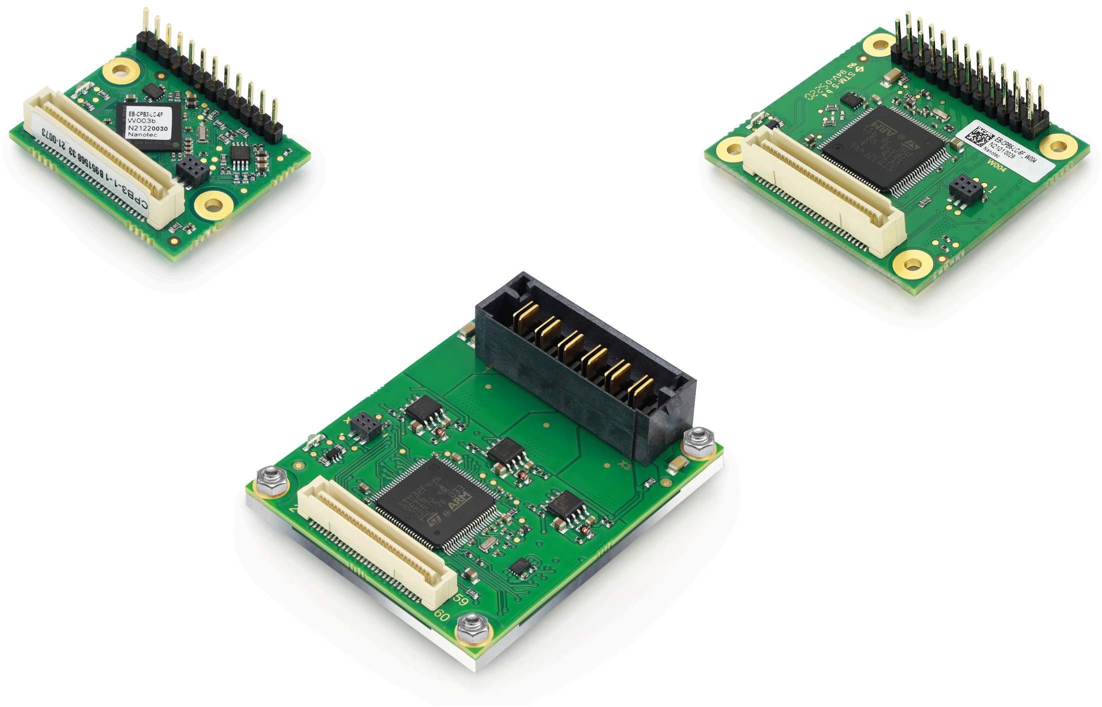
Technical Manual CPB

Fieldbus: Modbus TCP

For use with the following variants:

CPB3-1-4, CPB3-2-4, CPB6-1-4, CPB6-2-4, CPB15-4



Valid with firmware version FIR-v2213

Technical Manual Version: 1.0.1

Contents

1 Introduction

The products of the *CPB* series are motor controllers in plug#in module format for integration in your own development. The *CPB3-...* and *CPB6-...* variants can control both BLDC motors and stepper motors; the *CPB15* variant is suitable for BLDC motors only.

This manual describes the integration of the *CPB* in your motherboard and the functions of the controller. It also shows how you can address and program the controller via the communication interface.

You can find further information on the product on us.nanotec.com.

1.1 Version information

CPB3

Manual version	Date	Changes	Firmware version	Hardware version
1.0.0	07/2022	First edition	FIR-v2213	W003
1.0.1	09/2022	Correction of the routing example in the chapter Digital inputs.	FIR-v2213	W003

CPB6

Manual version	Date	Changes	Firmware version	Hardware version
1.0.0	07/2022	First edition	FIR-v2213	W004
1.0.1	09/2022	Correction of the routing example in the chapter Digital inputs.	FIR-v2213	W004

CPB15

Manual version	Date	Changes	Firmware version	Hardware version
1.0.0	07/2022	First edition	FIR-v2213	W005
1.0.1	09/2022	Correction of the routing example in the chapter Digital inputs.	FIR-v2213	W005

1.2 Copyright, marking and contact

Nanotec Electronic GmbH & Co. KG

Kapellenstraße 6

85622 Feldkirchen

Germany

Phone: +49 89 900 686-0

Fax: +49 (89) 900 686-50

us.nanotec.com

Microsoft® Windows® 98/NT/ME/2000/XP/7/10 are registered trademarks of the Microsoft Corporation.

1.3 Intended use

The *CPB* serves to control stepper motors or BLDC motors and is used as a component in drive systems in a wide range of industrial applications.

The controller must be connected to motors via a suitable motherboard. The system boundary of the *CPB* ends at the connectors.

Use the product as intended within the limits defined in the technical data (in particular, see Electrical properties and technical data) and the approved Environmental conditions.

Under no circumstances may this Nanotec product be integrated as a safety component in a product or system. All products containing a component manufactured by Nanotec must, upon delivery to the end user, be provided with corresponding warning notices including instructions for safe use and safe operation. All warning notices provided by Nanotec must be passed on directly to the end user.

1.4 Target group and qualification

The product and this documentation are directed towards technically trained specialists staff such as:

- Development engineers
- Plant engineers
- Installers/service personnel
- Application engineers

Only specialists may install, program and commission the product. Specialist staff are persons who

- have appropriate training and experience in working with motors and their control,
- are familiar with and understand the content of this technical manual,
- know the applicable regulations.

1.5 Warranty and disclaimer

Nanotec is not liable for damage and malfunction from installation errors, failure to observe this document, or improper repair. Responsible for the selection, operation, use of our products is the plant engineer, operator and user. Nanotec accepts no liability for product integration in the end system. The general terms and conditions at www.nanotec.com apply (customers of Nanotec Electronic USA please see us.nanotec.com). **Note:** Product modification / alteration is illicit.

1.6 EU directives for product safety

The following EU directives were observed:

- RoHS directive (2011/65/EU, 2015/863/EU)

1.7 Other applicable regulations

In addition to this technical manual, the following regulations are to be observed:

- Accident-prevention regulations
- Local regulations on occupational safety

1.8 Used icons

All notices are in the same format. The degree of the hazard is divided into the following classes.

CAUTION!

The CAUTION notice indicates a possibly dangerous situation.

Failure to observe the notice **may** result in moderately severe injuries.

► Describes how you can avoid the dangerous situation.

NOTICE

Indicates a possible incorrect operation of the product.

Failure to observe the notice may result in damage to this or other products.

► Describes how you can avoid the incorrect operation.

TIP

Shows a tip for the application or task.

1.9 Emphasis in the text

The following conventions are used in the document:

Underlined text indicates cross references and hyperlinks:

- The following bits in object 6041$_h$ (statusword) have a special function:
- A list of available system calls can be found in chapter NanoJ functions in the NanoJ program.

Text set in *italics* marks named objects:

- Read the *installation manual*.
- Use the *Plug & Drive Studio* software to perform the auto setup.
- For software: You can find the corresponding information in the *Operation* tab.
- For hardware: Use the *ON/OFF* switch to switch the device on.

A text set in `Courier` marks a code section or programming command:

- The line with the `od_write(0x6040, 0x00, 5 );` command has no effect.
- The NMT message is structured as follows: `000 | 81 2A`

A text in "quotation marks" marks user input:

- Start the NanoJ program by writing object 2300$_h$, bit 0 = "1".
- If a holding torque is already needed in this state, the value "1" must be written in 3212$_h$:01$_h$.

1.10 Numerical values

Numerical values are generally specified in decimal notation. The use of hexadecimal notation is indicated by a subscript *h* at the end of the number.

The objects in the object dictionary are written with index and subindex as follows: `<Index>:<Subindex>`

Both the index as well as the subindex are specified in hexadecimal notation. If no subindex is listed, the subindex is 00$_h$.

Example: Subindex 5 of object 1003$_h$ is addressed with `1003`$_h$`:05`$_h$, subindex 00 of object 6040$_h$ with `6040`$_h$.

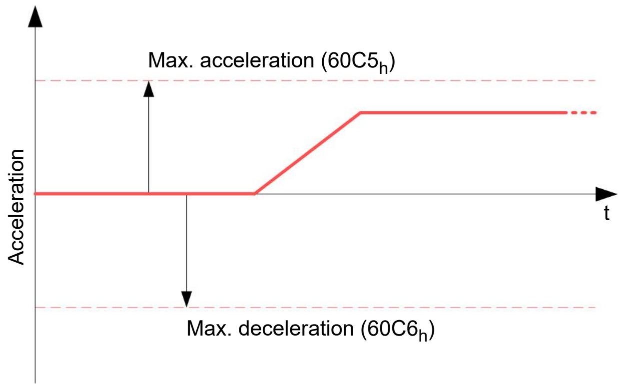
1.11 Bits

The numbering of individual bits in an object always begins with the LSB (bit number 0). See the following figure, which uses data type *UNSIGNED8* as an example.

	MSB							LSB	
Bit Nummer	7	6	5	4	3	2	1	0	
Bits	0	1	0	1	0	1	0	1	≙ 55_{hex} ≙ 85_{dec}

1.12 Counting direction (arrows)

In figures, the counting direction is always in the direction of an arrow. Objects `60C5`$_h$ and `60C6`$_h$ depicted as examples in the following figure are both specified as positive.



2 Safety and warning notices

NOTICE

Damage to the controller!

Changing the wiring during operation may damage the controller.

► Only change the wiring in a de-energized state. After switching off, wait until the capacitors have discharged.

NOTICE

Damage to the controller due to excitation voltage of the motor!

Voltage peaks during operation may damage the controller.

► Install suitable circuits (e. g., charging capacitor) that reduce voltage peaks.

NOTICE

Damage to the electronics through improper handling of ESD-sensitive components!

The device contains components that are sensitive to electrostatic discharge. Improper handling can damage the device.

► Observe the basic principles of ESD protection when handling the device.

NOTICE

Damage to the electronics if the supply voltage is connected with reversed polarity!

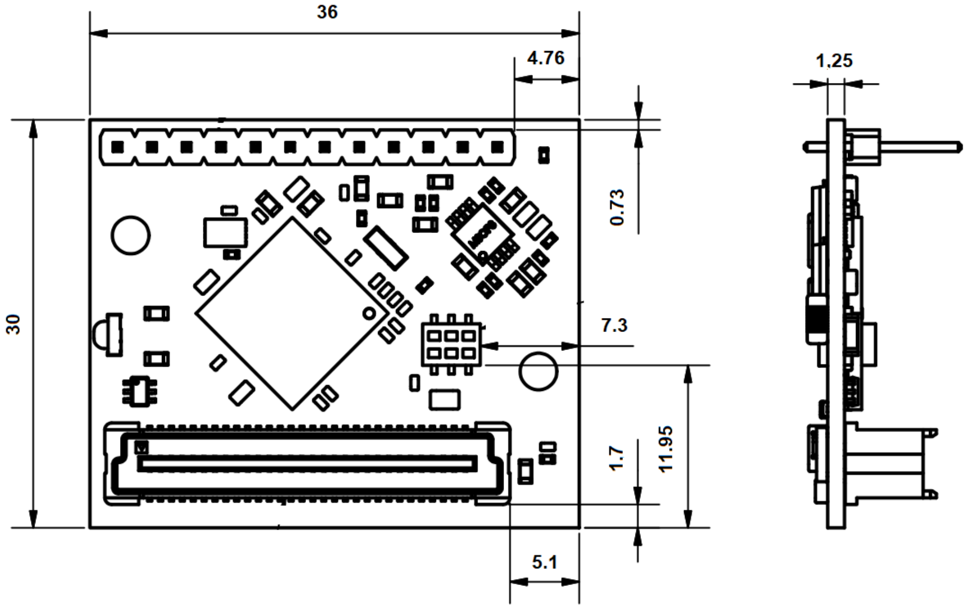
► Install a line protection device (fuse) in the supply line.

3 Technical details and pin assignment

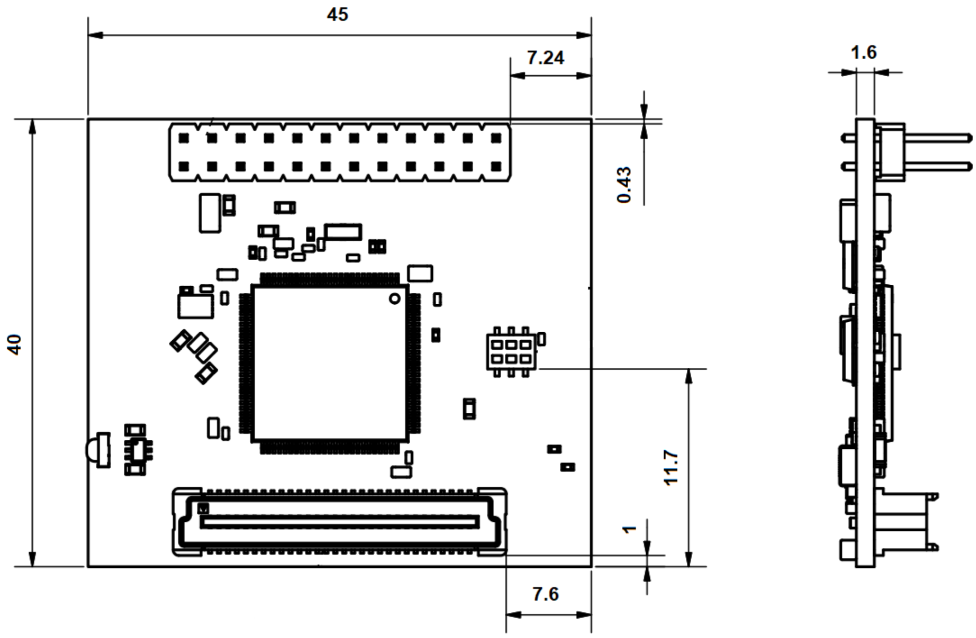
3.1 Dimensioned drawings

All dimensions are in millimeters.

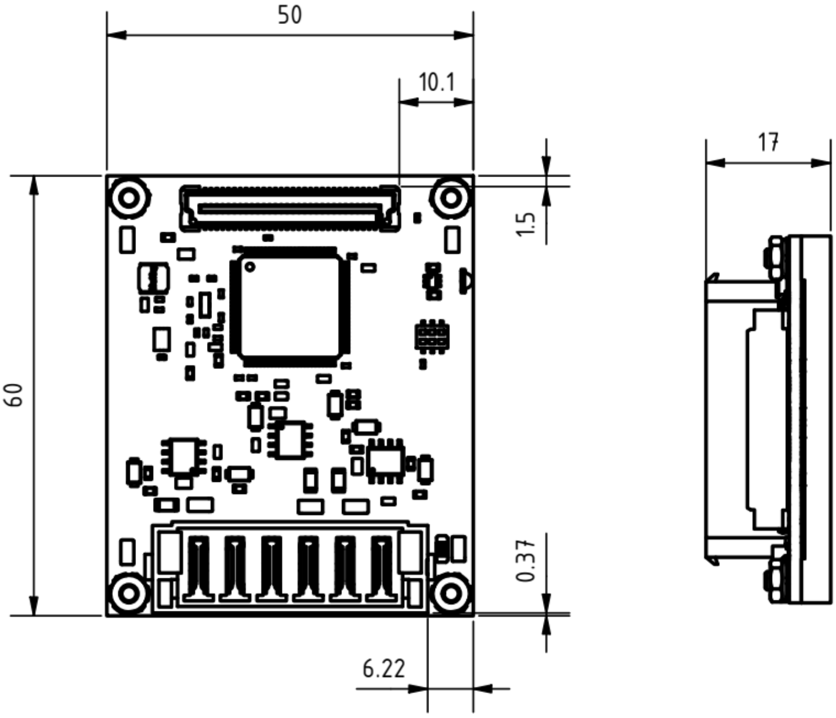
CPB3



CPB6



CPB15



3.2 Environmental conditions

Environmental condition	Value
Protection class	No IP protection
Ambient temperature (operation)	-10 … +40°C
Air humidity (non-condensing)	0 … 95 %
Max. Altitude of site above *sea level*	2000 m (drop in performance above 1000 m: -1%/100 m)
Ambient temperature (storage)	-25 … +85°C

3.3 Electrical properties and technical data

Property	Description / value
Operating voltage	12 … 57.6 V DC
Rated current @40°C	*CPB3-...*: 3 A_{rms}
	CPB6-...: 6 A_{rms} (for temperature derating, see Overtemperature protection)
	CPB15-...: ■ 15 A_{rms} (@25°C) ■ 10 A_{rms} (@40°C)

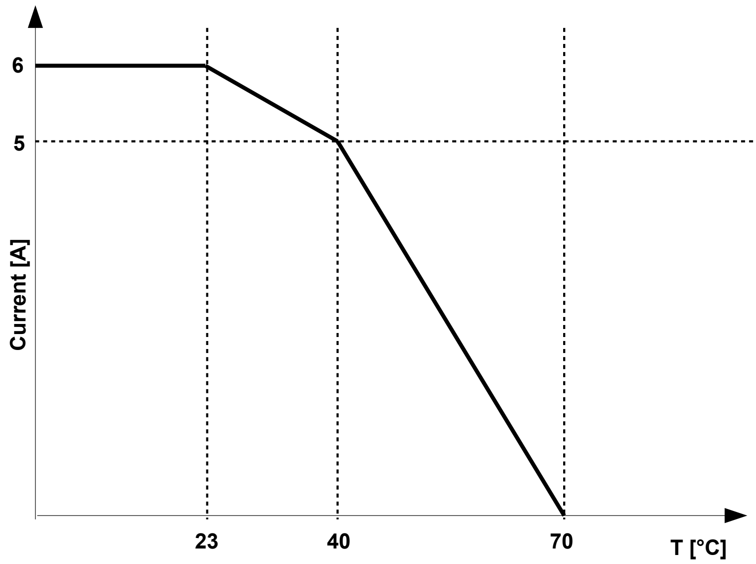
Property	Description / value
Peak current @40°C	■ *CPB3-1-...*: 3 A_{rms} ■ *CPB3-2-...*: 9 A_{rms} (for max. 5 seconds)
	■ *CPB6-1-...*: 6 A_{rms} ■ *CPB6-2-...*: 18 A_{rms} (for max. 5 seconds)
	CPB15-...: 45 A_{rms} (for max. 5 seconds)
Commutation	*CPB3-...*, *CPB6-...*: Stepper motor *open-loop*, stepper motor *closed-loop* with encoder, BLDC sine commutated via Hall sensor, BLDC sine commutated via encoder *CPB15-…*: BLDC sine commutated via Hall sensor, BLDC sine commutated via encoder
	Note: External wiring is required for encoder and Hall sensor!
Operating modes	*Profile Position Mode*, *Profile Velocity Mode*, *Profile Torque Mode*, *Velocity Mode*, *Homing Mode*, *Interpolated Position Mode*, *Cyclic Sync Position Mode*, *Cyclic Sync Velocity Mode*, *Cyclic Synchronous Torque Mode*, *Clock-Direction Mode*
Set value setting / programming	*Clock-direction*, *analog*, *NanoJ program*
Interfaces	USB, Ethernet (Modbus TCP, REST) **Note:** External wiring is required for USB and Ethernet!
Encoder/Hall	1x SSI encoder, 1x Hall sensor, 1x incremental encoder (second SSI encoder can be configured instead) **Note:** External wiring is required for encoder and Hall sensor!
I/O	12x general I/Os (one of which can be used as output for external brake), 2x analog inputs, 1x output for external ballast circuit
Overtemperature	Protection circuit at temperature > 80°C
Charging capacitor	For each ampere of rated current on the motor, Nanotec recommends a capacitance of approx. 1000 µF.

3.4 Overtemperature protection

Above a temperature of approx. 80 °C on the power board the power part of the controller switches off and the error bit is set (see objects 1001_h and 1003_h). After cooling down and confirming the error (see table for the controlword, "Fault reset"), the controller again functions normally.

Temperature-dependent power reduction for the CPB6

The following diagram shows the permissible continuous current as a function of the ambient temperature:



NOTICE

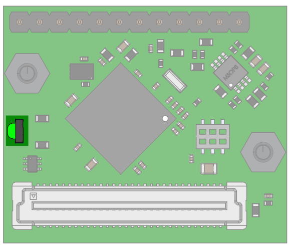
Aside from the motor, the exact temperature behavior is also dependent on the flange connection and the heat transfer there as well as on the convection in the application. For this reason, we recommend always performing an endurance test in the actual environment for applications in which current level and ambient temperature pose a problem.

3.5 LED signaling

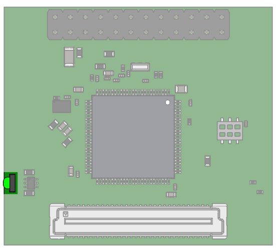
3.5.1 Power LED

The power LED indicates the current status.

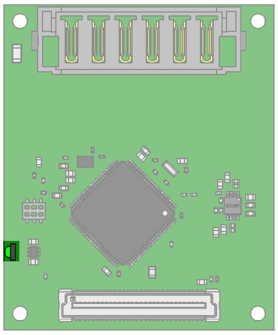
CPB3



CPB6

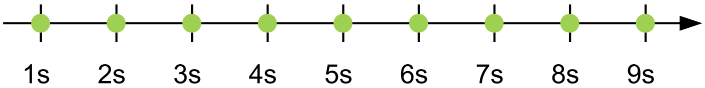


CPB15



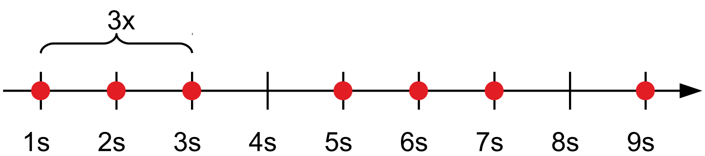
3.5.1.1 Normal operation

In normal operation, the green power LED flashes briefly once per second.



3.5.1.2 Case of an error

If an error has occurred, the LED turns red and signals an error number.In the following figure, the error number 3 is signaled.



The following table shows the meaning of the error numbers.

Flash rate	Error
1	General
2	Voltage
3	Temperature
4	Overcurrent
5	Controller
6	Watchdog-Reset

NOTICE

For each error that occurs, a more precise error code is stored in object 1003_h.

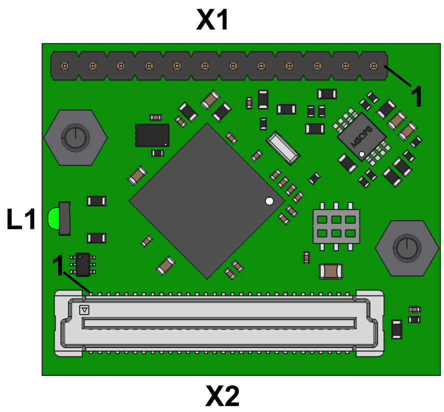
TIP

You can switch off the power LEDs with 3250_h:09_h.

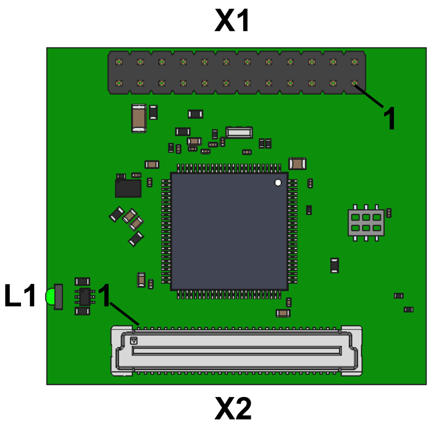
3.6 Pin assignment

3.6.1 Overview

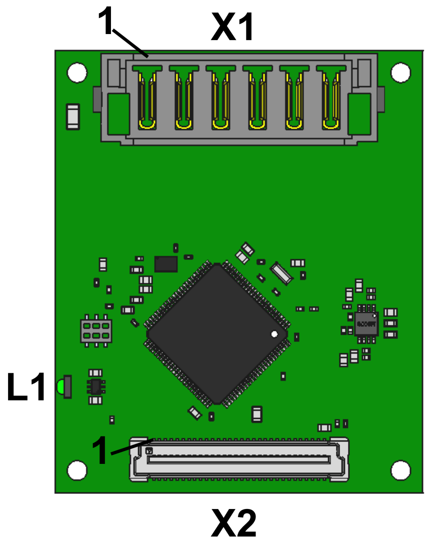
CPB3



CPB6



CPB15



3.6.2 X1 — voltage supply and motor

CPB3

Connector: Würth Elektronik 61301211121

Pin	Name	Description/function
1	GND	
2	BN_OUT	B\ (stepper motor)
3	BN_OUT	B\ (stepper motor)
4	B_OUT	B (stepper motor) or W (BLDC)
5	B_OUT	B (stepper motor) or W (BLDC)
6	AN_OUT	A\ (stepper motor) or V (BLDC)
7	AN_OUT	A\ (stepper motor) or V (BLDC)
8	A_OUT	A (stepper motor) or U (BLDC)
9	A_OUT	A (stepper motor) or U (BLDC)
10	+UB	12 - 57.6 V DC
11	+UB	12 - 57.6 V DC
12	GND	

CPB6

Connector: Würth Elektronik 61302421121

Pin	Name	Description/function
1	GND	
2	GND	
3	BN_OUT	B\ (stepper motor)
4	BN_OUT	B\ (stepper motor)

Pin	Name	Description/function
5	BN_OUT	B\ (stepper motor)
6	BN_OUT	B\ (stepper motor)
7	B_OUT	B (stepper motor) or W (BLDC)
8	B_OUT	B (stepper motor) or W (BLDC)
9	B_OUT	B (stepper motor) or W (BLDC)
10	B_OUT	B (stepper motor) or W (BLDC)
11	AN_OUT	A\ (stepper motor) or V (BLDC)
12	AN_OUT	A\ (stepper motor) or V (BLDC)
13	AN_OUT	A\ (stepper motor) or V (BLDC)
14	AN_OUT	A\ (stepper motor) or V (BLDC)
15	A_OUT	A (stepper motor) or U (BLDC)
16	A_OUT	A (stepper motor) or U (BLDC)
17	A_OUT	A (stepper motor) or U (BLDC)
18	A_OUT	A (stepper motor) or U (BLDC)
19	+UB	12 - 57.6 V DC
20	+UB	12 - 57.6 V DC
21	+UB	12 - 57.6 V DC
22	+UB	12 - 57.6 V DC
23	GND	
24	GND	

CP15

Connector: Samtec MPT-06-6.30-01-L-V

Pin	Name	Description/function
1	+UB	12 - 57.6 V DC
2	GND	
3	U_OUT	U (BLDC)
4	V_OUT	V (BLDC)
5	W_OUT	W (BLDC)
6	GND	

NOTICE

EMC: For a DC power supply line longer than 30 m or when using the motor on a DC bus, additional interference-suppression and protection measures are necessary.

► An EMI filter is to be inserted in the DC supply line as close as possible to the controller/ motor.

► Long data or supply lines are to be routed through ferrites.

3.6.3 X2 – inputs and outputs

Connectors:

- CPB3: Amphenol 10144518-063802LF

- CPB6: Amphenol 10144518-063802LF
- CPB15: Amphenol 10144518-062802LF

NOTICE

Some pins are freely configurable and support alternative functions. You can find details in chapter Defining input and output assignments.

Pin	Name	Description/function
1	GND	
2	GND	
3	ETH_RXD1	*Ethernet receive data* 1
4	ANA1	Analog input 1: 0…3.2 V
5	ETH_RXD0	*Ethernet receive data* 0
6	ANA2	Analog input 2: 0…3.2 V
7	ETH_CRS_DV	*Ethernet Carrier Sense / Data Valid*
8	Spare	not connected
9	ETH_REF_CLK	*RMII Reference Clock output*
10	ETH_MDC	not connected
11	ETH_TX_EN	*Ethernet transmit enable*
12	ETH_MDIO	not connected
13	ETH_TXD0	*Ethernet receive data* 0
14	INTRP	*Ethernet interrupt*, active low
15	ETH_TXD1	*Ethernet receive data* 1
16	GND	
17	GND	
18	H1	Hall sensor 1
19	DIO11	General I/O
20	H2	Hall sensor 2
21	DIO13	General I/O
22	H3	Hall sensor 3
23	ENC1B	Encoder 1, B
24	ENC1A	Encoder 1, A
25	SSI1_MISO	SSI encoder 1, Data IN
26	ENC1I	Encoder 1, index
27	SSI1_CLK	SSI encoder 1, clock
28	reserved	reserved
29	DIO14	General I/O
30	reserved	reserved
31	reserved	not connected
32	reserved	not connected
33	reserved	reserved
34	reserved	not connected
35	VBUS	USB VBUS (power supply)
36	DP	USB DP (Data+)

Pin	Name	Description/function
37	GND	
38	DM	USB DM (Data-)
39	DIO1	General I/O
40	GND	
41	DIO3	General I/O
42	DIO2	General I/O
43	DIO5	General I/O
44	DIO4	General I/O
45	DIO7	General I/O
46	DIO6	General I/O
47	DIO9	General I/O
48	BRAKE	Control of the external brake
49	GND	
50	DIO10	General I/O
51	uC Reset	System function, reserved
52	Ballast	For controlling the external ballast circuit
53	uC Supply	Voltage supply 3.3 V Must be provided by motherboard, limits: 3.2…3.4 V DC, min. 140 mA
54	uC Supply	Voltage supply 3.3 V Must be provided by motherboard, limits: 3.2…3.4 V DC, min. 140 mA
55	Driver Supply	Voltage supply 10.5 V Must be provided by motherboard, limits: 9.6…14 V DC, min. 70 mA
56	DIO12	General I/O
57	Ub_Logic	Optional external logic supply, 12…30 V DC
58	Spare	not connected
59	GND	
60	GND	

NOTICE

- For the digital inputs, the switch-on threshold is 1.79 V and the switch-off threshold is 1.11 V (depending on the power supply at pins 53/54, typical values @3.3 V). The maximum sampling frequency is 1 MHz. If the I/O pins are used as output (see Defining input and output assignments), the maximum admissible current is approx. 10 mA at 3.3 V DC.

TIP

You can find information on the design of your motherboard in chapter Hardware installation.

4 Hardware installation

NOTICE

Make certain that all components are de-energized.

NOTICE

- The device contains components that are sensitive to electrostatic discharge.
- Improper handling can damage the device.
- Observe the basic principles of ESD protection when handling the device.

4.1 Connecting the controller

NOTICE

EMC: Current-carrying cables – particularly around supply and motor cables – produce electromagnetic alternating fields. These can interfere with the motor and other devices.

Suitable measures may be:

- ▶ Use shielded cables and earth the cable shielding on both ends over a short distance.
- ▶ Keep power supply and motor cables as short as possible.
- ▶ Use cables with cores in twisted pairs.
- ▶ Earth motor housing with large contact area over a short distance.
- ▶ Lay supply, motor and control cables separately.

4.1.1 Integrating the CPB in a motherboard

The minimum wiring varies depending on motor type and any present feedback (stepper or BLDC motor, Hall sensors/encoders). For commissioning, the connection of the voltage supply (X1 — voltage supply and motor) of the motor and is sufficient.

The following table shows the necessary voltage rails and the requirements that your motherboard should satisfy for them. When calculating the current consumption, a low efficiency of 80% is assumed for the step-down converter.

Voltage rail	Used for	Current consumption	Ripple permitted	Tolerance of the regulated voltage	Recommended topology
+10-12V	Gate-Drivers, +3.3V, +5V	70…160 mA	100 mV p-p		Step-down converter for main supply UB
+5V	Encoder/ Hall, +3.3V	150 mA max.	100 mV p-p		Step-down converter for +10-12V
+3.3V	MCU	140 mA min.		+/-3%	Low-dropout regulator (LDO)

NOTICE

There is no polarity reversal protection. Install a fuse in the supply line dimensioned according to the current consumption in your application.

5 Commissioning

Described in this chapter is how you establish communication with the controller and set the necessary parameters to make the motor ready for operation.

5.1 Configuration via USB

5.1.1 General

NOTICE

External wiring of pins 36 (*CP*) and 38 (DM) is required for USB (see Pin assignment).

The following options are available for configuring the controller via USB:

Configuration file

This file can be saved to the controller via the USB connection. For further information, read chapters USB connection and Configuration file.

NanoJ program

This program can be programmed, compiled and then transferred to the controller with *NanoJ* via USB. For further information, read chapters NanoJ program and Programming with NanoJ.

After connecting to a voltage supply, the controller reads out the configuration in the following order:

1. The configuration file is read out and processed.
2. The NanoJ program is started.

5.1.2 USB connection

If the controller is connected to a PC via a USB cable, the controller behaves like a removable storage device. No further drivers are required.

Three files are displayed: the configuration file (`cfg.txt`), the *NanoJ program* (`vmmcode.usr`) and the information file (`info.txt`), where the serial numbers and firmware version of the product can be found.

You can thereby store the configuration file or the *NanoJ program* on the controller. The voltage supply of the controller must also be connected during USB operation.

NOTICE

- Only use a standard USB cable. Never use a USB cable that manufacturers of mobile phones include with their products. These USB cables could have a different plug shape or pin assignment.
- Do not save any files on the controller other than those listed below:

  1. `cfg.txt`
  2. `vmmcode.usr`
  3. `info.bin`
  4. `reset.txt`

  Any other file is deleted when the voltage supply of the controller is switched on!

TIP

Because it is often necessary during commissioning to copy the same file to the controller following an update, it is recommended that a script file be used to perform this task.

- Under Windows, you can create a text file with file extension `bat` and the following content:

```
copy <SOURCE> <TARGET>
```

- Under Linux, you can create a script with file extension `sh` and the following content:

```
#!/bin/bash
cp <SOURCE> <TARGET>
```

5.1.3 Configuration file

5.1.3.1 General

The `cfg.txt` configuration file is used to preset values for the object dictionary to a certain value during startup. This file uses a special syntax to make accessing the objects of the object dictionary as easy as possible. The controller evaluates all assignments in the file from top to bottom.

NOTICE

If you delete the configuration file, the controller recreates the file (without content) on the next restart.

5.1.3.2 Reading and writing the file

How to access the file:

1. Connect and switch on the voltage supply.
2. Connect the controller to your PC using the USB cable.
3. After the PC has detected the device as a removable storage device, navigate in the Explorer to the directory of the controller. File `cfg.txt` (for a PD4C, the file is named `pd4cfg.txt`) is stored there.
4. Open this file with a simple text editor, such as Notepad or Vi. Do not use any programs that use markup (LibreOffice or similar).

After you have made changes to the file, proceed as follows to apply the changes through a restart:

1. Save the file if you have not yet already done so. The motor stops.
2. Disconnect the USB cable from the controller.
3. Disconnect the voltage supply from the controller for approx. 1 second until the power LEDs stop flashing.
4. Reconnect the voltage supply. When the controller is now restarted, the values in the configuration file are read out and applied.

TIP

To restart the controller, you can also copy an empty `reset.txt` file to the controller. This restarts the controller. The `reset.txt` file is deleted on the next restart.

5.1.3.3 Structure of the configuration file

Comments

Lines that begin with a semicolon are ignored by the controller.

Example

```
; This is a comment line
```

Assignments

NOTICE

Before setting a value, determine its data type (see chapter Description of the object dictionary)! The controller does not validate entries for logical errors!

Values in the object dictionary can be set with the following syntax:

```
<Index>:<Subindex>=<Value>
```

<Index>
: This value corresponds to the index of the object and is interpreted as a hexadecimal number. The value must always be specified with four digits.

<Subindex>
: This value corresponds to the subindex of the object and is interpreted as a hexadecimal number. The value must always be specified with two digits and can be omitted if the subindex is 00_h.

<Value>
: The value that is to be written in the object is interpreted as a hexadecimal number. Hexadecimal numbers are to be prefixed with "`0x`".

You can also set individual bits:

Set bit

```
3202:00.03=1
```

Reset bit

```
3202:00.03=0
```

Bitwise OR

```
3202:00|=0x08
```

Bitwise AND

```
3202:00&=0x08
```

Example

Set object 203B$_h$:01 (rated current) to the value "600" (mA):

```
203B:01=600
```

Set object 3202$_h$:00 to the value "8" (activate current reduction while at a standstill in *open-loop* mode):

```
3202:00=8
```

or only set bit 3

```
3202:00.03=1
```

NOTICE

- There must be no blank characters to the left and right of the equal sign. The following assignments are not correct:
  `6040:00 =5`
  `6040:00= 5`
  `6040:00 = 5`
- The number of places must not be changed. The index must be four characters long and the subindex two characters long. The following assignments are not correct:
  `6040:0=6`
  `6040=6`
- Blank spaces at the start of the line are not permitted.

5.1.4 NanoJ program

A *NanoJ program* can be executed on the controller. To load and start a program on the controller, proceed as follows:

1. Write and compile your program as described in chapter Programming with NanoJ.
2. Connect the voltage supply to the controller and switch on the voltage supply.
3. Connect the controller to your PC using the USB cable.
4. After the PC has detected the device as a removable storage device, open an Explorer window and delete file `vmmcode.usr` on the controller.
5. Navigate in the Explorer to the directory with your program. The compiled file has the same name as the source code file, only with file extension `.usr`. Rename this file `vmmcode.usr`.
6. Copy file `vmmcode.usr` to the controller.
   To start the *NanoJ program* the next time the controller is restarted, add the following line to the configuration file:

   ```
   2300:00=1
   ```
7. Disconnect the voltage supply from the controller for approx. 1 second until the power LEDs stop flashing.
8. Reconnect the voltage supply. When the controller now starts, the new *NanoJ program* is read in and started.

TIP

To restart the controller, you can also copy an empty `reset.txt` file to the controller. This restarts the controller. The `reset.txt` file is deleted on the next restart.

NOTICE

- The *NanoJ program* on the controller must have file name `vmmcode.usr`.
- If the *NanoJ program* was deleted, an empty file named `vmmcode.usr` is created the next time the controller is started.

TIP

It is possible to automate the deletion of the old *NanoJ program* and the copying of the new one with a script file:

- Under Windows, you can create a file with file extension `bat` and the following content:

  ```
  copy <SOURCE_PATH>\<OUTPUT>.usr <TARGET>:\vmmcode.usr
  ```

  For example:

  ```
  copy c:\test\main.usr n:\vmmcode.usr
  ```

- Under Linux, you can create a script with file extension `sh` and the following content:

  ```
  #!/bin/bash
  cp <SOURCE_PATH>/<OUTPUT>.usr <TARGET_PATH>/vmmcode.usr
  ```

You can protect your *NanoJ program* from being read out/copied by activating the *hidden* attribute of the FAT file system.

5.2 Configuration of the Ethernet interface

5.2.1 Overview

5.2.1.1 Hardware address

The controller initially has no IP address, but is instead addressed via the printed hardware address (MAC address). This address consists of 6 hexadecimal numbers in form44-AA-E8-xx-xx-xx.

The hardware address is unique and unchangeable and is assigned during production. In general, this is only needed during a firmware update. As soon as the boot loader has ended and the actual firmware begins to operate, subsequent communication takes place via the TCP/IP protocol.

5.2.1.2 IP address

The controller needs a valid IP address. This can be obtained in the following ways:

- DHCP: A DHCP server assigns the IP address to the controller (default setting).
- AutoIP: The controller automatically determines a suitable IP address. The prerequisite here is that the communication partners are in the same physical subnet and also use AutoIP.
- Static IP address: This is defined by the user.

Which method is used depends on the network environment and is defined by the network administrator.

The IP address can be determined most easily using the *ping* tool. To do this, the NetBIOS service must be activated on the PC and the MAC address of the controller must be known.

Example

If controller with MAC address 44:AA:E8:00:02:9F is to be addressed, the tool is called in a shell or command line using:

```
ping MAC-44AAE800029F
```

5.2.2 Setting the IP address

Each of the connected devices (controller and communication partners) in an Ethernet network or with a point-to-point Ethernet connection requires a unique IP address. This can either be obtained automatically (DHCP) or generated (Auto-IP) or assigned statically. In the following, "communication partner" refers to a PC or laptop.

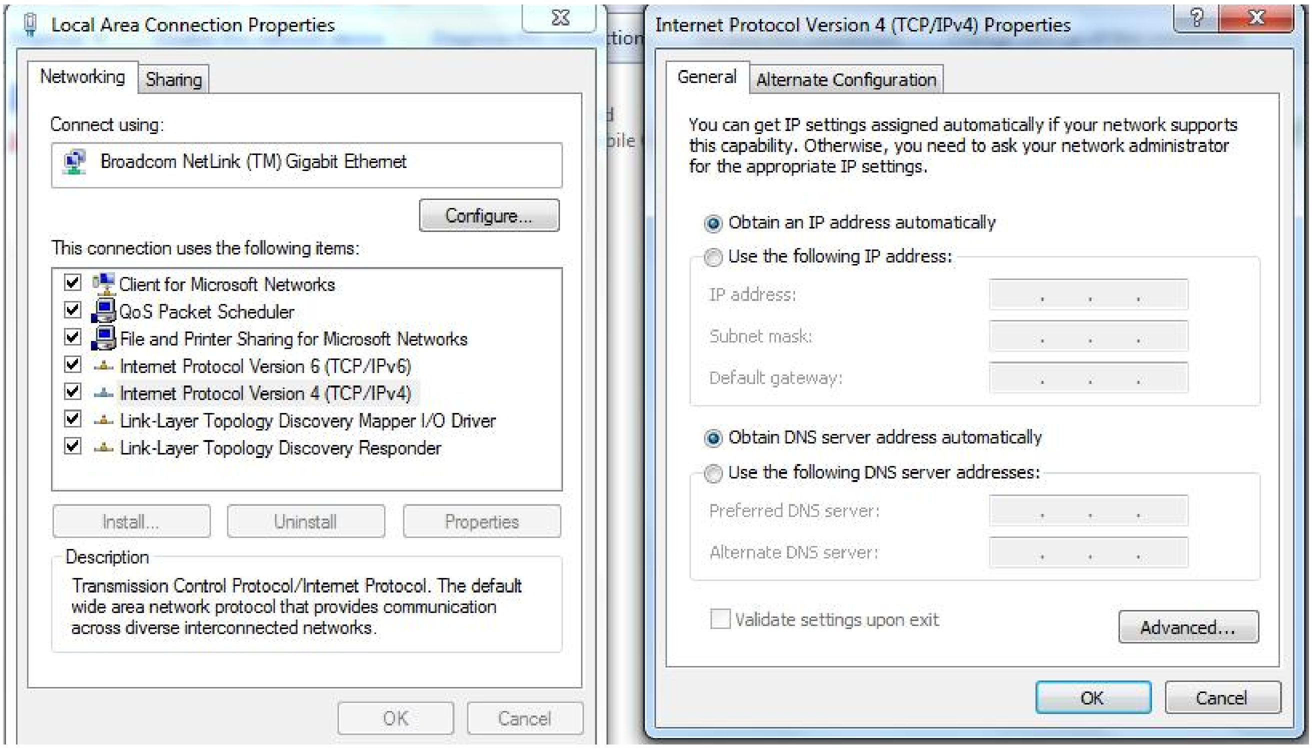
You can integrate the controller in an existing Ethernet network. To do this, you only need to establish the physical connection with a standard Ethernet cable. Provided DHCP is activated on the controller (factory setting), the controller is also automatically detected on the network and can immediately be operated via a PC located on the network.

5.2.3 Setting DHCP/Auto-IP

IP addresses can be obtained dynamically in a network from a DHCP server or, for example, in the case of a PC direct connection, can be automatically self-generated without DHCP by the two communication devices (e.g., PC and controller). DHCP is preset in the controller at the factory for automatically obtaining an IP address from a DHPC server or for automatic IP address generation.

To establish the connection to the controller, it may only be necessary to make a few settings on the communication partner (e.g., PC or laptop). Settings using the Windows 7 operating system as an example:

1. Press the Windows Start button and select *Control Panel*.
2. Select *Network and Sharing Center*.
3. Select *Change adapter settings*.
4. A list of the available network adapters is displayed. Open the properties on the adapter to which the controller is connected (e.g., click with the right mouse button).
5. *Select Internet Protocol version 4 (TCP/IPv4)* and press the *Properties* button.
6. Select the *Obtain an IP address automatically* option.
7. Confirm acceptance of the entries with the *OK* button.



5.2.4 Setting a static IP address

If static addresses are to be assigned to the controller and the communication partner, just a few settings need to be performed on the controller and the communication partner.

The controller can be assigned a static IP address and network mask (each IPv4) through OD entries. The following entries are relevant in the object dictionary:

Index	Description
2010_h	IP configuration, bit mask with the following meaning: Bit 0: A static IP address from object 2011_h and the network mask from object 2012_h are used.
2011_h	Static IP address, 4 bytes in hex coding
2012_h	Static IP subnet mask, 4 bytes in hex coding
2013_h	Gateway address
2014_h	Active IP address, 4 bytes in hex coding
2015_h	Active IP subnet mask, 4 bytes in hex coding
2016_h	Currently used gateway address
$200F_h$	MAC address

NOTICE

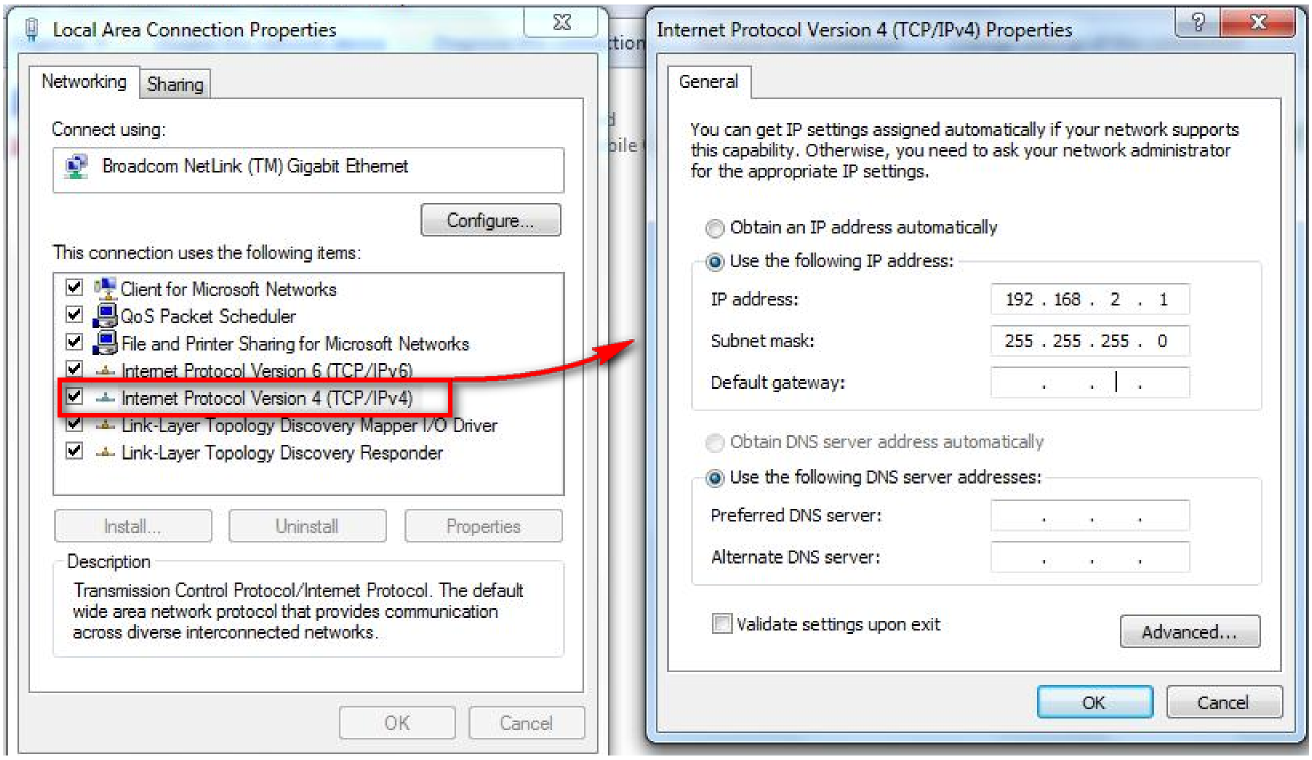
You must save the objects 2010_h…2013_h (*category: Ethernet*) after making changes (see the chapter Saving objects) by writing the value "65766173_h" in 1010_h:$0C_h$. The changes are not taken over until after the controller has been restarted.

Notes:

- If DHCP was activated and Auto-IP is not active, the controller uses the set static IP address if no address could be assigned via DHCP (e.g., because the DHCP service is temporarily unavailable).
- If both objects 2010_h and 2011_h are set to the value "0", an incorrect configuration is assumed and DHCP and Auto-IP are switched on.
- If bit 0 is set in object 2010_h, the static IP address is used. DHCP is not used in this case.
- If only DHCP is switched on and an IP address assignment did not function, an attempt is made independent of bit 0 to connect using the entered static IP address.

A static IP address is also assigned to the communication partner. Settings using the Windows 7 operating system as an example:

1. Press the Windows Start button and select *Control Panel*.
2. Select *Network and Sharing Center*.
3. Select *Change adapter settings*.
4. A list of the available network adapters is displayed. Open the properties on the adapter with which the controller is connected (e.g., right-mouse click and select *Properties*).
5. *Select Internet Protocol version 4 (TCP/IPv4)* and press the *Properties* button.
6. Select the *Use the following IP address:* option and enter the desired IP address and network mask in the *IP address* field.
7. Confirm acceptance of the entries with the OK button.



5.2.5 REST web services

5.2.5.1 Introduction

The protocol used by the web server is HTTP/1.0. The architecture here is realized according to REST (Representational State Transfer) which also offers the possibility to access objects/resources. The values in the object dictionary are an example of this.

The operations supported here are:

- `GET`: Request a resource
- `POST`: Add a new resource

5.2.5.2 Resource names

The name of a resource is always specified in the *URI (Uniform Resource Identifier)* notation familiar from the Internet. Via this *URI*, the controller supports access to the object dictionary. The identifier for this is:

- Od: Object dictionary

Example

Accessing a value in the object dictionary:

```
GET /od/6040/00 HTTP/1.0
```

This string is used to access entry 6040h subindex 00h in the object dictionary.

The reply is made as a JSON string and reflects the content of this object:

```
HTTP/1.0 200 OK
Server: uip/1.0
Cache-Control: no-cache, no-store, private
Content-type: application/json
"0006"
```

Writing a value to the object dictionary:

```
POST /od/6040/00 HTTP/1.0
Content-Type: application/x-www-form-urlencoded
```

`Form item: ""000F"" = ""`(Key: "000F", Value:)

This string is used to write value "15(0Fh)" to object 6040h subindex 00h.

The controller receives confirmation with status code `200 OK`:

```
HTTP/1.0 200 OK
Server: uip/1.0
```

5.2.5.3 Accessing the object dictionary

The following URIs enable access to the object dictionary:

<IP address>/od/xxxx/yy
Requests entry xxxx subindex yy from the object dictionary.

<IP address>/od/xxxx/data
Requests entry xxxx with all subindices.

Example

Accessing a value in the object dictionary:

```
http://192.168.2.100/od/6040/00
```

This string is used to access entry 6040h subindex 00h in the object dictionary.

The reply is made as a JSON string and reflects the content of this object.

5.3 Configuring via Modbus TCP

This controller is equipped with a Modbus TCP interface. All messages are sent to port 502 of the controller via TCP; only one connection is supported. No CRC (as it is used with Modbus RTU) takes place.

The I/O data with any preconfigured drive values (see Process data objects (PDO)) can be sent with the standard Modbus function codes. To configure your own I/O data, however, function code 2Bh (CAN

Encapsulation) must be supported by the Modbus master in order for the parameters to be read and written independent of the process image.

If the master does not support this function code, the I/O image can be configured and stored using *Plug & Drive Studio*. The master can then access the data using the standard Modbus function codes.

Read chapter Modbus TCP for further details.

Testing the interface

To test the interface, send bytes `00 00 00 00 00 0D 00 2B 0D 00 00 01 60 41 00 00 00 00 02` to the controller (you can find a detailed description of the Modbus function codes in chapter Modbus TCP).

Statusword (6041_h) was read; you receive this response: `00 00 00 00 0F 00 2B 0D 00 00 01 60 41 00 00 00 00 XX XX 06`.

5.4 Setting the motor data

Prior to commissioning, the motor controller requires a number of values from the motor data sheet.

- Number of pole pairs: Object 2030_h:00_h (pole pair count) The number of motor pole pairs is to be entered here. With a stepper motor, the number of pole pairs is calculated using the step angle, e.g., 1.8° = 50 pole pairs, 0.9° = 100 pole pairs (see step angle in motor data sheet). With BLDC motors, the number of pole pairs is specified directly in the motor data sheet.
- Object 2031_h:00_h: maximum permissible motor current (motor protection) in mA (see motor data sheet)
- Object 6075_h:00_h: rated current of the motor in mA (see motor data sheet), limited by 2031_h
- Object 6073_h:00_h: maximum current (for a stepper motor, generally corresponds to the rated current, bipolar) in tenths of a percent of the set rated current (see motor data sheet). Factory settings: "1000", which corresponds to 100% of the value in 6075_h. Is limited by 2031_h.
- Object $203B_h$:02_h Maximum duration of the maximum current (6073_h) in ms (for initial commissioning, Nanotec recommends a value of 100 ms; this value is to be adapted later to the specific application).
- Setting the motor type:
  - Stepper motor:
    - Object 3202_h:00_h (Motor Drive Submode Select): Defines motor type stepper motor, activates current reduction on motor standstill: 0000008h.See also chapter Commissioning *open-loop*.
  - BLDC motor:
    - Object 3202_h:00_h (Motor Drive Submode Select): Defines motor type BLDC: 00000040h
- Motor with encoder without index: You must set the encoder parameters after the Auto setup, see chapter Configuring the sensors.
-
  - Stepper motor, brake control (and current reduction) activated: 0000000Ch
  - BLDC motor, brake control activated: 00000044h

NOTICE

Due to the sine commutation and the sinusoidal current flow, the current of a motor winding can achieve an alternating current value that is briefly greater (by max. √2 times) than the set current.

At especially slow speeds or while at a standstill with full load, one of the windings can therefore be supplied with overcurrent for a longer period of time. Take this into account when dimensioning the motor and select a motor with larger torque reserve if necessary if required by the application.

5.5 Auto setup

To determine a number of parameters related to the motor and the connected sensors (encoders/Hall sensors), you must perform an auto setup.

TIP

As long as the motor connected to the controller or the sensors for feedback (encoders/Hall sensors) are not changed, auto setup is only to be performed once during initial commissioning.

NOTICE

Note the following prerequisites for performing the auto setup:

- ▶ The motor must be load-free.
- ▶ The motor must not be touched.
- ▶ The motor must be able to turn freely in any direction.
- ▶ No NanoJ programs may be running (object 2300_h:00_h bit 0 = "0", see 2300h NanoJ Control).

TIP

Execution of the auto setup requires a relatively large amount of processor computing power. During the auto setup, this may result in fieldbuses not being operated in a timely manner.

5.5.1 Parameter determination

Auto setup determines various parameters of the connected motor and of the present sensors by means of multiple test runs and measurement runs. To a certain extent, the type and number of parameters are dependent on the respective motor configuration.

Parameter	All motors independent of the configuration
Motor type (stepper motor or BLDC motor)	✓
Winding resistance	✓
Winding inductance	✓
Interlinking flux	✓

NOTICE

It is not possible to determine the interlinking flux on motors whose windings have widely differing inductances. These motors are, therefore, not suitable for sensorless *closed-loop* operation.

Parameter	Motor without encoder	Motor with encoder and index	Motor with encoder without index
Encoder resolution	-	✓	---
Alignment (shifting of the electrical zero to the index)	-	✓	---

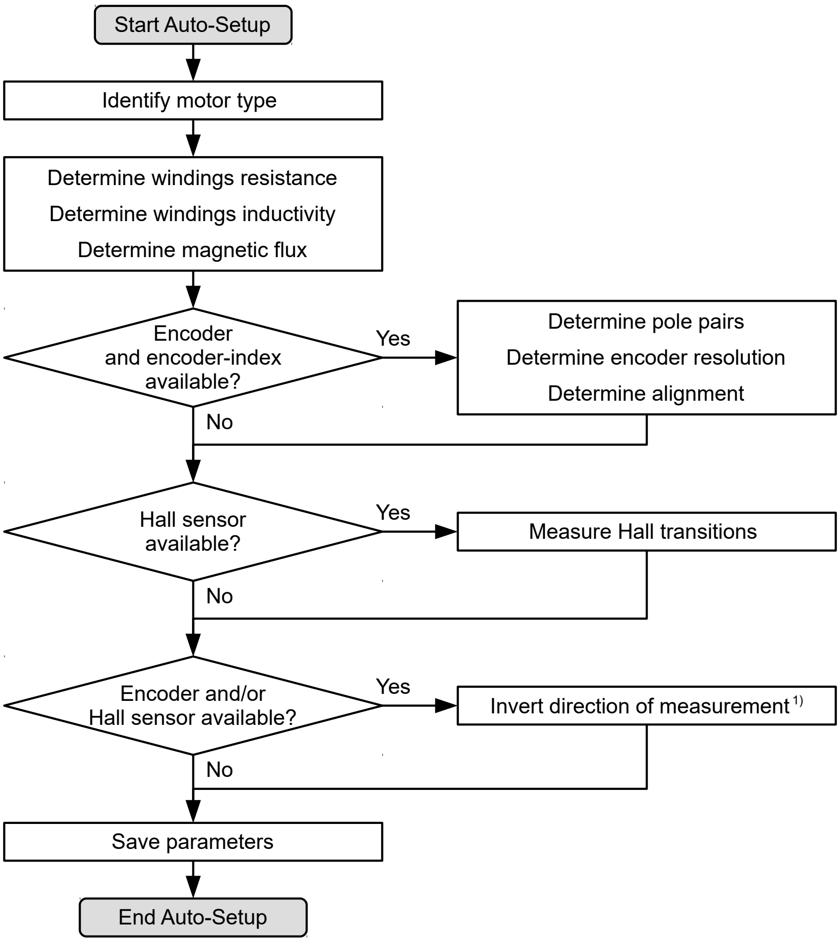
Parameter	Motor without Hall sensor	Motor with Hall sensor
Hall transitions	-	✓

5.5.2 Execution

Before performing the *auto setup*, make certain that you have correctly set the necessary parameters (see Setting the motor data).

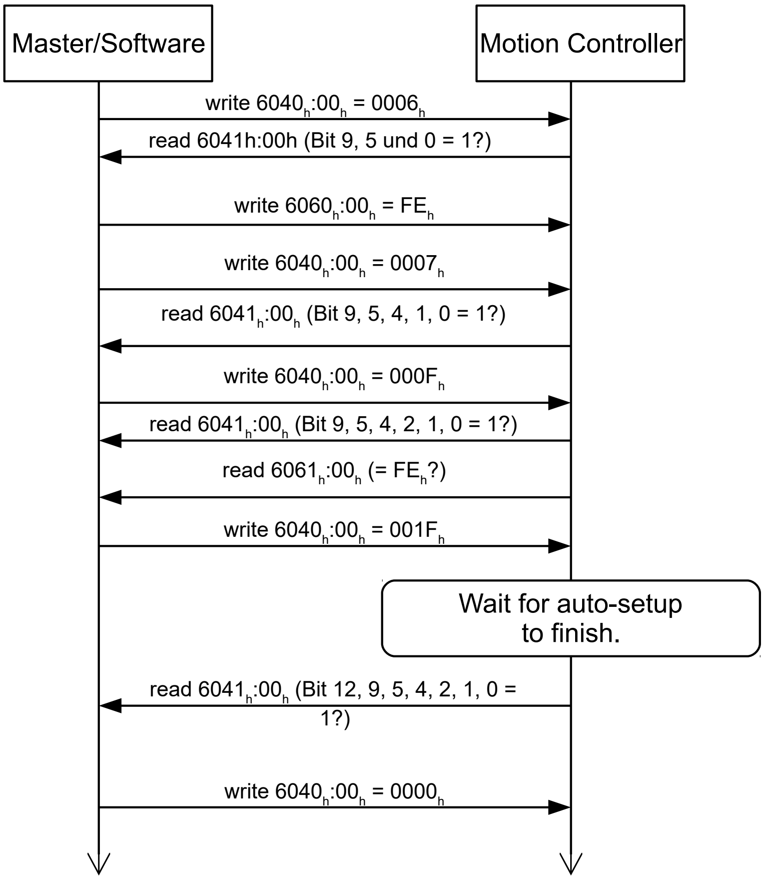
1. To preselect the *auto setup* operating mode, enter the value "-2" (="FE$_h$") in object 6060$_h$:00$_h$.
   The *power state machine* must now switch to the *Operation enabled* state, see CiA 402 Power State Machine.
2. Start *auto setup* by setting bit 4 *OMS* in object 6040$_h$:00$_h$ (controlword).

   While the auto setup is running, the following tests and measurements are performed in succession:



1) To determine the values, the direction of the measurement method is reversed and edge detection re-evaluated.

Value 1 in bit 12 *OMS* in object 6041$_h$:00$_h$ (statusword) indicates that the auto setup was completely executed and ended. In addition, bit 10 *TARG* in object 6041$_h$:00$_h$ can be used to query whether (= "1") or not (= "0") an encoder index was found.



5.5.3 Parameter memory

After a successful *auto setup*, the determined parameter values are automatically taken over into the corresponding objects and stored with the storage mechanism, see Saving objects and 1010h Store Parameters. Categories *Drive* 1010_h:05_h and *Tuning* 1010_h:06_h are used.

CAUTION!

Uncontrolled motor movements!

After the auto setup, the internal coordinate system is no longer valid. Unforeseen reactions can result.

► Restart the device after an auto setup. Homing alone does not suffice.

5.6 Configuring the sensors

The parameters (configuration, alignment, etc.) of each feedback are determined by Auto setup and stored in the following objects:

Object	Feedback	Description
3380_h	Sensorless	Contains measurement and configuration values for sensorless control
3390_h	Hall sensor (digital)	contains configuration values for the Hall sensors
$33A0_h$	Incremental encoder 1	contains configuration values for the first incremental encoder

Object	Feedback	Description
$33B0_h$	SSI encoder 1	contains configuration values for the first SSI encoder
$33B1_h$	SSI encoder 2	contains configuration values for the second SSI encoder. You can read how to activate the function of the specified pins in Defining input and output assignments.

NOTICE

It is not possible to determine the resolution of encoders without index or with more than one index per motor revolution.

In this case, you must enter and store the parameters in the corresponding objects (see 3204_h, $60E6_h$ and $60EB_h$) (category *Tuning*, see Saving objects).

For external sensors that are not mounted directly on the motor shaft, you must set and store the gear ratio according to the constructive features (objects $60E8_h$ and $60ED_h$) and/or the feed constant (objects $60E9_h$ and $60EE_h$) (category *Application*).

Example

An encoder with a resolution of 2000 increments/mm was connected that is to be used in the field directly at the process for a high-precision position measurement. The constructive design was realized as follows:

Motor	Gearbox	Process	Encoder
Rotary	Rotary | Rotary	Rotary | Translational	Translational
1	i=4	Diameter 40 mm | 125.6637... mm/ revolution	2000 incr./mm (62831.85 incr. per motor revolution)

You must set the resolution, gear ratio and feed constant as follows:

Object	Value
60E6h Additional Position Encoder Resolution - Encoder Increments	1256637
60EBh Additional Position Encoder Resolution - Motor Revolutions	20
60E8h Additional Gear Ratio - Motor Shaft Revolutions	4
60EDh Additional Gear Ratio - Driving Shaft Revolutions	1
60E9h Additional Feed Constant - Feed	2513274 incr. (corresponds to 1256.637 mm)
60EEh Additional Feed Constant - Driving Shaft Revolutions	10

You must still set the unit for the position to millimeters or other unit of length, see chapter User-defined units.

In object 3203h you can set which of the present feedbacks the controller takes into account for each controller (current controller/commutation, velocity controller, position controller) in *closed-loop* or the determination of the actual position and actual speed in *open-loop*. See also chapter Closed-Loop and Assignment of the feedbacks to the control loops.

NOTICE

The value "0" in a subindex of the object 60E6h means that the respective feedback is not connected and is not used. Thus, it is possible, for Example, to switch off the sensorless function to save computing time. This can be helpful if a *NanoJ* program needs the computing time.

If a value is not equal to "0" in a subindex, the controller checks the corresponding sensor when switching on. In case of an error (signal not present, invalid configuration/state), the error bit is set in the statusword and an error code stored in object 1003h.

