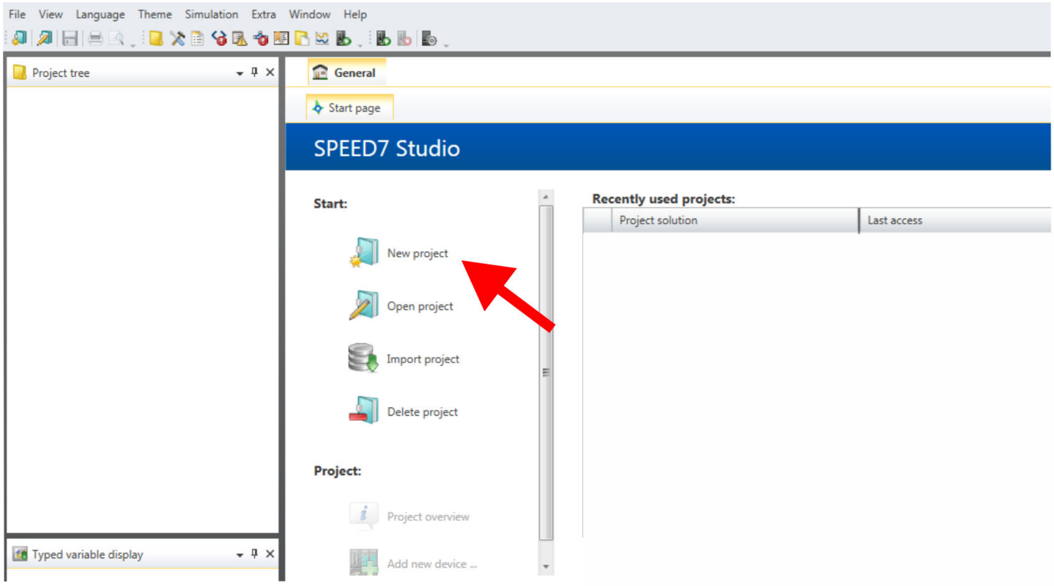
3. Click at *'Advanced configurations'* and select at *'Function X3'* the value *'PTP'*.

7.4.1.3 Hardware configuration System SLIO CPU 014 ... 017

Add CPU in the project

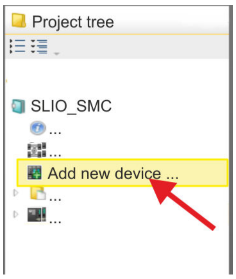
Please use the *SPEED7 Studio* V1.7.1 and up for the configuration.

1. Start the *SPEED7 Studio*.

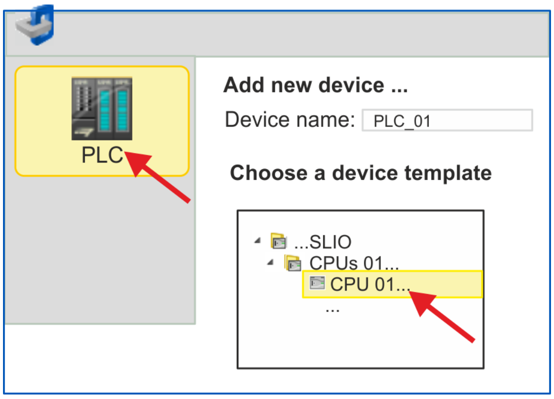


2. Create a new project at the start page with *'New project'* and assign a *'Project name'*.

   ⇨ A new project is created and the view *'Devices and networking'* is shown.



3. Click in the *Project tree* at *'Add new device ...'*.



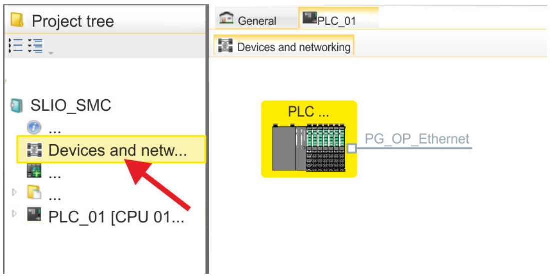
   ⇨ A dialog for device selection opens.

4. Select from the *'Device templates'* the corresponding System SLIO CPU and click at [OK].

   ⇨ The CPU is inserted in *'Devices and networking'* and the *'Device configuration'* is opened.

Configuration of Ethernet PG/OP channel

1. Click in the *Project tree* at *'Devices and networking'*.
   ⇨ You will get a graphical object view of your CPU.



2. Click at the network *'PG_OP_Ethernet'*.
3. Select *'Context menu ➔ Interface properties'*.
   ⇨ A dialog window opens. Here you can enter the IP address data for your Ethernet PG/OP channel. You get valid IP address parameters from your system administrator.
4. Confirm with [OK].
   ⇨ The IP address data are stored in your project listed in *'Devices and networking'* at *'Local components'*.

   After transferring your project your CPU can be accessed via Ethernet PG/OP channel with the set IP address data.

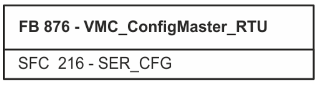
Enable PtP functionality

For the System SLIO CPUs 014 ... 017, the RS485 interface is set to PtP communication as standard. A hardware configuration to enable the PtP functionality is not necessary.

7.4.2 User program

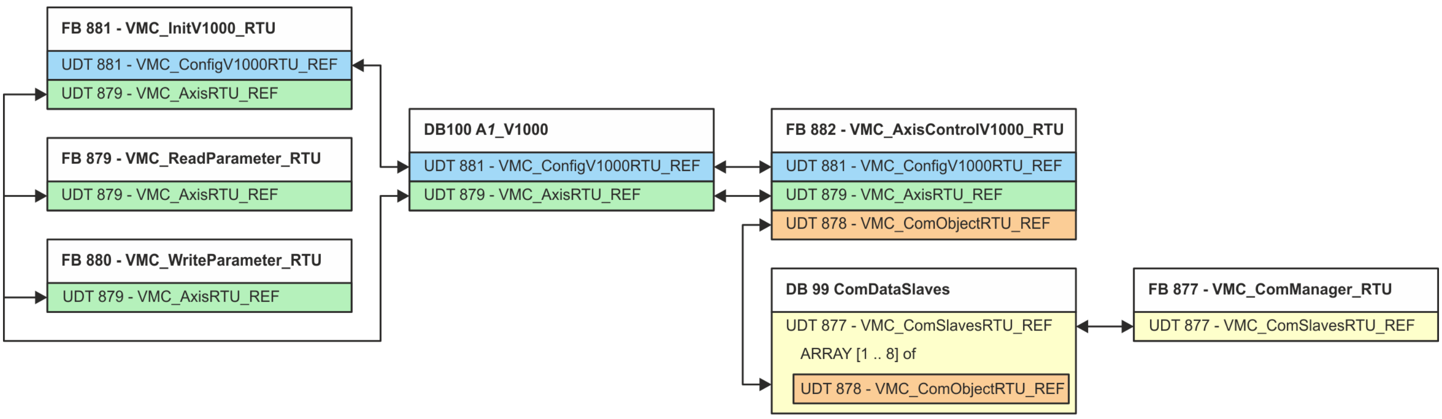
7.4.2.1 Program structure

OB 100



- FB 876 - VMC_ConfigMaster_RTU 🔗 *322*
  – This block is used to parametrize the serial interface of the CPU for Modbus RTU communication.
  – Internally block SFC 216 - SER_CFG is called.

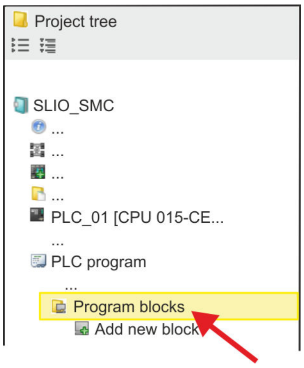
OB 1



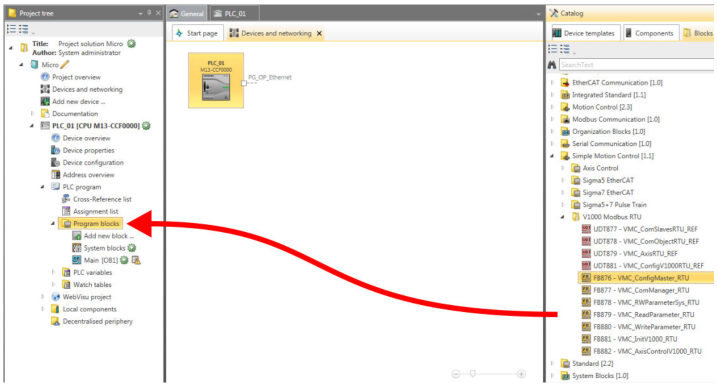
With the exception of blocks DB 99 and FB 877, you must create the blocks listed below for each connected inverter drive:

- FB 881 - VMC_InitV1000_RTU ↳ *325*
  – The FB 881 - VMC_InitV1000_RTU initializes the corresponding inverter drive with the user data.
  – Before an inverter drive can be controlled, it must be initialized.
  – UDT 881 - VMC_ConfigV1000RTU_REF ↳ *322*
  – UDT 879 - VMC_AxisRTU_REF ↳ *322*
- FB 879 - VMC_ReadParameter_RTU ↳ *324*
  – With this FB you have read access to the parameters of an inverter drive, which is connected serially via Modbus RTU.
  – The read data are recorded in a data block.
  – UDT 879 - VMC_AxisRTU_REF ↳ *322*
- FB 880 - VMC_WriteParameter_RTU ↳ *325*
  – With this FB you have read access to the parameters of an inverter drive, which is connected serially via Modbus RTU.
  – The data to be written must be stored in a data block.
  – UDT 879 - VMC_AxisRTU_REF ↳ *322*
- DB 100 - A1_V1000
  – For each inverter drive, which is serially connected via Modbus RTU, a data block must be created.
  – UDT 879 - VMC_AxisRTU_REF ↳ *322*
  – UDT 881 - VMC_ConfigV1000RTU_REF ↳ *322*
- FB 882 - VMC_AxisControlV1000_RTU ↳ *327*
  – With this block, you can control an inverter drive, which is serially connected via Modbus RTU and check its status.
  – UDT 881 - VMC_ConfigV1000RTU_REF ↳ *322*
  – UDT 879 - VMC_AxisRTU_REF ↳ *322*
  – UDT 878 - VMC_ComObjectRTU_REF ↳ *322*
- DB 99 - ComDataSlaves
  – For the communication data of all the inverter drives (max. 8), which are serially connected via Modbus RTU, a common data block is to be created.
  – UDT 877 - VMC_ComSlavesRTU_REF ↳ *322*
  – UDT 878 - VMC_ComObjectRTU_REF ↳ *322*
- FB 877 - VMC_ComManager_RTU ↳ *323*
  – The device ensures that only 1 inverter drive (Modbus slave) can use the serial interface. If several inverter drives are used, this block, as communication manager, sends the jobs to the respective Modbus slaves and evaluates their responses.
  – UDT 877 - VMC_ComSlavesRTU_REF ↳ *322*

7.4.2.2 Copy blocks into project



1. Click at *'Project tree ➔ ...CPU... ➔ PLC program ➔ Program blocks'*.



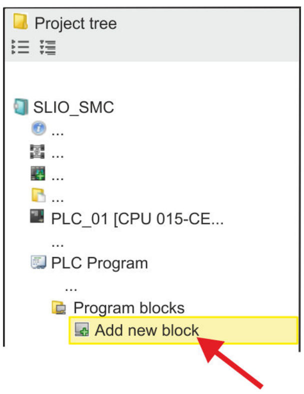
2. In the *'Catalog'* at *'Blocks ➔ Simple Motion Control'* open the collection *'V1000 Modbus RTU'* and drag and drop the following blocks into *'Program blocks'* of the *Project tree*:

   - FB 876 - VMC_ConfigMaster_RTU
   - FB 877 - VMC_ComManager_RTU
   - FB 878 - VMC_RWParameterSys_RTU
   - FB 879 - VMC_ReadParameter_RTU
   - FB 880 - VMC_WriteParameter_RTU
   - FB 881 - VMC_InitV1000_RTU
   - FB 882 - VMC_AxisControlV1000_RTU

   Here the following blocks are automatically added to the project:

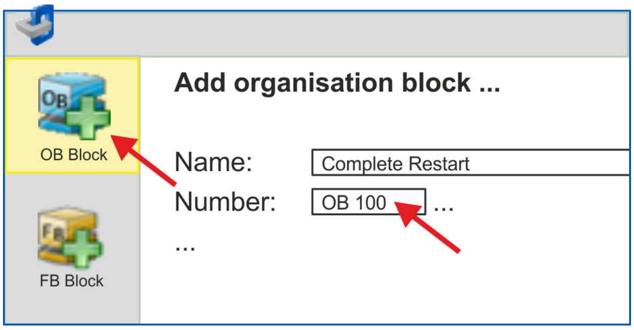
   - SEND (FB 60)
   - RECEIVE (FB 61)
   - RTU MB_MASTER (FB 72)
   - SER_CFG (FC 216)
   - SER_SND (FC 217)
   - SER_RCV (FC 218)
   - VMC_ComSlavesRTU_REF (UDT 877)
   - VMC_ComObjectRTU_REF (UDT 878)
   - VMC_AxisRTU_REF (UDT 879)
   - VMC_ConfigV1000RTU_REF (UDT 881)

7.4.2.3 Create OB 100 for serial communication



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

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



2. Enter OB 100 and confirm with [OK].
   ⇨ OB 100 is created and opened.
3. Add a `Call FB876, DB876` to the OB 100.
   ⇨ The block call is created and a dialog opens to specify the instance data block *'VMC_ConfigMaster_RTU_876'*.
4. Confirm the query of the instance data block with [OK].
5. Specify the following parameters:

`Call FB876, DB876` *Chap. 7.7.5 'FB 876 - VMC_ConfigMaster_RTU - Modbus RTU CPU interface' page 322*

Baudrate	:= `B#16#09`	// Baud rate: 09h (9600bit/s)	IN: BYTE
CharLen	:= `B#16#03`	// Number data bits: 03h (8bit)	IN: BYTE
Parity	:= `B#16#00`	// Parity: 0 (none)	IN: BYTE
StopBits	:= `B#16#01`	// Stop bits: 1 (1bit)	IN: BYTE
TimeOut	:= `W#16#1FFF`	// Error wait time: 1FFFh (high selected)	IN: WORD
Valid	:= `"ModbusConfigValid"`	// Configuration	OUT BOOL
Error	:= `"ModbusConfigError"`	// Error feedback	OUT BOOL
ErrorID	:= `"ModbusConfigErrorID"`	// Additional error information	OUT: WORD

Symbolic variable

You create the symbolic variables via *'Context menu ➔ Create / edit symbol'*. Here you can assign the corresponding operands via a dialog.

7.4.2.4 Create data block for Modbus slave

For each inverter drive, which is serially connected via Modbus RTU, a data block must be created.

1. For this click at *'Project tree ➔ ...CPU... ➔ PLC program ➔ Program blocks ➔ Add new block'*.
   ⇨ The dialog *'Add block'* is opened.
2. Select the block type *'DB block'* and assign it the name "A1_V1000". The DB number can freely be selected such as DB 100. Specify DB 100 and create this as a global DB with [OK].
   ⇨ The block is created and opened.
3. In "A1_V1000" create the following variables:
   - *'AxisData'* from Type UDT 879 - VMC_AxisRTU_REF
   - *'V1000Data'* from Type UDT 881 - VMC_ConfigV1000RTU_REF

7.4.2.5 Create data block for all Modbus slaves

For the communication data of the inverter drives, which are serially connected via Modbus RTU, a common data block is to be created.

1. For this click at *'Project tree ➔ ...CPU... ➔ PLC program ➔ Program blocks ➔ Add new block'*.

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

2. Select the block type *'DB block'* and assign it the name "ComDataSlaves". The DB number can freely be selected such as DB 99. Specify DB 99 and create this as a global DB with [OK].

   ⇨ The block is created and opened.

3. In "ComDataSlaves" create the following variable:
   - *'Slaves'* of Type UDT 877 - VMC_ComSlavesRTU_REF

7.4.2.6 OB 1 - Create instance of communication manager

The FB 877 - VMC_ComManager_RTU ensures that only 1 inverter drive (Modbus slave) can use the serial interface. As a communication manager, the block sends the jobs to the respective Modbus slaves and evaluates their responses.

1. Double-click at *'Project tree ➔ ...CPU... ➔ PLC program ➔ Program blocks ➔ Main [OB1]'*.

   ⇨ The programming window for OB 1 is opened.

2. Add a call `Call FB877, DB877` to OB 1.

   ⇨ The block call is created and a dialog opens to specify the instance data block *'VMC_ComManager_RTU_877'*.

3. Confirm the query of the instance data block with [OK].
4. Specify the following parameters:

`Call FB877, DB877` *Chap. 7.7.6 'FB 877 - VMC_ComManager_RTU - Modbus RTU communication manager' page 323*

NumberOfSlaves	:= `1`	// Number of connected inverter drives: 1	IN: INT
WaitCycles	:= `"ComWaitCycles"`	// Minimum number of waiting cycles	IN: DINT
SlavesComData	:= `"ComDataSlaves.Slave"`	// Reference to all communication objects	IN-OUT: UDT 877

7.4.2.7 OB 1 - Create instance of the V1000 initialization

The FB 881 - VMC_InitV1000_RTU initializes the corresponding inverter drive with the user data. Before an inverter drive can be controlled, it must be initialized.

1. Add a `Call FB881, DB881` to OB 1.

   ⇨ The block call is created and a dialog opens to specify the instance data block *'VMC_InitV1000_RTU _881'*.

2. Confirm the query of the instance data block with [OK].
3. Specify the following parameters:

`Call FB881, DB881` *Chap. 7.7.10 'FB 881 - VMC_InitV1000_RTU - Modbus RTU initialization' page 325*

Execute	:= `"A1_InitExecute"`	// The job is started with edge 0-1.	IN: BOOL

Hardware	:= "A1_InitHardware"	// Specification of the hardware, used // 1: System SLIO CP040, 2: SPEED7 CPU	IN: BYTE
Laddr	:= "A1_InitLaddr"	// Logical address when using CP040	IN: INT
UnitId	:= "A1_InitUnitId"	// Modbus address of the *V1000*	IN: BYTE
UserUnitsVelocity	:= "A1_InitUserUnitsVel"	// User unit for velocities: // 0: Hz, 1: %, 2: RPM	IN: INT
UserUnitsAcceleration	:= "A1_InitUserUnitsAcc"	// User units acceleration/deceleration // 0: 0.01s, 1: 0.1s	IN: INT
MaxVelocityApp	:= "A1_InitMaxVelocityApp"	// Max. velocity in user units	IN: REAL
Done	:= "A1_InitDone"	// Status job finished	OUT: BOOL
Busy	:= "A1_InitBusy"	// Status job in progress	OUT: BOOL
Error	:= "A1_InitError"	// Error feedback	OUT: BOOL
ErrorID	:= "A1_InitErrorID"	// Additional error information	OUT: WORD
Axis	:= "A1_V1000".AxisData	// Reference to the general axis data	IN-OUT: UDT 879
V1000	:= "A1_V1000".V1000Data	// Reference to the drive-specific data	IN-OUT: UDT 881

Input values

All parameters must be interconnected with the corresponding variables or operands. The following input parameters must be pre-assigned:

- Hardware
  Here specify the hardware you use to control your inverter drives:
  – 1: System SLIO CP040 whose logical address is to be specified via *Laddr*.
  – 2: SPEED7 CPU
- Laddr
  – Logical address for the System SLIO CP040 (*Hardware* = 1). Otherwise, this parameter is ignored.
- UnitId
  – Modbus address of the *V1000*.
- UserUnitsVelocity
  User unit for speeds:
  – 0: Hz
    Specified in hertz
  – 1: %
    Specified as a percentage of the maximum speed
    = $2*f_{max}/P$
    with f_{max}: max. output frequency (parameter E1-04)
    p: Number of motor poles (motor-dependent parameter E2-04, E4-04 or E5-04)
  – 2: RPM
    Data in revolutions per minute
- UserUnitsAcceleration
  User units for acceleration and deceleration
  – 0: 0.01s (range of values: 0.00s - 600.00s)
  – 1: 0.1s (range of values: 0.0 - 6000.0s)
- MaxVelocityApp
  Max. speed for the application. The specification must be made in user units and is used for synchronization in movement commands.

7.4.2.8 OB 1 - Create instance axis control V1000

With the FB 882 - VMC_AxisControlV1000_RTU you can control an inverter drive, which is serially connected via Modbus RTU and check its status.

1. Add a `Call FB882, DB882` to OB 1.
   ⇨ The block call is created and a dialog opens to specify the instance data block *'VMC_AxisControlV1000_RTU _882'*.
2. Confirm the query of the instance data block with [OK].
3. Specify the following parameters:

`Call FB882, DB882` *Chap. 7.7.11 'FB 882 - VMC_AxisControlV1000_RTU - Modbus RTU Axis control' page 327*

AxisEnable	:= `"A1_AxisEnable"`	// Activation of the axis	IN: BOOL
AxisReset	:= `"A1_AxisReset"`	// Command: Reset error of the *V1000*.	IN: BOOL
StopExecute	:= `"A1_StopExecute"`	// Command: *Stop* - Stop axis	IN: BOOL
MvVelocityExecute	:= `"A1_MvVelocityExecute"`	// Command: *MoveVelocity* (velocity control)	IN: BOOL
Velocity	:= `"A1_Velocity"`	// Parameter: Velocity setting for *MoveVelocity*	IN: REAL
AccelerationTime	:= `"A1_AccelerationTime"`	// Parameter: Acceleration time	IN: REAL
DecelerationTime	:= `"A1_DecelerationTime"`	// Parameter: Deceleration time	IN: REAL
JogPositive	:= `"A1_JogPositive"`	// Command: *JogPos*	IN: BOOL
JogNegative	:= `"A1_JogNegative"`	// Command: *JogNeg*	IN: BOOL
JogVelocity	:= `"A1_JogVelocity"`	// Parameter: Velocity setting for jogging	IN: REAL
JogAccelerationTime	:= `"A1_JogAccelerationTime"`	// Parameter: Acceleration time for jogging	IN: REAL
JogDecelerationTime	:= `"A1_JogDecelerationTime"`	// Parameter: Deceleration time for jogging	IN: REAL
AxisReady	:= `"A1_AxisReady"`	// Status: Axis ready	OUT: BOOL
AxisEnabled	:= `"A1_AxisEnabled"`	// Status: Activation of the axis	OUT: BOOL
AxisError	:= `"A1_AxisError"`	// Status: Axis error	OUT: BOOL
AxisErrorID	:= `"A1_AxisErrorID"`	// Status: Additional error information for *AxisError*	OUT: WORD
DriveError	:= `"A1_DriveError"`	// Status: Error on the inverter drive	OUT: BOOL
ActualVelocity	:= `"A1_ActualVelocity"`	// Status: Current velocity	OUT: REAL
InVelocity	:= `"A1_InVelocity"`	// Status target velocity	OUT: BOOL
CmdDone	:= `"A1_CmdDone"`	// Status: Command finished	OUT: BOOL
CmdBusy	:= `"A1_CmdBusy"`	// Status: Command in progress	OUT: BOOL
CmdAborted	:= `"A1_CmdAborted"`	// Status: Command aborted	OUT: BOOL
CmdError	:= `"A1_CmdError"`	// Status: Command error	OUT: BOOL
CmdErrorID	:= `"A1_CmdErrorID"`	// Status: Additional error information for *CmdError*	OUT: WORD
CmdActive	:= `"A1_CmdActive"`	// Status: Active command	OUT: INT
DirectionPositive	:= `"A1_DirectionPositive"`	// Status: Direction of rotation positive	OUT: BOOL
DirectionNegative	:= `"A1_DirectionNegative"`	// Status: Direction of rotation negative	OUT: BOOL
Axis	:= `"A1_V1000".AxisData`	// Reference to the general axis data	IN-OUT: UDT 879
V1000	:= `"A1_V1000".V1000Data`	// Reference to the general axis data // of the inverter drive	IN-OUT: UDT 881
AxisComData	:= `"ComDataSlaves".Slaves.Slave(1)`	// Reference to the communication data	IN-OUT: UDT 878

7.4.2.9 OB 1 - Create instance read parameter

With the FB 879 - VMC_ReadParameter_RTU you have read access to the parameters of an inverter drive, which is serially connected via Modbus RTU. For the parameter data a DB is to be created.

1. For this click at *'Project tree ➔ ...CPU... ➔ PLC program ➔ Program blocks ➔ Add new block'*.
   ⇨ The dialog *'Add block'* is opened.
2. Select the block type *'DB block'* and assign it the name "A1_TransferData". The DB number can freely be selected such as DB 98. Specify DB 98 and create this as a global DB with [OK].
   ⇨ The block is created and opened.
3. In "A1_TransferData" create the following variables:
   - *'Data_0'* of type WORD
   - *'Data_1'* of type WORD
   - *'Data_2'* of type WORD
   - *'Data_3'* of type WORD
4. Add a `Call FB879, DB879` to OB 1.
   ⇨ The block call is created and a dialog opens to specify the instance data block *'VMC_ReadParameter_RTU'*.
5. Confirm the query of the instance data block with [OK].
6. Specify the following parameters:

`Call FB879, DB879` *Chap. 7.7.8 'FB 879 - VMC_ReadParameter_RTU - Modbus RTU read parameters' page 324*

Execute	:= `"A1_RdParExecute"`	// The job is started with edge 0-1.	IN: BOOL
StartAddress	:= `"A1_RdParStartAddress"`	// Start address of the 1. register	IN: INT
Quantity	:= `"A1_RdParQuantity"`	// Number of registers to read	IN: INT
Done	:= `"A1_RdParDone"`	// Status job finished	IN: REAL
Busy	:= `"A1_RdParBusy"`	// Status job in progress	OUT: BOOL
Error	:= `"A1_RdParError"`	// Error feedback	OUT: BOOL
ErrorID	:= `"A1_RdParErrorID"`	// Additional error information	OUT: BOOL
Data	:= `P#DB98.DBX0.0 BYTES 8`	// Location of the parameter data	OUT: WORD
Axis	:= `"A1_V1000".AxisData`	// Reference to the general axis data	IN-OUT: UDT 879

Please note that only whole registers can be read as WORD. To evaluate individual bits, you must swap high and low byte!

7.4.2.10 OB 1 - Create instance write parameter

With the FB 880 - VMC_WriteParameter_RTU you have write access to the parameters of an inverter drive, which is serially connected via Modbus RTU. For the data you can use the DB created for read access - here DB 98.

1. Add a `Call FB880, DB880` to OB 1.
   ⇨ The block call is created and a dialog opens to specify the instance data block *'VMC_WriteParameter_RTU'*.

2. Confirm the query of the instance data block with [OK].
3. Specify the following parameters:

`Call FB880, DB880` *Chap. 7.7.9 'FB 880 - VMC_WriteParameter_RTU - Modbus RTU write parameters' page 325*

Execute	:= `"A1_WrParExecute"`	// The job is started with edge 0-1.	IN: BOOL
StartAddress	:= `"A1_WrParStartAddress"`	// Start address of the 1. register	IN: INT
Quantity	:= `"A1_WrParQuantity"`	// Number of registers to write	IN: INT
Done	:= `"A1_WrParDone"`	// Status job finished	IN: REAL
Busy	:= `"A1_WrParBusy"`	// Status job in progress	OUT: BOOL
Error	:= `"A1_WrParError"`	// Error feedback	OUT: BOOL
ErrorID	:= `"A1_WrParErrorID"`	// Additional error information	OUT: BOOL
Data	:= `P#DB98.DBX0.0 BYTES 8`	// Location of the parameter data	OUT: WORD
Axis	:= `"A1_V1000".AxisData`	// Reference to the general axis data	IN-OUT: UDT 879

7.4.2.11 Sequence of operations

1. Select *'Project → Compile all'* and transfer the project into your CPU.

   You can find more information on the transfer of your project in the online help of the *SPEED7 Studio*.

   ⇨ You can now take your application into operation via the existing communication connection.

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

2. A watch table allows you to manually control the inverter drive. Double-click at *'Project tree → ...CPU... → PLC program → Watch tables → Add watch table'*.
3. Enter a name for the watch table such as *'V1000'* and confirm with [OK]

   ⇨ The watch table is created and opened for editing.

4. First adjust the waiting time between 2 jobs. This is at least 200ms for a V1000 inverter drive. For this enter in the watch table at *'Name'* the designation *'ComWaitCycles'* as *'Decimal'* and enter at *'Control value'* a value between 200 and 400.

To increase performance, you can later correct this to a smaller value as long as you do not receive a timeout error (80C8h). Please note that some commands, such as MoveVelocity, can consist of several jobs.

5. Before you can control an inverter drive, it must be initialized with FB 881 - VMC_InitV1000_RTU. *Chap. 7.7.10 'FB 881 - VMC_InitV1000_RTU - Modbus RTU initialization' page 325*

   For this enter in the watch table at *'Name'* the designation *'A1_InitExecute'* as *'Boolean'* and enter at *'Control value'* the value *'True'*. Activate *'Control'* and start the transfer of the control values.

   ⇨ The inverter drive is initialized. After execution, the output *Done* returns TRUE. In the event of a fault, you can determine the error by evaluating the *ErrorID*.

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

6. After successful initialization, the registers of the connected inverter drives are cyclically processed, i.e. they receive cyclical jobs. For manual control, you can use the FB 882 - VMC_AxisControlV1000_RTU to send control commands to the appropriate inverter drive. *Chap. 7.7.11 'FB 882 - VMC_AxisControlV1000_RTU - Modbus RTU Axis control' page 327*
7. Create the parameters of the FB 882 - VMC_AxisControlV1000_RTU for control and query in the watch table.
8. Activate the corresponding axis by setting *AxisEnable*. As soon as this reports *AxisReady* = TRUE, you can control it with the corresponding drive commands.

7.5 Usage in Siemens SIMATIC Manager

7.5.1 Precondition

Overview

- Please use for configuration the Siemens SIMATIC Manager V 5.5 SP2 and up.
- With a System MICRO CPU, plugging the expansion module activates the PtP functionality. The configuration happens in the Siemens SIMATIC Manager by means of a virtual PROFINET IO device. The PROFINET IO device is to be installed in the hardware catalog by means of a GSDML.
- With a System SLIO 013C CPU the configuration of PtP functionality happens in the Siemens SIMATIC Manager by means of a virtual PROFINET IO device. The PROFINET IO device is to be installed in the hardware catalog by means of a GSDML.
- With the System SLIO CPUs 014 ... 017, the RS485 interface is set to PtP communication as standard. The configuration happens in the Siemens SIMATIC Manager by means of a virtual PROFINET IO device. The PROFINET IO device is to be installed in the hardware catalog by means of a GSDML.

Installing the VIPA IO device

The installation of the PROFINET VIPA IO device 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 ...'*.

7.5.2 Hardware configuration

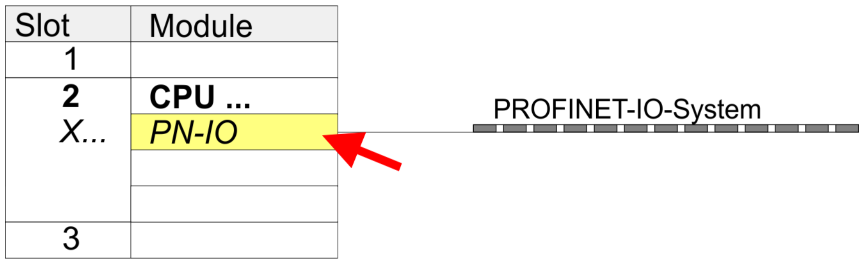
7.5.2.1 Hardware configuration System MICRO

Add CPU in the project

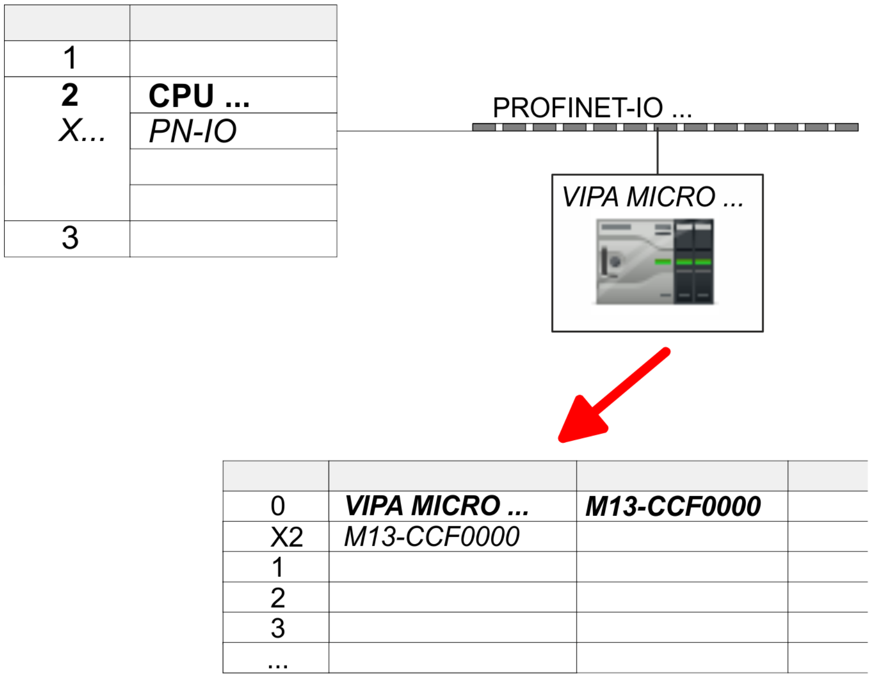
Slot	Module
1	
2	**CPU 314C-2PN/DP**
X1	*MPI/DP*
X2	*PN-IO*
X2...	*Port 1*
X2...	*Port 2*
...	...
3	

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 314C-2 PN/DP (314-6EH04-0AB0 V3.3).
4. Click at the sub module *'PN-IO'* of the CPU.
5. Select *'Context menu ➔ Insert PROFINET IO System'*.



6. Create with [New] a new sub net and assign valid address data.
7. Click at the sub module *'PN-IO'* of the CPU and open with *'Context menu ➔ Properties'* the properties dialog.
8. Enter at *'General'* a *'Device name'*. The device name must be unique at the Ethernet subnet.



9. Navigate in the hardware catalog to the directory *'PROFINET IO ➔ Additional field devices ➔ I/O ➔ VIPA ...'* and connect e.g. for the System MICRO the IO device *'M13-CCF0000'* to your PROFINET system.
   ⇨ In the *Device overview* of the PROFINET IO device *'VIPA MICRO PLC'* the CPU is already placed at slot 0.

Configuration of Ethernet PG/OP channel

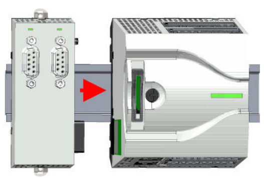
Slot	Module
1	
2	**CPU ...**
X...	*PN-IO*
3	
4	343-1EX30
5	
...	

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!

Enable PtP functionality

A hardware configuration to enable the PtP functionality is not necessary.

1. Turn off the power supply.



2. Mount the extension module.
3. Establish a cable connection to the communication partner.

4. Switch on the power supply.
   ⇨ After a short boot time the interface X1 PtP is ready for PtP communication.

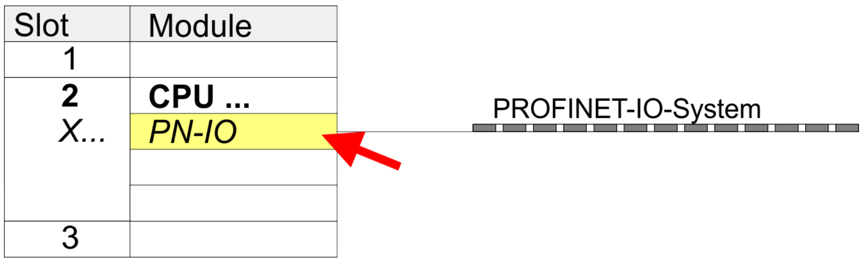
7.5.2.2 Hardware configuration System SLIO CPU 013C

Add CPU in the project

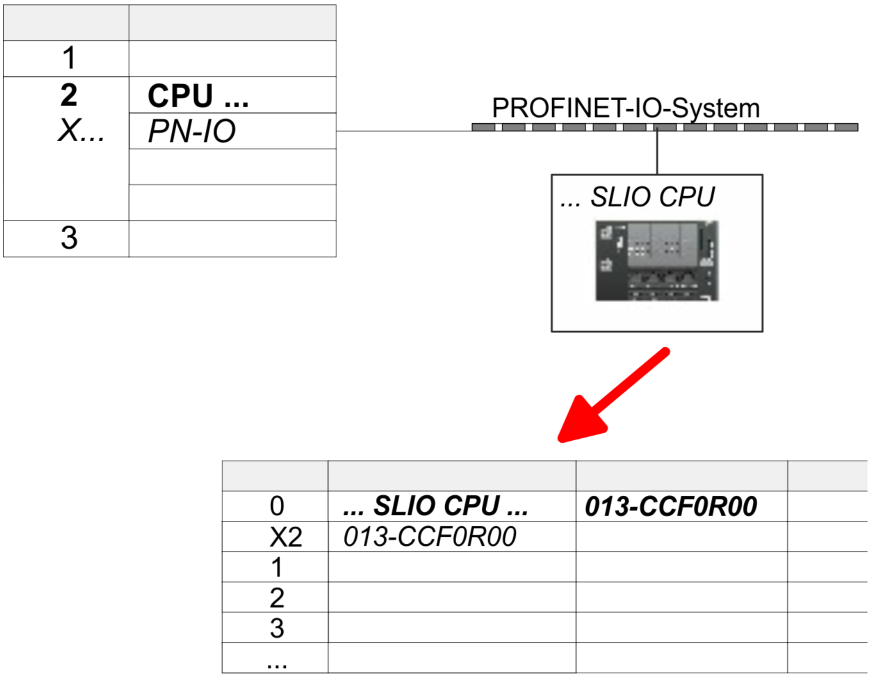
Slot	Module
1	
2	**CPU 314C-2PN/DP**
X1	*MPI/DP*
X2	*PN-IO*
X2...	*Port 1*
X2...	*Port 2*
...	...
3	

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 314C-2 PN/DP (314-6EH04-0AB0 V3.3).
4. Click at the sub module *'PN-IO'* of the CPU.
5. Select *'Context menu ➔ Insert PROFINET IO System'*.



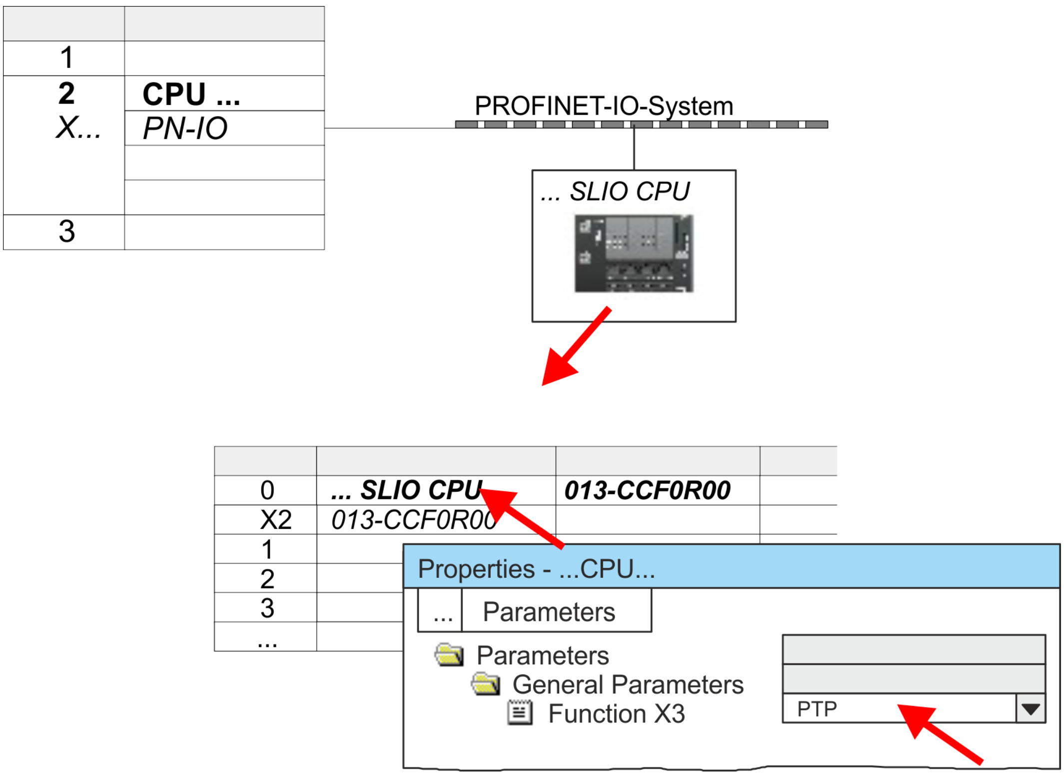
6. Use [New] to create a new subnet and assign valid IP address data for your PROFINET system.
7. Click at the sub module *'PN-IO'* of the CPU and open with *'Context menu ➔ Properties'* the properties dialog.
8. Enter at *'General'* a *'Device name'*. The device name must be unique at the Ethernet subnet.



9. Navigate in the hardware catalog to the directory *'PROFINET IO ➔ Additional field devices ➔ I/O ➔ VIPA ...'* and connect the IO device *'013-CCF0R00'* CPU 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.

Enable PtP functionality



1. Open the properties dialog by a double-click at *'VIPA SLIO CPU'*.

   ⇨ The VIPA specific parameters may be accessed by means of the properties dialog.

2. Select at *'Function X3'* the value *'PTP'*.

Configuration of Ethernet PG/OP channel

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!

7.5.2.3 Hardware configuration System SLIO CPU 014 ... 017

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	

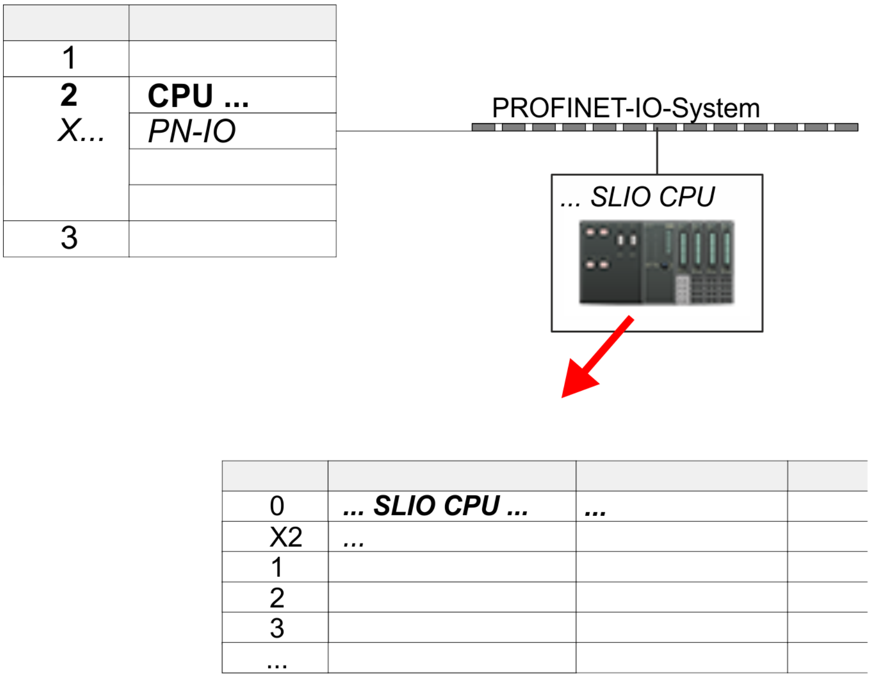
To be compatible with the Siemens SIMATIC Manager the following steps should be executed:

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 the CPU 315-2 PN/DP (315-2EH14-0AB0 V3.2).
4. Click at the sub module *'PN-IO'* of the CPU.

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

PROFINET-IO-System

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.



8. Navigate in the hardware catalog to the directory *'PROFINET IO ➔ Additional field devices ➔ I/O ➔ VIPA ...'* and connect the IO device, which corresponds to your CPU, 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

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!

Enable PtP functionality

For the System SLIO CPUs 014 ... 017, the RS485 interface is set to PtP communication as standard. A hardware configuration to enable the PtP functionality is not necessary.

7.5.3 User program

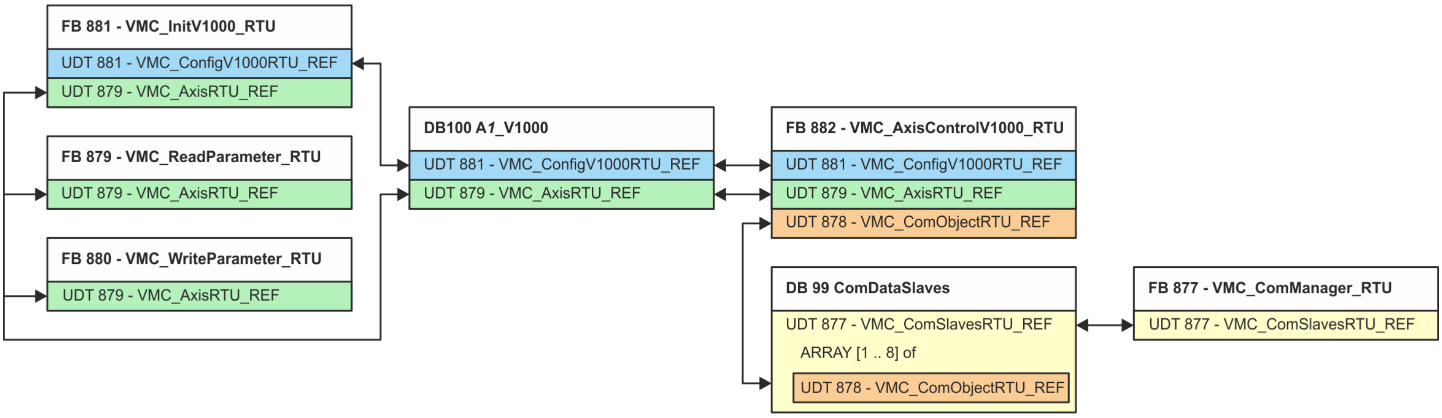
7.5.3.1 Program structure

OB 100

FB 876 - VMC_ConfigMaster_RTU
SFC 216 - SER_CFG

- FB 876 - VMC_ConfigMaster_RTU 🡆 *322*
  – This block is used to parametrize the serial interface of the CPU for Modbus RTU communication.
  – Internally block SFC 216 - SER_CFG is called.

