VIPA SPEED7 Library

Block library - Simple Motion Control
Table of contents

1 General.......................................................................................................................... 7
   1.1 Copyright © YASKAWA Europe GmbH................................................................. 7
   1.2 About this manual............................................................................................... 8

2 Overview......................................................................................................................... 9

3 Usage Sigma-5/7 EtherCAT.......................................................................................... 11
   3.1 Usage Sigma-5 EtherCAT.................................................................................... 11
      3.1.1 Overview..................................................................................................... 11
      3.1.2 Set the parameters on the drive.................................................................. 11
      3.1.3 Usage in VIPA SPEED7 Studio................................................................. 12
      3.1.4 Usage in Siemens SIMATIC Manager......................................................... 26
      3.1.5 Drive specific blocks.................................................................................. 45
      3.2 Usage Sigma-7S EtherCAT.............................................................................. 47
         3.2.1 Overview................................................................................................. 47
      3.2.2 Set the parameters on the drive.................................................................. 48
      3.2.3 Usage in VIPA SPEED7 Studio................................................................. 49
      3.2.4 Usage in Siemens SIMATIC Manager......................................................... 64
      3.2.5 Drive specific blocks.................................................................................. 83
      3.3 Usage Sigma-7W EtherCAT.............................................................................. 85
         3.3.1 Overview................................................................................................. 85
      3.3.2 Set the parameters on the drive.................................................................. 86
      3.3.3 Usage in VIPA SPEED7 Studio................................................................. 87
      3.3.4 Usage in Siemens SIMATIC Manager......................................................... 104
      3.3.5 Drive specific blocks.................................................................................. 124

4 Usage Sigma-5/7 PROFINET...................................................................................... 128
   4.1 Usage Sigma-5 PROFINET................................................................................ 128
      4.1.1 Overview..................................................................................................... 128
      4.1.2 Set the parameters on the drive.................................................................. 128
      4.1.3 Usage in VIPA SPEED7 Studio................................................................. 129
      4.1.4 Usage in Siemens SIMATIC Manager......................................................... 142
      4.1.5 Usage in Siemens TIA-Portal...................................................................... 155
   4.2 Usage Sigma-7 PROFINET................................................................................ 171
      4.2.1 Overview..................................................................................................... 171
      4.2.2 Set the parameters on the drive.................................................................. 171
      4.2.3 Usage in VIPA SPEED7 Studio................................................................. 172
      4.2.4 Usage in Siemens SIMATIC Manager......................................................... 185
      4.2.5 Usage in Siemens TIA-Portal...................................................................... 198
      4.3 Drive specific blocks....................................................................................... 213
         4.3.1 UDT 890 - VMC_ConfigSigmaPN_REF - Sigma-5/7 PROFINET Data
               structure axis configuration........................................................................ 213
         4.3.2 FB 890 - VMC_AxisControlSigma_PN - control block axis control for
               Sigma-5/7 PROFINET.................................................................................. 213
         4.3.3 FB 891 - VMC_InitSigma_PN - Sigma-5/7 PROFINET initialization........... 217

5 Usage Sigma-5/7 Pulse Train...................................................................................... 220
   5.1 Overview............................................................................................................. 220
   5.2 Set the parameters on the drive........................................................................ 220
   5.3 Wiring.................................................................................................................. 221
   5.4 Usage in VIPA SPEED7 Studio......................................................................... 223
# Table of contents

VIPA SPEED7 Library

5.4.1 Hardware configuration ................................................................. 223
5.4.2 User program .............................................................................. 225
5.5 Usage in Siemens SIMATIC Manager ................................................... 227
  5.5.1 Precondition ............................................................................. 227
  5.5.2 Hardware configuration ............................................................... 228
  5.5.3 User program ........................................................................... 229
5.6 Usage in Siemens TIA Portal .............................................................. 232
  5.6.1 Precondition ............................................................................. 232
  5.6.2 Hardware configuration ............................................................... 232
  5.6.3 User program ........................................................................... 235
5.7 Drive specific block ......................................................................... 238
  5.7.1 FB 875 - VMC_AxisControl_PT - Axis control via Pulse Train ...... 238

6 Usage inverter drive via PWM ............................................................... 247
  6.1 Overview ...................................................................................... 247
  6.2 Set the parameters on the inverter drive ........................................... 247
  6.3 Wiring ........................................................................................... 249
  6.3.1 Connecting the V1000 inputs ....................................................... 249
  6.3.2 Connecting the V1000 outputs ..................................................... 250
  6.4 Usage in VIPA SPEED7 Studio ......................................................... 250
  6.4.1 Hardware configuration ............................................................... 250
  6.4.2 User program ........................................................................... 253
  6.5 Usage in Siemens SIMATIC Manager ................................................... 255
    6.5.1 Precondition ........................................................................... 255
    6.5.2 Hardware configuration ............................................................. 256
    6.5.3 User program .......................................................................... 258
  6.6 Usage in Siemens TIA Portal .............................................................. 260
    6.6.1 Precondition ........................................................................... 260
    6.6.2 Hardware configuration ............................................................. 260
    6.6.3 User program .......................................................................... 263
  6.7 Drive specific block ........................................................................ 266
    6.7.1 FB 885 - VMC_AxisControlV1000_PWM - Axis control over PWM 266

7 Usage inverter drive via Modbus RTU .................................................. 270
  7.1 Overview ...................................................................................... 270
  7.2 Set the parameters on the inverter drive ........................................... 270
  7.3 Wiring ........................................................................................... 272
  7.4 Usage in VIPA SPEED7 Studio ......................................................... 275
    7.4.1 Hardware configuration ............................................................. 275
    7.4.2 User program .......................................................................... 280
  7.5 Usage in Siemens SIMATIC Manager ................................................... 289
    7.5.1 Precondition ........................................................................... 289
    7.5.2 Hardware configuration ............................................................. 290
    7.5.3 User program .......................................................................... 295
  7.6 Usage in Siemens TIA Portal .............................................................. 304
    7.6.1 Precondition ........................................................................... 304
    7.6.2 Hardware configuration ............................................................. 305
    7.6.3 User program .......................................................................... 313
  7.7 Drive specific blocks ...................................................................... 322
    7.7.1 UDT 877 - VMC_ComSlavesRTU_REF - Modbus RTU data structure communication data all slaves ... 322
8 Usage inverter drive via EtherCAT

8.1 Overview

8.2 Set the parameters on the inverter drive

8.3 Wiring

8.4 Usage in VIPA SPEED7 Studio

8.4.1 Hardware configuration

8.4.2 User program

8.5 Usage in Siemens SIMATIC Manager

8.5.1 Precondition

8.5.2 Hardware configuration

8.5.3 User program

8.6 Drive specific blocks

8.6.1 UDT 886 - VMC_ConfigInverterEC_REF - inverter drive EtherCAT Data structure axis configuration

8.6.2 FB 886 - VMC_KernelInverter_EC - inverter drive EtherCAT kernel

8.6.3 FB 887 - VMC_InitInverter_EC - inverter drive EtherCAT initialization

9 Blocks for axis control

9.1 Overview

9.2 Simple motion tasks

9.2.1 UDT 860 - MC_AXIS_REF - Data structure axis data

9.2.2 FB 860 - VMC_AxisControl - Control block axis control

9.3 Complex motion tasks - PLCopen blocks

9.3.1 UDT 860 - MC_AXIS_REF - Data structure axis data

9.3.2 UDT 861 - MC_TRIGGER_REF - Data structure trigger signal

9.3.3 FB 800 - MC_Power - enable/disable axis

9.3.4 FB 801 - MC_Home - home axis

9.3.5 FB 802 - MC_Stop - stop axis

9.3.6 FB 803 - MC_Halt - holding axis

9.3.7 FB 804 - MC_MoveRelative - move axis relative

9.3.8 FB 805 - MC_MoveVelocity - drive axis with constant velocity

9.3.9 FB 808 - MC_MoveAbsolute - move axis to absolute position

9.3.10 FB 811 - MC_Reset - reset axis

9.3.11 FB 812 - MC_ReadStatus - PLCopen status

9.3.12 FB 813 - MC_ReadAxisError - read axis error

9.3.13 FB 814 - MC_ReadParameter - read axis parameter data

9.3.14 FB 815 - MC_WriteParameter - write axis parameter data
9.3.15 FB 816 - MC_ReadActualPosition - reading current axis position........... 392
9.3.16 FB 817 - MC_ReadActualVelocity - read axis velocity.......................... 393
9.3.17 FB 818 - MC_ReadAxisInfo - read additional axis information............... 394
9.3.18 FB 819 - MC_ReadMotionState - read status motion job........................ 396
9.3.19 FB 823 - MC_TouchProbe - record axis position.................................. 398
9.3.20 FB 824 - MC_AbortTrigger - abort recording axis position...................... 400
9.3.21 FB 825 - MC_ReadBoolParameter - read axis boolean parameter data. 401
9.3.22 FB 826 - MC_WriteBoolParameter - write axis boolean parameter data 403
9.3.23 FB 827 - VMC_ReadDWORDParameter - read axis double word parameter data.......................... 405
9.3.24 FB 828 - VMC_WriteDWORDParameter - write axis double word parameter data......................................................... 407
9.3.25 FB 829 - VMC_ReadWordParameter - read axis word parameter data.. 409
9.3.26 FB 830 - VMC_WriteWordParameter - write axis word parameter data.. 411
9.3.27 FB 831 - VMC_ReadByteParameter - read axis byte parameter data.... 413
9.3.28 FB 832 - VMC_WriteByteParameter - write axis byte parameter data.... 415
9.3.29 FB 833 - VMC_ReadDriveParameter - read drive parameter............... 417
9.3.30 FB 834 - VMC_WriteDriveParameter - write drive parameter.............. 419
9.3.31 FB 835 - VMC_HomeInit_LimitSwitch - Initialisation of homing on limit switch............................................................... 421
9.3.32 FB 836 - VMC_HomeInit_HomeSwitch - Initialisation of homing on home switch............................................................. 423
9.3.33 FB 837 - VMC_HomeInit_ZeroPulse - Initialisation of homing on zero puls........................................................................... 426
9.3.34 FB 838 - VMC_HomeInit_SetPosition - Initialisation of homing mode set position.............................................................. 428
9.3.35 PLCopen parameter ........................................................................... 429
9.3.36 VIPA-specific parameter................................................................... 430

10 Controlling the drive via HMI............................................................................ 432
10.1 Overview..................................................................................................... 432
10.2 Create a new project.................................................................................. 433
10.3 Modify the project in Movicon..................................................................... 437
10.4 Commissioning........................................................................................... 448
10.4.1 Transfer project to target device.............................................................. 448
10.4.2 Controlling the VMC_AxisControl via the panel................................. 449

11 States and behavior of the outputs................................................................. 452
11.1 States.......................................................................................................... 452
11.2 Replacement behavior of motion jobs......................................................... 453
11.3 Behavior of the inputs and outputs............................................................. 455

12 ErrorID - Additional error information.......................................................... 457
1 General

1.1 Copyright © YASKAWA Europe GmbH

All Rights Reserved

This document contains proprietary information of YASKAWA and is not to be disclosed or used except in accordance with applicable agreements.

This material is protected by copyright laws. It may not be reproduced, distributed, or altered in any fashion by any entity (either internal or external to YASKAWA) except in accordance with applicable agreements, contracts or licensing, without the express written consent of YASKAWA and the business management owner of the material.

For permission to reproduce or distribute, please contact: YASKAWA Europe GmbH, European Headquarters, Hauptstraße 185, 65760 Eschborn, Germany
Tel.: +49 6196 569 300
Fax.: +49 6196 569 398
Email: info@yaskawa.eu.com
Internet: www.yaskawa.eu.com

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

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

EC conformity declaration

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

Conformity Information

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

Trademarks

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

SPEED7 is a registered trademark of YASKAWA Europe GmbH.

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

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

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

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

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SimpleMotion_S7_V0039.zip</td>
<td>Block library for Siemens SIMATIC Manager.</td>
</tr>
<tr>
<td></td>
<td>For use in VIPA CPUs or S7-300 CPUs from Siemens.</td>
</tr>
<tr>
<td>SimpleMotion_TIA_V0025.zip</td>
<td>Block library for Siemens TIA Portal V14/V15.</td>
</tr>
<tr>
<td></td>
<td>For use in VIPA CPUs or S7-300 CPUs from Siemens.</td>
</tr>
<tr>
<td>SimpleMotion_Movicon0007.zip</td>
<td>Symbol library for Movicon</td>
</tr>
<tr>
<td>Demo_S7_Movicon_V0023.zip</td>
<td>Demo project for Siemens SIMATIC Manager and Movicon.</td>
</tr>
<tr>
<td></td>
<td>For use in VIPA CPUs and TouchPanels or S7-300 CPUs from Siemens.</td>
</tr>
<tr>
<td>Demo_TIA_Movicon_V0018.zip</td>
<td>Demo project for Siemens TIA Portal V14 and Movicon.</td>
</tr>
<tr>
<td></td>
<td>For use in VIPA CPUs and TouchPanels or S7-300 CPUs from Siemens.</td>
</tr>
</tbody>
</table>

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
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
- 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)

<table>
<thead>
<tr>
<th>Servopack Parameter</th>
<th>Address:digit</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pn205</td>
<td>(2205h)</td>
<td>Multiturn Limit Setting</td>
<td>65535</td>
</tr>
<tr>
<td>Pn20E</td>
<td>(220Eh)</td>
<td>Electronic Gear Ratio (Numerator)</td>
<td>1</td>
</tr>
<tr>
<td>Pn210</td>
<td>(2210h)</td>
<td>Electronic Gear Ratio (Denominator)</td>
<td>1</td>
</tr>
<tr>
<td>PnB02</td>
<td>(2701h:01)</td>
<td>Position User Unit (Numerator)</td>
<td>1</td>
</tr>
<tr>
<td>PnB04</td>
<td>(2701h:02)</td>
<td>Position User Unit (Denominator)</td>
<td>1</td>
</tr>
<tr>
<td>PnB06</td>
<td>(2702h:01)</td>
<td>Velocity User Unit (Numerator)</td>
<td>1</td>
</tr>
<tr>
<td>PnB08</td>
<td>(2702h:02)</td>
<td>Velocity User Unit (Denominator)</td>
<td>1</td>
</tr>
</tbody>
</table>
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](http://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**

<table>
<thead>
<tr>
<th>Name</th>
<th>Index</th>
<th>Bit length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status word</td>
<td>0x6041:00</td>
<td>16bit</td>
</tr>
<tr>
<td>Position actual internal value</td>
<td>0x6063:00</td>
<td>32bit</td>
</tr>
<tr>
<td>Position actual value</td>
<td>0x6064:00</td>
<td>32bit</td>
</tr>
<tr>
<td>Torque actual value</td>
<td>0x6077:00</td>
<td>16bit</td>
</tr>
<tr>
<td>Following error actual value</td>
<td>0x60F4:00</td>
<td>32bit</td>
</tr>
<tr>
<td>Modes of operation display</td>
<td>0x6061:00</td>
<td>8bit</td>
</tr>
<tr>
<td>Digital inputs</td>
<td>0x60FD:00</td>
<td>32bit</td>
</tr>
</tbody>
</table>

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**

<table>
<thead>
<tr>
<th>Name</th>
<th>Index</th>
<th>Bit length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Touch probe status</td>
<td>0x60B9:00</td>
<td>16bit</td>
</tr>
<tr>
<td>Touch probe 1 position value</td>
<td>0x60BA:00</td>
<td>32bit</td>
</tr>
<tr>
<td>Touch probe 2 position value</td>
<td>0x60BC:00</td>
<td>32bit</td>
</tr>
<tr>
<td>Velocity actual value</td>
<td>0x606C:00</td>
<td>32bit</td>
</tr>
</tbody>
</table>

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 cannot be activated at the same time!
  - 1601: de-activated
  - 1602: de-activated
  - 1603: de-activated

**Entries**

<table>
<thead>
<tr>
<th>Name</th>
<th>Index</th>
<th>Bit length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control word</td>
<td>0x6040:00</td>
<td>16bit</td>
</tr>
<tr>
<td>Target position</td>
<td>0x607A:00</td>
<td>32bit</td>
</tr>
<tr>
<td>Target velocity</td>
<td>0x60FF:00</td>
<td>32bit</td>
</tr>
<tr>
<td>Modes of operation</td>
<td>0x6060:00</td>
<td>8bit</td>
</tr>
<tr>
<td>Touch probe function</td>
<td>0x60B8:00</td>
<td>16bit</td>
</tr>
</tbody>
</table>

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 cannot be activated at the same time!
  - 1600: de-activated
  - 1602: activated
  - 1603: activated

**Entries**

- Profile velocity: 0x6081:00 \(\rightarrow\) 32 Bit
- Profile acceleration: 0x6083:00 \(\rightarrow\) 32 Bit
- Profile deceleration: 0x6084:00 \(\rightarrow\) 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'.

<table>
<thead>
<tr>
<th>Device Editor</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PDO assign</strong></td>
</tr>
<tr>
<td><strong>Inputs</strong></td>
</tr>
<tr>
<td>☑ 1st Transmit PDO mapping</td>
</tr>
<tr>
<td>☑ 2nd Transmit PDO mapping</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Device Editor</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Distributed Clock</strong></td>
</tr>
<tr>
<td>Distributed Clock</td>
</tr>
</tbody>
</table>

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’

<table>
<thead>
<tr>
<th>Device Editor</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
</tr>
<tr>
<td>I/O addresses</td>
</tr>
<tr>
<td>Nr.</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

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

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_ConfigSigma5EC_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_InitSigma5_EC
  - The Init 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_KernelSigma5_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 Programming

Copy blocks into project

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

   ![Project tree screenshot]

   ⇒ 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]

<table>
<thead>
<tr>
<th>Addr...</th>
<th>Name</th>
<th>Data type...</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>Config</td>
<td>UDT [870]</td>
</tr>
<tr>
<td>...</td>
<td>Axis</td>
<td>UDT [860]</td>
</tr>
</tbody>
</table>

#### 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  

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

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
MinPosition :=1.048500e+003
MinUserPosition :="InitS5EC1_MinUserPos"
MaxPosition :=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  

CALL "VMC_KernelSigma5_EC", "DI_KernelSgm5ETC01"

Init :="KernelS5EC1_Init"
Config :="Axis01".Config
Axis :="Axis01".Axis
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.

- 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_DirectionPositive"
  - DirectionNegative := "AxCtrl1_DirectionNegative"
  - 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
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:

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

<table>
<thead>
<tr>
<th>Slot</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CPU 315-2 PN/DP</td>
</tr>
<tr>
<td>X1</td>
<td>MPI/DP</td>
</tr>
<tr>
<td>X2</td>
<td>PN-IO</td>
</tr>
<tr>
<td>X2...</td>
<td>Port 1</td>
</tr>
<tr>
<td>X2...</td>
<td>Port 2</td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

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 PROFINET 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 è Additional field devices è I/O è 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

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 è Additional field devices è I/O è 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.

2. 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.

3. Open in the SPEED7 EtherCAT Manager via ‘File ➔ ESI Manager’ the dialogue window ‘ESI Manager’.

4. 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.

5. 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**

<table>
<thead>
<tr>
<th>Name</th>
<th>Index</th>
<th>Bit length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status word</td>
<td>0x6041:00</td>
<td>16bit</td>
</tr>
<tr>
<td>Position actual internal value</td>
<td>0x6063:00</td>
<td>32bit</td>
</tr>
<tr>
<td>Position actual value</td>
<td>0x6064:00</td>
<td>32bit</td>
</tr>
<tr>
<td>Torque actual value</td>
<td>0x6077:00</td>
<td>16bit</td>
</tr>
<tr>
<td>Following error actual value</td>
<td>0x60F4:00</td>
<td>32bit</td>
</tr>
<tr>
<td>Modes of operation display</td>
<td>0x6061:00</td>
<td>8bit</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>8bit</td>
</tr>
<tr>
<td>Digital inputs</td>
<td>0x60FD:00</td>
<td>32bit</td>
</tr>
</tbody>
</table>

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**

<table>
<thead>
<tr>
<th>Name</th>
<th>Index</th>
<th>Bit length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Touch probe status</td>
<td>0x60B9:00</td>
<td>16bit</td>
</tr>
<tr>
<td>Touch probe 1 position value</td>
<td>0x60BA:00</td>
<td>32bit</td>
</tr>
<tr>
<td>Touch probe 2 position value</td>
<td>0x60BC:00</td>
<td>32bit</td>
</tr>
<tr>
<td>Velocity actual value</td>
<td>0x606C:00</td>
<td>32bit</td>
</tr>
</tbody>
</table>

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**

<table>
<thead>
<tr>
<th>Name</th>
<th>Index</th>
<th>Bit length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control word</td>
<td>0x6040:00</td>
<td>16bit</td>
</tr>
<tr>
<td>Target position</td>
<td>0x607A:00</td>
<td>32bit</td>
</tr>
<tr>
<td>Target velocity</td>
<td>0x60FF:00</td>
<td>32bit</td>
</tr>
<tr>
<td>Modes of operation</td>
<td>0x6060:00</td>
<td>8bit</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>8bit</td>
</tr>
<tr>
<td>Touch probe function</td>
<td>0x60B8:00</td>
<td>16bit</td>
</tr>
</tbody>
</table>

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**

<table>
<thead>
<tr>
<th>Name</th>
<th>Index</th>
<th>Bit length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profile velocity</td>
<td>0x6081:00</td>
<td>32bit</td>
</tr>
<tr>
<td>Profile acceleration</td>
<td>0x6083:00</td>
<td>32bit</td>
</tr>
<tr>
<td>Profile deceleration</td>
<td>0x6084:00</td>
<td>32bit</td>
</tr>
</tbody>
</table>

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’

![Device Editor Diagram]

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_ConfigSigma5EC_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_InitSigma5_EC**
  - The **Init** 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_KernelSigma5_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
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

<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>Typ</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Config</td>
<td>&quot;VMC_ConfigSigma5EC_REF&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Axis</td>
<td>&quot;MC_AXIS_REF&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>END_STRUCT</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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
  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.

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
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

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.*

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

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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Release of initialization</td>
</tr>
<tr>
<td>Logical address</td>
<td>INPUT</td>
<td>INT</td>
<td>Start address of the PDO input data</td>
</tr>
<tr>
<td>InputsStartAddressPDO</td>
<td>INPUT</td>
<td>INT</td>
<td>Start address of the input PDOs</td>
</tr>
<tr>
<td>OutputsStartAddressPDO</td>
<td>INPUT</td>
<td>INT</td>
<td>Start address of the output PDOs</td>
</tr>
<tr>
<td>Parameter</td>
<td>Declaration</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------</td>
<td>-----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>EncoderType</td>
<td>INPUT</td>
<td>INT</td>
<td>Encoder type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ 1: Absolute encoder</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ 2: Incremental encoder</td>
</tr>
<tr>
<td>EncoderResolutionBits</td>
<td>INPUT</td>
<td>INT</td>
<td>Number of bits corresponding to one encoder revolution. Default: 20</td>
</tr>
<tr>
<td>FactorPosition</td>
<td>INPUT</td>
<td>REAL</td>
<td>Factor for converting the position of user units [u] into drive units [increments] and back.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>It's valid: ( p_{\text{increments}} = p_{[u]} \times \text{FactorPosition} )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Please consider the factor which can be specified on the drive via the objects 0x2701: 1 and 0x2701: 2. This should be 1.</td>
</tr>
<tr>
<td>Velocity Factor</td>
<td>INPUT</td>
<td>REAL</td>
<td>Factor for converting the speed of user units [u/s] into drive units [increments/s] and back.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>It's valid: ( v_{\text{increments/s}} = v_{[u/s]} \times \text{FactorVelocity} )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>FactorAcceleration</td>
<td>INPUT</td>
<td>REAL</td>
<td>Factor to convert the acceleration of user units ([u/s^2]) in drive units ([10^{-4} \times \text{increments/s}^2]) and back.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>It's valid: ( a_{\text{increments/s}^2} = a_{[u/s^2]} \times \text{FactorAcceleration} )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>OffsetPosition</td>
<td>INPUT</td>
<td>REAL</td>
<td>Offset for the zero position [u].</td>
</tr>
<tr>
<td>MaxVelocityApp</td>
<td>INPUT</td>
<td>REAL</td>
<td>Maximum application speed [u/s].</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The command inputs are checked to the maximum value before execution.</td>
</tr>
<tr>
<td>MaxAccelerationApp</td>
<td>INPUT</td>
<td>REAL</td>
<td>Maximum acceleration of the application ([u/s^2]).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The command inputs are checked to the maximum value before execution.</td>
</tr>
<tr>
<td>MaxDecelerationApp</td>
<td>INPUT</td>
<td>REAL</td>
<td>Maximum application deceleration ([u/s^2]).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The command inputs are checked to the maximum value before execution.</td>
</tr>
<tr>
<td>MaxPosition</td>
<td>INPUT</td>
<td>REAL</td>
<td>Maximum position for monitoring the software limits [u].</td>
</tr>
<tr>
<td>MinPosition</td>
<td>INPUT</td>
<td>REAL</td>
<td>Minimum position for monitoring the software limits [u].</td>
</tr>
<tr>
<td>EnableMaxPosition</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Monitoring maximum position</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ TRUE: Activates the monitoring of the maximum position.</td>
</tr>
<tr>
<td>EnableMinPosition</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Monitoring minimum position</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ TRUE: Activation of the monitoring of the minimum position.</td>
</tr>
<tr>
<td>MinUserPosition</td>
<td>OUTPUT</td>
<td>REAL</td>
<td>Minimum user position based on the minimum encoder value of 0x80000000 and the FactorPosition [u].</td>
</tr>
<tr>
<td>Parameter</td>
<td>Declaration</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MaxUserPosition</td>
<td>OUTPUT</td>
<td>REAL</td>
<td>Maximum user position based on the maximum encoder value of 0x7FFFFFFF and the FactorPosition [u].</td>
</tr>
<tr>
<td>Valid</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Initialization</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TRUE: Initialization is valid.</td>
</tr>
<tr>
<td>Error</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TRUE: An error has occurred. Additional error information can be found in the parameter ErrorID. The axis is disabled.</td>
</tr>
<tr>
<td>ErrorID</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>Additional error information</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Chap. 12 'ErrorID - Additional error information' page 457</td>
</tr>
<tr>
<td>Config</td>
<td>IN_OUT</td>
<td>UDT870</td>
<td>Data structure for transferring axis-dependent configuration data to the AxisKernel.</td>
</tr>
<tr>
<td>Axis</td>
<td>IN_OUT</td>
<td>MC_AXIS_REF</td>
<td>Data structure for transferring axis-dependent information to the AxisKernel and PLCopen blocks.</td>
</tr>
</tbody>
</table>

### 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
- 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

**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)

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

*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.
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**

<table>
<thead>
<tr>
<th>Name</th>
<th>Index</th>
<th>Bit length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status word</td>
<td>0x6041:00</td>
<td>16bit</td>
</tr>
<tr>
<td>Position actual internal value</td>
<td>0x6063:00</td>
<td>32bit</td>
</tr>
<tr>
<td>Position actual value</td>
<td>0x6064:00</td>
<td>32bit</td>
</tr>
<tr>
<td>Torque actual value</td>
<td>0x6077:00</td>
<td>16bit</td>
</tr>
<tr>
<td>Following error actual value</td>
<td>0x60F4:00</td>
<td>32bit</td>
</tr>
<tr>
<td>Modes of operation display</td>
<td>0x6061:00</td>
<td>8bit</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>8bit</td>
</tr>
<tr>
<td>Digital inputs</td>
<td>0x60FD:00</td>
<td>32bit</td>
</tr>
</tbody>
</table>

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**

<table>
<thead>
<tr>
<th>Name</th>
<th>Index</th>
<th>Bit length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Touch probe status</td>
<td>0x60B9:00</td>
<td>16bit</td>
</tr>
<tr>
<td>Touch probe 1 position value</td>
<td>0x60BA:00</td>
<td>32bit</td>
</tr>
<tr>
<td>Touch probe 2 position value</td>
<td>0x60BC:00</td>
<td>32bit</td>
</tr>
<tr>
<td>Velocity actual value</td>
<td>0x606C:00</td>
<td>32bit</td>
</tr>
</tbody>
</table>

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**

<table>
<thead>
<tr>
<th>Name</th>
<th>Index</th>
<th>Bit length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control word</td>
<td>0x6040:00</td>
<td>16bit</td>
</tr>
<tr>
<td>Target position</td>
<td>0x607A:00</td>
<td>32bit</td>
</tr>
<tr>
<td>Target velocity</td>
<td>0x60FF:00</td>
<td>32bit</td>
</tr>
<tr>
<td>Modes of operation</td>
<td>0x6060:00</td>
<td>8bit</td>
</tr>
<tr>
<td>Touch probe function</td>
<td>0x60B8:00</td>
<td>16bit</td>
</tr>
</tbody>
</table>

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**

<table>
<thead>
<tr>
<th>Name</th>
<th>Index</th>
<th>Bit length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profile velocity</td>
<td>0x6081:00</td>
<td>32Bit</td>
</tr>
<tr>
<td>Profile acceleration</td>
<td>0x6083:00</td>
<td>32Bit</td>
</tr>
<tr>
<td>Profile deceleration</td>
<td>0x6084:00</td>
<td>32Bit</td>
</tr>
</tbody>
</table>

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.
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_ConfigSigma7EC_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_InitSigma7S_EC**
- The **Init** 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_KernelSigma7_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’.

   ![Add organisation block](image)

   - 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.

   **Data block structure**

<table>
<thead>
<tr>
<th>Addr</th>
<th>Name</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>Config</td>
<td>UDT [872]</td>
</tr>
<tr>
<td>...</td>
<td>Axis</td>
<td>UDT [860]</td>
</tr>
</tbody>
</table>
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
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
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.

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
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.
The installation of the PROFINET IO device ‘VIPA SLIO CPU’ happens in the hardware catalog with the following approach:

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’.

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’.

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

<table>
<thead>
<tr>
<th>Slot</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CPU 315-2 PN/DP</td>
</tr>
<tr>
<td></td>
<td>X1 MPI/DP</td>
</tr>
<tr>
<td></td>
<td>X2 PN-IO</td>
</tr>
<tr>
<td></td>
<td>X2... Port 1</td>
</tr>
<tr>
<td></td>
<td>X2... Port 2</td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

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 PROFINET 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.
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

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 128 byte
- Out 128 byte

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.

2. 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.

3. Open in the SPEED7 EtherCAT Manager via ‘File ➔ ESI Manager’ the dialogue window ‘ESI Manager’.

4. 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.

5. 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.
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**

<table>
<thead>
<tr>
<th>Name</th>
<th>Index</th>
<th>Bit length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status word</td>
<td>0x6041:00</td>
<td>16bit</td>
</tr>
<tr>
<td>Position actual internal value</td>
<td>0x6063:00</td>
<td>32bit</td>
</tr>
<tr>
<td>Position actual value</td>
<td>0x6064:00</td>
<td>32bit</td>
</tr>
<tr>
<td>Torque actual value</td>
<td>0x6077:00</td>
<td>16bit</td>
</tr>
<tr>
<td>Following error actual value</td>
<td>0x60F4:00</td>
<td>32bit</td>
</tr>
<tr>
<td>Modes of operation display</td>
<td>0x6061:00</td>
<td>8bit</td>
</tr>
<tr>
<td>Digital inputs</td>
<td>0x60FD:00</td>
<td>32bit</td>
</tr>
</tbody>
</table>

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**

<table>
<thead>
<tr>
<th>Name</th>
<th>Index</th>
<th>Bit length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Touch probe status</td>
<td>0x60B9:00</td>
<td>16bit</td>
</tr>
<tr>
<td>Touch probe 1 position value</td>
<td>0x60BA:00</td>
<td>32bit</td>
</tr>
<tr>
<td>Touch probe 2 position value</td>
<td>0x60BC:00</td>
<td>32bit</td>
</tr>
<tr>
<td>Velocity actual value</td>
<td>0x606C:00</td>
<td>32bit</td>
</tr>
</tbody>
</table>

Close the dialog ‘Edit PDO’ with [OK].
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**

<table>
<thead>
<tr>
<th>Name</th>
<th>Index</th>
<th>Bit length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control word</td>
<td>0x6040:00</td>
<td>16bit</td>
</tr>
<tr>
<td>Target position</td>
<td>0x607A:00</td>
<td>32bit</td>
</tr>
<tr>
<td>Target velocity</td>
<td>0x60FF:00</td>
<td>32bit</td>
</tr>
<tr>
<td>Modes of operation</td>
<td>0x6060:00</td>
<td>8bit</td>
</tr>
<tr>
<td>Touch probe function</td>
<td>0x60B8:00</td>
<td>16bit</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Index</th>
<th>Bit length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profile velocity</td>
<td>0x6081:00</td>
<td>32bit</td>
</tr>
<tr>
<td>Profile acceleration</td>
<td>0x6083:00</td>
<td>32bit</td>
</tr>
<tr>
<td>Profile deceleration</td>
<td>0x6084:00</td>
<td>32bit</td>
</tr>
</tbody>
</table>

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
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_ConfigSigma7EC_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_InitSigma7S_EC**
- The Init 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_KernelSigma7_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**

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.

