metronix
servo drives

PROFINET Manual

Important!
Read thoroughly before use!
Retain for future reference!

Original PROFINET Manual

› Copyright

The information and data in this document have been composed to the best of our knowledge. However, deviations between the document and the product cannot be excluded entirely. For the devices and the corresponding software in the version handed out to the customer, Metronix guarantees the contractual use in accordance with the user documentation. In the case of serious deviations from the user documentation, Metronix has the right and the obligation to repair, unless it would involve an unreasonable effort. A possible liability does not include deficiencies caused by deviations from the operating conditions intended for the device and described in the user documentation.

Metronix does not guarantee that the products meet the buyer's demands and purposes or that they work together with other products selected by the buyer. Metronix does not assume any liability for damage resulting from the combined use of its products with other products or resulting from improper handling of machines or systems.

Metronix reserves the right to modify, amend or improve the document or the product without prior notification.

› Trademarks

Any product names in this document may be registered trademarks. The sole purpose of any trademarks in this document is the identification of the corresponding products.

Metronix ServoCommander® is a registered trademark of Metronix Meßgeräte und Elektronik GmbH.

Contact

Metronix Meßgeräte und Elektronik GmbH
Kocherstraße 3
38120 Braunschweig
Germany

Telephone: +49 (0)531 8668 0
Fax: +49 (0)531 8668 555
E-mail: vertrieb@metronix.de
https://www.metronix.de

Revision Information

Manual title	PROFINET Manual
File name	PN-HB_1p 0_EN.pdf
Version	1.0
Year	2024

Table of Contents

1 About this manual

This manual describes how the servo drives of the smartServo BL 4000 device series can be integrated into a PROFINET network. The physical connection, activation of the fieldbus protocol, integration into the network and the parameters for adaptation to the respective application are described. It is intended for persons who are already familiar with the respective servo drive series and have read and understood the corresponding product manual.

The product manual contains instructions for the proper and professional transport, storage, assembly, installation, project planning and correct and safe operation of the servo drive.

The product manual contains safety instructions which must be strictly observed. The product manuals are available for download on our homepage (http://www.metronix.de).

1.1 Structure of the warning notes

Warning notes have the following structure:

- Signal word
- Type of hazard
- Measures to prevent the hazard

› Signal words

⚠ DANGER

Indicates an imminent hazard.
If the situation is not avoided, extremely serious and possibly fatal injuries will result.

⚠ WARNING

Indicates a potentially hazardous situation.
If the situation is not avoided, extremely serious and possibly fatal injuries may result.

⚠ CAUTION

Indicates a potentially hazardous situation.
If the situation is not avoided, slight or minor injuries may result.

NOTICE

Warns against damage to property.

› Warning signs as per ISO 7010

Warning sign	Explanation
	Warning against fatal electric voltage.

1.2 Notation in this manual

› Structure of notes

The notes in this manual have the following structure:

- Signal word "NOTE"
- Introductory phrase
- Explanations and special tips

› Operating elements, menus

Operating elements, menus and menu paths are written in orange.

Example: Double-clicking the desired device or clicking the button Establish connection will establish an online connection.

› PNUs, bit constants

Terms such as parameter names (PNUs) are written in blue. Bit constants are highlighted by a different font.

Example: The first case can be achieved by setting the `Change immediately` bit in Control word 1 when starting the second movement task.

› States, commands

Servo drive states (see section 4 *Device Control* on page 56) are set in a different font and are capitalised. Commands are highlighted with a white box.

Example:

`SWITCHING_ON_INHIBITED`			The servo drive has completed its self-test.	
4	Disable Operation	`0111`		Disabling the servo drive

2 Quick-start guide

This chapter describes how to connect the servo drives to a commercially available PROFINET controller and put them into operation in order to obtain a quick setup for starting application development.

Section 3 *Parameter* on page 23 then describes all available parameters, which can be used to adapt the servo drive to the respective application. The following chapter is intended for users who already have an industrial controller.

2.1 Basics: PROFINET and PROFIdrive

› PROFINET IO

PROFINET IO (Input - Output) provides the connection of decentralized field devices such as I/O, drives, valves, transmitters or analysis devices to a central automation device such as a PLC, PC or process control system. Data transmission is based on Fast Ethernet standard transmission at 100 Mbit/s. Three conformance classes (CC-A, CC-B and CC-C), which build on each other, specify the range of functions and the real-time properties of PROFINET IO. The servo drives in thesmartServo BL 4000 device series fulfill **Conformance Class B** (**CC-B**):

› PROFIdrive

The "PROFIBUS profile for drive technology", PROFIdrive for short, is a standard for manufacturers to implement PROFIBUS interfaces for drives. PROFIdrive specifies configuration, diagnostics, data exchange and state machines with a master. The servo drives of the smartServo device series have implemented a Metronix-specific application profile that is based on PROFIdrive version 3.1. For example, the control of the state machine via Control word 1 and Status word 1 and the meaning of the individual bits have largely been adopted, as well as the concept of parameter numbers (PNU) in order to access the drive parameters. In addition, it is also possible to access all parameters of the CANopen object dictionary. PROFIdrive telegrams such as MC-Servo are not supported.

› Metronix application protocol

The Metronix-specific application profile is based on cyclical communication between the controller and servo drive, which is described in section 6 *Cyclical communication* on page 71. Control via Control word 1 and Status word 1 is described in section 4 *Device Control* on page 56. An overview of all parameters can be found in section 3 *Parameter* on page 23. To simplify the fieldbus connection, all important functions and parameters are summarized in a function block. This is described in section 2.5 *Integrating the servo drive into TIA portal* on page 14.

2.2 Wiring and pin assignment

The PROFINET interface is integrated in the BL 4000-C servo drives and therefore always available. For servo drives of the BL 4000-M / BL 4000-D series, the PROFINET interface is only available with the PROFINET/EtherCAT field bus variant. More detailed information on this can be found in the *Product Description* section of the Product manual BL 4000-D and BL 4000-M.

› BL 4000-C

In accordance with the PROFINET specification, there are two RJ45 plugs for the bus connection [X21].

Figure 1: Connector view fieldbus interface BL 4000-C

The two connections RTE0 and RTE1 are RJ 45 sockets, Cat. 6

Pin	Name	Specification
1	RX-	Receive signal -
2	RX+	Receive signal +
3	TX-	Transmit signal -
4	-	-
5	-	-
6	TX+	Transmit signal +
7	-	-
8	-	-

› BL 4000-D and BL 4000-M (Fieldbus variant PROFINET/EtherCAT)

On these devices, the PROFINET bus connection is designed as an M8 connector according to IEC 61076-114 (4-pin, socket, D-coded). Note that although the fieldbus variant CAN uses the identical connectors, it is not electrically compatible. The fieldbus variants must not be mixed up and must never be used simultaneously in the same network!

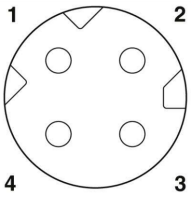

Figure 2: Pin assignment of the fieldbus connector BL 4000-D and BL 4000-M

Pin assignment **EtherCAT/PROFINET**:

Pin	Name	Description	Color
1	TD+	Transmit signal +	Yellow
2	RD+	Receive signal +	White
3	TD-	Transmit signal -	Orange
4	RD-	Receive signal -	Blue

For wiring, we recommend using the following pre-assembled cables or comparable products from other manufacturers:

Assembled network cable Phoenix Contact:

M8 plug to M8 plug: NBC-M8MSD/ 1,0-93C/M8MSD - 1423707
M8 plug to RJ45: NBC-M8MSD/ 1,0-93C/R4AC - 1423711
M8 plug to free cable end: NBC-M8MSD/ 1,0-93C - 1423703

2.2.1 Wiring instructions

Only cables approved for PROFINET with the corresponding manufacturer's declaration should be used in automation systems. The signal lines must be as far away from the power cables as possible. They should not be laid in parallel. The individual PROFINET nodes are generally connected to each other in a linear manner.

2.3 Status LEDs (BL 4000-C)

For easy indication of the CAN bus status, the servo drive is equipped with two fieldbus status LEDs. The behavior of the LEDs is defined by PROFINET.

The upper LED (SF) indicates system errors:

Blinkcode	Description
LED off	No system errors.
LED flashes red (1 Hz for 3 s)	PROFINET device identification
LED lights up red	One of the following errors has occurred: • Watchdog timeout • Channel diagnostics • General or extended diagnostics • System error

The lower LED (BF) indicates possible bus errors:

Blinkcode	Description
LED off	No bus errors.
LED lights up red	One of the following errors has occurred: • No configuration • Error on the physical link • No physical link
LED flashes red (2 Hz)	No data will be transmitted.

2.4 Operation parameter PROFINET

The operation parameter window can be called up in the menu bar of the Metronix ServoCommander® under Parameters/Field bus/PROFINET/Operation parameters.

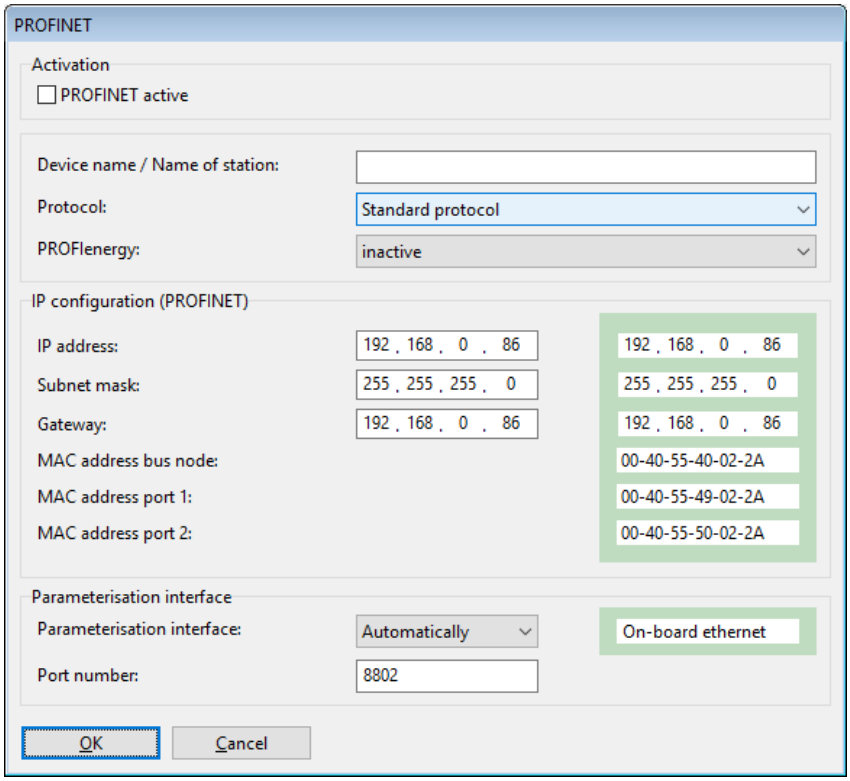

Figure 3: Setting the PROFINET operating parameters

› Activation

PROFINET communication is activated via the PROFINET active check box . It should be noted that the activation of the PROFINET communication is only effective after a Save & Reset. Deactivation of the communication, on the other hand, takes place immediately.

› Name of station, Protocol and PROFIenergy

To configure the communication on the servo drive side, only the Device name (Name of Station) is required. The IP address is assigned on the basis of the device name using the DCP protocol (Discovery and basic Configuration Protocol). For the assignment, it is necessary that a DCP-capable controller is in the network.

The application protocol used is displayed in the Protocol drop-down list. Currently, only the standard protocol is supported.

With the drop-down list PROFIEnergy, the standardised energy efficiency profile PROFIenergy can be activated or deactivated.

› IP configuration (PROFINET)

A unique IP address must be assigned to the servo drive. In the case of dynamic address assignment, the IP address as well as the associated subnet mask and the gateway are assigned via the DCP protocol (based on the Device name). A previously assigned static IP address is overwritten.

› Parameterisation interface

The servo drive can be parameterised or diagnosed using the Metronix ServoCommander® either via the on-board Ethernet interface [X18] or via the PROFINET network. To use the Metronix ServoCommander® in the PROFINET network, the Parameterisation interface must be configured to PROFINET. The Ethernet port to be used can be specified in the Port number input field.

› Save & Reset

The settings of the operating parameters only become valid when the Save & Reset button is pressed. This saves the settings in the parameter set and then resets the servo drive.

2.5 Integrating the servo drive into TIA portal

For fast integration of the servo drive into the TIA portal, a sample project including an Example Function block is provided. A DCO file matching the function block is available for the servo drive.

All sample projects and parameter files can be downloaded from our homepage (https://www.metronix.de).

NOTICE **Damage to property due to malfunction**

If you use the Example Function blocks or the sample project in your application, you must finally check whether all function and safety-relevant requirements of your application are fulfilled.

› Opening the sample project

First download the sample project for TIA Portal from our homepage (http://www.metronix.de). After extracting the ZIP file, you can open the following file in the TIA Portal:

UnifiedFunctionBlock\TIAv7_UnifiedFunctionBlock.ap*xx*

The placeholders xx indicate the version number of the TIA portal with which the file was created.

By using the sample project, the appropriate device description file (GSDML) and the Example Function block are automatically installed.

The project is designed for an S7-1511 CPU. If you use a different CPU, you must replace the S7-1511 in the topology view with your type. The example also uses a BL 4102-C, which you may have to replace with the servo drive type you are using. All servo drives of the BL 4000 servo drive series have the identical PROFINET interface.

› Loading the parameter file

Three requirements must be met on the servo drive side so that the servo drive and the controller can exchange data cyclically:

- The controller must have a suitable device name.
- The cyclical telegrams must be set appropriately.
- PROFINET communication must be activated.

The configuration on the servo drive side matching the example project is stored in the following parameter file, which are loaded into the servo drive using the Metronix ServoCommander®:

ProfiNET_UnifiedFunctionBlock.DCO

This file only contains the PROFINET-specific parameterisation, but does not change the motor or encoder-specific parameters of the servo drive (motor current, angle encoder, etc.).

2.5.1 Example Function block

Access to the servo drive is encapsulated in a function block, which you can find in the TIA Portal under Program blocks, Main.