OB 1



With the exception of blocks DB 99 and FB 877, you must create the blocks listed below for each connected inverter drive:

- FB 881 - VMC_InitV1000_RTU ↬ *325*
  - The FB 881 - VMC_InitV1000_RTU initializes the corresponding inverter drive with the user data.
  - Before an inverter drive can be controlled, it must be initialized.
  - UDT 881 - VMC_ConfigV1000RTU_REF ↬ *322*
  - UDT 879 - VMC_AxisRTU_REF ↬ *322*
- FB 879 - VMC_ReadParameter_RTU ↬ *324*
  - With this FB you have read access to the parameters of an inverter drive, which is connected serially via Modbus RTU.
  - The read data are recorded in a data block.
  - UDT 879 - VMC_AxisRTU_REF ↬ *322*
- FB 880 - VMC_WriteParameter_RTU ↬ *325*
  - With this FB you have read access to the parameters of an inverter drive, which is connected serially via Modbus RTU.
  - The data to be written must be stored in a data block.
  - UDT 879 - VMC_AxisRTU_REF ↬ *322*
- DB 100 - A1_V1000
  - For each inverter drive, which is serially connected via Modbus RTU, a data block must be created.
  - UDT 879 - VMC_AxisRTU_REF ↬ *322*
  - UDT 881 - VMC_ConfigV1000RTU_REF ↬ *322*
- FB 882 - VMC_AxisControlV1000_RTU ↬ *327*
  - With this block, you can control an inverter drive, which is serially connected via Modbus RTU and check its status.
  - UDT 881 - VMC_ConfigV1000RTU_REF ↬ *322*
  - UDT 879 - VMC_AxisRTU_REF ↬ *322*
  - UDT 878 - VMC_ComObjectRTU_REF ↬ *322*
- DB 99 - ComDataSlaves
  - For the communication data of all the inverter drives (max. 8), which are serially connected via Modbus RTU, a common data block is to be created.
  - UDT 877 - VMC_ComSlavesRTU_REF ↬ *322*
  - UDT 878 - VMC_ComObjectRTU_REF ↬ *322*
- FB 877 - VMC_ComManager_RTU ↬ *323*
  - The device ensures that only 1 inverter drive (Modbus slave) can use the serial interface. If several inverter drives are used, this block, as communication manager, sends the jobs to the respective Modbus slaves and evaluates their responses.
  - UDT 877 - VMC_ComSlavesRTU_REF ↬ *322*

7.5.3.2 Copy blocks into project

Include library

1. Go to the service area of www.vipa.com.
2. Download the *Simple Motion Control* library from the download area at *'VIPA Lib'*.
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 all the blocks of *'V1000 Modbus RTU'* into *'Blocks'* of your project:

- FB 876 - VMC_ConfigMaster_RTU
- FB 877 - VMC_ComManager_RTU
- FB 878 - VMC_RWParameterSys_RTU
- FB 879 - VMC_ReadParameter_RTU
- FB 880 - VMC_WriteParameter_RTU
- FB 881 - VMC_InitV1000_RTU
- FB 882 - VMC_AxisControlV1000_RTU
- FB 60 - SEND
- FB 61 - RECEIVE
- FB 72 - RTU MB_MASTER
- FC 216 - SER_CFG
- FC 217 - SER_SND
- FC 218 - SER_RCV
- UDT 877 - VMC_ComSlavesRTU_REF
- UDT 878 - VMC_ComObjectRTU_REF
- UDT 879 - VMC_AxisRTU_REF
- UDT 881 - VMC_ConfigV1000RTU_REF
- SFB 4 - TON

7.5.3.3 Create OB 100 for serial communication

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 the OB 100 to your project.
3. Open the OB 100.
4. Add a `Call FB876, DB876` to the OB 100.
   ⇨ The block call is created and a dialog opens to specify the instance data block *'VMC_ConfigMaster_RTU_876'*.
5. Specify the following parameters:

`Call FB876, DB876` *Chap. 7.7.5 'FB 876 - VMC_ConfigMaster_RTU - Modbus RTU CPU interface' page 322*

Baudrate	:= `B#16#09`	// Baud rate: 09h (9600bit/s)	IN: BYTE
CharLen	:= `B#16#03`	// Number data bits: 03h (8bit)	IN: BYTE
Parity	:= `B#16#00`	// Parity: 0 (none)	IN: BYTE
StopBits	:= `B#16#01`	// Stop bits: 1 (1bit)	IN: BYTE

TimeOut	:= `W#16#1FFF`	// Error wait time: 1FFFh (high selected)	IN: WORD
Valid	:= `"ModbusConfigValid"`	// Configuration	OUT BOOL
Error	:= `"ModbusConfigError"`	// Error feedback	OUT BOOL
ErrorID	:= `"ModbusConfigErrorID"`	// Additional error information	OUT: WORD

Symbolic variable

You create the symbolic variables via *'Context menu ➔ Edit symbol'*. Here you can assign the corresponding operand via a dialog.

7.5.3.4 Create data block for Modbus slave

For each inverter drive, which is serially connected via Modbus RTU, a data block must be created.

1. In your project, click at *'Blocks'* and choose *'Context menu ➔ Insert new object ➔ Data block'*.
   ⇨ The dialog *'Add block'* is opened.
2. Specify the following parameters:
   - Name and type
     – The DB number as *'Name'* can freely be chosen, such as DB 100. Enter DB 100.
     – Set *'Shared DB'* as the *'Type'*.
   - Symbolic name
     – Enter "A1_V1000".

   Confirm your input with [OK].

   ⇨ The block is created.
3. Open DB 100 "A1_V1000" by double-clicking.
4. In "A1_V1000" create the following variables:
   - *'AxisData'* of type UDT 879 - VMC_AxisRTU_REF
   - *'V1000Data'* of type UDT 881 - VMC_ConfigV1000RTU_REF

7.5.3.5 Create data block for all Modbus slaves

For the communication data of the inverter drives, which are serially connected via Modbus RTU, a common data block is to be created.

1. In your project, click at *'Blocks'* and choose *'Context menu ➔ Insert new object ➔ Data block'*.
   ⇨ The dialog *'Add block'* is opened.
2. Specify the following parameters:
   - Name and type
     – The DB number as *'Name'* can freely be chosen, such as DB 99. Enter DB 99.
     – Set *'Shared DB'* as the *'Type'*.
   - Symbolic name
     – Enter "ComDataSlaves".

   Confirm your input with [OK].

   ⇨ The block is created.

3. Open DB 99 "ComDataSlaves" by double-clicking.
4. In "ComDataSlaves" create the following variable:
   - *'Slaves'* of Type UDT 877 - VMC_ComSlavesRTU_REF

7.5.3.6 OB 1 - Create instance of communication manager

The FB 877 - VMC_ComManager_RTU ensures that only 1 inverter drive (Modbus slave) can use the serial interface. As a communication manager, the block sends the jobs to the respective Modbus slaves and evaluates their responses.

1. Open the OB 1.
2. Add a `Call FB877, DB877` to OB 1.
   ⇨ The block call is created and a dialog opens to specify the instance data block *'VMC_ComManager_RTU_877'*.
3. Confirm the query of the instance data block with [OK].
4. Specify the following parameters:

`Call FB877, DB877` *Chap. 7.7.6 'FB 877 - VMC_ComManager_RTU - Modbus RTU communication manager' page 323*

NumberOfSlaves	:= `1`	// Number of connected inverter drives: 1	IN: INT
WaitCycles	:= `"ComWaitCycles"`	// Minimum number of waiting cycles	IN: DINT
SlavesComData	:= `"ComDataSlaves.Slave"`	// Reference to all communication objects	IN-OUT: UDT 877

7.5.3.7 OB 1 - Create instance of the V1000 initialization

The FB 881 - VMC_InitV1000_RTU initializes the corresponding inverter drive with the user data. Before an inverter drive can be controlled, it must be initialized.

1. Add a `Call FB881, DB881` to OB 1.
   ⇨ The block call is created and a dialog opens to specify the instance data block *'VMC_InitV1000_RTU _881'*.
2. Confirm the query of the instance data block with [OK].
3. Specify the following parameters:

`Call FB881, DB881` *Chap. 7.7.10 'FB 881 - VMC_InitV1000_RTU - Modbus RTU initialization' page 325*

Execute	:= `"A1_InitExecute"`	// The job is started with edge 0-1.	IN: BOOL
Hardware	:= `"A1_InitHardware"`	// Specification of the hardware, used // 1: System SLIO CP040, 2: SPEED7 CPU	IN: BYTE
Laddr	:= `"A1_InitLaddr"`	// Logical address when using CP040	IN: INT
UnitId	:= `"A1_InitUnitId"`	// Modbus address of the *V1000*	IN: BYTE
UserUnitsVelocity	:= `"A1_InitUserUnitsVel"`	// User unit for velocities: // 0: Hz, 1: %, 2: RPM	IN: INT
UserUnitsAcceleration	:= `"A1_InitUserUnitsAcc"`	// User units acceleration/deceleration // 0: 0.01s, 1: 0.1s	IN: INT
MaxVelocityApp	:= `"A1_InitMaxVelocityApp"`	// Max. velocity in user units	IN: REAL
Done	:= `"A1_InitDone"`	// Status job finished	OUT: BOOL

Busy	:= "A1_InitBusy"	// Status job in progress	OUT: BOOL
Error	:= "A1_InitError"	// Error feedback	OUT: BOOL
ErrorID	:= "A1_InitErrorID"	// Additional error information	OUT: WORD
Axis	:= "A1_V1000".AxisData	// Reference to the general axis data	IN-OUT: UDT 879
V1000	:= "A1_V1000".V1000Data	// Reference to the drive-specific data	IN-OUT: UDT 881

Input values

All parameters must be interconnected with the corresponding variables or operands. The following input parameters must be pre-assigned:

- Hardware
  Here specify the hardware you use to control your inverter drives:
  – 1: System SLIO CP040 whose logical address is to be specified via *Laddr*.
  – 2: SPEED7 CPU
- Laddr
  – Logical address for the System SLIO CP040 (*Hardware* = 1). Otherwise, this parameter is ignored.
- UnitId
  – Modbus address of the *V1000*.
- UserUnitsVelocity
  User unit for speeds:
  – 0: Hz
  Specified in hertz
  – 1: %
  Specified as a percentage of the maximum speed
  = 2*f_{max}/P
  with f_{max}: max. output frequency (parameter E1-04)
  p: Number of motor poles (motor-dependent parameter E2-04, E4-04 or E5-04)
  – 2: RPM
  Data in revolutions per minute
- UserUnitsAcceleration
  User units for acceleration and deceleration
  – 0: 0.01s (range of values: 0.00s - 600.00s)
  – 1: 0.1s (range of values: 0.0 - 6000.0s)
- MaxVelocityApp
  Max. speed for the application. The specification must be made in user units and is used for synchronization in movement commands.

7.5.3.8 OB 1 - Create instance axis control V1000

With the FB 882 - VMC_AxisControlV1000_RTU you can control an inverter drive, which is serially connected via Modbus RTU and check its status.

1. Add a `Call FB882, DB882` to OB 1.
   ⇨ The block call is created and a dialog opens to specify the instance data block *'VMC_AxisControlV1000_RTU _882'*.
2. Confirm the query of the instance data block with [OK].
3. Specify the following parameters:

`Call FB882, DB882` *Chap. 7.7.11 'FB 882 - VMC_AxisControlV1000_RTU - Modbus RTU Axis control' page 327*

AxisEnable	:= "A1_AxisEnable"	// Activation of the axis	IN: BOOL

AxisReset	`:= "A1_AxisReset"`	// Command: Reset error of the *V1000*.	IN: BOOL
StopExecute	`:= "A1_StopExecute"`	// Command: *Stop* - Stop axis	IN: BOOL
MvVelocityExecute	`:= "A1_MvVelocityExecute"`	// Command: *MoveVelocity* (velocity control)	IN: BOOL
Velocity	`:= "A1_Velocity"`	// Parameter: Velocity setting for *MoveVelocity*	IN: REAL
AccelerationTime	`:= "A1_AccelerationTime"`	// Parameter: Acceleration time	IN: REAL
DecelerationTime	`:= "A1_DecelerationTime"`	// Parameter: Deceleration time	IN: REAL
JogPositive	`:= "A1_JogPositive"`	// Command: *JogPos*	IN: BOOL
JogNegative	`:= "A1_JogNegative"`	// Command: *JogNeg*	IN: BOOL
JogVelocity	`:= "A1_JogVelocity"`	// Parameter: Velocity setting for jogging	IN: REAL
JogAccelerationTime	`:= "A1_JogAccelerationTime"`	// Parameter: Acceleration time for jogging	IN: REAL
JogDecelerationTime	`:= "A1_JogDecelerationTime"`	// Parameter: Deceleration time for jogging	IN: REAL
AxisReady	`:= "A1_AxisReady"`	// Status: Axis ready	OUT: BOOL
AxisEnabled	`:= "A1_AxisEnabled"`	// Status: Activation of the axis	OUT: BOOL
AxisError	`:= "A1_AxisError"`	// Status: Axis error	OUT: BOOL
AxisErrorID	`:= "A1_AxisErrorID"`	// Status: Additional error information for *AxisError*	OUT: WORD
DriveError	`:= "A1_DriveError"`	// Status: Error on the inverter drive	OUT: BOOL
ActualVelocity	`:= "A1_ActualVelocity"`	// Status: Current velocity	OUT: REAL
InVelocity	`:= "A1_InVelocity"`	// Status target velocity	OUT: BOOL
CmdDone	`:= "A1_CmdDone"`	// Status: Command finished	OUT: BOOL
CmdBusy	`:= "A1_CmdBusy"`	// Status: Command in progress	OUT: BOOL
CmdAborted	`:= "A1_CmdAborted"`	// Status: Command aborted	OUT: BOOL
CmdError	`:= "A1_CmdError"`	// Status: Command error	OUT: BOOL
CmdErrorID	`:= "A1_CmdErrorID"`	// Status: Additional error information for *CmdError*	OUT: WORD
CmdActive	`:= "A1_CmdActive"`	// Status: Active command	OUT: INT
DirectionPositive	`:= "A1_DirectionPositive"`	// Status: Direction of rotation positive	OUT: BOOL
DirectionNegative	`:= "A1_DirectionNegative"`	// Status: Direction of rotation negative	OUT: BOOL
Axis	`:= "A1_V1000".AxisData`	// Reference to the general axis data	IN-OUT: UDT 879
V1000	`:= "A1_V1000".V1000Data`	// Reference to the general axis data // of the inverter drive	IN-OUT: UDT 881
AxisComData	`:= "ComDataSlaves".Slaves.Slave(1)`	// Reference to the communication data	IN-OUT: UDT 878

7.5.3.9 OB 1 - Create instance read parameter

With the FB 879 - VMC_ReadParameter_RTU you have read access to the parameters of an inverter drive, which is serially connected via Modbus RTU. For the parameter data a DB is to be created.

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

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

2. Specify the following parameters:
   - Name and type
     – The DB no. as *'Name'* can freely be chosen, such as DB 98. Enter DB 98.
     – Set *'Shared DB'* as the *'Type'*.
   - Symbolic name
     – Enter "A1_TransferData".

   Confirm your input with [OK].

   ⇨ The block is created.
3. Open DB 98 "A1_TransferData" by double-clicking.
4. In "A1_TransferData" create the following variables:
   - *'Data_0'* of type WORD
   - *'Data_1'* of type WORD
   - *'Data_2'* of type WORD
   - *'Data_3'* of type WORD
5. Add a `Call FB879, DB879` to OB 1.

   ⇨ The block call is created and a dialog opens to specify the instance data block *'VMC_ReadParameter_RTU'*.
6. Confirm the query of the instance data block with [OK].
7. Specify the following parameters:

`Call FB879, DB879` *Chap. 7.7.8 'FB 879 - VMC_ReadParameter_RTU - Modbus RTU read parameters' page 324*

Execute	:= `"A1_RdParExecute"`	// The job is started with edge 0-1.	IN: BOOL
StartAddress	:= `"A1_RdParStartAddress"`	// Start address of the 1. register	IN: INT
Quantity	:= `"A1_RdParQuantity"`	// Number of registers to read	IN: INT
Done	:= `"A1_RdParDone"`	// Status job finished	IN: REAL
Busy	:= `"A1_RdParBusy"`	// Status job in progress	OUT: BOOL
Error	:= `"A1_RdParError"`	// Error feedback	OUT: BOOL
ErrorID	:= `"A1_RdParErrorID"`	// Additional error information	OUT: BOOL
Data	:= `P#DB98.DBX0.0 BYTES 8`	// Location of the parameter data	OUT: WORD
Axis	:= `"A1_V1000".AxisData`	// Reference to the general axis data	IN-OUT: UDT 879

Please note that only whole registers can be read as WORD. To evaluate individual bits, you must swap high and low byte!

7.5.3.10 OB 1 - Create instance write parameter

With the FB 880 - VMC_WriteParameter_RTU you have write access to the parameters of an inverter drive, which is serially connected via Modbus RTU. For the data you can use the DB created for read access - here DB 98.

1. Add a `Call FB880, DB880` to OB 1.

   ⇨ The block call is created and a dialog opens to specify the instance data block *'VMC_WriteParameter_RTU'*.
2. Confirm the query of the instance data block with [OK].

3. Specify the following parameters:

`Call FB880, DB880` *Chap. 7.7.9 'FB 880 - VMC_WriteParameter_RTU - Modbus RTU write parameters' page 325*

Execute	:= `"A1_WrParExecute"`	// The job is started with edge 0-1.	IN: BOOL
StartAddress	:= `"A1_WrParStartAddress"`	// Start address of the 1. register	IN: INT
Quantity	:= `"A1_WrParQuantity"`	// Number of registers to write	IN: INT
Done	:= `"A1_WrParDone"`	// Status job finished	IN: REAL
Busy	:= `"A1_WrParBusy"`	// Status job in progress	OUT: BOOL
Error	:= `"A1_WrParError"`	// Error feedback	OUT: BOOL
ErrorID	:= `"A1_WrParErrorID"`	// Additional error information	OUT: BOOL
Data	:= `P#DB98.DBX0.0 BYTES 8`	// Location of the parameter data	OUT: WORD
Axis	:= `"A1_V1000".AxisData`	// Reference to the general axis data	IN-OUT: UDT 879

7.5.3.11 Sequence of operations

1. Safe your project with *'Station ➔ Safe and compile'*.
2. Transfer your project to your CPU.
   ⇨ You can take your application into operation now.

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

3. A watch table allows you to manually control the inverter drive. To create a watch table, choose *'PLC ➔ Monitor/Modify variables'*.
   ⇨ The watch table is created and opened for editing.
4. First adjust the waiting time between 2 jobs. This is at least 200ms for a V1000 inverter drive. For this enter in the watch table at *'Symbol'* the designation *'ComWaitCycles'* as *'Decimal'* and enter at *'Control value'* a value between 200 and 400.

> *To increase performance, you can later correct this to a smaller value as long as you do not receive a timeout error (80C8h). Please note that some commands, such as MoveVelocity, can consist of several jobs.*

5. Before you can control an inverter drive, it must be initialized with FB 881 - VMC_InitV1000_RTU. *Chap. 7.7.10 'FB 881 - VMC_InitV1000_RTU - Modbus RTU initialization' page 325*

   For this enter in the watch table at *'Symbol'* the designation *'A1_InitExecute'* as *'Boolean'* and enter at *'Control value'* the value *'True'*. Activate *'Control'* and start the transfer of the control values.

   ⇨ The inverter drive is initialized. After execution, the output *Done* returns TRUE. In the event of a fault, you can determine the error by evaluating the *ErrorID*.

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

6. After successful initialization, the registers of the connected inverter drives are cyclically processed, i.e. they receive cyclical jobs. For manual control, you can use the FB 882 - VMC_AxisControlV1000_RTU to send control commands to the appropriate inverter drive. *Chap. 7.7.11 'FB 882 - VMC_AxisControlV1000_RTU - Modbus RTU Axis control' page 327*
7. Create the parameters of the FB 882 - VMC_AxisControlV1000_RTU for control and query in the watch table.
8. Save the watch table under a name such as *'V1000'*.
9. Activate the corresponding axis by setting *AxisEnable*. As soon as this reports *AxisReady* = TRUE, you can control it with the corresponding drive commands.

7.6 Usage in Siemens TIA Portal

7.6.1 Precondition

Overview

- Please use the Siemens TIA Portal V 14 and up for the configuration.
- With a System MICRO CPU, plugging the expansion module activates the PtP functionality. The configuration happens in the Siemens TIA Portal by means of a virtual PROFINET IO device. The PROFINET IO device is to be installed in the hardware catalog by means of a GSDML.
- With a System SLIO 013C CPU the configuration of PtP functionality happens in the Siemens TIA Portal by means of a virtual PROFINET IO device. The PROFINET IO device is to be installed in the hardware catalog by means of a GSDML.
- With the System SLIO CPUs 014 ... 017, the RS485 interface is set to PtP communication as standard. The configuration happens in the Siemens TIA Portal by means of a virtual PROFINET IO device. The PROFINET IO device is to be installed in the hardware catalog by means of a GSDML.

Installing the VIPA IO device

The installation of the PROFINET VIPA IO device 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 ... > ....*

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

7.6.2 Hardware configuration

7.6.2.1 Hardware configuration System MICRO

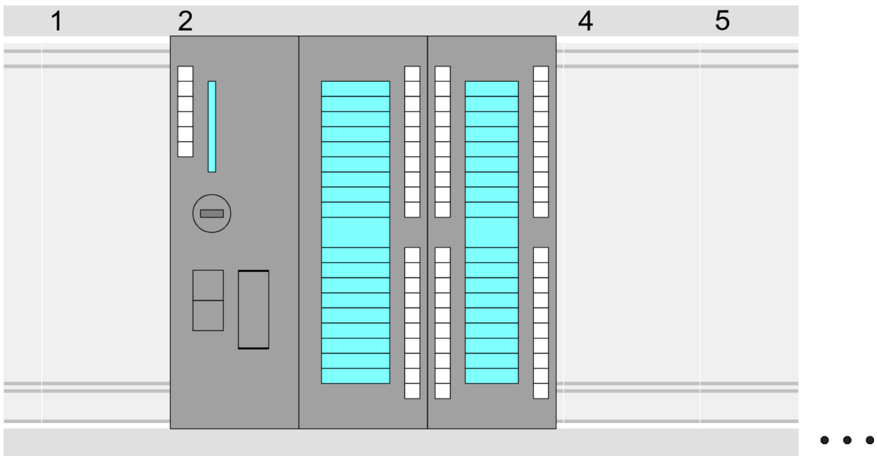
Add CPU in the project

To be compatible with the Siemens SIMATIC 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. Select the following CPU in the input dialog:

   SIMATIC S7-300 > CPU 314C-2 PN/DP (314-6EH04-0AB0 V3.3)

   ⇨ The CPU is inserted with a profile rail.



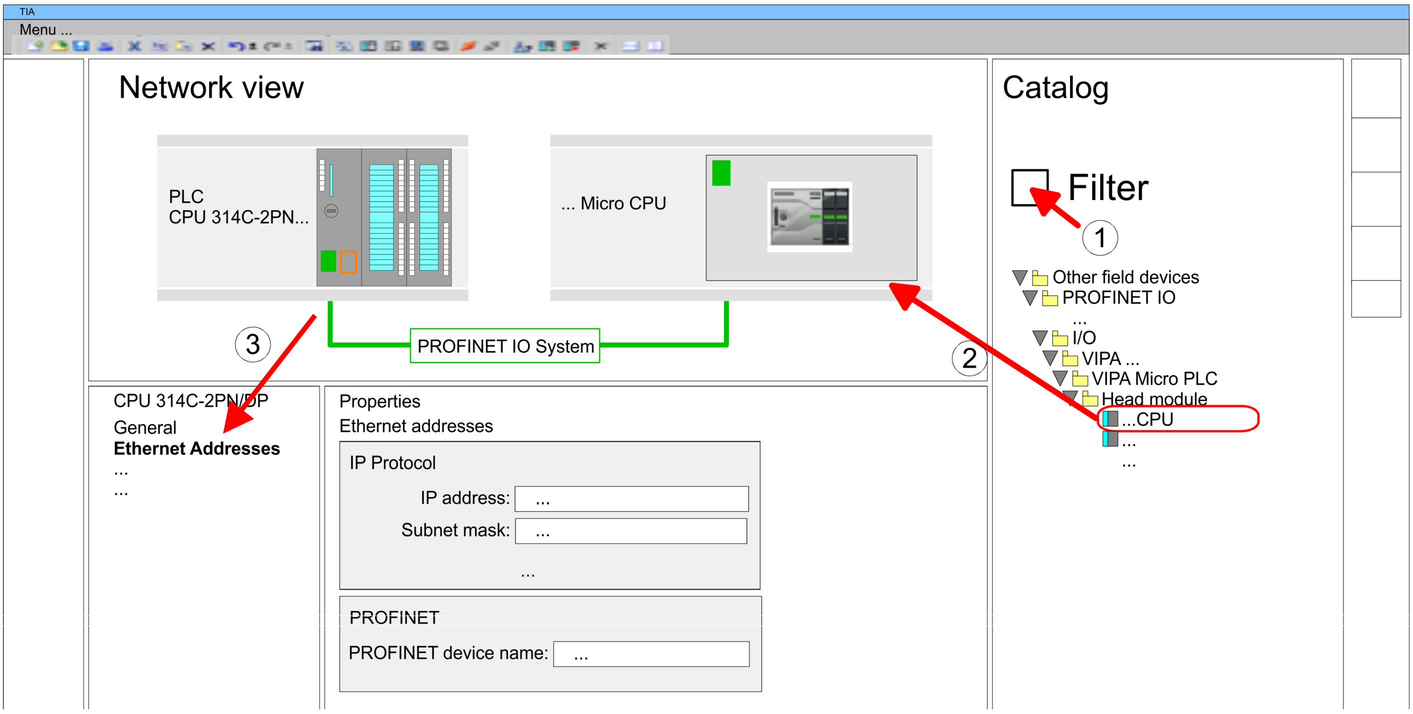
Device overview:

Module	...	Slot	...	Type	...
PLC...		2		CPU 314C-2PN/DP	
MPI interface...		2 X1		MPI/DP interface	
PROFINET interface...		2 X2		PROFINET interface	
DI24/DO16...		2 5		DI24/DO16	
AI5/AO2...		2 6		AI5/AO2	

Count...		2 7		Count	
...					

Connection CPU as PROFINET IO device

1. Switch in the *Project area* to *'Network view'*.
2. After installing the GSDML the IO device for the SLIO CPU may be found in the hardware catalog at *Other field devices > PROFINET > IO > VIPA ... > VIPA MICRO PLC*. 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 at 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.



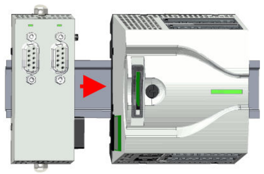
5. Select in the *Network view* the IO device *'VIPA MICRO PLC'* and switch to the *Device overview*.

   ⇨ In the *Device overview* of the PROFINET IO device *'VIPA MICRO PLC'* the CPU is already placed at slot 0.

Enable PtP functionality

A hardware configuration to enable the PtP functionality is not necessary.

1. Turn off the power supply.

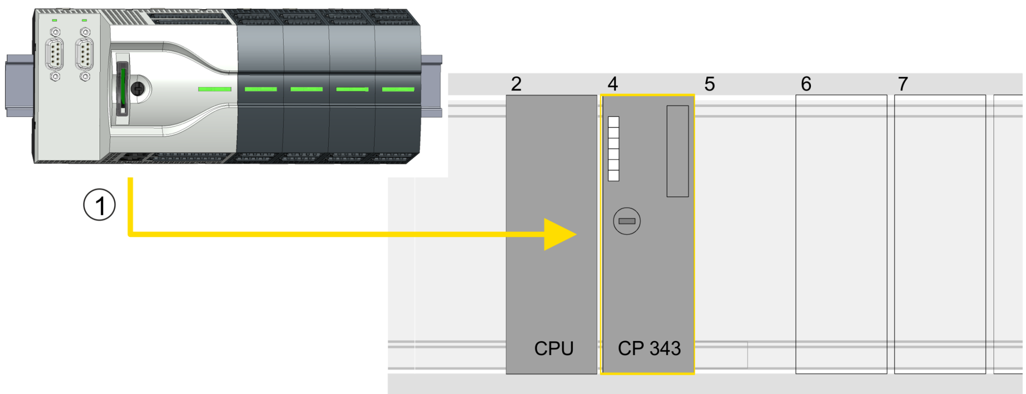


2. Mount the extension module.
3. Establish a cable connection to the communication partner.

4. Switch on the power supply.
   ⇨ After a short boot time the interface X1 PtP is ready for PtP communication.

Configuration of Ethernet PG/OP channel

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. You get valid IP address parameters from your system administrator.



1 Ethernet PG/OP channel

Device overview

Module	...	Slot	...	Type	...
PLC ...		2		CPU 314C-2PN/DP	
MPI/DP interface		2 X1		MPI/DP interface	
PROFINET inter-face		2 X2		PROFINET interface	
...		...		...	
CP 343-1		4		CP 343-1	
...		...		...	

7.6.2.2 Hardware configuration System SLIO CPU 013C

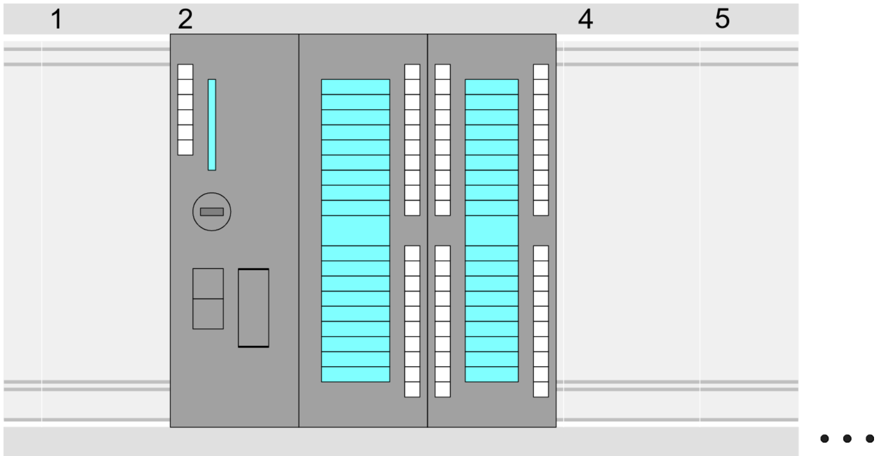
Add CPU in the project

To be compatible with the Siemens SIMATIC 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. Select the following CPU in the input dialog:

   SIMATIC S7-300 > CPU 314C-2 PN/DP (314-6EH04-0AB0 V3.3)

   ⇨ The CPU is inserted with a profile rail.

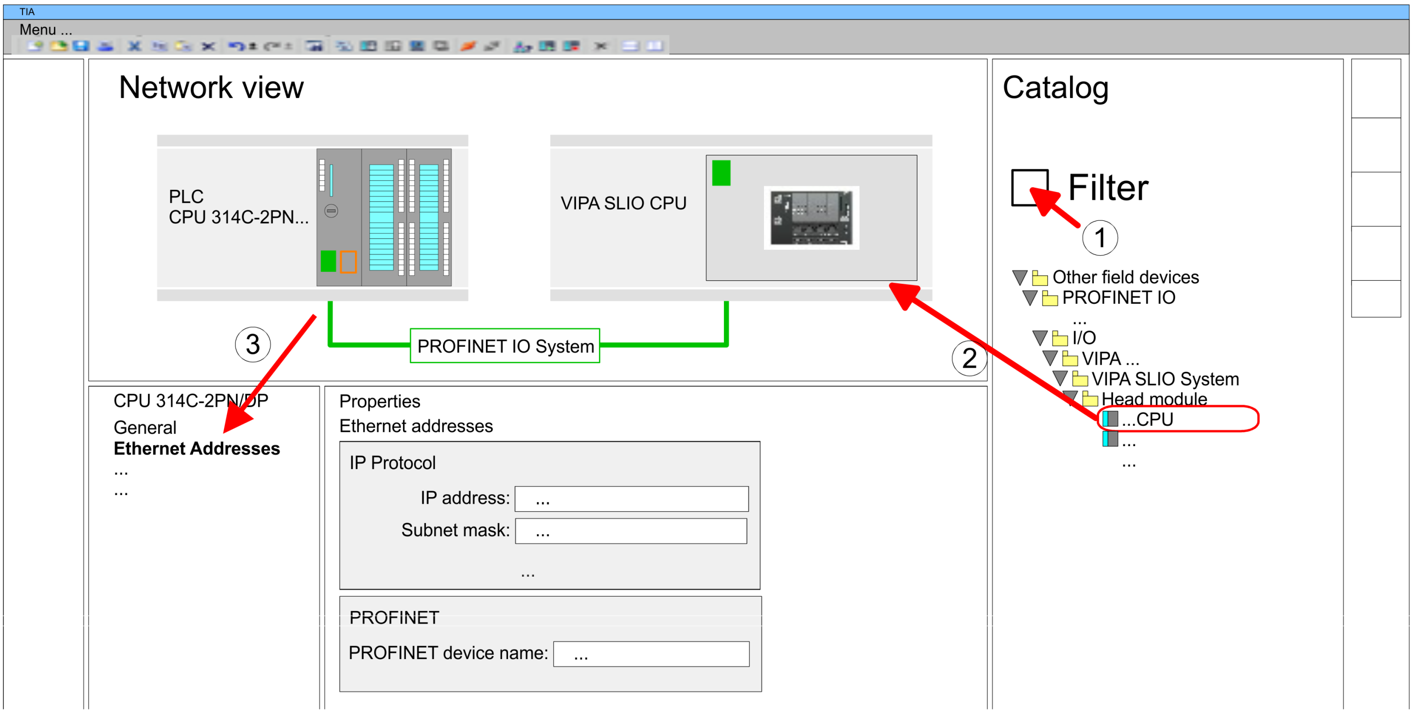


Device overview:

Module	...	Slot	...	Type	...
PLC...		2		CPU 314C-2PN/DP	
MPI interface...		2 X1		MPI/DP interface	
PROFINET interface...		2 X2		PROFINET interface	
DI24/DO16...		2 5		DI24/DO16	
AI5/AO2...		2 6		AI5/AO2	
Count...		2 7		Count	
...					

Connection CPU as PROFINET IO device

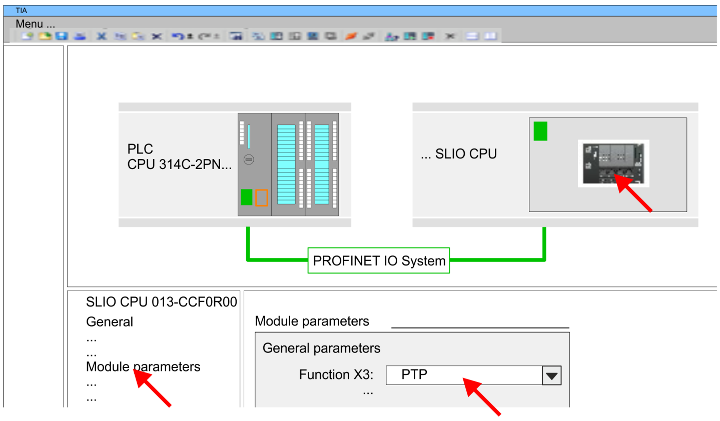
1. Switch in the *Project area* to *'Network view'*.
2. After installing the GSDML the IO device for the SLIO CPU may be found in the hardware catalog at *Other field devices > PROFINET > IO > VIPA ... > VIPA SLIO System*. Connect the slave system to the CPU by dragging&dropping it from the hardware 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 at 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.



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

⇨ In the *Device overview* of the PROFINET IO device *'VIPA SLIO CPU'* the CPU is already placed at slot 0.

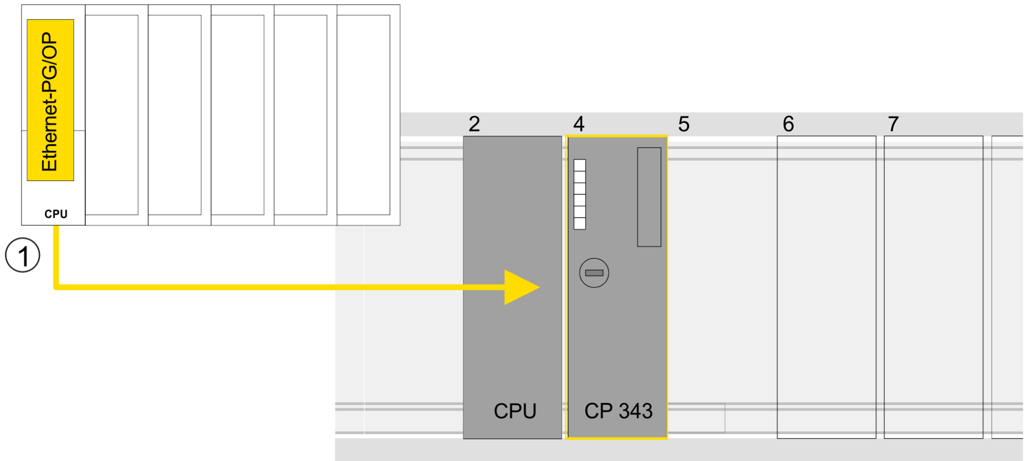
Enable PtP functionality



1. Open the properties dialog by a double-click at *'VIPA SLIO CPU'*.
2. Select at *'Function X3'* the value *'PTP'*.

Configuration of Ethernet PG/OP channel

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. You get valid IP address parameters from your system administrator.



1 Ethernet PG/OP channel

Device overview

Module	...	Slot	...	Type	...
PLC ...		2		CPU 315-2 PN/DP	
MPI/DP interface		2 X1		MPI/DP interface	
PROFINET inter-face		2 X2		PROFINET interface	
...		...		...	
CP 343-1		4		CP 343-1	
...		...		...	

7.6.2.3 Hardware configuration System SLIO CPU 014 ... 017

Add CPU in the project

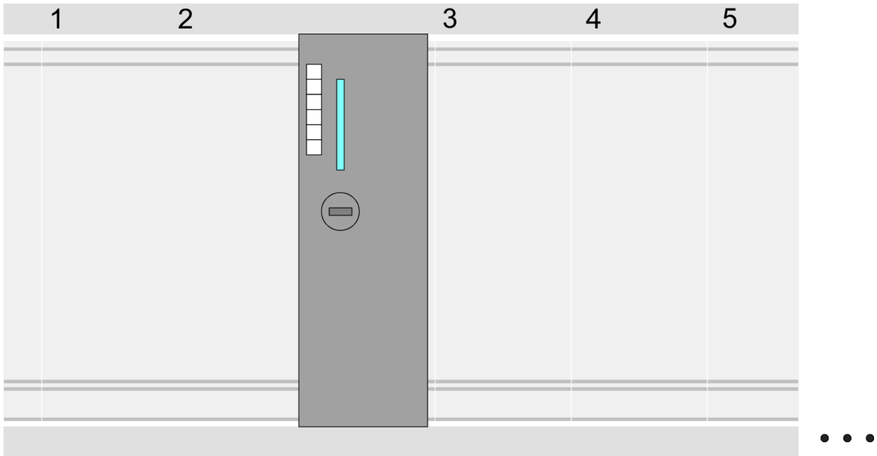
To be compatible with the Siemens SIMATIC TIA Portal the following steps should be executed:

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. Select the following CPU in the input dialog:

   SIMATIC S7-300 > CPU 315-2 PN/DP (315-2EH14-0AB0 V3.2)

   ⇨ The CPU is inserted with a profile rail.

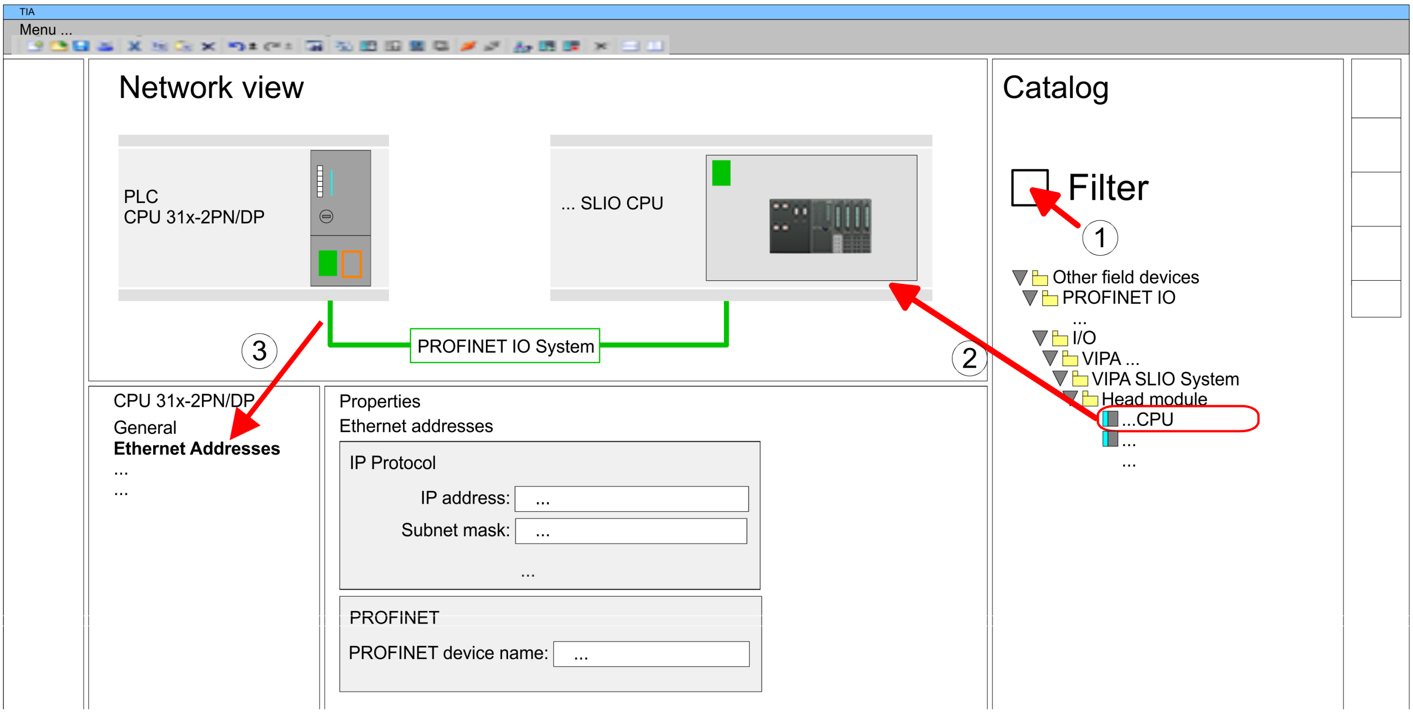


Device overview

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

Connection CPU as PROFINET IO device

1. Switch in the *Project area* to *'Network view'*.
2. After installing the GSDML the IO device for the SLIO CPU may be found in the hardware catalog at *Other field devices > PROFINET > IO > VIPA ... > VIPA SLIO System*. 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 at 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.



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

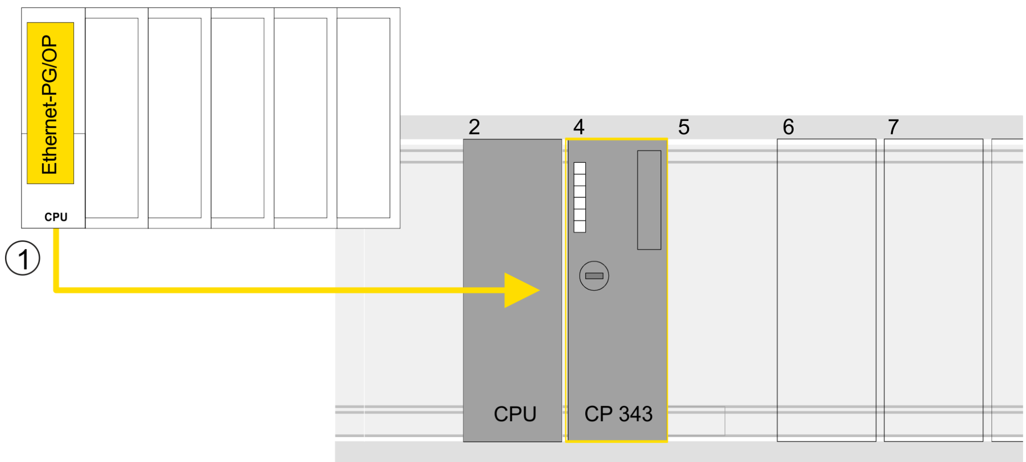
⇨ In the *Device overview* of the PROFINET IO device *'VIPA SLIO CPU'* the CPU is already placed at slot 0.

Enable PtP functionality

For the System SLIO CPUs 014 ... 017, the RS485 interface is set to PtP communication as standard. A hardware configuration to enable the PtP functionality is not necessary.

Configuration of Ethernet PG/OP channel

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. You get valid IP address parameters from your system administrator.



1 Ethernet PG/OP channel

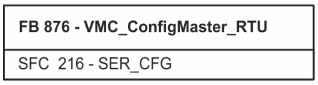
Device overview

Module	...	Slot	...	Type	...
PLC ...		2		CPU 315-2 PN/DP	
MPI/DP interface		2 X1		MPI/DP interface	
PROFINET inter-face		2 X2		PROFINET interface	
...		...		...	
CP 343-1		4		CP 343-1	
...		...		...	

7.6.3 User program

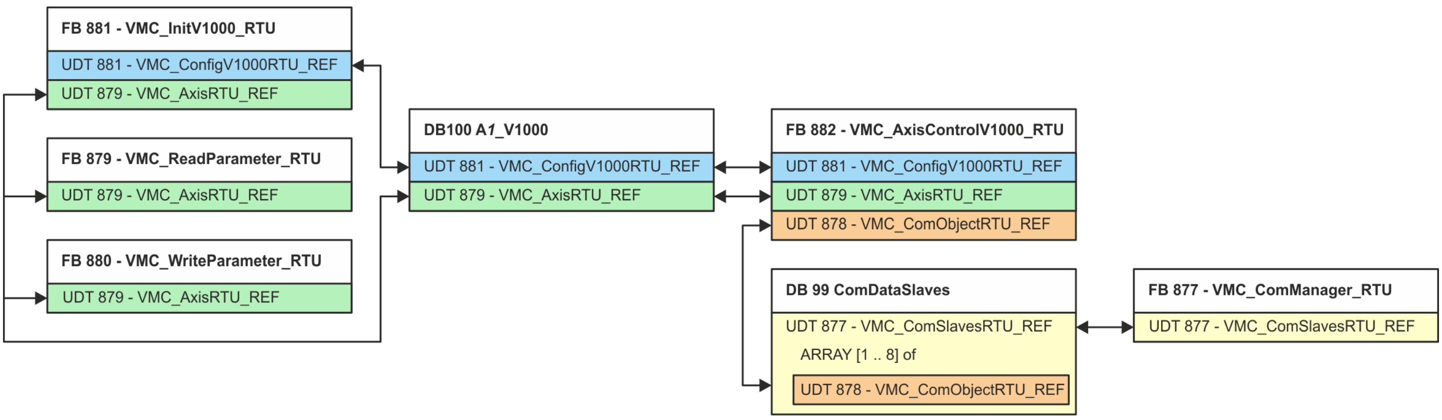
7.6.3.1 Program structure

OB 100



- FB 876 - VMC_ConfigMaster_RTU ⤷ *322*
  - This block is used to parametrize the serial interface of the CPU for Modbus RTU communication.
  - Internally block SFC 216 - SER_CFG is called.

OB 1



With the exception of blocks DB 99 and FB 877, you must create the blocks listed below for each connected inverter drive:

- FB 881 - VMC_InitV1000_RTU ⤷ *325*
  - The FB 881 - VMC_InitV1000_RTU initializes the corresponding inverter drive with the user data.
  - Before an inverter drive can be controlled, it must be initialized.
  - UDT 881 - VMC_ConfigV1000RTU_REF ⤷ *322*
  - UDT 879 - VMC_AxisRTU_REF ⤷ *322*
- FB 879 - VMC_ReadParameter_RTU ⤷ *324*
  - With this FB you have read access to the parameters of an inverter drive, which is connected serially via Modbus RTU.
  - The read data are recorded in a data block.
  - UDT 879 - VMC_AxisRTU_REF ⤷ *322*

- FB 880 - VMC_WriteParameter_RTU ↯ 325
  – With this FB you have read access to the parameters of an inverter drive, which is connected serially via Modbus RTU.
  – The data to be written must be stored in a data block.
  – UDT 879 - VMC_AxisRTU_REF ↯ 322
- DB 100 - A1_V1000
  – For each inverter drive, which is serially connected via Modbus RTU, a data block must be created.
  – UDT 879 - VMC_AxisRTU_REF ↯ 322
  – UDT 881 - VMC_ConfigV1000RTU_REF ↯ 322
- FB 882 - VMC_AxisControlV1000_RTU ↯ 327
  – With this block, you can control an inverter drive, which is serially connected via Modbus RTU and check its status.
  – UDT 881 - VMC_ConfigV1000RTU_REF ↯ 322
  – UDT 879 - VMC_AxisRTU_REF ↯ 322
  – UDT 878 - VMC_ComObjectRTU_REF ↯ 322
- DB 99 - ComDataSlaves
  – For the communication data of all the inverter drives (max. 8), which are serially connected via Modbus RTU, a common data block is to be created.
  – UDT 877 - VMC_ComSlavesRTU_REF ↯ 322
  – UDT 878 - VMC_ComObjectRTU_REF ↯ 322
- FB 877 - VMC_ComManager_RTU ↯ 323
  – The device ensures that only 1 inverter drive (Modbus slave) can use the serial interface. If several inverter drives are used, this block, as communication manager, sends the jobs to the respective Modbus slaves and evaluates their responses.
  – UDT 877 - VMC_ComSlavesRTU_REF ↯ 322

7.6.3.2 Copy blocks into project

Include library

1. Go to the service area of www.vipa.com.
2. Download the *Simple Motion Control* library from the download area at *'VIPA Lib'*.