<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Config</td>
<td>&quot;VMC_ConfigSigma7EC_REF&quot;</td>
</tr>
<tr>
<td></td>
<td>Axis</td>
<td>&quot;MC_AXIS_REF&quot;</td>
</tr>
<tr>
<td></td>
<td>END_STRUCT</td>
<td></td>
</tr>
</tbody>
</table>
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

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

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
MaxPosition           :=1.048500e+003
MinPosition           :=-1.048514e+003
EnableMaxPosition     :=TRUE
EnableMinPosition     :=TRUE
MinUserPosition       :="InitS5SEC1_MinUserPos"
MaxUserPosition       :="InitS5SEC1_MaxUserPos"
Valid                :="InitS5SEC1_Valid"
Error                :="InitS5SEC1_Error"
ErrorID              :="InitS5SEC1_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

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

For complex motion tasks, you can use the PLCopen blocks. Please specify the reference to the corresponding PLCopen 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
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.*

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

#### 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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Release of initialization</td>
</tr>
<tr>
<td>Logical address</td>
<td>INPUT</td>
<td>INT</td>
<td>Start address of the PDO input data</td>
</tr>
<tr>
<td>InputsStartAddressPDO</td>
<td>INPUT</td>
<td>INT</td>
<td>Start address of the input PDOs</td>
</tr>
<tr>
<td>OutputsStartAddressPDO</td>
<td>INPUT</td>
<td>INT</td>
<td>Start address of the output PDOs</td>
</tr>
<tr>
<td>Parameter</td>
<td>Declaration</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------------</td>
<td>-----------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>EncoderType</td>
<td>INPUT</td>
<td>INT</td>
<td>Encoder type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: Absolute encoder</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2: Incremental encoder</td>
</tr>
<tr>
<td>EncoderResolutionBits</td>
<td>INPUT</td>
<td>INT</td>
<td>Number of bits corresponding to one encoder revolution. Default: 20</td>
</tr>
<tr>
<td>FactorPosition</td>
<td>INPUT</td>
<td>REAL</td>
<td>Factor for converting the position of user units [u] into drive units [increments] and back.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>It's valid:  ( p_{\text{increments}} = p_{\text{u}} \times \text{FactorPosition} )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Please consider the factor which can be specified on the drive via the objects 0x2701: 1 and 0x2701: 2. This should be 1.</td>
</tr>
<tr>
<td>Velocity Factor</td>
<td>INPUT</td>
<td>REAL</td>
<td>Factor for converting the speed of user units [u/s] into drive units [increments/s] and back.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>It's valid:  ( v_{\text{increments/s}} = v_{\text{u/s}} \times \text{FactorVelocity} )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>FactorAcceleration</td>
<td>INPUT</td>
<td>REAL</td>
<td>Factor to convert the acceleration of user units [u/s^2] in drive units [10^{-4} x increments/s^2] and back.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>It's valid:  ( a_{\text{increments/s^2}} = a_{\text{u/s^2}} \times \text{FactorAcceleration} )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>OffsetPosition</td>
<td>INPUT</td>
<td>REAL</td>
<td>Offset for the zero position [u].</td>
</tr>
<tr>
<td>MaxVelocityApp</td>
<td>INPUT</td>
<td>REAL</td>
<td>Maximum application speed [u/s].</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The command inputs are checked to the maximum value before execution.</td>
</tr>
<tr>
<td>MaxAccelerationApp</td>
<td>INPUT</td>
<td>REAL</td>
<td>Maximum acceleration of application [u/s^2].</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The command inputs are checked to the maximum value before execution.</td>
</tr>
<tr>
<td>MaxDecelerationApp</td>
<td>INPUT</td>
<td>REAL</td>
<td>Maximum application delay [u/s^2].</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The command inputs are checked to the maximum value before execution.</td>
</tr>
<tr>
<td>MaxPosition</td>
<td>INPUT</td>
<td>REAL</td>
<td>Maximum position for monitoring the software limits [u].</td>
</tr>
<tr>
<td>MinPosition</td>
<td>INPUT</td>
<td>REAL</td>
<td>Minimum position for monitoring the software limits [u].</td>
</tr>
<tr>
<td>EnableMaxPosition</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Monitoring maximum position</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TRUE: Activates the monitoring of the maximum position.</td>
</tr>
<tr>
<td>EnableMinPosition</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Monitoring minimum position</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TRUE: Activation of the monitoring of the minimum position.</td>
</tr>
<tr>
<td>MinUserPosition</td>
<td>OUTPUT</td>
<td>REAL</td>
<td>Minimum user position based on the minimum encoder value of 0x80000000 and the FactorPosition [u].</td>
</tr>
</tbody>
</table>
### 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
- 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)**

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

**Axis 2 - Module 2 (24Bit Encoder)**

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

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.
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.

   ![Project tree screenshot](image)

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](http://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 cannot be edited because of the default settings. By de-activating already activated PDOs, you can release the processing of locked PDOs.

<table>
<thead>
<tr>
<th>PDO assign</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Inputs</strong></td>
<td><strong>Outputs</strong></td>
</tr>
<tr>
<td>☐ Module 1 (SGD7). 1st Transmit PDO mapping</td>
<td>☐ Module 1 (SGD7). 1st Receive PDO mapping</td>
</tr>
<tr>
<td>☐ Module 1 (SGD7). 2nd Transmit PDO mapping</td>
<td>☐ Module 1 (SGD7). 2nd Receive PDO mapping</td>
</tr>
<tr>
<td>☐ Module 2 (SGD7). 1st Transmit PDO mapping</td>
<td>☐ Module 2 (SGD7). 1st Receive PDO mapping</td>
</tr>
<tr>
<td>☐ Module 2 (SGD7). 2nd Transmit PDO mapping</td>
<td>☐ Module 2 (SGD7). 2nd Receive PDO mapping</td>
</tr>
</tbody>
</table>

⇒ 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**

<table>
<thead>
<tr>
<th>Module 1 (SGD7). 1st Transmit PDO mapping</th>
<th>Module 2 (SGD7). 1st Transmit PDO mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name: Module 1 (SGD7). 1st Transmit PDO mapping</td>
<td>Name: Module 2 (SGD7). 1st Transmit PDO mapping</td>
</tr>
<tr>
<td>Index: 0x1A00</td>
<td>Index: 0x1A10</td>
</tr>
<tr>
<td>Flags: Everything de-activated</td>
<td>Flags: Everything de-activated</td>
</tr>
<tr>
<td>Direction TxPdo (Input): activated</td>
<td>Direction TxPdo (Input): activated</td>
</tr>
<tr>
<td>Exclude: 1A01: de-activated</td>
<td>1A11: de-activated</td>
</tr>
</tbody>
</table>

Please note these settings, otherwise the PDO mappings can not be activated at the same time!

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

**Inputs: 2nd Transmit PDO**

<table>
<thead>
<tr>
<th>Module 1 (SGD7). 2nd Transmit PDO mapping</th>
<th>Module 2 (SGD7). 2nd Transmit PDO mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name: Module 1 (SGD7). 2nd Transmit PDO mapping</td>
<td>Name: Module 2 (SGD7). 2nd Transmit PDO mapping</td>
</tr>
<tr>
<td>Index: 0x1A01</td>
<td>Index: 0x1A11</td>
</tr>
<tr>
<td>Flags: Everything de-activated</td>
<td>Flags: Everything de-activated</td>
</tr>
<tr>
<td>Direction TxPdo (Input): activated</td>
<td>Direction TxPdo (Input): activated</td>
</tr>
<tr>
<td>Exclude: 1A00, 1A02, 1A03: de-activated</td>
<td>1A10, 1A12, 1A13: de-activated</td>
</tr>
</tbody>
</table>

Please note these settings, otherwise the PDO mappings can not be activated at the same time!

<table>
<thead>
<tr>
<th>Entries</th>
<th>Module 1 (axis 1)</th>
<th>Module 2 (axis 2)</th>
<th>Bit length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Index</td>
<td>Index</td>
<td></td>
</tr>
<tr>
<td>Touch probe status</td>
<td>0x60B9:00</td>
<td>0x68B9:00</td>
<td>16bit</td>
</tr>
<tr>
<td>Touch probe 1 position value</td>
<td>0x60BA:00</td>
<td>0x68BA:00</td>
<td>32bit</td>
</tr>
<tr>
<td>Touch probe 2 position value</td>
<td>0x60BC:00</td>
<td>0x68BC:00</td>
<td>32bit</td>
</tr>
<tr>
<td>Velocity actual value</td>
<td>0x606C:00</td>
<td>0x686C:00</td>
<td>32bit</td>
</tr>
</tbody>
</table>
5. Perform the following settings for the Receive PDOS:

### Outputs: 1st Receive PDO

<table>
<thead>
<tr>
<th>Module 1 (SGD7). 1st Receive PDO</th>
<th>Module 2 (SGD7). 1st Receive PDO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name: Module 1 (SGD7). 1st Receive PDO mapping</td>
<td>Name: Module 2 (SGD7). 1st Receive PDO mapping</td>
</tr>
<tr>
<td>Index: 0x1600</td>
<td>Index: 0x1610</td>
</tr>
<tr>
<td>Flags: Everything de-activated</td>
<td></td>
</tr>
<tr>
<td>Direction RxPdo (Output): activated</td>
<td></td>
</tr>
<tr>
<td>Exclude: 1601, 1602, 1603: de-activated</td>
<td>1611, 1612, 1613: de-activated</td>
</tr>
</tbody>
</table>

Please note these settings, otherwise the PDO mappings can not be activated at the same time!

#### Entries

<table>
<thead>
<tr>
<th>Name</th>
<th>Module 1 (axis 1)</th>
<th>Module 2 (axis 2)</th>
<th>Bit length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control word</td>
<td>0x6040:00</td>
<td>0x6840:00</td>
<td>16bit</td>
</tr>
<tr>
<td>Target position</td>
<td>0x607A:00</td>
<td>0x687A:00</td>
<td>32bit</td>
</tr>
<tr>
<td>Target velocity</td>
<td>0x60FF:00</td>
<td>0x68FF:00</td>
<td>32bit</td>
</tr>
<tr>
<td>Modes of operation</td>
<td>0x6060:00</td>
<td>0x6860:00</td>
<td>8bit</td>
</tr>
<tr>
<td>Touch probe function</td>
<td>0x60B8:00</td>
<td>0x68B8:00</td>
<td>16bit</td>
</tr>
</tbody>
</table>

### Outputs: 2nd Receive PDO

<table>
<thead>
<tr>
<th>Module 1 (SGD7). 2nd Receive PDO</th>
<th>Module 2 (SGD7). 2nd Receive PDO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name: Module 1 (SGD7). 2nd Receive PDO mapping</td>
<td>Name: Module 2 (SGD7). 2nd Receive PDO mapping</td>
</tr>
<tr>
<td>Index: 0x1601</td>
<td>Index: 0x1611</td>
</tr>
<tr>
<td>Flags: Everything de-activated</td>
<td></td>
</tr>
<tr>
<td>Direction RxPdo (Output): activated</td>
<td></td>
</tr>
<tr>
<td>Exclude: 1600, 1602, 1603: de-activated</td>
<td>1610, 1612, 1613: de-activated</td>
</tr>
</tbody>
</table>

Please note these settings, otherwise the PDO mappings can not be activated at the same time!

#### Entries

<table>
<thead>
<tr>
<th>Name</th>
<th>Module 1 (axis 1)</th>
<th>Module 2 (axis 2)</th>
<th>Bit length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profile velocity</td>
<td>0x6081:00</td>
<td>0x6881:00</td>
<td>32bit</td>
</tr>
<tr>
<td>Profile acceleration</td>
<td>0x6083:00</td>
<td>0x6883:00</td>
<td>32bit</td>
</tr>
<tr>
<td>Profile deceleration</td>
<td>0x6084:00</td>
<td>0x6884:00</td>
<td>32bit</td>
</tr>
</tbody>
</table>
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’.

<table>
<thead>
<tr>
<th>Input PDOs</th>
<th>Output PDOs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module 1 (SGD7): 1st Transmit PDO mapping</td>
<td>Module 1 (SGD7): 1st Receive PDO mapping</td>
</tr>
<tr>
<td>Module 1 (SGD7): 2nd Transmit PDO mapping</td>
<td>Module 1 (SGD7): 2nd Receive PDO mapping</td>
</tr>
<tr>
<td>Module 2 (SGD7): 1st Transmit PDO mapping</td>
<td>Module 2 (SGD7): 1st Receive PDO mapping</td>
</tr>
<tr>
<td>Module 2 (SGD7): 2nd Transmit PDO mapping</td>
<td>Module 2 (SGD7): 2nd Receive PDO mapping</td>
</tr>
</tbody>
</table>

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’.

   ![EtherCAT Manager](image)

   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.
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_ConfigSigma7EC_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_InitSigma7W_EC**
- The Init 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_KernelSigma7_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_KernelSigma7_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 Programming

#### Copy blocks into project

1. Click in the Project tree within the CPU at ‘PLC program’, ‘Program blocks’ at ‘Add New block’:

   ![Add organisation block dialog](image)

   - 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.
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**

<table>
<thead>
<tr>
<th>Addr...</th>
<th>Name</th>
<th>Data type...</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>Config</td>
<td>UDT</td>
</tr>
<tr>
<td>...</td>
<td>Axis</td>
<td>UDT</td>
</tr>
</tbody>
</table>

---

### 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.

<table>
<thead>
<tr>
<th>Addr...</th>
<th>Name</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>Config</td>
<td>UDT</td>
</tr>
<tr>
<td>...</td>
<td>Axis</td>
<td>UDT</td>
</tr>
</tbody>
</table>
Open OB 1 and program the following FB calls with associated DBs:

FB 874 - VMC_InitSigma7W_EC, DB 874

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

- 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_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_MaxPosition := 1.048500e+003
  - M2_MinPosition := -1.048514e+003
  - M2_EnableMaxPosition := TRUE
  - M2_EnableMinPosition := TRUE

Valid := "InitS7WEC1_Valid"
Error := "InitS7WEC1_Error"
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
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.

```plaintext
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
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:

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

<table>
<thead>
<tr>
<th>Slot</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CPU 315-2 PN/DP</td>
</tr>
<tr>
<td></td>
<td>X1 MPI/DP</td>
</tr>
<tr>
<td></td>
<td>X2 PN-IO</td>
</tr>
<tr>
<td></td>
<td>X2... Port 1</td>
</tr>
<tr>
<td></td>
<td>X2... Port 2</td>
</tr>
</tbody>
</table>

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.

1. 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**

<table>
<thead>
<tr>
<th>Slot</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CPU ... PN-IO</td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>343-1EX30</td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

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’**

<table>
<thead>
<tr>
<th>Slot</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CPU ... PN-IO</td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Slot</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CPU</td>
</tr>
<tr>
<td>X</td>
<td>PN-I/O</td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

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.

2. 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.

3. Open in the SPEED7 EtherCAT Manager via ‘File ➔ ESI Manager’ the dialogue window ‘ESI Manager’.

4. 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.

5. 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 cannot 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.
Perform the following settings for the Transmit PDOs:

**Inputs: 1st Transmit PDO**

<table>
<thead>
<tr>
<th>Module 1 (SGD7). 1st Transmit PDO mapping</th>
<th>Module 2 (SGD7). 1st Transmit PDO mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name: Module 1 (SGD7). 1st Transmit PDO mapping</td>
<td>Name: Module 2 (SGD7). 1st Transmit PDO mapping</td>
</tr>
<tr>
<td>Index: 0x1A00</td>
<td>Index: 0x1A10</td>
</tr>
<tr>
<td>Flags: Everything de-activated</td>
<td></td>
</tr>
<tr>
<td>Direction TxPdo (Input): activated</td>
<td></td>
</tr>
<tr>
<td>Exclude: 1A01: de-activated</td>
<td>1A11: de-activated</td>
</tr>
</tbody>
</table>

Please note these settings, otherwise the PDO mappings can not be activated at the same time!

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

**Inputs: 2nd Transmit PDO**

<table>
<thead>
<tr>
<th>Module 1 (SGD7). 2nd Transmit PDO mapping</th>
<th>Module 2 (SGD7). 2nd Transmit PDO mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name: Module 1 (SGD7). 2nd Transmit PDO mapping</td>
<td>Name: Module 2 (SGD7). 2nd Transmit PDO mapping</td>
</tr>
<tr>
<td>Index: 0x1A01</td>
<td>Index: 0x1A11</td>
</tr>
<tr>
<td>Flags: Everything de-activated</td>
<td></td>
</tr>
<tr>
<td>Direction TxPdo (Input): activated</td>
<td></td>
</tr>
<tr>
<td>Exclude: 1A00, 1A02, 1A03: de-activated</td>
<td>1A10, 1A12, 1A13: de-activated</td>
</tr>
</tbody>
</table>

Please note these settings, otherwise the PDO mappings can not be activated at the same time!

<table>
<thead>
<tr>
<th>Entries</th>
<th>Module 1 (axis 1)</th>
<th>Module 2 (axis 2)</th>
<th>Bit length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Index</td>
<td>Index</td>
<td></td>
</tr>
<tr>
<td>Touch probe status</td>
<td>0x60B9:00</td>
<td>0x68B9:00</td>
<td>16bit</td>
</tr>
<tr>
<td>Touch probe 1 position value</td>
<td>0x60BA:00</td>
<td>0x68BA:00</td>
<td>32bit</td>
</tr>
<tr>
<td>Touch probe 2 position value</td>
<td>0x60BC:00</td>
<td>0x68BC:00</td>
<td>32bit</td>
</tr>
<tr>
<td>Velocity actual value</td>
<td>0x606C:00</td>
<td>0x686C:00</td>
<td>32bit</td>
</tr>
</tbody>
</table>
13. Perform the following settings for the Receive PDOs:

### Outputs: 1st Receive PDO

<table>
<thead>
<tr>
<th>Module 1 (SGD7). 1st Receive PDO</th>
<th>Module 2 (SGD7). 1st Receive PDO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name: Module 1 (SGD7). 1st Receive PDO mapping</td>
<td>Name: Module 2 (SGD7). 1st Receive PDO mapping</td>
</tr>
<tr>
<td>Index: 0x1600</td>
<td>Index: 0x1610</td>
</tr>
<tr>
<td>Flags: Everything de-activated</td>
<td></td>
</tr>
<tr>
<td>Direction RxPdo (Output): activated</td>
<td></td>
</tr>
<tr>
<td>Exclude: 1601, 1602, 1603: de-activated</td>
<td>1611, 1612, 1613: de-activated</td>
</tr>
</tbody>
</table>

Please note these settings, otherwise the PDO mappings can not be activated at the same time!

<table>
<thead>
<tr>
<th>Entries</th>
<th>Module 1 (axis 1)</th>
<th>Module 2 (axis 2)</th>
<th>Bit length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Index</td>
<td>Index</td>
<td></td>
</tr>
<tr>
<td>Control word</td>
<td>0x6040:00</td>
<td>0x6840:00</td>
<td>16bit</td>
</tr>
<tr>
<td>Target position</td>
<td>0x607A:00</td>
<td>0x687A:00</td>
<td>32bit</td>
</tr>
<tr>
<td>Target velocity</td>
<td>0x60FF:00</td>
<td>0x68FF:00</td>
<td>32bit</td>
</tr>
<tr>
<td>Modes of operation</td>
<td>0x6060:00</td>
<td>0x6860:00</td>
<td>8bit</td>
</tr>
<tr>
<td>Touch probe function</td>
<td>0x60B8:00</td>
<td>0x68B8:00</td>
<td>16bit</td>
</tr>
</tbody>
</table>

### Outputs: 2nd Receive PDO

<table>
<thead>
<tr>
<th>Module 1 (SGD7). 2nd Receive PDO</th>
<th>Module 2 (SGD7). 2nd Receive PDO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name: Module 1 (SGD7). 2nd Receive PDO mapping</td>
<td>Name: Module 2 (SGD7). 2nd Receive PDO mapping</td>
</tr>
<tr>
<td>Index: 0x1601</td>
<td>Index: 0x1611</td>
</tr>
<tr>
<td>Flags: Everything de-activated</td>
<td></td>
</tr>
<tr>
<td>Direction RxPdo (Output): activated</td>
<td></td>
</tr>
<tr>
<td>Exclude: 1600, 1602, 1603: de-activated</td>
<td>1610, 1612, 1613: de-activated</td>
</tr>
</tbody>
</table>

Please note these settings, otherwise the PDO mappings can not be activated at the same time!

<table>
<thead>
<tr>
<th>Entries</th>
<th>Module 1 (axis 1)</th>
<th>Module 2 (axis 2)</th>
<th>Bit length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Index</td>
<td>Index</td>
<td></td>
</tr>
<tr>
<td>Profile velocity</td>
<td>0x6081:00</td>
<td>0x6881:00</td>
<td>32bit</td>
</tr>
<tr>
<td>Profile acceleration</td>
<td>0x6083:00</td>
<td>0x6883:00</td>
<td>32bit</td>
</tr>
<tr>
<td>Profile deceleration</td>
<td>0x6084:00</td>
<td>0x6884:00</td>
<td>32bit</td>
</tr>
</tbody>
</table>
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’.

<table>
<thead>
<tr>
<th>PDO assign</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Module 1 (SGD7). 1st Transmit PDO mapping</td>
</tr>
<tr>
<td></td>
<td>Module 1 (SGD7). 2nd Transmit PDO mapping</td>
</tr>
<tr>
<td></td>
<td>Module 2 (SGD7). 1st Transmit PDO mapping</td>
</tr>
<tr>
<td></td>
<td>Module 2 (SGD7). 2nd Transmit PDO mapping</td>
</tr>
</tbody>
</table>

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’ \(\rightarrow\) ‘M1_PdoInputs’ (here 0)
- Module 2: ‘S7 Input address’ \(\rightarrow\) ‘M2_PdoInputs’ (here 36)
- Module 1: ‘S7 Output address’ \(\rightarrow\) ‘M1_PdoOutputs’ (here 0)
- Module 2: ‘S7 Output address’ \(\rightarrow\) ‘M2_PdoOutputs’ (here 36)
17. Click on your CPU in the **SPEED7 EtherCAT Manager** and select the ‘Master’ tab in the ‘Device editor’.

![EtherCAT Manager Diagram]

- 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

![User Program Diagram]
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_ConfigSigma7EC_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_InitSigma7W_EC**
- The **Init** 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_KernelSigma7_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_KernelSigma7_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

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.

<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>Type</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Config</td>
<td>&quot;VMC_ConfigSigma7EC_REF&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Axis</td>
<td>&quot;MC_AXIS_REF&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>END_STRUCT</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>Type</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Config</td>
<td>&quot;VMC_ConfigSigma7EC_REF&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Axis</td>
<td>&quot;MC_AXIS_REF&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>END_STRUCT</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Open OB 1 and program the following FB calls with associated DBs:

FB 874 - VMC_InitSigma7W_EC, DB 874

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

CALL "VMC_Init Sigma7W_EC", "DI_Init Sgm7WETC01"

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 :=5.0000000e+001
M1_MaxAccelerationApp :=1.0000000e+002
M1_MaxDecelerationApp :=1.0000000e+002
M1_MaxPosition :=1.0485000e+003
M1_MinPosition :=-1.0485140e+003
M1_EnableMaxPosition :=TRUE
M1_EnableMinPosition :=TRUE
M1_MinUserPosition :=-1000.0
M1_MaxUserPosition :=1000.0

Valid :="InitS7WEC1_Valid"
Error :="InitS7WEC1_Error"
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 kernel for the respective axis

ErrorID := "InitS7WEC1_ErrorID"
M1_Config := "Axis01".Config
M1_Axis := "Axis01".Axis
M2_Config := "Axis02".Config
M2_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.

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
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!

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!

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.

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

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.*

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

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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Release of initialization</td>
</tr>
<tr>
<td>LogicalAddress</td>
<td>INPUT</td>
<td>INT</td>
<td>Start address of the PDO input data</td>
</tr>
<tr>
<td>M1_PdoInputs</td>
<td>INPUT</td>
<td>INT</td>
<td>Start address of the input PDOs for axis 1</td>
</tr>
<tr>
<td>M1_PdoOutputs</td>
<td>INPUT</td>
<td>INT</td>
<td>Start address of the output PDOs for axis 1</td>
</tr>
<tr>
<td>Parameter</td>
<td>Declaration</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| 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: \( p_{\text{increments}} = \frac{p_u}{\text{FactorPosition}} \)
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: \( v_{\text{increments/s}} = \frac{v_{u/s}}{\text{FactorVelocity}} \)
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/s²] in drive units \(10^{-4} \times \text{increments/s}^2\) and back of axis 1.
It's valid: \( a_{\text{increments/s}^2} = \frac{a_{u/s^2}}{\text{FactorAcceleration}} \)
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/s²].
The command inputs are checked to the maximum value before execution. |
| M1_MaxDecelerationApp   | INPUT       | REAL      | Maximum acceleration of application of axis 1 [u/s²].
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. |
<p>| M2_PdoInputs            | INPUT       | INT       | Start address of the input PDOs for axis 2 |</p>
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>M2_PdoOutputs</td>
<td>INPUT</td>
<td>INT</td>
<td>Start address of the output PDOs for axis 2</td>
</tr>
<tr>
<td>M2_EncoderType</td>
<td>INPUT</td>
<td>INT</td>
<td>Encoder type of axis 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: Absolute encoder</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2: Incremental encoder</td>
</tr>
<tr>
<td>M2_EncoderResolutionBits</td>
<td>INPUT</td>
<td>INT</td>
<td>Number of bits corresponding to one encoder revolution of axis 2. Default: 20</td>
</tr>
<tr>
<td>M2_FactorPosition</td>
<td>INPUT</td>
<td>REAL</td>
<td>Factor for converting the position of user units [u] into drive units [increments] and back of axis 2. It's valid: ( p_{\text{increments}} = p_u \times \text{FactorPosition} ) Please consider the factor which can be specified on the drive via the objects 0x2701: 1 and 0x2701: 2. This should be 1.</td>
</tr>
<tr>
<td>M2_FactorVelocity</td>
<td>INPUT</td>
<td>REAL</td>
<td>Factor for converting the speed of user units [u/s] into drive units [increments/s] and back of axis 2. It's valid: ( v_{\text{increments/s}} = v_{\text{u/s}} \times \text{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.</td>
</tr>
<tr>
<td>M2_FactorAcceleration</td>
<td>INPUT</td>
<td>REAL</td>
<td>Factor to convert the acceleration of user units [u/s^2] in drive units (10^{-4} \times \text{increments/s^2}) and back of axis 2. It's valid: ( a_{\text{increments/s^2}} = a_{\text{u/s^2}} \times \text{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.</td>
</tr>
<tr>
<td>M2_OffsetPosition</td>
<td>INPUT</td>
<td>REAL</td>
<td>Offset for the zero position of axis 2 [u].</td>
</tr>
<tr>
<td>M2_MaxVelocityApp</td>
<td>INPUT</td>
<td>REAL</td>
<td>Maximum application speed of axis 2 [u/s]. The command inputs are checked to the maximum value before execution.</td>
</tr>
<tr>
<td>M2_MaxAccelerationApp</td>
<td>INPUT</td>
<td>REAL</td>
<td>Maximum acceleration of application of axis 2 [u/s^2]. The command inputs are checked to the maximum value before execution.</td>
</tr>
<tr>
<td>M2_MaxDecelerationApp</td>
<td>INPUT</td>
<td>REAL</td>
<td>Maximum acceleration of application of axis 2 [u/s^2]. The command inputs are checked to the maximum value before execution.</td>
</tr>
<tr>
<td>M2_MaxPosition</td>
<td>INPUT</td>
<td>REAL</td>
<td>Maximum position for monitoring the software limits of axis 2 [u].</td>
</tr>
<tr>
<td>M2_MinPosition</td>
<td>INPUT</td>
<td>REAL</td>
<td>Minimum position for monitoring the software limits of axis 2 [u].</td>
</tr>
<tr>
<td>M2_EnableMaxPosition</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Monitoring maximum position of axis 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TRUE: Activates the monitoring of the maximum position.</td>
</tr>
<tr>
<td>M2_EnableMinPosition</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Monitoring minimum position of axis 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TRUE: Activation of the monitoring of the minimum position.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Declaration</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------------</td>
<td>-----------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>M1_MinUserPosition</td>
<td>OUTPUT</td>
<td>REAL</td>
<td>Minimum user position for axis 1 based on the minimum encoder value of 0x80000000 and the FactorPosition [u].</td>
</tr>
<tr>
<td>M1_MaxUserPosition</td>
<td>OUTPUT</td>
<td>REAL</td>
<td>Maximum user position for axis 1 based on the maximum encoder value of 0x7FFFFFFF and the FactorPosition [u].</td>
</tr>
<tr>
<td>M2_MinUserPosition</td>
<td>OUTPUT</td>
<td>REAL</td>
<td>Minimum user position for axis 2 based on the minimum encoder value of 0x80000000 and the FactorPosition [u].</td>
</tr>
<tr>
<td>M2_MaxUserPosition</td>
<td>OUTPUT</td>
<td>REAL</td>
<td>Maximum user position for axis 2 based on the maximum encoder value of 0x7FFFFFFF and the FactorPosition [u].</td>
</tr>
<tr>
<td>Valid</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Initialization</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- TRUE: Initialization is valid.</td>
</tr>
<tr>
<td>Error</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- TRUE: An error has occurred. Additional error information can be found in the parameter ErrorID. The axis is disabled.</td>
</tr>
<tr>
<td>ErrorID</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>Additional error information</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Chap. 12 ‘ErrorID - Additional error information’ page 457</td>
</tr>
<tr>
<td>M1_Config</td>
<td>IN_OUT</td>
<td>UDT872</td>
<td>Data structure for transferring axis-dependent configuration data to the AxisKernel for axis 1.</td>
</tr>
<tr>
<td>M1_Axis</td>
<td>IN_OUT</td>
<td>MC_AXIS_REF</td>
<td>Data structure for transferring axis-dependent information to the AxisKernel and PLCopen blocks for axis 1.</td>
</tr>
<tr>
<td>M2_Config</td>
<td>IN_OUT</td>
<td>UDT872</td>
<td>Data structure for transferring axis-dependent configuration data to the AxisKernel for axis 2.</td>
</tr>
<tr>
<td>M2_Axis</td>
<td>IN_OUT</td>
<td>MC_AXIS_REF</td>
<td>Data structure for transferring axis-dependent information to the AxisKernel and PLCopen blocks for axis 2.</td>
</tr>
</tbody>
</table>
4 Usage Sigma-5/7 PROFINET

4.1 Usage Sigma-5 PROFINET

4.1.1 Overview

Precondition

- SPEED7 Studio from V1.8
- 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

<table>
<thead>
<tr>
<th>Slot</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>CPU M13-CCF0000</td>
</tr>
<tr>
<td>-X2</td>
<td>MPI interface</td>
</tr>
<tr>
<td>-X3</td>
<td>PROFINET PG_OP IO-System</td>
</tr>
</tbody>
</table>

...
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.

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 smaller value of the start addresses of the input/output address range.

- **User program ↪ 139**

- **FB 891 - VMC InitSigma_PN ↪ 217**

### Example hardware configuration

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

<table>
<thead>
<tr>
<th>Slot</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>CPU 015-CEFPR01</td>
</tr>
<tr>
<td>-X1</td>
<td>PG_OP_Ethernet</td>
</tr>
<tr>
<td>-X3</td>
<td>MPI interface</td>
</tr>
<tr>
<td>-X4</td>
<td>PROFINET-IO-System</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
**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.

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

5. At ‘Catalog’ select the ‘Components’ tab.

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 \( \rightarrow \) 139
- FB 891 - VMC InitSigma_PN \( \rightarrow \) 217

### Example hardware configuration

<table>
<thead>
<tr>
<th>Slot</th>
<th>Component</th>
<th>I-Adr.</th>
<th>O-Adr.</th>
<th>Diagnostic address</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SGDV-OCB03A</td>
<td>2045</td>
<td>2045</td>
<td></td>
</tr>
<tr>
<td>X1</td>
<td>PN-IO</td>
<td>2039</td>
<td>2039</td>
<td></td>
</tr>
<tr>
<td>X1 P1</td>
<td>Port 1</td>
<td>2038</td>
<td>2038</td>
<td></td>
</tr>
<tr>
<td>X1 P2</td>
<td>Port 2</td>
<td>2037</td>
<td>2037</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>DO with YASKAWA telegr.100, PZD-16/14</td>
<td>2036</td>
<td>2036</td>
<td></td>
</tr>
<tr>
<td>1.1</td>
<td>Parameter Access Point</td>
<td>2036</td>
<td>2036</td>
<td></td>
</tr>
<tr>
<td>1.2</td>
<td>YASKAWA telegram, PZD-16/14</td>
<td>0-27</td>
<td>0-31</td>
<td>2036</td>
</tr>
</tbody>
</table>
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_ConfigSigmaPN_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_InitSigma_PN**
  - The `Init` 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_AxisControlSigma_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_AxisControlSigma_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_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

   | Addr... | Name | Data type | ...
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>Config</td>
<td>UDT</td>
</tr>
<tr>
<td>...</td>
<td>Axis</td>
<td>UDT</td>
</tr>
</tbody>
</table>

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**

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

**Example call**

```vhdl
CALL "VMC_InitSigma_PN", "VMC_InitSigma_PN_1"
Enable :="InitS5PN1_Enable"
LogicalAddress :=0 //HW-Konfig: Smallest IO 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
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:

```vhdl
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"
```
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

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:

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:

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

⇒ 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

<table>
<thead>
<tr>
<th>Slot</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CPU ...</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

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].

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CPU ... PN-IO PROFINET-IO-System</td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Axis-001</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>DO with ... teleg_100...</td>
</tr>
<tr>
<td></td>
<td>Parameter Access P...</td>
</tr>
<tr>
<td></td>
<td>YASKAWA telegram...</td>
</tr>
</tbody>
</table>

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

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

### 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](http://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. 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’.

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.

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].

More information about the initialization and the usage of the Ethernet PG/OP channel can be found in the manual of the CPU.
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: ‘InputsStartDateAddress’:
    Setting of the start address of the input address range.
  - FB 891 - VMC InitSigma_PN: ‘OutputsStartDateAddress’:
    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**

<table>
<thead>
<tr>
<th>Slot</th>
<th>Module</th>
<th>...</th>
<th>I Addr.</th>
<th>Q Addr.</th>
<th>Diagnostic address</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SGDV-OCB03A</td>
<td></td>
<td></td>
<td></td>
<td>2037</td>
</tr>
<tr>
<td>X1</td>
<td>PN-IO</td>
<td></td>
<td></td>
<td></td>
<td>2036</td>
</tr>
<tr>
<td>X1 P1</td>
<td>Port 1</td>
<td></td>
<td></td>
<td></td>
<td>2035</td>
</tr>
<tr>
<td>X1 P2</td>
<td>Port 2</td>
<td></td>
<td></td>
<td></td>
<td>2034</td>
</tr>
<tr>
<td>1</td>
<td>DO with YASKAWA telegr.100, PZD-16/14</td>
<td></td>
<td></td>
<td></td>
<td>2033</td>
</tr>
<tr>
<td>1.1</td>
<td>Parameter Access Point</td>
<td></td>
<td></td>
<td></td>
<td>2033</td>
</tr>
<tr>
<td>1.2</td>
<td>YASKAWA telegram, PZD-16/14</td>
<td>284-311</td>
<td>288-319</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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_ConfigSigmaPN_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_InitSigma_PN`
  - The `Init` 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_AxisControlSigma_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_AxisControlSigma_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**

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].
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.