6 General concepts

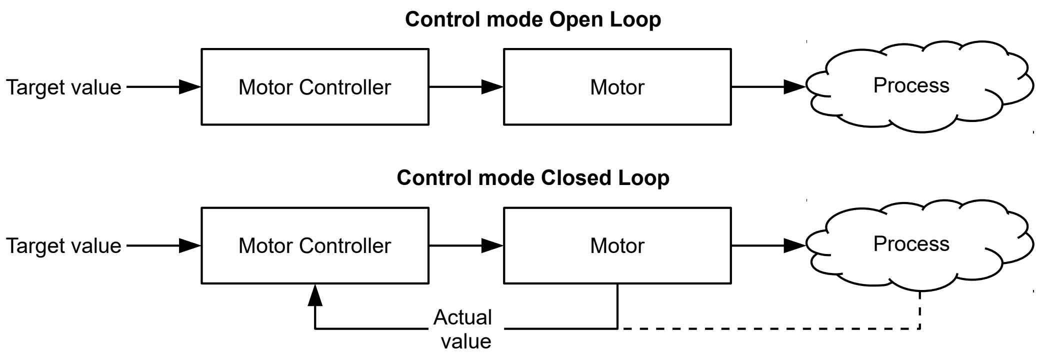
6.1 Control modes

6.1.1 General

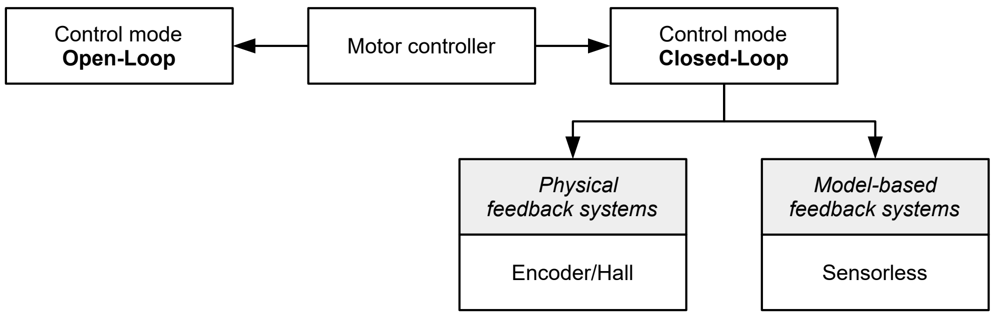
The control mode of systems without feedback is called *open-loop*, the mode with feedback is called *closed-loop*. In the *closed-loop* control mode, it is initially irrelevant whether the fed back signals come from the motor itself or from the influenced process.

For controllers with feedback, the measured control variable (actual value) is constantly compared with a set point (set value). In the event of deviations between these values, the controller readjusts according to the specified control parameters.

Pure controllers, on the other hand, have no feedback for the value that is to be regulated. The set point (set value) is only specified.



In addition to the physical feedback systems (e.g., via encoders or Hall sensors), model-based feedback systems, collectively referred to as *sensorless* systems, are also used. Both feedback systems can also be used in combination to further improve the control quality.



Summarized in the following are all possible combinations of control modes and feedback systems with respect to the motor technology. Support of the respective control mode and feedback is controller-specific and is described in chapters *Pin assignment* and Operating modes.

Control mode	Stepper motor	BLDC motor
Open-Loop	yes	no
Closed-Loop	yes	yes

Feedback	Stepper motor	BLDC motor
Hall	no	yes
Encoder	yes	yes

Feedback	Stepper motor	BLDC motor
Sensorless	yes	yes

Nanotec developed the Slow Speed control mode, which is a combination of *open-loop* and *closed-loop*, especially for applications in the low speed range. This control mode can be used if an encoder is present as feedback.

Various operating modes can be used depending on the control mode. The following list contains all the types of operation that are possible in the various control modes.

Operating mode	Control mode		
	Open-Loop	**Closed-Loop**	**Slow Speed**
Profile Position	yes	yes	yes
Velocity	yes	yes	yes
Profile Velocity	yes	yes	yes
Profile Torque	no[1)]	yes	no
Homing	yes[2)]	yes	yes
Interpolated Position Mode	yes[3)]	yes	yes
Cyclic Synchronous Position	yes[3)]	yes	yes
Cyclic Synchronous Velocity	yes[3)]	yes	yes
Cyclic Synchronous Torque	no[1)]	yes	no
Clock-direction	yes	yes	yes

1) The Profile Torque and Cyclic Synchronous Torque torque operating modes are not possible in the *open-loop* control mode due to a lack of feedback.

2) Exception: Homing on block is not possible due to a lack of feedback.

3) Because ramps and speeds in operating modes Cyclic Synchronous Position and Cyclic Synchronous Velocity follow from the specified points of the master, it is not normally possible to preselect these parameters and to ascertain whether a step loss can be excluded. It is therefore not advisable to use these operating modes in combination with *open-loop* control mode.

6.1.2 Open-Loop

6.1.2.1 Introduction

Open-loop mode is only used with stepper motors and is, by definition, a control mode without feedback. The field rotation in the stator is specified by the controller. The rotor directly follows the magnetic field rotation without step losses as long as no limit parameters, such as the maximum possible torque, are exceeded. Compared to *closed-loop*, no complex internal control processes are needed in the controller. As a result, the requirements on the controller hardware and the controller logic are very low. *Open-loop* mode is used primarily with price-sensitive applications and simple movement tasks.

Because, unlike *closed-loop*, there is no feedback for the current rotor position, no conclusion can be drawn on the counter torque being applied to the output side of the motor shaft. To compensate for any torque fluctuations that arise on the output shaft of the motor, in *open-loop* mode, the controller always supplies the maximum possible (e.g., specified by parameters) set current to the stator windings over the entire speed range. The high magnetic field strength thereby produced forces the rotor to assume the new steady state in a very short time. This torque is, however, opposite that of the inertia of the rotor and overall system. Under certain operating conditions, this combination is prone to resonances, comparable to a spring-mass system.

6.1.2.2 Commissioning

To use *open-loop* mode, the following settings are necessary:

- In object 2030$_h$ (Pole Pair Count), enter the number of pole pairs (see motor data sheet: for a stepper motor with 2 phases, a step angle of 1.8° corresponds to 50 pole pairs and 0.9° corresponds to 100 pole pairs).
- In object 2031$_h$:00$_h$, enter the maximum permissible motor current (motor protection) in mA (see motor data sheet)
- In object 6075$_h$:00$_h$, enter the rated current of the motor in mA (see motor data sheet).
- In object 6073$_h$:00$_h$, enter the maximum current (for a stepper motor, generally corresponds to the rated current, bipolar) in tenths of a percent of the set rated current (see motor data sheet). Factory settings: "1000", which corresponds to 100% of the value in 6073$_h$. A value greater than "1000" is limited internally to "1000".
- In object 3202$_h$ (Motor Drive Submode Select), set bit 0 (CL/OL) to the value "0".

Nanotec recommends to activate the current reduction on motor standstill in order to reduce the power loss and heat build-up. To activate current reduction, the following settings are necessary:

- In object 3202$_h$ (Motor Drive Submode Select), set bit 3 (CurRed) to "1".
- In object 2036$_h$ (open-loop current reduction idle time), the time in milliseconds is specified that the motor must be at a standstill (set value is checked) before current reduction is activated.
- In object 2037$_h$ (open-loop current reduction value/factor), the root mean square is specified to which the rated current is to be reduced if current reduction is activated in *open loop* and the motor is at a standstill.

6.1.2.3 Optimizations

Depending on the system, resonances may occur in *open-loop* mode; susceptibility to resonances is particularly high at low loads. Practical experience has shown that, depending on the application, various measures are effective for largely reducing resonances:

- Reduce or increase current, see objects 6073$_h$ and 6075$_h$, respectively. An excessive torque reserve promotes resonances.
- Reduce or increase the operating voltage, taking into account the product-specific ranges (with sufficient torque reserve). The permissible operating voltage range can be found in the product data sheet.
- Optimize the control parameters of the current controller via objects 3210$_h$:09$_h$ (I_P) and 3210$_h$:0A$_h$ (I_I) (generally not necessary).
- Adjustments to the acceleration, deceleration and/or target speed depending on the selected control mode:

  Profile Position operating mode

  Objects 6083$_h$ (Profile Acceleration), 6084$_h$ (Profile Deceleration) and 6081$_h$ (Profile Velocity).

  Velocity operating mode

  Objects 6048$_h$ (Velocity Acceleration), 6049$_h$ (Velocity Deceleration) and 6042$_h$ (Target Velocity).

  Profile Velocity operating mode

  Objects 6083$_h$ (Profile Acceleration), 6084$_h$ (Profile Deceleration) and 6081$_h$ (Profile Velocity).

  Homing operating mode

  Objects 609A$_h$ (Homing Acceleration), 6099$_h$:01$_h$ (Speed During Search For Switch) and 6099$_h$:02$_h$ (Speed During Search For Zero).

  Interpolated Position Mode operating mode

  The acceleration and deceleration ramps can be influenced with the higher-level controller.

  Cyclic Synchronous Position operating mode

  The acceleration and deceleration ramps can be influenced via the external "position specification / time unit" targets.

  Cyclic Synchronous Velocity operating mode

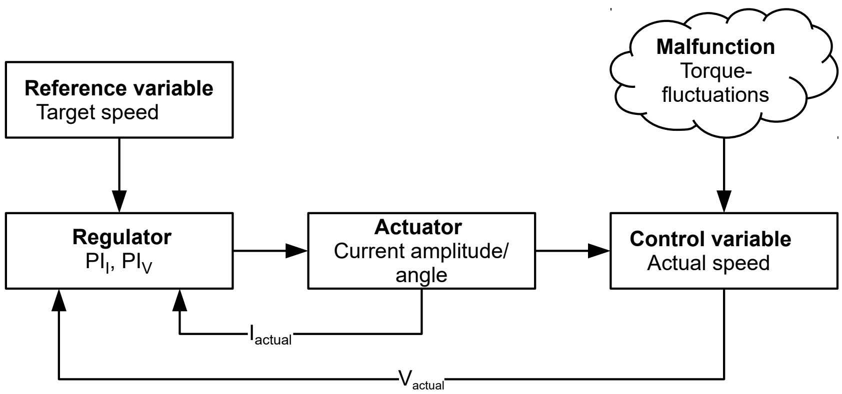
  The acceleration and deceleration ramps can be influenced via the external "position specification / time unit" targets.

6.1.3 Closed-Loop

6.1.3.1 Introduction

The *closed-loop* theory is based on the idea of a control loop. A disturbance acting on a system should be compensated for quickly and without lasting deviation to adjust the control variable back to the set point.

Closed loop using a speed control as an example:



PI_I	=	Proportional-integral current control loop
PI_V	=	Proportional-integral velocity control loop
I_{actual}	=	Actual current
V_{actual}	=	Actual speed

The *closed-loop* method is also referred to as "sine commutation via an encoder with field-oriented control". At the heart of *closed-loop* technology is the performance-adjusted current control as well as the feedback of the actual values of the process. Using sensor signals, the rotor orientation is recorded and sinusoidal phase currents generated in the motor windings. Vector control of the magnetic field ensures that the magnetic field of the stator is always perpendicular to that of the rotor and that the field strength corresponds precisely to the desired torque. The current thereby controlled in the windings provides a uniform motor force and results in an especially smooth-running motor that can be precisely regulated.

The feedback of the control variables necessary for *closed-loop* mode can be realized with various technologies. In addition to the physical feedback with encoders or Hall sensors, it is also possible to virtually record the motor parameters through a software-based model calculation. Physical variables, such as speed or back-EMF, can be reconstructed with the help of a so-called "observer" from the data of the current controller. With this sensorless technology, one has a "virtual rotary encoder", which – above a certain minimum speed – supplies the position and speed information with the same precision as a real optical or magnetic encoder.

All controllers from Nanotec that support *closed-loop* mode implement a field oriented control with sine commutated current control. Thus, the stepper motors and BLDC motor are controlled in the same way as a servo motor. With *closed-loop* mode, step angle errors can be compensated for during travel and load angle errors corrected within one full step.

6.1.3.2 Controller structure

The controller consists of three cascaded PI controllers (proportional-integral): the current controller (commutation), the velocity controller and the position controller.

The current controller is active in all operating modes. The velocity controller is as well with the sole exception of the "Real Torque" modes (torque mode without speed limiting if bit 5 in 3202_h is set to "1").

The position controller is active in the following operating modes:

- Profile Position

- Homing
- Interpolated Position Mode
- Cyclic Synchronous Position
- Clock-direction mode
- Velocity/Profile Velocity/Cyclic Synchronous Velocity if bit 1 in 3202_h is set to "1"

Each controller consists of a proportional component with the *gain factor* K_p and an integral component with the *integrator time* T_i. The control variable (the output signal of the controller, which is the set point for the next controller) is limited by the maximum speed (position controller), the maximum current (velocity controller) or the maximum PWM signal (current controller), respectively.

Object	Name	Unit	Description
$321A_h:01_h$	Current controller Proportional Gain Kp for Iq	[mV/A]	Proportional component of torque-forming component
$321A_h:02_h$	Current controller Integrator Time Ti for Iq	[µs]	Integrator time of torque-forming component
$321A_h:03_h$	Current controller Proportional Gain Kp for Id	[mV/A]	Proportional component of field-forming component
$321A_h:04_h$	Current controller Integrator Time Ti for Id	[µs]	Integrator time of field-forming component
$321B_h:01_h$	Velocity controller Proportional Gain Kp	[mA/Hz]	Proportional component
$321B_h:02_h$	Velocity controller Integrator Time Ti	[µs]	Integrator time
$321C_h:01_h$	Position controller Proportional Gain Kp	[Hz] (Controller deviation in mech. revolutions per second)	Proportional component
$321C_h:02_h$	Position controller Integrator Time Ti	[µs]	Integrator time

The *gain factor* K_p has a direct influence on the current control variable: at the same deviation, the control variable is proportional to the gain factor.

Each controller also has an integral component that is determined by the *integrator time* (T_i). The smaller the integrator time, the faster the control variable increases. If the integrator time is 0, the integral component is internally set to "0" and the controller only has the proportional component.

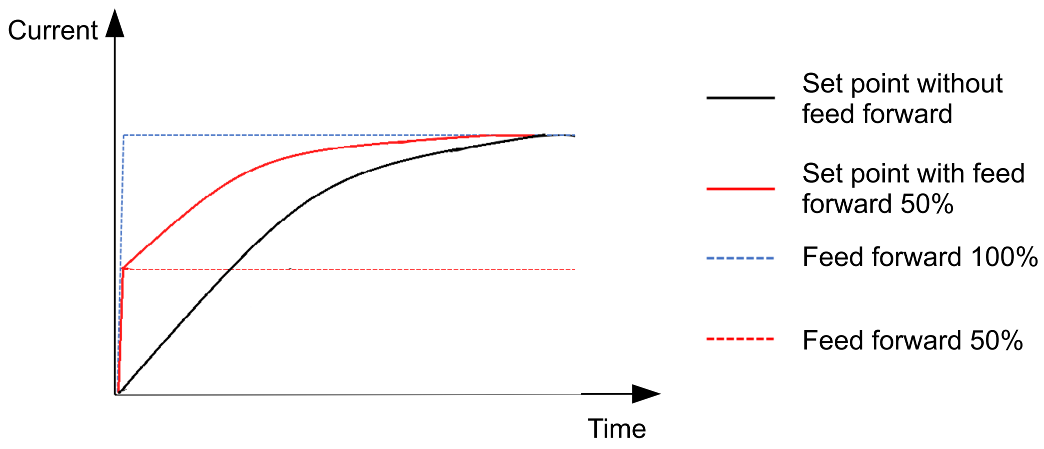
6.1.3.3 Feed forward

It is also possible to set a *velocity feed forward*, an *acceleration feed forward* (that corresponds to a torque/current value) and a *voltage feed forward*.

You can use the *feed forward* to add an already known or anticipated control variable to the set point ("predictive"). You can, e. g., compensate for the inertia of the load by adding an acceleration feed forward value to the output of the velocity controller.

The feed forward values are additionally fed to the speed/current control loop or added to the voltage value and are immediately available. A more dynamic control can thereby be achieved.

The following figure shows the current (produced by the acceleration) during the acceleration phase as a function of the *acceleration feed forward*. At a feed forward value of "50%", the current is at "50%" already at the start of the acceleration phase; the current controller is thereby "relieved".

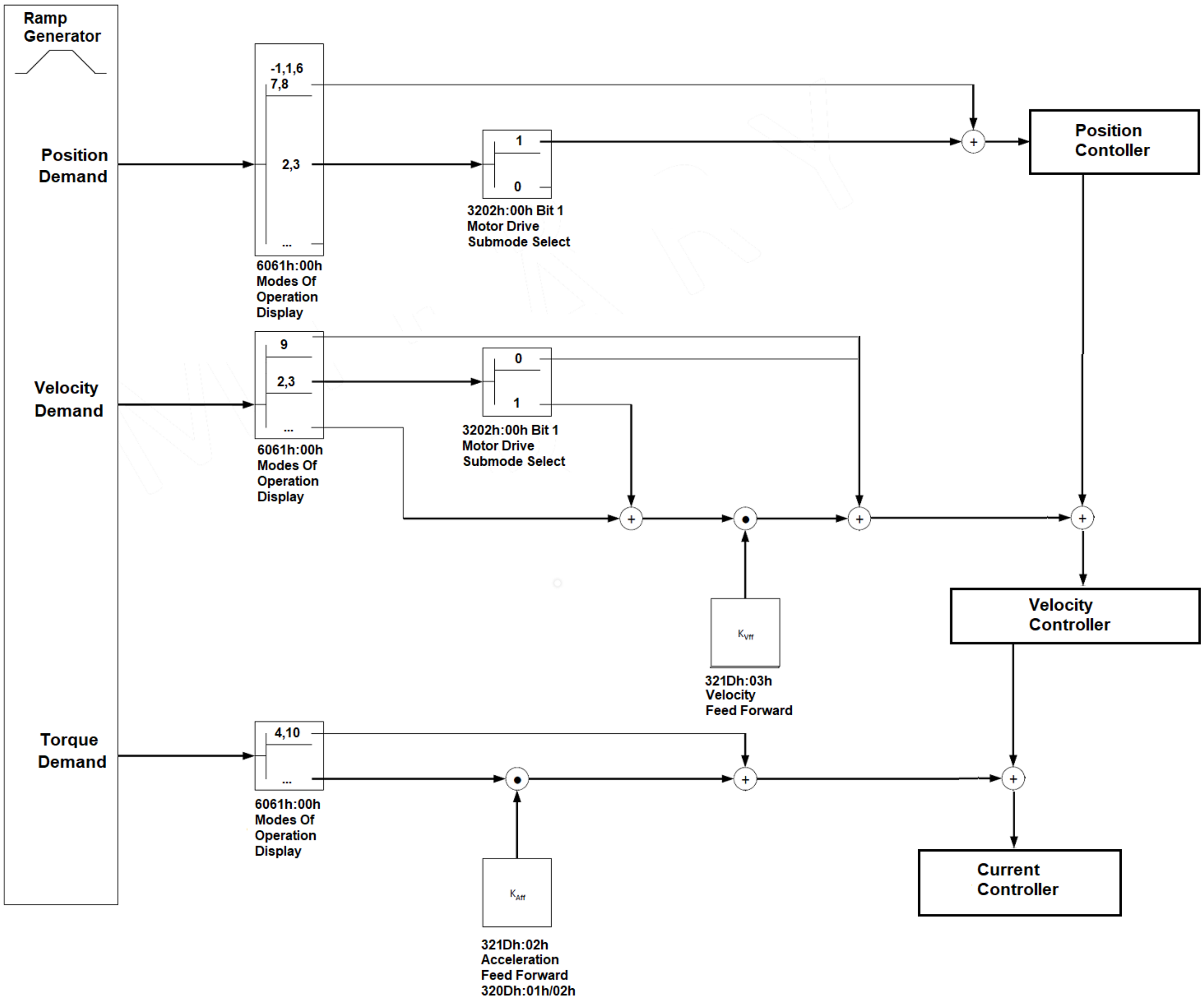


The factor for the *velocity feed forward* is set in object 321D$_h$:03$_h$ in tenths of a percent of the output of the ramp generator (606B$_h$) and added to the output of the position controller before the velocity controller. The *velocity feed forward* is active in all modes with position control loop:

- Profile Position
- Homing
- Interpolated Position Mode
- Cyclic Synchronous Position
- Clock-direction mode
- Velocity/Profile Velocity if bit 1 in 3202$_h$ is set to "1"

The factor for the *acceleration feed forward* is set in object 321D$_h$:02$_h$ in tenths of a percent of the factor of 320D$_h$ and multiplied by the output of the ramp generator (6074$_h$). The value is added to the output of the velocity controller before the current controller. The *acceleration feed forward* is active in all modes, with the exception of the torque modes.

The following figure shows the cases in which the feed forward is active and the position of the feed forward within the controller cascade.



The factor for the *voltage feed forward* is specified in object 321D$_h$:01$_h$ in tenths of a percent of the voltage that is needed to produce the rated current. If the factor is 1000‰ (factory setting), the voltage is immediately available and the actual current quickly reaches the rated current. As a result, there is practically no control deviation during acceleration and the current controller is relieved.

The *voltage feed forward* is active in all modes. To switch it off, set 321D$_h$:01$_h$ to "0".

6.1.3.4 Assignment of the feedbacks to the control loops

In object 3203$_h$, you define which of the existing feedbacks the controller takes into account for the individual controllers (current controller/commutation, velocity, position). You can also use a second sensor for the commutation (see Commutation help).

Each subindex of the object contains a bit mask for the respective feedback of a sensor. The bits have the following meaning here:

- Bit 0: If the bit is set to "1", this sensor is used for position feedback.
- Bit 1: If the bit is set to "1", this sensor is used for velocity feedback.
- Bit 2: If the bit is set to "1", this sensor is used for commutation feedback in Closed-Loop.

Subindex 01$_h$ always corresponds to the first (and always existing) *sensorless* feedback. The order of the remaining feedbacks corresponds to the table in chapter Configuring the sensors.

Which sensor the controller takes into account for the individual controllers (commutation, velocity, position) is implicitly specified by the order of the sensors.

The search always begins with sensor 2 and continues in ascending order until all existing sensors have been queried. If a sensor is found whose feedback is set, it is assigned to the corresponding controller and the search ended.

Example

The controller has two physical interfaces. Hall sensors and a (non-absolute) incremental encoder were connected.

Bit	Controller	Feedback 1 Sensorless	Feedback 2 Hall	Feedback 3 Incremental encoder
0	Position	0	0	1
1	Velocity	0	1	1[1]
2	Commutation	0	1[2]	1
	Index:Subindex	$3203_h:01_h$	$3203_h:02_h$	$3203_h:03_h$

[1]The Hall sensors should be used for velocity control, the encoder for the positioning and commutation. Although the bit for the velocity was also set for the third feedback, this is not taken into account.

[2]Immediately after switching on − and until the index of the encoder is passed over for the first time − commutation is to take place via the Hall sensors and immediately enable *closed-loop* mode.

Commutation help

Some sensors are initially lacking the alignment necessary for the commutation (offset between the index of the encoder and the magnets of the rotor). This means that the rotor orientation cannot be determined using only the position information of the sensor.

For assistance, you can set a second sensor as commutation sensor (bit 2 of the corresponding subindex in 3203_h). It is thereby possible, for example, for each (electric) absolute sensor with alignment (such as a Hall sensor), to offer commutation assistance, e. g., for an incremental encoder without index or still missing alignment (index signal not yet seen since a restart). The controller automatically uses the better sensor for the commutation.

If no second commutation sensor is selected or if the alignment is missing for the selected sensors, an auto-alignment is determined in *open-loop* if necessary (independent of bit 4 in 3202_h).

6.1.3.5 Commissioning

An auto setup should be performed before using *closed-loop* mode. The auto setup operating mode automatically determines the necessary parameters (e.g., motor data, feedback systems) that are necessary for optimum operation of the field oriented control. All information necessary for performing the auto setup can be found in chapter Auto setup.

To use *closed-loop* mode, certain settings are necessary depending on the motor type and feedback; see chapter Setting the motor data.

Bit 0 in 3202_h must be set . The bit is set automatically after a successfully completed auto setup.

Activation

If an (electric) absolute sensor (e.g., Hall sensor) is used for the commutation, the *closed-loop* is activated automatically already when switching on.

If an encoder is used for the commutation, the index of the encoder must be passed over at least once after switching on before *closed-loop* can be activated (remains in *open-loop* mode until this takes place).

If no index is present or if it cannot be used, you can:

- use a second sensor for commutation (see Assignment of the feedbacks to the control loops)
- or have an *auto alignment* determined in *open-loop* by setting bit 4 in 3202_h to "1".
  Auto alignment is determined once every time the controller is restarted after the first command that switches the CiA 402 Power State Machine to the *Operation enabled* state.
  In doing so, the rotor is moved up to a magnetic pole. After the alignment has been determined, the *Operation enabled* state is reached and travel continues if applicable.

NOTICE

To be able to determine the *auto alignment*, you must ensure that the (automatic or manual) brake control is deactivated (see chapter Automatic brake control).

CAUTION!

Uncontrolled motor movements!

Unforeseeable reactions can result if the alignment is not correctly determined.

Please observe the following requirements for the use of auto alignment:

► The motor shaft must ideally be load-free. If this is not possible, the motor must be designed so that there is a large torque reserve (at least 25%).

► Use an encoder with sufficiently high resolution (at least 500 counts per revolution, after quadrature, for a motor with 50 pole pairs)

Bit 15 in 6041h Statusword indicates whether or not *closed-loop* is active (if the state of CiA 402 Power State Machine is *Operation enabled*).

6.1.3.6 Optimizations

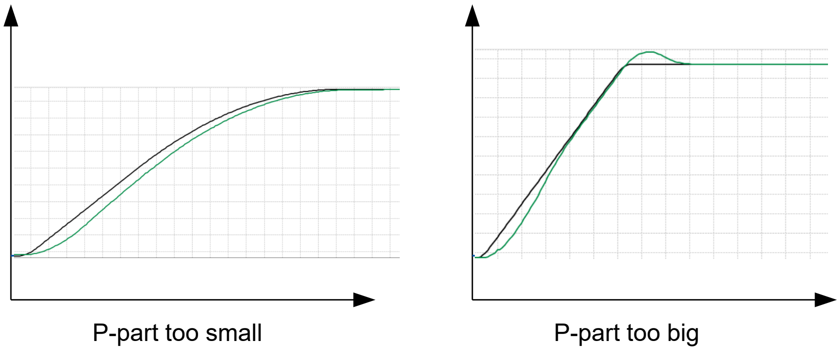
In *closed-loop*, the measured control variable (actual value) is constantly compared with a set point (set value). In the event of deviations between these values, the controller readjusts according to the specified control parameters.

The objective of control parameter optimization (the so-called *tuning* of the controller) is the smoothest possible running of the motor, high accuracy and high dynamics in the reaction of the controller to faults. All control deviations should be eliminated as quickly as possible.

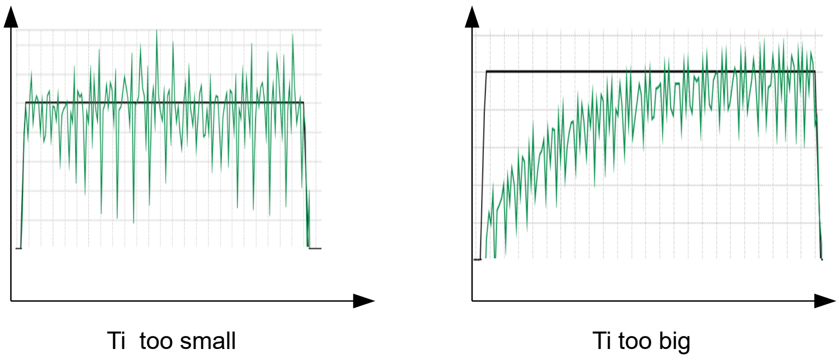
Due to the cascaded Controller structure, it is useful to start the optimization of the inner-most controller (current controller) before the velocity and – if applicable – the position controller are optimized. Each of the three controllers consists of a proportional and an integral component, which should normally be adjusted in this order.

The following figures show the reaction of the controller to a change in set value.

If the proportional component is too small, the actual value remains below the set value. A proportional component that is too large, on the other hand, results in "overshooting".



If the integrator time is too small, the system tends toward oscillations. If the integrator time is too large, the deviations are compensated for too slowly.



CAUTION!

Risk of injury through uncontrolled motor movements!

Incorrect control parameters may result in an unstable control behavior. Unforeseen reactions can result.

► Increase the control parameters slowly and incrementally. Do not increase these further if you notice strong vibrations/oscillations.

► Do not reach for moving parts during operation. After switching off, wait until all movements have ended.

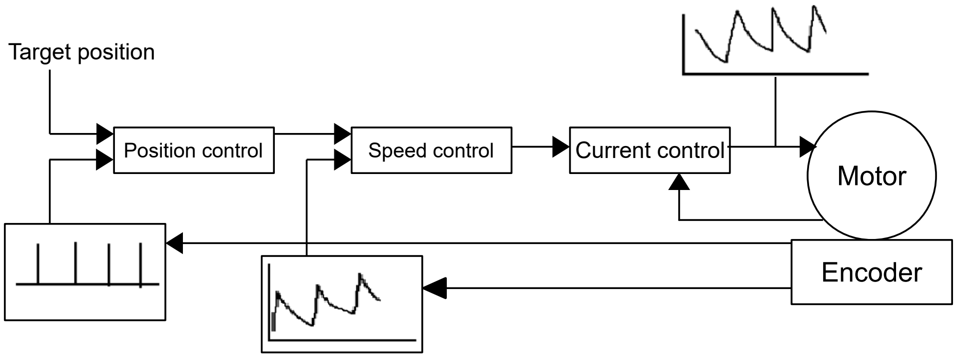
6.1.4 Slow Speed

6.1.4.1 Introduction

The *slow speed* mode combines the advantages of *open-loop* and *closed-loop* technologies in a low speed range and can be used if an encoder is present as feedback. *Slow speed* offers following error monitoring but is more smooth-running than in pure *closed-loop* mode at low speeds.

The rotor orientation is detected via the signals of the encoder. To calculate the speed, the change of position is divided by the (fixed) cycle time. At low speeds, the controller counts fewer (or even no) encoder increments in one cycle, which leads to a speed curve with a relatively high number of peaks (in spite of the used low-pass filter).

Due to the cascaded control loop, this results in current peaks in *closed-loop* mode, which can lead to uneven running, as the following figure shows.



In the *slow speed* mode, the motor instead operates with constant phase current, as in *open-loop*. The following error is, however, monitored by means of the encoder and the vector control of the magnetic field is activated if necessary, as in *closed-loop*.

6.1.4.2 Activation

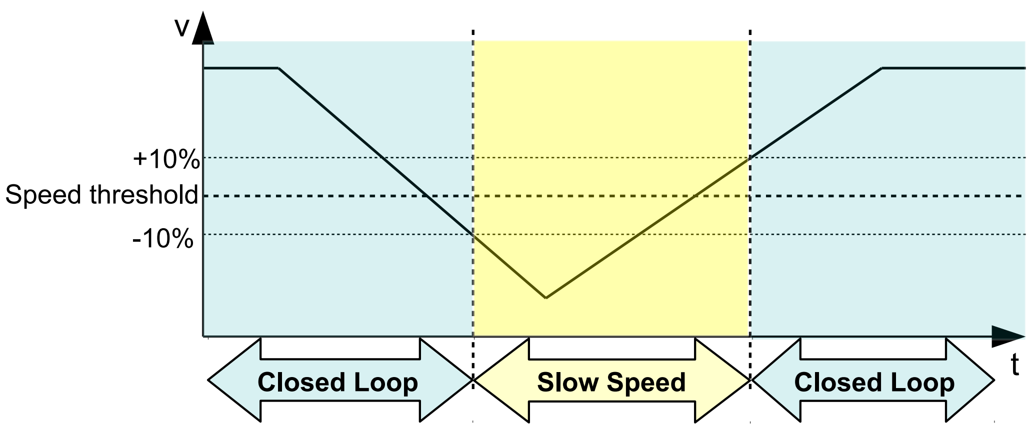
To activate the *slow speed* mode, you must:

1. activate closed-loop,
2. in object 3202_h (Motor Drive Submode Select), set bit 7 to "1".

The changeover between *slow speed* and *closed-loop* occurs automatically at a speed that is dependent on the physical encoder resolution, with a hysteresis of 10%. This fixed changeover speed is calculated in revolutions per minute as follows:

$$\frac{4000}{\text{Encoder resolution (ppr)}} \times 60$$

The following figure shows the changeover as a function of speed in both directions.



While at a standstill, the motor is in *closed-loop* mode.

6.1.4.3 Optimizations

The entire phase current remains constant as in *open-loop*. Depending on the system, resonances may occur that you can avoid by adjusting the motor current and/or the acceleration ramp. See also chapter Open-Loop.

During operation at various speed ranges, if changing between *closed-loop* and *slow speed*, it may be necessary to:

- reduce the motor current (objects 6075_h, 6073_h) if changing from *closed-loop* to *slow speed*,
- ascertain various control parameters (see Controller structure) for each speed range.

6.2 CiA 402 Power State Machine

6.2.1 State machine

6.2.1.1 CiA 402

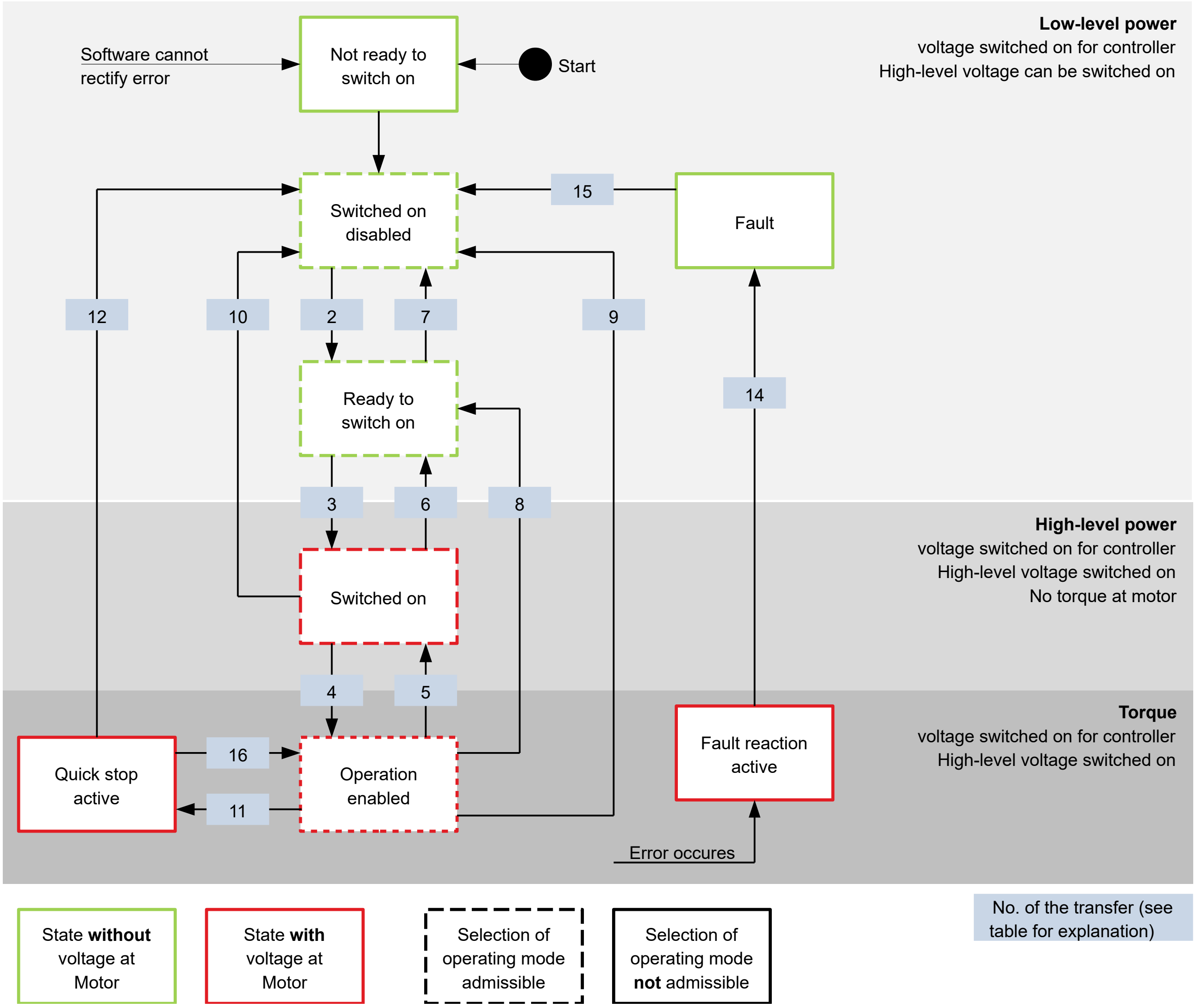
To switch the controller to the ready state, it is necessary to run through a *state machine*. This is defined in *CANopen standard 402*. State changes are requested in object 6040_h (controlword). The actual state of the state machine can be found in object 6041_h (statusword).

6.2.1.2 Controlword

State changes are requested via object 6040_h (controlword).

State transitions

The diagram shows the possible state transitions.



Listed in the following table are the bit combinations for the controlword that result in the corresponding state transitions. An X here corresponds to a bit state that requires no further consideration. Exceptions are the resetting of the error (fault reset) and the changeover from *Quick Stop Active* to *Operation Enabled*: the transition is only requested by the rising edge of the bit.

Command	**Bit in object 6040$_h$**					**Transition**
	Bit 7	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**	
Shutdown	0	X	1	1	0	2, 6, 8
Switch on	0	0	1	1	1	3
Disable voltage	0	X	X	0	X	7, 10, 9, 12
Quick stop	0	X	0	1	X	11
Disable operation	0	0	1	1	1	5
Enable operation	0	1	1	1	1	4
Enable operation after Quick stop	0	1		1	1	16

Command	Bit in object 6040_h					Transition
	Bit 7	Bit 3	Bit 2	Bit 1	Bit 0	
Fault / warning reset	↑	X	X	X	X	15

6.2.1.3 Statusword

Listed in the following table are the bit masks that break down the state of the controller.

Statusword (6041_h)	State
xxxx xxxx x0xx 0000	Not ready to switch on
xxxx xxxx x1xx 0000	Switch on disabled
xxxx xxxx x01x 0001	Ready to switch on
xxxx xxxx x01x 0011	Switched on
xxxx xxxx x01x 0111	Operation enabled
xxxx xxxx x00x 0111	Quick stop active
xxxx xxxx x0xx 1111	Fault reaction active
xxxx xxxx x0xx 1000	Fault

After switching on and successfully completing the self-test, the controller reaches the *Switch on disabled* state.

NOTICE

If an unrecoverable error occurs, the controller changes to the *Not ready to switch on* state and remains there.

6.2.1.4 Operating mode

The operating mode is set in object 6060_h. The actually active operating mode is displayed in 6061_h.

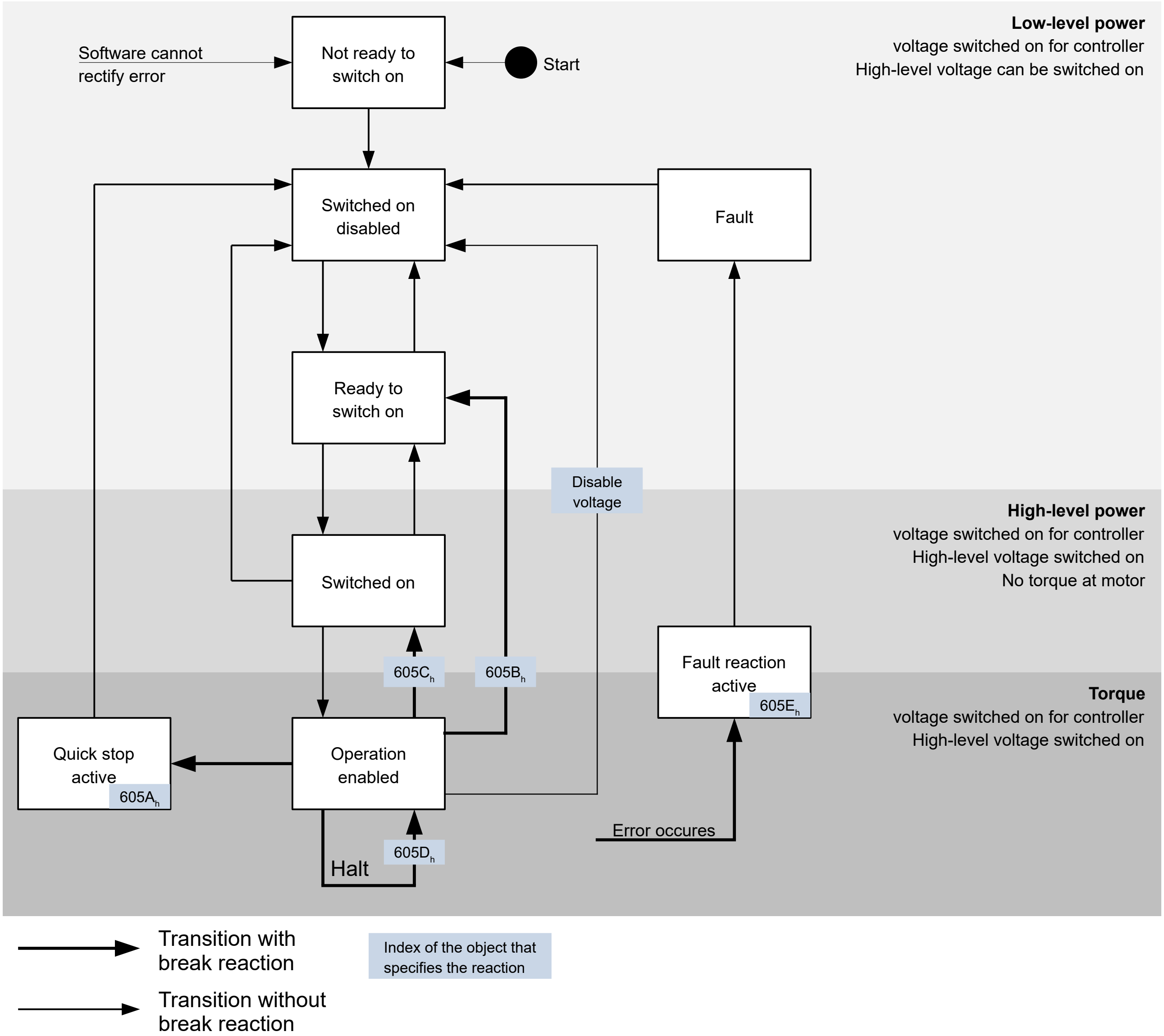
The operating mode can be set or changed at any time.

6.2.2 Behavior upon exiting the *Operation enabled* state

6.2.2.1 Halt motion reactions

Various halt motion reactions can be programmed upon exiting the *Operation enabled* state.

The following graphic shows an overview of the halt motion reactions.



6.2.2.2 Quick stop active

Transition to the *Quick stop active* state (quick stop option):

In this case, the action stored in object 605A$_h$ is executed (see following table).

Value in object 605A$_h$	Description
0	Switch off driver without deceleration ramp; drive function blocked – motor can turn freely
1	Braking with *slow down ramp* (deceleration ramp depending on operating mode) and subsequent state change to *Switch on disabled*
2	Braking with *quick stop ramp* (6085$_h$) and subsequent state change to *Switch on disabled*
5	Braking with *slow down ramp* (deceleration ramp depending on operating mode) and subsequent state change to *Quick stop active*; control does not switch off and the motor remains energized. You can switch back to the *Operation enabled* state.

Value in object $605A_h$	Description
6	Braking with *quick stop ramp* (6085_h) and subsequent state change to *Quick Stop Active*; control does not switch off and the motor remains energized. You can switch back to the *Operation enabled* state.

The *Quick stop active* state can also be reached when a limit switch is actuated; see Limitation of the range of motion.

6.2.2.3 Ready to switch on

Transition to the *Ready to switch on* state (shutdown option):

In this case, the action stored in object $605B_h$ is executed (see following table).

Value in object $605B_h$	Description
-32768 … -1	Reserved
0	Switch off driver without deceleration ramp; drive function blocked – motor can turn freely
1	Braking with *slow down ramp* (braking deceleration depending on operating mode) and subsequent state change to *Ready to switch on*
2 … 32767	Reserved

6.2.2.4 Switched on

Transition to the *Switched on* state (disable operation option):

In this case, the action stored in object $605C_h$ is executed (see following table).

Value in object $605C_h$	Description
-32768 … -1	Reserved
0	Switch off driver without deceleration ramp; drive function blocked – motor can turn freely
1	Braking with *slow down ramp* (braking deceleration depending on operating mode) and subsequent state change to *Switched on*
2 … 32767	Reserved

6.2.2.5 Halt

The bit is valid in the following modes:

- Profile Position
- Velocity
- Profile Velocity
- Profile Torque
- Interpolated Position Mode

When setting bit 8 in object 6040_h (controlword), the action stored in $605D_h$ is executed (see following table):

Value in object $605D_h$	Description
-32768 … 0	Reserved
1	Braking with *slow down ramp* (braking deceleration depending on operating mode)

Value in object $605D_h$	Description
2	Braking with *quick stop ramp* (6085_h)
3 … 32767	Reserved

6.2.2.6 Fault

Case of an error (fault):

If an error occurs, the motor will brake according to the value stored in object $605E_h$.

Value in object $605E_h$	Description
-32768 … -1	Reserved
0	Switch off driver without deceleration ramp; drive function blocked – motor can turn freely
1	Braking with *slow down ramp* (braking deceleration depending on operating mode)
2	Braking with *quick stop ramp* (6085_h)
3 … 32767	Reserved

For each error that occurs, a more precise error code is stored in object 1003_h.

6.2.2.7 Following/slippage error

If a following or slippage error occurs, the motor is braked according to the value stored in object 3700_h.

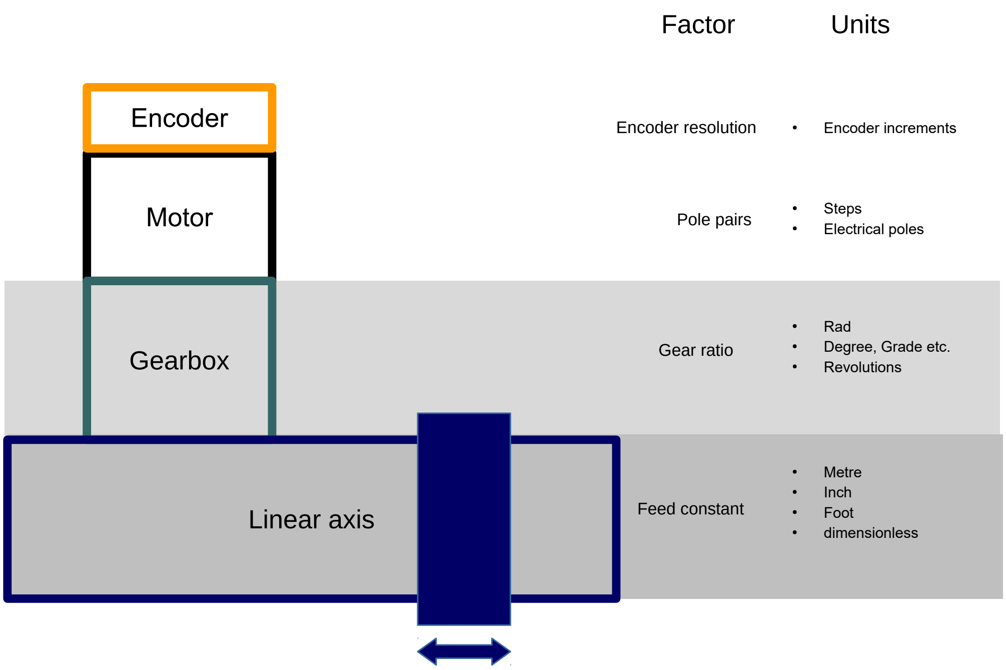
Value	Description
-32768 … -2	Reserved
-1	no reaction
0	Switch off driver without deceleration ramp; drive function blocked – motor can turn freely
1	Braking with *slow down ramp* (braking deceleration depending on operating mode)
2	Braking with *quick stop ramp* (6085_h)
3 … 32767	reserved

You can deactivate error monitoring by setting object 6065_h to the value "-1" ($FFFFFFFF_h$) or object $60F8_h$ to the value "$7FFFFFFF_h$".

6.3 User-defined units

The controller offers you the possibility to set user-defined units. It is thereby possible to set and read out the corresponding parameters, e.g., directly in degrees [°], millimeter [mm], etc.

Depending on the mechanical circumstances, you can also define a Gear ratio and/or a Feed constant.



NOTICE

Value changes of all objects that are described in this chapter are not immediately applied in the *Operation enabled* state of the CiA 402 Power State Machine. For this to happen, the *Operation enabled* state must be exited.

6.3.1 Units

Units of the international unit system (*SI*) as well as a number of specific units are supported. It is also possible to specify a power of ten as a factor.

Listed in the following table are all supported units for the position and their values for $60A8_h$ (Position unit) or $60A9_h$ (Speed unit). Depending on the unit that is used, Feed constant (6092_h) and/or Gear ratio (6091_h) are/is taken into account.

Name	Unit symbol	Value	6091_h	6092_h	Description
meter	m	01_h	yes	yes	*Meter*
inch	in	$C1_h$	yes	yes	*Inch* (=0.0254 m)
foot	ft	$C2_h$	yes	yes	*Foot* (=0.3048 m)
grade	g	40_h	yes	no	*Gradian* (unit of angle, 400 corresponds to 360°)
radian	rad	10_h	yes	no	*Radian*
degree	°	41_h	yes	no	*Degrees*
arcminute	'	42_h	yes	no	*Arcminute* (60'=1°)
arcsecond	"	43_h	yes	no	*Arcsecond* (60"=1')
mechanical revolution		$B4_h$	yes	no	*Revolution*

Name	Unit symbol	Value	6091_h	6092_h	Description
encoder increment		$B5_h$	no	no	*Encoder increments.* Dependent on the used sensor (encoder/Hall sensor) and control mode. In *open-loop* and *sensorless* mode, the number of pole pairs (2030_h) multiplied by 65536 corresponds to one motor revolution.
step		AC_h	no	no	*Steps.* With 2-phase stepper motors, the number of pole pairs (2030_h) multiplied by 4 is equivalent to one revolution. With 3-phase BLDC motors, the number of pole pairs (2030_h) multiplied by 6 is equivalent to one revolution.
electrical pole		$C0_h$	no	no	*Electric poles.* With a stepper motor that has, e.g., 50 pole pairs (2030_h), the unit corresponds to 1/50 of a revolution.
dimensionless		00_h	yes	yes	*Dimensionless length unit*

Listed in the following table are all supported units for the time and their values for $60A9_h$ (Speed unit):

Name	Unit symbol	Value	Description
second	s	03_h	*Second*
minute	min	47_h	*Minute*
hour	h	48_h	*Hour*
day	d	49_h	*Day*
year	a	$4A_h$	*Year* (=365.25 days)

Listed in the following table are the possible exponents and their values for $60A8_h$ (Position unit) and $60A9_h$ (Speed unit):

Factor	Exponent	Value
10^6	6	06_h
10^5	5	05_h
...	...	...
10^1	1	01_h
10^0	0	00_h
10^{-1}	-1	FF_h
...	..	...
10^{-5}	-5	FB_h
10^{-6}	-6	FA_h

6.3.2 Encoder resolution

The physical resolution for position measurement of the used encoder/sensor is calculated from the encoder increments ($60E6_h$ (Encoder Increments)) per motor revolutions ($60EB_h$ (Motor Revolutions)).

6.3.3 Gear ratio

The gear ratio is calculated from motor revolutions ($60E8_h$ (Motor Shaft Revolutions)) per axis rotations ($60ED_h$ (Driving Shaft Revolutions)).

6.3.4 Feed constant

The feed constant is calculated in user-defined position units from the feed (60E9$_h$ (Feed) per revolution of the output shaft (60EE$_h$ (Driving Shaft Revolutions).

The feed constant is useful for specifying the lead screw pitch for a linear axis and is used if the unit is based on length dimensions or if it is dimensionless.

6.3.5 Calculation formulas for user units

6.3.5.1 Position unit

Object 60A8$_h$ contains:

- Bits 16 to 23: The position unit (see chapter Units)
- Bits 24 to 31: The exponent of a power of ten (see chapter Units)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Factor								Unit							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved (00h)								reserved (00h)							

Example

If 60A8$_h$ is written with the value "FF410000$_h$" (bits 16-23=41$_h$ and bits 24-31=FF$_h$), the unit is set to *tenths of degree* (factory setting).

With a relative target position (607A$_h$) of 3600, the motor moves exactly one mechanical revolution, if Gear ratio is 1:1. The Feed constant plays no role in this case.

Example

If 60A8$_h$ is written with the value "FD010000$_h$" (bits 16-23=01$_h$ and bits 24-31=FD$_h$(=-3)), the unit is set to *millimeter*.

With a relative target position (607A$_h$) of 1, the motor moves exactly one mechanical revolution, if Feed constant and Gear ratio are 1:1.

If the Feed constant is set according to the lead screw pitch of a linear axis, the motor turns far enough that a feed of 1 mm is achieved.

Described in chapter Assignment of the feedbacks to the control loops is how you can determine which encoder/sensor is to be used for position control and measurement.

6.3.5.2 Speed unit

Object 60A9$_h$ contains:

- Bits 8 to 15: The time unit (see chapter Units)
- Bits 16 to 23: The position unit (see chapter Units)
- Bits 24 to 31: The exponent of a power of ten (see chapter Units)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Factor								Nominator (Position)							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Denominator (Time)								reserved (00h)							

Example

If 60A9$_h$ is written with the value "00B44700$_h$" (bits 8-15=00$_h$, bits 16-23=B4$_h$ and bits 24-31=47$_h$), the unit is set to *revolutions per minute* (factory setting).

Example

If 60A9$_h$ is written with the value "FD010300$_h$" (bits 8-15=FD$_h$(=-3), bits 16-23=01$_h$ and bits 24-31=03$_h$), the unit is set to *millimeters per second*.

Described in chapter Assignment of the feedbacks to the control loops is how you can determine which encoder/sensor is to be used for speed control and measurement.

NOTICE

The speed unit in Velocity mode is preset to *revolutions per minute*. You can only set the unit via the 604Ch VI Dimension Factor.

Conversion factor for the speed unit

You can set an additional factor for the speed unit. Thus, a unit of, e.g., 1/3 revolutions/minute is possible. The factor n is calculated from the factor for numerator (6096$_h$:01$_h$) divided by the factor for denominator (6096$_h$:02$_h$).

$$n_{velocity} = \frac{6096_h{:}01}{6096_h{:}02}$$

6.3.5.3 Acceleration unit

The acceleration unit is speed unit per second.

Conversion factor for the acceleration unit

The factor n for the acceleration unit is calculated from the numerator (6097$_h$:01$_h$) divided by the denominator (6097$_h$:02$_h$).

$$n_{acceleration} = \frac{6097_h{:}01}{6097_h{:}02}$$

6.3.5.4 Jerk unit

The jerk unit is Acceleration unit per second.

Conversion factor for jerk

The factor n for the jerk is calculated from the numerator (60A2$_h$:01$_h$) divided by the denominator (60A2$_h$:02$_h$).

$$n_{jerk} = \frac{60A2_h{:}01}{60A2_h{:}02}$$

6.4 Limitation of the range of motion

The digital inputs can be used as limit switches, as is described in chapter Digital inputs, if you activate this function for the inputs. The controller also supports software limit switches.

6.4.1 Behavior upon reaching the limit switch

If a limit switch is triggered, the limit switch position is stored internally, bit 7 (*Warning*) in 6041_h (*statusword*) is set and the action stored in object 3701_h is executed (see following table).

Value in object 3701_h	Description
-2	No reaction, discard the limit switch position
-1 (factory settings)	No reaction (e. g., to execute a homing operation) except noting the limit switch position
0	Switch off driver without deceleration ramp; drive function blocked – motor can turn freely (*Switch on disabled* state)
1	Braking with *slow down ramp* (deceleration ramp depending on operating mode) and subsequent state change to *Switch on disabled*
2	Braking with *quick stop ramp* and subsequent state change to *Switch on disabled*
5	Braking with *slow down ramp* (deceleration ramp depending on operating mode) and subsequent state change to *Quick stop active*; control does not switch off and the motor remains energized. You can switch back to the *Operation enabled* state.
6	Braking with *quick stop ramp* and subsequent state change to *Quick Stop Active*; control does not switch off and the motor remains energized. You can switch back to the *Operation enabled* state.

Continued travel behind the limit switch position is prevented provided the value in 3701_h is not "-1" or "-2". In any case, it is possible to move in the opposite direction.

If the value "-2" is used, bit 7 in 6041_h (Warning) is deleted as soon as the limit switches no longer trigger. Otherwise, it is not deleted until the internally noted limit switch position has been returned to.

NOTICE

To avoid automatically returning from the *Quick stop active* state to *Operation enabled* when using options "5" or "6" — the quick-stop bit (bit 2) in 6040_h is not used upon triggering of the limit switches — a change of the quick-stop bit from "0" to "1" is expected in order to changed back to the *Operation enabled* state.

Discarding the limit switch position

NOTICE

It is necessary to discard the limit switch positions if both limit switches were actuated simultaneously or the movement range is dynamically limited by a shifting of the limit switches.

To delete internally stored limit switch positions in the event of triggering and to release or clear the limit switches, briefly set object 3701_h to "-2".

If, when using the values "5" or "6" in 3701_h, the state of the *State Machine* is *Quick stop active* and the motor is to remain energized, proceed as follows to avoid an automatic change to the *Switch on disabled* state:

1. Use a rising edge of bit 2 (quick stop) in 6040_h to switch back to the *Operation enabled* state without, however, starting a movement (set bit 4 in 6040_h to 0 or target speed or target torque to "0").
2. Set 3701_h to "-2" .
3. Release the limit switch again.
4. Reset 3701_h back to "5" or "6".

6.4.2 Software limit switches

The controller takes into account software limit switches ($607D_h$ (Software Position Limit)). Target positions ($607A_h$) are limited by $607D_h$; the absolute target position may not be larger than the limits in $607D_h$. If the motor is located outside of the permissible range when setting up the limit switches, only travel commands in the direction of the permissible range are accepted.

6.5 Cycle times

The controller operates with a cycle time of 1 ms. This means that data are processed every 1 ms; multiple changes to a value (e.g., value of an object or level at a digital input) within one ms cannot be detected.

The following table includes an overview of the cycle times of the various processes.

Task	Cycle time
Application	1 ms
NanoJ application	1 ms
Current controller	62.5 µs (16 kHz)
Velocity controller	250 µs (4 kHz)
Position controller	1 ms

7 Operating modes

7.1 Profile Position

7.1.1 Overview

7.1.1.1 Description

Profile Position Mode is used to move to positions relative to the last target position or to an absolute position (last reference position). During the movement, the limit values for the speed, starting acceleration/ braking deceleration and jerks are taken into account.

7.1.1.2 Activation

To activate the mode, the value "1" must be set in object 6060_h (Modes Of Operation) (see "CiA 402 Power State Machine").

7.1.1.3 Controlword

The following bits in object 6040_h (controlword) have a special function:

- Bit 4 starts a travel command. This is carried out on a transition from "0" to "1". An exception occurs if changing from another operating mode to *profile position*: If bit 4 is already set, it does not need to be set to "0" and then back to "1" in order to start the travel command.
- Bit 5: If this bit is set to "1", a travel command triggered by bit 4 is immediately executed. If it is set to "0", the just executed travel command is completed and only then is the next travel command started.
- Bit 6: With "0", the target position ($607A_h$) is absolute and with "1" the target position is relative. The reference position is dependent on bits 0 and 1 of object $60F2_h$.
- Bit 8 (Halt): If this bit is set to "1", the motor stops. On a transition from "1" to "0", the motor accelerates with the set start ramp to the target speed. On a transition from "0" to "1", the motor brakes and comes to a standstill. The braking deceleration is dependent here on the setting of the "Halt Option Code" in object $605D_h$.
- Bit 9 (Change on setpoint): If this bit is set, the speed is not changed until the first target position is reached. This means that, before the first target is reached, no braking is performed, as the motor should not come to a standstill at this position.

Controlword 6040_h		
Bit 9	**Bit 5**	**Definition**
X	1	The new target position is moved to immediately.
0	0	Positioning is completed before moving to the next target position with the new limits.
1	0	The current target position is only passed through; afterwards, the new target position is moved to with the new values.

For further information, see figure in "Setting travel commands".

NOTICE

Bit 9 in the controlword is ignored if the ramp speed is not met at the target point. In this case, the controller would need to reset and take a run-up to reach the preset.

7.1.1.4 Statusword

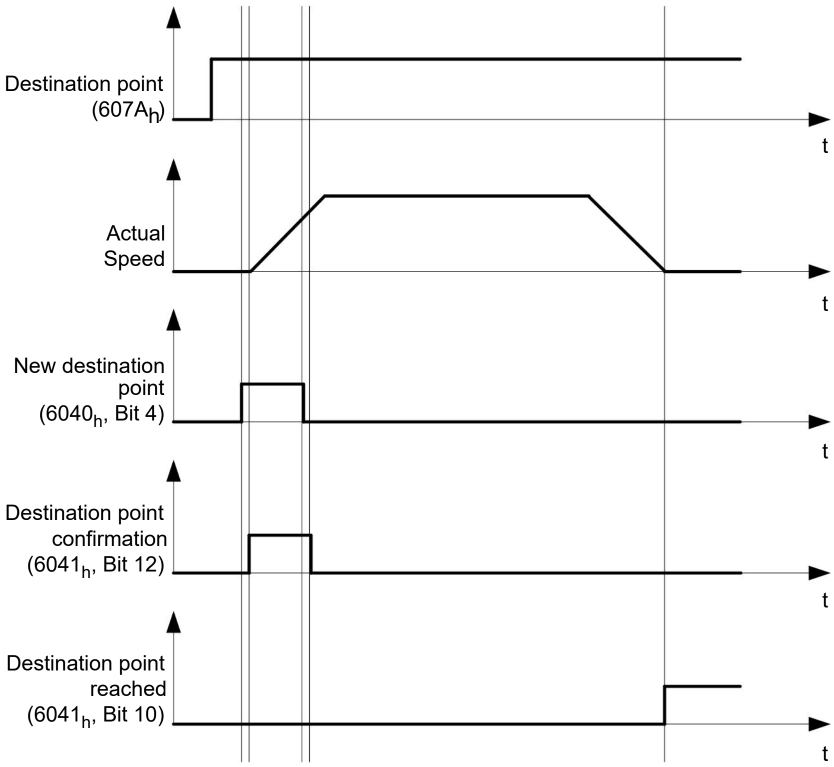
The following bits in object 6041_h (statusword) have a special function:

- Bit 10 (Target Reached): This bit is set to "1" if the last target was reached and the motor remains within a tolerance window (6067$_h$) for a preset time (6068$_h$). The bit is also set to "1" if the halt bit (bit 8) in 6040$_h$ has been set and as soon as the motor is at a standstill.
- Bit 11: Limit exceeded: The demand position is above or below the limit values set in 607D$_h$.
- Bit 12 (Set-point acknowledge): This bit confirms receipt of a new and valid set point. It is set and reset in sync with the "New set-point" bit in the controlword.
  There is an exception in the event that a new movement is started before another one has completed and the next movement is not to occur until after the first one has finished. In this case, the bit is reset if the command was accepted and the controller is ready to execute new travel commands. If a new travel command is sent even though this bit is still set, the newest travel command is ignored.
  The bit is not set if one of the following conditions is met:
  - The new target position can no longer be reached while adhering to all boundary conditions.
  - A target position was already traveled to and a target position was already specified. A new target position can only be specified after the current positioning has been concluded.
- Bit 13 (Following Error): This bit is set in *closed loop* mode if the following error is greater than the set limits (6065$_h$ (Following Error Window) and 6066$_h$ (Following Error Time Out)).

7.1.2 Setting travel commands

7.1.2.1 Travel command

In object 607A$_h$ (Target Position), the new target position is specified in user units (see User-defined units). The travel command is then triggered by setting bit 4 in object 6040$_h$ (controlword). If the target position is valid, the controller responds with bit 12 in object 6041$_h$ (statusword) and begins the positioning move. As soon as the position is reached, bit 10 in the statusword is set to "1".