   The library is available as packed zip file for the corresponding TIA Portal version.
3. Start your un-zip application with a double click on the file ...TIA_V*xx*.zip and copy all the files and folders in a work directory for the Siemens TIA Portal.
4. Switch to the *Project view* of the Siemens TIA Portal.
5. Choose "Libraries" from the task cards on the right side.
6. Click at "Global library".
7. Click on the free area inside the *'Global Library'* and select *'Context menu → Retrieve library'*.
8. Navigate to your work directory and load the file ...Simple Motion.zal*xx*.

Copy blocks into project

Copy all blocks from the library into the *'Program blocks'* of the *Project tree* of your project.

- FB 876 - VMC_ConfigMaster_RTU
- FB 877 - VMC_ComManager_RTU
- FB 878 - VMC_RWParameterSys_RTU
- FB 879 - VMC_ReadParameter_RTU
- FB 880 - VMC_WriteParameter_RTU
- FB 881 - VMC_InitV1000_RTU
- FB 882 - VMC_AxisControlV1000_RTU
- FB 60 - SEND
- FB 61 - RECEIVE
- FB 72 - RTU MB_MASTER
- FC 216 - SER_CFG
- FC 217 - SER_SND
- FC 218 - SER_RCV
- UDT 877 - VMC_ComSlavesRTU_REF
- UDT 878 - VMC_ComObjectRTU_REF
- UDT 879 - VMC_AxisRTU_REF
- UDT 881 - VMC_ConfigV1000RTU_REF
- SFB 4 - TON

7.6.3.3 Create OB 100 for serial communication

1. Click at *'Project tree ➔ ...CPU...PLC program ➔ Program blocks ➔ Add new block'*.
   ⇨ The dialog *'Add block'* is opened.
2. Enter OB 100 and confirm with [OK].
   ⇨ OB 100 is created and opened.
3. Add a `Call FB876, DB876` to the OB 100.
   ⇨ The block call is created and a dialog opens to specify the instance data block *'VMC_ConfigMaster_RTU_876'*.
4. Confirm the query of the instance data block with [OK].
5. Specify the following parameters:

`Call FB876, DB876` *Chap. 7.7.5 'FB 876 - VMC_ConfigMaster_RTU - Modbus RTU CPU interface' page 322*

Baudrate	:= `B#16#09`	// Baud rate: 09h (9600bit/s)	IN: BYTE
CharLen	:= `B#16#03`	// Number data bits: 03h (8bit)	IN: BYTE
Parity	:= `B#16#00`	// Parity: 0 (none)	IN: BYTE
StopBits	:= `B#16#01`	// Stop bits: 1 (1bit)	IN: BYTE
TimeOut	:= `W#16#1FFF`	// Error wait time: 1FFFh (high selected)	IN: WORD
Valid	:= `"ModbusConfigValid"`	// Configuration	OUT BOOL
Error	:= `"ModbusConfigError"`	// Error feedback	OUT BOOL
ErrorID	:= `"ModbusConfigErrorID"`	// Additional error information	OUT: WORD

7.6.3.4 Create data block for Modbus slave

For each inverter drive, which is serially connected via Modbus RTU, a data block must be created.

1. Click at *'Project tree ➔ ...CPU...PLC program ➔ Program blocks ➔ Add new block'*.
   ⇨ The dialog *'Add block'* is opened.
2. Select the block type *'DB block'* and assign it the name "A1_V1000". The DB number can freely be selected such as DB100. Specify DB 100 and create this as a global DB with [OK].
   ⇨ The block is created and opened.
3. In "A1_V1000" create the following variables:
   - *'AxisData'* of type UDT 879 - VMC_AxisRTU_REF
   - *'V1000Data'* of type UDT 881 - VMC_ConfigV1000RTU_REF

7.6.3.5 Create data block for all Modbus slaves

For the communication data of the inverter drives, which are serially connected via Modbus RTU, a common data block is to be created.

1. Click at *'Project tree ➔ ...CPU...PLC program ➔ Program blocks ➔ Add new block'*.
   ⇨ The dialog *'Add block'* is opened.
2. Select the block type *'DB block'* and assign it the name "ComDataSlaves". The DB number can freely be selected such as DB99. Specify DB 99 and create this as a global DB with [OK].
   ⇨ The block is created and opened.
3. In "ComDataSlaves" create the following variable:
   - *'Slaves'* of Type UDT 877 - VMC_ComSlavesRTU_REF

7.6.3.6 OB 1 - Create instance of communication manager

The FB 877 - VMC_ComManager_RTU ensures that only 1 inverter drive (Modbus slave) can use the serial interface. As a communication manager, the block sends the jobs to the respective Modbus slaves and evaluates their responses.

1. Open the OB 1.
2. Add a `Call FB877, DB877` to OB 1.
   ⇨ The block call is created and a dialog opens to specify the instance data block *'VMC_ComManager_RTU_877'*.
3. Confirm the query of the instance data block with [OK].
4. Specify the following parameters:

`Call FB877, DB877` *Chap. 7.7.6 'FB 877 - VMC_ComManager_RTU - Modbus RTU communication manager' page 323*

NumberOfSlaves	:= `1`	// Number of connected inverter drives: 1	IN: INT
WaitCycles	:= `"ComWaitCycles"`	// Minimum number of waiting cycles	IN: DINT
SlavesComData	:= `"ComDataSlaves.Slave"`	// Reference to all communication objects	IN-OUT: UDT 877

7.6.3.7 OB 1 - Create instance of the V1000 initialization

The FB 881 - VMC_InitV1000_RTU initializes the corresponding inverter drive with the user data. Before an inverter drive can be controlled, it must be initialized.

1. Add a `Call FB881, DB881` to OB 1.

   ⇨ The block call is created and a dialog opens to specify the instance data block *'VMC_InitV1000_RTU _881'*.

2. Confirm the query of the instance data block with [OK].
3. Specify the following parameters:

`Call FB881, DB881` *Chap. 7.7.10 'FB 881 - VMC_InitV1000_RTU - Modbus RTU initialization' page 325*

Execute	:= `"A1_InitExecute"`	// The job is started with edge 0-1.	IN: BOOL
Hardware	:= `"A1_InitHardware"`	// Specification of the hardware, used // 1: System SLIO CP040, 2: SPEED7 CPU	IN: BYTE
Laddr	:= `"A1_InitLaddr"`	// Logical address when using CP040	IN: INT
UnitId	:= `"A1_InitUnitId"`	// Modbus address of the *V1000*	IN: BYTE
UserUnitsVelocity	:= `"A1_InitUserUnitsVel"`	// User unit for velocities: // 0: Hz, 1: %, 2: RPM	IN: INT
UserUnitsAcceleration	:= `"A1_InitUserUnitsAcc"`	// User units acceleration/deceleration // 0: 0.01s, 1: 0.1s	IN: INT
MaxVelocityApp	:= `"A1_InitMaxVelocityApp"`	// Max. velocity in user units	IN: REAL
Done	:= `"A1_InitDone"`	// Status job finished	OUT: BOOL
Busy	:= `"A1_InitBusy"`	// Status job in progress	OUT: BOOL
Error	:= `"A1_InitError"`	// Error feedback	OUT: BOOL
ErrorID	:= `"A1_InitErrorID"`	// Additional error information	OUT: WORD
Axis	:= `"A1_V1000".AxisData`	// Reference to the general axis data	IN-OUT: UDT 879
V1000	:= `"A1_V1000".V1000Data`	// Reference to the drive-specific data	IN-OUT: UDT 881

Input values

All parameters must be interconnected with the corresponding variables or operands. The following input parameters must be pre-assigned:

- Hardware
  Here specify the hardware you use to control your inverter drives:
  – 1: System SLIO CP040 whose logical address is to be specified via *Laddr*.
  – 2: SPEED7 CPU
- Laddr
  – Logical address for the System SLIO CP040 (*Hardware* = 1). Otherwise, this parameter is ignored.
- UnitId
  – Modbus address of the *V1000*.

- UserUnitsVelocity
  User unit for speeds:
  – 0: Hz
    Specified in hertz
  – 1: %
    Specified as a percentage of the maximum speed
    = 2*f_{max}/P
    with f_{max}: max. output frequency (parameter E1-04)
    p: Number of motor poles (motor-dependent parameter E2-04, E4-04 or E5-04)
  – 2: RPM
    Data in revolutions per minute
- UserUnitsAcceleration
  User units for acceleration and deceleration
  – 0: 0.01s (range of values: 0.00s - 600.00s)
  – 1: 0.1s (range of values: 0.0 - 6000.0s)
- MaxVelocityApp
  Max. speed for the application. The specification must be made in user units and is used for synchronization in movement commands.

7.6.3.8 OB 1 - Create instance axis control V1000

With the FB 882 - VMC_AxisControlV1000_RTU you can control an inverter drive, which is serially connected via Modbus RTU and check its status.

1. Add a `Call FB882, DB882` to OB 1.
   ⇨ The block call is created and a dialog opens to specify the instance data block *'VMC_AxisControlV1000_RTU _882'*.
2. Confirm the query of the instance data block with [OK].
3. Specify the following parameters:

`Call FB882, DB882` *Chap. 7.7.11 'FB 882 - VMC_AxisControlV1000_RTU - Modbus RTU Axis control' page 327*

AxisEnable	:= `"A1_AxisEnable"`	// Activation of the axis	IN: BOOL
AxisReset	:= `"A1_AxisReset"`	// Command: Reset error of the *V1000*.	IN: BOOL
StopExecute	:= `"A1_StopExecute"`	// Command: *Stop* - Stop axis	IN: BOOL
MvVelocityExecute	:= `"A1_MvVelocityExecute"`	// Command: *MoveVelocity* (velocity control)	IN: BOOL
Velocity	:= `"A1_Velocity"`	// Parameter: Velocity setting for *MoveVelocity*	IN: REAL
AccelerationTime	:= `"A1_AccelerationTime"`	// Parameter: Acceleration time	IN: REAL
DecelerationTime	:= `"A1_DecelerationTime"`	// Parameter: Deceleration time	IN: REAL
JogPositive	:= `"A1_JogPositive"`	// Command: *JogPos*	IN: BOOL
JogNegative	:= `"A1_JogNegative"`	// Command: *JogNeg*	IN: BOOL
JogVelocity	:= `"A1_JogVelocity"`	// Parameter: Velocity setting for jogging	IN: REAL
JogAccelerationTime	:= `"A1_JogAccelerationTime"`	// Parameter: Acceleration time for jogging	IN: REAL
JogDecelerationTime	:= `"A1_JogDecelerationTime"`	// Parameter: Deceleration time for jogging	IN: REAL
AxisReady	:= `"A1_AxisReady"`	// Status: Axis ready	OUT: BOOL
AxisEnabled	:= `"A1_AxisEnabled"`	// Status: Activation of the axis	OUT: BOOL
AxisError	:= `"A1_AxisError"`	// Status: Axis error	OUT: BOOL

AxisErrorID	:= `"A1_AxisErrorID"`	// Status: Additional error information for *AxisError*	OUT: WORD
DriveError	:= `"A1_DriveError"`	// Status: Error on the inverter drive	OUT: BOOL
ActualVelocity	:= `"A1_ActualVelocity"`	// Status: Current velocity	OUT: REAL
InVelocity	:= `"A1_InVelocity"`	// Status target velocity	OUT: BOOL
CmdDone	:= `"A1_CmdDone"`	// Status: Command finished	OUT: BOOL
CmdBusy	:= `"A1_CmdBusy"`	// Status: Command in progress	OUT: BOOL
CmdAborted	:= `"A1_CmdAborted"`	// Status: Command aborted	OUT: BOOL
CmdError	:= `"A1_CmdError"`	// Status: Command error	OUT: BOOL
CmdErrorID	:= `"A1_CmdErrorID"`	// Status: Additional error information for *CmdError*	OUT: WORD
CmdActive	:= `"A1_CmdActive"`	// Status: Active command	OUT: INT
DirectionPositive	:= `"A1_DirectionPositive"`	// Status: Direction of rotation positive	OUT: BOOL
DirectionNegative	:= `"A1_DirectionNegative"`	// Status: Direction of rotation negative	OUT: BOOL
Axis	:= `"A1_V1000".AxisData`	// Reference to the general axis data	IN-OUT: UDT 879
V1000	:= `"A1_V1000".V1000Data`	// Reference to the general axis data // of the inverter drive	IN-OUT: UDT 881
AxisComData	:= `"ComDataSlaves".Slaves.Slave(1)`	// Reference to the communication data	IN-OUT: UDT 878

7.6.3.9 OB 1 - Create instance read parameter

With the FB 879 - VMC_ReadParameter_RTU you have read access to the parameters of an inverter drive, which is serially connected via Modbus RTU. For the parameter data a DB is to be created.

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

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

2. Select the block type *'DB block'* and assign it the name "A1_TransferData". The DB number can freely be selected. Specify DB 98 and create this as a global DB with [OK].

   ⇨ The block is created and opened.

3. In "A1_TransferData" create the following variables:
   - *'Data_0'* of type WORD
   - *'Data_1'* of type WORD
   - *'Data_2'* of type WORD
   - *'Data_3'* of type WORD

4. Add a `Call FB879, DB879` to OB 1.

   ⇨ The block call is created and a dialog opens to specify the instance data block *'VMC_ReadParameter_RTU'*.

5. Confirm the query of the instance data block with [OK].

6. Specify the following parameters:

`Call FB879, DB879` *Chap. 7.7.8 'FB 879 - VMC_ReadParameter_RTU - Modbus RTU read parameters' page 324*

Execute	:= `"A1_RdParExecute"`	// The job is started with edge 0-1.	IN: BOOL
StartAddress	:= `"A1_RdParStartAddress"`	// Start address of the 1. register	IN: INT

Quantity	:= "A1_RdParQuantity"	// Number of registers to read	IN: INT
Done	:= "A1_RdParDone"	// Status job finished	IN: REAL
Busy	:= "A1_RdParBusy"	// Status job in progress	OUT: BOOL
Error	:= "A1_RdParError"	// Error feedback	OUT: BOOL
ErrorID	:= "A1_RdParErrorID"	// Additional error information	OUT: BOOL
Data	:= P#DB98.DBX0.0 BYTES 8	// Location of the parameter data	OUT: WORD
Axis	:= "A1_V1000".AxisData	// Reference to the general axis data	IN-OUT: UDT 879

Please note that only whole registers can be read as WORD. To evaluate individual bits, you must swap high and low byte!

7.6.3.10 OB 1 - Create instance write parameter

With the FB 880 - VMC_WriteParameter_RTU you have write access to the parameters of an inverter drive, which is serially connected via Modbus RTU. For the data you can use the DB created for read access - here DB 98.

1. Add a `Call FB880, DB880` to OB 1.
   ⇨ The block call is created and a dialog opens to specify the instance data block *'VMC_WriteParameter_RTU'*.
2. Confirm the query of the instance data block with [OK].
3. Specify the following parameters:

`Call FB880, DB880` *Chap. 7.7.9 'FB 880 - VMC_WriteParameter_RTU - Modbus RTU write parameters' page 325*

Execute	:= "A1_WrParExecute"	// The job is started with edge 0-1.	IN: BOOL
StartAddress	:= "A1_WrParStartAddress"	// Start address of the 1. register	IN: INT
Quantity	:= "A1_WrParQuantity"	// Number of registers to write	IN: INT
Done	:= "A1_WrParDone"	// Status job finished	IN: REAL
Busy	:= "A1_WrParBusy"	// Status job in progress	OUT: BOOL
Error	:= "A1_WrParError"	// Error feedback	OUT: BOOL
ErrorID	:= "A1_WrParErrorID"	// Additional error information	OUT: BOOL
Data	:= P#DB98.DBX0.0 BYTES 8	// Location of the parameter data	OUT: WORD
Axis	:= "A1_V1000".AxisData	// Reference to the general axis data	IN-OUT: UDT 879

7.6.3.11 Sequence of operations

1. Safe and translate your project.
2. Transfer your project to your CPU.
   ⇨ You can take your application into operation now.

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

3. A watch table allows you to manually control the inverter drive. To create a watch table, double-click *'Project tree ➔ ...CPU... ➔ Watch and force tables ➔ Add new watch table'*.
   ⇨ The watch table is created and opened for editing.
4. First adjust the waiting time between 2 jobs. This is at least 200ms for a V1000 inverter drive. For this enter in the watch table at *'Name'* the designation *'ComWaitCycles'* as *'DEC'* and enter at *'Modify value'* a value between 200 and 400.

   > *To increase performance, you can later correct this to a smaller value as long as you do not receive a timeout error (80C8h). Please note that some commands, such as MoveVelocity, can consist of several jobs.*

5. Before you can control an inverter drive, it must be initialized with FB 881 - VMC_InitV1000_RTU. *Chap. 7.7.10 'FB 881 - VMC_InitV1000_RTU - Modbus RTU initialization' page 325*

   For this enter in the watch table at *'Name'* the designation *'A1_InitExecute'* as *'Boolean'* and enter at *'Modify value'* the value *'True'*. Activate the modification of the variables and start the transmission of the modified values.

   ⇨ The inverter drive is initialized. After execution, the output *Done* returns TRUE. In the event of a fault, you can determine the error by evaluating the *ErrorID*.

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

6. After successful initialization, the registers of the connected inverter drives are cyclically processed, i.e. they receive cyclical jobs. For manual control, you can use the FB 882 - VMC_AxisControlV1000_RTU to send control commands to the appropriate inverter drive. *Chap. 7.7.11 'FB 882 - VMC_AxisControlV1000_RTU - Modbus RTU Axis control' page 327*
7. Create the parameters of the FB 882 - VMC_AxisControlV1000_RTU for control and query in the watch table.
8. Save the watch table under a name such as *'V1000'*.
9. Activate the corresponding axis by setting *AxisEnable*. As soon as this reports *AxisReady* = TRUE, you can control it with the corresponding drive commands.

7.7 Drive specific blocks

7.7.1 UDT 877 - VMC_ComSlavesRTU_REF - Modbus RTU data structure communication data all slaves

This is a user-defined data structure for the communication data of the connected Modbus RTU slaves. The UDT is specially adapted to the use of inverter drives, which are connected via Modbus RTU.

7.7.2 UDT 878 - VMC_ComObjectRTU_REF - Modbus RTU data structure communication data slave

This is a user-defined data structure for the communication data of a connected Modbus RTU slave. The UDT is specially adapted to the use of inverter drives, which are connected via Modbus RTU.

7.7.3 UDT 879 - VMC_AxisRTU_REF - Modbus RTU data structure axis data

This is a user-defined data structure that contains status information about the inverter drive. This structure serves as a reference to the general axis data of the inverter drive.

7.7.4 UDT 881 - VMC_ConfigV1000RTU_REF - Modbus RTU data structure configuration

This is a user-defined data structure containing information about the configuration data of an inverter drive, which is connected via Modbus RTU.

7.7.5 FB 876 - VMC_ConfigMaster_RTU - Modbus RTU CPU interface

Description

This block is used to parametrize the serial interface of the CPU for Modbus RTU communication.

Please note that this block internally calls the SFC 216.

In the SPEED7 Studio, this module is automatically inserted into your project.

In Siemens SIMATIC Manager, you have to copy the SFC 216 from the Motion Control Library into your project.

Parameter

Parameter	Declaration	Data type	Description
Baudrate	INPUT	BYTE	Speed of data transmission in bit/s (baud). ■ 04h: 1200baud ■ 05h: 1800baud ■ 06h: 2400baud ■ 07h: 4800baud ■ 08h: 7200baud ■ 09h: 9600baud ■ 0Ah: 14400baud ■ 0Bh: 19200baud ■ 0Ch: 38400baud ■ 0Dh: 57600baud ■ 0Eh: 115200baud

Parameter	Declaration	Data type	Description
CharLen	INPUT	BYTE	Number of data bits to which a character is mapped ■ 0: 5bit ■ 1: 6bit ■ 2: 7bit ■ 3: 8bit
Parity	INPUT	BYTE	The parity is even or odd depending on the value. For parity control, the information bits are extended by the parity bit, which by its value ("0" or "1") adds the value of all bits to an agreed state. If no parity is specified, the parity bit is set to "1" but not evaluated. ■ 0: None ■ 1: Odd ■ 2: Even
StopBits	INPUT	BYTE	The stop bits are added to each character to be transmitted and signalize the end of a character ■ 1: 1bit ■ 2: 1.5bit ■ 3: 2bit
TimeOut	INPUT	WORD	Waiting time until an error is generated if a slave does not respond. The time for *TimeOut* must be specified as a hexadecimal value. The hexadecimal value is obtained by multiplying the desired time in seconds by the baud rate. Example: Desired time 8ms at a baud rate of 19200bit/s Calculation: 19200bit/s x 0.008s ≈ 154bit >>>> (9Ah) The hex value should be 9Ah.
Valid	OUTPUT	BOOL	Configuration ■ TRUE: The configuration is valid. ■ FALSE: The configuration is not valid.
Error	OUTPUT	BOOL	Error feedback ■ TRUE: An error has occurred - see *ErrorID*. ■ FALSE: There is no error.
ErrorID	OUTPUT	WORD	Additional error information *Chap. 12 'ErrorID - Additional error information' page 457*

7.7.6 FB 877 - VMC_ComManager_RTU - Modbus RTU communication manager

Description

This block regulates that only one slave can communicate in succession via the serial interface. Via the UDT 877 this block has access to the communication data of all slaves.

You can only use one FB 877 in your project per serial interface!

Parameter

Parameter	Declaration	Data type	Description
NumberOfSlaves	IN	INT	Number of currently used Modbus slaves
WaitCycles	IN	DINT	Minimum number of cycles to wait between two requests from a slave. This prevents overflows on the slave and resulting timeouts.
SlavesComData	IN_OUT	UDT 877	Reference to the data block with all communication objects

7.7.7 FB 878 - VMC_RWParameterSys_RTU - Modbus RTU read/write parameters system

Description

This block is used internally by the system for parameter transfer.

You must not call this module, as this can lead to a malfunction of your system!

7.7.8 FB 879 - VMC_ReadParameter_RTU - Modbus RTU read parameters

Description

With this block you can read parameters from the corresponding slave.

Please note that only whole registers can be read as WORD. To evaluate individual bits, you must swap high and low byte!

Parameter

Parameter	Declaration	Data type	Description
Execute	IN	BOOL	The job is started with edge 0-1.
StartAddress	IN	WORD	Start address of the register from which to read.
Quantity	IN	BYTE	Number of registers to read.
Done	OUT	BOOL	Status ■ TRUE: Job successfully done
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. 12 'ErrorID - Additional error information' page 457*
Data	IN-OUT	ANY	Reference where to store the read data
Axis	IN-OUT	UDT 879	Reference to the general axis data of the inverter drive

7.7.9 FB 880 - VMC_WriteParameter_RTU - Modbus RTU write parameters

Description

With this block you can write parameters in the registers of the corresponding slave.

Please note that only whole registers can be written as WORD. To set or reset individual bits, you must swap high and low byte!

Parameter

Parameter	Declaration	Data type	Description
Execute	INPUT	BOOL	The job is started with edge 0-1.
StartAddress	INPUT	WORD	Start address of the register from which to write.
Quantity	INPUT	BYTE	Number of registers to write.
Done	OUTPUT	BOOL	Status ■ TRUE: Job successfully done
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. 12 'ErrorID - Additional error information' page 457*
Data	IN_OUT	ANY	Reference to the data to be written.
Axis	IN_OUT	UDT 879	Reference to the general axis data of the inverter drive

7.7.10 FB 881 - VMC_InitV1000_RTU - Modbus RTU initialization

Description

This block is used to initialize the corresponding inverter drive with the user data and must be processed, before commands can be transferred. The block is specially adapted to the use of a inverter drive, which is connected via Modbus RTU.

Parameter

Parameter	Declaration	Data type	Description
Execute	INPUT	BOOL	The job is started with edge 0-1.
Hardware	INPUT	BYTE	Specification of the hardware, which is used ■ 1: System SLIO CP040 whose logical address is to be specified via *Laddr*. ■ 2: SPEED7 CPU
Laddr	INPUT	INT	Logical address for the System SLIO CP040 (*Hardware* = 1). Otherwise, this parameter is ignored.
UnitId	INPUT	BYTE	Modbus address of the *V1000*.

Parameter	Declaration	Data type	Description
UserUnitsVelocity	INPUT	INT	User unit for speeds ■ 0: Hz – Specified in hertz ■ 1: % – Specified as a percentage of the maximum speed – = $2*f_{max}/p$ with f_{max}: max. output frequency (parameter E1-04) p: Number of motor poles (motor-dependent parameter E2-04, E4-04 or E5-04) ■ 2: RPM – Data in revolutions per minute
UserUnitsAcceleration	INPUT	INT	User units for acceleration and deceleration ■ 0: 0.01s (range of values: 0.00s - 600.00s) ■ 1: 0.1s (range of values: 0.0 - 6000.0s)
MaxVelocityApp	INPUT	REAL	Max. speed for the application. The specification must be made in user units and is used for synchronization in movement commands.
Done	OUTPUT	BOOL	Status ■ TRUE: Job successfully done
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. 12 'ErrorID - Additional error information' page 457*
Axis	IN_OUT	UDT 879	Reference to the general axis data of the inverter drive
V1000	IN_OUT	UDT 881	Reference to the user data of the inverter drive

7.7.11 FB 882 - VMC_AxisControlV1000_RTU - Modbus RTU Axis control

Description

With the FB 882 *VMC_AxisControlV1000_RTU* you can control an inverter drive, which is serially connected via Modbus RTU and check its status.

The control of a V1000 inverter drive, which is connected via Modbus RTU, takes place exclusively with FB 882 VMC_AxisControlV1000_RTU. PLCopen blocks are not supported!

Parameter

Parameter	Declaration	Data type	Description
AxisEnable	INPUT	BOOL	Activation of the axis ■ TRUE: Switch on axis → *AxisEnabled* = 1, commands can be executed. ■ FALSE: Switch off the axis → *AxisEnabled* = 0, no commands can be executed.
AxisReset	INPUT	BOOL	Command: Reset inverter drive faults. → *CmdActive* = 1
StopExecute	INPUT	BOOL	Command: *Stop* - Stop axis → *CmdActive* = 1
MvVelocityExecute	INPUT	BOOL	Command: *MoveVelocity* (velocity control) → *CmdActive* = 2
Velocity	INPUT	REAL	Parameter: Velocity setting for MoveVelocity in user units. See example below the table
AccelerationTime	INPUT	REAL	Parameter: Acceleration time in seconds (accuracy depending on *UserUnitsAcceleration* at Init block). Always related to time, from standstill to the maximum set velocity. See example below the table This parameter is used for the command MoveVelocity (*MvVelocityExecute*).
DecelerationTime	INPUT	REAL	Parameter: Deceleration time in seconds (accuracy depending on *UserUnitsAcceleration* at Init block). Always related to time, from standstill to the maximum set velocity. See example below This parameter is used for the commands Stop (*StopExecute*) MoveVelocity (*MvVelocityExecute*).
JogPositive	INPUT	BOOL	Command: *JogPos* ■ Edge 0-1: Start axis in positive direction (jogging positive) ■ Edge 1-0: Stop axis
JogNegative	INPUT	BOOL	Command: *JogNeg* ■ Edge 0-1: Start axis in negative direction (jogging negative) ■ Edge 1-0: Stop axis
JogVelocity	INPUT	REAL	Parameter: Velocity setting for jogging in user units. Note: *JogPositive* and *JogNegative* use the absolute value of the velocity.

Parameter	Declaration	Data type	Description
JogAcceleration-Time	INPUT	REAL	Parameter: Acceleration time for jogging in seconds (accuracy depending on *UserUnitsAcceleration* at Init block). Is always based on the time, from standstill to the maximum set speed. See example below the table
JogDeceleration-Time	INPUT	REAL	Parameter: Deceleration time for jogging in seconds (accuracy depending on *UserUnitsAcceleration* of FB 881). Parameter always refers to the time from standstill to the maximum set velocity. See example below the table
AxisReady	OUTPUT	BOOL	Status: Axis ready ■ TRUE: The axis is ready to switch on. ■ FALSE: The axis is not ready to switch on.
AxisEnabled	OUTPUT	BOOL	Status: Activation of the axis ■ TRUE: The axis is switched on ■ FALSE: The axis is switched off
AxisError	OUTPUT	BOOL	Status: Axis error ■ TRUE: Axis reports an error and is locked. Further error information can be found in *AxisErrorID*. ■ FALSE: Axis does not report any errors.
AxisErrorID	OUTPUT	WORD	Status: Additional error information for *AxisError* *Chap. 12 'ErrorID - Additional error information' page 457*
DriveError	OUTPUT	BOOL	Status: Error on the inverter drive ■ TRUE: Inverter drive reports an error and is locked. ■ FALSE: Inverter drive does not report any errors.
ActualVelocity	OUTPUT	REAL	Status: Current velocity in user units
InVelocity	OUTPUT	BOOL	Status target velocity ■ TRUE: The target velocity *Velocity* has been reached. ■ FALSE: The target velocity *Velocity* has not yet been reached.
CmdDone	OUTPUT	BOOL	Status: Command finished ■ TRUE: Command was executed successfully. ■ FALSE: Command has not yet been executed or is still in progress.
CmdBusy	OUTPUT	BOOL	Status: Command in progress ■ TRUE: Command is in progress ■ FALSE: Currently no command is executed.
CmdAborted	OUTPUT	BOOL	Status: Command aborted ■ TRUE: Command was aborted ■ FALSE: Command was not aborted
CmdError	OUTPUT	BOOL	Status: Command error ■ TRUE: An error occurred while executing a command ■ FALSE: The execution of a command proceeded correctly.
CmdErrorID	OUTPUT	WORD	Status: Additional error information for *CmdError* *Chap. 12 'ErrorID - Additional error information' page 457*

Parameter	Declaration	Data type	Description
CmdActive	OUTPUT	INT	Status: Active command ■ 0: NoCmd - no command active ■ 1: Stop ■ 2: MvVelocity ■ 3: MvRelative ■ 4: JogPos ■ 5: JogNeg
DirectionPositive	OUTPUT	BOOL	Status: Direction of rotation positive ■ TRUE: Current direction of rotation is positive ■ FALSE: Current direction of rotation is not positive
DirectionNegative	OUTPUT	BOOL	Status: Direction of rotation negative ■ TRUE: Current direction of rotation is negative ■ FALSE: Current direction of rotation is not negative
Axis	IN_OUT	UDT 879	Reference to the general axis data of the inverter drive
V1000	IN_OUT	UDT 881	Reference to the user data of the inverter drive
AxisComData	IN_OUT	UDT 878	Reference to the communication data of the current slave

Example *AccelerationTime*

The values for *Velocity*, *AccelerationTime* and *DecelerationTime* must be specified in the user units of the FB 881 - VMC_InitV1000_RTU. *AccelerationTime* or *DecelerationTime* always refer to the time from standstill to the maximum set velocity or from the maximum velocity to standstill.

The maximum velocity results from the formula

$$v_{max} = \frac{2 \cdot f}{p}$$

v_{max} max. velocity in 1/s
f max. Output frequency (parameter E1-04)
p Number of motor poles (motor-dependent parameter E2-04, E4-04 or E5-04)

Sequence of operations

1. Select *'Project ➔ Compile all'* and transfer the project into your CPU.

   You can find more information on the transfer of your project in the online help of the *SPEED7 Studio*.

   ⇨ You can take your application into operation now.

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

2. Bring your CPU into RUN and turn on your inverter drive.

   ⇨ The FB 882 - VMC_AxisControlV1000_RTU is executed cyclically.

3. As soon as *AxisReady* = TRUE, you can use *AxisEnable* to enable the axis.
4. You now have the possibility to control your drive via its parameters and to check its status.

8 Usage inverter drive via EtherCAT

8.1 Overview

Precondition

- SPEED7 Studio from V1.8
  or
- Siemens SIMATIC Manager from V 5.5, SP2 & *SPEED7 EtherCAT Manager* & *Simple Motion Control Library*
- CPU with EtherCAT master, such as CPU 015-CEFNR00
- Inverter drive with EtherCAT option card

Steps of configuration

1. Set the parameters on the inverter drive.
   - The setting of the parameters happens by means of the software tool *Drive Wizard+*.
2. Hardware configuration in the VIPA *SPEED7 Studio* or Siemens SIMATIC Manager.
   - Configuring the CPU.
3. Programming in the VIPA *SPEED7 Studio* or Siemens SIMATIC Manager.
   - *Init* block for the configuration of the axis.
   - *Kernel* block for communication with the axis.
   - Connecting the blocks for motion sequences.

8.2 Set the parameters on the inverter drive

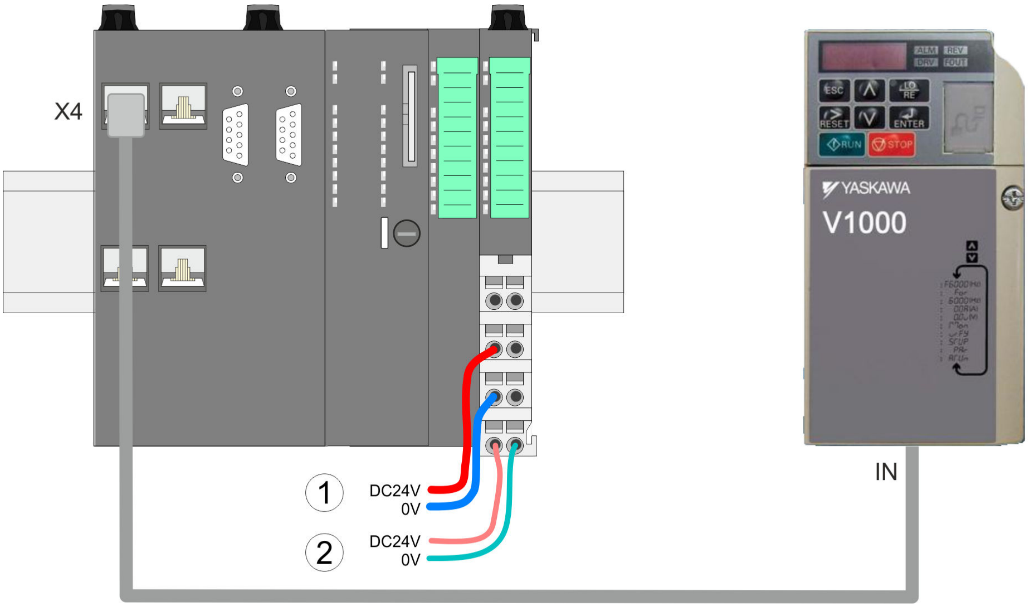
CAUTION!
Before the commissioning, you have to adapt your inverter drive to your application with the *Drive Wizard+* software tool! More may be found in the manual of your inverter drive.

The following table shows all parameters which do not correspond to the default values. The following parameters must be set via *Drive Wizard+* to match the *Simple Motion Control Library*.

No.	Designation	Range of values	Setting for *Simple Motion Control Library*
B1-01	Input source frequency setpoint 1	0, 1, 2, 3, 4	■ 3: Option card
B1-02	Input source start command 1	0, 1, 2, 3	■ 3: Option card
O1-03	Display scaling	0, 1, 2, 3, 4	■ 2: min-1 unit

For all settings to be accepted, you must restart the inverter drive after parametrization!

8.3 Wiring



(1) DC 24V for power section supply I/O area (max. 10A)
(2) DC 24V for electronic power supply CPU and I/O area

Proceeding

1. Turn off power supply of the CPU and the inverter drive.
2. If not already installed, install the EtherCAT option card in your inverter drive.
3. Connect the option card and the inverter drive via the enclosed ground cable.
4. Connect the EtherCAT jack *'X4'* of the CPU to the *'IN'* jack of the option card via an EtherCAT cable.

⇨ Your system is now ready for commissioning.

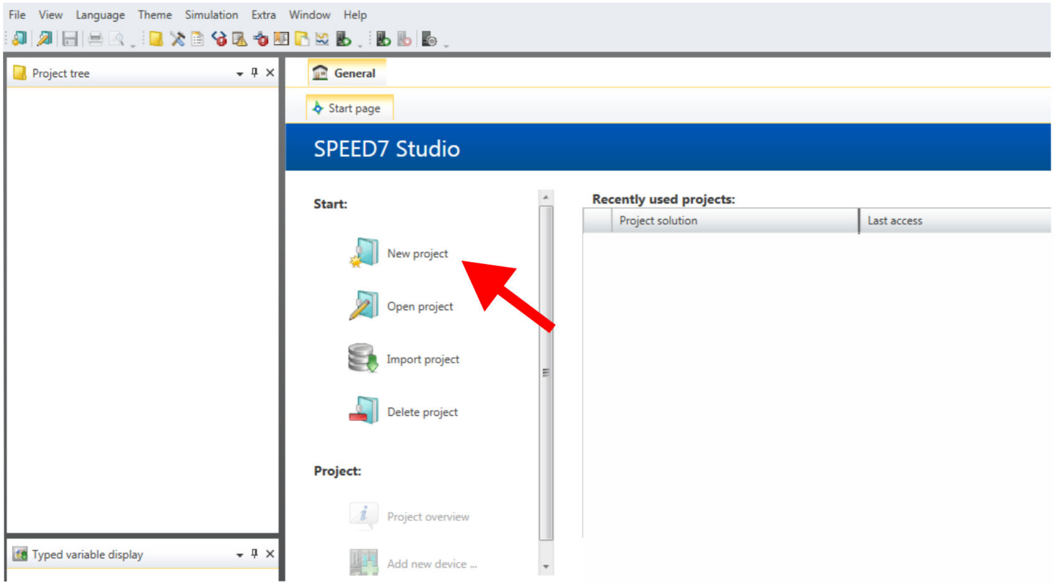
8.4 Usage in VIPA *SPEED7 Studio*

8.4.1 Hardware configuration

Add CPU in the project

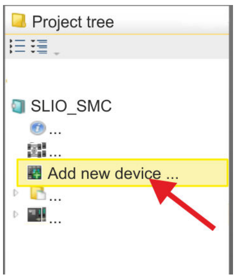
Please use the *SPEED7 Studio* V1.8 and up for the configuration.

1. Start the *SPEED7 Studio*.

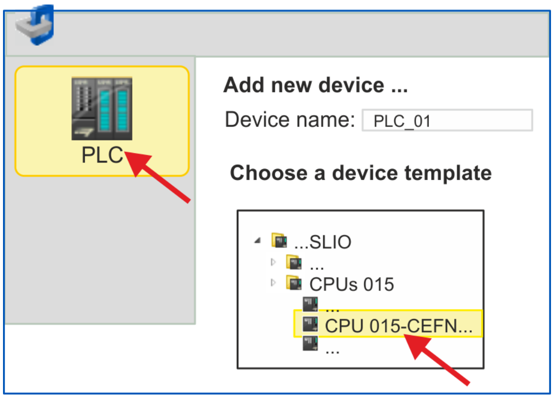


2. Create a new project at the start page with *'New project'* and assign a *'Project name'*.

⇨ A new project is created and the view *'Devices and networking'* is shown.



3. Click in the *Project tree* at *'Add new device ...'*.

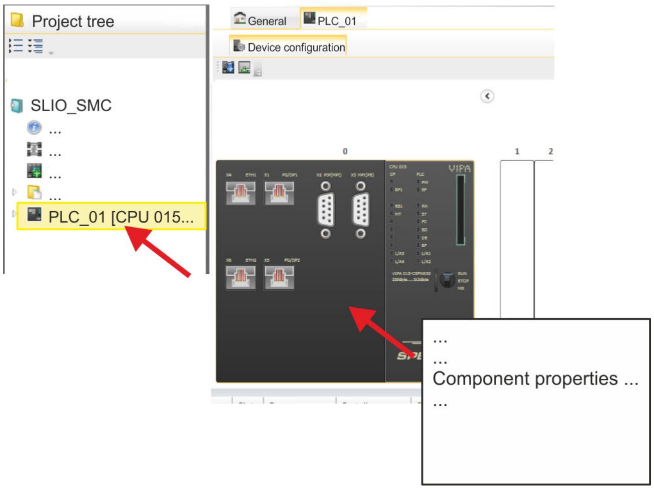


⇨ A dialog for device selection opens.

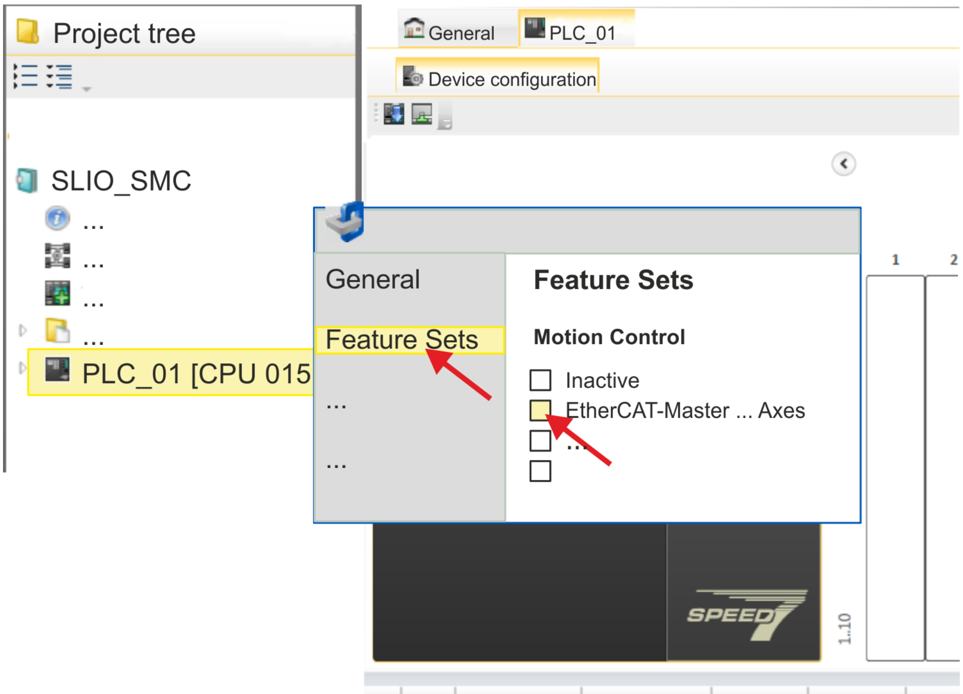
4. Select from the *'Device templates'* a CPU with EtherCAT master functionality such as the CPU 015-CEFNR00 and click at [OK].

⇨ The CPU is inserted in *'Devices and networking'* and the *'Device configuration'* is opened.

Activate motion control functions



1. Click at the CPU in the *'Device configuration'* and select *'Context menu ➔ Components properties'*.
   ⇨ The properties dialog of the CPU is opened.

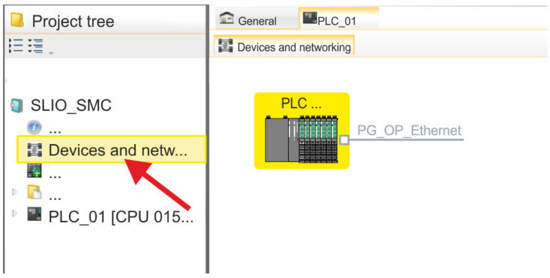


2. Click at *'Feature Sets'* and activate at *'Motion Control'* the parameter *'EtherCAT-Master... Axes'*. The number of axes is not relevant in this example.
3. Confirm your input with [OK].
   ⇨ The motion control functions are now available in your project.

CAUTION!
Please note due to the system, with every change to the feature set settings, the EtherCAT field bus system and its motion control configuration will be deleted from your project!

Configuration of Ethernet PG/OP channel

1. Click in the *Project tree* at *'Devices and networking'*.
   ⇨ You will get a graphical object view of your CPU.



2. Click at the network *'PG_OP_Ethernet'*.
3. Select *'Context menu ➔ Interface properties'*.
   ⇨ A dialog window opens. Here you can enter the IP address data for your Ethernet PG/OP channel. You get valid IP address parameters from your system administrator.
4. Confirm with [OK].
   ⇨ The IP address data are stored in your project listed in *'Devices and networking'* at *'Local components'*.

   After transferring your project your CPU can be accessed via Ethernet PG/OP channel with the set IP address data.

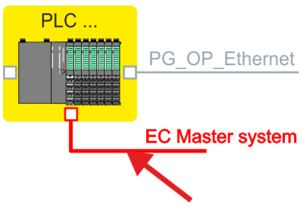
Installing the ESI file

For the inverter drive can be configured in the *SPEED7 EtherCAT Manager*, the corresponding ESI file must be installed. Usually, the *SPEED7 Studio* is delivered with current ESI files and you can skip this part. If your ESI file is not up-to date, you will find the latest ESI file for the inverter drive under www.yaskawa.eu.com at *'Service ➔ Drives & Motion Software'*.

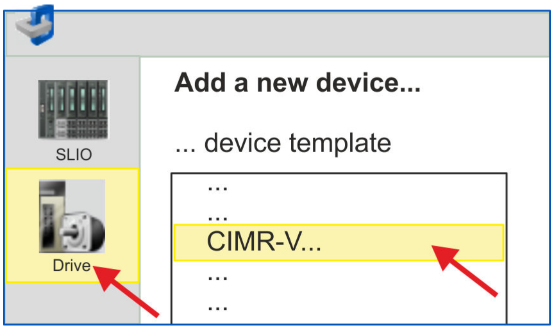
1. Download the according ESI file for your inverter drive. Unzip this if necessary.
2. Navigate to your *SPEED7 Studio*.
3. Open the corresponding dialog window by clicking on *'Extra ➔ Install device description (EtherCAT - ESI)'*.
4. Under *'Source path'*, specify the ESI file and install it with [Install].
   ⇨ The devices of the ESI file are now available.

Add an inverter drive

1. Click in the Project tree at *'Devices and networking'*.
2. Click here at *'EC-Mastersystem'* and select *'Context menu ➔ Add new device'*.



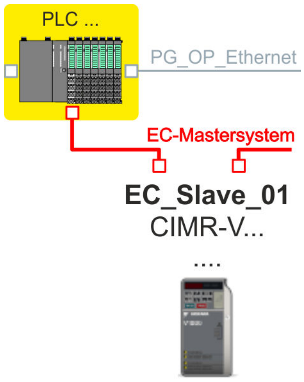
   ⇨ The device template for selecting an EtherCAT device opens.



3. Select your inverter drive:

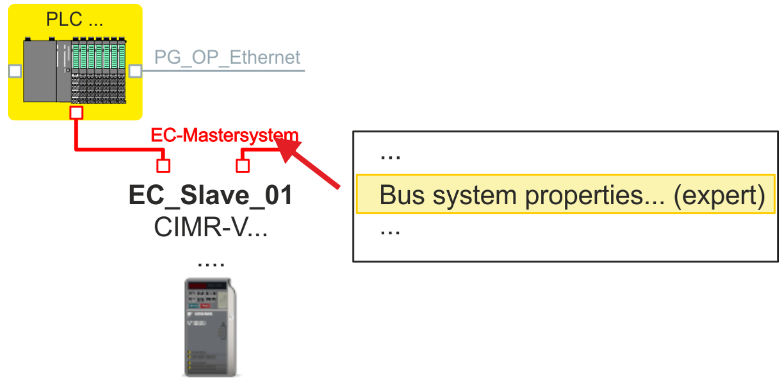
- CIMR-Vxxxx...
- CIPR-GA70xxxx...