Create interrupt OBs

Create interrupt OBs

Create axis DB

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
- FB 891 - VMC InitSigma_PN

### Example hardware configuration

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

### Example call

CALL "VMC_InitSigma_PN", "VMC_InitSigma_PN_1"
Enable :="InitS5PN1_Enable"
LogicalAddress :=284 //HW-Konfig: Smallest IO addr.
ParaAccessPointAddress :=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
### 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.

```plaintext
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

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**

- 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:

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

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

Device overview:

<table>
<thead>
<tr>
<th>Module</th>
<th>Slot</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLC...</td>
<td>2</td>
<td>CPU 314C-2PN/DP</td>
</tr>
<tr>
<td>MPI interface...</td>
<td>2 X1</td>
<td>MPI/DI interface</td>
</tr>
<tr>
<td>PROFINET interface...</td>
<td>2 X2</td>
<td>PROFINET interface</td>
</tr>
<tr>
<td>DI24/DO16...</td>
<td>2 5</td>
<td>DI24/DO16</td>
</tr>
<tr>
<td>AI5/AO2...</td>
<td>2 6</td>
<td>AI5/AO2</td>
</tr>
<tr>
<td>Count...</td>
<td>2 7</td>
<td>Count</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Module</th>
<th>Slot</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLC</td>
<td>2</td>
<td>CPU 314C-2PN/DP</td>
</tr>
<tr>
<td>MPI/DP interface</td>
<td>2 X1</td>
<td>MPI/DP interface</td>
</tr>
<tr>
<td>PROFINET interface</td>
<td>2 X2</td>
<td>PROFINET interface</td>
</tr>
<tr>
<td>CP 343-1</td>
<td>4</td>
<td>CP 343-1</td>
</tr>
</tbody>
</table>

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 \(\rightarrow\) Device configuration’ the ‘Device overview’.

3. Assign a suitable ‘Device name’ such as Axis-001.

4. **Device overview**

<table>
<thead>
<tr>
<th>Module</th>
<th>Slot</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axis-001</td>
<td>0</td>
<td>SGDV-0CB03A</td>
</tr>
<tr>
<td>PN-IO</td>
<td>0 X1</td>
<td>SGDV-0CB03A</td>
</tr>
<tr>
<td>DO w/ Yaskawa telegr.100,PZD...</td>
<td>1</td>
<td>DO w/ Yaskawa telegr.100,PZD-16/14</td>
</tr>
<tr>
<td>Parameter Access Point</td>
<td>1 1</td>
<td>Parameter Access Point</td>
</tr>
<tr>
<td>Yaskawa telegram, PZD-16/14</td>
<td>1 2</td>
<td>Yaskawa telegram, PZD-16/14</td>
</tr>
</tbody>
</table>

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.

\(\Rightarrow\) 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.
4.1.5.2 Hardware configuration System 300S

Precondition

- 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 ....’.

Example hardware configuration

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

<table>
<thead>
<tr>
<th>Module</th>
<th>Slot</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLC</td>
<td>2</td>
<td>CPU 315-2PN/DP</td>
</tr>
<tr>
<td>MPI/DP interface</td>
<td>2 X1</td>
<td>MPI/DP interface</td>
</tr>
<tr>
<td>PROFINET interface</td>
<td>2 X2</td>
<td>PROFINET interface</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Module</th>
<th>Slot</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLC...</td>
<td>2</td>
<td>CPU 315-2PN/DP</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>DI...</td>
<td>4</td>
<td>DI...</td>
</tr>
<tr>
<td>DO...</td>
<td>5</td>
<td>DO...</td>
</tr>
<tr>
<td>DIO...</td>
<td>6</td>
<td>DIO...</td>
</tr>
<tr>
<td>AI...</td>
<td>7</td>
<td>AI...</td>
</tr>
<tr>
<td>AO...</td>
<td>8</td>
<td>AO...</td>
</tr>
<tr>
<td>CP 343-1</td>
<td>9</td>
<td>CP 343-1</td>
</tr>
</tbody>
</table>
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**

<table>
<thead>
<tr>
<th>Module</th>
<th>Slot</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axis-001</td>
<td>0</td>
<td>SGDV-0CB03A</td>
</tr>
<tr>
<td>PN-IO</td>
<td>0 X1</td>
<td>SGDV-0CB03A</td>
</tr>
<tr>
<td>DO w/ Yaskawa telegr.100,PZD...</td>
<td>1</td>
<td>DO w/ Yaskawa telegr.100,PZD-16/14</td>
</tr>
<tr>
<td>Parameter Access Point</td>
<td>1 1</td>
<td>Parameter Access Point</td>
</tr>
<tr>
<td>Yaskawa telegram, PZD-16/14</td>
<td>1 2</td>
<td>Yaskawa telegram, PZD-16/14</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

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...’:
  - 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 smaller value of the start addresses of the input/output address range.

- User program § 166
- FB 891 - VMC InitSigma_PN § 217

### Example hardware configuration

<table>
<thead>
<tr>
<th>Slot</th>
<th>Component</th>
<th>I-Adr.</th>
<th>O-Adr.</th>
<th>Diagnostic address</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SGDV-OGB03A</td>
<td></td>
<td></td>
<td>2037</td>
</tr>
<tr>
<td>X1</td>
<td>PN-IO</td>
<td></td>
<td>2036</td>
<td></td>
</tr>
<tr>
<td>X1 P1</td>
<td>Port 1</td>
<td></td>
<td>2035</td>
<td></td>
</tr>
<tr>
<td>X1 P2</td>
<td>Port 2</td>
<td></td>
<td>2034</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>DO with YASKAWA telegr.100, PZD-16/14</td>
<td></td>
<td>2033</td>
<td></td>
</tr>
<tr>
<td>1.1</td>
<td>Parameter Access Point</td>
<td></td>
<td></td>
<td>2033</td>
</tr>
<tr>
<td>1.2</td>
<td>YASKAWA telegram, PZD-16/14</td>
<td>284-311</td>
<td>288-319</td>
<td></td>
</tr>
</tbody>
</table>
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_ConfigSigmaPN**
    - 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_InitSigma_PN**
  - The **Init** 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_AxisControlSigma_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_AxisControlSigma_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**

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_Vxx.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.zalxx.

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...':
  - 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.

### Example hardware configuration

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

### Example call

```plaintext
CALL "VMC_InitSigma_PN" , "VMC_InitSigma_PN_1"
Enable :="InitS5PN1_Enable"
LogicalAddress :=284 //HW-Konfig: Smallest IO addr.
ParaAccessPointAddress :=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 :=6.000000e+001
MaxPosition :=1.048500e+003
MinPosition :=-1.048514e+003
EnableMaxPosition :=TRUE
EnableMinPosition :=TRUE
```
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
- 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)**

<table>
<thead>
<tr>
<th>Servopack Parameter</th>
<th>Address</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PnB32</td>
<td>606Dh</td>
<td>Velocity Window</td>
<td>1000 Velocity units</td>
</tr>
<tr>
<td>PnB34</td>
<td>606Eh</td>
<td>Velocity Window Time</td>
<td>50 ms</td>
</tr>
<tr>
<td>PnC20</td>
<td>0922h</td>
<td>Telegram Selection (100: General Telegram: All OP modes)</td>
<td>100</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Slot</th>
<th>Module</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>CPU M13-CCF0000</td>
<td></td>
</tr>
<tr>
<td>-X2</td>
<td>MPI interface</td>
<td></td>
</tr>
<tr>
<td>-X3</td>
<td>PROFINET PG_OP IO-System</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
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.

4. Click on the *Sigma-7* drive.

   The *Sigma-7* drive is connected to your PROFINET IO controller.

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

<table>
<thead>
<tr>
<th>Slot</th>
<th>Component</th>
<th>...</th>
<th>I-Adr.</th>
<th>O-Adr.</th>
<th>Diagnostic address</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SGD7S-xxxAC0xxxx</td>
<td></td>
<td>2035</td>
<td>2035</td>
<td></td>
</tr>
<tr>
<td>X1</td>
<td>PN-IO</td>
<td></td>
<td>2034</td>
<td>2034</td>
<td></td>
</tr>
<tr>
<td>X1 P1</td>
<td>Port 1</td>
<td></td>
<td>2033</td>
<td>2033</td>
<td></td>
</tr>
<tr>
<td>X1 P2</td>
<td>Port 2</td>
<td></td>
<td>2032</td>
<td>2032</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>DO with YASKAWA telegr.100, PZD-16/14</td>
<td></td>
<td>2044</td>
<td>2044</td>
<td></td>
</tr>
<tr>
<td>1.1</td>
<td>Parameter Access Point</td>
<td></td>
<td>2044</td>
<td>2044</td>
<td></td>
</tr>
<tr>
<td>1.2</td>
<td>YASKAWA telegram, PZD-16/14</td>
<td></td>
<td>28-55</td>
<td>32-63</td>
<td>2044</td>
</tr>
</tbody>
</table>
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-CEPFPR01 and click at [OK].
   ⇒ The CPU is inserted in 'Devices and networking' and the 'Device configuration' is opened.

Device configuration

<table>
<thead>
<tr>
<th>Slot</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>CPU 015-CEPFPR01</td>
</tr>
<tr>
<td>-X1</td>
<td>PG_OP_Ethernet</td>
</tr>
<tr>
<td>-X3</td>
<td>MPI interface</td>
</tr>
<tr>
<td>-X4</td>
<td>PROFINET-IO-System</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>

...
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 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 ** 182
- **FB 891 - VMC InitSigma_PN ** 217

<table>
<thead>
<tr>
<th>Slot</th>
<th>Component</th>
<th>...</th>
<th>I-Adr.</th>
<th>O-Adr.</th>
<th>Diagnostic address</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SGD7S-xxxAC0xxxx</td>
<td></td>
<td>2035</td>
<td>2035</td>
<td></td>
</tr>
<tr>
<td>X1</td>
<td>PN-IO</td>
<td></td>
<td>2034</td>
<td>2034</td>
<td></td>
</tr>
<tr>
<td>X1 P1</td>
<td>Port 1</td>
<td></td>
<td>2033</td>
<td>2033</td>
<td></td>
</tr>
<tr>
<td>X1 P2</td>
<td>Port 2</td>
<td></td>
<td>2032</td>
<td>2032</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>DO with YASKAWA telegr.100, PZD-16/14</td>
<td></td>
<td>2044</td>
<td>2044</td>
<td></td>
</tr>
<tr>
<td>1.1</td>
<td>Parameter Access Point</td>
<td></td>
<td>2044</td>
<td>2044</td>
<td></td>
</tr>
<tr>
<td>1.2</td>
<td>YASKAWA telegram, PZD-16/14</td>
<td></td>
<td>28-55</td>
<td>32-63</td>
<td>2044</td>
</tr>
</tbody>
</table>
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_ConfigSigmaPN_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_InitSigma_PN**
  - The *Init* 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_AxisControlSigma_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_AxisControlSigma_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 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.

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

<table>
<thead>
<tr>
<th>Slot</th>
<th>Component</th>
<th>I-Adr.</th>
<th>O-Adr.</th>
<th>Diagnostic address</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SGD7S-xxxAC0xxxx</td>
<td>2035</td>
<td>2035</td>
<td></td>
</tr>
<tr>
<td>X1</td>
<td>PN-IO</td>
<td>2034</td>
<td>2034</td>
<td></td>
</tr>
<tr>
<td>X1 P1</td>
<td>Port 1</td>
<td>2033</td>
<td>2033</td>
<td></td>
</tr>
<tr>
<td>X1 P2</td>
<td>Port 2</td>
<td>2032</td>
<td>2032</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>DO with YASKAWA telegr.100, PZD-16/14</td>
<td>2044</td>
<td>2044</td>
<td></td>
</tr>
<tr>
<td>1.1</td>
<td>Parameter Access Point</td>
<td>2044</td>
<td>2044</td>
<td></td>
</tr>
<tr>
<td>1.2</td>
<td>YASKAWA telegram, PZD-16/14</td>
<td>28-55</td>
<td>32-63</td>
<td>2044</td>
</tr>
</tbody>
</table>

Example call

```plaintext
CALL "VMC_InitSigma_PN", "VMC_InitSigma_PN_1"
Enable :="InitS7PN1_Enable"
LogicalAddress :=28 //HW-Konfig: Smallest IO 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+003
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.

```plaintext
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:

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:

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

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.
Navigate in the hardware catalog to the directory ‘PROFINET IO è Additional field devices è 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

<table>
<thead>
<tr>
<th>Slot</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CPU ... PN-IO</td>
</tr>
<tr>
<td>2</td>
<td>cpu ... PN-IO</td>
</tr>
<tr>
<td>3</td>
<td>343-1EX30</td>
</tr>
</tbody>
</table>

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 è Properties’ the properties dialog.
3. Assign a suitable ‘Device name’ such as Axis-001.
4. Confirm your input with [OK].

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td><strong>2</strong></td>
<td><strong>CPU ...</strong></td>
</tr>
<tr>
<td><strong>X...</strong></td>
<td><strong>PN-IO</strong></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

In the hardware catalog, expand the Sigma-7 PROFINET drive ‘SGD7S-xxxAC0xxx’ 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...’:
  - FB 891 - VMC InitSigma_PN: ‘InputsStartAddress’:
  - FB 891 - VMC InitSigma_PN: ‘OutputsStartAddress’:
  - FB 891 - VMC InitSigma_PN: ‘LogicalAddress’:

- User program 195
- FB 891 - VMC InitSigma_PN 217
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](http://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

<table>
<thead>
<tr>
<th>Slot</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CPU 315-2 PN/DP</td>
</tr>
<tr>
<td>X1</td>
<td>MPI/DP</td>
</tr>
<tr>
<td>X2</td>
<td>PN-IO</td>
</tr>
<tr>
<td>X2...</td>
<td>Port 1</td>
</tr>
<tr>
<td>X2...</td>
<td>Port 2</td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

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’.

To configure the modules on the standard bus:

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.

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

<table>
<thead>
<tr>
<th>Slot</th>
<th>Component</th>
<th>I-Adr.</th>
<th>O-Adr.</th>
<th>Diagnostic address</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SGD7S-xxxAC0xxxx</td>
<td></td>
<td></td>
<td>2035</td>
</tr>
<tr>
<td>X1</td>
<td>PN-IO</td>
<td></td>
<td></td>
<td>2034</td>
</tr>
<tr>
<td>X1 P1</td>
<td>Port 1</td>
<td></td>
<td></td>
<td>2033</td>
</tr>
<tr>
<td>X1 P2</td>
<td>Port 2</td>
<td></td>
<td></td>
<td>2032</td>
</tr>
<tr>
<td>1</td>
<td>DO with YASKAWA telegr.100, PZD-16/14</td>
<td></td>
<td></td>
<td>2044</td>
</tr>
<tr>
<td>1.1</td>
<td>Parameter Access Point</td>
<td></td>
<td></td>
<td>2044</td>
</tr>
<tr>
<td>1.2</td>
<td>YASKAWA telegram, PZD-16/14</td>
<td>28-55</td>
<td>32-63</td>
<td></td>
</tr>
</tbody>
</table>

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_ConfigSigmaPN_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_InitSigma_PN**
    
    - The **Init** 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_AxisControlSigma_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_AxisControlSigma_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

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.

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...’:
  - 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

<table>
<thead>
<tr>
<th>Slot</th>
<th>Component</th>
<th>I-Adr.</th>
<th>O-Adr.</th>
<th>Diagnostic address</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SGD7S-xxxAC0xxxx</td>
<td>2035</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X1</td>
<td>PN-IO</td>
<td>2034</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X1 P1</td>
<td>Port 1</td>
<td>2033</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X1 P2</td>
<td>Port 2</td>
<td>2032</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>DO with YASKAWA telegr.100, PZD-16/14</td>
<td></td>
<td>2044</td>
<td></td>
</tr>
<tr>
<td>1.1</td>
<td>Parameter Access Point</td>
<td></td>
<td>2044</td>
<td></td>
</tr>
<tr>
<td>1.2</td>
<td>YASKAWA telegram, PZD-16/14</td>
<td>28-55</td>
<td>32-63</td>
<td></td>
</tr>
</tbody>
</table>

### Example call

```plaintext
CALL "VMC_InitSigma_PN", "VMC_InitSigma_PN_1"
Enable :="InitS7PN1_Enable"
LogicalAddress :=28 //HW-Konfig: Smallest IO 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.

```plaintext
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"
```
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 \textit{Init} block FB 891 - VMC\textunderscore InitSigma\textunderscore PN with \texttt{Enable = TRUE}.

   \textbullet\ The output \texttt{Valid} returns \texttt{TRUE}. In the event of a fault, you can determine the error by evaluating the \texttt{ErrorID}.

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

   \begin{center}
   \textit{Do not continue until the \textit{Init} block does not report any errors!}
   \end{center}

3. Program your application with the FB 890 - VMC\textunderscore AxisControlSigma\textunderscore PN or with the PLCopen blocks.

4.2.5 Usage in Siemens TIA-Portal

4.2.5.1 Hardware configuration System MICRO respectively SLIO

\textbf{Precondition}

\begin{itemize}
\item Please use the Siemens TIA Portal from V.14 for the configuration.
\item 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.
\item For the PROFINET drive can be configured in the Siemens TIA Portal, the corresponding GSDML file must be installed.
\end{itemize}

\textbf{Install GSDML file for System MICRO respectively SLIO}

The installation of the PROFINET IO device happens in the hardware catalog with the following approach:

\begin{enumerate}
\item Go to the service area of www.vipa.com.
\item Download the configuration file for your System MICRO or SLIO CPU from the download area via 'Config files \textbullet PROFINET'.
\item Extract the file into your working directory.
\item Start the Siemens TIA Portal.
\item Close all the projects.
\item Switch to the \textit{Project view}.
\item Select 'Options \textbullet Install general station description file (GSD)'.
\item Navigate to your working directory and install the according GSDML file.
\end{enumerate}

\textbullet\ 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 \textbullet PROFINET IO \textbullet I/O \textbullet VIPA ...'.

From YASKAWA there are the following PROFINET IO devices:

\begin{itemize}
\item System MICRO: ‘VIPA Micro PLC’
\item System SLIO: ‘VIPA System SLIO’
\end{itemize}
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](http://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’.
Depending on the VIPA CPU used, select the following CPU from Siemens:

<table>
<thead>
<tr>
<th>VIPA CPU</th>
<th>to configure as SIMATIC S7-300 &gt; ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>M13-CCF0000 from V2.4.12</td>
<td>CPU 314C-2 PN/DP (6ES7 314-6EH04-0AB0 V3.3)</td>
</tr>
<tr>
<td>013-CCFR00 from V2.4.12</td>
<td>CPU 314C-2 PN/DP (6ES7 314-6EH04-0AB0 V3.3)</td>
</tr>
<tr>
<td>014-CEFR01 from V2.4.12</td>
<td>CPU 315-2 PN/DP (6ES7 315-2EH14-0AB0 V3.2)</td>
</tr>
<tr>
<td>015-CEFNR00 from V2.4.16</td>
<td>CPU 315-2 PN/DP (6ES7 315-2EH14-0AB0 V3.2)</td>
</tr>
<tr>
<td>015-CEFPR01 from V2.4.12</td>
<td>CPU 315-2 PN/DP (6ES7 315-2EH14-0AB0 V3.2)</td>
</tr>
<tr>
<td>017-CEFPR00 from V2.4.12</td>
<td>CPU 317-2PN/DP (6ES7 317-2EK14-0AB0 V3.2)</td>
</tr>
</tbody>
</table>

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

---

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’.

---

Device overview:

<table>
<thead>
<tr>
<th>Module</th>
<th>Slot</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLC...</td>
<td>2</td>
<td>CPU 314C-2PN/DP</td>
</tr>
<tr>
<td>MPI interface</td>
<td>2 X1</td>
<td>MPI/DP interface</td>
</tr>
<tr>
<td>PROFINET interface</td>
<td>2 X2</td>
<td>PROFINET interface</td>
</tr>
<tr>
<td>DI24/DO16...</td>
<td>2 5</td>
<td>DI24/DO16</td>
</tr>
<tr>
<td>AI5/AO2...</td>
<td>2 6</td>
<td>AI5/AO2</td>
</tr>
<tr>
<td>Count...</td>
<td>2 7</td>
<td>Count</td>
</tr>
</tbody>
</table>

...
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

<table>
<thead>
<tr>
<th>Module</th>
<th>Slot</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLC ...</td>
<td>2</td>
<td>CPU 314C-2PN/DP</td>
</tr>
<tr>
<td>MPI/DP interface</td>
<td>2 X1</td>
<td>MPI/DP interface</td>
</tr>
<tr>
<td>PROFINET interface</td>
<td>2 X2</td>
<td>PROFINET interface</td>
</tr>
<tr>
<td>CP 343-1</td>
<td>4</td>
<td>CP 343-1</td>
</tr>
</tbody>
</table>

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**

<table>
<thead>
<tr>
<th>Module</th>
<th>...</th>
<th>Slot</th>
<th>...</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axis-001</td>
<td></td>
<td>0</td>
<td></td>
<td>SGD7S-xxxAC0xxxx</td>
</tr>
<tr>
<td>PN-IO</td>
<td></td>
<td>0 X1</td>
<td></td>
<td>SGD7S-xxxAC0xxxx</td>
</tr>
<tr>
<td>DO w/ Yaskawa telegr.100,PZD...</td>
<td></td>
<td>1</td>
<td></td>
<td>DO w/ Yaskawa telegr.100,PZD-16/14</td>
</tr>
<tr>
<td>Parameter Access Point</td>
<td>1</td>
<td>1</td>
<td></td>
<td>Parameter Access Point</td>
</tr>
<tr>
<td>Yaskawa telegram, PZD-16/14</td>
<td>1</td>
<td>2</td>
<td></td>
<td>Yaskawa telegram, PZD-16/14</td>
</tr>
</tbody>
</table>

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.

---

**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.
Usage Sigma-5/7 PROFINET

Example hardware configuration

<table>
<thead>
<tr>
<th>Slot</th>
<th>Component</th>
<th>I-Adr.</th>
<th>O-Adr.</th>
<th>Diagnostic address</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SGD7S-xxxAC0xxxx</td>
<td></td>
<td>2035</td>
<td></td>
</tr>
<tr>
<td>X1</td>
<td>PN-I0</td>
<td></td>
<td>2034</td>
<td></td>
</tr>
<tr>
<td>X1 P1</td>
<td>Port 1</td>
<td></td>
<td>2033</td>
<td></td>
</tr>
<tr>
<td>X1 P2</td>
<td>Port 2</td>
<td></td>
<td>2032</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>DO with YASKAWA telegr.100, PZD-16/14</td>
<td></td>
<td>2044</td>
<td></td>
</tr>
<tr>
<td>1.1</td>
<td>Parameter Access Point</td>
<td></td>
<td>2044</td>
<td></td>
</tr>
<tr>
<td>1.2</td>
<td>YASKAWA telegram, PZD-16/14</td>
<td>28-55</td>
<td>32-63</td>
<td></td>
</tr>
</tbody>
</table>

4.2.5.2 Hardware configuration System 300S

Precondition

- 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](http://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

<table>
<thead>
<tr>
<th>Module</th>
<th>Slot</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLC</td>
<td>2</td>
<td>CPU 315-2PN/DP</td>
</tr>
<tr>
<td>MPI/DP interface</td>
<td>2 X1</td>
<td>MPI/DP interface</td>
</tr>
<tr>
<td>PROFINET interface</td>
<td>2 X2</td>
<td>PROFINET interface</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
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]

### Device overview

<table>
<thead>
<tr>
<th>Module</th>
<th>Slot</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLC...</td>
<td>2</td>
<td>CPU 315-2PN/DP</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>DI...</td>
<td>4</td>
<td>DI...</td>
</tr>
<tr>
<td>DO...</td>
<td>5</td>
<td>DO...</td>
</tr>
<tr>
<td>DIO...</td>
<td>6</td>
<td>DIO...</td>
</tr>
<tr>
<td>Al...</td>
<td>7</td>
<td>Al...</td>
</tr>
<tr>
<td>AO...</td>
<td>8</td>
<td>AO...</td>
</tr>
<tr>
<td>CP 343-1</td>
<td>9</td>
<td>CP 343-1</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Module</th>
<th>Slot</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axis-001</td>
<td>0</td>
<td>SGD7S-xxxAC0xxxx</td>
</tr>
<tr>
<td>PN-IO</td>
<td>0 X1</td>
<td>SGD7S-xxxAC0xxxx</td>
</tr>
<tr>
<td>DO w/ Yaskawa telegr.100,PZD...</td>
<td>1</td>
<td>DO w/ Yaskawa telegr.100,PZD-16/14</td>
</tr>
<tr>
<td>Parameter Access Point</td>
<td>1 1</td>
<td>Parameter Access Point</td>
</tr>
<tr>
<td>Yaskawa telegram, PZD-16/14</td>
<td>1 2</td>
<td>Yaskawa telegram, PZD-16/14</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

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.

Example hardware configuration

<table>
<thead>
<tr>
<th>Slot</th>
<th>Component</th>
<th>I-Adr.</th>
<th>O-Adr.</th>
<th>Diagnostic address</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SGD7S-xxxAC0xxxx</td>
<td></td>
<td></td>
<td>2035</td>
</tr>
<tr>
<td>X1</td>
<td>PN-IO</td>
<td></td>
<td></td>
<td>2034</td>
</tr>
<tr>
<td>X1 P1</td>
<td>Port 1</td>
<td></td>
<td></td>
<td>2033</td>
</tr>
<tr>
<td>X1 P2</td>
<td>Port 2</td>
<td></td>
<td></td>
<td>2032</td>
</tr>
<tr>
<td>1</td>
<td>DO with YASKAWA telegr.100, PZD-16/14</td>
<td>2044</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.1</td>
<td>Parameter Access Point</td>
<td></td>
<td></td>
<td>2044</td>
</tr>
<tr>
<td>1.2</td>
<td>YASKAWA telegram, PZD-16/14</td>
<td>28-55</td>
<td>32-63</td>
<td></td>
</tr>
</tbody>
</table>

4.2.5.3 User program
4.2.5.3.1 Program structure
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_ConfigSigmaPN_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_InitSigma_PN**
  - The Init 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_AxisControlSigma_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_AxisControlSigma_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**

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_Vxx.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.zalxx.

**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.
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

### Copy blocks into project

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

   FB 891 - VMC_InitSigma_PN, DB 891

   **Note:**
   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.

### Create axis DB

1. Click at ‘Project tree \( \rightarrow \) ...CPU... \( \rightarrow \) Program blocks \( \rightarrow \) 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

**Note:**
- Hardware configuration \( \downarrow \) 198
- FB 891 - VMC_InitSigma_PN \( \downarrow \) 217
**Example hardware configuration**

<table>
<thead>
<tr>
<th>Slot</th>
<th>Component</th>
<th>I-Adr.</th>
<th>O-Adr.</th>
<th>Diagnostic address</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SGD7S-xxxAC0xxxx</td>
<td></td>
<td></td>
<td>2035</td>
</tr>
<tr>
<td>X1</td>
<td>PN-IO</td>
<td></td>
<td></td>
<td>2034</td>
</tr>
<tr>
<td>X1 P1</td>
<td>Port 1</td>
<td></td>
<td></td>
<td>2033</td>
</tr>
<tr>
<td>X1 P2</td>
<td>Port 2</td>
<td></td>
<td></td>
<td>2032</td>
</tr>
<tr>
<td>1</td>
<td>DO with YASKAWA telegr.100, PZD-16/14</td>
<td></td>
<td></td>
<td>2044</td>
</tr>
<tr>
<td>1.1</td>
<td>Parameter Access Point</td>
<td></td>
<td></td>
<td>2044</td>
</tr>
<tr>
<td>1.2</td>
<td>YASKAWA telegram, PZD-16/14</td>
<td>28-55</td>
<td>32-63</td>
<td></td>
</tr>
</tbody>
</table>

**Example call**

```c
CALL "VMC_InitSigma_PN", "VMC_InitSigma_PN_1"
Enable :="InitS7PN1_Enable"
LogicalAddress :=28 //HW-Konfig: Smallest IO 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.

