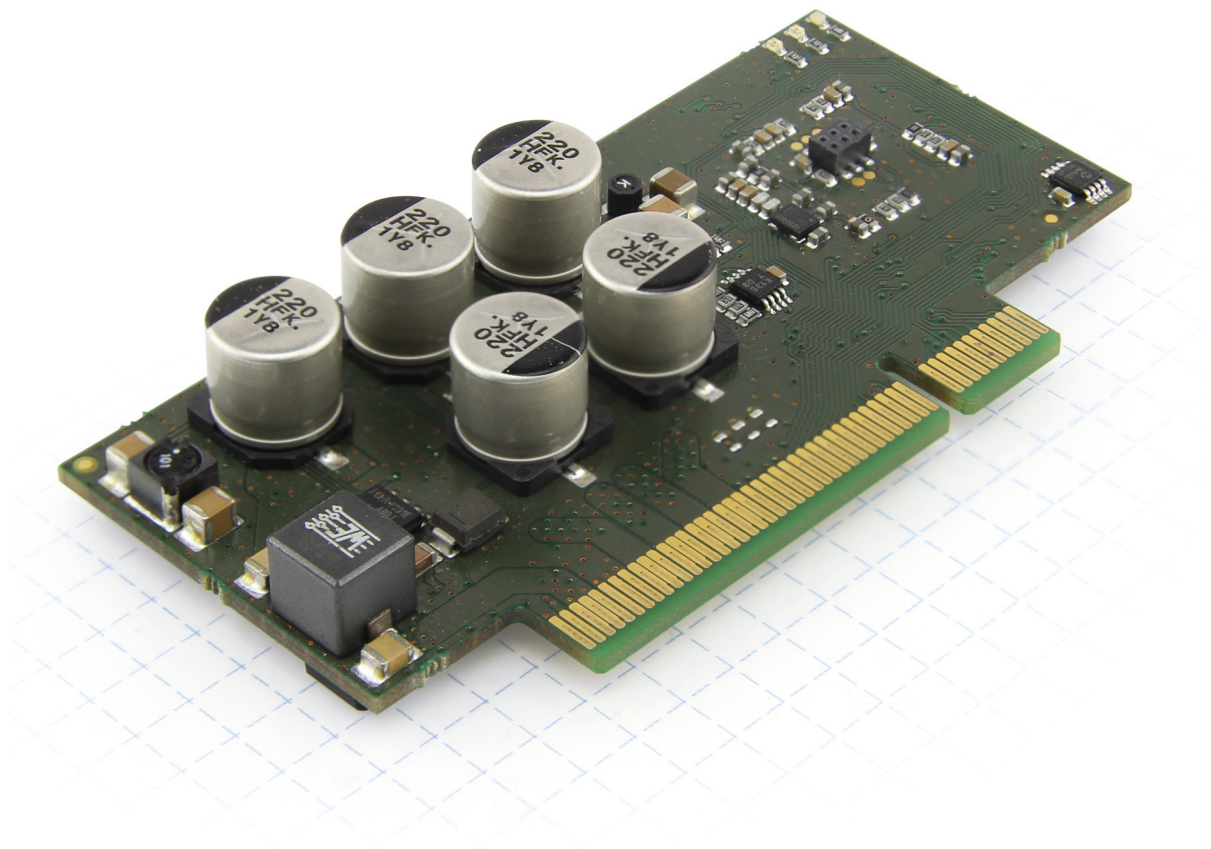


# Technical Manual NP5-02

Fieldbus: Modbus RTU



Valid with firmware version FIR-v2039  
and since hardware version W003b

Technical Manual Version: 1.3.0

# Contents

<b>1 Introduction.....</b>	<b>9</b>
1.1 Version information.....	9
1.2 Copyright, marking and contact.....	9
1.3 Intended use.....	10
1.4 Target group and qualification.....	10
1.5 Warranty and disclaimer.....	10
1.6 EU directives for product safety.....	11
1.7 Other applicable regulations.....	11
1.8 Used icons.....	11
1.9 Emphasis in the text.....	11
1.10 Numerical values.....	12
1.11 Bits.....	12
1.12 Counting direction (arrows).....	12
<b>2 Safety and warning notices.....</b>	<b>13</b>
<b>3 Technical details and pin assignment.....</b>	<b>14</b>
3.1 Environmental conditions.....	14
3.2 Dimensioned drawings.....	14
3.3 Electrical properties and technical data.....	15
3.4 Overtemperature protection.....	16
3.5 LED signaling.....	17
3.5.1 Power LED.....	17
3.6 Pin assignment.....	19
<b>4 Hardware installation.....</b>	<b>23</b>
4.1 Connecting the controller.....	23
4.1.1 Integrating the NP5.....	23
4.1.2 Connecting Modbus RTU.....	26
4.1.3 RS-485 line polarization.....	28
4.1.4 Connecting the NP5 controller via the <i>Discovery Board</i> .....	29
<b>5 Commissioning.....</b>	<b>37</b>
5.1 Communication settings.....	37
5.1.1 Modbus RTU.....	37
5.2 Establishing communication.....	40
5.2.1 Modbus RTU.....	40
5.3 Setting the motor data.....	40
5.4 Connecting the motor.....	41
5.5 Auto setup.....	41
5.5.1 Parameter determination.....	42
5.5.2 Execution.....	42
5.5.3 Parameter memory.....	44
5.6 Configuring the sensors.....	44
5.7 Test run.....	46
<b>6 General concepts.....</b>	<b>47</b>

6.1 Control modes.....	47
6.1.1 General.....	47
6.1.2 Open Loop.....	48
6.1.3 Closed Loop.....	50
6.1.4 Slow Speed.....	58
6.2 CiA 402 Power State Machine.....	60
6.2.1 State machine.....	60
6.2.2 Behavior upon exiting the <i>Operation enabled state</i> .....	62
6.3 User-defined units.....	65
6.3.1 Units.....	66
6.3.2 Encoder resolution.....	67
6.3.3 Gear ratio.....	67
6.3.4 Feed constant.....	68
6.3.5 Calculation formulas for user units.....	68
6.4 Limitation of the range of motion.....	70
6.4.1 Behavior upon reaching the limit switch.....	70
6.4.2 Software limit switches.....	70
6.5 Cycle times.....	70
<b>7 Operating modes.....</b>	<b>72</b>
7.1 Profile Position.....	72
7.1.1 Overview.....	72
7.1.2 Setting travel commands.....	73
7.1.3 Loss of accuracy for relative movements.....	77
7.1.4 Boundary conditions for a positioning move.....	78
7.1.5 Jerk-limited mode and non-jerk-limited mode.....	79
7.2 Velocity.....	80
7.2.1 Description.....	80
7.2.2 Activation.....	80
7.2.3 Controlword.....	80
7.2.4 Statusword.....	80
7.2.5 Object entries.....	80
7.3 Profile Velocity.....	81
7.3.1 Description.....	81
7.3.2 Activation.....	81
7.3.3 Controlword.....	81
7.3.4 Statusword.....	81
7.3.5 Object entries.....	82
7.4 Profile Torque.....	84
7.4.1 Description.....	84
7.4.2 Activation.....	84
7.4.3 Controlword.....	84
7.4.4 Statusword.....	84
7.4.5 Object entries.....	85
7.5 Homing.....	86
7.5.1 Overview.....	86
7.5.2 Homing method.....	87
7.6 Interpolated Position Mode.....	93
7.6.1 Overview.....	93
7.6.2 Activation.....	93
7.6.3 Controlword.....	93
7.6.4 Statusword.....	93
7.6.5 Use.....	93
7.6.6 Setup.....	94
7.6.7 Operation.....	94
7.7 Cyclic Synchronous Position.....	94
7.7.1 Overview.....	94
7.7.2 Object entries.....	95

7.8 Cyclic Synchronous Velocity.....	96
7.8.1 Overview.....	96
7.8.2 Object entries.....	96
7.9 Cyclic Synchronous Torque.....	97
7.9.1 Overview.....	97
7.9.2 Object entries.....	97
7.10 Clock-direction mode.....	98
7.10.1 Description.....	98
7.10.2 Activation.....	98
7.10.3 General.....	98
7.10.4 Statusword.....	99
7.10.5 Subtypes of the clock-direction mode.....	99
7.11 Auto setup.....	100
7.11.1 Description.....	100
7.11.2 Activation.....	100
7.11.3 Controlword.....	100
7.11.4 Statusword.....	100
<b>8 Special functions.....</b>	<b>101</b>
8.1 Digital inputs and outputs.....	101
8.1.1 Defining input and output assignments.....	101
8.1.2 Bit assignment.....	102
8.1.3 Digital inputs.....	102
8.1.4 Digital outputs.....	106
8.2 Automatic brake control.....	111
8.2.1 Description.....	111
8.2.2 Activation and connection.....	111
8.2.3 Brake control.....	111
8.2.4 Brake PWM.....	112
8.3 External ballast circuit.....	113
8.3.1 Control of the ballast resistor.....	113
8.3.2 Activating the ballast.....	114
8.3.3 Ballast monitoring.....	114
8.3.4 Example of a ballast circuit.....	115
8.4 I <sup>2</sup> t Motor overload protection.....	116
8.4.1 Description.....	116
8.4.2 Object entries.....	116
8.4.3 Activation.....	116
8.4.4 Function of I <sup>2</sup> t.....	116
8.5 Saving objects.....	117
8.5.1 General.....	117
8.5.2 Category: communication.....	118
8.5.3 Category: application.....	118
8.5.4 Category: customer.....	120
8.5.5 Category: drive.....	120
8.5.6 Category: tuning.....	120
8.5.7 Category: Modbus RTU.....	120
8.5.8 Starting the save process.....	120
8.5.9 Discarding the saved data.....	121
8.5.10 Verifying the configuration.....	121
<b>9 Modbus RTU.....</b>	<b>123</b>
9.1 RS-485.....	123
9.2 Modbus Modicon notation with PLCs.....	123
9.3 General.....	123
9.4 Communication settings.....	124
9.5 Function codes.....	124

9.6 Function code descriptions.....	125
9.6.1 FC 3 (03 <sub>h</sub> ) Read Input Registers / FC 4 (04 <sub>h</sub> ) Read Holding Registers.....	125
9.6.2 FC 6 (06 <sub>h</sub> ) Write Single Register.....	126
9.6.3 FC 16 (10 <sub>h</sub> ) Write Multiple Registers.....	127
9.6.4 FC 17 (11 <sub>h</sub> ) Report Server ID.....	128
9.6.5 FC 23 (17 <sub>h</sub> ) Read/Write Multiple registers.....	129
9.6.6 FC 8 (08 <sub>h</sub> ) Diagnostics.....	130
9.6.7 FC 43 (2B <sub>h</sub> ) Encapsulated Interface Transport.....	135
9.6.8 FC 101 (65 <sub>h</sub> ) Read complete object dictionary.....	141
9.6.9 FC 102 (66 <sub>h</sub> ) Read complete array or record.....	144
9.6.10 Exception codes.....	148
9.7 Process data objects (PDO).....	148
9.7.1 Configuration.....	148
9.7.2 Transfer.....	148
9.8 NanoJ objects.....	149
<b>10 Programming with <i>NanoJ</i>.....</b>	<b>151</b>
10.1 NanoJ program.....	151
10.2 Mapping in the NanoJ program.....	155
10.3 NanoJ functions in the NanoJ program.....	156
10.4 Restrictions and possible problems.....	158
<b>11 Description of the object dictionary.....</b>	<b>160</b>
11.1 Overview.....	160
11.2 Structure of the object description.....	160
11.3 Object description.....	160
11.4 Value description.....	161
11.5 Description.....	162
1000h Device Type.....	163
1001h Error Register.....	164
1003h Pre-defined Error Field.....	165
1008h Manufacturer Device Name.....	169
1009h Manufacturer Hardware Version.....	169
100Ah Manufacturer Software Version.....	170
1010h Store Parameters.....	170
1011h Restore Default Parameters.....	174
1018h Identity Object.....	177
1020h Verify Configuration.....	179
1F50h Program Data.....	180
1F51h Program Control.....	181
1F57h Program Status.....	182
2028h MODBUS Slave Address.....	183
202Ah MODBUS RTU Baudrate.....	183
202Ch MODBUS RTU Stop Bits.....	184
202Dh MODBUS RTU Parity.....	184
2030h Pole Pair Count.....	185
2031h Max Motor Current.....	185
2034h Upper Voltage Warning Level.....	186
2035h Lower Voltage Warning Level.....	186
2036h Open Loop Current Reduction Idle Time.....	187
2037h Open Loop Current Reduction Value/factor.....	187
2038h Brake Controller Timing.....	188
2039h Motor Currents.....	190
203Ah Homing On Block Configuration.....	192
203Bh I2t Parameters.....	193
203Dh Torque Window.....	196
203Eh Torque Window Time Out.....	196

203Fh Max Slippage Time Out.....	197
2057h Clock Direction Multiplier.....	197
2058h Clock Direction Divider.....	198
205Ah Absolute Sensor Boot Value (in User Units).....	198
205Bh Clock Direction Or Clockwise/Counter Clockwise Mode.....	199
2084h Bootup Delay.....	199
2101h Fieldbus Module Availability.....	200
2102h Fieldbus Module Control.....	201
2103h Fieldbus Module Status.....	202
2290h PDI Control.....	204
2291h PDI Input.....	204
2292h PDI Output.....	206
2300h NanoJ Control.....	207
2301h NanoJ Status.....	208
2302h NanoJ Error Code.....	209
230Fh Uptime Seconds.....	210
2310h NanoJ Input Data Selection.....	210
2320h NanoJ Output Data Selection.....	212
2330h NanoJ In/output Data Selection.....	213
2400h NanoJ Inputs.....	214
2410h NanoJ Init Parameters.....	215
2500h NanoJ Outputs.....	216
2600h NanoJ Debug Output.....	217
2701h Customer Storage Area.....	218
2800h Bootloader And Reboot Settings.....	219
3202h Motor Drive Submode Select.....	220
3203h Feedback Selection.....	221
3204h Feedback Mapping.....	223
320Dh Torque Of Inertia Factor.....	225
320Eh Closed Loop Controller Parameter.....	226
320Fh Open Loop Controller Parameter.....	231
3210h Motor Drive Parameter Set.....	233
3212h Motor Drive Flags.....	237
3220h Analog Inputs.....	239
3225h Analogue Inputs Switches.....	240
3231h Flex IO Configuration.....	241
3240h Digital Inputs Control.....	243
3242h Digital Input Routing.....	246
3243h Digital Input Homing Capture.....	248
3250h Digital Outputs Control.....	249
3252h Digital Output Routing.....	252
3320h Read Analogue Input.....	254
3321h Analogue Input Offset.....	255
3322h Analogue Input Factor Numerator.....	256
3323h Analogue Input Factor Denominator.....	257
3380h Feedback Sensorless.....	259
3390h Feedback Hall.....	260
33A0h Feedback Incremental A/B/I 1.....	263
33A1h Feedback Incremental A/B/I 2.....	264
3502h MODBUS Rx PDO Mapping.....	266
3602h MODBUS Tx PDO Mapping.....	269
3700h Deviation Error Option Code.....	273
3701h Limit Switch Error Option Code.....	274
4012h HW Information.....	275
4013h HW Configuration.....	276
4014h Operating Conditions.....	277
4021h Ballast Configuration.....	279
4040h Drive Serial Number.....	281
4041h Device Id.....	281

4042h Bootloader Infos.....	282
603Fh Error Code.....	283
6040h Controlword.....	284
6041h Statusword.....	285
6042h VI Target Velocity.....	286
6043h VI Velocity Demand.....	287
6044h VI Velocity Actual Value.....	287
6046h VI Velocity Min Max Amount.....	288
6048h VI Velocity Acceleration.....	289
6049h VI Velocity Deceleration.....	290
604Ah VI Velocity Quick Stop.....	291
604Ch VI Dimension Factor.....	292
605Ah Quick Stop Option Code.....	293
605Bh Shutdown Option Code.....	294
605Ch Disable Option Code.....	294
605Dh Halt Option Code.....	295
605Eh Fault Option Code.....	296
6060h Modes Of Operation.....	296
6061h Modes Of Operation Display.....	297
6062h Position Demand Value.....	297
6063h Position Actual Internal Value.....	298
6064h Position Actual Value.....	298
6065h Following Error Window.....	299
6066h Following Error Time Out.....	299
6067h Position Window.....	300
6068h Position Window Time.....	301
606Bh Velocity Demand Value.....	301
606Ch Velocity Actual Value.....	302
606Dh Velocity Window.....	302
606Eh Velocity Window Time.....	303
606Fh Velocity Threshold.....	303
6070h Velocity Threshold Time.....	304
6071h Target Torque.....	304
6072h Max Torque.....	305
6073h Max Current.....	305
6074h Torque Demand.....	306
6075h Motor Rated Current.....	307
6077h Torque Actual Value.....	307
607Ah Target Position.....	308
607Bh Position Range Limit.....	308
607Ch Home Offset.....	309
607Dh Software Position Limit.....	309
607Eh Polarity.....	310
607Fh Max Profile Velocity.....	311
6080h Max Motor Speed.....	312
6081h Profile Velocity.....	313
6082h End Velocity.....	313
6083h Profile Acceleration.....	314
6084h Profile Deceleration.....	314
6085h Quick Stop Deceleration.....	314
6086h Motion Profile Type.....	315
6087h Torque Slope.....	315
608Fh Position Encoder Resolution.....	316
6090h Velocity Encoder Resolution.....	317
6091h Gear Ratio.....	319
6092h Feed Constant.....	320
6096h Velocity Factor.....	321
6097h Acceleration Factor.....	322
6098h Homing Method.....	323



6099h Homing Speed.....	323
609Ah Homing Acceleration.....	325
60A2h Jerk Factor.....	325
60A4h Profile Jerk.....	326
60A8h SI Unit Position.....	328
60A9h SI Unit Velocity.....	328
60B0h Position Offset.....	329
60B1h Velocity Offset.....	329
60B2h Torque Offset.....	330
60C1h Interpolation Data Record.....	330
60C2h Interpolation Time Period.....	331
60C4h Interpolation Data Configuration.....	332
60C5h Max Acceleration.....	335
60C6h Max Deceleration.....	335
60E4h Additional Position Actual Value.....	335
60E5h Additional Velocity Actual Value.....	337
60E6h Additional Position Encoder Resolution - Encoder Increments.....	338
60E8h Additional Gear Ratio - Motor Shaft Revolutions.....	339
60E9h Additional Feed Constant - Feed.....	340
60EBh Additional Position Encoder Resolution - Motor Revolutions.....	341
60EDh Additional Gear Ratio - Driving Shaft Revolutions.....	342
60EEh Additional Feed Constant - Driving Shaft Revolutions.....	343
60F2h Positioning Option Code.....	344
60F4h Following Error Actual Value.....	346
60F8h Max Slippage.....	346
60FAh Control Effort.....	347
60FCh Position Demand Internal Value.....	347
60FDh Digital Inputs.....	348
60FEh Digital Outputs.....	349
60FFh Target Velocity.....	350
6502h Supported Drive Modes.....	350
6503h Drive Catalogue Number.....	351
6505h Http Drive Catalogue Address.....	352

## **12 Copyrights.....353**

12.1 Introduction.....	353
12.2 AES.....	353
12.3 MD5.....	353
12.4 uIP.....	354
12.5 DHCP.....	354
12.6 CMSIS DSP Software Library.....	354
12.7 FatFs.....	354
12.8 Protothreads.....	355
12.9 lwIP.....	355
12.10 littlefs.....	356



## 1 Introduction

The *NP5* is a controller for BLDC and stepper motors in plug-in module format (PCI-format connector strip) for integration in your own developments.

### Note



The PCI-format connector strip is not electrically compatible with PCI Express. Under no circumstances is it to be plugged into the PC mainboard.

This manual describes the integration of the *NP5* 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](http://us.nanotec.com).

### 1.1 Version information

Manual version	Date	Changes	Firmware version	Hardware version
1.0.0	03/2018	First edition	FIR-v1748	W003a
1.0.1	08/2018	Additions and error corrections	FIR-v1748	W003a
1.0.2	09/2018	Additions and error corrections	FIR-v1748	W003a
1.1.0	08/2019	<ul style="list-style-type: none"> <li>■ New chapter <a href="#">Configuring the sensors</a></li> <li>■ Changes and additions in chapter <a href="#">Closed Loop</a></li> <li>■ New sections in chapter <i>Control modes</i>: <a href="#">Controller structure</a>, <a href="#">Feed forward</a>, <a href="#">Assignment of the feedbacks to the control loops</a> and <a href="#">Slow Speed</a></li> <li>■ Minor additions and error corrections in the object dictionary</li> </ul>	FIR-v1926	W003b
1.2.0	10/2019	Error corrections	FIR-v1939	W003b
1.3.0	11/2020	<ul style="list-style-type: none"> <li>■ New chapter <a href="#">External ballast circuit</a></li> <li>■ New object <a href="#">3231<sub>h</sub>:03<sub>h</sub></a> for activating the alternative function of the inputs/outputs</li> <li>■ New objects <a href="#">606F<sub>h</sub></a> and <a href="#">6070<sub>h</sub></a> for monitoring the actual speed in <i>Profile Velocity</i> mode</li> <li>■ New object <a href="#">3250<sub>h</sub>:09<sub>h</sub></a> for switching the LEDs on/off</li> <li>■ New object <a href="#">320E<sub>h</sub>:0D<sub>h</sub></a> for setting a voltage feed forward (see <a href="#">Feed forward</a>)</li> <li>■ New objects <a href="#">320E<sub>h</sub>:0F<sub>h</sub></a> and <a href="#">320F<sub>h</sub>:05<sub>h</sub></a> for setting the maximum PWM voltage</li> </ul>	FIR-v2039	W003b

### 1.2 Copyright, marking and contact

© 2013 – 2020 Nanotec Electronic GmbH & Co. KG. All rights reserved.



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](http://us.nanotec.com)

### 1.3 Intended use

The *NP5* serves to control stepper motors and 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 PCI-format connector strip and a suitable motherboard. The system boundary of the *NP5* ends at the PCI connector strip.

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 and 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 assumes no liability for damages and malfunctions resulting from installation errors, failure to observe this manual or improper repairs. The selection and use of Nanotec products is the responsibility of the plant engineer or end user. Nanotec accepts no responsibility for the integration of the product in the end system.

Our general terms and conditions apply: [en.nanotec.com/service/general-terms-and-conditions/](http://en.nanotec.com/service/general-terms-and-conditions/).

Customers of Nanotec Electronic US Inc. please refer to [us.nanotec.com/service/general-terms-andconditions/](http://us.nanotec.com/service/general-terms-andconditions/).



#### Note

Changes or modifications to the product are not permitted.

## 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.

<b>CAUTION</b>	
	<p><b>The CAUTION notice indicates a possibly dangerous situation.</b></p> <p>Failure to observe the notice <b>may</b> result in moderately severe injuries.</p> <ul style="list-style-type: none"> <li>▶ Describes how you can avoid the dangerous situation.</li> </ul>
<b>Note</b>	
	<p><b>Indicates a possible incorrect operation of the product.</b></p> <p>Failure to observe the notice may result in damage to this or other products.</p> <ul style="list-style-type: none"> <li>▶ Describes how you can avoid the incorrect operation.</li> </ul>
<b>Tip</b>	
	<p>Shows a tip for the application or task.</p>

## 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<sub>h</sub> (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<sub>h</sub>, bit 0 = "1".

- If a holding torque is already needed in this state, the value "1" must be written in 3212<sub>h</sub>:01<sub>h</sub>.

### 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.

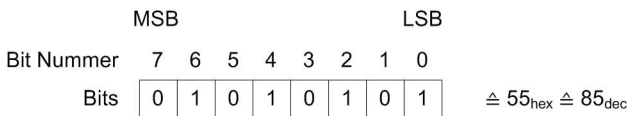
The objects in the object dictionary are written with index and subindex as follows: <Index>:<Subindex>

Both the index as well as the subindex are specified in hexadecimal notation. If no subindex is listed, the subindex is 00<sub>h</sub>.

Example: Subindex 5 of object 1003<sub>h</sub> is addressed with 1003<sub>h</sub>:05<sub>h</sub>, subindex 00 of object 6040<sub>h</sub> with 6040<sub>h</sub>.

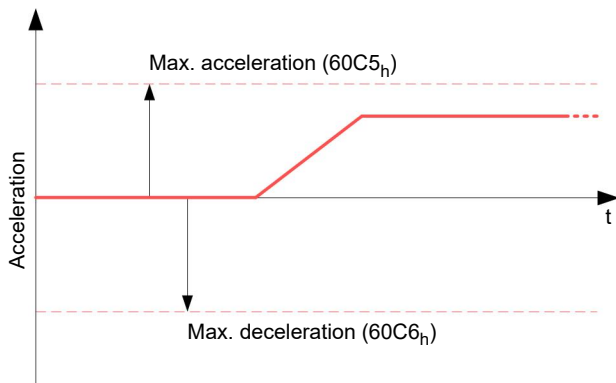
### 1.11 Bits

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.



### 1.12 Counting direction (arrows)

In figures, the counting direction is always in the direction of an arrow. Objects 60C5<sub>h</sub> and 60C6<sub>h</sub> depicted as examples in the following figure are both specified as positive.



## 2 Safety and warning notices

### Note



#### **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.

### Note



#### **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.

### Note



#### **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.

### Note



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

Polarity reversal results in a short-circuit between supply voltage and GND (earth) via the power diode.

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

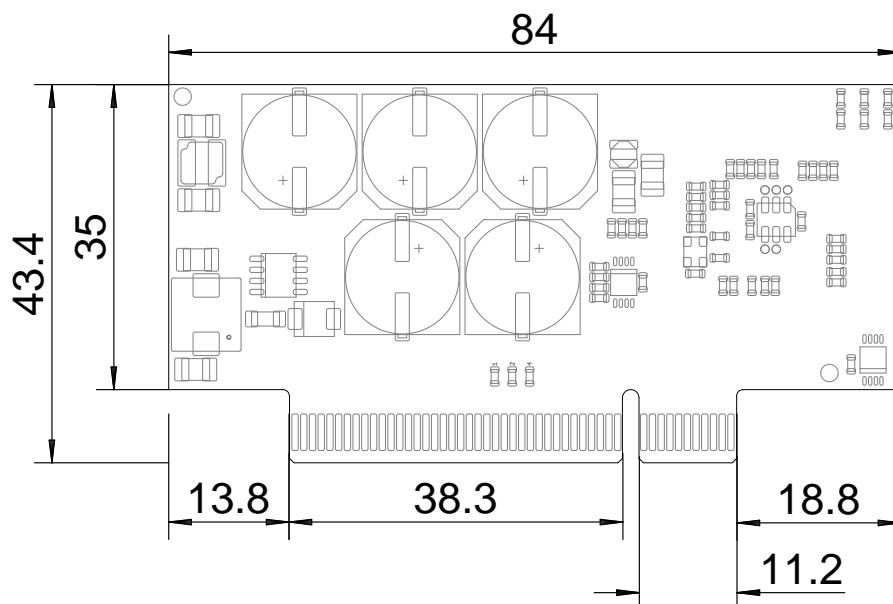
## 3 Technical details and pin assignment

### 3.1 Environmental conditions

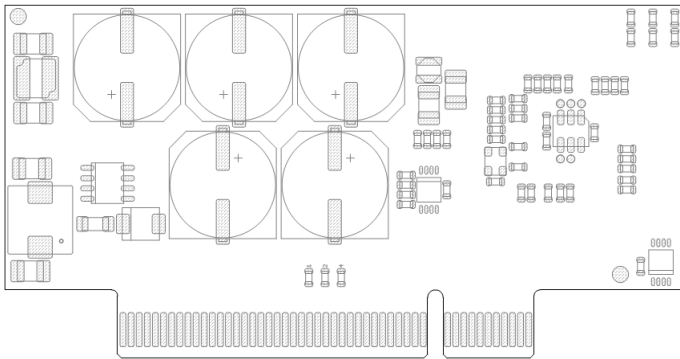
Environmental condition	Value
Protection class	No IP protection
Ambient temperature (operation)	-10 ... +40°C
Air humidity (non-condensing)	0 ... 95 %
Max. Altitude of site above <i>sea level</i> (without drop in performance)	1500 m
Ambient temperature (storage)	-25 ... +85°C

### 3.2 Dimensioned drawings

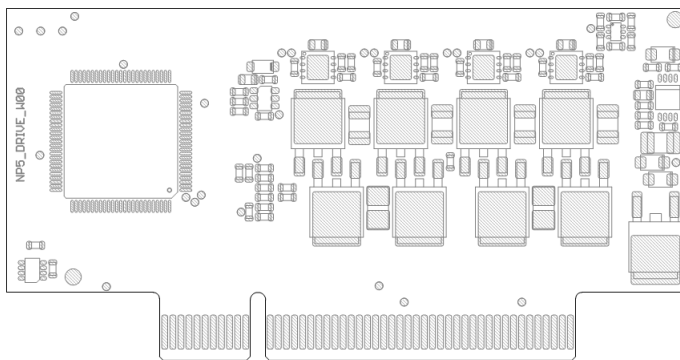
All dimensions are in millimeters.



The following figures show the board layout.



Side A



Side B

### 3.3 Electrical properties and technical data

Property	Description / value
Operating voltage	12 - 48 V DC $\pm 4\%$
Rated current	6 A <sub>rms</sub>
Peak current	10 A <sub>rms</sub> (for 1 second)
Commutation	Stepper motor <i>open loop</i> , stepper motor <i>closed loop</i> with encoder, BLDC sine commutated via Hall sensor, BLDC sine commutated via encoder <b>Note:</b> External wiring is required for encoder and Hall sensor!
Operating modes	<i>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</i>
Set value setting / programming	<i>Clock-direction, analog, NanoJ program</i>
Interfaces	2x SPI, 1x I <sup>2</sup> C
Encoder/Hall	2x encoder 1x Hall sensor <b>Note:</b> External wiring is required for encoder and Hall sensor!
I/O	6x general I/O, 2x analog input, 1x output for the external brake (open drain), 1x output for the external ballast circuit



Property	Description / value
Connector	PCI Express 8x, 1.0 mm RM, 2x49 contacts
Overtemperature	Protection circuit at temperature > 75°C
Polarity reversal protection	Polarity reversal protection by power diode (short-circuit between +UB and GND, fuse necessary in supply line)
Fuse size for polarity reversal protection:	$I_{\max}(\text{controller}) < I(\text{tripping current for fuse}) < I_{\max}(\text{voltage supply})$
Charging capacitor	For each ampere of rated current on the motor, Nanotec recommends a capacitance of approx. 1000 µF.

#### Note



- For the digital inputs, the switch-on threshold is 1.86 V, the switch-off threshold is 0.91 V.
- For the digital inputs, the maximum sampling frequency is 1 MHz.
- The range of the analog inputs is 0 ... 3.3 V.

#### Tip



If the fuse value ( $I$  tripping current for fuse) is very close to the maximum current consumption of the controller ( $I_{\max}$  controller), a *medium / slow* tripping characteristics should be used.

### 3.4 Overtemperature protection

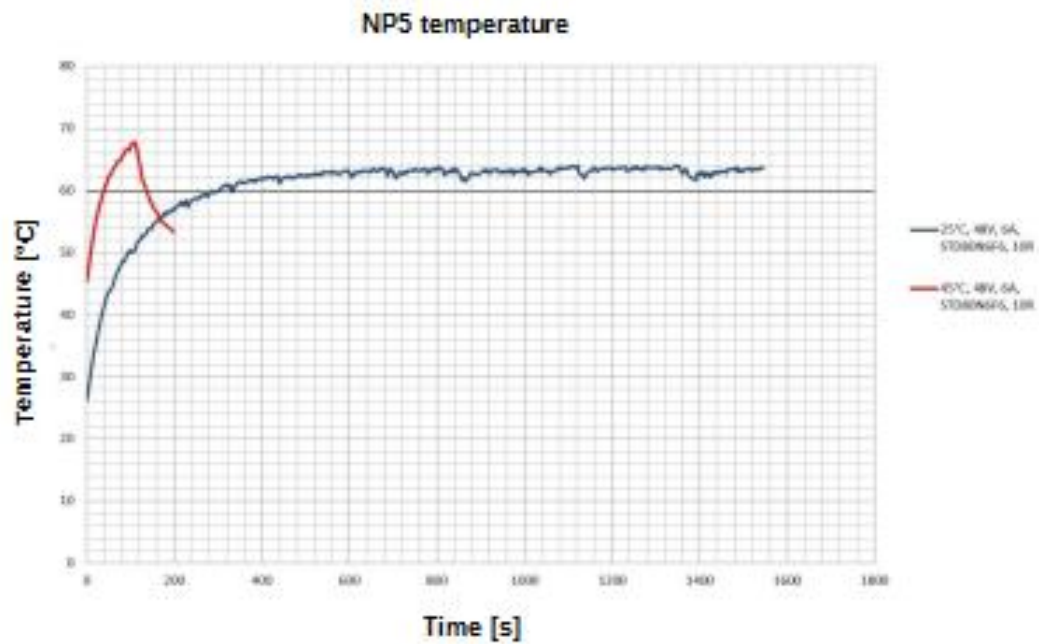
Above a temperature of approx. 75 °C on the power board the power part of the controller switches off and the error bit is set (see objects [1001<sub>h</sub>](#) and [1003<sub>h</sub>](#)). After cooling down and confirming the error (see [table for the controlword](#), "Fault reset"), the controller again functions normally.

The following temperature test results provide information on the temperature behavior of this controller.

Temperature tests are performed under the following conditions:

- Operating voltage: 48 V DC
- Motor current: 6 A rms
- Operation mode: Velocity Mode, full step, 30 rpm
- Ambient temperature: 25 °C / 45 °C
- Altitude of site: 500 m above sea level
- No external cooling in the climatic chamber, e. g., via fan

The following graphic shows the results of the temperature tests:



### Summary:

At 25°C (+48 V, 6 A rms, Velocity Mode 30 rpm), the controller was in operation for longer than 2 hours without having been switched off. The temperature was stable at approx. 62°C.

At 45°C (+48 V, 6 A rms, Velocity Mode 30 rpm), temperature protection switched off the controller in less than 2 minutes.

### Note

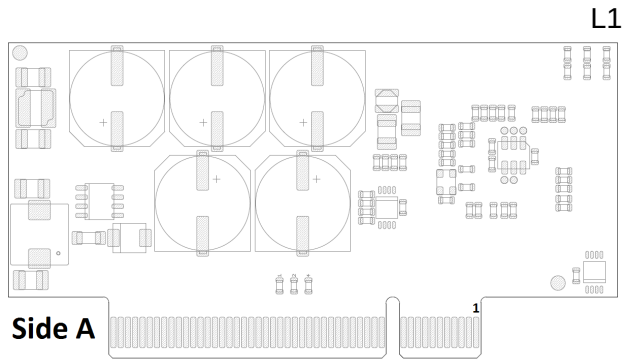


Aside from the motor, the exact temperature behavior is also dependent on the flange connection and the heat transfer there as well as on the convection in the application. For this reason, we recommend always performing an endurance test in the actual environment for applications in which current level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

## 3.5 LED signaling

### 3.5.1 Power LED

The power LED indicates the current status.