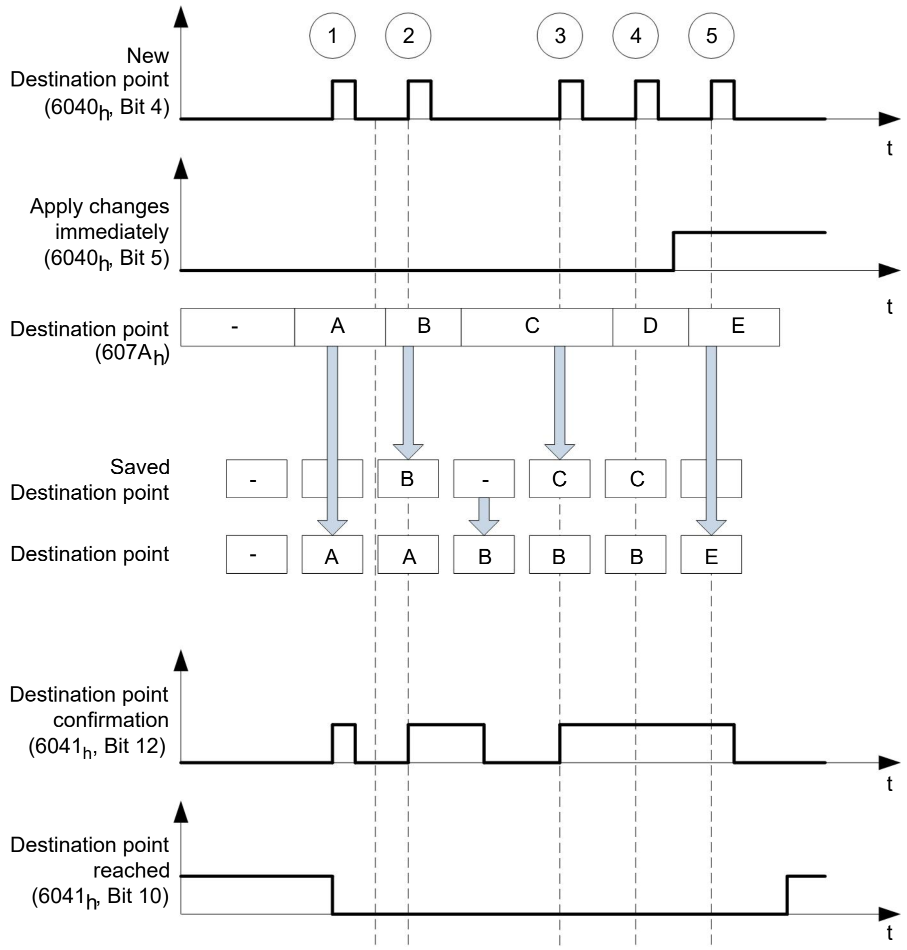
The controller can also reset bit 4 in object 6040$_h$ (controlword) on its own. This is set with bits 4 and 5 of object 60F2$_h$.

7.1.2.2 Other travel commands

Bit 12 in object 6041$_h$ (statusword, set-point acknowledge) changes to "0" if another travel command can be buffered (see time 1 in the following figure). As long as a target position is being moved to, a second target position can be passed to the controller in preparation. All parameters – such as speed, acceleration, braking deceleration, etc. – can thereby be reset (time 2). If the buffer is empty, the next time can be queued up (time 3).

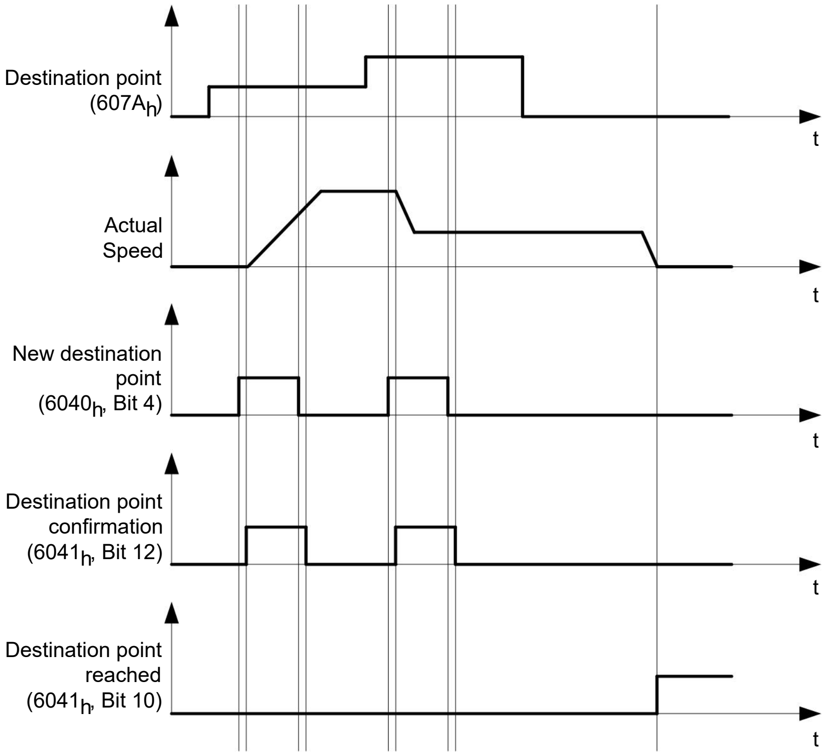
If the buffer is already full, a new set point is ignored (time 4). If bit 5 in object 6040_h (controlword, bit: "Change Set-Point Immediately") is set, the controller operates without the buffer; new travel commands are implemented directly (time 5).

Times



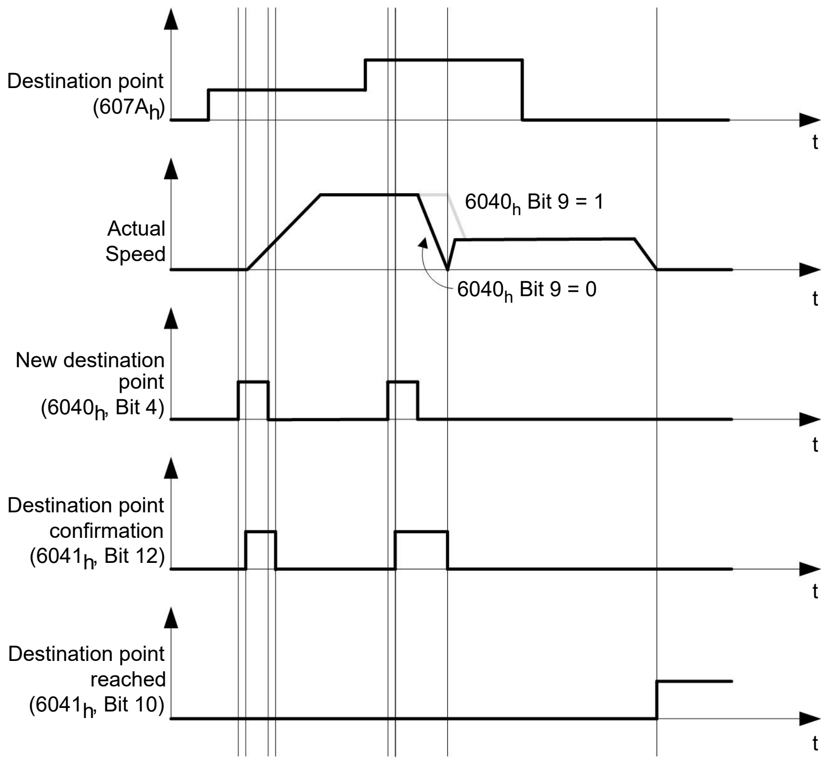
Transition procedure for second target position

The following graphic shows the transition procedure for the second target position while moving to the first target position. In this figure, bit 5 of object 6040_h (controlword) is set to "1"; the new target value is, thus, taken over immediately.



Possibilities for moving to a target position

If bit 9 in object 6040h (controlword) is equal to "0", the current target position is first moved to completely. In this example, the final speed (6082h) of the target position is equal to zero. If bit 9 is set to "1", the profile speed (6081h) is maintained until the target position is reached; only then do the new boundary conditions apply.

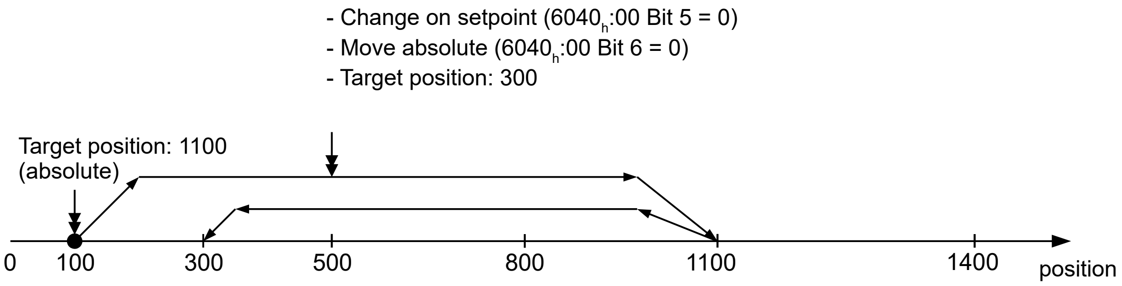


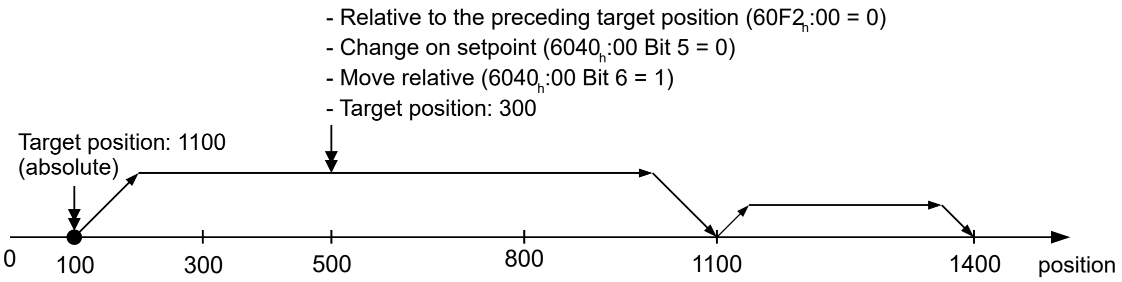
Possible combinations of travel commands

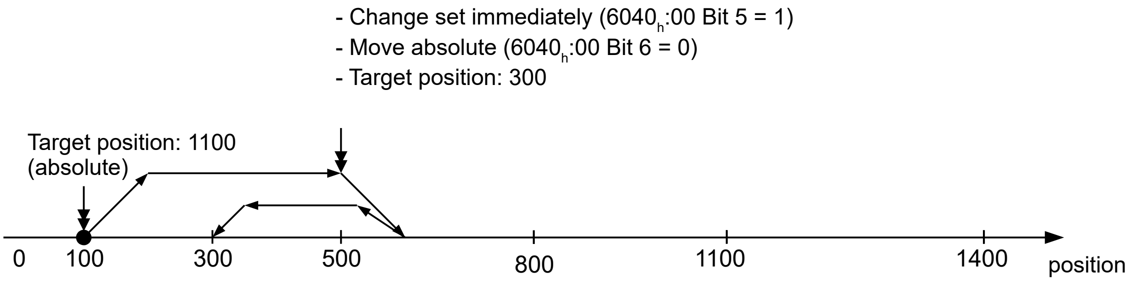
To provide a better overview of the travel commands, combinations of travel commands are listed and depicted in this chapter.

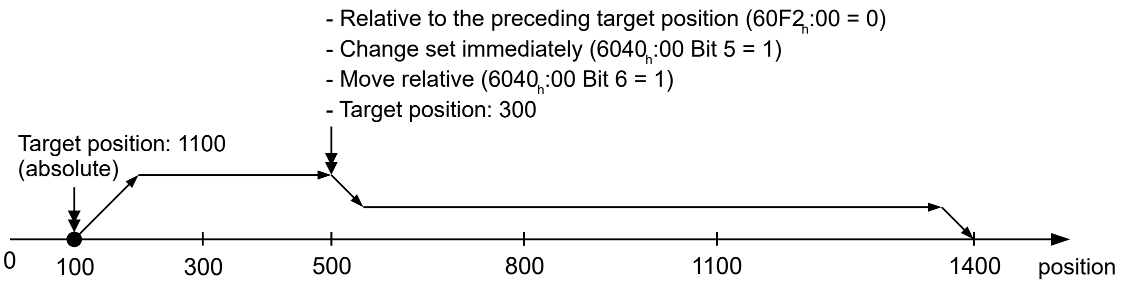
The following applies for the figures below:

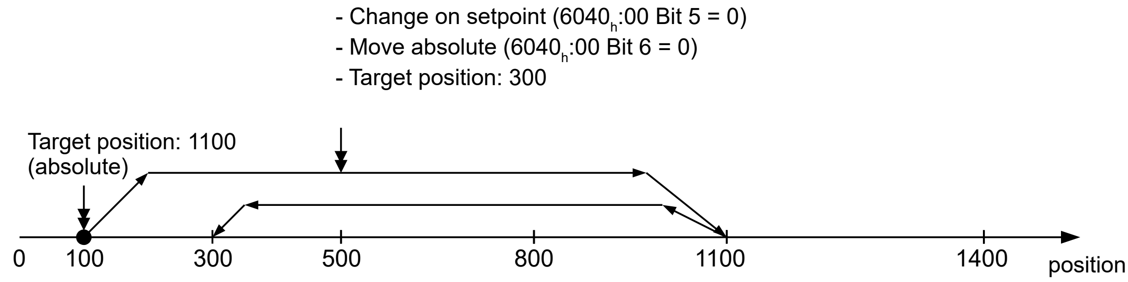
- A double arrow indicates a new travel command.
- The first travel command at the start is always an absolute travel command to position 1100.
- The second movement is performed at a lower speed so as to present the graphs in a clear manner.

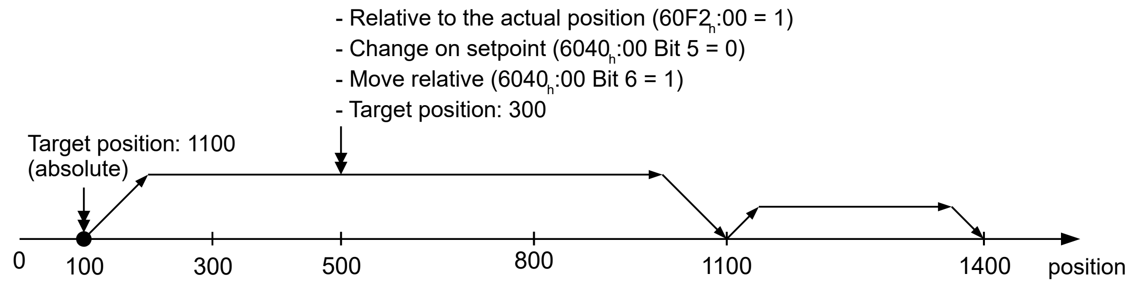


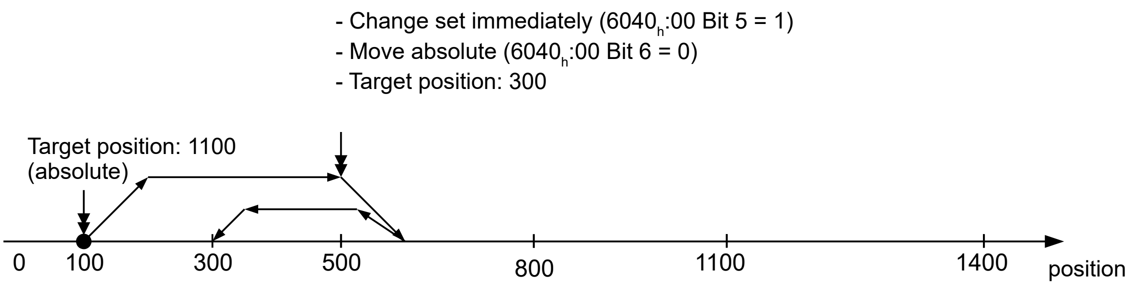






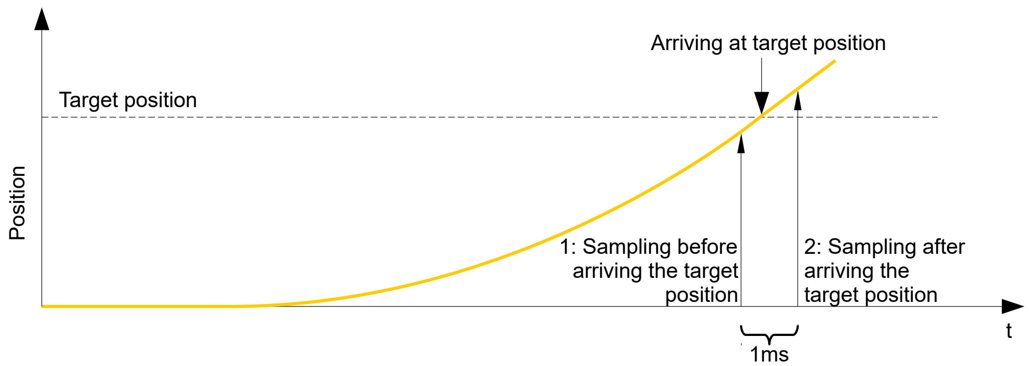






7.1.3 Loss of accuracy for relative movements

When linking together relative movements, a loss of accuracy may occur if the final speed is not set to zero. The following graphic illustrates the reason.



The current position is sampled once per millisecond. It is possible that the target position is reached between two samples. If the final speed is not equal to zero, then, after the target position is reached, the sample is used as an offset as the basis for the subsequent movement. As a result, the subsequent movement may go somewhat farther than expected.

7.1.4 Boundary conditions for a positioning move

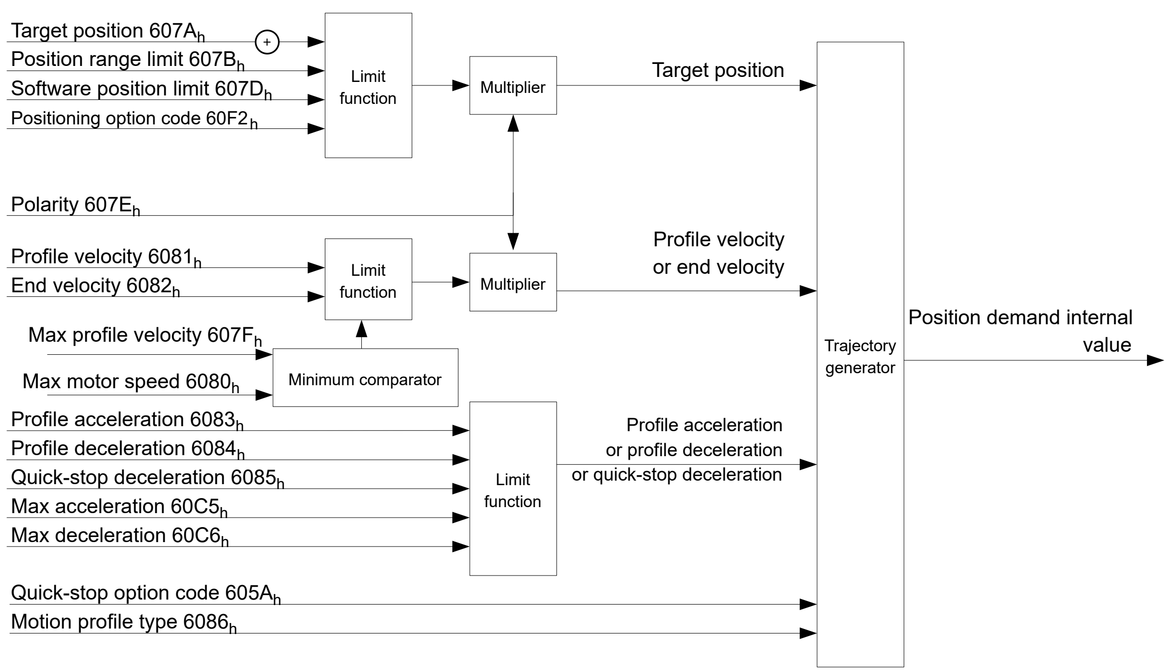
7.1.4.1 Object entries

The boundary conditions for the position that has been moved to can be set in the following entries of the object dictionary:

- $607A_h$: (Target Position): Planned target position
- $607D_h$: (Software Position Limit): Definition of the limit stops (see chapter Software limit switches)
- $607C_h$ (Home Offset): Specifies the difference between the zero position of the controller and the reference point of the machine in user-defined units. (See "Homing")
- $607B_h$ (Position Range Limit): Limits of a modulo operation for replicating an endless rotation axis
- 607_h (Polarity): Direction of rotation
- 6081_h (Profile Velocity): Maximum speed with which the position is to be approached
- 6082_h (End Velocity): Speed upon reaching the target position
- 6083_h (Profile Acceleration): Desired starting acceleration
- 6084_h (Profile Deceleration): Desired braking deceleration
- 6085_h (Quick Stop Deceleration): Emergency-stop braking deceleration in case of the "Quick stop active" state of the "CiA 402 Power State Machine"
- 6086_h (Motion Profile Type): Type of ramp to be traveled; if the value is "0", the jerk is not limited; if the value is "3", the values of $60A4_h$:1_h–4_h are set as limits for the jerk.
- $60C5_h$ (Max Acceleration): The maximum acceleration that may not be exceeded when moving to the end position
- $60C6_h$ (Max Deceleration): The maximum braking deceleration that may not be exceeded when moving to the end position
- $60A4_h$ (Profile Jerk), subindex 01_h to 04_h: Objects for specifying the limit values for the jerk.
- The speed is is limited by $607F_h$ (Max Profile Velocity) and 6080_h (Max Motor Speed); the smaller value is used as the limit.
- $60F2_h$: (Positioning Option Code): Defines the positioning behavior

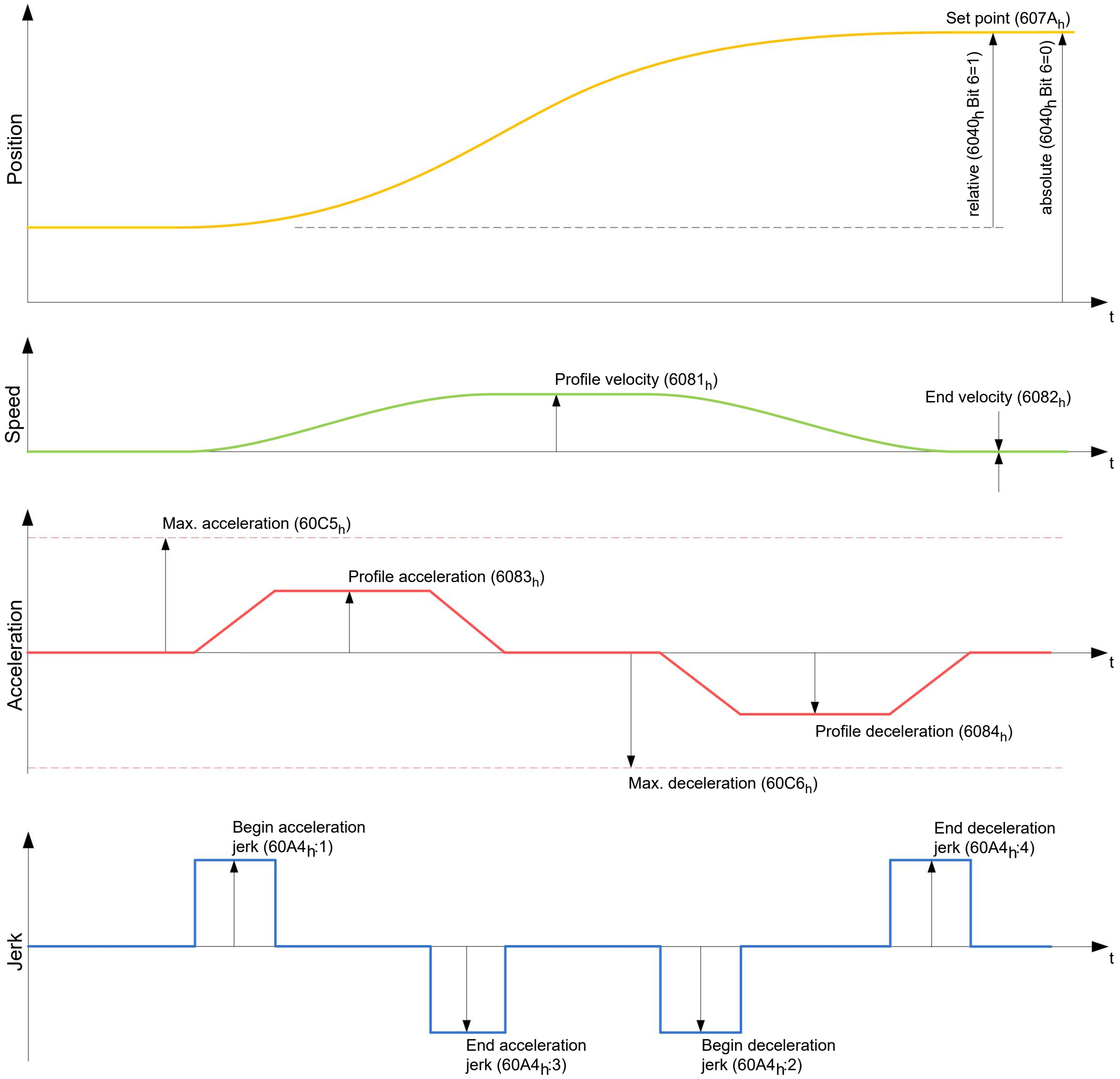
7.1.4.2 Objects for the positioning move

The following graphic shows the objects involved in the boundary conditions of the positioning move.



7.1.4.3 Parameters for the target position

The following graphic shows an overview of the parameters that are used for moving to a target position (figure not to scale).



7.1.5 Jerk-limited mode and non-jerk-limited mode

7.1.5.1 Description

A distinction is made between the "jerk-limited" and "non-jerk-limited" modes.

7.1.5.2 Jerk-limited mode

Jerk-limited positioning can be achieved by setting object 6086_h to "3". The entries for the jerks in subindices $:1_h$–4_h of object 60A4 thereby become valid.

7.1.5.3 Non-jerk-limited mode

A "non-jerk-limited" ramp is traveled if the entry in object 6086_h is set to "0" (default setting).

7.2 Velocity

7.2.1 Description

This mode operates the motor at a preset target speed, similar to a frequency inverter. Unlike the *profile velocity mode*, this mode does not permit the selection of jerk-limited ramps.

7.2.2 Activation

To activate the mode, the value "2" must be set in object 6060_h (Modes Of Operation) (see "CiA 402 Power State Machine").

7.2.3 Controlword

The following bits in object 6040_h (controlword) have a special function:

- Bit 8 (Halt): If this bit is set to "1", the motor stops. On a transition from "1" to "0", the motor accelerates with the acceleration ramp to the target speed. On a transition from "0" to "1", the motor brakes according to the deceleration ramp and comes to a standstill.

7.2.4 Statusword

The following bits in object 6041_h (statusword) have a special function:

- Bit 11: Limit exceeded: The target speed is above or below the set limit values.

7.2.5 Object entries

The following objects are necessary for controlling this mode:

- $604C_h$ (Dimension Factor):
  The unit for speed values is defined here for the following objects.
  Subindex 1 contains the denominator (multiplier) and subindex 2 contains the numerator (divisor) with which the internal speed values are converted to revolutions per minute. If, for example, subindex 1 is set to the value "60" and subindex 2 is set to the value "1", the speed is specified in revolutions per second (60 revolutions per 1 minute).
- 6042_h: Target Velocity.
  The target speed is set here in user-defined units.
- 6048_h: Velocity Acceleration
  This object defines the acceleration. Subindex 1 contains the change in speed, subindex 2 the corresponding time in seconds. Both together are used to calculate the acceleration:

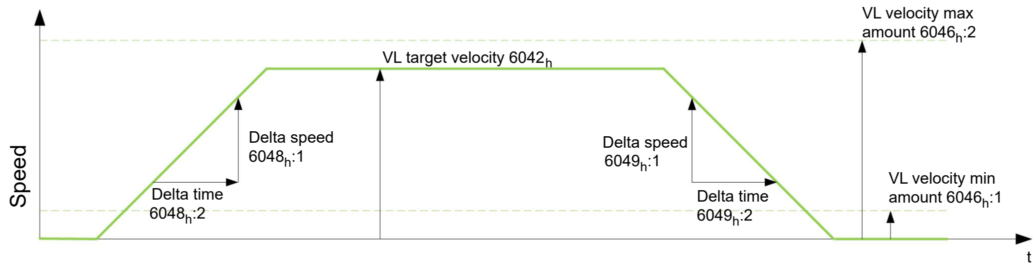
$$\text{VL velocity acceleration} = \frac{\text{Delta speed } (6048_h{:}1)}{\text{Delta time } (6048_h{:}2)}$$

- 6049_h (Velocity Deceleration):
  This object defines the deceleration (deceleration ramp). The subindices here are arranged as described in object 6048_h; the change in speed is to be specified with positive sign.
- 6046_h (Velocity Min Max Amount):
  The limitations of the target speeds are specified in this object.
  The minimum speed is set in $6046_h{:}1_h$. If the target speed (6042_h) falls below the minimum speed, the value is limited to the minimum speed $6046_h{:}1_h$.
  The maximum speed is set in $6046_h{:}2_h$. If the target speed (6042_h) exceeds the maximum speed, the value is limited to the maximum speed $6046_h{:}2_h$.
- $604A_h$ (Velocity Quick Stop):
  This object can be used to set the quick-stop ramp. Subindices 1 and 2 are identical to those described for object 6048_h.

The following objects can be used to check the function:

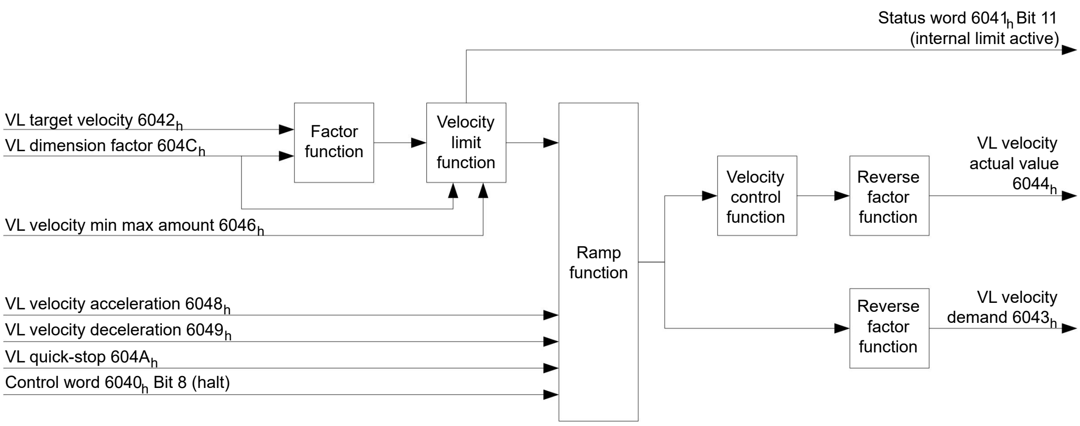
- 6043_h (Vl Velocity Demand)
- 6044_h (Vl Velocity Actual Value)

7.2.5.1 Speeds in Velocity Mode



7.2.5.2 Objects for Velocity Mode

The ramp generator follows the target speed, remaining within the set speed and acceleration limits. As long as a limit is active, bit 11 in object 6041_h is set (internal limit active).



7.3 Profile Velocity

7.3.1 Description

This mode operates the motor in Velocity Mode with extended (jerk-limited) ramps. Unlike *Velocity Mode* (see "Velocity"), the statusword is used in this mode to indicate whether the target speed is reached.

7.3.2 Activation

To activate the mode, the value "3" must be set in object 6060_h (Modes Of Operation) (see "CiA 402 Power State Machine").

7.3.3 Controlword

The following bits in object 6040_h (controlword) have a special function:

- Bit 8 (Halt): If this bit is set to "1", the motor stops. On a transition from "1" to "0", the motor accelerates with the set start ramp to the target speed. On a transition from "0" to "1", the motor brakes and comes to a standstill.

7.3.4 Statusword

The following bits in object 6041_h (statusword) have a special function:

- Bit 10 (target speed reached; Target Reached): In combination with bit 8 in the controlword, this bit specifies whether the target speed is reached, if braking is taking place or if the motor is at a standstill (see table).

6041$_h$ Bit 10	6040$_h$ Bit 8	Description
0	0	Target speed not reached
0	1	Axis braking
1	0	Target speed within target window (defined in 606D$_h$h and 606E$_h$)
1	1	Axis speed is 0

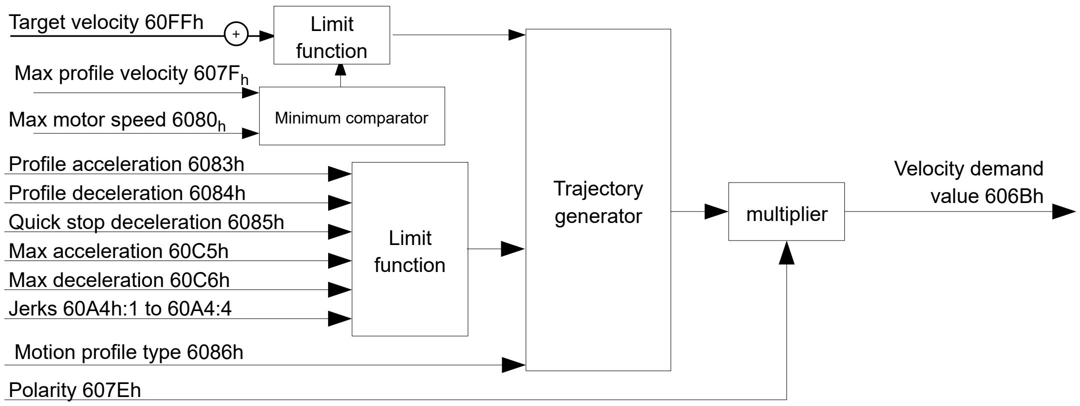
- Bit 12: This bit indicates whether the actual speed is zero.
  If the actual speed is greater than the value in 606F$_h$(Velocity Threshold) for a time of 6070$_h$(Velocity Threshold Time), this bit has the value "0". The bit otherwise remains set to "1".
- Bit 13 (Deviation Error): This bit is set in *closed loop* mode if the slippage error is greater than the set limits (60F8h Max Slippage and 203Fh Max Slippage Time Out).

7.3.5 Object entries

The following objects are necessary for controlling this mode:

- 606B$_h$ (Velocity Demand Value):
  This object contains the output of the ramp generator, which simultaneously serves as the preset value for the velocity controller.
- 606C$_h$ (Velocity Actual Value):
  Indicates the current actual speed.
- 606D$_h$ (Velocity Window):
  This value specifies by how much the actual speed may vary from the set speed for bit 10 (target speed reached; Target Reached") in object 6041$_h$ (statusword) to be set to "1".
- 606E$_h$ (Velocity Window Time):
  This object specifies how long the actual speed and the set speed must be close to one another (see 606D$_h$ "Velocity Window") for bit 10 "Target speed reached" in object 6041$_h$ (statusword) to be set to "1".
- 607E$_h$ (Polarity):
  If bit 6 is set to "1" here, the sign of the target speed is reversed.
- 6083$_h$ (Profile acceleration):
  Sets the value for the acceleration ramp.
- 6084$_h$ (Profile Deceleration):
  Sets the value for the deceleration ramp.
- 6085$_h$ (Quick Stop Deceleration):
  Sets the value for the deceleration ramp for rapid braking.
- 6086$_h$ (Motion Profile Type):
  The ramp type can be selected here ("0" = trapezoidal ramp, "3" = jerk-limited ramp).
- 60FF$_h$ (Target Velocity):
  Specifies the target speed that is to be reached.
- The speed is is limited by 607F$_h$ (Max Profile Velocity) and 6080$_h$ (Max Motor Speed); the smaller value is used as the limit.

7.3.5.1 Objects in Profile Velocity Mode

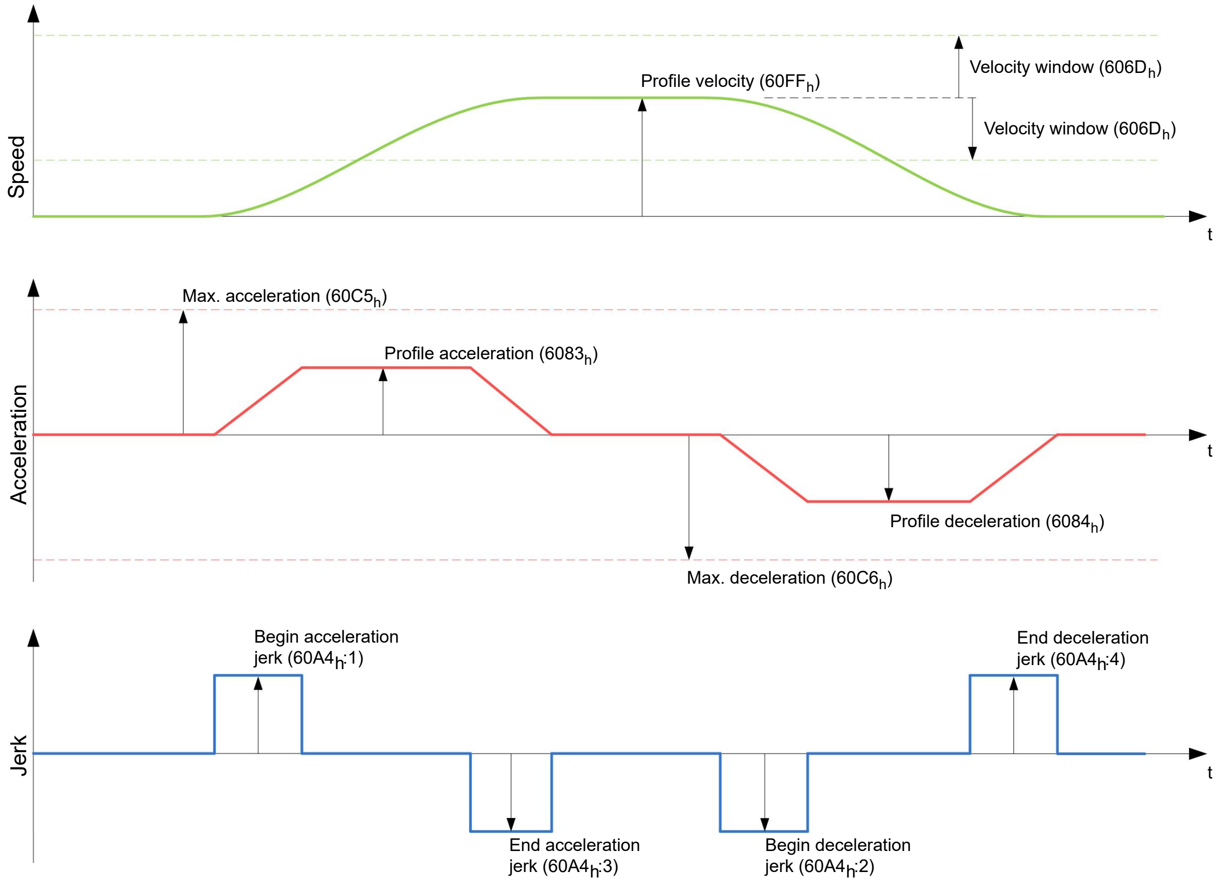


7.3.5.2 Activation

After the mode is selected in object 6060$_h$ (Modes Of Operation) and the "Power State machine" (see "CiA 402 Power State Machine") is switched to *Operation enabled*, the motor is accelerated to the target speed in object 60FF$_h$ (see following figures). The speed and acceleration values are taken into account here; for jerk-limited ramps, the jerk-limit values are also taken into account.

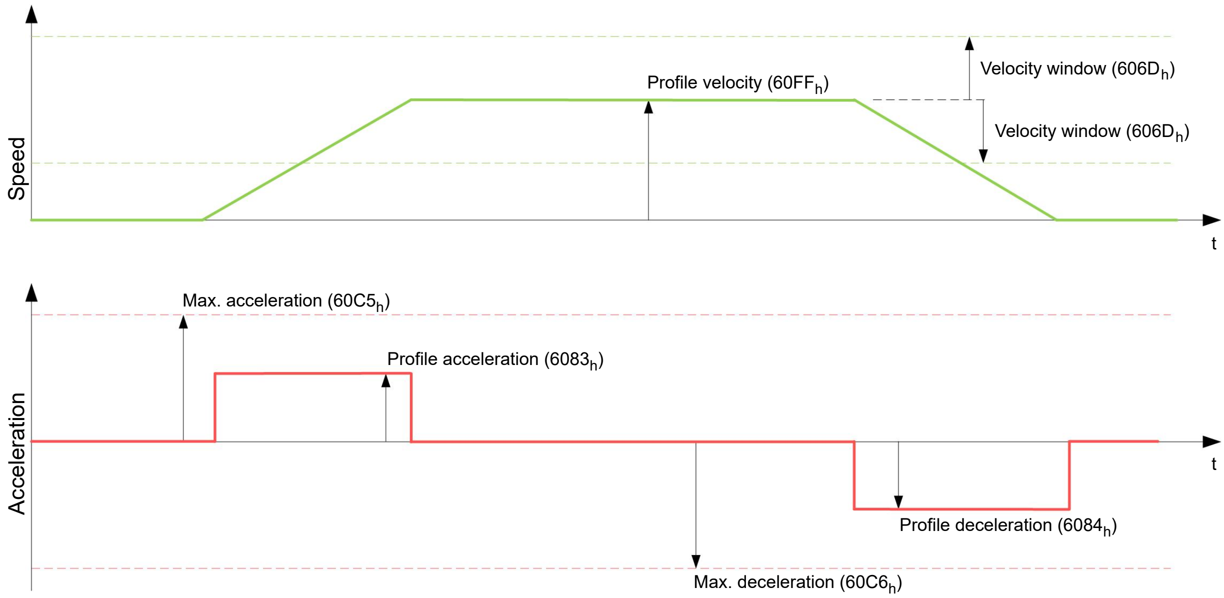
7.3.5.3 Limitations in the jerk-limited case

The following figure shows the adjustable limits in the jerk-limited case (6086$_h$ = 3).



7.3.5.4 Limitations in the trapezoidal case

This figure shows the adjustable limitations for the trapezoidal case (6086_h = 0).



7.4 Profile Torque

7.4.1 Description

In this mode, the torque is preset as a set value and reached via a ramp function.

7.4.2 Activation

To activate the mode, the value "4" must be set in object 6060_h (Modes Of Operation) (see "CiA 402 Power State Machine").

7.4.3 Controlword

The following bits in object 6040_h (controlword) have a special function:

- Bit 8 (Halt): If this bit is set to "1", the motor stops. If this bit is set from "1" to "0", the motor is started up according to the presets. When setting from "0" to "1", the motor is again brought to a standstill, taking the preset values into consideration.

7.4.4 Statusword

The following bits in object 6041_h (statusword) have a special function:

- Bit 10 (Target Reached): In combination with bit 8 of object 6040_h (controlword), this bit indicates whether the specified torque is reached (see following table). The target is considered having been met if the current torque (6077h Torque Actual Value) is within a tolerance window (203Dh Torque Window) for a specified time (203Eh Torque Window Time Out).

6040_h Bit 8	6041_h Bit 10	Description
0	0	Specified torque not reached
0	1	Specified torque reached
1	0	Axis brakes
1	1	Axis speed is 0

- Bit 11: Limit exceeded: The target torque (6071_h) exceeds the maximum torque entered in 6072_h.

7.4.5 Object entries

All values of the following entries in the object dictionary are to be specified as a thousandth of the maximum torque, which corresponds to the rated current ($203B_h$:01_h) . This includes the objects:

- 6071_h (Target Torque):
  Target torque
- 6072_h (Max Torque):
  Maximum torque during the entire ramp (accelerate, maintain torque, decelerate)
- 6073_h (Max Current):
  Maximum current. The minimum of 6073_h and 6072_h is used as limit for the torque in 6071_h.
- 6074_h (Torque Demand):
  Current output value of the ramp generator (torque) for the controller
- 6087_h (Torque Slope):
  Max. change in torque per second

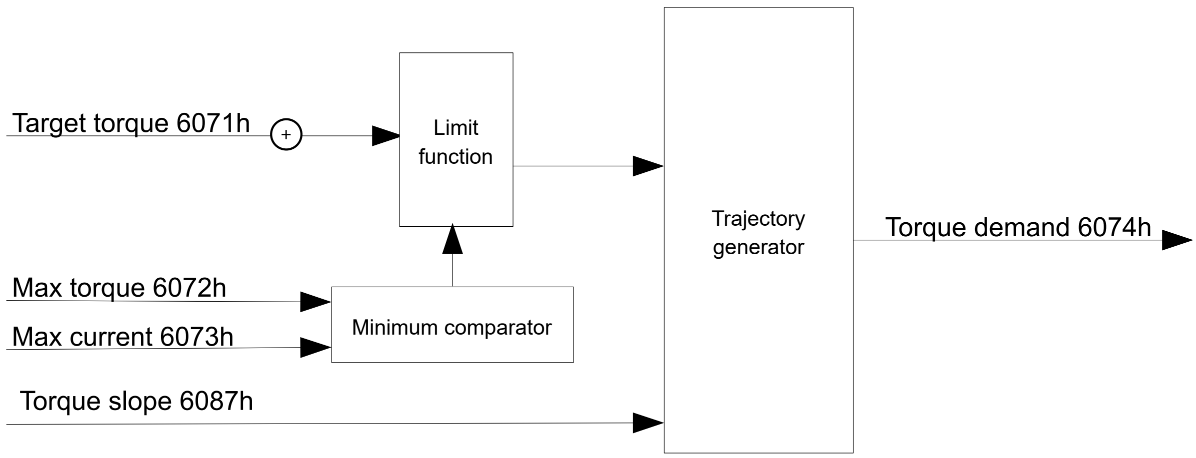
NOTICE

These values are not limited to 100% of the rated current ($203B_h$:01_h). Torque values greater than the rated torque (generated from the rated current) can be achieved if the maximum duration ($203B_h$:02_h) of the maximum current (6073_h) is set (see I2t Motor overload protection). All torque objects are limited by the maximum motor current (2031_h).

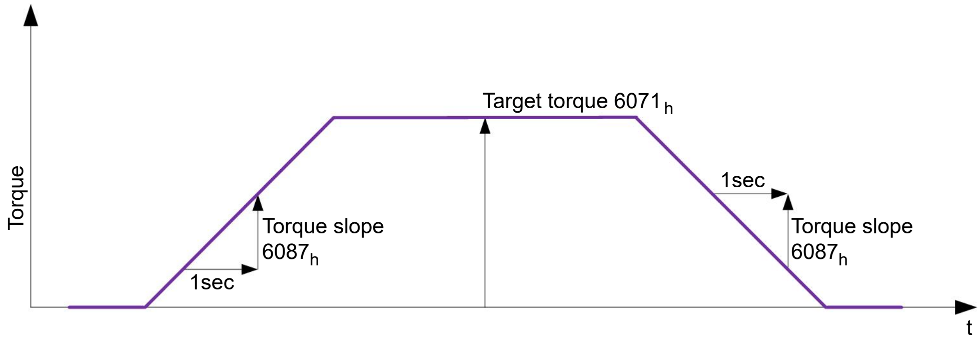
The following objects are also needed for this operating mode:

- 3202_h Bit 5 (Motor Drive Submode Select):
  If this bit is set to "0", the drive controller is operated in the torque-limited Velocity Mode, i.e., the maximum speed can be limited in object 6080_h and the controller can operate in field weakening mode.
  If this bit is set to "1", the controller operates in the ("Real") Torque Mode; the maximum speed cannot be limited here and field weakening mode is not possible.

7.4.5.1 Objects of the ramp generator



7.4.5.2 Torque curve



7.5 Homing

7.5.1 Overview

7.5.1.1 Description

The purpose of the homing method is to align the position zero point of the controller with an encoder index or position switch.

7.5.1.2 Activation

To activate the mode, the value "6" must be set in object 6060_h (Modes Of Operation) (see "CiA 402 Power State Machine").

TIP

If home switches and/or limit switches are used, these special functions must first be activated in the I/O configuration (see "Digital inputs and outputs").

To use the limit switch, you must also set object 3701_h to "-1" (factory setting) to prevent blocking the further travel of the motor.

7.5.1.3 Controlword

The following bits in object 6040_h (controlword) have a special function:

- Bit 4: If the bit is set to "1", referencing is started. This is performed until either the reference position is reached or bit 4 is reset to "0".

7.5.1.4 Statusword

The following bits in object 6041_h (statusword) have a special function:

Bit 13	Bit 12	Bit 10	Description
0	0	0	Homing is performed
0	0	1	Homing is interrupted or not started
0	1	0	Homing has been performed since the last restart but target is not currently reached
0	1	1	Homing completed
1	0	0	Error during homing, motor still turning
1	0	1	Error during homing, motor at standstill

NOTICE

Bit 12 in *Homing* mode is set to 1 after the first fully completed homing operation since the restart. It is only reset to 0

- during all subsequent homing operations
- in the event of an error during a homing operation (permanently deleted until a new homing operation is fully completed).

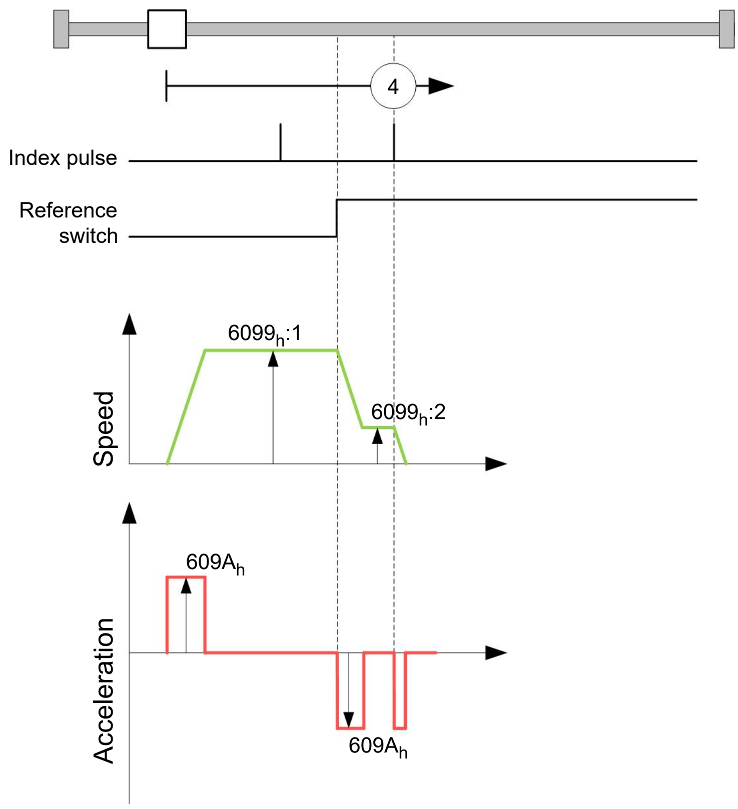
7.5.1.5 Object entries

The following objects are necessary for controlling this mode:

- $607C_h$ (Home Offset): Specifies the difference between the zero position of the controller and the reference point of the machine in user-defined units.
- 6098_h (Homing Method):
  Method to be used for referencing (see "Homing method")
- 6099_h:01_h (Speed During Search For Switch):
  Speed for the search of the switch
- 6099_h:02_h (Speed During Search For Zero):
  Speed for the search of the index
- 6080_h (Max Motor Speed): Maximum speed
- $609A_h$ (Homing Acceleration):
  Starting acceleration and braking deceleration for homing
- $203A_h$:01_h (Minimum Current For Block Detection):
  Minimum current threshold which, if exceeded, is to detect the blocking of the motor at a block.
- $203A_h$:02_h (Period Of Blocking):
  Specifies the time in ms that the motor is to continue to run against the block after block detection.

Homing speeds

The figure shows the homing speeds using method 4 as an example:



7.5.2 Homing method

7.5.2.1 Description

The homing method is written as a number in object 6098$_h$ and decides whether, on a switch edge (rising/falling), a current threshold for block detection or an index pulse is referenced or in which direction homing starts. Methods that use the index pulse of the encoder lie in the number range 1 to 14, 33 and 34. Methods that do not use the index pulse of the encoder lie between 17 and 30, but are identical to methods 1 to 14 with respect to the travel profiles. These number are shown in circles in the following figures. Methods for which no limit switches are used and, instead, travel against a block is to be detected, a minus must be placed before the method number when making the call.

In the following graphics, the negative movement direction is to the left. The *limit switch* is located before the respective mechanical block; the *home switch* is located between the two limit switches. The index pulses come from the connected encoder.

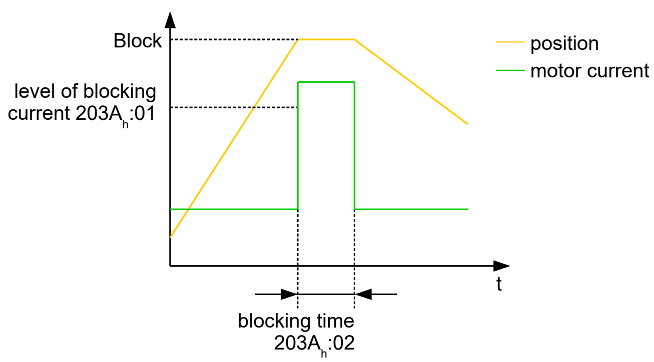
For methods that use homing on block, the same figures apply as for the methods with limit switch. Because nothing is different aside from the missing limit switches, the same figures are used. For the figures here, the limit switches must be replaced with a mechanical block.

7.5.2.2 Homing on block

Homing on block currently only functions in *closed loop* mode.

"Homing on block" functions like every homing method with the difference that instead of a limit switch, a block (limit stop) is used for positioning. Two settings are to be made here:

1. Current level: In object 203A$_h$:01, the current level is defined above which movement against the block is detected.
2. Blocking duration: In object 203A$_h$:02, the duration during which the motor moves against the block is set.



7.5.2.3 Overview of methods

Methods 1 to 14 as well as 33 and 34 use the index pulse of the encoder.

Methods 17 to 32 are identical to methods 1 to 14 with the difference that only limit or home switches are used for referencing and not the index pulse.

- Methods 1 to 14 use an index pulse.
- Methods 17 to 30 do not use an index pulse.
- Methods 33 and 34 reference only to the next index pulse.
- Method 35 references to the current position.

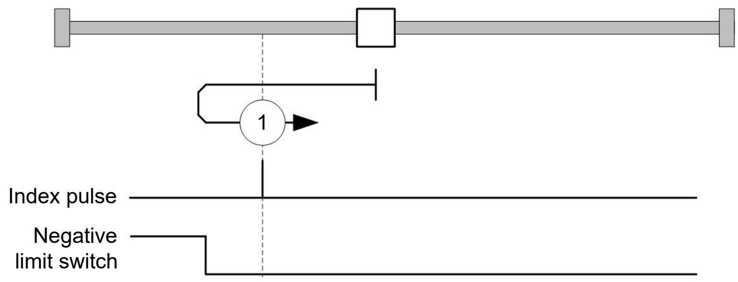
The following methods can be used for homing on block:

- Methods -1 to -2 and -7 to -14 contain an index pulse
- Methods -17 to -18 and -23 to -30 have no index pulse

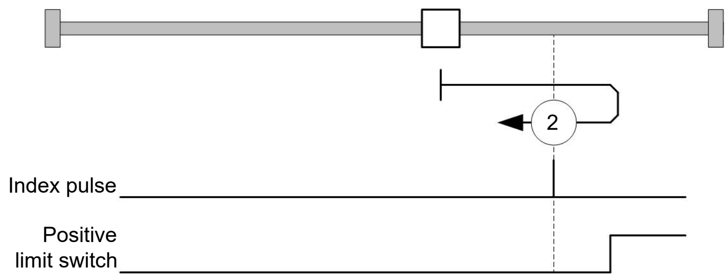
7.5.2.4 Methods 1 and 2

Reference to limit switches and index pulse.

Method 1 references to negative limit switch and index pulse:



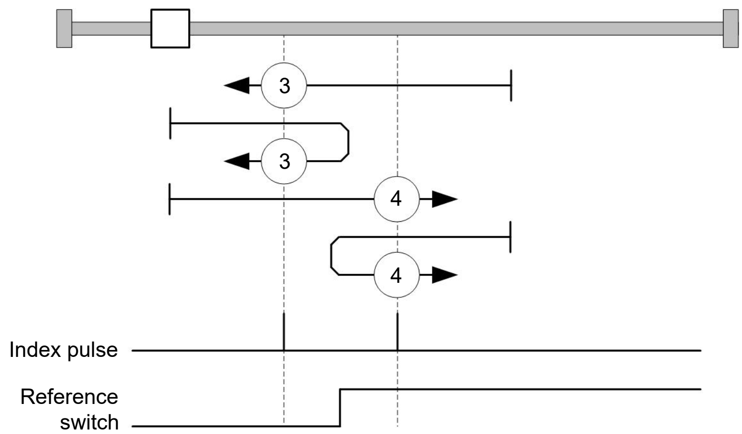
Method 2 references to positive limit switch and index pulse:



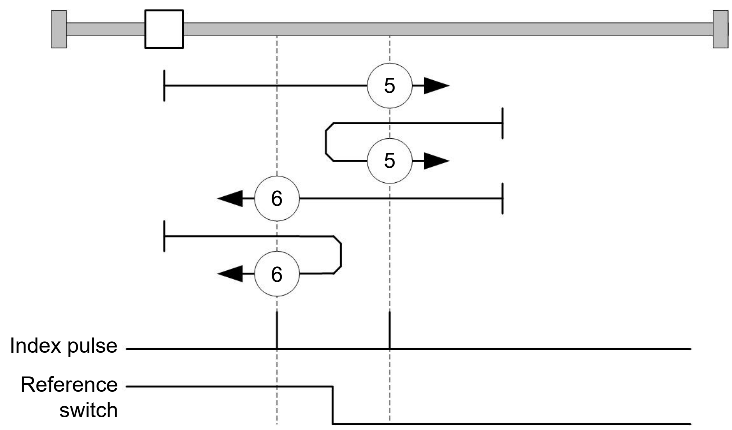
7.5.2.5 Methods 3 to 6

Reference to the switching edge of the home switch and index pulse.

With methods 3 and 4, the left switching edge of the home switch is used as reference:



With methods 5 and 6, the right switching edge of the home switch is used as reference:

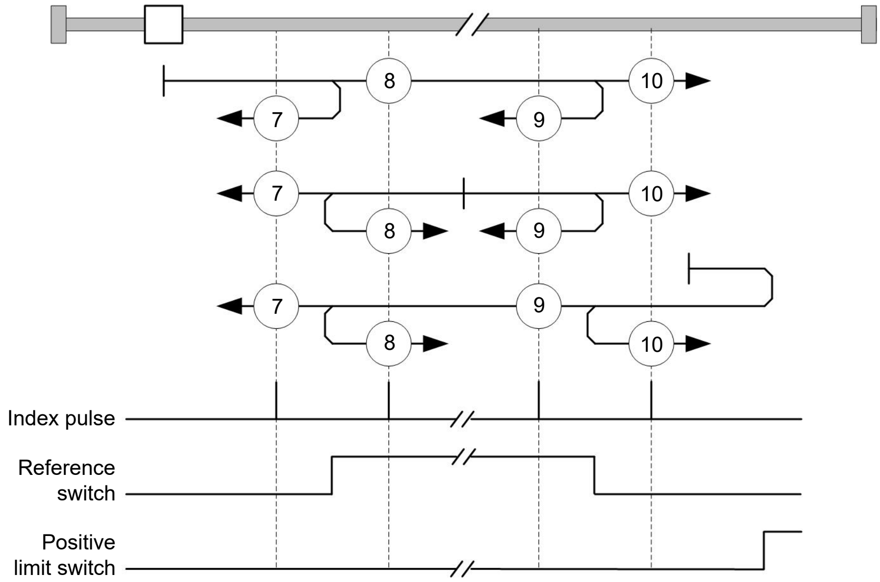


7.5.2.6 Methods 7 to 14

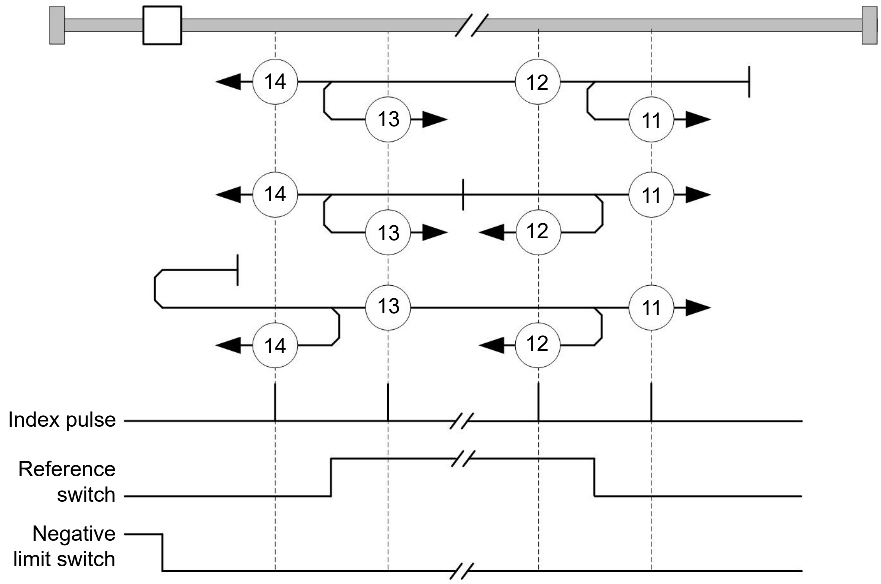
Reference to the home switch and index pulse (with limit switches).

With these methods, the current position relative to the home switch is not important. With method 10, for example, referencing is always performed to the index pulse to the right of the right edge of the home switch.

Methods 7 to 10 take the positive limit switch into account:



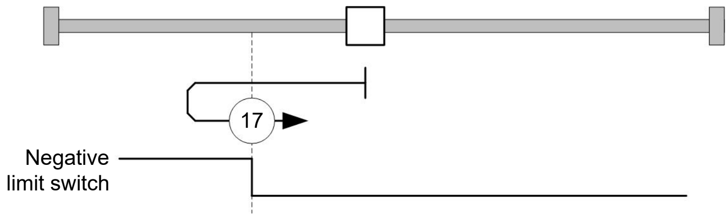
Methods 11 to 14 take the negative limit switch into account:



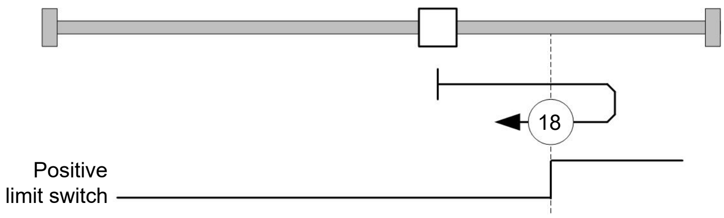
7.5.2.7 Methods 17 and 18

Reference to the limit switch without the index pulse.

Method 17 references to the negative limit switch:



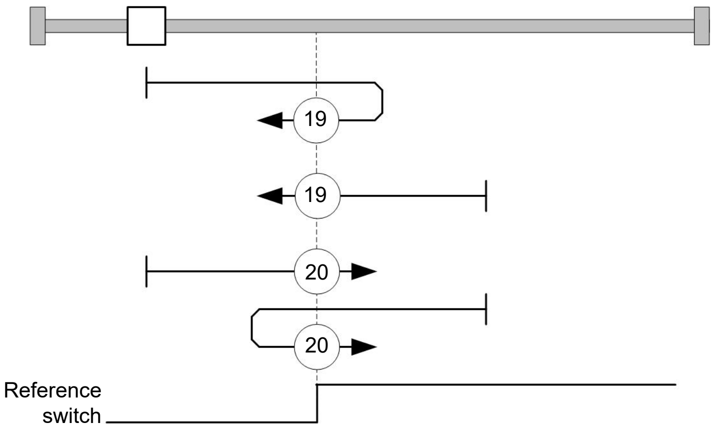
Method 18 references to the positive limit switch:



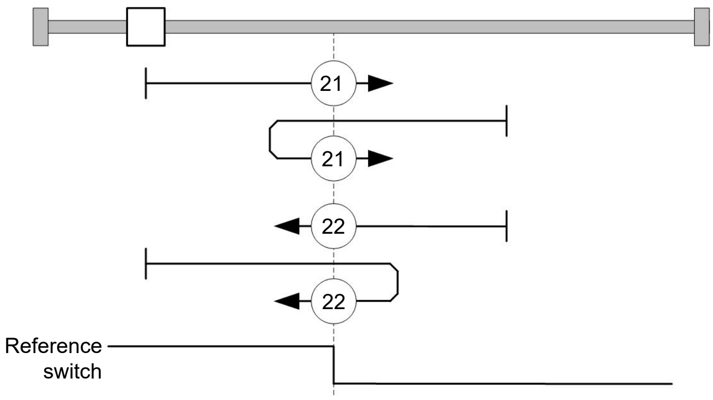
7.5.2.8 Methods 19 to 22

Reference to the switching edge of the home switch without the index pulse.