```c
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"
```
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 \textit{Init} block \texttt{FB 891 - VMC\_InitSigma\_PN with} \texttt{Enable = TRUE}.

\begin{itemize}
  \item The output \texttt{Valid} returns \texttt{TRUE}. In the event of a fault, you can determine the error by evaluating the \texttt{ErrorID}.
  \item You have to call the \textit{Init} block again if you load a new axis DB or you have changed parameters on the \textit{Init} block.
\end{itemize}

\textit{Do not continue until the Init block does not report any errors!}

3. Program your application with the \texttt{FB 890 - VMC\_AxisControlSigma\_PN} or with the PLCopen blocks.

4.3 Drive specific blocks

\begin{itemize}
  \item The PLCopen blocks for axis control can be found here: \texttt{Chap. 9 'Blocks for axis control' page 361}
\end{itemize}

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

\textbf{Description}

The \texttt{FB VMC\_AxisControlSigma\_PN} is a combination of a \texttt{Kernel} for Sigma-5/7 axes for PROFINET and an \texttt{Axis\_Control} for controlling the motion control functions. With the \texttt{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.

\textit{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.}

\textit{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 MoveAbsolute job, the corresponding drive is stopped and then the new move job is executed.

### Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AxisEnable</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Enable/disable axis</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- TRUE: The axis is enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- FALSE: The axis is disabled.</td>
</tr>
<tr>
<td>AxisReset</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Reset axis</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Edge 0-1: Axis reset is performed.</td>
</tr>
<tr>
<td>HomeExecute</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Homing</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Edge 0-1: Homing is started.</td>
</tr>
<tr>
<td>HomePosition</td>
<td>INPUT</td>
<td>REAL</td>
<td>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.</td>
</tr>
<tr>
<td>StopExecute</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Stop axis</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Edge 0-1: Stopping of the axis is started.</td>
</tr>
<tr>
<td>MvVelocityExecute</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Start moving the axis</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Edge 0-1: The axis is accelerated / decelerated to the speed specified.</td>
</tr>
<tr>
<td>MvRelativeExecute</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Start moving the axis</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Edge 0-1: The relative positioning of the axis is started.</td>
</tr>
<tr>
<td>MvAbsoluteExecute</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Start moving the axis</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Edge 0-1: The absolute positioning of the axis is started.</td>
</tr>
<tr>
<td>Direction *</td>
<td>INPUT</td>
<td>BYTE</td>
<td>Mode for absolute positioning:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 0: shortest distance</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 1: positive direction</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 2: negative direction</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 3: current direction</td>
</tr>
<tr>
<td>PositionDistance</td>
<td>INPUT</td>
<td>REAL</td>
<td>Absolute position or relative distance depending on the command in [user units].</td>
</tr>
<tr>
<td>Velocity</td>
<td>INPUT</td>
<td>REAL</td>
<td>Velocity setting (signed value) in [user units / s].</td>
</tr>
<tr>
<td>Acceleration</td>
<td>INPUT</td>
<td>REAL</td>
<td>Acceleration in [user units / s²].</td>
</tr>
<tr>
<td>Deceleration</td>
<td>INPUT</td>
<td>REAL</td>
<td>Deceleration in [user units / s²].</td>
</tr>
<tr>
<td>JogPositive</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Drive axis with constant velocity in positive direction</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Edge 0-1: Drive axis with constant velocity is started.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Edge 1-0: The axis is stopped.</td>
</tr>
<tr>
<td>JogNegative</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Drive axis with constant velocity in negative direction</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Edge 0-1: Drive axis with constant velocity is started.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Edge 1-0: The axis is stopped.</td>
</tr>
<tr>
<td>JogVelocity</td>
<td>INPUT</td>
<td>REAL</td>
<td>Speed setting for jogging (positive value) in [user units / s].</td>
</tr>
<tr>
<td>Parameter</td>
<td>Declaration</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------</td>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>JogAcceleration</td>
<td>INPUT</td>
<td>REAL</td>
<td>Acceleration in [user units / s²].</td>
</tr>
<tr>
<td>JogDeceleration</td>
<td>INPUT</td>
<td>REAL</td>
<td>Delay for jogging in [user units / s²].</td>
</tr>
<tr>
<td>KernelInitReset</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Reset the kernel functions. Caution, running commands are aborted!</td>
</tr>
</tbody>
</table>
| 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**  
<p>|                   |             |           |   - TRUE: The axis is homed.  |</p>
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ModeOfOperation</td>
<td>OUTPUT</td>
<td>INT</td>
<td>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)</td>
</tr>
<tr>
<td>PLCopenState</td>
<td>OUTPUT</td>
<td>INT</td>
<td>Current PLCopenState: 1: Disabled 2: Standstill 3: Homing 4: Discrete Motion 5: Continuous Motion 7: Stopping 8: Errorstop</td>
</tr>
<tr>
<td>ActualPosition</td>
<td>OUTPUT</td>
<td>REAL</td>
<td>Position of the axis in [user unit].</td>
</tr>
<tr>
<td>ActualVelocity</td>
<td>OUTPUT</td>
<td>REAL</td>
<td>Velocity of the axis in [user unit / s]</td>
</tr>
</tbody>
</table>
| 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direction Negative</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status motion job: Position decreasing</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: The position of the axis is decreasing</td>
</tr>
<tr>
<td>SWLimitMinActive</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Software limit switch</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: Software Limit switch Minimum active (Minimum position in negative direction exceeded).</td>
</tr>
<tr>
<td>SWLimitMaxActive</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Software limit switch</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: Software limit switch Maximum active (Maximum position in positive direction exceeded).</td>
</tr>
<tr>
<td>HWLimitMinActive</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Hardware limit switch</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: Negative hardware limit switch active on the drive (NOT- Negative Overtravel).</td>
</tr>
<tr>
<td>HWLimitMaxActive</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Hardware limit switch</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: Positive hardware limit switch active on the drive (POT- Positive Overtravel).</td>
</tr>
<tr>
<td>Config</td>
<td>IN_OUT</td>
<td>VMC_Config-SigmaPN_REF</td>
<td>Reference to the configuration of the axis.</td>
</tr>
<tr>
<td>Axis</td>
<td>IN_OUT</td>
<td>MC_AXIS_REF</td>
<td>Reference to the axis.</td>
</tr>
</tbody>
</table>

*) 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Release of initialization</td>
</tr>
<tr>
<td>Logical address</td>
<td>INPUT</td>
<td>INT</td>
<td>Smallest address of the input/output address range of the hardware configuration of the axis.</td>
</tr>
<tr>
<td>ParaAccessPointAdress</td>
<td>INPUT</td>
<td>INT</td>
<td>Diagnostic address of slot 1 of the hardware configuration of the axis.</td>
</tr>
<tr>
<td>InputsStartAddress</td>
<td>INPUT</td>
<td>INT</td>
<td>Start address of the input address range of the hardware configuration of the axis.</td>
</tr>
<tr>
<td>OutputsStartAddress</td>
<td>INPUT</td>
<td>INT</td>
<td>Start address of the output address range of the hardware configuration of the axis.</td>
</tr>
<tr>
<td>EncoderType</td>
<td>INPUT</td>
<td>INT</td>
<td>Encoder type</td>
</tr>
<tr>
<td>1: Absolute encoder</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2: Incremental encoder</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EncoderResolutionBits</td>
<td>INPUT</td>
<td>INT</td>
<td>Number of bits corresponding to one encoder revolution. Default: 20</td>
</tr>
<tr>
<td>FactorPosition</td>
<td>INPUT</td>
<td>REAL</td>
<td>Factor for converting the position of user units [u] into drive units [increments] and back.</td>
</tr>
<tr>
<td>It's valid: $p_{[\text{increments}]} = p_{[\text{u}]} \times \text{FactorPosition}$</td>
<td></td>
<td></td>
<td>Please consider the factor which can be specified on the drive via the objects 0x2301: 1 and 0x2301: 2. This should be 1.</td>
</tr>
<tr>
<td>Velocity Factor</td>
<td>INPUT</td>
<td>REAL</td>
<td>Factor for converting the speed of user units [u/s] into drive units [increments/s] and back.</td>
</tr>
<tr>
<td>It's valid: $v_{[\text{increments/s}]} = v_{[\text{u/s}]} \times \text{FactorVelocity}$</td>
<td></td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>FactorAcceleration</td>
<td>INPUT</td>
<td>REAL</td>
<td>Factor to convert the acceleration of user units [u/s^2] in drive units [10^{-4} x increments/s^2] and back.</td>
</tr>
<tr>
<td>It's valid: $a_{[\text{increments/s^2}]} = a_{[\text{u/s^2}]} \times \text{FactorAcceleration}$</td>
<td></td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>OffsetPosition</td>
<td>INPUT</td>
<td>REAL</td>
<td>Offset for the zero position [u].</td>
</tr>
<tr>
<td>MaxVelocityApp</td>
<td>INPUT</td>
<td>REAL</td>
<td>Maximum application speed [u/s]. The command inputs are checked to the maximum value before execution.</td>
</tr>
<tr>
<td>MaxAccelerationApp</td>
<td>INPUT</td>
<td>REAL</td>
<td>Maximum acceleration of the application [u/s^2]. The command inputs are checked to the maximum value before execution.</td>
</tr>
<tr>
<td>MaxDecelerationApp</td>
<td>INPUT</td>
<td>REAL</td>
<td>Maximum application deceleration [u/s^2]. The command inputs are checked to the maximum value before execution.</td>
</tr>
<tr>
<td>MaxPosition</td>
<td>INPUT</td>
<td>REAL</td>
<td>Maximum position for monitoring the software limits [u].</td>
</tr>
<tr>
<td>Parameter</td>
<td>Declaration</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MinPosition</td>
<td>INPUT</td>
<td>REAL</td>
<td>Minimum position for monitoring the software limits [u].</td>
</tr>
<tr>
<td>EnableMaxPosition</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Monitoring maximum position</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TRUE: Activates the monitoring of the maximum position.</td>
</tr>
<tr>
<td>EnableMinPosition</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Monitoring minimum position</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TRUE: Activation of the monitoring of the minimum position.</td>
</tr>
<tr>
<td>MinUserPosition</td>
<td>OUTPUT</td>
<td>REAL</td>
<td>Minimum user position based on the minimum encoder value of 0x80000000 and the FactorPosition [u].</td>
</tr>
<tr>
<td>MaxUserPosition</td>
<td>OUTPUT</td>
<td>REAL</td>
<td>Maximum user position based on the maximum encoder value of 0x7FFFFFFF and the FactorPosition [u].</td>
</tr>
<tr>
<td>Valid</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Initialization</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TRUE: Initialization is valid.</td>
</tr>
<tr>
<td>Error</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TRUE: An error has occurred. Additional error information can be found in the parameter ErrorID. The axis is disabled.</td>
</tr>
<tr>
<td>ErrorID</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>Additional error information</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Chap. 12 ‘ErrorID - Additional error information’ page 457</td>
</tr>
<tr>
<td>Config</td>
<td>IN_OUT</td>
<td>VMC_Config-SigmaPN_REF</td>
<td>Data structure for transferring axis-dependent configuration data to the AxisKernel.</td>
</tr>
<tr>
<td>Axis</td>
<td>IN_OUT</td>
<td>MC_AXIS_REF</td>
<td>Data structure for transferring axis-dependent information to the AxisKernel and PLCopen blocks.</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>Servopack Parameter</th>
<th>Address:digit</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pn000</td>
<td>(2000h:01)</td>
<td>Basic Function Selection Switch 0</td>
<td>1: Position control (pulse train reference)</td>
</tr>
<tr>
<td>Pn002</td>
<td>(2002h:02)</td>
<td>Application Function Select Switch 2</td>
<td>1: Uses absolute encoder as incremental encoder</td>
</tr>
<tr>
<td>Pn200</td>
<td>(2200h:03)</td>
<td>Position Control Reference From Selection Switch</td>
<td>1: Uses reference input filter for open collector signal</td>
</tr>
<tr>
<td>Pn20E</td>
<td>(220Eh)</td>
<td>Electronic Gear Ratio (Numerator)</td>
<td>1024</td>
</tr>
<tr>
<td>Pn216</td>
<td>(2216h)</td>
<td>Position Reference Acceleration / Deceleration Time Constant</td>
<td>0</td>
</tr>
<tr>
<td>Pn217</td>
<td>(2217h)</td>
<td>Average Movement Time of Position Reference</td>
<td>0</td>
</tr>
</tbody>
</table>
### Servopack Parameter

<table>
<thead>
<tr>
<th>Servopack Parameter</th>
<th>Address:digit</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pn50A</td>
<td>(250Ah:02)</td>
<td>/P-CON Signal Mapping</td>
<td>8: Sets signal off</td>
</tr>
<tr>
<td>Pn50A</td>
<td>(250Ah:03)</td>
<td>P-OT Signal Mapping</td>
<td>8: Forward run allowed</td>
</tr>
<tr>
<td>Pn50B</td>
<td>(250Bh:00)</td>
<td>N-OT Signal Mapping</td>
<td>8: Reverse run allowed</td>
</tr>
<tr>
<td>Pn50B</td>
<td>(250Bh:02)</td>
<td>/P-CL Signal Mapping</td>
<td>8: Sets signal off</td>
</tr>
<tr>
<td>Pn50B</td>
<td>(250Bh:03)</td>
<td>/N-CL Signal Mapping</td>
<td>8: Sets signal off</td>
</tr>
</tbody>
</table>

#### 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.
### Usage Sigma-5/7 Pulse Train

#### X2

<table>
<thead>
<tr>
<th>Function</th>
<th>Type</th>
<th>LED</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DO 0.7</td>
<td>O</td>
<td><img src="green.png" alt="Green" /></td>
<td>Digital output DO 7</td>
</tr>
<tr>
<td>DO 0.6</td>
<td>O</td>
<td><img src="green.png" alt="Green" /></td>
<td>Digital output DO 6</td>
</tr>
<tr>
<td>DO 0.2</td>
<td>O</td>
<td><img src="green.png" alt="Green" /></td>
<td>Digital output DO 2</td>
</tr>
<tr>
<td>DO 0.1</td>
<td>O</td>
<td><img src="red.png" alt="Red" /></td>
<td>Pulse Train Channel 1</td>
</tr>
<tr>
<td>DO 0.0</td>
<td>O</td>
<td><img src="green.png" alt="Green" /></td>
<td>Pulse Train Channel 0</td>
</tr>
<tr>
<td>0 V</td>
<td>I</td>
<td><img src="red.png" alt="Red" /></td>
<td>4M: GND for Pulse Train</td>
</tr>
<tr>
<td>DC 24V</td>
<td>I</td>
<td><img src="green.png" alt="Green" /></td>
<td>4L+: DC 24V power supply for Pulse Train</td>
</tr>
</tbody>
</table>

#### X1

<table>
<thead>
<tr>
<th>Function</th>
<th>Type</th>
<th>LED</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DI 0.2</td>
<td>I</td>
<td><img src="green.png" alt="Green" /></td>
<td>Digital input DI 2</td>
</tr>
<tr>
<td>DI 0.0</td>
<td>I</td>
<td><img src="green.png" alt="Green" /></td>
<td>Digital input DI 0</td>
</tr>
<tr>
<td>0 V</td>
<td>I</td>
<td><img src="green.png" alt="Green" /></td>
<td>3M: GND power section supply for on-board DI</td>
</tr>
<tr>
<td>DC 24V</td>
<td>I</td>
<td><img src="green.png" alt="Green" /></td>
<td>3L+: DC 24V power section supply for on-board DI</td>
</tr>
</tbody>
</table>

#### X6

<table>
<thead>
<tr>
<th>Function</th>
<th>Type</th>
<th>LED</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sys DC 24V</td>
<td>I</td>
<td><img src="green.png" alt="Green" /></td>
<td>1L+: DC 24V for electronic section supply</td>
</tr>
<tr>
<td>Sys 0V</td>
<td>I</td>
<td><img src="green.png" alt="Green" /></td>
<td>1M: GND for electronic section supply</td>
</tr>
</tbody>
</table>
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.
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
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.
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:

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

<table>
<thead>
<tr>
<th>Slot</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CPU 314C-2PN/DP</td>
</tr>
<tr>
<td>X1</td>
<td>MPI/DP</td>
</tr>
<tr>
<td>X2</td>
<td>PN-IO</td>
</tr>
<tr>
<td>X2...</td>
<td>Port 1</td>
</tr>
<tr>
<td>X2...</td>
<td>Port 2</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

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 ➔ VI PA ...’ 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.

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’.

Configuration of Ethernet PG/OP channel

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!
3. Leave all values unchanged.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CPU 314C-2 PN/DP</td>
</tr>
<tr>
<td>2</td>
<td>X1 MPI/DP</td>
</tr>
<tr>
<td></td>
<td>X2 PN-IO</td>
</tr>
<tr>
<td>X2 P1 R</td>
<td>Port 1</td>
</tr>
<tr>
<td>X2 P2 R</td>
<td>Port 2</td>
</tr>
<tr>
<td>2.5</td>
<td>DI24/DO16</td>
</tr>
<tr>
<td>2.6</td>
<td>AI5/ AO2</td>
</tr>
<tr>
<td>2.7</td>
<td>Count</td>
</tr>
<tr>
<td>2.8</td>
<td>Position</td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

4. Close the dialog with [OK].
5. Select ‘Station ➔ Save and compile’.
6. Close the hardware configurator.

5.5.3 User program

**Include library**

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:

```plaintext
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
DriveError := MW 126
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. See Chap. 5.7.1 ‘FB 875 - VMC_AxisControl_PT - Axis control via Pulse Train’ page 238
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. 

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:

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.

<table>
<thead>
<tr>
<th>Module</th>
<th>Slot</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLC...</td>
<td>2</td>
<td>CPU 314C-2PN/DP</td>
</tr>
<tr>
<td>MPI interface...</td>
<td>2 X1</td>
<td>MPI/DP interface</td>
</tr>
<tr>
<td>PROFINET interface...</td>
<td>2 X2</td>
<td>PROFINET interface</td>
</tr>
<tr>
<td>DI24/DO16...</td>
<td>2 5</td>
<td>DI24/DO16</td>
</tr>
<tr>
<td>AI5/AO2...</td>
<td>2 6</td>
<td>AI5/AO2</td>
</tr>
<tr>
<td>Count...</td>
<td>2 7</td>
<td>Count</td>
</tr>
</tbody>
</table>

Device overview:

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.
### 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


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_Vxx.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.zalxx*.
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 \textit{FactorPosition} and \textit{FactorVelocity}:

\begin{verbatim}
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
\end{verbatim}

The addresses of \textit{S\_Ready} and \textit{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 \textit{S\_On}, \textit{S\_Direction} and \textit{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 \textit{AxisReady} = TRUE, you can use \textit{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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>S_ChannelNumberPWM</td>
<td>INPUT</td>
<td>INT</td>
<td>Channel number of the PWM output, which is used for the control of the Pulse Train input of the servo (signal PULS).</td>
</tr>
<tr>
<td>S_Ready</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Digital input for connecting the S_Ready signal (S-RDY)</td>
</tr>
<tr>
<td>S_Alarm</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Digital input for connecting the S_Alarm signal (ALM)</td>
</tr>
<tr>
<td>FactorPosition</td>
<td>INPUT</td>
<td>REAL</td>
<td>Factor for converting the position of user units into drive units (increments) and back.</td>
</tr>
<tr>
<td>FactorVelocity</td>
<td>INPUT</td>
<td>REAL</td>
<td>Factor for converting the velocity of user units into drive units (increments) and back.</td>
</tr>
<tr>
<td>AxisEnable</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Enable/disable axis</td>
</tr>
<tr>
<td>AxisReset</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Reset axis</td>
</tr>
<tr>
<td>StopExecute</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Stop axis</td>
</tr>
</tbody>
</table>

Note: StopExecute = 1: No other command can be started!
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MvVelocityExecute</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Start moving the axis</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Edge 0-1: The axis is accelerated / decelerated to the speed specified.</td>
</tr>
<tr>
<td>MvRelativeExecute</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Start moving the axis</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Edge 0-1: The relative positioning of the axis is started.</td>
</tr>
<tr>
<td>JogPositive</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Jog operation positive</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Drive axis with constant velocity in positive direction</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Edge 0-1: Drive axis with constant velocity is started.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Edge 1-0: The axis is stopped.</td>
</tr>
<tr>
<td>JogNegative</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Jog operation negative</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Drive axis with constant velocity in negative direction</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Edge 0-1: Drive axis with constant velocity is started.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Edge 1-0: The axis is stopped.</td>
</tr>
<tr>
<td>PositionDistance</td>
<td>INPUT</td>
<td>REAL</td>
<td>Absolute position or relative distance for MvRelativeExecute in [user units].</td>
</tr>
<tr>
<td>Velocity</td>
<td>INPUT</td>
<td>REAL</td>
<td>Velocity setting (signed value) in [user units / s].</td>
</tr>
<tr>
<td>S_ON</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Digital output for controlling the S_On signal (S-ON)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- TRUE: turns on the servo.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- TRUE: turns off the servo.</td>
</tr>
<tr>
<td>S_Direction</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Digital output for controlling the S_Direction signal (SIGN)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- TRUE: Presetting of the direction of rotation positive direction for the servo.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- FALSE: Presetting of the direction of rotation negative direction for the servo.</td>
</tr>
<tr>
<td>S_AlarmReset</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Digital output for controlling the S_AlarmReset signal (ALM-RST)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- TRUE: Alarms are reset in the servo.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- FALSE: Alarms in the servo remain.</td>
</tr>
<tr>
<td>MinUserDistance</td>
<td>OUTPUT</td>
<td>REAL</td>
<td>Minimum drive distance (1 increment) of the servo [user units].</td>
</tr>
<tr>
<td>MinUserDistance</td>
<td>OUTPUT</td>
<td>REAL</td>
<td>Maximum drive distance (8388607 increments = maximum number of pulses of the PWM output) of the servo [user units].</td>
</tr>
<tr>
<td>MinUserVelocity</td>
<td>OUTPUT</td>
<td>REAL</td>
<td>Minimum speed (period duration = 65535μs = maximum period of the PWM output) of the servo [user units].</td>
</tr>
<tr>
<td>MinUserVelocity</td>
<td>OUTPUT</td>
<td>REAL</td>
<td>Maximum speed (period duration = 20μs = minimum period duration of the PWM output) of the servo [user units].</td>
</tr>
<tr>
<td>AxisReady</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Axis is ready to switch on.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- TRUE: The axis is ready to switch on.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- FALSE: The axis is not ready to switch on.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>→ Check and fix AxisError (see AxisErrorID).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>→ Check and fix DriveError.</td>
</tr>
</tbody>
</table>
### Parameter Declaration Data type Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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.

\[
\text{FactorPosition} = \frac{\text{Resolution}}{\text{Numerator}} \cdot \frac{\text{Denominator}}{\text{Resolution}}
\]

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

\[\text{Resolution} = 2^{20} = 1048576\]

Numerator - Numerator: Electronic Gear Ratio (Pn20E) of the servo parameter

\[\text{Numerator} = 1024\]

Denominator - Denominator: Electronic Gear Ratio (Pn210) of the servo parameter

\[\text{Denominator} = 1\]

\[
\text{FactorPosition} = \frac{\text{Resolution}}{\text{Numerator}} \cdot \frac{\text{Denominator}}{\text{Resolution}} = \frac{1048576}{1024} \cdot 1 = 1024
\]

**Example minimum distance**

MinPos - Minimum distance in rotations

Resolution - Number of increments per user unit

\[\text{Resolution} = 2^{20} = 1048576\]

Numerator - Numerator: Electronic Gear Ratio (Pn20E) of the servo parameter

\[\text{Numerator} = 1024\]

Period - Minimum period

\[\text{Period} = 1\]

\[
\text{MinPos} = \text{Numerator} \cdot \frac{\text{Period}}{\text{Resolution}} = 1024 \cdot \frac{1}{1048576} = \frac{1}{1024}
\]
Example maximum distance

- MaxPos: Maximum distance in revolutions
  \[ \text{Resolution} = 2^{20} = 1048576 \]
- Resolution: Number of increments per user unit
- Numerator: Electronic Gear Ratio (Pn20E) of the servo parameter
  \[ \text{Numerator} = 1024 \]
- Period: Maximum period
  \[ \text{Period} = 8388607 \]

\[ \text{MaxPos} = \frac{\text{Numerator} \times \text{Period}}{\text{Resolution}} = 1024 \times \frac{8388607}{1048576} = 8192 \]

FactorVelocity

The calculation of FactorVelocity is only valid if servo parameter Reference Pulse Multiplier (Pn218) = 1.

\[ \text{FactorVelocity} = \frac{\text{Numerator}}{\text{Denominator} \times \text{Resolution}} \]

- Time: Time for 1 revolution in μs
- Numerator: Electronic Gear Ratio (Pn20E) of the servo parameter
- 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

\[ Time = 1\text{min} = 60 \cdot 10^6\mu s \]

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

\[ Resolution = 2^{20} = 1048576 \]

\[ \frac{\text{Numerator}}{\text{Denominator}} = \frac{1024}{1} = 1024 \]

\[ \frac{\text{Time}}{\text{Resolution}} = \frac{60 \cdot 10^6}{1048576} = 58593.75 \]

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

\[ Time = 1\text{s} = 10^6\mu s \]

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

\[ Resolution = 2^{20} = 1048576 \]

\[ \frac{\text{Numerator}}{\text{Denominator}} = \frac{1024}{1} = 1024 \]

\[ \frac{\text{Time}}{\text{Resolution}} = \frac{10^6}{1048576} = 976.5625 \]
Minimum velocity for revolutions/min

\[
\text{MinVel} = \frac{\text{FactorVelocity}}{65535} = \frac{58593.75}{65535} = 0.89
\]

Maximum velocity for revolutions/min

\[
\text{MaxVel} = \frac{\text{FactorVelocity}}{20} = \frac{58593.75}{20} = 2929.69
\]

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_Move-Velocity

- 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.

\[ \text{MaxVelError} = \frac{\text{FactorVelocity}}{20} - \frac{\text{FactorVelocity}}{21} \]

**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
- Siemens SIMATIC Manager from V 5.5, SP2 & Simple Motion Control Library
- 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.

<table>
<thead>
<tr>
<th>No.</th>
<th>Parameters that differ from the standard</th>
<th>Setting for Simple Motion Control Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>B1-01</td>
<td>Reference selection</td>
<td>4: Pulse train input</td>
</tr>
<tr>
<td>B1-02</td>
<td>Operation method selection</td>
<td>1: Control circuit terminal</td>
</tr>
<tr>
<td>H1-01</td>
<td>Terminal S1 function selection</td>
<td>0040: Forward Run Command</td>
</tr>
<tr>
<td>H1-02</td>
<td>Terminal S2 function selection</td>
<td>0041: Reverse Run Command</td>
</tr>
<tr>
<td>H2-01</td>
<td>Terminal MA/MB-MC selection</td>
<td>000E: Fault</td>
</tr>
<tr>
<td>H2-02</td>
<td>P1 terminal selection</td>
<td>0006</td>
</tr>
<tr>
<td>H6-01</td>
<td>Pulse train input function selection</td>
<td>0: Frequency reference</td>
</tr>
<tr>
<td>H6-02</td>
<td>Pulse train input scaling</td>
<td>20000Hz</td>
</tr>
<tr>
<td>H6-03</td>
<td>Pulse train input gain</td>
<td>100.0%</td>
</tr>
</tbody>
</table>
### Parameters that differ from the standard

<table>
<thead>
<tr>
<th>No.</th>
<th>Parameters that differ from the standard</th>
<th>Setting for <em>Simple Motion Control Library</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>H6-04</td>
<td>Pulse train input bias</td>
<td>0.0%</td>
</tr>
<tr>
<td>H6-05</td>
<td>Pulse train input filter time</td>
<td>0.10s</td>
</tr>
<tr>
<td>H6-06</td>
<td>Pulse train monitor selection</td>
<td>102: Output frequency</td>
</tr>
<tr>
<td>H6-07</td>
<td>Pulse train monitor scaling</td>
<td>20000Hz</td>
</tr>
</tbody>
</table>

### Parameters depending on the application

<table>
<thead>
<tr>
<th>No.</th>
<th>Parameters depending on the application</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1-01</td>
<td>Acceleration time 1</td>
<td>10.00s</td>
</tr>
<tr>
<td>C1-02</td>
<td>Deceleration time 1</td>
<td>10.00s</td>
</tr>
<tr>
<td>C1-10</td>
<td>Accel/Decel time setting unit</td>
<td>0: 0.01- second units</td>
</tr>
<tr>
<td>C1-11</td>
<td>Accel/Decel switching frequency</td>
<td>0.0Hz</td>
</tr>
<tr>
<td>O1-02</td>
<td>Monitor selection after power up</td>
<td>1: Frequency reference</td>
</tr>
<tr>
<td>O1-03</td>
<td>Display scaling</td>
<td>2: min-1 unit</td>
</tr>
</tbody>
</table>

---

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.

<table>
<thead>
<tr>
<th>R</th>
<th>Resistor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value: 4.7kΩ</td>
<td></td>
</tr>
<tr>
<td>Power dissipation: min. 0.25W</td>
<td></td>
</tr>
<tr>
<td>Resistance example: Carbon film resistor 0207 wired with 0.25W power dissipation</td>
<td></td>
</tr>
</tbody>
</table>

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’.
The operating parameters required for PWM are internally adapted to the corresponding values. Leave all values unchanged.

Close the dialog with [OK].

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
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_S2ReverseRun"
   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.

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:

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**

<table>
<thead>
<tr>
<th>Slot</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CPU 314C-2PN/DP</td>
</tr>
<tr>
<td>X1</td>
<td>MPI/DP</td>
</tr>
<tr>
<td>X2</td>
<td>PN-IO</td>
</tr>
<tr>
<td>X2...</td>
<td>Port 1</td>
</tr>
<tr>
<td>X2...</td>
<td>Port 2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

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**

<table>
<thead>
<tr>
<th>Slot</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CPU ...</td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>343-1EX30</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>1</th>
<th>CPU 314C-2 PN/DP</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>MPI/DP</td>
</tr>
<tr>
<td>X1</td>
<td>PN-IO</td>
</tr>
<tr>
<td>X2</td>
<td>Port 1</td>
</tr>
<tr>
<td>X2 P1 R</td>
<td>Port 2</td>
</tr>
<tr>
<td>X2 P2 R</td>
<td>DI24/DO16</td>
</tr>
<tr>
<td>2.5</td>
<td>AI5/AO2</td>
</tr>
<tr>
<td>2.7</td>
<td>Count</td>
</tr>
<tr>
<td>2.8</td>
<td>Position</td>
</tr>
<tr>
<td>3</td>
<td>PROFINET-IO</td>
</tr>
<tr>
<td></td>
<td>VIPA MICRO...</td>
</tr>
</tbody>
</table>

4. Close the dialog with [OK].

5. Select ‘Station ➔ Save and compile’.

6. Close the hardware configurator.

6.5.3 User program

**Include library**

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:

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:

<table>
<thead>
<tr>
<th>Module</th>
<th>Slot</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLC...</td>
<td>2</td>
<td>CPU 314C-2PN/DP</td>
</tr>
<tr>
<td>MPI interface...</td>
<td>2 X1</td>
<td>MPI/DP interface</td>
</tr>
<tr>
<td>PROFINET interface</td>
<td>2 X2</td>
<td>PROFINET interface</td>
</tr>
<tr>
<td>DI24/DO16...</td>
<td>2 5</td>
<td>DI24/DO16</td>
</tr>
<tr>
<td>AI5/AO2...</td>
<td>2 6</td>
<td>AI5/AO2</td>
</tr>
<tr>
<td>Count...</td>
<td>2 7</td>
<td>Count</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Module</th>
<th>...</th>
<th>Slot</th>
<th>...</th>
<th>Type</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLC ...</td>
<td></td>
<td>2</td>
<td></td>
<td>CPU 314C-2PN/DP</td>
<td></td>
</tr>
</tbody>
</table>
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

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_Vxx.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.zalxx.
Copy the following block from the library into the "Program blocks" of the Project tree of your project.

- V1000 PWM
  - FB885 – VMC_AxisControlV1000PWM \( \text{\textit{Chap. 6.7.1 'FB 885 - VMC_Axis-ControlV1000_PWM - Axis control over PWM'}} \) 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:

\[
\begin{align*}
&\text{CALL FB "VMC\_AxisControlV1000PWM", } \\
&\quad \text{"VMC\_AxisCtrlV1000PWM\_885"} \\
&I\_ChannelNumberPWM := \text{"Ax1\_I\_ChannelNumberPWM"} \\
&I\_MA\_Alarm := \text{"Ax1\_MA\_Alarm"} \\
&I\_P1\_Ready := \text{"I\_P1\_Ready"} \\
&\text{MaxVelocityDrive} := 1.000000e+002 \\
&AxisEnable := \text{"Ax1\_AxisEnable"} \\
&AxisReset := \text{"Ax1\_AxisReset"} \\
&\text{StopExecute} := \text{"Ax1\_StopExecute"} \\
&MvVelocityExecute := \text{"Ax1\_MvVelExecute"} \\
&\text{JogPositive} := \text{"Ax1\_JogPositive"} \\
&\text{JogNegative} := \text{"Ax1\_JogNegative"} \\
&\text{Velocity} := \text{"Ax1\_Velocity"} \\
&I\_S1\_ForwardRun := \text{"Ax1\_S1\_ForwardRun"} \\
&I\_S2\_ReverseRun := \text{"Ax1\_S2\_ReverseRun"} \\
&I\_S4\_AlarmReset := \text{"Ax1\_S4\_AlarmReset"} \\
&\text{MinUserVelocity} := \text{"Ax1\_MinUserVelocity"} \\
&\text{MaxUserVelocity} := \text{"Ax1\_MaxUserVelocity"} \\
&AxisReady := \text{"Ax1\_AxisReady"} \\
&AxisEnabled := \text{"Ax1\_AxisEnabled"} \\
&AxisError := \text{"Ax1\_AxisError"} \\
&AxisErrorID := \text{"Ax1\_AxisErrorID"} \\
&\text{DriveError} := \text{"Ax1\_DriveError"} \\
&CmdActive := \text{"Ax1\_CmdActive"} \\
&CmdDone := \text{"Ax1\_CmdDone"} \\
&CmdBusy := \text{"Ax1\_CmdBusy"} \\
&CmdAborted := \text{"Ax1\_CmdAborted"} \\
&CmdError := \text{"Ax1\_CmdError"} \\
&CmdErrorID := \text{"Ax1\_CmdErrorID"}
\end{align*}
\]

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.

\[
\Rightarrow \text{You can take your application into operation now.}
\]

\[\text{CAUTION!}\]

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

2. Bring your CPU into RUN and turn on your drive.

\[
\Rightarrow \text{The FB 875 - VMC\_AxisControl\_PT is executed cyclically.}
\]

3. As soon as \(AxisReady = \text{TRUE}\), 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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I_Channel-NumberPWM</td>
<td>INPUT</td>
<td>INT</td>
<td>Channel number of the PWM output used to drive the PWM input of the inverter drive.</td>
</tr>
<tr>
<td>I_MA_Alarm</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Digital input for connecting the I_MA_Alarm signal (MA)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TRUE: The inverter drive has detected an error.</td>
</tr>
<tr>
<td>I_P1_Ready</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Digital input for connecting the I_P1_Ready signal</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>FALSE: The inverter drive is ready.</td>
</tr>
<tr>
<td>MaxVelocity-Drive</td>
<td>INPUT</td>
<td>REAL</td>
<td>Maximum speed of the inverter drive [user units].</td>
</tr>
<tr>
<td>AxisEnable</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Enable/disable axis</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TRUE: The axis is enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>FALSE: The axis is disabled.</td>
</tr>
<tr>
<td>AxisReset</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Reset axis</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Edge 0-1: Axis reset is performed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The status of a reset, started with AxisReset, is not indicated at the outputs CmdActive, CmdDone, CmdBusy, CmdAborted, CmdError and CmdErrorID.</td>
</tr>
<tr>
<td>StopExecute</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Stop axis</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Edge 0-1: Stopping of the axis is started.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Note: StopExecute = 1: No other command can be started!</td>
</tr>
<tr>
<td>MvVelocityExecute</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Start moving the axis</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Edge 0-1: The axis is accelerated/decelerated to the speed specified.</td>
</tr>
<tr>
<td>JogPositive</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Jog operation positive</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Drive axis with constant velocity in positive direction</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Edge 0-1: Drive axis with constant velocity is started.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Edge 1-0: The axis is stopped.</td>
</tr>
<tr>
<td>JogNegative</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Jog operation negative</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Drive axis with constant velocity in negative direction</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Edge 0-1: Drive axis with constant velocity is started.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Edge 1-0: The axis is stopped.</td>
</tr>
<tr>
<td>Velocity</td>
<td>INPUT</td>
<td>REAL</td>
<td>Velocity setting (signed value) in [user units / s].</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Note: JogPositive and JogNegative use the absolute value of the speed.</td>
</tr>
<tr>
<td>I_S1_ForwardRun</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Digital output for controlling the inverter drive signal S1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TRUE: Enables the inverter drive in positive direction.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Declaration</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------------</td>
<td>-----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| I_S2_ReversRun            | OUTPUT      | BOOL      | Digital output for controlling the inverter drive signal S2  
- TRUE: Enables the inverter drive in negative direction.                                                                                     |
| I_S4_AlarmReset          | 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.                                                                                       |
| MinUserVelocity           | OUTPUT      | REAL      | Minimum speed (period duration = 65535μs = maximum period of the PWM output) of the inverter drive [user units].                                                                                         |
| MinUserVelocity           | 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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**

\[ n = 2 \cdot 60 \cdot \frac{f_{\text{max}, \text{out}}}{\text{poles}} \cdot \frac{l}{\text{min}} \]

- Maximum frequency (parameter E1-04)
- Number of motor poles (parameter E5-04)
- 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
- Siemens SIMATIC Manager from V5.5, SP2 & Simple Motion Control Library
- Siemens TIA Portal V14 & 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.

<table>
<thead>
<tr>
<th>No.</th>
<th>Designation</th>
<th>Range of values</th>
<th>Setting for Simple Motion Control Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>H5-01</td>
<td>Slave address inverter drive</td>
<td>00h, 20h</td>
<td>By default, the slave address is set to 1Fh. Please note that addresses in the network must not be assigned more than once!</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H5-02</td>
<td>Communication speed MEMOBUS/Modbus</td>
<td>0, 1, 2, ..., 8</td>
<td>3: 9600bit/s</td>
</tr>
<tr>
<td>H5-03</td>
<td>Transmission parity MEMOBUS/Modbus</td>
<td>0, 1, 2</td>
<td>0: no parity</td>
</tr>
</tbody>
</table>
### Simple Motion Control Library

<table>
<thead>
<tr>
<th>No.</th>
<th>Designation</th>
<th>Range of values</th>
<th>Setting for Simple Motion Control Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>H5-04</td>
<td>Stop method after communication error (CE error)</td>
<td>0, 1, 2, 3</td>
<td>3: Operation continues with alarm</td>
</tr>
<tr>
<td>H5-05</td>
<td>Stop method after communication error (CE error)</td>
<td>0, 1</td>
<td>1: Activated - If the connection is aborted for longer than 2s (adjustable via H2-09), a CE error is triggered.</td>
</tr>
<tr>
<td>H5-06</td>
<td>Waiting time between receiving and sending data from the inverter drive</td>
<td>5 ... 65ms</td>
<td>5ms</td>
</tr>
<tr>
<td>H5-07</td>
<td>Request to send (RTS) control</td>
<td>0, 1</td>
<td>1: Activated - RTS is activated only when sending (RS485 or RS422 and multi-drop)</td>
</tr>
<tr>
<td>H5-09</td>
<td>Time after which a communication error (CE error) is detected.</td>
<td>0,0 ... 10,0s</td>
<td>2s</td>
</tr>
<tr>
<td>H5-10</td>
<td>Step size (resolution) for the MEMOBUS/Modbus register 0025h</td>
<td>0, 1</td>
<td>By default, the resolution is set to 0.1V increments (0). By default, the resolution is set to 0.1V increments (0). 0: 0.1V increments 1: 1V increments</td>
</tr>
<tr>
<td>H5-11</td>
<td>ENTER function for connections</td>
<td>0, 1</td>
<td>1: Enter command not required</td>
</tr>
<tr>
<td>H5-12</td>
<td>Selection start command method</td>
<td>0, 1</td>
<td>1: Run/Stop</td>
</tr>
<tr>
<td>B1-01</td>
<td>Input source frequency setpoint 1</td>
<td>0, 1, 2, 3, 4</td>
<td>2: MEMOBUS/Modbus communication</td>
</tr>
<tr>
<td>B1-02</td>
<td>Input source start command 1</td>
<td>0, 1, 2, 3</td>
<td>2: MEMOBUS/Modbus communication</td>
</tr>
<tr>
<td>B1-15</td>
<td>Input source frequency setpoint 2</td>
<td>0, 1, 2, 3, 4</td>
<td>2: MEMOBUS/Modbus communication</td>
</tr>
<tr>
<td>B1-16</td>
<td>Input source start command 2</td>
<td>0, 1, 2, 3</td>
<td>2: MEMOBUS/Modbus communication</td>
</tr>
</tbody>
</table>

For all settings to be accepted, you must restart the inverter drive after parametrization!
7.3 Wiring

RS485 cabling

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 (Point-to-Point).