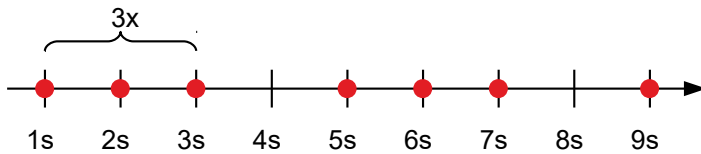
### 3.5.1.1 Normal operation

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



#### Note

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

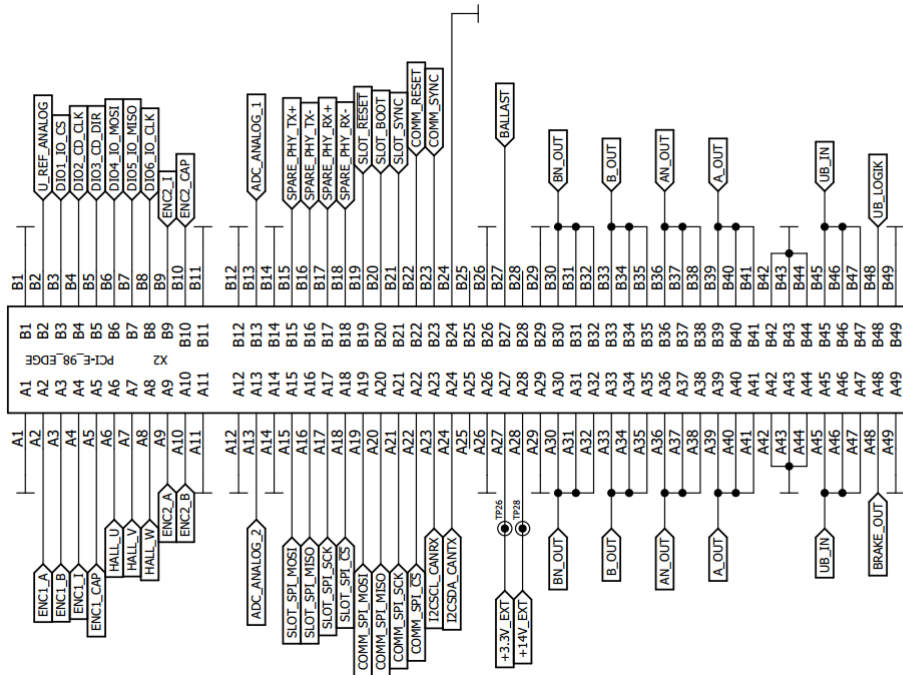


#### Tip

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

### 3.6 Pin assignment

PIN assignment of the PCI connector strip



**Note**



- For digital inputs 1 to 6, the switch-on threshold is 1.86 V, the switch-off threshold is 0.91 V DC. 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.
- The range of the analog inputs is 0 ... 3.3 V DC.
- The encoder signal is single-ended, the switch-on threshold is 1.86 V, the switch-off threshold is 0.91 V DC. The maximum sampling frequency is 1 MHz.
- The current consumption of the UB\_LOGIC logic supply is approx. 30 mA at 24 V DC.

PCI pin assignment:

Pin	Name	Description/function
A1	GND	
A2	ENC1_A	Encoder 1, A
A3	ENC1_B	Encoder 1, B
A4	ENC1_I	Encoder 1, Index
A5	ENC1_CAP	Not used
A6	HALL_U (H1)	Hall sensor 1 (U)
A7	HALL_V (H2)	Hall sensor 2 (V)
A8	HALL_W (H3)	Hall sensor 3 (W)
A9	ENC2_A	Encoder 2, A
A10	ENC2_B	Encoder 2, B
A11	GND	

Pin	Name	Description/function
A12	GND	
A13	ADC_ANALOG_2	Analog input 2: 0 ... 3.3 V Slave address, see <a href="#">Setting slave address and baud rate</a>
A14	GND	
A15	SLOT_SPI_MOSI	
A16	SLOT_SPI_MISO	<i>Modbus RX Data</i>
A17	SLOT_SPI_SCK	<i>Modbus TX Data</i>
A18	SLOT_SPI_C $\bar{S}$	
A19	COMM_SPI_MOSI	<i>Receiver Enable</i>
A20	COMM_SPI_MISO	
A21	COMM_SPI_SCK	
A22	COMM_SPI_C $\bar{S}$	
A23	I2CSCL_CANRX	
A24	I2CSDA_CANTX	
A25	n.c.	reserved
A26	GND	
A27	+3.3V_EXT	Not used
A28	+14V_EXT	Not used
A29	GND	
A30	BN_OUT	B\ (stepper motor)
A31		
A32		
A33	B_OUT	B\ (stepper motor) or W (BLDC)
A34		
A35		
A36	AN_OUT	A\ (stepper motor) or V (BLDC)
A37		
A38		
A39	A_OUT	A (stepper motor) or U (BLDC)
A40		
A41		
A42	GND	
A43		
A44		
A45	UB_IN	12 ... 48 V DC $\pm$ 4%
A46		
A47		
A48	BRAKE_OUT	Control of the external brake, open-drain output, max. 1 A
A49	GND	

Pin	Name	Description/function
B1	GND	
B2	U_REF_ANALOG	3.3 V DC, reference voltage for analog inputs Slave address, see <a href="#">Setting slave address and baud rate</a>
B3	DIO1_IO_CS	General I/O
B4	DIO2_CD_CLK	General I/O (clock input in <a href="#">clock-direction mode</a> )
B5	DIO3_CD_DIR	General I/O (direction input in <a href="#">clock-direction mode</a> )
B6	DIO4_IO_MOSI	General I/O
B7	DIO5_IO_MISO	General I/O
B8	DIO6_IO_CLK	General I/O
B9	ENC2_I	Encoder 2, Index
B10	ENC2_CAP	Not used
B11	GND	
B12	GND	
B13	ADC_ANALOG_1	Analog input 1: 0 ... 3.3 V
B14	GND	
B15	SPARE_PHY_TX+	reserved
B16	SPARE_PHY_TX-	reserved
B17	SPARE_PHY_RX+	reserved
B18	SPARE_PHY_RX-	reserved
B19	SLOT_RESET	System function, reserved
B20	SLOT_BOOT	System function, reserved
B21	SLOT_SYNC	System function, reserved
B22	COMM_RESET	<i>Driver Enable</i>
B23	COMM_SYNC	
B24	GND	
B25	n.c.	reserved
B26	GND	
B27	BALLAST	For controlling an <a href="#">external ballast circuit</a>
B28	n.c.	reserved
B29	GND	
B30	BN_OUT	B\ (stepper motor)
B31		
B32		
B33	B_OUT	B (stepper motor) or W (BLDC)
B34		
B35		
B36	AN_OUT	A\ (stepper motor) or V (BLDC)
B37		
B38		
B39	A_OUT	A (stepper motor) or U (BLDC)

Pin	Name	Description/function
B40		
B41		
B42	GND	
B43		
B44		
B45	UB_IN	12 ... 48 V DC ±4%
B46		
B47		
B48	UB_LOGIK	External logic supply, 24 V DC
B49	GND	



## 4 Hardware installation



### Note

Make certain that all components are de-energized.



### Note

- 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

For easy connection, Nanotec recommends the *Discovery Board DK-NP5-4A*. If you operate your controller using this *Discovery Board*, read the chapter [Connecting the NP5 controller via the Discovery Board](#).

#### 4.1.1 Integrating the NP5



### Note

**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:

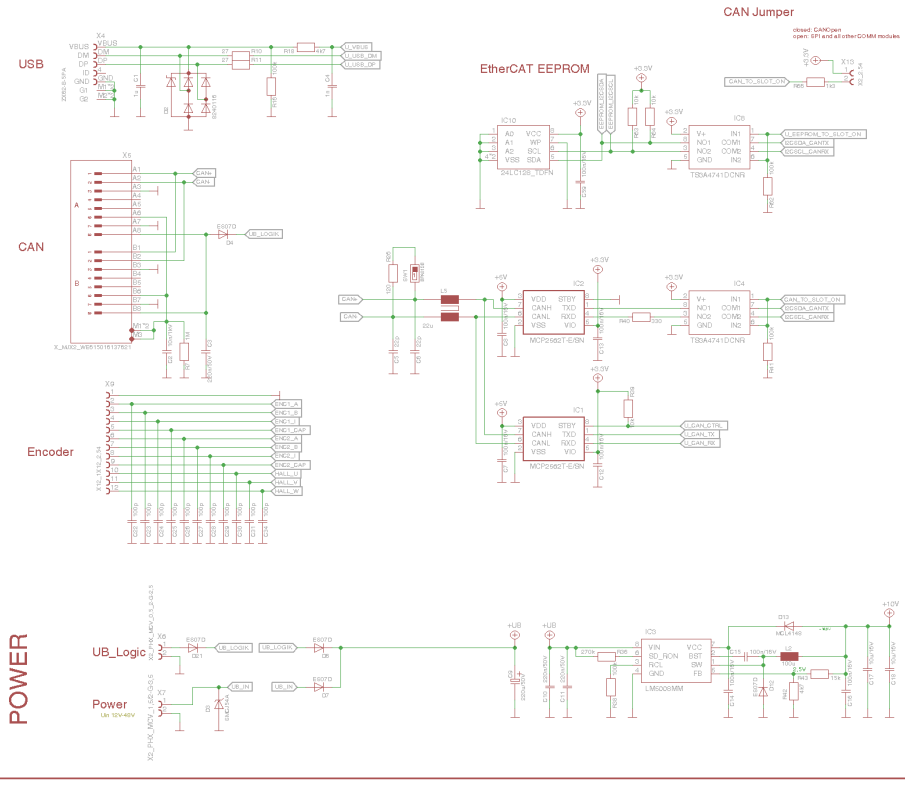
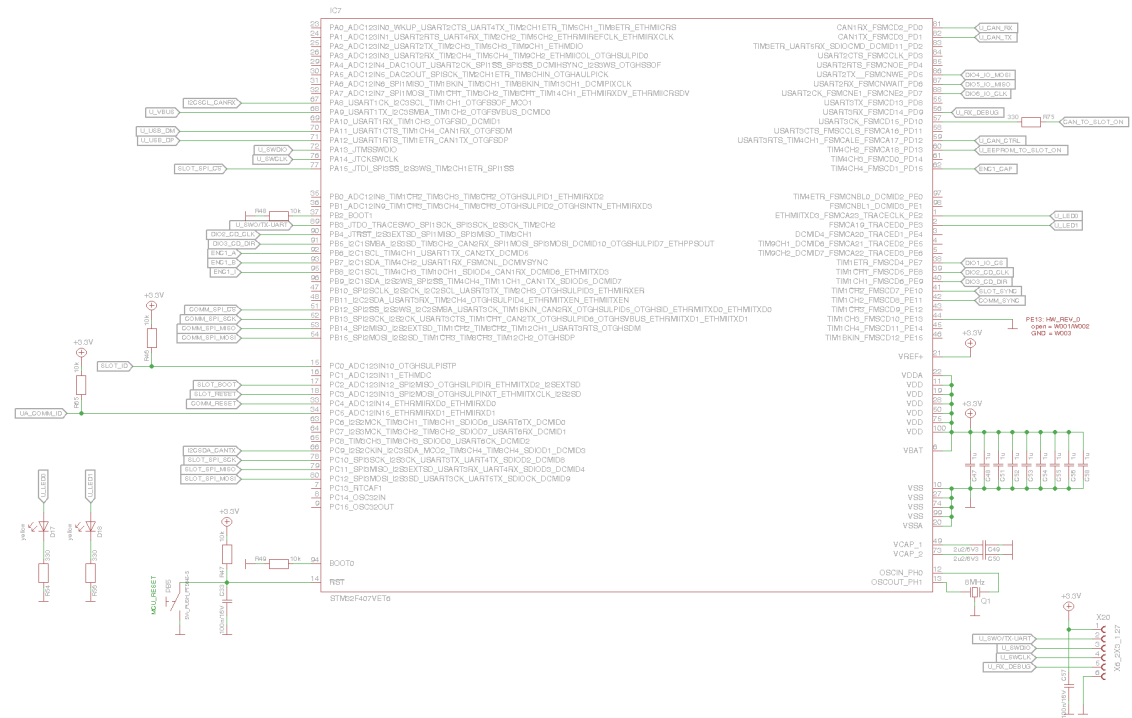
- ▶ Use shielded cables and earth the cable shielding on both ends over a short distance.
- ▶ Keep power supply and motor cables as short as possible.
- ▶ Use cables with cores in twisted pairs.
- ▶ Earth motor housing with large contact area over a short distance.
- ▶ Lay supply, motor and control cables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

Shown in the following figures is the circuit diagram of the *NP5 Discovery Board*, which can serve as a reference for the development of your own motherboard. You can find the pin assignment of the PCI connector strip in chapter [Pin assignment](#).

#### 1. Prepare your 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 (*POWER*) of the motor and of an RS485 transceiver (see also [Connecting Modbus RTU](#)) is sufficient.









**Note**

For the standard assignment of the connections, see [Pin assignment](#).

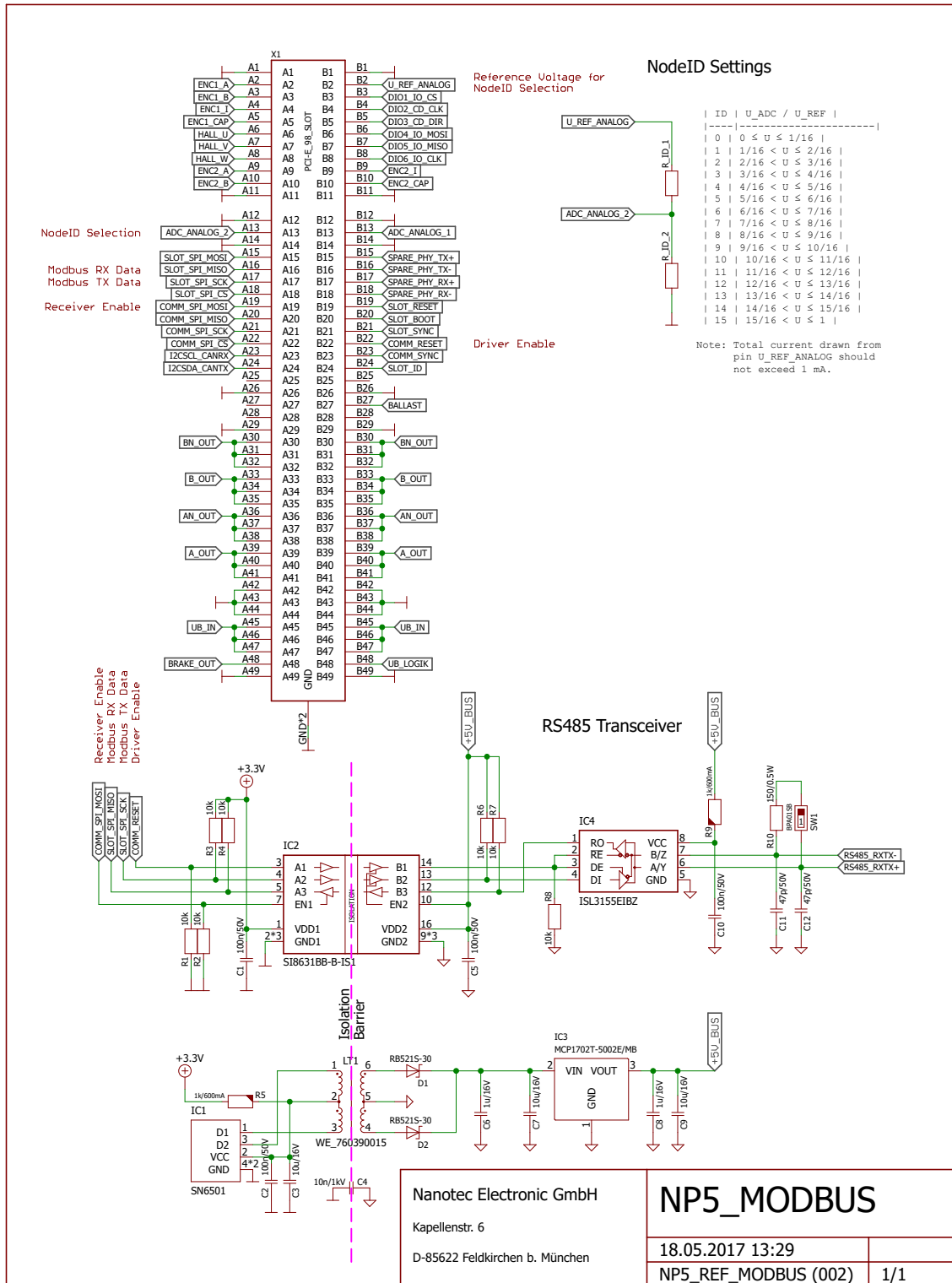


Figure 1:

PCI-specific pin assignment for Modbus RTU:

Nanotec Electronic GmbH	
Kapellenstr. 6	
D-85622 Feldkirchen b. München	
<b>NP5_MODBUS</b>	
18.05.2017 13:29	
NP5_REF_MODBUS (002)	1/1

Pin	Name	Description
A13	ADC_ANALOG_2	The slave address is defined via the applied voltage, see <a href="#">Setting slave address and baud rate</a>
A16	SLOT_SPI_MISO	<i>Modbus RX Data</i>
A17	SLOT_SPI_SCK	<i>Modbus TX Data</i>
A19	COMM_SPI_MOSI	<i>Receiver Enable</i>
B2	U_REF_ANALOG	Is used as a reference voltage of 3.3 V DC to define the slave address, see <a href="#">Setting slave address and baud rate</a> .
B22	COMM_RESET	<i>Driver Enable</i>

### Tip



If you do not use the ADC\_ANALOG\_2 analog input to set the address and baud rate, connect pin A13 to GND.

### 4.1.3 RS-485 line polarization

### Note



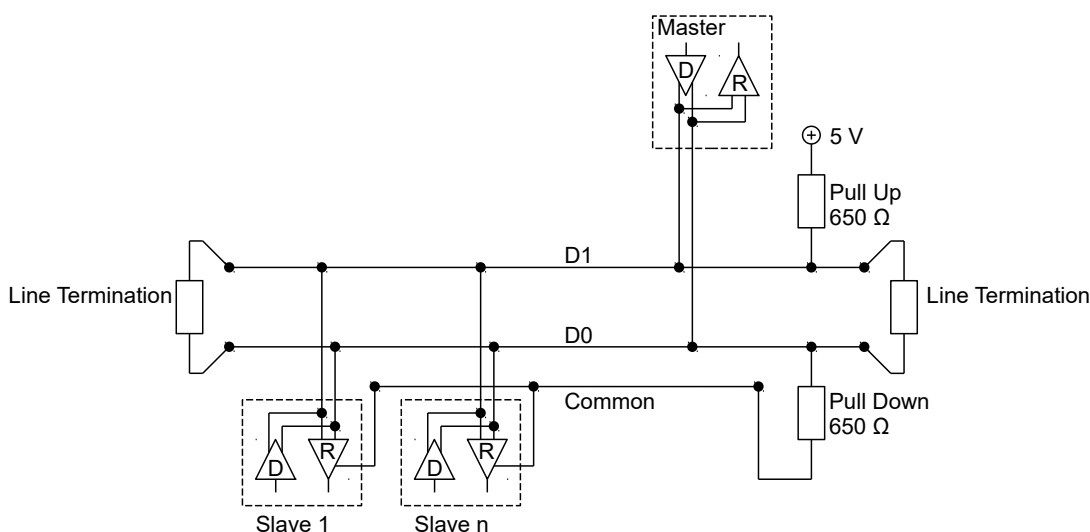
The controller is **not** equipped with line polarization and expects the master device to have one.

If the master device on the bus does not have line polarization of its own, a pair of resistors must be attached to the RS-485 balanced cables:

- A pull-up resistor to a 5V voltage on the RS-485+ (D1) cable
- A pull-down resistor to earth (GND) on the RS-485- (D0) cable

The value of these resistors must be between 450 ohm and 650 ohm. A 650 ohm resistor permits a higher number of devices on the bus.

In this case, a line polarization must be attached at a location for the entire serial bus. In general, this location should be on the master device or its connection. All other devices then no longer need to implement line polarization.



#### 4.1.4 Connecting the NP5 controller via the *Discovery Board*

The *NP5 Discover Board* helps you during tests and during the evaluation of the *NP5* controller.

The connectors necessary for the boards are supplied already installed.

Jumper X13 must be set if CANopen (*NP5-08*) is used; otherwise, you must remove it.

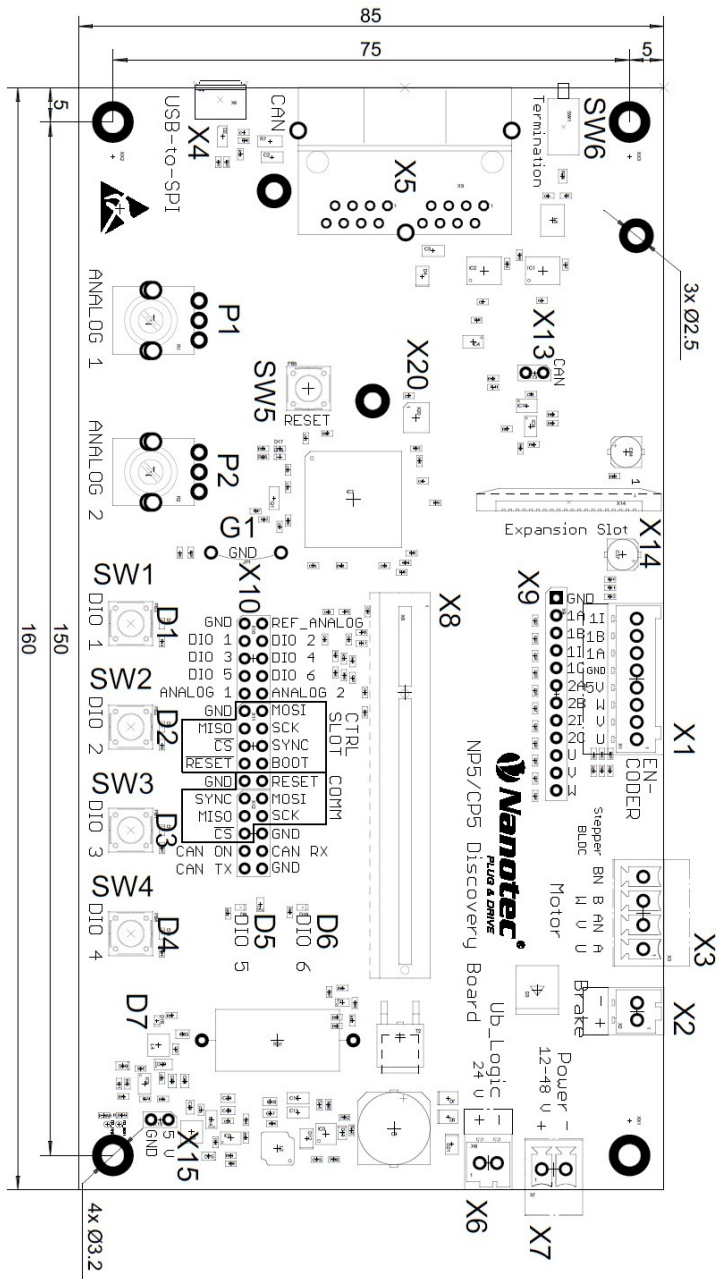
##### 4.1.4.1 Technical data – *NP5 Discovery Board*

Property	Description / value
Operating voltage +UB:	12 ... 48 V DC $\pm 5\%$
Logic voltage +UB_Logic:	24 V DC $\pm 5\%$
Current consumption +UB:	Max. 100 mA (without connected NP5)
Current consumption +UB_Logic:	Max. 100 mA (without connected NP5)
Communication interface:	SPI, CANopen, Modbus RTU
Analog reference voltage:	3.3 V DC $\pm 5\%$ , max. 10 mA
Digital input voltage:	Max. 3.3 V DC
DC output voltage:	5 V DC $\pm 3\%$ , max. 300 mA
Status indicator:	4x green LEDs for GPIO 1 to 4 2x blue LEDs for GPIO 5 and GPIO 6 1x green LED for Discovery Board (+3.3 V DC)
<u>Ballast resistor:</u>	15 $\Omega$ /5 W
Mounting holes:	4x $\varnothing$ 3.2 mm for Discovery Board
Additional board Modbus RTU:	3x $\varnothing$ 2.5 mm
Weight:	0.12 kg

##### 4.1.4.2 Dimensioned drawings – *NP5 Discovery Board*

Dimensions are in [mm].





4.1.4.3 Pin assignment – NP5 Discovery Board

Connector	Function
X1	Encoder 1 and Hall sensor
X2	Brake
X3	Motor
X5	CANopen
X6	Logic voltage
X7	Voltage supply
X8	Slot for NP5 controller, see also <a href="#">Dimensioned drawings</a> and <a href="#">Pin assignment</a>
X9	Encoder 1/2 and Hall sensor
X10	GPIO and communication interface
X13	Jumper for activating / deactivating the CANopen communication
X14	Ribbon cable socket for Modbus RTU additional board

Connector	Function
X15	+5 V DC output
P1	Potentiometer for analog input 1
P2	Potentiometer for analog input 2 (for setting the slave address and baud rate for Modbus RTU, see also <a href="#">P2 - Analog input 2</a> )
SW1 to SW4	Buttons for GPIO 1 to GPIO 4
SW5	Reset button for the <i>Discovery Board</i>
SW6	Switch for 120 ohm termination resistor (CANopen)
D1 to D6	Status indicator for GPIO 1 to GPIO 6
D7	Status indicator for the <i>Discovery Board</i> (+3.3 V DC)
G1	Earth connection

### Connector X1 – encoder 1 and Hall sensor

Connector X1 has the following features:

- Connector type: Phoenix base strip. MCV-0.5/8-G-2.5
- Voltage level: +5 V logic level
- Maximum admissible current: Max. 300 mA (together with +5 V DC output voltage on pin header X15)
- Hall inputs: Internally by means of 2.7 kΩ pull-up resistor connected to +5 V DC

Pin	Name / function
1	Hall_U (H1)
2	Hall_V (H2)
3	Hall_W (H3)
4	+5 V DC
5	GND
6	ENC1_A
7	ENC1_B
8	ENC1_I

### Connector X2 – brake

Connector X2 has the following features:

- Connector type: Phoenix base strip. MCV-0.5/2-G-2.5

Pin	Name / function
1	Brake + (connected with +UB)
2	Brake – (PWM-controlled open-drain output, max 1.5 A)

### Connector X3 – motor

Connector X3 has the following features:

- Connector type: Phoenix base strip. MCV-1.5/4-G-3.5
- Max. rated current 6 A RMS
- Max. peak current 10 A RMS (for 1 s)

Pin	Stepper motor	BLDC motor
1	A	U
2	A\	V
3	B	W
4	B\	

### Connector X5 – CANopen

Connector X5 has the following features:

- Connector type: RJ45 Duo Port, horizontal

Pin	Name / function
1	CAN_H
2	CAN_L
3	GND
4	N.C
5	N.C
6	CAN_Shield
7	GND
8	+UB_Logic (24 V DC $\pm 5\%$ )

### Connector X6 – logic voltage

Connector X6 has the following features:

- Connector type: Phoenix base strip. MCV-0.5/2-G-2.5

Pin	Name / function
1	+UB_Logic (24 V DC $\pm 5\%$ )
2	GND

### Connector X7 – operating voltage

Connector X7 has the following features:

- Connector type: Phoenix base strip. MCV-1.5/2-G-3.5

Pin	Name / function
1	+UB (12 ... 48 V DC $\pm 5\%$ )
2	GND

### Connector X9 – encoder and Hall sensors

Connector X9 has the following features:

- Connector type: Pin header, single row, RM 2.54 mm, 12-pin, vertical
- Voltage level: +5 V DC logic level

Pin	Name / function
1	GND
2	ENC1_A
3	ENC1_B
4	ENC1_I
5	ENC1_CAP
6	ENC2_A
7	ENC2_B
8	ENC2_I
9	ENC2_CAP
10	Hall_U (H1)
11	Hall_V (H2)
12	Hall_W (H3)

### Connector X10 – I/O and communication interface

Connector X10 has the following features:

- Connector type: Pin header, two rows, RM 2.54 mm, 2x 15-pin, vertical

Pin	Name	Type	Note
1	GND	Earth	
2	U_REF_ANALOG	Out	Analog reference voltage
3	DIO1_IO_CS	I/O	General I/O
4	DIO2_CD_CLK	I/O	General I/O
5	DIO3_CD_DIR	I/O	General I/O
6	DIO4_IO_MOSI	I/O	General I/O
7	DIO5_IO_MISO	I/O	General I/O
8	DIO6_IO_CLK	I/O	General I/O
9	ADC_ANALOG_1	In	AD converter 1
10	ADC_ANALOG_2	In	AD converter 2
11	GND	Earth	
12	SLOT_SPI_MOSI	-	SPI 1
13	SLOT_SPI_MISO	-	SPI 1
14	SLOT_SPI_SCK	-	SPI 1
15	SLOT_SPI_CS	-	SPI 1
16	SLOT_SYNC	-	System function, reserved
17	SLOT_RESET	-	System function, reserved
18	SLOT_BOOT	-	System function, reserved
19	GND	Earth	
20	COMM_RESET	-	System function, reserved
21	COMM_SYNC	-	System function, reserved
22	COMM_SPI_MOSI	-	SPI 2
23	COMM_SPI_MISO	-	SPI 2
24	COMM_SPI_SCK	-	SPI 2
25	COMM_SPI_CS	-	SPI 2

Pin	Name	Type	Note
26	GND	Earth	
27	CANopen ON	-	CANopen ON
28	I2CSCL_CANRX	-	I <sup>2</sup> C Clock or CANopen RX
29	I2CSDA_CANTX	-	I <sup>2</sup> C Data or CANopen TX
30	GND	Earth	

### Connector X13 – jumper for activating / deactivating the CANopen communication

Connector X13 has the following features:

- Connector type: Pin header, RM 2.54 mm, 2-pin, vertical
- Bridged with jumper: CANopen activated
- Not bridged with jumper: CANopen deactivated, Modbus RTU activated

Pin	Name / function
1	+3.3V
2	CANopen ON

### Connector X15 – +5 V DC output

Connector X15 has the following features:

- Connector type: Pin header, RM 2.54 mm, 2-pin, vertical
- Maximum admissible current: Max. 300 mA (together with +5 V DC output voltage on pin header X1)

Pin	Name / function
1	+5 V DC
2	GND

## P2 - Analog input 2

The slave address and the baud rate are defined via analog input 2. You can find further details in chapter [Setting slave address and baud rate](#).

To use the factory settings, set the potentiometer to 0 (turn counterclockwise as far as it will go). The factory settings are:

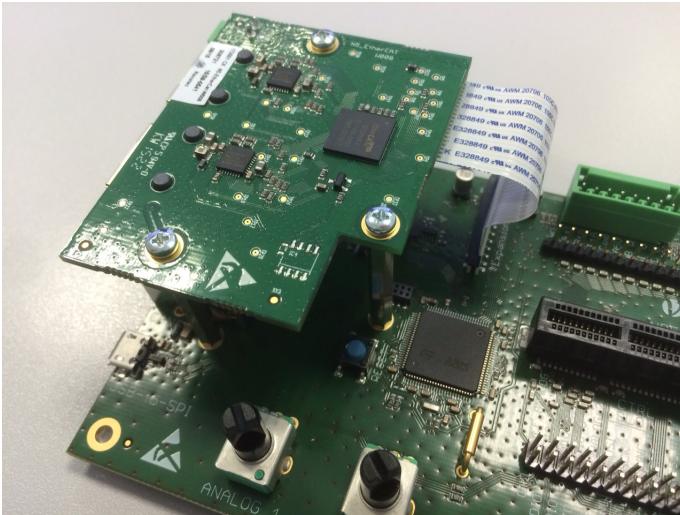
Configuration	Object	Factory settings
Slave address	2028 <sub>h</sub>	5
Baud rate	202A <sub>h</sub>	19200
Parity	202D <sub>h</sub>	0x04 (Even)

The number of data bits is always "8" here. The number of stop bits is dependent on the parity setting:

- No parity: 2 stop bits
- "Even" or "Odd" parity: 1 stop bit

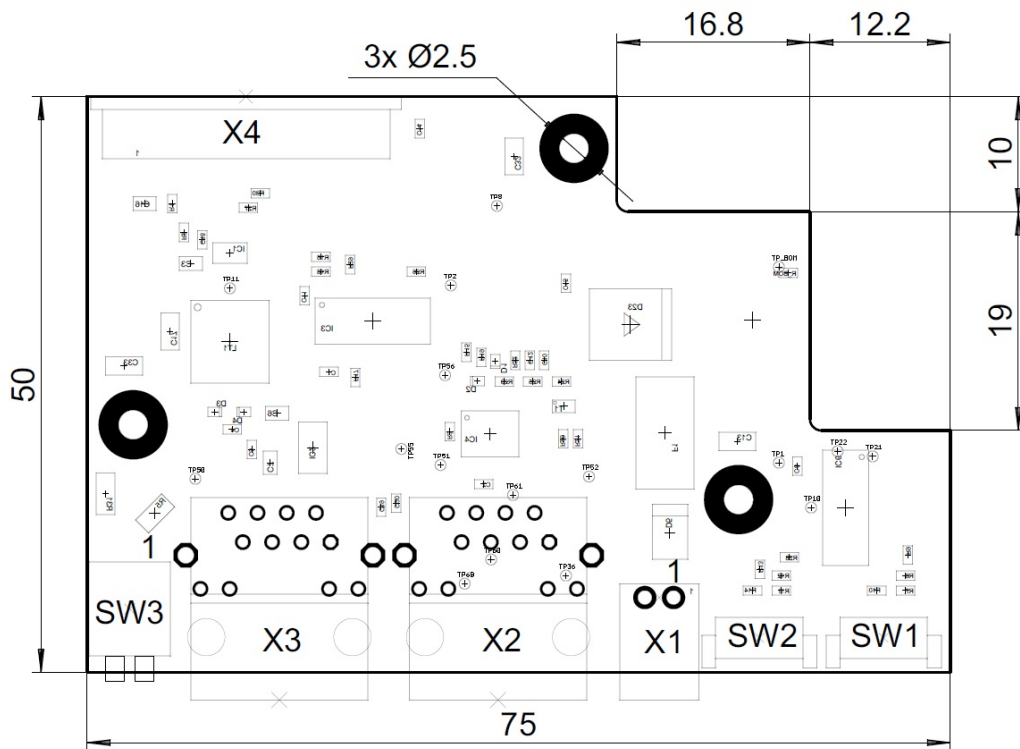
#### 4.1.4.4 Extension for Modbus RTU (additional board)

Discovery Board DK-NP5-4A is equipped with an additional board for communication via Modbus RTU (RS485).



**Dimensioned drawings – Modbus RTU additional board**

Dimensions are in [mm].