Confirm with [OK]. If your drive does not exist, you must install the corresponding ESI file as described above.



⇨ The inverter drive is connected to your EC-Mastersystem.

Configure inverter drive

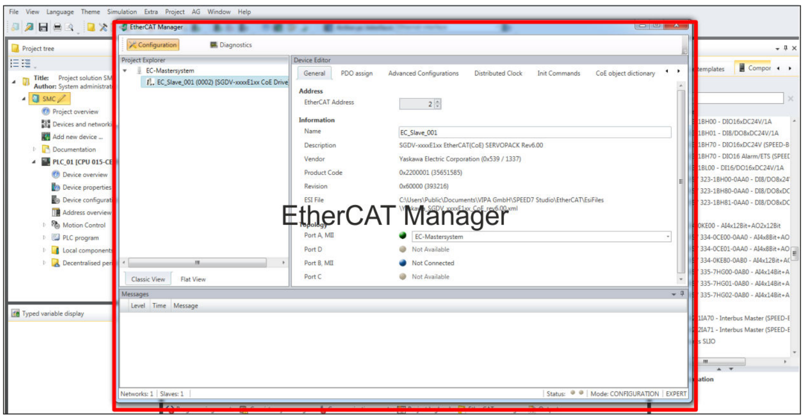


1. Click here at *'EC-Mastersystem'* and select *'Context menu → Bus system properties (expert)'*.

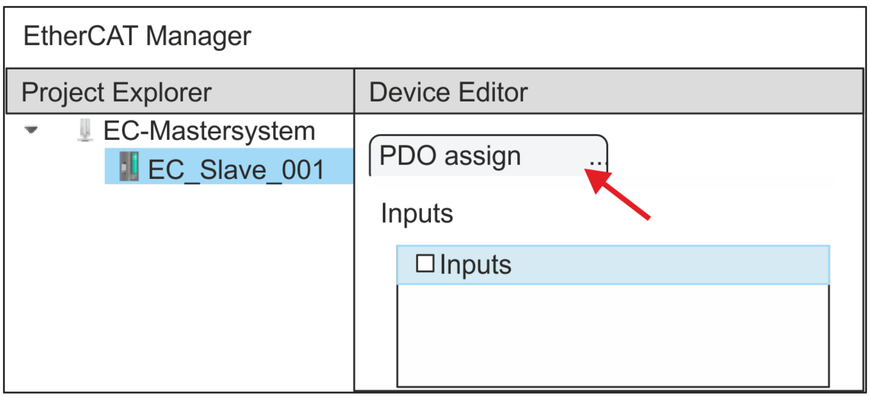
> *You can only edit PDOs in 'Expert mode'! Otherwise, the buttons are hidden.*

⇨ The *SPEED7 EtherCAT Manager* opens. Here you can configure the EtherCAT communication to your inverter drive.

More information about the usage of the *SPEED7 EtherCAT Manager* may be found in the online help of the *SPEED7 Studio*.



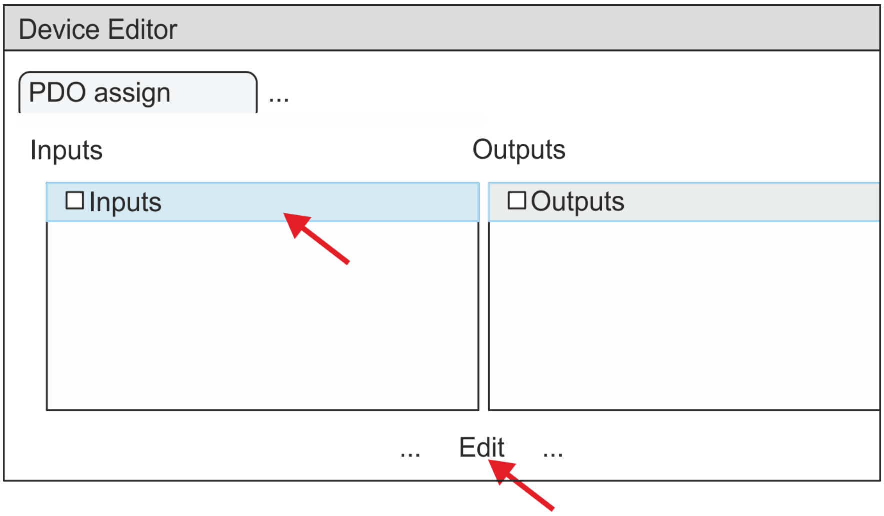
2. Click on the slave in the *SPEED7 EtherCAT Manager* and select the *'PDO assign'* tab in the *'Device editor'*.



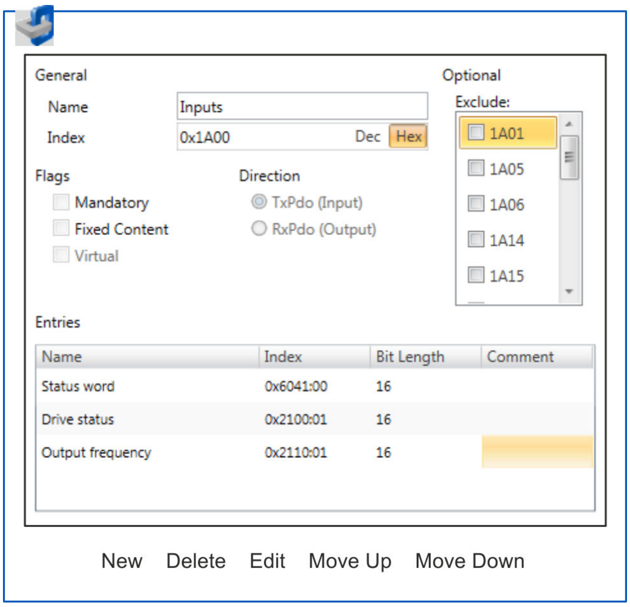
⇨ This dialog shows a list of the PDOs.

3. By selecting the appropriate mapping, you can edit the PDOs with [Edit]. Select the mapping *'Inputs'* and click at [Edit].

Please note that some PDOs can not be edited because of the default settings. By de-activating already activated PDOs, you can release the processing of locked PDOs.



⇨ The dialog *'Edit PDO'* is opened. Please check the PDO settings listed here and adjust them if necessary. Please also take into account the order of the *'Entries'* and add them accordingly.



The following functions are available for editing the *'Entries'*:

- New
  – Here you can create a new entry in a dialog by selecting the corresponding entry from the *'CoE object dictionary'* and making your settings. The entry is accepted with [OK] and is listed in the list of entries.
- Delete
  – This allows you to delete a selected entry.

- Edit
  – This allows you to edit the general data of an entry.
- Move Up/Down
  – This allows you to move the selected entry up or down in the list.

4. Perform the following settings:

Inputs

- General
  – Name: Inputs
  – Index: 0x1A00
- Flags
  – Everything de-activated
- Direction
  – TxPdo (Input): activated
- Exclude
  Please note these settings, otherwise the PDO mappings can not be activated at the same time!
  – Everything de-activated
- Entries

Name	Index	Bit length
Status word	0x6041:00	16bit
Drive status value	0x2100:01	16bit
Output frequency value	0x2110:01	16bit

Close the dialog *'Edit PDO'* with [OK].

5. Select the mapping *'Outputs'* and click at [Edit]. Perform the following settings:

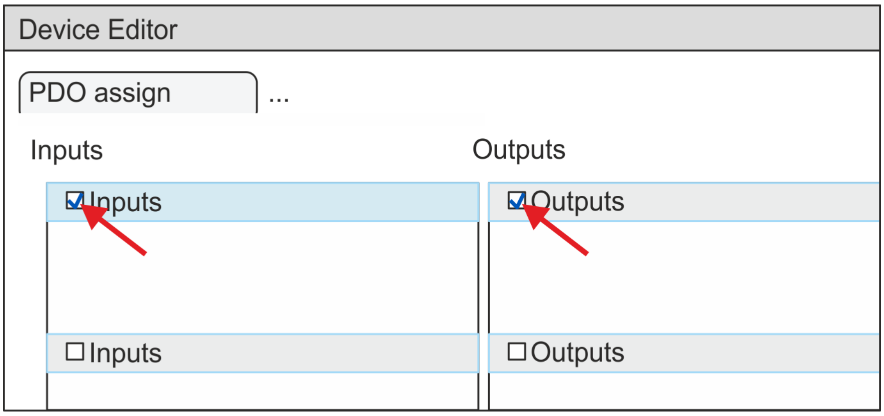
Outputs

- General
  – Name: Outputs
  – Index: 0x1600
- Flags
  – Everything de-activated
- Direction
  – RxPdo (Output): activated
- Exclude
  Please note these settings, otherwise the PDO mappings can not be activated at the same time!
  – Everything de-activated
- Entries

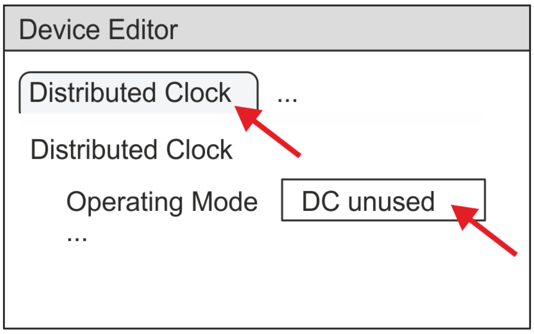
Name	Index	Bit length
Control word	0x6040:00	16bit
vl target velocity	0x6042:00	16bit
vl velocity acceleration: Delta speed	0x6048:01	32bit
vl velocity acceleration: Delta time	0x6048:02	16bit

Close the dialog *'Edit PDO'* with [OK].

6. In PDO assignment, activate each 1. PDOs "Inputs" and "Outputs". All subsequent PDOs must remain de-activated. If this is not possible, please check the respective PDO parameter *'Exclude'*.

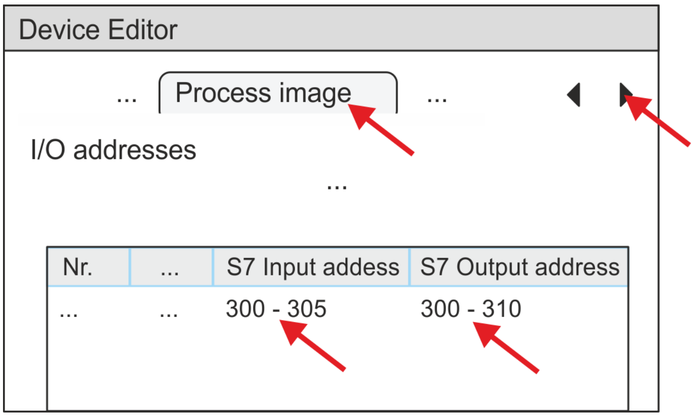


7. In the *'Device Editor'* of the *SPEED7 EtherCAT Manager*, select the *'Distributed clocks'* tab and set *'DC unused'* as *'Operating mode'*.



8. Select the *'Process image'* tab via the arrow key in the *'Device editor'* and note for the parameter of the block FB 887 - VMC_InitInverter_EC the following PDO.

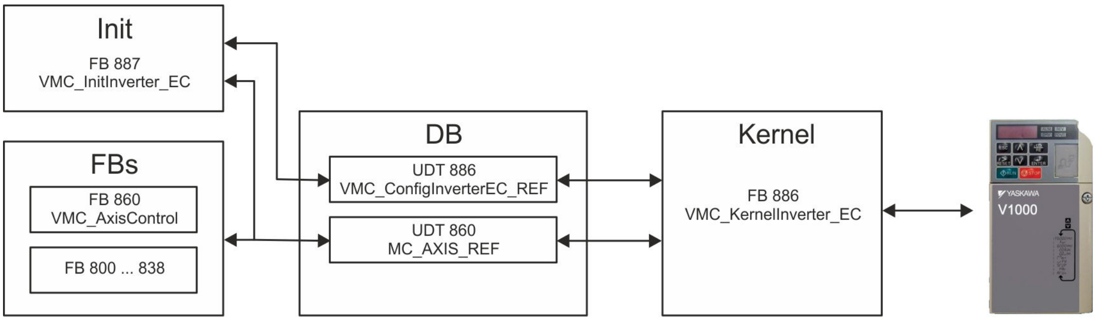
- *'S7 Input address'* → *'InputsStartAddressPDO'*
- *'S7 Output address'* → *'OutputsStartAddressPDO'*



9. By closing the dialog of the *SPEED7 EtherCAT Manager* with [X] the configuration is taken to the *SPEED7 Studio*.

8.4.2 User program

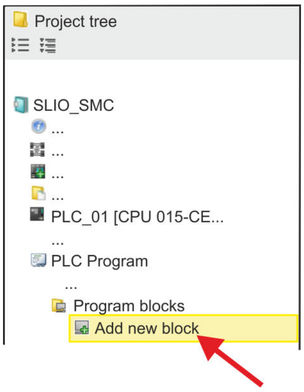
8.4.2.1 Program structure



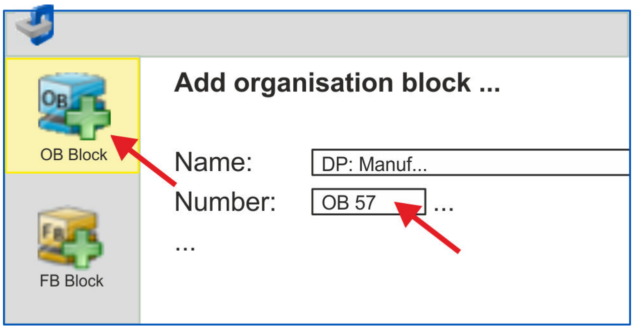
- DB
  A data block (axis DB) for configuration and status data must be created for each axis of a drive. The data block consists of the following data structures:
  – UDT 886 - *VMC_Config*InverterEC_REF
    The data structure describes the structure of the configuration of the drive.
    Specific data structure for inverter drive with EtherCAT.
  – UDT 860 - *MC_AXIS_REF*
    The data structure describes the structure of the parameters and status information of drives.
    General data structure for all drives and bus systems.
- FB 887 - *VMC_Init*Inverter_EC
  – The *Init*t block is used to configure an axis.
  – Specific block for inverter drive with EtherCAT.
  – The configuration data for the initialization must be stored in the *axis DB*.
- FB 886 - *VMC_Kernel*Inverter_EC
  – The *Kernel* block communicates with the drive via the appropriate bus system, processes the user requests and returns status messages.
  – Specific block for inverter drive with EtherCAT.
  – The exchange of the data takes place by means of the *axis DB*.
- FB 860 - *VMC_AxisControl*
  – General block for all drives and bus systems.
  – Supports simple motion commands and returns all relevant status messages.
  – The exchange of the data takes place by means of the *axis DB*.
  – For motion control and status query, via the instance data of the block you can link a visualization.
  – In addition to the FB 860 - *VMC_AxisControl*, *PLCopen* blocks can be used.
- FB 800 ... FB 838 - *PLCopen*
  – The PLCopen blocks are used to program motion sequences and status queries.
  – General blocks for all drives and bus systems.

8.4.2.2 Programming

Copy blocks into project

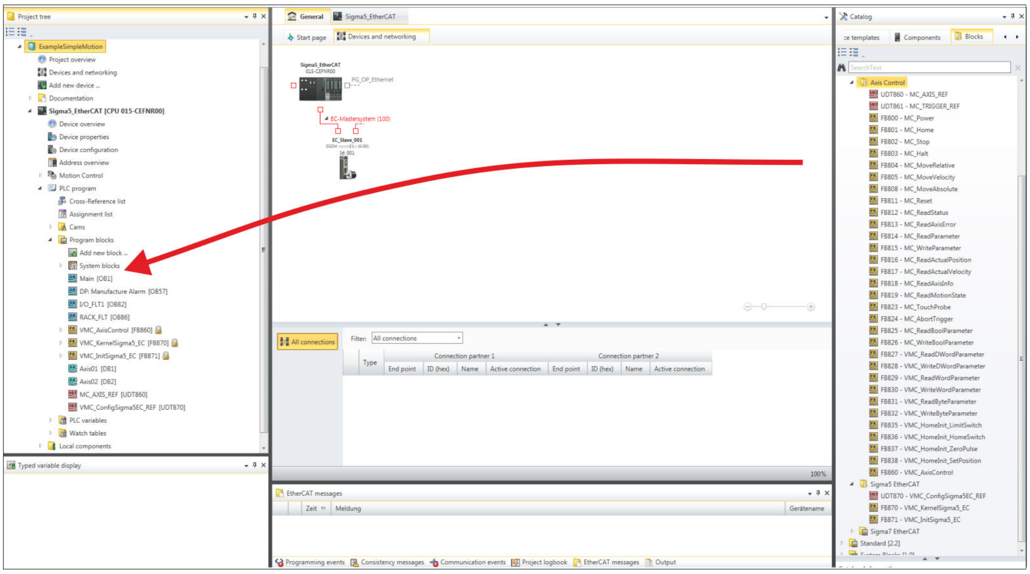


1. Click in the *Project tree* within the CPU at *'PLC program'*, *'Program blocks'* at *'Add New block'*.



⇨ The dialog *'Add block'* is opened.

2. Select the block type *'OB block'* and add OB 57, OB 82 and OB 86 to your project.



3. In the *'Catalog'*, open the *'Simple Motion Control'* library at *'Blocks'* and drag and drop the following blocks into *'Program blocks'* of the *Project tree*:
   - *Inverter* EtherCAT:
     – UDT 886 - VMC_ConfigInverterEC_REF
     – FB 886 - VMC_KernelInverter_EC
     – FB 887 - VMC_InitInverter_EC
   - Axis Control
     – UDT 860 - MC_AXIS_REF
     – Blocks for your movement sequences

Create axis DB

1. Add a new DB as your *axis DB* to your project. Click in the *Project tree* within the CPU at *'PLC program'*, *'Program blocks'* at *'Add New block'*, select the block type *'DB block'* and assign the name "Axis01" to it. The DB number can freely be selected such as DB 10.

⇨ The block is created and opened.

2.
- In "Axis01", create the variable "Config" of type UDT 886. These are specific axis configuration data.
- In "Axis01", create the variable "Axis" of type UDT 860. During operation, all operating data of the axis are stored here.

Axis01 [DB10]
Data block structure

		Adr...	Name	Data type	...
		...	Config	UDT	[886]
		...	Axis	UDT	[860]

OB 1

Configuration of the axis

Open OB 1 and program the following FB calls with associated DBs:

→ FB 887 - VMC_InitInverter_EC, DB 887 *Chap. 8.6.3 'FB 887 - VMC_InitInverter_EC - inverter drive EtherCAT initialization' page 359*

At *InputsStartAddressPDO* respectively *OutputsStartAddressPDO*, enter the address from the *SPEED7 EtherCAT Manager*. *339*

⇨
```
CALL  "VMC_InitInverter_EC" , "DI_InitInvEC01"
Enable                  :="InitInvEC1_Enable"
LogicalAddress          :=300
InputsStartAddressPDO :=300 (EtherCAT-Man.: S7 Input
address)
OutputsStartAddressPDO:=300 (EtherCAT-Man.: S7 Output
address)
MaxVelocityDrive        :=1.000000e+002
MaxOutputFrequency      :=6.000000e+001
NumberOfPoles           :=6
Valid                   :="InitInvEC1_Valid"
Error                   :="InitInvEC1_Error"
ErrorID                 :="InitInvEC1_ErrorID"
MaxVelocity             :="InitInvEC1_MaxVelocityRPM"
Config                  :="Axis01".Config
Axis                    :="Axis01".Axis
```

Connecting the Kernel for the axis

The *Kernel* processes the user commands and passes them appropriately processed on to the drive via the respective bus system.

→ FB 886 - VMC_KernelInverter_EC, DB 886 *Chap. 8.6.2 'FB 886 - VMC_KernelInverter_EC - inverter drive EtherCAT kernel' page 359*

⇨
```
      CALL  "VMC_KernelInverter_EC" , "DI_KernelInvEC01"
       Init   :="KernelInvEC1_Init"
       Config:="Axis01".Config
       Axis   :="Axis01".Axis
```

Connecting the block for motion sequences

For simplicity, the connection of the FB 860 - VMC_AxisControl is to be shown here. This universal block supports simple motion commands and returns status messages. The inputs and outputs can be individually connected. Please specify the reference to the corresponding axis data at *'Axis'* in the *axis DB*.

FB 860 - VMC_AxisControl, DB 860 *Chap. 9.2.2 'FB 860 - VMC_AxisControl - Control block axis control' page 363*

```
        CALL  "VMC_AxisControl" , "DI_AxisControl01"
         AxisEnable         :="AxCtrl1_AxisEnable"
         AxisReset          :="AxCtrl1_AxisReset"
         HomeExecute*       :="AxCtrl1_HomeExecute"
         HomePosition*      :="AxCtrl1_HomePosition"
         StopExecute        :="AxCtrl1_StopExecute"
         MvVelocityExecute  :="AxCtrl1_MvVelExecute"
         MvRelativeExecute* :="AxCtrl1_MvRelExecute"
         MvAbsoluteExecute* :="AxCtrl1_MvAbsExecute"
         PositionDistance*  :="AxCtrl1_PositionDistance"
         Velocity           :="AxCtrl1_Velocity"
         Acceleration       :="AxCtrl1_Acceleration"
         Deceleration       :="AxCtrl1_Deceleration"
         JogPositive        :="AxCtrl1_JogPositive"
         JogNegative        :="AxCtrl1_JogNegative"
         JogVelocity        :="AxCtrl1_JogVelocity"
         JogAcceleration    :="AxCtrl1_JogAcceleration"
         JogDeceleration    :="AxCtrl1_JogDeceleration"
         AxisReady          :="AxCtrl1_AxisReady"
         AxisEnabled        :="AxCtrl1_AxisEnabled"
         AxisError          :="AxCtrl1_AxisError"
         AxisErrorID        :="AxCtrl1_AxisErrorID"
         DriveWarning       :="AxCtrl1_DriveWarning"
         DriveError         :="AxCtrl1_DriveError"
         DriveErrorID       :="AxCtrl1_DriveErrorID"
         IsHomed*           :="AxCtrl1_IsHomed"
         ModeOfOperation    :="AxCtrl1_ModeOfOperation"
         PLCopenState       :="AxCtrl1_PLCopenState"
         ActualPosition*    :="AxCtrl1_ActualPosition"
         ActualVelocity     :="AxCtrl1_ActualVelocity"
         CmdDone            :="AxCtrl1_CmdDone"
         CmdBusy            :="AxCtrl1_CmdBusy"
         CmdAborted         :="AxCtrl1_CmdAborted"
         CmdError           :="AxCtrl1_CmdError"
         CmdErrorID         :="AxCtrl1_CmdErrorID"
         DirectionPositive  :="AxCtrl1_DirectionPos"
         DirectionNegative  :="AxCtrl1_DirectionNeg"
         SWLimitMinActive*  :="AxCtrl1_SWLimitMinActive"
         SWLimitMaxActive*  :="AxCtrl1_SWLimitMaxActive"
         HWLimitMinActive*  :="AxCtrl1_HWLimitMinActive"
         HWLimitMaxActive*  :="AxCtrl1_HWLimitMaxActive"
         Axis               :="Axis01".Axis

*) This Parameter is not supported by an inverter.
```

For complex motion tasks, you can use the PLCopen blocks. Please specify the reference to the corresponding axis data at Axis in the axis DB.

Your project now includes the following blocks:

- OB 1 - Main
- OB 57 - DP Manufacturer Alarm
- OB 82 - I/O_FLT1

- OB 86 - Rack_FLT
- FB 860 - VMC_AxisControl with instance DB
- FB 886 - VMC_KernelInverter_EC with instance DB
- FB 887 - VMC_InitInverter_EC with instance DB
- UDT 860 - MC_Axis_REF
- UDT 886 - VMC_ConfigInverterEC_REF

Sequence of operations

1. Select *'Project ➔ Compile all'* and transfer the project into your CPU.

   You can find more information on the transfer of your project in the online help of the *SPEED7 Studio*.

   ⇨ 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. To do this, call the *Init* block FB 887 - VMC_InitInverter_EC with *Enable* = TRUE.

   ⇨ The output *Valid* returns TRUE. In the event of a fault, you can determine the error by evaluating the *ErrorID*.

   You have to call the *Init* block again if you load a new axis DB or you have changed parameters on the *Init* block.

   Do not continue until the Init block does not report any errors!

3. Ensure that the *Kernel* block FB 886 - VMC_KernelInverter_EC is cyclically called. In this way, control signals are transmitted to the drive and status messages are reported.

4. Program your application with the FB 860 - VMC_AxisControl or with the PLCopen blocks.

Controlling the drive via HMI

There is the possibility to control your drive via HMI. For this, a predefined symbol library is available for Movicon to access the VMC_AxisControl function block. *Chap. 10 'Controlling the drive via HMI' page 432*

8.5 Usage in Siemens SIMATIC Manager

8.5.1 Precondition

Overview

- Please use for configuration the Siemens SIMATIC Manager V 5.5 SP2 and up.
- 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 CPU'* is to be installed in the hardware catalog by means of the GSDML.
- The configuration of the EtherCAT masters happens in the Siemens SIMATIC Manager by means of a virtual PROFINET IO device *'EtherCAT network'*. The *'EtherCAT network'* is to be installed in the hardware catalog by means of the GSDML.
- The *'EtherCAT network'* can be configured with the VIPA Tool *SPEED7 EtherCAT Manager*.
- For the configuration of the drive in the *SPEED7 EtherCAT Manager* the installation of the according ESI file is necessary.

Installing the IO device *'VIPA SLIO System'*

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'*.

Installing the IO device EtherCAT network

The installation of the PROFINET IO devices *'EtherCAT Network'* happens in the hardware catalog with the following approach:

1. Go to the service area of www.vipa.com
2. Load from the download area at *'Config files ➔ EtherCAT'* the GSDML file for your EtherCAT master.
3. Extract the files 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 *'EtherCAT Network'* can be found at *'PROFINET IO ➔ Additional field devices ➔ I/O ➔ VIPA EtherCAT System'*.

Installing the *SPEED7 EtherCAT Manager*

The configuration of the PROFINET IO device *'EtherCAT Network'* happens by means of the VIPA *SPEED7 EtherCAT Manager*. This may be found in the service area of www.vipa.com at *'Service/Support ➔ Downloads ➔ Software'*.

The installation happens with the following proceeding:

1. Close the Siemens SIMATIC Manager.
2. Go to the service area of www.vipa.com
3. Load the *SPEED7 EtherCAT Manager* and unzip it on your PC.

4. For installation start the file EtherCATManager_v... .exe.
5. Select the language for the installation.
6. Accept the licensing agreement.
7. Select the installation directory and start the installation.
8. After installation you have to reboot your PC.
   ⇨ The *SPEED7 EtherCAT Manager* is installed and can now be called via the context menu of the Siemens SIMATIC Manager.

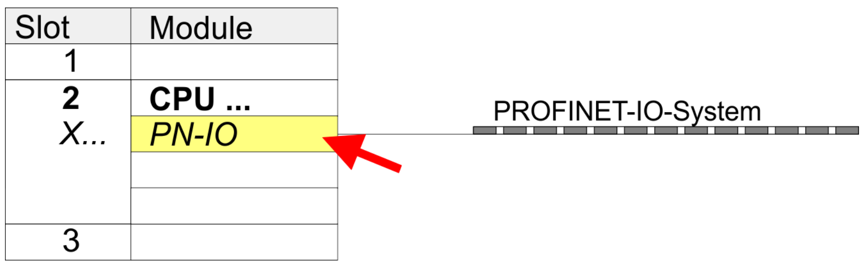
8.5.2 Hardware configuration

Configuring the 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	

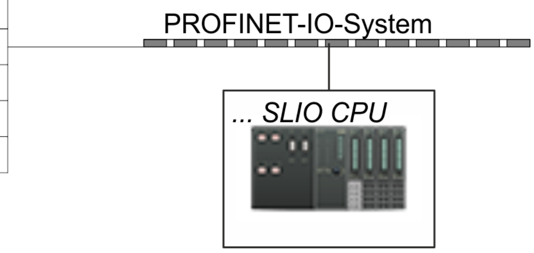
To be compatible with the Siemens SIMATIC Manager the following steps should be executed:

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 the CPU 315-2 PN/DP (315-2EH14 V3.2).
4. The integrated PROFIBUS DP master (jack X3) is to be configured and connected via the sub module *'X1 MPI/DP'*.
5. The integrated EtherCAT master is to be configured via the sub module *'X2 PN-IO'* as a virtual PROFINET network.
6. Click at the sub module *'PN-IO'* of the CPU.
7. Select *'Context menu → Insert PROFINET IO System'*.



8. Create with [New] a new sub net and assign valid address data.
9. Click at the sub module *'PN-IO'* of the CPU and open with *'Context menu → Properties'* the properties dialog.
10. Enter at *'General'* a *'Device name'*. The device name must be unique at the Ethernet subnet.

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



Slot	Module	Order number	
0	***... SLIO CPU ...***	***015-...***	
X2	*015-...*		
1			
2			
3			
...			

11. Navigate in the hardware catalog to the directory *'PROFINET IO ➔ Additional field devices ➔ I/O ➔ VIPA SLIO System'* and connect the IO device *'015-CFFNR00 CPU'* to your PROFINET system.

⇨ In the Device 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

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

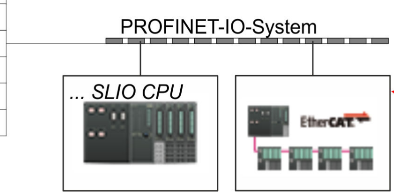
1. Place for the Ethernet PG/OP channel at slot 4 the Siemens CP 343-1 (SIMATIC 300 \ CP 300 \ Industrial Ethernet \CP 343-1 \ 6GK7 343-1EX30 0XE0 V3.0).

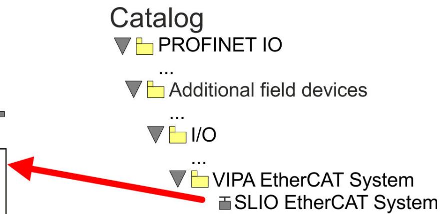
2. Open the properties dialog by clicking on the CP 343-1EX30 and enter for the CP at *'Properties'* the IP address data. You get valid IP address parameters from your system administrator.

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

Insert *'EtherCAT network'*

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



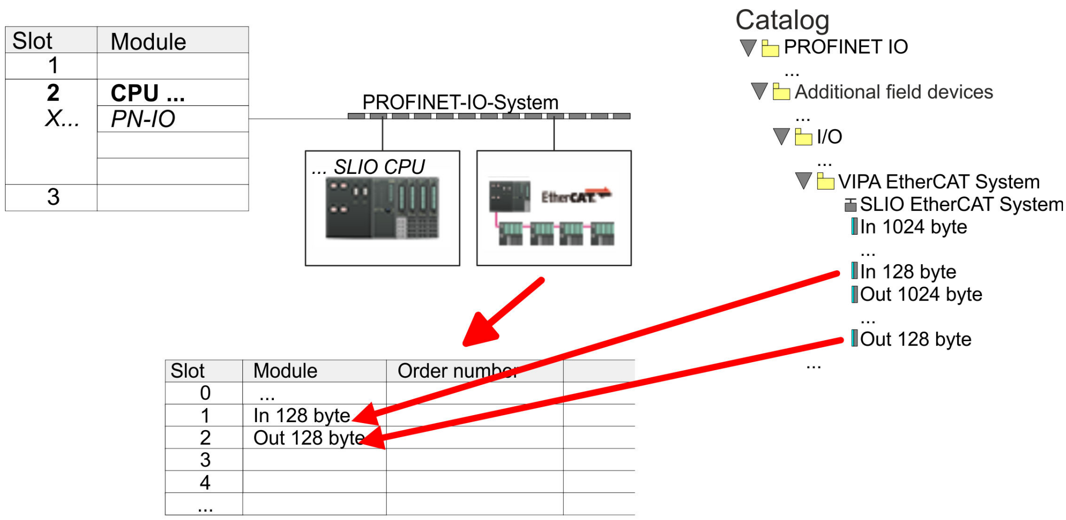


1. Navigate in the hardware catalog to the directory *'PROFINET IO ➔ Additional field devices ➔ I/O ➔ VIPA EtherCAT System'* and connect the IO device *'SLIO EtherCAT System'* to your PROFINET system.

2. Click at the inserted IO device *'EtherCAT Network'* and define the areas for in and output by drag and dropping the according *'Out'* or *'In'* area to a slot.

   Create the following areas:

   - In 128byte
   - Out 128byte

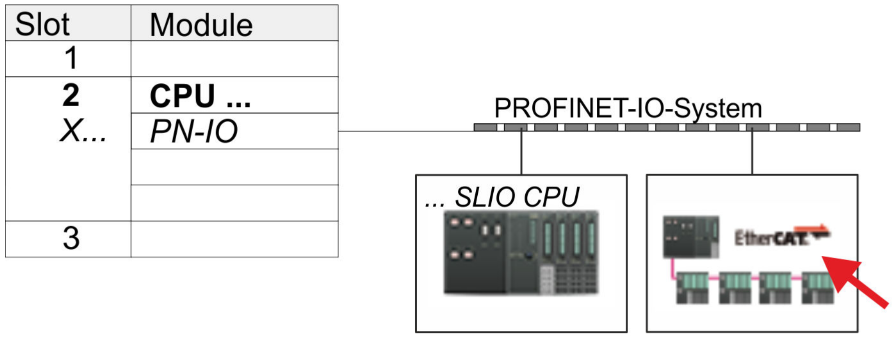


3. Select *'Station → Save and compile'*

Configure inverter drive

The drive is configured in the *SPEED7 EtherCAT Manager*.

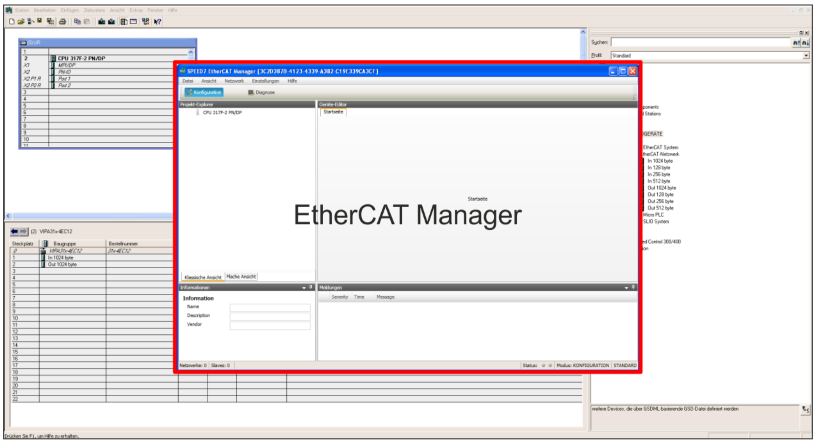
Before calling the SPEED7 EtherCAT Manager you have always to save your project with 'Station ➔ Save and compile'.



1. Click at an inserted IO device *'EtherCAT Network'* and select *'Context menu ➔ Start Device-Tool ➔ SPEED7 EtherCAT Manager'*.

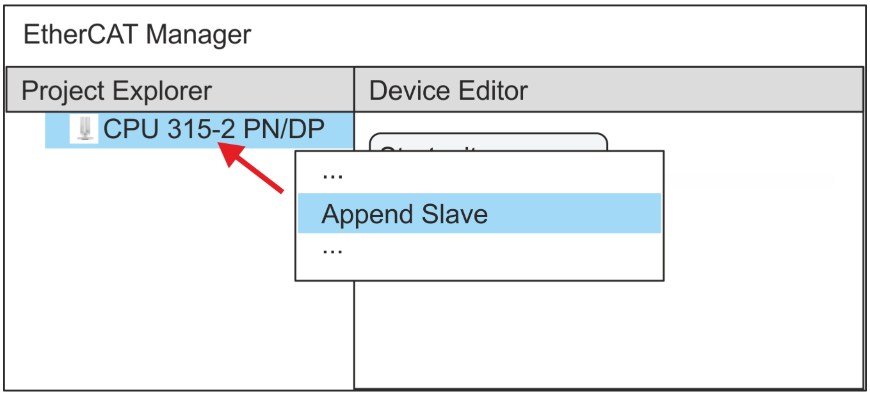
   ⇨ The *SPEED7 EtherCAT Manager* opens. Here you can configure the EtherCAT communication to your inverter drive.

   More information about the usage of the *SPEED7 EtherCAT Manager* may be found in the according manual or online help.



3. For the inverter drive to be configured in the *SPEED7 EtherCAT Manager*, the corresponding ESI file must be installed. The ESI file for the inverter drive can be found under <u>*www.yaskawa.eu.com*</u> at *'Service ➔ Drives & Motion Software'*. Download the according ESI file for your drive. Unzip this if necessary.
4. Open in the *SPEED7 EtherCAT Manager* via *'File ➔ ESI Manager'* the dialog window *'ESI Manager'*.
5. In the *'ESI Manager'* click at [Add File] and select your ESI file. With [Open], the ESI file is installed in the *SPEED7 EtherCAT Manager*.
6. Close the *'ESI Manager'*.

   ⇨ Your inverter drive is now available for configuration.



7. In the EtherCAT Manager, click on your CPU and open via *'Context menu ➔ Append Slave'* the dialog box for adding an EtherCAT slave.

   ⇨ The dialog window for selecting an EtherCAT slave is opened.

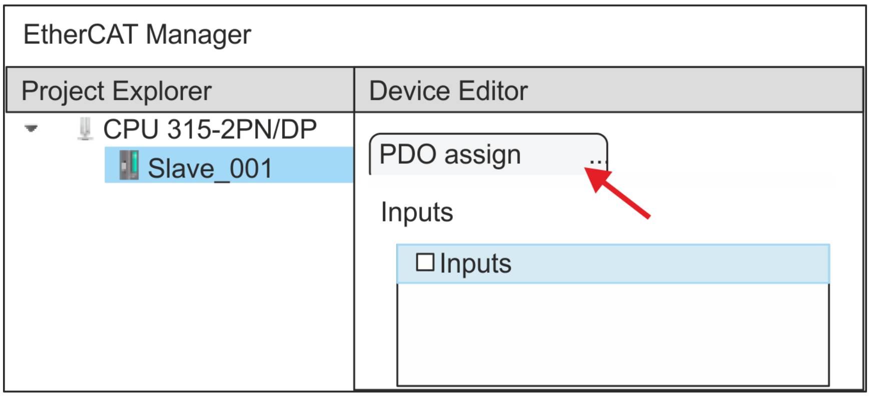
8. Select your inverter drive and confirm your selection with [OK].

   ⇨ The inverter drive is connected to the master and can now be configured.

9. *You can only edit PDOs in 'Expert mode'! Otherwise, the buttons are hidden. By activating the 'Expert mode' you can switch to advanced setting.*

   By activating *'View ➔ Expert'* you can switch to the *Expert mode*.

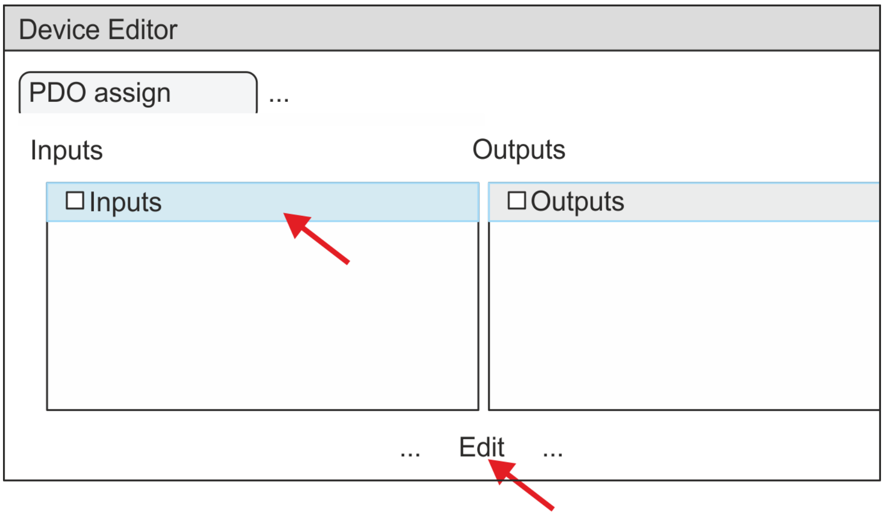
10. Click on the inverter drive EtherCAT Slave in the *SPEED7 EtherCAT Manager* and select the *'PDO assign'* tab in the *'Device editor'*.



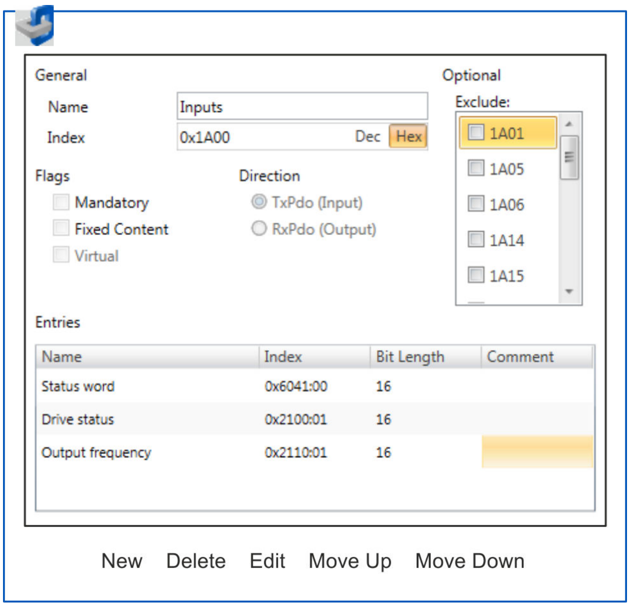
   ⇨ This dialog shows a list of the PDOs.

11. By selecting the appropriate PDO mapping, you can edit the PDOs with [Edit]. Select the mapping *'Inputs'* and click at [Edit].

> *Please note that some PDOs can not be edited because of the default settings. By de-activating already activated PDOs, you can release the processing of locked PDOs.*



⇨ The dialog *'Edit PDO'* is opened. Please check the PDO settings listed here and adjust them if necessary. Please also take into account the order of the *'Entries'* and add them accordingly.



The following functions are available for editing the *'Entries'*:

- New
  – Here you can create a new entry in a dialog by selecting the corresponding entry from the *'CoE object dictionary'* and making your settings. The entry is accepted with [OK] and is listed in the list of entries.
- Delete
  – This allows you to delete a selected entry.

- Edit
  - This allows you to edit the general data of an entry.
- Move Up/Down
  - This allows you to move the selected entry up or down in the list.

12. Perform the following settings:

Inputs

- General
  - Name: Inputs
  - Index: 0x1A00
- Flags
  - Everything de-activated
- Direction
  - TxPdo (Input): activated
- Exclude
  Please note these settings, otherwise the PDO mappings can not be activated at the same time!
  - Everything de-activated
- Entries

Name	Index	Bit length
Status word	0x6041:00	16bit
Drive status value	0x2100:01	16bit
Output frequency value	0x2110:01	16bit

Close the dialog *'Edit PDO'* with [OK].

13. Select the mapping *'1st Receive PDO mapping'* and click at [Edit]. Perform the following settings:

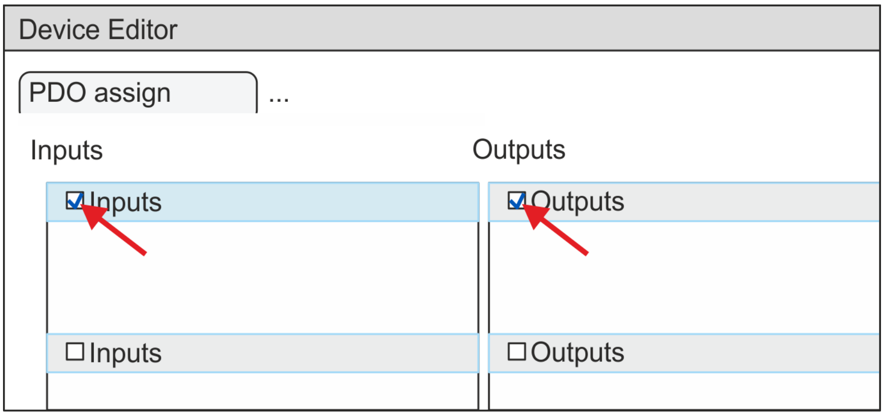
Outputs

- General
  - Name: Outputs
  - Index: 0x1600
- Flags
  - Everything de-activated
- Direction
  - RxPdo (Output): activated
- Exclude
  Please note these settings, otherwise the PDO mappings can not be activated at the same time!
  - Everything de-activated
- Entries

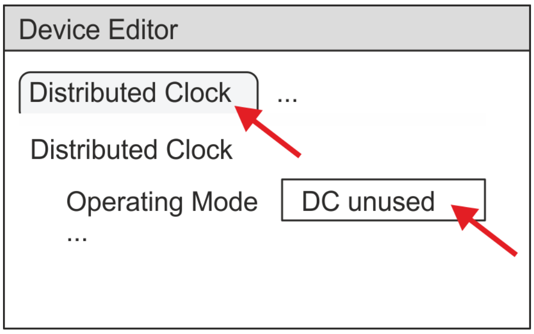
Name	Index	Bit length
Control word	0x6040:00	16bit
vl target velocity	0x6042:00	16bit
vl velocity acceleration: Delta speed	0x6048:01	32bit
vl velocity acceleration: Delta time	0x6048:02	16bit

Close the dialog *'Edit PDO'* with [OK].

14. In PDO assignment, activate each 1. PDOs "Inputs" and "Outputs". All subsequent PDOs must remain de-activated. If this is not possible, please check the respective PDO parameter *'Exclude'*.

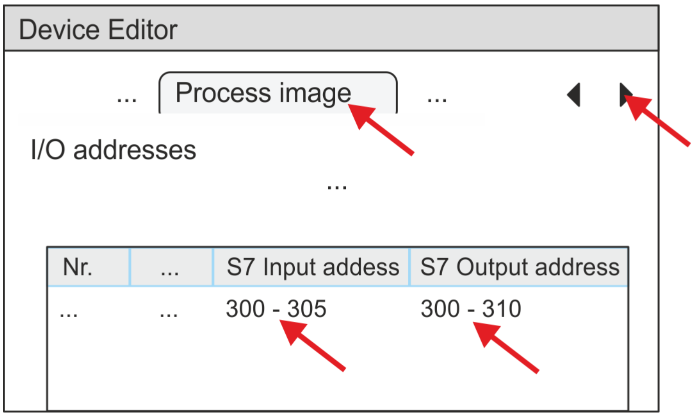


15. In the *'Device Editor'* of the *SPEED7 EtherCAT Manager*, select the *'Distributed clocks'* tab and set *'DC unused'* as *'Operating mode'*.



16. Select the *'Process image'* tab via the arrow key in the *'Device editor'* and note for the parameter of the block FB 887 - VMC_InitInverter_EC the following PDO.

- *'S7 Input address'* → *'InputsStartAddressPDO'*
- *'S7 Output address'* → *'OutputsStartAddressPDO'*

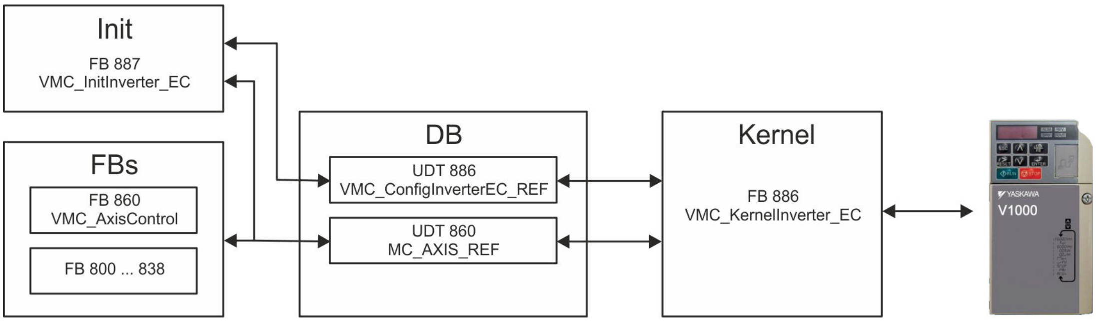


17. By closing the *SPEED7 EtherCAT Manager* with [X] the configuration is taken to the project. You can always edit your EtherCAT configuration in the *SPEED7 EtherCAT Manager*, since the configuration is stored in your project.