- 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

<table>
<thead>
<tr>
<th>CPU</th>
<th>Connection</th>
<th>Comment</th>
</tr>
</thead>
</table>
| MICRO CPU M13C  | ![Connection Diagram](image1) | - 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 | ![Connection Diagram](image2) | - 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. |
| System SLIO CPU 014 ... 017 | ![Connection Diagram](image3) | - 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. |

---

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
## Connection of the YASKAWA inverter drives

### Wiring

<table>
<thead>
<tr>
<th>FU</th>
<th>Connection continuous</th>
<th>Connection termination</th>
</tr>
</thead>
<tbody>
<tr>
<td>J1000</td>
<td><img src="image1.png" alt="Diagram" /></td>
<td><img src="image2.png" alt="Diagram" /></td>
</tr>
<tr>
<td>V1000</td>
<td><img src="image3.png" alt="Diagram" /></td>
<td><img src="image4.png" alt="Diagram" /></td>
</tr>
<tr>
<td>A1000</td>
<td><img src="image5.png" alt="Diagram" /></td>
<td><img src="image6.png" alt="Diagram" /></td>
</tr>
<tr>
<td>GA700</td>
<td><img src="image7.png" alt="Diagram" /></td>
<td><img src="image8.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>

*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.
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.
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.
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

- FB 880 - VMC_WriteParameter_RTU
  – UDT 879 - VMC_AxisRTU_REF

- FB 877 - VMC_CmdManager_RTU
  – UDT 878 - VMC_CmdSlaveRefRTU_REF

- DB 99 ComDataSlaves
  – UDT 878 - VMC_CmdHeader1RTU_REF
  – ARRAY [1 ... 8] of
    – UDT 877 - VMC_CmdSlaveRefRTU_REF

- DB100.AY_V1000
  – UDT 880 - VMC_CmdSlaveRefRTU_REF
  – UDT 879 - VMC_AxisRTU_REF

- UDT 881 - VMC_ConfigG1000RTU_REF

- FB 887 - VMC_CmdManager_RTU
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**
  - 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
  - UDT 879 - VMC_AXISRTU_REF

- **FB 879 - VMC_ReadParameter_RTU**
  - 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

- **FB 880 - VMC_WriteParameter_RTU**
  - 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

- **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
  - UDT 881 - VMC_ConfigV1000RTU_REF

- **FB 882 - VMC_AxisControlV1000_RTU**
  - 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
  - UDT 879 - VMC_AXISRTU_REF
  - UDT 878 - VMC_ComObjectRTU_REF

- **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
  - UDT 878 - VMC_ComObjectRTU_REF

- **FB 877 - VMC_ComManager_RTU**
  - 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
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
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baudrate</td>
<td>:= B#16#09</td>
<td>Baud rate: 09h (9600bit/s)</td>
<td>IN: BYTE</td>
</tr>
<tr>
<td>CharLen</td>
<td>:= B#16#03</td>
<td>Number data bits: 03h (8bit)</td>
<td>IN: BYTE</td>
</tr>
<tr>
<td>Parity</td>
<td>:= B#16#00</td>
<td>Parity: 0 (none)</td>
<td>IN: BYTE</td>
</tr>
<tr>
<td>StopBits</td>
<td>:= B#16#01</td>
<td>Stop bits: 1 (1bit)</td>
<td>IN: BYTE</td>
</tr>
<tr>
<td>TimeOut</td>
<td>:= W#16#1FFF</td>
<td>Error wait time: 1FFFh (high selected)</td>
<td>IN: WORD</td>
</tr>
<tr>
<td>Valid</td>
<td>:= &quot;ModbusConfigValid&quot;</td>
<td>Configuration</td>
<td>OUT BOOL</td>
</tr>
<tr>
<td>Error</td>
<td>:= &quot;ModbusConfigError&quot;</td>
<td>Error feedback</td>
<td>OUT BOOL</td>
</tr>
<tr>
<td>ErrorID</td>
<td>:= &quot;ModbusConfigErrorID&quot;</td>
<td>Additional error information</td>
<td>OUT: WORD</td>
</tr>
</tbody>
</table>

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

HB00 | OPL_SP7-LIB | SW90MS0MA V10.017 | en | 20-18
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 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
### 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 velocities:
  - 0: Hz
    
    Specified in hertz
  - 1: %
    
    Specified as a percentage of the maximum speed $= 2\frac{f_{\text{max}}}{P}$
    
    with $f_{\text{max}}$: 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.

---

**VIPA SPEED7 Library**

**Usage inverter drive via Modbus RTU**

**Usage in VIPA SPEED7 Studio > User program**

```plaintext
Hardware := "A1_InitHardware" // Specification of the hardware, used
           // 1: System SLIO CP040, 2: SPEED7 CPU
Laddr := "A1_InitLaddr" // Logical address when using CP040
UnitId := "A1_InitUnitId" // Modbus address of the V1000
UserUnitsVelocity := "A1_InitUserUnitsVel" // User unit for velocities:
                        // 0: Hz, 1: %, 2: RPM
UserUnitsAcceleration := "A1_InitUserUnitsAcc" // User units acceleration/deceleration
MaxVelocityApp := "A1_InitMaxVelocityApp" // Max. velocity in user units
Done := "A1_InitDone" // Status job finished
Busy := "A1_InitBusy" // Status job in progress
Error := "A1_InitError" // Error feedback
ErrorID := "A1_InitErrorID" // Additional error information
Axis := "A1_V1000".AxisData // Reference to the general axis data
V1000 := "A1_V1000".V1000Data // Reference to the drive-specific data
```

---

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

285
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
   Velocity   := "A1_Velocity"    // Parameter: Velocity setting for MoveVelocity  IN: REAL
   AccelerationTime := "A1_AccelerationTime" // Parameter: Acceleration time  IN: REAL
   JogNegative := "A1_JogNegative" // Command: JogNeg  IN: BOOL
   AxisReady  := "A1_AxisReady"    // Status: Axis ready  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
   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  IN-OUT: UDT 881
   // of the inverter drive
   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**  := "Al_RdParExecute"  // The job is started with edge 0-1.  IN: BOOL
   **StartAddress**  := "Al_RdParStartAddress"  // Start address of the 1. register  IN: INT
   **Quantity**  := "Al_RdParQuantity"  // Number of registers to read  IN: INT
   **Done**  := "Al_RdParDone"  // Status job finished  IN: REAL
   **Busy**  := "Al_RdParBusy"  // Status job in progress  OUT: BOOL
   **Error**  := "Al_RdParError"  // Error feedback  OUT: BOOL
   **ErrorID**  := "Al_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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute</td>
<td>The job is started with edge 0-1.</td>
<td>IN:</td>
</tr>
<tr>
<td>StartAddress</td>
<td>Start address of the 1. register</td>
<td>IN:</td>
</tr>
<tr>
<td>Quantity</td>
<td>Number of registers to write</td>
<td>IN:</td>
</tr>
<tr>
<td>Done</td>
<td>Status job finished</td>
<td>IN:</td>
</tr>
<tr>
<td>Busy</td>
<td>Status job in progress</td>
<td>OUT:</td>
</tr>
<tr>
<td>Error</td>
<td>Error feedback</td>
<td>OUT:</td>
</tr>
<tr>
<td>ErrorID</td>
<td>Additional error information</td>
<td>OUT:</td>
</tr>
<tr>
<td>Data</td>
<td>Location of the parameter data</td>
<td>OUT:</td>
</tr>
<tr>
<td>Axis</td>
<td>Reference to the general axis data</td>
<td>IN-OUT:</td>
</tr>
</tbody>
</table>

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 Axis-Ready = 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:

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

<table>
<thead>
<tr>
<th>Slot</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CPU 314C-2PN/DP</td>
</tr>
<tr>
<td>X1</td>
<td>MPI/DP</td>
</tr>
<tr>
<td>X2</td>
<td>PN-IO</td>
</tr>
<tr>
<td>X2...</td>
<td>Port 1</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

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’.

<table>
<thead>
<tr>
<th>Slot</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CPU ...</td>
</tr>
<tr>
<td>X...</td>
<td>PN-IO</td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

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.
Navigate in the hardware catalog to the directory ‘PROFINET IO è Additional field devices è I/O è VIKA ...’ 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 ‘VIKA MICRO PLC’ the CPU is already placed at slot 0.

## Configuration of Ethernet PG/OP channel

<table>
<thead>
<tr>
<th>Slot</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CPU ... PN-IO</td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>343-1EX30</td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

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.
Usage in Siemens SIMATIC Manager > Hardware configuration

7.5.2.2 Hardware configuration System SLIO CPU 013C

Add CPU in the project

<table>
<thead>
<tr>
<th>Slot</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CPU 314C-2PN/DP</td>
</tr>
<tr>
<td>X1</td>
<td>MPI/DP</td>
</tr>
<tr>
<td>X2</td>
<td>PN-IO</td>
</tr>
<tr>
<td>X2...</td>
<td>Port 1</td>
</tr>
<tr>
<td>X2...</td>
<td>Port 2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

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’.

<table>
<thead>
<tr>
<th>Slot</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CPU ... PROFINET-IO-System</td>
</tr>
<tr>
<td>X...</td>
<td>PN-IO</td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

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’.
1. Place 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**

<table>
<thead>
<tr>
<th>Slot</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CPU 315-2 PN/DP</td>
</tr>
<tr>
<td></td>
<td>X1</td>
</tr>
<tr>
<td></td>
<td>X2</td>
</tr>
<tr>
<td></td>
<td>X2... Port 1</td>
</tr>
<tr>
<td></td>
<td>X2... Port 2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Slot</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CPU ... PROFINET-IO-System</td>
</tr>
<tr>
<td></td>
<td>PN-IO</td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Slot</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CPU ... PN-IO</td>
</tr>
<tr>
<td>2</td>
<td>X...</td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>343-1EX30</td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

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** ➔ 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.
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**  
  - 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  
  - UDT 879 - VMC_AxisRTU_REF

- **FB 879 - VMC_ReadParameter_RTU**  
  - 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

- **FB 880 - VMC_WriteParameter_RTU**  
  - 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

- **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
  - UDT 881 - VMC_ConfigV1000RTU_REF

- **FB 882 - VMC_AxisControlV1000_RTU**  
  - 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
  - UDT 879 - VMC_AxisRTU_REF
  - UDT 878 - VMC_ComObjectRTU_REF

- **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
  - UDT 878 - VMC_ComObjectRTU_REF

- **FB 877 - VMC_ComManager_RTU**  
  - 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
7.5.3.2 Copy blocks into project

Include library

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
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 := "Al_InitExecute" // The job is started with edge 0-1. IN: BOOL
   - Hardware := "Al_InitHardware" // Specification of the hardware, used IN: BYTE
     // 1: System SLIO CP040, 2: SPEED7 CPU
   - Laddr := "Al_InitLaddr" // Logical address when using CP040 IN: INT
   - UnitId := "Al_InitUnitId" // Modbus address of the V1000 IN: BYTE
   - UserUnitsVelocity := "Al_InitUserUnitsVel" // User unit for velocities:
     // 0: Hz, 1: %, 2: RPM IN: INT
   - UserUnitsAcceleration := "Al_InitUserUnitsAcc" // User units acceleration/deceleration
     // 0: 0.01s, 1: 0.1s IN: INT
   - MaxVelocityApp := "Al_InitMaxVelocityApp" // Max. velocity in user units IN: REAL
   - Done := "Al_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
    \[ \frac{2 \times f_{\text{max}}}{P} \]
    with \( f_{\text{max}} \): 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:
StopExecute := "A1_StopExecute" // Command: Stop - Stop axis
Velocity := "A1_Velocity" // Parameter: Velocity setting for MoveVelocity
AccelerationTime := "A1_AccelerationTime" // Parameter: Acceleration time
JogNegative := "A1_JogNegative" // Command: JogNeg
JogVelocity := "A1_JogVelocity" // Parameter: Velocity setting for jogging
AxisReady := "A1_AxisReady" // Status: Axis ready
AxisEnabled := "A1_AxisEnabled" // Status: Activation of the axis
AxisError := "A1_AxisError" // Status: Axis error
AxisErrorID := "A1_AxisErrorID" // Status: Additional error information for AxisError
DriveError := "A1_DriveError" // Status: Error on the inverter drive
ActualVelocity := "A1_ActualVelocity" // Status: Current velocity
InVelocity := "A1_InVelocity" // Status: Target velocity
CmdDone := "A1_CmdDone" // Status: Command finished
CmdBusy := "A1_CmdBusy" // Status: Command in progress
CmdAborted := "A1_CmdAborted" // Status: Command aborted
CmdError := "A1_CmdError" // Status: Command error
CmdErrorID := "A1_CmdErrorID" // Status: Additional error information for CmdError
CmdActive := "A1_CmdActive" // Status: Active command
DirectionPositive := "A1_DirectionPositive" // Status: Direction of rotation positive
DirectionNegative := "A1_DirectionNegative" // Status: Direction of rotation negative
Axis := "A1_V1000".AxisData // Reference to the general axis data
V1000 := "A1_V1000".V1000Data // Reference to the general axis data of the inverter drive
AxisComData := "ComDataSlaves".SlavesSlave{1} // Reference to the communication data

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute</td>
<td>The job is started with edge 0-1.</td>
<td>IN: BOOL</td>
</tr>
<tr>
<td>StartAddress</td>
<td>Start address of the 1. register</td>
<td>IN: INT</td>
</tr>
<tr>
<td>Quantity</td>
<td>Number of registers to read</td>
<td>IN: INT</td>
</tr>
<tr>
<td>Done</td>
<td>Status job finished</td>
<td>IN: REAL</td>
</tr>
<tr>
<td>Busy</td>
<td>Status job in progress</td>
<td>OUT: BOOL</td>
</tr>
<tr>
<td>Error</td>
<td>Error feedback</td>
<td>OUT: BOOL</td>
</tr>
<tr>
<td>ErrorID</td>
<td>Additional error information</td>
<td>OUT: BOOL</td>
</tr>
<tr>
<td>Data</td>
<td>Location of the parameter data</td>
<td>OUT: WORD</td>
</tr>
<tr>
<td>Axis</td>
<td>Reference to the general axis data</td>
<td>IN-OUT: UDT 879</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute</td>
<td>:= &quot;A1_WrParExecute&quot; // The job is started with edge 0-1.</td>
</tr>
<tr>
<td>StartAddress</td>
<td>:= &quot;A1_WrParStartAddress&quot; // Start address of the 1. register</td>
</tr>
<tr>
<td>Quantity</td>
<td>:= &quot;A1_WrParQuantity&quot; // Number of registers to write</td>
</tr>
<tr>
<td>Done</td>
<td>:= &quot;A1_WrParDone&quot; // Status job finished</td>
</tr>
<tr>
<td>Busy</td>
<td>:= &quot;A1_WrParBusy&quot; // Status job in progress</td>
</tr>
<tr>
<td>Error</td>
<td>:= &quot;A1_WrParError&quot; // Error feedback</td>
</tr>
<tr>
<td>ErrorID</td>
<td>:= &quot;A1_WrParErrorID&quot; // Additional error information</td>
</tr>
<tr>
<td>Data</td>
<td>:= $DB98.DBX0.0 BYTES 8 // Location of the parameter data</td>
</tr>
<tr>
<td>Axis</td>
<td>:= &quot;A1_V1000&quot;.AxisData // Reference to the general axis data</td>
</tr>
</tbody>
</table>

IN: BOOL

IN: INT

IN: REAL

OUT: BOOL

OUT: BOOL

OUT: WORD

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. 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.

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:

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:

<table>
<thead>
<tr>
<th>Module</th>
<th>Slot</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLC...</td>
<td>2</td>
<td>CPU 314C-2PN/DP</td>
</tr>
<tr>
<td>MPI interface...</td>
<td>2 X1</td>
<td>MPI/DP interface</td>
</tr>
<tr>
<td>PROFINET interface...</td>
<td>2 X2</td>
<td>PROFINET interface</td>
</tr>
<tr>
<td>DI24/DO16...</td>
<td>2 5</td>
<td>DI24/DO16</td>
</tr>
<tr>
<td>AI5/AO2...</td>
<td>2 6</td>
<td>AI5/AO2</td>
</tr>
</tbody>
</table>
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.

Device overview

<table>
<thead>
<tr>
<th>Module</th>
<th>Slot</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLC</td>
<td>2</td>
<td>CPU 314C-2PN/DP</td>
</tr>
<tr>
<td>MPI/DP interface</td>
<td>2 X1</td>
<td>MPI/DP interface</td>
</tr>
<tr>
<td>PROFINET interface</td>
<td>2 X2</td>
<td>PROFINET interface</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CP 343-1</td>
<td>4</td>
<td>CP 343-1</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>Module</th>
<th>Slot</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLC...</td>
<td>2</td>
<td>CPU 314C-2PN/DP</td>
</tr>
<tr>
<td>MPI interface...</td>
<td>2 X1</td>
<td>MPI/DP interface</td>
</tr>
<tr>
<td>PROFINET interface</td>
<td>2 X2</td>
<td>PROFINET interface</td>
</tr>
<tr>
<td>DI24/DO16...</td>
<td>2 5</td>
<td>DI24/DO16</td>
</tr>
<tr>
<td>AI5/AO2...</td>
<td>2 6</td>
<td>AI5/AO2</td>
</tr>
<tr>
<td>Count...</td>
<td>2 7</td>
<td>Count</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.
Usage inverter drive via Modbus RTU

Usage in Siemens TIA Portal > Hardware configuration

### Device overview

<table>
<thead>
<tr>
<th>Module</th>
<th>Slot</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLC</td>
<td>2</td>
<td>CPU 315-2 PN/DP</td>
</tr>
<tr>
<td>MPI/DP interface</td>
<td>2 X1</td>
<td>MPI/DP interface</td>
</tr>
<tr>
<td>PROFINET interface</td>
<td>2 X2</td>
<td>PROFINET interface</td>
</tr>
<tr>
<td>CP 343-1</td>
<td>4</td>
<td>CP 343-1</td>
</tr>
</tbody>
</table>

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**

<table>
<thead>
<tr>
<th>Module</th>
<th>Slot</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLC</td>
<td>2</td>
<td>CPU 315-2 PN/DP</td>
</tr>
<tr>
<td>MPI/DP interface</td>
<td>2 X1</td>
<td>MPI/DP interface</td>
</tr>
<tr>
<td>PROFINET interface</td>
<td>2 X2</td>
<td>PROFINET interface</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**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.
Device overview

<table>
<thead>
<tr>
<th>Module</th>
<th>Slot</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLC</td>
<td>2</td>
<td>CPU 315-2 PN/DP</td>
</tr>
<tr>
<td>MPI/DP interface</td>
<td>2 X1</td>
<td>MPI/DP interface</td>
</tr>
<tr>
<td>PROFINET interface</td>
<td>2 X2</td>
<td>PROFINET interface</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CP 343-1</td>
<td>4</td>
<td>CP 343-1</td>
</tr>
</tbody>
</table>

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

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_Vxx.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.zalxx.
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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>NumberOfSlaves</td>
<td>1</td>
</tr>
<tr>
<td>WaitCycles</td>
<td>&quot;ComWaitCycles&quot;</td>
</tr>
<tr>
<td>SlavesComData</td>
<td>&quot;ComDataSlaves.Slave&quot;</td>
</tr>
</tbody>
</table>
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  

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute</td>
<td>&quot;A1_InitExecute&quot;  // The job is started with edge 0-1.</td>
</tr>
<tr>
<td>Hardware</td>
<td>&quot;A1_InitHardware&quot;  // Specification of the hardware, used</td>
</tr>
<tr>
<td></td>
<td>1: System SLIO CP040, 2: SPEED7 CPU</td>
</tr>
<tr>
<td>Laddr</td>
<td>&quot;A1_InitLaddr&quot;  // Logical address when using CP040</td>
</tr>
<tr>
<td>UnitId</td>
<td>&quot;A1_InitUnitId&quot;  // Modbus address of the V1000</td>
</tr>
<tr>
<td>UserUnitsVelocity</td>
<td>&quot;A1_InitUserUnitsVel&quot;  // User unit for velocities:</td>
</tr>
<tr>
<td></td>
<td>0: Hz, 1: %, 2: RPM</td>
</tr>
<tr>
<td>UserUnitsAcceleration</td>
<td>&quot;A1_InitUserUnitsAcc&quot;  // User units acceleration/deceleration</td>
</tr>
<tr>
<td></td>
<td>0: 0.01s, 1: 0.1s</td>
</tr>
<tr>
<td>MaxVelocityApp</td>
<td>&quot;A1_InitMaxVelocityApp&quot;  // Max. velocity in user units</td>
</tr>
<tr>
<td>Done</td>
<td>&quot;A1_InitDone&quot;  // Status job finished</td>
</tr>
<tr>
<td>Busy</td>
<td>&quot;A1_InitBusy&quot;  // Status job in progress</td>
</tr>
<tr>
<td>Error</td>
<td>&quot;A1_InitError&quot;  // Error feedback</td>
</tr>
<tr>
<td>ErrorID</td>
<td>&quot;A1_InitErrorID&quot;  // Additional error information</td>
</tr>
<tr>
<td>Axis</td>
<td>&quot;A1_V1000&quot;.AxisData  // Reference to the general axis data</td>
</tr>
<tr>
<td>V1000</td>
<td>&quot;A1_V1000&quot;.V1000Data  // Reference to the drive-specific data</td>
</tr>
</tbody>
</table>

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.
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
Velocity := "A1_Velocity" // Parameter: Velocity setting for MoveVelocity  IN: REAL
AccelerationTime := "A1_AccelerationTime" // Parameter: Acceleration time  IN: REAL
JogNegative := "A1_JogNegative" // Command: JogNeg  IN: BOOL
AxisReady := "A1_AxisReady" // Status: Axis ready  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
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".SlavesSlave{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

VIPA SPEED7 Library
Usage inverter drive via Modbus RTU
Usage in Siemens TIA Portal > User program
Usage inverter drive via Modbus RTU

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 Axis-Ready = 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baudrate</td>
<td>INPUT</td>
<td>BYTE</td>
<td>Speed of data transmission in bit/s (baud).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>04h: 1200baud</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>05h: 1800baud</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>06h: 2400baud</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>07h: 4800baud</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>08h: 7200baud</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>09h: 9600baud</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0Ah: 14400baud</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0Bh: 19200baud</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0Ch: 38400baud</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0Dh: 57600baud</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0Eh: 115200baud</td>
</tr>
<tr>
<td>Parameter</td>
<td>Declaration</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>CharLen</td>
<td>INPUT</td>
<td>BYTE</td>
<td>Number of data bits to which a character is mapped</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0: 5bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: 6bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2: 7bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3: 8bit</td>
</tr>
<tr>
<td>Parity</td>
<td>INPUT</td>
<td>BYTE</td>
<td>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 (&quot;0&quot; or &quot;1&quot;) adds the value of all bits to an agreed state. If no parity is specified, the parity bit is set to &quot;1&quot; but not evaluated.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0: None</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: Odd</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2: Even</td>
</tr>
<tr>
<td>StopBits</td>
<td>INPUT</td>
<td>BYTE</td>
<td>The stop bits are added to each character to be transmitted and signalize the end of a character</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: 1bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2: 1.5bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3: 2bit</td>
</tr>
<tr>
<td>TimeOut</td>
<td>INPUT</td>
<td>WORD</td>
<td>Waiting time until an error is generated if a slave does not respond.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Example: Desired time 8ms at a baud rate of 19200bit/s</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Calculation: 19200bit/s x 0.008s ≈ 154bit &gt;&gt;&gt;&gt; (9Ah)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The hex value should be 9Ah.</td>
</tr>
<tr>
<td>Valid</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Configuration</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TRUE: The configuration is valid.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>FALSE: The configuration is not valid.</td>
</tr>
<tr>
<td>Error</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Error feedback</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TRUE: An error has occurred - see ErrorID.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>FALSE: There is no error.</td>
</tr>
<tr>
<td>ErrorID</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>Additional error information</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Chap. 12 ‘ErrorID - Additional error information’ page 457</td>
</tr>
</tbody>
</table>

### 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!
### 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!

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute</td>
<td>IN</td>
<td>BOOL</td>
<td>The job is started with edge 0-1.</td>
</tr>
<tr>
<td>StartAddress</td>
<td>IN</td>
<td>WORD</td>
<td>Start address of the register from which to read.</td>
</tr>
<tr>
<td>Quantity</td>
<td>IN</td>
<td>BYTE</td>
<td>Number of registers to read.</td>
</tr>
<tr>
<td>Done</td>
<td>OUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ TRUE: Job successfully done</td>
</tr>
<tr>
<td>Busy</td>
<td>OUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ TRUE: Job is running</td>
</tr>
<tr>
<td>Error</td>
<td>OUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ TRUE: An error has occurred. Additional error information can be found in the parameter ErrorID.</td>
</tr>
<tr>
<td>ErrorID</td>
<td>OUT</td>
<td>WORD</td>
<td>Additional error information</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Chap. 12 'ErrorID - Additional error information' page 457</td>
</tr>
<tr>
<td>Data</td>
<td>IN-OUT</td>
<td>ANY</td>
<td>Reference where to store the read data</td>
</tr>
<tr>
<td>Axis</td>
<td>IN-OUT</td>
<td>UDT 879</td>
<td>Reference to the general axis data of the inverter drive</td>
</tr>
</tbody>
</table>
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!*  

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute</td>
<td>INPUT</td>
<td>BOOL</td>
<td>The job is started with edge 0-1.</td>
</tr>
<tr>
<td>StartAddress</td>
<td>INPUT</td>
<td>WORD</td>
<td>Start address of the register from which to write.</td>
</tr>
<tr>
<td>Quantity</td>
<td>INPUT</td>
<td>BYTE</td>
<td>Number of registers to write.</td>
</tr>
</tbody>
</table>
| 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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute</td>
<td>INPUT</td>
<td>BOOL</td>
<td>The job is started with edge 0-1.</td>
</tr>
</tbody>
</table>
| 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UserUnitsVelocity</td>
<td>INPUT</td>
<td>INT</td>
<td>User unit for speeds&lt;br&gt; 0: Hz&lt;br&gt; 1: %&lt;br&gt; = 2*f&lt;sub&gt;max&lt;/sub&gt;/p&lt;br&gt; with f&lt;sub&gt;max&lt;/sub&gt;: max. output frequency (parameter E1-04)&lt;br&gt; p: Number of motor poles (motor-dependent parameter E2-04, E4-04 or E5-04)&lt;br&gt; 2: RPM&lt;br&gt; 3: Data in revolutions per minute</td>
</tr>
<tr>
<td>UserUnitsAcceleration</td>
<td>INPUT</td>
<td>INT</td>
<td>User units for acceleration and deceleration&lt;br&gt; 0: 0.01s (range of values: 0.00s - 600.00s)&lt;br&gt; 1: 0.1s (range of values: 0.0 - 6000.0s)</td>
</tr>
<tr>
<td>MaxVelocityApp</td>
<td>INPUT</td>
<td>REAL</td>
<td>Max. speed for the application. The specification must be made in user units and is used for synchronization in movement commands.</td>
</tr>
<tr>
<td>Done</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status&lt;br&gt; TRUE: Job successfully done</td>
</tr>
<tr>
<td>Busy</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status&lt;br&gt; TRUE: Job is running</td>
</tr>
<tr>
<td>Error</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status&lt;br&gt; TRUE: An error has occurred. Additional error information can be found in the parameter ErrorID.</td>
</tr>
<tr>
<td>ErrorID</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>Additional error information&lt;br&gt; Chap. 12 'ErrorID - Additional error information' page 457</td>
</tr>
<tr>
<td>Axis</td>
<td>IN_OUT</td>
<td>UDT 879</td>
<td>Reference to the general axis data of the inverter drive</td>
</tr>
<tr>
<td>V1000</td>
<td>IN_OUT</td>
<td>UDT 881</td>
<td>Reference to the user data of the inverter drive</td>
</tr>
</tbody>
</table>
With the FB 882 VMC_AxisControlV1000_RTU you can control an inverter drive, which is serially connected via Modbus RTU and check its status. PLCopen blocks are not supported!

### Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AxisEnable</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Activation of the axis</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TRUE: Switch on axis $\rightarrow$ AxisEnabled = 1, commands can be executed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>FALSE: Switch off the axis $\rightarrow$ AxisEnabled = 0, no commands can be executed.</td>
</tr>
<tr>
<td>AxisReset</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Command: Reset inverter drive faults. $\rightarrow$ CmdActive = 1</td>
</tr>
<tr>
<td>StopExecute</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Command: Stop - Stop axis $\rightarrow$ CmdActive = 1</td>
</tr>
<tr>
<td>MvVelocityExecute</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Command: MoveVelocity (velocity control) $\rightarrow$ CmdActive = 2</td>
</tr>
<tr>
<td>Velocity</td>
<td>INPUT</td>
<td>REAL</td>
<td>Parameter: Velocity setting for MoveVelocity in user units. See example below the table</td>
</tr>
<tr>
<td>AccelerationTime</td>
<td>INPUT</td>
<td>REAL</td>
<td>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</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This parameter is used for the command MoveVelocity (MvVelocityExecute).</td>
</tr>
<tr>
<td>DecelerationTime</td>
<td>INPUT</td>
<td>REAL</td>
<td>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</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This parameter is used for the commands Stop (StopExecute) MoveVelocity (MvVelocityExecute).</td>
</tr>
<tr>
<td>JogPositive</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Command: JogPos</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Edge 0-1: Start axis in positive direction (jogging positive)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Edge 1-0: Stop axis</td>
</tr>
<tr>
<td>JogNegative</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Command: JogNeg</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Edge 0-1: Start axis in negative direction (jogging negative)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Edge 1-0: Stop axis</td>
</tr>
<tr>
<td>JogVelocity</td>
<td>INPUT</td>
<td>REAL</td>
<td>Parameter: Velocity setting for jogging in user units.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Note: JogPositive and JogNegative use the absolute value of the velocity.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Declaration</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------------</td>
<td>-----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>JogAcceleration-Time</td>
<td>INPUT</td>
<td>REAL</td>
<td>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</td>
</tr>
<tr>
<td>JogDeceleration-Time</td>
<td>INPUT</td>
<td>REAL</td>
<td>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</td>
</tr>
</tbody>
</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

\[ v_{\text{max}} = \frac{2 \cdot f}{p} \]

- \( v_{\text{max}} \) 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
- 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.

<table>
<thead>
<tr>
<th>No.</th>
<th>Designation</th>
<th>Range of values</th>
<th>Setting for Simple Motion Control Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>B1-01</td>
<td>Input source frequency setpoint 1</td>
<td>0, 1, 2, 3, 4</td>
<td>3: Option card</td>
</tr>
<tr>
<td>B1-02</td>
<td>Input source start command 1</td>
<td>0, 1, 2, 3</td>
<td>3: Option card</td>
</tr>
<tr>
<td>O1-03</td>
<td>Display scaling</td>
<td>0, 1, 2, 3, 4</td>
<td>2: min-1 unit</td>
</tr>
</tbody>
</table>

*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!
Usage inverter drive via EtherCAT

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.

   ![Graphical object view of CPU](image)

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.
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

<table>
<thead>
<tr>
<th>Name</th>
<th>Index</th>
<th>Bit length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status word</td>
<td>0x6041:00</td>
<td>16bit</td>
</tr>
<tr>
<td>Drive status value</td>
<td>0x2100:01</td>
<td>16bit</td>
</tr>
<tr>
<td>Output frequency value</td>
<td>0x2110:01</td>
<td>16bit</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Index</th>
<th>Bit length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control word</td>
<td>0x6040:00</td>
<td>16bit</td>
</tr>
<tr>
<td>vl target velocity</td>
<td>0x6042:00</td>
<td>16bit</td>
</tr>
<tr>
<td>vl velocity acceleration: Delta speed</td>
<td>0x6048:01</td>
<td>32bit</td>
</tr>
<tr>
<td>vl velocity acceleration: Delta time</td>
<td>0x6048:02</td>
<td>16bit</td>
</tr>
</tbody>
</table>

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’.

```
Device Editor

<table>
<thead>
<tr>
<th>PDO assign</th>
<th>...</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>☑ Inputs</td>
<td>☑ Outputs</td>
</tr>
<tr>
<td>☐ Inputs</td>
<td>☐ Outputs</td>
</tr>
</tbody>
</table>
```

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

```
Device Editor

Distributed Clock ... 

Distributed Clock

Operating Mode DC unused
```

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’

```
Device Editor

... Process image ...

I/O addresses ...

<table>
<thead>
<tr>
<th>Nr.</th>
<th>S7 Input address</th>
<th>S7 Output address</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>300 - 305</td>
<td>300 - 310</td>
</tr>
</tbody>
</table>
```

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_ConfigInverterEC_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_InitInverter_EC
- The Init 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_KernelInverter_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.