**Hardware overview – Modbus RTU additional board**

Name	Function	Note
X2	Modbus RTU IN	
X3	Modbus RTU OUT	
X4	Connection to <i>Discovery Board NP5</i>	
SW1	Reserved	
SW2	Reserved	
SW3	Switch for 150 Ω termination resistor	

Name	Function	Note
Pin 1		150 $\Omega$ termination resistor (switch set to ON)
Pin 2		Reserved

PIN	Function	Note
1	n.c.	
2	n.c.	
3	n.c.	
4	D1 (RS-485 +)	
5	D0 (RS-485 -)	
6	n.c.	
7	n.c.	PIN 7 on X2 is internally connected to PIN 7 on X3
8	Common	Common signal and supply

#### 4.1.4.5 Commissioning the Modbus RTU via the *Discovery Board*

To establish a connection with the *NP5-02*, proceed as follows:

1. Plug in the *NP5-02* at X8.
2. Unplug jumper X13.
3. Set potentiometer P2 to 0 (counterclockwise as far as it will go)
4. Switch on the termination resistor (switch SW3 of the Modbus board to ON).
5. Connect your RS485 cable to X2 of the Modbus board.
6. Connect your supply voltage to X7.



## 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.

The *Plug & Drive Studio* software offers a convenient option for performing the configuration and adapting the controller to the connected motor. You can find further information in document *Plug & Drive Studio: Quick Start Guide* at [us.nanotec.com](http://us.nanotec.com).

### 5.1 Communication settings

#### 5.1.1 Modbus RTU

Described in the following chapters is how you can change the communication settings.

The controller is set at the factory to slave address 5, baud rate 19200 baud, even parity, 1 stop bit.

##### 5.1.1.1 Setting slave address and baud rate

Slave address, baud rate and parity are dependent on the position of a *virtual rotary switch* and, if applicable, on objects 2028h MODBUS Slave Address, 202Ah MODBUS RTU Baudrate and 202Dh MODBUS RTU Parity. The position of the *virtual rotary switch* is dependent on the analog value at input ADC\_ANALOG\_2 (see Pin assignment).

Described in the following table is how the position of the *virtual rotary switch* is composed of the analog value applied at ADC\_ANALOG\_2 and the voltage divider:

Position of the virtual rotary switch	Upper limit digit	Formula for the voltage divider calculation
0	64	$\frac{32}{1023} \times U\_REF\_ANALOG$
1	128	$\frac{96}{1023} \times U\_REF\_ANALOG$
2	192	$\frac{160}{1023} \times U\_REF\_ANALOG$
3	256	$\frac{224}{1023} \times U\_REF\_ANALOG$
4	320	$\frac{288}{1023} \times U\_REF\_ANALOG$
5	384	$\frac{352}{1023} \times U\_REF\_ANALOG$

Position of the virtual rotary switch	Upper limit digit	Formula for the voltage divider calculation
6	448	$\frac{416}{1023} \times U\_REF\_ANALOG$
7	512	$\frac{480}{1023} \times U\_REF\_ANALOG$
8	576	$\frac{544}{1023} \times U\_REF\_ANALOG$
9	640	$\frac{608}{1023} \times U\_REF\_ANALOG$
10	704	$\frac{672}{1023} \times U\_REF\_ANALOG$
11	768	$\frac{736}{1023} \times U\_REF\_ANALOG$
12	832	$\frac{800}{1023} \times U\_REF\_ANALOG$
13	896	$\frac{864}{1023} \times U\_REF\_ANALOG$
14	960	$\frac{928}{1023} \times U\_REF\_ANALOG$
15	1023	$\frac{992}{1023} \times U\_REF\_ANALOG$

<b>Note</b>
-------------

Observe the following for your voltage divider calculation:



- If you do not want to use the analog input for setting the address, connect the ADC\_ANALOG\_2 pin to earth or U\_REF\_ANALOG.
- The table shows suggestions for the ratio of the voltage divider; the concrete values of all resistors are to be calculated.
- U\_REF\_ANALOG is the reference voltage for the analog input, 3.3 V, see [Pin assignment](#).
- The current consumption of the voltage divider must not exceed 1 mA.

Described in the following is how you define the slave address, baud rate and parity via the *virtual rotary switch*.

Position of the virtual rotary switch	Slave address	Baud rate and parity
0	Object <u>2028</u> <sub>hh</sub>	Object <u>202A</u> <sub>h</sub> or <u>202D</u> <sub>h</sub>
1 ... 14	Number of the rotary switch	Object <u>202A</u> <sub>h</sub> or <u>202D</u> <sub>h</sub>
15	5	19200, even Parity

Proceed as follows to define the position of the *virtual rotary switch*.

1. Calculate the resistances for your voltage divider. You can find an example in the circuit diagram in chapter [Connecting Modbus RTU](#). The reference voltage U\_REF\_ANALOG (pin B2, see [Pin assignment](#)) is 3.3 V DC.
2. Wire input ADC\_ANALOG\_2 to the voltage divider.

### 5.1.1.2 Communication settings

The following settings can be performed:

Configuration	Object	Value range	Factory settings
Slave address	<u>2028</u> <sub>h</sub>	1 to 247	5
Baud rate	<u>202A</u> <sub>h</sub>	7200 to 256000	19200
Parity	<u>202D</u> <sub>h</sub>	<ul style="list-style-type: none"> <li>■ None: 0x00</li> <li>■ Even: 0x04</li> <li>■ Odd: 0x06</li> </ul>	0x04 (Even)

The number of data bits is always "8" here. The number of stop bits is dependent on the parity setting:

- No parity: 2 stop bits
- "Even" or "Odd" parity: 1 stop bit

The following baud rates are supported:

- 7200
- 9600
- 14400
- 19200
- 38400
- 56000
- 57600

- 115200
- 128000
- 256000

The slave address and baud rate can also be defined via the analog input, see [Setting slave address and baud rate](#)

You must save the changes by writing value "65766173<sub>h</sub>" in object 1010<sub>h</sub>:0B<sub>h</sub>. The changes are not taken over until after the controller has been restarted.

## 5.2 Establishing communication

### 5.2.1 Modbus RTU

1. Connect the *Modbus master* to the controller via the RS-485+ and RS-485- cables.
2. Supply the controller with voltage.
3. Change the configuration values if necessary.

The controller is set to slave address 5 ex works, baud rate 19200 baud, even parity, 1 stop bit.

4. To test the interface, send bytes 05 65 55 00 2F A7 to the controller (you can find a detailed description of the Modbus function codes in chapter [Modbus RTU](#)).  
The object dictionary is read out.

## 5.3 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<sub>h</sub>:00<sub>h</sub> (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<sub>h</sub>:00<sub>h</sub>: maximum permissible motor current (motor protection) in mA (see motor data sheet)
- Object 6075<sub>h</sub>:00<sub>h</sub>: rated current of the motor in mA (see motor data sheet), limited by 2031<sub>h</sub>
- Object 6073<sub>h</sub>:00<sub>h</sub>: 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<sub>h</sub>. Is limited by 2031<sub>h</sub>.
- Object 203B<sub>h</sub>:02<sub>h</sub> Maximum duration of the maximum current (6073<sub>h</sub>) 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<sub>h</sub>:00<sub>h</sub> (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<sub>h</sub>:00<sub>h</sub> (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](#).
- Motor with brake: Object 3202<sub>h</sub>:00<sub>h</sub> (Motor Drive Submode Select): The brake control is activated for the initial commissioning. Depending on the specific application, this configuration can be deactivated later if necessary. One of the following values is to be entered depending on the motor type:
  - Stepper motor, brake control (and [current reduction](#)) activated: 0000000Ch
  - BLDC motor, brake control activated: 00000044h

### Note



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.  $\sqrt{2}$  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.

## 5.4 Connecting the motor

After setting the motor parameters, see [Setting the motor data](#), connect the motor and, if applicable, the present sensors (encoders / Hall sensors) and the brake.

### Note



**Damage to the electronics if motor is connected incorrectly!**

- ▶ Observe the PIN assignment in chapter [Pin assignment](#) and the motor data sheet.

- Connect the motor:
  - to the corresponding pins of the PCI connector strip, see [Pin assignment](#)
  - to connector X3 of the Discovery Board, if it is used; see [Connector X3 – motor](#)
- Connect encoders / Hall sensors:
  - to the corresponding pins of the PCI connector strip, see [Pin assignment](#)
  - to connector X1 of the Discovery Board, if it is used; see [Connector X1 – encoder 1 and Hall sensor](#)
- Connect the brake:
  - negative to pin A48 of the PCI connector strip, see [Pin assignment](#)
  - positive to UB\_IN of the PCI connector strip or directly to the voltage supply, see [Pin assignment](#)
  - to connector X2 of the Discovery Board, if it is used; see [Connector X2 – brake](#)

How the automatic brake control can be activated is described in chapter [Automatic brake control](#).

## 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.

### Note



**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<sub>h</sub>:00<sub>h</sub> 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	✓

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)	-	✓	---

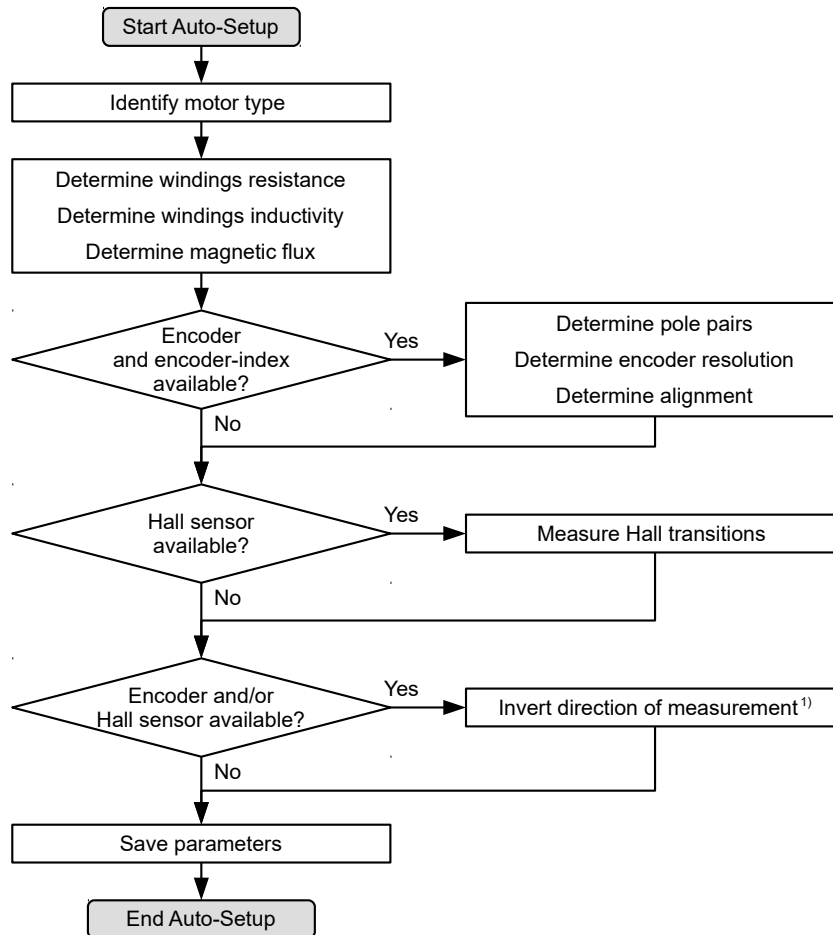
Parameter	Motor without Hall sensor	Motor with Hall sensor
Hall transitions	-	✓

### 5.5.2 Execution

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

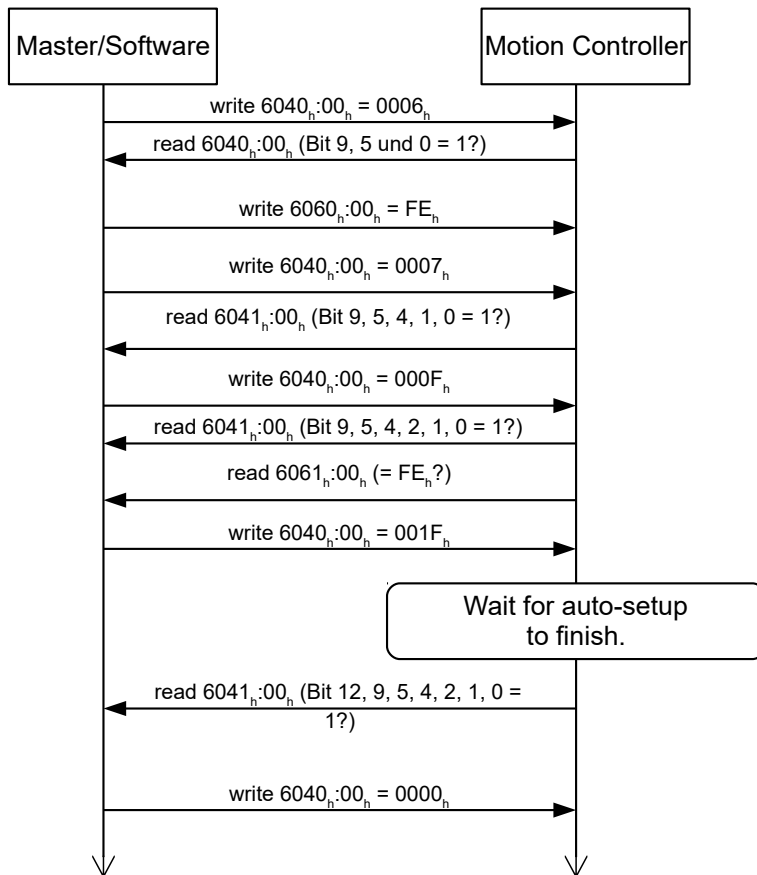
1. To preselect the *auto setup* operating mode, enter the value "-2" ("FE<sub>h</sub>") in object 6060<sub>h</sub>:00<sub>h</sub>. The *power state machine* must now switch to the *Operation enabled* state, see [CiA 402 Power State Machine](#).
2. Start *auto setup* by setting bit 4 *OMS* in object 6040<sub>h</sub>:00<sub>h</sub> (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<sub>h</sub>:00<sub>h</sub> (statusword) indicates that the auto setup was completely executed and ended. In addition, bit 10 *TARG* in object 6041<sub>h</sub>:00<sub>h</sub> 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* 1010h:05h and *Tuning* 1010h:06h 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
3380h	Sensorless	Contains measurement and configuration values for sensorless control
3390h	Hall sensor (digital)	contains configuration values for the Hall sensors
33A0h	Incremental encoder 1	contains configuration values for the first incremental encoder



Object	Feedback	Description
<u>33A1<sub>h</sub></u>	Incremental encoder 2	contains configuration values for the second incremental encoder

#### Note



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<sub>h</sub>, 60E6<sub>h</sub> and 60EB<sub>h</sub>) (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<sub>h</sub> and 60ED<sub>h</sub>) and/or the feed constant (objects 60E9<sub>h</sub> and 60EE<sub>h</sub>) (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
<u>60E6<sub>h</sub></u> Additional Position Encoder Resolution - Encoder Increments	1256637
<u>60EB<sub>h</sub></u> Additional Position Encoder Resolution - Motor Revolutions	20
<u>60E8<sub>h</sub></u> Additional Gear Ratio - Motor Shaft Revolutions	4
<u>60ED<sub>h</sub></u> Additional Gear Ratio - Driving Shaft Revolutions	1
<u>60E9<sub>h</sub></u> Additional Feed Constant - Feed	2513274 incr. (corresponds to 1256.637 mm)
<u>60EE<sub>h</sub></u> 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 3203<sub>h</sub> 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](#).

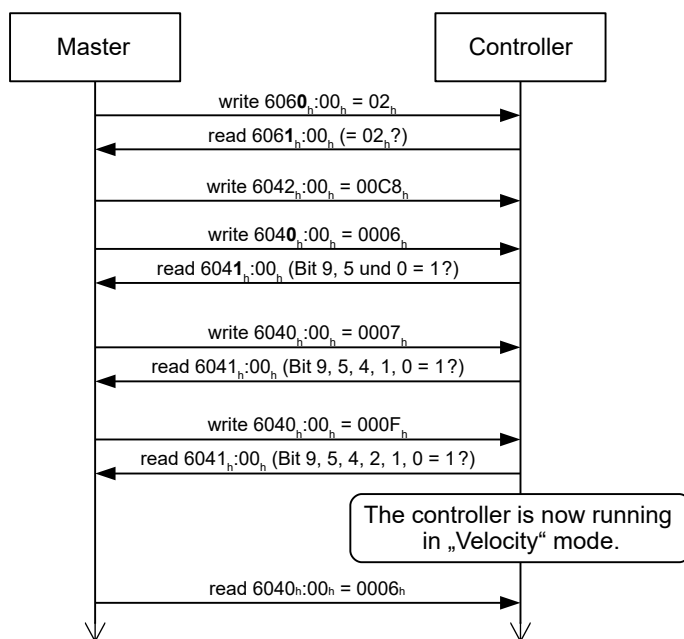
## 5.7 Test run

After configuring and the auto setup, a test run can be performed. As an example, the [Velocity](#) operating mode is used.

The values are transferred from your *Modbus master* to the controller. After every transfer, the *master* should use the status objects of the controller to ensure successful parameterization.

1. Select the *Velocity* mode by setting object [6060<sub>h</sub>](#) (Modes Of Operation) to the value "2".
2. Write the desired speed in [6042<sub>h</sub>](#).
3. Switch the *power state machine* to the *Operation enabled* state, see [CiA 402 Power State Machine](#).

The following sequence starts *Velocity* mode; the motor turns at 200 rpm.



4. To stop the motor, set controlword ([6040<sub>h</sub>](#)) to "6".

## 6 General concepts

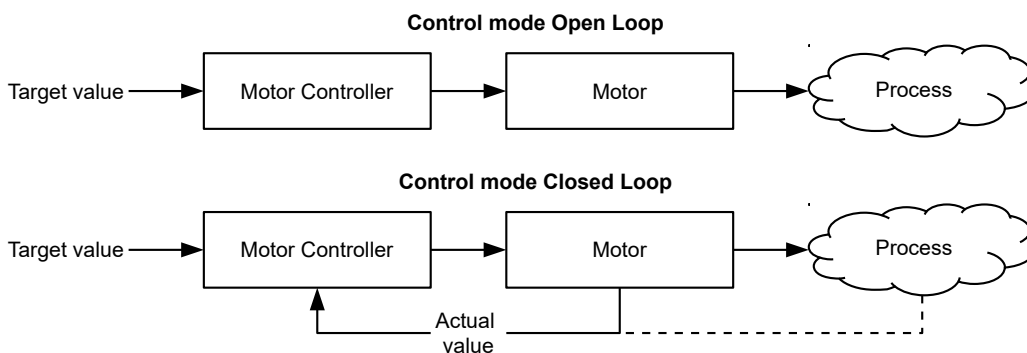
### 6.1 Control modes

#### 6.1.1 General

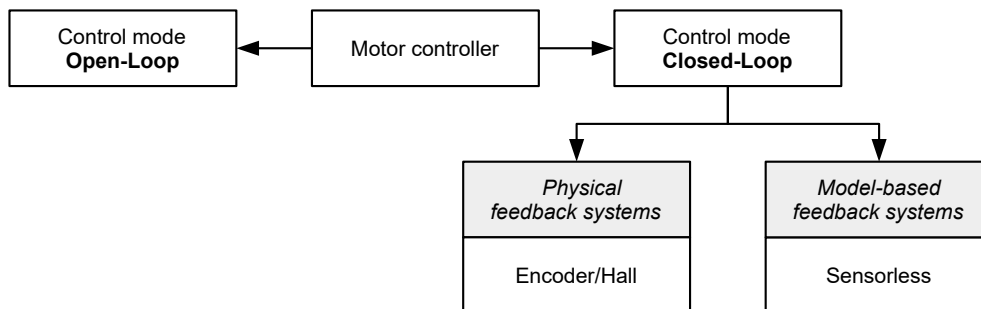
The control mode of systems without feedback is called *open loop*, the mode with feedback is called *closed loop*. In the *closed loop* control mode, it is initially irrelevant whether the fed back signals come from the motor itself or from the influenced process.

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 <sup>1)</sup>	yes	no
Homing	yes <sup>2)</sup>	yes	yes
Interpolated Position Mode	yes <sup>3)</sup>	yes	yes
Cyclic Synchronous Position	yes <sup>3)</sup>	yes	yes
Cyclic Synchronous Velocity	yes <sup>3)</sup>	yes	yes
Cyclic Synchronous Torque	no <sup>1)</sup>	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<sub>h</sub> (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<sub>h</sub>:00<sub>h</sub>, enter the maximum permissible motor current (motor protection) in mA (see motor data sheet)
- In object 6075<sub>h</sub>:00<sub>h</sub>, enter the rated current of the motor in mA (see motor data sheet).
- In object 6073<sub>h</sub>:00<sub>h</sub>, 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<sub>h</sub>. A value greater than "1000" is limited internally to "1000".
- In object 3202<sub>h</sub> (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<sub>h</sub> (Motor Drive Submode Select), set bit 3 (CurRed) to "1".
- In object 2036<sub>h</sub> (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<sub>h</sub> (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<sub>h</sub> and 6075<sub>h</sub>, 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<sub>h</sub>:09<sub>h</sub> (I\_P) and 3210<sub>h</sub>:0A<sub>h</sub> (I\_I) or 320F<sub>h</sub> (generally not necessary).

The current controller operates optimally if the actual current of both windings (square root of the sum  $I_a^2 + I_b^2$ , 2039<sub>h</sub>:03<sub>h</sub>:04<sub>h</sub>) divided by 2 at any point in time corresponds to the set rated current (203B<sub>h</sub>:01<sub>h</sub>).

- Adjustments to the acceleration, deceleration and/or target speed depending on the selected control mode:

#### Profile Position operating mode

Objects 6083<sub>h</sub> (Profile Acceleration), 6084<sub>h</sub> (Profile Deceleration) and 6081<sub>h</sub> (Profile Velocity).

#### Velocity operating mode

Objects 6048<sub>h</sub> (Velocity Acceleration), 6049<sub>h</sub> (Velocity Deceleration) and 6042<sub>h</sub> (Target Velocity).

#### Profile Velocity operating mode

Objects 6083<sub>h</sub> (Profile Acceleration), 6084<sub>h</sub> (Profile Deceleration) and 6081<sub>h</sub> (Profile Velocity).

#### Homing operating mode

Objects 609A<sub>h</sub> (Homing Acceleration), 6099<sub>h</sub>:01<sub>h</sub> (Speed During Search For Switch) and 6099<sub>h</sub>:02<sub>h</sub> (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.

### Clock-direction operating mode

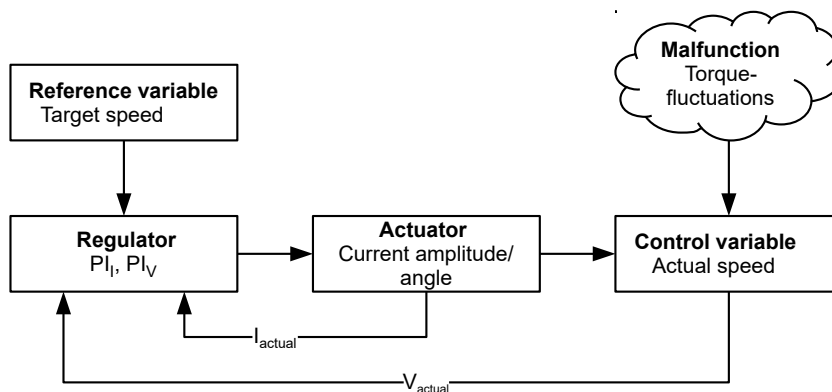
Change of the step resolution via objects 2057<sub>h</sub> (Clock Direction Multiplier) and 2058<sub>h</sub> (Clock Direction Divider). Optimize acceleration / deceleration ramps by adjusting the pulse frequency to pass through the resonance range as quickly as possible.

## 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"

#### Note

For firmware versions from FIR-v19xx upwards, the new schema described here for the Controller structure applies.

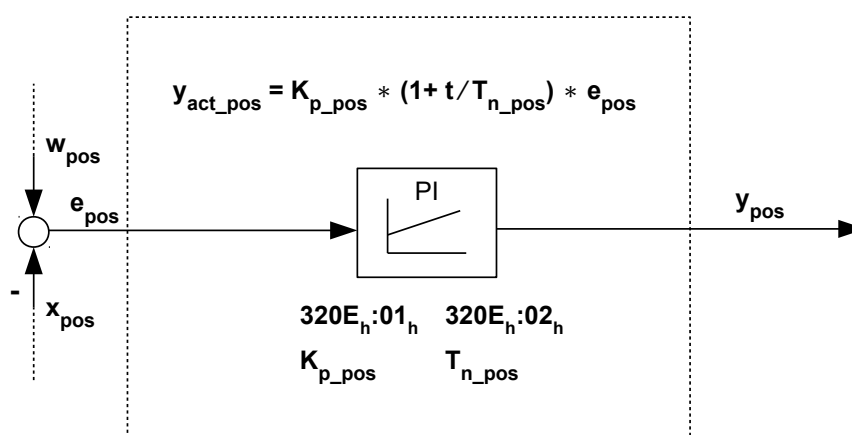


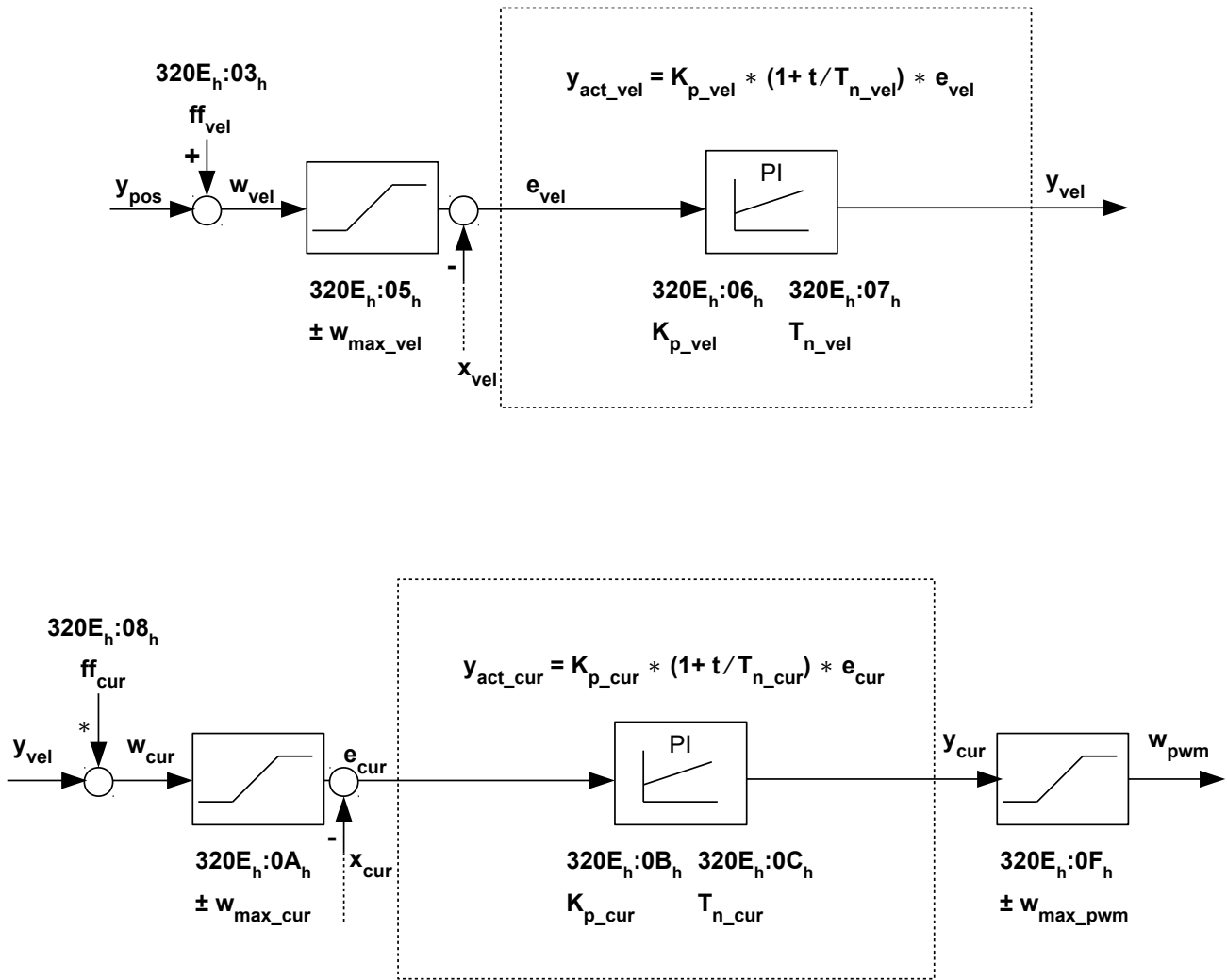
The old control parameters (object  $3210_h$ ) are still activated in the factory settings for compatibility reasons. For new applications, Nanotec recommends using the new control parameters.

To use the new parameters, you must set  $3210_h:07_h$  (for *closed loop*) or  $3210_h:09_h$  (for *open loop*) to "0". When the controller is switched on, the old values are converted and entered in the new object  $320E_h$  or  $320F_h$ . You must save both objects (see Saving objects).

Each controller consists of a proportional component with the *gain factor*  $K_p$  and an integral component with the *reset time*  $T_n$ . 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.

The following figures show the structure of the three cascaded controllers.





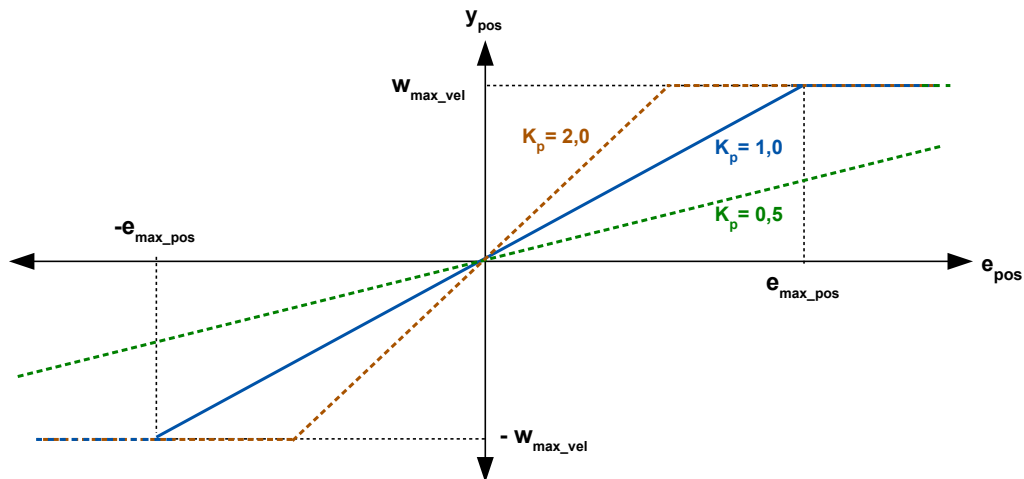
For each controller you can set a maximum control deviation ( $e_{max}$ ) and a *gain factor* ( $K_p$ ) that determine the output of the controller (control variable), taking into account the limitation of the control variable ( $y_{max}$ ).

The following figure shows the relationship between the maximum control deviation ( $e$ ), the control variable ( $y$ ) and the *gain factor* ( $K_p$ ) using the position controller as an example.

With a  $K_p$  of 100%, a maximum deviation set in 320E<sub>h</sub>:04<sub>h</sub> ( $e_{max\_pos}$ ) leads to the set maximum control variable set in 320E<sub>h</sub>:05<sub>h</sub> (in the case of the maximum speed,  $y_{max\_vel}$ ). For smaller deviations, the control variable is also correspondingly smaller.

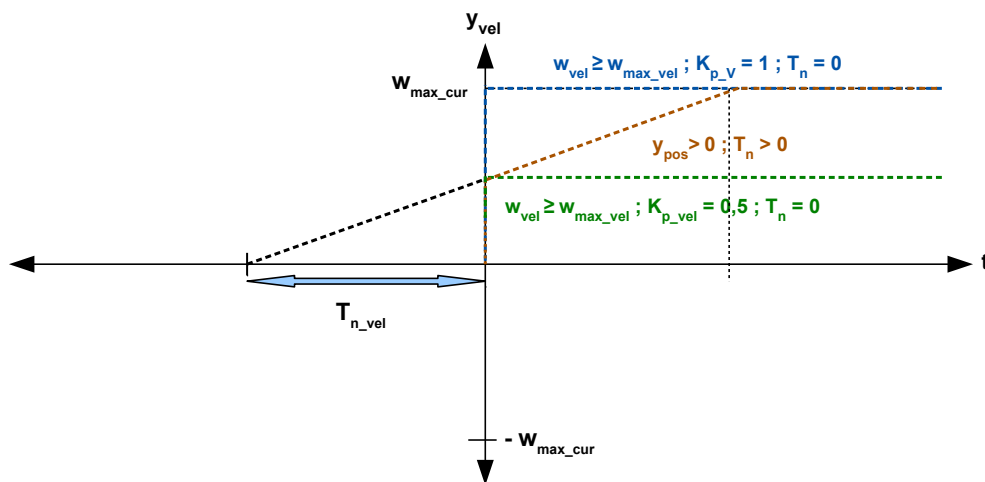
The *gain factor*  $K_p$  has a direct influence on the current control variable: at the same deviation, the control variable is proportional to the gain factor.





Each controller also has an integral component that is determined by the *reset time* ( $T_n$ ). The following figure shows the influence of the reset time on the control variable using the velocity controller as an example.

The smaller the reset time, the greater the influence of the integral component and the faster the control variable increases. If the reset time is 0, the integral component is internally set to "0" and the controller only has the proportional component.



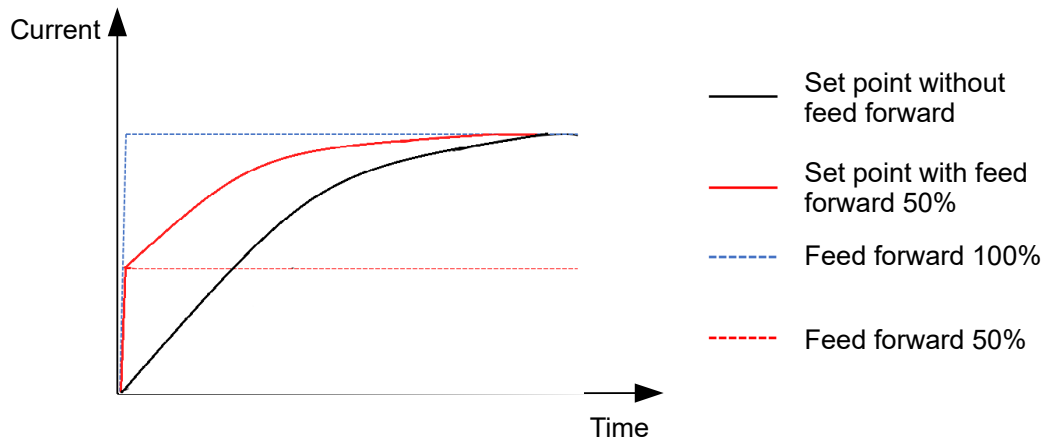
### 6.1.3.3 Feed forward

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".