18. Save and compile your configuration

8.5.3 User program

8.5.3.1 Program structure



- DB
  A data block (axis DB) for configuration and status data must be created for each axis of a drive. The data block consists of the following data structures:
  – UDT 886 - *VMC_Config*InverterEC_REF
    The data structure describes the structure of the configuration of the drive.
    Specific data structure for inverter drive with EtherCAT.
  – UDT 860 - *MC_AXIS_REF*
    The data structure describes the structure of the parameters and status information of drives.
    General data structure for all drives and bus systems.
- FB 887 - *VMC_Init*Inverter_EC
  – The *Init*t block is used to configure an axis.
  – Specific block for inverter drive with EtherCAT.
  – The configuration data for the initialization must be stored in the *axis DB*.
- FB 886 - *VMC_Kernel*Inverter_EC
  – The *Kernel* block communicates with the drive via the appropriate bus system, processes the user requests and returns status messages.
  – Specific block for inverter drive with EtherCAT.
  – The exchange of the data takes place by means of the *axis DB*.
- FB 860 - *VMC_AxisControl*
  – General block for all drives and bus systems.
  – Supports simple motion commands and returns all relevant status messages.
  – The exchange of the data takes place by means of the *axis DB*.
  – For motion control and status query, via the instance data of the block you can link a visualization.
  – In addition to the FB 860 - *VMC_AxisControl*, *PLCopen* blocks can be used.
- FB 800 ... FB 838 - *PLCopen*
  – The PLCopen blocks are used to program motion sequences and status queries.
  – General blocks for all drives and bus systems.

8.5.3.2 Programming

Include library

1. Go to the service area of www.vipa.com.
2. Download the *Simple Motion Control* library from the download area at *‘VIPA Lib’*.
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:

- Inverter EtherCAT:
  – UDT 886 - VMC_ConfigInverterEC_REF
  – FB 886 - VMC_KernelInverter_EC
  – FB 887 - VMC_InitInverter_EC
- Axis Control
  – UDT 860 - MC_AXIS_REF
  – Blocks for your movement sequences

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.

Create axis DB

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

   Specify the following parameters:

   - Name and type
     – The DB no. as *'Name'* can freely be chosen, such as DB 10.
     – Set *'Shared DB'* as the *'Type'*.
   - Symbolic name
     – Specify "Axis01".

   Confirm your input with [OK].

   ⇨ The block is created.

2. Open DB 10 "Axis01" by double-click.

   - In "Axis01", create the variable "Config" of type UDT 886. These are specific axis configuration data.
   - In "Axis01", create the variable "Axis" of type UDT 860. During operation, all operating data of the axis are stored here.

DB10

Address	Name	Typ	...
		Struct	
...	Config	"VMC_ConfigInverterEC_REF"	
...	Axis	"MC_AXIS_REF	
...		END_STRUCT	

OB 1

Configuration of the axis

Open OB 1 and program the following FB calls with associated DBs:

FB 887 - VMC_InitInverter_EC, DB 887 *Chap. 8.6.3 'FB 887 - VMC_InitInverter_EC - inverter drive EtherCAT initialization' page 359*

At *InputsStartAddressPDO* respectively *OutputsStartAddressPDO*, enter the address from the *SPEED7 EtherCAT Manager*. *354*

```
CALL  "VMC_InitInverter_EC" , "DI_InitInvEC01"
Enable                 :="InitInvEC1_Enable"
LogicalAddress         :=300
InputsStartAddressPDO :=300 (EtherCAT-Man.: S7 Input
address)
OutputsStartAddressPDO:=300 (EtherCAT-Man.: S7 Output
address)
MaxVelocityDrive       :=1.000000e+002
MaxOutputFrequency     :=6.000000e+001
NumberOfPoles          :=6
Valid                  :="InitInvEC1_Valid"
Error                  :="InitInvEC1_Error"
ErrorID                :="InitInvEC1_ErrorID"
MaxVelocity            :="InitInvEC1_MaxVelocityRPM"
Config                 :="Axis01".Config
Axis                   :="Axis01".Axis
```

Connecting the Kernel for the axis

The *Kernel* processes the user commands and passes them appropriately processed on to the drive via the respective bus system.

FB 886 - VMC_KernelInverter_EC, DB 886 *Chap. 8.6.2 'FB 886 - VMC_KernelInverter_EC - inverter drive EtherCAT kernel' page 359*

```
      CALL  "VMC_KernelInverter_EC" , "DI_KernelInvEC01"
       Init   :="KernelInvEC1_Init"
       Config:="Axis01".Config
       Axis   :="Axis01".Axis
```

Connecting the block for motion sequences

For simplicity, the connection of the FB 860 - VMC_AxisControl is to be shown here. This universal block supports simple motion commands and returns status messages. The inputs and outputs can be individually connected. Please specify the reference to the corresponding axis data at *'Axis'* in the *axis DB*.

FB 860 - VMC_AxisControl, DB 860 *Chap. 9.2.2 'FB 860 - VMC_AxisControl - Control block axis control' page 363*

```
CALL  "VMC_AxisControl" , "DI_AxisControl01"
 AxisEnable        :="AxCtrl1_AxisEnable"
 AxisReset         :="AxCtrl1_AxisReset"
 HomeExecute       :="AxCtrl1_HomeExecute"
 HomePosition      :="AxCtrl1_HomePosition"
 StopExecute       :="AxCtrl1_StopExecute"
 MvVelocityExecute:="AxCtrl1_MvVelExecute"
 MvRelativeExecute:="AxCtrl1_MvRelExecute"
 MvAbsoluteExecute:="AxCtrl1_MvAbsExecute"
 PositionDistance :="AxCtrl1_PositionDistance"
 Velocity          :="AxCtrl1_Velocity"
 Acceleration      :="AxCtrl1_Acceleration"
 Deceleration      :="AxCtrl1_Deceleration"
 JogPositive       :="AxCtrl1_JogPositive"
 JogNegative       :="AxCtrl1_JogNegative"
 JogVelocity       :="AxCtrl1_JogVelocity"
 JogAcceleration   :="AxCtrl1_JogAcceleration"
 JogDeceleration   :="AxCtrl1_JogDeceleration"
 AxisReady         :="AxCtrl1_AxisReady"
 AxisEnabled       :="AxCtrl1_AxisEnabled"
 AxisError         :="AxCtrl1_AxisError"
 AxisErrorID       :="AxCtrl1_AxisErrorID"
 DriveWarning      :="AxCtrl1_DriveWarning"
 DriveError        :="AxCtrl1_DriveError"
 DriveErrorID      :="AxCtrl1_DriveErrorID"
 IsHomed           :="AxCtrl1_IsHomed"
 ModeOfOperation   :="AxCtrl1_ModeOfOperation"
 PLCopenState      :="AxCtrl1_PLCopenState"
 ActualPosition    :="AxCtrl1_ActualPosition"
 ActualVelocity    :="AxCtrl1_ActualVelocity"
 CmdDone           :="AxCtrl1_CmdDone"
 CmdBusy           :="AxCtrl1_CmdBusy"
 CmdAborted        :="AxCtrl1_CmdAborted"
 CmdError          :="AxCtrl1_CmdError"
 CmdErrorID        :="AxCtrl1_CmdErrorID"
 DirectionPositive:="AxCtrl1_DirectionPos"
 DirectionNegative:="AxCtrl1_DirectionNeg"
 SWLimitMinActive :="AxCtrl1_SWLimitMinActive"
 SWLimitMaxActive :="AxCtrl1_SWLimitMaxActive"
 HWLimitMinActive :="AxCtrl1_HWLimitMinActive"
 HWLimitMaxActive :="AxCtrl1_HWLimitMaxActive"
 Axis              :="Axis01".Axis
```

For complex motion tasks, you can use the PLCopen blocks. Please specify the reference to the corresponding axis data at Axis in the axis DB.

Your project now includes the following blocks:

- OB 1 - Main
- OB 57 - DP Manufacturer Alarm
- OB 82 - I/O_FLT1
- OB 86 - Rack_FLT
- FB 860 - VMC_AxisControl with instance DB

- FB 886 - VMC_KernelInverter_EC with instance DB
- FB 887 - VMC_InitInverter_EC with instance DB
- UDT 860 - MC_Axis_REF
- UDT 886 - VMC_ConfigInverterEC_REF

Sequence of operations

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

   The transfer can only be done by the Siemens SIMATIC Manager - not hardware configurator!

   Since slave and module parameters are transmitted by means of SDO respectively SDO Init command, the configuration remains active, until a power cycle is performed or new parameters for the same SDO objects are transferred.

   With an overall reset the slave and module parameters are not reset!

   ⇨ 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. To do this, call the *Init* block FB 887 - VMC_InitInverter_EC with *Enable* = TRUE.

   ⇨ The output *Valid* returns TRUE. In the event of a fault, you can determine the error by evaluating the *ErrorID*.

   You have to call the *Init* block again if you load a new axis DB or you have changed parameters on the *Init* block.

   Do not continue until the Init block does not report any errors!

3. Ensure that the *Kernel* block FB 886 - VMC_KernelInverter_EC is cyclically called. In this way, control signals are transmitted to the drive and status messages are reported.
4. Program your application with the FB 860 - VMC_AxisControl or with the PLCopen blocks.

Controlling the drive via HMI

There is the possibility to control your drive via HMI. For this, a predefined symbol library is available for Movicon to access the VMC_AxisControl function block. *Chap. 10 'Controlling the drive via HMI' page 432*

8.6 Drive specific blocks

The PLCopen blocks for axis control can be found here: *Chap. 9 'Blocks for axis control' page 361*

8.6.1 UDT 886 - VMC_ConfigInverterEC_REF - inverter drive EtherCAT Data structure axis configuration

This is a user-defined data structure that contains information about the configuration data. The UDT is specially adapted to the use of an inverter drive, which is connected via EtherCAT.

8.6.2 FB 886 - VMC_KernelInverter_EC - inverter drive EtherCAT kernel

Description

This block converts the drive commands for an inverter drive via EtherCAT and communicates with the drive. For each inverter drive, an instance of this FB is to be cyclically called.

Please note that this module calls the SFB 238 internally.

In the SPEED7 Studio, this module is automatically inserted into your project.

In Siemens SIMATIC Manager, you have to copy the SFB 238 from the Motion Control Library into your project.

Parameter	Declaration	Data type	Description
Init	INPUT	BOOL	The block is internally reset with an edge 0-1. Existing motion commands are aborted and the block is initialized.
Config	IN_OUT	UDT 886	Data structure for transferring axis-dependent configuration data to the *AxisKernel*.
Axis	IN_OUT	UDT 860	Data structure for transferring axis-dependent information to the *AxisKernel* and PLCopen blocks.

8.6.3 FB 887 - VMC_InitInverter_EC - inverter drive EtherCAT initialization

Description

This block is used to configure the axis. The block is specially adapted to the use of an inverter drive, which is connected via EtherCAT.

Parameter	Declaration	Data type	Description
Enable	INPUT	BOOL	Release of initialization
LogicalAddress	INPUT	INT	Start address of the PDO input data
InputsStartAddressPDO	INPUT	INT	Start address of the input PDOs
OutputsStartAddressPDO	INPUT	INT	Start address of the output PDOs

Parameter	Declaration	Data type	Description
MaxVelocityDrive	INPUT	REAL	Maximum application speed [u].
MaxOutputFrequency	INPUT	REAL	Maximum output frequency [Hz]. Please transfer the value from the software tool *Drive Wizard+* here.
NumberOfPoles	INPUT	INT	Number of poles. Please transfer the value from the software tool *Drive Wizard+* here.
Valid	OUTPUT	BOOL	Initialization ■ TRUE: Initialization is valid.
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. 12 'ErrorID - Additional error information' page 457*
MaxVelocity	OUTPUT	INT	Maximum velocity in [rpm]. This value is determined automatically.
Config	IN_OUT	UDT 886	Data structure for transferring axis-dependent configuration data to the *AxisKernel*.
Axis	IN_OUT	UDT 860	Data structure for transferring axis-dependent information to the *AxisKernel* and PLCopen blocks.

9 Blocks for axis control

9.1 Overview

At Axis Control the blocks for programming motion tasks and status queries can be found. The following components can only be used to control the following drive systems.

- *Sigma-5 EtherCAT*
- *Sigma-7S EtherCAT*
- *Sigma-7W EtherCAT*
- *Sigma-5/7 PROFINET*
- *Inverter drive (inverter) via EtherCAT*

Please note that there are also restrictions here. The supported blocks can be found in the following table.

Simple motion tasks

Supported blocks	Sigma-5/7 PROFINET	Sigma-5/7 EtherCAT	Inverter EtherCAT	Page
UDT 860 - MC_AXIS_REF - data structure for axis	yes	yes	yes	*363*
FB 860 - VMC_AxisControl - control of drive functions and query of drive states	no	yes	yes	*363*

Complex motion tasks - PLCopen blocks

Supported blocks	Sigma-5/7 PROFINET	Sigma-5/7 EtherCAT	Inverter EtherCAT	Page
UDT 860 - MC_AXIS_REF - data structure for axis	yes	yes	yes	*367*
UDT 861 - MC_TRIGGER_REF - data structure	no	yes	no	*367*
FB 800 - MC_Power - enable respectively disable axis	no	yes	yes	*368*
FB 801 - MC_Home - home axis	no	yes	no	*370*
FB 802 - MC_Stop - stop axis	no	yes	yes	*372*
FB 803 - MC_Halt - stop axis	no	yes	yes	*374*
FB 804 - MC_MoveRelative - move axis relative	no	yes	no	*376*
FB 805 - MC_MoveVelocity - drive axis with constant velocity	no	yes	yes	*378*
FB 808 - MoveAbsolute - move axis to absolute position	no	yes	no	*380*
FB 811 - MC_Reset - reset axis	no	yes	yes	*382*
FB 812 - MC_ReadStatus - read PLCopen-State of the axis	no	yes	yes	*384*
FB 813 - MC_ReadAxisError - read axis error	no	yes	yes	*386*
FB 814 - MC_ReadParameter - read parameter data from axis	yes	yes	yes	*388*
FB 815 - MC_WriteParameter - write parameter data to axis	yes	yes	yes	*390*
FB 816 - MC_ReadActualPosition - read the current position of the axis	no	yes	no	*392*
FB 817 - MC_ReadActualVelocity - read the current velocity of the axis	no	yes	yes	*393*
FB 818 - MC_ReadAxisInfo - read axis additional information	no	yes	yes	*394*

Supported blocks	Sigma-5/7 PROFINET	Sigma-5/7 EtherCAT	Inverter EtherCAT	Page
FB 819 - MC_ReadMotionState - read state motion job	no	yes	yes	396
FB 823 - MC_TouchProbe - touch probe	yes	yes	no	398
FB 824 - MC_AbortTrigger - abort touch probe	yes	yes	no	400
FB 825 - MC_ReadBoolParameter - read boolean parameter from axis	yes	yes	yes	401
FB 826 - MC_WriteBoolParameter - write boolean parameter to axis	yes	yes	yes	403
FB 827 - VMC_ReadDWordParameter - read double-word parameter from axis	yes	yes	yes	405
FB 828 - VMC_WriteDWordParameter - write double-word parameter to axis	yes	yes	yes	407
FB 829 - VMC_ReadDWordParameter - read word parameter from axis	yes	yes	yes	409
FB 830 - VMC_WriteDWordParameter - write word parameter to axis	yes	yes	yes	411
FB 831 - VMC_ReadByteParameter - read byte parameter from axis	yes	yes	yes	413
FB 832 - MC_WriteParameter - write byte parameter to axis	yes	yes	yes	415
FB 833 - VMC_ReadDriveParameter - read drive parameter from drive	yes	yes	yes	417
FB 834 - VMC_WriteParameter - write drive parameter to drive	yes	yes	yes	419
FB 835 - VMC_HomeInit_LimitSwitch - initialization of homing on limit switch	yes	yes	no	421
FB 836 - VMC_HomeInit_HomeSwitch - initialization of homing on home switch	yes	yes	no	423
FB 837 - VMC_HomeInit_ZeroPulse - initialization of homing on zero pulse	yes	yes	no	426
FB 838 - VMC_HomeInit_SetPosition - initialization of homing mode set position	yes	yes	no	428

9.2 Simple motion tasks

9.2.1 UDT 860 - MC_AXIS_REF - Data structure axis data

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

9.2.2 FB 860 - VMC_AxisControl - Control block axis control

Description

With the FB *VMC_AxisControl* you can control the connected axis. You can check the status of the drive, turn the drive on or off, or execute various motion commands. A separate memory area is located in the instance data of the block. You can control your axis by means of an HMI. *Chap. 10 'Controlling the drive via HMI' page 432*

> *The VMC_AxisControl block should never be used simultaneously with the PLCopen module MC_Power. Since the VMC_AxisControl contains functionalities of the MC_Power and the latest command from the VMC_Kernel module is always executed, this can lead to a faulty behavior of the drive.*

Parameter

Parameter	Declaration	Data type	Description
AxisEnable	INPUT	BOOL	■ Enable/disable axis – TRUE: The axis is enabled. – FALSE: The axis is disabled.
AxisReset	INPUT	BOOL	■ Reset axis – Edge 0-1: Axis reset is performed.
HomeExecute	INPUT	BOOL	■ Homing – Edge 0-1: Homing is started.
HomePosition	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 application unit.
StopExecute	INPUT	BOOL	■ Stop axis – Edge 0-1: Stopping of the axis is started.
MvVelocityExecute	INPUT	BOOL	■ Start moving the axis – Edge 0-1: The axis is accelerated / decelerated to the speed specified.
MvRelativeExecute	INPUT	BOOL	■ Start moving the axis – Edge 0-1: The relative positioning of the axis is started.
MvAbsoluteExecute	INPUT	BOOL	■ Start moving the axis – Edge 0-1: The absolute positioning of the axis is started.
Direction *	INPUT	BYTE	Mode for absolute positioning: ■ 0: shortest distance ■ 1: positive direction ■ 2: negative direction ■ 3: current direction
PositionDistance	INPUT	REAL	Absolute position or relative distance depending on the command in [user units].

Parameter	Declaration	Data type	Description
Velocity	INPUT	REAL	Velocity setting (signed value) in [user units / s].
Acceleration	INPUT	REAL	Acceleration in [user units / s^2].
Deceleration	INPUT	REAL	Deceleration in [user units / s^2].
JogPositive	INPUT	BOOL	■ Drive axis with constant velocity in positive direction – Edge 0-1: Drive axis with constant velocity is started. – Edge 1-0: The axis is stopped.
JogNegative	INPUT	BOOL	■ Drive axis with constant velocity in negative direction – Edge 0-1: Drive axis with constant velocity is started. – Edge 1-0: The axis is stopped.
JogVelocity	INPUT	REAL	Speed setting for jogging (positive value) in [user units / s].
JogAcceleration	INPUT	REAL	Acceleration in [user units / s^2].
JogDeceleration	INPUT	REAL	Delay for jogging in [user units / s^2].
AxisReady	OUTPUT	BOOL	■ AxisReady – TRUE: The axis is ready to switch on. – FALSE: The axis is not ready to switch on. → Check and fix AxisError (see *AxisErrorID*). → Check and fix DriveError (see *DriveErrorID*). → Check initialization FB (input and output addresses or PDO mapping correct?)
AxisEnabled	OUTPUT	BOOL	■ Status axis – TRUE: Axis is switched on and accepts motion commands. – FALSE: Axis is not switched on and does not accepts motion commands.
AxisError	OUTPUT	BOOL	■ Motion axis error – TRUE: An error has occurred. Additional error information can be found in the parameter *AxisErrorID*. → The axis is disabled.
AxisErrorID	OUTPUT	WORD	Additional error information *Chap. 12 'ErrorID - Additional error information' page 457*
DriveWarning	OUTPUT	BOOL	■ Warning – TRUE: There is a warning on the drive. Additional information can be found in the manufacturer's manual.
DriveError	OUTPUT	BOOL	■ Error on the drive – TRUE: An error has occurred. Additional error information can be found in the parameter *DriveErrorID*. → The axis is disabled.
DriveErrorID	OUTPUT	WORD	■ Error – TRUE: There is an error on the drive. Additional information can be found in the manufacturer's manual.

Parameter	Declaration	Data type	Description
IsHomed	OUTPUT	BOOL	■ Information axis: homed – TRUE: The axis is homed.
ModeOfOperation	OUTPUT	INT	Drive-specific mode. For further information see drive manual. Example *Sigma-5*: 0: No mode changed/no mode assigned 1: Profile Position mode 2: Reserved (keep last mode) 3: Profile Velocity mode 4: Torque Profile mode 6: Homing mode 7: Interpolated Position mode 8: Cyclic Sync Position mode 9: Cyclic Sync Velocity mode 10: Cyclic Sync Torque mode Other Reserved (keep last mode)
PLCopenState	OUTPUT	INT	Current PLCopenState: 1: Disabled 2: Standstill 3: Homing 4: Discrete Motion 5: Continous Motion 7: Stopping 8: Errorstop
ActualPosition	OUTPUT	REAL	Position of the axis in [user unit].
ActualVelocity	OUTPUT	REAL	Velocity of the axis in [user unit / s]
CmdDone	OUTPUT	BOOL	■ Status – TRUE: Job ended without error.
CmdBusy	OUTPUT	BOOL	■ Status – TRUE: Job is running.
CmdAborted	OUTPUT	BOOL	■ Status – TRUE: The job was aborted during processing by another job.
CmdError	OUTPUT	BOOL	■ Status – TRUE: An error has occurred. Additional error information can be found in the parameter *CmdErrorID*.
CmdErrorID	OUTPUT	WORD	Additional error information *Chap. 12 'ErrorID - Additional error information' page 457*
DirectionPositive	OUTPUT	BOOL	■ Status motion job: Position increasing – TRUE: The position of the axis is increasing

Parameter	Declaration	Data type	Description
DirectionNegative	OUTPUT	BOOL	■ Status motion job: Position decreasing – TRUE: The position of the axis is decreasing
SWLimitMinActive	OUTPUT	BOOL	■ Software limit switch – TRUE: Software Limit switch Minimum active (Minimum position in negative direction exceeded).
SWLimitMaxActive	OUTPUT	BOOL	■ Software limit switch – TRUE: Software limit switch Maximum active (Maximum position in positive direction exceeded).
HWLimitMinActive	OUTPUT	BOOL	■ Hardware limit switch – TRUE: Negative hardware limit switch active on the drive (NOT- Negative Overtravel).
HWLimitMaxActive	OUTPUT	BOOL	■ Hardware limit switch – TRUE: Positive hardware limit switch active on the drive (POT- Positive Overtravel).
Axis	IN_OUT	MC_AXIS_REF	Reference to the axis.
*) This parameter is not supported by all drives, e.g. *Sigma 5 via EtherCAT* does not support this parameter.			

9.3 Complex motion tasks - PLCopen blocks

9.3.1 UDT 860 - MC_AXIS_REF - Data structure axis data

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

9.3.2 UDT 861 - MC_TRIGGER_REF - Data structure trigger signal

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

9.3.3 FB 800 - MC_Power - enable/disable axis

Description

An overview of the drive systems, which can be controlled with this block can be found here: *↳ Chap. 9.1 'Overview' page 361*

With MC_Power an axis can be enabled or disabled.

Parameter

Parameter	Declaration	Data type	Description
Enable	INPUT	BOOL	■ Enable/disable axis – TRUE: The axis is enabled – FALSE: The axis is disabled
EnablePositive	INPUT	BOOL	Parameter is currently not supported; call with FALSE
EnableNegative	INPUT	BOOL	Parameter is currently not supported; call with FALSE
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. 12 'ErrorID - Additional error information' page 457*
Axis	IN_OUT	MC_AXIS_REF	Reference to the axis

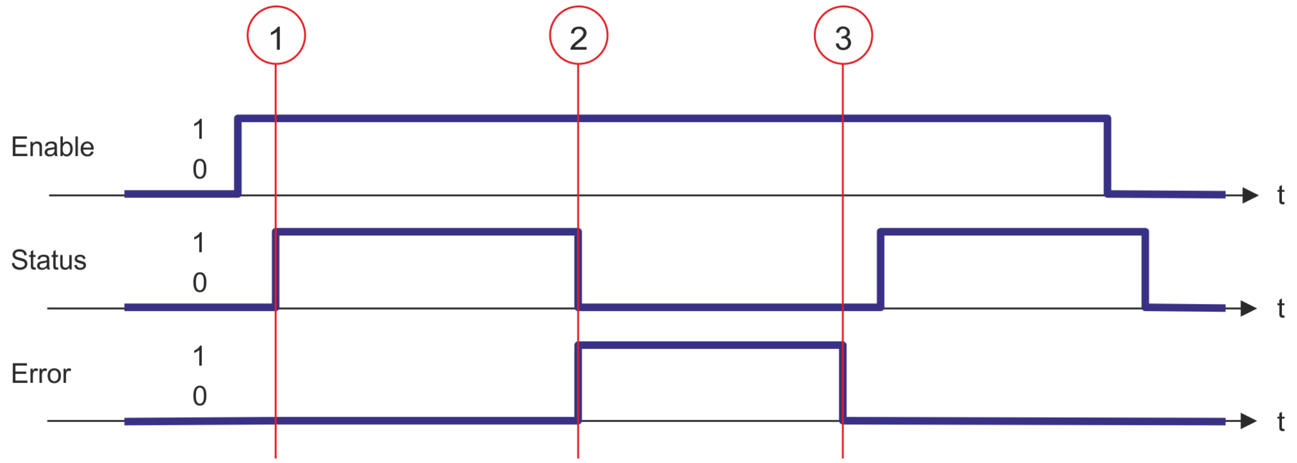
Enable axis

Call MC_Power with *Enable* = TRUE. If *Status* shows a value of 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 a value of 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 to disable 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.

9.3.4 FB 801 - MC_Home - home axis

Description

An overview of the drive systems, which can be controlled with this block can be found here: *Chap. 9.1 'Overview' page 361*

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 VMC_HomeInit_... blocks.

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 application unit.
BufferMode	INPUT	BYTE	Parameter is currently not supported; call with B#16#0
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. 12 'ErrorID - Additional error information' page 457*
Axis	IN_OUT	MC_AXIS_REF	Reference to the axis

PLCopen-State

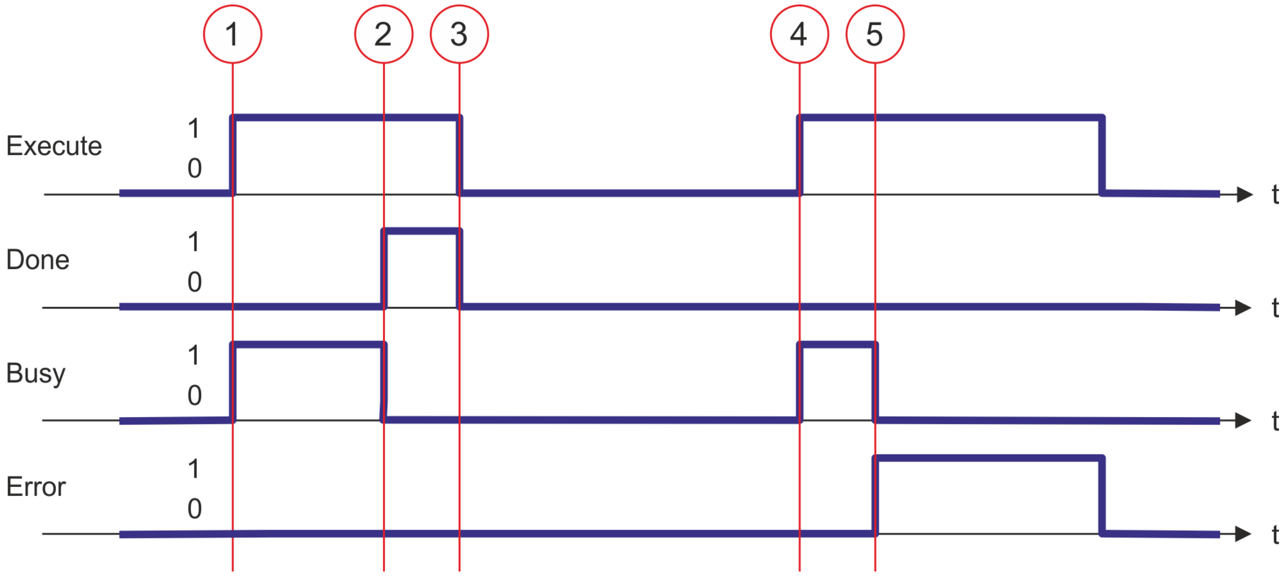
Start of the job only in the PLCopen-State *Standstill* possible.

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* den 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* den value TRUE.

9.3.5 FB 802 - MC_Stop - stop axis

Description

An overview of the drive systems, which can be controlled with this block can be found here: *Chap. 9.1 'Overview' page 361*

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]
Jerk	INPUT	REAL	Parameter is currently not supported; call with 0.0
Done	OUTPUT	BOOL	■ Status – TRUE: Job successfully done
Busy	OUTPUT	BOOL	■ Status – TRUE: Job is running
CommandA-borted	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. 12 'ErrorID - Additional error information' page 457*
Axis	IN_OUT	MC_AXIS_REF	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.

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

9.3.6 FB 803 - MC_Halt - holding axis

Description

An overview of the drive systems, which can be controlled with this block can be found here: *Chap. 9.1 'Overview' page 361*

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]
Jerk	INPUT	REAL	Parameter is currently not supported; call with 0.0
BufferMode	INPUT	BYTE	Parameter is currently not supported; call with B#16#0
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. 12 'ErrorID - Additional error information' page 457*
Axis	IN_OUT	MC_AXIS_REF	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*.

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 job continues until the axis stops even when Execute is set to FALSE.*
– *A running 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* den value TRUE.
(3) At the time (3) the job is completed and *Execute* becomes FALSE and thus each output parameter FALSE respectively 0.

9.3.7 FB 804 - MC_MoveRelative - move axis relative

Description

An overview of the drive systems, which can be controlled with this block can be found here: *Chap. 9.1 'Overview' page 361*

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
ContinuousUpdate	INPUT	BOOL	Parameter is currently not supported; call with FALSE
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]
Jerk	INPUT	REAL	Parameter is currently not supported; call with 0.0
BufferMode	INPUT	BYTE	Parameter is currently not supported; call with B#16#0
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. 12 'ErrorID - Additional error information' page 457*
Axis	IN_OUT	MC_AXIS_REF	Reference to the axis

PLCopen-State

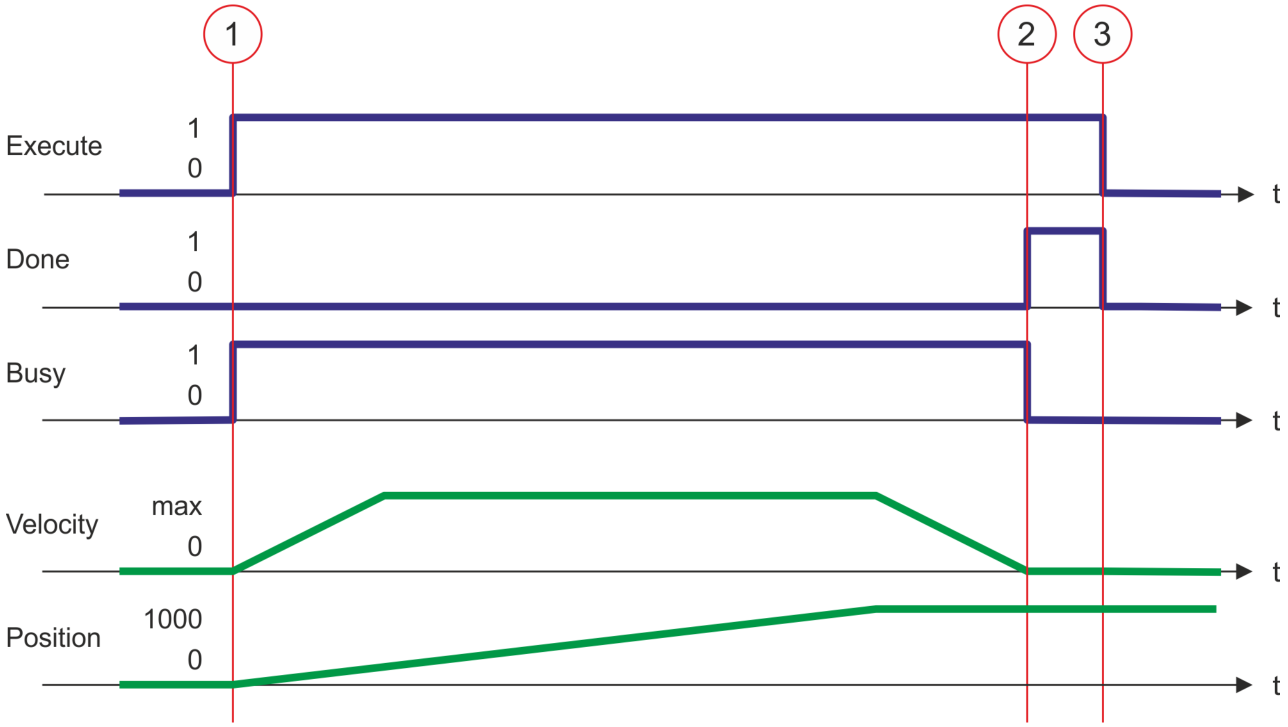
- Start of the job in the PLCopen-States *Standstill*, *Discrete Motion* and *Continuous Motion* possible.
- MC_MoveRelative switches the axis to the PLCopen-State *Discrete Motion*.

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

9.3.8 FB 805 - MC_MoveVelocity - drive axis with constant velocity

Description

An overview of the drive systems, which can be controlled with this block can be found here: *Chap. 9.1 'Overview' page 361*

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
ContinuousUp-date	INPUT	BOOL	Parameter is currently not supported; call with FALSE
Velocity	INPUT	REAL	Velocity setting (signed value) in [user units/s]
Acceleration	INPUT	REAL	Acceleration in [user units/s^2]
Deceleration	INPUT	REAL	Delay in breaking in [user units/s^2]
Jerk	INPUT	REAL	Parameter is currently not supported; call with 0.0
BufferMode	INPUT	BYTE	Parameter is currently not supported; call with B#16#0
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
CommandA-borted	OUTPUT	BOOL	■ Status – TRUE: The job was aborted during processing by another job
Error	OUTPUT	BOOL	■ Status – TRUE: An error has occurred. Additional error infor-mation can be found in the parameter *ErrorID*.
ErrorID	OUTPUT	WORD	Additional error information *Chap. 12 'ErrorID - Additional error information' page 457*
Axis	IN_OUT	MC_AXIS_REF	Reference to the axis

PLCopen-State

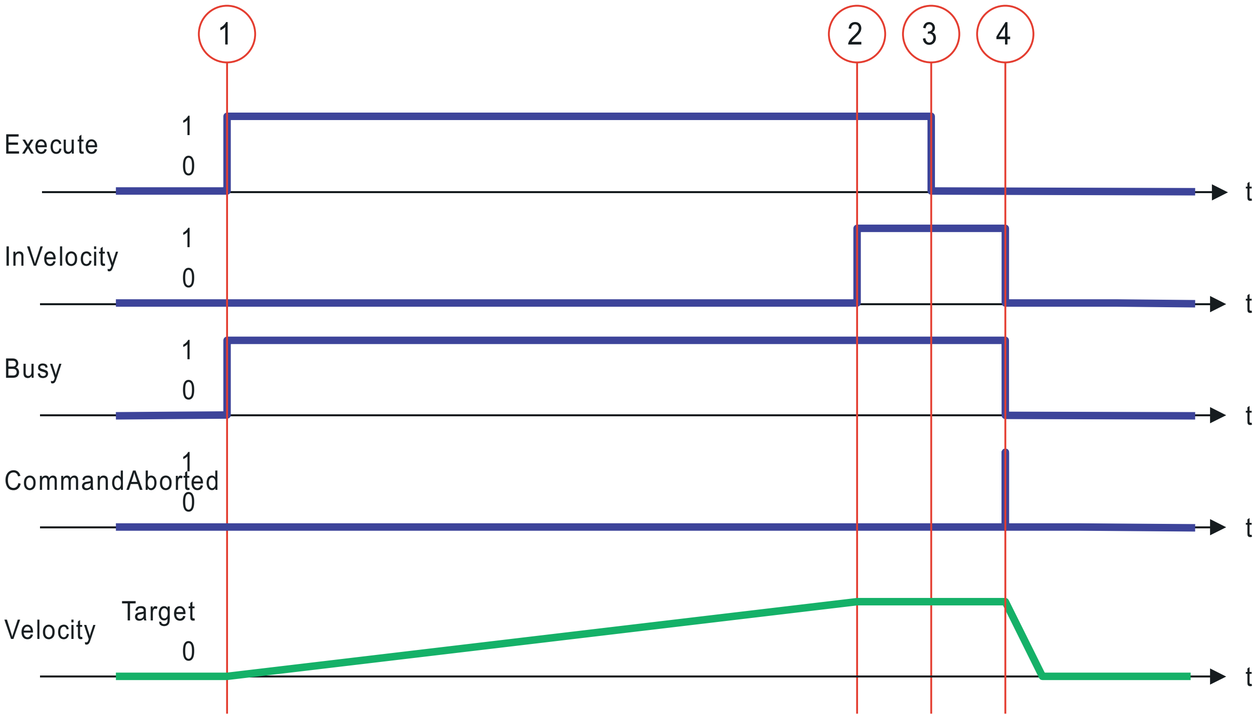
- Start of the job in the PLCopen-States *Standstill*, *Discrete Motion* and *Continuous Motion* possible.
- MC_MoveVelocity switches the axis to the PLCopen-State *Continuous Motion*.

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 *InVelocity* has 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_Halt job. The axis is decelerated to stop and *Busy* has the value FALSE.

9.3.9 FB 808 - MC_MoveAbsolute - move axis to absolute position

Description

An overview of the drive systems, which can be controlled with this block can be found here: *Chap. 9.1 'Overview' page 361*

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
ContinuousUp-date	INPUT	BOOL	Parameter is currently not supported; call with FALSE
Position	INPUT	REAL	Absolute position in [user units]
Velocity	INPUT	REAL	Maximum velocity (needs not necessarily be reached) signed value in [user units/s]
Acceleration	INPUT	REAL	Acceleration in [user units/s^2]
Deceleration	INPUT	REAL	Delay in breaking in [user units/s^2]
Jerk	INPUT	REAL	Parameter is currently not supported; call with 0.0
Direction	INPUT	Byte	■ Direction – 0: Shortest way – 1: Positive direction – 2: Negative direction – 3: Current direction
BufferMode	INPUT	BYTE	Parameter is currently not supported; call with B#16#0
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
CommandA-borted	OUTPUT	BOOL	■ Status – TRUE: The job was aborted during processing by another job
Error	OUTPUT	BOOL	■ Status – TRUE: An error has occurred. Additional error infor-mation can be found in the parameter *ErrorID*.
ErrorID	OUTPUT	WORD	Additional error information *Chap. 12 'ErrorID - Additional error information' page 457*
Axis	IN_OUT	MC_AXIS_REF	Reference to the axis

PLCopen-State

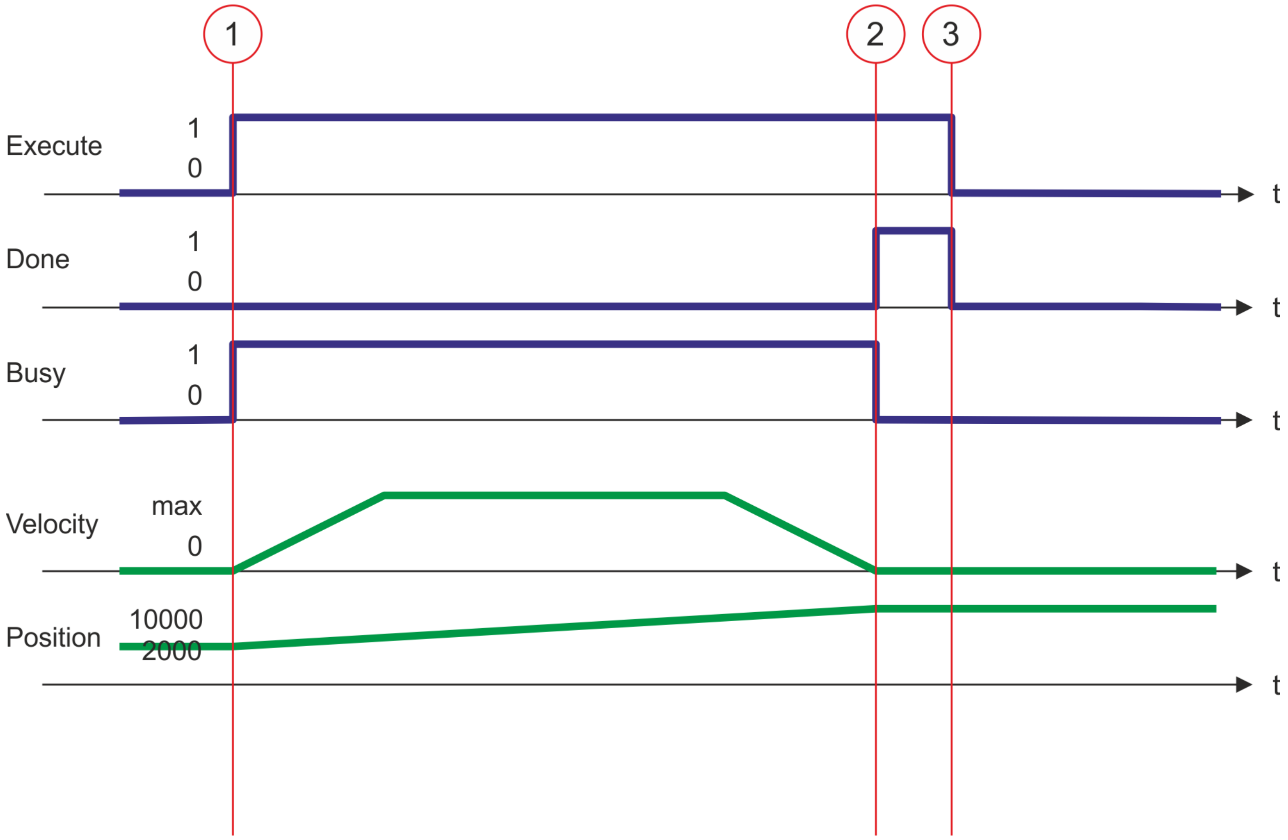
- Start of the job in the PLCopen-States *Standstill*, *Discrete Motion* and *Continuous Motion* possible.
- MC_MoveVelocity switches the axis to the PLCopen-State *Discrete Motion*.

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.

– *With Sigma-5 EtherCAT the target position is always reached via the shortest way.*
– *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* den value TRUE.
(3) At the time (3) the job is completed and *Execute* becomes FALSE and thus each output parameter FALSE respectively 0.

9.3.10 FB 811 - MC_Reset - reset axis

Description

An overview of the drive systems, which can be controlled with this block can be found here: *Chap. 9.1 'Overview' page 361*

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

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. 12 'ErrorID - Additional error information' page 457*
Axis	IN_OUT	MC_AXIS_REF	Reference to the axis

PLCopen-State

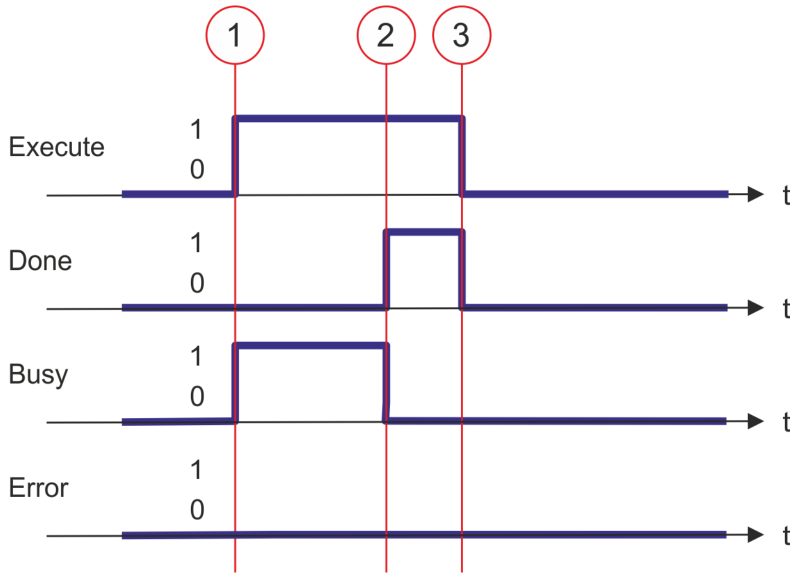
- Job start in PLCopen-State *ErrorStop* possible.
- MC_Reset switches the axis depending on MC_Power either to PLCopen-State *Standstill* (call MC_Power with *Enable* = TRUE) or *Disabled* (call MC_Power with *Enable* = FALSE).

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

9.3.11 FB 812 - MC_ReadStatus - PLCopen status

Description

An overview of the drive systems, which can be controlled with this block can be found here: *Chap. 9.1 'Overview' page 361*

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. 12 'ErrorID - Additional error information' page 457*
ErrorStop	OUTPUT	BOOL	■ Axis errors – TRUE: An axis error has occurred, move job can not be activated
Disabled	OUTPUT	BOOL	■ Status axis: Disabled – TRUE: Axis is disabled, move job can not be activated
Stopping	OUTPUT	BOOL	■ Status axis: Stop – TRUE: Axis is stopped (MC_Stop is active)
Homing	OUTPUT	BOOL	■ Status axis: Homing – TRUE: Axis is just homing (MC_Homing is active)
Standstill	OUTPUT	BOOL	■ Status move job – TRUE: No move job is active; a move job can be activated
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)
Axis	IN_OUT	MC_AXIS_REF	Reference to the slave axis

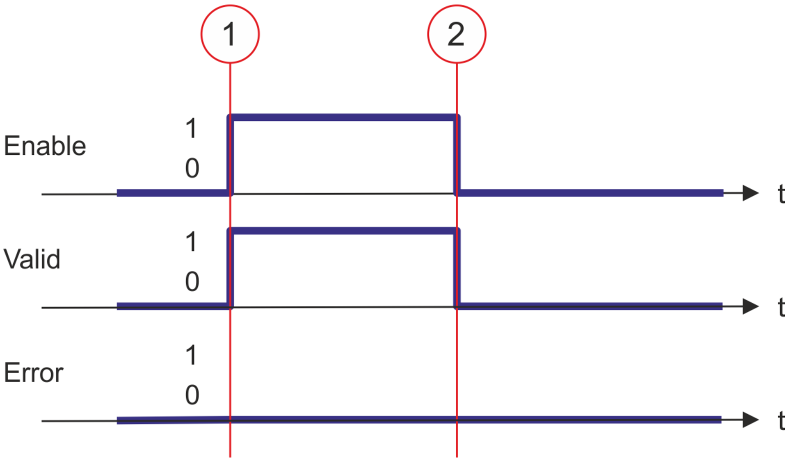
PLCopen-State

- Job start in each PLCopen-State possible.

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.