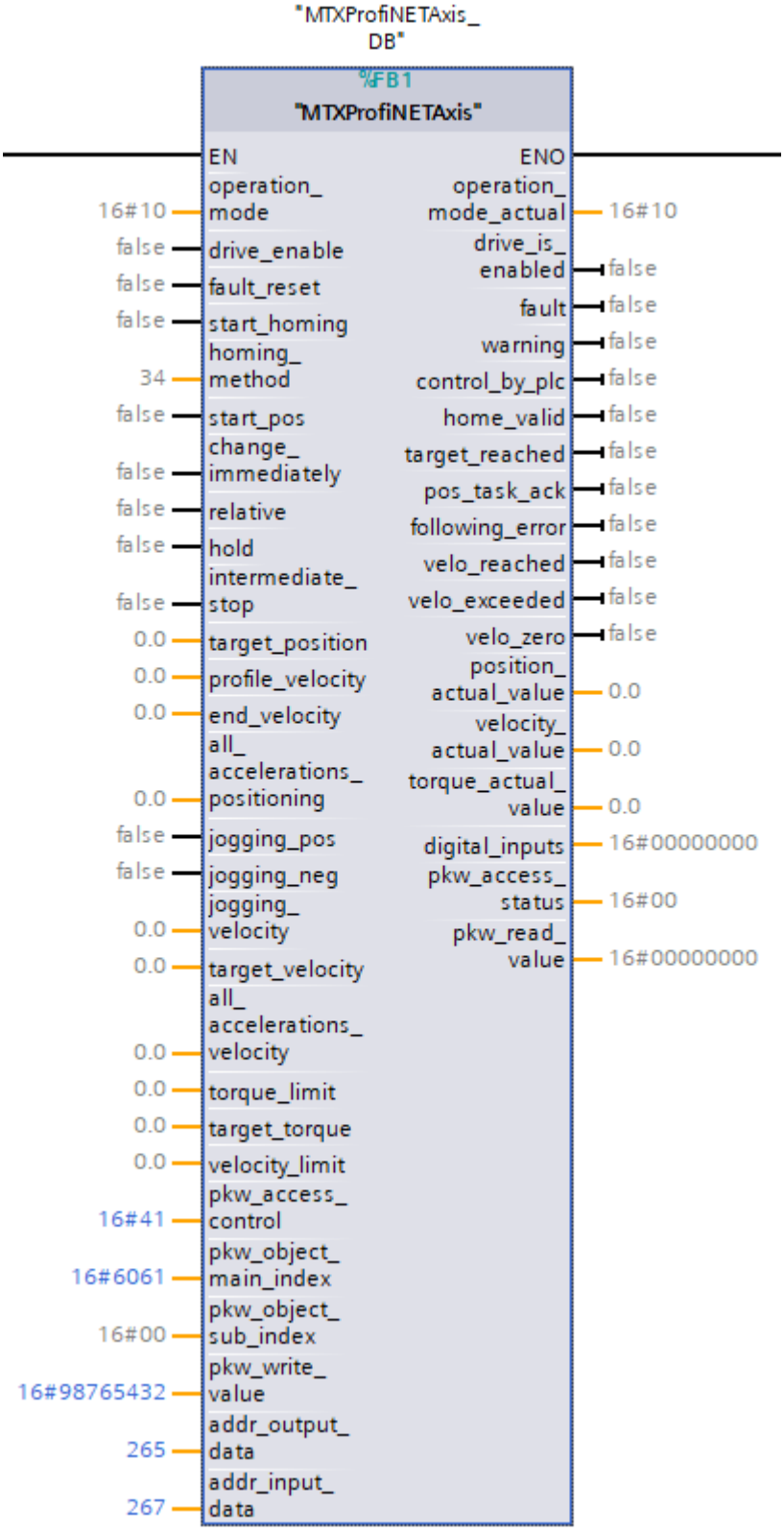

Figure 4: Example Function block

The function block can be used for the positioning and speed control operating modes as well as for torque control. In addition, a torque limitation or a speed limitation can be used. The function block has inputs to enable the controller, change the operating mode and specify setpoints. The current status (enabled, error, etc.), the operating mode and the actual values of the controller can be read out via the outputs. If additional parameters are required, these can be found in section 3 *Parameter* on page 23

⚠ CAUTION Risk of injury due to incorrectly parameterised servo drive

An incorrectly parameterised servo drive can cause uncontrolled rotary movements and thus personal injury or damage to property.

Before switching on the power stage for the very first time, make sure that the servo drive contains the desired parameters.

2.5.1.1 General inputs/outputs

Input	Meaning
addr_output_data	Hardware identifier of the **output data** of the connected servo drive. The FB is linked to the respective axis via this. The hardware identifier is displayed in the TIA Portal under the system constants of the respective controller. It is already set correctly in the example project.
addr_input_data	Hardware identifier of the **input data** of the connected servo drive.

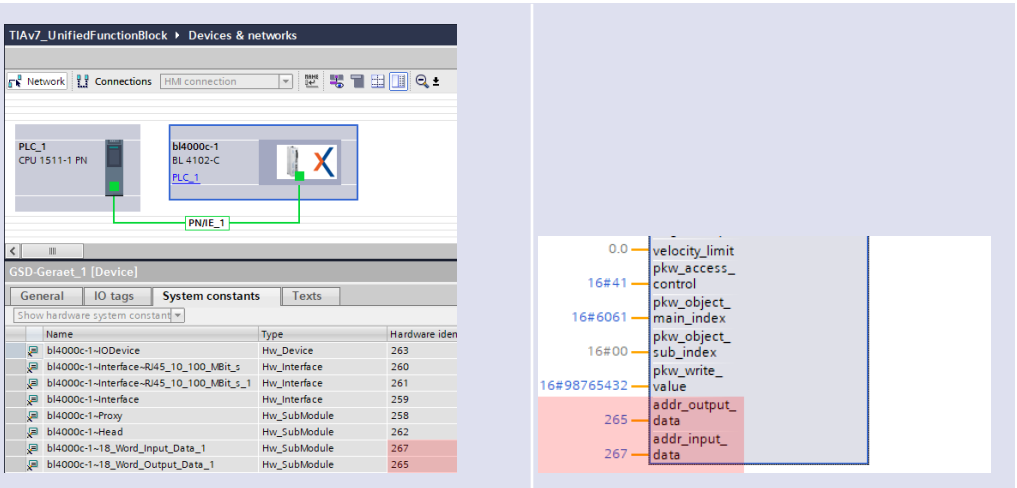

Figure 5: Determination of the HW identifier

Output	Meaning
Control by PLC	This output maps bit 9 of Status word 1 (control by the PLC). The corresponding bit in Control word 1 is set automatically by the Example Function block.

2.5.1.2 Inputs/outputs for all operating modes

The following inputs and outputs can be used regardless of the set operating mode.

Input	Meaning
operation_mode	This input defines the operating mode of the servo drive. The following operating modes can be set: Name / Value / Meaning OPMODE_POS / 10_h / Position mode OPMODE_VELO / 08_h / Velocity mode OPMODE_TORQUE / 04_h / Torque mode See also section 5 *Operating modes* on page 68.
drive_enable	Enables the controller, i.e. the output stage is switched on and the motor is controlled according to the set **operation mode**. It is therefore essential to ensure beforehand that the drive is correctly parameterised and that a corresponding setpoint value is zero. The input operates the state machine of the servo drive, see also section 4.4.1 *State diagram: States* on page 63.
fault_reset	Pending error messages are acknowledged on a rising edge at this input. The input corresponds to bit 7 of *PNU 967: Control word 1*.

Output	Meaning
operation_mode_actual	Current **operation mode** of the controller. Changing the operation mode requires several cycles. The change is only complete when the operating mode set at the input can be read via this object. See section 5 *Operating modes* on page 68.
drive_is_enabled	The servo drive is enabled.
fault	An error has occurred when the output is set. The servo drive is disabled. The output is reset after the error has been acknowledged.
warning	If the output is set, there is a warning in the servo drive. In contrast to an error, a warning does not cause the controller to switch off. The output is automatically reset when the reason for the warning has disappeared.
digital_inputs	This output returns the status of the digital inputs. It corresponds to parameter 1141.0, see section 3.3.7 *PNU 1141: Digital Inputs* on page 31
velo_zero	The drive is stopped when the output is set. The output corresponds to bit 13 of *PNU 968: Status word 1*
velo_reached	When the output is set, the actual speed is within the parameterised tolerance window of the speed setpoint (Comparison speed). The output corresponds to bit 8 of *PNU 968: Status word 1*

Output	Meaning
velo_exceeded	If the output is set, the actual speed is greater than the user-definable reference speed. The output corresponds to bit 10 of *PNU 968: Status word 1*
position_actual_value	This output returns the current position. It corresponds to parameter 1100.0, see section 3.3.2 *PNU 1100: Position Actual Value* on page 29.
velocity_actual_value	This output returns the current speed. It corresponds to parameter 1101.0, see section 3.3.3 *PNU 1101: Velocity Actual Value* on page 29.
current_actual_value	This output returns the actual current value. It corresponds to parameter 1102.0, see section 3.3.4 *PNU 1102: Current Actual Value* on page 30.

2.5.1.3 Inputs/outputs for position mode

The following inputs and outputs can be used in positioning mode. For further details, see section 5.4 *Positioning mode* on page 69.

Input	Meaning
start_homing	Starts the reference run defined via *PNU 1050: Homing Method*. The prerequisite is an enabled servo drive, i.e. the **drive_enable_ok** output must be set. Resetting the **start_homing** input during homing cancels it without error. The input corresponds to bit 11 of *PNU 967: Control word 1*.
start_pos	A rising edge signals that a new motion task should be executed. A falling edge has no effect. This input has no effect during a reference run. The input corresponds to bit 6 of *PNU 967: Control word 1*.
relative	If this input is set on a rising edge of the **start_pos** input, positioning is performed relative to the current position setpoint. If this input is not set on a rising edge of **start_pos**, absolute positioning will be executed. The input corresponds to bit 12 of *PNU 967: Control word 1*.
change_immediatly	If this input is set on a rising edge of the start_pos input, a positioning run is cancelled immediately and replaced by the new motion task. If this input is not set on a rising edge on the start_pos input, the new motion task is appended to the end of the current positioning. In this case, the output target_reached is not set at the end of the current positioning, but only at the end of the attached positioning. The input corresponds to bit 13 of *PNU 967: Control word 1*.
intermediate_stop	If this input is not set, a started positioning run is executed. If the input is set while positioning is in progress, the drive will be stopped and remain in position control. The current positioning is not finished. It is continued when the intermediate_stop input is reset. This input has no effect during a reference run. The input corresponds to the inverted bit 5 of *PNU 967: Control word 1*.

Input	Meaning
hold	If this input is set, the current positioning is cancelled. Braking is performed with the deceleration valid for this positioning task. The target_reached output is not set after the process is completed. Resetting the input has no effect. This input has no effect during a reference run. If the input is set, bits 4, 8 and 9 of *PNU 967: Control word 1* will be deleted.
homing_method	This input defines the homing method. It corresponds to parameter 1050.0, see section 3.4.8 *PNU 1050: Homing Method* on page 37.
target_position	This input defines the target position. It corresponds to parameter 1001.0, see section 3.4.2 *PNU 1001: Position Data* on page 33.
profile_velocity	This input defines the profile speed. It corresponds to parameter 1001.1, see section 3.4.2 *PNU 1001: Position Data* on page 33
end_velocity	This input defines the end velocity. It corresponds to parameter 1001.2, see section 3.4.2 *PNU 1001: Position Data* on page 33
all_accelerations_ positioning	This input defines both the acceleration and the deceleration for the positioning run. It corresponds to parameter 1001.5, see section 3.4.2 *PNU 1001: Position Data* on page 33

Output	Meaning
home_valid	This output is set if the reference position is valid. The output is not set during a reference run. The output corresponds to bit 11 of *PNU 968: Status word 1*
target_reached	This output is set if the current position is within the target window after positioning has been completed. The output corresponds to bit 10 of *PNU 968: Status word 1*
following_error	This output is set if there is a following error. The output corresponds to the inverted bit 10 of *PNU 968: Status word 1*
pos_task_ack	This output is reset if the servo drive can process a new positioning job. The output corresponds to the inverted bit 12 of *PNU 968: Status word 1*

2.5.1.4 Inputs/outputs for the speed control operating mode

The following inputs and outputs can be used in speed control mode. See also section 5.3 *Speed control operating mode* on page 69.

Inputs	Meaning
target_velocity	This input defines the target speed. It corresponds to parameter 1010.2, see section 3.5.1 *PNU 1010: Target Velocity* on page 46
current_limit	This input defines the maximum current in mA that is used in speed control mode. See section 3.5.3 *CAN Objekt 2415h: current_limitation* on page 47.

Output	Meaning
---	There are no outputs that are used only in this operating mode.

2.5.1.5 Inputs/outputs for torque control mode

The following inputs and outputs can be used in torque control mode. See also section 5.2 *Torque control operating mode* on page 68.

Input	Meaning
target_torque	This input defines the target current. It corresponds to CAN object $6071_h\_00_h$ (*target_torque*) and is specified in per mille of the nominal current.
velocity_limit	The maximum speed can be limited via this input. See section 3.6.2 *CAN Object 2416h: speed_limitation* on page 48.

Output	Meaning
---	There are no outputs that are used only in this operating mode.

2.5.1.6 Inputs/outputs for jogging mode

This section describes the inputs and outputs for jogging mode.

Input	Meaning
jog_pos	The drive moves in a positive direction as long as this input is set. The speed is specified via the **jogging_velocity** input. The acceleration/deceleration set in position set TIPP 0 is used for acceleration and braking. This input has no effect during a reference run. The input corresponds to bit 8 of *PNU 967: Control word 1*.
jog_neg	The drive moves in a negative direction as long as this input is set. The speed is specified via the **jogging_velocity** input. The acceleration/deceleration set in position set TIPP 1 is used for acceleration and braking. This input has no effect during a reference run. The input corresponds to bit 9 von *PNU 967: Control word 1*.
jogging_velocity	This input defines the speed of the jogging movement. It corresponds to parameter 1040.0, see section 3.7.1 *PNU 1040: Symmetrical Jogging* on page 49.

Output	Meaning
---	There are no outputs that are used only in this operating mode.

2.5.1.7 Inputs/outputs for parameter access (PKW)

Parameter access enables the asynchronous reading/writing of any PNU or any CAN object without the need for cyclical transmission. The inputs and outputs map the access to the PNU 2000, see section 3.8.3 *PNU 2000: PKW Access* on page 54.

Input	Meaning
pkw_object_main_index	This input defines the main index of the PNU or the CAN object that is to be read or written. It corresponds to bytes 1 and 2 of parameter 2000.0.
pkw_object_sub_index	This input defines the subindex of the PNU or the CAN object that is to be read or written. It corresponds to byte 3 of parameter 2000.0.
pkw_write_value	This input defines the value that is to be written to the PNU or the CAN object. It corresponds to bytes 4 to 7 of parameter 2000.0.
pkw_access_control	Reading and writing is controlled via this input. It corresponds to byte 0 of parameter 2000.0.

Output	Meaning
pkw_read_value	This output returns the value that was read from the PNU or from the CAN object. It corresponds to bytes 4 to 7 of parameter 2000.0.
pkw_access_status	The servo drive uses this output to mirror the value from **pkw_access_control** to indicate that reading and writing has been carried out. It corresponds to byte 0 of parameter 2000.0.

3 Parameter

This section describes the parameters (PNU) available under PROFINET that can be used to customise the drive to the respective application.

› Description of the parameters

All parameters of the drive are described in a uniform way. If the parameter is a simple data type (VAR), it is described as follows:

PNU	**Index (decimal)**		
Name	**Name of the Parameter**		
Info	Unit	rw	Data type
Value	Value range	Default value	

If the parameter is a structured data type (ARRAY/RECORD), it is described as follows:

PNU	**Index (decimal)**		
Name	**Name of Parameter group**		
Type	Object Code		Max
Sub-Index	**Subindex (decimal)**		
Name	**Name of parameter**		
Info	Unit	rw	Data type
Value	Value range	Default value	

The individual fields have the following meaning:

Field	Meaning
Index (hexadecimal)	The main index of the described parameter.
Subindex (hexadecimal)	The subindex of the described parameter. If this is not specified, the subindex is zero.
Name of the parameter group	Plain text name of the parameter group.
Name of the parameter	Plain text name of the parameter.
Object code	Specifies whether the data type is simple or structured: • VAR: Simple data type • ARRAY: Group of parameters that all have the same data type. • RECORD: Group of parameters that have different data types.
Max	Maximum subindex of the group.
Data type	Data type of the parameter or the ARRAY: A list of the supported data types can be found in section 0.1 *SDO-Zugriff* on page 1.
Unit	Physical unit of the parameter.
Access	Specifies whether the parameter may be read (ro), written (wr) or read and written (rw).

Field	Meaning
Value range	The range of permissible values for this parameter.
Default value	Value that is effective on factory setting.

3.1 Physical units

3.1.1 Overview

The scaling of the parameters that are transmitted via the bus can be adjusted for the physical units of position, speed and acceleration. This is not necessary when using the Example Function block. It is based on the default setting of the physical units. The following units are set by default:

Name	Description	Unit	Explanation
Position	Physical unit Position	Thousandth of one revolution	1000 = 1 Revolution
Velocity	Phys. Einheit Velocity	min^{-1}	Revolutions per minute
Acceleration	Phys. Einheit Acceleration	$(min^{-1})/s$	Speed increase in revolutions per minute per second

3.1.2 Parameterisation of the physical units

The physical units can be set via the following menu item in the parameterisation program Metronix ServoCommander®:

Parameters/Field bus/PROFINET/Display units

The parameters for the physical units should be set once and should not be changed during a running application. They can be set independently of each other for position, speed and acceleration. The gear factor and a feed constant are specified as separate parameters.

The specified physical units are used to calculate conversion factors consisting of numerator and denominator. The numerator and denominator must not be greater than 32 bits. If there is an overflow when entering the factors, the value is not accepted. In this case, the factors or the physical units must be corrected.