With methods 19 and 20 (equivalent to methods 3 and 4), the left switching edge of the home switch is used as reference:



With methods 21 and 22 (equivalent to methods 5 and 6), the right switching edge of the home switch is used as reference:

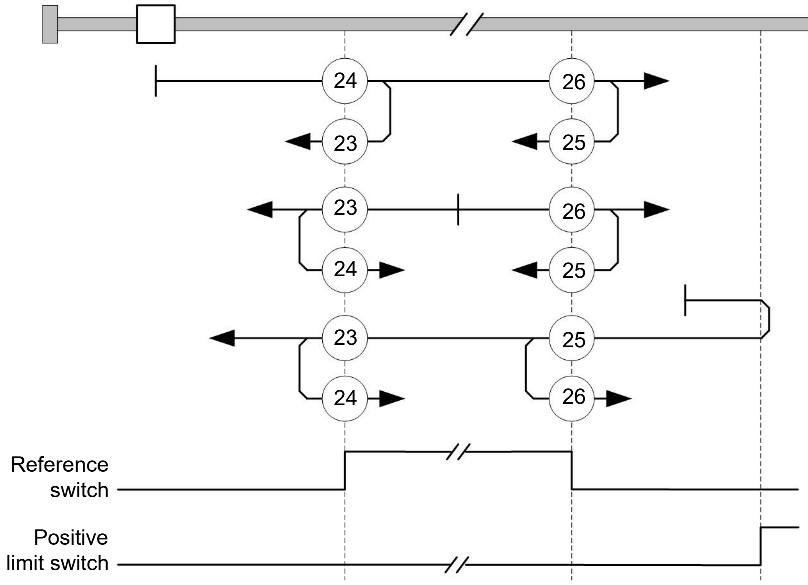


7.5.2.9 Methods 23 to 30

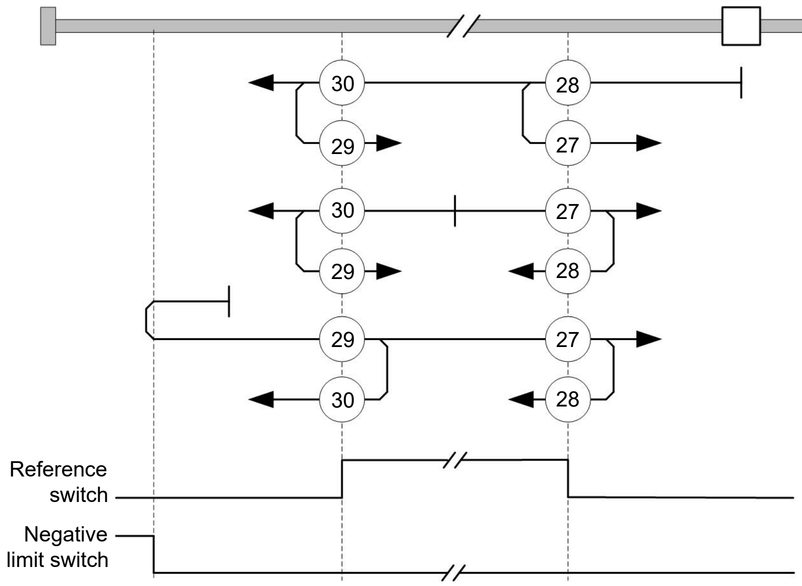
Reference to the home switch without the index pulse (with limit switches).

With these methods, the current position relative to the home switch is not important. With method 26, for example, referencing is always performed to the index pulse to the right of the right edge of the home switch.

Methods 23 to 26 take the positive home switch into account:



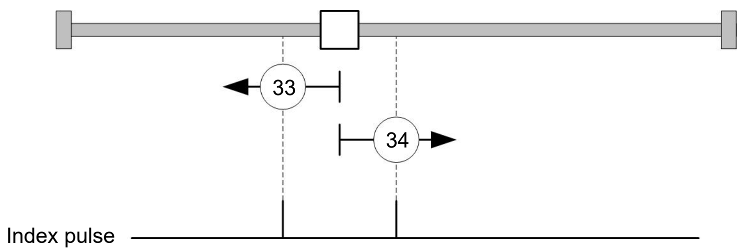
Methods 27 to 30 take the negative home switch into account:



7.5.2.10 Methods 33 and 34

Reference to the next index pulse.

With these methods referencing is only performed to the respective subsequent index pulse:



7.5.2.11 Method 35

References to the current position.

NOTICE

For homing mode 35, it is not necessary to switch the CiA 402 Power State Machine to the "Operation enabled" state. When energizing the motor windings in *open loop* mode, it is thereby possible to prevent the current position from not being exactly 0 after Homing Mode 35.

7.6 Interpolated Position Mode

7.6.1 Overview

7.6.1.1 Description

Interpolated position mode is used to synchronize multiple axes. For this purpose, a higher-level controller performs the ramp and path calculation and passes the respective demand position, at which the axis is to be located at a certain time, to the controller. The controller interpolates between these intermediate position points.

7.6.1.2 Synchronization with the SYNC object

For interpolated position mode, it is necessary that the controller synchronizes with the SYNC object (depending on the fieldbus). This SYNC object is to be sent by the higher-level controller in regular intervals. Synchronization occurs as soon as the controller is switched to the *Operational* NMT mode.

NOTICE

Where possible, it is recommended that a time interval of the *SYNC object* be used.

7.6.2 Activation

To activate the mode, the value "7" must be set in object 6060_h (Modes Of Operation) (see "CiA 402 Power State Machine").

7.6.3 Controlword

The following bits in object 6040_h (controlword) have a special function:

- Bit 4 activates the interpolation when it is set to "1".
- Bit 8 (Halt): If this bit is set to "1", the motor stops. On a transition from "1" to "0", the motor accelerates with the set start ramp to the target speed. On a transition from "0" to "1", the motor brakes and comes to a standstill. The braking deceleration is dependent here on the setting of the "Halt Option Code" in object $605D_h$.

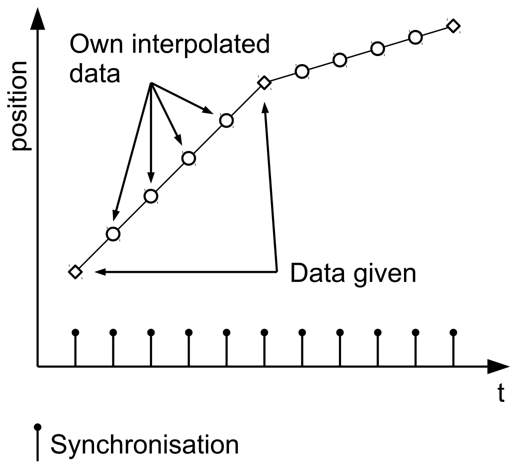
7.6.4 Statusword

The following bits in object 6041_h (statusword) have a special function:

- Bit 10: Target position reached: This bit is set to "1" if the target position was reached (if the halt bit in the controlword is "0") or the axis has speed 0 (if the halt bit in the last control word was "1").
- Bit 12 (IP mode active): This bit is set to "1" if interpolation is active.
- Bit 13 (Following Error): This bit is set in *closed loop* mode if the following error is greater than the set limits (6065_h (Following Error Window) and 6066_h (Following Error Time Out)).

7.6.5 Use

The controller follows a linearly interpolated path between the current position and the preset target position. The (next) target position must be written in record $60C1_h$:01_h.



In the current implementation, only

- linear interpolation
- and a target position

are supported.

7.6.6 Setup

The following setup is necessary:

- 60C2$_h$:01$_h$: Time between two passed target positions in ms.
- 60C4$_h$:06$_h$: This object is to be set to "1" to be able to modify the target position in object 60C1$_h$:01$_h$.
- 6081$_h$ (Profile Velocity): Maximum speed with which the position is to be approached
- 6084$_h$ (Profile Deceleration): Desired braking deceleration during braking
- 60C6$_h$: (Max Deceleration): The maximum allowed braking deceleration
- Only if closed loop is activated: The speed is limited by 607F$_h$ (Max Profile Velocity) and 6080$_h$ (Max Motor Speed); the smaller value is used as the limit.
- To be able to turn the motor, the *power state machine* is to be set to the *Operation enabled* state (see CiA 402 Power State Machine).

7.6.7 Operation

After setting up, the task of the higher-level controller is to write the target positions to object 60C1$_h$:01$_h$ in time.

7.7 Cyclic Synchronous Position

7.7.1 Overview

7.7.1.1 Description

In this mode, the controller receives an absolute position preset via the fieldbus at fixed time intervals (referred to in the following as a *cycle*). The controller then no longer calculates any ramps, but rather only follows the presets.

The target position is transferred cyclically (via *PDO*). Bit 4 in the controlword does not need to be set (unlike the Profile Position mode).

NOTICE

The target is absolute and, thus, independent of how often it was sent per *cycle*.

7.7.1.2 Synchronization with the SYNC object

To achieve smooth movement, the controller should synchronize with the SYNC object (depending on the field bus). This SYNC object is to be sent by the higher-level controller in regular intervals. Synchronization occurs as soon as the controller is switched to the *Operational* NMT mode.

NOTICE

Where possible, it is recommended that a time interval of the *SYNC object* be used for transfer of the target position.

7.7.1.3 Activation

To activate the mode, the value "8" must be set in object 6060_h (Modes Of Operation) (see "CiA 402 Power State Machine").

7.7.1.4 Controlword

In this mode, the bits of controlword 6040_h have no special function.

7.7.1.5 Statusword

The following bits in object 6041_h (statusword) have a special function:

Bit	Value	Description
8	0	The controller is not in sync with the fieldbus
8	1	The controller is in sync with the fieldbus
10	0	Reserved
10	1	Reserved
12	0	Controller does not follow the target; the preset of $607A_h$ (Target Position) is ignored
12	1	Controller follows the target; object $607A_h$ (Target Position) is used as the input for position control.
13	0	No following error
13	1	Following error

Bit 11: Limit exceeded: The demand position is above or below the limit values set in $607D_h$.

7.7.2 Object entries

The following objects are necessary for controlling this mode:

- $607A_h$ (Target Position): This object must be written cyclically with the position set value.
- $607B_h$ (Position Range Limit): This object contains the preset for an overrun or underrun of the position specification.
- $607D_h$ (Software Position Limit): This object defines the limitations within which the position specification ($607A_h$) must be located.
- 6065_h (Following Error Window): This object specifies a tolerance corridor in both the positive and negative direction from the set specification. If the actual position is outside of this corridor for longer than the specified time (6066_h), a following error is reported.
- 6066_h (Following Error Time Out): This object specifies the time range in milliseconds. If the actual position is outside of the position corridor (6065_h) for longer than this time range, a following error is triggered.
- 6085_h (Quick-Stop Deceleration): This object contains the braking deceleration for the case that a quick-stop is triggered.

- 605A$_h$ (Quick-Stop Option Code): This object contains the option that is to be executed in the event of a quick-stop.
- Only if closed loop is activated: 6080$_h$ (Max Motor Speed): Maximum speed
- 60C2$_h$:01$_h$ (Interpolation Time Period): This object specifies the time of a *cycle*; a new set value must be written in 607A$_h$ in these time intervals.
  The following applies here: cycle time = value of 60C2$_h$:01$_h$ * $10^{\text{value of 60C2:02}}$ seconds.
- 60C2$_h$:02$_h$ (Interpolation Time Index): This object specifies the time basis of the cycles. Currently, only value 60C2$_h$:02$_h$=-3 is supported; this yields a time basis of 1 millisecond.
- 60B0$_h$ (Position Offset): Offset for the position set value in user-defined units
- 60B1$_h$ (Velocity Offset): Offset for the speed set value in user-defined units
- 60B2$_h$ (Torque Offset): Offset for the torque set value in tenths of a percent

The following objects can be read in this mode:

- 6064$_h$ (Position Actual Value)
- 606C$_h$ (Velocity Actual Value)
- 60F4$_h$ (Following Error Actual Value)

7.8 Cyclic Synchronous Velocity

7.8.1 Overview

7.8.1.1 Description

In this mode, the controller passes a speed preset via the fieldbus at fixed time intervals (referred to in the following as a *cycle*). The controller then no longer calculates any ramps, but rather only follows the presets.

7.8.1.2 Activation

To activate the mode, the value "9" must be set in object 6060$_h$ (Modes Of Operation) (see "CiA 402 Power State Machine").

7.8.1.3 Controlword

In this mode, the bits of controlword 6040$_h$ have no special function.

7.8.1.4 Statusword

The following bits in object 6041$_h$ (statusword) have a special function:

Bit	Value	Description
8	0	The controller is not in sync with the fieldbus
8	1	The controller is in sync with the fieldbus
10	0	Reserved
10	1	Reserved
12	0	Controller does not follow the target; the preset of 60FF$_h$ (Target Velocity) is ignored
12	1	Controller follows the target; object 60FF$_h$ (Target Velocity) is used as the input for position control.
13	0	Reserved
13	1	Reserved

7.8.2 Object entries

The following objects are necessary for controlling this mode:

- 60FF$_h$ (Target Velocity): This object must be written cyclically with the speed set value.

- 6085$_h$ (Quick-Stop Deceleration): This object contains the braking deceleration for the case that a quick-stop is triggered (see "CiA 402 Power State Machine").
- 605A$_h$ (Quick-Stop Option Code): This object contains the option that is to be executed in the event of a quick-stop (see "CiA 402 Power State Machine").
- 6080$_h$ (Max Motor Speed): Maximum speed
- 60C2$_h$:01$_h$ (Interpolation Time Period): This object specifies the time of a *cycle*; a new set value must be written in 60FF$_h$ in these time intervals.
  The following applies here: cycle time = value of 60C2$_h$:01$_h$ * 10$^{\text{value of 60C2:02}}$ seconds.
- 60C2$_h$:02$_h$ (Interpolation Time Index): This object specifies the time basis of the cycles. Currently, only value 60C2$_h$:02$_h$=-3 is supported; this yields a time basis of 1 millisecond.
- 60B1$_h$ (Velocity Offset): Offset for the speed set value in user-defined units
- 60B2$_h$ (Torque Offset): Offset for the torque set value in tenths of a percent

The following objects can be read in this mode:

- 606C$_h$ (Velocity Actual Value)
- 607E$_h$ (Polarity)

7.9 Cyclic Synchronous Torque

7.9.1 Overview

7.9.1.1 Description

In this mode, the controller passes an absolute torque preset via the fieldbus at fixed time intervals (referred to in the following as a *cycle*). The controller then no longer calculates any ramps, but rather only follows the presets.

NOTICE

This mode only functions if closed loop is activated, see also Commissioning closed loop.

7.9.1.2 Activation

To activate the mode, the value "10" must be set in object 6060$_h$ (Modes Of Operation) (see "CiA 402 Power State Machine").

7.9.1.3 Controlword

In this mode, the bits of controlword 6040$_h$ have no special function.

7.9.1.4 Statusword

The following bits in object 6041$_h$ (statusword) have a special function:

Bit	Value	Description
8	0	The controller is not in sync with the fieldbus
8	1	The controller is in sync with the fieldbus
10	0	Reserved
10	1	Reserved
12	0	Controller does not follow the target; the preset of 6071$_h$ (Target Torque) is ignored
12	1	Controller follows the target; object 6071$_h$ (Target Torque) is used as the input for position control.
13	0	Reserved
13	1	Reserved

7.9.2 Object entries

The following objects are necessary for controlling this mode:

- 6071_h (Target Torque): This object must be written cyclically with the torque set value and is to be set relative to 6072_h.
- 6072_h (Max Torque): Describes the maximum permissible torque.
- 6073_h (Max Current):
  Maximum current. The minimum of 6073_h and 6072_h is used as limit for the torque in 6071_h.
- 6080_h (Max Motor Speed): Maximum speed
- $60C2_h$:01_h (Interpolation Time Period): This object specifies the time of a *cycle*; a new set value must be written in 6071_h in these time intervals.
  The following applies here: cycle time = value of $60C2_h$:01_h * $10^{\text{value of 60C2:02}}$ seconds.
- $60C2_h$:02_h (Interpolation Time Index): This object specifies the time basis of the cycles. Currently, only value $60C2_h$:02_h=-3 is supported; this yields a time basis of 1 millisecond.
- $60B2_h$ (Torque Offset): Offset for the torque set value in tenths of a percent

The following objects can be read in this mode:

- $606C_h$ (Velocity Actual Value)
- 6074_h (Torque Demand)

7.10 Clock-direction mode

7.10.1 Description

In clock-direction mode, the motor is operated via two inputs by a higher-level positioning controller with clock and direction signal. On each clock signal, the motor moves one step in the direction corresponding to the direction signal.

7.10.2 Activation

To activate the mode, the value "-1" (or "FFh") must be set in object 6060_h (Modes Of Operation) (see "CiA 402 Power State Machine").

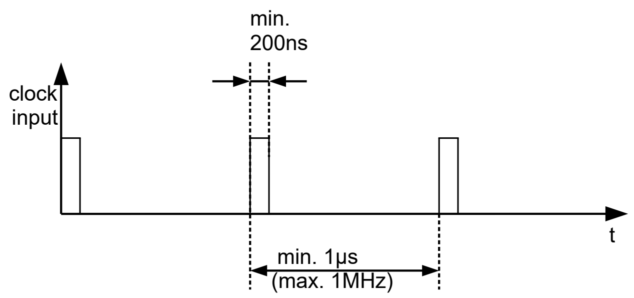
NOTICE

To use this mode, you must configure the pins for clock and direction appropriately as described in chapter Defining input and output assignments.

7.10.3 General

The following data apply for every subtype of the clock-direction mode:

- The maximum frequency of the input pulse is 1 MHz; the ON pulse should not be less than 200 ns.



- The demand position resulting from the input pulses is updated cyclically; the cycle time corresponds to the Interpolation Time Period ($60C2_h$). The input pulses that arrive within a cycle are collected and buffered in the controller.
- The steps are scaled using objects 2057_h and 2058_h. The following formula applies here:

$$\text{step width per pulse} = \frac{2057_h}{2058_h}$$

The "step size per pulse" value is set to 128 (2057_h=128 and 2058_h=1) ex works, which corresponds to a quarter step per pulse. A full step is the value "512", a half step per pulse corresponds to "256", etc.

NOTICE

For a stepper motor with 50 pole pairs, 200 full steps correspond to one mechanical revolution of the motor shaft.
In *clock-direction mode*, the BLDC motors are also handled as stepper motors by the controller. This means that for a BLDC motor with, e.g., 3 pole pairs, 12 (=4*3) full steps correspond to one revolution.

NOTICE

If there is a change of direction, a time of at least 35 µs must elapse before the new clock signal is applied.

7.10.4 Statusword

The following bits in object 6041_h (statusword) have a special function:

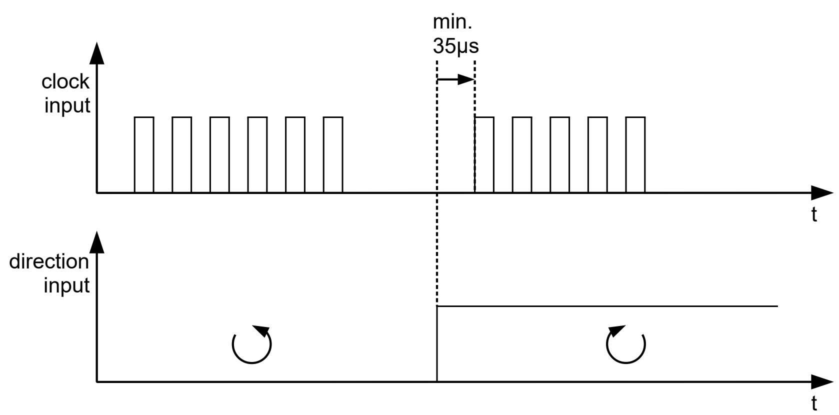
- Bit 13 (Following Error): This bit is set in *closed loop* mode if the following error is greater than the set limits (6065_h (Following Error Window) and 6066_h (Following Error Time Out)).

7.10.5 Subtypes of the clock-direction mode

7.10.5.1 Clock-direction mode (TR mode)

To activate the mode, object $205B_h$ must be set to the value "0" (factory settings).

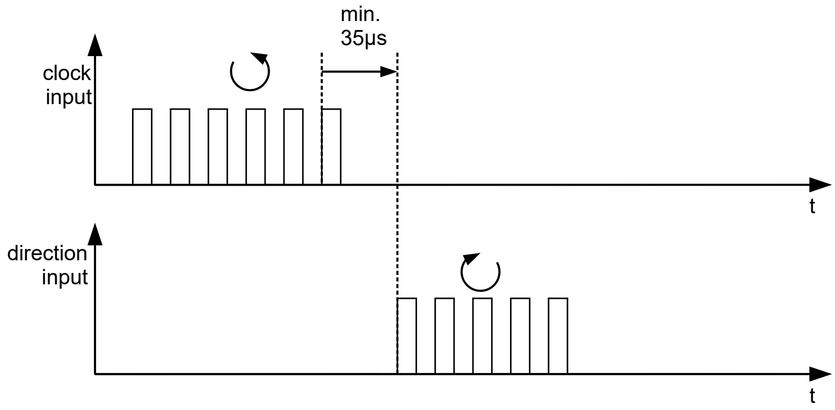
In this mode, the pulses must be preset via the clock input; the signal of the direction input specifies the direction of rotation here (see following graphic).



7.10.5.2 Right / left rotation mode (CW / CCW mode)

To activate the mode, object 205B$_h$ must be set to the value "1".

In this mode, the input that is used decides the direction of rotation (see following graphic).



7.11 Auto setup

7.11.1 Description

To determine a number of parameters related to the motor and the connected sensors (encoders/Hall sensors), an *auto setup* is performed. Closed-Loop operation requires a successfully completed *auto setup*. *Auto setup* is only to be performed once during commissioning as long as the motor/sensor connected to the controller is not changed. For details, see the corresponding section in chapter Commissioning.

7.11.2 Activation

To activate the mode, the value "-2" (="FE$_h$") must be set in object 6060$_h$ (Modes Of Operation) (see CiA 402 Power State Machine).

7.11.3 Controlword

The following bits in object 6040$_h$ (controlword) have a special function:

- Bit 4 starts a travel command. This is carried out on a transition from "0" to "1".

7.11.4 Statusword

The following bits in object 6041$_h$ (statusword) have a special function:

- Bit 10: Indexed: indicates whether (= "1") or not (= "0") an encoder index was found.
- Bit 12: Aligned: this bit is set to "1" after *auto setup* has concluded

8 Special functions

8.1 Digital inputs and outputs

This controller has 24 I/O pins, which can be configured as either input or output.

8.1.1 Defining input and output assignments

Some pins can be freely assigned. Which those are can be seen in the following table. You can find the assignment as set in the factory settings in chapter Pin assignment.

Define the function of each configurable pin in the corresponding subindex of object 3272h (Usage Of Pins Available On Connector). All of the pins listed in the following table can be assigned the following functions:

- Digital input or output
- *Input Range Selection*
- *Analog Input Control*
- *Capture Input*
- *SPI Chip Select*

The following pins also support alternative functions:

Pin	Alt. Function 1	Alt. Function 2	Subindex in 3272_h
4, ANA1	Analog input 1 (factory settings)		01_h
6, ANA2	Analog input 2 (factory settings)		02_h
18, H1	Hall sensor input 1 (factory settings)		03_h
19, DIO11			04_h
20, H2	Hall sensor input 2 (factory settings)		05_h
21, DIO13		SPI Data OUT (MOSI)	06_h
22, H3	Hall sensor input 3 (factory settings)		07_h
23, ENC1B	incr. Encoder 1, B (factory settings)	SSI encoder 2, Data IN	08_h
24, ENC1A	incr. Encoder 1, A (factory settings)	SSI encoder 2, clock	09_h
25, SSI1_MISO	SSI encoder 1, Data IN (factory settings)	SPI Data IN (MISO)	$0A_h$
26, ENC1I	incr. Encoder 1, Index (factory settings)		$0B_h$
27, SSI1_CLK	SSI encoder 1, clock (factory settings)	SPI Clock	$0C_h$
29, DIO14			$0D_h$
39, DIO1	Clock input in clock-direction mode	Channel A of the virtual encoder output	$0E_h$
41, DIO3			$0F_h$
42, DIO2	Direction input in clock-direction mode	Channel B of the virtual encoder output	10_h
43, DIO5			11_h
44, DIO4			12_h

Pin	Alt. Function 1	Alt. Function 2	Subindex in 3272_h
45, DIO7			13_h
46, DIO6			14_h
47, DIO9	PWM output 0		15_h
48, BRAKE	Brake output (factory settings)		16_h
50, DIO10	PWM output 1		17_h
56, DIO12			18_h

In object 3272_h, you define how each pin is to be used by writing the corresponding value in the corresponding subindex.

Value	Function
0	digital input
128	digital output
256/257	Input Range Selection
384/385	Analog Input Control
512	analog input
640	Hall sensor input
768	Encoder input (incremental)
896	Encoder input (SSI)
1024	PWM output / brake output
1152	Virtual encoder output
1280	Clock/direction input in Clock-direction mode
1408	Generic SPI
1536/1537	Capture Input

Then store your configuration by writing the value "65766173_h" in 1010_h:03_h (see chapterSaving objects) and restart the controller.

Example

Pin 39 (DIO1) is to be the clock input in Clock-direction mode.

Pin 42 (DIO2) is to be the direction input in Clock-direction mode.

1. Set 2372_h:$0E_h$ and 2372_h:10_h to "1280".
2. Set 1010_h:03_h to "65766173_h“.

Input Range Selection

You can assign up to two of the configurable pins this function. These pins are only digital outputs that can be controlled via the corresponding bit in 3240_h:06_h (set to *High* if bit=1):

Value in 3272_h:xx_h	Control bit in 3240_h:06_h
384	0
385	1

With these pins, you can control appropriate external circuits that toggle the switching level of the digital inputs, e.g., between 5/24 V.

Analog Input Control

You can assign up to two of the configurable pins this function. These pins are only digital outputs that can be controlled via the corresponding bit in 3221_h:06_h (set to *High* if bit=1):

Value in 3272_h:xx_h	Control bit in 3221_h:00_h
256	0
257	1

With these pins, you can control appropriate external circuits that toggle the corresponding analog input between voltage measurement and current measurement. The first pin (in the order given in the table of alternative functions) controls the first analog input and the second pin controls the second.

Capture Input

You can assign up to two of the configurable pins this function. These pins are capture inputs that are configured via the corresponding subindices in 3241_h:

Value in 3272_h:xx_h	Configuration in 3241_h
1536	Capture Input 1: Subindex 1 to 4
1537	Capture Input 2: Subindex 5 to 8

If there is a level change at these pins, the current encoder position is noted. The first pin assigned the function (in the order given in the table of alternative functions) is the first capture input (Capture Input 1).

Generic SPI

The controller can communicate with external devices via this SPI interface, e.g. port expanders or displays.

NOTICE

The used SPI peripheral must support a clock frequency of at least 164 KHz.

All configurable pins can be assigned the *Chip Select* function, with the exception of the pins that are intended for the following functions:

Pin	SPI function	Subindex in 3272_h
21, DIO13	Data OUT (MOSI)	06_h
25, SSI1_MISO	Data IN (MISO)	$0A_h$
27, SSI1_CLK	Clock	$0C_h$

To activate the respective function, write the value "1408" in 3272_h:xx_h.

NOTICE

Only the first (in the order given in the table of alternative functions) pin configured as *Chip Select* is used. If you need multiple pins with the function, use one pin as digital output.

The settings of the SPI interface are located in object 3273_h:01_h (Generic SPI Hardware Configuration):

- Bit 0 (Clock Phase):
  - Value = "0": Data transfer begins with the first falling clock edge (with polarity = 1)
  - Value = "1": Data transfer begins with the first rising clock edge (with polarity = 1)
- Bit 1 (Clock Polarity): With this bit, you can invert the polarity of the clock signal. The value 0 means the level remains on Low if the clock is idling.
- Bits 2 to 4 (baud rate): You set the clock frequency here:
  - 000_b: 21 MHz
  - 001_b: 10.5 MHz
  - 010_b: 5.25 MHz
  - 011_b: 2625 KHz
  - 100_b: 1312.5 KHz
  - 101_b: 656.25 KHz
  - 110_b: 328.125 KHz
  - 111_b: 164.0625 KHz
- Bit 10 (CS Polarity): With this bit, you can invert the polarity of the *Chip Select*. Value 0 means that the level remains on High if the signal is idling.

The data are sent/received via the following objects:

- 3274_h (Generic SPI Mosi Data):
  - Subindex 1 to $1F_h$ (Generic SPI Mosi Data Byte #1 to #31): You write the data that are to be sent here, divided into up to 31 bytes.
  - Subindex 0 (Length of SPI message to be sent): Then enter the number of bytes here (= subindicies) that are to be sent. In the next millisecond cycle, the data are sent and the subindex is reset to the value "0".
- 3275_h (Generic SPI Miso Data): You read the received data here.
  - Subindex 0 (Length of received SPI message): The value tells you how many data bytes (= subindices) were received.
  - Subindex 1 to $1F_h$ (Generic SPI Miso Data Byte #1 to #31): The data that were received are located here.

8.1.2 Bit assignment

The software of the controller assigns each input and output two bits in the respective object (e.g., 60FDh Digital Inputs or 60FEh Digital Outputs):

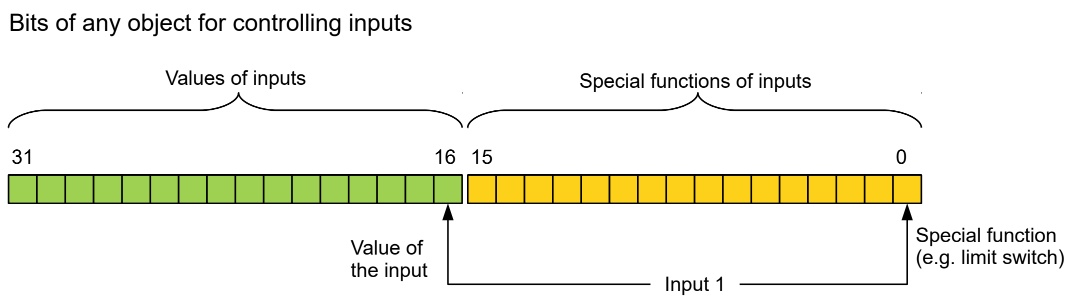
1. The first bit corresponds to the special function of an output or input. These functions are always available on bits 0 to 15 (inclusive) of the respective object. These include the limit switches and the home switch for the digital inputs and the brake control for the outputs.
2. The second bit shows the output/input as a level; these are then available on bits 16 to 31.

Example

To manipulate the value of output 2, always use bit 17 in $60FE_h$.

To activate the "negative limit switch" special function of input 1, set bit 0 in 3240_h:01_h; to query the status of the input, read bit 0 in $60FD_h$. Bit 16 in $60FD_h$ also shows the status of input 1 (independent of whether or not the special function of the input was activated).

This assignment is graphically illustrated in the following drawing.



8.1.3 Digital inputs

8.1.3.1 Overview

NOTICE

For digital inputs with 5 V, the length of the supply lines must not exceed 3 meters.

NOTICE

The digital inputs are sampled once per millisecond. Signal changes at the input less than one millisecond in duration are not processed.

The following inputs are available in the factory settings:

Pin	Name
39	DIO1
42	DIO2
41	DIO3
44	DIO4
43	DIO5
46	DIO6
45	DIO7
47	DIO9
50	DIO10
19	DIO11
56	DIO12
21	DIO13
29	DIO14

8.1.3.2 Object entries

The value of an input can be manipulated using the following OD settings, whereby only the corresponding bit acts on the input here.

- 3240_h:01_h (Special Function Enable): This bit allows special functions of an input to be switched off (value "0") or on (value "1"). If input 1 is not used as, e. g., a negative limit switch, the special function must be

switched off to prevent an erroneous response to the signal generator. The object has no effect on bits 16 to 31.
The firmware evaluates the following bits:

- Bit 0: Negative limit switch (see Limitation of the range of motion)
- Bit 1: Positive limit switch (see Limitation of the range of motion)
- Bit 2: Home switch (see Homing)
- Bit 3: Interlock (see interlock function)

If, for example, two limit switches and one home switch are used, bits 0–2 in 3240_h:01_h must be set to "1".

NOTICE

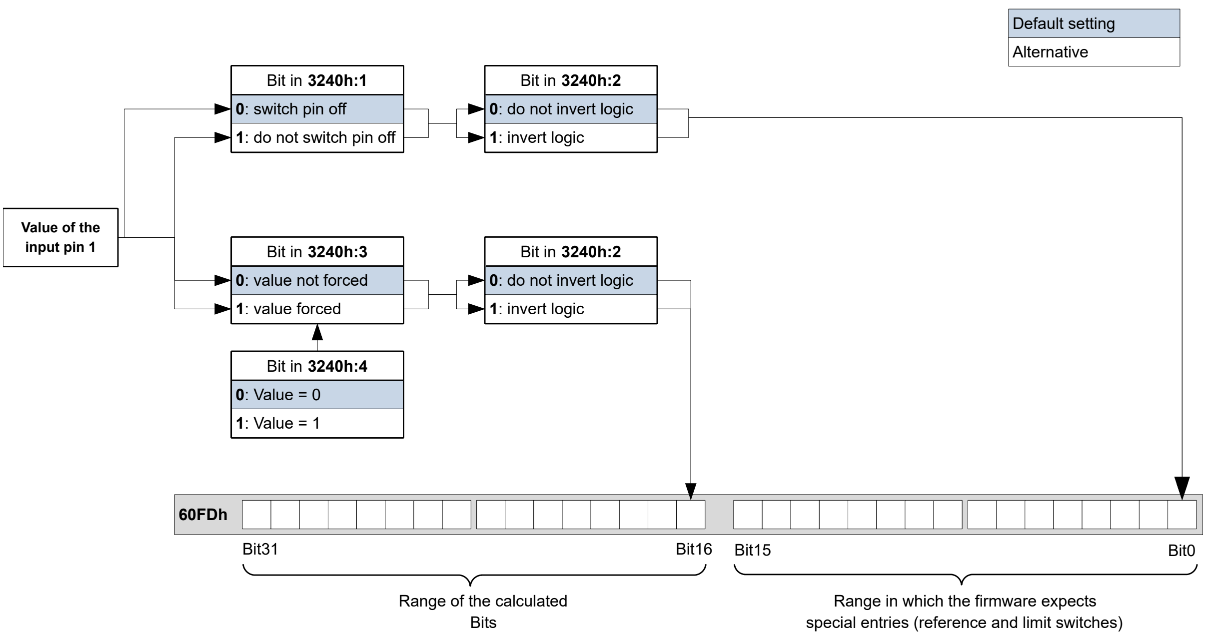
Because the *Input Routing* (see following chapter) is activated in the factory settings, object 3240_h:01_h has no function in this controller. To use the special functions, configure the source for bits 0 to 3 of $60FD_h$ in 3242_h:01_h to :04_h according to your cabling.

- 3240_h:02_h (Function Inverted): This subindex switches from normally open logic (a logical high level at the input yields the value "1" in object $60FD_h$) to normally closed logic (the logical high level at the input yields the value "0").
  This applies for the special functions (except for the clock and direction inputs) and for the normal inputs. If the bit has the value "0", normally open logic applies; for the value "1", normally closed logic applies. Bit 0 changes the logic of input 1, bit 1 changes the logic of input 2, etc.
- 3240_h:03_h (Force Enable): This subindex switches on the software simulation of input values if the corresponding bit is set to "1".
  In this case, the actual values are no longer used in object 3240_h:04_h, but rather the set values for the respective input. Bit 0 corresponds to input 1 here, bit 1 to input 2, etc.
- 3240_h:04_h (Force Value): This bit specifies the value that is to be read as the input value if the same bit was set in object 3240_h:03_h.
- 3240_h:05_h (Raw Value): This object contains the unmodified input value.
- $60FD_h$ (Digital Inputs): This object contains a summary of the inputs and the special functions.

8.1.3.3 Computation of the inputs

Computation of the input signal using the example of input 1:

The value at bit 0 of object $60FD_h$ is interpreted by the firmware as negative limit switch; the result of the complete computation is stored in bit 16.



8.1.3.4 Input Routing

Principle

To perform the assignment of the inputs more flexibly, there is a mode called *Input Routing Mode*. This assigns a signal of a source to a bit in object 60FD$_h$.

Activation

This mode is activated by setting object 3240$_h$:08$_h$ (Routing Enable) to "1" (this is the case in the factory setting).

NOTICE

Entries 3240$_h$:01$_h$ to 3240:04$_h$ then have **no** function until Input Routing is again switched off.

NOTICE

If *Input Routing* is switched on, the initial values of 3242$_h$ are changed and correspond to the function of the input as it was before activation of *Input Routing*. The inputs of the controller behave the same with activation of *Input Routing*. Therefore, you should not switch back and forth between the normal mode and *Input Routing*.

Routing

Object 3242$_h$ determines which signal source is routed to which bit of 60FD$_h$. Subindex 01$_h$ of 3242$_h$ determines bit 0, subindex 02$_h$ determines bit 1, and so forth. The signal sources in the factory settings and their numbers can be found in the following lists.

NOTICE

If you deactivate the *Input Routing*, bits 16 to 31 correspond to the first 16 subindices in the table of alternative functions.

Number		
dec	**hex**	**Signal source**
00	00	Signal is always 0
01	01	ANA1 (Pin 4)
02	02	ANA2 (Pin 6)
03	03	H1 (Pin 18)
04	04	DIO11 (Pin 19)
05	05	H2 (Pin 20)
06	06	DIO13 (Pin 21)
07	07	H3 (Pin 22)
08	08	ENC1B (Pin 23)
09	09	ENC1A (Pin 24)

Number		
dec	**hex**	**Signal source**
10	0A	SSI1_MISO (Pin 25)
11	0B	ENC1I (Pin 26)
12	0C	SSI1_CLK (Pin 27)
13	0D	DIO14 (Pin 29)
14	0E	DIO1 (Pin 39)
15	0F	DIO3 (Pin 41)
16	10	DIO2 (Pin 42)
17	11	DIO5 (Pin 43)
18	12	DIO4 (Pin 44)
19	13	DIO7 (Pin 45)
20	14	DIO6 (Pin 46)
21	15	DIO9 (Pin 47)
22	16	BRAKE (Pin 48)
23	17	DIO10 (Pin 50)
24	18	DIO12 (Pin 56)
65	41	Hall input "U"
66	42	Hall input "V"
67	43	Hall input "W"
68	44	Encoder input "A"
69	45	Encoder input "B"
70	46	Encoder input "Index"

The following table describes the inverted signals of the previous table.

Number		
dec	**hex**	**Signal source**
128	80	Signal is always 1
129	81	Inverted ANA1 (Pin 4)
130	82	Inverted ANA2 (Pin 6)
131	83	Inverted H1 (Pin 18)
132	84	Inverted DIO11 (Pin 19)
133	85	Inverted H2 (Pin 20)
134	86	Inverted DIO13 (Pin 21)
135	87	Inverted H3 (Pin 22)
136	88	Inverted ENC1B (Pin 23)
137	89	Inverted ENC1A (Pin 24)
138	8A	Inverted SSI1_MISO (Pin 25)
139	8B	Inverted ENC1I (Pin 26)
140	8C	Inverted SSI1_CLK (Pin 27)
141	8D	Inverted DIO14 (Pin 29)
142	8E	Inverted DIO1 (Pin 39)
143	8F	Inverted DIO3 (Pin 41)
144	90	Inverted DIO2 (Pin 42)
145	91	Inverted DIO5 (Pin 43)

Number		
dec	hex	Signal source
146	92	Inverted DIO4 (Pin 44)
147	93	Inverted DIO7 (Pin 45)
148	94	Inverted DIO6 (Pin 46)
149	95	Inverted DIO9 (Pin 47)
150	96	Inverted BRAKE (Pin 48)
151	97	Inverted DIO10 (Pin 50)
152	98	Inverted DIO12 (Pin 56)
193	C1	Inverted Hall input "U"
194	C2	Inverted Hall input "V"
195	C3	Inverted Hall input "W"
196	C4	Inverted encoder input "A"
197	C5	Inverted encoder input "B"
198	C6	Inverted encoder input "Index"

Example

Input 1 should be routed to bit 0 of object $60FD_h$ in order to be used as a negative limit switch.

1. To activate the *Input Routing*, set 3240_h:08_h to "1" (already set in the factory settings).
2. To route input 1 (DIO1) to bit 0, set 3242_h:01_h to "14".

8.1.3.5 Interlock function

The interlock function is a release that you control via bit 3 in $60FD_h$. If this bit is set to "1", the motor can move. If the bit is set to "0", the controller switches to the error state and the action stored in $605E_h$ is executed.

Use *Input Routing* to define which signal source is routed to bit 3 of $60FD_h$ and is to control the interlock function.

Example

Input 4 is to be routed to bit 3 of object $60FD_h$ to control the interlock function. A low level is to result in an error state.

1. To activate the *Input Routing*, set 3240_h:08_h to "1" (already set in the factory settings).
2. To route input 4 (DIO4) to bit 3, set 3242_h:04_h to "18".

8.1.4 Digital outputs

8.1.4.1 Outputs

The outputs are controlled via object $60FE_h$. Here, output 1 corresponds to bit 16 in object $60FE_h$, output 2 corresponds to bit 17, etc., as with the inputs. The outputs with special functions are again entered in the firmware in the lower bits 0 to 15. The only bit assigned at the present time is bit 0, which controls the motor brake.

You must have first defined the desired pins as output, see Defining input and output assignments.

8.1.4.2 Wiring

The digital outputs have a digital level of 3.3 V DC. The maximum admissible current is approx. 10 mA.

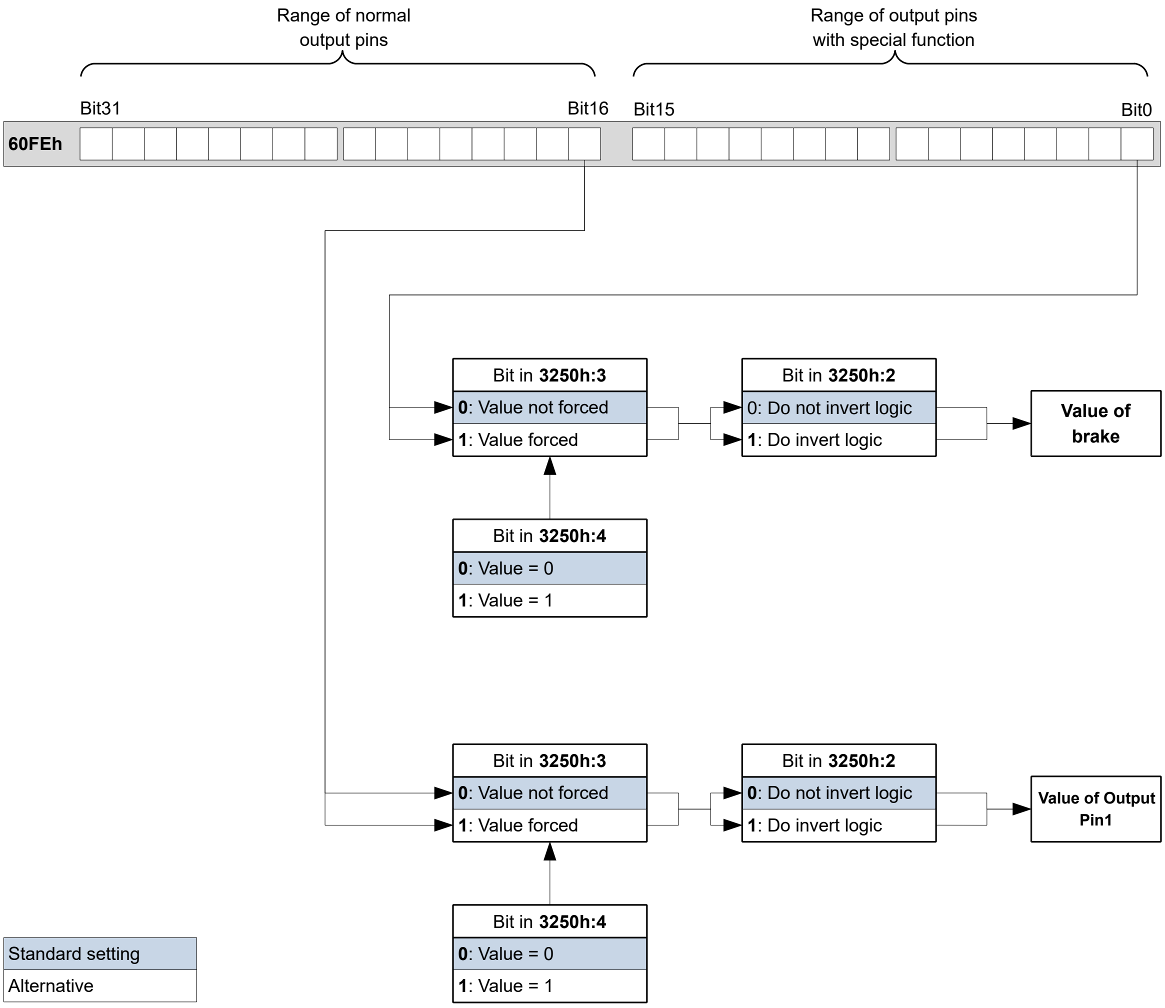
8.1.4.3 Object entries

Additional OD entries are available for manipulating the value of the outputs (see the following example for further information). As with the inputs, only the bit at the corresponding location acts on the respective output:

- 3250_h:01_h: No function.
- 3250_h:02_h: This is used to switch the logic from *normally open* to *normally closed*. Configured as *normally open*, the output outputs a logical high level if the bit is "1". With the *normally closed* configuration, a logical low level is output accordingly for a "1" in object $60FE_h$.
- 3250_h:03_h: If a bit is set here, the output is controlled manually. The value for the output is then in object 3250_h:4_h; this is also possible for the brake output.
- 3250_h:04_h: The bits in this object specify the output value that is to be applied at the output if manual control of the output is activated by means of object 3250_h:03_h.
- 3250_h:05_h: The bit combination applied to the outputs is stored in this subindex.
- 3250_h:08_h: For activating the Output Routing.
- 3250_h:09_h: For switching control of the Power LED on/off. If bit 0 is set to "1", the green LED is activated (flashes in normal operation). If bit 1 is set to "1", the red LED is activated (flashes in case of an error). If the bit is set to "0", the respective LED remains off.

8.1.4.4 Computation of the outputs

Example for calculating the bits of the outputs:

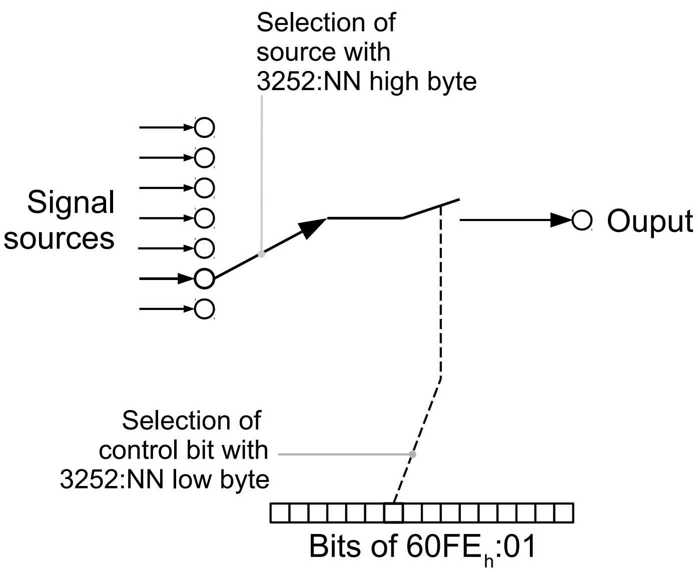


8.1.4.5 Output Routing

Principle

The "Output Routing Mode" assigns an output a signal source; a control bit in object 60FE$_h$:01$_h$ switches the signal on or off.

The source is selected with 3252$_h$:01 to n in the "high byte" (bit 15 to bit 8). The assignment of a control bit from object 60FE$_h$:01$_h$ is performed in the "low byte" (bit 7 to bit 0) of 3252$_h$:01$_h$ to n (see following figure).



Activation

This mode is activated by setting object 3250h:08h (Routing Enable) to "1" (this is the case in the factory setting).

NOTICE

Entries 3250h:01h to 3250:04h then have **no** function until *Output Routing* is again switched off.

Routing

The subindex of object 3252h determines which signal source is routed to which output. The output assignments are listed in the following:

Subindex 3252h	Output Pin
01h	Configuration of output 1 (pin 4)
02h	Configuration of output 2 (pin 6)
03h	Configuration of output 3 (pin 18)
04h	Configuration of output 4 (pin 19)
05h	Configuration of output 5 (pin 20)
06h	Configuration of output 6 (pin 21)
07h	Configuration of output 7 (pin 22)
08h	Configuration of output 8 (pin 23)
09h	Configuration of output 9 (pin 24)
0Ah	Configuration of output 10 (pin 25)
0Bh	Configuration of output 11 (pin 26)
0Ch	Configuration of output 12 (pin 27)
0Dh	Configuration of output 13 (pin 29)
0Eh	Configuration of output 14 (pin 39)
0Fh	Configuration of output 15 (pin 41)
10h	Configuration of output 16 (pin 42)
11h	Configuration of output 17 (pin 43)
12h	Configuration of output 18 (pin 44)

Subindex 3252_h	Output Pin
13_h	Configuration of output 19 (pin 45)
14_h	Configuration of output 20 (pin 46)
15_h	Configuration of output 21 (pin 47)
16_h	Configuration of output 22 (pin 48)
17_h	Configuration of output 23 (pin 50)
18_h	Configuration of output 24 (pin 56)

NOTICE

The maximum output frequency of the PWM output (software PWM) is 2 kHz. All other outputs can only produce signals up to 500 Hz.

Subindices 3252_h:01_h to $0n_h$ are 16 bits wide, whereby the high byte selects the signal source (e. g., the PWM generator) and the low byte determines the control bit in object $60FE_h$:01.

Bit 7 of 3252_h:01_h to $0n_h$ inverts the controller from object $60FE_h$:01. Normally, value "1" in object $60FE_h$:01_h switches on the signal; if bit 7 is set, the value "0" switches on the signal.

TIP

To deactivate routing, enter the value $FFFF_h$.

Number in 3252:01 to 0n	
$00XX_h$	Output is always "1"
$01XX_h$	Output is always "0"
$02XX_h$	Encoder signal (6063_h) with frequency divider 1
$03XX_h$	Encoder signal (6063_h) with frequency divider 2
$04XX_h$	Encoder signal (6063_h) with frequency divider 4
$05XX_h$	Encoder signal (6063_h) with frequency divider 8
$06XX_h$	Encoder signal (6063_h) with frequency divider 16
$07XX_h$	Encoder signal (6063_h) with frequency divider 32
$08XX_h$	Encoder signal (6063_h) with frequency divider 64
$09XX_h$	Position Actual Value (6064_h) with frequency divider 1
$0AXX_h$	Position Actual Value (6064_h) with frequency divider 2
$0BXX_h$	Position Actual Value (6064_h) with frequency divider 4
$0CXX_h$	Position Actual Value (6064_h) with frequency divider 8
$0DXX_h$	Position Actual Value (6064_h) with frequency divider 16
$0EXX_h$	Position Actual Value (6064_h) with frequency divider 32
$0FXX_h$	Position Actual Value (6064_h) with frequency divider 64
$10XX_h$	PWM signal that is configured with object 2038_h (brake output)
$11XX_h$	Inverted PWM signal that is configured with object 2038_h (brake output)
$12XX_h$	PWM signal that is configured with object 3260_h
$13XX_h$	PWM signal that is configured with object 3261_h

NOTICE

On any change of the "encoder signal" (6063$_h$) or the current position (6064$_h$ in user-defined units) by an increment, a pulse is output at the digital input (for frequency divider 1). Take this into account when selecting the frequency divider and the unit, especially when using sensors with low resolution (such as Hall sensors).

Example

The encoder signal (6063$_h$) is to be applied to output 1 with a frequency divider 4. The output is to be controlled with bit 5 of object 60FE:01.

- 3250$_h$:08$_h$ = 1 (activate routing)
- 3252$_h$:02$_h$ = 0405$_h$ (04XX$_h$ + 0005$_h$)
- 04XX$_h$: Encoder signal with frequency divider 4
- 0005$_h$: Selection of bit 5 of 60FE:01

The output is switched on by setting bit 5 in object 60FE:01.

Example

The brake PWM signal is to be applied to output 2. Because the automatic brake control uses bit 0 of 60FE:01$_h$, this should be used as control bit.

- 3250$_h$:08$_h$ = 1 (activate routing)
- 3252$_h$:03$_h$ = 1080$_h$ (=10XX$_h$ + 0080$_h$). Where:
  - 10XX$_h$: Brake PWM signal
  - 0080$_h$: Selection of the inverted bit 0 of object 60FE:01

8.1.5 Virtual encoder output

You have the option of outputting the actual position via two pins of the controller and passing it on to your PLC or another controller. The maximum frequency here is 200 kHz.

Activating the function of the pins

To activate the function, set 2372$_h$:0E$_h$ and 2372$_h$:10$_h$ to "1152".

See also Defining input and output assignments.

Selecting the type of output signals

You can select one of the following types in object 205C$_h$02$_h$:

- Value "0": two 90° phase-shifted channels on channel A (leading when moving in the positive direction) and B, analogous to an incremental encoder
- Value "1": a clock and direction signal on channel A and B, analogous to the signals in Clock-direction mode
- Value "2": two clock signals, analogous to the signals in Right / left rotation mode (CW / CCW mode)

Selecting the source of the position data

The position data of one of the existing feedbacks are reproduced via the encoder output.

To select the source, set bit 3 in the corresponding subindex of object 3203$_h$ to "1". If you do not set a bit, the value from 205C:01$_h$ is used.

Setting the resolution of the output signals

Define the conversion of source signal into virtual encoder signals via the numerator in 205C:03_h and the denominator in 205C:04_h.

8.2 Automatic brake control

8.2.1 Description

Automatic brake control is activated if the controller is switched to the *Operation enabled* state of the CiA 402 Power State Machine; the brake otherwise always remains closed.

The brake output of the controller results in a PWM signal that can be adjusted with respect to frequency and duty cycle.

For information on the interaction of the brake with the motor stopping behavior, see also chapter Power State machine – halt motion reactions.

8.2.2 Activation and connection

The brake can be controlled either automatically or manually:

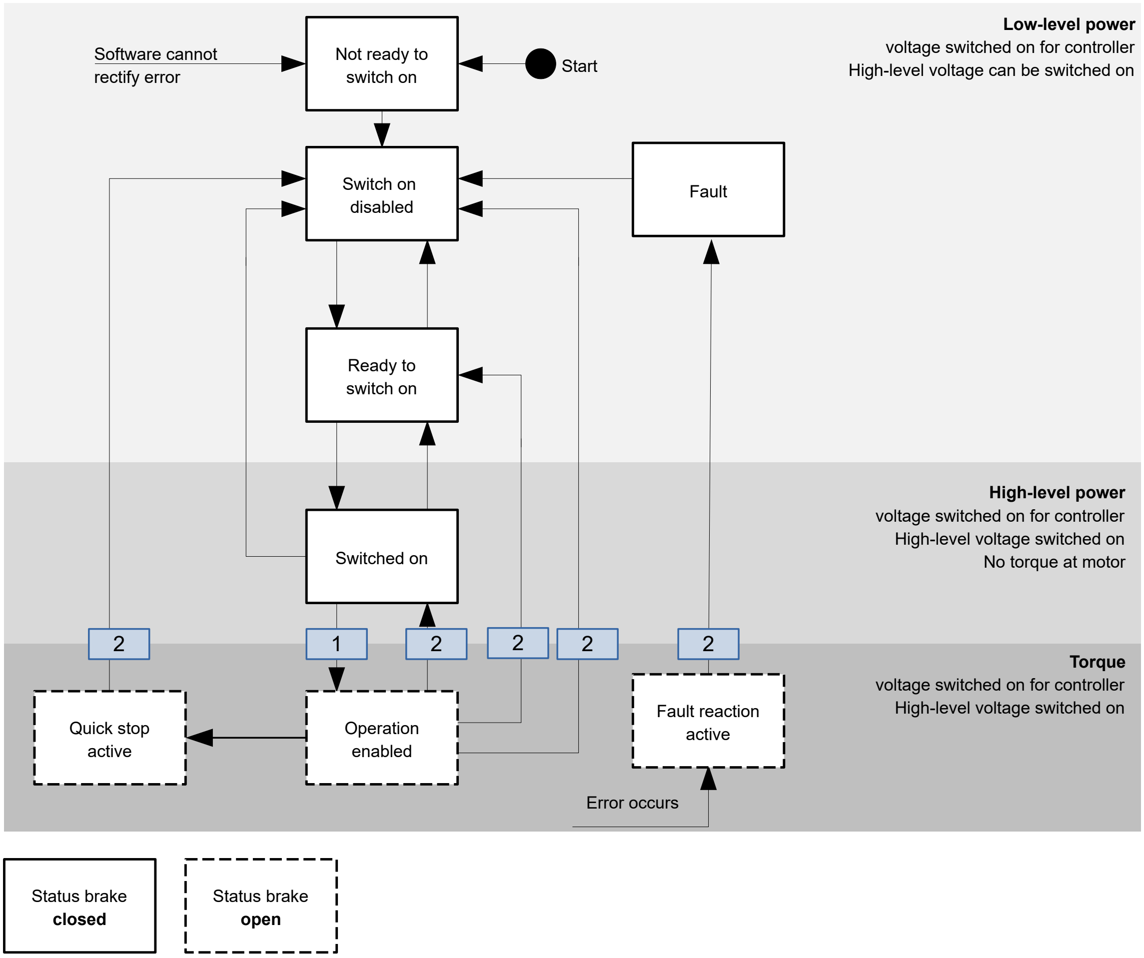
- Automatic: Setting bit 2 of object 3202_h to "1" activates the brake control.
- Manual: Setting bit 2 of object 3202_h to "0" deactivates the brake control; the brake can now be controlled with bit 0 in object $60FE_h$:01_h .

8.2.2.1 Connection

The brake output is located on pin 48 (see Pin assignment). To use the brake output, you must configure the pin appropriately (this is the case in the factory settings), (see Digital inputs and outputs).

8.2.3 Brake control

The following graphic shows the states of the CiA 402 Power State Machine together with the states of the brake for the automatic mode.



The following steps are performed on the transition, which is marked with 1:

1. The motor current is switched on.
2. The time stored in 2038h:3h is allowed to elapse.
3. The brake releases.
4. The time stored in 2038h:4h is allowed to elapse.
5. The *Operation enabled* state is reached, the motor controller can perform travel commands.

The following steps are performed on all transitions that are marked with 2:

1. The motor is brought to a standstill.
2. The time stored in 2038h:1h is allowed to elapse.
3. The brake is activated.
4. The time stored in 2038h:2h is allowed to elapse.
5. The motor current is switched off.

8.2.4 Brake PWM

The switched-on brake generates a PWM signal at the output of the controller that can be adjusted with respect to duty cycle and frequency. If an output pin without PWM is needed, a duty cycle of 100 percent can be set.

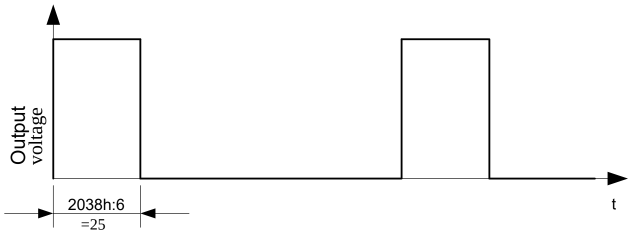
8.2.4.1 Frequency

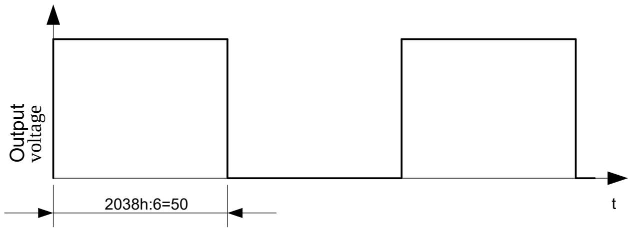
The frequency of the brake PWM can be set in object 2038h:5h. The unit is Hertz; a value less than 50 or greater than 20000 is not possible.

8.2.4.2 Duty cycle

The duty cycle – the ratio of pulse to period duration – is set in 2038$_h$:6$_h$. The value is a percentage and can be selected between 2 and 100. With a value of 100, the output pin is permanently switched on.

In the following figure, example duty cycles of 25 and 50 percent are shown, whereby the frequency is held constant.





8.3 External ballast circuit

During braking, electrical energy is fed back into the DC-link through self-induction of the motor. If not using a power supply with regenerative-feedback capability, the brake power can cause the DC-link voltage to increase which, if no additional measures are taken, is limited only by the internal consumption and capacitances in the DC-link.

To prevent damage to the controller through overvoltage, it may – depending on the level of the braking power – be necessary to dissipate excess energy in the form of heat. For this purpose, the controller provides an output at pin (pin 52) for controlling an external ballast circuit that consists of a driver, a MOSFET as switch, and a sufficiently dimensioned ballast resistor.

8.3.1 Control of the ballast resistor

A ballast controller and monitor that has two functions is implemented in the firmware of the controller:

- Limitation of the DC-link voltage through activation of the ballast resistor or shutdown of the output stage
- Protection of the ballast resistor against thermal overload

The parameters to be configured are described in the following chapters.

8.3.2 Activating the ballast

To activate the ballast, set bit 0 in 4021$_h$:01$_h$ to "1". If you would like to invert the polarity of the pin for controlling the external ballast circuit (pin 52, on delivery: *active high*), set bit 1 in 4021$_h$:01$_h$ to "1".

Enter the response threshold in millivolts as well as the hysteresis when switching on/off in 4021$_h$:02$_h$ and 4021$_h$:03$_h$, respectively.

If, in spite of the activation, the ballast is not able to limit the increase in the DC-link voltage, an error is generated and the driver output stage switched off when the overvoltage threshold (2034$_h$) is exceeded.

8.3.3 Ballast monitoring

The firmware constantly monitors the ballast resistor by adding up the energy it converts – taking into account the thermal energy that the resistor discharges to its surroundings through convection.

If the energy exceeds the permissible limit value, the ballast resistor is blocked from switching on and a warning generated with error code 7113_h (see 1003_h). After the resistor has cooled sufficiently, the block is automatically canceled.

To configure the monitoring, you must ascertain or determine the following resistor parameters from the data sheet of the ballast resistor and enter them in the corresponding subindex of 4021_h:

Nominal Resistance $R_{Ballast}$, [mOhm]

Rated value of the ballast resistor

Cooling Power $P_{Stat\_TA\_Max}$, [mW]

The amount of heat that the resistor can/may constantly discharge to its surroundings. You can calculate these as follows:

$$P_{Stat\_TA\_Max} = (T_{Ballast\_Max} - TA_{Max})/R_{th,A}$$

- $T_{Ballast\_Max}$: Maximum permissible surface temperature of the resistor. Limited by the data of the resistor (data sheet value) or by the installation position (temperature stability of adjacent components).
- TA_{Max}: Maximum temperature in the surroundings of the ballast
- $R_{th,A}$: Thermal resistance of the ballast resistor to the surroundings (data sheet value)

Short Term Energy Limit $E_{ST\_25°C}$, [mWs]

Amount of energy that can be supplied to the resistor within a short load surge (<1 second) without overloading it.

The material of the resistor element (wire, thick film) is the limiting factor here as, in the case of short pulses, practically only it can absorb energy and heats up.

For load resistors, the value is typically specified in the data sheet.

Long Term Energy Limit $E_{LT\_TA\_Max}$, [mWs]

Amount of energy that can be supplied to the resistor within the *Long Term Reference Time* (see below, typically between 1 and 5 seconds) without overloading it.

In the case of long pulses, the carrier material (cement or ceramic body) also absorbs energy and thereby slows the temperature rise.

The long-term overload capacity of a load resistor is typically specified in its data sheet in the form of an overload factor for a certain length of time (e.g. 5x rated power for 5 seconds).

Long Term Reference Time t_{LT_Ref}, [ms]

The reference time for the Long Term Energy Limit (typically between 1 and 5 seconds)

If the parameters are not valid or realistic, an error is generated with error code 7110_h (see 1003_h).