9.3.12 FB 813 - MC_ReadAxisError - read axis error

Description

An overview of the drive systems, which can be controlled with this block can be found here: *Chap. 9.1 'Overview' page 361*

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

Parameter

Parameter	Declaration	Data type	Description
Execute	INPUT	BOOL	■ Reset axis – Edge 0-1: Axis error is read.
Done	OUTPUT	BOOL	■ Status – TRUE: Job successfully done. Axis error 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. 12 'ErrorID - Additional error information' page 457*
AxisErrorID	OUTPUT	WORD	Axis error ID; the read value is vendor-specifically encoded.
Axis	IN_OUT	MC_AXIS_REF	Reference to the axis

PLCopen-State

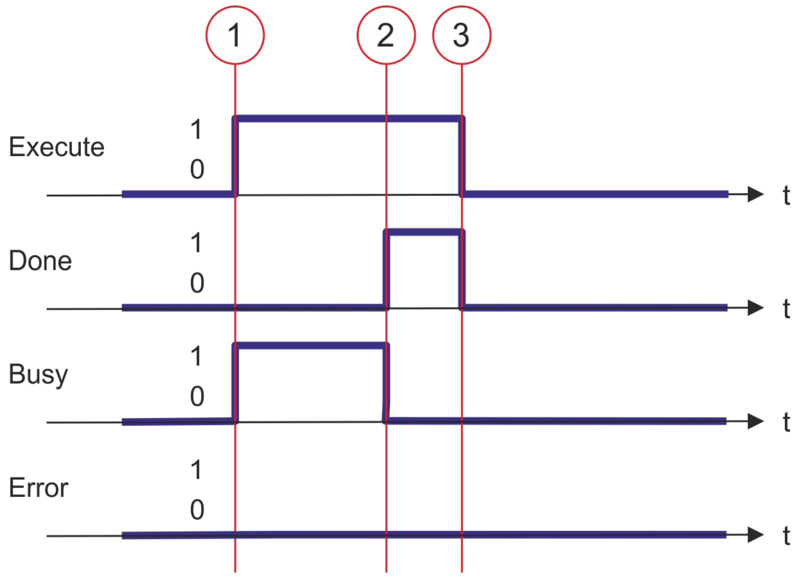
- Job start in each PLCopen-State possible.

Read error of the axis

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

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

9.3.13 FB 814 - MC_ReadParameter - read axis parameter data

Description

An overview of the drive systems, which can be controlled with this block can be found here: *Chap. 9.1 'Overview' page 361*

With MC_ReadParameter the parameter, that is defined by the parameter number, is read from the axis. *Chap. 9.3.35 'PLCopen parameter' page 429*

Parameter

Parameter	Declaration	Data type	Description
Execute	INPUT	BOOL	■ Read axis parameter data – Edge 0-1: The parameter data is read
Parameter Number	INPUT	INT	Number of the parameter to be read. *Chap. 9.3.35 'PLCopen parameter' page 429*
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. 12 'ErrorID - Additional error information' page 457*
Value	OUTPUT	REAL	Value of the read parameter
Axis	IN_OUT	MC_AXIS_REF	Reference to the axis

PLCopen-State

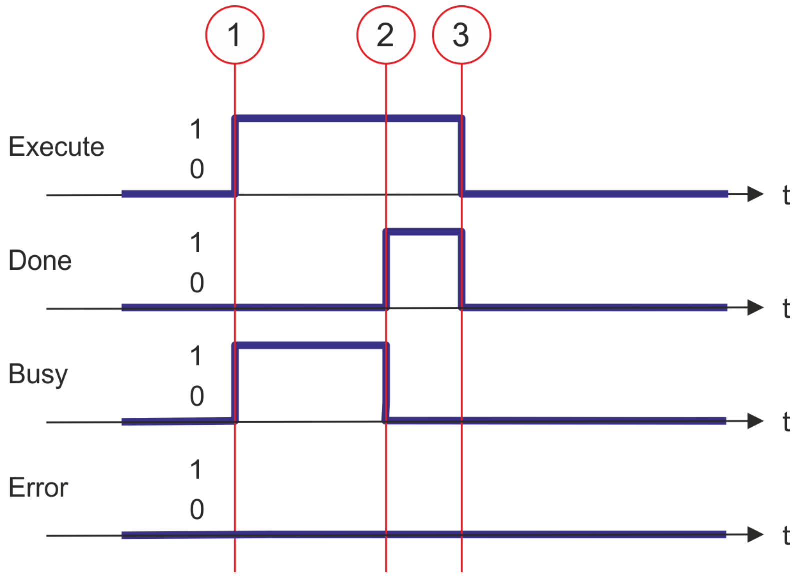
- Job start in each PLCopen-State possible.

Read axis parameter data

The reading of the axis 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.

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

9.3.14 FB 815 - MC_WriteParameter - write axis parameter data

Description

An overview of the drive systems, which can be controlled with this block can be found here: *Chap. 9.1 'Overview' page 361*

With MC_WriteParameter the value of the parameter, that is defined by the parameter number, is written to the axis. *Chap. 9.3.35 'PLCopen parameter' page 429*

Parameter

Parameter	Declaration	Data type	Description
Execute	INPUT	BOOL	■ Write axis parameter data – Edge 0-1: The parameter data is written
Parameter Number	INPUT	INT	Number of the parameter to be written. *Chap. 9.3.35 'PLCopen parameter' page 429*
Value	INPUT	REAL	Value of the written parameter
Done	OUTPUT	BOOL	■ Status – TRUE: Job successfully done. Parameter data was written
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. 12 'ErrorID - Additional error information' page 457*
Axis	IN_OUT	MC_AXIS_REF	Reference to the axis

PLCopen-State

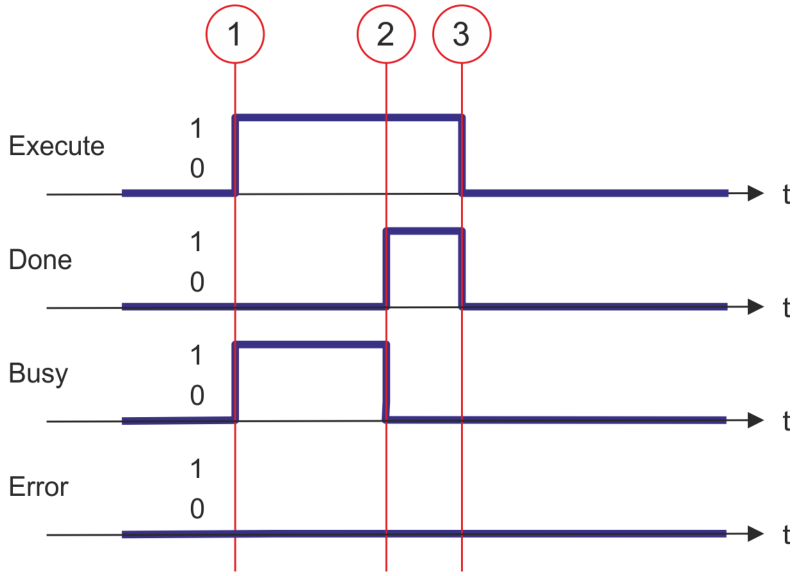
- Job start in each PLCopen-State possible.

Write axis parameter data

The writing of the axis 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.

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

9.3.15 FB 816 - MC_ReadActualPosition - reading current axis position

Description

An overview of the drive systems, which can be controlled with this block can be found here: *Chap. 9.1 'Overview' page 361*

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
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. 12 'ErrorID - Additional error information' page 457*
Position	OUTPUT	REAL	Position of the axis [user unit]
Axis	IN_OUT	MC_AXIS_REF	Reference to the axis

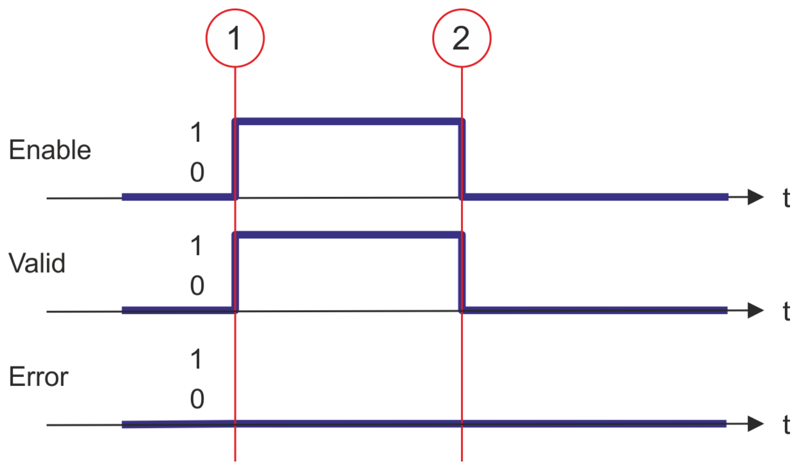
PLCopen-State

- Job start in each PLCopen-State possible.

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.

9.3.16 FB 817 - MC_ReadActualVelocity - read axis velocity

Description

An overview of the drive systems, which can be controlled with this block can be found here: *Chap. 9.1 'Overview' page 361*

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
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. 12 'ErrorID - Additional error information' page 457*
Velocity	OUTPUT	REAL	Velocity of the axis [user unit/s]
Axis	IN_OUT	MC_AXIS_REF	Reference to the axis

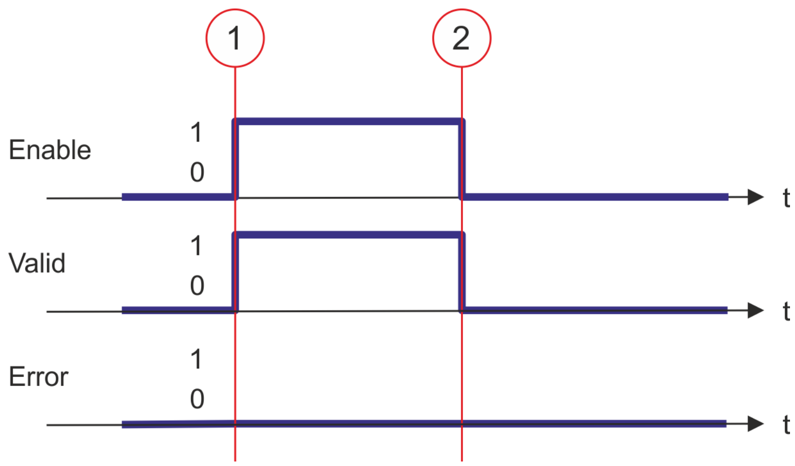
PLCopen-State

- Job start in each PLCopen-State possible.

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.

9.3.17 FB 818 - MC_ReadAxisInfo - read additional axis information

Description

An overview of the drive systems, which can be controlled with this block can be found here: *Chap. 9.1 'Overview' page 361*

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. 12 'ErrorID - Additional error information' page 457*
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
Communication-Ready	OUTPUT	BOOL	■ Information axis: Data exchange – TRUE: Data exchange with axis is initialized; axis is ready for communication
ReadyForPo-werOn	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
Axis	IN_OUT	MC_AXIS_REF	Reference to the axis

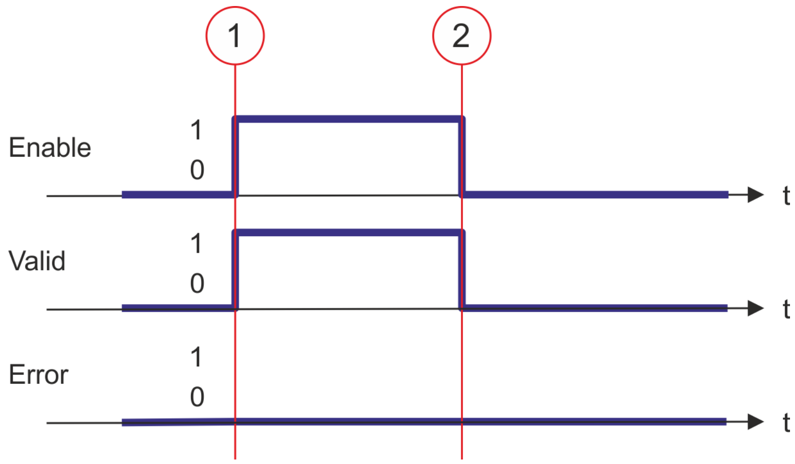
PLCopen-State

- Job start in each PLCopen-State possible.

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.

9.3.18 FB 819 - MC_ReadMotionState - read status motion job

Description

> *An overview of the drive systems, which can be controlled with this block can be found here:* *Chap. 9.1 'Overview' page 361*

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. 12 'ErrorID - Additional error information' page 457*
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	MC_AXIS_REF	Reference to the axis

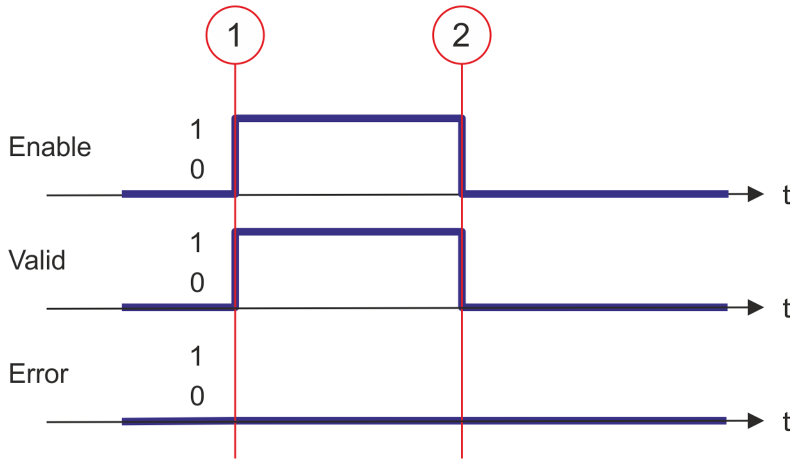
PLCopen-State

- Job start in each PLCopen-State possible.

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.

9.3.19 FB 823 - MC_TouchProbe - record axis position

Description

An overview of the drive systems, which can be controlled with this block can be found here: *Chap. 9.1 'Overview' page 361*

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.
CommandA-borted	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. 12 'ErrorID - Additional error information' page 457*
RecordedPosi-tion	OUTPUT	REAL	Recorded axis position where trigger event occurred [user units].
Axis	IN_OUT	MC_AXIS_REF	Reference to the axis.
TriggerInput	IN_OUT	MC_TRIGGER_REF	Reference to the trigger input. Structure ■ .Probe – 01: TouchProbe register 1 – 02: TouchProbe register 2 ■ .TriggerSource – 00: Input – 00: Encoder zero pulse ■ .Triggermode – 00: SingleTrigger (fix) ■ .Reserved (0 fix)

- *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. 11.3 'Behavior of the inputs and outputs' page 455*
- *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*.

9.3.20 FB 824 - MC_AbortTrigger - abort recording axis position

Description

An overview of the drive systems, which can be controlled with this block can be found here: *Chap. 9.1 'Overview' page 361*

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

Parameter

Parameter	Declaration	Data type	Description
Execute	INPUT	BOOL	The recording of the axis position is aborted with edge 0-1 at *Execute*.
Done	OUTPUT	BOOL	■ Status – TRUE: Job successfully done. The recording of the axis position was aborted.
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. 12 'ErrorID - Additional error information' page 457*
Axis	IN_OUT	MC_AXIS_REF	Reference to the axis.
TriggerInput	IN_OUT	MC_TRIGGER_REF	Reference to the trigger input. Structure ■ .Probe – 01: TouchProbe register 1 – 02: TouchProbe register 2 ■ .TriggerSource – 00: Input – 00: Encoder zero pulse ■ .Triggermode – 00: SingleTrigger (fix) ■ .Reserved (0 fix)

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.

9.3.21 FB 825 - MC_ReadBoolParameter - read axis boolean parameter data

Description

An overview of the drive systems, which can be controlled with this block can be found here: ⇨ Chap. 9.1 'Overview' page 361

With MC_ReadBoolParameter the parameter of data type BOOL, that is defined by the parameter number, is read from the axis. ⇨ *Chap. 9.3.35 'PLCopen parameter' page 429*

Parameter

Parameter	Declaration	Data type	Description
Execute	INPUT	BOOL	■ Read axis parameter data – Edge 0-1: The parameter data is read
Parameter Number	INPUT	INT	Number of the parameter to be read. ⇨ *Chap. 9.3.35 'PLCopen parameter' page 429*
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. 12 'ErrorID - Additional error information' page 457*
Value	OUTPUT	BOOL	Value of the read parameter
Axis	IN_OUT	MC_AXIS_REF	Reference to the axis

PLCopen-State

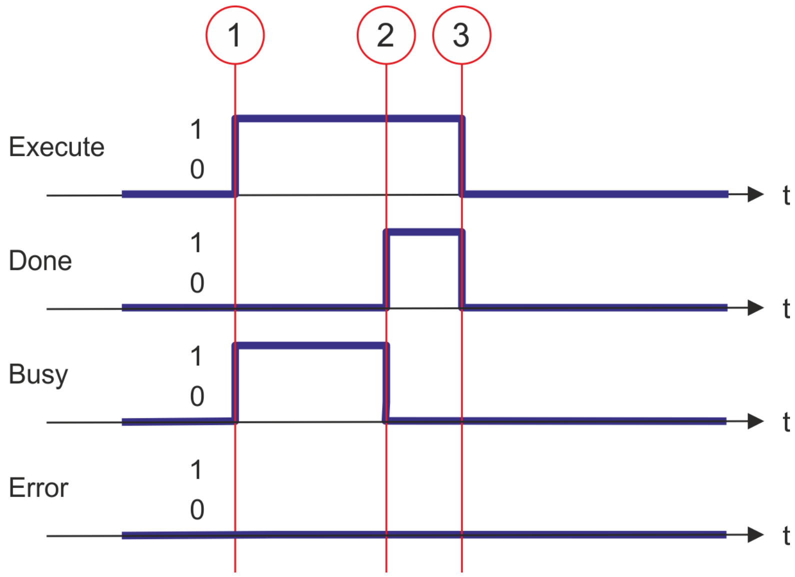
- Job start in each PLCopen-State possible.

Read axis parameter data

The reading of the axis 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.

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

9.3.22 FB 826 - MC_WriteBoolParameter - write axis boolean parameter data

Description

An overview of the drive systems, which can be controlled with this block can be found here: *Chap. 9.1 'Overview' page 361*

With MC_WriteBoolParameter the value of the parameter of data type BOOL, that is defined by the parameter number, is written to the axis. *Chap. 9.3.35 'PLCopen parameter' page 429*

Parameter

Parameter	Declaration	Data type	Description
Execute	INPUT	BOOL	■ Write axis parameter data – Edge 0-1: The parameter data is written
Parameter Number	INPUT	INT	Number of the parameter to be written. *Chap. 9.3.35 'PLCopen parameter' page 429*
Value	INPUT	BOOL	Value of the written parameter
Done	OUTPUT	BOOL	■ Status – TRUE: Job successfully done. Parameter data was written
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. 12 'ErrorID - Additional error information' page 457*
Axis	IN_OUT	MC_AXIS_REF	Reference to the axis

PLCopen-State

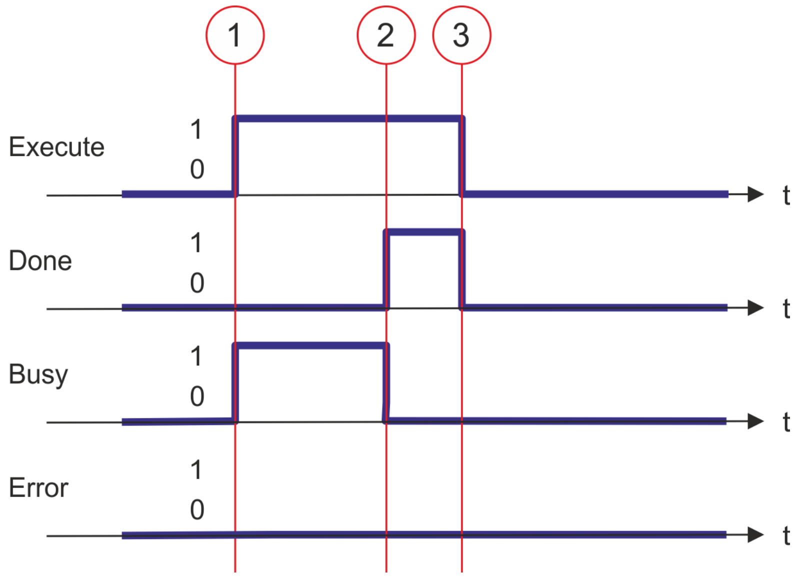
- Job start in each PLCopen-State possible.

Write axis parameter data

The writing of the axis 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.

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

9.3.23 FB 827 - VMC_ReadDWordParameter - read axis double word parameter data

Description

An overview of the drive systems, which can be controlled with this block can be found here: *Chap. 9.1 'Overview' page 361*

With MC_ReadDWordParameter the parameter of data type DWORD, that is defined by the parameter number, is read from the axis. *Chap. 9.3.35 'PLCopen parameter' page 429*

Parameter

Parameter	Declaration	Data type	Description
Execute	INPUT	BOOL	■ Read axis parameter data – Edge 0-1: The parameter data is read
Parameter-Number	INPUT	INT	Number of the parameter to be read. *Chap. 9.3.35 'PLCopen parameter' page 429*
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 information can be found in the parameter *ErrorID*.
ErrorID	OUTPUT	WORD	Additional error information *Chap. 12 'ErrorID - Additional error information' page 457*
Value	OUTPUT	DWORD	Value of the read parameter
Axis	IN_OUT	MC_AXIS_REF	Reference to the axis

PLCopen-State

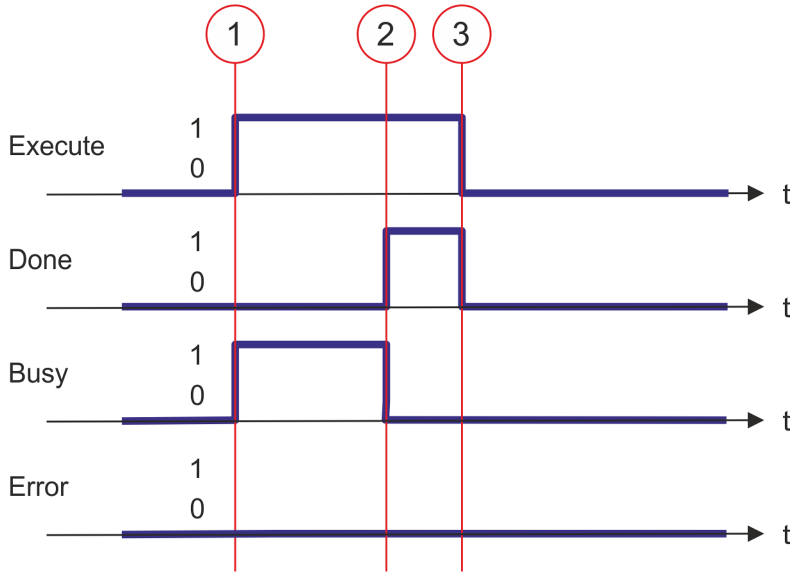
- Job start in each PLCopen-State possible.

Read axis parameter data

The reading of the axis 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.

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

9.3.24 FB 828 - VMC_WriteDWordParameter - write axis double word parameter data

Description

An overview of the drive systems, which can be controlled with this block can be found here: ⮫ Chap. 9.1 'Overview' page 361

With VMC_WriteDWordParameter the value of the parameter of data type DWORD, that is defined by the parameter number, is written to the axis. *⮫ Chap. 9.3.35 'PLCopen parameter' page 429*

Parameter

Parameter	Declaration	Data type	Description
Execute	INPUT	BOOL	■ Write axis parameter data – Edge 0-1: The parameter data is written
Parameter Number	INPUT	INT	Number of the parameter to be written. *⮫ Chap. 9.3.35 'PLCopen parameter' page 429*
Value	INPUT	DWORD	Value of the written parameter
Done	OUTPUT	BOOL	■ Status – TRUE: Job successfully done. Parameter data was written
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. 12 'ErrorID - Additional error information' page 457*
Axis	IN_OUT	MC_AXIS_REF	Reference to the axis

PLCopen-State

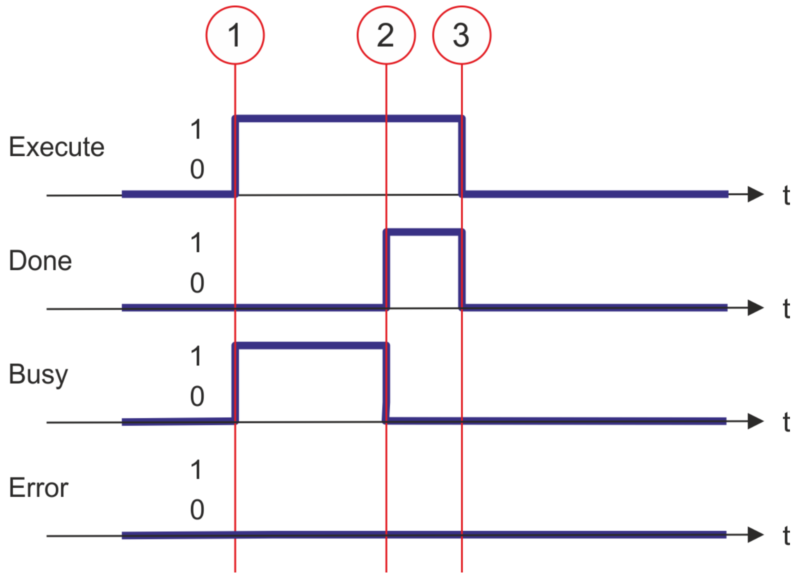
- Job start in each PLCopen-State possible.

Write axis parameter data

The writing of the axis 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.

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

9.3.25 FB 829 - VMC_ReadWordParameter - read axis word parameter data

Description

An overview of the drive systems, which can be controlled with this block can be found here: *Chap. 9.1 'Overview' page 361*

With VMC_ReadWordParameter the parameter of data type WORD, that is defined by the parameter number, is read from the axis. *Chap. 9.3.35 'PLCopen parameter' page 429*

Parameter

Parameter	Declaration	Data type	Description
Execute	INPUT	BOOL	■ Read axis parameter data – Edge 0-1: The parameter data is read
Parameter Number	INPUT	INT	Number of the parameter to be read. *Chap. 9.3.35 'PLCopen parameter' page 429*
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. 12 'ErrorID - Additional error information' page 457*
Value	OUTPUT	WORD	Value of the read parameter
Axis	IN_OUT	MC_AXIS_REF	Reference to the axis

PLCopen-State

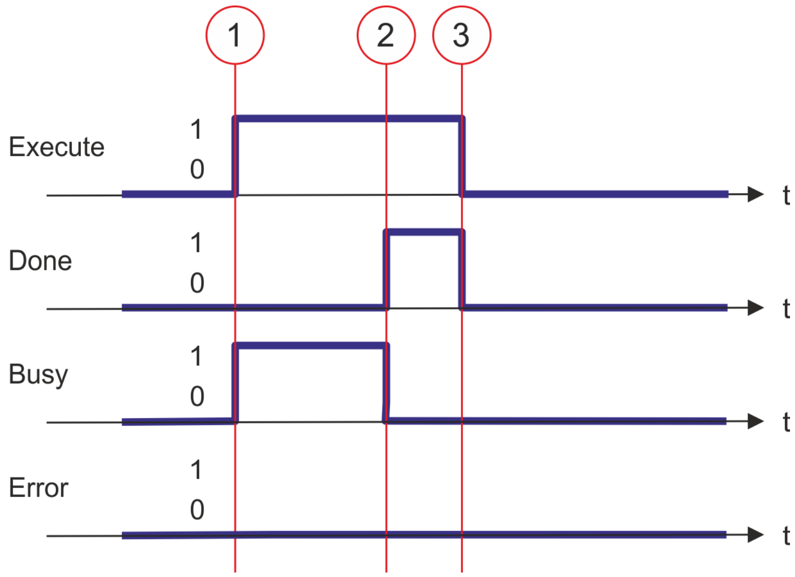
- Job start in each PLCopen-State possible.

Read axis parameter data

The reading of the axis 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.

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

9.3.26 FB 830 - VMC_WriteWordParameter - write axis word parameter data

Description

An overview of the drive systems, which can be controlled with this block can be found here: ⇨ Chap. 9.1 'Overview' page 361

With VMC_WriteWordParameter the value of the parameter of data type WORD, that is defined by the parameter number, is written to the axis. ⇨ *Chap. 9.3.35 'PLCopen parameter' page 429*

Parameter

Parameter	Declaration	Data type	Description
Execute	INPUT	BOOL	■ Write axis parameter data – Edge 0-1: The parameter data is written
Parameter Number	INPUT	INT	Number of the parameter to be written. ⇨ *Chap. 9.3.35 'PLCopen parameter' page 429*
Value	INPUT	WORD	Value of the written parameter
Done	OUTPUT	BOOL	■ Status – TRUE: Job successfully done. Parameter data was written
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. 12 'ErrorID - Additional error information' page 457*
Axis	IN_OUT	MC_AXIS_REF	Reference to the axis

PLCopen-State

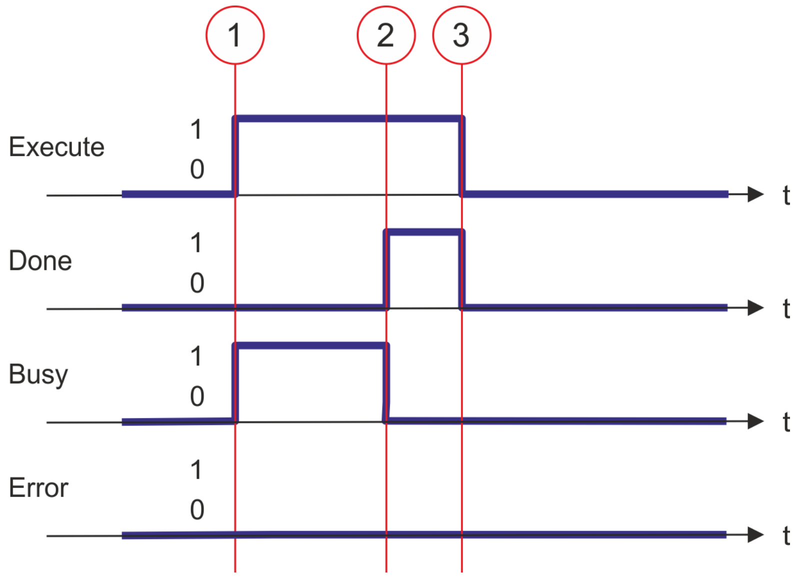
- Job start in each PLCopen-State possible.

Write axis parameter data

The writing of the axis 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.

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

9.3.27 FB 831 - VMC_ReadByteParameter - read axis byte parameter data

Description

An overview of the drive systems, which can be controlled with this block can be found here: *Chap. 9.1 'Overview' page 361*

With VMC_ReadByteParameter the parameter of data type BYTE, that is defined by the parameter number, is read from the axis. *Chap. 9.3.35 'PLCopen parameter' page 429*

Parameter

Parameter	Declaration	Data type	Description
Execute	INPUT	BOOL	■ Read axis parameter data – Edge 0-1: The parameter data is read
Parameter Number	INPUT	INT	Number of the parameter to be read. *Chap. 9.3.35 'PLCopen parameter' page 429*
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. 12 'ErrorID - Additional error information' page 457*
Value	OUTPUT	BYTE	Value of the read parameter
Axis	IN_OUT	MC_AXIS_REF	Reference to the axis

PLCopen-State

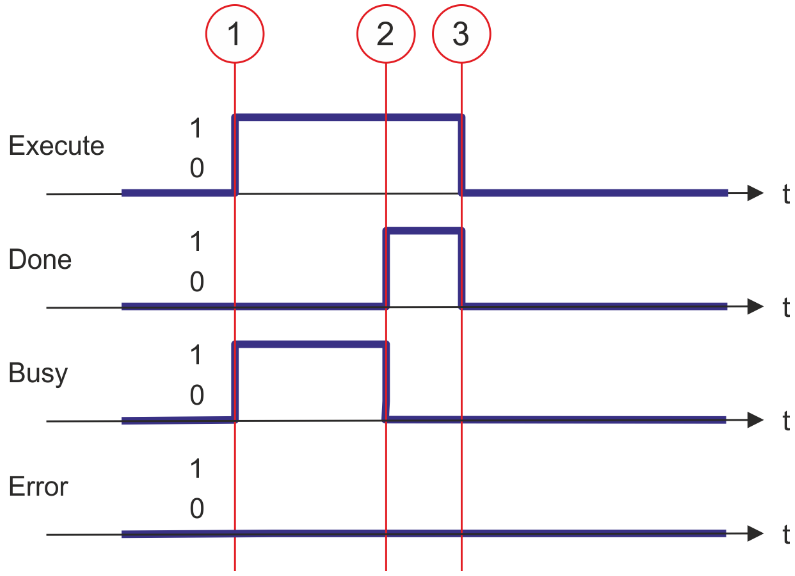
- Job start in each PLCopen-State possible.

Read axis parameter data

The reading of the axis 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.

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

9.3.28 FB 832 - VMC_WriteByteParameter - write axis byte parameter data

Description

An overview of the drive systems, which can be controlled with this block can be found here: *Chap. 9.1 'Overview' page 361*

With VMC_WriteByteParameter the value of the parameter of data type BYTE, that is defined by the parameter number, is written to the axis. *Chap. 9.3.35 'PLCopen parameter' page 429*

Parameter

Parameter	Declaration	Data type	Description
Execute	INPUT	BOOL	■ Write axis parameter data – Edge 0-1: The parameter data is written
Parameter Number	INPUT	INT	Number of the parameter to be written. *Chap. 9.3.35 'PLCopen parameter' page 429*
Value	INPUT	BYTE	Value of the written parameter
Done	OUTPUT	BOOL	■ Status – TRUE: Job successfully done. Parameter data was written
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. 12 'ErrorID - Additional error information' page 457*
Axis	IN_OUT	MC_AXIS_REF	Reference to the axis

PLCopen-State

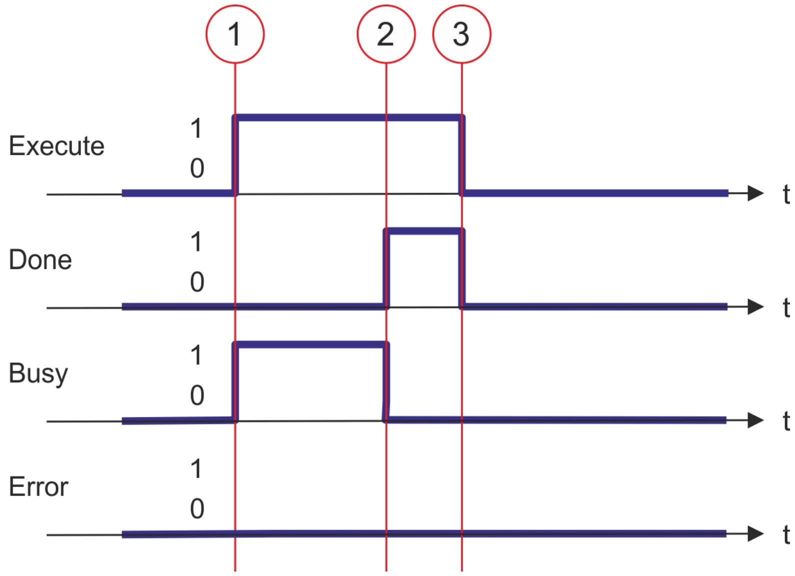
- Job start in each PLCopen-State possible.

Write axis parameter data

The writing of the axis 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.

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

9.3.29 FB 833 - VMC_ReadDriveParameter - read drive parameter

Description

An overview of the drive systems, which can be controlled with this block can be found here: *Chap. 9.1 'Overview' page 361*

With VMC_ReadDriveParameter 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
Length	INPUT	BYTE	Length of data ■ 1: BYTE ■ 2: WORD ■ 4: DWORD
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. 12 'ErrorID - Additional error information' page 457*
Value	OUTPUT	DWORD	Value of the read parameter
Axis	IN_OUT	MC_AXIS_REF	Reference to the axis

PLCopen-State

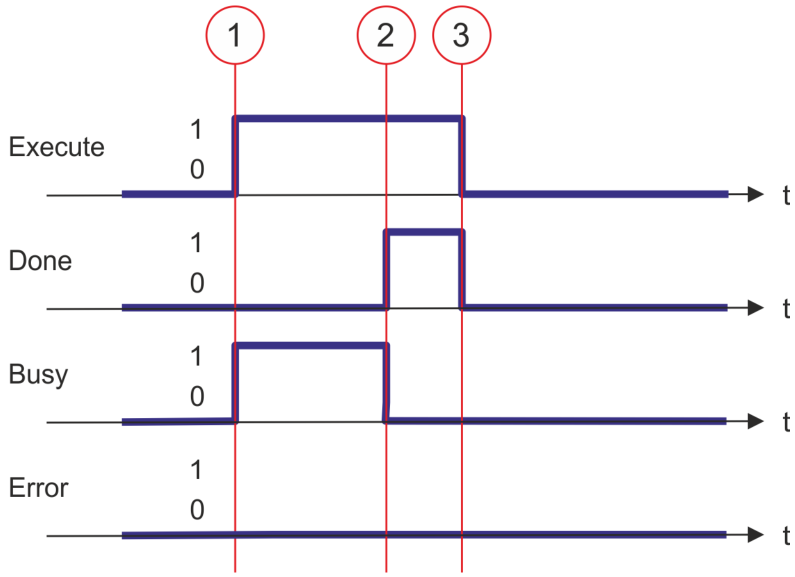
- Job start in each PLCopen-State possible.

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.

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

9.3.30 FB 834 - VMC_WriteDriveParameter - write drive parameter

Description

An overview of the drive systems, which can be controlled with this block can be found here: *Chap. 9.1 'Overview' page 361*

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

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
Length	INPUT	BYTE	Length of data ■ 1: BYTE ■ 2: WORD ■ 4: DWORD
Value	INPUT	DWORD	Value of the written parameter
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. 12 'ErrorID - Additional error information' page 457*
Axis	IN_OUT	MC_AXIS_REF	Reference to the axis

PLCopen-State

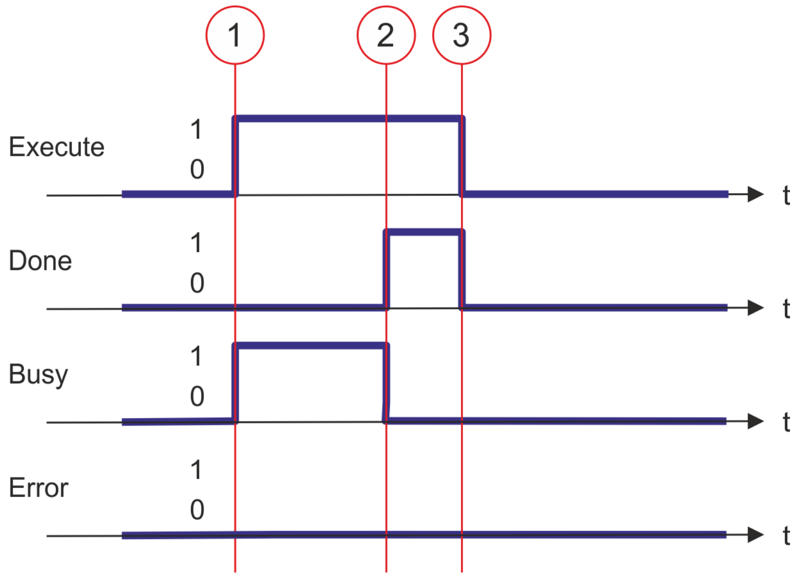
- Job start in each PLCopen-State possible.

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.

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

9.3.31 FB 835 - VMC_HomeInit_LimitSwitch - Initialisation of homing on limit switch

Description

An overview of the drive systems, which can be controlled with this block can be found here: *↳ Chap. 9.1 'Overview' page 361*

This block initialises homing on limit switch.

To use this block you must add the following blocks to your project:

- *↳ Chap. 9.3.24 'FB 828 - VMC_WriteDWordParameter - write axis double word parameter data' page 407*
- *↳ Chap. 9.3.28 'FB 832 - VMC_WriteByteParameter - write axis byte parameter data' page 415*

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
Velocity-SearchSwitch	INPUT	REAL	Velocity for search for the switch in [user units/s]
VelocitySearch-Zero	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. 12 'ErrorID - Additional error information' page 457*
AXIS	IN_OUT	MC_AXIS_REF	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 [UserUnits] > 0.0
   - VelocitySearchSwitch [InternalUnits] > 0
   - VelocitySearchSwitch [InternalUnits] ≤ VelocityMax
   - Input VelocitySearchZero [UserUnits] > 0.0
   - VelocitySearchZero [InternalUnits] > 0
   - VelocitySearchZero [InternalUnits] ≤ VelocityMax
   - Input Acceleration [UserUnits] > 0.0
   - Acceleration [InternalUnits] > 0
   - Acceleration [InternalUnits] ≤ AccelerationMax
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
1	false
2	true

9.3.32 FB 836 - VMC_HomeInit_HomeSwitch - Initialisation of homing on home switch

Description

An overview of the drive systems, which can be controlled with this block can be found here: *Chap. 9.1 'Overview' page 361*

This block initialises homing on home switch.

To use this block you must add the following blocks to your project:

- *Chap. 9.3.24 'FB 828 - VMC_WriteDWordParameter - write axis double word parameter data' page 407*
- *Chap. 9.3.28 'FB 832 - VMC_WriteByteParameter - write axis byte parameter data' page 415*

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.
InitialDirection	INPUT	BOOL	■ Initial 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
OnRisingEdge	INPUT	BOOL	■ Edge of home switch – TRUE: Edge 0-1 – FALSE: Edge 1-0
SameDirIndex-Pulse	INPUT	BOOL	■ Search 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
Velocity-SearchSwitch	INPUT	REAL	Velocity for search for the switch in [user units/s]
VelocitySearch-Zero	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*.

Parameter	Declaration	Data type	Description
ErrorID	OUTPUT	WORD	Additional error information *Chap. 12 'ErrorID - Additional error information' page 457*
AXIS	IN_OUT	MC_AXIS_REF	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 [UserUnits] > 0.0
   - VelocitySearchSwitch [InternalUnits] > 0
   - VelocitySearchSwitch [InternalUnits] ≤ VelocityMax
   - Input VelocitySearchZero [UserUnits] > 0.0
   - VelocitySearchZero [InternalUnits] > 0
   - VelocitySearchZero [InternalUnits] ≤ VelocityMax
   - Input Acceleration [UserUnits] > 0.0
   - Acceleration [InternalUnits] > 0
   - Acceleration [InternalUnits] ≤ AccelerationMax
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	InitialDirection	WithIndexPulse	OnRisingEdge	SameDirIndexPulse
7	positive	true	true	false
8	positive	true	true	true
9	positive	true	false	false
10	positive	true	false	true
11	negative	true	true	false
12	negative	true	true	true
13	negative	true	false	false
14	negative	true	false	true
24	positive	false	true	false
24	positive	false	true	true
24	positive	false	false	false

Homing Method	InitialDirection	WithIndexPulse	OnRisingEdge	SameDirIndexPulse
24	positive	false	false	true
28	negative	false	true	false
28	negative	false	true	true
28	negative	false	false	false
28	negative	false	false	true

9.3.33 FB 837 - VMC_HomeInit_ZeroPulse - Initialisation of homing on zero puls

Beschreibung

An overview of the drive systems, which can be controlled with this block can be found here: *Chap. 9.1 'Overview' page 361*

This block initialises homing on zero pulse.

To use this block you must add the following blocks to your project:

- *Chap. 9.3.24 'FB 828 - VMC_WriteDWordParameter - write axis double word parameter data' page 407*
- *Chap. 9.3.28 'FB 832 - VMC_WriteByteParameter - write axis byte parameter data' page 415*

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: Positive direction – FALSE: Negative direction
VelocitySearch-Zero	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. 12 'ErrorID - Additional error information' page 457*
AXIS	IN_OUT	MC_AXIS_REF	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 VelocitySearchZero [UserUnits] > 0.0
   - VelocitySearchZero [InternalUnits] > 0
   - VelocitySearchZero [InternalUnits] ≤ VelocityMax
   - Input Acceleration [UserUnits] > 0.0
   - Acceleration [InternalUnits] > 0
   - Acceleration [InternalUnits] ≤ AccelerationMax
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

9.3.34 FB 838 - VMC_HomeInit_SetPosition - Initialisation of homing mode set position

Description

> *An overview of the drive systems, which can be controlled with this block can be found here:* *Chap. 9.1 'Overview' page 361*

This block initialises homing on current position.

To use this block you must add the following block to your project:

- *Chap. 9.3.28 'FB 832 - VMC_WriteByteParameter - write axis byte parameter data' page 415*

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.
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. 12 'ErrorID - Additional error information' page 457*
AXIS	IN_OUT	MC_AXIS_REF	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

9.3.35 PLCopen parameter

PN	Name	Data type	R/W	Comments
1	CommandedPosition	REAL	R	Commanded position Access on: `#Axis.Status.Positioning.SetValues.CommandedPosition`
2	SWLimitPos	REAL	R/W	Positive software limit switch position Access on: `"Axis".AxisConfiguration.PositionLimits.MaxPosition`
3	SWLimitNeg	REAL	R/W	Negative software limit switch position Access on: `"Axis".AxisConfiguration.PositionLimits.MinPosition`
4	EnableLimitPos	BOOL	R/W	Enable positive software limit switch Access on: `"Axis".AxisConfiguration.PositionLimits.EnableMaxPos`
5	EnableLimitNeg	BOOL	R/W	Enable negative software limit switch Access on: `"Axis".AxisConfiguration.PositionLimits.EnableMinPos`
6	EnablePosLagMonitoring	BOOL	R/W	Enable monitoring of position lag Function is not supported
7	MaxPositionLag	REAL	R/W	Maximal position lag Function is not supported
8	MaxVelocitySystem	REAL	R	Maximal allowed velocity of the axis in the motion system This parameter is currently not supported
9	MaxVelocityAppl	REAL	R/W	Maximal allowed velocity of the axis in the application Access on: `#Axis.AxisConfiguration.DynamicLimits.MaxVelocityApp`
10	ActualVelocity	REAL	R	Actual velocity Access on: `#Axis.Status.Positioning.ActValues.Velocity`
11	CommandedVelocity	REAL	R	Commanded velocity Access on: `#Axis.Status.Positioning.SetValues.Velocity`
12	MaxAccelerationSystem	REAL	R	Maximal allowed acceleration of the axis in the motion system This parameter is currently not supported

PN	Name	Data type	R/W	Comments
13	MaxAccelerationAppl	REAL	R/W	Maximal allowed acceleration of the axis in the application Access on: `#Axis.AxisConfiguration.DynamicLimits.MaxAccelerationApp`
14	MaxDecelerationSystem	REAL	R	Maximal allowed deceleration of the axis in the motion system This parameter is currently not supported
15	MaxDecelerationAppl	REAL	R/W	Maximal allowed deceleration of the axis in the application Access on: `#Axis.AxisConfiguration.DynamicLimits.MaxDecelerationApp`
16	MaxJerkSystem	REAL	R	Maximum allowed jerk of the axis in the motion system This parameter is currently not supported
17	MaxJerkAppl	REAL	R/W	Maximum allowed jerk of the axis in the application This parameter is currently not supported.