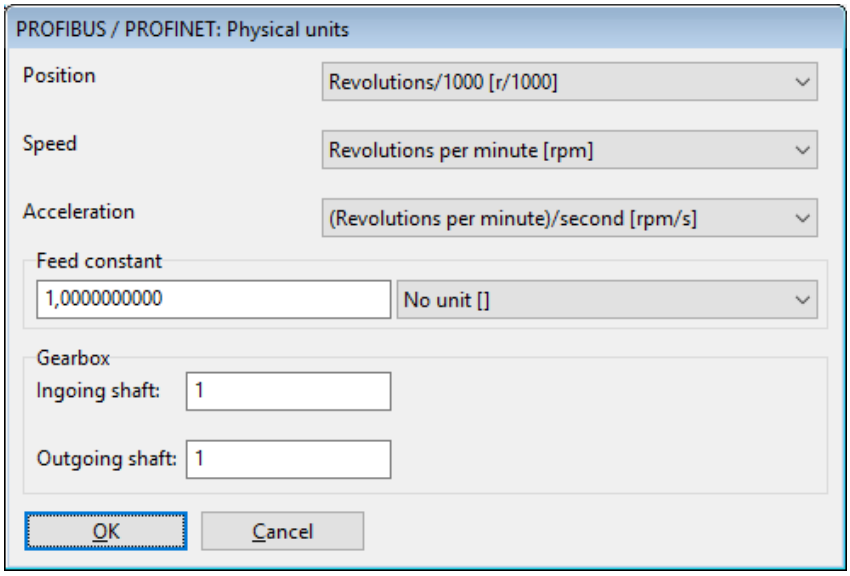

Figure 6: Setting the physical units for PROFIBUS

It should be noted that some quantities do not make sense in all cases. For a purely rotatory system, for example, no feed constant is needed. Furthermore, the feed constant has a physical unit. If this is not parameterised accordingly, the feed constant is not taken into account.

Example

The following cases illustrate the use of the feed constant::

1. Position in Revolutions, feed constant in mm/Revolution:
   => The feed constant is ignored.
2. Position in mm, feed constant without unit:
   => The feed constant is used like a gear factor.
3. Position in mm, feed constant in µm/Revolution
   => The feed constant is taken into account with the factor 1000.

The value of the feed constant is ignored for the respective physical unit if the feed constant has a translational unit and a rotational unit is selected for the physical variable.

Problems are to be expected during operation only if the internal value or the value entered externally can no longer be represented by the conversion. In this case, an error is triggered and the settings of the physical units must be checked.

When setting the parameters of the display units, transient states can occur that lead to an overflow of the physical units, triggering error 22-4. In this case, it is only possible to determine whether the parameterisation is actually invalid by saving and then resetting. If there is no error 22-4 afterwards, the settings are valid.

Gear factor and feed constant can only assume positive values. If the orientation of the application is to be rotated, this can be achieved via the gear factor of the angle encoder in the parameterisation software.

3.2 Overview

The following table provides an overview of the currently implemented PNUs:

PNU	Sub index	Description / Link	Type	Access
1000	0	*Position Set Number*	UINT16	rw
1001	-	*Position Data*		
	0	*Target Position*	INT32	rw
	1	*Profile Velocity*	INT32	rw
	2	*End Velocity*	INT32	rw
	3	*Acceleration Positioning*	UINT32	rw
	4	*Deceleration Positioning*	UINT32	rw
	5	*All Accelerations Positioning*	UINT32	rw
1002	0	*Start Set Number*	UINT8	rw
1003	0	*Position Profile Type*	UINT16	rw
1004	0	*Override Factor*	UINT16	rw
1005	-	*Software Position Limits*		
	0	*Lower software position limit switch*	INT32	rw
	1	*Upper software position limit switch*	INT32	rw
1006	-	*Rotary Axis*		
	0	*Rotary Axis Mode*Rotary Axis Mode (Modus Rundachse)	UINT8	rw
	1	*Lower rotary axis limit*	INT32	rw
	2	*Upper rotary axis limit*	INT32	rw
1010	0	*Target Velocity*	INT32	rw
1011	-	*Accelerations for Velocity Control*		
	0	*Acceleration Velocity Control*	UINT32	rw
	1	*Deceleration Velocity Control*	UINT32	rw
	2	*All Accelerations Velocity Control*	UINT32	rw
1040	-	*Symmetrical Jogging*		
	0	*Symmetrical Jogging Velocity*	INT32	rw
	1	*Symmetrical Jogging Accelerations*	UINT32	rw
1041	-	*Jogging Positive*		
	0	*Jogging Velocity Positive*	INT32	rw
	1	*Jogging Acceleration Positive*	UINT32	rw
	2	*Jogging Deceleration Positive*	UINT32	rw
	3	*Symmetrical Jogging Accelerations Positive*	UINT32	rw

PNU	Sub index	Description / Link	Type	Access
1042	-	*Jogging Negative*		
	0	*Jogging Velocity Negative*	INT32	rw
	1	*Jogging Acceleration Negative*	UINT32	rw
	2	*Jogging Deceleration Negative*	UINT32	rw
	3	*Symmetrical Jogging Accelerations Negative*	UINT32	rw
1050	0	*PNU 1050: Homing Method*	INT8	rw
1051	0	*Home Offset*	INT32	rw
1060	0	*Thread Speed (Einrichtdrehzahl)*	INT32	rw
1100	0	*Position Actual Value*	INT32	ro
1101	0	*Velocity Actual Value*	INT32	ro
1102	0	*Current Actual Value*	INT32	ro
1110	-	*Sampling Positions*		
	0	*Sampling Position Rising Edge*	INT32	ro
	1	*Sampling Position Falling Edge*	INT32	ro
1141	0	*Digital Inputs*	UINT32	ro
1270	-	*Position Control Parameters*		
	2	*Position error tolerance window*	UINT32	rw
1271	-	*Position Window Data*		
	0	*Target Window*	UINT32	rw
1272	-	*Following Error Data*		
	0	*Following Error Window*	UINT32	rw
1273	-	*Position Error Data*		
	0	*Position Error Limit*	UINT32	rw
1500	0	*Operating Mode*	UINT8	ro
1600	0	*Last Error Code*	UINT16	ro
2000	0	*PKW Access*	2xUINT32	rw
2010	-	*Placeholder*		
	0	*8 Bit*	UINT8	rw
	1	*16 Bit*	UINT16	rw
	2	*32 Bit*	UINT32	rw
2011	0	*Placeholder*	UINT32	rw

The Example Function block also uses the following CAN objects:

Main index	Sub index	Description	Type	Access
6071_h	00_h	*target_torque*	INT16	rw
6077_h	00_h	*torque_actual_value*	UINT16	ro
2415_h	-	*current_limitation*		
	01_h	*limit_current*	INT32	rw
2416_h	-	*speed_limitation*		
	01_h	*limit_speed*	INT32	rw

The parameters - grouped according to their intended use - are described in detail in the following sections.

3.3 PNUs for all operating modes

This section lists parameter numbers that can generally be used regardless of the operating mode, such as actual values..

3.3.1 PNU 1500: Operating Mode

This manufacturer-specific parameter allows the operating mode to be set/read. See also section 5 *Operating modes* on page 68.

PNU	**1500**		
Name	**Operating Mode**		
Info	--	rw	UINT8
Value	4_h, 8_h, 10_h	--	

Value	Meaning
4_h	Torque Mode
8_h	Velocity Mode
10_h	Position Mode

3.3.2 PNU 1100: Position Actual Value

The actual position value is returned via this parameter. This is scaled in the physical unit set for PROFINET. Errors can occur when calculating the actual position value, as the internal position of the servo drive has a larger value range than can be transmitted. However, this depends on the set physical units as well as the gear factor and the feed constant. Please contact the Technical Support if necessary.

PNU	**1100**		
Name	**Position Actual Value**		
Info	Physical unit Position	ro	INT32
Value	--	--	

3.3.3 PNU 1101: Velocity Actual Value

The actual speed value is returned via this parameter. This is scaled in the physical unit set for PROFINET.

PNU	**1101**		
Name	**Velocity Actual Value**		
Info	Physical unit speed	ro	INT32
Value	--	--	

3.3.4 PNU 1102: Current Actual Value

This parameter is used to read the actual current value. This is returned in relation to the rated motor current.

PNU	**1102**		
Name	**Current Actual Value**		
Info	Per mille of the rated motor current	ro	INT32
Value	--	--	

3.3.5 CAN Object 6077_h: torque_actual_value

This object can be used to read out the actual torque value of the motor in thousandths of the nominal torque (object 6076_h).

Index	**6077_h**			
Name	**torque_actual_value**			
Info	‰ (1000 = motor_rated torque)	ro	PDO	INT16
Value	--	--		

3.3.6 PNU 1110: Sampling Positions

These parameter numbers provide the positions that have been saved on the rising or falling edge of the so-called sample input. The desired input DIN8 or DIN9 can be parameterised for this purpose via the parameterisation program Metronix ServoCommander® in the „Parameter - IOs - Digital inputs“ window.

Index	**1110**	
Name	**Sampling Positions**	
Type	RECORD	1

Sub-Index	**00**		
Name	**Sampling Position Rising Edge**		
Info	Physical unit Position	ro	INT32
Value	--	--	

Sub-Index	**01**		
Name	**Sampling Position Falling Edge**		
Info	Physical unit Position	ro	INT32
Value	--	--	

3.3.7 PNU 1141: Digital Inputs

This parameter is used to read the status of the digital inputs. The available digital inputs depend on the parameterisation of the servo drive.

PNU	**1141**		
Name	**Digital Inputs**		
Info		ro	UINT32
Value	--	--	

Bit	Meaning
0	Reserved (= 0)
1	DIN 0
2	DIN 1
3	DIN 2
4	DIN 3
5	DIN 4
6	DIN 5 (Digital controller enable)
7	DIN 6 (Limit switch 0 left = negative rotation direction)
8	DIN 7 (Limit switch1 right = positive rotation direction)
9	DIN 8 (Default: Start input)
10	DIN 9 (Default: Sample input)
11...31	Reserved

3.3.8 PNU 1600: Last Error Code

The last error that occurred is output under this parameter number. If there is no error, the PNU returns 0. Drive errors are also signalled to the control unit via the diagnostic interface, see section 4.6 *Diagnosis - Alarms* on page 67.

PNU	**1600**		
Name	**Last Error Code**		
Info	--	ro	UINT16
Value	see Table	0	

Bit	Meaning
0...3	Sub error number (0...9)
4...15	Main error number (1...96)

3.4 PNUs for positioning mode

This section describes the parameters that are required for the Positioning operating mode.

3.4.1 PNU 1000: Position Set Number

This parameter can be used to select the position data set where the data transmitted via PROFINET is entered. This parameter can be used to access all position data sets of the servo drive. The position data set for PROFINET can be saved and can also be parameterised via the parameterisation program. This allows parameters to be permanently specified so that they do not have to be changed in an application during operation. For example, the accelerations can be entered once and do not need to be transferred.

The special position data records for homing or jogging can also be accessed via this parameter. However, due to the special structure of the data records, parameterisation via the Metronix ServoCommander® is recommended here.

PNU	**1000**		
Name	**Position Set Number**		
Info	--	rw	UINT16
Value	0...267	266 (PROFINET)	

Value	Meaning
0...255	Standard position data sets
256	Homing set 0
257	Homing set 1
258	Homing set 2
259	Jogging positive
260	Jogging negative
261...265	Reserved
266	Position data set PROFINET
267	Reserved

3.4.2 PNU 1001: Position Data

Parameters of the selected position data set (PNU1000) can be addressed under this parameter number. The following parameters are available:

- Target position
- Profile velocity
- End velocity
- Acceleration and deceleration, individually or as a combination for both accelerations

The data is parameterised in the currently set physical unit (siehe section 3.1 *Physical units* on page 24).

Parameters, e.g. for jogging, can also be parameterised under this PNU. To do this, the position set number must first be set accordingly, then, for example, the speed for jogging can be set via the profile velocity.

Index	**1001**		
Name	**Position Data**		
Type	RECORD		5

Sub-Index	**00**		
Name	**Target Position**		
Info	Physical unit position	rw	INT32
Value	--	0	

Sub-Index	**01**		
Name	**Profile Velocity**		
Info	Physical unit velocity	rw	INT32
Value	--	1000 min^{-1}	

Sub-Index	**02**		
Name	**End Velocity**		
Info	Physical unit velocity	rw	INT32
Value	--	0	

Sub-Index	**03**		
Name	**Acceleration Positioning**		
Info	Physical unit acceleration	rw	UINT32
Value	--	10.000 $(min^{-1})/s$	

Sub-Index	**04**		
Name	**Deceleration Positioning**		
Info	Physical unit acceleration	rw	UINT32
Value	--	10.000 $(min^{-1})/s$	

The All Acceleration Positioning parameter allows simultaneous access to the acceleration and deceleration. If both parameters should have the same value, only one data value needs to be transferred. This is then written internally to both accelerations. Please note that when reading, only the current value of the acceleration is read. The user must ensure that reading of this value is sufficient. This can be achieved, for example, by reading the value once and then writing it back.

Sub-Index	**05**		
Name	**All Accelerations Positioning**		
Info	Physical unit acceleration	rw	UINT32
Value	--	10.000 (min^{-1})/s	

3.4.3 PNU 1002: Start Set Number

This parameter can be used to select the position data set that is started when a positioning start command is issued via control word 1. The servo drive has 256 storable standard position data sets (0...255). To limit the Start Set Number to 8 bits, the PROFINET position data set is addressed under the last index. This means that the position data set 255 itself cannot be started via the bus.

PNU	**1002**		
Name	Start Set Number		
Info	--	rw	UINT8
Value	0...255	255 (PROFINET)	

Value	Meaning
0...254	Standard Position data sets
255	Position data set PROFINET

3.4.4 PNU 1003: Position Profile Type

This parameter can be used to switch the jerk limitation of the position data records between 0 and automatic determination. With automatic determination, the filter time for the jerk limitation is always recalculated when the position record is called up, depending on the acceleration and the profile velocity. This means that the jerk is also updated when these parameters are changed and does not have to be calculated in the control system. The "call" of the position data set differs from the "start" of the position data set. The actual start of the position data set can be delayed by the corresponding option, e.g. until the end of a current positioning run. The "call" here refers to the time at which the start command is transmitted via the fieldbus.

PNU	**1003**		
Name	**Position Profile Type**		
Info	--	rw	UINT16
Value	0...1	0	

Wert	Meaning
0	Jerk for position data set = 0. This is executed once after a reset or when this parameter is written. If the jerk is subsequently changed (e.g. via the parameterisation program), the filter time remains effective as entered until this parameter is written again or a reset.
1	Automatically calculate jerk for position data set when accessed.

3.4.5 PNU 1004: Override Factor

This parameter can be used to change the speed of a positioning run at any time. By changing the override factor to 50 %, for example, the speed of a current positioning run is reduced by half. This value is always 100 % after a reset and cannot be saved permanently by saving the parameter set.

The change does not affect the acceleration. This remains unchanged. Changes to the override during a braking phase therefore have no effect on the current positioning.

PNU	**1004**		
Name	**Override Factor**		
Info	‰ (1000 = 100 %)	rw	UINT16
Value	0...2000	1000	

3.4.6 PNU 1005: Software Position Limits

This parameter number writes and reads the software position limits. These have the function of software limit switches. They are only effective in positioning mode. If the target position is beyond the software limit switches, the positioning process is not started. If parameterised accordingly, a message is triggered in this case.

Index	**1005**		
Name	**Software Position Limits**		
Type	ARRAY		1

Sub-Index	**00**		
Name	**Lower software position limit switch**		
Info	Physical unit position	rw	INT32
Value	--	-2147483648	

Sub-Index	**01**		
Name	**Upper software position limit switch**		
Info	Physical unit position	rw	INT32
Value	--	2147483647	

3.4.7 PNU 1006: Rotary Axis

These parameter numbers are used to parameterise the rotary axis mode and its limits. When the rotary axis is active, the position setpoint and actual values are limited to the rotary axis limits. The upper and lower limits "coincide". For example, for a rotary axis range of 1 revolution, 0.0 U must be set as the lower limit and 1.0 U as the upper limit.

This mode only influences the setpoint generation in position mode. The "Fixed direction of rotation Positive" mode, for example, does not prevent movements in a negative direction. The position controller continues to supply setpoints in the negative direction. Setpoint generation in other operating modes is not affected.