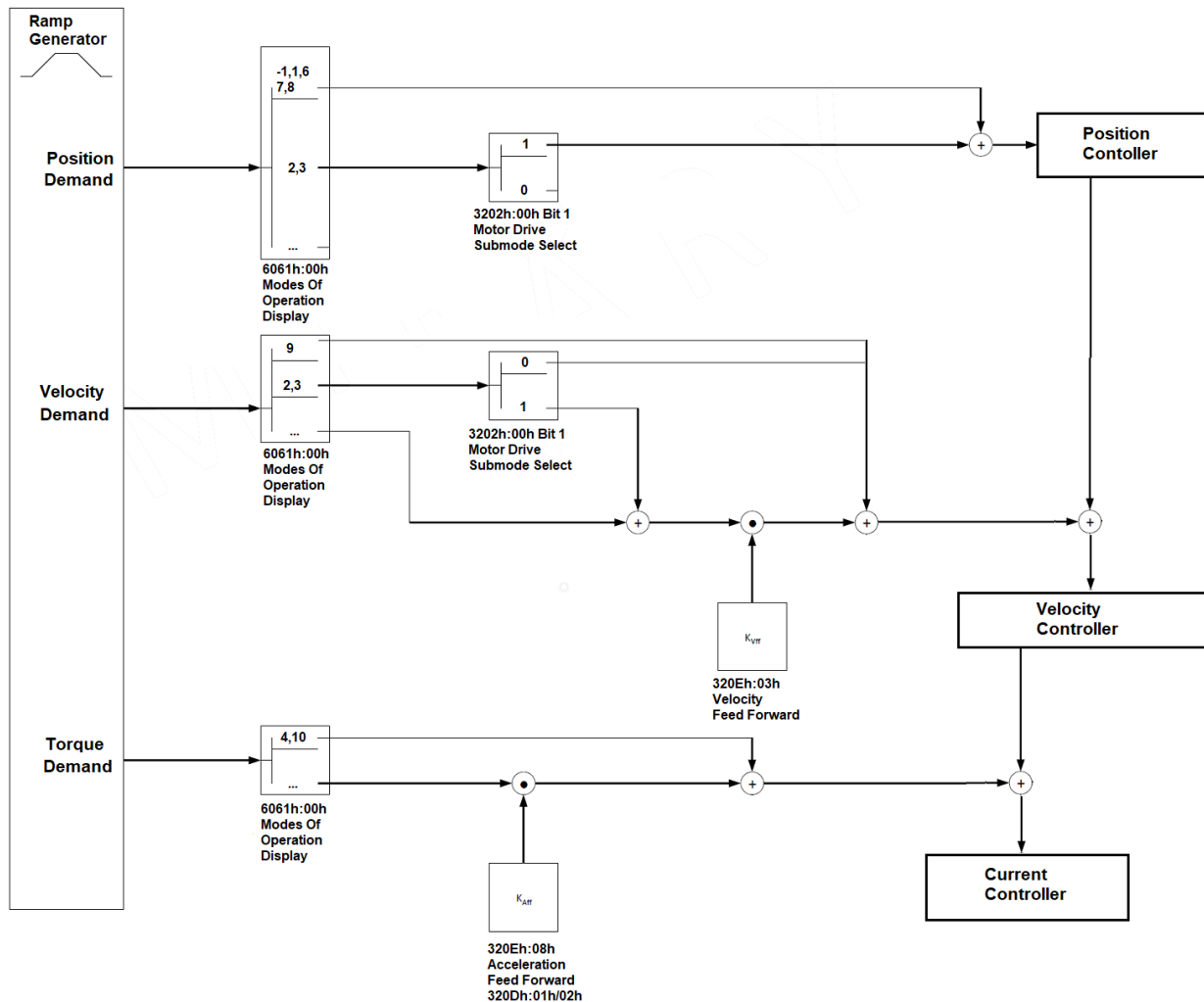


The factor for the *velocity feed forward* is set in object  $320E_h:03_h$  in tenths of a percent of the output of the ramp generator ( $606B_h$ ) and added to the output of the position controller before the velocity controller. The *velocity feed forward* is active in all modes with position control 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  $320E_h:08_h$  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  $320E_h:0D_h$  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  $320E_h:0D_h$  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 <sup>1</sup>
2	Commutation	0	1 <sup>2</sup>	1
<b>Index:Subindex</b>		3203 <sub>h</sub> :01 <sub>h</sub>	3203 <sub>h</sub> :02 <sub>h</sub>	3203 <sub>h</sub> :03 <sub>h</sub>

<sup>1</sup>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.

<sup>2</sup>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<sub>h</sub>). 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<sub>h</sub>).

#### 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<sub>h</sub> 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 `3202h` 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.

#### Note



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

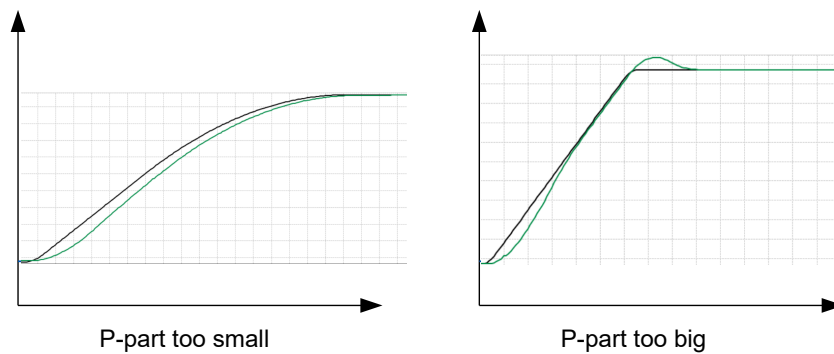
In *closed loop*, the measured control variable (actual value) is constantly compared with a set point (set value). In the event of deviations between these values, the controller readjusts according to the specified control parameters.

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.

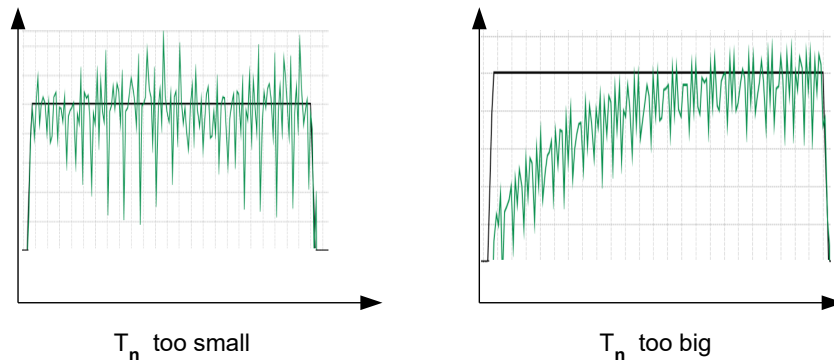
Due to the cascaded [Controller structure](#), it is useful to start the optimization of the inner-most controller (current controller) before the velocity and – if applicable – the position controller are optimized. Each of the three controllers consists of a proportional and an integral component, 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 reset time is too small, the system tends toward oscillations. If the reset 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.

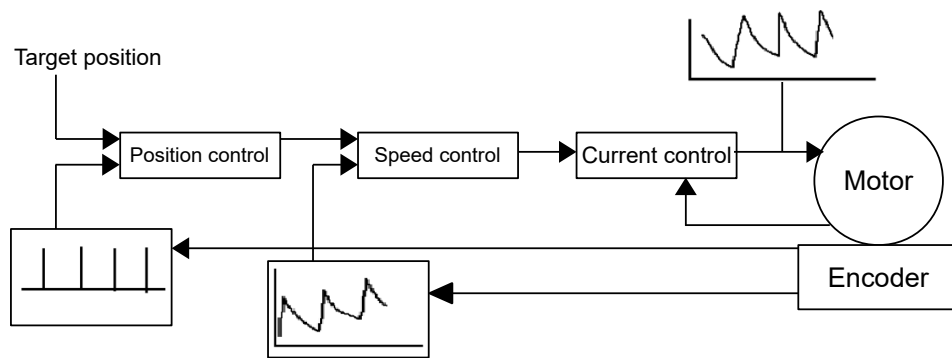
## 6.1.4 Slow Speed

### 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**

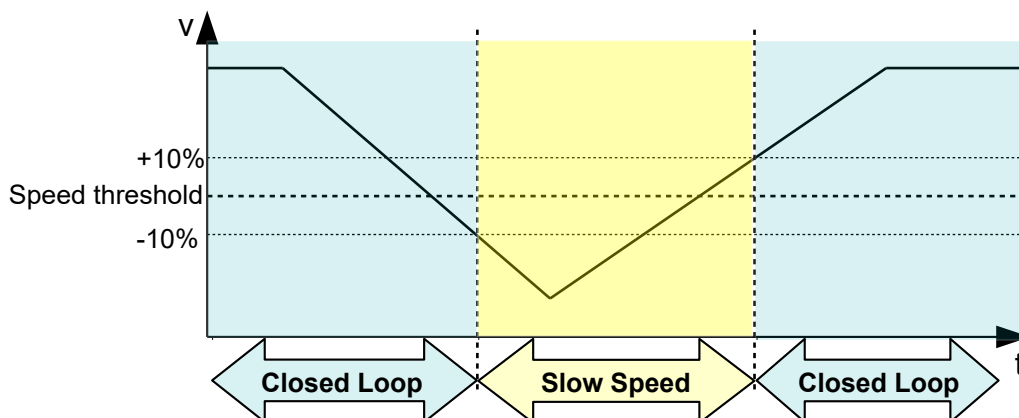
To activate the *slow speed* mode, you must:

1. activate closed loop,
2. in object `3202h` (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<sub>h</sub>](#), [6073<sub>h</sub>](#)) 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<sub>h</sub>](#) (controlword). The actual state of the state machine can be found in object [6041<sub>h</sub>](#) (statusword).

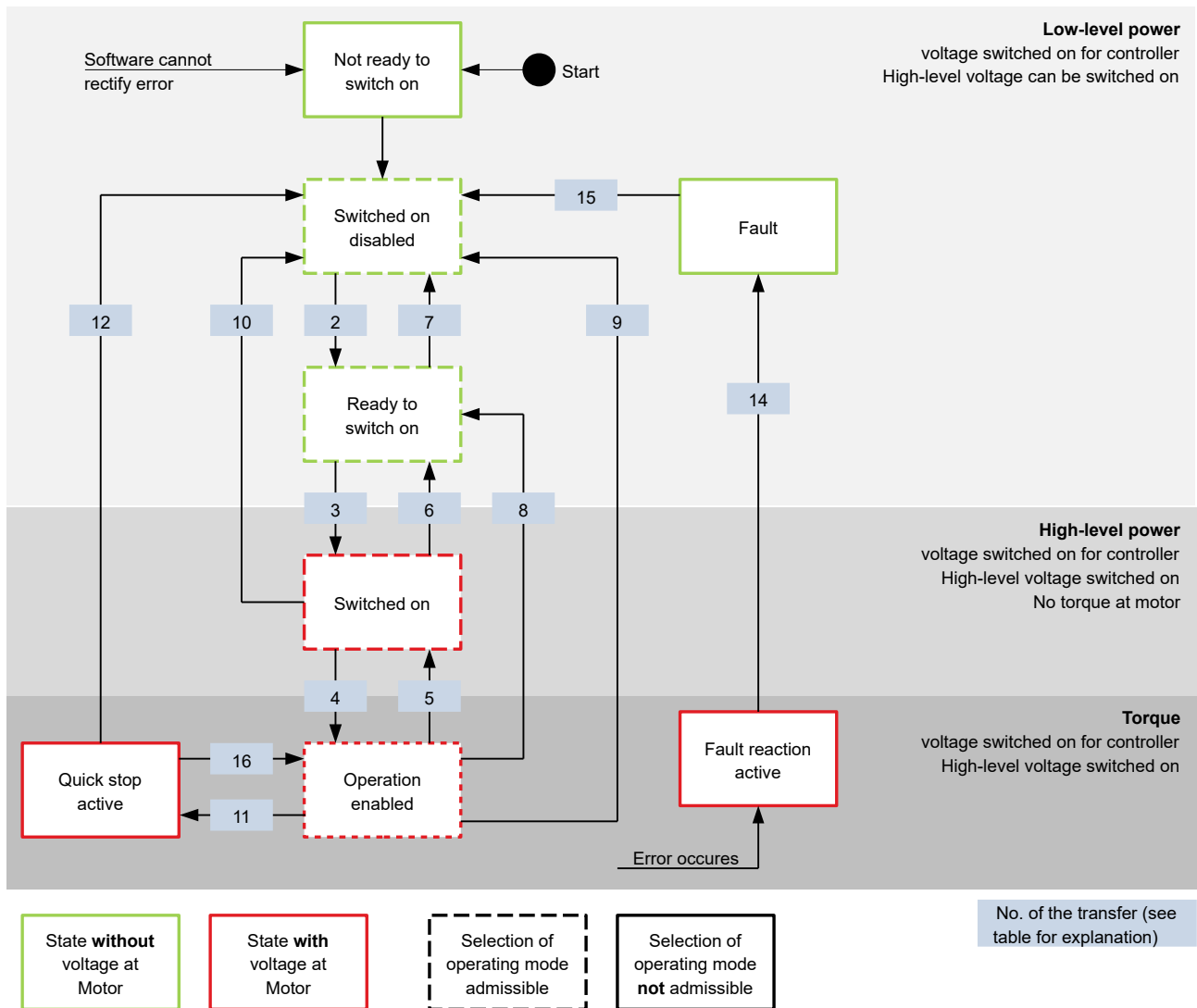
#### 6.2.1.2 Controlword

State changes are requested via object [6040<sub>h</sub>](#) (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 <sub>h</sub>					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 <sub>h</sub>					Transition
	Bit 7	Bit 3	Bit 2	Bit 1	Bit 0	
Fault 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 <sub>h</sub> )	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.

#### Note



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<sub>h</sub>. The actually active operating mode is displayed in 6061<sub>h</sub>.

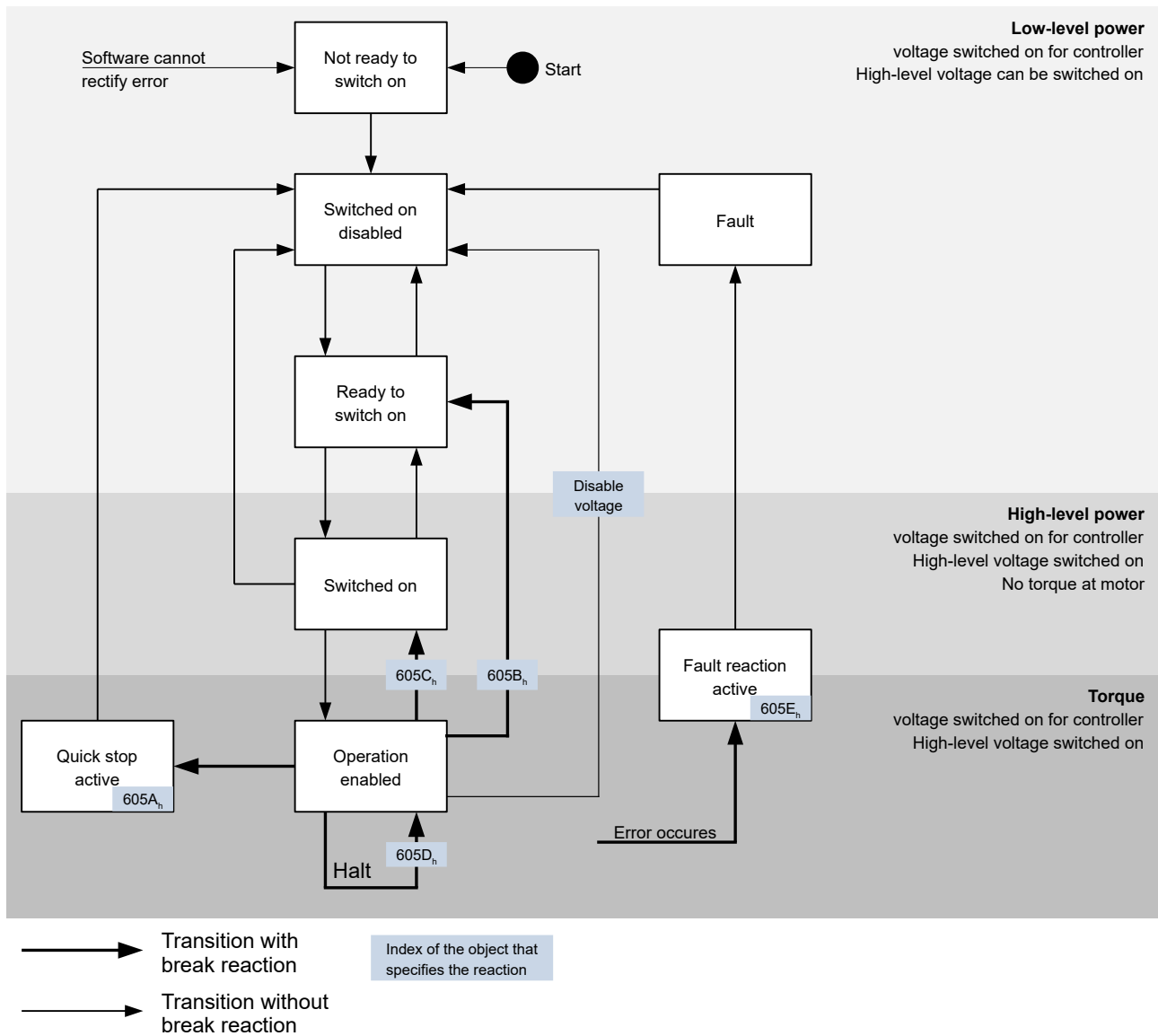
The operating mode can be set or changed at any time.

## 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<sub>n</sub> is executed (see following table).

Value in object 605A <sub>n</sub>	Description
0	Immediate stop with subsequent state change to <i>Switch on disabled</i>
1	Braking with <i>slow down ramp</i> (deceleration ramp depending on operating mode) and subsequent state change to <i>Switch on disabled</i>
2	Braking with <i>quick stop ramp</i> (6085 <sub>n</sub> ) and subsequent state change to <i>Switch on disabled</i>
5	Braking with <i>slow down ramp</i> (deceleration ramp depending on operating mode) and subsequent state change to <i>Quick stop active</i> ; control does not switch off and the motor remains energized. You can switch back to the <i>Operation enabled</i> state.

Value in object 605A <sub>h</sub>	Description
6	Braking with <i>quick stop ramp</i> (6085 <sub>h</sub> ) and subsequent state change to <i>Quick Stop Active</i> ; control does not switch off and the motor remains energized. You can switch back to the <i>Operation enabled</i> 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<sub>h</sub> is executed (see following table).

Value in object 605B <sub>h</sub>	Description
-32768 ... -1	Reserved
0	Blocking of the drive function – motor can turn freely
1	Braking with <i>slow down ramp</i> (braking deceleration depending on operating mode) and subsequent state change to <i>Switch on disabled</i>
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<sub>h</sub> is executed (see following table).

Value in object 605C <sub>h</sub>	Description
-32768 ... -1	Reserved
0	Blocking of the drive function – motor can turn freely
1	Braking with <i>slow down ramp</i> (braking deceleration depending on operating mode) and subsequent state change to <i>Switch on disabled</i>
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<sub>h</sub> (controlword), the action stored in 605D<sub>h</sub> is executed (see following table):

Value in object 605D <sub>h</sub>	Description
-32768 ... 0	Reserved
1	Braking with <i>slow down ramp</i> (braking deceleration depending on operating mode)
2	Braking with <i>quick stop ramp</i> (6085 <sub>h</sub> )
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<sub>h</sub>.

Value in object <u>605E<sub>h</sub></u>	Description
-32768 ... -1	Reserved
0	Blocking of the drive function – motor can turn freely
1	Braking with <i>slow down ramp</i> (braking deceleration depending on operating mode)
2	Braking with <i>quick stop ramp</i> ( <u>6085<sub>h</sub></u> )
3 ... 32767	Reserved

For each error that occurs, a more precise error code is stored in object 1003<sub>h</sub>.

### 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<sub>h</sub>.

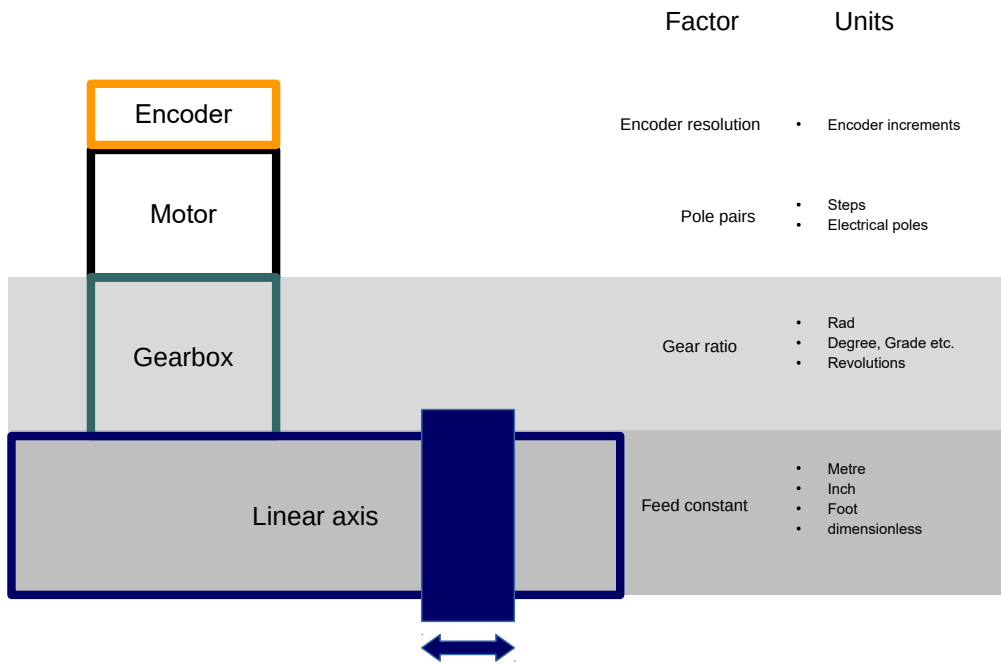
Value	Description
-32768 ... -1	Reserved
0	Blocking of the drive function – motor can turn freely
1	Braking with <i>slow down ramp</i> (braking deceleration depending on operating mode)
2	Braking with <i>quick stop ramp</i> ( <u>6085<sub>h</sub></u> )
3 ... 32767	Reserved

You can deactivate error monitoring by setting object 6065<sub>h</sub> to the value "-1" (FFFFFFFF<sub>h</sub>) or object 60F8<sub>h</sub> to the value "7FFFFFFFF<sub>h</sub>".

## 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.



**Note**

**i** 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<sub>h</sub> (Position unit) or 60A9<sub>h</sub> (Speed unit). Depending on the unit that is used, Feed constant (6092<sub>h</sub>) and/or Gear ratio (6091<sub>h</sub>) are/is taken into account.

Name	Unit symbol	Value	6091 <sub>h</sub>	6092 <sub>h</sub>	Description
meter	m	01 <sub>h</sub>	yes	yes	<i>Meter</i>
inch	in	C1 <sub>h</sub>	yes	yes	<i>Inch</i> (=0.0254 m)
foot	ft	C2 <sub>h</sub>	yes	yes	<i>Foot</i> (=0.3048 m)
grade	g	40 <sub>h</sub>	yes	no	<i>Gradian</i> (unit of angle, 400 corresponds to 360°)
radian	rad	10 <sub>h</sub>	yes	no	<i>Radian</i>
degree	°	41 <sub>h</sub>	yes	no	<i>Degrees</i>
arcminute	'	42 <sub>h</sub>	yes	no	<i>Arcminute</i> (60'=1°)
arcsecond	"	43 <sub>h</sub>	yes	no	<i>Arcsecond</i> (60"=1')
mechanical revolution		B4 <sub>h</sub>	yes	no	<i>Revolution</i>

Name	Unit symbol	Value	6091 <sub>h</sub>	6092 <sub>h</sub>	Description
encoder increment		B5 <sub>h</sub>	no	no	<i>Encoder increments.</i> Dependent on the used sensor (encoder/Hall sensor) and <i>control mode</i> . In <i>open loop</i> and <i>sensorless</i> mode, the number of pole pairs (2030 <sub>h</sub> ) multiplied by 65536 corresponds to one motor revolution.
step		AC <sub>h</sub>	no	no	<i>Steps.</i> With 2-phase stepper motors, the number of pole pairs (2030 <sub>h</sub> ) multiplied by 4 is equivalent to one revolution. With 3-phase BLDC motors, the number of pole pairs (2030 <sub>h</sub> ) multiplied by 6 is equivalent to one revolution.
electrical pole		C0 <sub>h</sub>	no	no	<i>Electric poles.</i> With a stepper motor that has, e.g., 50 pole pairs (2030 <sub>h</sub> ), the unit corresponds to 1/50 of a revolution.
dimensionless		00 <sub>h</sub>	yes	yes	<i>Dimensionless length unit</i>

Listed in the following table are all supported units for the time and their values for 60A9<sub>h</sub> (*Speed unit*):

Name	Unit symbol	Value	Description
second	s	03 <sub>h</sub>	<i>Second</i>
minute	min	47 <sub>h</sub>	<i>Minute</i>
hour	h	48 <sub>h</sub>	<i>Hour</i>
day	d	49 <sub>h</sub>	<i>Day</i>
year	a	4A <sub>h</sub>	<i>Year (=365.25 days)</i>

Listed in the following table are the possible exponents and their values for 60A8<sub>h</sub> (*Position unit*) and 60A9<sub>h</sub> (*Speed unit*):

Factor	Exponent	Value
10 <sup>6</sup>	6	06 <sub>h</sub>
10 <sup>5</sup>	5	05 <sub>h</sub>
...	...	...
10 <sup>1</sup>	1	01 <sub>h</sub>
10 <sup>0</sup>	0	00 <sub>h</sub>
10 <sup>-1</sup>	-1	FF <sub>h</sub>
...	..	...
10 <sup>-5</sup>	-5	FB <sub>h</sub>
10 <sup>-6</sup>	-6	FA <sub>h</sub>

### 6.3.2 Encoder resolution

The physical resolution for position measurement of the used encoder/sensor is calculated from the encoder increments (60E6<sub>h</sub> (*Encoder Increments*)) per motor revolutions (60EB<sub>h</sub> (*Motor Revolutions*)).

### 6.3.3 Gear ratio

The gear ratio is calculated from motor revolutions (60E8<sub>h</sub> (*Motor Shaft Revolutions*)) per axis rotations (60ED<sub>h</sub> (*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<sub>h</sub>" (bits 16-23=41<sub>h</sub> and bits 24-31=FF<sub>h</sub>), 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<sub>h</sub>" (bits 16-23=01<sub>h</sub> and bits 24-31=FD<sub>h</sub>(=-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<sub>h</sub> is written with the value "00B44700<sub>h</sub>" (bits 8-15=00<sub>h</sub>, bits 16-23=B4<sub>h</sub> and bits 24-31=47<sub>h</sub>), the unit is set to *revolutions per minute* (factory setting).

**Example**

If 60A9<sub>h</sub> is written with the value "FD010300<sub>h</sub>" (bits 8-15=FD<sub>h</sub>(=-3), bits 16-23=01<sub>h</sub> and bits 24-31=03<sub>h</sub>), 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.

**Note**

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<sub>h</sub>:01<sub>h</sub>) divided by the factor for denominator (6096<sub>h</sub>:02<sub>h</sub>).

$$n_{\text{velocity}} = \frac{6096_{\text{h}}:01}{6096_{\text{h}}:02}$$

**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<sub>h</sub>:01<sub>h</sub>) divided by the denominator (6097<sub>h</sub>:02<sub>h</sub>).

$$n_{\text{acceleration}} = \frac{6097_{\text{h}}:01}{6097_{\text{h}}:02}$$

**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<sub>h</sub>:01<sub>h</sub>) divided by the denominator (60A2<sub>h</sub>:02<sub>h</sub>).

$$n_{\text{jerk}} = \frac{60A2_{\text{h}}:01}{60A2_{\text{h}}:02}$$

## 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 passed over, bit 7 (*Warning*) is set in `6041h` (*statusword*) and the action that is stored in object `3701h` executed (see following table).

Value in object <code>3701<sub>h</sub></code>	Description
-1 (factory settings)	No reaction (e. g., to execute a homing operation)
1	Braking with <i>slow down ramp</i> (deceleration ramp depending on operating mode) and subsequent state change to <i>Switch on disabled</i>
2	Braking with <i>quick stop ramp</i> and subsequent state change to <i>Switch on disabled</i>
5	Braking with <i>slow down ramp</i> (deceleration ramp depending on operating mode) and subsequent state change to <i>Quick stop active</i> ; control does not switch off and the motor remains energized. You can switch back to the <i>Operation enabled</i> state.
6	Braking with <i>quick stop ramp</i> and subsequent state change to <i>Quick Stop Active</i> ; control does not switch off and the motor remains energized. You can switch back to the <i>Operation enabled</i> state.

As long as the limit switch is still active, travel in the direction of the limit switch is blocked; it is, however, possible to travel in the opposite direction.

Bit 7 (*Warning*) in `6041h` is not deleted until the limit switch is deactivated and the limit switch position has been passed back over.

#### Note



The quick-stop bit (bit 2) in `6040h` is not automatically set to "0" when the state changes to *Quick stop active*.

► If you want to change the *state machine* back to the *Operation enabled* state, you must set the bit to "0" and then to "1" again.

### 6.4.2 Software limit switches

The controller takes into account software limit switches (`607Dh` (Software Position Limit)). Target positions (`607Ah`) are limited by `607Dh`; the absolute target position may not be larger than the limits in `607Dh`. 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

Task	Cycle time
Current controller	62.5 $\mu$ s (16 kHz)
Velocity controller	250 $\mu$ s (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<sub>h</sub> (Modes Of Operation) (see "CiA 402 Power State Machine").

##### 7.1.1.3 Controlword

The following bits in object 6040<sub>h</sub> (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<sub>h</sub>) is absolute and with "1" the target position is relative. The reference position is dependent on bits 0 and 1 of object 60F2<sub>h</sub>.
- 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<sub>h</sub>.
- 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 <sub>h</sub>		
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".

#### Note



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<sub>h</sub> (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<sub>h</sub>) for a preset time (6068<sub>h</sub>).
- Bit 11: Limit exceeded: The demand position is above or below the limit values set in 607D<sub>h</sub>.
- 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.

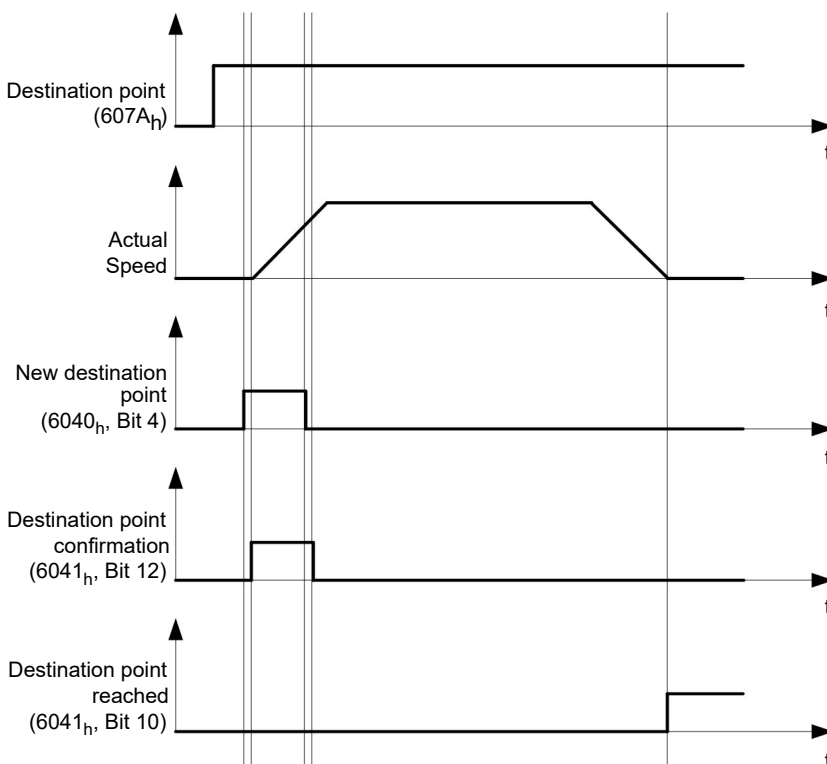
The bit is not set if one of the following conditions is met:

- 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<sub>h</sub> (Following Error Window) and 6066<sub>h</sub> (Following Error Time Out)).

## 7.1.2 Setting travel commands

### 7.1.2.1 Travel command

In object 607A<sub>h</sub> (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<sub>h</sub> (controlword). If the target position is valid, the controller responds with bit 12 in object 6041<sub>h</sub> (statusword) and begins the positioning move. As soon as the position is reached, bit 10 in the statusword is set to "1".



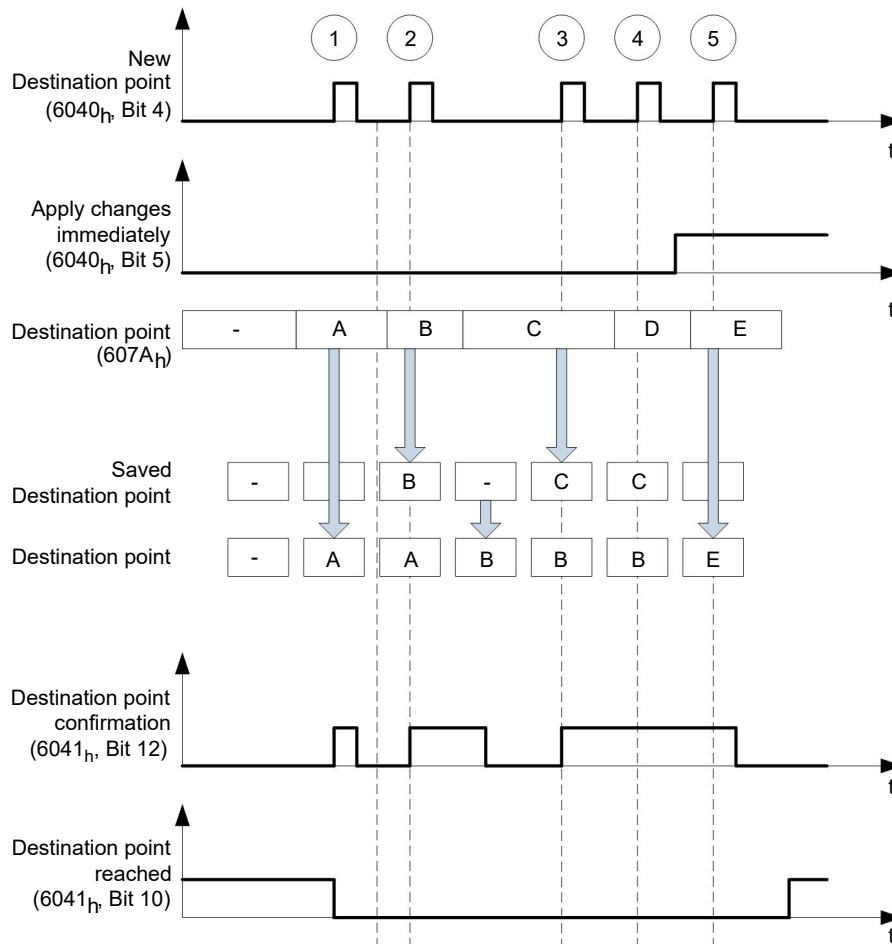
The controller can also reset bit 4 in object 6040<sub>h</sub> (controlword) on its own. This is set with bits 4 and 5 of object 60F2<sub>h</sub>.