9.3.36 VIPA-specific parameter

Positioning axis: Yaskawa *Sigma-5* / *Sigma-7* via EtherCAT

No.	Name	Data type	Index	Subindex	Access
900	HomingDone	BOOL	-	-	R/W [1, 2]
901	PositiveTorqueLimit	BOOL	-	-	R/W [1, 2]
902	NegativeTorqueLimit	BOOL	-	-	R/W [1, 2]
1000	ErrorCode	WORD	603F	0	R [3]
1001	HomeOffset	DWORD	607C	0	R/W [5, 6]
1002	HomingMethod	WORD	6098	0	R/W [3, 4]
1003	SpeedSearchSwitch	DWORD	6099	1	R/W [5, 6]
1004	SpeedSearchZero	DWORD	6099	2	R/W [5, 6]
1005	HomingAcceleration	DWORD	609A	0	R/W [5, 6]
1006	PositiveTorqueLimit	WORD	60E0	0	R/W [3, 4]
1007	NegativeTorqueLimit	WORD	0x60E1	0	R/W [3, 4]
1008	MotorRatedTorque	DWORD	0x6076	0	R/W [5, 6]

1) Access via *Chap. 9.3.21 'FB 825 - MC_ReadBoolParameter - read axis boolean parameter data' page 401*

2) Access via *Chap. 9.3.22 'FB 826 - MC_WriteBoolParameter - write axis boolean parameter data' page 403*

3) Access via *Chap. 9.3.25 'FB 829 - VMC_ReadWordParameter - read axis word parameter data' page 409*

4) Access via *Chap. 9.3.26 'FB 830 - VMC_WriteWordParameter - write axis word parameter data' page 411*

5) Access via *Chap. 9.3.23 'FB 827 - VMC_ReadDWordParameter - read axis double word parameter data' page 405*

6) Access via *Chap. 9.3.24 'FB 828 - VMC_WriteDWordParameter - write axis double word parameter data' page 407*

No.	Name	Data type	Index	Subindex	Access
1009	FollowingErrorWindow	DWORD	0x6065	0	R/W [5,6]
1010	FollowingErrorTimeOut	WORD	0x6066	0	R/W [3,4]
1011	PositionWindow	DWORD	0x6067	0	R/W [5,6]
1012	PositionTime	WORD	0x6068	0	R/W [3,4]
1013	Min Position Limit	DWORD	0x607D	1	R/W [5,6]
1014	Max Position Limit	DWORD	0x607D	2	R/W [5,6]
1015	Digital outputs/ physical outputs	DWORD	0x60FE	1	R/W [5,6]
1016	Digital outputs/ mask	DWORD	0x60FE	2	R/W [5,6]
1017	Quick stop deceleration	DWORD	0x6085	0	R/W [5,6]
1018	Forward external torque limit	WORD	0x2404	0	R/W [3,4]
1019	Reverse external torque limit	WORD	0x2405	0	R/W [3,4]

1) Access via ↳ ***Chap. 9.3.21 'FB 825 - MC_ReadBoolParameter - read axis boolean parameter data' page 401***

2) Access via ↳ ***Chap. 9.3.22 'FB 826 - MC_WriteBoolParameter - write axis boolean parameter data' page 403***

3) Access via ↳ ***Chap. 9.3.25 'FB 829 - VMC_ReadWordParameter - read axis word parameter data' page 409***

4) Access via ↳ ***Chap. 9.3.26 'FB 830 - VMC_WriteWordParameter - write axis word parameter data' page 411***

5) Access via ↳ ***Chap. 9.3.23 'FB 827 - VMC_ReadDWordParameter - read axis double word parameter data' page 405***

6) Access via ↳ ***Chap. 9.3.24 'FB 828 - VMC_WriteDWordParameter - write axis double word parameter data' page 407***

10 Controlling the drive via HMI

10.1 Overview

Drive control via an HMI is possible with the following library groups:

- *Sigma-5* EtherCAT ↳ *11*
- *Sigma-7S* EtherCAT ↳ *47*
- *Sigma-7W* EtherCAT ↳ *85*
- *Sigma-5/7* Pulse Train ↳ *220*

To control the corresponding drive via an HMI such as Touch Panel or Panel PC, there is a symbol library for Movicon. You can use the templates to control the corresponding VMC_AxisControl function block. The Symbol Library contains the following templates:

- Numeric Touchpad
  - This is an input field adapted to the VMC_AxisControl templates for different display resolutions.
  - You can use the touch pad instead of the default input field.
- VMC_AxisControl
  - Template for controlling the FB 860 - VMC_AxisControl function block in the CPU.
  - The template is available for different display resolutions.
- VMC_AxisControl ... Trend
  - Template for controlling the FB 860 - VMC_AxisControl function block in the CPU, which additionally shows the graphic trend of the drive.
  - The use of this template can affect the performance of the panel.
  - The template is available for different display resolutions.
- VMC_AxisControl_PT
  - Template for controlling the FB 875 - VMC_AxisControl_PT function block in the CPU, which drive is connected via Pulse Train.
  - The template is available for different display resolutions.

Please note that currently no ECO panels are supported!

Installation in Movicon

1. Go to the service area of www.vipa.com.
2. Download the *'Symbol library for Movicon'* from the download area at *'VIPA Lib'*.
3. Specify a target directory in which the blocks are to be stored and start the unzip process with [OK].
4. Open the library after unzipping and drag and drop the *Symbol library 'vipa simple motion control VX.X.X.msxz'* and the *Language table 'vipa simple motion control VX.X.X.CSV'* to the Movicon unser directory ...\Public\Documents\Progea\Movicon \Symbols.

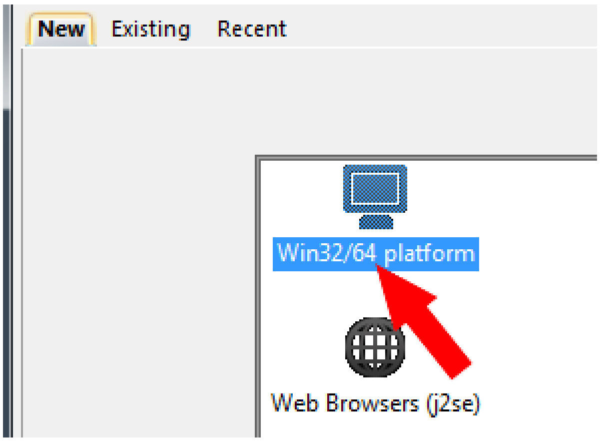
   ⇨ After restarting Movicon, the symbol library is available in Movicon via the *'Symbol libraries'*.

   In order for the texts of the templates to be displayed correctly, you must import the language table into your project. ↳ *'Import voice table' page 438*

10.2 Create a new project

Create a project

1. Start Movicon and open the project wizard via *'File ➔ New'*.
2. Select *'Win32/64 platform'* as target platform and click at [Open].

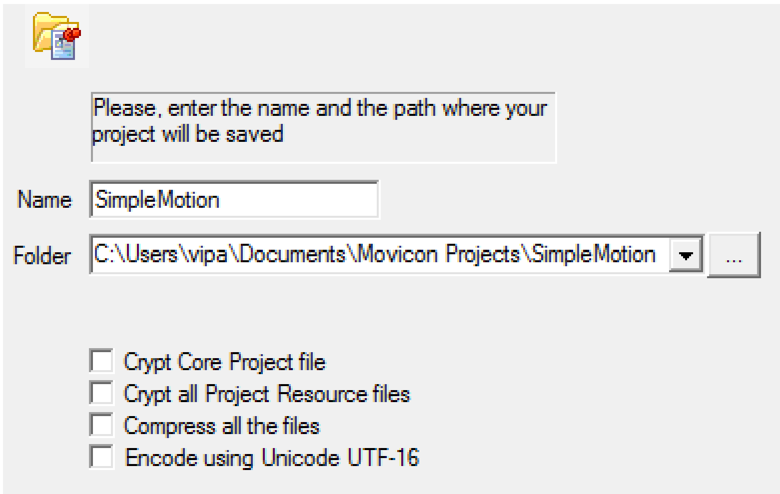


⇨ The dialog *'Device properties'* opens.

3. Specify a project name at *'Name'*.

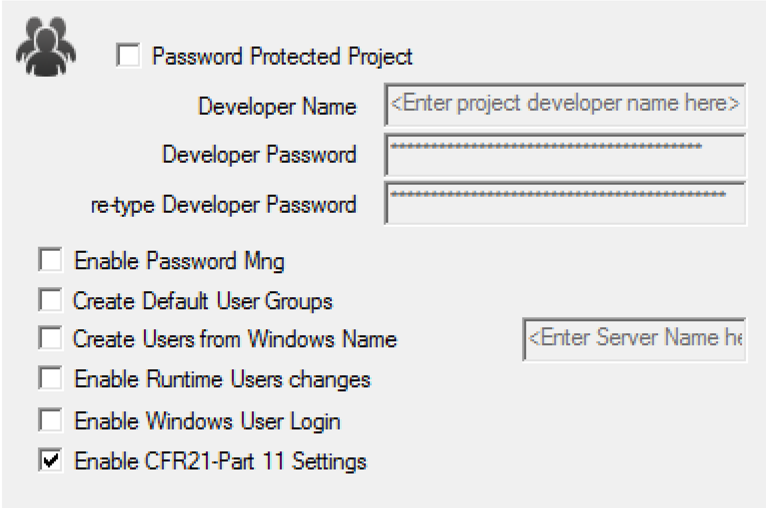
   Specify at *'Folder'* a storage area.

   Leave all settings disabled and click at [Next].



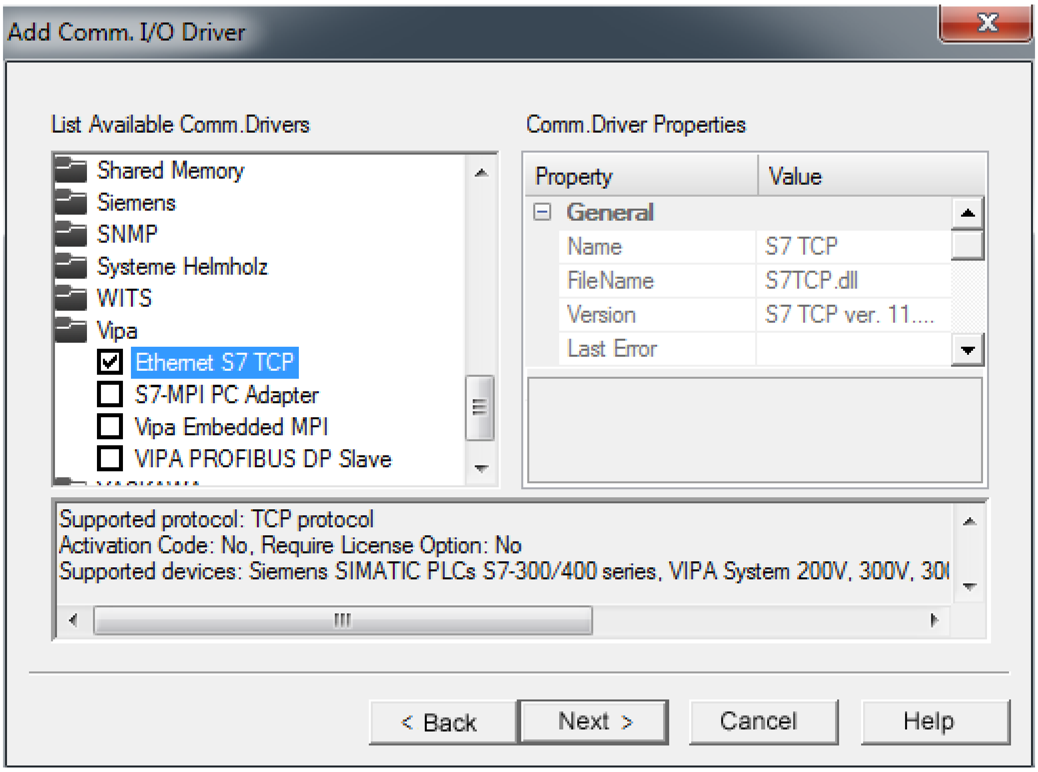
⇨ The dialog *'Users'* opens.

4. Make the appropriate user settings, if desired, or enable only *'CRF-21-Part...'* and click at [Next].



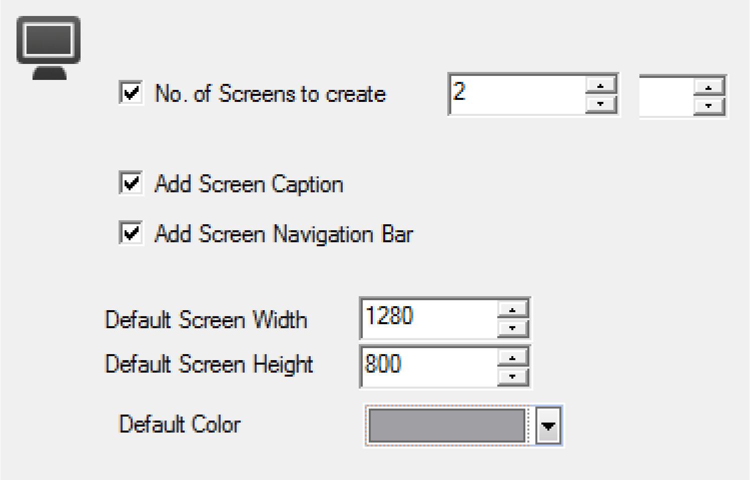
⇨ The dialog *'Add Comm. I/O Driver'* opens.

5. Since the connection to the CPU is via TCP/IP, enable in the *'List Available Comm.Drivers'* the driver *'VIPA'* > *'Ethernet S7 TCP'* and click at [Next].



⇨ The dialog *'Screens'* opens.

6. Enter 2 screens and their size, which matches your panel and click at [Next].



⇨ The dialog *'Data base settings (ODBC)'* opens.

7. If you want a database connection, you can make the corresponding settings here. Otherwise, click at [Next].

⇨ The dialog *'Data logger and recipe settings (ODBC)'* opens.

8. If templates are to be generated, you can make the corresponding settings here. Otherwise, click at [Next].

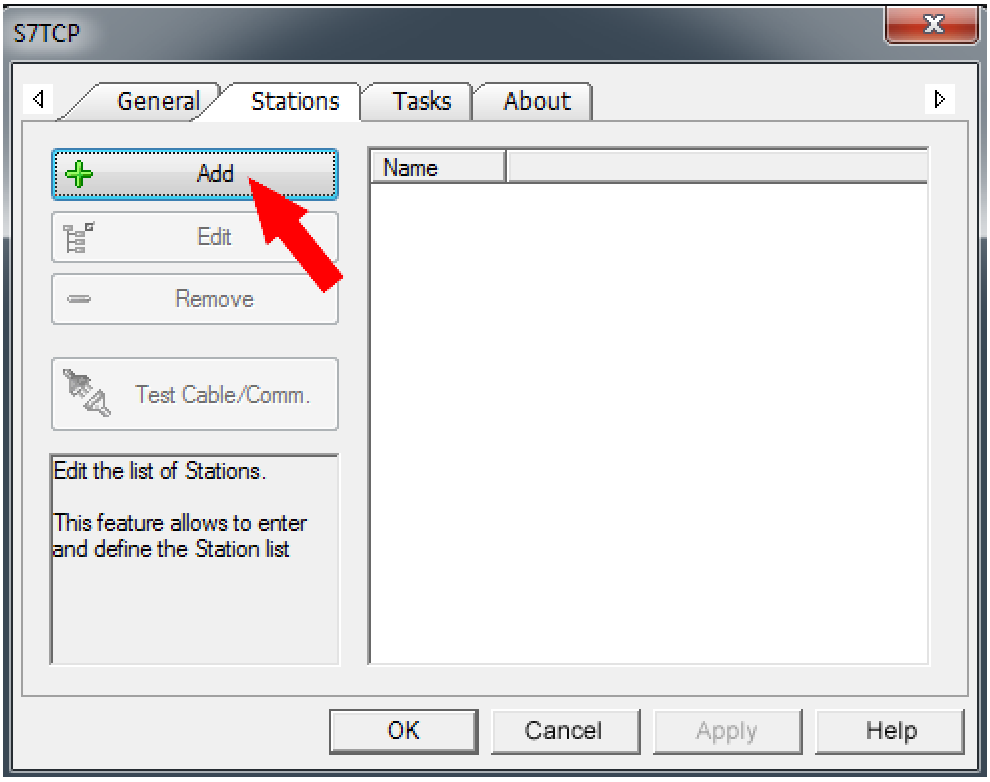
⇨ The dialog *'Alarm settings'* opens.

9. If alarms are to be generated, you can make the corresponding settings here. Otherwise, click at [Finish].

⇨ Your project is created with the settings you have made and the settings dialog for the *'S7TCP'* communication driver opens automatically.

10. Select the register *'Stations'*.

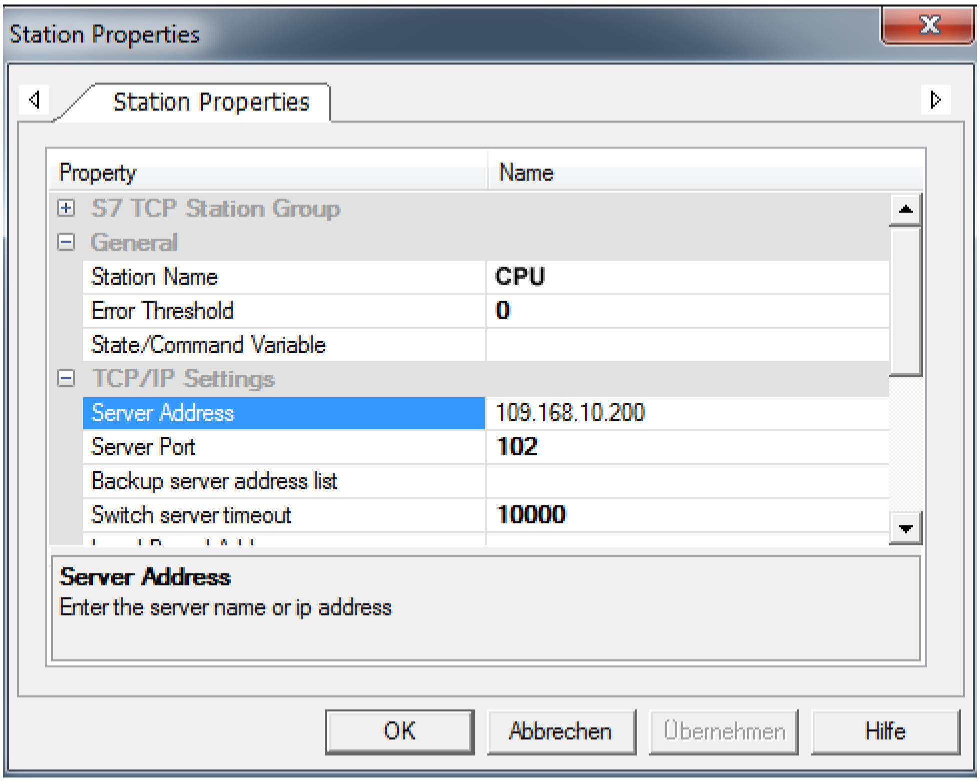
11. To add a new station, click [+ Add].



⇨ The dialog *'Station Properties'* opens.

12. Enter a station name at *'Station Name'*. You have to use this name for the screen in the initialization dialog further below. Allowed characters: *A-Z*, *a-z*, *0-9* space and the separators "_" and "-"

Enter at *'Server Address'* the IP address of your CPU and click at [OK].



13. Negate the query for importing variables from the PLC database and close the *'S7TCP'* dialog with [OK].

⇨ The project and the workspace are now enabled for use. In the project at *'Resourcen > SimpleMotion'* the standard elements were added by the following elements:

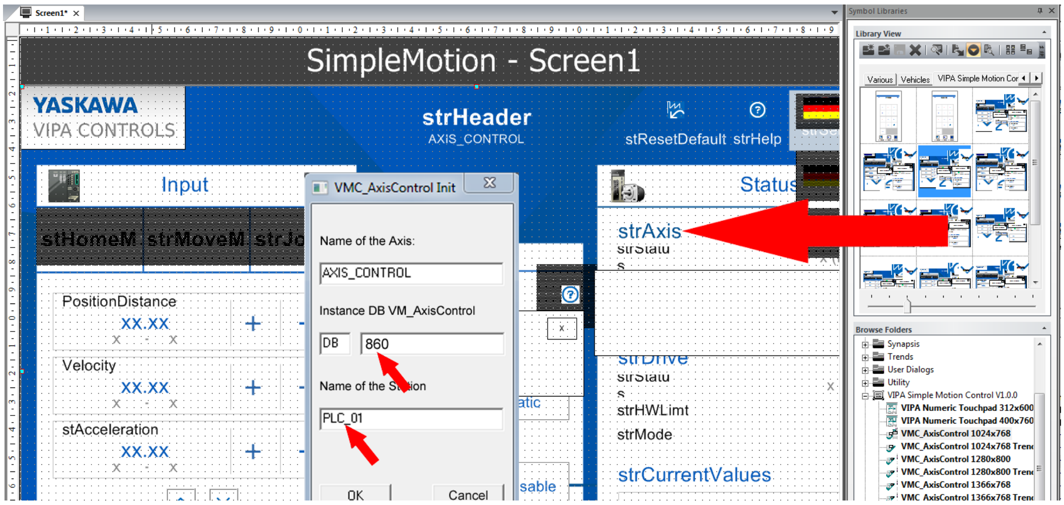
- Real Time DB
  - Comm.Drivers
    S7 TCP
- Screens
  - Screen1
  - Screen2
  - Footer Buttons

10.3 Modify the project in Movicon

Configuring the screen

1. Open via *'Resources > SimpleMotion >Screens' 'Screen1'*.

2. Navigate in *'Browse Folders'* at *'vipa simple motion control ...'* and drag & drop from the *'Library view'* the template to the *'Screen1'*, which matches the resolution of your panel.



⇨ The initialization dialog opens

3. Specify a name for the axis. Allowed characters: *A-Z*, *a-z*, *0-9*, space and the separators "_" and "-"

Specify the instance DB number that you use in your PLC program.

Specify the station name. This must match the *'Station Name'* from *'Station Properties'* of the *'S7 TCP'* communication settings. Allowed characters: *A-Z*, *a-z*, *0-9*, space and the separators "_" and "-"

⇨ With [OK] all variables as well as their structures are generated and the addresses are set to the specified destination address.

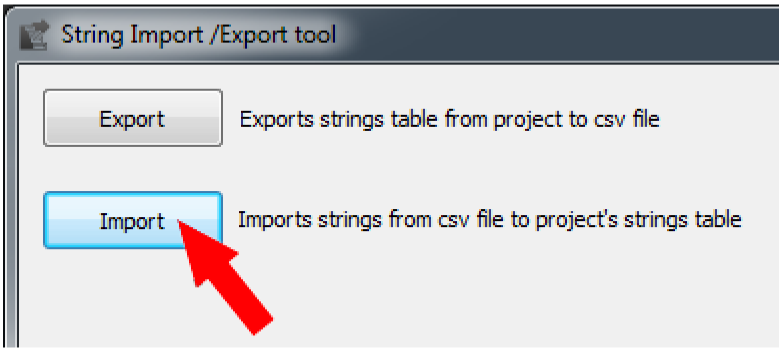
4. Place the template and adjust its size.

> *Variables are created for each template under the corresponding name. When deleting the template, the corresponding variables must be deleted again. You can select these at 'Resources > SimpleMotion > Real Time DB > Variables'. Delete these together with the higher-level directory. If no further templates access the 'Structure Prototypes' for the Axis control, these must also be deleted.*

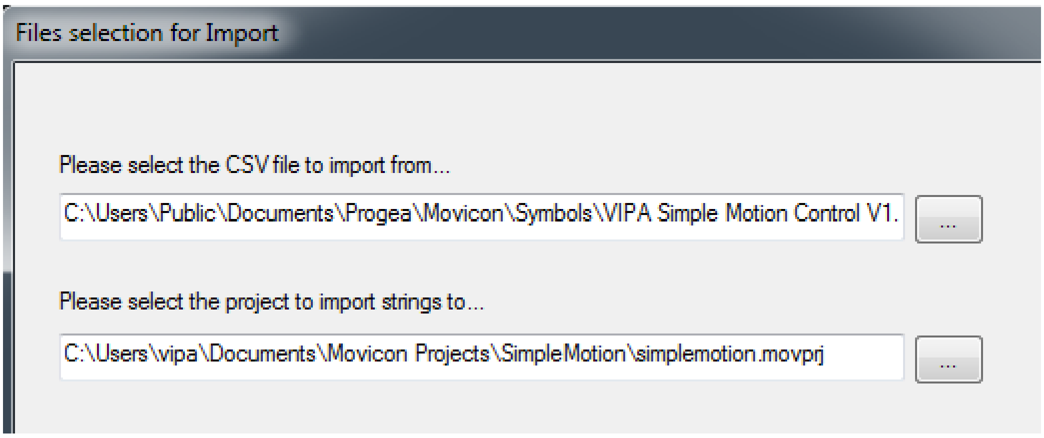
Import voice table

The templates refer to the displayed texts from a language table, which is to be imported from the working directory into your project.

1. Select *'Tools ➔ Csv String Importer-Exporter'*.
   ⇨ The *'String Import/Export tool'* opens.



2. Click at [Import].
3. For the CSV file, use [...] to navigate to your Movicon user directory ...\Public\Documents\Progea\Movicon\Symbols and select the file *'vipa simple motion control VX.X.X.CSV'*.
4. As a project directory, you specify the project file *'simplemotion.movprj'* which is located in the user directory such as ...\vipa\Documents\Movicon Projects\SimpleMotion.



5. Click at [Continue].
   ⇨ *'Language selection'* opens.

6. Select [Select all languages] and click at [Finish].

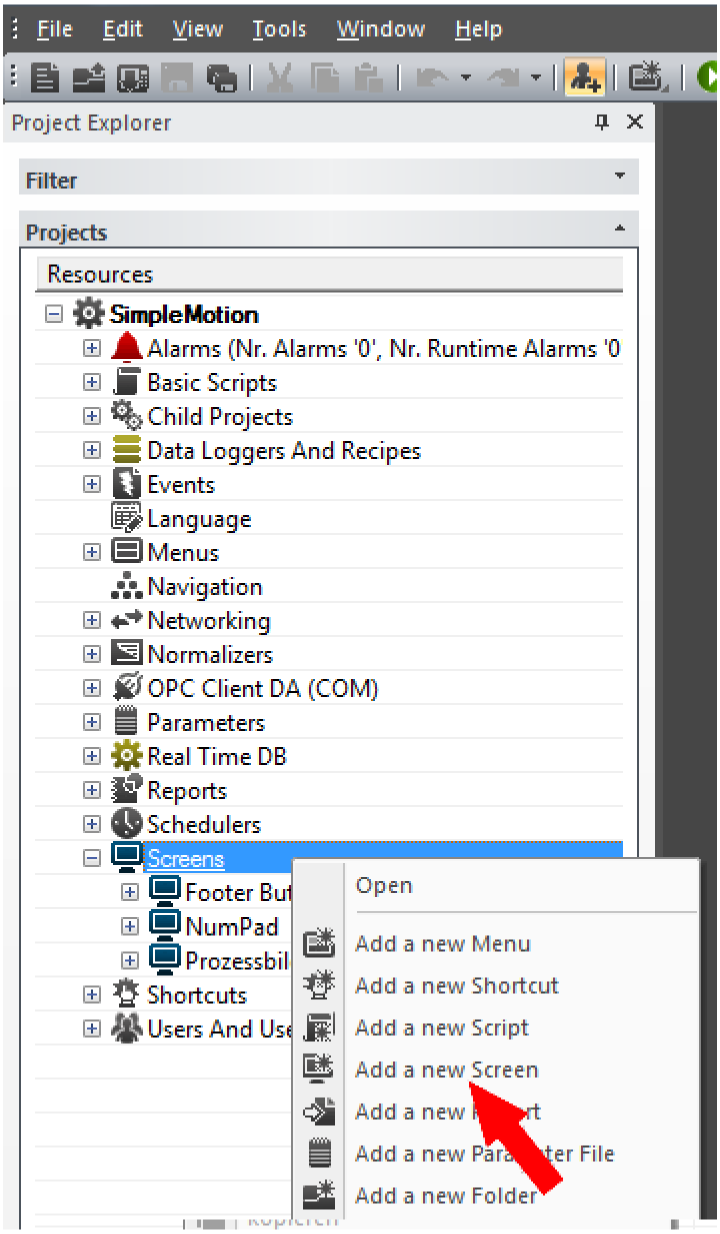
⇨ The language table is imported into your project.

7. After successful import, close the *'String Import/Export tool'*.

Adjust the numeric input field

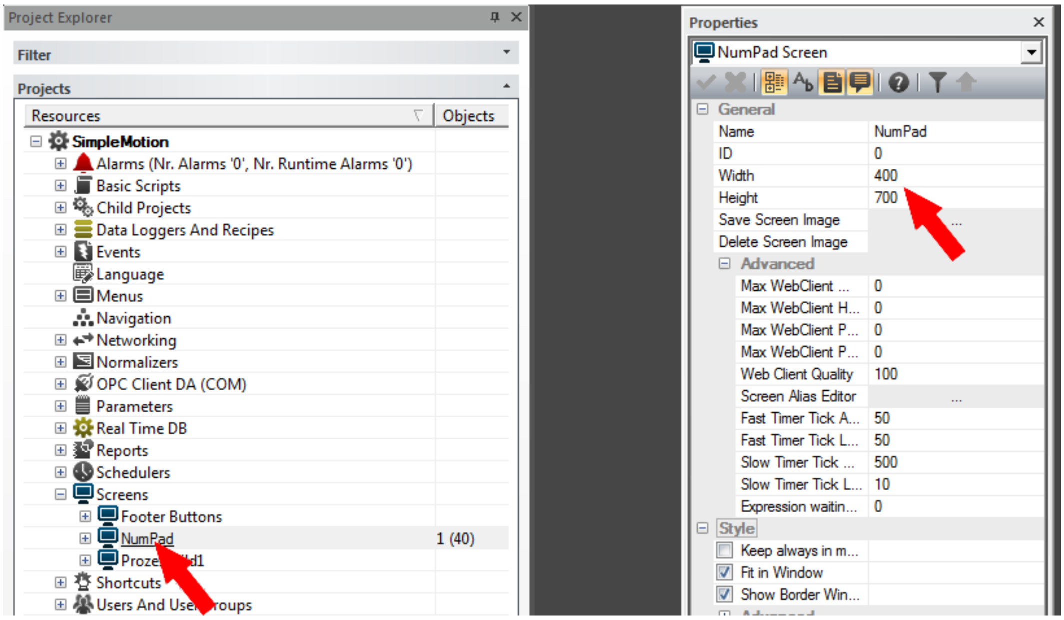
At the templates, you will find a *'Numeric Touchpad'* in various resolutions. This is an input field adapted to the VMC_AxisControl templates for different display resolutions. You can use this touch pad instead of the default input field using the following procedure.

1. Click at *'Resources > SimpleMotion > Screens'* and select *'Context menu ➔ Add a new screen'*.

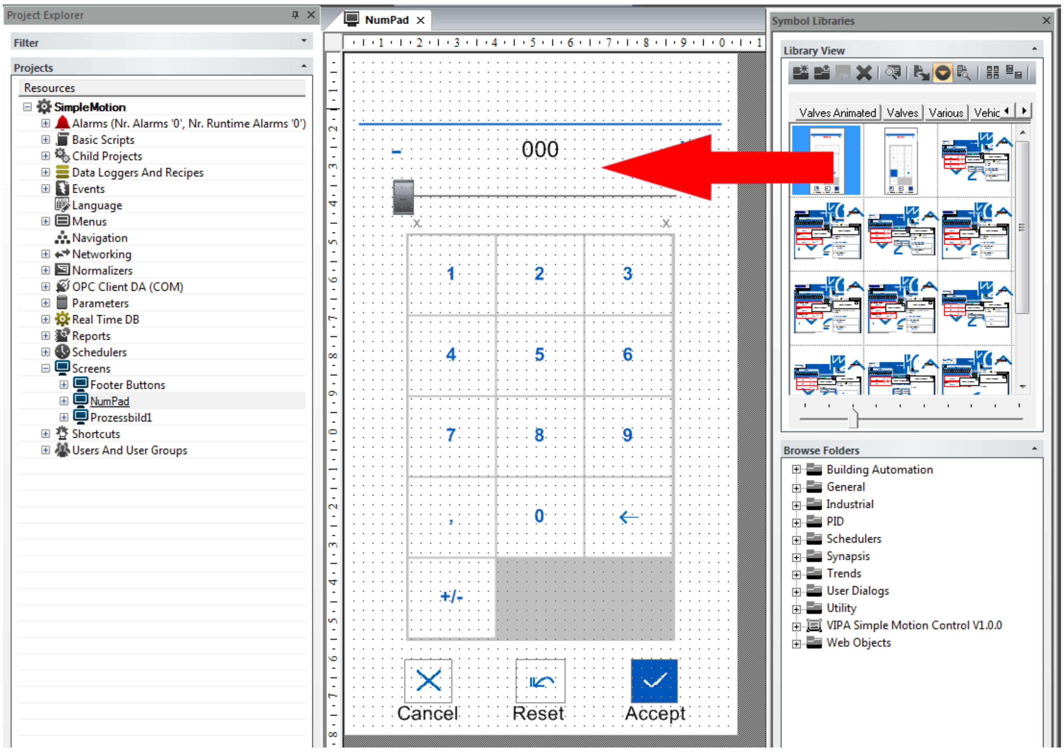


2. Assign a name such as *'NumPad'* and confirm with [OK].

3. Click at the screen *'NumPad'* and adjust via *'Context menu → Properties'* width and height such as *'Width'* = 400 and *'Height'* = 700. Confirm with ✔ your settings.



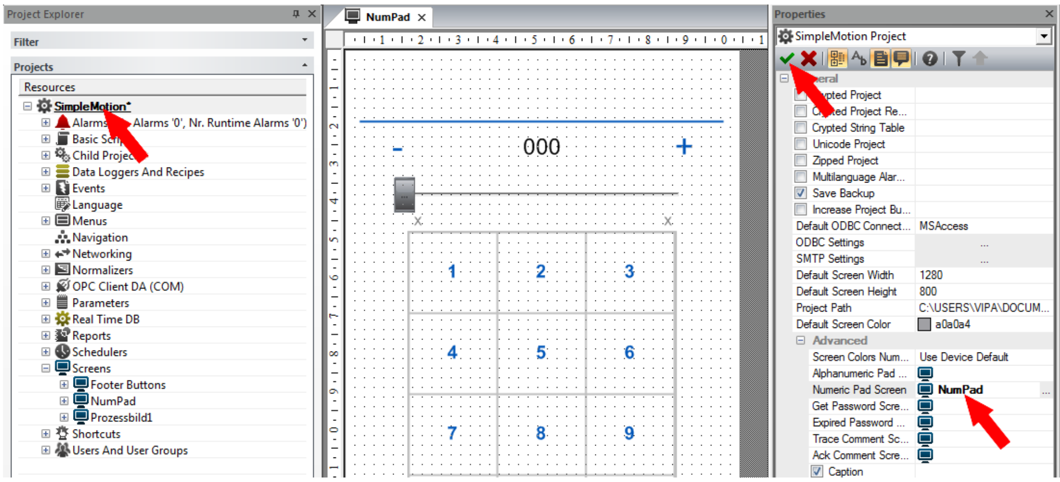
4. Select *'View → Symbol Libraries'*. Navigate in *'Browse Folders'* at *'vipa simple motion control ...'* and drag & drop from the *'Library view'* the *'Numeric Touchpad'* template to the *'NumPad'*, which matches the resolution of your panel.



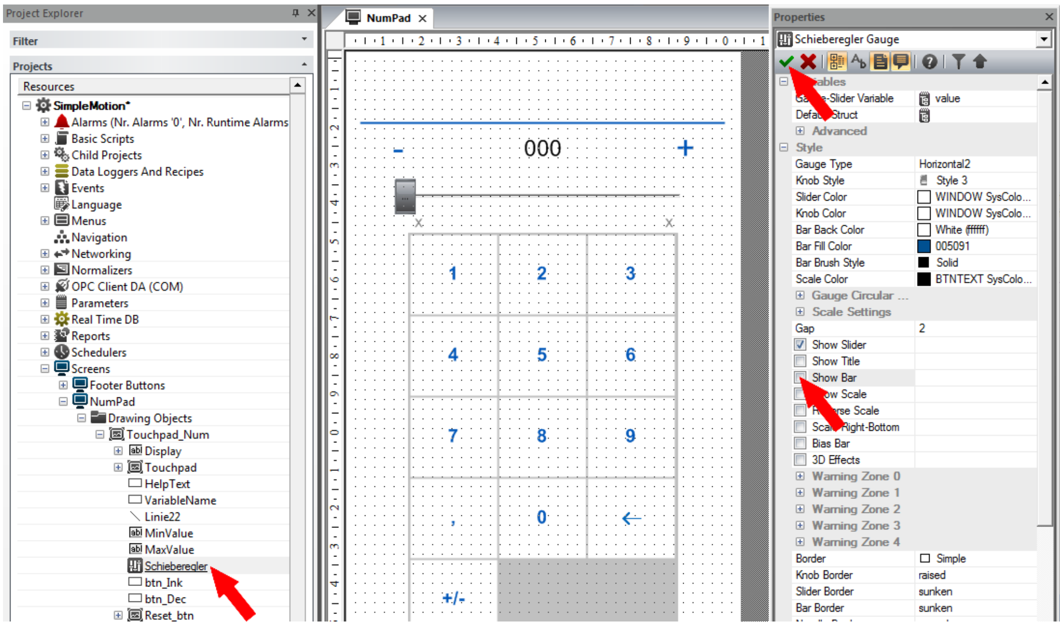
5. If necessary, adjust its size.

6. Click at *'Resources > SimpleMotion'* and select *'Context menu → Properties'*.

7. Select at *'General > Advanced'* the numeric touch pad *'NumPad'*. Confirm with your settings.



8. For optical adjustment click at *'Resource > SimpleMotion > Screens > NumPad > Drawing Objects > Touchpad_Num'* at *'Schieberegler'* (slide control) and select *'Context menu ➔ Properties'*. Expand the *'Style'* part and disable *'Show Bar'*.



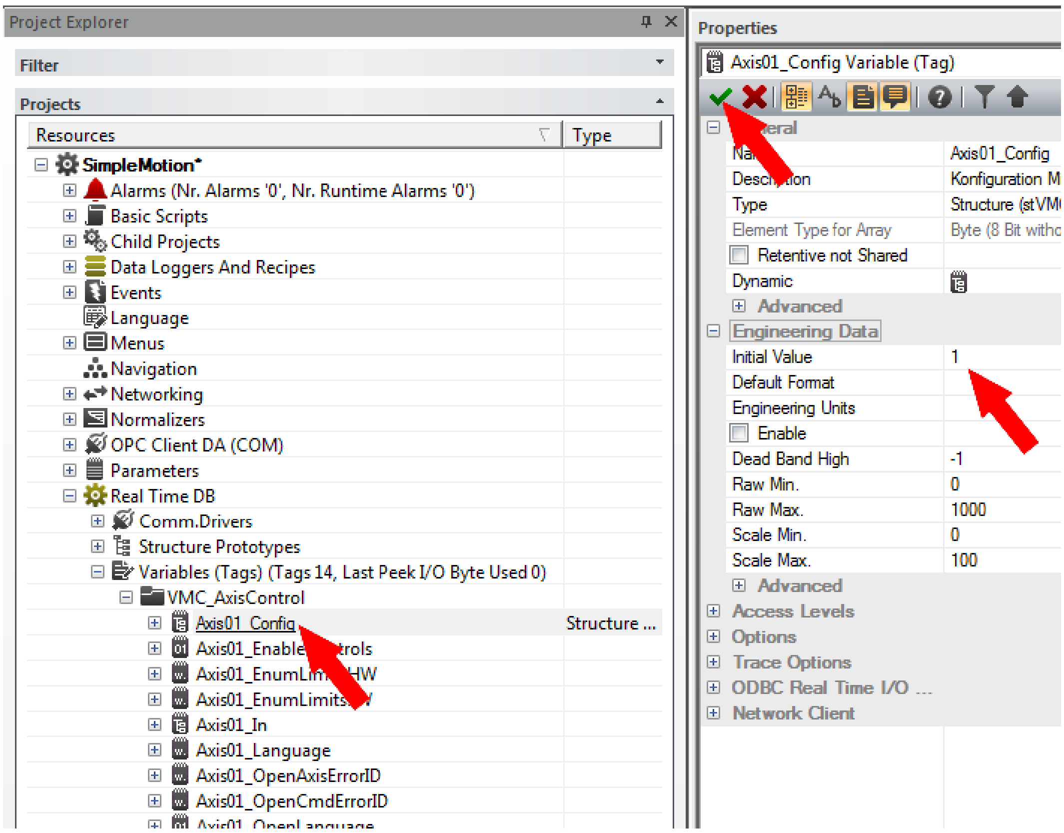
Adjust limit and default values

When a template is placed in a screen, the associated variables and structure definitions are automatically created at *'Resources > SimpleMotion > Real Time DB > Variables > VMC_AxisControl > ..._Config'*. Here the following variables are created and initial values are assigned:

AccelerationMaxValue	-	Maximum acceleration value
AccelerationMinValue	-	Minimum acceleration value
DecelerationMaxValue	-	Maximum delay value
DecelerationMinValue	-	Minimum delay value
HomePosMaxValue	-	Maximum home position
HomePosMinValue	-	Minimum home position
JogAccelerationMaxValue	-	Maximum acceleration value jog mode
JogAccelerationMinValue	-	Minimum acceleration value jog mode
JogDecelerationMaxValue	-	Maximum delay value jog mode
JogDecelerationMinValue	-	Minimum delay value jog mode
PositionMaxValue	-	Maximum position value
PositionMinValue	-	Minimum position value
VelocityMaxValue	-	Maximum speed value
VelocityMinValue	-	Minimum speed value

To adjust limit and default values click at *'Resources > SimpleMotion > Real Time DB > Variables > VMC_AxisControl > ..._Config'* and select *'Context menu ➔ Properties'*.

⇨ You can adjust the corresponding values at *'Engineering Data'*. Confirm with ✓ your settings.

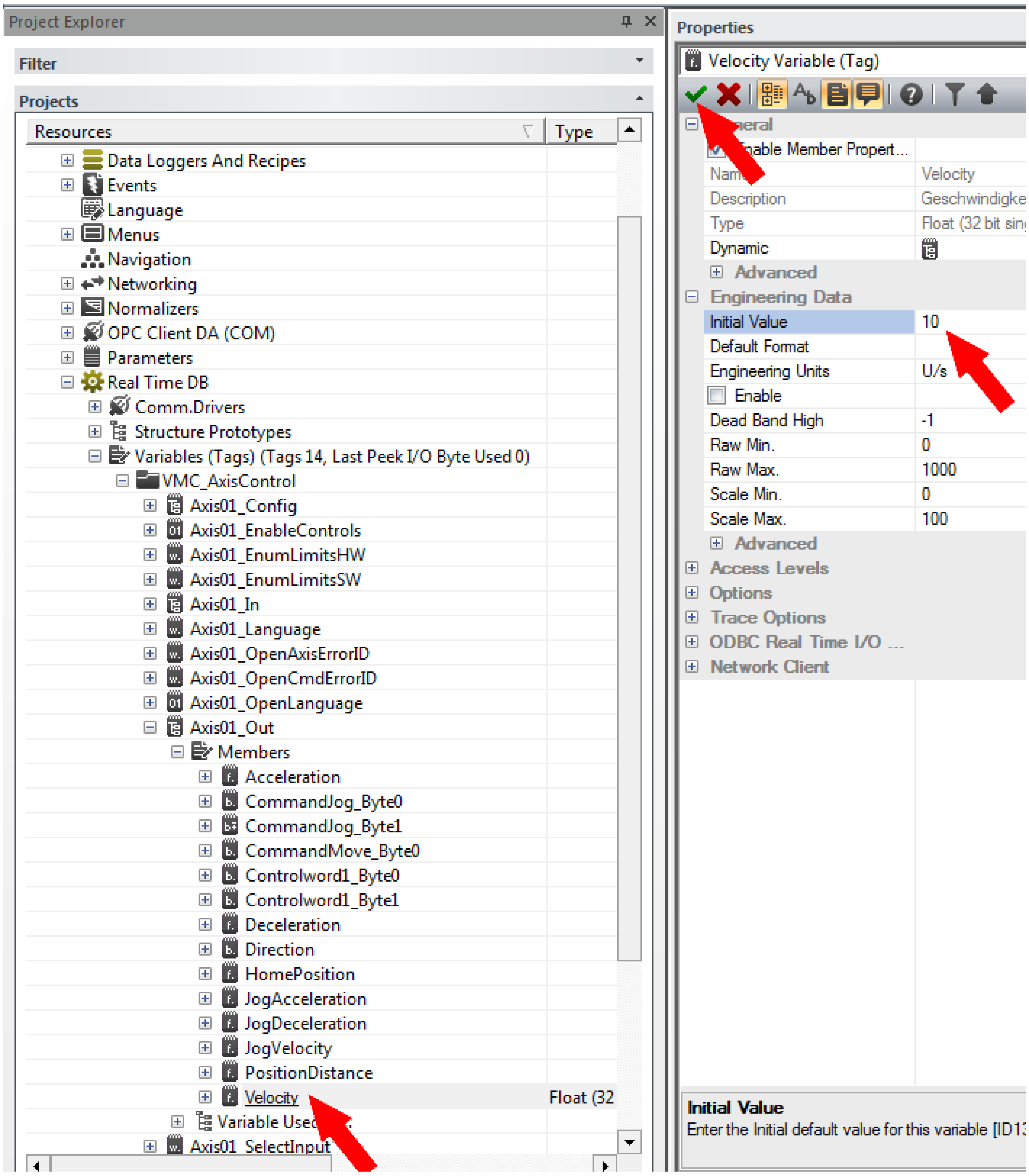


Adjust technical units

When a template is placed in a process picture, the associated variables are automatically generated with their technical units. These can be customized via the properties.

→ To adapt the technical units, e.g. for speed, click at *'Resources > SimpleMotion > Real Time DB > Variables > VMC_AxisControl > ..._Out > Members > Velocity'* and select *'Context menu ➔ Properties'*.

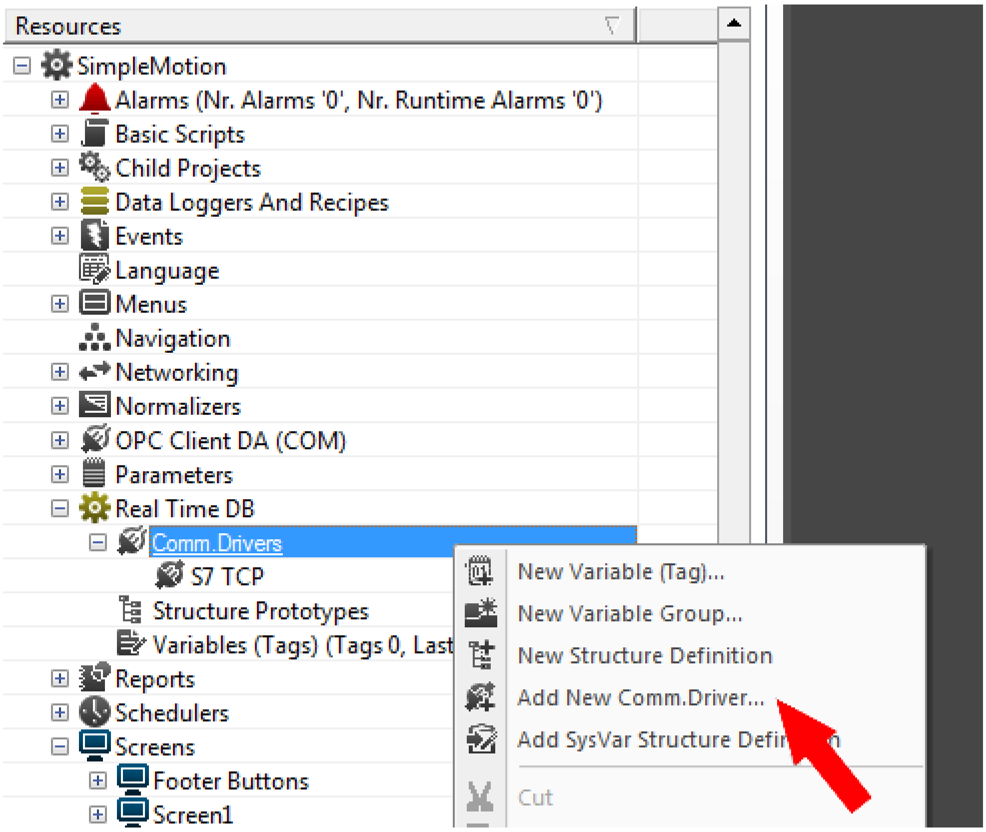
⇨ You can adjust the corresponding values at *'Engineering Data'*. Confirm with ✔ your settings.