Index	1006	
Name	Rotary Axis	
Type	RECORD	2

Sub-Index	00		
Name	Rotary Axis Mode		
Info	--	rw	UINT8
Value	--	0	

Value	Meaning
0	Off
1	Shortest distance
2	Direction of rotation from position set
3	Fixed direction of rotation "Positive"
4	Fixed direction of rotation "Negative“

Sub-Index	01		
Name	Lower rotary axis limit		
Info	Physical unit position	rw	INT32
Value	--	-2147483648	

Sub-Index	02		
Name	Upper rotary axis limit		
Info	Physical unit position	rw	INT32
Value	--	2147483647	

3.4.8 PNU 1050: Homing Method

A number of different methods are provided for a homing run. The variant required for the application can be selected using this parameter number. There are four possible homing signals: the negative and positive limit switches, the reference switch and the (periodic) zero pulse of the angle encoder.

In addition, the servo drive can reference to the negative or positive stop without any additional signal at all. If a method for referencing is set via the object Homing Method, the following settings are determined with this:

- The reference source (neg./pos. limit switch, the reference switch, neg. / pos. stop).
- The direction and the sequence of the homing
- The method of evaluation of the zero pulse from the used angle encoder

A detailed description of the homing methods can be found in section 3.4.10 *Homing sequences* on page 39.

	Direction	Target	Reference point for zero
-18	positive	Stop	Stop
-17	negative	Stop	Stop
-2	positive	Stop	Zero pulse
-1	negative	Stop	Zero pulse
1	negative	Limit switch	Zero pulse
2	positive	Limit switch	Zero pulse
7	positive	Reference switch	Zero pulse
11	negative	Reference switch	Zero pulse
17	negative	Limit switch	Limit switch
18	positive	Limit switch	Limit switch
23	positive	Reference switch	Reference switch
27	negative	Reference switch	Reference switch
32	negative	Zero pulse	Zero pulse
33	positive	Zero pulse	Zero pulse
34		No movement	Current actual position

3.4.9 PNU 1051: Home Offset

This parameter number specifies the distance between the reference position (zero position) and the reference point of a Homing run. Positive values shift the zero point in a positive direction from the reference point. The parameter corresponds to the Offset start position parameter in the Metronix ServoCommander®.

PNU	**1050**		
Name	**Home Offset**		
Info	Physical unit position	rw	INT32
Value		0	

3.4.10 Homing sequences

3.4.10.1 Methods -17 and -18: Stop

If this method is used, the drive moves in the positive direction (-18) or negative direction (-17) until it reaches the stop. Normally, a 50% increase of the i²t value is used as the criterion for detecting the stop. Alternatively, a comparison torque value at which the stop will be considered as detected can be specified (see). The mechanical design of the stop must be such that it cannot be damaged with the parameterised maximum current. The home position refers directly to the stop. Since, in this case, the home position would be located directly at the stop, the parameter Offset start position should be used to shift the home position in a suitable manner.

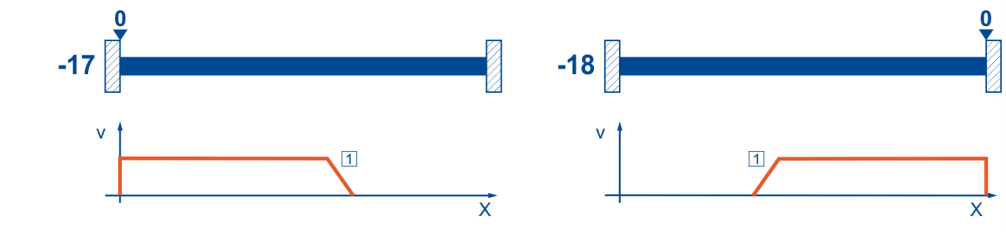

Figure 7: Homing run to the stop

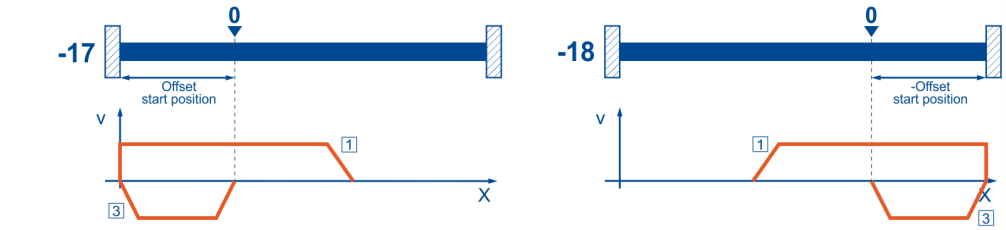

Figure 8: Use of "Offset start position"

3.4.10.2 Methods -1 and -2: stop with index pulse evaluation

These methods correspond to the methods -17 and -18. However, the home position also refers to the first index pulse of the angle encoder in the negative (-2) or positive (-1) direction as seen from the stop.

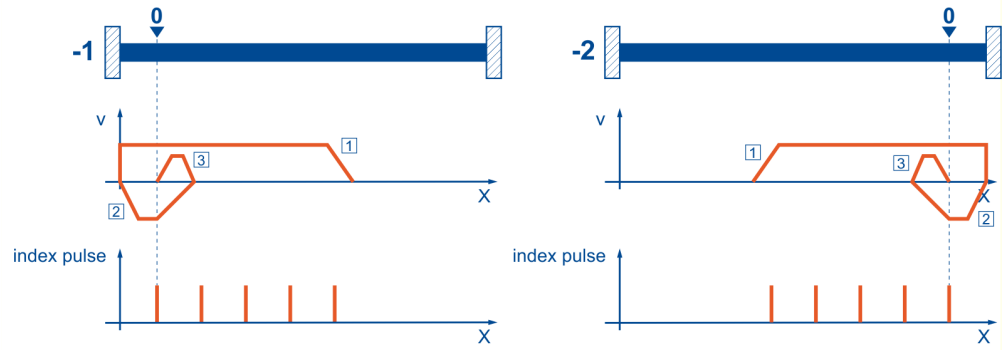

Figure 9: Homing run to the stop with index pulse evaluation

3.4.10.3 Methods 17 and 18: positive and negative limit switch

If these methods are used, the drive moves in the positive direction (18) or negative direction (17) at search speed until it reaches the limit switch. Then, the drive moves back at crawl speed and tries to find the exact position of the limit switch. The home position refers to the falling edge of the limit switch.

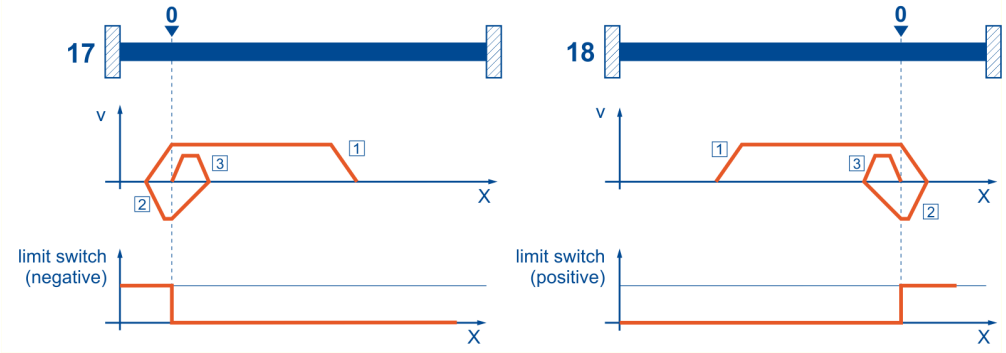

Figure 10: Homing run to the limit switch

3.4.10.4 Methods 1 and 2: positive and negative limit switch with index pulse evaluation

Like in the case of the previous method, the system tries to find the limit switch. However, in this case, the home position refers to the first index pulse of the angle encoder in the negative (1) or positive (2) direction as seen from the limit switch.

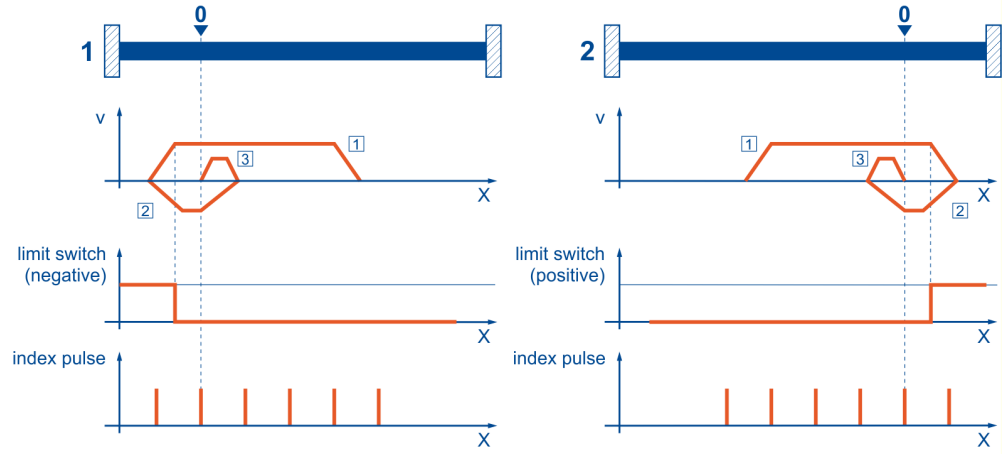

Figure 11: Homing run to the limit switch with index pulse evaluation

3.4.10.5 Methods 23 and 27: reference switch

These two methods use a reference switch which is active only over a certain part of the distance. This method is particularly suitable for rotary axis applications in which the reference switch is activated once during every rotation. If this method is used, the drive moves in the positive direction (23) or negative direction (27) at search speed until it reaches the reference switch. Then, the drive moves back at crawl speed and tries to find the exact position of the reference switch. The home position refers to the falling edge of the reference switch. If, at the beginning, the drive moves away from the reference switch, the associated limit switch causes a reversal of the direction of rotation so that the reference switch will be found.

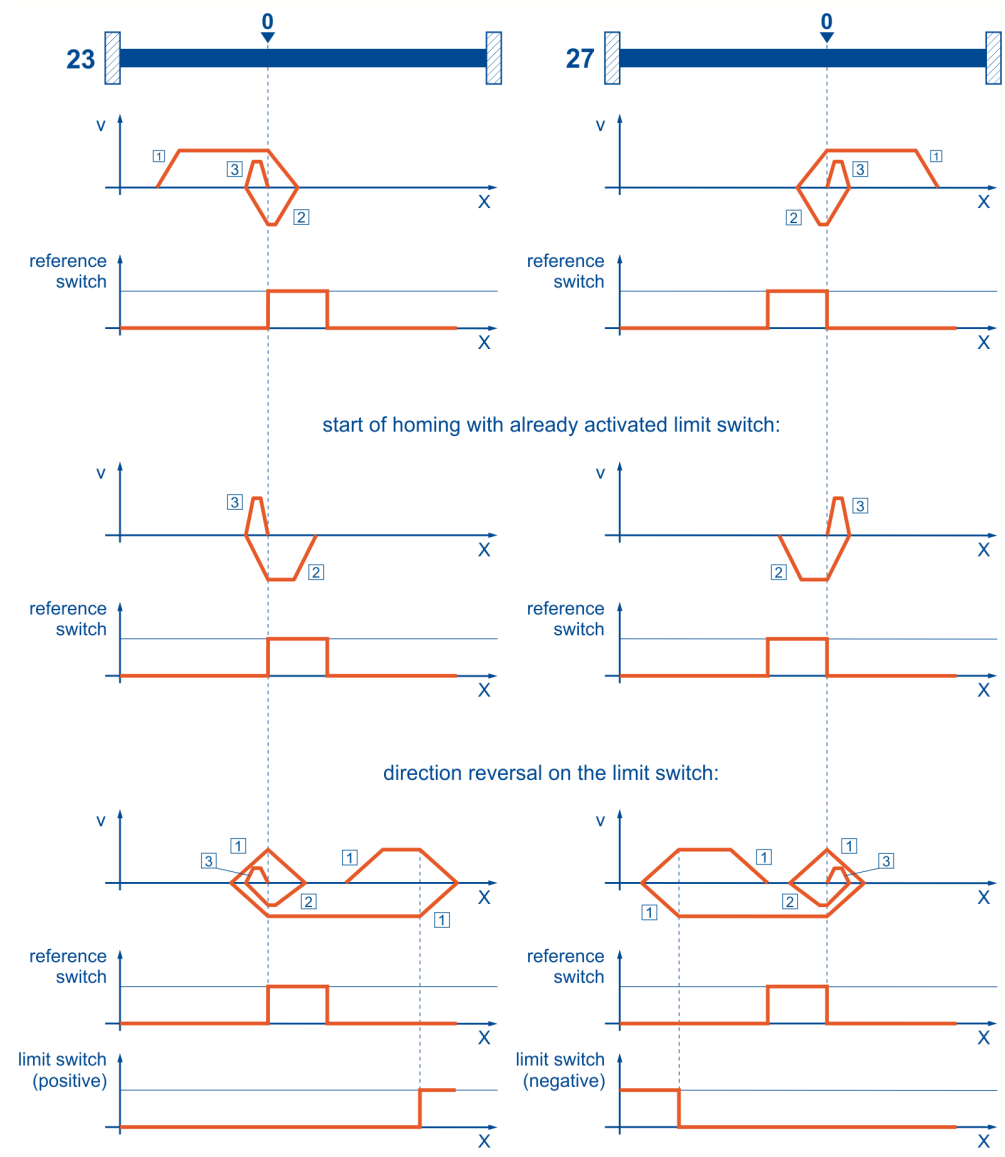

Figure 12: Homing run to the reference switch

3.4.10.6 Methods 7 and 11: reference switch and index pulse evaluation

Like methods 23 and 27, methods 7 and 11 use the reference switch. In addition, however, the home position refers to the first index pulse in the negative or positive direction as seen from the reference switch.

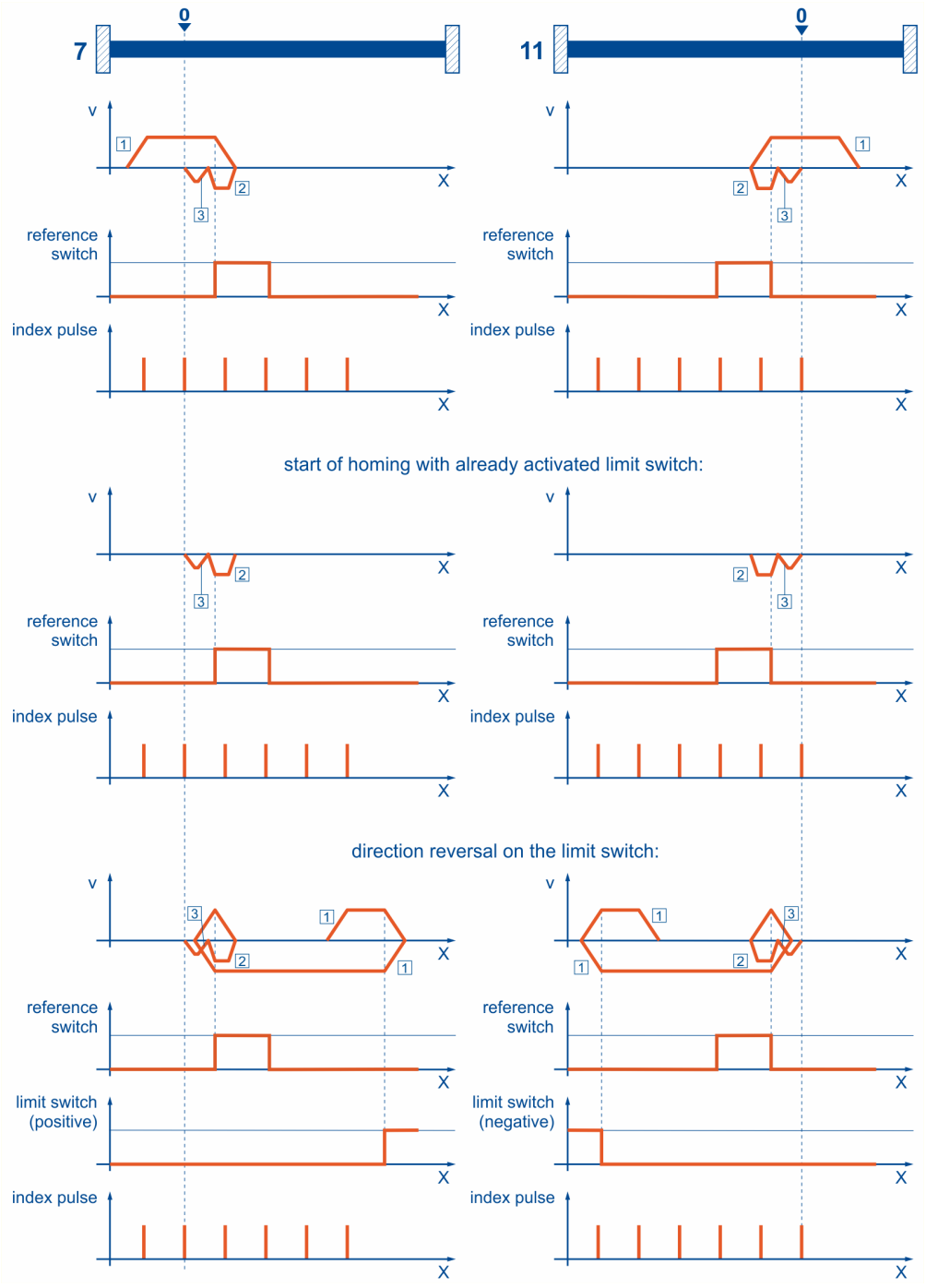

Figure 13: Homing run to the reference switch with index pulse evaluation

3.4.10.7 Methods -23 and -27: homing run (positive/negative) to the reference switch

These methods are similar to the methods 23 and 27. However, in this case, the system tries to locate the end of the range of movement, e.g. the stop or a limit switch, in a first step. It is only then that the system searches for the reference switch. As a result, several switches can be connected to the same input for the reference switch. During the homing run, the "last" switch in the search direction will be used as the reference switch. In the case of method -23, the drive moves in the positive direction first, and in the case of method -27, it moves in the negative direction first. The home position refers to the falling edge of the reference switch.