### 7.1.2.2 Other travel commands

Bit 12 in object 6041<sub>h</sub> (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).

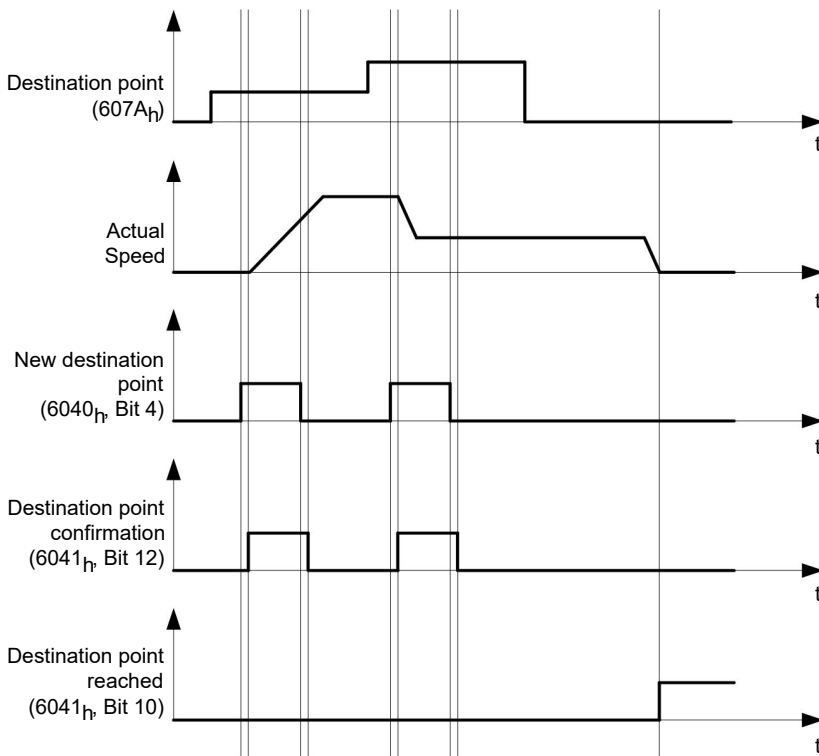
If the buffer is already full, a new set point is ignored (time 4). If bit 5 in object  $6040_h$  (controlword, bit: "Change Set-Point Immediately") is set, the controller operates without the buffer; new travel commands are implemented directly (time 5).

**Times**



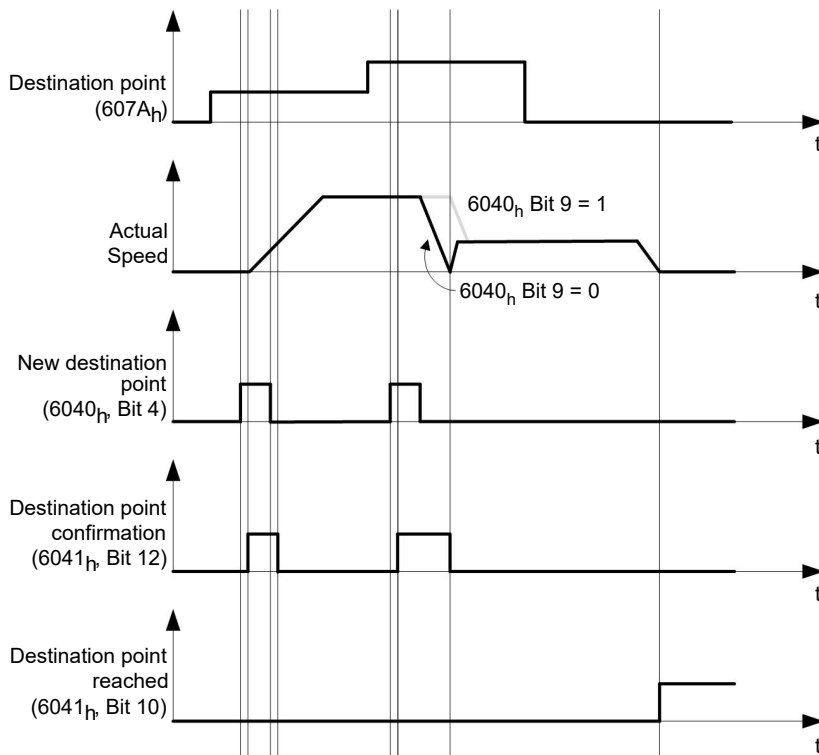
**Transition procedure for second target position**

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



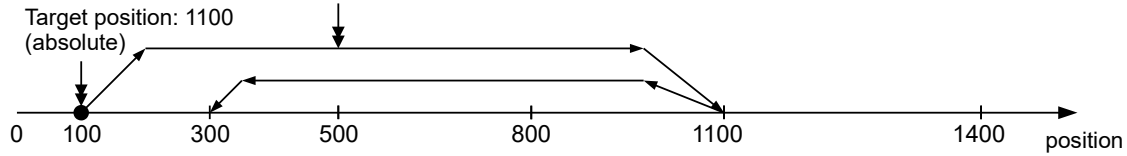
**Possible combinations of travel commands**

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

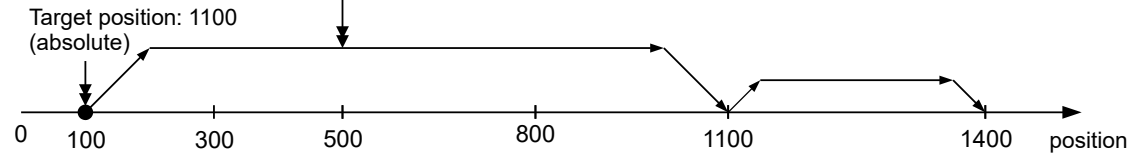
The following applies for the figures below:

- 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.

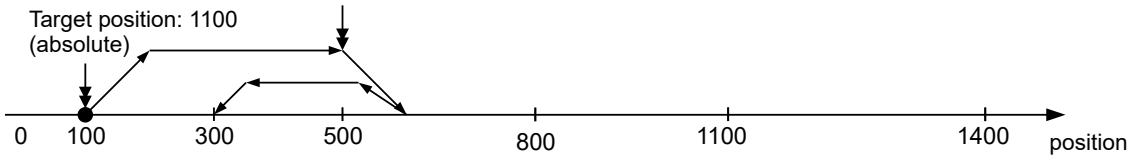
- Change on setpoint ( $6040_h;00$  Bit 5 = 0)
- Move absolute ( $6040_h;00$  Bit 6 = 0)
- Target position: 300



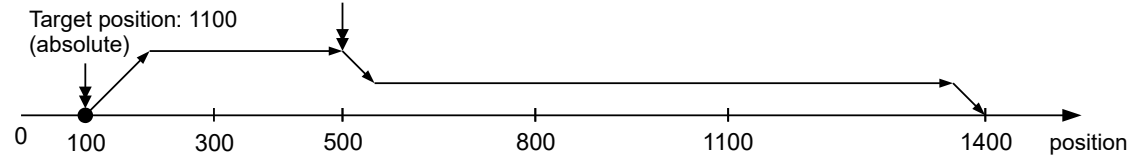
- Relative to the preceding target position ( $60F2_n;00 = 0$ )
- Change on setpoint ( $6040_h;00$  Bit 5 = 0)
- Move relative ( $6040_h;00$  Bit 6 = 1)
- Target position: 300



- Change set immediately ( $6040_h;00$  Bit 5 = 1)
- Move absolute ( $6040_h;00$  Bit 6 = 0)
- Target position: 300

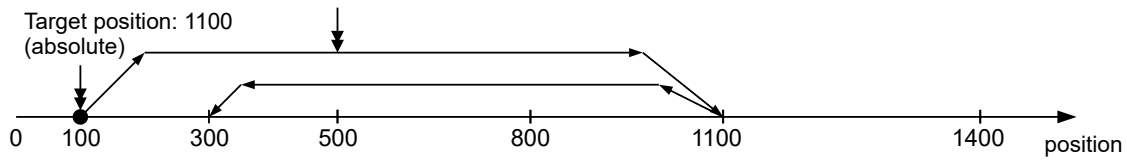


- Relative to the preceding target position ( $60F2_n;00 = 0$ )
- Change set immediately ( $6040_h;00$  Bit 5 = 1)
- Move relative ( $6040_h;00$  Bit 6 = 1)
- Target position: 300

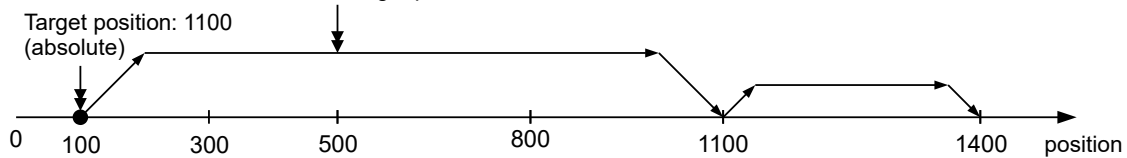




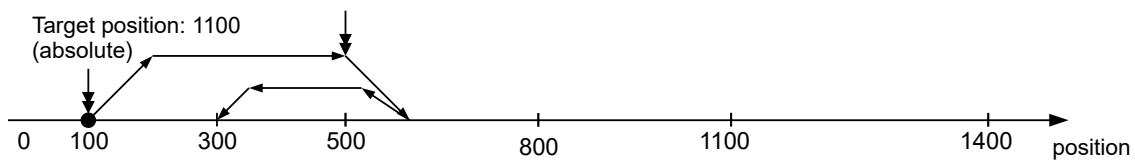
- Change on setpoint (6040<sub>h</sub>:00 Bit 5 = 0)
- Move absolute (6040<sub>h</sub>:00 Bit 6 = 0)
- Target position: 300



- Relative to the actual position (60F2<sub>h</sub>:00 = 1)
- Change on setpoint (6040<sub>h</sub>:00 Bit 5 = 0)
- Move relative (6040<sub>h</sub>:00 Bit 6 = 1)
- Target position: 300

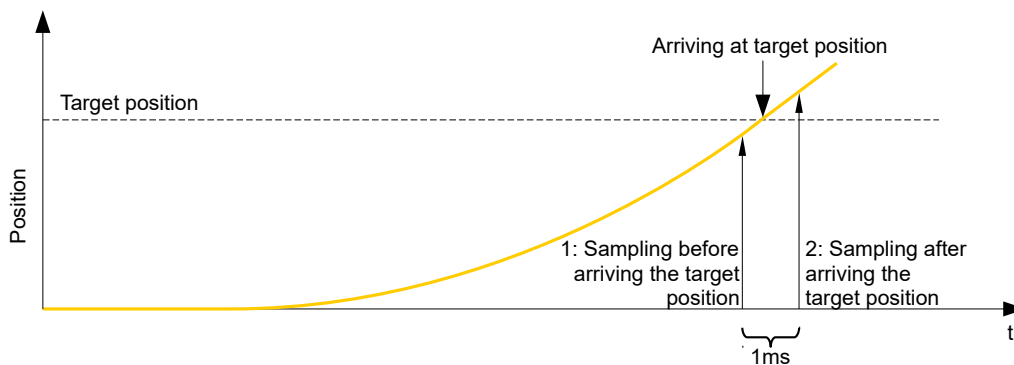


- Change set immediately (6040<sub>h</sub>:00 Bit 5 = 1)
- Move absolute (6040<sub>h</sub>:00 Bit 6 = 0)
- Target position: 300



### 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

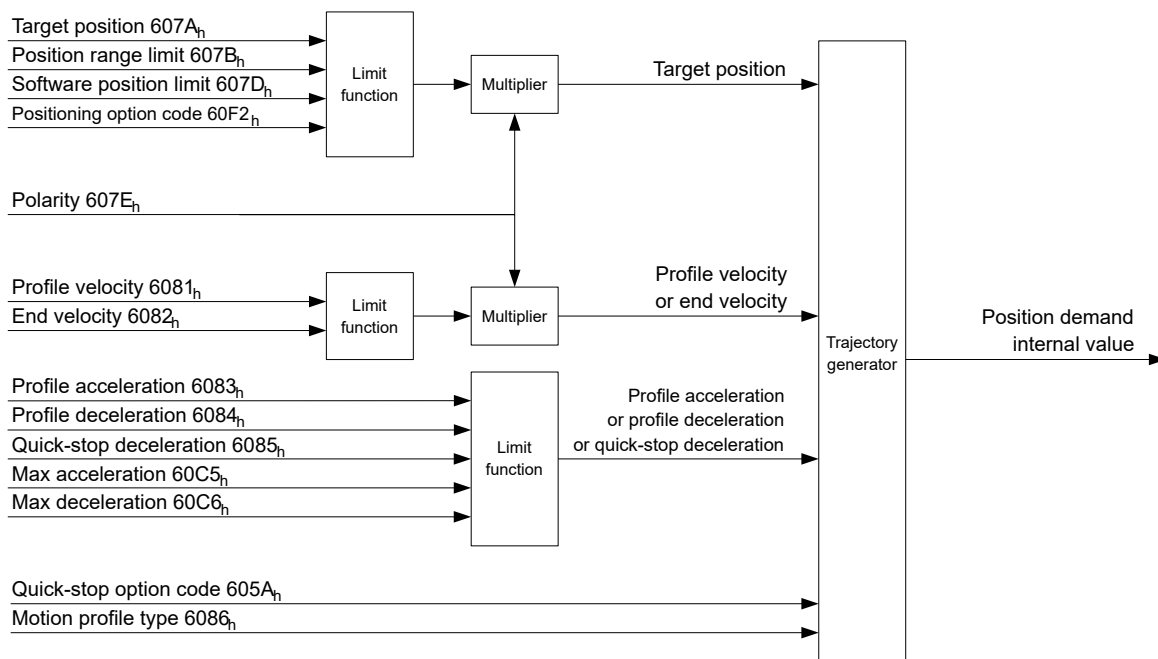
### 7.1.4.1 Object entries

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

- 607A<sub>h</sub>: (Target Position): Planned target position
- 607D<sub>h</sub>: (Software Position Limit): Definition of the limit stops (see chapter [Software limit switches](#))
- 607C<sub>h</sub>: (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<sub>h</sub>: (Position Range Limit): Limits of a modulo operation for replicating an endless rotation axis
- 607<sub>h</sub>: (Polarity): Direction of rotation
- 6081<sub>h</sub>: (Profile Velocity): Maximum speed with which the position is to be approached
- 6082<sub>h</sub>: (End Velocity): Speed upon reaching the target position
- 6083<sub>h</sub>: (Profile Acceleration): Desired starting acceleration
- 6084<sub>h</sub>: (Profile Deceleration): Desired braking deceleration
- 6085<sub>h</sub>: (Quick Stop Deceleration): Emergency-stop braking deceleration in case of the "Quick stop active" state of the "CiA 402 Power State Machine"
- 6086<sub>h</sub>: (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<sub>h</sub>:1<sub>h</sub>-4<sub>h</sub> are set as limits for the jerk.
- 60C5<sub>h</sub>: (Max Acceleration): The maximum acceleration that may not be exceeded when moving to the end position
- 60C6<sub>h</sub>: (Max Deceleration): The maximum braking deceleration that may not be exceeded when moving to the end position
- 60A4<sub>h</sub>: (Profile Jerk), subindex 01<sub>h</sub> to 04<sub>h</sub>: Objects for specifying the limit values for the jerk.
- The speed is limited by 607E<sub>h</sub> (Max Profile Velocity) and 6080<sub>h</sub> (Max Motor Speed); the smaller value is used as the limit.
- 60F2<sub>h</sub>: (Positioning Option Code): Defines the positioning behavior
- 60B0<sub>h</sub>: (Position Offset): Offset for the position set value in user-defined units

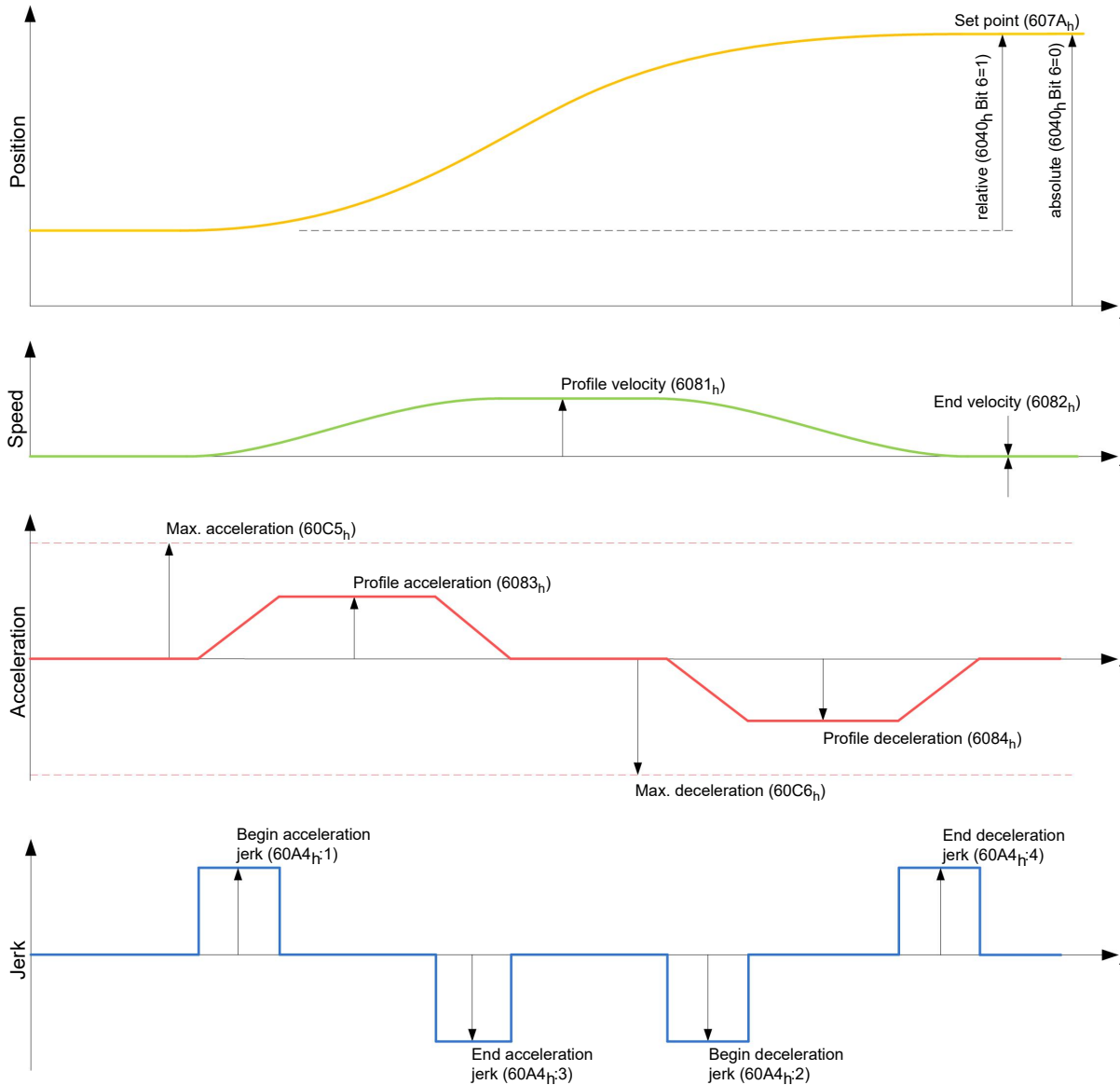
### 7.1.4.2 Objects for the positioning move

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<sub>h</sub> to "3". The entries for the jerks in subindices :1<sub>h</sub>–4<sub>h</sub> 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<sub>h</sub> 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<sub>h</sub> (Modes Of Operation) (see "CiA 402 Power State Machine").

### 7.2.3 Controlword

The following bits in object 6040<sub>h</sub> (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<sub>h</sub> (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<sub>h</sub> (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<sub>h</sub>: Target Velocity.  
The target speed is set here in user-defined units.
- 6048<sub>h</sub>: 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:

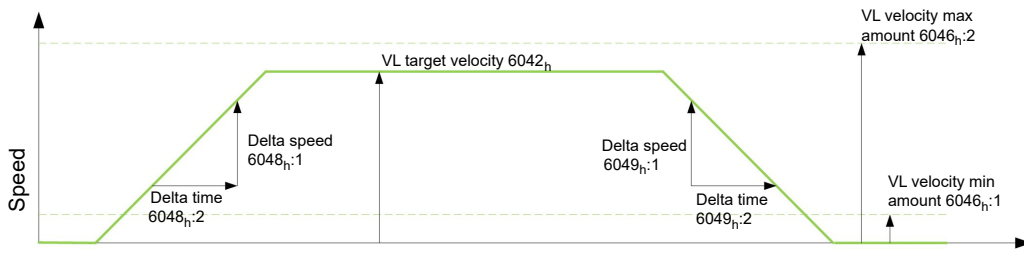
$$\text{VL velocity acceleration} = \frac{\text{Delta speed (6048}_{h}:1)}{\text{Delta time (6048}_{h}:2)}$$

- 6049<sub>h</sub> (Velocity Deceleration):  
This object defines the deceleration (deceleration ramp). The subindices here are arranged as described in object 6048<sub>h</sub>; the change in speed is to be specified with positive sign.
- 6046<sub>h</sub> (Velocity Min Max Amount):  
The limitations of the target speeds are specified in this object.  
The minimum speed is set in 6046<sub>h</sub>:1<sub>h</sub>. If the target speed (6042<sub>h</sub>) falls below the minimum speed, the value is limited to the minimum speed 6046<sub>h</sub>:1<sub>h</sub>.  
The maximum speed is set in 6046<sub>h</sub>:2<sub>h</sub>. If the target speed (6042<sub>h</sub>) exceeds the maximum speed, the value is limited to the maximum speed 6046<sub>h</sub>:2<sub>h</sub>.
- 604A<sub>h</sub> (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<sub>h</sub>.
- 60B1<sub>h</sub> (Velocity Offset): Offset for the speed set value in user-defined units

The following objects can be used to check the function:

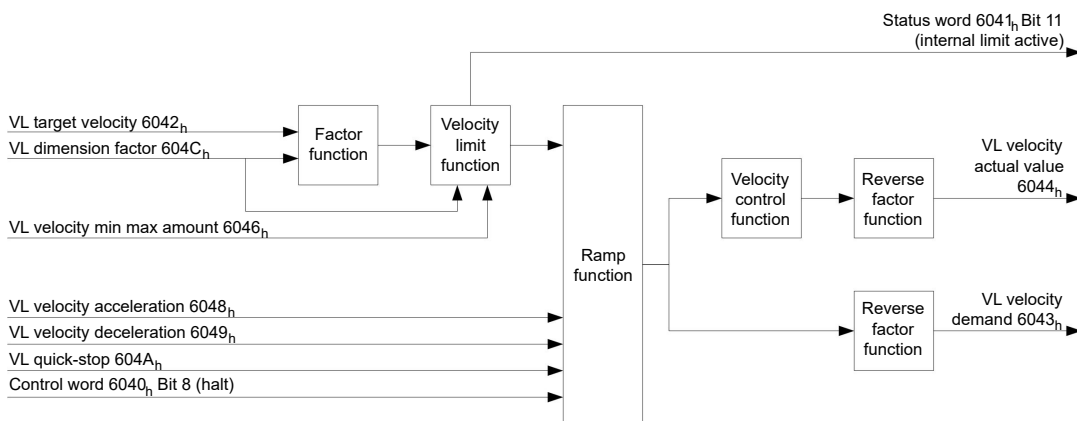
- 6043<sub>h</sub> (VI Velocity Demand)
- 6044<sub>h</sub> (VI Velocity Actual Value)

### 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<sub>h</sub> 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<sub>h</sub> (Modes Of Operation) (see "[CiA 402 Power State Machine](#)").

### 7.3.3 Controlword

The following bits in object 6040<sub>h</sub> (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<sub>h</sub> (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 <sub>h</sub> Bit 10	6040 <sub>h</sub> Bit 8	Description
0	0	Target speed not reached
0	1	Axis braking
1	0	Target speed within target window (defined in 606D <sub>h</sub> and 606E <sub>h</sub> )
1	1	Axis speed is 0

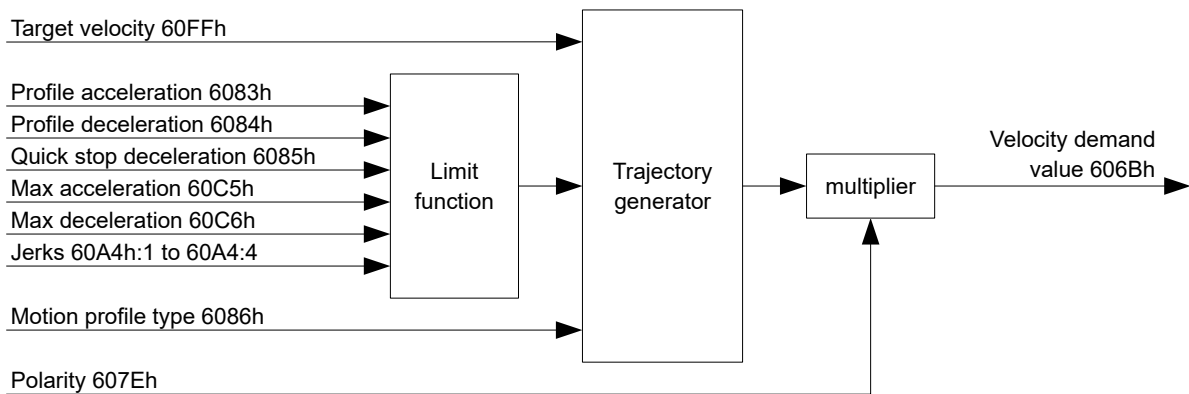
- Bit 12: This bit indicates whether the actual speed is zero. If the actual speed is greater than the value in 606F<sub>h</sub>(Velocity Threshold) for a time of 6070<sub>h</sub>(Velocity Threshold Time), this bit has the value "0". The bit otherwise remains set to "1".
- Bit 13 (Deviation Error): This 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<sub>h</sub> (Velocity Demand Value): This object contains the output of the ramp generator, which simultaneously serves as the preset value for the velocity controller.
- 606C<sub>h</sub> (Velocity Actual Value): Indicates the current actual speed.
- 606D<sub>h</sub> (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<sub>h</sub> (statusword) to be set to "1".
- 606E<sub>h</sub> (Velocity Window Time): This object specifies how long the actual speed and the set speed must be close to one another (see 606D<sub>h</sub> "Velocity Window") for bit 10 "Target speed reached" in object 6041<sub>h</sub> (statusword) to be set to "1".
- 607E<sub>h</sub> (Polarity): If bit 6 is set to "1" here, the sign of the target speed is reversed.
- 6083<sub>h</sub> (Profile acceleration): Sets the value for the acceleration ramp.
- 6084<sub>h</sub> (Profile Deceleration): Sets the value for the deceleration ramp.
- 6085<sub>h</sub> (Quick Stop Deceleration): Sets the value for the deceleration ramp for rapid braking.
- 6086<sub>h</sub> (Motion Profile Type): The ramp type can be selected here ("0" = trapezoidal ramp, "3" = jerk-limited ramp).
- 60FF<sub>h</sub> (Target Velocity): Specifies the target speed that is to be reached.
- The speed is limited by 607E<sub>h</sub> (Max Profile Velocity) and 6080<sub>h</sub> (Max Motor Speed); the smaller value is used as the limit.
- 60B1<sub>h</sub> (Velocity Offset): Offset for the speed set value in user-defined units

### 7.3.5.1 Objects in Profile Velocity Mode

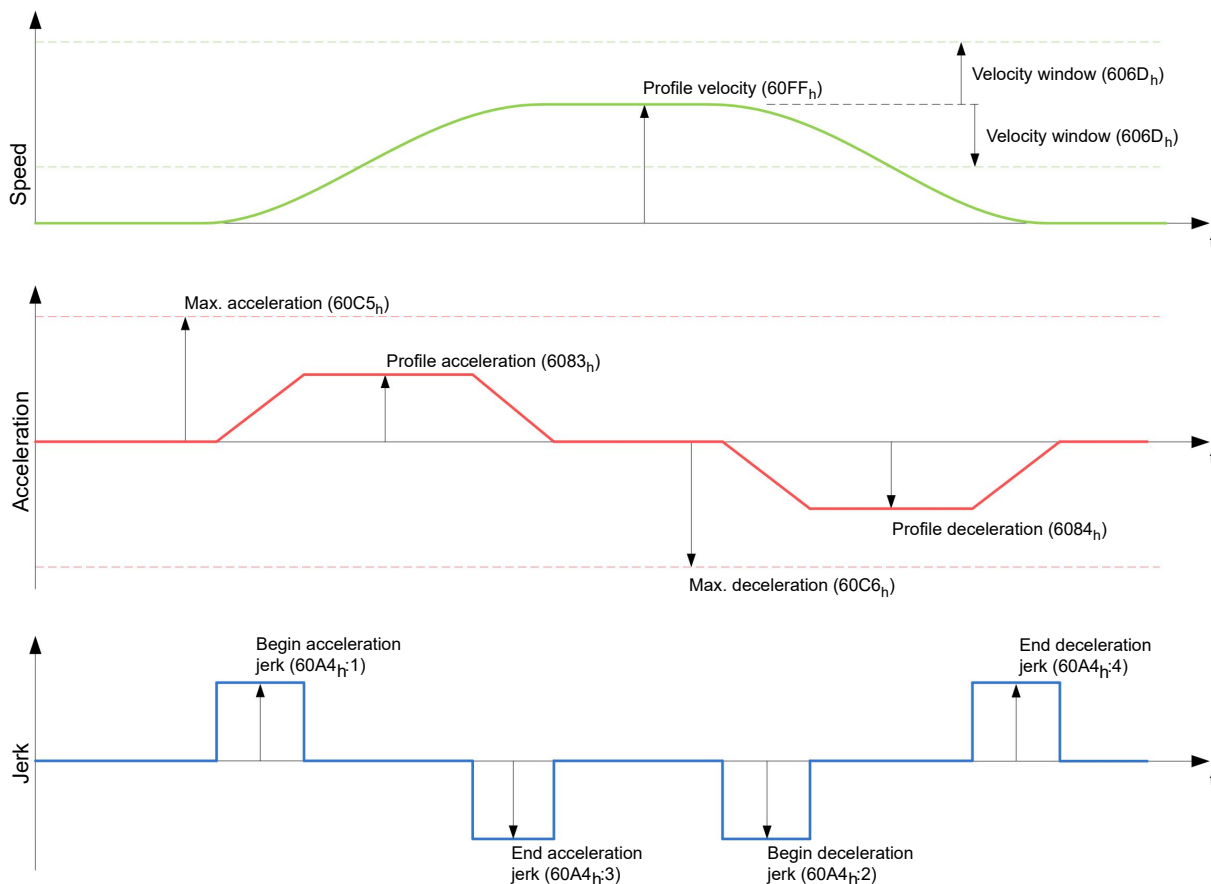


### 7.3.5.2 Activation

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.

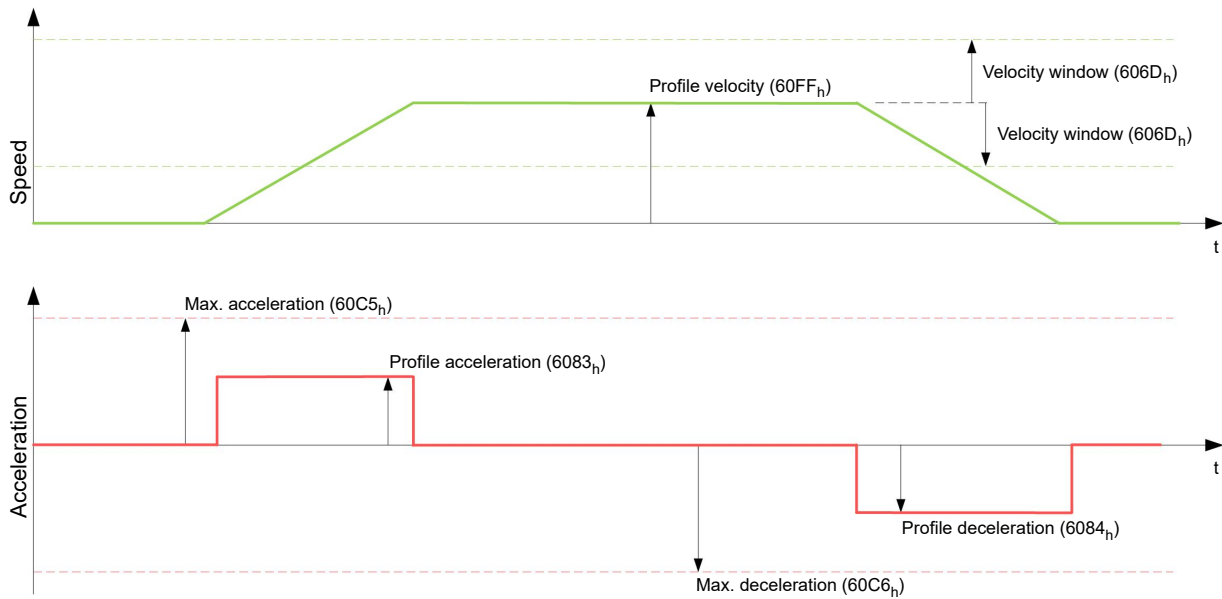
### 7.3.5.3 Limitations in the jerk-limited case

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.



**Note**

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

### 7.4.2 Activation

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

### 7.4.3 Controlword

The following bits in object 6040h (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 6041h (statusword) have a special function:

- Bit 10 (Target Reached): In combination with bit 8 of object 6040h (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).

6040h Bit 8	6041h 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
- $60B2_h$  (Torque Offset): Offset for the torque set value in tenths of a percent

**Note**

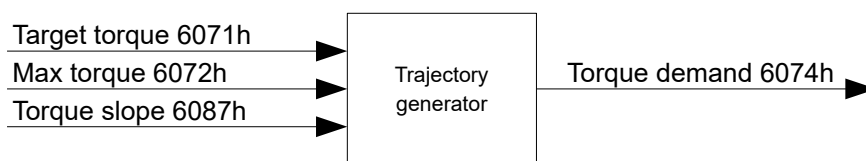


These values are not limited to 100% of the rated current ( $203B_h:01_h$ ). Torque values greater than the rated torque (generated from the rated current) can be achieved if the maximum duration ( $203B_h:02_h$ ) of the maximum current ( $6073_h$ ) is set (see [I2t Motor overload protection](#)). All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

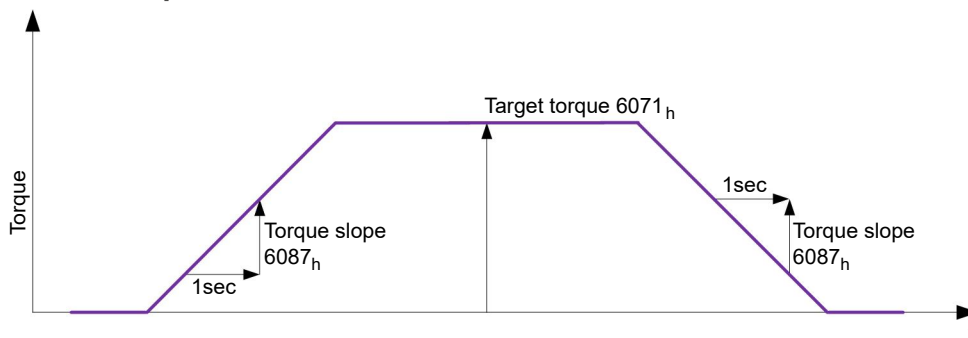
The following objects are also needed for this operating mode:

- $3202_h$  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 `6060h` (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 `3701h` to "-1" (factory setting) to prevent blocking the further travel of the motor.