8.4 I^2t Motor overload protection

8.4.1 Description

NOTICE

For stepper motors, only the rated current is specified, not a maximum current. No liability is therefore assumed when using I^2t with stepper motors.

The goal of I^2t motor overload protection is to protect the motor from damage and, at the same time, operate it normally up to its thermal limit.

This function is only available if the controller is in the closed loop mode (bit 0 of object 3202_h must be set to "1").

8.4.2 Object entries

The following objects affect I^2t motor overload protection:

- 2031_h: Max Motor Current – specifies the maximum permissible motor current in mA.
- $203B_h$:1_h Motor Rated Current – specifies the rated current in mA.
- 6073_h Max Current – specifies the maximum current in tenths of a percent of the set rated current.
- $203B_h$:2_h Maximum Duration Of Peak Current – specifies the maximum duration of the maximum current in ms.

The following objects indicate the current state of I^2t:

- $203B_h$:3_h Threshold – specifies the limit in A^2ms that determines whether the maximum current or rated current is switched to.
- $203B_h$:4_h CalcValue – specifies the calculated value in A^2ms that is compared with the threshold for setting the current.
- $203B_h$:5_h LimitedCurrent – shows the momentary current value in mA that was set by I^2t.
- $203B_h$:6_h Status:
  - Value = "0": I^2t deactivated
  - Value = "1": I^2t activated

8.4.3 Activation

Closed loop must be activated, (bit 0 of object 3202_h set to "1", see also chapter Closed-Loop).

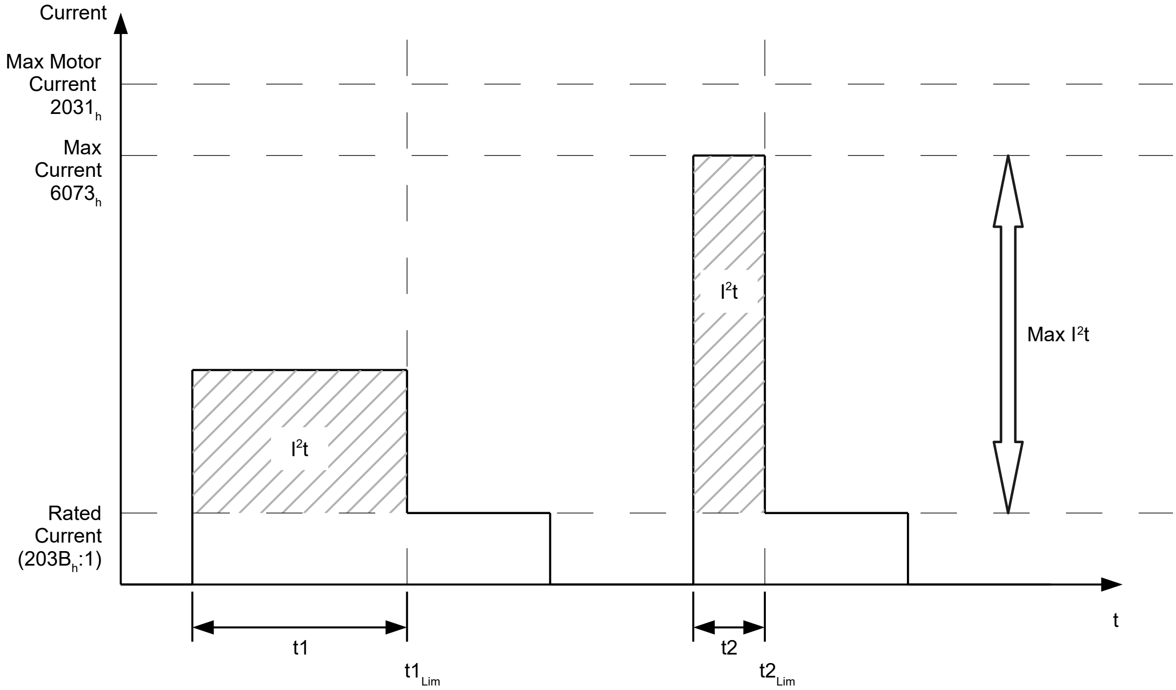
To activate the mode, you must appropriately specify the four object entries mentioned above (2031_h, 6073_h, $203B_h$:1_h, $203B_h$:2_h). This means that the maximum current must be greater than the rated current and a time value for the maximum duration of the maximum current must be entered. If these conditions are not met, the I^2t functionality remains deactivated.

8.4.4 Function of I^2t

From the specification of rated current, maximum current and maximum duration of the maximum current, an I^2t_{Lim} is calculated.

The motor can run with maximum current until the calculated I^2t_{Lim} is reached. The current is then immediately reduced to the rated current. The maximum current is limited by the maximum motor current (2031_h).

The relationships are illustrated again in the following diagrams.



In the first section, t1, the current value is higher than the rated current. At time $t1_{Lim}$, I^2t_{Lim} is reached and the current is limited to the rated current. A current that corresponds to the maximum current then occurs for a period of time t2. Hence, the value for I^2t_{Lim} is reached more quickly than in time t1.

8.5 Saving objects

NOTICE

Improper use of the function can result in it no longer being possible to start the controller. Therefore, carefully read the entire chapter before using the function.

NOTICE

As an alternative, objects can also be set and saved using the configuration file (see Configuration via USB) . Note that this file has higher priority. Objects that are saved both with the mechanism described here as well as in the configuration file take the value of the configuration file.

8.5.1 General

Many objects in the object dictionary can be saved and then automatically reloaded the next time the controller is switched on or reset. Furthermore, the saved values are also retained following a firmware update.

Only entire collections of objects (referred to in the following as *categories*) can be saved together; individual objects cannot be saved.

An object can be assigned one of the following *categories*:

- Communication: Parameters related to external interfaces, such as PDO configuration etc.
- Application: Parameters related to operating modes.
- Customer: Parameters that are written and read by the customer/user only and are ignored by the controller firmware.
- Drive: Parameters related to the motor and the sensors (BLDC/Stepper, *closed/open-loop*…). Some are set and saved by auto setup.

- Tuning: Parameters related to motor and encoder that are set either by auto setup or that can be found in the data sheets, e.g., pole pairs and maximum current.

If an object is not assigned one of these *categories*, it cannot be saved, e.g., statusword and all objects whose value is dependent on the current state of the controller.

The objects in each *category* are listed below. In chapter Description of the object dictionary, the corresponding *category* for each object is also specified.

8.5.2 Category: communication

- 2102_h: Fieldbus Module Control
- 3502_h: MODBUS Rx PDO Mapping
- 3602_h: MODBUS Tx PDO Mapping

8.5.3 Category: application

- 2034_h: Upper Voltage Warning Level
- 2035_h: Lower Voltage Warning Level
- 2036_h: Open Loop Current Reduction Idle Time
- 2037_h: Open Loop Current Reduction Value/factor
- 2038_h: Brake Controller Timing
- $203A_h$: Homing On Block Configuration
- $203D_h$: Torque Window
- $203E_h$: Torque Window Time Out
- $203F_h$: Max Slippage Time Out
- 2057_h: Clock Direction Multiplier
- 2058_h: Clock Direction Divider
- $205B_h$: Clock Direction Or Clockwise/Counter Clockwise Mode
- $205C_h$: Virtual Encoder Configuration
- 2084_h: Bootup Delay
- 2290_h: PDI Control
- 2300_h: NanoJ Control
- 2410_h: NanoJ Init Parameters
- 2800_h: Bootloader And Reboot Settings
- 3210_h: Motor Drive Parameter Set
- 3212_h: Motor Drive Flags
- $321A_h$: Current Controller Parameters
- $321B_h$: Velocity Controller Parameters
- $321C_h$: Position Controller Parameters
- $321D_h$: Pre-control
- $321E_h$: Voltage Limit
- 3221_h: Analog Inputs Control
- 3240_h: Digital Inputs Control
- 3241_h: Digital Input Capture
- 3242_h: Digital Input Routing
- 3243_h: Digital Input Homing Capture
- 3250_h: Digital Outputs Control
- 3252_h: Digital Output Routing
- 3260_h: Pwm Output 0
- 3261_h: Pwm Output 1
- 3273_h: Generic SPI Hardware Configuration
- 3274_h: Generic SPI Mosi Data
- 3321_h: Analog Input Offsets
- 3322_h: Analog Input Numerators
- 3323_h: Analog Input Denominators

- 3700$_h$: Deviation Error Option Code
- 3701$_h$: Limit Switch Error Option Code
- 4013$_h$: HW Configuration
- 6040$_h$: Controlword
- 6042$_h$: Vl Target Velocity
- 6046$_h$: Vl Velocity Min Max Amount
- 6048$_h$: Vl Velocity Acceleration
- 6049$_h$: Vl Velocity Deceleration
- 604A$_h$: Vl Velocity Quick Stop
- 604C$_h$: Vl Dimension Factor
- 605A$_h$: Quick Stop Option Code
- 605B$_h$: Shutdown Option Code
- 605C$_h$: Disable Option Code
- 605D$_h$: Halt Option Code
- 605E$_h$: Fault Option Code
- 6060$_h$: Modes Of Operation
- 6065$_h$: Following Error Window
- 6066$_h$: Following Error Time Out
- 6067$_h$: Position Window
- 6068$_h$: Position Window Time
- 606D$_h$: Velocity Window
- 606E$_h$: Velocity Window Time
- 606F$_h$: Velocity Threshold
- 6070$_h$: Velocity Threshold Time
- 6071$_h$: Target Torque
- 6072$_h$: Max Torque
- 607A$_h$: Target Position
- 607B$_h$: Position Range Limit
- 607C$_h$: Home Offset
- 607D$_h$: Software Position Limit
- 607E$_h$: Polarity
- 607F$_h$: Max Profile Velocity
- 6081$_h$: Profile Velocity
- 6082$_h$: End Velocity
- 6083$_h$: Profile Acceleration
- 6084$_h$: Profile Deceleration
- 6085$_h$: Quick Stop Deceleration
- 6086$_h$: Motion Profile Type
- 6087$_h$: Torque Slope
- 6091$_h$: Gear Ratio
- 6092$_h$: Feed Constant
- 6096$_h$: Velocity Factor
- 6097$_h$: Acceleration Factor
- 6098$_h$: Homing Method
- 6099$_h$: Homing Speed
- 609A$_h$: Homing Acceleration
- 60A2$_h$: Jerk Factor
- 60A4$_h$: Profile Jerk
- 60A8$_h$: SI Unit Position
- 60A9$_h$: SI Unit Velocity
- 60B0$_h$: Position Offset
- 60B1$_h$: Velocity Offset
- 60B2$_h$: Torque Offset

- 60C1$_h$: Interpolation Data Record
- 60C2$_h$: Interpolation Time Period
- 60C4$_h$: Interpolation Data Configuration
- 60C5$_h$: Max Acceleration
- 60C6$_h$: Max Deceleration
- 60E8$_h$: Additional Gear Ratio - Motor Shaft Revolutions
- 60E9$_h$: Additional Feed Constant - Feed
- 60ED$_h$: Additional Gear Ratio - Driving Shaft Revolutions
- 60EE$_h$: Additional Feed Constant - Driving Shaft Revolutions
- 60F2$_h$: Positioning Option Code
- 60F8$_h$: Max Slippage
- 60FE$_h$: Digital Outputs
- 60FF$_h$: Target Velocity

8.5.4 Category: customer

- 2701$_h$: Customer Storage Area

8.5.5 Category: drive

- 3202$_h$: Motor Drive Submode Select
- 320D$_h$: Torque Of Inertia Factor
- 6073$_h$: Max Current
- 6080$_h$: Max Motor Speed

8.5.6 Category: tuning

- 2030$_h$: Pole Pair Count
- 2031$_h$: Max Motor Current
- 203B$_h$: I2t Parameters
- 3203$_h$: Feedback Selection
- 3380$_h$: Feedback Sensorless
- 3390$_h$: Feedback Hall
- 33A0$_h$: Feedback Incremental A/B/I 1
- 33B0$_h$: Feedback SSI 1
- 33B1$_h$: Feedback SSI 2
- 4021$_h$: Ballast Configuration
- 6075$_h$: Motor Rated Current
- 608F$_h$: Position Encoder Resolution
- 6090$_h$: Velocity Encoder Resolution
- 60E6$_h$: Additional Position Encoder Resolution - Encoder Increments
- 60EB$_h$: Additional Position Encoder Resolution - Motor Revolutions

8.5.7 Category: Ethernet

- 2010$_h$: IP-Configuration
- 2011$_h$: Static-IPv4-Address
- 2012$_h$: Static-IPv4-Subnet-Mask
- 2013$_h$: Static-IPv4-Gateway-Address

8.5.8 Starting the save process

CAUTION!

Uncontrolled motor movements!

Control may be affected while saving. Unforeseen reactions can result.

► The motor must be at a standstill before starting the saving process. The motor must not be started while saving.

NOTICE

- Saving may take a few seconds. Never interrupt the power supply while saving. The state of the saved objects is otherwise undefined.
- Always wait until the controller has signaled that the save process has been successfully completed with the value "1" in the corresponding subindex in object 1010_h.

There is a subindex in object 1010_h for each *category*. To save all objects of this *category*, the value "65766173_h" must be written in the subindex. [1] The controller signals the end of the save process by overwriting the value with a "1".

The following table shows which subindex of object 1010_h is responsible for which *category*.

Subindex	Category
01_h	All categories
02_h	Communication
03_h	Application
04_h	Customer
05_h	Drive
06_h	Tuning

8.5.9 Discarding the saved data

If all objects or one *category* of saved objects is to be deleted, value "$64616F6C_h$" must be written in object 1011_h. [2] The following subindices correspond to a *category* here:

Subindex	Category
01_h	All categories (reset to factory settings) with the exception of 06_h (Tuning)
02_h	Communication
03_h	Application
04_h	Customer
05_h	Drive
06_h	Tuning

1 This corresponds to the decimal of 1702257011_d or the ASCII string `save`.
2 This corresponds to the decimal of 1684107116_d or the ASCII string `load`.

The saved objects are subsequently discarded; the change does not take effect until after the controller is restarted. You can restart the controller by entering the value "746F6F62$_h$" in 2800$_h$:01$_h$.

NOTICE

- Objects of *category* 06$_h$ (Tuning) are determined by Auto setup and are not reset when resetting to factory settings with subindex 01$_h$ (thereby making it unnecessary to again perform an auto setup). You can reset these objects with subindex 06$_h$.

8.5.10 Verifying the configuration

Object 1020$_h$ can be used to verify the configuration. It acts as a modification marker similar to common text editors: as soon as a file is modified in the editor, a marker (usually an asterisk) is added.

The entries of object 1020$_h$ can be written with a date and time and then saved together with all other savable objects with 1010$_h$:01.

The entries of 1020$_h$ are reset to "0" as soon as a savable object (including 1010$_h$:0x$_h$, except for 1010$_h$:01$_h$ and 1020$_h$) is written.

The following sequence makes verification possible:

1. An external tool or master configures the controller.
2. The tool or master sets the value in object 1020$_h$.
3. The tool or master activates the saving of all objects 1010$_h$:01$_h$ = 65766173$_h$. The date and time in object 1020$_h$ are also saved.

After the controller is restarted, the master can check the value in 1020$_h$:01$_h$ and 1020:01$_h$. If one of the values is "0", the object dictionary was changed after the saved values were loaded. If the date or time in 1020 does not correspond to the expected value, objects were probably saved with values other than those that were expected.

9 Modbus TCP

The controller can be controlled by means of Modbus TCP. In this chapter, the function codes of the Modbus communication structure are described.

Modbus references: www.modbus.org.

- *MODBUS APPLICATION PROTOCOL SPECIFICATION V1.1b3*, Date: 26.04.2014, Version: 1.1b3
- *MODBUS Messaging on TCP/IP Implementation Guide V1.0b*, Date: 24.10.2006, Version: 1.0b

All messages are sent to port 502 of the controller via TCP; only one connection is supported. No CRC (as it is used with Modbus RTU) takes place.

The I/O data with any preconfigured drive values (see Process data objects (PDO)) can be sent with the standard Modbus function codes. To configure your own I/O data, however, function code 2Bh (CAN Encapsulation) must be supported by the Modbus master in order for the parameters to be read and written independent of the process image.

If the master does not support this function code, the I/O image can be configured and stored using *Plug & Drive Studio*. The master can then access the data using the standard Modbus function codes.

Otherwise, configuration via the configuration file is possible (see chapter Configuration file) or the use of *Plug & Drive interface* (see document *Function description Plug & Drive interface*).

9.1 General

Modbus is generally big-endian based.

The only exceptions are the commands with function codes 43 ($2B_h$), 101 (65_h) and 102 (66_h), which are based on CANopen. For the data values of these commands, the little-endian format applies. The remainder of the Modbus message is, on the other hand, based on big-endian.

Example

Command $2B_h$: With this command, the value `12345678`$_h$ is written in object 0123_h (does not exist):

MBAP	FC	Data
`00 00 00 00 00 11 00`	`2B`	`0D 01 00 01 23 01 00 00 00 00 04` **`78 56 34 12`**

MBAP
: Modbus Application Protocol Header (see MBAP Header for details)

FC
: Function code

Data
: Data range, decoding is dependent on the used function code

9.2 MBAP Header

With Modbus TCP, a *Modbus Application Protocol Header* (*MBAP Header* for short) precedes the actual message.

MBAP Header	Function code	Data

This header consists of the following parts:

Name	Length	Value
Transaction Identifier	2 bytes	
Protocol Identifier	2 bytes	0000_h (Modbus)
Length	2 bytes	
Unit Identifier	1 byte	00_h

The MBAP Header is 7 bytes long:

Transaction Identifier

It is used for transaction pairing; the server (the controller) copies the value from the client's request in the response. If the client increases the number on each query, the response can be uniquely assigned to the query.

Protocol Identifier

Because a Modbus protocol is used, the field always has the value 0.

Length

The length of the data including the *Unit Identifier* field (1 byte), *the function code* (1 byte) and the data.

Unit Identifier

This field is used for internal system routing. Because the controller does not support routing, the field always has the value 0.

9.3 Function codes

The following "function codes" are supported:

Name		Function code	Subfunction code
Data access (16-bit)	Read Holding Registers	03 (03_h)	
	Read Input Register	04 (04_h)	
	Write Single Register	06 (06_h)	
	Write Multiple Registers	16 (10_h)	
	Read/Write Multiple Registers	23 (17_h)	
Miscellaneous	Encapsulated Interface Transport	43 ($2B_h$)	13 ($0D_h$)
	Read complete object dictionary start	101 (65_h)	85 (55_h)
	Read complete object dictionary next	101 (65_h)	170 (AA_h)
	Read complete array or record start	102 (66_h)	85 (55_h)
	Read complete array or record next	102 (66_h)	170 (AA_h)

9.4 Function code descriptions

9.4.1 FC 3 (03_h) Read Input Registers / FC 4 (04_h) Read Holding Registers

With this function code, one 16-bit value or multiple 16-bit values can be read. This function can be applied to NanoJ objects (see NanoJ objects) or process data objects (min. 4-byte alignment, see Process data objects (PDO)).

Request		
Name	**Length**	**Value**
Transaction Identifier	2 bytes	0000_h
Protocol Identifier	2 bytes	0000_h
Length	2 bytes	0006_h
Unit Identifier	1 byte	00_h
Function code	1 byte	03_h / 04_h
Start address	2 bytes	0000_h to $FFFF_h$
Number of registers	2 bytes	1 to ($7D_h$)

Response ("M" corresponds to the number of registers to be read)		
Name	**Length**	**Value**
Transaction Identifier	2 bytes	0000_h
Protocol Identifier	2 bytes	0000_h
Length	2 bytes	0003_h + 2*M
Unit Identifier	1 byte	00_h
Function code	1 byte	03_h / 04_h
Number of bytes	1 byte	2 * M
Register value	2 bytes	

Error		
Name	**Length**	**Value**
Transaction Identifier	2 bytes	0000_h
Protocol Identifier	2 bytes	0000_h
Length	2 bytes	0003_h
Unit Identifier	1 byte	00_h
Error code	1 byte	83_h / 84_h
Exception code (see Exception codes)	1 byte	01, 02, 03 or 04

Example

Below is an example of a read request and response of register 5000 (1388_h) and of the following register (2 registers):

Request

MBAP	FC	Data
00 00 00 00 00 06 00	03	13 88 00 02

Response

MBAP	FC	Data
00 00 00 00 00 07 00	03	04 02 40 00 00

9.4.2 FC 6 (06_h) Write Single Register

This function code can be used to write a single 16-bit value. The function can be used on process data objects (see Process data objects (PDO)).

Request		
Name	**Length**	**Value**
Transaction Identifier	2 bytes	0000_h
Protocol Identifier	2 bytes	0000_h
Length	2 bytes	0006_h
Unit Identifier	1 byte	00_h
Function code	1 byte	06_h
Register address	2 bytes	0000_h to $FFFF_h$
Register value	2 bytes	0000_h to $FFFF_h$

Response		
Name	**Length**	**Value**
Transaction Identifier	2 bytes	0000_h
Protocol Identifier	2 bytes	0000_h
Length	2 bytes	0006_h
Unit Identifier	1 byte	00_h
Function code	1 byte	06_h
Register address	2 bytes	0000_h to $FFFF_h$
Register value	2 bytes	0000_h to $FFFF_h$

Error		
Name	**Length**	**Value**
Transaction Identifier	2 bytes	0000_h
Protocol Identifier	2 bytes	0000_h
Length	2 bytes	0003_h
Unit Identifier	1 byte	00_h
Error code	1 byte	86_h
Exception code (see Exception codes)	1 byte	01, 02, 03 or 04

Example

Below is an example of a write request and response in register 6000 (1770_h) with the value "0001_h":

Request

MBAP	FC	Data
00 00 00 00 00 06 00	06	17 70 00 01

Response

MBAP	FC	Data
00 00 00 00 00 06 00	06	17 70 00 01

9.4.3 FC 16 (10_h) Write Multiple Registers

With this function code, one 16-bit value or multiple 16-bit values can be written. The function can be applied to NanoJ objects (see Process data objects (PDO)) or process data objects (see NanoJ objects).

Request ("N" is the number of registers to be written)		
Name	**Length**	**Value**
Transaction Identifier	2 bytes	0000_h
Protocol Identifier	2 bytes	0000_h
Length	2 bytes	0007_h + N * 2
Unit Identifier	1 byte	00_h
Function code	1 byte	10_h
Start address	2 bytes	0000_h to $FFFF_h$
Number of registers	2 bytes	0001_h to $007B_h$
Number of bytes	1 byte	2 * N
Register value	N * 2 bytes	

Response		
Name	**Length**	**Value**
Transaction Identifier	2 bytes	0000_h
Protocol Identifier	2 bytes	0000_h
Length	2 bytes	0006_h
Unit Identifier	1 byte	00_h
Function code	1 byte	10_h
Start address	2 bytes	0000_h to $FFFF_h$
Number of registers	2 bytes	0001_h to $007B_h$

Error		
Name	**Length**	**Value**
Transaction Identifier	2 bytes	0000_h
Protocol Identifier	2 bytes	0000_h

Error		
Name	**Length**	**Value**
Length	2 bytes	0003_h
Unit Identifier	1 byte	00_h
Error code	1 byte	90_h
Exception code (see Exception codes)	1 byte	01, 02, 03 or 04

Example

Below is an example for writing values "0102_h" and "0304_h" starting with register address 6000 (1770_h), number of registers is 2, length of the data is 4:

Request

MBAP	FC	Data
00 00 00 00 00 0B 00	10	17 70 00 02 04 01 02 03 04

Response

MBAP	FC	Data
00 00 00 00 00 06 00	10	17 70 00 02

9.4.4 FC 23 (17_h) Read/Write Multiple registers

With this function code, one 16-bit value or multiple 16-bit values can be simultaneously read and written. The function can be applied to NanoJ objects (see Process data objects (PDO)) or process data objects (see NanoJ objects).

Request ("N" is the number of registers to be read):		
Name	**Length**	**Value**
Transaction Identifier	2 bytes	0000_h
Protocol Identifier	2 bytes	0000_h
Length	2 bytes	$000B_h$ + 2 * N
Unit Identifier	1 byte	00_h
Function code	1 byte	17_h
Read: Start address	2 bytes	0000_h to $FFFF_h$
Read: Number of registers	2 bytes	0001_h to 0079_h
Write: Start address	2 bytes	0000_h to $FFFF_h$
Write: Number of registers	2 bytes	0001_h to 0079_h
Write: Number of bytes	1 byte	2 * N
Write: Register value	N * 2 bytes	

Response ("M" corresponds to the number of bytes to be written):		
Name	**Length**	**Value**
Transaction Identifier	2 bytes	0000_h
Protocol Identifier	2 bytes	0000_h

Response ("M" corresponds to the number of bytes to be written):		
Name	**Length**	**Value**
Length	2 bytes	0003_h + 2 * M
Unit Identifier	1 byte	00_h
Function code	1 byte	17_h
Number of bytes	1 byte	2 * M
Registers read	M * 2 bytes	

Error		
Name	**Length**	**Value**
Transaction Identifier	2 bytes	0000_h
Protocol Identifier	2 bytes	0000_h
Length	2 bytes	0003_h
Unit Identifier	1 byte	00_h
Error code	1 byte	97_h
Exception code (see Exception codes)	1 byte	01, 02, 03 or 04

Example

Below is an example for reading two registers beginning with register 5000 (1388_h) and for writing two registers beginning with register 6000 (1770_h) with 4 bytes and data "0102_h" and "0304_h":

Request

MBAP	FC	Data
00 00 00 00 00 0F 00	17	13 88 00 02 17 70 00 02 04 01 02 03 04

Response

MBAP	FC	Data
00 00 00 00 00 07 00	17	04 02 40 00 00

9.4.5 FC 43 ($2B_h$) Encapsulated Interface Transport

This function facilitates simple access of the CANopen object dictionary. Further details can be found in the following documentation:

1. *MODBUS APPLICATION PROTOCOL SPECIFICATION V1.1b3*, Date: 26.04.2014, Version: 1.1b3
2. *CiA 309 Draft Standard Proposal - Access from other networks - Part 2: Modbus/TCP mapping V1.3*, Date: 30.07.2015, Version: 1.3

NOTICE

For the messages of the Encapsulated Interface Transport, another byte sequence applies in part, see chapter General.

Definition of the request and response:

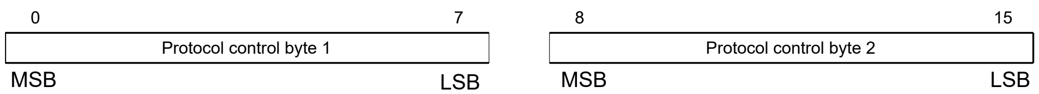
Name	Length	Example/number range
Transaction Identifier	2 bytes	0000_h
Protocol Identifier	2 bytes	0000_h
Length	2 bytes	$00NN_h$
Unit Identifier	1 byte	00_h
Function code	1 byte	$2B_h$ (43_d)
MEI type	1 byte	$0D_h$ (13_d)
Protocol options Range	2 to 5 bytes	
Address and data range	N bytes	

Protocol options Range

Name	Length	Example/number range
Protocol control	1 to 2 bytes	See description
Reserved	1 byte	Always 0
(Optional) Counter byte	1 byte	
(Optional) Network ID	1 byte	
(Optional) Encoded data	1 byte	

Protocol control:

The "Protocol control" field contains the flags that are needed for controlling the message protocols. The bytes of the "Protocol control" field are defined as follows if the "extended" flag was set (the second byte is otherwise omitted):



The most significant bit (MSB) is bit 0 for "protocol control" byte 1 and bit 8 for "protocol control" byte 2. The least significant bit (LSB) is bit 7 for "protocol control" byte 1 and bit 15 for "protocol control" byte 2.

Bit	Name	Description
0	"Extended" flag	This bit is used if the object dictionary data set is larger than would fit in a Modbus command. The data set then spans over multiple Modbus messages; each message contains part of the data set. "0" = No multiple message transaction or the end of the multiple message transaction. "1" = Part of a multiple message transaction.
1	Extended protocol control	Length of the protocol control, the value "0" indicates a length of 1 byte, the value "1" indicates a length of 2 bytes.
2	Counter byte option	This bit is set to "1" to indicate that the "counter byte" field is used in this message. If this bit is set to "0", the "counter byte" field does not exist in this message.
3 and 4	Reserved	0
5	Network ID option	Not supported, must be "0".

Bit	Name	Description
6	Encoded data option	Not supported, must be "0".
7	Access flag	This bit indicates the access method of the requested command. "0" = read, "1" = write.
8 to 15	Reserved	0

Address and data range

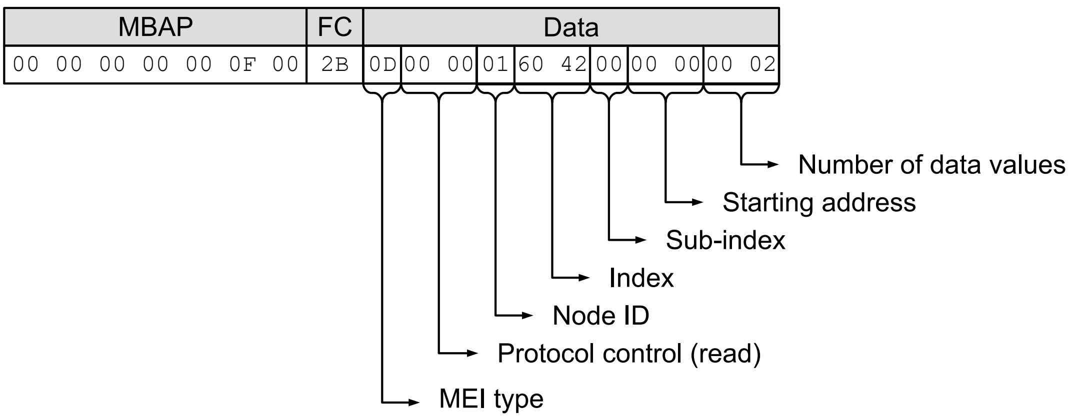
The address and data range is defined in the following table:

Name	Byte size and byte order	Example / range
Node-ID	1 byte	01_h to $7F_h$
Index	1 byte, high	0000_h to $FFFF_h$
	1 byte, low	
Subindex	1 byte	00_h to FF_h
Start address	1 byte, high	0000_h to $FFFF_h$
	1 byte, low	
Number of data values	1 byte, high	0000_h to $00FD_h$
	1 byte, low	
Write/read data	n bytes	The data are encoded as described in chapter General.

Example:

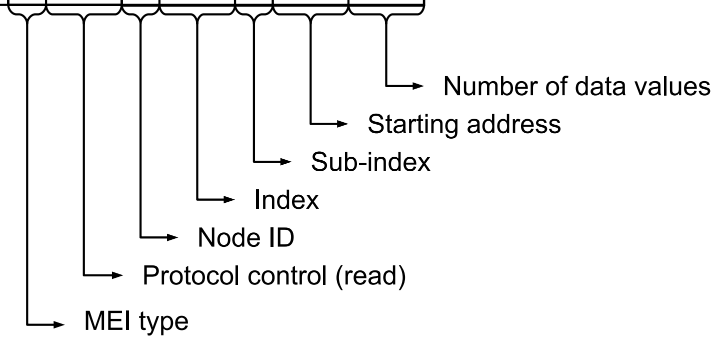
To read object $6042_h:00_h$ (16-bit value), the following message must be sent by the master (all values are in hexadecimal notation).

Request



Response

MBAP	FC	Data						
00 00 00 00 0F 00	2B	0D	00 00	01	60 42	00	00 00	00 02



Shown as an additional example below, a sequence of Modbus messages is sent from the master to the slave to rotate the motor in "Velocity" mode:

Set 6060 = "02$_h$" (Velocity mode)

Request

MBAP	FC	Data
00 00 00 00 00 0E 00	2B	0D 01 00 01 60 60 00 00 00 00 01 02

Response

MBAP	FC	Data
00 00 00 00 00 0D 00	2B	0D 01 00 01 60 60 00 00 00 00 00

Set 2031 = 03E8$_h$" (1000 mA)

Request

MBAP	FC	Data
00 01 00 00 00 12 00	2B	0D 01 00 01 20 31 00 00 00 00 04 E8 03 00 00

Response

MBAP	FC	Data
00 01 00 00 00 0D 00	2B	0D 01 00 01 20 31 00 00 00 00 00

Set 6040 = "00$_h$"

Request

MBAP	FC	Data
00 02 00 00 00 0F 00	2B	0D 01 00 01 60 40 00 00 00 00 02 00 00

Response

MBAP	FC	Data
00 02 00 00 00 0D 00	2B	0D 01 00 01 60 40 00 00 00 00 00

Set 6040 = "80$_h$"

Request

MBAP	FC	Data
00 03 00 00 00 0F 00	2B	0D 01 00 01 60 40 00 00 00 00 02 80 00

Response

MBAP	FC	Data
00 03 00 00 00 0D 00	2B	0D 01 00 01 60 40 00 00 00 00 00

Set 6040 = "06$_h$"

Request

MBAP	FC	Data
00 04 00 00 00 0F 00	2B	0D 01 00 01 60 40 00 00 00 00 02 06 00

Response

MBAP	FC	Data
00 04 00 00 00 0D 00	2B	0D 01 00 01 60 40 00 00 00 00 00

Set 6040 = "07$_h$"

Request

MBAP	FC	Data
00 05 00 00 00 0F 00	2B	0D 01 00 01 60 40 00 00 00 00 02 07 00

Response

MBAP	FC	Data
00 05 00 00 00 0D 00	2B	0D 01 00 01 60 40 00 00 00 00 00

Set 6040 = "0F$_h$"

Request

MBAP	FC	Data
00 06 00 00 00 0F 00	2B	0D 01 00 01 60 40 00 00 00 00 02 0F 00

Response

MBAP	FC	Data
00 06 00 00 00 0D 00	2B	0D 01 00 01 60 40 00 00 00 00 00

Below are two examples for reading an object:

Read 6041$_h$:00$_h$

Request

MBAP	FC	Data
00 00 00 00 00 0D 00	2B	0D 00 00 01 60 41 00 00 00 00 02

Response

Read 6061$_h$:00$_h$

Request

MBAP	FC	Data
00 00 00 00 00 0D 00	2B	0D 00 00 01 60 61 00 00 00 00 01

Response

MBAP	FC	Data
00 00 00 00 00 0D 00	2B	0D 00 00 01 60 61 00 00 00 00 01 00

9.4.5.1 Error reaction

In the event of an error, the following error message is sent:

Name	Length	Example value
Transaction Identifier	2 bytes	0000$_h$
Protocol Identifier	2 bytes	0000$_h$
Length	2 bytes	000B$_h$
Unit Identifier	1 byte	00$_h$
Function code	1 byte	2B$_h$ +80$_h$ (171$_d$ = 43$_d$ + 128$_d$) (indicates error)
Modbus exception code	1 byte	FF$_h$ ("extended exception")
Extended exception length	2 bytes	6
MEI type	1 byte	0D$_h$
Exception code	1 byte	CE$_h$
Error code	4 bytes	CANopen error code, see following table

CANopen error code	Description
FFFF0000$_h$	*Abort no error*
FFFF1003$_h$	Service is not supported
FFFF1004$_h$	Gap in counter byte of the *Protocol control* field

CANopen error code	Description
FFFF0003$_h$	Unknown or invalid command
FFFF0008$_h$	Access to the object is not supported
FFFF000E$_h$	General error in the parameter
FFFF0011$_h$	Length of parameter incorrect
FFFF0012$_h$	Parameter too long
FFFF0013$_h$	Parameter too short
FFFF0015$_h$	Parameter data outside of the permissible value range (for write commands)
FFFF0016$_h$	Parameter data exceed the permissible value range (for write commands)
FFFF0017$_h$	Parameter data below the permissible value range (for write commands)
FFFF0018$_h$	Maximum entered values less than minimum values
FFFF0019$_h$	General error
FFFF001E$_h$	Requested object is too large for single message
FFFF1004$_h$	Invalid sequence of messages (e. g., if the value of the *counter byte* is not correct according to the previous request or response)

In the event that the unsupported control option bit is set, the following error message is sent:

Name	Length	Example value
Transaction Identifier	2 bytes	0000$_h$
Protocol Identifier	2 bytes	0000$_h$
Length	2 bytes	0008/0009$_h$
Unit Identifier	1 byte	00$_h$
Function code	1 byte	2B$_h$ +80$_h$ (171$_d$ = 43$_d$ + 128$_d$) (indicates error)
Modbus exception code	1 byte	FF$_h$ ("extended exception")
Extended exception length	2 bytes	2 + length of "supported protocol control"
MEI type	1 byte	0D$_h$
Exception code	1 byte	AE$_h$
Supported protocol control	1 or 2 bytes	See following table

Bit	Name	Description
0	"Extended" flag	This bit is used if the object dictionary data set is larger than would fit in a Modbus command. The data set then spans over multiple Modbus messages; each message contains part of the data set. "0" = No multiple message transaction or the end of the multiple message transaction. "1" = Part of a multiple message transaction.
1	Extended protocol control	Length of the protocol control, the value "0" indicates a length of 1 byte, the value "1" indicates a length of 2 bytes.
2	Counter byte option	This bit is set to "1" to indicate that the "counter byte" field is used in this message. If this bit is set to "0", the "counter byte" field does not exist in this message.

Bit	Name	Description
3 and 4	Reserved	0
5	Network ID option	Not supported, must be "0".
6	Encoded data option	Not supported, must be "0".
7	Access flag	This bit indicates the access method of the requested command. "0" = read, "1" = write.
8 to 15	Reserved	0

The following example shows an error in the event of a faulty request. The request reads 6061$_h$:00 with a length of 2 bytes, but the object has a size of just 1 byte:

Request

MBAP	FC	Data
`00 00 00 00 00 0D 00`	`2B`	`0D 00 00 01 60 60 00 00 00 00 02`

Response

MBAP	FC	Data
`00 00 00 00 00 0B 00`	`2B`	`FF 00 06 0D CE 12 00 07 06`

9.4.6 FC 101 (65$_h$) Read complete object dictionary

This function code is used to read out the complete object dictionary.

To start or restart the reading out of the object dictionary, subfunction code `55`$_h$ must be sent. This code resets reading out of the object dictionary on object 0000$_h$. All subsequent object dictionary frames must then contain subfunction code `AA`$_h$. At the end, once all objects have been read out, an "Error Response" is generated with the abort code "No data available".

The format of each "read object" is as follows:

Request:

Name	Length	Value / note
Slave address	1 byte	
Function code	1 byte	65$_h$
Subfunction code	1 byte	55$_h$ or AA$_h$
Length of the data	1 byte	00$_h$
CRC	2 bytes	

Response:

Name	Length	Value / note
Slave address	1 byte	65$_h$
Function code	1 byte	
Subfunction code	1 byte	
Length of the data	1 byte	
n times "object dictionary frame"	1 - 252 bytes	
CRC	2 bytes	

An object dictionary frame consists of the following bytes:

Name	Value / note	
Index Low Byte	1 byte	
Index High Byte	1 byte	
Subindex	1 byte	
Number of bytes	1 byte	Number m of the valid data in the data field
Data byte	m-1 byte	

Example

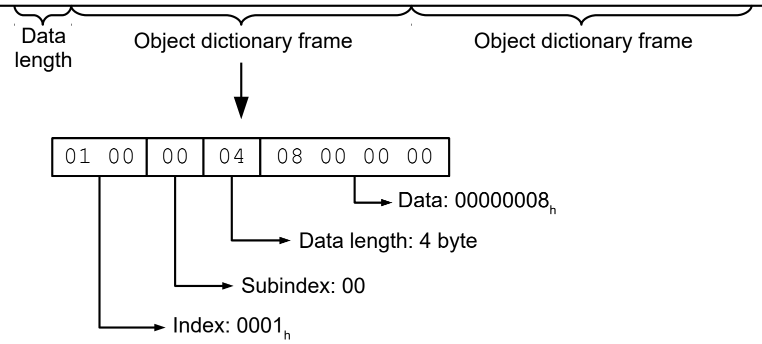
All of the following numerical values are in hexadecimal format.

Start reading of the object dictionary with request:

MBAP	FC	Data
00 00 00 00 00 04 00	65	55 00

The response is:

MBAP	FC	Data
00 00 00 00 NN NN 00	65	55 CE 01 00 00 04 08 00 00 00 02 00 00 04 08 00 00 00 ...



Read out the next part of the object dictionary with the request:

MBAP	FC	Data
00 01 00 00 00 04 00	65	AA 00

The response is:

MBAP	FC	Data
00 01 00 00 NN NN 00	65	AA CD 21 00 0A 02 07 00 21 00 0B 02 07 00 21 00 0C 02 ...

Repeat reading of the object dictionary with the previous request until the response is an error:

MBAP	FC	Data
NN NN 00 00 00 03 00	E5	0D

9.4.6.1 Error reaction

In the event of an error, the following error message is sent:

Name	Length	Example value
Transaction Identifier	2 bytes	0000_h
Protocol Identifier	2 bytes	0000_h
Length	2 bytes	$000B_h$
Unit Identifier	1 byte	00_h
Function code	1 byte	$2B_h$ +80_h ($171_d = 43_d + 128_d$) (indicates error)
Modbus exception code	1 byte	FF_h ("extended exception")
Extended exception length	2 bytes	6
MEI type	1 byte	$0D_h$
Exception code	1 byte	CE_h
Error code	4 bytes	CANopen error code, see following table

CANopen error code	Description
$FFFF0000_h$	*Abort no error*
$FFFF1003_h$	Service is not supported
$FFFF1004_h$	Gap in counter byte of the *Protocol control* field
$FFFF0003_h$	Unknown or invalid command
$FFFF0008_h$	Access to the object is not supported
$FFFF000E_h$	General error in the parameter
$FFFF0011_h$	Length of parameter incorrect
$FFFF0012_h$	Parameter too long
$FFFF0013_h$	Parameter too short
$FFFF0015_h$	Parameter data outside of the permissible value range (for write commands)
$FFFF0016_h$	Parameter data exceed the permissible value range (for write commands)
$FFFF0017_h$	Parameter data below the permissible value range (for write commands)
$FFFF0018_h$	Maximum entered values less than minimum values
$FFFF0019_h$	General error
$FFFF001E_h$	Requested object is too large for single message
$FFFF1004_h$	Invalid sequence of messages (e. g., if the value of the *counter byte* is not correct according to the previous request or response)

In the event that the unsupported control option bit is set, the following error message is sent:

Name	Length	Example value
Transaction Identifier	2 bytes	0000_h
Protocol Identifier	2 bytes	0000_h
Length	2 bytes	$0008/0009_h$
Unit Identifier	1 byte	00_h
Function code	1 byte	$2B_h$ +80_h ($171_d = 43_d + 128_d$) (indicates error)
Modbus exception code	1 byte	FF_h ("extended exception")
Extended exception length	2 bytes	2 + length of "supported protocol control"

Name	Length	Example value
MEI type	1 byte	$0D_h$
Exception code	1 byte	AE_h
Supported protocol control	1 or 2 bytes	See following table

Bit	Name	Description
0	"Extended" flag	This bit is used if the object dictionary data set is larger than would fit in a Modbus command. The data set then spans over multiple Modbus messages; each message contains part of the data set. "0" = No multiple message transaction or the end of the multiple message transaction. "1" = Part of a multiple message transaction.
1	Extended protocol control	Length of the protocol control, the value "0" indicates a length of 1 byte, the value "1" indicates a length of 2 bytes.
2	Counter byte option	This bit is set to "1" to indicate that the "counter byte" field is used in this message. If this bit is set to "0", the "counter byte" field does not exist in this message.
3 and 4	Reserved	0
5	Network ID option	Not supported, must be "0".
6	Encoded data option	Not supported, must be "0".
7	Access flag	This bit indicates the access method of the requested command. "0" = read, "1" = write.
8 to 15	Reserved	0

The following example shows an error in the event of a faulty request. The request reads 6061_h:00 with a length of 2 bytes, but the object has a size of just 1 byte:

Request

MBAP	FC	Data
00 00 00 00 00 0D 00	2B	0D 00 00 01 60 60 00 00 00 00 02

Response

MBAP	FC	Data
00 00 00 00 00 0B 00	2B	FF 00 06 0D CE 12 00 07 06

9.4.7 FC 102 (66_h) Read complete array or record

This function code is used to read out the complete array or record from the object dictionary.

To start or restart the reading out of the array, subfunction code 55_h must be sent. This code resets reading out on the object with subindex 00_h. All subsequent requests must then contain subfunction code AA_h. At the end, once all objects have been read out, an "Error Response" is generated.

The format of each "read object" is as follows:

Request:

Name	Length	Value / note
Transaction Identifier	2 bytes	0000_h
Protocol Identifier	2 bytes	0000_h
Length	2 bytes	0007_h
Unit Identifier	1 byte	00_h
Function code	1 byte	66_h
Subfunction code	1 byte	55_h or AA_h
Length of the data	1 byte	00_h
Index of the array to be read	2 bytes	

Response:

Name	Length	Value / note
Transaction Identifier	2 bytes	0000_h
Protocol Identifier	2 bytes	0000_h
Length	2 bytes	0004_h+n
Unit Identifier	1 byte	00_h
Function code	1 byte	
Subfunction code	1 byte	
Length of the data	1 byte	
n times object dictionary frame	1 - 252 bytes	

An object dictionary frame consists of the following bytes:

Name		Value / note
Index Low Byte	1 byte	
Index High Byte	1 byte	
Subindex	1 byte	
Number of bytes	1 byte	Number m of the valid data in the data field
Data byte	m-1 byte	

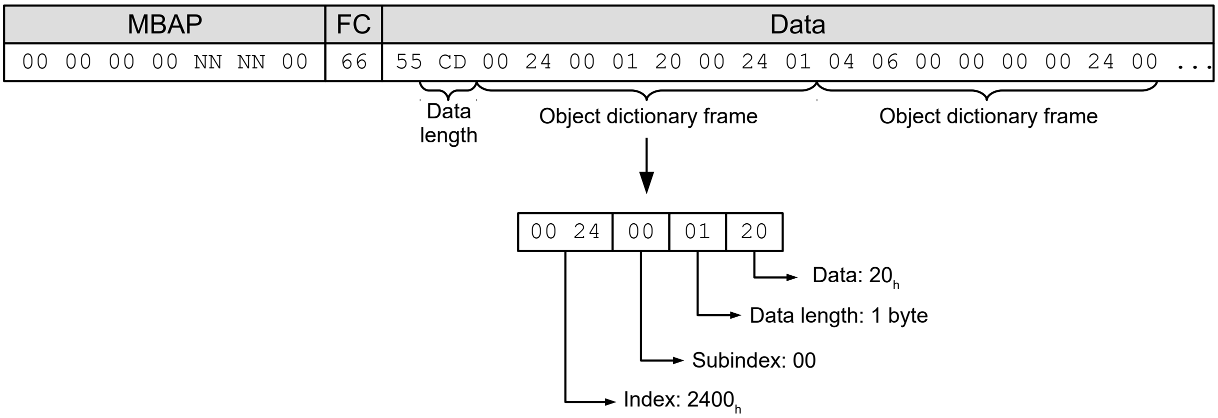
Example

All of the following numerical values are in hexadecimal format; the index of the object that is to be read is 2400_h.

Start reading of the array with request:

MBAP	FC	Data
00 00 00 00 00 06 00	66	55 00 24 00

The response is:



9.4.7.1 Error reaction

In the event of an error, the following error message is sent:

Name	Length	Example value
Transaction Identifier	2 bytes	0000_h
Protocol Identifier	2 bytes	0000_h
Length	2 bytes	$000B_h$
Unit Identifier	1 byte	00_h
Function code	1 byte	$2B_h$ +80_h (171_d = 43_d + 128_d) (indicates error)
Modbus exception code	1 byte	FF_h ("extended exception")
Extended exception length	2 bytes	6
MEI type	1 byte	$0D_h$
Exception code	1 byte	CE_h
Error code	4 bytes	CANopen error code, see following table

CANopen error code	Description
$FFFF0000_h$	*Abort no error*
$FFFF1003_h$	Service is not supported
$FFFF1004_h$	Gap in counter byte of the *Protocol control* field
$FFFF0003_h$	Unknown or invalid command
$FFFF0008_h$	Access to the object is not supported
$FFFF000E_h$	General error in the parameter
$FFFF0011_h$	Length of parameter incorrect
$FFFF0012_h$	Parameter too long
$FFFF0013_h$	Parameter too short
$FFFF0015_h$	Parameter data outside of the permissible value range (for write commands)
$FFFF0016_h$	Parameter data exceed the permissible value range (for write commands)
$FFFF0017_h$	Parameter data below the permissible value range (for write commands)
$FFFF0018_h$	Maximum entered values less than minimum values
$FFFF0019_h$	General error

CANopen error code	Description
FFFF001E$_h$	Requested object is too large for single message
FFFF1004$_h$	Invalid sequence of messages (e. g., if the value of the *counter byte* is not correct according to the previous request or response)

In the event that the unsupported control option bit is set, the following error message is sent:

Name	Length	Example value
Transaction Identifier	2 bytes	0000$_h$
Protocol Identifier	2 bytes	0000$_h$
Length	2 bytes	0008/0009$_h$
Unit Identifier	1 byte	00$_h$
Function code	1 byte	2B$_h$ +80$_h$ (171$_d$ = 43$_d$ + 128$_d$) (indicates error)
Modbus exception code	1 byte	FF$_h$ ("extended exception")
Extended exception length	2 bytes	2 + length of "supported protocol control"
MEI type	1 byte	0D$_h$
Exception code	1 byte	AE$_h$
Supported protocol control	1 or 2 bytes	See following table

Bit	Name	Description
0	"Extended" flag	This bit is used if the object dictionary data set is larger than would fit in a Modbus command. The data set then spans over multiple Modbus messages; each message contains part of the data set. "0" = No multiple message transaction or the end of the multiple message transaction. "1" = Part of a multiple message transaction.
1	Extended protocol control	Length of the protocol control, the value "0" indicates a length of 1 byte, the value "1" indicates a length of 2 bytes.
2	Counter byte option	This bit is set to "1" to indicate that the "counter byte" field is used in this message. If this bit is set to "0", the "counter byte" field does not exist in this message.
3 and 4	Reserved	0
5	Network ID option	Not supported, must be "0".
6	Encoded data option	Not supported, must be "0".
7	Access flag	This bit indicates the access method of the requested command. "0" = read, "1" = write.
8 to 15	Reserved	0

The following example shows an error in the event of a faulty request. The request reads 6061$_h$:00 with a length of 2 bytes, but the object has a size of just 1 byte:

Request

MBAP	FC	Data
00 00 00 00 00 0D 00	2B	0D 00 00 01 60 60 00 00 00 00 02

Response

MBAP	FC	Data
00 00 00 00 00 0B 00	2B	FF 00 06 0D CE 12 00 07 06

9.4.8 Exception codes

In case of an error, the following exception codes may be contained in the response depending on the function code:

Code	Name	Description
01	Illegal Function	Function code not recognized/allowed
02	Illegal Data Address	Register address not valid or does not exist
03	Illegal Data Value	Value not valid
04	Device Failure	Unrecoverable error

For further details, refer to Modbus specification *MODBUS APPLICATION PROTOCOL SPECIFICATION V1.1b3*.

9.5 Process data objects (PDO)

As with CANopen, a process image can be configured for input and output values with Modbus. This image only contains the data values of one or more objects without additional information, such as length, index or subindex. A single message can thereby be used to read or write multiple objects at the same time.

9.5.1 Configuration

The configuration of the image is referred to as "mapping" and is written in the following objects:

- 3502_h for the Modbus Rx (master → slave) PDO mapping
- 3602_h for Modbus Tx (slave → master) PDO mapping

Both objects contain an array of 16 entries each. Subindex 00 specifies the number of valid entries here.

Objects 3502_h and 3602_h can be written with messages with Modbus function code $2B_h$.

9.5.2 Transfer

The data are written sequentially in the message without gaps and alignment.

If alignment is required (e.g., 16-bit alignment), additional "dummy objects" can be incorporated in the message. Dummy objects are only ever transferred with the data value "0". These objects are listed in the following table.

Index	Data type
0002_h	Signed integer (8 bit)
0003_h	Signed integer (16 bit)
0004_h	Signed integer (32 bit)
0005_h	Unsigned integer (8 bit)

Index	Data type
0006$_h$	Unsigned integer (16 bit)
0007$_h$	Unsigned integer (32 bit)

Mapping is as follows:

- The PDO RX image begins at Modbus register address 6000$_d$ (1770$_h$).
- The PDO TX image begins at Modbus register address 5000$_d$ (1388$_h$).

Read/write access can be performed simultaneously with function code 17$_h$ or with the 03$_h$, 04$_h$, 06$_h$, 10$_h$ commands on the respective RX/TX images.

NOTICE

To be able to change the mapping, you must first deactivate it by setting the corresponding subindex 0$_h$ to "0".

After writing the objects to the respective subindices, enter the number of mapped objects in subindex 0$_h$.

Example

The following objects are to be set in the mapping:

- 3602$_h$:00$_h$ = "0$_h$" (mapping is deactivated)
- 3602$_h$:01$_h$ = "60410010$_h$" (object 6041$_h$:00$_h$, length 16 bits is mapped)
- 3602$_h$:02$_h$ = "00050008$_h$" (dummy object 0005$_h$:00$_h$, length 8 bits is mapped)
- 3602$_h$:03$_h$ = "60610008$_h$" (object 6061$_h$:00$_h$, length 8 bits is mapped)
- 3602$_h$:04$_h$ = "60640020$_h$" (object 6064$_h$:00$_h$, length 32 bits is mapped)
- 3602$_h$:05$_h$ = "60440010$_h$" (object 6044$_h$:00$_h$, length 16 bits is mapped)
- 3602$_h$:06$_h$ = "60FD0020$_h$" (object 60FD$_h$:00$_h$, length 32 bits is mapped)
- 3602$_h$:00$_h$ = "6$_h$" (6 values are mapped)

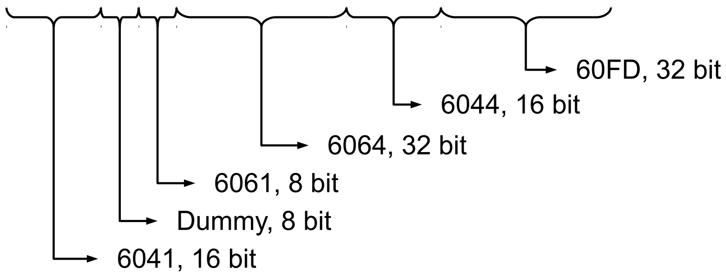
After the mapping for object 6061$_h$:00$_h$, a dummy object is inserted so that the next object 6064$_h$:00$_h$ can be aligned to 32 bit.

Rx message: The master sends the slave the following message:

MBAP	FC	Data
00 00 00 00 00 06 00	04	13 88 00 07

Tx message: The slave sends following response to the master:

MBAP	FC	Data						
00 00 00 00 00 12 00	04	0E	06 40	00	02	00 00 00 00	00 00	00 00 00 00



9.6 NanoJ objects

NanoJ objects 2400_h NanoJ Input and 2500_h (NanoJ Output) are, like the process image, mapped to the Modbus register:

- 2500_h with 32 x 32 bit values is mapped to the Modbus register address beginning with 2000_d ($7D0_h$) and can only be read in this way.
- 2400_h with 32 x 32 bit values is mapped to the Modbus register address beginning with 3000_d ($BB8_h$) and can only be written in this way.

To access, commands with function codes 03_h, 04_h, 10_h and 17_h can be used. For purposes of data consistency, the restriction that the address must be 32-bit aligned and that at least 32 bits must always be written during a write operation applies.

Example

Request: The master sends the slave the following message:

MBAP	FC	Data
00 00 00 00 00 1B 00	17	07 D0 00 08 0B B8 00 08 10 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

Reply: The slave sends the master the following response:

MBAP	FC	Data
00 00 00 00 00 13 00	17	10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

10 Programming with *NanoJ*

NanoJ is a programming language similar to *C* or *C++*. NanoJ is integrated in the *Plug & Drive Studio 3* software. You can find further information in document *Plug & Drive Studio 3: User Manual* at us.nanotec.com.

10.1 NanoJ program

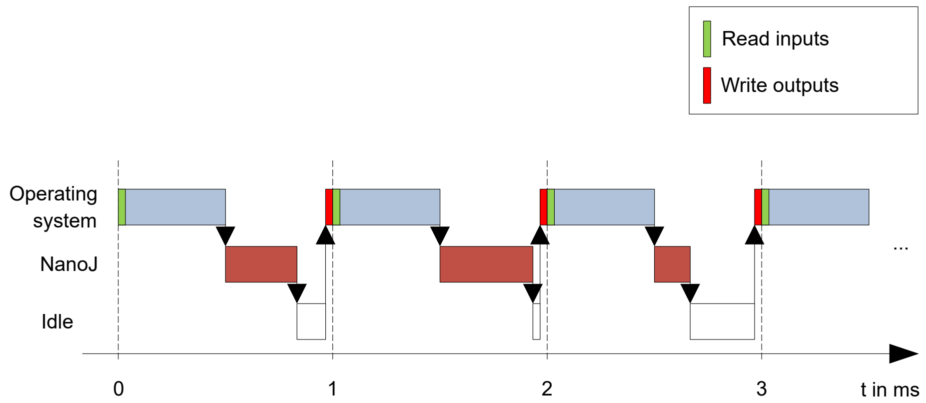
A *NanoJ program* makes a protected runtime environment available within the firmware. Here, the user can create his own processes. These can then trigger functions in the controller by, for example, reading or writing entries in the object dictionary.

Through the use of protective mechanisms, a *NanoJ program* is prevented from crashing the firmware. In the worst case, the execution is interrupted with an error code stored in the object dictionary.

If the *NanoJ program* was loaded on the controller, it is automatically executed after the controller is switched on or restarted, as long as you do not set bit 0 in object 2300_h to "0".

10.1.1 Available computing time

A *NanoJ program* receives computing time cyclically in a 1 ms clock (see following figure). Because computing time is lost through interrupts and system functions of the firmware, only approx. 30% – 50% of computing time is available to the user program (depending on control mode and application). In this time, the user program must run through the cycle and either complete the cycle or yield the computing time by calling the `yield()` function. In the former case, the user program is restarted with the start of the next 1 ms cycle; the latter results in the program being continued on the next 1 ms cycle with the command that follows the `yield()` function.



If the *NanoJ program* needs more time than was allotted, it is ended and an error code set in the object dictionary.

TIP

When developing user programs, the runtime behavior must be carefully examined, especially for more time-intensive tasks. For example, it is therefore recommended that tables be used instead of calculating a sine value using a `sin` function.

NOTICE

If the *NanoJ program* does not yield the computing time after too long a time, it is ended by the operating system. In this case, the number `4` is entered in the statusword for object 2301_h; in the error register for object 2302_h, the number `5` (timeout) is noted, see 2301h NanoJ Status and 2302h NanoJ Error Code.

To keep the *NanoJ program* from stopping, you can activate *AutoYield* mode by writing value "5" in 2300_h. In *AutoYield* mode, however, the *NanoJ program* is no longer real-time capable and no longer runs every 1 ms.

10.1.2 Protected runtime environment

Using process-specific properties, a so-called *protected runtime environment* is generated. A user program in the protected runtime environment is only able to access specially allocated memory areas and system resources. For example, an attempt to directly write to a processor IO register is acknowledged with an *MPU Fault* and the user program terminated with the corresponding error code in the object dictionary.

10.1.3 NanoJ program – communication possibilities

A *NanoJ program* has a number of possibilities for communicating with the controller:

- Read and write OD values using PDO mapping
- Directly read and write OD values via NanoJ functions
- Call other NanoJ functions (e.g., write debug output)

The OD values of the user program are made available in the form of variables via *PDO mapping*. Before a user program receives the 1 ms time slot, the firmware transfers the values from the object dictionary to the variables of the user program. As soon as the user program receives computing time, it can manipulate these variables as regular C variables. At the end of the time slot, the new values are then automatically copied by the firmware back to the respective OD entries.

To optimize the performance, three types of mapping are defined: input, output, and input/output (In, Out, InOut).

- *Input mappings* can only be read; they are not transferred back to the object dictionary.
- *Output mappings* can only be written.
- *Input/output mappings*, on the other hand, can both be read and written.

The set mappings can be read and checked via the GUI for objects 2310_h, 2320_h, and 2330_h. Up to 16 entries are allowed for each mapping.

Whether a variable is stored in the input, output or data range is controlled in *Plug & Drive Studio* via the specification of the *linker section*.

NanoJ inputs and NanoJ outputs

To communicate with the *NanoJ program* via the respective interface, you can use the following objects:

- 2400h NanoJ Inputs: Array with thirty-two S32 values for passing values to the *NanoJ program*
- 2410h NanoJ Init Parameters: Array with thirty-two S32 values. This object can be stored, unlike 2400_h.
- 2500h NanoJ Outputs: Array with thirty-two S32 values, where the *NanoJ program* can store values that can be read out via the fieldbus

10.1.4 Executing a NanoJ program

When executing a cycle, the *NanoJ program* essentially consists of the following three steps with respect to the PDO mapping:

1. Read values from the object dictionary and copy them to the input and output areas
2. Execute a user program
3. Copy values from the output and input areas back to the object dictionary

The configuration of the copy processes is based on the CANopen standard.

In addition, values of the object dictionary can be accessed via NanoJ functions. This is generally slower; mappings are therefore to be preferred. The number of mappings is limited (16 entries each in In/Out/InOut).

TIP

Nanotec recommends: Map OD entries that are used and changed frequently and use NanoJ function to access OD entries that are used less frequently.

A list of available NanoJ functions can be found in chapter NanoJ functions in the NanoJ program.

TIP

Nanotec recommends accessing a given OD value either by mapping or using a NanoJ function with `od_write()`. If both are used simultaneously, the NanoJ function has no effect.

10.1.5 NanoJ program – OD entries

The *NanoJ program* is controlled and configured in object range 2300_h to 2330_h (see 2300h NanoJ Control).

OD-Index	Name and description
2300_h	2300h NanoJ Control
2301_h	2301h NanoJ Status
2302_h	2302h NanoJ Error Code
2310_h	2310h NanoJ Input Data Selection
2320_h	2320h NanoJ Output Data Selection
2330_h	2330h NanoJ In/output Data Selection

Example:

To start the *TEST1.USR* user program, the following sequence can, for example, be used:

- Check entry 2302_h for error code.
- If no error:
  Start the *NanoJ program* by writing object 2300_h, bit 0 = "1" or by restarting the controller.

NOTICE

It can take up to 200 ms for the NanoJ program to start.

- Check entry 2302_h for error code and object 2301_h, bit 0 = "1".

To stop a running program: write entry 2300_h with bit 0 value = "0".

10.1.6 Structure of a NanoJ program

A user program consists of at least two instructions:

- the preprocessor instruction `#include "wrapper.h"`
- the `void user(){}` function

The code to be executed can be stored in the `void user()` function.

NOTICE

The file names of the user programs must not be longer than eight characters plus three characters in the suffix; file name `main.cpp` is permissible, file name `aLongFileName.cpp` is not permissible.

NOTICE

In *NanoJ programs*, global variables may only be initialized within functions. It then follows:

- No `new` operator
- No constructors
- No initialization of global variables outside of functions

Examples:

The global variable is to be initialized within the `void user()` function:

```
unsigned int i;
void user(){
 i = 1;
 i += 1;
}
```

The following assignment results in an error during compilation:

```
unsigned int i = 1;
 void user() {
 i += 1;
}
```

10.1.7 NanoJ program example

The example shows the programming of a square wave signal in object 2500_h:01_h.

```
// file main.cpp
map S32 outputReg1 as inout 0x2500:1
#include "wrapper.h"

// user program
void user()
{
  U16 counter = 0;
  while( 1 )
  {
    ++counter;

    if( counter < 100 )
     InOut.outputReg1 = 0;
    else if( counter < 200 )
      InOut.outputReg1 = 1;
    else
      counter = 0;

    // yield() 5 times (delay 5ms)
    for(U08 i = 0; i < 5; ++i )
      yield();
  }
}// eof
```

You can find other examples at us.nanotec.com.

10.2 Mapping in the NanoJ program

With this method, a variable in the *NanoJ program* is linked directly with an entry in the object dictionary. The creation of the mapping must be located at the start of the file here, even before the `#include "wrapper.h"` instruction.

TIP

Nanotec recommends:

- Use mapping if you need to access an object in the object dictionary frequently, e. g., *controlword* 6040_h or *statusword* 6041_h.
- The `od_write()` and `od_read()` functions are better suited for accessing objects a single time, see Accessing the object dictionary.