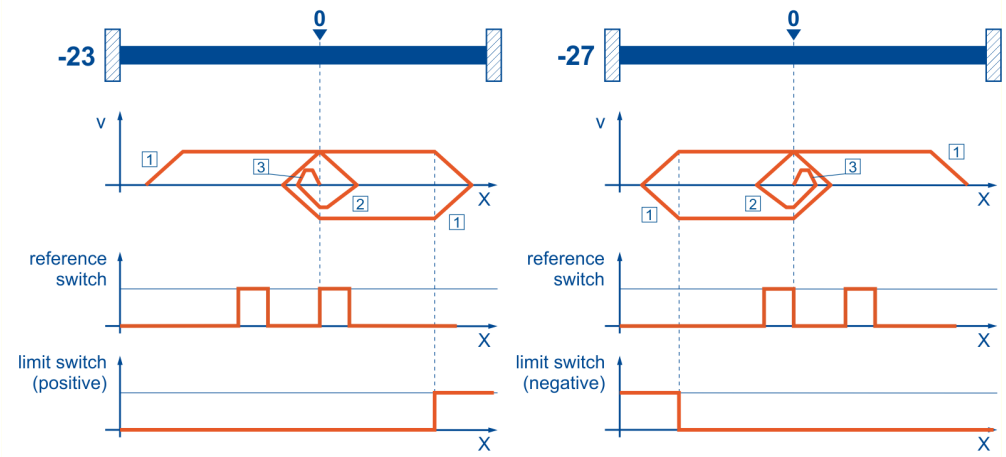

Figure 14: Reference switch with an initial movement in the positive and negative direction

3.4.10.8 Methods 32 and 33: homing to the index pulse

In the case of method 32 and method 33, the direction of the homing run is negative or positive. The home position refers to the first index pulse of the angle encoder in the search direction.

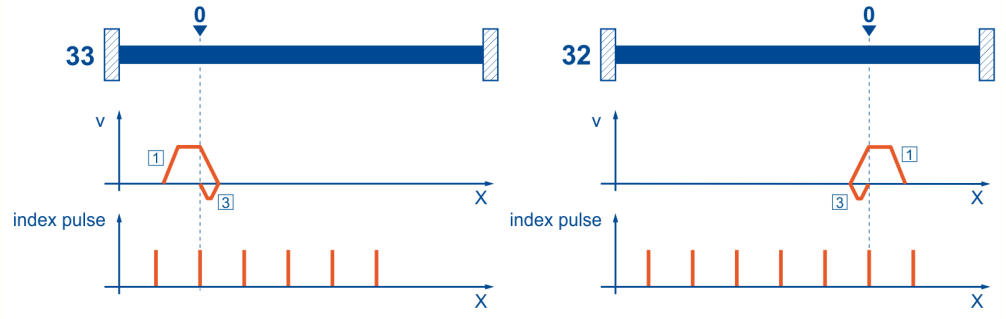

Figure 15: Index pulse with a negative (32) and positive (33) initial movement

3.4.10.9 Method 34: homing to the current position

In the case of method 34, the home position refers to the current position, i.e. the current position of the drive is set to zero.

3.4.11 PNU 1060: Thread Speed

This parameter can be used to change the set-up speed. Set-up mode can be activated or deactivated via a corresponding digital input. When set-up mode is active, reduced speed limit values apply in both speed control and positioning mode. The set-up speed is written directly in speed units. The parameterisation program Metronix ServoCommander®, on the other hand, sets this as a percentage of the limit speed. Both change the same internal variable.

PNU	**1060**		
Name	**Thread Speed (Einrichtdrehzahl)**		
Info	Physical unit velocity	rw	INT32
Value	--	3276 min^{-1}	

3.4.12 PNU 1270: Position Control Parameters

This parameter number can be used to make settings for the position controller. The dead band describes the range of the control deviation within which the position controller does not generate an output value (speed setpoint). This can be advantageous for drives with gear backlash, for example.

Index	**1270**	
Name	**Position Control Parameters**	
Type	RECORD	2

Sub-Index	**02**		
Name	**Position error tolerance window**		
Info	Physical unit position	rw	UINT32
Value	0,001 U...1 U	0,01°	

3.4.13 PNU 1271: Position Window Data

This parameter number defines the window for the „Target reached“ message.

Index	**1271**	
Name	**Position Window Data**	
Type	RECORD	0

Sub-Index	**00**		
Name	**Target Window**		
Info	Physical unit position	rw	UINT32
Value	--	10°	

3.4.14 PNU 1272: Following Error Data

This parameter number can be used to make settings concerning the range for a following error message. A position window can be defined here, beyond which, a warning is generated, for example (depending on the parameterisation of the reaction).

Index	**1272**		
Name	**Following Error Data**		
Type	RECORD		0

Sub-Index	**00**		
Name	**Following Error Window**		
Info	Physical unit position	rw	UINT32
Value	0...101 U	50°	

3.4.15 PNU 1273: Position Error Data

This parameter number can be used to modify parameters of the position window, outside of which the servo drive executes a parameterisable reaction. In addition to the following error window, a second position window can be defined here, which can, for example, lead to the servo drive being switched off with an error message (depending on the parameterisation of the reaction).

Index	**1273**		
Name	**Position Error Data**		
Type	RECORD		0

Sub-Index	**00**		
Name	**Position Error Limit**		
Info	Physical unit position	rw	UINT32
Value	0...2^{31}-1	180°	

3.5 PNUs for speed control operating mode

This section describes the parameters that are required for the speed control operating mode.

3.5.1 PNU 1010: Target Velocity

This parameter is used to set the speed setpoint. The fixed setpoint 1 is intended for this purpose. In speed control mode, this setpoint is selected automatically. In principle, the fieldbus setpoint 1 can also be saved as a fixed setpoint in the parameter set. If PROFINET communication is active in the parameter set, the fieldbus setpoint is always set to zero. This always overwrites the value saved in the parameter set.

The value 0 for Control Word 1 means that the fieldbus setpoint may not be routed to the ramp (setpoint not enabled). The setpoint selector for speed control must be set in advance when PROFINET communication is inactive. Further information on this can be found in section 5 *Operating modes* on page 68.

PNU	**1010**		
Name	**Target Velocity**		
Info	Physical unit velocity	rw	INT32
Value		0	

3.5.2 PNU 1011: Accelerations for Velocity Control

The acceleration values for the speed control operating mode can be parameterised under this parameter number. The servo drive defines 4 different acceleration ramps. As several ramps are parameterised in the same way in most applications, the following selection is available::

- Acceleration, combined for positive and negative direction of rotation
- Deceleration, combined for positive and negative direction of rotation
- Combination for acceleration and deceleration for positive and negative direction of rotation

The data is parameterised in the currently set physical unit, see section 3.1 *Physical units* on page 24.

The parameters with the sub-indices 0 and 1 allow access to the acceleration for both directions of rotation. The accelerations for both directions of rotation are therefore always written internally. When reading, however, only the current value of the acceleration ramp for the positive direction of rotation is returned. The user must ensure that reading of one value is sufficient. This can be achieved, for example, by reading the value once and then writing it back.

Index	**1011**	
Name	**Accelerations for Velocity Control**	
Type	RECORD	2

Sub-Index	**00**		
Name	**Acceleration Velocity Control**		
Info	Physical unit acceleration	rw	UINT32
Value	--	14.100 $(min^{-1})/s$	

Sub-Index	**01**		
Name	**Deceleration Velocity Control**		
Info	Physical unit acceleration	rw	UINT32
Value	--	14.100 $(min^{-1})/s$	

The All Accelerations Velocity Control parameter allows simultaneous writing of acceleration and deceleration for both directions of rotation, so that only one data value needs to be transferred. When reading, however, only the current value of the acceleration ramp for the positive direction of rotation is returned. To ensure that all four values are consistent, the parameter can be read once and then written back.

Sub-Index	**02_h**		
Name	**All Accelerations Velocity Control**		
Info	Physical unit acceleration	rw	UINT32
Value	--	14.100 $(min^{-1})/s$	

3.5.3 CAN Objekt 2415_h: current_limitation

The current_limitation object record can be used to limit the maximum current for the motor, thus allowing torque-limited speed operation, for example. The limiting torque in mA is specified via the limit_current object.

Index	**2415_h**	
Name	**current_limitation**	
Type	RECORD	02_h

Sub-Index	**02_h**			
Name	**limit_current**			
Info	mA	rw	PDO	INT32
Value	--	--		

3.6 PNUs for torque control mode

This section describes the parameters that are required for the torque control operating mode.

3.6.1 CAN Object 6071_h: target_torque

This parameter is the input value for the torque controller in torque-controlled mode. It is specified in thousandths of the nominal current.

Index	**6071_h**			
Name	**target_torque**			
Info	‰ (1000 = motor_rated torque)	rw	PDO	INT16
Value	--	--		

3.6.2 CAN Object 2416_h: speed_limitation

The speed_limitation object group can be used to limit the maximum speed of the motor in Torque operating mode, thus allowing speed-limited torque operation. This object is used to specify the limiting speed (source = fixed value) in speed_units. When using this CAN object, it must be observed that the CANopen Factor Group for speed must be set to match the Physical unit speed (see section 3.1 *Physical units* on page 24).

Index	**2416_h**	
Name	**speed_limitation**	
Type	RECORD	02_h

Sub-Index	**02_h**			
Name	**limit_speed**			
Info	speed_unit	rw	~~PDO~~	INT32
Value	--	--		

3.7 PNUs for jogging mode

This section describes the parameters used to parameterise the jogging mode.

3.7.1 PNU 1040: Symmetrical Jogging

This parameter number can be used to access the two speeds and all four acceleration values for jogging in a simplified form. In this way, less data has to be transmitted in the cyclical data telegram. When reading the value of the positive speed/acceleration is returned. The parameter for the jog speeds takes effect immediately. This means that a change also has an effect if jogging is already active.

Index	**1040**		
Name	**Symmetrical Jogging**		
Type	RECORD		1

Sub-Index	**00**		
Name	**Symmetrical Jogging Velocity**		
Info	Physical unit velocity	rw	INT32
Value	--	100 min^{-1}	

Sub-Index	**01**		
Name	**Symmetrical Jogging Accelerations**		
Info	Physical unit acceleration	rw	UINT32
Value		1.000 $(min^{-1})/s$	

3.7.2 PNU 1041: Jogging Positive

The parameters for jogging in the positive direction (TIPP0) are parameterised in more detail under this parameter number. Depending on the application, an individual parameter can also be changed in this way. The parameter for the jog speed takes effect immediately. This means that a change also has an effect if jogging in the positive direction is already active. The accelerations can be set separately for acceleration and deceleration (subindex 1 and 2) or in combination (subindex 3). When reading, subindex 3 returns the value of subindex 1.

Index	**1041**	
Name	**Jogging Positive**	
Type	RECORD	3

Sub-Index	**00**		
Name	**Jogging Velocity Positive**		
Info	Physical unit velocity	rw	INT32
Value	--	100 min^{-1}	

Sub-Index	**01**		
Name	**Jogging Acceleration Positive**		
Info	Physical unit acceleration	rw	UINT32
Value		1.000 $(min^{-1})/s$	

Sub-Index	**02**		
Name	**Jogging Deceleration Positive**		
Info	Physical unit acceleration	rw	UINT32
Value		1.000 $(min^{-1})/s$	

Sub-Index	**03**		
Name	**Symmetrical Jogging Accelerations Positive**		
Info	Physical unit acceleration	rw	UINT32
Value		1.000 $(min^{-1})/s$	

3.7.3 PNU 1042: Jogging Negative

The parameters for jogging in the negative direction (TIPP1) are parameterised in more detail under this parameter number. Depending on the application, an individual parameter can also be changed in this way. The parameter for the jog speed takes effect immediately. This means that a change also has an effect if jogging in the negative direction is already active. The accelerations can be set separately for acceleration and deceleration (subindex 1 and 2) or in combination (subindex 3). When reading, subindex 3 returns the value of subindex 1.

Index	**1042**		
Name	**Jogging Negative**		
Type	RECORD		3

Sub-Index	**00**		
Name	**Jogging Velocity Negative**		
Info	Physical unit velocity	rw	INT32
Value	--	100 min^{-1}	

Sub-Index	**01**		
Name	**Jogging Acceleration Negative**		
Info	Physical unit acceleration	rw	UINT32
Value		1.000 $(min^{-1})/s$	

Sub-Index	**02**		
Name	**Jogging Deceleration Negative**		
Info	Physical unit acceleration	rw	UINT32
Value		1.000 $(min^{-1})/s$	

Sub-Index	**03**		
Name	**Symmetrical Jogging Accelerations Negative**		
Info	Physical unit acceleration	rw	UINT32
Value		1.000 $(min^{-1})/s$	

3.8 Further PNUs

This section summarises PNUs from different areas.

3.8.1 PNU 964: Device Identification

The Device Identification parameter provides information on the connected device.

Index	**964**	
Name	**Device Identification**	
Type	RECORD	4

Sub-Index	**00**		
Name	**Manufacturer**		
Info	--	ro	UINT16
Value	277 (115_h)	277 (115_h)	

Sub-Index	**01**		
Name	**Device Type**		
Info	--	ro	UINT16
Value	see Table	--	

Value	Meaning
8202_h	BL 4102-C
8203_h	BL 4104-C
8204_h	BL 4108-C
8208_h	BL 4304-C
8209_h	BL 4308-C
8212_h	BL 4312-C
8213_h	BL 4320-C
8214_h	BL 4340-C
8215_h	BL 4360W-C

Value	Meaning
$820A_h$	BL 4104-M ETH
$820C_h$	BL 4104-D ETH
$820D_h$	BL 4840-M ETH
$820F_h$	BL 4840-D ETH
$820B_h$	BL 4104-M CAN
8210_h	BL 4104-D CAN
$820E_h$	BL 4840-M CAN
8211_h	BL 4840-D CAN

Sub-Index	02		
Name	**Version**		
Info	$MMSS_h$	ro	UINT16
Value	--	--	

Value	Meaning
M	main version
S	sub version

The PNUs with subindices 3 and 4 indicate the creation date of the PROFINET implementation. This can also be the same in different product levels or revisions if nothing has changed in the basic implementation.

Sub-Index	03		
Name	**Creation date: Year**		
Info	--	ro	UINT16
Value	--	--	

Sub-Index	04		
Name	**Creation date: Day/Month**		
Info	$DDMM_h$	ro	UINT16
Value	--	--	

Value	Meaning
D	Day
M	Month

3.8.2 PNU 971: Transfer into a non-volatile memory

This parameter number can be used to save various sets of parameters in the servo drive.