#### 7.5.1.3 Controlword

The following bits in object `6040h` (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 `6041h` (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

#### Note

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

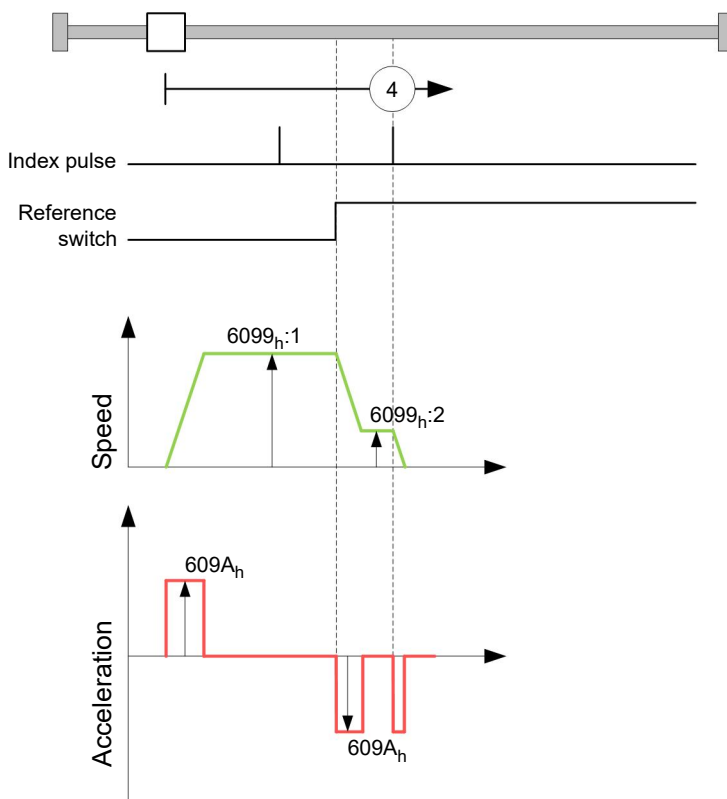
The following objects are necessary for controlling this mode:

- `607Ch` (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.

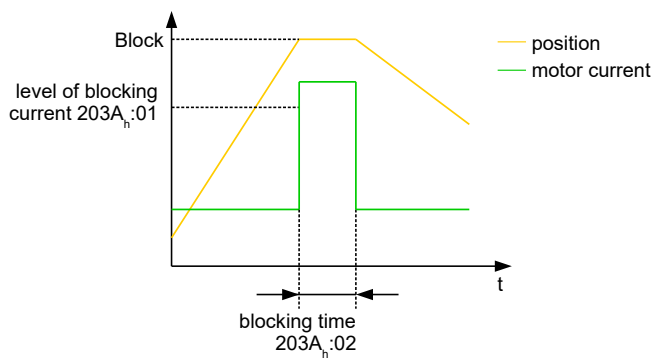
For methods that use homing on block, the same figures apply as for the methods with limit switch. Because nothing is different aside from the missing limit switches, the same figures are used. For the figures here, the limit switches must be replaced with a mechanical block.

### 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 `203Ah:01`, the current level is defined above which movement against the block is detected.
2. Blocking duration: In object `203Ah: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.

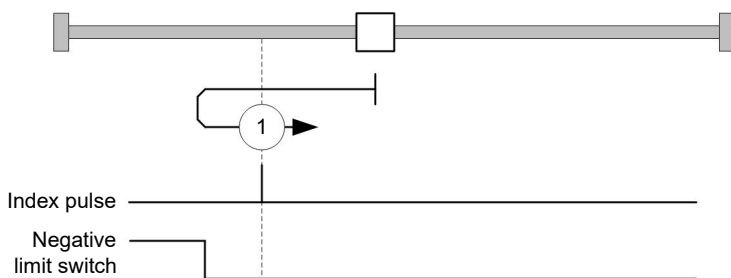
The following methods can be used for homing on block:

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

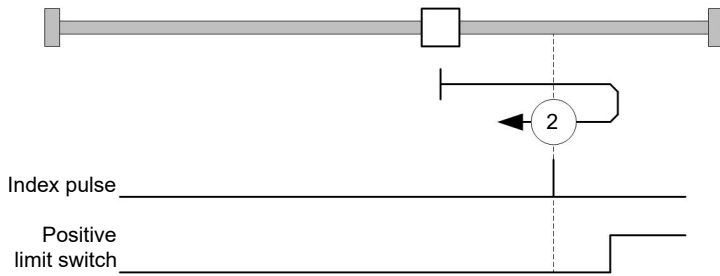
### 7.5.2.4 Methods 1 and 2

Reference to limit switches and index pulse.

Method 1 references to negative limit switch and index pulse:



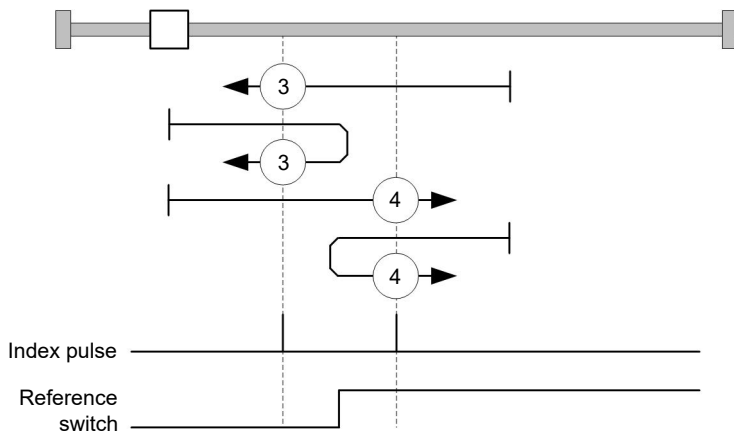
Method 2 references to positive limit switch and index pulse:



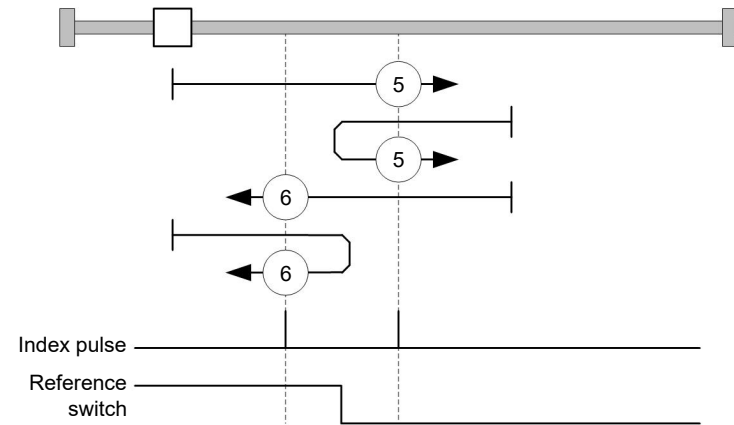
**7.5.2.5 Methods 3 to 6**

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:

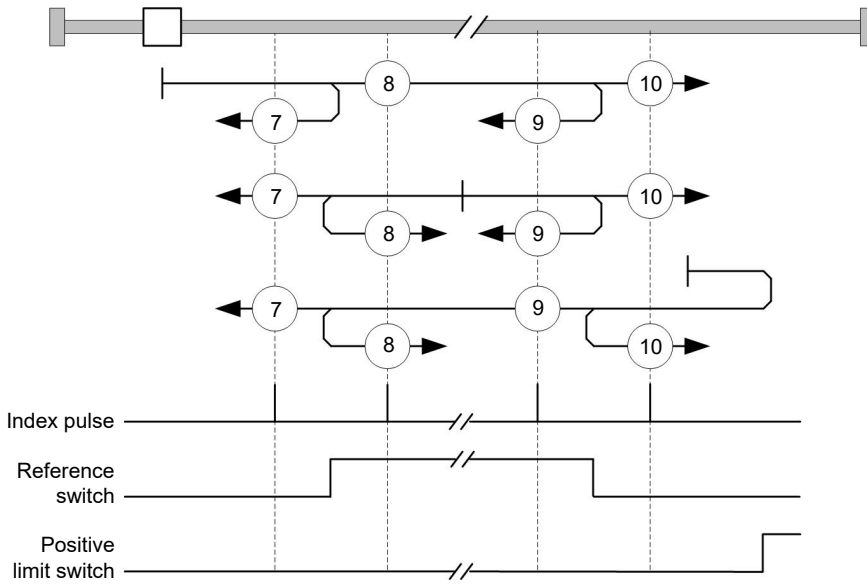


**7.5.2.6 Methods 7 to 14**

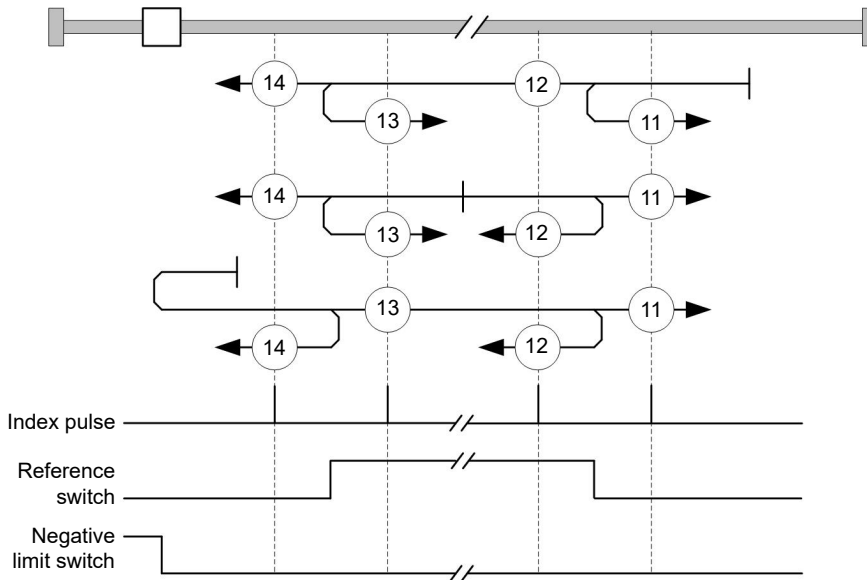
Reference to the home switch and index pulse (with limit switches).

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:



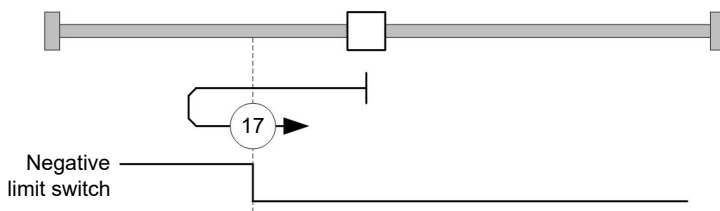
Methods 11 to 14 take the negative limit switch into account:



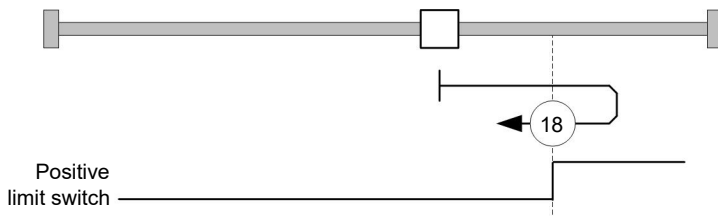
### 7.5.2.7 Methods 17 and 18

Reference to the limit switch without the index pulse.

Method 17 references to the negative limit switch:



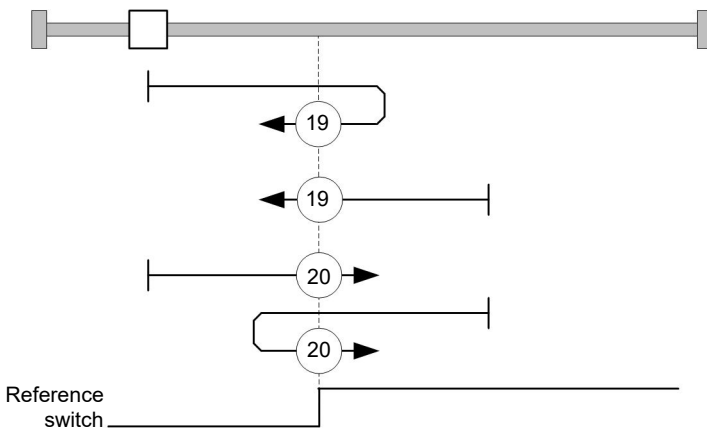
Method 18 references to the positive limit switch:



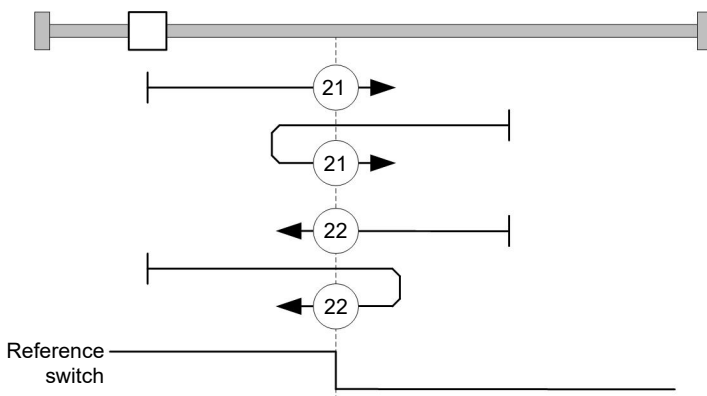
### 7.5.2.8 Methods 19 to 22

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:

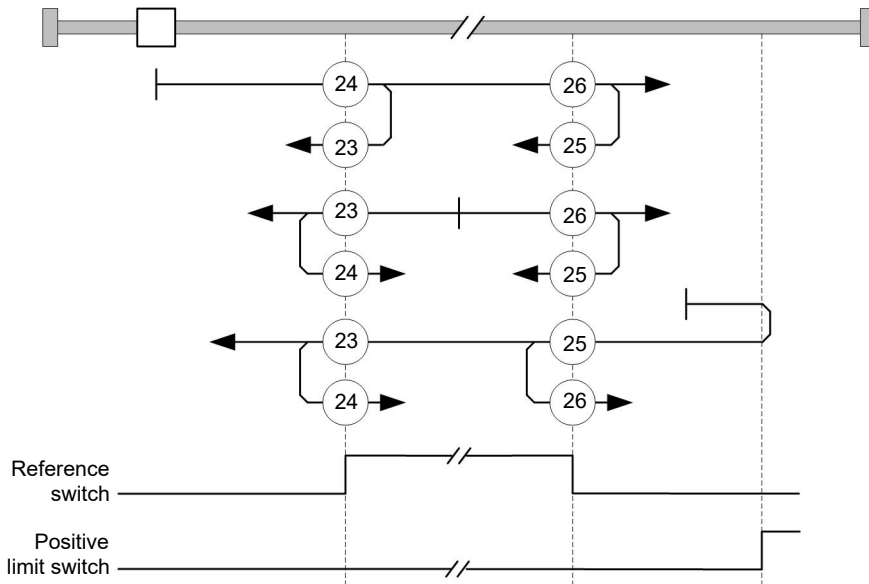


### 7.5.2.9 Methods 23 to 30

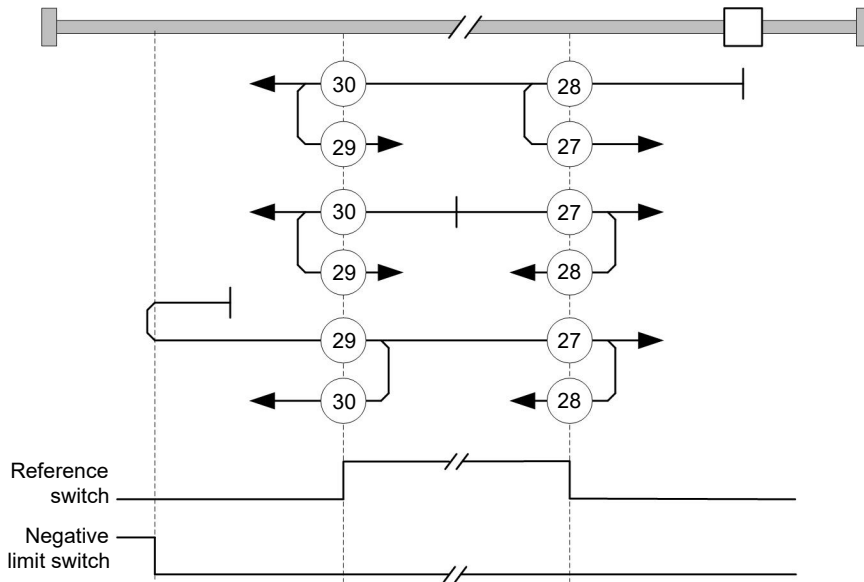
Reference to the home switch without the index pulse (with limit switches).

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:



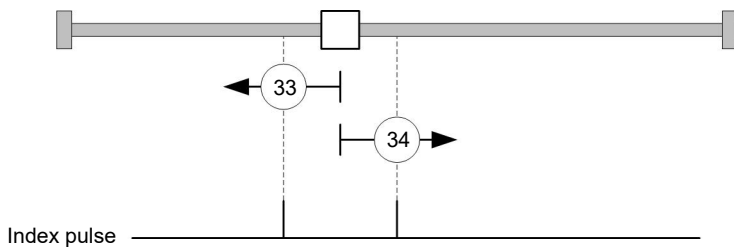
Methods 27 to 30 take the negative home switch into account:



### 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.



### Note



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.

### Note



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<sub>h</sub> (Modes Of Operation) (see "CiA 402 Power State Machine").

### 7.6.3 Controlword

The following bits in object 6040<sub>h</sub> (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<sub>h</sub>.

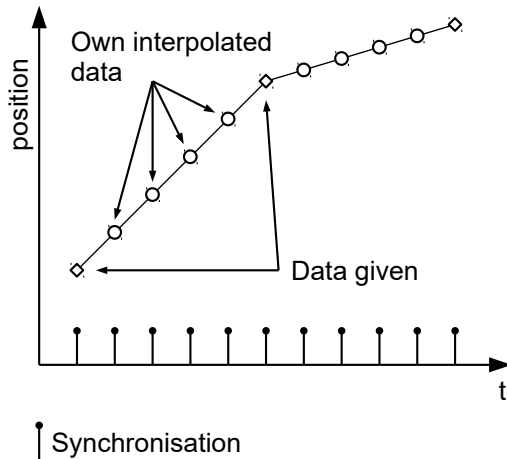
### 7.6.4 Statusword

The following bits in object 6041<sub>h</sub> (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<sub>h</sub> (Following Error Window) and 6066<sub>h</sub> (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<sub>h</sub>:01<sub>h</sub>.



In the current implementation, only

- linear interpolation
- and a target position

are supported.

### 7.6.6 Setup

The following setup is necessary:

- `60C2h:01h`: Time between two passed target positions in ms.
- `60C4h:06h`: This object is to be set to "1" to be able to modify the target position in object `60C1h:01h`.
- `6081h` (Profile Velocity): Maximum speed with which the position is to be approached
- `6084h` (Profile Deceleration): Desired braking deceleration during braking
- `60C6h`: (Max Deceleration): The maximum allowed braking deceleration
- Only if closed loop is activated: The speed is limited by `607Fh` (Max Profile Velocity) and `6080h` (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 `60C1h:01h` 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).



#### Note

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

### 7.7.1.2 Activation

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

### 7.7.1.3 Controlword

In this mode, the bits of controlword 6040<sub>h</sub> have no special function.

### 7.7.1.4 Statusword

The following bits in object 6041<sub>h</sub> (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 <u>607A<sub>h</sub></u> (Target Position) is ignored
12	1	Controller follows the target; object <u>607A<sub>h</sub></u> (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<sub>h</sub>.

## 7.7.2 Object entries

The following objects are necessary for controlling this mode:

- 607A<sub>h</sub> (Target Position): This object must be written cyclically with the position set value.
- 607B<sub>h</sub> (Position Range Limit): This object contains the preset for an overrun or underrun of the position specification.
- 607D<sub>h</sub> (Software Position Limit): This object defines the limitations within which the position specification (607A<sub>h</sub>) must be located.
- 6065<sub>h</sub> (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<sub>h</sub>), a following error is reported.
- 6066<sub>h</sub> (Following Error Time Out): This object specifies the time range in milliseconds. If the actual position is outside of the position corridor (6065<sub>h</sub>) for longer than this time range, a following error is triggered.
- 6085<sub>h</sub> (Quick-Stop Deceleration): This object contains the braking deceleration for the case that a quick-stop is triggered.
- 605A<sub>h</sub> (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<sub>h</sub> (Max Motor Speed): Maximum speed
- 60C2<sub>h</sub>:01<sub>h</sub> (Interpolation Time Period): This object specifies the time of a *cycle*; a new set value must be written in 607A<sub>h</sub> in these time intervals.  
The following applies here: cycle time = value of 60C2<sub>h</sub>:01<sub>h</sub> \* 10<sup>value of 60C2:02</sup> seconds.
- 60C2<sub>h</sub>:02<sub>h</sub> (Interpolation Time Index): This object specifies the time basis of the cycles. Currently, only value 60C2<sub>h</sub>:02<sub>h</sub>=-3 is supported; this yields a time basis of 1 millisecond.
- 60B0<sub>h</sub> (Position Offset): Offset for the position set value in user-defined units

The following objects can be read in this mode:

- 6064<sub>h</sub> (Position Actual Value)

- 606C<sub>h</sub> (Velocity Actual Value)
- 60F4<sub>h</sub> (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<sub>h</sub> (Modes Of Operation) (see "CiA 402 Power State Machine").

#### 7.8.1.3 Controlword

In this mode, the bits of controlword 6040<sub>h</sub> have no special function.

#### 7.8.1.4 Statusword

The following bits in object 6041<sub>h</sub> (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 <u>60FF<sub>h</sub></u> (Target Velocity) is ignored
12	1	Controller follows the target; object <u>60FF<sub>h</sub></u> (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<sub>h</sub> (Target Velocity): This object must be written cyclically with the speed set value.
- 6085<sub>h</sub> (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<sub>h</sub> (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<sub>h</sub> (Max Motor Speed): Maximum speed
- 60C2<sub>h</sub>:01<sub>h</sub> (Interpolation Time Period): This object specifies the time of a *cycle*; a new set value must be written in 60FF<sub>h</sub> in these time intervals.  
The following applies here: cycle time = value of 60C2<sub>h</sub>:01<sub>h</sub> \* 10<sup>value of 60C2:02</sup> seconds.
- 60C2<sub>h</sub>:02<sub>h</sub> (Interpolation Time Index): This object specifies the time basis of the cycles. Currently, only value 60C2<sub>h</sub>:02<sub>h</sub>=-3 is supported; this yields a time basis of 1 millisecond.
- 60B1<sub>h</sub> (Velocity Offset): Offset for the speed set value in user-defined units

The following objects can be read in this mode:

- 606C<sub>h</sub> (Velocity Actual Value)
- 607E<sub>h</sub> (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.



#### Note

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<sub>h</sub> (Modes Of Operation) (see "CiA 402 Power State Machine").

#### 7.9.1.3 Controlword

In this mode, the bits of controlword 6040<sub>h</sub> have no special function.

#### 7.9.1.4 Statusword

The following bits in object 6041<sub>h</sub> (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 <u>6071<sub>h</sub></u> (Target Torque) is ignored
12	1	Controller follows the target; object <u>6071<sub>h</sub></u> (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<sub>h</sub> (Target Torque): This object must be written cyclically with the torque set value and is to be set relative to 6072<sub>h</sub>.
- 6072<sub>h</sub> (Max Torque): Describes the maximum permissible torque.
- 6073<sub>h</sub> (Max Current):  
Maximum current. The minimum of 6073<sub>h</sub> and 6072<sub>h</sub> is used as limit for the torque in 6071<sub>h</sub>.
- 6080<sub>h</sub> (Max Motor Speed): Maximum speed
- 60C2<sub>h</sub>:01<sub>h</sub> (Interpolation Time Period): This object specifies the time of a *cycle*; a new set value must be written in 6071<sub>h</sub> in these time intervals.  
The following applies here: cycle time = value of 60C2<sub>h</sub>:01<sub>h</sub> \* 10<sup>value of 60C2:02</sup> seconds.
- 60C2<sub>h</sub>:02<sub>h</sub> (Interpolation Time Index): This object specifies the time basis of the cycles. Currently, only value 60C2<sub>h</sub>:02<sub>h</sub>=-3 is supported; this yields a time basis of 1 millisecond.
- 60B2<sub>h</sub> (Torque Offset): Offset for the torque set value in tenths of a percent

The following objects can be read in this mode:

- 606C<sub>h</sub> (Velocity Actual Value)
- 6074<sub>h</sub> (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<sub>h</sub> (Modes Of Operation) (see "CiA 402 Power State Machine").

#### Note

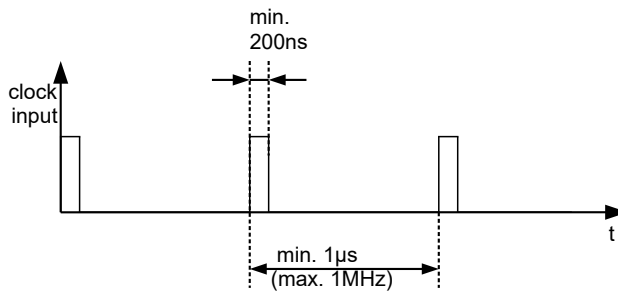


If this mode is activated, the pins for clock and direction are automatically configured accordingly and the settings in 3231<sub>h</sub> are overwritten.

### 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<sub>h</sub>). The input pulses that arrive within a cycle are collected and buffered in the controller.
- The steps are scaled using objects 2057<sub>h</sub> and 2058<sub>h</sub>. 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<sub>h</sub>=128 and 2058<sub>h</sub>=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.

#### Note



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.

### Note



If there is a change of direction, a time of at least  $35\ \mu\text{s}$  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:

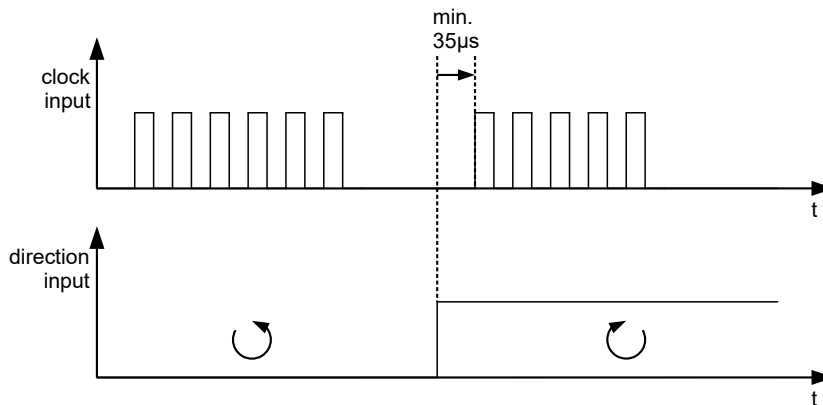
- Bit 13 (Following Error): This bit is set in *closed loop* mode if the following error is greater than the set limits ( $6065_h$  (Following Error Window) and  $6066_h$  (Following Error Time Out)).

## 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).

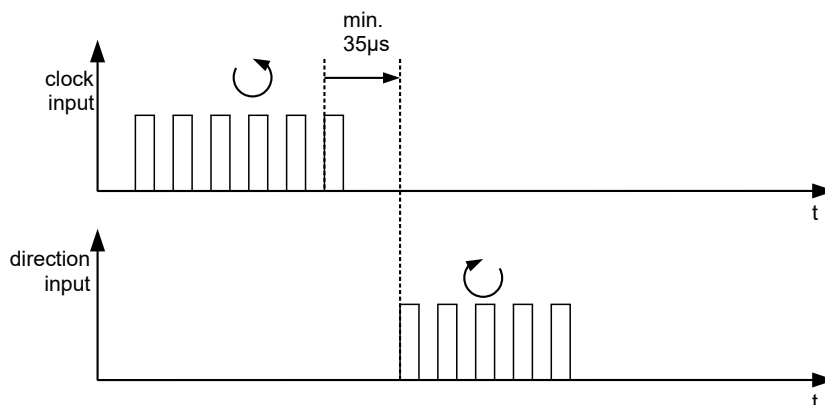
In this mode, the pulses must be preset via the clock input; the signal of the direction input specifies the direction of rotation here (see following graphic).



### 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<sub>h</sub>") must be set in object 6060<sub>h</sub> (Modes Of Operation) (see CiA 402 Power State Machine).

### 7.11.3 Controlword

The following bits in object 6040<sub>h</sub> (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<sub>h</sub> (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 is equipped with 6 digital I/O pins. Of these, 4 can be configured as either input or output. Pins *DIO5\_IO\_MISO* and *DIO6\_IO\_CLK* are preset as inputs.

#### 8.1.1 Defining input and output assignments

Digital inputs/outputs 1...4 can be freely assigned on the PCI connector strip of the device, see also [Pin assignment](#) and [3231h Flex IO Configuration](#).

- Pin 1: *DIO1\_IO\_CS*
- Pin 2: *DIO2\_CD\_CLK*
- Pin 3: *DIO3\_CD\_DIR*
- Pin 4: *DIO4\_IO\_MOSI*

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
													...	Pin 2	Pin 1

- Subindex 03<sub>h</sub> *Alternate Function Mask*: This bit mask defines whether the alternative function of the pin is to be activated.  
To activate an alternative function, set the corresponding bit to "1":

Bit	Pin	Alternative function
1	<i>DIO2_CD_CLK</i>	Clock input in clock-direction mode
2	<i>DIO3_CD_DIR</i>	Direction input in clock-direction mode

#### Note



If you do not activate the alternative function, you can no longer use the corresponding pin as a normal input/output.

If the [Clock-direction mode](#) is activated, the pins for clock and direction are automatically configured accordingly and the settings in 3231<sub>h</sub> are overwritten.

- Subindex 01<sub>h</sub> *Output Mask*: This bit mask defines whether the pin is used as input or output (depending on whether an alternative function was activated for the pin in subindex 03<sub>h</sub>):
  - Bit = "0": Pin is input (default)
  - Bit = "1": Pin is output
- Subindex 02<sub>h</sub> *Pullup Mask*: This bit mask defines whether the pin is a *pullup* or *pulldown*:
  - Bit = "0": Pin is *pulldown* (default)
  - Bit = "1": Pin is *pullup*

Subindex 02<sub>h</sub> is only active for the pin if it is defined as an input.

Example for subindex 01<sub>h</sub>: Pin 2 and pin 3 are to be outputs, value = "6" (=0110<sub>b</sub>)

1. Check which of the pins you would like to assign an alternative function to and set the corresponding bits in 3231<sub>h</sub>:03<sub>h</sub> to "1".
2. Determine which pins you would like to define as input or output.
3. Determine which inputs you would like to define as *pull down* or *pull up*.
4. Set the values in 3231<sub>h</sub>:01<sub>h</sub> and 3321<sub>h</sub>:02<sub>h</sub> accordingly.
5. Store the object by writing the value "65766173<sub>h</sub>" in 1010<sub>h</sub>:03<sub>h</sub> (see chapter [Saving objects](#)) and restart the controller.

## 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):

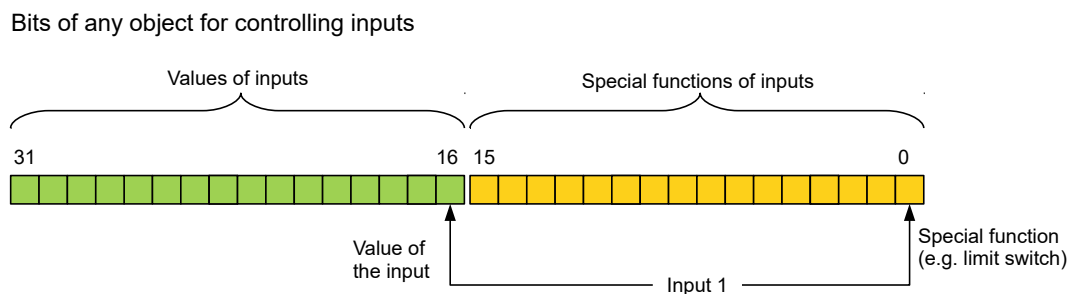
1. The first bit corresponds to the special function of an output or input. These functions are always available on bits 0 to 15 (inclusive) of the respective object. These include the limit switches and the home switch for the digital inputs and the brake control for the outputs.
2. The second bit shows 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 60FEh.

To activate the "negative limit switch" special function of input 1, set bit 0 in 3240h:01h; to query the status of the input, read bit 0 in 60FDh. Bit 16 in 60FDh 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.



### Tip



The first 4 I/O pins can also be configured as outputs, see Defining input and output assignments. If these are configured as outputs, the current status can still be read back in bits 16 to 19 of object 60FDh. The assignment of the bits in 60FDh thereby remains unchanged; bit 20 corresponds to input 5 and bit 21 to input 6.

## 8.1.3 Digital inputs

### 8.1.3.1 Overview

#### Note

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

#### Note

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:

PIN/input	Name for Input Routing
B3/DIO1_IO_CS	physical input 1
B4/DIO2_CD_CLK	physical input 2
B5/DIO3_CD_DIR	physical input 3
B6/DIO4_IO_MOSI	physical input 4
B7/DIO5_IO_MISO	physical input 5
B8/DIO6_IO_CLK	physical input 6

### 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<sub>h</sub>:01<sub>h</sub>** (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<sub>h</sub>:01<sub>h</sub>** must be set to "1".

- **3240<sub>h</sub>:02<sub>h</sub>** (Function Inverted): This subindex switches from normally open logic (a logical high level at the input yields the value "1" in object **60FD<sub>h</sub>**) to normally closed logic (the logical high level at the input yields the value "0").