10.2.1 Declaration of the mapping

The declaration of the mapping is structured as follows:

```
map <TYPE> <NAME> as <input|output|inout> <INDEX>:<SUBINDEX>
```

Where:

- ```
  <TYPE>
  ```
  The data type of the variable; U32, U16, U08, S32, S16 or S08.
- `<NAME>`
  The name of the variable as it is used in the user program.
- ```
  <input|output|inout>
  ```
  The read and write permission of a variable: a variable can be declared as an `input`, `output` or `inout`. This defines whether a variable is readable (`input`), writable (`output`) or both (`inout`) and the structure by means of which it must be addressed in the program.
- ```
  <INDEX>:<SUBINDEX>
  ```
  Index and subindex of the object to be mapped in the object dictionary.

Each declared variable is addressed in the user program via one of the three structures: *In*, *Out* or *InOut* depending on the defined write and read direction.

NOTICE

A comment is only permitted above the respective mapping declaration in the code, not on the same line.

10.2.2 Example of mapping

Example of a mapping and the corresponding variable accesses:

```
// 6040h:00h is UNSIGNED16
map U16 controlWord as output 0x6040:00
// 6041h:00h is UNSIGNED16
map U16 statusWord as input 0x6041:00

// 6060h:00h is SIGNED08 (INTEGER8)
map S08 modeOfOperation as inout 0x6060:00
```
```

```
#include "wrapper.h"

void user()
{
  [...]
  Out.controlWord = 1;
  U16 tmpVar = In.statusword;
  InOut.modeOfOperation = tmpVar;
  [...]
}
```

10.2.3 Possible error at `od_write()`

A possible source of errors is a write access with the `od_write()` function (see NanoJ functions in the NanoJ program) of an object in the object dictionary that was simultaneously created as mapping. The code listed in the following is incorrect:

```
map U16 controlWord as output 0x6040:00
#include " wrapper.h"
void user()
{
 [...]
  Out.controlWord = 1;
  [...]
  od_write(0x6040, 0x00, 5 ); // der Wert wird durch das Mapping überschrieben
  [...]
}
```

The line with the `od_write(0x6040, 0x00, 5 );` command has no effect. As described in the introduction, all mappings are copied to the object dictionary at the end of each millisecond.

This results in the following sequence:

1. The `od_write` function writes the value `5` in object 6040_h:00_h.
2. At the end of the 1 ms cycle, the mapping is written that also specifies object 6040_h:00_h, however, with the value `1`.
3. From the perspective of the user, the `od_write` command thus serves no purpose.

10.3 NanoJ functions in the NanoJ program

With NanoJ functions, it is possible to call up functions integrated in the firmware directly from a user program. Code can only be directly executed in the protected area of the protected execution environment and is realized via so-called *Cortex Supervisor Calls* (Svc Calls). Here, an interrupt is triggered when the function is called, thereby giving the firmware the possibility to temporarily permit code execution outside of the protected execution environment. Developers of user programs do not need to worry about this mechanism – for them, the NanoJ functions can be called up like normal C functions. Only the *wrapper.h* file needs to be integrated as usual.

10.3.1 Accessing the object dictionary

void **od_write** (U32 index, U32 subindex, U32 value)

This function writes the transferred value to the specified location in the object dictionary.

index	Index of the object to be written in the object dictionary
subindex	Subindex of the object to be written in the object dictionary
value	Value to be written

NOTICE

It is highly recommended that the processor time be passed on with `yield()` after calling a `od_write()`. The value is immediately written to the OD. For the firmware to be able to trigger actions that are dependent on this, however, it must receive computing time. This, in turn, means that the user program must either be ended or interrupted with `yield()`.

U32 **od_read** (U32 index, U32 subindex)

This function reads the value at the specified location in the object dictionary and returns it.

index	Index of the object to be read in the object dictionary
subindex	Subindex of the object to be read in the object dictionary
Output value	Content of the OD entry

NOTICE

Active waiting for a value in the object dictionary should always be associated with a `yield()`.

Example

```
while (od_read(2400,2) != 0) // wait until 2400:2 is set
{ yield(); }
```

10.3.2 Process control

```
void yield()
```

This function returns the processor time to the operating system. In the next time slot, the program continues at the location after the call.

```
void sleep (U32 ms)
```

This function returns the processor time to the operating system for the specified number of milliseconds. The user program is then continued at the location after the call.

ms	Time to be waited in milliseconds

10.3.3 Debug output

The following functions output a value in the debug console. They differ with respect to the data type of the parameter to be passed.