<table>
<thead>
<tr>
<th>Addr...</th>
<th>Name</th>
<th>Data type...</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>Config</td>
<td>UDT</td>
</tr>
<tr>
<td>...</td>
<td>Axis</td>
<td>UDT</td>
</tr>
</tbody>
</table>

**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.

  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
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.

\[ \text{CALL "VMC\_AxisControl", "DI\_AxisControl01"} \]
\[ \text{AxisEnable} := "\text{AxCtrl1\_AxisEnable}" \]
\[ \text{AxisReset} := "\text{AxCtrl1\_AxisReset}" \]
\[ \text{HomeExecute}^{*} := "\text{AxCtrl1\_HomeExecute}" \]
\[ \text{HomePosition}^{*} := "\text{AxCtrl1\_HomePosition}" \]
\[ \text{StopExecute} := "\text{AxCtrl1\_StopExecute}" \]
\[ \text{MvVelocityExecute} := "\text{AxCtrl1\_MvVelExecute}" \]
\[ \text{MvRelativeExecute}^{*} := "\text{AxCtrl1\_MvRelExecute}" \]
\[ \text{MvAbsoluteExecute}^{*} := "\text{AxCtrl1\_MvAbsExecute}" \]
\[ \text{PositionDistance}^{*} := "\text{AxCtrl1\_PositionDistance}" \]
\[ \text{Velocity} := "\text{AxCtrl1\_Velocity}" \]
\[ \text{Acceleration} := "\text{AxCtrl1\_Acceleration}" \]
\[ \text{Deceleration} := "\text{AxCtrl1\_Deceleration}" \]
\[ \text{JogPositive} := "\text{AxCtrl1\_JogPositive}" \]
\[ \text{JogNegative} := "\text{AxCtrl1\_JogNegative}" \]
\[ \text{JogVelocity} := "\text{AxCtrl1\_JogVelocity}" \]
\[ \text{JogAcceleration} := "\text{AxCtrl1\_JogAcceleration}" \]
\[ \text{JogDeceleration} := "\text{AxCtrl1\_JogDeceleration}" \]
\[ \text{AxisReady} := "\text{AxCtrl1\_AxisReady}" \]
\[ \text{AxisEnabled} := "\text{AxCtrl1\_AxisEnabled}" \]
\[ \text{AxisError} := "\text{AxCtrl1\_AxisError}" \]
\[ \text{AxisErrorID} := "\text{AxCtrl1\_AxisErrorID}" \]
\[ \text{DriveWarning} := "\text{AxCtrl1\_DriveWarning}" \]
\[ \text{DriveError} := "\text{AxCtrl1\_DriveError}" \]
\[ \text{DriveErrorID} := "\text{AxCtrl1\_DriveErrorID}" \]
\[ \text{IsHomed}^{*} := "\text{AxCtrl1\_IsHomed}" \]
\[ \text{ModeOfOperation} := "\text{AxCtrl1\_ModeOfOperation}" \]
\[ \text{PLCopenState} := "\text{AxCtrl1\_PLCopenState}" \]
\[ \text{ActualPosition}^{*} := "\text{AxCtrl1\_ActualPosition}" \]
\[ \text{ActualVelocity} := "\text{AxCtrl1\_ActualVelocity}" \]
\[ \text{CmdDone} := "\text{AxCtrl1\_CmdDone}" \]
\[ \text{CmdBusy} := "\text{AxCtrl1\_CmdBusy}" \]
\[ \text{CmdAborted} := "\text{AxCtrl1\_CmdAborted}" \]
\[ \text{CmdError} := "\text{AxCtrl1\_CmdError}" \]
\[ \text{CmdErrorID} := "\text{AxCtrl1\_CmdErrorID}" \]
\[ \text{DirectionPositive} := "\text{AxCtrl1\_DirectionPos}" \]
\[ \text{DirectionNegative} := "\text{AxCtrl1\_DirectionNeg}" \]
\[ \text{SWLimitMinActive}^{*} := "\text{AxCtrl1\_SWLimitMinActive}" \]
\[ \text{SWLimitMaxActive}^{*} := "\text{AxCtrl1\_SWLimitMaxActive}" \]
\[ \text{HWLimitMinActive}^{*} := "\text{AxCtrl1\_HWLimitMinActive}" \]
\[ \text{HWLimitMaxActive}^{*} := "\text{AxCtrl1\_HWLimitMaxActive}" \]
\[ \text{Axis} := "\text{Axis01}.Axis" \]

*) 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
Usage in VIPA SPEED7 Studio > User program

- OB 86 - RackFLT
- 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:
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

<table>
<thead>
<tr>
<th>Slot</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CPU 315-2 PN/DP</td>
</tr>
<tr>
<td>X1</td>
<td>MPI/DP</td>
</tr>
<tr>
<td>X2</td>
<td>PN-IO</td>
</tr>
<tr>
<td></td>
<td>Port 1</td>
</tr>
<tr>
<td>X2</td>
<td>Port 2</td>
</tr>
</tbody>
</table>

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’.

<table>
<thead>
<tr>
<th>Slot</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CPU 315-2 PN/DP</td>
</tr>
<tr>
<td>X1</td>
<td>x1_MPI/DP</td>
</tr>
<tr>
<td>X2</td>
<td>PN-IO</td>
</tr>
<tr>
<td></td>
<td>Port 1</td>
</tr>
</tbody>
</table>

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.
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.

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!

---

Configuration of Ethernet PG/OP channel

<table>
<thead>
<tr>
<th>Slot</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CPU ... PN-I0</td>
</tr>
<tr>
<td>2</td>
<td>CPU ... PN-I0</td>
</tr>
<tr>
<td>3</td>
<td>434-1EX30</td>
</tr>
</tbody>
</table>

Insert ‘EtherCAT network’

<table>
<thead>
<tr>
<th>Slot</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CPU ... PN-I0</td>
</tr>
<tr>
<td>2</td>
<td>CPU ... PN-I0</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Slot</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CPU</td>
</tr>
<tr>
<td></td>
<td>X...PN-IO</td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

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'.

<table>
<thead>
<tr>
<th>Slot</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CPU...</td>
</tr>
<tr>
<td>X...</td>
<td>PN-IO</td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

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.

2. 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 www.yaskawa.eu.com at ‘Service ➔ Drives & Motion Software’. Download the according ESI file for your drive. Unzip this if necessary.

3. Open in the SPEED7 EtherCAT Manager via ‘File ➔ ESI Manager’ the dialog window ‘ESI Manager’.

4. 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.

5. 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.
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**

<table>
<thead>
<tr>
<th>Name</th>
<th>Index</th>
<th>Bit length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status word</td>
<td>0x6041:00</td>
<td>16bit</td>
</tr>
<tr>
<td>Drive status value</td>
<td>0x2100:01</td>
<td>16bit</td>
</tr>
<tr>
<td>Output frequency value</td>
<td>0x2110:01</td>
<td>16bit</td>
</tr>
</tbody>
</table>

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**

<table>
<thead>
<tr>
<th>Name</th>
<th>Index</th>
<th>Bit length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control word</td>
<td>0x6040:00</td>
<td>16bit</td>
</tr>
<tr>
<td>vl target velocity</td>
<td>0x6042:00</td>
<td>16bit</td>
</tr>
<tr>
<td>vl velocity acceleration: Delta speed</td>
<td>0x6048:01</td>
<td>32bit</td>
</tr>
<tr>
<td>vl velocity acceleration: Delta time</td>
<td>0x6048:02</td>
<td>16bit</td>
</tr>
</tbody>
</table>

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

- **Init**
  
  FB 887
  
  VMC_InitInverter_EC

- **FBs**
  
  - FB 860
    
    VMC_AxisControl
  
  - FB 800 ... 838

- **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_ConfigInverterEC_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_InitInverter_EC**

  - The *Init* 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_KernelInverter_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**

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.

<table>
<thead>
<tr>
<th>Address</th>
<th>Name</th>
<th>Typ</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Struct</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>Config</td>
<td>&quot;VMC_ConfigInverterEC_REF&quot;</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>Axis</td>
<td>&quot;MC_AXIS_REF&quot;</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>END_STRUCT</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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

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.

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
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 hard-ware 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

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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Init</td>
<td>INPUT</td>
<td>BOOL</td>
<td>The block is internally reset with an edge 0-1. Existing motion commands are aborted and the block is initialized.</td>
</tr>
<tr>
<td>Config</td>
<td>IN_OUT</td>
<td>UDT 886</td>
<td>Data structure for transferring axis-dependent configuration data to the AxisKernel.</td>
</tr>
<tr>
<td>Axis</td>
<td>IN_OUT</td>
<td>UDT 860</td>
<td>Data structure for transferring axis-dependent information to the AxisKernel and PLCopen blocks.</td>
</tr>
</tbody>
</table>

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.

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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Release of initialization</td>
</tr>
<tr>
<td>LogicalAddress</td>
<td>INPUT</td>
<td>INT</td>
<td>Start address of the PDO input data</td>
</tr>
<tr>
<td>InputsStartAddressPDO</td>
<td>INPUT</td>
<td>INT</td>
<td>Start address of the input PDOs</td>
</tr>
<tr>
<td>OutputsStartAddressPDO</td>
<td>INPUT</td>
<td>INT</td>
<td>Start address of the output PDOs</td>
</tr>
</tbody>
</table>
### Parameter Declaration

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxVelocityDrive</td>
<td>INPUT</td>
<td>REAL</td>
<td>Maximum application speed [μ].</td>
</tr>
<tr>
<td>MaxOutputFrequency</td>
<td>INPUT</td>
<td>REAL</td>
<td>Maximum output frequency [Hz]. Please transfer the value from the software tool Drive Wizard+ here.</td>
</tr>
<tr>
<td>NumberOfPoles</td>
<td>INPUT</td>
<td>INT</td>
<td>Number of poles. Please transfer the value from the software tool Drive Wizard+ here.</td>
</tr>
<tr>
<td>Valid</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Initialization</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- TRUE: Initialization is valid.</td>
</tr>
<tr>
<td>Error</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- TRUE: An error has occurred. Additional error information can be found in the parameter ErrorID. The axis is disabled.</td>
</tr>
<tr>
<td>ErrorID</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>Additional error information</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Chap. 12 ‘ErrorID - Additional error information’ page 457</td>
</tr>
<tr>
<td>MaxVelocity</td>
<td>OUTPUT</td>
<td>INT</td>
<td>Maximum velocity in [rpm]. This value is determined automatically.</td>
</tr>
<tr>
<td>Config</td>
<td>IN_OUT</td>
<td>UDT 886</td>
<td>Data structure for transferring axis-dependent configuration data to the AxisKernel.</td>
</tr>
<tr>
<td>Axis</td>
<td>IN_OUT</td>
<td>UDT 860</td>
<td>Data structure for transferring axis-dependent information to the AxisKernel and PLCopen blocks.</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Supported blocks</th>
<th>Sigma-5/7 PROFINET</th>
<th>Sigma-5/7 EtherCAT</th>
<th>Inverter EtherCAT</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDT 860 - MC_AXIS_REF - data structure for axis</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>363</td>
</tr>
<tr>
<td>FB 860 - VMC_AxisControl - control of drive functions and query of drive states</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>363</td>
</tr>
</tbody>
</table>

## Complex motion tasks - PLCopen blocks

<table>
<thead>
<tr>
<th>Supported blocks</th>
<th>Sigma-5/7 PROFINET</th>
<th>Sigma-5/7 EtherCAT</th>
<th>Inverter EtherCAT</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDT 860 - MC_AXIS_REF - data structure for axis</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>367</td>
</tr>
<tr>
<td>UDT 861 - MC_TRIGGER_REF - data structure</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>367</td>
</tr>
<tr>
<td>FB 800 - MC_Power - enable respectively disable axis</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>368</td>
</tr>
<tr>
<td>FB 801 - MC_Home - home axis</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>370</td>
</tr>
<tr>
<td>FB 802 - MC_Stop - stop axis</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>372</td>
</tr>
<tr>
<td>FB 803 - MC_Halt - stop axis</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>374</td>
</tr>
<tr>
<td>FB 804 - MC_MoveRelative - move axis relative</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>376</td>
</tr>
<tr>
<td>FB 805 - MC_MoveVelocity - drive axis with constant velocity</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>378</td>
</tr>
<tr>
<td>FB 808 - MoveAbsolute - move axis to absolute position</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>380</td>
</tr>
<tr>
<td>FB 811 - MC_Reset - reset axis</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>382</td>
</tr>
<tr>
<td>FB 812 - MC_ReadStatus - read PLCopen-State of the axis</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>384</td>
</tr>
<tr>
<td>FB 813 - MC_ReadAxisError - read axis error</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>386</td>
</tr>
<tr>
<td>FB 814 - MC_ReadParameter - read parameter data from axis</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>388</td>
</tr>
<tr>
<td>FB 815 - MC_WriteParameter - write parameter data to axis</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>390</td>
</tr>
<tr>
<td>FB 816 - MC_ReadActualPosition - read the current position of the axis</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>392</td>
</tr>
<tr>
<td>FB 817 - MC_ReadActualVelocity - read the current velocity of the axis</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>393</td>
</tr>
<tr>
<td>FB 818 - MC_ReadAxisInfo - read axis additional information</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>394</td>
</tr>
</tbody>
</table>
## Blocks for axis control

<table>
<thead>
<tr>
<th>Supported blocks</th>
<th>Sigma-5/7</th>
<th>Sigma-5/7</th>
<th>Inverter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB 819 - MC_ReadMotionState - read state motion job</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>396</td>
</tr>
<tr>
<td>FB 823 - MC_TouchProbe - touch probe</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>398</td>
</tr>
<tr>
<td>FB 824 - MC_AbortTrigger - abort touch probe</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>400</td>
</tr>
<tr>
<td>FB 825 - MC_ReadBoolParameter - read boolean parameter from axis</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>401</td>
</tr>
<tr>
<td>FB 826 - MC_WriteBoolParameter - write boolean parameter to axis</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>403</td>
</tr>
<tr>
<td>FB 827 - VMC_ReadDWordParameter - read double-word parameter from axis</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>405</td>
</tr>
<tr>
<td>FB 828 - VMC_WriteDWordParameter - write double-word parameter to axis</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>407</td>
</tr>
<tr>
<td>FB 829 - VMC_ReadDWordParameter - read word parameter from axis</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>409</td>
</tr>
<tr>
<td>FB 830 - VMC_WriteDWordParameter - write word parameter to axis</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>411</td>
</tr>
<tr>
<td>FB 831 - VMC_ReadByteParameter - read byte parameter from axis</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>413</td>
</tr>
<tr>
<td>FB 832 - MC_WriteParameter - write byte parameter to axis</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>415</td>
</tr>
<tr>
<td>FB 833 - VMC_ReadDriveParameter - read drive parameter from drive</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>417</td>
</tr>
<tr>
<td>FB 834 - VMC_WriteParameter - write drive parameter to drive</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>419</td>
</tr>
<tr>
<td>FB 835 - VMC_HomeInit_LimitSwitch - initialization of homing on limit switch</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>421</td>
</tr>
<tr>
<td>FB 836 - VMC_HomeInit_HomeSwitch - initialization of homing on home switch</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>423</td>
</tr>
<tr>
<td>FB 837 - VMC_HomeInit_ZeroPulse - initialization of homing on zero pulse</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>426</td>
</tr>
<tr>
<td>FB 838 - VMC_HomeInit_SetPosition - initialization of homing mode set position</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>428</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AxisEnable</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Enable/disable axis&lt;br&gt;– TRUE: The axis is enabled.&lt;br&gt;– FALSE: The axis is disabled.</td>
</tr>
<tr>
<td>AxisReset</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Reset axis&lt;br&gt;– Edge 0-1: Axis reset is performed.</td>
</tr>
<tr>
<td>HomeExecute</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Homing&lt;br&gt;– Edge 0-1: Homing is started.</td>
</tr>
<tr>
<td>HomePosition</td>
<td>INPUT</td>
<td>REAL</td>
<td>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.</td>
</tr>
<tr>
<td>StopExecute</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Stop axis&lt;br&gt;– Edge 0-1: Stopping of the axis is started.</td>
</tr>
<tr>
<td>MvVelocityExecute</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Start moving the axis&lt;br&gt;– Edge 0-1: The axis is accelerated / decelerated to the speed specified.</td>
</tr>
<tr>
<td>MvRelativeExecute</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Start moving the axis&lt;br&gt;– Edge 0-1: The relative positioning of the axis is started.</td>
</tr>
<tr>
<td>MvAbsoluteExecute</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Start moving the axis&lt;br&gt;– Edge 0-1: The absolute positioning of the axis is started.</td>
</tr>
<tr>
<td>Direction *</td>
<td>INPUT</td>
<td>BYTE</td>
<td>Mode for absolute positioning:&lt;br&gt;0: shortest distance&lt;br&gt;1: positive direction&lt;br&gt;2: negative direction&lt;br&gt;3: current direction</td>
</tr>
<tr>
<td>PositionDistance</td>
<td>INPUT</td>
<td>REAL</td>
<td>Absolute position or relative distance depending on the command in [user units].</td>
</tr>
</tbody>
</table>
### Blocks for axis control

**VIPASPEED7 Library**

Simple motion tasks > FB 860 - VMC_AxisControl - Control block axis control

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Velocity</td>
<td>INPUT</td>
<td>REAL</td>
<td>Velocity setting (signed value) in [user units / s].</td>
</tr>
<tr>
<td>Acceleration</td>
<td>INPUT</td>
<td>REAL</td>
<td>Acceleration in [user units / s²].</td>
</tr>
<tr>
<td>Deceleration</td>
<td>INPUT</td>
<td>REAL</td>
<td>Deceleration in [user units / s²].</td>
</tr>
<tr>
<td>JogPositive</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Drive axis with constant velocity in positive direction</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– Edge 0-1: Drive axis with constant velocity is started.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– Edge 1-0: The axis is stopped.</td>
</tr>
<tr>
<td>JogNegative</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Drive axis with constant velocity in negative direction</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– Edge 0-1: Drive axis with constant velocity is started.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– Edge 1-0: The axis is stopped.</td>
</tr>
<tr>
<td>JogVelocity</td>
<td>INPUT</td>
<td>REAL</td>
<td>Speed setting for jogging (positive value) in [user units / s].</td>
</tr>
<tr>
<td>JogAcceleration</td>
<td>INPUT</td>
<td>REAL</td>
<td>Acceleration in [user units / s²].</td>
</tr>
<tr>
<td>JogDeceleration</td>
<td>INPUT</td>
<td>REAL</td>
<td>Delay for jogging in [user units / s²].</td>
</tr>
<tr>
<td>AxisReady</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>AxisReady</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: The axis is ready to switch on.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– FALSE: The axis is not ready to switch on.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>→ Check and fix AxisError (see AxisErrorID).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>→ Check and fix DriveError (see DriveErrorID).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>→ Check initialization FB (input and output addresses or PDO mapping correct?)</td>
</tr>
<tr>
<td>AxisEnabled</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status axis</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: Axis is switched on and accepts motion commands.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– FALSE: Axis is not switched on and does not accepts motion commands.</td>
</tr>
<tr>
<td>AxisError</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Motion axis error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: An error has occurred.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Additional error information can be found in the parameter AxisErrorID.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>→ The axis is disabled.</td>
</tr>
<tr>
<td>AxisErrorID</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>Additional error information</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Chap. 12 ‘ErrorID - Additional error information’ page 457</td>
</tr>
<tr>
<td>DriveWarning</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Warning</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: There is a warning on the drive.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Additional information can be found in the manufacturer’s manual.</td>
</tr>
<tr>
<td>DriveError</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Error on the drive</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: An error has occurred.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Additional error information can be found in the parameter DriveErrorID.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>→ The axis is disabled.</td>
</tr>
<tr>
<td>DriveErrorID</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>Error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: There is an error on the drive.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Additional information can be found in the manufacturer’s manual.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Declaration</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>IsHomed</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Information axis: homed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: The axis is homed.</td>
</tr>
<tr>
<td>ModeOfOperation</td>
<td>OUTPUT</td>
<td>INT</td>
<td>Drive-specific mode. For further information see drive manual.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Example Sigma-5:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0: No mode changed/no mode assigned</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: Profile Position mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2: Reserved (keep last mode)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3: Profile Velocity mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4: Torque Profile mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6: Homing mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>7: Interpolated Position mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>8: Cyclic Sync Position mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>9: Cyclic Sync Velocity mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>10: Cyclic Sync Torque mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Other Reserved (keep last mode)</td>
</tr>
<tr>
<td>PLCopenState</td>
<td>OUTPUT</td>
<td>INT</td>
<td>Current PLCopenState:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: Disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2: Standstill</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3: Homing</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4: Discrete Motion</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5: Continuous Motion</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>7: Stopping</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>8: Errorstop</td>
</tr>
<tr>
<td>ActualPosition</td>
<td>OUTPUT</td>
<td>REAL</td>
<td>Position of the axis in [user unit].</td>
</tr>
<tr>
<td>ActualVelocity</td>
<td>OUTPUT</td>
<td>REAL</td>
<td>Velocity of the axis in [user unit / s]</td>
</tr>
<tr>
<td>CmdDone</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: Job ended without error.</td>
</tr>
<tr>
<td>CmdBusy</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: Job is running.</td>
</tr>
<tr>
<td>CmdAborted</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: The job was aborted during processing by another job.</td>
</tr>
<tr>
<td>CmdError</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: An error has occurred.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Additional error information can be found in the parameter CmdErrorID.</td>
</tr>
<tr>
<td>CmdErrorID</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>Additional error information</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>³ Chap. 12 ‘ErrorID - Additional error information’ page 457</td>
</tr>
<tr>
<td>DirectionPositive</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status motion job: Position increasing</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: The position of the axis is increasing</td>
</tr>
</tbody>
</table>
### Parameter Declaration Data type Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DirectionNegative</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status motion job: Position decreasing</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- TRUE: The position of the axis is decreasing</td>
</tr>
<tr>
<td>SWLimitMinActive</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Software limit switch</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- TRUE: Software Limit switch Minimum active (Minimum</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>position in negative direction exceeded).</td>
</tr>
<tr>
<td>SWLimitMaxActive</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Software limit switch</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- TRUE: Software limit switch Maximum active (Maximum</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>position in positive direction exceeded).</td>
</tr>
<tr>
<td>HWLimitMinActive</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Hardware limit switch</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- TRUE: Negative hardware limit switch active on the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>drive (NOT- Negative Overtravel).</td>
</tr>
<tr>
<td>HWLimitMaxActive</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Hardware limit switch</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- TRUE: Positive hardware limit switch active on the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>drive (POT- Positive Overtravel).</td>
</tr>
<tr>
<td>Axis</td>
<td>IN_OUT</td>
<td>MC_AXIS_REF</td>
<td>Reference to the axis.</td>
</tr>
</tbody>
</table>

*) 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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Enable/disable axis</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- TRUE: The axis is enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- FALSE: The axis is disabled</td>
</tr>
<tr>
<td>EnablePositive</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Parameter is currently not supported; call with FALSE</td>
</tr>
<tr>
<td>EnableNegative</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Parameter is currently not supported; call with FALSE</td>
</tr>
<tr>
<td>Status</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status axis</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- TRUE: The axis is ready to execute motion control jobs</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- FALSE: The axis is not ready to execute motion control jobs</td>
</tr>
<tr>
<td>Valid</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Always FALSE</td>
</tr>
<tr>
<td>Error</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- TRUE: An error has occurred. Additional error information can be found</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>in the parameter ErrorID. The axis is disabled</td>
</tr>
<tr>
<td>ErrorID</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>Additional error information</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Chap. 12 ‘ErrorID - Additional error information’ page 457</td>
</tr>
<tr>
<td>Axis</td>
<td>IN_OUT</td>
<td>MC_AXIS_REF</td>
<td>Reference to the axis</td>
</tr>
</tbody>
</table>

**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.
The axis is enabled with Enable = TRUE. At the time (1) it is enabled. Then motion control jobs can be activated.

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.

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_Homelnit_... blocks.

Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Homing</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– Edge 0-1: Homing is started</td>
</tr>
<tr>
<td>Position</td>
<td>INPUT</td>
<td>REAL</td>
<td>With a successful homing the current position of the axis is uniquely set to Position.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Position is to be entered in the used application unit.</td>
</tr>
<tr>
<td>BufferMode</td>
<td>INPUT</td>
<td>BYTE</td>
<td>Parameter is currently not supported; call with B#16#0</td>
</tr>
<tr>
<td>Done</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: Job successfully done</td>
</tr>
<tr>
<td>Busy</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: Job is running</td>
</tr>
<tr>
<td>CommandAborted</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: The job was aborted during processing by another job.</td>
</tr>
<tr>
<td>Error</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: An error has occurred. Additional error information can be found in the parameter ErrorID.</td>
</tr>
<tr>
<td>ErrorID</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>Additional error information</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Chap. 12 ‘ErrorID - Additional error information’ page 457</td>
</tr>
<tr>
<td>Axis</td>
<td>IN_OUT</td>
<td>MC_AXIS_REF</td>
<td>Reference to the axis</td>
</tr>
</tbody>
</table>

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).
The homing is started with edge 0-1 at Execute and Busy becomes TRUE.

At the time (2) the homing is completed. Busy has the value FALSE and Done den value TRUE.

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

At the time (4) with an edge 0-1 at Execute the homing is started again and Busy becomes TRUE.

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute</td>
<td>INPUT</td>
<td>BOOL</td>
<td>- Stop axis</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– Edge 0-1: Stopping of the axis is started</td>
</tr>
<tr>
<td>Deceleration</td>
<td>INPUT</td>
<td>REAL</td>
<td>- Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: Job successfully done</td>
</tr>
<tr>
<td>Jerk</td>
<td>INPUT</td>
<td>REAL</td>
<td>- Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: The job was aborted during processing by another job.</td>
</tr>
<tr>
<td>Done</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>- Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: Job is running</td>
</tr>
<tr>
<td>Busy</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>- Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: Job is running</td>
</tr>
<tr>
<td>CommandAborted</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>- Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: The job was aborted during processing by another job.</td>
</tr>
<tr>
<td>Error</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>- Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: An error has occurred. Additional error information can be found in the parameter ErrorID.</td>
</tr>
<tr>
<td>ErrorID</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>Additional error information</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>&amp; Chap. 12 ‘ErrorID - Additional error information’ page 457</td>
</tr>
<tr>
<td>Axis</td>
<td>IN_OUT</td>
<td>MC_AXIS_REF</td>
<td>Reference to the axis</td>
</tr>
</tbody>
</table>

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).
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.

At time (2) stopping the axis is completed, the axis is stopped. Busy has the value FALSE and Done den value TRUE.

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Stop axis</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– Edge 0-1: Stopping of the axis is started</td>
</tr>
<tr>
<td>Deceleration</td>
<td>INPUT</td>
<td>REAL</td>
<td>Delay in breaking in [user units/s²]</td>
</tr>
<tr>
<td>Jerk</td>
<td>INPUT</td>
<td>REAL</td>
<td>Parameter is currently not supported; call with 0.0</td>
</tr>
<tr>
<td>BufferMode</td>
<td>INPUT</td>
<td>BYTE</td>
<td>Parameter is currently not supported; call with B#16#0</td>
</tr>
<tr>
<td>Done</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: Job successfully done</td>
</tr>
<tr>
<td>Busy</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: Job is running</td>
</tr>
<tr>
<td>Active</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: Block controls the axis</td>
</tr>
<tr>
<td>CommandAborted</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: The job was aborted during processing by another job</td>
</tr>
<tr>
<td>Error</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: An error has occurred. Additional error information can be found in the parameter ErrorID.</td>
</tr>
<tr>
<td>ErrorID</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>Additional error information</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Chap. 12 ‘ErrorID - Additional error information’ page 457</td>
</tr>
<tr>
<td>Axis</td>
<td>IN_OUT</td>
<td>MC_AXIS_REF</td>
<td>Reference to the axis</td>
</tr>
</tbody>
</table>

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).
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 $Deceleration$. 

At time (2) slowing down the axis is completed, the axis is stopped. $Busy$ has the value FALSE and $Done$ den value TRUE.

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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/s²] |
| Deceleration    | INPUT       | REAL      | Delay in breaking in [user units/s²] |
| 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Drive axis with constant velocity</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– Edge 0-1: Drive axis with constant velocity is started</td>
</tr>
<tr>
<td>ContinuousUpdate</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Parameter is currently not supported; call with FALSE</td>
</tr>
<tr>
<td>Velocity</td>
<td>INPUT</td>
<td>REAL</td>
<td>Velocity setting (signed value) in [user units/s]</td>
</tr>
<tr>
<td>Acceleration</td>
<td>INPUT</td>
<td>REAL</td>
<td>Acceleration in [user units/s^2]</td>
</tr>
<tr>
<td>Deceleration</td>
<td>INPUT</td>
<td>REAL</td>
<td>Delay in breaking in [user units/s^2]</td>
</tr>
<tr>
<td>Jerk</td>
<td>INPUT</td>
<td>REAL</td>
<td>Parameter is currently not supported; call with 0.0</td>
</tr>
<tr>
<td>BufferMode</td>
<td>INPUT</td>
<td>BYTE</td>
<td>Parameter is currently not supported; call with B#16#0</td>
</tr>
<tr>
<td>InVelocity</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Velocity setting</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: Velocity setting reached</td>
</tr>
<tr>
<td>Busy</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: Job is running</td>
</tr>
<tr>
<td>Active</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: Block controls the axis</td>
</tr>
<tr>
<td>CommandAborted</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: The job was aborted during processing by another job</td>
</tr>
<tr>
<td>Error</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: An error has occurred. Additional error information can be found in</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>the parameter ErrorID.</td>
</tr>
<tr>
<td>ErrorID</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>Additional error information</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Chap. 12 ‘ErrorID - Additional error information’ page 457</td>
</tr>
<tr>
<td>Axis</td>
<td>IN_OUT</td>
<td>MC_AXIS_REF</td>
<td>Reference to the axis</td>
</tr>
</tbody>
</table>

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 \textit{Execute}. \textit{Busy} is TRUE and \textit{InVelocity} FALSE as soon as the set velocity is not reached. If the set velocity is reached, \textit{Busy} becomes FALSE and \textit{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 \textit{Execute} is set to FALSE.
- A running job can be aborted by a move job (e.g. \textit{MC_MoveAbsolute}).

Status diagram of the block parameters

1. Moving the axis with set velocity is started with edge 0-1 at \textit{Execute} and \textit{Busy} becomes TRUE.
2. At time (2) the axis reaches the set velocity and \textit{InVelocity} has the value TRUE.
3. Resetting \textit{Execute} to FALSE at time (3) does not influence the axis. The axis is further moved with constant set velocity and \textit{InVelocity} is further TRUE.
4. At the time (4) the \textit{MC_Velocity} job is aborted by a \textit{MC_Halt} job. The axis is decelerated to stop and \textit{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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Move the axis&lt;br&gt;– Edge 0-1: The movement of the axis is started</td>
</tr>
<tr>
<td>ContinuousUpdate</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Parameter is currently not supported; call with FALSE</td>
</tr>
<tr>
<td>Position</td>
<td>INPUT</td>
<td>REAL</td>
<td>Absolute position in [user units]</td>
</tr>
<tr>
<td>Velocity</td>
<td>INPUT</td>
<td>REAL</td>
<td>Maximum velocity (needs not necessarily be reached) signed value in [user units/s]</td>
</tr>
<tr>
<td>Acceleration</td>
<td>INPUT</td>
<td>REAL</td>
<td>Acceleration in [user units/s²]</td>
</tr>
<tr>
<td>Deceleration</td>
<td>INPUT</td>
<td>REAL</td>
<td>Delay in breaking in [user units/s²]</td>
</tr>
<tr>
<td>Jerk</td>
<td>INPUT</td>
<td>REAL</td>
<td>Parameter is currently not supported; call with 0.0</td>
</tr>
<tr>
<td>Direction</td>
<td>INPUT</td>
<td>Byte</td>
<td>Direction&lt;br&gt;– 0: Shortest way&lt;br&gt;– 1: Positive direction&lt;br&gt;– 2: Negative direction&lt;br&gt;– 3: Current direction</td>
</tr>
<tr>
<td>BufferMode</td>
<td>INPUT</td>
<td>BYTE</td>
<td>Parameter is currently not supported; call with B#16#0</td>
</tr>
<tr>
<td>Done</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status&lt;br&gt;– TRUE: Job successfully done. Target position was reached.</td>
</tr>
<tr>
<td>Busy</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status&lt;br&gt;– TRUE: Job is running</td>
</tr>
<tr>
<td>Active</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status&lt;br&gt;– TRUE: Block controls the axis</td>
</tr>
<tr>
<td>CommandAborted</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status&lt;br&gt;– TRUE: The job was aborted during processing by another job</td>
</tr>
<tr>
<td>Error</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status&lt;br&gt;– TRUE: An error has occurred. Additional error information can be found in the parameter ErrorID.</td>
</tr>
<tr>
<td>ErrorID</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>Additional error information</td>
</tr>
<tr>
<td>Axis</td>
<td>IN_OUT</td>
<td>MC_AXIS_REF</td>
<td>Reference to the axis</td>
</tr>
</tbody>
</table>

---

**VIPA SPEED7 Library**

Complex motion tasks - PLCopen blocks > FB 808 - MC_MoveAbsolute - move axis to absolute position

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

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Reset axis</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– Edge 0-1: Axis reset is performed</td>
</tr>
<tr>
<td>Done</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: Job successfully done. Reset was performed</td>
</tr>
<tr>
<td>Busy</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: Job is running</td>
</tr>
<tr>
<td>Error</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: An error has occurred. Additional error information can be found in the parameter ErrorID.</td>
</tr>
<tr>
<td>ErrorID</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>Additional error information</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Chap. 12 ‘ErrorID - Additional error information’ page 457</td>
</tr>
<tr>
<td>Axis</td>
<td>IN_OUT</td>
<td>MC_AXIS_REF</td>
<td>Reference to the axis</td>
</tr>
</tbody>
</table>

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.
At time (1) the reset of the axis is started with edge 0-1 at Execute and Busy becomes TRUE.

At the time (2) the reset is successfully completed. Busy has the value FALSE and Done den value TRUE.

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Status indication</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: The status is permanently displayed at the outputs</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– FALSE: All the outputs are FALSE respectively 0</td>
</tr>
<tr>
<td>Valid</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>State is valid</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: The shown state is valid</td>
</tr>
<tr>
<td>Error</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: An error has occurred. Additional error information can be found in</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>the parameter ErrorID.</td>
</tr>
<tr>
<td>ErrorID</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>Additional error information</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>☞ Chap. 12 ‘ErrorID - Additional error information’ page 457</td>
</tr>
<tr>
<td>ErrorStop</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Axis errors</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: An axis error has occurred, move job can not be activated</td>
</tr>
<tr>
<td>Disabled</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status axis: Disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: Axis is disabled, move job can not be activated</td>
</tr>
<tr>
<td>Stopping</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status axis: Stop</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: Axis is stopped (MC_Stop is active)</td>
</tr>
<tr>
<td>Homing</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status axis: Homing</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: Axis is just homing (MC_Homing is active)</td>
</tr>
<tr>
<td>Standstill</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status move job</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: No move job is active; a move job can be activated</td>
</tr>
<tr>
<td>DiscreteMotion</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status axis motion: Discrete</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: Axis is moved by a discrete movement (MC_MoveRelative, MC_MoveAbsolute or MC_Halt is active)</td>
</tr>
<tr>
<td>ContinuousMotion</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status axis motion: Continuous</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: Axis is moved by a continuous movement (MC_MoveVelocity is active)</td>
</tr>
<tr>
<td>Axis</td>
<td>IN_OUT</td>
<td>MC_AXIS_REF</td>
<td>Reference to the slave axis</td>
</tr>
</tbody>
</table>
With $Enable = \text{TRUE}$ the outputs represent the state of the axis according to the PLCopen-State diagram.