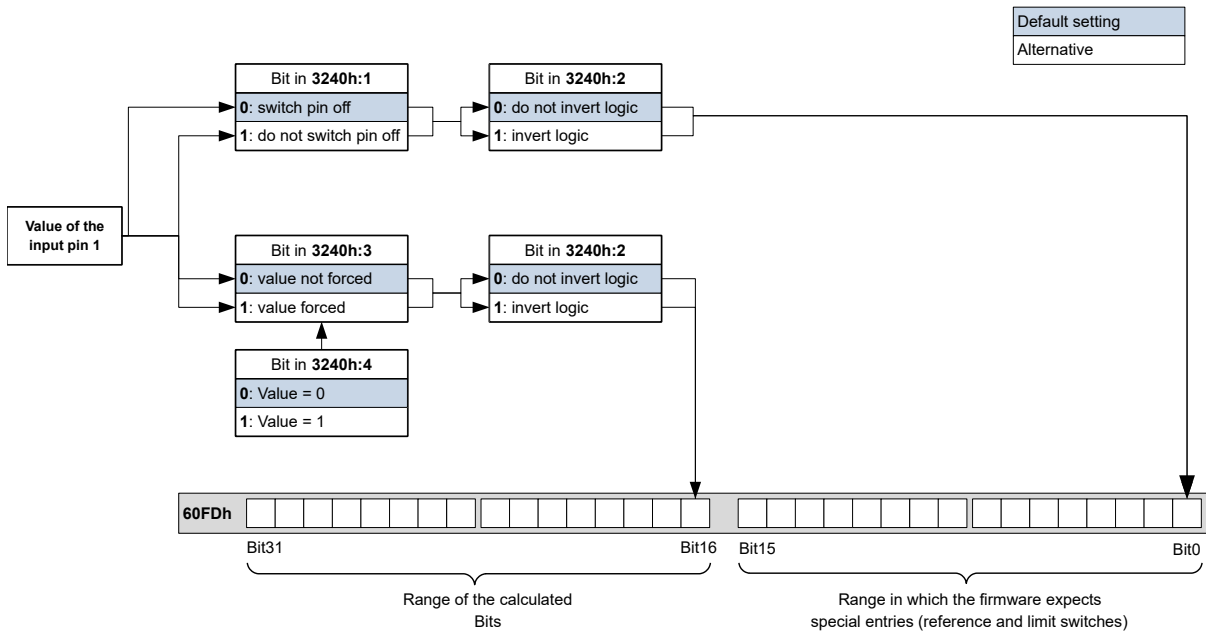
This applies for the special functions (except for the clock and direction inputs) and for the normal inputs. If the bit has the value "0", normally open logic applies; for the value "1", normally closed logic applies. Bit 0 changes the logic of input 1, bit 1 changes the logic of input 2, etc.

- **3240<sub>h</sub>:03<sub>h</sub>** (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<sub>h</sub>:04<sub>h</sub>**, but rather the set values for the respective input. Bit 0 corresponds to input 1 here, bit 1 to input 2, etc.
- **3240<sub>h</sub>:04<sub>h</sub>** (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<sub>h</sub>:03<sub>h</sub>**.
- **3240<sub>h</sub>:05<sub>h</sub>** (Raw Value): This object contains the unmodified input value.
- **60FD<sub>h</sub>** (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:

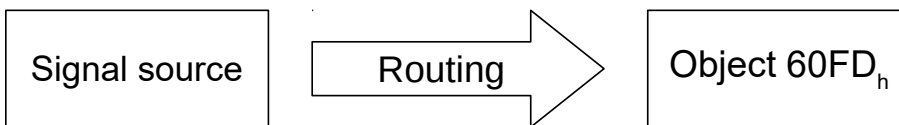
The value at bit 0 of object **60FD<sub>h</sub>** is interpreted by the firmware as negative limit switch; the result of the complete computation is stored in bit 16.



### 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<sub>h</sub>.



#### Activation

This mode is activated by setting object 3240<sub>h</sub>:08<sub>h</sub> (Routing Enable) to "1".



#### Note

Entries 3240<sub>h</sub>:01<sub>h</sub> to 3240:04<sub>h</sub> then have **no** function until Input Routing is again switched off.



#### Note

If *Input Routing* is switched on, the initial values of 3242<sub>h</sub> 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<sub>h</sub> determines which signal source is routed to which bit of 60FD<sub>h</sub>. Subindex 01<sub>h</sub> of 3242<sub>h</sub> determines bit 0, subindex 02<sub>h</sub> determines bit 1, and so forth. You can find the signal sources and their numbers in the following lists.

Number		
dec	hex	Signal source
00	00	Signal is always 0
01	01	Physical input 1
02	02	Physical input 2
03	03	Physical input 3
04	04	Physical input 4
05	05	Physical input 5
06	06	Physical input 6
07	07	Physical input 7
08	08	Physical input 8
09	09	Physical input 9
10	0A	Physical input 10
11	0B	Physical input 11
12	0C	Physical input 12
13	0D	Physical input 13
14	0E	Physical input 14
15	0F	Physical input 15
16	10	Physical input 16
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 physical input 1
130	82	Inverted physical input 2
131	83	Inverted physical input 3
132	84	Inverted physical input 4
133	85	Inverted physical input 5
134	86	Inverted physical input 6
135	87	Inverted physical input 7
136	88	Inverted physical input 8
137	89	Inverted physical input 9
138	8A	Inverted physical input 10
139	8B	Inverted physical input 11
140	8C	Inverted physical input 12
141	8D	Inverted physical input 13
142	8E	Inverted physical input 14
143	8F	Inverted physical input 15

Number		Signal source
dec	hex	
144	90	Inverted physical input 16
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 is to be routed to bit 16 of object  $60FD_h$ :

The number of the signal source for input 1 is "1". The routing for bit 16 is written in  $3242_h:11_h$ .

Hence, object  $3242_h:11_h$  must be set to the value "1".

### 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.

To activate the interlock function, you must switch on the special function by setting bit 3 in  $3240:01_h$  to "1".

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 *Input Routing*, set  $3240_h:08_h$  to "1".
2. To route input 4 to bit 3, set  $3242_h:04_h$  to "4".

## 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 first 4 I/O pins can be configured as outputs, see [Defining input and output assignments](#). 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.

### 8.1.4.2 Wiring

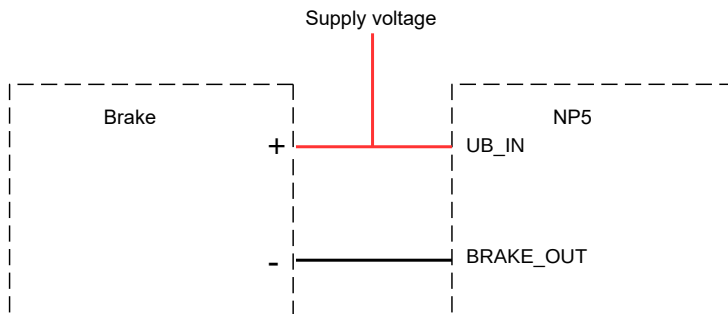


#### Note

Always observe the maximum capacity of the output (see [Pin assignment](#)).

The digital outputs, with the exception of the brake output, have a digital level of 3.3 V DC. The maximum admissible current is 10 mA.

The brake output is implemented as *open drain*. Hence, an external voltage supply as shown in the following figure is always necessary. See also [Automatic brake control](#).



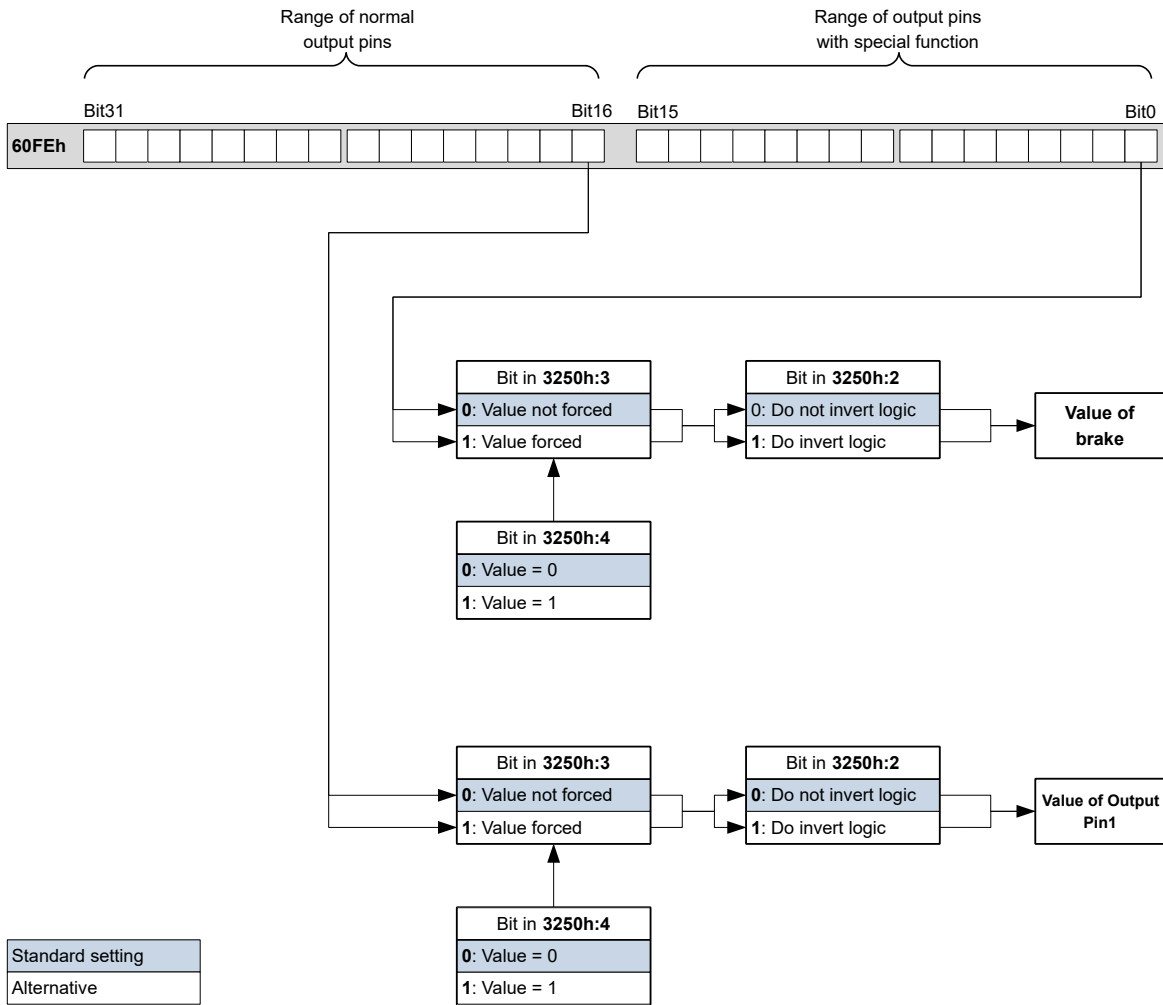
### 8.1.4.3 Object entries

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<sub>h</sub>:01<sub>h</sub>: No function.
- 3250<sub>h</sub>:02<sub>h</sub>: 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<sub>h</sub>.
- 3250<sub>h</sub>:03<sub>h</sub>: If a bit is set here, the output is controlled manually. The value for the output is then in object 3250<sub>h</sub>:4<sub>h</sub>; this is also possible for the brake output.
- 3250<sub>h</sub>:04<sub>h</sub>: 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<sub>h</sub>:03<sub>h</sub>.
- 3250<sub>h</sub>:05<sub>h</sub>: The bit combination applied to the outputs is stored in this subindex.
- 3250<sub>h</sub>:08<sub>h</sub>: For activating the [Output Routing](#).
- 3250<sub>h</sub>:09<sub>h</sub>: 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:



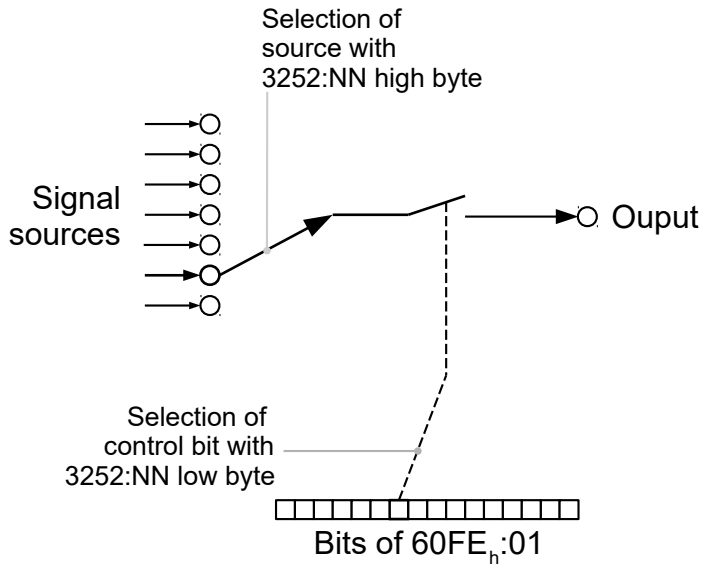
### 8.1.4.5 Output Routing

#### Principle

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

The source is selected with **3252h:01** to **05** in the "high byte" (bit 15 to bit 8). The assignment of a control bit from object **60FEh:01h** is performed in the "low byte" (bit 7 to bit 0) of **3252h:01h** to **05** (see following figure).





### Activation

This mode is activated by setting object 3250<sub>h</sub>:08<sub>h</sub> (Routing Enable) to "1".



**Note**

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

### Routing

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

Subindex <u>3252<sub>h</sub></u>	Output Pin
01 <sub>h</sub>	Configuration of the PWM output (software PWM)
02 <sub>h</sub>	Configuration of output 1
03 <sub>h</sub>	Configuration of output 2 (if available)
...	...
0n <sub>h</sub>	Configuration of output n (if available)



**Note**

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<sub>h</sub>:01<sub>h</sub> to 0n<sub>h</sub> 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<sub>h</sub>:01.

Bit 7 of 3252<sub>h</sub>:01<sub>h</sub> to 0n<sub>h</sub> inverts the controller from object 60FE<sub>h</sub>:01. Normally, value "1" in object 60FE<sub>h</sub>:01<sub>h</sub> switches on the signal; if bit 7 is set, the value "0" switches on the signal.

Number in <u>3252:01</u> to 0n	
00XX <sub>h</sub>	Output is always "1"

Number in 3252:01 to 0n	
01XX <sub>h</sub>	Output is always "0"
02XX <sub>h</sub>	Encoder signal (6063 <sub>h</sub> ) with frequency divider 1
03XX <sub>h</sub>	Encoder signal (6063 <sub>h</sub> ) with frequency divider 2
04XX <sub>h</sub>	Encoder signal (6063 <sub>h</sub> ) with frequency divider 4
05XX <sub>h</sub>	Encoder signal (6063 <sub>h</sub> ) with frequency divider 8
06XX <sub>h</sub>	Encoder signal (6063 <sub>h</sub> ) with frequency divider 16
07XX <sub>h</sub>	Encoder signal (6063 <sub>h</sub> ) with frequency divider 32
08XX <sub>h</sub>	Encoder signal (6063 <sub>h</sub> ) with frequency divider 64
09XX <sub>h</sub>	Position Actual Value (6064 <sub>h</sub> ) with frequency divider 1
0AXX <sub>h</sub>	Position Actual Value (6064 <sub>h</sub> ) with frequency divider 2
0BXX <sub>h</sub>	Position Actual Value (6064 <sub>h</sub> ) with frequency divider 4
0CXX <sub>h</sub>	Position Actual Value (6064 <sub>h</sub> ) with frequency divider 8
0DXX <sub>h</sub>	Position Actual Value (6064 <sub>h</sub> ) with frequency divider 16
0EXX <sub>h</sub>	Position Actual Value (6064 <sub>h</sub> ) with frequency divider 32
0FXX <sub>h</sub>	Position Actual Value (6064 <sub>h</sub> ) with frequency divider 64
10XX <sub>h</sub>	PWM signal that is configured with object 2038 <sub>h</sub> :05 <sub>h</sub> and 06 <sub>h</sub>
11XX <sub>h</sub>	Inverted PWM signal that is configured with object 2038 <sub>h</sub> :05 <sub>h</sub> and 06 <sub>h</sub>

#### Note



On any change of the "encoder signal" (6063<sub>h</sub>) or the current position (6064<sub>h</sub> 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<sub>h</sub>) 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<sub>h</sub>:08<sub>h</sub> = 1 (activate routing)
- 3252<sub>h</sub>:02<sub>h</sub> = 0405<sub>h</sub> (04XX<sub>h</sub> + 0005<sub>h</sub>)
- 04XX<sub>h</sub>: Encoder signal with frequency divider 4
- 0005<sub>h</sub>: 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<sub>h</sub>, this should be used as control bit.

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

## 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<sub>h</sub>](#) to "1" activates the brake control.
- Manual: Setting bit 2 of object [3202<sub>h</sub>](#) to "0" deactivates the brake control; the brake can now be controlled with bit 0 in object [60FE<sub>h</sub>:01<sub>h</sub>](#).

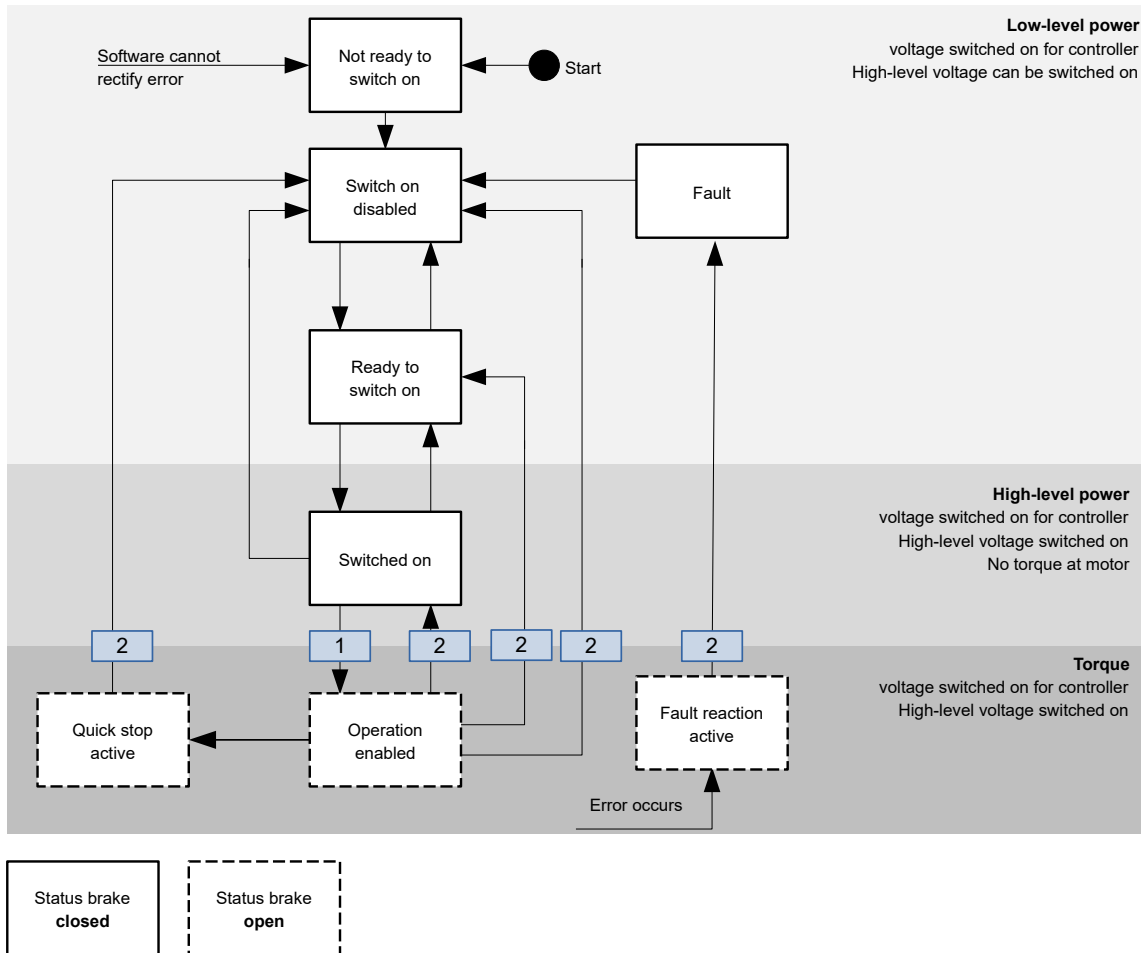
#### 8.2.2.1 Connection

The brake output is located:

- On pin A48 of the PCI connector strip; see [Pin assignment](#) and [Wiring of the outputs](#)
- On connector X2 of the Discovery Board, if this is used; see [Connector X2 – brake](#)

### 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  $2038_h:3_h$  is allowed to elapse.
3. The brake releases.
4. The time stored in  $2038_h:4_h$  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  $2038_h:1_h$  is allowed to elapse.
3. The brake is activated.
4. The time stored in  $2038_h:2_h$  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.

### Note

The *Brake +* pin of the brake output is connected to the voltage supply of the controller via the *Discovery Board*.



If the operating voltage of the brake is greater than the supply voltage of the controller, you cannot use the brake output of the controller; you must supply the brake externally.

If the supply voltage of the controller is greater than the operating voltage of the brake (and up to 48 V DC), it is recommended that the PWM controller from Nanotec with order designation *EB-BRAKE-48V* be used and the duty cycle of the controller brake output be set to "100".

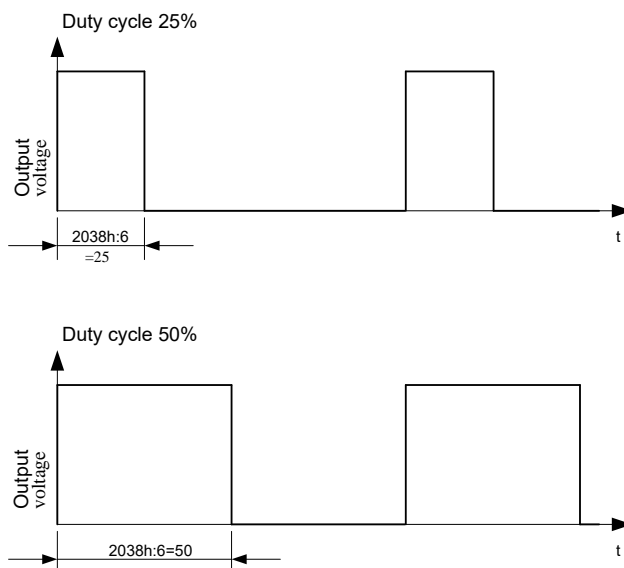
#### 8.2.4.1 Frequency

The frequency of the brake PWM can be set in object `2038h:5h`. The unit is Hertz; a value greater than 2000 is not possible.

#### 8.2.4.2 Duty cycle

The duty cycle – the ratio of pulse to period duration – is set in `2038h:6h`. 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 B27) for controlling an external ballast circuit that consists of a driver, a MOSFET as switch, and a sufficiently dimensioned ballast resistor (see [Example of a ballast circuit](#)).

#### 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 (B27, 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:

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

- $T_{Ballast\_Max}$ : 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).
- $T_{A\_Max}$ : 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 $E_{ST\_25^\circ C}$ [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 (such as 5x rated power for 5 seconds).

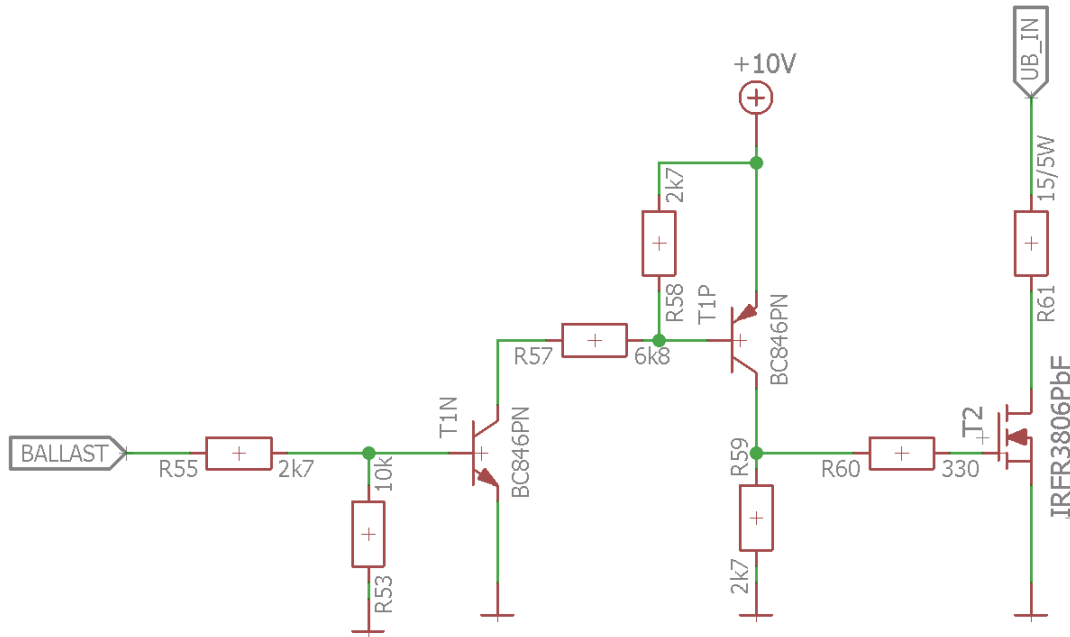
**Long Term Reference Time  $t_{LT\_Ref}$  [ms]**

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<sub>h</sub> (see 1003<sub>h</sub>).

**8.3.4 Example of a ballast circuit**

The following section from the circuit diagram of the *NP5 Discovery Board* (see Hardware installation), can serve as a reference for the development of your own ballast circuit.



The used ballast resistor is a CR257-05T15R from VITROHM. The circuit is controlled via pin *BALLAST* (B27).



**Tip**

Nanotec recommends using a wirewound resistor due to its pulse strength.

The following parameters, which you enter in the corresponding subindex of 4021<sub>h</sub>, are based on the data sheet values and the installation position on the *Discovery Board*:

Subindex	Parameter	Value
04 <sub>h</sub>	Nominal Resistance	15000
05 <sub>h</sub>	Long Term Energy Limit	60996
06 <sub>h</sub>	Long Term Reference Time	1000
07 <sub>h</sub>	Short Term Energy Limit	6375
08 <sub>h</sub>	Cooling Power	2258

## 8.4 I<sup>2</sup>t Motor overload protection

### 8.4.1 Description

#### Note



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

The goal of I<sup>2</sup>t 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<sub>h</sub> must be set to "1").

### 8.4.2 Object entries

The following objects affect I<sup>2</sup>t motor overload protection:

- 2031<sub>h</sub>: Max Motor Current – specifies the maximum permissible motor current in mA.
- 203B<sub>h</sub>:1<sub>h</sub> Motor Rated Current – specifies the rated current in mA.
- 6073<sub>h</sub> Max Current – specifies the maximum current in tenths of a percent of the set rated current.
- 203B<sub>h</sub>:2<sub>h</sub> Maximum Duration Of Peak Current – specifies the maximum duration of the maximum current in ms.

The following objects indicate the current state of I<sup>2</sup>t:

- 203B<sub>h</sub>:3<sub>h</sub> Threshold – specifies the limit in mAs that determines whether the maximum current or rated current is switched to.
- 203B<sub>h</sub>:4<sub>h</sub> CalcValue – specifies the calculated value that is compared with the threshold for setting the current.
- 203B<sub>h</sub>:5<sub>h</sub> LimitedCurrent – shows the momentary current value that was set by I<sup>2</sup>t.
- 203B<sub>h</sub>:6<sub>h</sub> Status:
  - Value = "0": I<sup>2</sup>t deactivated
  - Value = "1": I<sup>2</sup>t activated

### 8.4.3 Activation

*Closed loop* must be activated, (bit 0 of object 3202<sub>h</sub> set to "1", see also chapter Closed Loop).

To activate the mode, you must appropriately specify the four object entries mentioned above (2031<sub>h</sub>, 6073<sub>h</sub>, 203B<sub>h</sub>:1<sub>h</sub>, 203B<sub>h</sub>:2<sub>h</sub>). 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<sup>2</sup>t functionality remains deactivated.

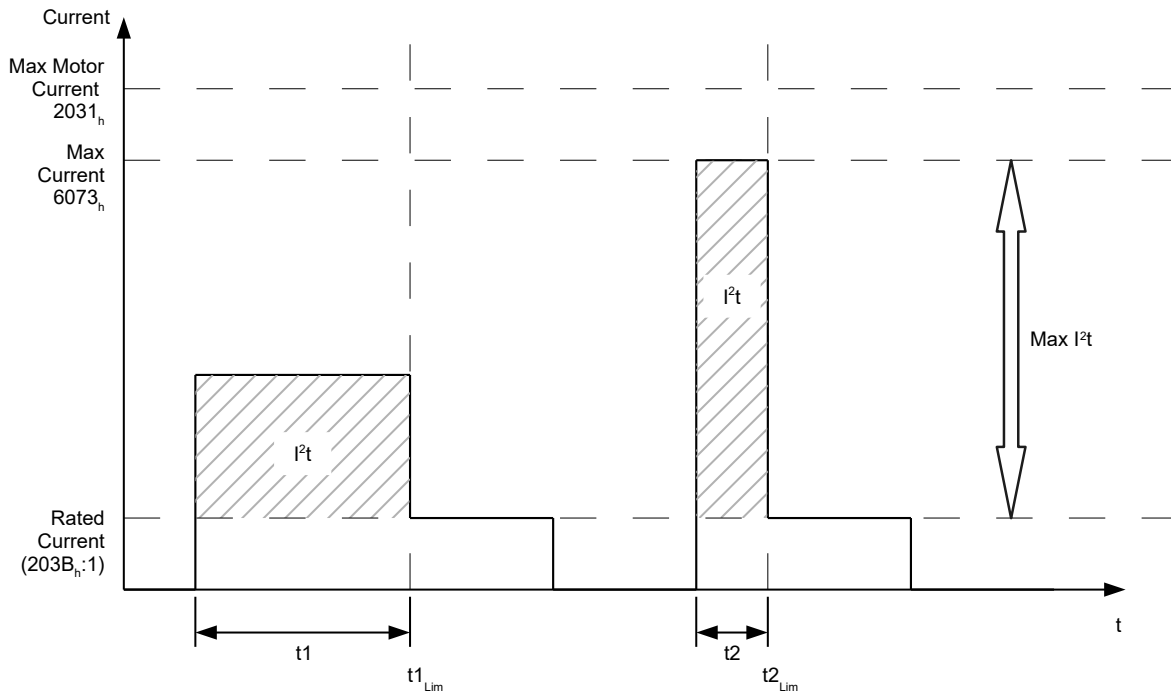
### 8.4.4 Function of I<sup>2</sup>t

From the specification of rated current, maximum current and maximum duration of the maximum current, an I<sup>2</sup>t<sub>Lim</sub> is calculated.

The motor can run with maximum current until the calculated I<sup>2</sup>t<sub>Lim</sub> is reached. The current is then immediately reduced to the rated current. The maximum current is limited by the maximum motor current (2031<sub>h</sub>).

The relationships are illustrated again in the following diagrams.





In the first section,  $t_1$ , the current value is higher than the rated current. At time  $t_{1\_Lim}$ ,  $I^2_{t\_Lim}$  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  $t_2$ . Hence, the value for  $I^2_{t\_Lim}$  is reached more quickly than in time  $t_1$ .

## 8.5 Saving objects

### Note



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.

### 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.
- Modbus RTU: Parameters related to Modbus RTU communication

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<sub>h</sub>](#): Fieldbus Module Control
- [3502<sub>h</sub>](#): MODBUS Rx PDO Mapping
- [3602<sub>h</sub>](#): MODBUS Tx PDO Mapping

### 8.5.3 Category: application

- [2034<sub>h</sub>](#): Upper Voltage Warning Level
- [2035<sub>h</sub>](#): Lower Voltage Warning Level
- [2036<sub>h</sub>](#): Open Loop Current Reduction Idle Time
- [2037<sub>h</sub>](#): Open Loop Current Reduction Value/factor
- [2038<sub>h</sub>](#): Brake Controller Timing
- [203A<sub>h</sub>](#): Homing On Block Configuration
- [203D<sub>h</sub>](#): Torque Window
- [203E<sub>h</sub>](#): Torque Window Time Out
- [203F<sub>h</sub>](#): Max Slippage Time Out
- [2057<sub>h</sub>](#): Clock Direction Multiplier
- [2058<sub>h</sub>](#): Clock Direction Divider
- [205B<sub>h</sub>](#): Clock Direction Or Clockwise/Counter Clockwise Mode
- [2084<sub>h</sub>](#): Bootup Delay
- [2290<sub>h</sub>](#): PDI Control
- [2300<sub>h</sub>](#): NanoJ Control
- [2410<sub>h</sub>](#): NanoJ Init Parameters
- [2800<sub>h</sub>](#): Bootloader And Reboot Settings
- [3210<sub>h</sub>](#): Motor Drive Parameter Set
- [3212<sub>h</sub>](#): Motor Drive Flags
- [3231<sub>h</sub>](#): Flex IO Configuration
- [3240<sub>h</sub>](#): Digital Inputs Control
- [3242<sub>h</sub>](#): Digital Input Routing
- [3243<sub>h</sub>](#): Digital Input Homing Capture
- [3250<sub>h</sub>](#): Digital Outputs Control
- [3252<sub>h</sub>](#): Digital Output Routing
- [3321<sub>h</sub>](#): Analogue Input Offset
- [3322<sub>h</sub>](#): Analogue Input Factor Numerator
- [3323<sub>h</sub>](#): Analogue Input Factor Denominator
- [3700<sub>h</sub>](#): Deviation Error Option Code
- [3701<sub>h</sub>](#): Limit Switch Error Option Code
- [4013<sub>h</sub>](#): HW Configuration
- [6040<sub>h</sub>](#): Controlword
- [6042<sub>h</sub>](#): VI Target Velocity
- [6046<sub>h</sub>](#): VI Velocity Min Max Amount
- [6048<sub>h</sub>](#): VI Velocity Acceleration
- [6049<sub>h</sub>](#): VI Velocity Deceleration
- [604A<sub>h</sub>](#): VI Velocity Quick Stop
- [604C<sub>h</sub>](#): VI Dimension Factor
- [605A<sub>h</sub>](#): Quick Stop Option Code
- [605B<sub>h</sub>](#): Shutdown Option Code
- [605C<sub>h</sub>](#): Disable Option Code
- [605D<sub>h</sub>](#): Halt Option Code
- [605E<sub>h</sub>](#): Fault Option Code

- 6060<sub>h</sub>: Modes Of Operation
- 6065<sub>h</sub>: Following Error Window
- 6066<sub>h</sub>: Following Error Time Out
- 6067<sub>h</sub>: Position Window
- 6068<sub>h</sub>: Position Window Time
- 606D<sub>h</sub>: Velocity Window
- 606E<sub>h</sub>: Velocity Window Time
- 606F<sub>h</sub>: Velocity Threshold
- 6070<sub>h</sub>: Velocity Threshold Time
- 6071<sub>h</sub>: Target Torque
- 6072<sub>h</sub>: Max Torque
- 607A<sub>h</sub>: Target Position
- 607B<sub>h</sub>: Position Range Limit
- 607C<sub>h</sub>: Home Offset
- 607D<sub>h</sub>: Software Position Limit
- 607E<sub>h</sub>: Polarity
- 607F<sub>h</sub>: Max Profile Velocity
- 6081<sub>h</sub>: Profile Velocity
- 6082<sub>h</sub>: End Velocity
- 6083<sub>h</sub>: Profile Acceleration
- 6084<sub>h</sub>: Profile Deceleration
- 6085<sub>h</sub>: Quick Stop Deceleration
- 6086<sub>h</sub>: Motion Profile Type
- 6087<sub>h</sub>: Torque Slope
- 6091<sub>h</sub>: Gear Ratio
- 6092<sub>h</sub>: Feed Constant
- 6096<sub>h</sub>: Velocity Factor
- 6097<sub>h</sub>: Acceleration Factor
- 6098<sub>h</sub>: Homing Method
- 6099<sub>h</sub>: Homing Speed
- 609A<sub>h</sub>: Homing Acceleration
- 60A2<sub>h</sub>: Jerk Factor
- 60A4<sub>h</sub>: Profile Jerk
- 60A8<sub>h</sub>: SI Unit Position
- 60A9<sub>h</sub>: SI Unit Velocity
- 60B0<sub>h</sub>: Position Offset
- 60B1<sub>h</sub>: Velocity Offset
- 60B2<sub>h</sub>: Torque Offset
- 60C1<sub>h</sub>: Interpolation Data Record
- 60C2<sub>h</sub>: Interpolation Time Period
- 60C4<sub>h</sub>: Interpolation Data Configuration
- 60C5<sub>h</sub>: Max Acceleration
- 60C6<sub>h</sub>: Max Deceleration
- 60E8<sub>h</sub>: Additional Gear Ratio - Motor Shaft Revolutions
- 60E9<sub>h</sub>: Additional Feed Constant - Feed
- 60ED<sub>h</sub>: Additional Gear Ratio - Driving Shaft Revolutions
- 60EE<sub>h</sub>: Additional Feed Constant - Driving Shaft Revolutions
- 60F2<sub>h</sub>: Positioning Option Code
- 60F8<sub>h</sub>: Max Slippage
- 60FE<sub>h</sub>: Digital Outputs
- 60FF<sub>h</sub>: Target Velocity

### 8.5.4 Category: customer

- 2701<sub>h</sub>: Customer Storage Area

### 8.5.5 Category: drive

- 3202<sub>h</sub>: Motor Drive Submode Select
- 320D<sub>h</sub>: Torque Of Inertia Factor
- 320E<sub>h</sub>: Closed Loop Controller Parameter
- 320F<sub>h</sub>: Open Loop Controller Parameter
- 6073<sub>h</sub>: Max Current
- 6080<sub>h</sub>: Max Motor Speed

### 8.5.6 Category: tuning

- 2030<sub>h</sub>: Pole Pair Count
- 2031<sub>h</sub>: Max Motor Current
- 203B<sub>h</sub>: I2t Parameters
- 3203<sub>h</sub>: Feedback Selection
- 3380<sub>h</sub>: Feedback Sensorless
- 3390<sub>h</sub>: Feedback Hall
- 33A0<sub>h</sub>: Feedback Incremental A/B/I 1
- 33A1<sub>h</sub>: Feedback Incremental A/B/I 2
- 4021<sub>h</sub>: Ballast Configuration
- 6075<sub>h</sub>: Motor Rated Current
- 608F<sub>h</sub>: Position Encoder Resolution
- 6090<sub>h</sub>: Velocity Encoder Resolution
- 60E6<sub>h</sub>: Additional Position Encoder Resolution - Encoder Increments
- 60EB<sub>h</sub>: Additional Position Encoder Resolution - Motor Revolutions

### 8.5.7 Category: Modbus RTU

- 2028<sub>h</sub>: MODBUS Slave Address
- 202A<sub>h</sub>: MODBUS RTU Baudrate
- 202D<sub>h</sub>: MODBUS RTU Parity

### 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.

#### Note



- Saving may take a few seconds. Under no circumstances may you interrupt the voltage 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<sub>h</sub>.