PNU	971		
Name	**Transfer into a non-volatile memory**		
Info		ro	UINT16
Value	0, 1, 256, 257, 258	0	

Value	Meaning
0	No action
1	Saving the current parameter set and all position data sets
256	Saving all position data sets without standard parameters
257	Saving in the angle encoder: Standard parameters only
258	Saving in the angle encoder: Standard parameter + absolute position

When saving the standard parameter set (writing a 1), it can be determined by reading this PNU whether the saving process has been completed. If the saving process is still running, a 1 is returned, otherwise a 0.

3.8.3 PNU 2000: PKW Access

This parameter number must be entered in a telegram in order to be able to perform asynchronous access to any drive parameter during runtime. This PNU may only be entered once in the receive and response telegrams 1..3. This must be ensured by the user accordingly.

This PNU must be entered in both the receive and response telegrams. For example, to carry out a read access, the controller writes the PNU number in bytes 1...2 and 41h in byte 0. The servo drive responds with 41h in byte 0, the PNU in bytes 1...2 and the read value in bytes 4...7 of the response telegram. In the case of a write access, the controller must set bytes 4...7 in the receive telegram accordingly.

In order to distinguish between successive accesses of the same type, byte 0 must be written with 0 in the meantime. A new access may only take place when the controller mirrors this value.

PNU	**2000**		
Name	**PKW Access**		
Info	--	rw	2 * UINT32
Value	--	0	

Byte	Meaning
0	Access type: 00_h: No access 41_h: Read access 42_h: Write access All remaining values are reserved
1...2	Parameter number (PNU) or CAN index When using CAN objects, please note that the index and subindex are given in hexadecimal form and the objects are scaled with the CANopen factor group.
3	Subindex PNU or CAN subindex
4..7	Data

3.8.4 PNU 2010: Placeholder

This parameter is required for parameterising the cyclical telegrams, see also section 6.1 *Telegram editor* on page 71.

It allows telegrams to be filled. In this way, data areas (e.g. data blocks) can be created in such a way that parameters with a length of 2 bytes or 4 bytes are located on even memory addresses.

Index	**2010**		
Name	**Placeholder**		
Type	RECORD		2

Sub-Index	**00**		
Name	**8 Bit**		
Info	--	rw	UINT8
Value	--	0	

Sub-Index	**01**		
Name	**16 Bit**		
Info	--	rw	UINT16
Value	--	0	

Sub-Index	**02**		
Name	**32 Bit**		
Info	--	rw	UINT32
Value	--	0	

3.8.5 PNU 2011: Element 0

This parameter is required for parameterising the cyclical telegrams, see also section 6.1 *Telegram editor* on page 71.

It behaves identically to the parameter with PNU 2010.2 with the difference that it is not displayed in the Metronix ServoCommander® telegram editor if it is entered at the end of a telegram. The number of entries in a telegram is always set to 10. Entries that are not required therefore receive this PNU.

Index	**2011**		
Name	**Element 0**		
Type	RECORD		0

Sub-Index	**00**		
Name	**32 Bit**		
Info	--	rw	UINT32
Value	--	0	

4 Device Control

Two data words are specified to control the servo drive. Control word 1 is used by the master to control the main device functions, while the status of the device is read back in Status word 1. Device control is based on the PROFIdrive specification, although some functions are implemented on a manufacturer-specific basis.

In the following, Control word 1 and Status word 1 are described first. Device control using these two data words is explained afterwards.

4.1 Terms used

The following terms are used to describe the device control:

Term	Explanation
State	The servo drive is in different states depending on whether the power stage is switched on or an error has occurred. The states defined under PROFINET are presented in the following chapter. Example: `SWITCHING_ON_INHIBITED`
State Transition	Like the states, it is also defined how to move from one state to another (e.g. to acknowledge an error). State transitions are triggered by the host by setting bits in the Control word 1 or internally by the servo drive if it detects an error, for example.
Command	To trigger state transitions, certain combinations of bits must be set in the controlword. Such a combination is called a command. Example: Enable Operation
State Machine	The states and state transitions together form the State Machine diagram, i.e. the overview of all states and possible transitions.

4.2 PNU 967: Control word 1

Control word 1 is used to control various device functions, e.g. controller enable via bits 0 to 3. The use of these bits is described in section 4.4 *State Machine* on page 62. Other bits are directly assigned to a specific action, e.g. the start of a positioning task. These bits can also have different meanings depending on the operating mode.

PNU	**967**		
Name	**Control word 1**		
Info	--	rw	UINT16
Value	--	0	

Control word 1 is contained at a fixed position in the receive telegrams 0..2. It is always evaluated as the last item. This means that a new target position, for example, is written first. A simultaneously transmitted command to start a positioning task therefore always refers to the data that was transmitted in the same telegram.

Bit	Value	Meaning
0	0001_h	Control of the state transitions.
1	0002_h	(These bits are evaluated together)
2	0004_h	
3	0008_h	
4	0010_h	`Enable ramp Generator / Do not reject traversing task`
5	0020_h	`Unfreeze ramp Generator / No intermediate stop`
6	0040_h	`Enable setpoint / Activate traversing task`
7	0080_h	`Fault reset`
8	0100_h	`Jog 1 on`
9	0200_h	`Jog 2 on`
10	0400_h	`Control by PLC`
11	0800_h	`Start homing`
12	1000_h	`Relative`
13	2000_h	`Change immediately`
14	4000_h	Reserved, write 0
15	8000_h	Reserved, write 0

Bit 4	Depending on the operating mode:
`Do not reject traversing task`	In Positioning mode: 0: A positioning run is cancelled 1: No effect
`Enable ramp Generator`	In Speed control mode: 0: All speed setpoints disabled ⌐↑: Acceleration to target velocity 1: All speed setpoints enabled ↓¬: Deceleration to 0
Bit 5	**Depending on the operating mode:**
`No intermediate stop`	In Positioning mode: 0: Positioning stopped ⌐↑: Acceleration (according to current position set) to profile velocity 1: Positioning permitted/running ↓¬: Interrupts the current positioning by accelerating to 0 (with deceleration from the current position set)
`Unfreeze ramp Generator`	In Speed control mode: 0: Setpoint ramp active 1: Setpoint ramp frozen The currently active setpoint is "frozen" and therefore remains active. Bit 4 (`Enable ramp generator`) has no effect in this case.
Bit 6	**Depending on the operating mode:**
`Activate traversing task`	In Positioning mode: 0: Positioning stopped ⌐↑: Start positioning if the following conditions are met: • Bit 4 = 1 (Do not reject traversing task) • Bit 5 = 1 (No intermediate stop) • No active homing 1: Positioning permitted/running ↓¬: Positioning is interrupted. Decelerating to 0. The position set to be used is defined via PNU 1002.0. If the PROFINET position set is selected, the options (e.g. relative) from Control word 1 are used, otherwise those from the respective set.
Bit 6	**Depending on the operating mode:**
`Enable setpoint`	In Speed control mode: 0: Speed setpoint disabled 1: Speed setpoint enabled When this bit is used, the setpoints are immediately set to 0 (bit is cleared) or enabled (bit is set) without ramping the setpoint up or down.

Bit 7	
`Fault reset`	⌐↑: Acknowledgement of pending errors if their cause has been eliminated
Bit 8	
`Jog 1 on`	Movement in positive direction which is determined by the position set TIPP 0 (pos): 0: Movement stopped ⌐↑: Start in positive direction 1: Movement active ↓¬: Decelerating
Bit 9	
`Jog 2 on`	Movement in positive direction which is determined by the position set TIPP 1 (neg): 0: Movement stopped ⌐↑: Start in negative direction 1: Movement active ↓¬: Decelerating
Bit 10	
`Control by PLC`	0: Control word 1 is NOT evaluated. 1: Control word 1 is evaluated.
Bit 11	
`Start homing`	⌐↑: Start homing ↓¬: Cancellation of the homing run without error The options set in the Metronix ServoCommander® are taken into account during the homing run.
Bit 12	
`Relative`	Only in positioning mode: When starting a positioning: 0: Absolute positioning 1: Relative positioning
Bit 13	
`Change immediately`	Only in positioning mode: When starting a positioning: 0: Append new positioning to current positioning 1: Interrupt current positioning and start new positioning immediately

4.3 PNU 968: Status word 1

The Status word 1 is used to indicate various device states, e.g. an active controller enable. Individual bits have a specific meaning. This is described in section 4.4 *State Machine* on page 62.

PNU	**968**		
Name	**Status word 1**		
Info	--	rw	UINT16
Value	--	0	

Bit	Value	Name
0	0001_h	State of the servo drive, see section 4.4.1 *State diagram: States* on page 63. These bits must be evaluated together.
1	0002_h	
2	0004_h	
3	0008_h	Fault present
4	0010_h	Coast Stop Not Activated (No OFF2)
5	0020_h	Quick Stop Not Activated (No OFF3)
6	0040_h	State of the servo drive, see Bits 0...2
7	0080_h	Warning Present
8	0100_h	Following Error Within Tolerance Range / Speed error Within Tolerance Range
9	0200_h	Control Requested
10	0400_h	Target Position Reached / f Or n Reached Or Exceeded
11	0800_h	Home Position Set
12	1000_h	Traversing Task Acknowledgment
13	2000_h	Drive Stopped
14	4000_h	Reserved
15	8000_h	Reserved

Bit 3	
`Fault Present`	An error is present.
Bit 4	
`Coast Stop Not Activated (No OFF2)`	No Coast Stop active. See section 4.4 *State Machine* on page 62.
Bit 5	
`Quick Stop Not Activated (No OFF3)`	No Quick Stop active. See section 4.4 *State Machine* on page 62.
Bit 7	
`Warning Present`	A warning or a setpoint lock is active due to a limit switch.
Bit 8	**Depending on the operating mode:**
`Following Error Within Tolerance Range`	In Positioning mode: No following error
`Speed error Within Tolerance Range`	In Speed control mode: The actual speed is within the parameterisable message window around the setpoint speed. The message window is parameterised in Metronix ServoCommander® under the Parameters | Signals, Speed signal tab (Speed signal).
Bit 9	
`Control by PLC`	Mirroring bit 10 from the Control word 1
Bit 10	**Depending on the operating mode:**
`Target Position Reached`	In Positioning mode: The current positioning is complete and the actual position is within the target window.
`f Or n Reached Or Exceeded`	In Speed control mode: The actual speed is equal to or greater than the speed threshold. The comparison is made taking the sign into account! The free comparison speed is parameterised in Metronix ServoCommander® under Parameters | Signals, Speed signal tab (Threshold).
Bit 11	
`Home Position Set`	The reference position is valid (by homing or due to the connected encoder system)
Bit 12	
`Traversing Task Acknowledgment`	This bit is cleared when the servo drive can process a new positioning job.
Bit 13	
`Drive Stopped`	Speed = 0 and no positioning active or intermediate stop active.

4.4 State Machine

The states are largely taken from the PROFIdrive specification. PROFIdrive distinguishes between Ramp stop and Quick stop. In both cases, the servo drive switches off the controller enable, resulting in the simplified state diagram shown in *Figure 16: State diagram of the servo drive* ergibt.

⚠ CAUTION Risk of injury due to incorrectly parameterised servo drive

An incorrectly parameterised servo drive can cause uncontrolled rotary movements and thus personal injury or damage to property.

Before switching on the power stage for the very first time, make sure that the servo drive contains the desired parameters.

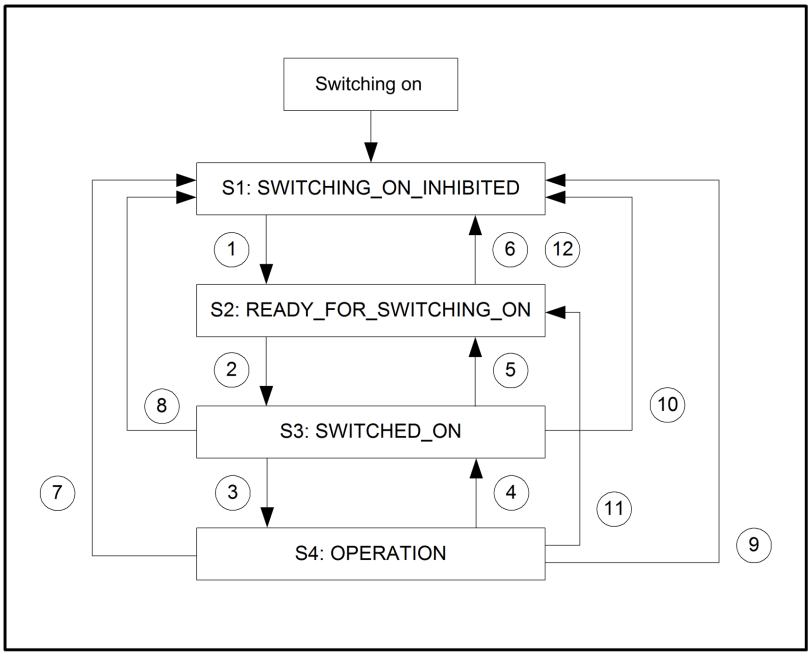

Figure 16: State diagram of the servo drive

After switching on, the servo drive initialises and finally reaches the `SWITCHING_ON_INHIBITED` state. The output stage is deactivated and the motor shaft can rotate freely. State transitions 1, 2 and 3 lead to the `OPERATION` state. This corresponds to enabling the controller. In this state, the output stage is switched on and the drive is controlled according to the set operating mode. It is therefore essential to ensure beforehand that the servo drive is correctly parameterised and that a corresponding setpoint value is zero.

State transition 4 corresponds to the deactivation of the controller enable, i.e. a motor that is still running is braked to standstill in a controlled manner in accordance with the set emergency stop ramp. State transition 7 corresponds to an immediate switch-off of the output stage, i.e. a motor that is still running would coast to a stop without control.

If an error occurs, the system ultimately branches to the `SWITCHING_ON_INHIBITED` state (regardless of the state). Depending on the severity of the error, certain actions, such as emergency braking, are carried out beforehand.

4.4.1 State diagram: States

The following table lists all states and their meaning:

Name	Meaning
Switching on	The servo drive performs a self-test. PROFINET communication is not yet working.
SWITCHING_ON_INHIBITED	The servo drive has completed its self-test. PROFINET communication is possible.
READY_FOR_SWITCHING_ON	The servo drive waits until the digital input "Controller enable" is at 24 V. (Only if Controller enable logic = "Digital input and PROFINET").
SWITCHED_ON	The power stage is switched on.
OPERATION	The motor is at voltage and is controlled according to the operating mode.

› Reading the servo drive state

In the same way as various state transitions can be triggered by combining several bits of the Control word 1, the status of the servo drive can be read out by combining different bits of the Status word 1. The following table lists the possible states of the state diagram and the corresponding bit combination with which they are displayed in the Status word 1.

State	Bit 6	Bit 2	Bit 1	Bit 0	Mask	Value
	0040_h	0004_h	0002_h	0001_h		
SWITCHING_ON_INHIBITED	0	0	0	0	0047_h	0040_h
READY_FOR_SWITCHING_ON	1	0	0	0	0047_h	0001_h
SWITCHED_ON	0	0	0	1	0047_h	0003_h
OPERATION	0	0	1	1	0047_h	0007_h

EXAMPLE

This example shows how the current status of the servo drive is determined from the Status word 1. In order to clearly determine the state of the servo drive, cleared bits in Status word 1 must also be recognised. Status word 1 must therefore be masked accordingly.

State	
SWITCHING_ON_INHIBITED	(Status word 1 & 0047_h) = 0040_h
READY_FOR_SWITCHING_ON	(Status word 1 & 0047_h) = 0001_h
SWITCHED_ON	(Status word 1 & 0047_h) = 0003_h
OPERATION	(Status word 1 & 0047_h) = 0007_h

4.4.2 Statediagram: State transitions

⚠ DANGER ⚠ Danger to life due to electric shock!

Power stage disabled means that the power semiconductors are no longer driven. If this state is entered when the motor is rotating, it coasts down unbraked. A mechanical motor brake, if present, is automatically applied.

The signal does not guarantee that the motor is in fact voltage-free.