### Determine the status of the axis

At time (1) $Enable$ is set to TRUE. So $Valid$ gets TRUE and the outputs correspond to the status of the PLCopen-State.

At time (2) $Enable$ is set to FALSE. So all the outputs are set to FALSE respectively 0.

### Status diagram of the block parameters

![Status diagram](image)

(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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Reset axis</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– Edge 0-1: Axis error is read.</td>
</tr>
<tr>
<td>Done</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: Job successfully done. Axis error read.</td>
</tr>
<tr>
<td>Busy</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: Job is running.</td>
</tr>
<tr>
<td>Error</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: An error has occurred. Additional error information can be found in the parameter ErrorID.</td>
</tr>
<tr>
<td>ErrorID</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>Additional error information</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Chap. 12 ‘ErrorID - Additional error information’ page 457</td>
</tr>
<tr>
<td>AxisErrorID</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>Axis error ID; the read value is vendor-specifically encoded.</td>
</tr>
<tr>
<td>Axis</td>
<td>IN_OUT</td>
<td>MC_AXIS_REF</td>
<td>Reference to the axis</td>
</tr>
</tbody>
</table>

**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.
(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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Read axis parameter data</td>
</tr>
<tr>
<td>Parameter Number</td>
<td>INPUT</td>
<td>INT</td>
<td>Number of the parameter to be read. Chap. 9.3.35 ‘PLCopen parameter’ page 429</td>
</tr>
<tr>
<td>Done</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status: TRUE: Job successfully done. Parameter data was read</td>
</tr>
<tr>
<td>Busy</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status: TRUE: Job is running</td>
</tr>
<tr>
<td>Error</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status: TRUE: An error has occurred. Additional error information can be found in the parameter ErrorID.</td>
</tr>
<tr>
<td>ErrorID</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>Additional error information Chap. 12 ‘ErrorID - Additional error information’ page 457</td>
</tr>
<tr>
<td>Value</td>
<td>OUTPUT</td>
<td>REAL</td>
<td>Value of the read parameter</td>
</tr>
<tr>
<td>Axis</td>
<td>IN_OUT</td>
<td>MC_AXIS_REF</td>
<td>Reference to the axis</td>
</tr>
</tbody>
</table>

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.
At time (1) the reading of the parameter data is started with edge 0-1 at $Execute$ and $Busy$ becomes TRUE.

At the time (2) reading of the parameter data is successfully completed. $Busy$ has the value FALSE and $Done$ den value TRUE.

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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Write axis parameter data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– Edge 0-1: The parameter data is written</td>
</tr>
<tr>
<td>Parameter Number</td>
<td>INPUT</td>
<td>INT</td>
<td>Number of the parameter to be written. Chap. 9.3.35 ‘PLCopen parameter’ page 429</td>
</tr>
<tr>
<td>Value</td>
<td>INPUT</td>
<td>REAL</td>
<td>Value of the written parameter</td>
</tr>
<tr>
<td>Done</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: Job successfully done. Parameter data was written</td>
</tr>
<tr>
<td>Busy</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: Job is running</td>
</tr>
<tr>
<td>Error</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: An error has occurred. Additional error information can be found in the parameter ErrorID.</td>
</tr>
<tr>
<td>ErrorID</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>Additional error information</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Chap. 12 ‘ErrorID - Additional error information’ page 457</td>
</tr>
<tr>
<td>Axis</td>
<td>IN_OUT</td>
<td>MC_AXIS_REF</td>
<td>Reference to the axis</td>
</tr>
</tbody>
</table>

**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.
(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

With MC_ReadActualPosition the current position of the axis is read.

Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Read axis position</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: The position of the axis is continuously read</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– FALSE: All the outputs are FALSE respectively 0</td>
</tr>
<tr>
<td>Valid</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Position valid</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: The read position is valid</td>
</tr>
<tr>
<td>Error</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: An error has occurred. Additional error information can be found in the parameter ErrorID.</td>
</tr>
<tr>
<td>ErrorID</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>Additional error information</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Chap. 12 ‘ErrorID - Additional error information’ page 457</td>
</tr>
<tr>
<td>Position</td>
<td>OUTPUT</td>
<td>REAL</td>
<td>Position of the axis [user unit]</td>
</tr>
<tr>
<td>Axis</td>
<td>IN_OUT</td>
<td>MC_AXIS_REF</td>
<td>Reference to the axis</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Read axis velocity</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: The velocity of the axis is continuously read</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– FALSE: All the outputs are FALSE respectively 0</td>
</tr>
<tr>
<td>Valid</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Velocity valid</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: The read velocity is valid</td>
</tr>
<tr>
<td>Error</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: An error has occurred. Additional error information can be found in the parameter ErrorID.</td>
</tr>
<tr>
<td>ErrorID</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>Additional error information</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Chap. 12 ‘ErrorID - Additional error information’ page 457</td>
</tr>
<tr>
<td>Velocity</td>
<td>OUTPUT</td>
<td>REAL</td>
<td>Velocity of the axis [user unit/s]</td>
</tr>
<tr>
<td>Axis</td>
<td>IN_OUT</td>
<td>MC_AXIS_REF</td>
<td>Reference to the axis</td>
</tr>
</tbody>
</table>

**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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Read additional information from axis</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: The additional information of the axis are read</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– FALSE: All the outputs are FALSE respectively 0</td>
</tr>
<tr>
<td>Valid</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Additional information valid</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: The read additional information are valid</td>
</tr>
<tr>
<td>Error</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: An error has occurred. Additional error information can be found in the parameter ErrorID.</td>
</tr>
<tr>
<td>ErrorID</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>Additional error information</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Chap. 12 ‘ErrorID - Additional error information’ page 457</td>
</tr>
<tr>
<td>HomeAbsSwitch</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Homing switch</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: Homing switch is activated</td>
</tr>
<tr>
<td>LimitSwitchPos</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Limit switch positive direction</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: Limit switch positive direction is activated</td>
</tr>
<tr>
<td>LimitSwitchNeg</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Limit switch negative direction (NOT bit of the drive)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: Limit switch negative direction is activated</td>
</tr>
<tr>
<td>Simulation</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Parameter is currently not supported; always FALSE</td>
</tr>
<tr>
<td>Communication-Ready</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Information axis: Data exchange</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: Data exchange with axis is initialized; axis is ready for communication</td>
</tr>
<tr>
<td>ReadyForPowerOn</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Information axis: Enable possible</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: Enabling the axis is possible</td>
</tr>
<tr>
<td>PowerOn</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Information axis: Enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: Enabling of the axis is carried out</td>
</tr>
<tr>
<td>IsHomed</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Information axis: Homed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: The axis is homed</td>
</tr>
<tr>
<td>AxisWarning</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Information axis: Error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: At least 1 error is reported from the axis</td>
</tr>
<tr>
<td>Axis</td>
<td>IN_OUT</td>
<td>MC_AXIS_REF</td>
<td>Reference to the axis</td>
</tr>
</tbody>
</table>

PLCopen-State

Job start in each PLCopen-State possible.
The additional information of the axis are shown at the outputs with *Enable* set to TRUE.

**Determine the status 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 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable</td>
<td>INPUT</td>
<td>BOOL</td>
<td>- Read motion state&lt;br&gt;TRUE: The status of the motion job is continuously read&lt;br&gt;FALSE: All the outputs are FALSE respectively 0</td>
</tr>
<tr>
<td>Source</td>
<td>INPUT</td>
<td>Byte</td>
<td>Only Source = 0 is supported; at the outputs the current status of the motion job is shown.</td>
</tr>
<tr>
<td>Valid</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>- Status valid&lt;br&gt;TRUE: The read status of the motion job is valid</td>
</tr>
<tr>
<td>Error</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>- Status&lt;br&gt;TRUE: An error has occurred. Additional error information can be found in the parameter ErrorID.</td>
</tr>
<tr>
<td>ErrorID</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>Additional error information&lt;br&gt;Chap. 12 ‘ErrorID - Additional error information’ page 457</td>
</tr>
<tr>
<td>ConstantVelocity</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>- Status motion job: Velocity&lt;br&gt;TRUE: Velocity is constant</td>
</tr>
<tr>
<td>Accelerating</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Please note that this parameter is not supported when using inverter drives via EtherCAT!&lt;br&gt;- Status motion job: Acceleration&lt;br&gt;TRUE: The axis is accelerated; the velocity of the axis is increasing</td>
</tr>
<tr>
<td>Decelerating</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Please note that this parameter is not supported when using inverter drives via EtherCAT!&lt;br&gt;- Status motion job: Braking process&lt;br&gt;TRUE: Axis is decelerated; the velocity of the axis is getting smaller</td>
</tr>
<tr>
<td>DirectionPositive</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>- Status motion job: Position increasing&lt;br&gt;TRUE: The position of the axis is increasing</td>
</tr>
<tr>
<td>DirectionNegative</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>- Status motion job: Position decreasing&lt;br&gt;TRUE: The position of the axis is decreasing</td>
</tr>
<tr>
<td>Axis</td>
<td>IN_OUT</td>
<td>MC_AXIS_REF</td>
<td>Reference to the axis</td>
</tr>
</tbody>
</table>

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.
At time (1) Enable is set to TRUE. So Valid gets TRUE and the outputs correspond to the status of motion job.

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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute</td>
<td>INPUT</td>
<td>BOOL</td>
<td>The recording of the axis position is activated with edge 0-1 at <code>Execute</code>.</td>
</tr>
<tr>
<td>Done</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- TRUE: Job successfully done. The axis position was recorded.</td>
</tr>
<tr>
<td>Busy</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- TRUE: Job is running.</td>
</tr>
<tr>
<td>CommandAborted</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- TRUE: The job was aborted during processing by another job.</td>
</tr>
<tr>
<td>Error</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- TRUE: An error has occurred. Additional error information can be found in the parameter <code>ErrorID</code>.</td>
</tr>
<tr>
<td>ErrorID</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>Additional error information</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Chap. 12 ‘ErrorID - Additional error information’ page 457</td>
</tr>
<tr>
<td>RecordedPosition</td>
<td>OUTPUT</td>
<td>REAL</td>
<td>Recorded axis position where trigger event occurred [user units].</td>
</tr>
<tr>
<td>Axis</td>
<td>IN_OUT</td>
<td>MC_AXIS_REF</td>
<td>Reference to the axis.</td>
</tr>
<tr>
<td>TriggerInput</td>
<td>IN_OUT</td>
<td>MC_TRIGGER_REF</td>
<td>Reference to the trigger input.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Structure</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- .Probe</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 01: TouchProbe register 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 02: TouchProbe register 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- .TriggerSource</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 00: Input</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 00: Encoder zero pulse</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- .Triggermode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 00: SingleTrigger (fix)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- .Reserved (0 fix)</td>
</tr>
</tbody>
</table>
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

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

Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute</td>
<td>INPUT</td>
<td>BOOL</td>
<td>The recording of the axis position is aborted with edge 0-1 at Execute.</td>
</tr>
<tr>
<td>Done</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>- Status: TRUE: Job successfully done. The recording of the axis position was aborted.</td>
</tr>
<tr>
<td>Busy</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>- Status: TRUE: Job is running.</td>
</tr>
<tr>
<td>Error</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>- Status: TRUE: An error has occurred. Additional error information can be found in the parameter ErrorID.</td>
</tr>
<tr>
<td>ErrorID</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>Additional error information</td>
</tr>
<tr>
<td>Axis</td>
<td>IN_OUT</td>
<td>MC_AXIS_REF</td>
<td>Reference to the axis.</td>
</tr>
<tr>
<td>TriggerInput</td>
<td>IN_OUT</td>
<td>MC_TRIGGER_REF</td>
<td>Reference to the trigger input. Structure</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- .Probe</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 01: TouchProbe register 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 02: TouchProbe register 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- .TriggerSource</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 00: Input</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 00: Encoder zero pulse</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- .Triggermode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- 00: SingleTrigger (fix)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- .Reserved (0 fix)</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Read axis parameter data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– Edge 0-1: The parameter data is read</td>
</tr>
<tr>
<td>Parameter Number</td>
<td>INPUT</td>
<td>INT</td>
<td>Number of the parameter to be read. Chap. 9.3.35 ‘PLCopen parameter’ page 429</td>
</tr>
<tr>
<td>Done</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: Job successfully done. Parameter data was read</td>
</tr>
<tr>
<td>Busy</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: Job is running</td>
</tr>
<tr>
<td>Error</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: An error has occurred. Additional error information can be found in the parameter ErrorID.</td>
</tr>
<tr>
<td>ErrorID</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>Additional error information</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Chap. 12 ‘ErrorID - Additional error information’ page 457</td>
</tr>
<tr>
<td>Value</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Value of the read parameter</td>
</tr>
<tr>
<td>Axis</td>
<td>IN_OUT</td>
<td>MC_AXIS_REF</td>
<td>Reference to the axis</td>
</tr>
</tbody>
</table>

**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.
(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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Write axis parameter data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Edge 0-1: The parameter data is written</td>
</tr>
<tr>
<td>Parameter Number</td>
<td>INPUT</td>
<td>INT</td>
<td>Number of the parameter to be written. Chap. 9.3.35 'PLCopen parameter' page 429</td>
</tr>
<tr>
<td>Value</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Value of the written parameter</td>
</tr>
<tr>
<td>Done</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- TRUE: Job successfully done. Parameter data was written</td>
</tr>
<tr>
<td>Busy</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- TRUE: Job is running</td>
</tr>
<tr>
<td>Error</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- TRUE: An error has occurred. Additional error information can be found in the parameter ErrorID.</td>
</tr>
<tr>
<td>ErrorID</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>Additional error information</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Chap. 12 'ErrorID - Additional error information' page 457</td>
</tr>
<tr>
<td>Axis</td>
<td>IN_OUT</td>
<td>MC_AXIS_REF</td>
<td>Reference to the axis</td>
</tr>
</tbody>
</table>

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.
(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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Read axis parameter data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– Edge 0-1: The parameter data is read</td>
</tr>
<tr>
<td>Parameter-Number</td>
<td>INPUT</td>
<td>INT</td>
<td>Number of the parameter to be read. Chap. 9.3.35 ‘PLCopen parameter’ page 429</td>
</tr>
<tr>
<td>Done</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: Job successfully done. Parameter data was read</td>
</tr>
<tr>
<td>Busy</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: Job is running</td>
</tr>
<tr>
<td>Error</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: An error has occurred. Additional error information can be found in the parameter ErrorID.</td>
</tr>
<tr>
<td>ErrorID</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>Additional error information</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Chap. 12 ‘ErrorID - Additional error information’ page 457</td>
</tr>
<tr>
<td>Value</td>
<td>OUTPUT</td>
<td>DWORD</td>
<td>Value of the read parameter</td>
</tr>
<tr>
<td>Axis</td>
<td>IN_OUT</td>
<td>MC_AXIS_REF</td>
<td>Reference to the axis</td>
</tr>
</tbody>
</table>

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.
At time (1) the reading of the parameter data is started with edge 0-1 at *Execute* and *Busy* becomes TRUE.

At the time (2) reading of the parameter data is successfully completed. *Busy* has the value FALSE and *Done* den value TRUE.

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Write axis parameter data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Edge 0-1: The parameter data is written</td>
</tr>
<tr>
<td>Parameter Number</td>
<td>INPUT</td>
<td>INT</td>
<td>Number of the parameter to be written. Chap. 9.3.35 ‘PLCopen parameter’ page 429</td>
</tr>
<tr>
<td>Value</td>
<td>INPUT</td>
<td>DWORD</td>
<td>Value of the written parameter</td>
</tr>
<tr>
<td>Done</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- TRUE: Job successfully done. Parameter data was written</td>
</tr>
<tr>
<td>Busy</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- TRUE: Job is running</td>
</tr>
<tr>
<td>Error</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- TRUE: An error has occurred. Additional error information can be found in the parameter ErrorID.</td>
</tr>
<tr>
<td>ErrorID</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>Additional error information</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Chap. 12 ‘ErrorID - Additional error information’ page 457</td>
</tr>
<tr>
<td>Axis</td>
<td>IN_OUT</td>
<td>MC_AXIS_REF</td>
<td>Reference to the axis</td>
</tr>
</tbody>
</table>

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.
At time (1) the writing of the parameter data is started with edge 0-1 at Execute and Busy becomes TRUE.

At the time (2) writing of the parameter data is successfully completed. Busy has the value FALSE and Done den value TRUE.

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Read axis parameter data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Edge 0-1: The parameter data is read</td>
</tr>
<tr>
<td>Parameter Number</td>
<td>INPUT</td>
<td>INT</td>
<td>Number of the parameter to be read.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Chap. 9.3.35 ‘PLCopen parameter’ page 429</td>
</tr>
<tr>
<td>Done</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- TRUE: Job successfully done. Parameter data was read</td>
</tr>
<tr>
<td>Busy</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- TRUE: Job is running</td>
</tr>
<tr>
<td>Error</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- TRUE: An error has occurred. Additional error information can be found in the parameter ErrorID.</td>
</tr>
<tr>
<td>ErrorID</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>Additional error information</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Chap. 12 ‘ErrorID - Additional error information’ page 457</td>
</tr>
<tr>
<td>Value</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>Value of the read parameter</td>
</tr>
<tr>
<td>Axis</td>
<td>IN_OUT</td>
<td>MC_AXIS_REF</td>
<td>Reference to the axis</td>
</tr>
</tbody>
</table>

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.
At time (1) the reading of the parameter data is started with edge 0-1 at Execute and Busy becomes TRUE.

At the time (2) reading of the parameter data is successfully completed. Busy has the value FALSE and Done den value TRUE.

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

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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.
At time (1) the writing of the parameter data is started with edge 0-1 at Execute and Busy becomes TRUE.

At the time (2) writing of the parameter data is successfully completed. Busy has the value FALSE and Done den value TRUE.

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**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Read axis parameter data</td>
</tr>
<tr>
<td>Parameter Number</td>
<td>INPUT</td>
<td>INT</td>
<td>Number of the parameter to be read. Chap. 9.3.35 ‘PLCopen parameter’ page 429</td>
</tr>
<tr>
<td>Done</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td>Busy</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td>Error</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td>ErrorID</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>Additional error information Chap. 12 ‘ErrorID - Additional error information’ page 457</td>
</tr>
<tr>
<td>Value</td>
<td>OUTPUT</td>
<td>BYTE</td>
<td>Value of the read parameter</td>
</tr>
<tr>
<td>Axis</td>
<td>IN_OUT</td>
<td>MC_AXIS_REF</td>
<td>Reference to the axis</td>
</tr>
</tbody>
</table>

**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.
At time (1) the reading of the parameter data is started with edge 0-1 at *Execute* and *Busy* becomes TRUE.

At the time (2) reading of the parameter data is successfully completed. *Busy* has the value FALSE and *Done* den value TRUE.

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Write axis parameter data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Edge 0-1: The parameter data is written</td>
</tr>
<tr>
<td>Parameter Number</td>
<td>INPUT</td>
<td>INT</td>
<td>Number of the parameter to be written. Chap. 9.3.35 ‘PLCopen parameter’ page 429</td>
</tr>
<tr>
<td>Value</td>
<td>INPUT</td>
<td>BYTE</td>
<td>Value of the written parameter</td>
</tr>
<tr>
<td>Done</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- TRUE: Job successfully done. Parameter data was written</td>
</tr>
<tr>
<td>Busy</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- TRUE: Job is running</td>
</tr>
<tr>
<td>Error</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- TRUE: An error has occurred. Additional error information can be found in the parameter ErrorID.</td>
</tr>
<tr>
<td>ErrorID</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>Additional error information</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Chap. 12 ‘ErrorID - Additional error information’ page 457</td>
</tr>
<tr>
<td>Axis</td>
<td>IN_OUT</td>
<td>MC_AXIS_REF</td>
<td>Reference to the axis</td>
</tr>
</tbody>
</table>

**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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Read drive parameter data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– Edge 0-1: The drive parameter data is reading.</td>
</tr>
<tr>
<td>Index</td>
<td>INPUT</td>
<td>WORD</td>
<td>Index of the drive parameter</td>
</tr>
<tr>
<td>Subindex</td>
<td>INPUT</td>
<td>BYTE</td>
<td>Subindex of the drive parameter</td>
</tr>
<tr>
<td>Length</td>
<td>INPUT</td>
<td>BYTE</td>
<td>Length of data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– 1: BYTE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– 2: WORD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– 4: DWORD</td>
</tr>
<tr>
<td>Done</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: Job successfully done. Parameter data was read</td>
</tr>
<tr>
<td>Busy</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: Job is running</td>
</tr>
<tr>
<td>Error</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: An error has occurred. Additional error information can be found in the parameter ErrorID.</td>
</tr>
<tr>
<td>ErrorID</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>Additional error information</td>
</tr>
<tr>
<td>Value</td>
<td>OUTPUT</td>
<td>DWORD</td>
<td>Value of the read parameter</td>
</tr>
<tr>
<td>Axis</td>
<td>IN_OUT</td>
<td>MC_AXIS_REF</td>
<td>Reference to the axis</td>
</tr>
</tbody>
</table>

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.
At time (1) the reading of the parameter data is started with edge 0-1 at *Execute* and *Busy* becomes TRUE.

At the time (2) reading of the parameter data is successfully completed. *Busy* has the value FALSE and *Done* den value TRUE.

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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Write drive parameter data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– Edge 0-1: The drive parameter data is written.</td>
</tr>
<tr>
<td>Index</td>
<td>INPUT</td>
<td>WORD</td>
<td>Index of the drive parameter</td>
</tr>
<tr>
<td>Subindex</td>
<td>INPUT</td>
<td>BYTE</td>
<td>Subindex of the drive parameter</td>
</tr>
<tr>
<td>Length</td>
<td>INPUT</td>
<td>BYTE</td>
<td>Length of data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– 1: BYTE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– 2: WORD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– 4: DWORD</td>
</tr>
<tr>
<td>Value</td>
<td>INPUT</td>
<td>DWORD</td>
<td>Value of the written parameter</td>
</tr>
<tr>
<td>Done</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: Job successfully done. Parameter data was read</td>
</tr>
<tr>
<td>Busy</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: Job is running</td>
</tr>
<tr>
<td>Error</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– TRUE: An error has occurred. Additional error information can be found in the parameter ErrorID.</td>
</tr>
<tr>
<td>ErrorID</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>Additional error information</td>
</tr>
<tr>
<td>Axis</td>
<td>IN_OUT</td>
<td>MC_AXIS_REF</td>
<td>Reference to the axis</td>
</tr>
</tbody>
</table>

**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.
(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

This block initialise 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Initialisation of the homing method</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Edge 0-1: Values of the input parameter are accepted and the initialisation of the homing method is started.</td>
</tr>
<tr>
<td>Direction</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Direction of homing</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- TRUE: on positive limit switch</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- FALSE: on negative limit switch</td>
</tr>
<tr>
<td>VelocitySearchSwitch</td>
<td>INPUT</td>
<td>REAL</td>
<td>Velocity for search for the switch in [user units/s]</td>
</tr>
<tr>
<td>VelocitySearchZero</td>
<td>INPUT</td>
<td>REAL</td>
<td>Velocity for search for zero in [user units/s]</td>
</tr>
<tr>
<td>Acceleration</td>
<td>INPUT</td>
<td>REAL</td>
<td>Acceleration in [user units/s²]</td>
</tr>
<tr>
<td>Done</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- TRUE: Initialisation successfully done.</td>
</tr>
<tr>
<td>Busy</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- TRUE: Initialisation is active.</td>
</tr>
<tr>
<td>Error</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- TRUE: An error has occurred. Additional error information can be found in the parameter ErrorID.</td>
</tr>
<tr>
<td>ErrorID</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>Additional error information</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Chap. 12 ‘ErrorID - Additional error information’ page 457</td>
</tr>
<tr>
<td>AXIS</td>
<td>IN_OUT</td>
<td>MC_AXIS_REF</td>
<td>Reference to the axis</td>
</tr>
</tbody>
</table>
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.

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/s²]

<table>
<thead>
<tr>
<th>Homing Method</th>
<th>Direction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>false</td>
</tr>
<tr>
<td>2</td>
<td>true</td>
</tr>
</tbody>
</table>

Initialisation of the homing method

Initialisation homing on limit switch

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/s²]
9.3.32 FB 836 - VMC_HomeInit_HomeSwitch - Initialisation of homing on home switch

Description

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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 |
| SameDirIndexPulse          | 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 |
| VelocitySearchSwitch       | INPUT       | REAL      | Velocity for search for the switch in [user units/s] |
| VelocitySearchZero         | INPUT       | REAL      | Velocity for search for zero in [user units/s] |
| Acceleration               | INPUT       | REAL      | Acceleration in [user units/s²] |
| 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. |
### 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 [UserUnits] > 0
   - VelocitySearchSwitch [UserUnits] ≤ VelocityMax
   - Input VelocitySearchZero [UserUnits] > 0.0
   - VelocitySearchZero [UserUnits] > 0
   - VelocitySearchZero [UserUnits] ≤ VelocityMax
   - Input Acceleration [UserUnits] > 0.0
   - Acceleration [UserUnits] > 0
   - Acceleration [UserUnits] ≤ 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/s²]

### Homing Method

<table>
<thead>
<tr>
<th>Homing Method</th>
<th>InitialDirection</th>
<th>WithIndexPulse</th>
<th>OnRisingEdge</th>
<th>SameDirIndexPulse</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>positive</td>
<td>true</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>8</td>
<td>positive</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>9</td>
<td>positive</td>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>10</td>
<td>positive</td>
<td>true</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>11</td>
<td>negative</td>
<td>true</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>12</td>
<td>negative</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>13</td>
<td>negative</td>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>14</td>
<td>negative</td>
<td>true</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>24</td>
<td>positive</td>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>24</td>
<td>positive</td>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>24</td>
<td>positive</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
</tbody>
</table>

**Parameter Declaration Data type Description**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ErrorID</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>Additional error information</td>
</tr>
<tr>
<td>AXIS</td>
<td>IN_OUT</td>
<td>MC_AXIS_REF</td>
<td>Reference to the axis</td>
</tr>
</tbody>
</table>

Chap. 12 ‘ErrorID - Additional error information’ page 457
<table>
<thead>
<tr>
<th>Homing Method</th>
<th>InitialDirection</th>
<th>WithIndexPulse</th>
<th>OnRisingEdge</th>
<th>SameDirIndexPulse</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>positive</td>
<td>false</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>28</td>
<td>negative</td>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>28</td>
<td>negative</td>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>28</td>
<td>negative</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>28</td>
<td>negative</td>
<td>false</td>
<td>false</td>
<td>true</td>
</tr>
</tbody>
</table>
### FB 837 - VMC_HomeInit_ZeroPulse - Initialisation of homing on zero pulses

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Initialisation of the homing method</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Edge 0-1: Values of the input parameter are accepted and the initialisation of the homing method is started.</td>
</tr>
<tr>
<td>Direction</td>
<td>INPUT</td>
<td>BOOL</td>
<td>Direction of homing</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TRUE: Positive direction</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>FALSE: Negative direction</td>
</tr>
<tr>
<td>VelocitySearch-Zero</td>
<td>INPUT</td>
<td>REAL</td>
<td>Velocity for search for zero in [user units/s]</td>
</tr>
<tr>
<td>Acceleration</td>
<td>INPUT</td>
<td>REAL</td>
<td>Acceleration in [user units/s²]</td>
</tr>
<tr>
<td>Done</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TRUE: Initialisation successfully done.</td>
</tr>
<tr>
<td>Busy</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TRUE: Initialisation is active.</td>
</tr>
<tr>
<td>Error</td>
<td>OUTPUT</td>
<td>BOOL</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TRUE: An error has occurred. Additional error information can be found in the parameter ErrorID.</td>
</tr>
<tr>
<td>ErrorID</td>
<td>OUTPUT</td>
<td>WORD</td>
<td>Additional error information</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>§ Chap. 12 'ErrorID - Additional error information' page 457</td>
</tr>
<tr>
<td>AXIS</td>
<td>IN_OUT</td>
<td>MC_AXIS_REF</td>
<td>Reference to the axis</td>
</tr>
</tbody>
</table>

### 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/s²]

<table>
<thead>
<tr>
<th>Homing Method</th>
<th>Direction</th>
</tr>
</thead>
<tbody>
<tr>
<td>33</td>
<td>false</td>
</tr>
<tr>
<td>34</td>
<td>true</td>
</tr>
</tbody>
</table>
9.3.34 FB 838 - VMC_HomeInit_SetPosition - Initialisation of homing mode set position

Description

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Declaration</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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 |

Initialization 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.

Initialization 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

<table>
<thead>
<tr>
<th>PN</th>
<th>Name</th>
<th>Data type</th>
<th>R/W</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>EnableLimitPos</td>
<td>BOOL</td>
<td>R/W</td>
<td>Enable positive software limit switch&lt;br&gt;Access on: &quot;Axis&quot;.AxisConfiguration.PositionLimits.EnableMaxPos</td>
</tr>
<tr>
<td>5</td>
<td>EnableLimitNeg</td>
<td>BOOL</td>
<td>R/W</td>
<td>Enable negative software limit switch&lt;br&gt;Access on: &quot;Axis&quot;.AxisConfiguration.PositionLimits.EnableMinPos</td>
</tr>
<tr>
<td>6</td>
<td>EnablePosLagMonitoring</td>
<td>BOOL</td>
<td>R/W</td>
<td>Enable monitoring of position lag&lt;br&gt;Function is not supported</td>
</tr>
<tr>
<td>7</td>
<td>MaxPositionLag</td>
<td>REAL</td>
<td>R/W</td>
<td>Maximal position lag&lt;br&gt;Function is not supported</td>
</tr>
<tr>
<td>8</td>
<td>MaxVelocitySystem</td>
<td>REAL</td>
<td>R</td>
<td>Maximal allowed velocity of the axis in the motion system&lt;br&gt;This parameter is currently not supported</td>
</tr>
<tr>
<td>9</td>
<td>MaxVelocityAppl</td>
<td>REAL</td>
<td>R/W</td>
<td>Maximal allowed velocity of the axis in the application&lt;br&gt;Access on: #Axis.AxisConfiguration.DynamicLimits.MaxVelocityAppl</td>
</tr>
<tr>
<td>12</td>
<td>MaxAccelerationSystem</td>
<td>REAL</td>
<td>R</td>
<td>Maximal allowed acceleration of the axis in the motion system&lt;br&gt;This parameter is currently not supported</td>
</tr>
<tr>
<td>No.</td>
<td>Name</td>
<td>Data type</td>
<td>Index</td>
<td>Subindex</td>
</tr>
<tr>
<td>-----</td>
<td>-----------------------------</td>
<td>------------</td>
<td>-------</td>
<td>----------</td>
</tr>
<tr>
<td>13</td>
<td>MaxAccelerationAppl</td>
<td>REAL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>MaxDecelerationSystem</td>
<td>REAL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>MaxDecelerationAppl</td>
<td>REAL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>MaxJerkSystem</td>
<td>REAL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>MaxJerkAppl</td>
<td>REAL</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Comments**

Maximal allowed acceleration of the axis in the application
Access on: 
#Axis.AxisConfiguration.DynamicLimits.MaxAccelerationAppl

Maximal allowed deceleration of the axis in the motion system
This parameter is currently not supported

Maximal allowed deceleration of the axis in the application
Access on: 
#Axis.AxisConfiguration.DynamicLimits.MaxDecelerationAppl

Maximum allowed jerk of the axis in the motion system
This parameter is currently not supported.

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**

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Data type</th>
<th>Index</th>
<th>Subindex</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>900</td>
<td>HomingDone</td>
<td>BOOL</td>
<td>-</td>
<td>-</td>
<td>R/W</td>
</tr>
<tr>
<td>901</td>
<td>PositiveTorqueLimit</td>
<td>BOOL</td>
<td>-</td>
<td>-</td>
<td>R/W</td>
</tr>
<tr>
<td>902</td>
<td>NegativeTorqueLimit</td>
<td>BOOL</td>
<td>-</td>
<td>-</td>
<td>R/W</td>
</tr>
<tr>
<td>1000</td>
<td>ErrorCode</td>
<td>WORD</td>
<td>603F</td>
<td>0</td>
<td>R</td>
</tr>
<tr>
<td>1001</td>
<td>HomeOffset</td>
<td>DWORD</td>
<td>607C</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>1002</td>
<td>HomingMethod</td>
<td>WORD</td>
<td>6098</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>1003</td>
<td>SpeedSearchSwitch</td>
<td>DWORD</td>
<td>6099</td>
<td>1</td>
<td>R/W</td>
</tr>
<tr>
<td>1004</td>
<td>SpeedSearchZero</td>
<td>DWORD</td>
<td>6099</td>
<td>2</td>
<td>R/W</td>
</tr>
<tr>
<td>1005</td>
<td>HomingAcceleration</td>
<td>DWORD</td>
<td>609A</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>1006</td>
<td>PositiveTorqueLimit</td>
<td>WORD</td>
<td>60E0</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>1007</td>
<td>NegativeTorqueLimit</td>
<td>WORD</td>
<td>0x60E1</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>1008</td>
<td>MotorRatedTorque</td>
<td>DWORD</td>
<td>0x6076</td>
<td>0</td>
<td>R/W</td>
</tr>
</tbody>
</table>

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
<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Data type</th>
<th>Index</th>
<th>Subindex</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>1009</td>
<td>FollowingErrorWindow</td>
<td>DWORD</td>
<td>0x6065</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>1010</td>
<td>FollowingErrorTimeOut</td>
<td>WORD</td>
<td>0x6066</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>1011</td>
<td>PositionWindow</td>
<td>DWORD</td>
<td>0x6067</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>1012</td>
<td>PositionTime</td>
<td>WORD</td>
<td>0x6068</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>1013</td>
<td>Min Position Limit</td>
<td>DWORD</td>
<td>0x607D</td>
<td>1</td>
<td>R/W</td>
</tr>
<tr>
<td>1014</td>
<td>Max Position Limit</td>
<td>DWORD</td>
<td>0x607D</td>
<td>2</td>
<td>R/W</td>
</tr>
<tr>
<td>1015</td>
<td>Digital outputs/ physical outputs</td>
<td>DWORD</td>
<td>0x60FE</td>
<td>1</td>
<td>R/W</td>
</tr>
<tr>
<td>1016</td>
<td>Digital outputs/ mask</td>
<td>DWORD</td>
<td>0x60FE</td>
<td>2</td>
<td>R/W</td>
</tr>
<tr>
<td>1017</td>
<td>Quick stop deceleration</td>
<td>DWORD</td>
<td>0x6085</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>1018</td>
<td>Forward external torque limit</td>
<td>WORD</td>
<td>0x2404</td>
<td>0</td>
<td>R/W</td>
</tr>
<tr>
<td>1019</td>
<td>Reverse external torque limit</td>
<td>WORD</td>
<td>0x2405</td>
<td>0</td>
<td>R/W</td>
</tr>
</tbody>
</table>

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_ReadDWordParameter - read axis double word parameter data’ page 409
4) Access via Chap. 9.3.26 ‘FB 830 - VMC_WriteDWordParameter - write axis double word parameter data’ page 407
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

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 user 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. See ‘Import voice table’ page 438.
Create a new 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].