```
bool VmmDebugOutputString (const char *outstring)
```

```
bool VmmDebugOutputInt (const U32 val)
```

```
bool VmmDebugOutputByte (const U08 val)
```

```
bool VmmDebugOutputHalfWord (const U16 val)
```

```
bool VmmDebugOutputWord (const U32 val)
```

```
bool VmmDebugOutputFloat (const float val)
```

NOTICE

The debug outputs are first written to a separate area of the object dictionary and read from there by the *Plug & Drive Studio*.

This OD entry has index 2600_h and is 64 characters long, see 2600h NanoJ Debug Output. Subindex 00 always contains the number of characters already written.

If the buffer is full, `VmmDebugOutputxxx()` initially fails; execution of the user program ceases and it stops at the location of the debug output. Only after the GUI has read the buffer and after subindex 00 has been reset does the program continue and `VmmDebugOutputxxx()` returns to the user program.

NOTICE

Debug outputs may therefore only be used during the test phase when developing a user program.

NOTICE

Do not use the debug output if *AutoYield* mode is activated (see Available computing time).

10.4 Restrictions and possible problems

Restrictions and possible problems when working with NanoJ are listed below:

Restriction/problem	Measure
If an object is mapped, e. g., 0x6040, the object is reset to its previous value every 1 ms. This makes it impossible to control this object via the fieldbus or the *Plug & Drive Studio*.	Instead use `od_read` / `od_write` to access the object.
If an object was mapped as output and the value of the object was never defined before starting the *NanoJ program*, the value of this object may be random.	Initialize the values of the mapped objects in your NanoJ program to ensure that it behaves deterministically.

Restriction/problem	Measure
The array initialization must not be used with more than 16 entries.	Use `constant array` instead.
Too many local variables and arrays within functions may result in a stack overflow.	Declare the variables globally. Memory requirements are monitored already during compilation; errors do not occur at runtime.
Functions that are too deeply nested may result in a stack overflow.	Observe a maximum nesting depth of 2.
`float` must not be used with comparison operators.	Use `int` instead.
`double` must not be used.	
If a NanoJ program restarts the controller (either directly with an explicit restart or indirectly, e. g., through the use of the Reset function), the controller may fall into a restart loop that can be exited only with difficulty if at all.	
`math` or `cmath` cannot be included.	

11 Description of the object dictionary

11.1 Overview

This chapter contains a description of all objects.

You will find information here on:

- Functions
- Object descriptions ("Index")
- Value descriptions ("Subindices")
- Descriptions of bits
- Description of the object

11.2 Structure of the object description

The description of the object entries always has the same structure and usually consists of the following sections:

Function

The function of the object dictionary is briefly described in this section.

Object description

This table provides detailed information on the data type, preset values and similar. An exact description can be found in section "Object description"

Value description

This table is only available with the "Array" or "Record" data type and provides exact information about the sub-entries. A more exact description of the entries can be found in section "Value description"

Description

Here, more exact information on the individual bits of an entry is provided or any compositions explained. A more exact description can be found in section "Description"

11.3 Object description

The object description consists of a table that contains the following entries:

Index

Designates the object index in hexadecimal notation.

Object name

The name of the object.

Object Code

The type of object. This can be one of the following entries:

- VARIABLE: In this case, the object consists of only a variable that is indexed with subindex 0.
- ARRAY: These objects always consists of a subindex 0 – which specifies the number of sub-entries – and the sub-entries themselves, beginning with index 1. The data type within an array never changes, i.e., sub-entry 1 and all subsequent entries are always of the same data type.
- RECORD: These objects always consists of a subindex 0 – which specifies the number of sub-entries – and the sub-entries themselves, beginning with index 1. Unlike an ARRAY, the data type of the sub-entries can vary. This means that, e.g., sub-entry 1 may be of a different data type than sub-entry 2.

- VISIBLE_STRING: The object describes a character string coded in ASCII. The length of the string is specified in subindex 0; the individual characters are stored beginning in subindex 1. These character strings are **not** terminated by a null character.

Data type

The size and interpretation of the object is specified here. The following notation is used for the "VARIABLE" object code:

- A distinction is made between entries that are signed; these are designated with the prefix "SIGNED". For entries that are unsigned, the prefix "UNSIGNED" is used.
- The size of the variable in bits is placed before the prefix and can be 8, 16 or 32.

Savable

Described here is whether this object is savable and, if so, in which category.

Firmware version

The firmware version beginning with which the object is available is entered here.

Change history (ChangeLog)

Any changes to the object are noted here.

There are also the following table entries for the "VARIABLE" data type:

Access

The access restriction is entered here. The following restrictions are available:

- "read/write": The object can both be read as well as written
- "read only": The object can only be read from the object dictionary. It is not possible to set a value.

PDO mapping

Some bus systems, such as CANopen or EtherCAT, support PDO mapping. Described in this table entry is whether the object can be inserted into a mapping and, if so, into which. The following designations are available here:

- "no": The object may not be entered in a mapping.
- "TX-PDO": The object may be entered in an RX mapping.
- "RX-PDO": The object may be entered in a TX mapping.

Allowed values

In some cases, only certain values may be written in the object. If this is the case, these values are listed here. If there are no restrictions, the field is empty.

Preset value

To bring the controller to a secured state when switching on, it is necessary to preset a number of objects with values. The value that is written in the object when the controller is started is noted in this table entry.

11.4 Value description

NOTICE

For the sake of clarity, a number of subindices are grouped together if the entries all have the same name.

Listed in the table with the "Value description" heading are all data for sub-entries with subindex 1 or higher. The table contains the following entries:

Subindex

Number of the currently written sub-entry.

Name

Name of the sub-entry.

Data type

The size and interpretation of the sub-entry is specified here. The following notation always applies here:

- A distinction is made between entries that are signed; these are designated with the prefix "SIGNED". For entries that are unsigned, the prefix "UNSIGNED" is used.
- The size of the variable in bits is placed before the prefix and can be 8, 16 or 32.

Access

The access restriction for the sub-entry is entered here. The following restrictions are available:

- "read/write": The object can both be read as well as written
- "read only": The object can only be read from the object dictionary. It is not possible to set a value.

PDO mapping

Some bus systems, such as CANopen or EtherCAT, support PDO mapping. Described in this table entry is whether the sub-entry can be inserted into a mapping and, if so, into which. The following designations are available here:

- "no": The object may not be entered in a mapping.
- "TX-PDO": The object may be entered in an RX mapping.
- "RX-PDO": The object may be entered in a TX mapping.

Allowed values

In some cases, only certain values may be written in the sub-entry. If this is the case, these values are listed here. If there are no restrictions, the field is empty.

Preset value

To bring the controller to a secured state when switching on, it is necessary to preset a number of sub-entries with values. The value that is written in the sub-entry when the controller is started is noted in this table entry.

11.5 Description

This section may be present if use requires additional information. If individual bits of an object or sub-entry have different meaning, diagrams as shown in the following example are used.

Example: The object is 8 bits in size; bit 0 and bit 1 have different functions. Bits 2 and 3 are grouped into one function; the same applies for bits 4 to 7.

7	6	5	4	3	2	1	0
Example [4]				Example [2]		B	A

Example [4]

Description of bit 4 up to and including bit 7; these bits are logically related. The 4 in square brackets specifies the number of related bits. A list with possible values and their description is often attached at this point.

Example [2]

Description of bits 3 and 2; these bits are logically related. The 2 in square brackets specifies the number of related bits.

- Value 00_b: The description here applies if bit 2 and bit 3 are "0".
- Value 01_b: The description here applies if bit 2 is "0" and bit 3 is "1".
- Value 10_b: The description here applies if bit 2 is "1" and bit 3 is "0".
- Value 11_b: The description here applies if bit 2 and bit 3 are "1".

B

Description of bit B; no length is specified for a single bit.

A

Description of bit A; bits with a gray background are not used.

1000h Device Type

Function

Describes the controller type.

Object description

Index	1000_h
Object name	Device Type
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	00060192_h
Firmware version	FIR-v1426
Change history	

Description

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Motor Type [16]															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Device profile number [16]															

Motor Type[16]

Describes the supported motor type. The following values are possible:

- Bit 23 to bit 16: Value "2": BLDC motor
- Bit 23 to bit 16: Value "4": Stepper motor
- Bit 23 to bit 16: Value "6": Stepper motor as well as BLDC motor

Device profile number[16]

Describes the supported CANopen standard.

Values:

0192_h or 0402_d (preset value): The CiA 402 standard is supported.

1001h Error Register

Function

Error register: The corresponding error bit is set in case of an error. If the error no longer exists, it is deleted automatically.

NOTICE

For each error that occurs, a more precise error code is stored in object 1003_h.

Object description

Index	1001_h
Object name	Error Register
Object Code	VARIABLE
Data type	UNSIGNED8
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00_h
Firmware version	FIR-v1426
Change history	

Description

7	6	5	4	3	2	1	0
MAN	RES	PROF	COM	TEMP	VOL	CUR	GEN

GEN

General error

CUR

Current

VOL

Voltage

TEMP

Temperature

COM

Communication

PROF

Relates to the device profile

RES

Reserved, always "0"

MAN

Manufacturer-specific

1003h Pre-defined Error Field

Function

This object contains an error stack with up to eight entries.

Object description

Index	1003_h
Object name	Pre-defined Error Field
Object Code	ARRAY
Data type	UNSIGNED32
Savable	no
Firmware version	FIR-v1426
Change history	

Value description

Subindex	00_h
Name	Number Of Errors
Data type	UNSIGNED8
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00_h

Subindex	01_h
Name	1st Standard Error Field
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	02_h
Name	2nd Standard Error Field
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	

Preset value	00000000_h

Subindex	03_h
Name	3th Standard Error Field
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	04_h
Name	4th Standard Error Field
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	05_h
Name	5th Standard Error Field
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	06_h
Name	6th Standard Error Field
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	07_h
Name	7th Standard Error Field
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	08_h

Name	8th Standard Error Field
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h

Description

General function

If a new error occurs, it is entered in subindex 1. The already existing entries in subindices 1 to 7 are moved back one position. The error in subindex 7 is thereby removed.

The number of errors that have already occurred can be read from the object with subindex 0. If no error is currently entered in the error stack, it is not possible to read one of the eight subindices 1–8 and an error (abort code = 08000024_h) is sent in response. If a "0" is written in subindex 0, counting starts again from the beginning.

Bit description

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Error Number [8]								Error Class [8]							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Error Code [16]															

Error Number [8]

This can be used to pinpoint the cause of the error. The meaning of the number can be found in the following table.

Error number	Description
0	Watchdog-Reset
1	Input voltage (+Ub) too high
2	Output current too high
3	Input voltage (+Ub) too low
4	Error at fieldbus
6	CANopen only: NMT master takes too long to send Nodeguarding request
7	Sensor 1 (see 3204_h): Error through electrical fault or defective hardware
8	Sensor 2 (see 3204_h): Error through electrical fault or defective hardware
9	Sensor 3 (see 3204_h): Error through electrical fault or defective hardware
10	Warning: Positive limit switch exceeded
11	Warning: Negative limit switch exceeded
12	Overtemperature error
13	The values of object 6065_h (Following Error Window) and object 6066_h (Following Error Time Out) were exceeded; a fault was triggered.
14	Warning: Nonvolatile memory full. The current save process could not be completed; parts of the data of the save process are lost. Controller must be restarted for cleanup work.
15	Motor blocked
16	Warning: Nonvolatile memory damaged; controller must be restarted for cleanup work (all saved objects are reset to default).
17	CANopen only: Slave took too long to send PDO messages.

Error number	Description
18	Sensor n (see 3204h), where n is greater than 3: Error through electrical fault or defective hardware
19	CANopen only: PDO not processed due to a length error
20	CANopen only: PDO length exceeded
21	Warning: Restart the controller to avoid future errors when saving (nonvolatile memory full/corrupt).
22	Rated current must be set (203Bh:01h/6075h)
23	Encoder resolution, number of pole pairs and some other values are incorrect.
24	Motor current is too high, adjust the PI parameters.
25	Internal software error, generic
26	Current too high at digital output
27	CANopen only: Unexpected sync length
30	Error in speed monitoring: slippage error too large
32	Internal error: Correction factor for reference voltage missing in the OTP
39	Error in the ballast configuration: Invalid/unrealistic parameters entered (see Ballast monitoring)
40	Warning: Ballast resistor thermally overloaded
46	Interlock error: Bit 3 in 60FDh is set to "0", the motor may not start (see the section *Interlock function* in the chapter Digital inputs)

Error Class[8]

This byte is identical to object 1001h

Error Code[16]

Refer to the following table for the meaning of the bytes.

Error Code	Description
1000h	General error
2300h	Current at the controller output too large
3100h	Overvoltage/undervoltage at controller input
4200h	Temperature error within the controller
5440h	Interlock error: Bit 3 in 60FDh is set to "0", the motor may not start (see the section *Interlock function* in the chapter Digital inputs)
6010h	Software reset (watchdog)
6100h	Internal software error, generic
6320h	Rated current must be set (203Bh:01h/6075h)
7110h	Error in the ballast configuration: Invalid/unrealistic parameters entered (see Ballast monitoring)
7113h	Warning: Ballast resistor thermally overloaded
7121h	Motor blocked
7200h	Internal error: Correction factor for reference voltage missing in the OTP
7305h	Sensor 1 (see 3204h) faulty
7306h	Sensor 2 (see 3204h) faulty
7307h	Sensor n (see 3204h), where n is greater than 2

Error Code	Description
7600_h	Warning: Nonvolatile memory full or corrupt; restart the controller for cleanup work
8100_h	Error during fieldbus monitoring
8130_h	CANopen only: "Life Guard" error or "Heartbeat" error
8200_h	CANopen only: Slave took too long to send PDO messages.
8210_h	CANopen only: PDO was not processed due to a length error
8220_h	CANopen only: PDO length exceeded
8240_h	CANopen only: unexpected sync length
8400_h	Error in speed monitoring: slippage error too large
8611_h	Position monitoring error: Following error too large
8612_h	Position monitoring error: Limit switch exceeded

1008h Manufacturer Device Name

Function

Contains the device name as character string.

Object description

Index	1008_h
Object name	Manufacturer Device Name
Object Code	VARIABLE
Data type	VISIBLE_STRING
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	■ CPB6-1-4: CPB6-1-4 ■ CPB6-1-S-4: CPB6-1-S-4 ■ CPB6-2-4: CPB6-2-4 ■ CPB6-2-S-4: CPB6-2-S-4 ■ CPB3-1-4: CPB3-1-4 ■ CPB3-2-4: CPB3-2-4 ■ CPB15-4: CPB15-4 ■ CPB15-S-4: CPB15-S-4
Firmware version	FIR-v1426
Change history	

1009h Manufacturer Hardware Version

Function

This object contains the hardware version as character string.

Object description

Index	1009_h
Object name	Manufacturer Hardware Version
Object Code	VARIABLE
Data type	VISIBLE_STRING
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	0
Firmware version	FIR-v1426
Change history	

100Ah Manufacturer Software Version

Function

This object contains the software version as character string.

Object description

Index	$100A_h$
Object name	Manufacturer Software Version
Object Code	VARIABLE
Data type	VISIBLE_STRING
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	FIR-v2213-B1030801
Firmware version	FIR-v1426
Change history	

1010h Store Parameters

Function

This object is used to start the saving of objects. See chapter Saving objects.

Object description

Index	1010_h
Object name	Store Parameters
Object Code	ARRAY
Data type	UNSIGNED32
Savable	no
Access	read only

PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1436: "Object name" entry changed from "Store Parameter" to "Store Parameters". Firmware version FIR-v1436: The number of entries was changed from 3 to 4. Firmware version FIR-v1512: The number of entries was changed from 4 to 5. Firmware version FIR-v1540: The number of entries was changed from 5 to 7. Firmware version FIR-v1738-B501312: The number of entries was changed from 7 to 14.

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	$0D_h$

Subindex	01_h
Name	Save All Parameters To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001_h

Subindex	02_h
Name	Save Communication Parameters To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001_h

Subindex	03_h
Name	Save Application Parameters To Non-volatile Memory
Data type	UNSIGNED32

Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001_h

Subindex	04_h
Name	Save Customer Parameters To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001_h

Subindex	05_h
Name	Save Drive Parameters To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001_h

Subindex	06_h
Name	Save Tuning Parameters To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001_h

Subindex	07_h
Name	Save Miscellaneous Configurations To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001_h

Subindex	08_h
Name	Save Reserved1 Configurations To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	

Preset value	00000000$_h$

Subindex	09$_h$
Name	Save Reserved2 Configurations To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000$_h$

Subindex	0A$_h$
Name	Save CANopen Configurations To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001$_h$

Subindex	0B$_h$
Name	Save Modbus RTU Configurations To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001$_h$

Subindex	0C$_h$
Name	Save Ethernet Configurations To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001$_h$

Subindex	0D$_h$
Name	Save Profibus Configurations To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001$_h$

Description

Each subindex of the object stands for a certain memory class. By reading out the entry, it is possible to determine whether (value "1") or not (value="0") this memory category can be saved.

To start the save process of a memory category, value "65766173_h" must be written in the corresponding subindex. This corresponds to the decimal of 1702257011_d or the ASCII string `save`. As soon as the saving process is completed, the save command is again overwritten with the value "1", since saving is possible again.

For a detailed description, see chapter Saving objects.

1011h Restore Default Parameters

Function

This object can be used to reset all or part of the object dictionary to the default values. See chapter Saving objects.

Object description

Index	1011_h
Object name	Restore Default Parameters
Object Code	ARRAY
Data type	UNSIGNED32
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1436: "Object Name" entry changed from "Restore Default Parameter" to "Restore Default Parameters". Firmware version FIR-v1436: The number of entries was changed from 2 to 4. Firmware version FIR-v1512: The number of entries was changed from 4 to 5. Firmware version FIR-v1512: "Name" entry changed from "Restore The Comm Default Parameters" to "Restore Communication Default Parameters". Firmware version FIR-v1512: "Name" entry changed from "Restore The Application Default Parameters" to "Restore Application Default Parameters". Firmware version FIR-v1540: The number of entries was changed from 5 to 7. Firmware version FIR-v1738-B501312: The number of entries was changed from 7 to 14.

Value description

Subindex	00_h

Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	$0D_h$

Subindex	01_h
Name	Restore All Default Parameters
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001_h

Subindex	02_h
Name	Restore Communication Default Parameters
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001_h

Subindex	03_h
Name	Restore Application Default Parameters
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001_h

Subindex	04_h
Name	Restore Customer Default Parameters
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001_h

Subindex	05_h
Name	Restore Drive Default Parameters
Data type	UNSIGNED32
Access	read / write

PDO mapping	no
Allowed values	
Preset value	00000001_h

Subindex	06_h
Name	Restore Tuning Default Parameters
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001_h

Subindex	07_h
Name	Restore Miscellaneous Configurations
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001_h

Subindex	08_h
Name	Restore Reserved1 Configurations To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	09_h
Name	Restore Reserved2 Configurations To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	$0A_h$
Name	Restore CANopen Configurations To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001_h

Subindex	$0B_h$
Name	Restore Modbus RTU Configurations To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001_h

Subindex	$0C_h$
Name	Restore Ethernet Configurations To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001_h

Subindex	$0D_h$
Name	Restore Profibus Configurations To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001_h

Description

If the value $64616F6C_h$ (or 1684107116_d or ASCII `load`) is written in this object, part or all of the object dictionary is reset to the default values. The subindex that is used decides which range is reset.

For a detailed description, see chapter Discarding the saved data.

1018h Identity Object

Function

This object returns general information on the device, such as manufacturer, product code, revision and serial number.

TIP

Have these values ready in the event of service inquiries.

Object description

Index	1018_h
Object name	Identity Object
Object Code	RECORD
Data type	IDENTITY

Savable	no
Firmware version	FIR-v1426
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	04_h

Subindex	01_h
Name	Vendor-ID
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	$0000026C_h$

Subindex	02_h
Name	Product Code
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	■ CPB6-1-4: 00000193_h ■ CPB6-1-S-4: $0000019D_h$ ■ CPB6-2-4: $000001B1_h$ ■ CPB6-2-S-4: $000001BB_h$ ■ CPB3-1-4: $000001C5_h$ ■ CPB3-2-4: $000001D9_h$ ■ CPB15-4: $0000020D_h$ ■ CPB15-S-4: 00000217_h

Subindex	03_h
Name	Revision Number
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	$08A50000_h$

Subindex	04_h
Name	Serial Number
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h

1020h Verify Configuration

Function

This object indicates the date and time that the configuration was stored.

A configuration tool or a master can use this object to verify the configuration after a reset and, if necessary, perform a new configuration.

The tool must set the date and time before the storage mechanism is started (see chapter Saving objects).

Object description

Index	1020_h
Object name	Verify Configuration
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: verify
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1540
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Configuration Date
Data type	UNSIGNED32
Access	read / write
PDO mapping	no

Allowed values	
Preset value	00000000_h

Subindex	02_h
Name	Configuration Time
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Description

Subindex 01_h (configuration date) is to contain the number of days since 1 January 1984.

Subindex 02_h (configuration time) is to contain the number of milliseconds since midnight.

1F50h Program Data

Function

This object is used to program memory areas of the controller. Each entry stands for a certain memory area.

Object description

Index	$1F50_h$
Object name	Program Data
Object Code	ARRAY
Data type	DOMAIN
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1540
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Program Data Bootloader/firmware
Data type	DOMAIN
Access	read / write
PDO mapping	no
Allowed values	
Preset value	0

Subindex	02_h
Name	Program Data NanoJ
Data type	DOMAIN
Access	read / write
PDO mapping	no
Allowed values	
Preset value	0

1F51h Program Control

Function

This object is used to control the programming of memory areas of the controller. Each entry stands for a certain memory area.

Object description

Index	$1F51_h$
Object name	Program Control
Object Code	ARRAY
Data type	UNSIGNED8
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1540
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Program Control Bootloader/firmware
Data type	UNSIGNED8
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00_h

Subindex	02_h
Name	Program Control NanoJ
Data type	UNSIGNED8
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00_h

1F57h Program Status

Function

This object indicates the programming status during the programming of memory areas of the controller. Each entry stands for a certain memory area.

Object description

Index	$1F57_h$
Object name	Program Status
Object Code	ARRAY
Data type	UNSIGNED32
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1540
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Program Status Bootloader/firmware
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	02_h
Name	Program Status NanoJ
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h

200Fh IEEE 802 MAC Address

Function

This object contains the MAC address of the controller as a character string.

Object description

Index	$200F_h$
Object name	IEEE 802 MAC Address
Object Code	VARIABLE
Data type	VISIBLE_STRING
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	0
Firmware version	FIR-v1748-B533384
Change history	

2010h IP-Configuration

Function

Use this object to configure the Ethernet interface. The object is only taken into consideration once when restarting the controller. You can find further details in chapter *Commissioning*.

Object description

Index	2010_h
Object name	IP-Configuration
Object Code	VARIABLE

Data type	UNSIGNED32
Savable	yes, category: Ethernet
Access	read / write
PDO mapping	no
Allowed values	
Preset value	$0000006C_h$
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1748-B533384: "Savable" entry changed from "yes, category: communication" to "yes, category: Ethernet".

Description

IP

Value = "1": A static IP address from object 2011_h is used and the network mask from object 2012_h is used.

DHCP

Value = "1": IP address assignment using a DHCP server is activated

AUTO

Value = "1": IP address assignment using the AUTO-IP protocol is activated

NBIOS

Value = "1": The NetBIOS protocol is activated; this is necessary before resolving a hostname (e.g., with a ping command).

LLMNR

Value = "1": The LLMNR protocol is activated; this is necessary before resolving a hostname (e.g., with a ping command).

NOTICE

To avoid conflicts, never simultaneously set bit 0 (static IP address) and bits 2 and 3 (DHCP or Auto-IP) to "1".

2011h Static-IPv4-Address

Function

Contains the static IPv4 address in the form of a 32-bit word.

Object description

Index	2011_h
Object name	Static-IPv4-Address
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: Ethernet
Access	read / write
PDO mapping	no

Allowed values	
Preset value	$C0A80792_h$
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1450: "Object Name" entry changed from "Static-IP-Address" to "Static-IPv4-Address". Firmware version FIR-v1748-B533384: "Savable" entry changed from "yes, category: communication" to "yes, category: Ethernet".

Description

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IP Address Part 1 [8]								IP Address Part 2 [8]							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IP Address Part 3 [8]								IP Address Part 4 [8]							

IP Address Part 1 [8]
Specifies the first part of the IP address

IP Address Part 2 [8]
Specifies the second part of the IP address

IP Address Part 3 [8]
Specifies the third part of the IP address

IP Address Part 4 [8]
Specifies the fourth part of the IP address

Example

Address `192.168.2.0` is first converted to hexadecimal format, resulting in the following configuration value:

192 => $C0_h$

168 => $A8_h$

2 => 02_h

0 => 0

The corresponding adjustment value is then `C0A80200`$_h$.

2012h Static-IPv4-Subnet-Mask

Function

Contains the subnet mask of the static IP address in the form of a 32-bit word.

Object description

Index	2012_h
Object name	Static-IPv4-Subnet-Mask
Object Code	VARIABLE

Data type	UNSIGNED32
Savable	yes, category: Ethernet
Access	read / write
PDO mapping	no
Allowed values	
Preset value	FFFFFF00$_h$
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1450: "Object Name" entry changed from "Static-IP-Subnet-Mask" to "Static-IPv4-Subnet-Mask". Firmware version FIR-v1748-B533384: "Savable" entry changed from "yes, category: communication" to "yes, category: Ethernet".

Description

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Subnet Mask Part 1 [8]								Subnet Mask Part 2 [8]							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Subnet Mask Part 3 [8]								Subnet Mask Part 4 [8]							

Subnet Mask Part 1 [8]
Specifies the first part of the subnet mask

Subnet Mask Part 2 [8]
Specifies the second part of the subnet mask

Subnet Mask Part 3 [8]
Specifies the third part of the subnet mask

Subnet Mask Part 4 [8]
Specifies the fourth part of the subnet mask

Example

The class C network mask `255.255.255.0` is first converted to hexadecimal format, resulting in the following configuration value:

255 => FF$_h$

0 => 0

The corresponding adjustment value is then `FFFFFF00`$_h$.

2013h Static-IPv4-Gateway-Address

Function

Contains the static IP gateway address in the form of a 32-bit word.

Object description

Index	2013$_h$
Object name	Static-IPv4-Gateway-Address

Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: Ethernet
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h
Firmware version	FIR-v1446
Change history	Firmware version FIR-v1512: "Object Name" entry changed from "Static-IP-Gateway-Address" to "Static-IPv4-Gateway-Address". Firmware version FIR-v1748-B533384: "Savable" entry changed from "yes, category: communication" to "yes, category: Ethernet".

Description

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IP-Gateway-Address Part 1 [8]								IP-Gateway-Address Part Part 2 [8]							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IP-Gateway-Address Part 3 [8]								IP-Gateway-Address Part Part 4 [8]							

IP-Gateway-Address Part 1 [8]

Specifies the first part of the IP gateway address

IP-Gateway-Address Part 2 [8]

Specifies the second part of the IP gateway address

IP-Gateway-Address Part 3 [8]

Specifies the third part of the IP gateway address

IP-Gateway-Address Part 4 [8]

Specifies the fourth part of the IP gateway address

Example

Address `192.168.2.0` is first converted to hexadecimal format, resulting in the following configuration value:

192 => $C0_h$

168 => $A8_h$

2 => 02_h

0 => 0

The corresponding adjustment value is then $\mathtt{C0A80200}_h$.

2014h Current-IPv4-Address

Function

Contains the currently active IP address in the form of a 32-bit word.

Object description

Index	2014_h
Object name	Current-IPv4-Address
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1450: "Object Name" entry changed from "Current-IP-Address" to "Current-IPv4-Address".

Description

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IP Address Part 1 [8]								IP Address Part 2 [8]							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IP Address Part 3 [8]								IP Address Part 4 [8]							

IP Address Part 1 [8]
Specifies the first part of the IP address

IP Address Part 2 [8]
Specifies the second part of the IP address

IP Address Part 3 [8]
Specifies the third part of the IP address

IP Address Part 4 [8]
Specifies the fourth part of the IP address

Example

Address `192.168.2.0` is first converted to hexadecimal format, resulting in the following configuration value:

192 => $C0_h$

168 => $A8_h$

2 => 02_h

0 => 0

The corresponding adjustment value is then `C0A80200`$_h$.

2015h Current-IPv4-Subnet-Mask

Function

Contains the currently active subnet mask of the static IP address in the form of a 32-bit word.

Object description

Index	2015_h
Object name	Current-IPv4-Subnet-Mask
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1450: "Object Name" entry changed from "Current-IP-Subnet-Mask" to "Current-IPv4-Subnet-Mask".

Description

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Subnet Mask Part 1 [8]								Subnet Mask Part 2 [8]							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Subnet Mask Part 3 [8]								Subnet Mask Part 4 [8]							

Subnet Mask Part 1 [8]
Specifies the first part of the subnet mask

Subnet Mask Part 2 [8]
Specifies the second part of the subnet mask

Subnet Mask Part 3 [8]
Specifies the third part of the subnet mask

Subnet Mask Part 4 [8]
Specifies the fourth part of the subnet mask

Example

The class C network mask `255.255.255.0` is first converted to hexadecimal format, resulting in the following configuration value:

255 => FF_h

0 => 0

The corresponding adjustment value is then `FFFFFF00`$_h$.

2016h Current-IPv4-Gateway-Address

Function

This object contains the currently active gateway IP address in the form of a 32-bit word.

Object description

Index	2016_h
Object name	Current-IPv4-Gateway-Address
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h
Firmware version	FIR-v1540
Change history	

2030h Pole Pair Count

Function

Contains the number of pole pairs of the connected motor.

Object description

Index	2030_h
Object name	Pole Pair Count
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: tuning
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000032_h
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1540: "Savable" entry changed from "no" to "yes, category: tuning".

2031h Max Motor Current

Function

Enter the maximum permissible motor current in milliamperes here. All current values are limited by this value.

Within the controller, the entered value is always interpreted as the root mean square.

Object description

Index	2031$_h$
Object name	Max Motor Current
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: tuning
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000258$_h$
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1614: "Savable" entry changed from "yes, category: application" to "yes, category: tuning". Firmware version FIR-v1614: "Object Name" entry changed from "Peak Current" to "Max Current". Firmware version FIR-v1748-B538662: "Object Name" entry changed from "Maximum Current" to "Max Motor Current". Firmware version FIR-v1825-B577172: "Object Name" entry changed from "Max Motor Current" to "Maximum Current". Firmware version FIR-v1825-B577172: "Object Name" entry changed from "Maximum Current" to "Max Motor Current". Firmware version FIR-v1825-B577172: "Object Name" entry changed from "Max Motor Current" to "Maximum Current". Firmware version FIR-v1825-B577172: "Object Name" entry changed from "Maximum Current" to "Max Motor Current".

2034h Upper Voltage Warning Level

Function

This object contains the threshold value for the "overvoltage" error in millivolts.

Object description

Index	2034$_h$
Object name	Upper Voltage Warning Level
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: application
Access	read / write
PDO mapping	no
Allowed values	
Preset value	■ CPB6-1-4: 0000EE48$_h$ ■ CPB6-1-S-4: 0000EE48$_h$ ■ CPB6-2-4: 00011365$_h$ ■ CPB6-2-S-4: 00011365$_h$

- CPB3-1-4: 0000EE48$_h$
- CPB3-2-4: 0000EE48$_h$
- CPB15-4: 0000EE48$_h$
- CPB15-S-4: 0000EE48$_h$

Firmware version	FIR-v1426
Change history	

Description

If the input voltage of the controller exceeds this threshold value, the motor is switched off and an error triggered. This error is reset automatically if the input voltage is less than (voltage of object 2034$_h$ minus 2 volts).

2035h Lower Voltage Warning Level

Function

This object contains the threshold value for the "Undervoltage" error in millivolts.

Object description

Index	2035$_h$
Object name	Lower Voltage Warning Level
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: application
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00002D78$_h$
Firmware version	FIR-v1426
Change history	

Description

If the input voltage of the controller falls below this threshold value, the motor is switched off and an error triggered. The error is reset automatically if the input voltage exceeds the voltage of object 2035$_h$ plus 1.5 volts.

2036h Open Loop Current Reduction Idle Time

Function

This object describes the time in milliseconds that the motor must be at a standstill before current reduction is activated.

Object description

Index	2036$_h$
Object name	Open Loop Current Reduction Idle Time
Object Code	VARIABLE

Data type	UNSIGNED32
Savable	yes, category: application
Access	read / write
PDO mapping	no
Allowed values	
Preset value	$000003E8_h$
Firmware version	FIR-v1426
Change history	

2037h Open Loop Current Reduction Value/factor

Function

This object describes the rms current to which the motor current is to be reduced if current reduction is activated in open loop (bit 3 in 3202_h = "1") and the motor is at a standstill.

Object description

Index	2037_h
Object name	Open Loop Current Reduction Value/factor
Object Code	VARIABLE
Data type	INTEGER32
Savable	yes, category: application
Access	read / write
PDO mapping	no
Allowed values	
Preset value	$FFFFFFCE_h$
Firmware version	FIR-v1426
Change history	

Description

Value of 2037_h greater than or equal to 0 and less than value 6075_h

Current is reduced to the value entered here. The value is in mA and interpreted as root mean square.

Value of 2037_h in the range from -1 to -100

The entered value is interpreted as a percentage and determines the reduction of the rated current in 2037_h. The value in 6075_h is used for the calculation.

Example: Object 6075_h has the value 4200 mA. The value -60 in 2037_h reduces the current by 60% of 6075_h. The result is a current reduction to a root mean square of 6075_h * (2037_h + 100) / 100 = 1680 mA.

The value -100 in 2037_h would, for example, mean that a current reduction is set to a root mean square of 0 mA.

2038h Brake Controller Timing

Function

This object contains the times for the *brake control* in milliseconds as well as the PWM frequency and the duty cycle.

Object description

Index	2038_h
Object name	Brake Controller Timing
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Firmware version	FIR-v1426
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	06_h

Subindex	01_h
Name	Close Brake Idle Time
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	$000003E8_h$

Subindex	02_h
Name	Shutdown Power Idle Time
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	$000003E8_h$

Subindex	03_h
Name	Open Brake Delay Time
Data type	UNSIGNED32

Access	read / write
PDO mapping	no
Allowed values	
Preset value	000003E8$_h$

Subindex	04$_h$
Name	Start Operation Delay Time
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000$_h$

Subindex	05$_h$
Name	PWM Frequency
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	between 0 and 50 (brake output) and 20000 (4E20$_h$)
Preset value	00000000$_h$

Subindex	06$_h$
Name	PWM Duty Cycle
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	0, between 2 and 100 (64$_h$)
Preset value	00000000$_h$

Description

The subindices have the following functions:

- 01$_h$: Time between motor standstill and the closing of the brake.
- 02$_h$: Time between the closing of the brake and the switching off of the motor current.
- 03$_h$: Time between the switching on of the motor current and opening of the brake.
- 04$_h$: Time between the opening of the brake and when the *Operation enabled* state of the CiA 402 Power State Machine is reached.
- 05$_h$: Frequency of the PWM signal in hertz.
- 06$_h$: Duty cycle of the PWM signal in percent.

2039h Motor Currents

Function

This object contains the measured motor currents in mA. All values are peak values, (#2*rms).

Object description

Index	2039_h
Object name	Motor Currents
Object Code	ARRAY
Data type	INTEGER32
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1504: "PDO mapping" table entry for subindex 01 changed from "no" to "TX-PDO". Firmware version FIR-v1504: "PDO mapping" table entry for subindex 02 changed from "no" to "TX-PDO". Firmware version FIR-v1504: "PDO mapping" table entry for subindex 03 changed from "no" to "TX-PDO". Firmware version FIR-v1504: "PDO mapping" table entry for subindex 04 changed from "no" to "TX-PDO". Firmware version FIR-v2213: subindex 05_h, "Actual Current" added. Phase currents Ia and Ib changed to Iα and Iβ (Clarke transformation).

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	05_h

Subindex	01_h
Name	Id
Data type	INTEGER32
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h

Subindex	02_h
Name	Iq
Data type	INTEGER32
Access	read only

PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h

Subindex	03_h
Name	Iα
Data type	INTEGER32
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h

Subindex	04_h
Name	Iβ
Data type	INTEGER32
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h

Subindex	05_h
Name	Actual Current
Data type	INTEGER32
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h

Description

- 01_h: Field-forming components of the current
- 02_h: Torque-forming components of the current
- 03_h: Iα
- 04_h: Iβ
- 05_h: total current divided by √2, i.e., calculated down to a motor phase. In *closed-loop*, the sign of Iq is also used. The current value can then be placed on a scale to compare with the current from 6075_h, 2031_h and $203B_h$:05_h.
  open-loop: $I = \sqrt{(I\alpha^2 + I\beta^2)} / \sqrt{2}$
  Closed Loop:$I = \mathrm{sgn}(Iq) * \sqrt{(I\alpha^2 + I\beta^2)} / \sqrt{2}$

NOTICE

Motor currents I_d (subindex 01_h) and I_q (subindex 02_h) are only displayed if closed-loop was activated; the value 0 is otherwise output.

203Ah Homing On Block Configuration

Function

This object contains the parameters for *Homing on Block* (see chapter Homing).

Object description

Index	$203A_h$
Object name	Homing On Block Configuration
Object Code	ARRAY
Data type	INTEGER32
Savable	yes, category: application
Access	
PDO mapping	
Allowed values	
Preset value	
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1540: The number of entries was changed from 4 to 3. Firmware version FIR-v1540: "Name" entry changed from "Period Of Blocking" to "Block Detection time". Firmware version FIR-v1614: "Data Type" entry changed from "UNSIGNED32" to "INTEGER32". Firmware version FIR-v1614: "Savable" entry changed from "no" to "yes, category: application". Firmware version FIR-v1614: "Data Type" entry changed from "UNSIGNED32" to "INTEGER32". Firmware version FIR-v1614: "Data Type" entry changed from "UNSIGNED32" to "INTEGER32".

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Minimum Current For Block Detection
Data type	INTEGER32
Access	read / write
PDO mapping	no
Allowed values	

Preset value	$000005DC_h$

Subindex	02_h
Name	Block Detection Time
Data type	INTEGER32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	$000000C8_h$

Description

The subindices have the following function:

- 01_h: Specifies the current limit value above which blocking is to be detected. Positive numerical values specify the current limit in mA, negative numbers specify a percentage of object 2031_h. Example: The value "1000" corresponds to 1000 mA (= 1 A); the value "-70" corresponds to 70% of 2031_h.
- 02_h: Specifies the time in ms that the motor is to continue to travel against the block after block detection.

203Bh I2t Parameters

Function

This object contains the parameters for I^2t monitoring.

I^2t monitoring is activated by entering a value greater than 0 in $203B_h$:01 and $203B_h$:02 and a value greater than 1000 in 6073_h (see I2t Motor overload protection).

With one exception, I^2t monitoring can only be used for *closed loop* mode: If I^2t is activated in *open loop* mode, the current is reduced to the smaller of $203B_h$:01_h, 6073_h and 2031_h.

Object description

Index	$203B_h$
Object name	I2t Parameters
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: tuning
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1512: "Savable" entry changed from "no" to "yes, category: application". Firmware version FIR-v1512: The number of entries was changed from 7 to 8. Firmware version FIR-v1614: "Savable" entry changed from "yes, category: application" to "yes, category: tuning". Firmware version FIR-v1748-B538662: "Name" entry changed from "Nominal Current" to "Motor Rated Current". Firmware version FIR-v1825-B577172: "Name" entry changed from "Motor Rated Current" to "Nominal Current". Firmware version FIR-v1825-B577172: "Name" entry changed from "Nominal Current" to "Motor Rated Current".

	Firmware version FIR-v1825-B577172: "Name" entry changed from "Motor Rated Current" to "Nominal Current". Firmware version FIR-v1825-B577172: "Name" entry changed from "Nominal Current" to "Motor Rated Current". Firmware version FIR-v1825-B577172: The number of entries was changed from 8 to 7. Firmware version FIR-v1926-B648637: "Name" entry changed from "Maximum Duration Of Peak Current" to "Maximum Duration Of Max Current".

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	06_h

Subindex	01_h
Name	Motor Rated Current
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000258_h

Subindex	02_h
Name	Maximum Duration Of Max Current
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	03_h
Name	Threshold
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	04_h
Name	CalcValue
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	05_h
Name	LimitedCurrent
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	06_h
Name	Status
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Description

The subindices are divided into two groups: subindex 01_h and 02_h contain parameters for control, subindices 03_h to 06_h are status values. The functions are as follows:

- 01_h: The rated current specified in the motor data sheet is entered here in mA. This must be smaller than the current entered in 2031_h and 6073_h, otherwise monitoring is not activated. The specified value is interpreted as root mean square.
- 02_h: Specifies the maximum duration of the maximum current (6073_h) in ms.
- 03_h: Threshold, specifies the limit in A^2ms that determines whether the maximum current or rated current is switched to.
- 04_h: CalcValue, specifies the calculated value in A^2ms that is compared with the threshold for setting the current.
- 05_h: LimitedCurrent, contains the momentary current as root mean square set by I^2t.
- 06_h: Current status. If the sub-entry value is "0", I^2t is deactivated; if the value is "1", I^2t is activated.

203Dh Torque Window

Function

Specifies a symmetrical range relative to the target torque within which the target is considered having been met.

If the value is set to "FFFFFFFF"$_h$, monitoring is switched off, the "Target reached" bit in object 6041_h (statusword) is never set.

Object description

Index	$203D_h$
Object name	Torque Window
Object Code	VARIABLE
Data type	UNSIGNED16
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000_h
Firmware version	FIR-v1540
Change history	Firmware version FIR-v1614: "Savable" entry changed from "no" to "yes, category: application".

203Eh Torque Window Time Out

Function

The current torque must be within the "Torque Window" ($203D_h$) for this time (in milliseconds) for the target torque to be considered having been met.

Object description

Index	$203E_h$
Object name	Torque Window Time Out
Object Code	VARIABLE
Data type	UNSIGNED16
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000_h
Firmware version	FIR-v1540
Change history	Firmware version FIR-v1614: "Savable" entry changed from "no" to "yes, category: application". Firmware version FIR-v1738-B501312: "Object Name" entry changed from "Torque Window Time" to "Torque Window Time Out".

203Fh Max Slippage Time Out

Function

Time in milliseconds until an excessively large slippage error in Profile Velocity mode results in an error message.

Object description

Index	$203F_h$
Object name	Max Slippage Time Out
Object Code	VARIABLE
Data type	UNSIGNED16
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0064_h
Firmware version	FIR-v1738-B501312
Change history	

Description

If the actual speed deviates so much from the set speed that the value (absolute value) of the object $60F8_h$ (Max Slippage) is exceeded, bit 13 in object 6041_h is set. The deviation must last longer than the time in object $203F_h$.

A reaction to the slippage error can be set in object 3700_h. If a reaction is defined, an error is also entered in object 1003_h.

2057h Clock Direction Multiplier

Function

The clock count value in Clock-direction mode is multiplied by this value before it is processed further.

Object description

Index	2057_h
Object name	Clock Direction Multiplier
Object Code	VARIABLE
Data type	INTEGER32
Savable	yes, category: application
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000080_h
Firmware version	FIR-v1426
Change history	

2058h Clock Direction Divider

Function

The clock count value in Clock-direction mode is divided by this value before it is processed further.

Object description

Index	2058_h
Object name	Clock Direction Divider
Object Code	VARIABLE
Data type	INTEGER32
Savable	yes, category: application
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001_h
Firmware version	FIR-v1426
Change history	

205Ah Absolute Sensor Boot Value (in User Units)

Function

TIP

This object only has a function when using an absolute encoder. If an absolute encoder is not used, the value is always 0.

The initial encoder position when switching on the controller (in user-defined units) can be read from this object.

Object description

Index	$205A_h$
Object name	Absolute Sensor Boot Value (in User Units)
Object Code	VARIABLE
Data type	INTEGER32
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h
Firmware version	FIR-v1446
Change history	Firmware version FIR-v1512: "Access" table entry for subindex 00 changed from "read/write" to "read only". Firmware version FIR-v1738-B501312: "Object Name" entry changed from "Encoder Boot Value" to "Absolute Sensor Boot Value (in User Units)". Firmware version FIR-v1738-B501312: "Data type" entry changed from "UNSIGNED32" to "INTEGER32".

205Bh Clock Direction Or Clockwise/Counter Clockwise Mode

Function

This object can be used to switch the clock-direction mode (value = "0") to the right/left rotation mode (value = "1").

Object description

Index	$205B_h$
Object name	Clock Direction Or Clockwise/Counter Clockwise Mode
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: application
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h
Firmware version	FIR-v1504
Change history	

205Ch Virtual Encoder Configuration

Function

Use this object to configure the virtual encoder output. You can find details in chapter Virtual encoder output.

Object description

Index	$205C_h$
Object name	Virtual Encoder Configuration
Object Code	ARRAY
Data type	INTEGER16
Savable	yes, category: application
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	
Firmware version	FIR-v2115-B1016293
Change history	Firmware version FIR-v2115-B1016293: "Name" entry changed from "Number Of Increments To Be Sent When No Sensor In 0x3203 Is Choosen" to "Number Of Increments To Be Sent When No Sensor In 0x3203 Is Choosen". Firmware version FIR-v2115-B1016293: "Name" entry changed from "Output Signal Mode: 0=AB, 1=ClkDir, 2=CW/CCW" to "Output Signal Mode: 0=AB, 1=ClkDir, 2=CW/CCW". Firmware version FIR-v2115-B1016293: "Name" entry changed from "Numerator For Conversion Of Sensor Increments To Virtual Encoder Increments" to "Numerator For Conversion Of Sensor Increments To Virtual Encoder Increments".

	Firmware version FIR-v2115-B1016293: "Name" entry changed from "Denominator For Conversion Of Sensor Increments To Virtual Encoder Increments" to "Denominator For Conversion Of Sensor Increments To Virtual Encoder Increments".

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	04_h

Subindex	01_h
Name	Number Of Increments To Be Sent When No Sensor In 0x3203 Is Choosen
Data type	INTEGER16
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000_h

Subindex	02_h
Name	Output Signal Mode: 0=AB, 1=ClkDir, 2=CW/CCW
Data type	INTEGER16
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000_h

Subindex	03_h
Name	Numerator For Conversion Of Sensor Increments To Virtual Encoder Increments
Data type	INTEGER16
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000_h

Subindex	04_h
Name	Denominator For Conversion Of Sensor Increments To Virtual Encoder Increments

Data type	INTEGER16
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0001_h

2084h Bootup Delay

Function

Defines the period between the time that supply voltage is applied to the controller and the functional readiness of the controller in milliseconds.

Object description

Index	2084_h
Object name	Bootup Delay
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: application
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h
Firmware version	FIR-v1426
Change history	

2101h Fieldbus Module Availability

Function

Shows the available fieldbuses.

Object description

Index	2101_h
Object name	Fieldbus Module Availability
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	$001B0011_h$
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1626: "Object Name" entry changed from "Fieldbus Module" to "Fieldbus Module Availability".

Description

Bits 0 to 15 represent the physical interface, bits 16 to 31 the used protocol (if necessary).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
													E-IP	MTCP	MRTU

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									SPI	E-CAT	E-NET	CAN	RS232	RS485	USB

USB
Value = "1": The USB fieldbus is available.

RS-485
Value = "1": An RS-485 interface is available.

RS-232
Value = "1": An RS-232 interface is available.

CAN
Value = "1": The CANopen fieldbus is available.

E-NET
Value = "1": An Ethernet interface is available.

E-CAT
Value = "1": An EtherCAT interface is available.

SPI
Value = "1": An SPI interface is available.

MRTU
Value = "1": The used protocol is Modbus RTU.

MTCP
Value = "1": The used protocol is Modbus TCP

E-IP
Value = "1": The used protocol is EtherNet/IP™

2102h Fieldbus Module Control

Function

This object can be used to activate/deactivate certain fieldbuses (physical interfaces and protocols).

Object description

Index	2102_h
Object name	Fieldbus Module Control
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: communication
Access	read / write
PDO mapping	no
Allowed values	

Preset value	001B0011$_h$
Firmware version	FIR-v1540
Change history	Firmware version FIR-v1626: "Savable" entry changed from "yes, category: application" to "yes, category: communication".

Description

Object 2103$_h$:1$_h$ contains all physical interfaces/protocols that can be activated/deactivated. These can be switched in this object (2102$_h$). The current status of the activated fieldbuses is in object 2103$_h$:2$_h$.

The following distribution of the bits applies here:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
													E-IP	MTCP	MRTU

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									SPI	E-CAT	E-NET	CAN	RS232	RS485	USB

USB
: USB interface

RS-485
: RS-485 interface

RS-232
: RS-232 interface

CAN
: CANopen interface

E-NET
: EtherNet interface

E-CAT
: EtherCAT interface

SPI
: SPI interface

MRTU
: Modbus RTU protocol

MTCP
: Modbus TCP protocol

E-IP
: EtherNet/IP™ protocol

2103h Fieldbus Module Status

Function

Shows the active fieldbuses.

Object description

Index	2103_h
Object name	Fieldbus Module Status
Object Code	ARRAY
Data type	UNSIGNED32
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1540
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Fieldbus Module Disable Mask
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	02_h
Name	Fieldbus Module Enabled
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	$001B0011_h$

Description

Subindex 1 (Fieldbus Module Disable Mask): This subindex contains all physical interfaces and protocols that can be activated or deactivated. A value "1" means that this fieldbus can be deactivated.

Subindex 2 (Fieldbus Module Enabled): This subindex contains all currently activated physical interfaces and protocols. The value "1" means that that the fieldbus is active.

The following distribution of the bits applies for subindices 1 and 2:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
													E-IP	MTCP	MRTU

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									SPI	E-CAT	E-NET	CAN	RS232	RS485	USB

USB
USB interface

RS-485
RS-485 interface

RS-232
RS-232 interface

CAN
CANopen interface

E-NET
EtherNet interface

E-CAT
EtherCAT interface

SPI
SPI interface

MRTU
Modbus RTU protocol

MTCP
Modbus TCP protocol

E-IP
EtherNet/IP™ protocol

2290h PDI Control

Function

With this object, you can activate the *Plug&Drive interface*. You can find additional information in document *Function description Plug&Drive interface*.

Object description

Index	2290_h
Object name	PDI Control
Object Code	VARIABLE
Data type	UNSIGNED8
Savable	yes, category: application
Access	read / write

PDO mapping	no
Allowed values	
Preset value	01_h
Firmware version	FIR-v1748-B531667
Change history	Firmware version FIR-v1748-B538662: "Access" table entry for subindex 00 changed from "read only" to "read/write".

Description

To activate the *Plug&Drive interface*, set bit 0 to "1".

2291h PDI Input

Function

If you use the *Plug&Drive interface*, you can use this object to select and start the operating mode and set the corresponding target values (target position, speed, etc.). You can find additional information in document *Function description Plug&Drive interface*.

Object description

Index	2291_h
Object name	PDI Input
Object Code	RECORD
Data type	PDI_INPUT
Savable	no
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	
Firmware version	FIR-v1748-B531667
Change history	Firmware version FIR-v2013-B726332: "Savable" entry changed from "yes, category: application" to "no".

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	04_h

Subindex	01_h
Name	PDI Set Value 1
Data type	INTEGER32

Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h

Subindex	02_h
Name	PDI Set Value 2
Data type	INTEGER16
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000_h

Subindex	03_h
Name	PDI Set Value 3
Data type	INTEGER8
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00_h

Subindex	04_h
Name	PDI Command
Data type	INTEGER8
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00_h

2292h PDI Output

Function

If you use the *Plug&Drive interface*, you can, in this object, read the status and a return value that is dependent on the used operating mode. You can find additional information in document *Function description Plug&Drive interface*.

Object description

Index	2292_h
Object name	PDI Output
Object Code	RECORD
Data type	PDI_OUTPUT
Savable	no
Access	read only
PDO mapping	TX-PDO

Allowed values	
Preset value	
Firmware version	FIR-v1748-B531667
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	02_h

Subindex	01_h
Name	PDI Status
Data type	INTEGER16
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	0000_h

Subindex	02_h
Name	PDI Return Value
Data type	INTEGER32
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h

2300h NanoJ Control

Function

Controls the execution of a *NanoJ program*.

Object description

Index	2300_h
Object name	NanoJ Control
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: application
Access	read / write

PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1436: "Object Name" entry changed from "VMM Control" to "NanoJ Control".

Description

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
													AYield		ON

ON

Switches the *NanoJ program* on (value = "1") or off (value = "0").

With a rising edge in bit 0, the program is first reloaded and the variable range reset.

NOTICE

Startup of the *NanoJ program* can take up to 200 ms.

When switching on, a check is performed to determine whether a *NanoJ program* is present. If present, "1" is entered in 2300 and the *NanoJ program* is started.

AYield (AutoYield)

If this feature is activated (bit set to "1"), the *NanoJ program* is no longer stopped if it runs longer than it is allowed to. The *NanoJ program* is, thus, no longer real-time capable and no longer runs every 1 ms (see Available computing time).

NOTICE

Do not use the Debug output if *AutoYield* mode is activated.

2301h NanoJ Status

Function

Indicates the operating state of the user program.

Object description

Index	2301_h
Object name	NanoJ Status
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	no

Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1436: "Object Name" entry changed from "VMM Status" to "NanoJ Status".

Description

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
													ERR	RES	RUN

RUN

Value = "0": Program is stopped, value = "1": NanoJ program is running.

RES

Reserved.

ERR

Program was ended with an error. Cause of the error can be read from object 2302_h.

2302h NanoJ Error Code

Function

Indicates which error occurred during the execution of the user program.

Object description

Index	2302_h
Object name	NanoJ Error Code
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1436: "Object Name" entry changed from "VMM Error Code" to "NanoJ Error Code".

Description

Error codes during program execution:

Number	Description
0001_h	Firmware does not support the used function (e.g., `sin`, `cosin`, etc.)
0005_h	Time Out: Code executed too long without `yield()` or `sleep()`
0007_h	Too many variables on the stack
0100_h	Invalid NanoJ program file
0101_h	Invalid NanoJ version of the program file
0102_h	CRC error in the NanoJ program file

Error when accessing an object:

Number	Description
$1xxxxyy_h$	Invalid mapping in the NanoJ program file: The value in "xxxx" specifies the index, the value in "yy" specifies the subindex of the object that should – but cannot – be mapped.
2000000_h	Invalid mapping in the NanoJ program file: too many variables of type `input` were declared (see 2310h NanoJ Input Data Selection)
3000000_h	Invalid mapping in the NanoJ program file: too many variables of type `output` were declared (see 2320h NanoJ Output Data Selection)
4000000_h	Invalid mapping in the NanoJ program file: too many variables of type `inout` were declared (see 2330h NanoJ In/output Data Selection)
1000_h	Access of a nonexistent object in the object dictionary
1001_h	Write access of a write-protected entry in the OD
1002_h	An attempt was made to write a value that is too low or too high to an object.
1003_h	An attempt was made to read out an object that permits only write access.
$1FFF_h$	Unauthorized access of an object

230Eh Timer

Function

This object contains the operating time in milliseconds since the last time the controller was started.

NOTICE

This object is not stored; counting begins with "0" again after switching on or an overflow.

Object description

Index	$230E_h$
Object name	Timer
Object Code	ARRAY
Data type	UNSIGNED32
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	

Firmware version	FIR-v2139-B1020888
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	01_h

Subindex	01_h
Name	1ms Timer
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h

230Fh Uptime Seconds

Function

This object contains the operating time in seconds since the last time the controller was started.

NOTICE

This object is not stored; counting begins with "0" again after switching on or an overflow.

Object description

Index	$230F_h$
Object name	Uptime Seconds
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h
Firmware version	FIR-v1436
Change history	

2310h NanoJ Input Data Selection

Function

Describes the object dictionary entries that are copied to the PDO mapping input of the NanoJ program.

Object description

Index	2310_h
Object name	NanoJ Input Data Selection
Object Code	ARRAY
Data type	UNSIGNED32
Savable	no
Access	read / write
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1650-B472161
Change history	Firmware version FIR-v1436: "Object Name" entry changed from "VMM Input Data Selection" to "NanoJ Input Data Selection". Firmware version FIR-v1650-B472161: "Savable" entry changed from "yes, category: application" to "no". Firmware version FIR-v1650-B472161: "Access" table entry for subindex 00 changed from "read/write" to "read only". Firmware version FIR-v1650-B472161: "Access" table entry for subindex 01 changed from "read/write" to "read only".

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	10_h

Subindex	01_h - 10_h
Name	Mapping #1 - #16
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h

Description

Each subindex (1–16) describes a different mapped object.

A mapping entry consists of four bytes, which are structured according to the following graphic.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Index [16]															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SubIndex [8]								Length [8]							

Index [16]

This contains the index of the object to be mapped

Subindex [8]

This contains the subindex of the object to be mapped

Length [8]

This contains the length of the object to be mapped in units of bits.

2320h NanoJ Output Data Selection

Function

Describes the object dictionary entries that are copied into the output PDO mapping of the *NanoJ program* after it is executed.

Object description

Index	2320_h
Object name	NanoJ Output Data Selection
Object Code	ARRAY
Data type	UNSIGNED32
Savable	no
Access	read / write
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1650-B472161
Change history	Firmware version FIR-v1436: "Object Name" entry changed from "VMM Output Data Selection" to "NanoJ Output Data Selection". Firmware version FIR-v1650-B472161: "Savable" entry changed from "yes, category: application" to "no". Firmware version FIR-v1650-B472161: "Access" table entry for subindex 00 changed from "read/write" to "read only". Firmware version FIR-v1650-B472161: "Access" table entry for subindex 01 changed from "read/write" to "read only".

Value description

Subindex	00_h

Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	10_h

Subindex	01_h - 10_h
Name	Mapping #1 - #16
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h

Description

Each subindex (1–16) describes a different mapped object.

A mapping entry consists of four bytes, which are structured according to the following graphic.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Index [16]															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SubIndex [8]								Length [8]							

Index [16]
: This contains the index of the object to be mapped

Subindex [8]
: This contains the subindex of the object to be mapped

Length [8]
: This contains the length of the object to be mapped in units of bits.

2330h NanoJ In/output Data Selection

Function

Describes the object dictionary entries that are first copied to the input PDO mapping of the NanoJ program and, after it is executed, are copied back to the output PDO mapping.

Object description

Index	2330_h
Object name	NanoJ In/output Data Selection
Object Code	ARRAY
Data type	UNSIGNED32
Savable	no
Access	read / write
PDO mapping	no

Allowed values	
Preset value	
Firmware version	FIR-v1650-B472161
Change history	Firmware version FIR-v1436: "Object Name" entry changed from "VMM In/output Data Selection" to "NanoJ In/output Data Selection". Firmware version FIR-v1650-B472161: "Savable" entry changed from "yes, category: application" to "no". Firmware version FIR-v1650-B472161: "Access" table entry for subindex 00 changed from "read/write" to "read only". Firmware version FIR-v1650-B472161: "Access" table entry for subindex 01 changed from "read/write" to "read only".

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	10_h

Subindex	01_h - 10_h
Name	Mapping #1 - #16
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h

Description

Each subindex (1–16) describes a different mapped object.

A mapping entry consists of four bytes, which are structured according to the following graphic.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Index [16]															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SubIndex [8]								Length [8]							

Index [16]

This contains the index of the object to be mapped

Subindex [8]

This contains the subindex of the object to be mapped

Length [8]

This contains the length of the object to be mapped in units of bits.

2400h NanoJ Inputs

Function

Located here is an array with 32, 32-bit integer values that is not used within the firmware and serves only for communicating with the user program via the fieldbus.

Object description

Index	2400_h
Object name	NanoJ Inputs
Object Code	ARRAY
Data type	INTEGER32
Savable	no
Firmware version	FIR-v1426
Change history	The number of entries was changed from 2 to 33 Firmware version FIR-v1436: "Object Name" entry changed from "VMM Inputs" to "NanoJ Inputs". Firmware version FIR-v1436: "Name" entry changed from "VMM Input N#" to "NanoJ Input N#".

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	20_h

Subindex	01_h - 20_h
Name	NanoJ Input #1 - #32
Data type	INTEGER32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h

Description

Here, it is possible to pass, e.g., preset values, to the *NanoJ program*.

2410h NanoJ Init Parameters

Function

This object functions identically to object 2400_h with the difference that this object can be stored.

Object description

Index	2410_h
Object name	NanoJ Init Parameters
Object Code	ARRAY
Data type	INTEGER32
Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1450
Change history	Firmware version FIR-v1450: "Data Type" entry changed from "INTEGER32" to "UNSIGNED8".

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	20_h

Subindex	01_h - 20_h
Name	NanoJ Init Parameter #1 - #32
Data type	INTEGER32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h

2500h NanoJ Outputs

Function

Located here is an array with 32, 32-bit integer values that is not used within the firmware and serves only for communicating with the user program via the fieldbus.

Object description

Index	2500_h
Object name	NanoJ Outputs
Object Code	ARRAY
Data type	INTEGER32
Savable	no
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1436: "Object Name" entry changed from "VMM Outputs" to "NanoJ Outputs". Firmware version FIR-v1436: "Name" entry changed from "VMM Output N#" to "NanoJ Output N#".

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	20_h

Subindex	01_h - 20_h
Name	NanoJ Output #1 - #32
Data type	INTEGER32
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h

Description

Here, the *NanoJ program* can store results which can then be read out via the fieldbus.

2600h NanoJ Debug Output

Function

This object contains debug output of a user program.

Object description

Index	2600_h
Object name	NanoJ Debug Output
Object Code	ARRAY
Data type	UNSIGNED8

Savable	no
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1436: "Object Name" entry changed from "VMM Debug Output" to "NanoJ Debug Output".

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00_h

Subindex	01_h - 40_h
Name	Value #1 - #64
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	00_h

Description

Here, the NanoJ program stores the debug output that was called up with the `VmmDebugOutputString()` and `VmmDebugOutputInt()`.

2701h Customer Storage Area

Function

Data can be deposited and stored in this object.

Object description

Index	2701_h
Object name	Customer Storage Area
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: customer
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1540

Change history	Firmware version FIR-v1540: "Data Type" entry changed from "UNSIGNED32" to "UNSIGNED8".

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	FE_h

Subindex	01_h - FE_h
Name	Storage #1 - #254
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

2800h Bootloader And Reboot Settings

Function

With this object, a reboot of the firmware can be triggered and the short circuiting of the motor windings in boot loader mode switched off and on.

Object description

Index	2800_h
Object name	Bootloader And Reboot Settings
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1540
Change history	

Value description

Subindex	00_h

Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	03_h

Subindex	01_h
Name	Reboot Command
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	02_h
Name	Reboot Delay Time In Ms
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	03_h
Name	Bootloader HW Config
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Description

The subindices have the following function:

- 01_h: If the value "$746F6F62_h$" is entered here, the firmware is rebooted.
- 02_h: Time in milliseconds: delays the reboot of the firmware by the respective time.
- 03_h: Bit 0 can be used to switch short circuiting of the motor windings in boot loader mode off and on:
  - Bit 0 = 1: Short circuiting of the motor windings in boot loader mode is switched off.
  - Bit 0 = 0: Short circuiting of the motor windings in boot loader mode is switched on.

3202h Motor Drive Submode Select

Function

Controls the controller mode, such as the changeover between *closed loop* / *open loop* and whether Velocity Mode is simulated via the S-controller or functions with a real V-controller in *closed loop*.

Object description

Index	3202_h
Object name	Motor Drive Submode Select
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: drive
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1540: "Savable" entry changed from "yes category: application" to "yes, category: travel". Firmware version FIR-v1540: "Savable" entry changed from "yes category: travel" to "yes, category: movement".

Description

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								Slow	BLDC	Torque	AutoAl	CurRed	Brake	VoS	CL/OL

CL/OL

Changeover between *open loop* and *closed loop* (see chapter Control modes)

- Value = "0": *open loop*
- Value = "1": *closed loop*

Toggling is not possible in the *Operation enabled* state.

VoS

Value = "1": Simulate V-controller with an S-ramp: simulate the speed modes through continuous position changes

Brake

Value = "1": Switch on automatic brake control.

CurRed (Current Reduction)

Value = "1": Current reduction activated in *open loop*

AutoAl (*auto alignment*)

For the case that operation in *closed loop* is required (bit 0 in 3202_h is set).

Value = "1": The *auto alignment* process is activated; immediately after switching on, an alignment is determined in *open loop* and a switch is immediately made to *closed loop* mode without the encoder index having been seen.

The rotor is moved slightly during this process.

Value = "0": No *auto alignment*, the motor operates in *open loop* until the encoder index is seen (maximum one revolution of the motor shaft).

If the incremental encoder used for commutation does not have an index (bit 0 in $33A0_h$ is "0"), an *auto alignment* is always determined.

Torque

only active in operating modes Profile Torque and Cyclic Synchronous Torque

Value = "1": M-controller is active, otherwise a V-controller is superimposed: no V-controller is used in the torque modes for speed limiting, thus object 6080_h is ignored; 3210_h:3 and 3210_h:4 have no effect on the control.

BLDC

Value = "1": Motor type "BLDC" (brushless DC motor)

3203h Feedback Selection

Function

In this object, the sources of the presets are defined for the commutation and the velocity and position control.

A value change in the *Operation enabled* state shows no immediate effect. Value changes in objects are buffered and read out upon changing to the *Operation enabled* state.

Object description

Index	3203_h
Object name	Feedback Selection
Object Code	ARRAY
Data type	UNSIGNED8
Savable	yes, category: tuning
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	
Firmware version	FIR-v1748-B538662
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	05_h

Subindex	01_h
Name	1st Feedback Interface
Data type	UNSIGNED8
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00_h

Subindex	02_h
Name	2nd Feedback Interface
Data type	UNSIGNED8
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00_h

Subindex	03_h
Name	3rd Feedback Interface
Data type	UNSIGNED8
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00_h

Subindex	04_h
Name	4th Feedback Interface
Data type	UNSIGNED8
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00_h

Subindex	05_h
Name	5th Feedback Interface
Data type	UNSIGNED8
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00_h

Description

The subindices have the following function:

- 00_h: Value="1" to "n", where "n" is the number of existing feedbacks.

- n_h:
  Subindex n contains a bit mask for the respective feedback n. The bits have the following meaning here:
- Bit 0: If the bit is set to "1", this sensor is used for position feedback.
- Bit 1: If the bit is set to "1", this sensor is used for velocity feedback.
- Bit 2: If the bit is set to "1", this sensor is used for commutation feedback in Closed-Loop.

Subindex 01_h always corresponds to the first (and always existing) *sensorless* feedback.

Which sensor the controller takes into account for the individual controllers (commutation, velocity, position) is implicitly specified by the order of the sensors.

The search always begins with sensor 2 and continues in ascending order until all existing sensors have been queried. If a sensor is found whose feedback is set, it is assigned to the corresponding controller and the search ended.

NOTICE

If bit 0 in 3202_h is set to "0", *closed loop* is deactivated; bit 2 (commutation) then has no meaning. Bit 1 for the velocity and bit 0 for the position in the respective subindices are still used for the display of the actual position and speed values.

3204h Feedback Mapping

Function

This object contains information on the existing feedbacks.

Object description

Index	3204_h
Object name	Feedback Mapping
Object Code	ARRAY
Data type	UNSIGNED16
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	
Firmware version	FIR-v1748-B538662
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	05_h

Subindex	01_h
Name	Index Of 1st Feedback Interface
Data type	UNSIGNED16
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	3380_h

Subindex	02_h
Name	Index Of 2nd Feedback Interface
Data type	UNSIGNED16
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	3390_h

Subindex	03_h
Name	Index Of 3rd Feedback Interface
Data type	UNSIGNED16
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	$33A0_h$

Subindex	04_h
Name	Index Of 4th Feedback Interface
Data type	UNSIGNED16
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	$33B0_h$

Subindex	05_h
Name	Index Of 5th Feedback Interface
Data type	UNSIGNED16
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	$33B1_h$

Description

The subindices have the following function:

- 00_h: Value="1" to "n", where "n" is the number of existing feedbacks.

- n_h:
  Subindex n refers to the index of the respective object for the configuration of the corresponding feedback.
  Subindex 01_h always corresponds to the first (and always existing) *sensorless* feedback.

320Dh Torque Of Inertia Factor

Function

This factor is used for calculating the acceleration feed forward (see 321D). Default is 0 (feed forward inactive).

Acceleration feed forward applies during deceleration as well.

Object description

Index	$320D_h$
Object name	Torque Of Inertia Factor
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: drive
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1825-B577172
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Current
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

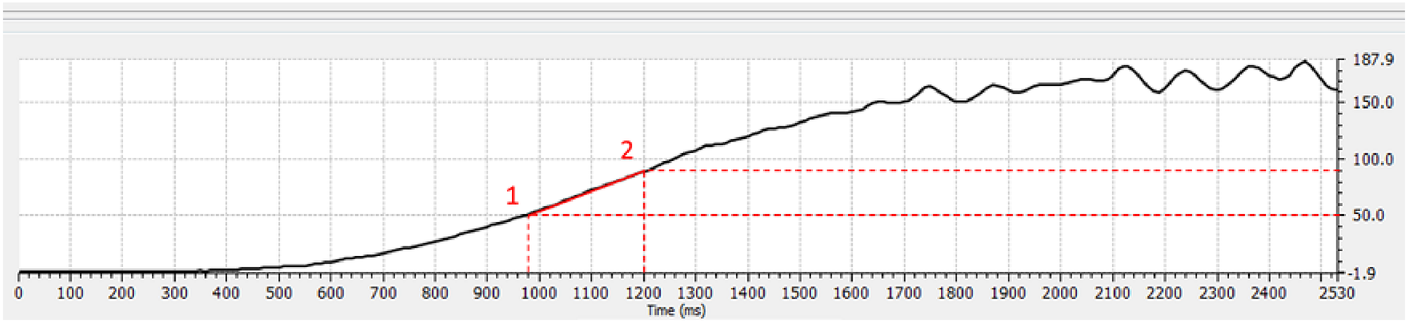
Subindex	02_h
Name	Acceleration

Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000$_h$

Description

The value is dependent on the inertia of the load. To determine the factor:

1. Activate *closed loop* and select the *profile torque* mode.
2. Set a target for the torque and enter the corresponding current value (mA) in 320D$_h$:01$_h$.
3. Record (e. g., in *Plug & Drive Studio*) the current speed (object 606C$_h$). Calculate the acceleration in the set user-defined units for the speed range, where this is constant. Enter the value in 320D$_h$:02$_h$.
   Using the speed curve in the following figure as an example:
   (90-50)/(1200-980)=182 rpm/s.



3210h Motor Drive Parameter Set

Function

Contains the P and I components of the current, speed and position controllers for *open-loop* (only current controller activated) and *closed-loop*.

Object description

Index	3210$_h$
Object name	Motor Drive Parameter Set
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1626: "Name" entry changed from "S_P" to "Position Loop, Proportional Gain (closed-loop)". Firmware version FIR-v1626: "Name" entry changed from "S_I" to "Position Loop, Integral Gain (closed-loop)". Firmware version FIR-v1626: "Name" entry changed from "V_P" to "Velocity Loop, Proportional Gain (closed-loop)".

	Firmware version FIR-v1626: "Name" entry changed from "V_I" to "Velocity Loop, Integral Gain (closed-loop)".
	Firmware version FIR-v1626: "Name" entry changed from "Id_P" to "Flux Current Loop, Proportional Gain (closed-loop)".
	Firmware version FIR-v1626: "Name" entry changed from "Id_I" to "Flux Current Loop, Integral Gain (closed-loop)".
	Firmware version FIR-v1626: "Name" entry changed from "Iq_P" to "Torque Current Loop, Proportional Gain (closed-loop)".
	Firmware version FIR-v1626: "Name" entry changed from "Iq_I" to "Torque Current Loop, Integral Gain (closed-loop)".
	Firmware version FIR-v1626: "Name" entry changed from "I_P" to "Torque Current Loop, Proportional Gain (dspDrive – Stepper Motor, open-loop)".
	Firmware version FIR-v1626: "Name" entry changed from "I_I" to "Torque Current Loop, Integral Gain (dspDrive – Stepper Motor, open-loop)".
	Firmware version FIR-v1650-B472161: "Name" entry changed from "Torque Current Loop, Proportional Gain (dspDrive – Stepper Motor, open-loop)" to "Torque Current Loop, Proportional Gain (open-loop)".
	Firmware version FIR-v1650-B472161: "Name" entry changed from "Torque Current Loop, Integral Gain (dspDrive – Stepper Motor, open-loop)" to "Torque Current Loop, Integral Gain (open-loop)".
	Firmware version FIR-v1650-B472161: "Data type" entry changed from "INTEGER32" to "UNSIGNED32".
	Firmware version FIR-v1650-B472161: "Data type" entry changed from "INTEGER32" to "UNSIGNED32".
	Firmware version FIR-v1738-B501312: The number of entries was changed from 11 to 13.
	Firmware version FIR-v1738-B501312: "PDO mapping" table entry for subindex 00 to 0A changed from "no" to "RX-PDO".

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	$0C_h$

Subindex	01_h
Name	Position Loop, Proportional Gain (closed Loop)
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO

Allowed values	
Preset value	00000800$_h$

Subindex	02$_h$
Name	Position Loop, Integral Gain (closed Loop)
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000$_h$

Subindex	03$_h$
Name	Velocity Loop, Proportional Gain (closed Loop)
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00002EE0$_h$

Subindex	04$_h$
Name	Velocity Loop, Integral Gain (closed Loop)
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000001E$_h$

Subindex	05$_h$
Name	Flux Current Loop, Proportional Gain (closed Loop)
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00881EE0$_h$

Subindex	06$_h$
Name	Flux Current Loop, Integral Gain (closed Loop)
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0007C740$_h$

Subindex	07_h
Name	Torque Current Loop, Proportional Gain (closed Loop)
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	$00881EE0_h$

Subindex	08_h
Name	Torque Current Loop, Integral Gain (closed Loop)
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	$0007C740_h$

Subindex	09_h
Name	Torque Current Loop, Proportional Gain (open Loop)
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	$004DC880_h$

Subindex	$0A_h$
Name	Torque Current Loop, Integral Gain (open Loop)
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	$001D2B30_h$

Subindex	$0B_h$
Name	Velocity Feed Forward Factor In Per Mille
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	$000003E8_h$

Subindex	$0C_h$
Name	Acceleration Feed Forward Factor
Data type	UNSIGNED32

Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h

Description

- Subindex 00_h: Number of entries
- Subindex 01_h: Proportional component of the S-controller (position)
- Subindex 02_h: Integral component of the S-controller (position)
- Subindex 03_h: Proportional component of the V-controller (speed)
- Subindex 04_h: Integral component of the V-controller (speed)
- Subindex 05_h: (Closed loop) Proportional component of the current controller of the field-forming component
- Subindex 06_h: (Closed loop) Integral component of the current controller of the field-forming component
- Subindex 07_h: (Closed loop) Proportional component of the current controller of the torque-forming component
- Subindex 08_h: (Closed loop) Integral component of the current controller of the torque-forming component
- Subindex 09_h: (Open-loop) Proportional component of the current controller of the field-building component
- Subindex $0A_h$: (Open-loop) Integral component of the current controller of the field-forming component
- Subindex $0B_h$: (Closed loop) Speed feed forward in tenths of a percent. Default is 1000 and, thus, a factor of 1.
- Subindex $0C_h$: (Closed loop) Acceleration feed forward. Default is 0 (feed forward inactive). It applies during deceleration as well.

3212h Motor Drive Flags

Function

This object is used to specify whether or not auto setup is to adapt the controller parameters. In addition, the direction of the rotary field and the objects for the control parameters can be changed.

NOTICE

Changes in subindex 02_h do not take effect until after the controller is restarted. Afterwards, Auto setup must again be performed.

Object description

Index	3212_h
Object name	Motor Drive Flags
Object Code	ARRAY
Data type	INTEGER8
Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1450

Change history	Firmware version FIR-v1512: The number of entries was changed from 2 to 3. Firmware version FIR-v1738-B501312: "Name" entry changed from "Enable Legacy Power Mode" to "Reserved". Firmware version FIR-v2213: The number of entries was changed from 3 to 4.

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	04_h

Subindex	01_h
Name	Reserved
Data type	INTEGER8
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00_h

Subindex	02_h
Name	Override Field Inversion
Data type	INTEGER8
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00_h

Subindex	03_h
Name	Auto-setup With Current Controller Parameters From The OD
Data type	INTEGER8
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00_h

Subindex	04_h
Name	Use 321Ah, 321Bh, 321Ch, 321Dh, Instead Of 3210h

Data type	INTEGER8
Access	read / write
PDO mapping	no
Allowed values	
Preset value	01_h

Description

Valid values for subindex 02_h:

- Value = "0": Use default values of the firmware
- Value = "1": Force non-inversion of the rotating field (mathematically positive)
- Value = "-1": Force inversion of the rotating field (mathematically negative)

Valid values for subindex 03_h:

- Value = "0": Auto setup detects the motor type (stepper motor or BLDC motor) and uses the corresponding pre-configured parameter set.
- Value = "1": Perform auto setup with the control parameters that were entered in object 3210_h or $321A_h$ to $321E_h$ before the auto setup. The control parameters are not changed.

Valid values for subindex 04_h:

- Value = "0": The old control parameters from object 3210_h are used.
- Value = "1": The new control parameters (see Controller structure) are used.

321Ah Current Controller Parameters

Function

Contains the parameters for the current controller (commutation). As a rule, the values for Iq (subindex 01_h/02_h) and Id (subindex 03_h/04_h) should be the same. See chapter Controller structure.

Object description

Index	$321A_h$
Object name	Current Controller Parameters
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v2213-B1028181
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only

PDO mapping	no
Allowed values	
Preset value	04_h

Subindex	01_h
Name	Proportional Gain Kp For Iq [mV/A]
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	$000027E4_h$

Subindex	02_h
Name	Integrator Time Ti For Iq [µs]
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000446_h

Subindex	03_h
Name	Proportional Gain Kp For Id [mV/A]
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	$000027E4_h$

Subindex	04_h
Name	Integrator Time Ti For Id [µs]
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000446_h

321Bh Velocity Controller Parameters

Function

Contains the parameters for the velocity controller. See chapter Controller structure.

Object description

Index	$321B_h$

Object name	Velocity Controller Parameters
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v2213-B1028181
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Proportional Gain Kp [mA/Hz]
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000180_h

Subindex	02_h
Name	Integrator Time Ti [µs]
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	$000186A0_h$

321Ch Position Controller Parameters

Function

Contains the parameters for the position controller. See chapter Controller structure.

Object description

Index	$321C_h$
Object name	Position Controller Parameters
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v2213-B1028181
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Proportional Gain Kp [Hz]
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000032_h

Subindex	02_h
Name	Integrator Time Ti [µs]
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

321Dh Pre-control

Function

Contains the parameters for the feed forward. See chapter Controller structure.

Object description

Index	321Dh
Object name	Pre-control
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v2213-B1028181
Change history	

Value description

Subindex	00h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	03h

Subindex	01h
Name	Voltage Pre-control [‰]
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	000003E8h

Subindex	02h
Name	Acceleration Pre-control [‰]
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	000003E8h

Subindex	03h
Name	Velocity Pre-control [‰]
Data type	UNSIGNED32
Access	read / write

PDO mapping	no
Allowed values	
Preset value	$000003E8_h$

321Eh Voltage Limit

Function

Maximum permissible PWM voltage (duty cycle). Values ≤ 1000 are interpreted as per mil values (of the available voltage). Values > 1000 as millivolt. See also chapter Controller structure.

Object description

Index	$321E_h$
Object name	Voltage Limit
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: application
Access	read / write
PDO mapping	no
Allowed values	
Preset value	$000186A0_h$
Firmware version	FIR-v2213-B1028181
Change history	

Description

Also dependent on this value is whether the *overmodulation* of the voltage vector is used. If *overmodulation* is used, a higher torque can be achieved. The resulting voltage is no longer sinusoidal, which can result in harmonics and higher losses.

Value in mV	Overmodulation
1001…U_{o_low}	None; the voltage vector describes a circle.
U_{o_low}…U_{o_high}	The voltage vector describes a circle that is increasingly flattened on four/six sides in proportion to the set value.
≥U_{o_high}	Full; the voltage vector describes a square or a hexagon.

U_{o_low}

The lowest voltage above which overmodulation occurs. Is calculated as follows:

Operating voltage*0.9425

U_{o_high}

The maximum overmodulation occurs above this voltage. Is calculated as follows:

With two-phase stepper motors: operating voltage*1.063

With three-phase BLDC motors: operating voltage*0.99

3220h Analog Input Digits

Function

Displays the instantaneous values of the analog inputs in *ADC digits*.

With object 3221_h, the respective analog input can be configured as current or voltage input.

Object description

Index	3220_h
Object name	Analog Input Digits
Object Code	ARRAY
Data type	INTEGER16
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1426
Change history	

Value description

Subindex	00_h
Name	Number Of Analog Input Digits
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h - 02_h
Name	Analog Input #1 - #2
Data type	INTEGER16
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	0000_h

Description

Formulas for converting from [digits] to the respective unit:

- Current input (if configurable): x digits * 20 mA / 1023 digits

3221h Analog Inputs Control

Function

With this object, an analog input can be switched from voltage measurement to current measurement if permitted by the hardware (see technical data).

See chapter Analog Input Control

Object description

Index	3221_h
Object name	Analog Inputs Control
Object Code	VARIABLE
Data type	INTEGER32
Savable	yes, category: application
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h
Firmware version	FIR-v1426
Change history	

Description

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
														AC2	AC1

In general: If a bit is set to the value "0", the analog input measures the voltage; if the bit is set to the value "1", the current is measured.

AC1

Setting for analog input 1

AC2

Setting for analog input 2

3240h Digital Inputs Control

Function

With this object, digital inputs can be manipulated as described in chapter Digital inputs and outputs .

Object description

Index	3240_h
Object name	Digital Inputs Control
Object Code	ARRAY
Data type	UNSIGNED32

Savable	yes, category: application
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1426: Subindex 01_h: "Name" entry changed from "Special Function Disable" to "Special Function Enable" Firmware version FIR-v1512: The number of entries was changed from 8 to 9.

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	08_h

Subindex	01_h
Name	Special Function Enable
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h

Subindex	02_h
Name	Function Inverted
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h

Subindex	03_h
Name	Force Enable
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h

Subindex	04_h
Name	Force Value

Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h

Subindex	05_h
Name	Raw Value
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h

Subindex	06_h
Name	Input Range Select
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h

Subindex	07_h
Name	Differential Select
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h

Subindex	08_h
Name	Routing Enable
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001_h

Description

The subindices have the following function:

- 3240_h:01_h (Special Function Enable): This bit allows special functions of an input to be switched off (value "0") or on (value "1"). If input 1 is not used as, e.g., a negative limit switch, the special function must be switched off to prevent an erroneous response to the signal generator. The object has no effect on bits 16 to 31.

The firmware evaluates the following bits:

- Bit 0: Negative limit switch
- Bit 1: Positive limit switch
- Bit 2: Home switch
- Bit 3: Interlock

If, for example, two limit switches and one home switch are used, bits 0–2 in 3240_h:01_h must be set to "1".

- 3240_h:02_h (Function Inverted): This subindex switches from normally open logic (a logical high level at the input yields the value "1" in object $60FD_h$) to normally closed logic (the logical high level at the input yields the value "0").
This applies for the special functions (except for the clock and direction inputs) and for the normal inputs. If the bit has the value "0", normally open logic applies; for the value "1", normally closed logic applies. Bit 0 changes the logic of input 1, bit 1 changes the logic of input 2, etc.
- 3240_h:03_h (Force Enable): This subindex switches on the software simulation of input values if the corresponding bit is set to "1".
In this case, the actual values are no longer used in object 3240_h:04_h, but rather the set values for the respective input.
- 3240_h:04_h (Force Value): This bit specifies the value that is to be read as the input value if the same bit was set in object 3240_h:03_h.
- 3240_h:05_h (Raw Value): This object contains the unmodified input value.
- 3240_h:07_h (Differential Select): With the inputs, this subindex switches between "single-ended input" (value "0" in the subindex) and "differential input" (value "1" in the subindex) if the input supports this function.
- 3240_h:08_h (Routing Enable): The value "1" in this subindex activates Input Routing.

3241h Digital Input Capture

Function

With this object, the encoder position can be noted automatically if a level change occurs at digital input .

Object description

Index	3241_h
Object name	Digital Input Capture
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	
Firmware version	FIR-v1446
Change history	Firmware version FIR-v1446: "Data type" entry changed from "UNSIGNED32" to "UNSIGNED8". Firmware version FIR-v1738-B501312: "Name" entry changed from "Encoder Raw Value" to "Sensor Raw Value". Firmware version FIR-v1748-B531667: "PDO mapping" table entry for subindex 00 changed from "no" to "TX-PDO". Firmware version FIR-v1748-B531667: "PDO mapping" table entry for subindex 01 changed from "RX-PDO" to "TX-PDO".

	Firmware version FIR-v1748-B531667: "PDO mapping" table entry for subindex 02 changed from "RX-PDO" to "TX-PDO".
	Firmware version FIR-v1748-B531667: "PDO mapping" table entry for subindex 03 changed from "RX-PDO" to "TX-PDO".
	Firmware version FIR-v1748-B531667: "PDO mapping" table entry for subindex 04 changed from "RX-PDO" to "TX-PDO".

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	08_h

Subindex	01_h
Name	Control For Capture Input 1
Data type	UNSIGNED32
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h

Subindex	02_h
Name	Capture Count For Capture Input 1
Data type	UNSIGNED32
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h

Subindex	03_h
Name	Value For Capture Input 1
Data type	UNSIGNED32
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h

Subindex	04_h
Name	Sensor Raw Value For Capture Input 1

Data type	UNSIGNED32
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h

Subindex	05_h
Name	Control For Capture Input 2
Data type	UNSIGNED32
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h

Subindex	06_h
Name	Capture Count For Capture Input 2
Data type	UNSIGNED32
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h

Subindex	07_h
Name	Value For Capture Input 2
Data type	UNSIGNED32
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h

Subindex	08_h
Name	Sensor Raw Value For Capture Input 2
Data type	UNSIGNED32
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h

Description

- Subindex 01_h: This is used to select the type of level change:
  - Deactivate function: Value "0"
  - With rising edge: Value "1"
  - With falling edge: Value "2"

  - Both edges: Value "3"
- Subindex 02_h: Specifies the number of the noted level changes since the time the function was started; is reset to 0 if subindex 01_h is set to 1,2 or 3
- Subindex 03_h: Encoder position of the level change (in absolute user units from 6064_h)
- Subindex 04_h: Encoder position of the level change

3242h Digital Input Routing

Function

This object determines the source of the input routing that ends in $60FD_h$.

Object description

Index	3242_h
Object name	Digital Input Routing
Object Code	ARRAY
Data type	UNSIGNED8
Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1504
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	20_h

Subindex	01_h
Name	Input Source For Bit #1 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	00_h

Subindex	02_h
Name	Input Source For Bit #2 In 60FDh
Data type	UNSIGNED8

Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	00_h

Subindex	03_h
Name	Input Source For Bit #3 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	00_h

Subindex	04_h
Name	Input Source For Bit #4 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	00_h

Subindex	05_h
Name	Input Source For Bit #5 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	00_h

Subindex	06_h
Name	Input Source For Bit #6 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	00_h

Subindex	07_h
Name	Input Source For Bit #7 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	

Preset value	00_h

Subindex	08_h
Name	Input Source For Bit #8 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	00_h

Subindex	09_h
Name	Input Source For Bit #9 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	00_h

Subindex	$0A_h$
Name	Input Source For Bit #10 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	00_h

Subindex	$0B_h$
Name	Input Source For Bit #11 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	00_h

Subindex	$0C_h$
Name	Input Source For Bit #12 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	00_h

Subindex	$0D_h$

Name	Input Source For Bit #13 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	00_h

Subindex	$0E_h$
Name	Input Source For Bit #14 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	00_h

Subindex	$0F_h$
Name	Input Source For Bit #15 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	00_h

Subindex	10_h
Name	Input Source For Bit #16 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	00_h

Subindex	11_h
Name	Input Source For Bit #17 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	$0E_h$

Subindex	12_h
Name	Input Source For Bit #18 In 60FDh
Data type	UNSIGNED8
Access	read / write

PDO mapping	TX-PDO
Allowed values	
Preset value	10_h

Subindex	13_h
Name	Input Source For Bit #19 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	$0F_h$

Subindex	14_h
Name	Input Source For Bit #20 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	12_h

Subindex	15_h
Name	Input Source For Bit #21 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	11_h

Subindex	16_h
Name	Input Source For Bit #22 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	14_h

Subindex	17_h
Name	Input Source For Bit #23 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	13_h

Subindex	18_h
Name	Input Source For Bit #24 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	16_h

Subindex	19_h
Name	Input Source For Bit #25 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	15_h

Subindex	$1A_h$
Name	Input Source For Bit #26 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	17_h

Subindex	$1B_h$
Name	Input Source For Bit #27 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	04_h

Subindex	$1C_h$
Name	Input Source For Bit #28 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	18_h

Subindex	$1D_h$
Name	Input Source For Bit #29 In 60FDh
Data type	UNSIGNED8

Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	00_h

Subindex	$1E_h$
Name	Input Source For Bit #30 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	00_h

Subindex	$1F_h$
Name	Input Source For Bit #31 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	00_h

Subindex	20_h
Name	Input Source For Bit #32 In 60FDh
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	00_h

3243h Digital Input Homing Capture

Function

With this object, the current position can be noted automatically if a level change occurs at the digital input that is used for the home switch.

NOTICE

Do not use this function in combination with a homing operation. The homing operation cannot otherwise be successfully completed.

Object description

Index	3243_h
Object name	Digital Input Homing Capture

Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1738-B501312
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	04_h

Subindex	01_h
Name	Control
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h

Subindex	02_h
Name	Capture Count
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h

Subindex	03_h
Name	Value
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h

Subindex	04_h
Name	Sensor Raw Value
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h

Description

- Subindex 01_h: This is used to select the type of level change:
  - Deactivate function: Value "0"
  - With rising edge: Value "1"
  - With falling edge: Value "2"
  - Both edges: Value "3"
- Subindex 02_h: Specifies the number of the noted level changes since the time the function was started; is reset to 0 if subindex 01_h is set to 1,2 or 3
- Subindex 03_h: Encoder position of the level change (in absolute user units from 6064_h)
- Subindex 04_h: Encoder position of the level change

3250h Digital Outputs Control

Function

This object can be used to control the digital outputs as described in chapter " Digital inputs and outputs" .

The following applies for all subindices:

- Bits 0 to 15 control the special functions.
- Bits 16 to 31 control the level of the outputs.

Object description

Index	3250_h
Object name	Digital Outputs Control
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1426: Subindex 01_h: "Name" entry changed from "Special Function Disable" to "Special Function Enable" Firmware version FIR-v1446: "Name" entry changed from "Special Function Enable" to "No Function". Firmware version FIR-v1512: The number of entries was changed from 6 to 9. Firmware version FIR-v2039: Subindex 09 added

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	09_h

Subindex	01_h
Name	No Function
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h

Subindex	02_h
Name	Function Inverted
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h

Subindex	03_h
Name	Force Enable
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h

Subindex	04_h
Name	Force Value
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h

Subindex	05_h

Name	Raw Value
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h

Subindex	06_h
Name	Reserved1
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h

Subindex	07_h
Name	Reserved2
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h

Subindex	08_h
Name	Routing Enable
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001_h

Subindex	09_h
Name	Enable Mask [Bit0=StatusLed, Bit1=ErrorLed]
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	$FFFFFFFF_h$

Description

The subindices have the following function:

- 01_h: No function.
- 02_h: This subindex is used to invert the logic (from normally closed logic to normally open logic).

- 03_h: This subindex is used to force the output value if the bit has the value "1". The level of the output is defined in subindex 4_h.
- 04_h: This subindex is used to define the level to be applied to the output. The value "0" returns a logical low level at the digital output; the value "1", on the other hand, returns a logical high level.
- 05_h: The bit combination applied to the outputs is stored in this subindex.
- 08_h: If the subindex is set to "1", *Output Routing* is activated.

NOTICE

Entries 3250_h:01_h to 3250:04_h then have **no** function until *Output Routing* is again switched off.

- 09_h: For switching control of the Power LED on/off. If bit 0 is set to "1", the green LED is activated (flashes in normal operation). If bit 1 is set to "1", the red LED is activated (flashes in case of an error). If the bit is set to "0", the respective LED remains off.

3252h Digital Output Routing

Function

This object assigns a signal source to an output; this signal source can be controlled with $60FE_h$. You can find details in chapter *Output Routing*.

Object description

Index	3252_h
Object name	Digital Output Routing
Object Code	ARRAY
Data type	UNSIGNED16
Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1540
Change history	Firmware version FIR-v2213-B1030801: entry "Name" changed from "Control Bit Of 60FEh:1h And Source For Output #1" to "Control Bit Of 60FEh:1h And Source For Output #1". Firmware version FIR-v2213-B1030801: entry "Name" changed from "Control Bit Of 60FEh:1h And Source For Output #2" to "Control Bit Of 60FEh:1h And Source For Output #2". Firmware version FIR-v2213-B1030801: entry "Name" changed from "Control Bit Of 60FEh:1h And Source For Output #3" to "Control Bit Of 60FEh:1h And Source For Output #3". Firmware version FIR-v2213-B1030801: entry "Name" changed from "Control Bit Of 60FEh:1h And Source For Output #4" to "Control Bit Of 60FEh:1h And Source For Output #4". Firmware version FIR-v2213-B1030801: entry "Name" changed from "Control Bit Of 60FEh:1h And Source For Output #5" to "Control Bit Of 60FEh:1h And Source For Output #5". Firmware version FIR-v2213-B1030801: entry "Name" changed from "Control Bit Of 60FEh:1h And Source For Output #6" to "Control Bit Of 60FEh:1h And Source For Output #6".

Firmware version FIR-v2213-B1030801: entry "Name" changed from "Control Bit Of 60FEh:1h And Source For Output #7" to "Control Bit Of 60FEh:1h And Source For Output #7".

Firmware version FIR-v2213-B1030801: entry "Name" changed from "Control Bit Of 60FEh:1h And Source For Output #8" to "Control Bit Of 60FEh:1h And Source For Output #8".

Firmware version FIR-v2213-B1030801: entry "Name" changed from "Control Bit Of 60FEh:1h And Source For Output #9" to "Control Bit Of 60FEh:1h And Source For Output #9".