⚠ CAUTION Uncontrolled behaviour

Power stage enabled means that the motor is controlled according to the selected operating mode. A mechanical motor brake, if present, is automatically released.

In the event of a defect or incorrect parameterisation (motor current, number of poles, resolver offset angle, etc.), the drive may behave in an uncontrolled manner.

The following table lists the required command and the required condition for all state transitions (see section 4.4 *State Machine* on page 62). The "Bits 0...3" column directly indicates the bit combination that must be set in Control word 1. An x indicates that this bit is not relevant.

Nr.	Command	Bits 0...3	Condition	Action
1	OFF	x110	Power stage enable + Controller enable + No quick stop + No coast stop	-
2	On	x111		Switching on the power stage enable
3	Enable Operation	1111		Control according to set operating mode
4	Disable Operation	0111		Removing the controller enable
5	OFF	x110		Power stage is disabled. Motor is freely rotatable
6	Coast Stop	xx0x		-
7	Coast Stop	xx0x		Power stage is disabled. Motor coasts down and is freely rotatable
8	Coast Stop	xx0x		Power stage is disabled. Motor coasts down and is freely rotatable
9	Quick Stop	x01x		Removing the controller enable
10	Quick Stop	x01x		Removing the controller enable
11	OFF	x110		Removing the controller enable

Nr.	Command	Bits 0...3	Condition	Action
12	Quick Stop	x01x		Removing the controller enable

After writing a command (by setting bits in Control word 1), you must wait until the requested status can be read back in Status word 1.

EXAMPLE

The servo drive is to be enabled. It is initially in the state SWITCHING_ON_INHIBITED.

Transition	State	Actions
1	SWITCHING_ON_INHIBITED → READY_FOR_SWITCHING_ON	(Control word 1 = 0406_h) Wait until (Status word 1 & 0047_h) = 0001_h
2	READY_FOR_SWITCHING_ON → SWITCHED_ON	(Control word 1 = 0407_h) Wait until (Status word 1 & 0047_h) = 0003_h
3	SWITCHED_ON → OPERATION	(Control word 1 = $040F_h$) Wait until (Status word 1 & 0047_h) = 0007_h

The example assumes that no other bits are set in Control word 1. Bit 10 must be set to enable control by the master.

4.5 Display of the current control/status word (Control word 1/Status word 1)

In this menu, the current values of the PROFIBUS/PROFINET control/status word are displayed.

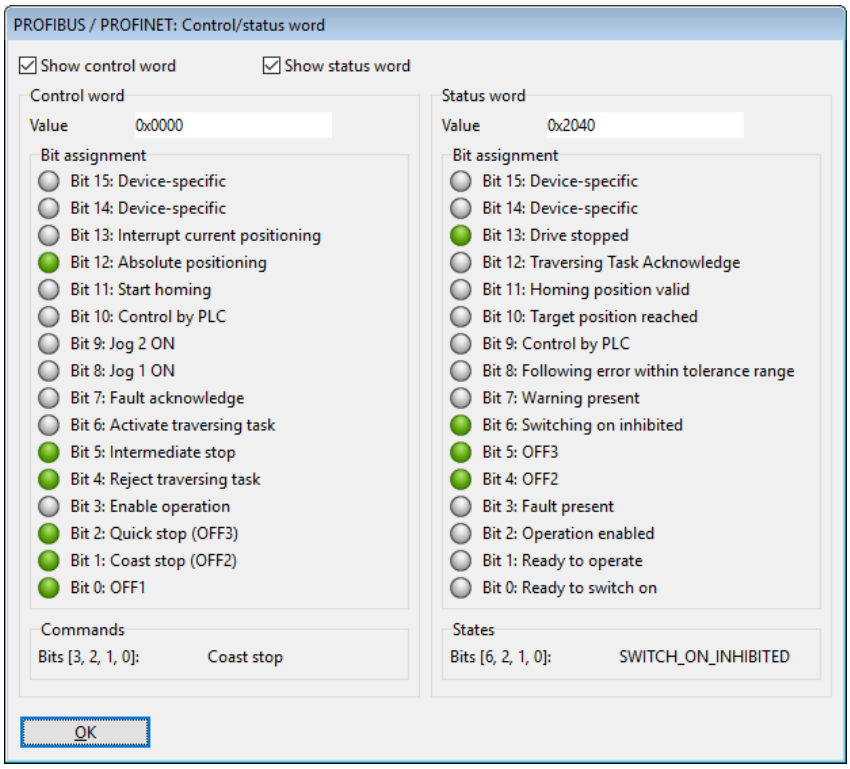

Figure 17: "PROFIBUS/PROFINET: Control-/status word" window

The display of the status or control word can be switched on or off via the respective check box Show status word or Show control word. The display of the control word cannot be switched off if the status word is not displayed and vice versa.

The current value of the control or status word is displayed both as a hexadecimal number and broken down by meaning. The individual bits are displayed as LEDs. The top LED represents bit 15 and the bottom LED bit 0 of the respective word. Since the control word and status word are assigned differently in the different operating modes, the displayed texts are adapted depending on the operating mode and bit meaning. In addition, the last command received is displayed in plain text for the control word (Commands) and the current status is displayed in plain text for the status word (States). The bits that are evaluated to determine the command/status are indicated in square brackets.

4.6 Diagnosis - Alarms

The servo drive supports PROFINET diagnosis alarms to report faults to the control unit. If a specific problem occurs in the servo drive, such as a following error, angle encoder error, etc., a diagnosis alarm is sent to the control unit. This triggers the red diagnostic LED in the control unit. For a fast analysis, the cause is specified in plain text in the PROFINET controller (e.g. in the channel diagnosis), e.g:

E08: Angle encoder: E08-6: Angle encoder communication error

The error number is made up of a main index (HH, in this example 08) and a subindex (S, in this example 6).

The main index is transmitted in the manufacturer-specific area of the channel diagnosis (ChannelErrorType) from 1000_h bis $7FFF_h$. The subindex is transmitted in the manufacturer-specific range of the extended channel diagnosis (ExtChannelErrorType) from 1000_h bis $100F_h$.

This would result in the following values for error 08-6 as an example

Error number	ChannelErrorType	ExtChannelErrorType
08-6	$HH_h + 1000_h = 1008_h$	$S_h + 1000_h = 1006_h$

5 Operating modes

5.1 Overview

The servo drives have 3 basic operating modes:

- Torque control
- Speed control
- Positioning

When using the Example Function block, the operating mode is switched via the **operation_mode** input (see section 2.5.1.2 *Inputs/outputs for all operating modes* on page 17) without any further boundary conditions having to be observed. When using your own cyclical telegrams, please note that the operating mode is permanently linked to the first two telegrams:

Operating mode	Receive telegram
Positioning	0
Speed control mode	1

Changing the receive telegram automatically changes the operating mode. This is explained in more detail in the section 6.1 *Telegram editor* on page 71. In contrast, when using Receive telegrams 2 and 3, the operating mode must be specified explicitly using the parameter *PNU 1500: Operating Mode*

As changing the operating mode can take some time, the currently valid operating mode must be determined by reading PNU 1500. It should therefore be entered in all Response telegrams (0...3).

5.2 Torque control operating mode

In torque control mode, the master directly specifies a setpoint for the torque controller. This is routed via a setpoint ramp to limit the torque change.

As there is no PNU for specifying a torque setpoint, the setpoint is specified via CAN object 6071_h (section 3.6.1 *CAN Object 6071h: target_torque* on page 48).

5.3 Speed control operating mode

In speed control mode, the controller directly specifies a setpoint for the speed controller. This is routed via a setpoint ramp to limit acceleration on speed changes. The output value of the speed ramp can be retained ("frozen").

When speed control is activated, the following setting is made in addition to the operating mode:

- Fixed setpoint 1 is activated for the adder in the setpoint selector when PROFINET communication is activated. In this case, this selector is labelled PROFINET in the Metronix ServoCommander®.
- If the setpoint is disabled by the `Enable setpoint` bit in *PNU 967: Control word 1*, then no setpoint is activated in the adder (i.e. no checkbox is selected in the Metronix ServoCommander®).

5.4 Positioning mode

In positioning mode, the servo drive can move to a target position autonomously. The following parameters (see section 3.4.2 *PNU 1001: Position Data* on page 33) are taken into account during the movement process:

- Profile velocity (Driving speed)
- End velocity (Speed at the target, usually 0)
- Acceleration and deceleration

Positioning is started on a rising edge of bit 6 (`Activate traversing task`) in *PNU 967: Control word 1*. Further bits in Control word 1 can be used to specify

- whether the positioning should be absolute or relative (bit 12, `Relative`).
- whether a running positioning task should be interrupted at the start of a new positioning task or whether the new positioning task should be appended (Bit 13, `Change immediately`).

In some applications, an sequence of positionings should be carried out without gaps, as shown on the right-hand side of the following illustration:

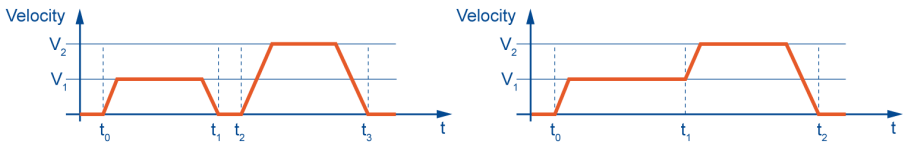

Figure 18: Single driving task (left) and sequence of driving tasks (right)

This can be achieved in two different ways:

- By interrupting the current positioning
- By starting a follow-up positioning, whereby the end velocity speed is equal to the profile velocity of the first positioning task.

The first case can be achieved by setting the `Change immediately` bit in Control word 1 when the second driving task is started. In this case, the time t_1 is not precisely defined and depends on the processing in the control system and the servo drive.

If the second positioning should start at a specific position, the end velocity (PNU 1001.2, End Velocity) of the second driving taks must be set equal to the profile velocity (PNU 1001.1, Profile Velocity) of the first driving task. The second driving task can then be initiated immediately after the start of the first (without Change Immediately). The drive then reaches the target position of the first positioning at time t_1 and then moves to target 2 without braking to zero.

The positioning mode is also used to execute the homing of the drive (see section 3.4.8 *PNU 1050: Homing Method* on page 37).

6 Cyclical communication

The data exchange of commands, setpoints and actual values between the control unit and the servo drive is based on cyclical communication. To establish cyclical communication, the telegrams in the controller and in the servo drive must be parameterised to match each other. In the following section, the configuration of the servo drive is described first and followed by the configuration of the TIA Portal.

6.1 Telegram editor

The telegram editor is used to define how the servo drive has to interpret the data it receives and sends. The operating parameter window can be opened in the Metronix ServoCommander® menu bar under:

PROFINET: Parameter/Field bus/PROFINET/Telegram editor

The data is exchanged cyclically with so-called telegrams. A distinction is made between the following two groups:

- **Receive telegrams**: Transmitted data from the master to the slave, also referred to as output data.
- **Response telegrams**: Data to be transmitted from the slave to the master, also referred to as input data.

Each telegram can have a maximum of 10 entries.

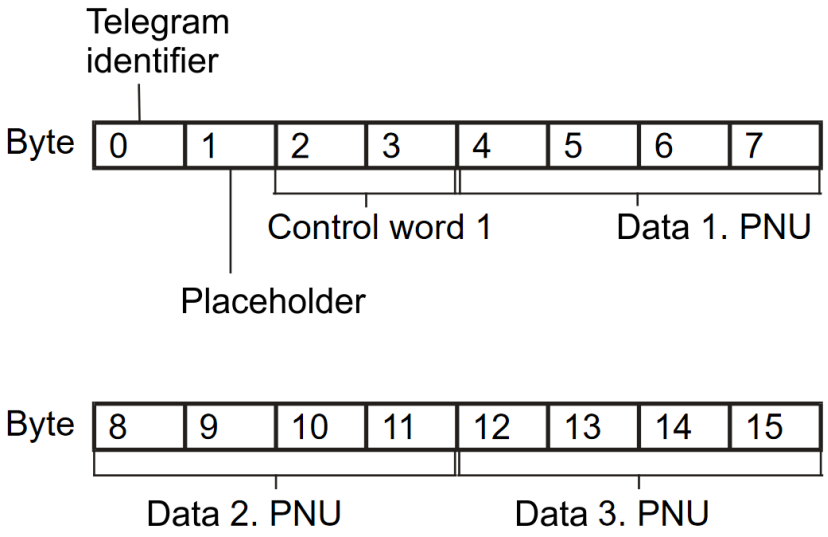

Figure 19: Example of telegram format

The illustration shows an example of a standard telegram from the master to the slave. In addition to the identifier in byte 0, this telegram type requires the PROFIdrive Control word 1 in bytes 2 and 3 for device control. The content of the following bytes can be freely configured. In this example, 3 further data are transmitted, each with a size of 4 bytes. This results in a length of 16 bytes for the entire telegram.

Data areas are created in the project of the master, e.g. data modules. The input and output data of the master and slave are stored in these data areas. When configuring, the user must specify the contents and the order of the contents as well as the size of

the two data areas in the same way on the master and slave sides. These parameterisations should be carried out before activating the communication.

› Receive telegrams

The servo drive supports 4 receive telegrams. Some of these telegrams are firmly bound to an operating mode. This makes it easier for the user to switch between different operating modes. An additional parameter for the operating mode does not have to be transmitted. The following table gives an overview of the operating mode binding of the receive telegrams:

Telegram	Operating mode
Receive telegram 0	Positioning
Receive telegram 1	Velocity control
Receive telegram 2	none (reserved for torque control)
Receive telegram 3	none (free telegram format)

As soon as a corresponding telegram identifier is read in the servo drive, the corresponding operating mode is checked and, if necessary, parameterised.

The parameter numbers must be entered for each received telegram. In this way the information about the meaning of the data in the telegram is stored in the servo drive. It should be noted that for the receive telegrams 0..2, the so-called control word is entered at address 2 (length: 2 bytes). This uniform definition facilitates the creation of applications and the use of the sample projects for SIEMENS SIMATIC S7. The other entries can be selected at will from the object directory of the parameter numbers. Only the suitability has to be considered. Pure actual value data, for example, cannot be entered in receive telegrams.

For the reception telegrams, the response telegrams must also be selected. The user can define and configure a separate response telegram for each reception telegram. In most cases, however, it is easier to use the same response telegram for all operating modes (reception telegrams 0..2). This reduces the programming effort on the part of the master. In addition, the master usually requires the same actual value data from the servo drive in all operating modes.

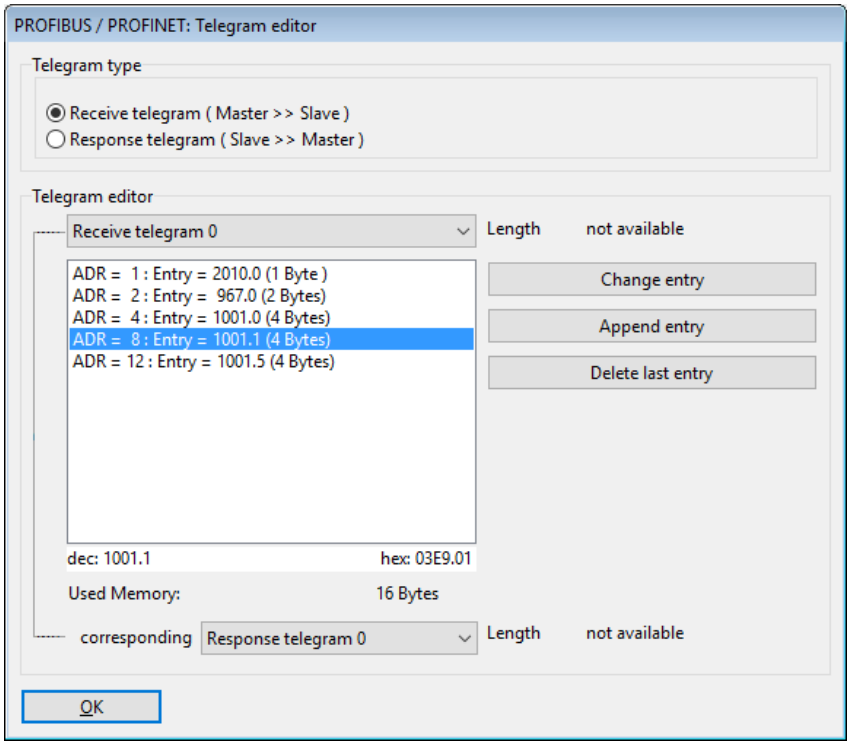

Figure 20: Composing a Receive telegram

The illustrated telegram editor window shows an example for the reception telegram 0 (positioning mode). The entries can be changed directly by marking them or successively deleted starting from the last entry. When marking an entry, an additional field appears in which the parameter number can be entered:

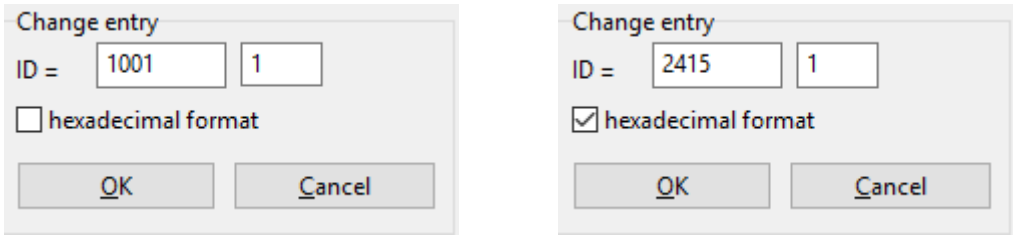

Figure 21: Input of a PNU (left) or a CAN object (right)

To enter PNUs, the number is entered in decimal form (hexadecimal format check box not selected). If objects from the CANopen object dictionary are used, the hexadecimal format checkbox must be ticked. The number listed in the CANopen manual can then be used directly. As soon as an entry is made in the ID field and confirmed with OK, a handshake with the connected servo drive takes place. It is checked whether the parameter exists and the number of bytes for this parameter is determined. Therefore, this function is also not available in the offline mode of the parameterisation program.