![Diagram of screen size settings]

⇒ 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.

   ⇒ 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\Simple-Motion.

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.


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 ‘Resourcen > 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 Ù Properties’.

You can adjust the corresponding values at ‘Engineering Data’. Confirm with your settings.
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.
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.
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’.

Click at ‘S7 TCP’ and select ‘Context menu ➔ Comm. I/O Driver Settings’.

The ‘S7 TCP’ dialog opens.

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’ your 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.

Connect your CPU to your panel and turn on your application.

⇒ The panel starts with the screen to control your drive.

10.4.2.2 Operation

‘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/s²
  - 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. $\S$ 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

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**

![State Diagram]

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
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
- 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.  

```
Example
VIPA SPEED7 Library
States and behavior of the outputs
Replacement behavior of motion jobs
```

```
HB00 | OPL_SP7-LIB | SW90MS0MA V10.017 | en | 20-18
```
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 a 1-0 edge 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 concern 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.
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

<table>
<thead>
<tr>
<th>ErrorID</th>
<th>Description</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>No Error</td>
<td></td>
</tr>
<tr>
<td>0x8y24</td>
<td>Error in block parameter y, with y:</td>
<td>VMC_ConfigMaster_RTU</td>
</tr>
<tr>
<td></td>
<td>1: Error in PROTOKOLL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2: Error in PARAMETER</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3: Error in BAUDRATE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4: Error in CHARLENGTH</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5: Error in PARITY</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6: Error in STOPBITS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>7: Error in FLOWCONTROL (parameter missing)</td>
<td></td>
</tr>
<tr>
<td>0x8001</td>
<td>Invalid value at parameter Position.</td>
<td></td>
</tr>
<tr>
<td>0x8002</td>
<td>Invalid value at parameter Distance.</td>
<td></td>
</tr>
<tr>
<td>0x8003</td>
<td>Invalid value at parameter Velocity.</td>
<td></td>
</tr>
<tr>
<td>0x8004</td>
<td>Invalid value at parameter Acceleration.</td>
<td></td>
</tr>
<tr>
<td>0x8005</td>
<td>Invalid value at parameter Deceleration.</td>
<td></td>
</tr>
<tr>
<td>0x8007</td>
<td>Invalid value at parameter ContinuousUpdate</td>
<td></td>
</tr>
<tr>
<td>0x8008</td>
<td>Invalid value at parameter BufferMode.</td>
<td></td>
</tr>
<tr>
<td>0x8009</td>
<td>Invalid value at parameter EnablePositive</td>
<td></td>
</tr>
<tr>
<td>0x800A</td>
<td>Invalid value at parameter EnableNegative</td>
<td></td>
</tr>
<tr>
<td>0x800B</td>
<td>Invalid value at parameter MasterOffset.</td>
<td></td>
</tr>
<tr>
<td>0x800C</td>
<td>Invalid value at parameter SlaveOffset.</td>
<td></td>
</tr>
<tr>
<td>0x800D</td>
<td>Invalid value at parameter MasterScaling</td>
<td></td>
</tr>
<tr>
<td>0x800E</td>
<td>Invalid value at parameter SlaveScaling</td>
<td></td>
</tr>
<tr>
<td>0x800F</td>
<td>Invalid value at parameter StartMode.</td>
<td></td>
</tr>
<tr>
<td>0x8010</td>
<td>Invalid value at parameter ActivationMode</td>
<td></td>
</tr>
<tr>
<td>0x8011</td>
<td>Invalid value at parameter Source.</td>
<td></td>
</tr>
<tr>
<td>0x8012</td>
<td>Invalid value at parameter Direction.</td>
<td></td>
</tr>
<tr>
<td>0x8014</td>
<td>Invalid parameter of physical axis.</td>
<td>MC_ReadParameter</td>
</tr>
<tr>
<td>0x8015</td>
<td>Invalid index or subindex.</td>
<td>MC_ReadParameter</td>
</tr>
<tr>
<td>0x8016</td>
<td>Invalid parameter length.</td>
<td>MC_ReadParameter</td>
</tr>
<tr>
<td>0x8017</td>
<td>Invalid LADDR if e.g. the corresponding drive system is switched off or cannot be reached.</td>
<td>MC_ReadParameter</td>
</tr>
<tr>
<td>0x8018</td>
<td>Invalid value at parameter RatioDenominator.</td>
<td>MC_Gearln</td>
</tr>
<tr>
<td>0x8019</td>
<td>Invalid value at parameter RatioNumerator.</td>
<td>MC_Gearln</td>
</tr>
<tr>
<td>0x801A</td>
<td>Unknown parameter number.</td>
<td>MC_ReadParameter, MC_Write-Parameter</td>
</tr>
<tr>
<td>0x801B</td>
<td>Parameter can not be written, parameter is write protected</td>
<td>MC_WriteParameter</td>
</tr>
<tr>
<td>0x801C</td>
<td>Parameter communication with unknown mode.</td>
<td>MC_Home, MC_WriteParameter</td>
</tr>
<tr>
<td>ErrorID</td>
<td>Description</td>
<td>Remark</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------------------------------------------------</td>
</tr>
<tr>
<td>0x801D</td>
<td>Parameter communication with general error. The cause of the error is not described in detail.</td>
<td></td>
</tr>
<tr>
<td>0x801E</td>
<td>SDO parameter value out of range.</td>
<td>MC_Home, MC_WriteParameter</td>
</tr>
<tr>
<td>0x801F</td>
<td>The Type in ANY is not BYTE.</td>
<td>Read/write parameter</td>
</tr>
<tr>
<td>0x8020</td>
<td>Different configuration of the user units in cam and master axis.</td>
<td></td>
</tr>
<tr>
<td>0x8021</td>
<td>Different configuration of the user units in cam and slave axis.</td>
<td></td>
</tr>
<tr>
<td>0x8022</td>
<td>There is no PROFIBUS/PROFINET device at the logical address specified in LADDR, from which you can read consistent data.</td>
<td>Read/write parameter</td>
</tr>
<tr>
<td>0x8023</td>
<td>An access error has been detected when accessing an I/O device.</td>
<td>Read/write parameter</td>
</tr>
<tr>
<td>0x8024</td>
<td>Slave error at external DP slave.</td>
<td>Read/write parameter</td>
</tr>
<tr>
<td>0x8025</td>
<td>System error at external DP slave.</td>
<td>Read/write parameter</td>
</tr>
<tr>
<td>0x8026</td>
<td>System error at external DP slave.</td>
<td>Read/write parameter</td>
</tr>
<tr>
<td>0x8027</td>
<td>The data haven't yet been read by the module.</td>
<td>Read/write parameter</td>
</tr>
<tr>
<td>0x8028</td>
<td>System error at external DP slave.</td>
<td>Read/write parameter</td>
</tr>
<tr>
<td>0x8029</td>
<td>Attempt to write a read only object.</td>
<td>Read/write parameter</td>
</tr>
<tr>
<td>0x802A</td>
<td>Attempt to read a write only object.</td>
<td>Read/write parameter</td>
</tr>
<tr>
<td>0x802B</td>
<td>Unsupported access to an object.</td>
<td>Read/write parameter</td>
</tr>
<tr>
<td>0x802C</td>
<td>Wrong data type.</td>
<td>Read/write parameter</td>
</tr>
<tr>
<td>0x802D</td>
<td>Error in device profile.</td>
<td>Read/write parameter</td>
</tr>
<tr>
<td>0x802E</td>
<td>Error command type.</td>
<td>Read/write parameter</td>
</tr>
<tr>
<td>0x802F</td>
<td>No system resources available.</td>
<td>Read/write parameter</td>
</tr>
<tr>
<td>0x8030</td>
<td>Invalid value at parameter Hardware (1 = SLIO CP; 2 = VIPA CPU).</td>
<td>Modbus; Init</td>
</tr>
<tr>
<td>0x8031</td>
<td>Invalid value at parameter Unitld.</td>
<td>Modbus; Init</td>
</tr>
<tr>
<td>0x8032</td>
<td>Invalid value at parameter UserUnitsVelocity (0 = Hz, 1 = %, 2 = RPM).</td>
<td>Modbus; Init</td>
</tr>
<tr>
<td>0x8033</td>
<td>Invalid value at parameter UserUnitsAcceleration (0 = 0.00s, 1 = 0.0s).</td>
<td>Modbus; Init</td>
</tr>
<tr>
<td>0x8034</td>
<td>Invalid value at parameter MaxVelocityApp (must be &gt; 0).</td>
<td>Modbus; Init</td>
</tr>
<tr>
<td>0x8035</td>
<td>Error while read access at MonitorData.</td>
<td>Modbus; Init</td>
</tr>
<tr>
<td>0x8036</td>
<td>Error while read access at NumberOfPoles.</td>
<td>Modbus; Init</td>
</tr>
<tr>
<td>0x8037</td>
<td>Error while write access to UserUnitsVelocity.</td>
<td>Modbus; Init</td>
</tr>
<tr>
<td>0x8038</td>
<td>Error while read access at MinOutputFrequency.</td>
<td>Modbus; Init</td>
</tr>
<tr>
<td>0x8039</td>
<td>Error while read access at MaxOutputFrequency.</td>
<td>Modbus; Init</td>
</tr>
<tr>
<td>0x803A</td>
<td>Error while write access to StoppingMethodSelection.</td>
<td>Modbus; Init</td>
</tr>
<tr>
<td>0x803B</td>
<td>Error while write access to UserUnitsAcceleration.</td>
<td>Modbus; Init</td>
</tr>
<tr>
<td>0x8041</td>
<td>Invalid value at parameter AccelerationTime.</td>
<td>Modbus V1000</td>
</tr>
<tr>
<td>0x8042</td>
<td>Invalid value at parameter DecelerationTime.</td>
<td>Modbus V1000</td>
</tr>
<tr>
<td>0x8043</td>
<td>Invalid value at parameter JogAccelerationTime.</td>
<td>Modbus V1000</td>
</tr>
<tr>
<td>ErrorID</td>
<td>Description</td>
<td>Remark</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>0x8044</td>
<td>Invalid value at parameter JogDecelerationTime.</td>
<td>Modbus V1000</td>
</tr>
<tr>
<td>0x8045</td>
<td>Invalid value at parameter JogVelocity (≤ MaxVelocityApp).</td>
<td>Modbus V1000</td>
</tr>
<tr>
<td>0x80C8</td>
<td>Modbus communication error: No response from the server in the defined period (timeout can be parametrized via interface).</td>
<td>Modbus V1000</td>
</tr>
<tr>
<td>0x809y</td>
<td>Error in value of the block parameter y, with y:</td>
<td>VMC_ConfigMaster_RTU</td>
</tr>
<tr>
<td></td>
<td>1: Error in PROTOCOL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3: Error in BAUDRATE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4: Error in CHARLENGTH</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5: Error in PARITY</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6: Error in STOPBITS</td>
<td></td>
</tr>
<tr>
<td>0x8092</td>
<td>Access error on parameter DB (DB too short).</td>
<td>VMC_ConfigMaster_RTU</td>
</tr>
<tr>
<td>0x809A</td>
<td>Interface not available or used with PROFIBUS.</td>
<td>VMC_ConfigMaster_RTU</td>
</tr>
<tr>
<td>0x8101</td>
<td>No cyclic communication with axis possible.</td>
<td></td>
</tr>
<tr>
<td>0x8102</td>
<td>Command is in current PLCopen-State not allowed.</td>
<td></td>
</tr>
<tr>
<td>0x8103</td>
<td>Command is not supported by the axis.</td>
<td></td>
</tr>
<tr>
<td>0x8104</td>
<td>Axis is not ready to switch on, possible reasons:</td>
<td>PreOperational has also to be set in Operational.</td>
</tr>
<tr>
<td></td>
<td>Communication to the axis is not ready.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Drive is not in status ‘switched on’→ reset drive error possibly with MC_Reset.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Communication was interrupted, e.g. by CPU power cycle. Reset error with MC_Reset.</td>
<td></td>
</tr>
<tr>
<td>0x8105</td>
<td>Command is not supported by virtual axes.</td>
<td></td>
</tr>
<tr>
<td>0x8106</td>
<td>PLCopen-State is not defined.</td>
<td></td>
</tr>
<tr>
<td>0x8107</td>
<td>Command is not permitted if drive is deactivated.</td>
<td>VMC_AxisControl_PT, ModbusV1000</td>
</tr>
<tr>
<td>0x8188</td>
<td>Modbus communication error: Internal error MB_FUNCTION invalid.</td>
<td>Modbus V1000</td>
</tr>
<tr>
<td>0x8189</td>
<td>Modbus communication error: Internal error MB_DATA_ADDR invalid.</td>
<td>Modbus V1000</td>
</tr>
<tr>
<td>0x818A</td>
<td>Modbus communication error: Internal error MB_DATA_LEN invalid.</td>
<td>Modbus V1000</td>
</tr>
<tr>
<td>0x818B</td>
<td>Modbus communication error: Internal error MB_DATA_PTR invalid.</td>
<td>Modbus V1000</td>
</tr>
<tr>
<td>0x8201</td>
<td>Command cannot be executed temporarily because of lack of internal resources (no free slot in CommandBuffer).</td>
<td></td>
</tr>
<tr>
<td>0x8202</td>
<td>Error when writing the offset for homing (no free slot in the CommandBuffer).</td>
<td>DriveManager → Homing (active command)</td>
</tr>
<tr>
<td>0x8210</td>
<td>Modbus communication error: The hardware is incompatible with the Modbus RTU/TCP block library.</td>
<td>Modbus V1000</td>
</tr>
<tr>
<td>0x828y</td>
<td>Error in parameter y of DB parameters, with y:</td>
<td>VMC_ConfigMaster_RTU</td>
</tr>
<tr>
<td></td>
<td>1: Error in 1. Parameter</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2: Error in the 2. Parameter</td>
<td></td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>ErrorID</td>
<td>Description</td>
<td>Remark</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>0x8301</td>
<td>No cyclic communication with master axis possible.</td>
<td></td>
</tr>
<tr>
<td>0x8302</td>
<td>Command is in current PLCopen-State of the master axis not allowed.</td>
<td></td>
</tr>
<tr>
<td>0x8303</td>
<td>Command is not supported by the master axis.</td>
<td></td>
</tr>
<tr>
<td>0x8304</td>
<td>Master axis is not in status Pre-Operational.</td>
<td></td>
</tr>
<tr>
<td>0x8305</td>
<td>Master axis data block number has been changed.</td>
<td></td>
</tr>
<tr>
<td>0x8306</td>
<td>Communication errors at the master axis. Slave axis is stopped with fast stop.</td>
<td></td>
</tr>
<tr>
<td>0x8311</td>
<td>No cyclic communication with slave axis possible.</td>
<td></td>
</tr>
<tr>
<td>0x8312</td>
<td>Command is in current PLCopen-State of the slave axis not allowed.</td>
<td></td>
</tr>
<tr>
<td>0x8313</td>
<td>Command is not supported by the slave axis.</td>
<td></td>
</tr>
<tr>
<td>0x8314</td>
<td>Slave axis is not in status Pre-Operational.</td>
<td></td>
</tr>
<tr>
<td>0x8315</td>
<td>Slave axis data block number has been changed.</td>
<td></td>
</tr>
<tr>
<td>0x8317</td>
<td>Block was not called within OB 1.</td>
<td>VMC_AxisControl_PT</td>
</tr>
<tr>
<td>0x8321</td>
<td>Coupling with StartMode = relative and ActivationMode = nextcycle is not permitted.</td>
<td></td>
</tr>
<tr>
<td>0x8322</td>
<td>Coupling or switching with StartMode = absolute and ActivationMode = nextcycle is not permitted.</td>
<td></td>
</tr>
<tr>
<td>0x8323</td>
<td>Switching with a different StartMode (StartMode of the coupling is to be used).</td>
<td></td>
</tr>
<tr>
<td>0x8331</td>
<td>MC_CamIn is not active.</td>
<td></td>
</tr>
<tr>
<td>0x8332</td>
<td>MC_GearIn is not active.</td>
<td></td>
</tr>
<tr>
<td>0x8340</td>
<td>Invalid value at TriggerInput.Probe.</td>
<td>MC_TouchProbe and MC_Abort-Trigger</td>
</tr>
<tr>
<td>0x8341</td>
<td>Invalid value at TriggerInput.Source.</td>
<td>MC_TouchProbe and MC_Abort-Trigger</td>
</tr>
<tr>
<td>0x8342</td>
<td>Invalid value at TriggerInput.TriggerMode.</td>
<td>MC_TouchProbe and MC_Abort-Trigger</td>
</tr>
<tr>
<td>0x8350</td>
<td>Invalid value at VelocitySearchSwitch.</td>
<td>Homing, initialization</td>
</tr>
<tr>
<td>0x8351</td>
<td>Invalid value at VelocitySearchZero.</td>
<td>Homing, initialization</td>
</tr>
<tr>
<td>0x8352</td>
<td>Invalid combination of inputs.</td>
<td>Homing, initialization</td>
</tr>
<tr>
<td>0x8360</td>
<td>The CPU does not support Pulse Train.</td>
<td>VMC_AxisControl_PT</td>
</tr>
<tr>
<td>0x8361</td>
<td>Wrong value in S_ChannelNumberPWM.</td>
<td>VMC_AxisControl_PT</td>
</tr>
<tr>
<td>0x8362</td>
<td>General error in Pulse Train output.</td>
<td>VMC_AxisControl_PT</td>
</tr>
<tr>
<td>0x8363</td>
<td>Move command with the StopExecute set.</td>
<td>VMC_AxisControl_PT, ModbusV1000</td>
</tr>
<tr>
<td>0x8381</td>
<td>Modbus communication error: Server returns Exception code 01h.</td>
<td>Modbus V1000</td>
</tr>
<tr>
<td>0x8382</td>
<td>Modbus communication error: Server returns Exception code 03h or wrong start address.</td>
<td>Modbus V1000</td>
</tr>
<tr>
<td>ErrorID</td>
<td>Description</td>
<td>Remark</td>
</tr>
<tr>
<td>---------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>0x8383</td>
<td>Modbus communication error: Server returns Exception code 02h.</td>
<td>Modbus V1000</td>
</tr>
<tr>
<td>0x8384</td>
<td>Modbus communication error: Server returns Exception code 04h.</td>
<td>Modbus V1000</td>
</tr>
<tr>
<td>0x8386</td>
<td>Modbus communication error: Server returns wrong function code.</td>
<td>Modbus V1000</td>
</tr>
<tr>
<td>0x8388</td>
<td>Modbus communication error: Server returns wrong value or wrong number.</td>
<td>Modbus V1000</td>
</tr>
<tr>
<td>0x8400</td>
<td>MC_Power: Unexpected Drive-State Drive-State &lt;&gt; Operation enabled</td>
<td>MC_Power</td>
</tr>
<tr>
<td>0x8401</td>
<td>MC_Power: Unexpected Drive-State Drive-State = Quick stop active</td>
<td>MC_Power</td>
</tr>
<tr>
<td>0x8402</td>
<td>MC_Power: Unexpected Drive-State Drive-State = Fault reaction active</td>
<td>MC_Power</td>
</tr>
<tr>
<td>0x8403</td>
<td>MC_Power: Unexpected Drive-State Drive-State = Fault</td>
<td>MC_Power</td>
</tr>
<tr>
<td>0x8410</td>
<td>Timeout while trying to reset the drive.</td>
<td>Kernel FB --&gt; MC_Reset</td>
</tr>
<tr>
<td>0x8500</td>
<td>Wrong value in EncoderType (1 or 2).</td>
<td>Init block</td>
</tr>
<tr>
<td>0x8501</td>
<td>Wrong value in EncoderResolutionBits (&gt;0 and ≤32).</td>
<td>Init block</td>
</tr>
<tr>
<td>0x8502</td>
<td>Wrong value in LogicalAddress ( ≥0).</td>
<td>Init block</td>
</tr>
<tr>
<td>0x8503</td>
<td>Wrong value in StartInputAddress ( ≥0).</td>
<td>Init block</td>
</tr>
<tr>
<td>0x8504</td>
<td>Wrong value in StartOutputAddress ( ≥0).</td>
<td>Init block</td>
</tr>
<tr>
<td>0x8505</td>
<td>Wrong value in FactorPosition (&gt;0.0).</td>
<td>Init block</td>
</tr>
<tr>
<td>0x8506</td>
<td>Wrong value in FactorVelocity (&gt;0.0).</td>
<td>Init block</td>
</tr>
<tr>
<td>0x8507</td>
<td>Wrong value in FactorAcceleration (&gt;0.0).</td>
<td>Init block</td>
</tr>
<tr>
<td>0x8508</td>
<td>Wrong value in MaxVelocityApp (&gt;0.0).</td>
<td>Init block</td>
</tr>
<tr>
<td>0x8509</td>
<td>Wrong value in MaxAccelerationApp (&gt;0.0).</td>
<td>Init block</td>
</tr>
<tr>
<td>0x850A</td>
<td>Wrong value in MaxDecelerationApp (&gt;0.0).</td>
<td>Init block</td>
</tr>
<tr>
<td>0x850B</td>
<td>Wrong value in MaxVelocityDrive (&gt;0.0).</td>
<td>Init block</td>
</tr>
<tr>
<td>0x850C</td>
<td>Wrong value in MaxAccelerationDrive (&gt;0.0).</td>
<td>Init block</td>
</tr>
<tr>
<td>0x850D</td>
<td>Wrong value in MaxDecelerationDrive (&gt;0.0).</td>
<td>Init block</td>
</tr>
<tr>
<td>0x850E</td>
<td>Wrong value in MinPosition (≥MinUserPos).</td>
<td>Init block</td>
</tr>
<tr>
<td>0x850F</td>
<td>Wrong value in MaxPosition (≥MaxUserPos).</td>
<td>Init block</td>
</tr>
<tr>
<td>0x8510</td>
<td>Wrong value in M2_ENCODERTYPE.</td>
<td>VMC_InitSigma7W_EC</td>
</tr>
<tr>
<td>0x8511</td>
<td>Wrong value in M2_ENCODERRESOLUTIONBITS.</td>
<td>VMC_InitSigma7W_EC</td>
</tr>
<tr>
<td>0x8513</td>
<td>Wrong value in M2_PDOI.</td>
<td>VMC_InitSigma7W_EC</td>
</tr>
<tr>
<td>0x8514</td>
<td>Wrong value in M2_PDO.</td>
<td>VMC_InitSigma7W_EC</td>
</tr>
<tr>
<td>0x8515</td>
<td>Wrong value in M2_FACTORPOSITION.</td>
<td>VMC_InitSigma7W_EC</td>
</tr>
<tr>
<td>0x8516</td>
<td>Wrong value in M2_FACTORVELOCITY.</td>
<td>VMC_InitSigma7W_EC</td>
</tr>
<tr>
<td>0x8517</td>
<td>Wrong value in M2_FACTORACCELERATION.</td>
<td>VMC_InitSigma7W_EC</td>
</tr>
<tr>
<td>ErrorID</td>
<td>Description</td>
<td>Remark</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------------------------------------------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>0x8518</td>
<td>Wrong value in $M2_{MaxVelocityApp}$.</td>
<td>VMC_InitSigma7W_EC</td>
</tr>
<tr>
<td>0x8519</td>
<td>Wrong value in $M2_{MaxAccelerationApp}$.</td>
<td>VMC_InitSigma7W_EC</td>
</tr>
<tr>
<td>0x851A</td>
<td>Wrong value in $M2_{MaxDecelerationApp}$.</td>
<td>VMC_InitSigma7W_EC</td>
</tr>
<tr>
<td>0x851D</td>
<td>Wrong value in $ParaAccessPointAddress$.</td>
<td>VMC_InitSigma_PN</td>
</tr>
<tr>
<td>0x8603</td>
<td>Error homing at the drive, speed $&lt;&gt;$ 0.</td>
<td>MC_Home</td>
</tr>
<tr>
<td>0x8604</td>
<td>Error homing at the drive, speed $= 0$.</td>
<td>MC_Home</td>
</tr>
<tr>
<td>0x8700</td>
<td>Error: Invalid size.</td>
<td></td>
</tr>
<tr>
<td>0x8710</td>
<td>SDO error: Toggle bit has not changed.</td>
<td></td>
</tr>
<tr>
<td>0x8711</td>
<td>SDO error: SDO protocol timeout.</td>
<td></td>
</tr>
<tr>
<td>0x8712</td>
<td>SDO error: Client / server command is not valid or unknown.</td>
<td></td>
</tr>
<tr>
<td>0x8713</td>
<td>SDO error: Invalid block size (only in block mode).</td>
<td></td>
</tr>
<tr>
<td>0x8714</td>
<td>SDO error: Invalid sequence number (only in block mode).</td>
<td></td>
</tr>
<tr>
<td>0x8715</td>
<td>SDO error: CRC error (only in block mode).</td>
<td></td>
</tr>
<tr>
<td>0x8716</td>
<td>SDO error: Out of memory.</td>
<td></td>
</tr>
<tr>
<td>0x8717</td>
<td>SDO error: Unsupported access to an object.</td>
<td></td>
</tr>
<tr>
<td>0x8718</td>
<td>SDO error: Attempt to read a write only object.</td>
<td></td>
</tr>
<tr>
<td>0x8719</td>
<td>SDO error: Attempt to write a read only object.</td>
<td></td>
</tr>
<tr>
<td>0x871A</td>
<td>SDO error: Object does not exist in the object dictionary.</td>
<td></td>
</tr>
<tr>
<td>0x871B</td>
<td>SDO error: Object can not be mapped to a PDO.</td>
<td></td>
</tr>
<tr>
<td>0x871C</td>
<td>SDO error: The number and length of objects to be mapped exceeds the PDO length.</td>
<td></td>
</tr>
<tr>
<td>0x871D</td>
<td>SDO error: General parameter incompatibility.</td>
<td></td>
</tr>
<tr>
<td>0x871E</td>
<td>SDO error: General internal incompatibility in the device.</td>
<td></td>
</tr>
<tr>
<td>0x871F</td>
<td>SDO error: Access failed due to a hardware error.</td>
<td></td>
</tr>
<tr>
<td>0x8720</td>
<td>SDO error: Data type does not match, length of service parameter does not match.</td>
<td></td>
</tr>
<tr>
<td>0x8721</td>
<td>SDO error: Data type does not match, service parameter too long.</td>
<td></td>
</tr>
<tr>
<td>0x8722</td>
<td>SDO error: Data type does not match, service parameter too short.</td>
<td></td>
</tr>
<tr>
<td>0x8723</td>
<td>SDO error: There is no subindex.</td>
<td></td>
</tr>
<tr>
<td>0x8724</td>
<td>SDO error: Write access - Parameter value out of range.</td>
<td></td>
</tr>
<tr>
<td>0x8725</td>
<td>SDO error: Write access - Parameter value out of high limit</td>
<td></td>
</tr>
<tr>
<td>0x8726</td>
<td>SDO error: Write access - Parameter value out of low limit.</td>
<td></td>
</tr>
<tr>
<td>0x8727</td>
<td>SDO error: Maximum value $&lt;$ Minimum value.</td>
<td></td>
</tr>
<tr>
<td>0x8728</td>
<td>SDO error: General error.</td>
<td></td>
</tr>
<tr>
<td>0x8729</td>
<td>SDO error: Unable to transfer or store data to application.</td>
<td></td>
</tr>
<tr>
<td>0x872A</td>
<td>SDO error: Unable to transfer or store data to application because of local.</td>
<td></td>
</tr>
<tr>
<td>ErrorID</td>
<td>Description</td>
<td>Remark</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>0x872B</td>
<td>SDO error: Unable to transfer or store data to application because of present device state.</td>
<td></td>
</tr>
<tr>
<td>0x872C</td>
<td>SDO error: The dynamic generation of the object dictionary failed or missing object dictionary.</td>
<td></td>
</tr>
<tr>
<td>0x872D</td>
<td>SDO error: Unknown code.</td>
<td></td>
</tr>
<tr>
<td>0x8750</td>
<td>Wrong value in LADDR.</td>
<td></td>
</tr>
<tr>
<td>0x8751</td>
<td>Type other than BYTE in ANY pointer.</td>
<td></td>
</tr>
<tr>
<td>0x8752</td>
<td>There is no PROFIBUS DP module or PROFINET IO device on the address, specified via LADDR, from which consistent data can be read.</td>
<td></td>
</tr>
<tr>
<td>0x8753</td>
<td>Access error when accessing a PROFINET IO device.</td>
<td></td>
</tr>
<tr>
<td>0x8754</td>
<td>Slave error on the external PROFIBUS DP slave.</td>
<td></td>
</tr>
<tr>
<td>0x8755</td>
<td>Length of the SFB data does not match the length of the user data.</td>
<td></td>
</tr>
<tr>
<td>0x8756</td>
<td>Error on external PROFIBUS DP slave.</td>
<td></td>
</tr>
<tr>
<td>0x8757</td>
<td>System error on external PROFIBUS DP slave.</td>
<td></td>
</tr>
<tr>
<td>0x8758</td>
<td>The data has not yet been read by the device.</td>
<td></td>
</tr>
<tr>
<td>0x8759</td>
<td>System error on external PROFIBUS DP slave.</td>
<td></td>
</tr>
<tr>
<td>0x875A</td>
<td>No system resources are available.</td>
<td></td>
</tr>
<tr>
<td>0x8799</td>
<td>SDO error: An other error appeared, for more information, see the data of Info1 and Info2.</td>
<td></td>
</tr>
<tr>
<td>0x8888</td>
<td>Internal: BufferIndex error</td>
<td>VMC_AxisControl</td>
</tr>
<tr>
<td>0x8A00</td>
<td>Access to unavailable parameter.</td>
<td>VMC_AxisControlSigma_PN</td>
</tr>
<tr>
<td>0x8A01</td>
<td>Access to a parameter value that cannot be changed.</td>
<td>VMC_AxisControlSigma_PN</td>
</tr>
<tr>
<td>0x8A02</td>
<td>Access with value outside the limits.</td>
<td>VMC_AxisControlSigma_PN</td>
</tr>
<tr>
<td>0x8A03</td>
<td>Access to unavailable subindex.</td>
<td>VMC_AxisControlSigma_PN</td>
</tr>
<tr>
<td>0x8A04</td>
<td>Access with subindex to non-indexed parameter.</td>
<td>VMC_AxisControlSigma_PN</td>
</tr>
<tr>
<td>0x8A05</td>
<td>Invalid data type</td>
<td>VMC_AxisControlSigma_PN</td>
</tr>
<tr>
<td>0x8A06</td>
<td>Access with value ≠ 0 when this is not permitted.</td>
<td>VMC_AxisControlSigma_PN</td>
</tr>
<tr>
<td>0x8A07</td>
<td>Access to a description element that cannot be changed.</td>
<td>VMC_AxisControlSigma_PN</td>
</tr>
<tr>
<td>0x8A09</td>
<td>Access to an unavailable description.</td>
<td>VMC_AxisControlSigma_PN</td>
</tr>
<tr>
<td>0x8A0B</td>
<td>Access without rights to change parameters.</td>
<td>VMC_AxisControlSigma_PN</td>
</tr>
<tr>
<td>0x8A0F</td>
<td>Access to text array that is not available.</td>
<td>VMC_AxisControlSigma_PN</td>
</tr>
<tr>
<td>0x8A11</td>
<td>Access is temporarily not possible.</td>
<td>VMC_AxisControlSigma_PN</td>
</tr>
<tr>
<td>0x8A14</td>
<td>Access with a value that is within limits but not currently not possible.</td>
<td>VMC_AxisControlSigma_PN</td>
</tr>
<tr>
<td>0x8A15</td>
<td>The length of the current response exceeds the maximum possible length.</td>
<td>VMC_AxisControlSigma_PN</td>
</tr>
<tr>
<td>0x8A16</td>
<td>Invalid value respectively value is not supported by the parameter type.</td>
<td>VMC_AxisControlSigma_PN</td>
</tr>
<tr>
<td>ErrorID</td>
<td>Description</td>
<td>Remark</td>
</tr>
<tr>
<td>---------</td>
<td>--------------------------------------------------------------------------------------------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>0x8A17</td>
<td>Error while write access to parameter: Invalid format</td>
<td>VMC_AxisControlSigma_PN</td>
</tr>
<tr>
<td>0x8A18</td>
<td>Error while write access to parameter: Number of parameter does not match the number of elements at the parameter address.</td>
<td>VMC_AxisControlSigma_PN</td>
</tr>
<tr>
<td>0x8A19</td>
<td>Error while write access to a digital output, which does not exist.</td>
<td>VMC_AxisControlSigma_PN</td>
</tr>
<tr>
<td>0x8A20</td>
<td>Write access to a parameter text, which cannot be changed</td>
<td>VMC_AxisControlSigma_PN</td>
</tr>
<tr>
<td>0x8A21</td>
<td>Invalid request ID</td>
<td>VMC_AxisControlSigma_PN</td>
</tr>
<tr>
<td>0x8A22</td>
<td>Max number of requested parameters is reached.</td>
<td>VMC_AxisControlSigma_PN</td>
</tr>
<tr>
<td>0xC000</td>
<td>Internal error: Status Init is undefined.</td>
<td>Modbus; Init</td>
</tr>
<tr>
<td>0xC001</td>
<td>Internal error: Invalid value at parameter <code>Cmd.ActiveType</code>.</td>
<td>Modbus V1000</td>
</tr>
<tr>
<td>0xC002</td>
<td>Internal Error: Invalid value at parameter <code>Cmd.State</code>.</td>
<td>Modbus V1000</td>
</tr>
</tbody>
</table>