Firmware version FIR-v2213-B1030801: entry "Name" changed from "Control Bit Of 60FEh:1h And Source For Output #10" to "Control Bit Of 60FEh:1h And Source For Output #10".

Firmware version FIR-v2213-B1030801: entry "Name" changed from "Control Bit Of 60FEh:1h And Source For Output #11" to "Control Bit Of 60FEh:1h And Source For Output #11".

Firmware version FIR-v2213-B1030801: entry "Name" changed from "Control Bit Of 60FEh:1h And Source For Output #12" to "Control Bit Of 60FEh:1h And Source For Output #12".

Firmware version FIR-v2213-B1030801: entry "Name" changed from "Control Bit Of 60FEh:1h And Source For Output #13" to "Control Bit Of 60FEh:1h And Source For Output #13".

Firmware version FIR-v2213-B1030801: entry "Name" changed from "Control Bit Of 60FEh:1h And Source For Output #14" to "Control Bit Of 60FEh:1h And Source For Output #14".

Firmware version FIR-v2213-B1030801: entry "Name" changed from "Control Bit Of 60FEh:1h And Source For Output #15" to "Control Bit Of 60FEh:1h And Source For Output #15".

Firmware version FIR-v2213-B1030801: entry "Name" changed from "Control Bit Of 60FEh:1h And Source For Output #16" to "Control Bit Of 60FEh:1h And Source For Output #16".

Firmware version FIR-v2213-B1030801: entry "Name" changed from "Control Bit Of 60FEh:1h And Source For Output #17" to "Control Bit Of 60FEh:1h And Source For Output #17".

Firmware version FIR-v2213-B1030801: entry "Name" changed from "Control Bit Of 60FEh:1h And Source For Output #18" to "Control Bit Of 60FEh:1h And Source For Output #18".

Firmware version FIR-v2213-B1030801: entry "Name" changed from "Control Bit Of 60FEh:1h And Source For Output #19" to "Control Bit Of 60FEh:1h And Source For Output #19".

Firmware version FIR-v2213-B1030801: entry "Name" changed from "Control Bit Of 60FEh:1h And Source For Output #20" to "Control Bit Of 60FEh:1h And Source For Output #20".

Firmware version FIR-v2213-B1030801: entry "Name" changed from "Control Bit Of 60FEh:1h And Source For Output #21" to "Control Bit Of 60FEh:1h And Source For Output #21".

Firmware version FIR-v2213-B1030801: entry "Name" changed from "Control Bit Of 60FEh:1h And Source For Output #22" to "Control Bit Of 60FEh:1h And Source For Output #22".

Firmware version FIR-v2213-B1030801: entry "Name" changed from "Control Bit Of 60FEh:1h And Source For Output #23" to "Control Bit Of 60FEh:1h And Source For Output #23".

	Firmware version FIR-v2213-B1030801: entry "Name" changed from "Control Bit Of 60FEh:1h And Source For Output #24" to "Control Bit Of 60FEh:1h And Source For Output #24".

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	18_h

Subindex	01_h
Name	Control Bit Of 60FEh:1h And Source For Output #1
Data type	UNSIGNED16
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	$FFFF_h$

Subindex	02_h
Name	Control Bit Of 60FEh:1h And Source For Output #2
Data type	UNSIGNED16
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	$FFFF_h$

Subindex	03_h
Name	Control Bit Of 60FEh:1h And Source For Output #3
Data type	UNSIGNED16
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	$FFFF_h$

Subindex	04_h
Name	Control Bit Of 60FEh:1h And Source For Output #4
Data type	UNSIGNED16
Access	read / write
PDO mapping	TX-PDO
Allowed values	

Preset value	001A$_h$

Subindex	05$_h$
Name	Control Bit Of 60FEh:1h And Source For Output #5
Data type	UNSIGNED16
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	FFFF$_h$

Subindex	06$_h$
Name	Control Bit Of 60FEh:1h And Source For Output #6
Data type	UNSIGNED16
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	001C$_h$

Subindex	07$_h$
Name	Control Bit Of 60FEh:1h And Source For Output #7
Data type	UNSIGNED16
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	FFFF$_h$

Subindex	08$_h$
Name	Control Bit Of 60FEh:1h And Source For Output #8
Data type	UNSIGNED16
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	FFFF$_h$

Subindex	09$_h$
Name	Control Bit Of 60FEh:1h And Source For Output #9
Data type	UNSIGNED16
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	FFFF$_h$

Subindex	0A$_h$

Name	Control Bit Of 60FEh:1h And Source For Output #10
Data type	UNSIGNED16
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	$FFFF_h$

Subindex	$0B_h$
Name	Control Bit Of 60FEh:1h And Source For Output #11
Data type	UNSIGNED16
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	$FFFF_h$

Subindex	$0C_h$
Name	Control Bit Of 60FEh:1h And Source For Output #12
Data type	UNSIGNED16
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	$FFFF_h$

Subindex	$0D_h$
Name	Control Bit Of 60FEh:1h And Source For Output #13
Data type	UNSIGNED16
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	$001D_h$

Subindex	$0E_h$
Name	Control Bit Of 60FEh:1h And Source For Output #14
Data type	UNSIGNED16
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	0010_h

Subindex	$0F_h$
Name	Control Bit Of 60FEh:1h And Source For Output #15
Data type	UNSIGNED16
Access	read / write

PDO mapping	TX-PDO
Allowed values	
Preset value	0012_h

Subindex	10_h
Name	Control Bit Of 60FEh:1h And Source For Output #16
Data type	UNSIGNED16
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	0011_h

Subindex	11_h
Name	Control Bit Of 60FEh:1h And Source For Output #17
Data type	UNSIGNED16
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	0014_h

Subindex	12_h
Name	Control Bit Of 60FEh:1h And Source For Output #18
Data type	UNSIGNED16
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	0013_h

Subindex	13_h
Name	Control Bit Of 60FEh:1h And Source For Output #19
Data type	UNSIGNED16
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	0016_h

Subindex	14_h
Name	Control Bit Of 60FEh:1h And Source For Output #20
Data type	UNSIGNED16
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	0015_h

Subindex	15_h
Name	Control Bit Of 60FEh:1h And Source For Output #21
Data type	UNSIGNED16
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	0018_h

Subindex	16_h
Name	Control Bit Of 60FEh:1h And Source For Output #22
Data type	UNSIGNED16
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	1080_h

Subindex	17_h
Name	Control Bit Of 60FEh:1h And Source For Output #23
Data type	UNSIGNED16
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	0019_h

Subindex	18_h
Name	Control Bit Of 60FEh:1h And Source For Output #24
Data type	UNSIGNED16
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	$001B_h$

3260h Pwm Output 0

Function

Use this object to configure the first PWM output. You must define the output as PWM output using *Output Routing*.

Object description

Index	3260_h
Object name	Pwm Output 0
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application

Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1939-B682906
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Pwm Frequency
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00002710_h

Subindex	02_h
Name	Pwm Duty Cycle
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Description

The subindices have the following function:

- 01_h: Frequency of the PWM signal in hertz. 50…20000
- 02_h: Duty cycle of the PWM signal: 0…100

3261h Pwm Output 1

Function

Use this object to configure the second PWM output. You must define the output as PWM output using *Output Routing*.

Object description

Index	3261_h
Object name	Pwm Output 1
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1939-B682906
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Pwm Frequency
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00002710_h

Subindex	02_h
Name	Pwm Duty Cycle
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Description

The subindices have the following function:

- 01_h: Frequency of the PWM signal in hertz. 50…20000
- 02_h: Duty cycle of the PWM signal: 0…100

3273h Generic SPI Hardware Configuration

Function

See chapter Generic SPI.

Object description

Index	3273_h
Object name	Generic SPI Hardware Configuration
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v2213-B1029645
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	01_h

Subindex	01_h
Name	Hardware Feature Control
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

3274h Generic SPI Mosi Data

Function

See chapter Generic SPI.

Object description

Index	3274_h

Object name	Generic SPI Mosi Data
Object Code	ARRAY
Data type	UNSIGNED8
Savable	yes, category: application
Access	read / write
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v2213-B1029645
Change history	

Value description

Subindex	00_h
Name	Length Of SPI Message To Be Sent
Data type	UNSIGNED8
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00_h

Subindex	01_h - $1F_h$
Name	Generic SPI Mosi Data Byte #1 - #31
Data type	UNSIGNED8
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00_h

3275h Generic SPI Miso Data

Function

See chapter Generic SPI.

Object description

Index	3275_h
Object name	Generic SPI Miso Data
Object Code	ARRAY
Data type	UNSIGNED8
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	

Firmware version	FIR-v2213-B1029645
Change history	

Value description

Subindex	00_h
Name	Length Of Received SPI Message
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	00_h

Subindex	01_h - $1F_h$
Name	Generic SPI Miso Data Byte #1 - #31
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	00_h

3320h Analog Input Values

Function

This object displays the instantaneous values of the analog inputs in user-defined units.

Object description

Index	3320_h
Object name	Analog Input Values
Object Code	ARRAY
Data type	INTEGER32
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1426
Change history	

Value description

Subindex	00_h
Name	Number Of Analog Input Values
Data type	UNSIGNED8

Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h - 02_h
Name	Analog Input #1 - #2
Data type	INTEGER32
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h

Description

The user-defined units are made up of offset (3321_h) and scaling value (3322_h/ 3323_h). If both are still set to the default values, the value in 3320_h is specified in the *ADC Digits* unit.

Formula for converting from digits to the respective unit:

- Current input (if configurable): x digits * 20 mA / 1023 digits

The following applies for the sub-entries:

- Subindex 00_h: Number of analog inputs
- Subindex 01_h: Analog value 1
- Subindex 02_h: Analog value 2 (if present)

3321h Analog Input Offsets

Function

Offset that is added to the read analog value (3220_h) before scaling (multiplier from object 3322 and divisor from object 3323_h).

Object description

Index	3321_h
Object name	Analog Input Offsets
Object Code	ARRAY
Data type	INTEGER16
Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v2139-B1022383
Change history	

Value description

Subindex	00_h
Name	Number Of Analog Input Offsets
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h - 02_h
Name	Analog Input #1 - #2
Data type	INTEGER16
Access	read / write
PDO mapping	no
Allowed values	
Preset value	0000_h

3322h Analog Input Numerators

Function

Value by which the read analog value (3220_h, 3321_h) is multiplied before it is written in object 3320_h.

Object description

Index	3322_h
Object name	Analog Input Numerators
Object Code	ARRAY
Data type	INTEGER16
Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1426
Change history	

Value description

Subindex	00_h
Name	Number Of Analog Input Numerators
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	

Preset value	02_h

Subindex	01_h - 02_h
Name	Analog Input #1 - #2
Data type	INTEGER16
Access	read / write
PDO mapping	no
Allowed values	
Preset value	0001_h

Description

The subindices contain:

- Subindex 01_h: Multiplier for analog input 1
- Subindex 02_h: Multiplier for analog input 2 (if present)

3323h Analog Input Denominators

Function

Value by which the read analog value (3220_h+ 3321_h) is divided before it is written in object 3320_h.

Object description

Index	3323_h
Object name	Analog Input Denominators
Object Code	ARRAY
Data type	INTEGER16
Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1926-B648637
Change history	

Value description

Subindex	00_h
Name	Number Of Analog Input Denominators
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h - 02_h
Name	Analog Input #1 - #2
Data type	INTEGER16
Access	read / write
PDO mapping	no
Allowed values	
Preset value	0001_h

Description

The subindices contain:

- Subindex 01_h: Divisor for analog input 1
- Subindex 02_h: Divisor for analog input 2 (if present)

3380h Feedback Sensorless

Function

Contains measurement and configuration values that are necessary for the sensorless control and field weakening in Closed-Loop.

Object description

Index	3380_h
Object name	Feedback Sensorless
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: tuning
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	
Firmware version	FIR-v2013-B726332
Change history	Firmware version FIR-v2013-B726332: The number of entries was changed from 7 to 6.

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	05_h

Subindex	01_h

Name	Resistance [Ohm]
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h

Subindex	02_h
Name	Inductance [H]
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h

Subindex	03_h
Name	Magnetic Flux [Vs]
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h

Subindex	04_h
Name	Switch On Speed [rpm]
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000078_h

Subindex	05_h
Name	Switch Off Speed [rpm]
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000064_h

Description

The subindices have the following function:

- 01_h: Winding resistance. Float value, shown here as UNSIGNED32. Is determined by Auto setup.
- 02_h: Winding inductance. Float value, shown here as UNSIGNED32. Is determined by Auto setup.

- 03_h: Interlinking flux. Float value, shown here as UNSIGNED32. Is determined by Auto setup.
- 04_h: Switch-on speed in RPM. *Closed loop* (*sensorless*) is activated above this speed if no sensors were detected by Auto setup.
- 05_h: Switch-off speed in RPM. *Closed loop* (*sensorless*) is deactivated below this speed if no sensors were detected by Auto setup.

3390h Feedback Hall

Function

Contains configuration values for the Hall sensors. The values are determined by the Auto setup.

Object description

Index	3390_h
Object name	Feedback Hall
Object Code	ARRAY
Data type	UNSIGNED16
Savable	yes, category: tuning
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	
Firmware version	FIR-v1748-B531667
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	$0C_h$

Subindex	01_h
Name	1st Alignment
Data type	UNSIGNED16
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000_h

Subindex	02_h
Name	2nd Alignment
Data type	UNSIGNED16

Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000_h

Subindex	03_h
Name	3rd Alignment
Data type	UNSIGNED16
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000_h

Subindex	04_h
Name	4th Alignment
Data type	UNSIGNED16
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000_h

Subindex	05_h
Name	5th Alignment
Data type	UNSIGNED16
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000_h

Subindex	06_h
Name	6th Alignment
Data type	UNSIGNED16
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000_h

Subindex	07_h
Name	7th Alignment
Data type	UNSIGNED16
Access	read / write
PDO mapping	RX-PDO
Allowed values	

Preset value	0000_h

Subindex	08_h
Name	8th Alignment
Data type	UNSIGNED16
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000_h

Subindex	09_h
Name	9th Alignment
Data type	UNSIGNED16
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000_h

Subindex	$0A_h$
Name	10th Alignment
Data type	UNSIGNED16
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000_h

Subindex	$0B_h$
Name	11th Alignment
Data type	UNSIGNED16
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000_h

Subindex	$0C_h$
Name	12th Alignment
Data type	UNSIGNED16
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000_h

33A0h Feedback Incremental A/B/I 1

Function

Contains configuration values for the first incremental encoder. The values are determined by the Auto setup.

Object description

Index	$33A0_h$
Object name	Feedback Incremental A/B/I 1
Object Code	ARRAY
Data type	UNSIGNED16
Savable	yes, category: tuning
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	
Firmware version	FIR-v1738-B501312
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Configuration
Data type	UNSIGNED16
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000_h

Subindex	02_h
Name	Alignment
Data type	UNSIGNED16
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000_h

Description

The subindices have the following function:

- 01_h (Configuration): The following bits have a meaning:
  - Bit 0: Value = "0": The encoder does not have an index. Value = "1": Encoder index exists and is to be used.
- 02_h (Alignment): This value specifies the offset between the index of the encoder and the rotor's magnets. The exact determination is possible via auto setup. The presence of this value is necessary for *closed loop* mode with encoder.

33B0h Feedback SSI 1

Function

Contains configuration values for the external SSI encoder.

Object description

Index	$33B0_h$
Object name	Feedback SSI 1
Object Code	RECORD
Data type	SSI ENCODER
Savable	yes, category: tuning
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1939-B682906
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	$0C_h$

Subindex	01_h
Name	Configuration
Data type	UNSIGNED16
Access	read / write
PDO mapping	no
Allowed values	
Preset value	0000_h

Subindex	02_h
Name	Alignment
Data type	UNSIGNED16
Access	read / write
PDO mapping	no
Allowed values	
Preset value	0000_h

Subindex	03_h
Name	Home Position Low
Data type	INTEGER32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	04_h
Name	Home Position High
Data type	INTEGER32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	05_h
Name	Number Of Bits For Transfer
Data type	UNSIGNED8
Access	read / write
PDO mapping	no
Allowed values	
Preset value	15_h

Subindex	06_h
Name	Baud Rate
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	$00280DE8_h$

Subindex	07_h
Name	Position Bitmask Low
Data type	UNSIGNED32

Access	read / write
PDO mapping	no
Allowed values	
Preset value	001FFFFE$_h$

Subindex	08$_h$
Name	Position Bitmask High
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000$_h$

Subindex	09$_h$
Name	Status Bitmask Low
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000$_h$

Subindex	0A$_h$
Name	Status Bitmask High
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000$_h$

Subindex	0B$_h$
Name	Status Value Low
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000$_h$

Subindex	0C$_h$
Name	Status Value High
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	

Preset value	00000000_h

Description

The subindices have the following function:

- 01_h (Configuration):
  - Bit 0: Value = "0": Alignment has not yet been determined or is not to be used. Value = "1": Alignment exists and is to be used.
- 02_h (Alignment): This value specifies the offset between the zero position of the encoder and the rotor's magnets.
  The exact determination is only possible via auto setup. The presence of this value is necessary for *closed-loop* mode with encoder.
- 03_h (Home Position Low) and 04_h (Home Position High): The absolute encoder position after a homing has been completed is entered in these subindices.
- 05_h (Number Of Bits For Transfer): Number of bits in a message (encoder data). Maximum 64 bits.
- 06_h (Baud Rate): Baud rate of the interface in hertz. The following frequencies are supported: 21 MHz, 10.5 MHz, 5.25 MHz, 2.625 MHz, 1.3125 MHz, 656.25 KHz, 328.125 KHz, 164.0625 KHz. If the values are different, the valid frequency with the smallest difference is selected.
- 07_h (Position Bitmask Low) and 08_h (Position Bitmask High): In these subindices, you enter a bitmask that determines which bits of the encoder data contain the position data (see following instructions).
- 09_h (Status Bitmask Low) and $0A_h$ (Status Bitmask High): In these subindices, you enter a bitmask that determines which bits of the encoder data contain the status information (see following instructions).
- $0B_h$ (Status Value Low) and $0C_h$ (Status Value High): In these subindices, you enter a bitmask that determines which value the status information bits (subindices 09_h and $0A_h$) must have (see following instructions). A different value at this point of the encoder data is interpreted as an error by the controller.

To set the configuration according to your encoder:

1. Set the baud rate in subindex 06_h and the number of bits in subindex 05_h according to the encoder data sheet.
2. Define which bits the position data should include and set subindices 07_h and 08_h to the corresponding value.
3. Define which bits the status information (e. g., status, error, etc.) should include and set subindices 09_h and $0A_h$ to the corresponding value.
4. Define which value, "0" or "1", the status information bits must have and set the corresponding bits in subindices 09_h and $0A_h$ to the value.
5. Store the object by writing the value "65766173_h" in 1010_h:06_h and restart the controller.

Example

The encoder sends the data in a 32-bit message. Bits 4…23 contain the position. The status information is divided into the following bits:

- Bits 0…2 are status bits that must always have the value "0"
- Bit 3 is the error bit that has the value "0" if an error has occurred
- Bit 31 signals the start of the message and always has the value "1"

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L								POS	POS	POS	POS	POS	POS	POS	POS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POS	POS	POS	POS	POS	POS	POS	POS	POS	POS	POS	POS	E	S	S	S

You must enter the following values in the subindices:

- 05_h (Number Of Bits For Transfer): 20_h

- 07_h(Position Bitmask Low) $00FFFFF0_h$
- 09_h (Status Bitmask Low): 8000 $000F_h$
- $0B_h$ (Status Value Low): 8000 $000F8_h$

Subindices 08_h, $0A_h$ and $0C_h$, which would contain the most-significant 32 bits of a 64-bit message, have the value "0".

33B1h Feedback SSI 2

Function

Contains configuration values for the second external SSI encoder.

Object description

Index	$33B1_h$
Object name	Feedback SSI 2
Object Code	RECORD
Data type	SSI ENCODER
Savable	yes, category: tuning
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v2139-B1019507
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	$0C_h$

Subindex	01_h
Name	Configuration
Data type	UNSIGNED16
Access	read / write
PDO mapping	no
Allowed values	
Preset value	0000_h

Subindex	02_h
Name	Alignment

Data type	UNSIGNED16
Access	read / write
PDO mapping	no
Allowed values	
Preset value	0000_h

Subindex	03_h
Name	Home Position Low
Data type	INTEGER32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	04_h
Name	Home Position High
Data type	INTEGER32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	05_h
Name	Number Of Bits For Transfer
Data type	UNSIGNED8
Access	read / write
PDO mapping	no
Allowed values	
Preset value	15_h

Subindex	06_h
Name	Baud Rate
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	$00280DE8_h$

Subindex	07_h
Name	Position Bitmask Low
Data type	UNSIGNED32
Access	read / write
PDO mapping	no

Allowed values	
Preset value	001FFFFE$_h$

Subindex	08$_h$
Name	Position Bitmask High
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000$_h$

Subindex	09$_h$
Name	Status Bitmask Low
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000$_h$

Subindex	0A$_h$
Name	Status Bitmask High
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000$_h$

Subindex	0B$_h$
Name	Status Value Low
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000$_h$

Subindex	0C$_h$
Name	Status Value High
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000$_h$

Description

The subindices have the following function:

- 01_h (Configuration):
  - Bit 0: Value = "0": Alignment has not yet been determined or is not to be used. Value = "1": Alignment exists and is to be used.
- 02_h (Alignment): This value specifies the offset between the zero position of the encoder and the rotor's magnets.
  The exact determination is only possible via auto setup. The presence of this value is necessary for *closed-loop* mode with encoder.
- 03_h (Home Position Low) and 04_h (Home Position High): The absolute encoder position after a homing has been completed is entered in these subindices.
- 05_h (Number Of Bits For Transfer): Number of bits in a message (encoder data). Maximum 64 bits.
- 06_h (Baud Rate): Baud rate of the interface in hertz. The following frequencies are supported: 21 MHz, 10.5 MHz, 5.25 MHz, 2.625 MHz, 1.3125 MHz, 656.25 KHz, 328.125 KHz, 164.0625 KHz. If the values are different, the valid frequency with the smallest difference is selected.
- 07_h (Position Bitmask Low) and 08_h (Position Bitmask High): In these subindices, you enter a bitmask that determines which bits of the encoder data contain the position data (see following instructions).
- 09_h (Status Bitmask Low) and $0A_h$ (Status Bitmask High): In these subindices, you enter a bitmask that determines which bits of the encoder data contain the status information (see following instructions).
- $0B_h$ (Status Value Low) and $0C_h$ (Status Value High): In these subindices, you enter a bitmask that determines which value the status information bits (subindices 09_h and $0A_h$) must have (see following instructions). A different value at this point of the encoder data is interpreted as an error by the controller.

To set the configuration according to your encoder:

1. Set the baud rate in subindex 06_h and the number of bits in subindex 05_h according to the encoder data sheet.
2. Define which bits the position data should include and set subindices 07_h and 08_h to the corresponding value.
3. Define which bits the status information (e. g., status, error, etc.) should include and set subindices 09_h and $0A_h$ to the corresponding value.
4. Define which value, "0" or "1", the status information bits must have and set the corresponding bits in subindices 09_h and $0A_h$ to the value.
5. Store the object by writing the value "65766173_h" in 1010_h:06_h and restart the controller.

Example

The encoder sends the data in a 32-bit message. Bits 4…23 contain the position. The status information is divided into the following bits:

- Bits 0…2 are status bits that must always have the value "0"
- Bit 3 is the error bit that has the value "0" if an error has occurred
- Bit 31 signals the start of the message and always has the value "1"

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L								POS	POS	POS	POS	POS	POS	POS	POS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
POS	POS	POS	POS	POS	POS	POS	POS	POS	POS	POS	POS	E	S	S	S

You must enter the following values in the subindices:

- 05_h (Number Of Bits For Transfer): 20_h
- 07_h(Position Bitmask Low) $00FFFFF0_h$
- 09_h (Status Bitmask Low): 8000 $000F_h$
- $0B_h$ (Status Value Low): 8000 $000F8_h$

Subindices 08_h, $0A_h$ and $0C_h$, which would contain the most-significant 32 bits of a 64-bit message, have the value "0".

3502h MODBUS Rx PDO Mapping

Function

The objects for RX mapping can be written in this object.

NOTICE

To be able to change the mapping, you must first deactivate it by setting subindex 0_h to "0".

After writing the objects to the respective subindices, enter the number of mapped objects in subindex 0_h.

Object description

Index	3502_h
Object name	MODBUS Rx PDO Mapping
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: communication
Access	read / write
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1748-B538662
Change history	Firmware version FIR-v1738-B505321: "Object Name" entry changed from "MODBUS Rx PDO-Mapping" to "MODBUS Rx PDO Mapping".

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read / write
PDO mapping	no
Allowed values	
Preset value	07_h

Subindex	01_h
Name	1st Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no

Allowed values	
Preset value	60400010_h

Subindex	02_h
Name	2nd Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00050008_h

Subindex	03_h
Name	3rd Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	60600008_h

Subindex	04_h
Name	4th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	$607A0020_h$

Subindex	05_h
Name	5th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	60810020_h

Subindex	06_h
Name	6th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	60420010_h

Subindex	07_h
Name	7th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	$60FE0120_h$

Subindex	08_h
Name	8th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	09_h
Name	9th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	$0A_h$
Name	10th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	$0B_h$
Name	11th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	$0C_h$
Name	12th Object To Be Mapped
Data type	UNSIGNED32

Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	$0D_h$
Name	13th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	$0E_h$
Name	14th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	$0F_h$
Name	15th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	10_h
Name	16th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

3602h MODBUS Tx PDO Mapping

Function

The objects for TX mapping can be written in this object.

NOTICE

To be able to change the mapping, you must first deactivate it by setting subindex 0_h to "0".

After writing the objects to the respective subindices, enter the number of mapped objects in subindex 0_h.

Object description

Index	3602_h
Object name	MODBUS Tx PDO Mapping
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: communication
Access	read / write
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1748-B538662
Change history	Firmware version FIR-v1738-B505321: "Object Name" entry changed from "MODBUS Tx PDO-Mapping" to "MODBUS Tx PDO Mapping".

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read / write
PDO mapping	no
Allowed values	
Preset value	06_h

Subindex	01_h
Name	1st Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	60410010_h

Subindex	02_h
Name	2nd Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no

Allowed values	
Preset value	00050008$_h$

Subindex	03$_h$
Name	3rd Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	60610008$_h$

Subindex	04$_h$
Name	4th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	60640020$_h$

Subindex	05$_h$
Name	5th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	60440010$_h$

Subindex	06$_h$
Name	6th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	60FD0020$_h$

Subindex	07$_h$
Name	7th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000$_h$

Subindex	08_h
Name	8th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	09_h
Name	9th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	$0A_h$
Name	10th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	$0B_h$
Name	11th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	$0C_h$
Name	12th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	$0D_h$
Name	13th Object To Be Mapped
Data type	UNSIGNED32

Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	$0E_h$
Name	14th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	$0F_h$
Name	15th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	10_h
Name	16th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

3700h Deviation Error Option Code

Function

The object contains the action that is to be executed if a following or slippage error is triggered.

Object description

Index	3700_h
Object name	Deviation Error Option Code
Object Code	VARIABLE
Data type	INTEGER16
Savable	yes, category: application
Access	read / write
PDO mapping	no
Allowed values	

Preset value	$FFFF_h$
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1738-B501312: "Object Name" entry changed from "Following Error Option Code" to "Deviation Error Option Code".

Description

Value	Description
-32768 … -2	Reserved
-1	no reaction
0	Switch off driver without deceleration ramp; drive function blocked – motor can turn freely
1	Braking with *slow down ramp* (braking deceleration depending on operating mode)
2	Braking with *quick stop ramp* (6085_h)
3 … 32767	reserved

3701h Limit Switch Error Option Code

Function

If a limit switch is passed over, bit 7 (*Warning*) is set in 6041_h (*statusword*) and the action that is stored in this object executed. See chapter Limitation of the range of motion.

Object description

Index	3701_h
Object name	Limit Switch Error Option Code
Object Code	VARIABLE
Data type	INTEGER16
Savable	yes, category: application
Access	read / write
PDO mapping	no
Allowed values	
Preset value	$FFFF_h$
Firmware version	FIR-v1748-B538662
Change history	

Description

Value in object 3701_h	Description
-2	No reaction, discard the limit switch position
-1 (factory settings)	No reaction (e. g., to execute a homing operation) except noting the limit switch position
0	Switch off driver without deceleration ramp; drive function blocked – motor can turn freely (*Switch on disabled* state)

Value in object 3701_h	Description
1	Braking with *slow down ramp* (deceleration ramp depending on operating mode) and subsequent state change to *Switch on disabled*
2	Braking with *quick stop ramp* and subsequent state change to *Switch on disabled*
5	Braking with *slow down ramp* (deceleration ramp depending on operating mode) and subsequent state change to *Quick stop active*; control does not switch off and the motor remains energized. You can switch back to the *Operation enabled* state.
6	Braking with *quick stop ramp* and subsequent state change to *Quick Stop Active*; control does not switch off and the motor remains energized. You can switch back to the *Operation enabled* state.

4012h HW Information

Function

This object contains information about the hardware.

Object description

Index	4012_h
Object name	HW Information
Object Code	ARRAY
Data type	UNSIGNED32
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1540
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	01_h

Subindex	01_h
Name	EEPROM Size In Bytes
Data type	UNSIGNED32
Access	read only
PDO mapping	no

Allowed values	
Preset value	00000000_h

Description

Subindex 01: Contains the size of the connected EEPROM in bytes. The value "0" means that no EEPROM is connected.

4013h HW Configuration

Function

This object is used to set certain hardware configurations.

Object description

Index	4013_h
Object name	HW Configuration
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1540
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	01_h

Subindex	01_h
Name	HW Configuration #1
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Description

reserved

4014h Operating Conditions

Function

This object is used to read out the current environment values for the controller.

Object description

Index	4014_h
Object name	Operating Conditions
Object Code	ARRAY
Data type	INTEGER32
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1540
Change history	Firmware version FIR-v1650-B472161: "Access" table entry for subindex 01 changed from "read/write" to "read only". Firmware version FIR-v1650-B472161: "Access" table entry for subindex 02 changed from "read/write" to "read only". Firmware version FIR-v1650-B472161: "Name" entry changed from "Temperature PCB [d?C]" to "Temperature PCB [Celsius * 10]". Firmware version FIR-v1650-B472161: "Access" table entry for subindex 03 changed from "read/write" to "read only". Firmware version FIR-v1738-B501312: The number of entries was changed from 4 to 6.

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	05_h

Subindex	01_h
Name	Voltage UB Power [mV]
Data type	INTEGER32
Access	read only
PDO mapping	TX-PDO

Allowed values	
Preset value	00000000_h

Subindex	02_h
Name	Voltage UB Logic [mV]
Data type	INTEGER32
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h

Subindex	03_h
Name	Temperature PCB [Celsius * 10]
Data type	INTEGER32
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h

Subindex	04_h
Name	Temperature Motor [Celsius * 10]
Data type	INTEGER32
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h

Subindex	05_h
Name	Temperature Microcontroller Chip [Celsius * 10]
Data type	INTEGER32
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h

Description

The subindices contain:

- 01_h: Current voltage supply voltage in [mV]
- 02_h: Current logic voltage in [mV]
- 03_h: Current temperature of the control board in [d°C] (tenths of degree)
- 04_h: Reserves
- 05_h: Reserves

4021h Ballast Configuration

Function

Object description

Index	4021_h
Object name	Ballast Configuration
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: tuning
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v2013-B726332
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	08_h

Subindex	01_h
Name	Settings [Bit0: On/Off, Bit1: Polarity]
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	02_h
Name	UB Power Limit [mV]
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	■ CPB6-1-4: $0000EC54_h$ ■ CPB6-1-S-4: $0000EC54_h$ ■ CPB6-2-4: 00011172_h

- CPB6-2-S-4: 00011172_h
- CPB3-1-4: $0000EC54_h$
- CPB3-2-4: $0000EC54_h$
- CPB15-4: $0000EC54_h$
- CPB15-S-4: $0000EC54_h$

Subindex	03_h
Name	UB Power Hysteresis [mV]
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	$000001F4_h$

Subindex	04_h
Name	Nominal Resistance [mOhm]
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001_h

Subindex	05_h
Name	Long Term Energy Limit [mWs]
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001_h

Subindex	06_h
Name	Long Term Reference Time [ms]
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	07_h
Name	Short Term Energy Limit [mWs]
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	

Preset value	00000000_h

Subindex	08_h
Name	Cooling Power [mW]
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000_h

Description

The subindices have the following function:

- 01_h:
  - Bit 0: Switches the ballast on (value = "1") or off (value = "0")
- 02_h: Response threshold (switch on/off) of the ballast circuit
- 03_h: Hysteresis for the response threshold (switch on/off)

4040h Drive Serial Number

Function

This object contains the serial number of the controller.

Object description

Index	4040_h
Object name	Drive Serial Number
Object Code	VARIABLE
Data type	VISIBLE_STRING
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	0
Firmware version	FIR-v1450
Change history	

4041h Device Id

Function

This object contains the ID of the device.

Object description

Index	4041_h
Object name	Device Id

Object Code	VARIABLE
Data type	OCTET_STRING
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	0
Firmware version	FIR-v1540
Change history	

4042h Bootloader Infos

Object description

Index	4042_h
Object name	Bootloader Infos
Object Code	ARRAY
Data type	UNSIGNED32
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v2013-B726332
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	03_h

Subindex	01_h
Name	Bootloader Version
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	02_h
Name	Bootloader Supported Fieldbus
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h

Subindex	03_h
Name	Bootloader Hw-group
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000_h

Description

The subindices have the following functions:

- 01_h: Version of the boot loader. The 4 most significant bytes contain the main version number; the 4 least significant bytes contain the minor version number. Example for version 4.2: 00040002_h
- 02_h: Fieldbuses supported by the boot loader. The bits have the same function as the bits of object 2101h Fieldbus Module Availability.

603Fh Error Code

Function

This object returns the error code of the last error that occurred.

It corresponds to the lower 16 bits of object 1003_h. For the description of the error codes, refer to object 1003_h.

Object description

Index	$603F_h$
Object name	Error Code
Object Code	VARIABLE
Data type	UNSIGNED16
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	0000_h
Firmware version	FIR-v1426
Change history	

Description

For the meaning of the error, see object 1003_h (Pre-defined Error Field).

If the error is reset by setting bit 7 in 6040h Controlword, this object is also automatically reset to "0".

6040h Controlword

Function

This object controls the CiA 402 Power State Machine.

Object description

Index	6040_h
Object name	Controlword
Object Code	VARIABLE
Data type	UNSIGNED16
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000_h
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1626: "Savable" entry changed from "no" to "yes, category: application".

Description

Parts of the object are, with respect to function, dependent on the currently selected mode.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						OMS	HALT	FR		OMS [3]		EO	QS	EV	SO

SO (Switched On)

Value = "1": Switches to the "Switched on" state

EV (Enable Voltage)

Value = "1": Switches to the "Enable voltage" state

QS (Quick Stop)

Value = "0": Switches to the "Quick stop" state

EO (Enable Operation)

Value = "1": Switches to the "Enable operation" state

OMS (Operation Mode Specific)

Meaning is dependent on the selected operating mode

FR (Fault Reset)

Resets an error or a warning (if possible)

HALT

Value = "1": Triggers a halt; valid in the following modes:

- Profile Position
- Velocity
- Profile Velocity
- Profile Torque
- Interpolated Position Mode

6041h Statusword

Function

This object returns information about the status of the CiA 402 Power State Machine.

Object description

Index	6041_h
Object name	Statusword
Object Code	VARIABLE
Data type	UNSIGNED16
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	0000_h
Firmware version	FIR-v1426
Change history	

Description

Parts of the object are, with respect to function, dependent on the currently selected mode. Refer to the corresponding section in chapter Operating modes.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLA		OMS [2]		ILA	TARG	REM	SYNC	WARN	SOD	QS	VE	FAULT	OE	SO	RTSO

RTSO (Ready To Switch On)
  Value = "1": Controller is in the "Ready to switch on" state

SO (Switched On)
  Value = "1": Controller is in the "Switched on" state

OE (Operation Enabled)
  Value = "1": Controller is in the "Operation enabled" state

FAULT
  Error occurred (see 1003_h)

VE (Voltage Enabled)
  Voltage applied

QS (Quick Stop)
  Value = "0": Controller is in the "Quick stop" state

SOD (Switched On Disabled)
Value = "1": Controller is in the "Switched on disabled" state

WARN (Warning)
Value = "1": Warning

SYNC (synchronization)
Value = "1": Controller is in sync with the fieldbus; value = "0": Controller is not in sync with the fieldbus

REM (Remote)
Remote (value of the bit is always "1")

TARG
Target reached

ILA (Internal Limit Active)
Limit exceeded

OMS (Operation Mode Specific)
Meaning is dependent on the selected operating mode

CLA (Closed Loop Active)
Value = "1": The controller is in the *Operation enabled* state and the Closed-Loop is activated.

Listed in the following table are the bit masks that break down the state of the controller.

Statusword (6041_h)	State
xxxx xxxx x0xx 0000	Not ready to switch on
xxxx xxxx x1xx 0000	Switch on disabled
xxxx xxxx x01x 0001	Ready to switch on
xxxx xxxx x01x 0011	Switched on
xxxx xxxx x01x 0111	Operation enabled
xxxx xxxx x00x 0111	Quick stop active
xxxx xxxx x0xx 1111	Fault reaction active
xxxx xxxx x0xx 1000	Fault

6042h Vl Target Velocity

Function

Specifies the target speed in user-defined units for Velocity mode.

Object description

Index	6042_h
Object name	Vl Target Velocity
Object Code	VARIABLE
Data type	INTEGER16
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO

Allowed values	
Preset value	$00C8_h$
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1626: "Savable" entry changed from "no" to "yes, category: application".

6043h Vl Velocity Demand

Function

Speed specification in user-defined units for the controller in Velocity mode.

Object description

Index	6043_h
Object name	Vl Velocity Demand
Object Code	VARIABLE
Data type	INTEGER16
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	0000_h
Firmware version	FIR-v1426
Change history	

6044h Vl Velocity Actual Value

Function

Specifies the current actual speed in user-defined units in Velocity mode.

Object description

Index	6044_h
Object name	Vl Velocity Actual Value
Object Code	VARIABLE
Data type	INTEGER16
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	0000_h
Firmware version	FIR-v1426
Change history	

6046h Vl Velocity Min Max Amount

Function

This object can be used to set the minimum speed and maximum speed in user-defined units.

Object description

Index	6046_h
Object name	Vl Velocity Min Max Amount
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Firmware version	FIR-v1426
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	MinAmount
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h

Subindex	02_h
Name	MaxAmount
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00007530_h

Description

Subindex 1 contains the minimum speed.

Subindex 2 contains the maximum speed.

If the value of the target speed (object 6042_h) specified here is less than the minimum speed, the minimum speed applies and bit 11 (Internal Limit Reached) in 6041h Statusword$_h$ is set.

A target speed greater than the maximum speed sets the speed to the maximum speed and bit 11 (Internal Limit Reached) in 6041h Statusword$_h$ is set.

6048h Vl Velocity Acceleration

Function

Sets the acceleration ramp in Velocity Mode (see Velocity).

Object description

Index	6048_h
Object name	Vl Velocity Acceleration
Object Code	RECORD
Data type	VELOCITY_ACCELERATION_DECELERATION
Savable	yes, category: application
Firmware version	FIR-v1426
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	DeltaSpeed
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	$000001F4_h$

Subindex	02_h
Name	DeltaTime
Data type	UNSIGNED16
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0001_h

Description

The acceleration is specified as a fraction in user-defined units:

Speed change per change in time.

Subindex 01_h: Contains the change in speed.

Subindex 02_h: Contains the change in time.

6049h Vl Velocity Deceleration

Function

Sets the deceleration (deceleration ramp) in Velocity Mode (see Velocity).

Object description

Index	6049_h
Object name	Vl Velocity Deceleration
Object Code	RECORD
Data type	VELOCITY_ACCELERATION_DECELERATION
Savable	yes, category: application
Firmware version	FIR-v1426
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	DeltaSpeed
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	$000001F4_h$

Subindex	02_h
Name	DeltaTime
Data type	UNSIGNED16
Access	read / write
PDO mapping	RX-PDO
Allowed values	

Preset value	0001_h

Description

The deceleration is specified as a fraction in user-defined units:

Speed change per change in time.

Subindex 01_h: Contains the change in speed.

Subindex 02_h: Contains the change in time.

604Ah VI Velocity Quick Stop

Function

This object defines the deceleration (deceleration ramp) if the Quick Stop state is initiated in velocity mode.

Object description

Index	$604A_h$
Object name	VI Velocity Quick Stop
Object Code	RECORD
Data type	VELOCITY_ACCELERATION_DECELERATION
Savable	yes, category: application
Firmware version	FIR-v1426
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	DeltaSpeed
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00001388_h

Subindex	02_h
Name	DeltaTime
Data type	UNSIGNED16

Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0001_h

Description

The deceleration is specified as a fraction in user-defined units:

Speed change per change in time.

Subindex 01_h: Contains the change in speed.

Subindex 02_h: Contains the change in time.

604Ch Vl Dimension Factor

Function

The unit for speed values is defined here for the objects associated with velocity mode.

Object description

Index	$604C_h$
Object name	Vl Dimension Factor
Object Code	ARRAY
Data type	INTEGER32
Savable	yes, category: application
Firmware version	FIR-v1426
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Vl Dimension Factor Numerator
Data type	INTEGER32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001_h

Subindex	02_h
Name	Vl Dimension Factor Denominator
Data type	INTEGER32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001_h

Description

Subindex 1 contains the numerator (multiplier) and subindex 2 contains the denominator (divisor) with which the internal speed values are converted to revolutions per minute. If, for example, subindex 1 is set to the value "60" and subindex 2 is set to the value "1", the speed is specified in revolutions per second (60 revolutions per 1 minute).

605Ah Quick Stop Option Code

Function

The object contains the action that is to be executed on a transition of the CiA 402 Power State Machine to the *Quick Stop active* state.

Object description

Index	$605A_h$
Object name	Quick Stop Option Code
Object Code	VARIABLE
Data type	INTEGER16
Savable	yes, category: application
Access	read / write
PDO mapping	no
Allowed values	
Preset value	0002_h
Firmware version	FIR-v1426
Change history	

Description

Value in object $605A_h$	Description
0	Immediate stop with subsequent state change to *Switch on disabled*
1	Braking with *slow down ramp* (deceleration ramp depending on operating mode) and subsequent state change to *Switch on disabled*
2	Braking with *quick stop ramp* (6085_h) and subsequent state change to *Switch on disabled*
5	Braking with *slow down ramp* (deceleration ramp depending on operating mode) and subsequent state change to *Quick stop active*; control does not switch off and the motor remains energized. You can switch back to the *Operation enabled* state.
6	Braking with *quick stop ramp* (6085_h) and subsequent state change to *Quick Stop Active*; control does not switch off and the motor

Value in object $605A_h$	Description
	remains energized. You can switch back to the *Operation enabled* state.

605Bh Shutdown Option Code

Function

This object contains the action that is to be executed on a transition of the CiA 402 Power State Machine from the *Operation enabled* state to the *Ready to switch on* state.

Object description

Index	$605B_h$
Object name	Shutdown Option Code
Object Code	VARIABLE
Data type	INTEGER16
Savable	yes, category: application
Access	read / write
PDO mapping	no
Allowed values	
Preset value	0001_h
Firmware version	FIR-v1426
Change history	

Description

Value in object $605B_h$	Description
-32768 … -1	Reserved
0	Blocking of the drive function – motor can turn freely
1	Braking with *slow down ramp* (braking deceleration depending on operating mode) and subsequent state change to *Ready to switch on*
2 … 32767	Reserved

605Ch Disable Option Code

Function

This object contains the action that is to be executed on a transition of the CiA 402 Power State Machine from the *Operation enabled* state to the *Switched on* state.

Object description

Index	$605C_h$
Object name	Disable Option Code
Object Code	VARIABLE
Data type	INTEGER16
Savable	yes, category: application

Access	read / write
PDO mapping	no
Allowed values	
Preset value	0001_h
Firmware version	FIR-v1426
Change history	

Description

Value in object $605C_h$	Description
-32768 … -1	Reserved
0	Blocking of the drive function – motor can turn freely
1	Braking with *slow down ramp* (braking deceleration depending on operating mode) and subsequent state change to *Switched on*
2 … 32767	Reserved

605Dh Halt Option Code

Function

The object contains the action that is to be executed if bit 8 (Halt) is set in controlword 6040_h.

Object description

Index	$605D_h$
Object name	Halt Option Code
Object Code	VARIABLE
Data type	INTEGER16
Savable	yes, category: application
Access	read / write
PDO mapping	no
Allowed values	
Preset value	0001_h
Firmware version	FIR-v1426
Change history	

Description

Value in object $605D_h$	Description
-32768 … 0	Reserved
1	Braking with *slow down ramp* (braking deceleration depending on operating mode)
2	Braking with *quick stop ramp* (6085_h)
3 … 32767	Reserved

605Eh Fault Option Code

Function

The object contains the action specifying how the motor is to be brought to a standstill in case of an error.

Object description

Index	$605E_h$
Object name	Fault Option Code
Object Code	VARIABLE
Data type	INTEGER16
Savable	yes, category: application
Access	read / write
PDO mapping	no
Allowed values	
Preset value	0002_h
Firmware version	FIR-v1426
Change history	

Description

Value in object $605E_h$	Description
-32768 … -1	Reserved
0	Blocking of the drive function – motor can turn freely
1	Braking with *slow down ramp* (braking deceleration depending on operating mode)
2	Braking with *quick stop ramp* (6085_h)
3 … 32767	Reserved

6060h Modes Of Operation

Function

The desired operating mode is entered in this object.

Object description

Index	6060_h
Object name	Modes Of Operation
Object Code	VARIABLE
Data type	INTEGER8
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00_h
Firmware version	FIR-v1426

Change history	Firmware version FIR-v1626: "Savable" entry changed from "no" to "yes, category: application".

Description

Mode	Description
-2	Auto setup
-1	Clock-direction mode
0	No mode change/no mode assigned
1	Profile Position Mode
2	Velocity Mode
3	Profile Velocity Mode
4	Profile Torque Mode
5	Reserved
6	Homing Mode
7	Interpolated Position Mode
8	Cyclic Synchronous Position Mode
9	Cyclic Synchronous Velocity Mode
10	Cyclic Synchronous Torque Mode

6061h Modes Of Operation Display

Function

Indicates the current operating mode. See also 6060h Modes Of Operation.

Object description

Index	6061_h
Object name	Modes Of Operation Display
Object Code	VARIABLE
Data type	INTEGER8
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00_h
Firmware version	FIR-v1426
Change history	

6062h Position Demand Value

Function

Indicates the current demand position in user-defined units.

Object description

Index	6062_h
Object name	Position Demand Value
Object Code	VARIABLE
Data type	INTEGER32
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h
Firmware version	FIR-v1426
Change history	

6063h Position Actual Internal Value

Function

Contains the current actual position in encoder increments. Unlike objects 6062_h and 6064_h, this value is not set to "0" following a Homing operation. The source is determined in 3203h Feedback Selection.

NOTICE

If the encoder resolution in object $608F_h$ = zero, the numerical values of this object are invalid.

Object description

Index	6063_h
Object name	Position Actual Internal Value
Object Code	VARIABLE
Data type	INTEGER32
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h
Firmware version	FIR-v1426
Change history	

6064h Position Actual Value

Function

Contains the current actual position in user-defined units. The source is determined in 3203h Feedback Selection.

Object description

Index	6064_h
Object name	Position Actual Value
Object Code	VARIABLE
Data type	INTEGER32
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h
Firmware version	FIR-v1426
Change history	

6065h Following Error Window

Function

Defines the maximum allowed following error in user-defined units symmetrically to the demand position.

Object description

Index	6065_h
Object name	Following Error Window
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000100_h
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1504: "Savable" entry changed from "no" to "yes, category: application".

Description

If the actual position deviates so much from the demand position that the value of this object is exceeded, bit 13 in object 6041_h is set. The deviation must last longer than the time in object 6066_h.

If the value of the "Following Error Window" is set to "FFFFFFFF"$_h$, following error monitoring is switched off.

A reaction to the following error can be set in object 3700_h. If a reaction is defined, an error is also entered in object 1003_h.

6066h Following Error Time Out

Function

Time in milliseconds until a larger following error results in an error message.

Object description

Index	6066_h
Object name	Following Error Time Out
Object Code	VARIABLE
Data type	UNSIGNED16
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0064_h
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1504: "Savable" entry changed from "no" to "yes, category: application".

Description

If the actual position deviates so much from the demand position that the value of object 6065_h is exceeded, bit 13 in object 6041_h is set. The deviation must persist for longer than the time defined in this object.

A reaction to the following error can be set in object 3700_h. If a reaction is defined, an error is also entered in object 1003_h.

6067h Position Window

Function

Specifies a range symmetrical to the target position within which that target is considered having been met in modes Profile Position and Interpolated Position Mode.

Object description

Index	6067_h
Object name	Position Window
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	$0000000A_h$
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1504: "Savable" entry changed from "no" to "yes, category: application".

Description

If the current position deviates from the target position by less than the value of this object, bit 10 in object 6041_h is set. The condition must be satisfied for longer than the time defined in object 6068_h.

If the value is set to "FFFFFFFF"$_h$, monitoring is switched off.

6068h Position Window Time

Function

The current position must be within the "Position Window" (6067$_h$) for this time in milliseconds for the target position to be considered having been met in the Profile Position and Interpolated Position Mode modes.

Object description

Index	6068$_h$
Object name	Position Window Time
Object Code	VARIABLE
Data type	UNSIGNED16
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0064$_h$
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1504: "Savable" entry changed from "no" to "yes, category: application".

Description

If the current position deviates from the target position by less than the value of object 6067$_h$, bit 10 in object 6041$_h$ is set. The condition must be satisfied for longer than the time defined in object 6068$_h$.

606Bh Velocity Demand Value

Function

Speed specification in user-defined units for the velocity controller.

Object description

Index	606B$_h$
Object name	Velocity Demand Value
Object Code	VARIABLE
Data type	INTEGER32
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000$_h$
Firmware version	FIR-v1426
Change history	

Description

This object contains the output of the ramp generator, which simultaneously serves as the preset value for the velocity controller.

606Ch Velocity Actual Value

Function

Current actual speed in user-defined units.

Object description

Index	$606C_h$
Object name	Velocity Actual Value
Object Code	VARIABLE
Data type	INTEGER32
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h
Firmware version	FIR-v1426
Change history	

606Dh Velocity Window

Function

Specifies a symmetrical range relative to the target speed within which the target is considered having been met in the Profile Velocity mode.

Object description

Index	$606D_h$
Object name	Velocity Window
Object Code	VARIABLE
Data type	UNSIGNED16
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	$001E_h$
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1614: "Savable" entry changed from "no" to "yes, category: application".

Description

If the current speed deviates from the set speed by less than the value of this object, bit 10 in object 6041h is set. The condition must be satisfied for longer than the time defined in object 606Eh (see also statusword in Profile Velocity Mode).

606Eh Velocity Window Time

Function

The current speed must be within the "Velocity Window" (606Dh) for this time (in milliseconds) for the target to be considered having been met.

Object description

Index	606Eh
Object name	Velocity Window Time
Object Code	VARIABLE
Data type	UNSIGNED16
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000h
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1614: "Savable" entry changed from "no" to "yes, category: application".

Description

Description

If the current speed deviates from the set speed by less than the value of object 606Dh, bit 10 in object 6041h is set. The condition must be satisfied for longer than the time defined in object 606E (see also statusword in Profile Velocity Mode).

606Fh Velocity Threshold

Function

Speed in user-defined units above which the actual speed in Profile Velocity mode is considered to be nonzero.

Object description

Index	606Fh
Object name	Velocity Threshold
Object Code	VARIABLE
Data type	UNSIGNED16
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO

Allowed values	
Preset value	0000_h
Firmware version	FIR-v2013-B726332
Change history	

Description

If the actual speed is greater than the value in $606F_h$(Velocity Threshold) for a time of 6070_h(Velocity Threshold Time), bit 12 in 6041_h(Statusword) has the value "0". The bit otherwise remains set to "1".

6070h Velocity Threshold Time

Function

Time in milliseconds above which an actual speed greater than the value in $606F_h$ in Profile Velocity mode is considered to be nonzero.

Object description

Index	6070_h
Object name	Velocity Threshold Time
Object Code	VARIABLE
Data type	UNSIGNED16
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000_h
Firmware version	FIR-v2013-B726332
Change history	

Description

If the actual speed is greater than the value in $606F_h$(Velocity Threshold) for a time of 6070_h(Velocity Threshold Time), bit 12 in 6041_h(Statusword) has the value "0". The bit otherwise remains set to "1".

6071h Target Torque

Function

This object contains the target torque for the Profile Torque and Cyclic Synchronous Torque modes in tenths of a percent of the rated torque.

Object description

Index	6071_h
Object name	Target Torque
Object Code	VARIABLE
Data type	INTEGER16
Savable	yes, category: application
Access	read / write

PDO mapping	RX-PDO
Allowed values	
Preset value	0000_h
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1626: "Savable" entry changed from "no" to "yes, category: application".

Description

This object is calculated as thousandths of the torque, e.g., the value "500" means "50%" of the rated torque; "1100" is equivalent to 110%. The rated torque corresponds to the rated current in object $203B_h$:01.

The minimum of 6073_h and 6072_h is used as limit for the torque in 6071_h.

The target torque may not exceed the peak torque (proportional to the maximum motor current in 2031_h).

6072h Max Torque

Function

The object describes the maximum torque for the Profile Torque and Cyclic Synchronous Torque modes in tenths of a percent of the rated torque.

Object description

Index	6072_h
Object name	Max Torque
Object Code	VARIABLE
Data type	UNSIGNED16
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0064_h
Firmware version	FIR-v1426
Change history	

Description

This object is calculated as thousandths of the torque, e.g., the value "500" means "50%" of the rated torque; "1100" is equivalent to 110%. The rated torque corresponds to the rated current in object $203B_h$:01.

The minimum of 6073_h and 6072_h is used as limit for the torque in 6071_h.

The target torque may not exceed the peak torque (proportional to the maximum motor current in 2031_h).

6073h Max Current

Function

Contains the maximum current in tenths of a percent of the set rated current. Is limited by the maximum motor current (2031_h). See also I2t Motor overload protection.

NOTICE

For stepper motors, only the rated current is specified, not a maximum current. Therefore, the value of 6073_h should generally not exceed the value 1000 (100%).

Object description

Index	6073_h
Object name	Max Current
Object Code	VARIABLE
Data type	UNSIGNED16
Savable	yes, category: drive
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	$03E8_h$
Firmware version	FIR-v1825-B577172
Change history	

Description

The maximum current is calculated in tenths of a percent of the rated current as follows:

$(6073_h*203B_h:01)/1000$

The maximum current determines:

- the maximum current for the I2t Motor overload protection
- the rated current in *open loop* mode.

6074h Torque Demand

Function

Current torque set value requested by the ramp generator in tenths of a percent of the rated torque for the internal controller.

Object description

Index	6074_h
Object name	Torque Demand
Object Code	VARIABLE
Data type	INTEGER16
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	0000_h
Firmware version	FIR-v1426
Change history	

Description

This object is calculated as thousandths of the torque, e.g., the value "500" means "50%" of the rated torque; "1100" is equivalent to 110%. The rated torque corresponds to the rated current in object $203B_h$:01.

The minimum of 6073_h and 6072_h is used as limit for the torque in 6071_h.

The target torque may not exceed the peak torque (proportional to the maximum motor current in 2031_h).

6075h Motor Rated Current

Function

Contains the rated current entered in $203B_h$:01_h in mA.

6077h Torque Actual Value

Function

This object indicates the current torque value in tenths of a percent of the rated torque for the internal controller.

Object description

Index	6077_h
Object name	Torque Actual Value
Object Code	VARIABLE
Data type	INTEGER16
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	0000_h
Firmware version	FIR-v1540
Change history	

Description

This object is calculated as thousandths of the torque, e.g., the value "500" means "50%" of the rated torque; "1100" is equivalent to 110%. The rated torque corresponds to the rated current in object $203B_h$:01.

The minimum of 6073_h and 6072_h is used as limit for the torque in 6071_h.

The target torque may not exceed the peak torque (proportional to the maximum motor current in 2031_h).

607Ah Target Position

Function

This object specifies the target position in user-defined units for the Profile Position and Cyclic Synchronous Position modes.

Object description

Index	$607A_h$
Object name	Target Position

Object Code	VARIABLE
Data type	INTEGER32
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	$00000FA0_h$
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1626: "Savable" entry changed from "no" to "yes, category: application".

607Bh Position Range Limit

Function

Contains the minimum and maximum position in user-defined units.

Object description

Index	$607B_h$
Object name	Position Range Limit
Object Code	ARRAY
Data type	INTEGER32
Savable	yes, category: application
Firmware version	FIR-v1426
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Min Position Range Limit
Data type	INTEGER32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h

Subindex	02_h
Name	Max Position Range Limit
Data type	INTEGER32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h

Description

If this range is exceeded or not reached, an overflow occurs. To prevent this overflow, limit values for the target position can be set in object $607D_h$ ("Software Position Limit").

607Ch Home Offset

Function

Specifies the difference between the zero position of the controller and the reference point of the machine in user-defined units.

Object description

Index	$607C_h$
Object name	Home Offset
Object Code	VARIABLE
Data type	INTEGER32
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h
Firmware version	FIR-v1426
Change history	

607Dh Software Position Limit

Function

Defines the limit positions relative to the reference point of the application in user-defined units.

Object description

Index	$607D_h$
Object name	Software Position Limit
Object Code	ARRAY
Data type	INTEGER32
Savable	yes, category: application
Firmware version	FIR-v1426
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Min Position Limit
Data type	INTEGER32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h

Subindex	02_h
Name	Max Position Limit
Data type	INTEGER32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h

Description

The absolute target position must lie within the limits set here. The Home Offset ($607C_h$) is not taken into account.

607Eh Polarity

Function

With this object, the direction of rotation can be reversed.

Object description

Index	$607E_h$
Object name	Polarity
Object Code	VARIABLE
Data type	UNSIGNED8
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	

Preset value	00_h
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1738-B501312: "PDO mapping" table entry for subindex 00 changed from "no" to "RX-PDO".

Description

The following generally applies for direction reversal: If a bit is set to the value "1", reversal is activated. If the value is "0", the direction of rotation is as described in the respective mode.

7	6	5	4	3	2	1	0
POS	VEL						

VEL (Velocity)

Direction of rotation reversal in the following modes:

- Profile Velocity Mode
- Cyclic Synchronous Velocity Mode

POS (Position)

Direction of rotation reversal in the following modes:

- Profile Position Mode
- Cyclic Synchronous Position Mode

TIP

You can force an inversion of the rotary field that affects all operating modes. See object 3212_h:02_h.

607Fh Max Profile Velocity

Function

Specifies the maximum speed in user-defined units for which the Mod i Profile Position , Interpolated Position Mode (only if closed loop is activated) and Profile Velocity.

Object description

Index	$607F_h$
Object name	Max Profile Velocity
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00007530_h
Firmware version	FIR-v1540
Change history	Firmware version FIR-v1738-B501312: "Object Name" entry changed from "Max profile velocity" to "Max Profile Velocity".

	Firmware version FIR-v1738-B501312: "Data type" entry changed from "INTEGER16" to "UNSIGNED32".
	Firmware version FIR-v1738-B501312: "Savable" entry changed from "no" to "yes, category: application".
	Firmware version FIR-v1738-B501312: "Access" table entry for subindex 00 changed from "read only" to "read/write".
	Firmware version FIR-v1738-B501312: "PDO mapping" table entry for subindex 00 changed from "TX-PDO" to "RX-PDO".

6080h Max Motor Speed

Function

Contains the maximum permissible speed of the motor in user-defined units.

Object description

Index	6080_h
Object name	Max Motor Speed
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: drive
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00007530_h
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1614: "Savable" entry changed from "yes, category: application" to "yes, category: tuning".
	Firmware version FIR-v1738-B501312: "Object Name" entry changed from "Maximum Speed" to "Max Motor Speed".
	Firmware version FIR-v1738-B501312: "PDO mapping" table entry for subindex 00 changed from "no" to "RX-PDO".
	Firmware version FIR-v1748-B538662: "Savable" entry changed from "yes, category: tuning" to "yes, category: movement".
	Firmware version FIR-v1825-B577172: "Savable" entry changed from "yes, category: movement" to "yes, category: tuning".
	Firmware version FIR-v1825-B577172: "Savable" entry changed from "yes, category: tuning" to "yes, category: movement".

6081h Profile Velocity

Function

Specifies the maximum travel speed in user-defined units.

Object description

Index	6081_h
Object name	Profile Velocity
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	$000001F4_h$
Firmware version	FIR-v1426
Change history	

6082h End Velocity

Function

Specifies the speed at the end of the traveled ramp in user-defined units.

Object description

Index	6082_h
Object name	End Velocity
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h
Firmware version	FIR-v1426
Change history	

6083h Profile Acceleration

Function

Specifies the maximum acceleration in user-defined units.

Object description

Index	6083_h
Object name	Profile Acceleration
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: application
Access	read / write

PDO mapping	RX-PDO
Allowed values	
Preset value	000001F4$_h$
Firmware version	FIR-v1426
Change history	

6084h Profile Deceleration

Function

Specifies the maximum deceleration (deceleration ramp) in user-defined units. Is limited by 60C6$_h$.

Object description

Index	6084$_h$
Object name	Profile Deceleration
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	000001F4$_h$
Firmware version	FIR-v1426
Change history	

6085h Quick Stop Deceleration

Function

Specifies the maximum Quick Stop Deceleration in user-defined units. Depending on the operating mode, is limited by 60C6$_h$ (Max Deceleration) and, if applicable, 60A4$_h$ (Profile Jerk).

Object description

Index	6085$_h$
Object name	Quick Stop Deceleration
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00001388$_h$
Firmware version	FIR-v1426
Change history	

6086h Motion Profile Type

Function

Specifies the ramp type for the Profile Position and Profile Velocity modes.

Object description

Index	6086_h
Object name	Motion Profile Type
Object Code	VARIABLE
Data type	INTEGER16
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000_h
Firmware version	FIR-v1426
Change history	

Description

Value = "0": = Trapezoidal ramp

Value = "3": Ramp with limited jerk

6087h Torque Slope

Function

This object contains the slope of the torque in Torque mode.

Object description

Index	6087_h
Object name	Torque Slope
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000064_h
Firmware version	FIR-v1426
Change history	

Description

This object is calculated as thousandths of the torque, e.g., the value "500" means "50%" of the rated torque; "1100" is equivalent to 110%. The rated torque corresponds to the rated current in object $203B_h$:01.

The minimum of 6073_h and 6072_h is used as limit for the torque in 6071_h.

The target torque may not exceed the peak torque (proportional to the maximum motor current in 2031h).

608Fh Position Encoder Resolution

Function

Contains the physical resolution (see objects 60E6h/ 60EBh) of the encoder/sensor that is used for position control (see 3203h Feedback Selection).

Object description

Index	608Fh
Object name	Position Encoder Resolution
Object Code	ARRAY
Data type	INTEGER32
Savable	yes, category: tuning
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1738-B501312: "Savable" entry changed from "yes, category: application" to "yes, category: tuning". Firmware version FIR-v1738-B501312: "PDO mapping" table entry for subindex 01 changed from "no" to "RX-PDO". Firmware version FIR-v1738-B501312: "PDO mapping" table entry for subindex 02 changed from "no" to "RX-PDO". Firmware version FIR-v1748-B538662: "Data type" entry changed from "UNSIGNED32" to "INTEGER32".

Value description

Subindex	00h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02h

Subindex	01h
Name	Encoder Increments
Data type	INTEGER32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000h

Subindex	02h
Name	Motor Revolutions

Data type	INTEGER32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001_h

Description

Position Encoder Resolution = Encoder Increments ($608F_h$:01_h) / Motor Revolutions ($608F_h$:02_h)

6090h Velocity Encoder Resolution

Function

Contains the physical resolution (see objects $60E6_h$/ $60EB_h$) of the encoder/sensor that is used for speed control (see 3203h Feedback Selection).

Object description

Index	6090_h
Object name	Velocity Encoder Resolution
Object Code	ARRAY
Data type	INTEGER32
Savable	yes, category: tuning
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1738-B501312
Change history	Firmware version FIR-v1748-B538662: "Data type" entry changed from "UNSIGNED32" to "INTEGER32". Firmware version FIR-v1748-B538662: "Data type" entry changed from "UNSIGNED32" to "INTEGER32". Firmware version FIR-v1748-B538662: "Data type" entry changed from "UNSIGNED32" to "INTEGER32". Firmware version FIR-v1825-B577172: "Data type" entry changed from "INTEGER32" to "UNSIGNED32". Firmware version FIR-v1825-B577172: "Data type" entry changed from "INTEGER32" to "UNSIGNED32". Firmware version FIR-v1825-B577172: "Data type" entry changed from "INTEGER32" to "UNSIGNED32". Firmware version FIR-v1825-B577172: "Data type" entry changed from "UNSIGNED32" to "INTEGER32". Firmware version FIR-v1825-B577172: "Data type" entry changed from "UNSIGNED32" to "INTEGER32". Firmware version FIR-v1825-B577172: "Data type" entry changed from "UNSIGNED32" to "INTEGER32".

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Encoder Increments Per Second
Data type	INTEGER32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h

Subindex	02_h
Name	Motor Revolutions Per Second
Data type	INTEGER32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001_h

Description

Velocity Encoder Resolution = Encoder Increments per second (6090_h:01_h) / Motor Revolutions per second (6090_h:02_h)

6091h Gear Ratio

Function

Contains the gear ratio (number of motor revolutions per revolution of the output shaft) of the encoder/sensor that is used for position control (see 3203h Feedback Selection).

Object description

Index	6091_h
Object name	Gear Ratio
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1738-B501312: "PDO mapping" table entry for subindex 01 changed from "no" to "RX-PDO".

	Firmware version FIR-v1738-B501312: "PDO mapping" table entry for subindex 02 changed from "no" to "RX-PDO".

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Motor Revolutions
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001_h

Subindex	02_h
Name	Shaft Revolutions
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001_h

Description

Gear Ratio = Motor Revolutions (6091_h:01_h) / Shaft Revolutions (6091_h:02_h)

6092h Feed Constant

Function

Contains the feed constant (feed in user-defined units per revolution of the output shaft) of the encoder/ sensor that is used for position control (see 3203h Feedback Selection).

Object description

Index	6092_h
Object name	Feed Constant
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application

Firmware version	FIR-v1426
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Feed
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001_h

Subindex	02_h
Name	Shaft Revolutions
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001_h

Description

Feed Constant = Feed (6092_h:01_h) / Shaft Revolutions (6092_h:02_h)

6096h Velocity Factor

Function

This object contains the factor that is used for converting from user-defined speed units. See chapter User-defined units.

Object description

Index	6096_h
Object name	Velocity Factor
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application

Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1738-B501312
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Numerator
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001_h

Subindex	02_h
Name	Divisor
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001_h

Description

The subindices have the following functions:

- 01_h: Numerator of the factor
- 02_h: Denominator of the factor

6097h Acceleration Factor

Function

This object contains the factor that is used for converting from user-defined acceleration units. See chapter User-defined units.

Object description

Index	6097_h
Object name	Acceleration Factor
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1738-B501312
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Numerator
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001_h

Subindex	02_h
Name	Divisor
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001_h

Description

The subindices have the following functions:

- 01_h: Numerator of the factor
- 02_h: Denominator of the factor

6098h Homing Method

Function

This object defines the Homing method in Homing mode.

Object description

Index	6098_h
Object name	Homing Method
Object Code	VARIABLE
Data type	INTEGER8
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	23_h
Firmware version	FIR-v1426
Change history	

6099h Homing Speed

Function

Specifies the speeds for homing mode (6098_h) in user-defined units.

Object description

Index	6099_h
Object name	Homing Speed
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Firmware version	FIR-v1426
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h

Name	Speed During Search For Switch
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000032_h

Subindex	02_h
Name	Speed During Search For Zero
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	$0000000A_h$

Description

The speed for the search for the switch is specified in subindex 1.

The (lower) speed for the search for the reference position is specified in subindex 2.

NOTICE

- The speed in subindex 2 is simultaneously the initial speed when starting the acceleration ramp. If this is set too high, the motor loses steps or fails to turn at all.
  If the setting is too high, the index marking will be overlooked, especially with high-resolution encoders. The minimum detectable width of the index pulse is 31.25 µs.
  .
- The speed in subindex 1 must be greater than the speed in subindex 2.

609Ah Homing Acceleration

Function

Specifies the acceleration ramp for homing mode in user-defined units.

Object description

Index	$609A_h$
Object name	Homing Acceleration
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	$000001F4_h$
Firmware version	FIR-v1426
Change history	

Description

The ramp is only used when starting up. When the switch is reached, the motor immediately switches to the lower speed; when the end position is reached, it immediately stops.

60A2h Jerk Factor

Function

This object contains the factor that is used for converting from user-defined jerk units. See chapter User-defined units.

Object description

Index	$60A2_h$
Object name	Jerk Factor
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1738-B501312
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Numerator
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001_h

Subindex	02_h
Name	Divisor
Data type	UNSIGNED32
Access	read / write

PDO mapping	RX-PDO
Allowed values	
Preset value	00000001_h

Description

The subindices have the following functions:

- 01_h: Numerator of the factor
- 02_h: Denominator of the factor

60A4h Profile Jerk

Function

In the case of a ramp with limited jerk, the size of the jerk can be entered in this object. An entry with the value "0" means that the jerk is not limited.

Object description

Index	$60A4_h$
Object name	Profile Jerk
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1614: "Name" entry changed from "End Acceleration Jerk" to "Begin Deceleration Jerk". Firmware version FIR-v1614: "Name" entry changed from "Begin Deceleration Jerk" to "End Acceleration Jerk".

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	04_h

Subindex	01_h
Name	Begin Acceleration Jerk
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	$000003E8_h$

Subindex	02_h
Name	Begin Deceleration Jerk
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	$000003E8_h$

Subindex	03_h
Name	End Acceleration Jerk
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	$000003E8_h$

Subindex	04_h
Name	End Deceleration Jerk
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	$000003E8_h$

Description

- Subindex 01_h (*Begin Acceleration Jerk*): Initial jerk during acceleration
- Subindex 02_h (*Begin Deceleration Jerk*): Initial jerk during braking
- Subindex 03_h (*End Acceleration Jerk*): Final jerk during acceleration
- Subindex 04_h (*End Deceleration Jerk*): Final jerk during braking

60A8h SI Unit Position

Function

This object contains the position unit. See chapter User-defined units.

Object description

Index	$60A8_h$
Object name	SI Unit Position
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: application
Access	read / write
PDO mapping	no
Allowed values	

Preset value	FF410000$_h$
Firmware version	FIR-v1738-B501312
Change history	

Description

Object 60A8$_h$ contains:

- Bits 16 to 23: The position unit (see chapter Units)
- Bits 24 to 31: The exponent of a power of ten (see chapter Units)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Factor								Unit							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
reserved (00h)								reserved (00h)							

60A9h SI Unit Velocity

Function

This object contains the speed unit. See chapter User-defined units.

Object description

Index	60A9$_h$
Object name	SI Unit Velocity
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: application
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00B44700$_h$
Firmware version	FIR-v1738-B501312
Change history	

Description

Object 60A9$_h$ contains:

- Bits 8 to 15: The time unit (see chapter Units)
- Bits 16 to 23: The position unit (see chapter Units)
- Bits 24 to 31: The exponent of a power of ten (see chapter Units)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Factor								Nominator (Position)							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Denominator (Time)								reserved (00h)							

60B0h Position Offset

Function

Offset for the position set value in user-defined units. Is taken into account in mode Cyclic Synchronous Position.

Object description

Index	$60B0_h$
Object name	Position Offset
Object Code	VARIABLE
Data type	INTEGER32
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h
Firmware version	FIR-v1738-B505321
Change history	

60B1h Velocity Offset

Function

Offset for the speed set value in user-defined units. Is taken into account in the Cyclic Synchronous Position, Cyclic Synchronous Velocity and Clock-direction mode modes.

Object description

Index	$60B1_h$
Object name	Velocity Offset
Object Code	VARIABLE
Data type	INTEGER32
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h
Firmware version	FIR-v1738-B505321
Change history	

60B2h Torque Offset

Function

Offset for the torque set value in tenths of a percent. Is taken into account in the Cyclic Synchronous Position, Cyclic Synchronous Velocity, Cyclic Synchronous Torque and Clock-direction mode modes.

Object description

Index	$60B2_h$
Object name	Torque Offset
Object Code	VARIABLE
Data type	INTEGER16
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000_h
Firmware version	FIR-v1738-B505321
Change history	

60C1h Interpolation Data Record

Function

This object contains the demand position in user-defined units for the interpolation algorithm for the interpolated position operating mode.

Object description

Index	$60C1_h$
Object name	Interpolation Data Record
Object Code	ARRAY
Data type	INTEGER32
Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1512
Change history	Firmware version FIR-v1626: "Savable" entry changed from "no" to "yes, category: application".

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	01_h

Subindex	01_h
Name	1st Set-point
Data type	INTEGER32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h

Description

The value is taken over at the next synchronization time.

60C2h Interpolation Time Period

Function

This object contains the interpolation time.

Object description

Index	$60C2_h$
Object name	Interpolation Time Period
Object Code	RECORD
Data type	INTERPOLATION_TIME_PERIOD
Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1426
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02_h

Subindex	01_h
Name	Interpolation Time Period Value
Data type	UNSIGNED8
Access	read / write
PDO mapping	no

Allowed values	
Preset value	01_h

Subindex	02_h
Name	Interpolation Time Index
Data type	INTEGER8
Access	read / write
PDO mapping	no
Allowed values	
Preset value	FD_h

Description

The subindices have the following functions:

- 01_h: Interpolation time.
- 02_h: Power of ten of the interpolation time: must have the value -3 (corresponds to the time basis in milliseconds).

The following applies here: cycle time = value of $60C2_h$:01_h * 10 $^{\text{value of 60C2:02}}$ seconds.

60C4h Interpolation Data Configuration

Function

This object offers the maximum buffer size, specifies the configured buffer organization of the interpolated data and offers objects for defining the size of the record and for deleting the buffer.

It is also used to store the position of other data points.

Object description

Index	$60C4_h$
Object name	Interpolation Data Configuration
Object Code	RECORD
Data type	INTERPOLATION_DATA_CONFIGURATION
Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1512
Change history	Firmware version FIR-v1540: "Access" table entry for subindex 05 changed from "read/write" to "write only". Firmware version FIR-v1540: "Access" table entry for subindex 06 changed from "read/write" to "write only". Firmware version FIR-v1626: "Savable" entry changed from "no" to "yes, category: application". Firmware version FIR-v1650-B472161: "Access" table entry for subindex 01 changed from "read/write" to "read only".

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	06_h

Subindex	01_h
Name	MaximumBufferSize
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000001_h

Subindex	02_h
Name	ActualBufferSize
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001_h

Subindex	03_h
Name	BufferOrganization
Data type	UNSIGNED8
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00_h

Subindex	04_h
Name	BufferPosition
Data type	UNSIGNED16
Access	read / write
PDO mapping	no
Allowed values	
Preset value	0001_h

Subindex	05_h

Name	SizeOfDataRecord
Data type	UNSIGNED8
Access	write only
PDO mapping	no
Allowed values	
Preset value	04_h

Subindex	06_h
Name	BufferClear
Data type	UNSIGNED8
Access	write only
PDO mapping	no
Allowed values	
Preset value	00_h

Description

The value of subindex 01_h contains the maximum possible number of interpolated records.

The value of subindex 02_h contains the current number of interpolated records.

If subindex 03_h is "00_h", this means a FIFO buffer organization; if it is "01_h", it specifies a ring buffer organization.

The value of subindex 04_h is unitless and specifies the next free buffer entry point.

The value of subindex 05_h is specified in units of "byte".

If the value "00_h" is written in subindex 06_h, it deletes the received data in the buffer, deactivates access and deletes all interpolated records.

If the value "01_h" is written in subindex 06_h, it activates access to the input buffer.

60C5h Max Acceleration

Function

This object contains the maximum permissible acceleration for the Profile Position and Profile Velocity modes.

Object description

Index	$60C5_h$
Object name	Max Acceleration
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00001388_h
Firmware version	FIR-v1426
Change history	

60C6h Max Deceleration

Function

This object contains the maximum permissible deceleration (deceleration ramp) for the Profile Position, Profile Velocity and Interpolated Position Mode operating modes.

Object description

Index	$60C6_h$
Object name	Max Deceleration
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00001388_h
Firmware version	FIR-v1426
Change history	

60E4h Additional Position Actual Value

Function

Contains the current actual position of all existing feedbacks in user-defined units.

Object description

Index	$60E4_h$
Object name	Additional Position Actual Value
Object Code	ARRAY
Data type	INTEGER32
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	
Firmware version	FIR-v1738-B501312
Change history	Firmware version FIR-v1748-B538662: "Data type" entry changed from "UNSIGNED32" to "INTEGER32". Firmware version FIR-v1748-B538662: "Data type" entry changed from "UNSIGNED32" to "INTEGER32".

Value description

Subindex	00_h
Name	Number Of Entries

Data type	UNSIGNED8
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	05_h

Subindex	01_h - 05_h
Name	Additional Position Actual Value #1 - #5
Data type	INTEGER32
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h

Description

The subindices have the following function:

- 00_h: Value="1" to "n", where "n" is the number of existing feedbacks.
- n_h:
  Subindex n contains the current actual position of the corresponding feedback.
  Subindex 01_h always corresponds to the first (and always existing) *sensorless* feedback.

60E5h Additional Velocity Actual Value

Function

Contains the current actual speed of all existing feedbacks in user-defined units.

Object description

Index	$60E5_h$
Object name	Additional Velocity Actual Value
Object Code	ARRAY
Data type	INTEGER32
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	
Firmware version	FIR-v1738-B501312
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only

PDO mapping	TX-PDO
Allowed values	
Preset value	05_h

Subindex	01_h - 05_h
Name	Additional Velocity Actual Value #1 - #5
Data type	INTEGER32
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h

Description

The subindices have the following function:

- 00_h: Value="1" to "n", where "n" is the number of existing feedbacks.
- n_h:
  Subindex n contains the current actual speed of the corresponding feedback.
  Subindex 01_h always corresponds to the first (and always existing) *sensorless* feedback.

60E6h Additional Position Encoder Resolution - Encoder Increments

Function

With this object and with 60EB$_h$, the resolution of each existing feedback is calculated.

Object description

Index	$60E6_h$
Object name	Additional Position Encoder Resolution - Encoder Increments
Object Code	ARRAY
Data type	INTEGER32
Savable	yes, category: tuning
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	
Firmware version	FIR-v1748-B538662
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	RX-PDO
Allowed values	

Preset value	05_h

Subindex	01_h - 05_h
Name	Additional Position Encoder Resolution - Encoder Increments Feedback Interface #1 - #5
Data type	INTEGER32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h

Description

The subindices have the following function:

- 00_h: Value="1" to "n", where "n" is the number of existing feedbacks.
- n_h:
  Subindex n contains the number of increments of the corresponding feedback.
  Subindex 01_h always corresponds to the first (and always existing) *sensorless* feedback.

The resolution of feedback "n" is calculated as follows:

Position Encoder Resolution = Encoder Increments ($60E6_h$:01_h) / Motor Revolutions ($60EB_h$:02_h)

NOTICE

The value "0" in a subindex means that the respective feedback is not connected and is not used. Thus, it is possible, for Example, to switch off the sensorless function to save computing time. This can be helpful if a *NanoJ* program needs the computing time.

If a value is not equal to "0" in a subindex, the controller checks the corresponding sensor when switching on. In case of an error (signal not present, invalid configuration/state), the error bit is set in the statusword and an error code stored in object 1003h.

60E8h Additional Gear Ratio - Motor Shaft Revolutions

Function

In this object and in $60ED_h$, you can set the gear ratio of each existing feedback.

Object description

Index	$60E8_h$
Object name	Additional Gear Ratio - Motor Shaft Revolutions
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	
Firmware version	FIR-v1738-B501312

Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	05_h

Subindex	01_h - 05_h
Name	Additional Gear Ratio - Motor Shaft Revolutions Feedback Interface #1 - #5
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001_h

Description

The subindices have the following function:

- 00_h: Value = "n", where "n" is the number of existing feedbacks.
- n_h: Subindex "n" contains the number of motor revolutions for the corresponding feedback. Subindex 01_h always corresponds to the first (and always existing) *sensorless* feedback.

The gear ratio of feedback "n" is calculated as follows:

Gear Ratio = Motor Shaft Revolutions ($60E8_h$:n_h) / Driving Shaft Revolutions ($60ED_h$:n_h)

60E9h Additional Feed Constant - Feed

Function

In this object and in $60EE_h$, you can set a feed constant for each existing feedback.

Object description

Index	$60E9_h$
Object name	Additional Feed Constant - Feed
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	

Firmware version	FIR-v1738-B501312
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	05_h

Subindex	01_h - 05_h
Name	Additional Feed Constant - Feed Feedback Interface #1 - #5
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001_h

Description

The subindices have the following function:

- 00_h: Value = "n", where "n" is the number of existing feedbacks.
- n_h: Subindex "n" contains the feed in user-defined units for the corresponding feedback. Subindex 01_h always corresponds to the first (and always existing) *sensorless* feedback.

The feed constant of feedback "n" is calculated as follows:

Feed Constant = Feed ($60E9_h$:n_h) / Driving Shaft Revolutions ($60EE_h$:n_h)

60EBh Additional Position Encoder Resolution - Motor Revolutions

Function

With this object and with $60E6_h$, the resolution of each existing feedback is calculated.

Object description

Index	$60EB_h$
Object name	Additional Position Encoder Resolution - Motor Revolutions
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: tuning
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	

Firmware version	FIR-v1738-B501312
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	05_h

Subindex	01_h - 05_h
Name	Additional Position Encoder Resolution - Motor Revolutions Feedback Interface #1 - #5
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001_h

Description

The subindices have the following function:

- 00_h: Value="1" to "n", where "n" is the number of existing feedbacks.
- n_h:
  Subindex n contains the number of motor revolutions of the corresponding feedback.
  Subindex 01_h always corresponds to the first (and always existing) *sensorless* feedback.

The resolution of feedback "n" is calculated as follows:

Position Encoder Resolution = Encoder Increments ($60E6_h$:n_h) / Motor Revolutions ($60EB_h$:n_h)

60EDh Additional Gear Ratio - Driving Shaft Revolutions

Function

In this object and in $60E8_h$, you can set the gear ratio of each existing feedback.

Object description

Index	$60ED_h$
Object name	Additional Gear Ratio - Driving Shaft Revolutions
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Access	read only
PDO mapping	RX-PDO

Allowed values	
Preset value	
Firmware version	FIR-v1738-B501312
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	05_h

Subindex	01_h - 05_h
Name	Additional Gear Ratio - Driving Shaft Revolutions Feedback Interface #1 - #5
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001_h

Description

The subindices have the following function:

- 00_h: Value = "n", where "n" is the number of existing feedbacks.
- n_h: Subindex "n" contains the number of revolutions of the output shaft for the corresponding feedback. Subindex 01_h always corresponds to the first (and always existing) *sensorless* feedback.

The gear ratio of feedback "n" is calculated as follows:

Gear Ratio = Motor Shaft Revolutions ($60E8_h$:n_h) / Driving Shaft Revolutions ($60ED_h$:n_h)

60EEh Additional Feed Constant - Driving Shaft Revolutions

Function

In this object and in $60E9_h$, you can set a feed constant for each existing feedback.

Object description

Index	$60EE_h$
Object name	Additional Feed Constant - Driving Shaft Revolutions
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Access	read only

PDO mapping	RX-PDO
Allowed values	
Preset value	
Firmware version	FIR-v1738-B501312
Change history	

Value description

Subindex	00_h
Name	Number Of Entries
Data type	UNSIGNED8
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	05_h

Subindex	01_h - 05_h
Name	Additional Feed Constant - Driving Shaft Revolutions Feedback Interface #1 - #5
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001_h

Description

The subindices have the following function:

- 00_h: Value = "n", where "n" is the number of existing feedbacks.
- n_h: Subindex "n" contains the number of revolutions of the output shaft for the corresponding feedback. Subindex 01_h always corresponds to the first (and always existing) *sensorless* feedback.

The feed constant of feedback "n" is calculated as follows:

Feed Constant = Feed ($60E9_h$:n_h) / Driving Shaft Revolutions ($60EE_h$:n_h)

60F2h Positioning Option Code

Function

The object describes the positioning behavior in Profile Position mode.

Object description

Index	$60F2_h$
Object name	Positioning Option Code
Object Code	VARIABLE
Data type	UNSIGNED16
Savable	yes, category: application

Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0001_h
Firmware version	FIR-v1446
Change history	Firmware version FIR-v1614: "Savable" entry changed from "no" to "yes, category: application".

Description

Only the following bits are supported at the present time:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MS	RESERVED [3]			IP OPTION [4]				RADO [2]		RRO [2]		CIO [2]		REL. OPT. [2]	

REL. OPT. (Relative Option)

These bits determine the behavior with relative rotating movement in "profile position" mode if bit 6 of controlword 6040_h = "1" is set.

Bit 1	Bit 0	Definition
0	0	Position movements are executed relative to the previous (internal absolute) target position (each relative to 0 if there is no previous target position)
0	1	Position movements are executed relative to the preset value (or output) of the ramp generator.
1	0	Position movements are performed relative to the current position (object 6064_h).
1	1	Reserved

RRO (Request-Response Option)

These bits determine the behavior when passing controlword 6040_h bit 4 ("new setpoint") – in this case, the controller releases the bit itself. This eliminates the need to externally reset the bit to "0" afterwards. After the bit is set to the value "0" by the controller, bit 12 ("setpoint acknowledgment") is also set to the value "0" in statusword 6041_h.

NOTICE

These options cause the controller to modify object controlword 6040_h.

Bit 5	Bit 4	Definition
0	0	The functionality is as described under Setting travel commands.
0	1	The controller releases the "new setpoint" bit as soon as the current targeted movement has reached its target.
1	0	The controller releases the "new setpoint" bit as soon this is possible for the controller.

Bit 5	Bit 4	Definition
1	1	Reserved

RADO (Rotary Axis Direction Option)

These bits determine the direction of rotation in "profile position" mode.

Bit 7	Bit 6	Definition
0	0	Normal positioning similar to a linear axis: If one of the "Position Range Limits" – $607B_h$:01_h and 02_h – is reached or exceeded, the preset is automatically transferred to the other end of the limit. Only with this bit combination is a movement greater than the modulo value possible.
0	1	Positioning only in negative direction: If the target position is greater than the current position, the axis moves to the target position via the "Min Position Range Limit" from object $607D_h$:01_h.
1	0	Positioning only in positive direction: If the target position is less than the current position, the axis moves to the target position via the "Max Position Range Limit" from object $607D_h$:01_h.
1	1	Positioning with the shortest distance to the target position. If the difference between the current position and the target position in a 360° system is less than 180°, the axis moves in the positive direction.

60F4h Following Error Actual Value

Function

This object contains the current following error in user-defined units.

Object description

Index	$60F4_h$
Object name	Following Error Actual Value
Object Code	VARIABLE
Data type	INTEGER32
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h
Firmware version	FIR-v1426
Change history	

60F8h Max Slippage

Function

Defines the maximum allowed slippage error in user-defined units symmetrically to the set speed in Profile Velocity mode.

Object description

Index	$60F8_h$
Object name	Max Slippage
Object Code	VARIABLE
Data type	INTEGER32
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000190_h
Firmware version	FIR-v1738-B501312
Change history	

Description

If the actual speed deviates so much from the set speed that the value (absolute value) of this object is exceeded, bit 13 in object 6041_h is set. The deviation must last longer than the time in object $203F_h$.

If the value of $60F8_h$ is set to "7FFFFFFF"$_h$, slippage error monitoring is switched off.

A reaction to the slippage error can be set in object 3700_h. If a reaction is defined, an error is also entered in object 1003_h.

60FAh Control Effort

Function

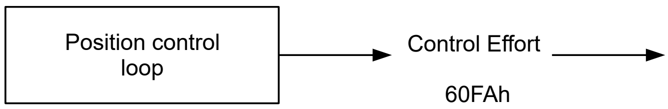
This object contains the correction speed (control variable) in user-defined units that is fed to the velocity controller by the position controller.

Object description

Index	$60FA_h$
Object name	Control Effort
Object Code	VARIABLE
Data type	INTEGER32
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h
Firmware version	FIR-v1748-B531667
Change history	

Description

The position controller calculates a correction speed (in user-defined units) from the difference between the current position and the demand position which is then passed on to the velocity controller. This correction value is dependent on the proportional component and integral component of the position controller. See also chapter Closed-Loop.



60FCh Position Demand Internal Value

Function

Indicates the current preset value for the position controller in increments of the sensor selected for the position (see Controller structure).

Object description

Index	$60FC_h$
Object name	Position Demand Internal Value
Object Code	VARIABLE
Data type	INTEGER32
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000_h
Firmware version	FIR-v1738-B501312
Change history	

60FDh Digital Inputs

Function

With this object, the digital inputs of the motor can be read.

Object description

Index	$60FD_h$
Object name	Digital Inputs
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	no
Access	read only
PDO mapping	TX-PDO

Allowed values	
Preset value	00000000_h
Firmware version	FIR-v1426
Change history	

Description

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
							...	IN 8	IN 7	IN 6	IN 5	IN 4	IN 3	IN 2	IN 1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												IL	HS	PLS	NLS

NLS (Negative Limit Switch)
Negative limit switch

PLS (Positive Limit Switch)
Positive limit switch

HS (Home Switch)
Home switch

IL (Interlock)
Interlock

IN n (Input n)
Input n – the number of used bits is dependent on the given controller.

60FEh Digital Outputs

Function

With this object, the digital outputs of the motor can be written.

Object description

Index	$60FE_h$
Object name	Digital Outputs
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1626: "Savable" entry changed from "no" to "yes, category: application". Firmware version FIR-v2213-B1028181: "Name" entry changed from "Digital Outputs #1" to "Physical Outputs".

Value description

Subindex	00_h
Name	Number Of Entries

Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	01_h

Subindex	01_h
Name	Physical Outputs
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001_h

Description

To write the outputs, the entries in object 3250_h, subindex 02_h to 05_h, must also be taken into account.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
											...	OUT4	OUT3	OUT2	OUT1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
															BRK

BRK (Brake)

Bit for the brake output (if the controller supports this function):

Value "1" means that the brake is activated (no current can flow between the two pins of the brake connection; the brake is closed).

OUT n (Output No n)

Bit for the respective digital output; the exact number of digital outputs is dependent on the controller.

60FFh Target Velocity

Function

In this object, the target speed for the profile velocity and cyclic synchronous velocity modes is entered in user-defined units.

Object description

Index	$60FF_h$
Object name	Target Velocity
Object Code	VARIABLE
Data type	INTEGER32
Savable	yes, category: application
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000_h

Firmware version	FIR-v1426
Change history	Firmware version FIR-v1626: "Savable" entry changed from "no" to "yes, category: application".

6502h Supported Drive Modes

Function

The object describes the supported operating modes in object 6060_h.

Object description

Index	6502_h
Object name	Supported Drive Modes
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	$000003EF_h$
Firmware version	FIR-v1426
Change history	

Description

The set bit specifies whether the respective mode is supported. If the value of the bit is "0", the mode is not supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						CST	CSV	CSP	IP	HM		TQ	PV	VL	PP

PP

Profile Position Mode

VL

Velocity Mode

PV

Profile Velocity Mode

TQ

Torque Mode

HM

Homing Mode

IP

Interpolated Position Mode

CSP
Cyclic Synchronous Position Mode

CSV
Cyclic Synchronous Velocity Mode

CST
Cyclic Synchronous Torque Mode

6503h Drive Catalogue Number

Function

Contains the device name as character string.

Object description

Index	6503_h
Object name	Drive Catalogue Number
Object Code	VARIABLE
Data type	VISIBLE_STRING
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	0
Firmware version	FIR-v1426
Change history	

6505h Http Drive Catalogue Address

Function

This object contains the manufacturer's web address as a character string.

Object description

Index	6505_h
Object name	Http Drive Catalogue Address
Object Code	VARIABLE
Data type	VISIBLE_STRING
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	0
Firmware version	FIR-v1426
Change history	

12 Copyrights

12.1 Introduction

Integrated in the Nanotec software are components from products from external software manufacturers. In this chapter, you will find the copyright information regarding the used external software sources.

12.2 AES

FIPS-197 compliant AES implementation

Based on XySSL: Copyright (C) 2006-2008 Christophe Devine

Copyright (C) 2009 Paul Bakker <polarssl_maintainer at polarssl dot org>

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution; or, the application vendor's website must provide a copy of this notice.
- Neither the names of PolarSSL or XySSL nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The AES block cipher was designed by Vincent Rijmen and Joan Daemen.

http://csrc.nist.gov/encryption/aes/rijndael/Rijndael.pdf

http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

12.3 MD5

MD5C.C - RSA Data Security, Inc., MD5 message-digest algorithm

Copyright (C) 1991-2, RSA Data Security, Inc. Created 1991. All rights reserved.

License to copy and use this software is granted provided that it is identified as the "RSA Data Security, Inc. MD5 Message-Digest Algorithm" in all material mentioning or referencing this software or this function.

License is also granted to make and use derivative works provided that such works are identified as "derived from the RSA Data Security, Inc. MD5 Message-Digest Algorithm" in all material mentioning or referencing the derived work.

RSA Data Security, Inc. makes no representations concerning either the merchantability of this software or the suitability of this software for any particular purpose. It is provided "as is" without express or implied warranty of any kind.

These notices must be retained in any copies of any part of this documentation and/or software.

12.4 uIP

12.5 DHCP

12.6 CMSIS DSP Software Library

12.7 FatFs

FatFs - FAT file system module include file R0.08 (C)ChaN, 2010

FatFs module is a generic FAT file system module for small embedded systems.

This is a free software that opened for education, research and commercial

developments under license policy of following trems.

12.8 Protothreads

Protothread class and macros for lightweight, stackless threads in C++.

This was "ported" to C++ from Adam Dunkels' protothreads C library at: http://www.sics.se/~adam/pt/

Originally ported for use by Hamilton Jet (www.hamiltonjet.co.nz) by Ben Hoyt, but stripped down for public release. See his blog entry about it for more information: http://blog.micropledge.com/2008/07/protothreads/

Original BSD-style license

12.9 lwIP

This file is part of the lwIP TCP/IP stack.

Author: Adam Dunkels <adam@sics.se>

12.10 littlefs

```
/*
* The little filesystem
*
* Copyright (c) 2017, Arm Limited. All rights reserved.
* SPDX-License-Identifier: BSD-3-Clause
*/
```