Manually add communication driver

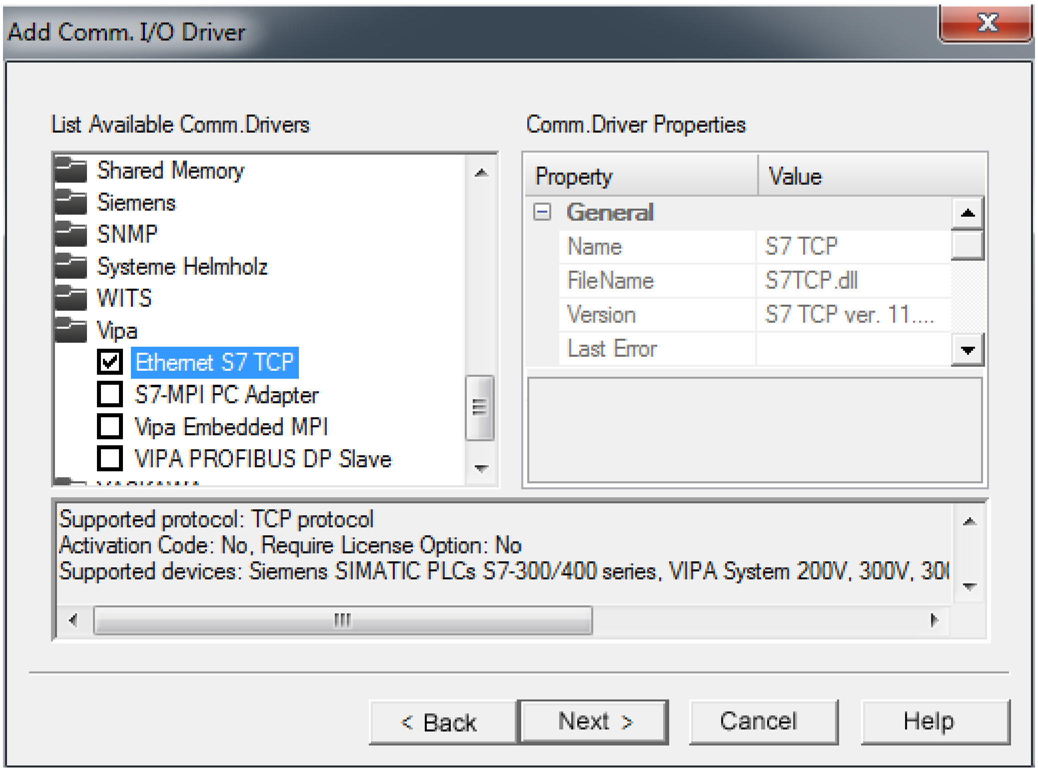
Instead of using the wizard, you can also manually add the communication driver:

1. Click at *'Resources > SimpleMotion > Real Time DB'* at *'Comm.Drivers'* and select *'Context menu ➔ Add new Comm.Driver'*.



⇨ The dialog window *'New comm. I/O Driver'* is opened.

2. Since the connection to the CPU is via TCP/IP, enable in the *'List available comm drivers'* the driver *'VIPA'* > *'Ethernet S7 TCP'* and click at [Next].



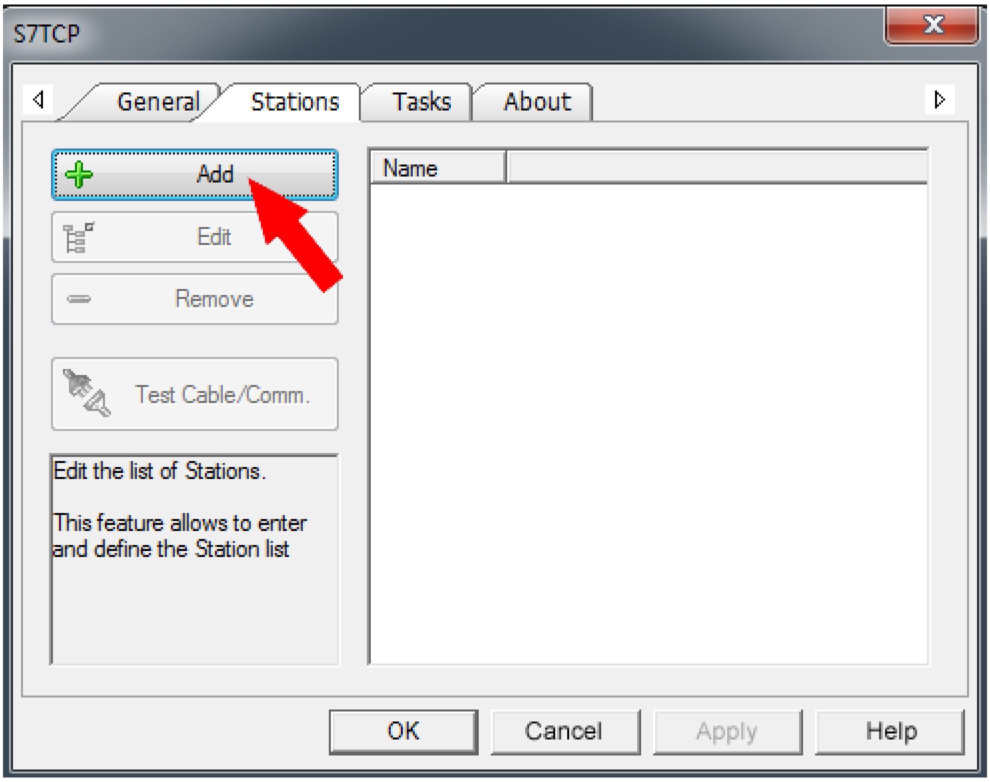
⇨ The communication driver *'S7 TCP'* is listed at *'Resources > SimpleMotion > Real Time DB > Comm.Drivers'*.

3. Click at *'S7 TCP'* and select *'Context menu ➔ Comm. I/O Driver Settings'*.

⇨ The *'S7 TCP'* dialog opens.

4. Select the register *'Stations'*.

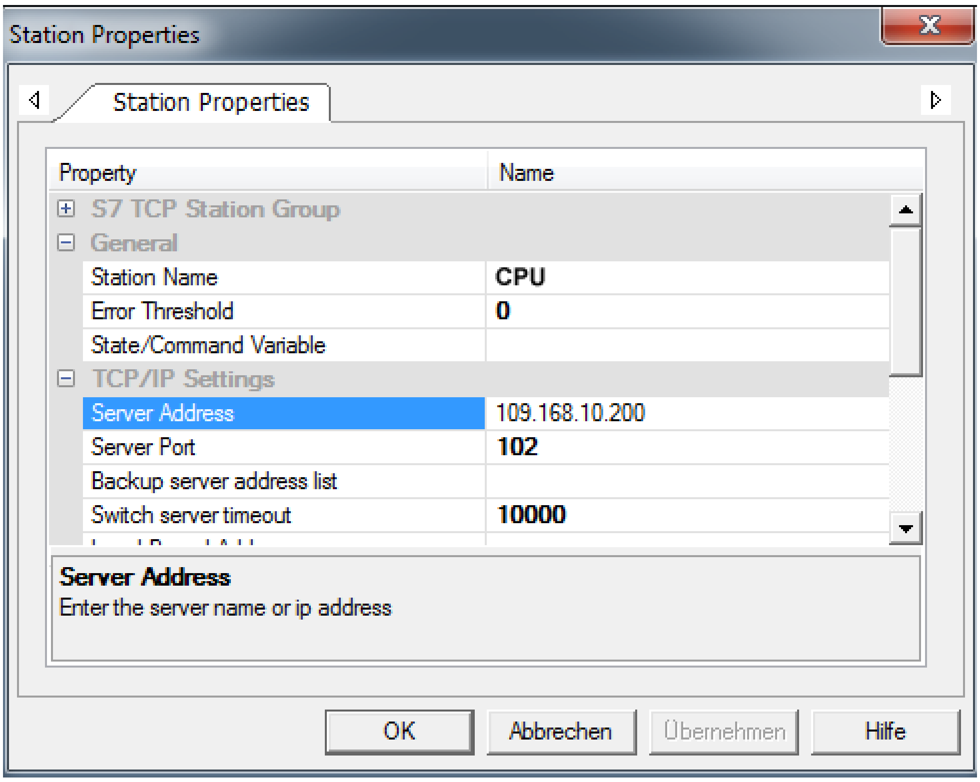
5. To add a new station, click [+ Add].



⇨ The dialog *'Station Properties'* opens.

6. Enter a station name at *'Station Name'*. Allowed characters: *A-Z*, *a-z*, *0-9* space and the separators "_" and "-"

Enter at *'Server Address'* the IP address of your CPU and click at [OK].



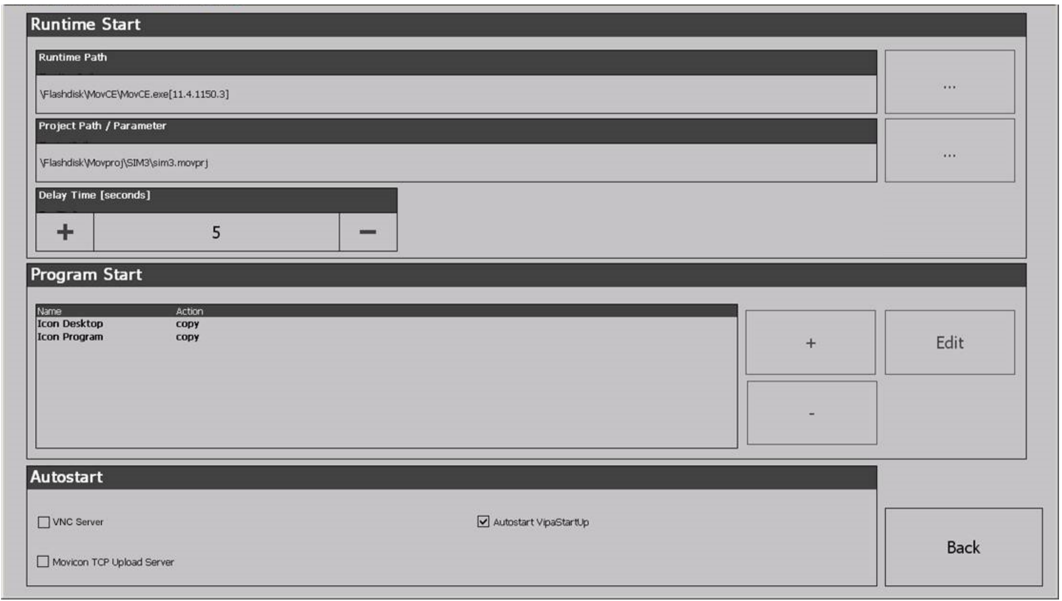
7. Negate the query for importing variables from the PLC database and close the *'S7 TCP'* dialog with [OK].

10.4 Commissioning

10.4.1 Transfer project to target device

You can transfer your project to your panel via Ethernet. The Movicon runtime version, which is pre-installed in your panel, will make your project executable.

1. Connect your PC and your panel via Ethernet.
2. Start your panel and determine the IP address of your panel in the *'Startup-Manager'*.
3. Call in your *'Startup-Manager'* the *'Autostart'* menu item.
4. To enable Movicon to transfer a project to your panel via Ethernet, you have to enable the option *'Movicon TCP Upload Server'* at *'Autostart'*.



⇨ Confirm the query for activation.

5. Now you can transfer your project to your panel from Movicon. For this in Movicon click in *'Resources'* at *'SimpleMotion'* and select *'Context menu ➔ Upload project to Device/FTP'*.

⇨ The Transfer dialog opens.

6. Select at *'PlugIn Type'* *'TCP'*.

Specify at *'Server'* the IP address of the panel.

Enter at *'User name'* and *'Password'* the access for your panel.

The following access data are used per default:

- Username: wince
- Password: vipatp

Specify at *'Upload Device Path'* you memory card and create a new project directory.

7. Start the transfer with [Upload project].
8. After successful transfer, you can add your project on the panel in the autostart directory and start it up.

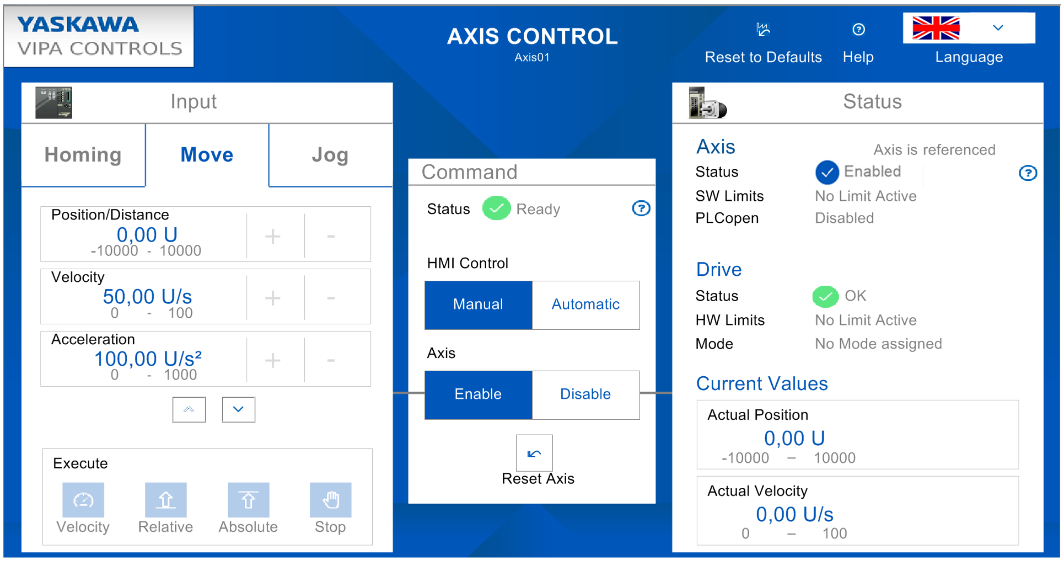
CAUTION!
Please always observe the safety instructions for your drive, especially during commissioning!

10.4.2 Controlling the *VMC_AxisControl* via the panel

10.4.2.1 Commissioning

It is assumed that you have set up your application and you can control your drive with the VMC_AxisControl function block.

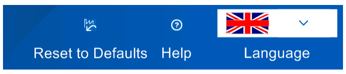
1. Connect your CPU to your panel and turn on your application.
   ⇨ The panel starts with the screen to control your drive.



In order to control your drive via the panel, you have to switch 'HMI Control' to [Manual]. If the status does not return any errors, you can activate the drive with [Enable] for the control. You can now control your drive via the corresponding buttons.

10.4.2.2 Operation

User panel



'Reset to Defaults'

- By *'Reset to Defaults'* the following values are reset to default values of the application, which you can adapt accordingly as described above:
  – Velocity: 50U/s
  – Acceleration/Deceleration: 100U/s^2
  – Position/Home Position: 0U

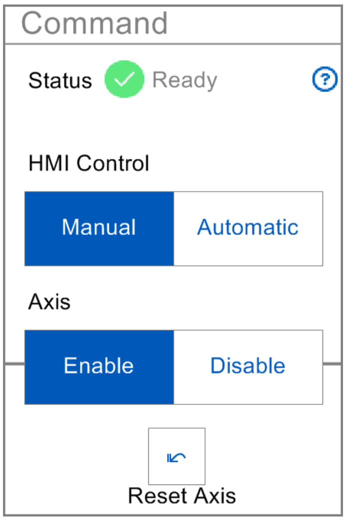
'Help'

- You can access your own help file via *'Help'*. This is to be integrated within Movicon accordingly.

'Language'

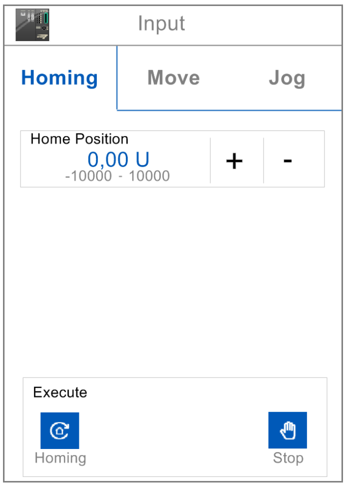
- You can use *'Language'* to specify the appropriate language for the user interface.

'Command'



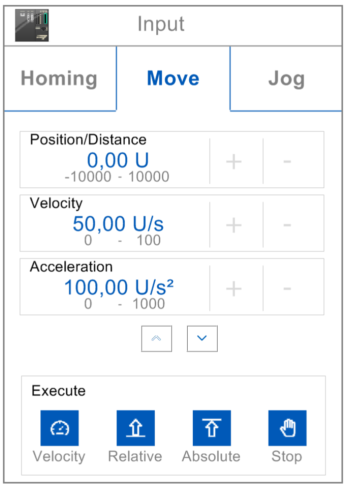
- *'Status'*
  - Here you can see the current status of your driving command.
- *'HMI Control'*
  - *'Manual'*: When activated, the drive can be controlled via the panel.
  - *'Automatic'*: In the activated state, the drive is controlled via the PLC program of your CPU and can not be influenced by the panel.
- *'Axis'*
  - *'Enable'*: The drive is enabled in the activated state and when *'Manual'* of *'HMI Control'* is activated and you can control this via the *'Input'* area.
  - *'Disable'*: When activated, the drive is disabled and no control is possible.
- *'Reset Axis'*
  - On error, the control buttons become inactive. With *'Reset Axis'* you can acknowledge error messages and and reactivate buttons.

'Input'



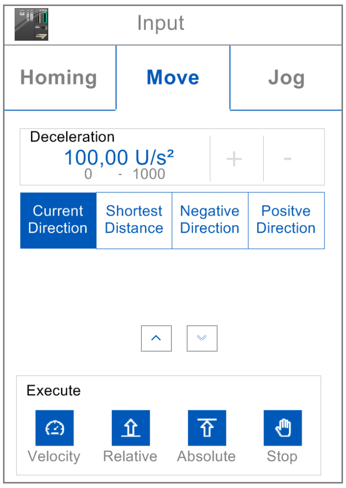
'Homing'

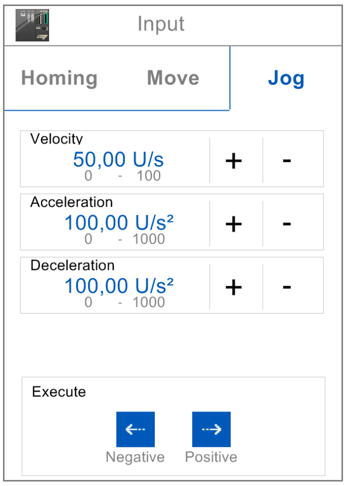
- You can use the input field or [+] and [-] to specify a homing position and move to this via *'Execute > Homing'* as a reference point.
- You can stop the homing with *'Execute > Stop'*.



'Move'

- Via the corresponding input field or [+] and [-] you can specify *'Position/Distance'*, *'Velocity'*, *'Acceleration'* and *'Deceleration'* and execute them via the corresponding driving command at *'Execute'*. Use [v] to navigate down.
  - *'Velocity'*: When actuated, the drive executes the drive command at a constant velocity.
  - *'Relative'*: When actuated, the drive moves to the relative position, which can be pre-set at *'Position/Distance'*.
  - *'Absolute'*: When actuated, the drive moves to the absolute position, which can be pre-set at *'Position/Distance'*.
  - *'Stop'*: When actuated, the drive is stopped.
  - *'Current direction'*: When activated, the current drive direction is used.
  - *'Shortest distance'*: When activated, the shortest distance to the specified position is used.
  - *'Negative direction'*: When activated, the negative drive direction is used.
  - *'Positive direction'*: When activated, the positive drive direction is used.

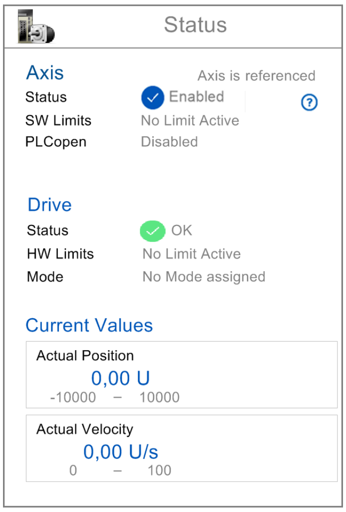




'Jog'

- Via the corresponding input field or [+] and [-] you can specify *'Velocity'*, *'Acceleration'* and *'Deceleration'* and execute the according drive command to positive respectively negative direction via the direction buttons at *'Execute'*.
- As long as you press one of the direction buttons, the drive is accelerated to the required speed with the specified acceleration.
- When the direction button is released, the drive is stopped with the specified deceleration.

'Status'



'Axis'

- *'Status'* The status of your axis is shown here.
  – *'Enabled'*: The axis is switched on.
  – *'Ready'*: The axis is ready to switch on.
  – *'Disabled'*: The axis is disabled.
  – *'Axis error'*: An axis error is pending, indicating the error number. *Chap. 12 'ErrorID - Additional error information' page 457*
- *'SW Limits'*: As soon as SW limits exist, this is shown here.
- *'PLCopen'*: The PLCopen status is shown here.

'Drive'

- *'Status'*: The status of the drive controller is shown here.
- *'HW-Limits'*: Here, a possible limitation in your drive controller is shown here.
- *'Mode'*: Here you can get information about the currently selected drive profile.

'Current Values'

- The current values of *'Position'* and *'Velocity'* are shown here.
- Values that are outside the defined limits are framed in red.

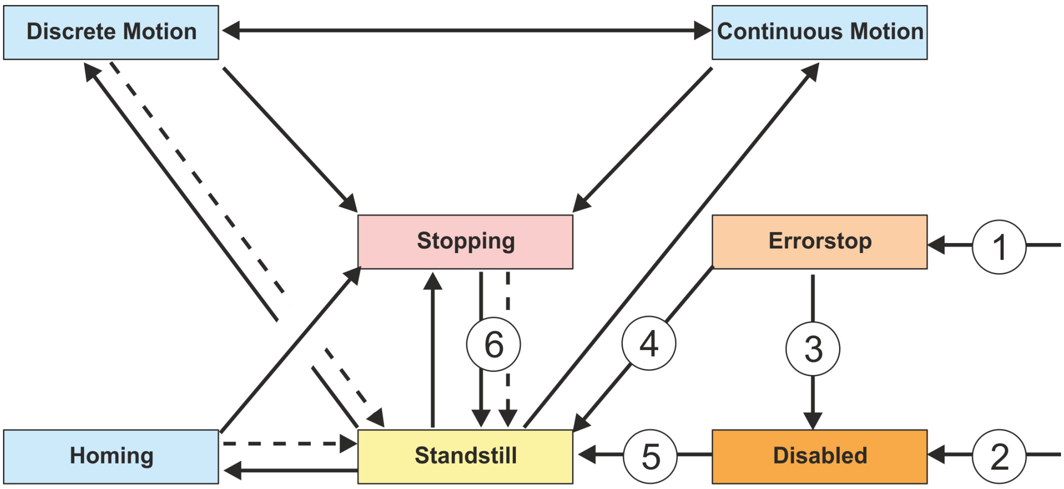
11 States and behavior of the outputs

11.1 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. 9.3.11 'FB 812 - MC_ReadStatus - PLCopen status' page 384*
- Parameter *PLCopenState* from *Chap. 9.2.2 'FB 860 - VMC_AxisControl - Control block axis control' page 363*



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

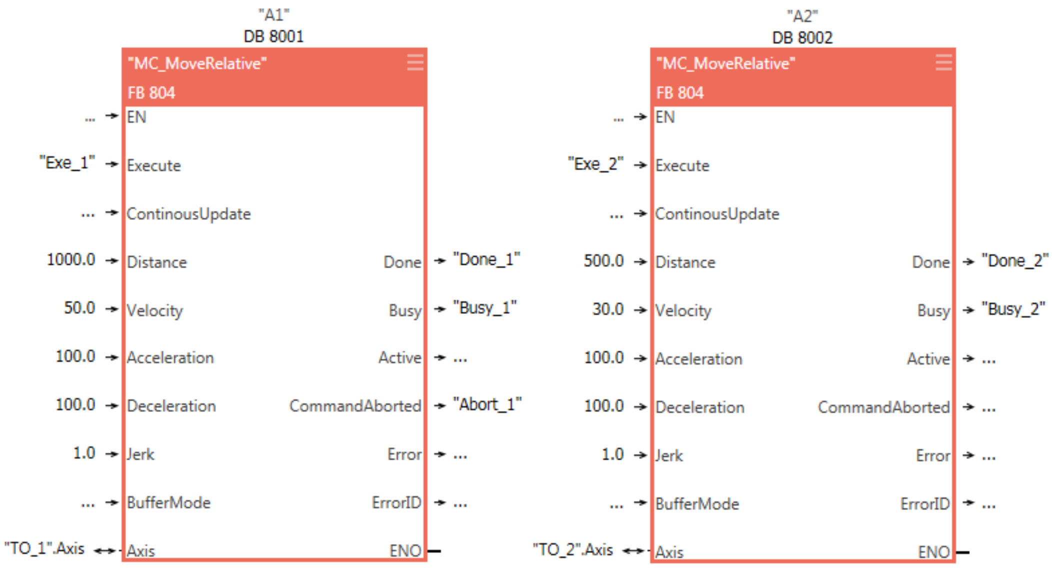
- Disabled
  - Basic state of an axis.
  - Axis can not be moved by any function block.
- Error Stop
  - An error has occurred on the axis.
  - Axis is stopped and is blocked for further motion tasks.
  - Axis remains in this state until the error is solved and a RESET is triggered.
  - Errors on an axis are also reported via the corresponding function block.
  - Errors on a function block do not lead to this state
- Stand Still
  - 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. 9.3.5 'FB 802 - MC_Stop - stop axis' page 372
    Chap. 9.2.2 'FB 860 - VMC_AxisControl - Control block axis control' page 363
  - 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*.

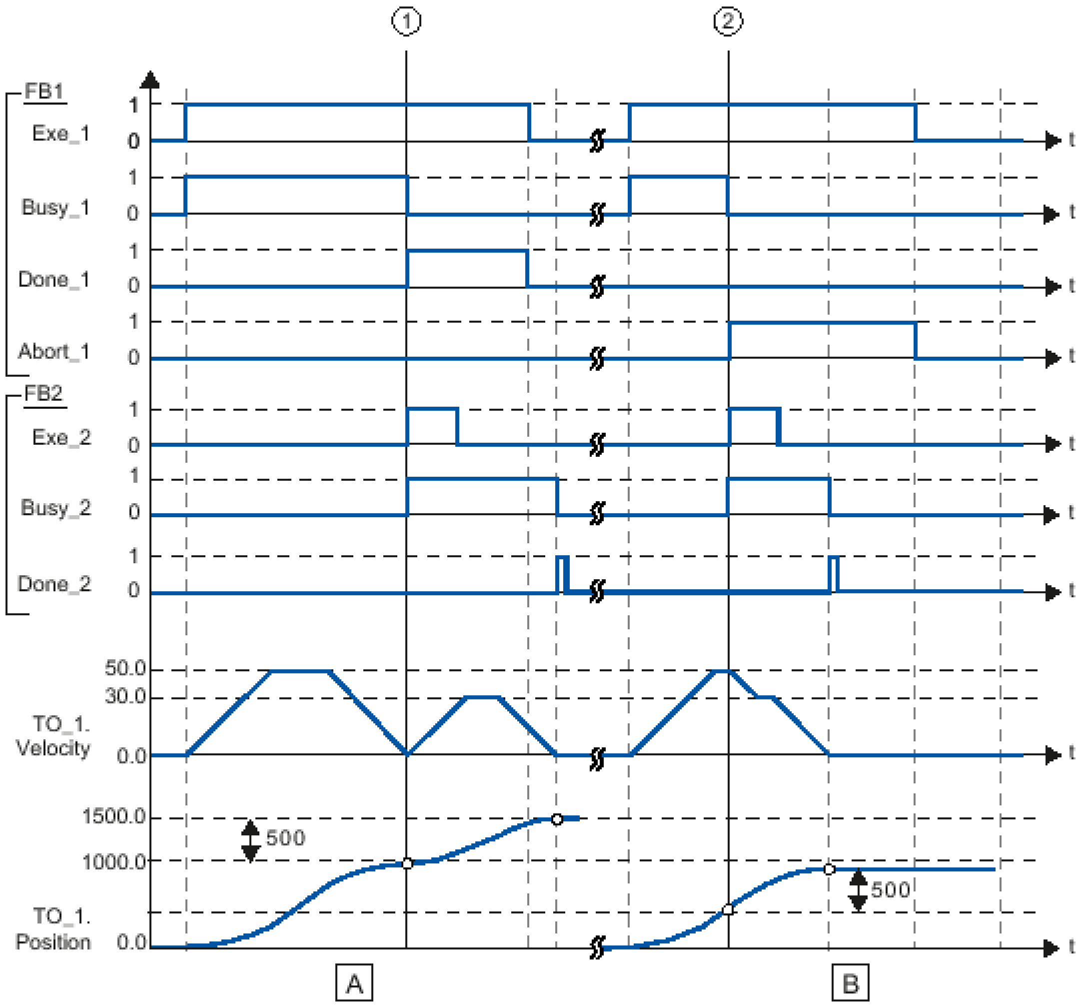
- Homing
  – The axis is currently homing:
    Chap. 9.3.4 'FB 801 - MC_Home - home axis' page 370
    Chap. 9.2.2 'FB 860 - VMC_AxisControl - Control block axis control' page 363
  – As soon as the axis is homed, the state automatically changes to *Standstill*.
- Discrete Motion
  – The axis is currently executing a motion task:
    Chap. 9.3.9 'FB 808 - MC_MoveAbsolute - move axis to absolute position' page 380
    Chap. 9.3.7 'FB 804 - MC_MoveRelative - move axis relative' page 376
    Chap. 9.3.6 'FB 803 - MC_Halt - holding axis' page 374
    Chap. 9.2.2 'FB 860 - VMC_AxisControl - Control block axis control' page 363
  – 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. 9.3.8 'FB 805 - MC_MoveVelocity - drive axis with constant velocity' page 378
    Chap. 9.2.2 'FB 860 - VMC_AxisControl - Control block axis control' page 363

11.2 Replacement behavior of motion jobs

Example

In the following with an example of MC_MoveRelative the replacement behavior of motion jobs is explained. *Chap. 9.3.7 'FB 804 - MC_MoveRelative - move axis relative' page 376*





(A) The axis is moved by the "MC_MoveRelative" job (A1) by the *Distance* 1000.0 (starting position is the position 0.0).

(1) Reaching the target position is reported at the time (1) *Done_1*. At this time (1) a further MC_MoveRelative order (A2) is started with the route 500.0. The successful achievement of the new target position is reported via *Done_2*. Since *Exe_2* was reset before, *Done_2* is only set for one cycle

(B) A running MC_MoveRelative job (A1) is replaced by a further MC_MoveRelative job (A2).

(2) The abort is reported at time (2) via *Abort_1*. The axis is then moved with the new velocity by the new distance *Distance* 500.0. The successful achievement of the new target position is reported via *Done_2*.

11.3 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.

BufferMode

- *BufferMode* is not supported.

12 ErrorID - Additional error information

ErrorID	Description	Remark
0x0000	No Error	
0x8y24	Error in block parameter y, with y: ■ 1: Error in PROTOKOLL ■ 2: Error in PARAMETER ■ 3: Error in BAUDRATE ■ 4: Error in CHARLENGTH ■ 5: Error in PARITY ■ 6: Error in STOPBITS ■ 7: Error in FLOWCONTROL (parameter missing)	VMC_ConfigMaster_RTU
0x8001	Invalid value at parameter *Position*.	
0x8002	Invalid value at parameter *Distance*.	
0x8003	Invalid value at parameter *Velocity*.	
0x8004	Invalid value at parameter *Acceleration*.	
0x8005	Invalid value at parameter *Deceleration*.	
0x8007	Invalid value at parameter *ContinuousUpdate*.	
0x8008	Invalid value at parameter *BufferMode*.	
0x8009	Invalid value at parameter *EnablePositive*.	
0x800A	Invalid value at parameter *EnableNegative*.	
0x800B	Invalid value at parameter *MasterOffset*.	
0x800C	Invalid value at parameter *SlaveOffset*.	
0x800D	Invalid value at parameter *MasterScaling*.	
0x800E	Invalid value at parameter *SlaveScaling*.	
0x800F	Invalid value at parameter *StartMode*.	
0x8010	Invalid value at parameter *ActivationMode*.	
0x8011	Invalid value at parameter *Source*.	
0x8012	Invalid value at parameter *Direction*.	
0x8014	Invalid parameter of physical axis.	MC_ReadParameter
0x8015	Invalid index or subindex.	MC_ReadParameter
0x8016	Invalid parameter length.	MC_ReadParameter
0x8017	Invalid LADDR if e.g. the corresponding drive system is switched off or cannot be reached.	MC_ReadParameter
0x8018	Invalid value at parameter *RatioDenominator*.	MC_GearIn
0x8019	Invalid value at parameter *RatioNumerator*.	MC_GearIn
0x801A	Unknown parameter number.	MC_ReadParameter, MC_Write-Parameter
0x801B	Parameter can not be written, parameter is write protected	MC_WriteParameter
0x801C	Parameter communication with unknown mode.	MC_Home, MC_WriteParameter

ErrorID	Description	Remark
0x801D	Parameter communication with general error. The cause of the error is not described in detail.	
0x801E	SDO parameter value out of range.	MC_Home, MC_WriteParameter
0x801F	The Type in ANY is not BYTE.	Read/write parameter
0x8020	Different configuration of the user units in cam and master axis.	
0x8021	Different configuration of the user units in cam and slave axis.	
0x8022	There is no PROFIBUS/PROFINET device at the logical address specified in LADDR, from which you can read consistent data.	Read/write parameter
0x8023	An access error has been detected when accessing an I/O device.	Read/write parameter
0x8024	Slave error at external DP slave.	Read/write parameter
0x8025	System error at external DP slave.	Read/write parameter
0x8026	System error at external DP slave.	Read/write parameter
0x8027	The data haven't yet been read by the module.	Read/write parameter
0x8028	System error at external DP slave.	Read/write parameter
0x8029	Attempt to write a read only object.	Read/write parameter
0x802A	Attempt to read a write only object.	Read/write parameter
0x802B	Unsupported access to an object.	Read/write parameter
0x802C	Wrong data type.	Read/write parameter
0x802D	Error in device profile.	Read/write parameter
0x802E	Error command type.	Read/write parameter
0x802F	No system resources available.	Read/write parameter
0x8030	Invalid value at parameter *Hardware* (1 = SLIO CP; 2 = VIPA CPU).	Modbus; Init
0x8031	Invalid value at parameter *UnitId*.	Modbus; Init
0x8032	Invalid value at parameter *UserUnitsVelocity* (0 = Hz, 1 = %, 2 = RPM).	Modbus; Init
0x8033	Invalid value at parameter *UserUnitsAcceleration* (0 = 0.00s, 1 = 0.0s).	Modbus; Init
0x8034	Invalid value at parameter *MaxVelocityApp* (must be > 0).	Modbus; Init
0x8035	Error while read access at *MonitorData*.	Modbus; Init
0x8036	Error while read access at *NumberOfPoles*.	Modbus; Init
0x8037	Error while write access to *UserUnitsVelocity*.	Modbus; Init
0x8038	Error while read access at *MinOutputFrequency*.	Modbus; Init
0x8039	Error while read access at *MaxOutputFrequency*.	Modbus; Init
0x803A	Error while write access to *StoppingMethodSelection*.	Modbus; Init
0x803B	Error while write access to *UserUnitsAcceleration*.	Modbus; Init
0x8041	Invalid value at parameter *AccelerationTime*.	Modbus V1000
0x8042	Invalid value at parameter *DecelerationTime*.	Modbus V1000
0x8043	Invalid value at parameter *JogAccelerationTime*.	Modbus V1000

ErrorID	Description	Remark
0x8044	Invalid value at parameter *JogDecelerationTime*.	Modbus V1000
0x8045	Invalid value at parameter *JogVelocity* (≤ *MaxVelocityApp*).	Modbus V1000
0x80C8	Modbus communication error: No response from the server in the defined period (timeout can be parametrized via interface).	Modbus V1000
0x809y	Error in value of the block parameter y, with y: ■ 1: Error in PROTOKOLL ■ 3: Error in BAUDRATE ■ 4: Error in CHARLENGTH ■ 5: Error in PARITY ■ 6: Error in STOPBITS	VMC_ConfigMaster_RTU
0x8092	Access error on parameter DB (DB too short).	VMC_ConfigMaster_RTU
0x809A	Interface not available or used with PROFIBUS.	VMC_ConfigMaster_RTU
0x8101	No cyclic 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 *'switched on'* → reset drive error possibly with MC_Reset. ■ Communication was interrupted, e.g. by CPU power cycle. Reset error with MC_Reset.	*PreOperational* has also to be set in *Operational*.
0x8105	Command is not supported by virtual axes.	
0x8106	PLCopen-State is not defined.	
0x8107	Command is not permitted if drive is deactivated.	VMC_AxisControl_PT, Mod-busV1000
0x8188	Modbus communication error: Internal error MB_FUNCTION invalid.	Modbus V1000
0x8189	Modbus communication error: Internal error MB_DATA_ADDR invalid.	Modbus V1000
0x818A	Modbus communication error: Internal error MB_DATA_LEN invalid.	Modbus V1000
0x818B	Modbus communication error: Internal error MB_DATA_PTR invalid.	Modbus V1000
0x8201	Command cannot be executed temporarily because of lack of internal resources (no free slot in CommandBuffer).	
0x8202	Error when writing the offset for homing (no free slot in the Com-mandBuffer).	DriveManager → Homing (active command)
0x8210	Modbus communication error: The hardware is incompatible with the Modbus RTU/TCP block library.	Modbus V1000
0x828y	Error in parameter y of DB parameters, with y: ■ 1: Error in 1. Parameter ■ 2: Error in the 2. Parameter ■ ...	VMC_ConfigMaster_RTU

ErrorID	Description	Remark
0x8301	No cyclic communication with master axis possible.	
0x8302	Command is in current PLCopen-State of the master axis not allowed.	
0x8303	Command is not supported by the master axis.	
0x8304	Master axis is not in status *Pre-Operational*.	
0x8305	Master axis data block number has been changed.	
0x8306	Communication errors at the master axis. Slave axis is stopped with fast stop.	
0x8311	No cyclic communication with slave axis possible.	
0x8312	Command is in current PLCopen-State of the slave axis not allowed.	
0x8313	Command is not supported by the slave axis.	
0x8314	Slave axis is not in status *Pre-Operational*.	
0x8315	Slave axis data block number has been changed.	
0x8317	Block was not called within OB 1.	VMC_AxisControl_PT
0x8321	Coupling with *StartMode* = relative and *ActivationMode* = nextcycle is not permitted.	
0x8322	Coupling or switching with *StartMode* = absolute and *Activation-Mode* = nextcycle is not permitted.	
0x8323	Switching with a different *StartMode* (*StartMode* of the coupling is to be used).	
0x8331	MC_CamIn is not active.	
0x8332	MC_GearIn is not active.	
0x8340	Invalid value at TriggerInput.Probe.	MC_TouchProbe and MC_Abort-Trigger
0x8341	Invalid value at TriggerInput.Source.	MC_TouchProbe and MC_Abort-Trigger
0x8342	Invalid value at TriggerInput.TriggerMode.	MC_TouchProbe and MC_Abort-Trigger
0x8350	Invalid value at VelocitySearchSwitch.	Homing, initialization
0x8351	Invalid value at VelocitySearchZero.	Homing, initialization
0x8352	Invalid combination of inputs.	Homing, initialization
0x8360	The CPU does not support Pulse Train.	VMC_AxisControl_PT
0x8361	Wrong value in *S_ChannelNumberPWM*.	VMC_AxisControl_PT
0x8362	General error in Pulse Train output.	VMC_AxisControl_PT
0x8363	Move command with the *StopExecute* set.	VMC_AxisControl_PT, Mod-busV1000
0x8381	Modbus communication error: Server returns Exception code 01h.	Modbus V1000
0x8382	Modbus communication error: Server returns Exception code 03h or wrong start address.	Modbus V1000

ErrorID	Description	Remark
0x8383	Modbus communication error: Server returns Exception code 02h.	Modbus V1000
0x8384	Modbus communication error: Server returns Exception code 04h.	Modbus V1000
0x8386	Modbus communication error: Server returns wrong function code.	Modbus V1000
0x8388	Modbus communication error: Server returns wrong value or wrong number.	Modbus V1000
0x8400	MC_Power: Unexpected Drive-State Drive-State <> Operation enabled	MC_Power
0x8401	MC_Power: Unexpected Drive-State Drive-State = Quick stop active	MC_Power
0x8402	MC_Power: Unexpected Drive-State Drive-State = Fault reaction active	MC_Power
0x8403	MC_Power: Unexpected Drive-State Drive-State = Fault	MC_Power
0x8410	Timeout while trying to reset the drive.	Kernel FB --> MC_Reset
0x8500	Wrong value in *EncoderType* (1 or 2).	Init block
0x8501	Wrong value in *EncoderResolutionBits* (>0 and ≤32).	Init block
0x8502	Wrong value in *LogicalAddress* (≥0).	Init block
0x8503	Wrong value in *StartInputAddress* (≥0).	Init block
0x8504	Wrong value in *StartOutputAddress* (≥0).	Init block
0x8505	Wrong value in *FactorPosition* (>0.0).	Init block
0x8506	Wrong value in *FactorVelocity* (>0.0).	Init block
0x8507	Wrong value in *FactorAcceleration* (>0.0).	Init block
0x8508	Wrong value in *MaxVelocityApp* (>0.0).	Init block
0x8509	Wrong value in *MaxAccelerationApp* (>0.0).	Init block
0x850A	Wrong value in *MaxDecelerationApp* (>0.0).	Init block
0x850B	Wrong value in *MaxVelocityDrive* (>0.0).	Init block
0x850C	Wrong value in *MaxAccelerationDrive* (>0.0).	Init block
0x850D	Wrong value in *MaxDecelerationDrive* (>0.0).	Init block
0x850E	Wrong value in *MinPosition* (≥MinUserPos).	Init block
0x850F	Wrong value in *MaxPosition* (≥MaxUserPos).	Init block
0x8510	Wrong value in *M2_EncoderType*.	VMC_InitSigma7W_EC
0x8511	Wrong value in *M2_EncoderResolutionBits*.	VMC_InitSigma7W_EC
0x8513	Wrong value in *M2_PdoInputs*.	VMC_InitSigma7W_EC
0x8514	Wrong value in *M2_PdoOutputs*.	VMC_InitSigma7W_EC
0x8515	Wrong value in *M2_FactorPosition*.	VMC_InitSigma7W_EC
0x8516	Wrong value in *M2_FactorVelocity*.	VMC_InitSigma7W_EC
0x8517	Wrong value in *M2_FactorAcceleration*.	VMC_InitSigma7W_EC

ErrorID	Description	Remark
0x8518	Wrong value in *M2_MaxVelocityApp*.	VMC_InitSigma7W_EC
0x8519	Wrong value in *M2_MaxAccelerationApp*.	VMC_InitSigma7W_EC
0x851A	Wrong value in *M2_MaxDecelerationApp*.	VMC_InitSigma7W_EC
0x851D	Wrong value in *ParaAccessPointAddress*.	VMC_InitSigma_PN
0x8603	Error homing at the drive, speed <> 0.	MC_Home
0x8604	Error homing at the drive, speed = 0.	MC_Home
0x8700	Error: Invalid size.	
0x8710	SDO error: Toggle bit has not changed.	
0x8711	SDO error: SDO protocol timeout.	
0x8712	SDO error: Client / server command is not valid or unknown.	
0x8713	SDO error: Invalid block size (only in block mode).	
0x8714	SDO error: Invalid sequence number (only in block mode).	
0x8715	SDO error: CRC error (only in block mode).	
0x8716	SDO error: Out of memory.	
0x8717	SDO error: Unsupported access to an object.	
0x8718	SDO error: Attempt to read a write only object.	
0x8719	SDO error: Attempt to write a read only object.	
0x871A	SDO error: Object does not exist in the object dictionary.	
0x871B	SDO error: Object can not be mapped to a PDO.	
0x871C	SDO error: The number and length of objects to be mapped exceeds the PDO length.	
0x871D	SDO error: General parameter incompatibility.	
0x871E	SDO error: General internal incompatibility in the device.	
0x871F	SDO error: Access failed due to a hardware error.	
0x8720	SDO error: Data type does not match, length of service parameter does not match.	
0x8721	SDO error: Data type does not match, service parameter too long.	
0x8722	SDO error: Data type does not match, service parameter too short.	
0x8723	SDO error: There is no subindex.	
0x8724	SDO error: Write access - Parameter value out of range.	
0x8725	SDO error: Write access - Parameter value out of high limit	
0x8726	SDO error: Write access - Parameter value out of low limit.	
0x8727	SDO error: Maximum value < Minimum value.	
0x8728	SDO error: General error.	
0x8729	SDO error: Unable to transfer or store data to application.	
0x872A	SDO error: Unable to transfer or store data to application because of local.	

ErrorID	Description	Remark
0x872B	SDO error: Unable to transfer or store data to application because of present device state.	
0x872C	SDO error: The dynamic generation of the object dictionary failed or missing object dictionary.	
0x872D	SDO error: Unknown code.	
0x8750	Wrong value in *LADDR*.	
0x8751	Type other than BYTE in ANY pointer.	
0x8752	There is no PROFIBUS DP module or PROFINET IO device on the address, specified via *LADDR*, from which consistent data can be read.	
0x8753	Access error when accessing a PROFINET IO device.	
0x8754	Slave error on the external PROFIBUS DP slave.	
0x8755	Length of the SFB data does not match the length of the user data.	
0x8756	Error on external PROFIBUS DP slave.	
0x8757	System error on external PROFIBUS DP slave.	
0x8758	The data has not yet been read by the device.	
0x8759	System error on external PROFIBUS DP slave.	
0x875A	No system resources are available.	
0x8799	SDO error: An other error appeared, for more information, see the data of *Info1* and *Info2*.	
0x8888	Internal: BufferIndex error	VMC_AxisControl
0x8A00	Access to unavailable parameter.	VMC_AxisControlSigma_PN
0x8A01	Access to a parameter value that cannot be changed.	VMC_AxisControlSigma_PN
0x8A02	Access with value outside the limits.	VMC_AxisControlSigma_PN
0x8A03	Access to unavailable subindex.	VMC_AxisControlSigma_PN
0x8A04	Access with subindex to non-indexed parameter.	VMC_AxisControlSigma_PN
0x8A05	Invalid data type	VMC_AxisControlSigma_PN
0x8A06	Access with value ≠ 0 when this is not permitted.	VMC_AxisControlSigma_PN
0x8A07	Access to a description element that cannot be changed.	VMC_AxisControlSigma_PN
0x8A09	Access to an unavailable description.	VMC_AxisControlSigma_PN
0x8A0B	Access without rights to change parameters.	VMC_AxisControlSigma_PN
0x8A0F	Access to text array that is not available.	VMC_AxisControlSigma_PN
0x8A11	Access is temporarily not possible.	VMC_AxisControlSigma_PN
0x8A14	Access with a value that is within limits but not currently not possible.	VMC_AxisControlSigma_PN
0x8A15	The length of the current response exceeds the maximum possible length.	VMC_AxisControlSigma_PN
0x8A16	Invalid value respectively value is not supported by the parameter type.	VMC_AxisControlSigma_PN

ErrorID	Description	Remark
0x8A17	Error while write access to parameter: Invalid format	VMC_AxisControlSigma_PN
0x8A18	Error while write access to parameter: Number of parameter does not match the number of elements at the parameter address.	VMC_AxisControlSigma_PN
0x8A19	Error while write access to a digital output, which does not exist.	VMC_AxisControlSigma_PN
0x8A20	Write access to a parameter text, which cannot be changed	VMC_AxisControlSigma_PN
0x8A21	Invalid request ID	VMC_AxisControlSigma_PN
0x8A22	Max number of requested parameters is reached.	VMC_AxisControlSigma_PN
0xC000	Internal error: Status Init is undefined.	Modbus; Init
0xC001	Internal error: Invalid value at parameter *Cmd.ActiveType*.	Modbus V1000
0xC002	Internal Error: Invalid value at parameter *Cmd.State*.	Modbus V1000