New telegram entries are added at the end. If communication could be established between the master and slave, additional diagnostic information is displayed via the Change entry button: The actual length of the telegram from the master to the slave configured by the master.

In the example shown above, the following parameters are transferred:

Address	Content (parameter number)	Description
0	Identifier(= 0xE0)	Fixed identifier
1	8 bit Placeholder (PNU 2010.0)	Free
2	Control word 1 (PNU 967.0)	Control word for unit control, must be fixed at this address
4	Target position (PNU 1001.0)	Target position, given in the configured physical "position" unit.
12	Acceleration (PNU 1001.5)	Combination of the values for acceleration and deceleration, given in the configured physical "acceleration " unit.

› Response telegrams

The servo drive supports 4 response telegrams, which are parameterised in the same way as the receive telegrams. The parameter numbers must be entered for each response telegram. In this way the information about the meaning of the data in the telegram is stored in the servo drive. It should be noted that for the response telegrams 0..2, the so-called status word is entered at address 2 (length: 2 bytes). This uniform definition facilitates the creation of applications and the use of the sample projects for SIEMENS SIMATIC S7. The other entries can be selected at will from the object directory of the parameter numbers. Only the suitability has to be considered. Parameters that can only be written, for example, cannot be entered in response telegrams.

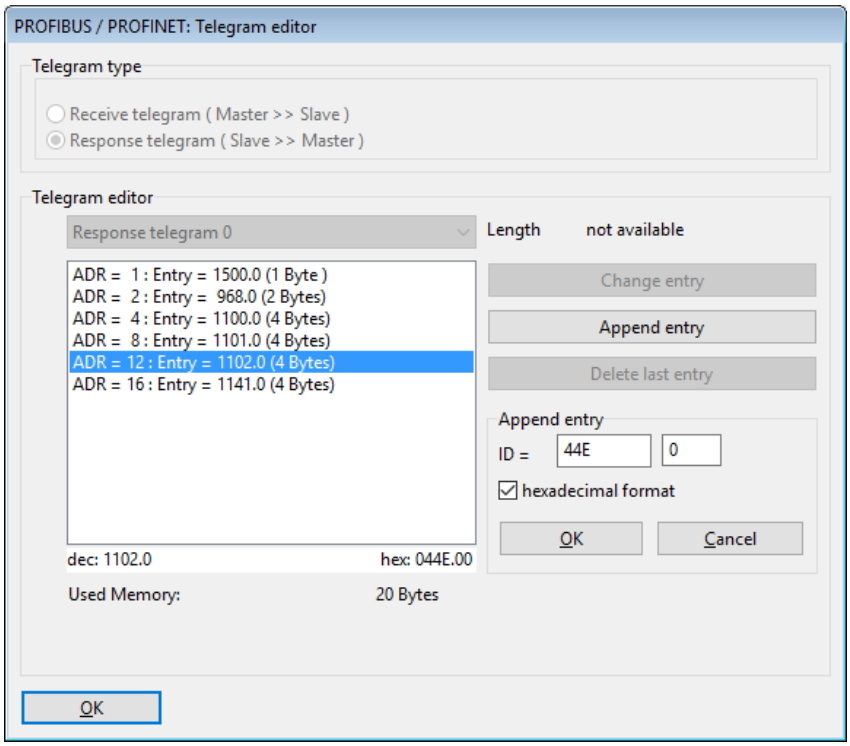

Figure 22: Composing a response telegram

The illustrated telegram editor window shows an example of response telegram 0. The entries can be changed directly by marking them or deleted successively starting from the last entry. When marking an entry, an additional field appears in which the parameter number can be entered. New telegram entries are added at the end. If communication could be established between master and slave, additional diagnostic information is displayed via the Change entry button. The actual length of the telegram from the slave to the master configured by the master is displayed.

In the example shown above, the following parameters are transmitted:

Address	Content (parameter number)	Description
0	Identifier (= 0xF0)	Fixed identifier
1	Operating mode (PNU 1500.0)	Current operating mode of the servo drive
2	Status word 1 (PNU 968.0)	Status word for unit control, must be fixed at this address
4	Actual position (PNU 1100.0)	Current actual position, given in the configured physical "position" unit
8	Actual velocity (PNU 1101.0)	Current actual speed value, given in the configured physical "speed" unit.
12	Actual active current (PNU 1102.0)	The actual active current value is read via this parameter. This is returned in relation to the rated motor current.
16	Status digital inputs (PNU 1141.0)	Current status of the digital inputs, bit assignment see description of the PNU.

6.1.1 Display of the current telegram data

The current data of the receive and response telegram is displayed in this window. Thereby mapping errors of data or mix-ups of data fields can be detected. The order of the bytes for 2-byte or 4-byte data types always corresponds to "Low Byte ... High Byte".

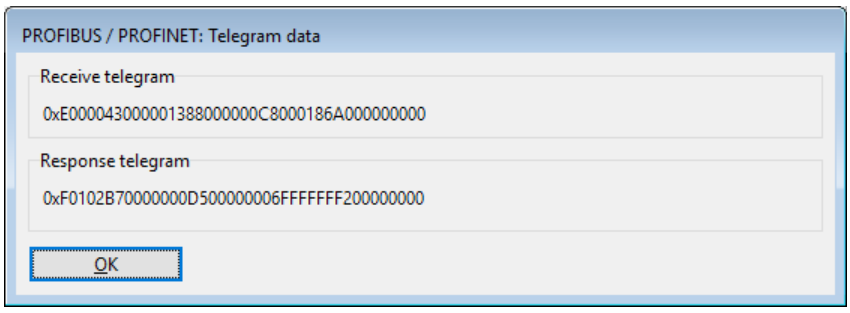

Figure 23: "PROFIBUS/PROFINET: Telegram data"

The displayed data can be interpreted as follows: The whole left byte corresponds to the telegram header (1 byte), followed by the data as specified in the telegram editor (see section Telegram Editor), i.e. entries from top to bottom in the editor correspond to values from left to right in the telegram data.

6.2 Configuration of the telegrams in the TIA Portal

The length of the input and output data must be defined in the TIA Portal to match the configuration in the servo drive. To do this, the Metronix servo drives must be selected in the device view. Then the telegram data can be configured. The IN and OUT modules are moved from the hardware catalogue to the empty lines in the device view by drag & drop.

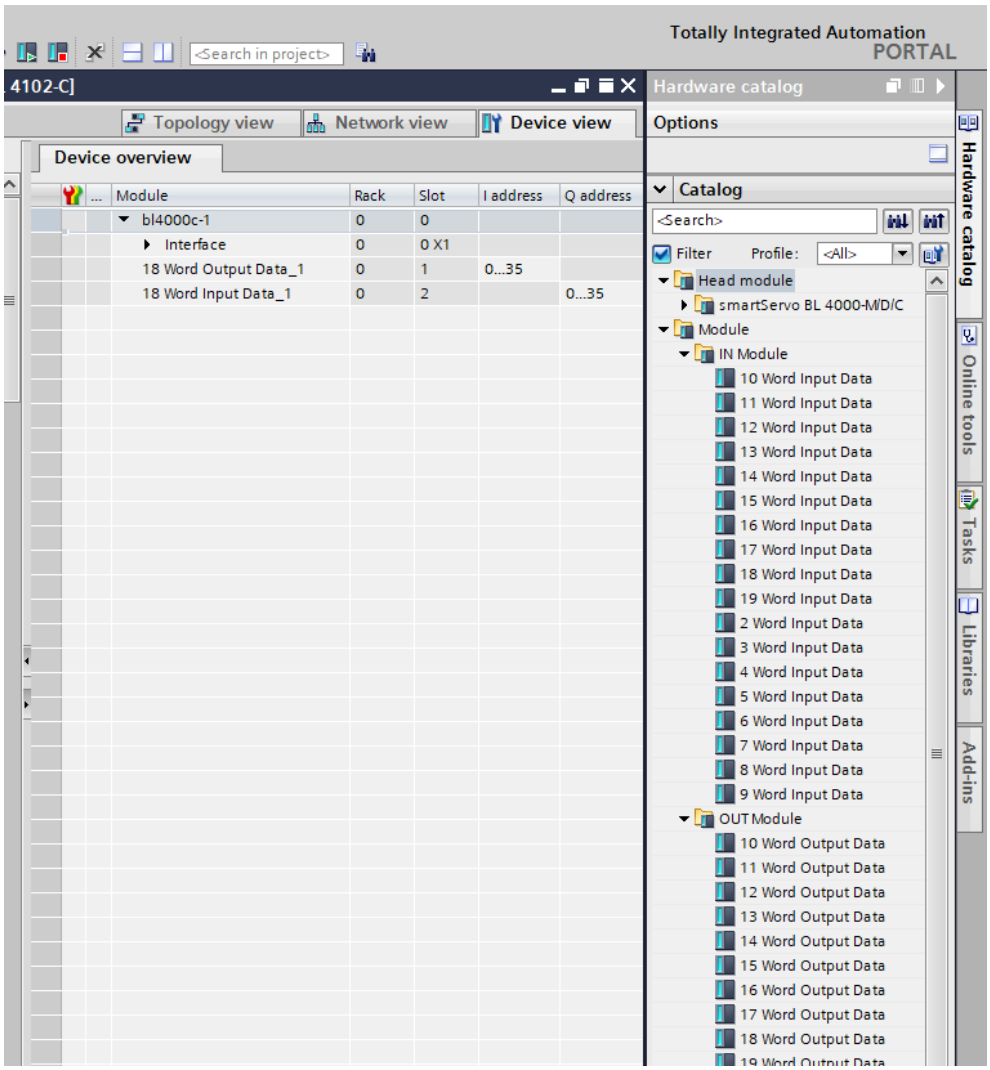

Figure 24: Configuration of the input/output data in the TIA-Portall

Unlike the configuration in the servo drive, the data's exact meaning does not need to be parameterised at this point, only its length. Modules ranging from 2 to 19 words are available for this purpose.

The receive telegram corresponds to the input data, while the response telegram corresponds to the output data.

7 Appendix

7.1 Parameterisation for the Example Function block

When using the Example Function block, the following telegram settings are required, which can be set simultaneously in the servo drive using the appropriate parameter file (section 2.5 *Integrating the servo drive into TIA portal* on page 14). All telegrams have a standardised length of 36 bytes:

› Telegram 0

Address	PNU	Description
0	$E0_h$	Fixed identifier
1	1050.0	*PNU 1050: Homing Method*
2	967.0	*PNU 967: Control word 1*
4	1001.0	*PNU 968: Status word 1*
8	1001.1	Profile Velocity, *PNU 1001: Position Data*
12	1001.2	End Velocity, *PNU 1001: Position Data*
16	1001.5	All Accelerations Positioning, *PNU 1001: Position Data*
20	1040.0	Jogging Velocity, *PNU 1040: Symmetrical Jogging*
24	2000.0	*PNU 2000: PKW Access*
32	$2415_h\_02_h$	*CAN Objekt 2415h: current_limitation*

Tabelle 1: Receive telegram 0 for Example Function block

Address	PNU	Description
0	$F0_h$	Fixed identifier
1	1500.0	*PNU 1500: Operating Mode*
2	968.0	*PNU 968: Status word 1*
4	1100.0	*PNU 1100: Position Actual Value*
8	1101.0	*PNU 1101: Velocity Actual Value*
12	6077_h	*CAN Object 6077h: torque_actual_value*
14	1141.0	*PNU 1141: Digital Inputs*
18	2010.1	*PNU 2010: Placeholder*
20	2010.2	*PNU 2010: Placeholder*
24	2000.0	*PNU 2000: PKW Access*
32	2010.2	*PNU 2010: Placeholder*

Tabelle 2: Response telegram 0 for Example Function block

› Telegram 1

Address	PNU	Description
0	$E1_h$	Fixed identifier
1	2010.0	*PNU 2010: Placeholder*
2	967.0	*PNU 967: Control word 1*
4	1010.0	*PNU 1010: Target Velocity*
8	2010.2	*PNU 2010: Placeholder*
12	2010.2	*PNU 2010: Placeholder*
16	1011.2	All Accelerations Velocity Control, *PNU 1011: Accelerations for Velocity Control*
20	1040.0	Jogging Velocity, *PNU 1040: Symmetrical Jogging*
24	2000.0	*PNU 2000: PKW Access*
32	$2415_h\_02_h$	*CAN Objekt 2415h: current_limitation*

Tabelle 3: Receive telegram 1 for Example Function block

Address	PNU	Description
0	$F1_h$	Fixed identifier
1	1500.0	*PNU 1500: Operating Mode*
2	968.0	*PNU 968: Status word 1*
4	1100.0	*PNU 1100: Position Actual Value*
8	1101.0	*PNU 1101: Velocity Actual Value*
12	6077_h	*CAN Object 6077h: torque_actual_value*
14	1141.0	*PNU 1141: Digital Inputs*
18	2010.1	*PNU 2010: Placeholder*
20	2010.2	*PNU 2010: Placeholder*
24	2000.0	*PNU 2000: PKW Access*
32	2010.2	*PNU 2010: Placeholder*

Tabelle 4: Response telegram 1 for Example Function block

› Telegram 2

Address	PNU	Description
0	$E2_h$	Fixed identifier
1	2010.2	*PNU 2010: Placeholder*
2	967.0	*PNU 967: Control word 1*
4	6071_h	*CAN Object 6071h: target_torque*
6	2010.1	*PNU 2010: Placeholder*
8	2010.2	*PNU 2010: Placeholder*
12	2010.2	*PNU 2010: Placeholder*
16	2010.2	*PNU 2010: Placeholder*
20	2010.2	*PNU 2010: Placeholder*
24	2000.0	*PNU 2000: PKW Access*
32	$2416_h\_02_h$	*CAN Object 2416h: speed_limitation*

Tabelle 5: Receive telegram 2 for Example Function block

Address	PNU	Description
0	$F2_h$	Fixed identifier
1	1500.0	*PNU 1500: Operating Mode*
2	968.0	*PNU 968: Status word 1*
4	1100.0	*PNU 1100: Position Actual Value*
8	1101.0	*PNU 1101: Velocity Actual Value*
12	6077_h	*CAN Object 6077h: torque_actual_value*
14	1141.0	*PNU 1141: Digital Inputs*
18	2010.1	*PNU 2010: Placeholder*
20	2010.2	*PNU 2010: Placeholder*
24	2000.0	*PNU 2000: PKW Access*
32	2010.2	*PNU 2010: Placeholder*

Tabelle 6: Response telegram 2 for Example Function block