There is a subindex in object  $1010_h$  for each *category*. To save all objects of this *category*, the value "65766173<sub>h</sub>" must be written in the subindex.<sup>1</sup> 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 <sub>h</sub>	All categories with the exception of 0B <sub>h</sub> (Modbus RTU)
02 <sub>h</sub>	Communication
03 <sub>h</sub>	Application
04 <sub>h</sub>	Customer
05 <sub>h</sub>	Drive
06 <sub>h</sub>	Tuning
0B <sub>h</sub>	Modbus RTU

### 8.5.9 Discarding the saved data

If all objects or one *category* of saved objects is to be deleted, value "64616F6C<sub>h</sub>" must be written in object  $1011_h$ .<sup>2</sup> The following subindices correspond to a *category* here:

Subindex	Category
01 <sub>h</sub>	All categories (reset to factory settings) with the exception of 06 <sub>h</sub> (Tuning) and 0B <sub>h</sub> (Modbus RTU)
02 <sub>h</sub>	Communication
03 <sub>h</sub>	Application
04 <sub>h</sub>	Customer
05 <sub>h</sub>	Drive
06 <sub>h</sub>	Tuning
0B <sub>h</sub>	Modbus RTU

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<sub>h</sub>" in  $2800_h:01_h$ .

#### Note



- Objects of *category* 06<sub>h</sub> (Tuning) are determined by Auto setup and are not reset when resetting to factory settings with subindex 01<sub>h</sub> (thereby making it unnecessary to again perform an auto setup). You can reset these objects with subindex 06<sub>h</sub>.
- Objects of *category* 0B<sub>h</sub> (Modbus RTU) are not reset with subindex 01<sub>h</sub>.

### 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  $1010_h:01$ .

<sup>1</sup> This corresponds to the decimal of 1702257011<sub>d</sub> or the ASCII string `save`.

<sup>2</sup> This corresponds to the decimal of 1684107116<sub>d</sub> or the ASCII string `load`.

The entries of 1020<sub>h</sub> are reset to "0" as soon as a savable object (including 1010<sub>h</sub>:0x<sub>h</sub>, except for 1010<sub>h</sub>:01<sub>h</sub> and 1020<sub>h</sub>) 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<sub>h</sub>.
3. The tool or master activates the saving of all objects 1010<sub>h</sub>:01<sub>h</sub> = 65766173<sub>h</sub>. The date and time in object 1020<sub>h</sub> are also saved.

After the controller is restarted, the master can check the value in 1020<sub>h</sub>:01<sub>h</sub> and 1020:01<sub>h</sub>. 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 RTU

Modbus references: [www.modbus.org](http://www.modbus.org).

- MODBUS APPLICATION PROTOCOL SPECIFICATION V1.1b3, Date: 26.04.2014, Version: 1.1b3
- MODBUS over Serial Line Specification and Implementation Guide V1.02, Date: 20.12.2006, Version: 1.02

The controller can be controlled by means of Modbus RTU. The I/O data, with, e.g., the preconfigured drive values (see [Process data objects \(PDO\)](#)), can be handled with the standard Modbus function codes. To configure your own I/O data, however, function code 2Bh (CAN Encapsulation) must be supported by the 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 by means of *Plug & Drive interface* is possible (see document *Function description Plug & Drive interface*).

### 9.1 RS-485

The "Two-Wire Modbus Interface" electrical interface is supported by the controller in accordance with standard EIA/

TIA-485 (RS-485) (see chapter [Connecting Modbus RTU](#) for further information).

### 9.2 Modbus Modicon notation with PLCs

Many PLCs use the Modicon addressing model. This notation is not used in the Modbus standard.

The following address notation is relevant for Nanotec controllers:

- Input register 30001 - 39999 is mapped to Modbus telegram address 0 (0<sub>h</sub>) - 9998 (270E<sub>h</sub>).
- Holding register 40001 - 49999 is mapped to Modbus telegram address 0 (0<sub>h</sub>) - 9998 (270E<sub>h</sub>).

**Note**



Where Modbus addresses are mentioned in the manual, it may be necessary to implement the register addresses in the PLC in accordance with *Modicon notation*.

### 9.3 General

Modbus is generally big-endian based.

The only exceptions are the commands with function codes 43 (2B<sub>h</sub>), 101 (65<sub>h</sub>) and 102 (66<sub>h</sub>), 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<sub>h</sub>: With this command, the value 12345678<sub>h</sub> is written in object 0123<sub>h</sub> (does not exist):

SA	FC	Data	CRC
05	2B	0D 01 00 01 23 01 00 00 00 00 04 <b>78 56 34 12</b>	67 35

**SA**

Slave address

**FC**

Function code

**Data**

Data range, decoding is dependent on the used function code

**CRC**

Cyclic redundancy check

**Note**

In case of more than one Modbus slaves in the network, the Modbus master must wait for at least 3 ms after receiving a response before sending the next request.

## 9.4 Communication settings

The following settings can be performed:

Configuration	Object	Value range	Factory settings
Slave address	<u>2028</u> <sub>h</sub>	1 to 247	5
Baud rate	<u>202A</u> <sub>h</sub>	7200 to 256000	19200
Parity	<u>202D</u> <sub>h</sub>	<ul style="list-style-type: none"> <li>■ None: 0x00</li> <li>■ Even: 0x04</li> <li>■ Odd: 0x06</li> </ul>	0x04 (Even)

The number of data bits is always "8" here. The number of stop bits is dependent on the parity setting:

- No parity: 2 stop bits
- "Even" or "Odd" parity: 1 stop bit

The following baud rates are supported:

- 7200
- 9600
- 14400
- 19200
- 38400
- 56000
- 57600
- 115200
- 128000
- 256000

The slave address and baud rate can also be defined via the analog input, see [Setting slave address and baud rate](#)

You must save the changes by writing value "65766173<sub>h</sub>" in object 1010<sub>h</sub>:0B<sub>h</sub>. The changes are not taken over until after the controller has been restarted.

## 9.5 Function codes

The following "function codes" are supported:



	Name	Function code	Subfunction code
Data access (16-bit)	Read Holding Registers	03 (03 <sub>h</sub> )	
	Read Input Register	04 (04 <sub>h</sub> )	
	Write Single Register	06 (06 <sub>h</sub> )	
	Write Multiple Registers	16 (10 <sub>h</sub> )	
	Read/Write Multiple Registers	23 (17 <sub>h</sub> )	
Diagnosis	Clear Counters and Diagnostic Register	08 (08 <sub>h</sub> )	10 (0A <sub>h</sub> )
	Return Bus Message Count	08 (08 <sub>h</sub> )	11 (0B <sub>h</sub> )
	Return Bus Communication Error Count	08 (08 <sub>h</sub> )	12 (0C <sub>h</sub> )
	Return Bus Exception Error Count	08 (08 <sub>h</sub> )	13 (0D <sub>h</sub> )
	Return Server Message Count	08 (08 <sub>h</sub> )	14 (0E <sub>h</sub> )
	Return Server No Response Count	08 (08 <sub>h</sub> )	15 (0F <sub>h</sub> )
	Return Server NAK Count	08 (08 <sub>h</sub> )	16 (10 <sub>h</sub> )
	Return Server Busy Count	08 (08 <sub>h</sub> )	17 (11 <sub>h</sub> )
	Return Bus Character Overrun Count	08 (08 <sub>h</sub> )	18 (12 <sub>h</sub> )
Miscellaneous	Encapsulated Interface Transport	43 (2B <sub>h</sub> )	13 (0D <sub>h</sub> )
	Read complete object dictionary start	101 (65 <sub>h</sub> )	85 (55 <sub>h</sub> )
	Read complete object dictionary next	101 (65 <sub>h</sub> )	170 (AA <sub>h</sub> )
	Read complete array or record start	102 (66 <sub>h</sub> )	85 (55 <sub>h</sub> )
	Read complete array or record next	102 (66 <sub>h</sub> )	170 (AA <sub>h</sub> )

## 9.6 Function code descriptions

### 9.6.1 FC 3 (03<sub>h</sub>) Read Input Registers / FC 4 (04<sub>h</sub>) 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
Slave address	1 byte	
Function code	1 byte	03 <sub>h</sub> / 04 <sub>h</sub>
Start address	2 bytes	0000 <sub>h</sub> to FFFF <sub>h</sub>
Number of registers	2 bytes	1 to (7D <sub>h</sub> )
CRC	2 bytes	

Response ("M" corresponds to the number of registers to be read)		
Name	Length	Value
Slave address	1 byte	
Function code	1 byte	03 <sub>h</sub> / 04 <sub>h</sub>
Number of bytes	1 byte	2 * M
Register value	2 bytes	
CRC	2 bytes	

Error		
Name	Length	Value
Slave address	1 byte	
Error code	1 byte	83 <sub>h</sub> / 84 <sub>h</sub>
Exception code (see <a href="#">Exception codes</a> )	1 byte	01, 02, 03 or 04
CRC	2 bytes	

### Example

Below is an example of a read request and response of register 5000 (1388<sub>h</sub>) and of the following register (2 registers):

#### Request

SA	FC	Data	CRC
05	03	13 88 00 02	41 21

#### Response

SA	FC	Data	CRC
05	03	04 02 40 00 00	41 21

## 9.6.2 FC 6 (06<sub>h</sub>) 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
Slave address	1 byte	
Function code	1 byte	06 <sub>h</sub>
Register address	2 bytes	0000 <sub>h</sub> to FFFF <sub>h</sub>
Register value	2 bytes	0000 <sub>h</sub> to FFFF <sub>h</sub>
CRC	2 bytes	

Response		
Name	Length	Value
Slave address	1 byte	
Function code	1 byte	06 <sub>h</sub>
Register address	2 bytes	0000 <sub>h</sub> to FFFF <sub>h</sub>
Register value	2 bytes	0000 <sub>h</sub> to FFFF <sub>h</sub>
CRC	2 bytes	

Error		
Name	Length	Value
Slave address	1 byte	
Error code	1 byte	86 <sub>h</sub>
Exception code (see <a href="#">Exception codes</a> )	1 byte	01, 02, 03 or 04
CRC	2 bytes	

### Example

Below is an example of a write request and response in register 6000 (1770<sub>h</sub>) with the value "0001<sub>h</sub>":

#### Request

SA	FC	Data	CRC
05	06	17 70 00 01	4D E1

#### Response

SA	FC	Data	CRC
05	06	17 70 00 01	4D E1

### 9.6.3 FC 16 (10<sub>h</sub>) 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
Slave address	1 byte	
Function code	1 byte	10 <sub>h</sub>
Start address	2 bytes	0000 <sub>h</sub> to FFFF <sub>h</sub>
Number of registers	2 bytes	0001 <sub>h</sub> to 007B <sub>h</sub>
Number of bytes	1 byte	2 * N
Register value	N * 2 bytes	
CRC	2 bytes	

Response		
Name	Length	Value
Slave address	1 byte	
Function code	1 byte	10 <sub>h</sub>
Start address	2 bytes	0000 <sub>h</sub> to FFFF <sub>h</sub>
Number of registers	2 bytes	0001 <sub>h</sub> to 007B <sub>h</sub>
CRC	2 bytes	

Error		
Name	Length	Value
Slave address	1 byte	
Error code	1 byte	90 <sub>h</sub>
Exception code (see <a href="#">Exception codes</a> )	1 byte	01, 02, 03 or 04
CRC	2 bytes	

**Example**

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

**Request**

SA	FC	Data	CRC
05	10	17 70 00 02 04 01 02 03 04	AB 44

**Response**

SA	FC	Data	CRC
05	10	17 70 00 02	44 23

**9.6.4 FC 17 (11<sub>h</sub>) Report Server ID**

This function code can be used to read the description of the type, the current status and other information about the device.

Request		
Name	Length	Value
Slave address	1 byte	
Function code	1 byte	11 <sub>h</sub>
CRC	2 bytes	

Response		
Name	Length	Value
Slave address	1 byte	
Function code	1 byte	03 <sub>h</sub>
Number of bytes	1 byte	01 <sub>h</sub>
Run Indicator Status	1 byte	00 <sub>h</sub> = OFF, FF <sub>h</sub> = ON
Additional data		
CRC	2 bytes	

Error		
Name	Length	Value
Slave address	1 byte	

Error		
Name	Length	Value
Error code	1 byte	91 <sub>h</sub>
Exception code (see <a href="#">Exception codes</a> )	1 byte	01 or 04
CRC	2 bytes	

### Example

Below is an example of a request/response for ID and status:

#### Request

SA	FC	CRC
05	11	C2 EC

#### Response

SA	FC	Data	CRC
05	11	02 05 FF	0F EC

### 9.6.5 FC 23 (17<sub>h</sub>) 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
Slave address	1 byte	
Function code	1 byte	17 <sub>h</sub>
Read: Start address	2 bytes	0000 <sub>h</sub> to FFFF <sub>h</sub>
Read: Number of registers	2 bytes	0001 <sub>h</sub> to 0079 <sub>h</sub>
Write: Start address	2 bytes	0000 <sub>h</sub> to FFFF <sub>h</sub>
Write: Number of registers	2 bytes	0001 <sub>h</sub> to 0079 <sub>h</sub>
Write: Number of bytes	1 byte	2 * N
Write: Register value	N * 2 bytes	
CRC	2 bytes	

Response ("M" corresponds to the number of bytes to be written):		
Name	Length	Value
Slave address	1 byte	
Function code	1 byte	17 <sub>h</sub>
Number of bytes	1 byte	2 * M
Registers read	M * 2 bytes	
CRC	2 bytes	

Error		
Name	Length	Value
Slave address	1 byte	
Error code	1 byte	97 <sub>h</sub>
Exception code (see <a href="#">Exception codes</a> )	1 byte	01, 02, 03 or 04
CRC	2 bytes	

**Example**

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

**Request**

SA	FC	Data	CRC
05	17	13 88 00 02 17 70 00 02 04 01 02 03 04	56 6A

**Response**

SA	FC	Data	CRC
05	17	04 02 40 00 00	0F EC

**9.6.6 FC 8 (08<sub>h</sub>) Diagnostics**

Modbus function code FC08 offers numerous tests for checking the communication system between client and server or for checking various internal error states within the server.

This function uses a two-byte subfunction code in the request for defining the type of test. In a normal response, the server repeats both, the function and the subfunction code. Some diagnoses contain data of the device in the data field of the normal response.

Request:

Name	Length	Value
Function code	1 byte	08 <sub>h</sub>
Subfunction code	2 bytes	
Data	N x 2 bytes	

Response:

Name	Length	Value
Function code	1 byte	08 <sub>h</sub>
Subfunction code	2 bytes	
Data	N x 2 bytes	

Error:

Name	Length	Value
Function code	1 byte	88 <sub>h</sub>
Exception code (see <a href="#">Exception codes</a> )	1 byte	01 or 03 or 04

### 9.6.6.1 FC 8.10 (08<sub>h</sub>.0A<sub>h</sub>) Clear Counters and Diagnostic Register

The objective of this request is to reset all counters and diagnosis registers. Counters are also reset when the controller is switched on.

Subfunction	Data range	
	Request	Response
00 <sub>h</sub> 0A <sub>h</sub>	00 <sub>h</sub> - 00 <sub>h</sub>	Echo of the request data

#### Example

##### Request

SA	FC	Data	CRC
05	08	00 0A 00 00	56 6A

##### Response

SA	FC	Data	CRC
05	08	00 0A 00 00	C1 8D

### 9.6.6.2 FC 8.11 (08<sub>h</sub>.0B<sub>h</sub>) Return Bus Message Count

The response data range returns the number of messages detected by the communications system since the last restart, "Clear Counters and Diagnostic Register" request, or switching on of the controller.

Subfunction	Data range	
	Request	Response
00 <sub>h</sub> 0B <sub>h</sub>	00 <sub>h</sub> - 00 <sub>h</sub>	Total Message Count

### 9.6.6.3 FC 8.12 (08<sub>h</sub>.0C<sub>h</sub>) Return Bus Communication Error Count

The response data range returns the number of CRC errors since the last restart, "Clear Counters and Diagnostic Register" request, or switching on of the controller.

Subfunction	Data range	
	Request	Response
00 <sub>h</sub> 0C <sub>h</sub>	00 <sub>h</sub> - 00 <sub>h</sub>	CRC Error Count

**Example****Request**

SA	FC	Data	CRC
05	08	00 0C 00 00	21 8C

**Response**

SA	FC	Data	CRC
05	08	00 0C 00 00	21 8C

**9.6.6.4 FC 8.13 (08<sub>h</sub>.0D<sub>h</sub>) Return Bus Exception Error Count**

The response data range returns the number of Modbus exceptions since the last restart, "Clear Counters and Diagnostic Register" request, or switching on of the controller.

Subfunction	Data range	
	Request	Response
00 <sub>h</sub> 0D <sub>h</sub>	00 <sub>h</sub> - 00 <sub>h</sub>	Exception Error Count

**Example****Request**

SA	FC	Data	CRC
05	08	00 0D 00 00	70 4C

**Response**

SA	FC	Data	CRC
05	08	00 0D 00 00	70 4C

**9.6.6.5 FC 8.14 (08<sub>h</sub>.0E<sub>h</sub>) Return Server Message Count**

The response data range returns the number of messages addressed to the device and the number of broadcast messages that were processed by the controller. The number of messages since the last restart, "Clear Counters and Diagnostic Register" request, or switching on of the controller are counted.

Subfunction	Data range	
	Request	Response
00 <sub>h</sub> 0E <sub>h</sub>	00 <sub>h</sub> - 00 <sub>h</sub>	Server Message Count



**Example**

**Request**

SA	FC	Data	CRC
05	08	00 0E 00 00	80 4C

**Response**

SA	FC	Data	CRC
05	08	00 0E 00 00	80 4C

**9.6.6.6 FC 8.15 (08<sub>h</sub>.0F<sub>h</sub>) Return Server No Response Count**

The response data range returns the number of messages addressed to the controller for which no response was returned (neither normal response nor exception response). The number of messages since the last restart, "Clear Counters and Diagnostic Register" request, or switching on of the controller are counted.

Subfunction	Data range	
	Request	Response
00 <sub>h</sub> 0F <sub>h</sub>	00 <sub>h</sub> - 00 <sub>h</sub>	No Response Count

**Example**

**Request**

SA	FC	Data	CRC
05	08	00 0F 00 00	D1 8C

**Response**

SA	FC	Data	CRC
05	08	00 0F 00 00	D1 8C

**9.6.6.7 FC 8.16 (08<sub>h</sub>.10<sub>h</sub>) Return Server NAK Count**

The response data range returns the number of messages for which a "Negative Acknowledge (NAK)" exception response was returned. The number of messages since the last restart, "Clear Counters and Diagnostic Register" request, or switching on of the controller are counted.

Subfunction	Data range	
	Request	Response
00 <sub>h</sub> - 10 <sub>h</sub>	00 <sub>h</sub> - 00 <sub>h</sub>	Server NAK Count

**Example****Request**

SA	FC	Data	CRC
05	08	00 10 00 00	E0 4A

**Response**

SA	FC	Data	CRC
05	08	00 10 00 00	E0 4A

**9.6.6.8 FC 8.17 (08<sub>h</sub>.11<sub>h</sub>) Return Server Busy Count**

The response data range returns the number of messages for which a "Server Device Busy" exception response was returned. The number of messages since the last restart, "Clear Counters and Diagnostic Register" request, or switching on of the controller are counted.

Subfunction	Data range	
	Request	Response
00 <sub>h</sub> - 11 <sub>h</sub>	00 <sub>h</sub> - 00 <sub>h</sub>	Server NAK Count

**Example****Request**

SA	FC	Data	CRC
05	08	00 11 00 00	B1 8A

**Response**

SA	FC	Data	CRC
05	08	00 11 00 00	B1 8A

**9.6.6.9 FC 8.18 (08<sub>h</sub>.12<sub>h</sub>) Return Bus Character Overrun Count**

The response data range returns the number of messages addressed to the controller that could not be processed due to a character overrun. The number of messages since the last restart, "Clear Counters and Diagnostic Register" request, or switching on of the controller are counted. A character overrun occurs when characters arrive at the controller faster than they can be stored or by the loss of a character due to a hardware malfunction.

Subfunction	Data range	
	Request	Response
00 <sub>h</sub> - 12 <sub>h</sub>	00 <sub>h</sub> - 00 <sub>h</sub>	Server Character Overrun Count

**Example****Request**

SA	FC	Data	CRC
05	08	00 12 00 00	41 8A

**Response**

SA	FC	Data	CRC
05	08	00 12 00 00	41 8A

### 9.6.7 FC 43 (2B<sub>h</sub>) 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

<b>Note</b>
-------------



For the messages of the Encapsulated Interface Transport, another byte sequence applies in part, see chapter [General](#).

Definition of the request and response:

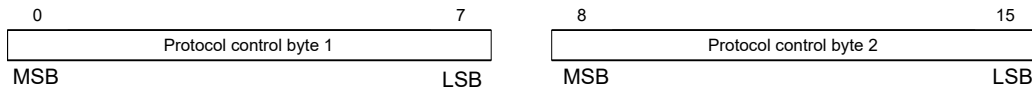
Name	Length	Example/number range
Slave address	1 byte	
Function code	1 byte	2B <sub>h</sub> (43 <sub>d</sub> )
MEI type	1 byte	0D <sub>h</sub> (13 <sub>d</sub> )
Protocol options Range	2 to 5 bytes	
Address and data range	N bytes	
CRC	2 bytes	

#### 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".
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

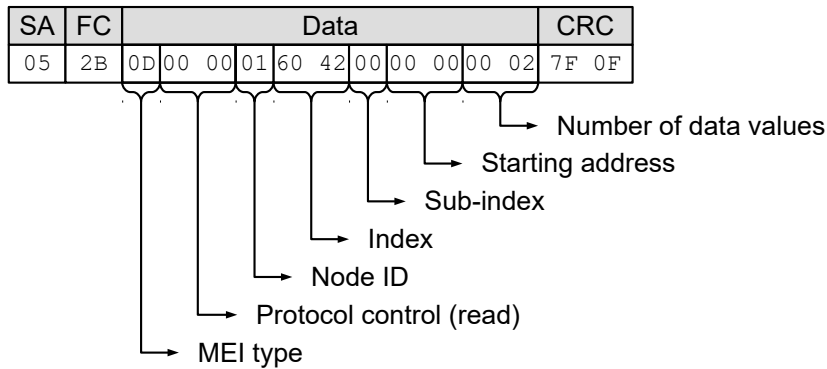
The address and data range is defined in the following table:

Name	Byte size and byte order	Example / range
Node-ID	1 byte	01 <sub>h</sub> to 7F <sub>h</sub>
Index	1 byte, high	0000 <sub>h</sub> to FFFF <sub>h</sub>
	1 byte, low	
Subindex	1 byte	00 <sub>h</sub> to FF <sub>h</sub>
Start address	1 byte, high	0000 <sub>h</sub> to FFFF <sub>h</sub>
	1 byte, low	
Number of data values	1 byte, high	0000 <sub>h</sub> to 00FD <sub>h</sub>
	1 byte, low	
Write/read data	n bytes	The data are encoded as described in chapter <a href="#">General</a> .

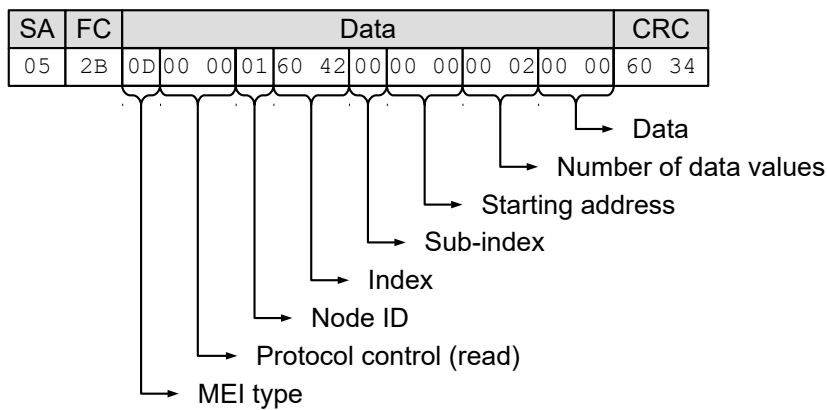
### Example:

To read object 6042<sub>h</sub>:00<sub>h</sub> (16-bit value), the following message must be sent by the master (all values are in hexadecimal notation, the slave ID of the controller is "5").

**Request**



**Response**



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<sub>h</sub>" (Velocity mode)**

**Request**

SA	FC	Data										CRC			
05	2B	0D	01	00	01	60	60	00	00	00	00	01	02	C9	2F

**Response**

SA	FC	Data										CRC			
05	2B	0D	01	00	01	60	60	00	00	00	00	00	00	A9	89

**Set 2031 = 03E8<sub>h</sub>" (1000 mA)**

**Request**

SA	FC	Data										CRC						
05	2B	0D	01	00	01	20	31	00	00	00	00	04	E8	03	00	00	C3	53

**Response**

SA	FC	Data	CRC
05	2B	0D 01 00 01 20 31 00 00 00 00 00	E5 CC

Set 6040 = "00<sub>h</sub>"**Request**

SA	FC	Data	CRC
05	2B	0D 01 00 01 60 40 00 00 00 00 02 00 00	1C 2E

**Response**

SA	FC	Data	CRC
05	2B	0D 01 00 01 60 40 00 00 00 00 00	AE E9

Set 6040 = "80<sub>h</sub>"**Request**

SA	FC	Data	CRC
05	2B	0D 01 00 01 60 40 00 00 00 00 02 80 00	7D EE

**Response**

SA	FC	Data	CRC
05	2B	0D 01 00 01 60 40 00 00 00 00 00	AE E9

Set 6040 = "06<sub>h</sub>"**Request**

SA	FC	Data	CRC
05	2B	0D 01 00 01 60 40 00 00 00 00 02 06 00	1F 8E

**Response**

SA	FC	Data	CRC
05	2B	0D 01 00 01 60 40 00 00 00 00 00	AE E9

Set 6040 = "07<sub>h</sub>"**Request**

SA	FC	Data	CRC
05	2B	0D 01 00 01 60 40 00 00 00 00 02 07 00	1E 1E

**Response**

SA	FC	Data	CRC
05	2B	0D 01 00 01 60 40 00 00 00 00 00	AE E9

Set **6040** = "0F<sub>h</sub>"

**Request**

SA	FC	Data	CRC
05	2B	0D 01 00 01 60 40 00 00 00 00 02 0F 00	19 DE

**Response**

SA	FC	Data	CRC
05	2B	0D 01 00 01 60 40 00 00 00 00 00	AE E9

Below are two examples for reading an object:

Read **6041<sub>h</sub>:00<sub>h</sub>**

**Request**

SA	FC	Data	CRC
05	2B	0D 00 00 01 60 41 00 00 00 00 02	7F 3C

**Response**

SA	FC	Data	CRC
05	2B	0D 00 00 01 60 41 00 00 00 00 02 37 06	B6 13

Read **6061<sub>h</sub>:00<sub>h</sub>**

**Request**

SA	FC	Data	CRC
05	2B	0D 00 00 01 60 61 00 00 00 00 01	38 5D

**Response**

SI	FC	Data	CRC
05	2B	0D 00 00 01 60 61 00 00 00 00 01 00	5C D2

**9.6.7.1 Error reaction**

In the event of an error, the following error message is sent:

Name	Length	Example value
Slave address	1 byte	

Name	Length	Example value
Function code	1 byte	2B <sub>h</sub> +80 <sub>h</sub> (171 <sub>d</sub> = 43 <sub>d</sub> + 128 <sub>d</sub> ) (indicates error)
Modbus exception code	1 byte	FF <sub>h</sub> ("extended exception")
Extended exception length	2 bytes	6
MEI type	1 byte	0D <sub>h</sub>
Exception code	1 byte	CE <sub>h</sub>
Error code	4 bytes	CANopen error code, see following table
CRC	2 bytes	

CANopen error code	Description
FFFF0000 <sub>h</sub>	<i>Abort no error</i>
FFFF1003 <sub>h</sub>	Service is not supported
FFFF1004 <sub>h</sub>	Gap in counter byte of the <i>Protocol control</i> field
FFFF0003 <sub>h</sub>	Unknown or invalid command
FFFF0008 <sub>h</sub>	Access to the object is not supported
FFFF000E <sub>h</sub>	General error in the parameter
FFFF0011 <sub>h</sub>	Length of parameter incorrect
FFFF0012 <sub>h</sub>	Parameter too long
FFFF0013 <sub>h</sub>	Parameter too short
FFFF0015 <sub>h</sub>	Parameter data outside of the permissible value range (for write commands)
FFFF0016 <sub>h</sub>	Parameter data exceed the permissible value range (for write commands)
FFFF0017 <sub>h</sub>	Parameter data below the permissible value range (for write commands)
FFFF0018 <sub>h</sub>	Maximum entered values less than minimum values
FFFF0019 <sub>h</sub>	General error
FFFF001E <sub>h</sub>	Requested object is too large for single message
FFFF1004 <sub>h</sub>	Invalid sequence of messages (e. g., if the value of the <i>counter byte</i> 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
Slave address	1 byte	
Function code	1 byte	2B <sub>h</sub> +80 <sub>h</sub> (171 <sub>d</sub> = 43 <sub>d</sub> + 128 <sub>d</sub> ) (indicates error)
Modbus exception code	1 byte	FF <sub>h</sub> ("extended exception")
Extended exception length	2 bytes	2 + length of "supported protocol control"
MEI type	1 byte	0D <sub>h</sub>
Exception code	1 byte	AE <sub>h</sub>
Supported protocol control	1 or 2 bytes	See following table
CRC	2 bytes	



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 `6061h:00` with a length of 2 bytes, but the object has a size of just 1 byte:

#### Request

SA	FC	Data	CRC
05	2B	0D 00 00 01 60 60 00 00 00 00 02	79 8D

#### Response

SA	FC	Data	CRC
05	2B	FF 00 06 0D CE 12 00 07 06	AC 3C

### 9.6.8 FC 101 (65<sub>h</sub>) 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 `55h` must be sent. This code resets reading out of the object dictionary on object `0000h`. All subsequent object dictionary frames must then contain subfunction code `AAh`. 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	<code>65<sub>h</sub></code>
Subfunction code	1 byte	<code>55<sub>h</sub></code> or <code>AA<sub>h</sub></code>
Length of the data	1 byte	<code>00<sub>h</sub></code>
CRC	2 bytes	

**Response:**

Name	Length	Value / note
Slave address	1 byte	65 <sub>h</sub>
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

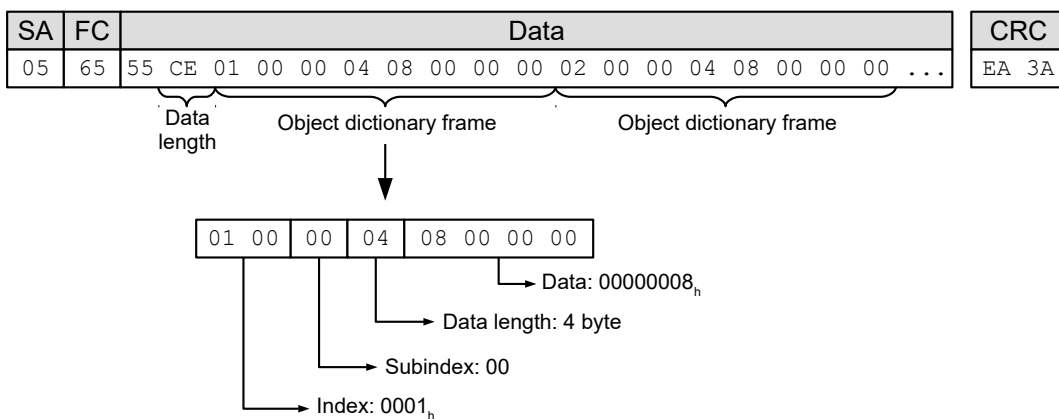
**Example**

All of the following numerical values are in hexadecimal format. The address of the slave is "5".

Start reading of the object dictionary with request:

SA	FC	Data	CRC
05	65	55 00	2F A7

The response is:



Read out the next part of the object dictionary with the request:

SA	FC	Data	CRC
05	65	AA 00	6E 57

The response is:

SA	FC	Data	CRC
05	65	AA CD 21 00 0A 02 07 00 21 00 0B 02 07 00 21 00 0C 02 ...	NN NN

Repeat reading of the object dictionary with the previous request until the response is an error:

SA	FC	Data	CRC
05	E5	0D	EA 94

### 9.6.8.1 Error reaction

In the event of an error, the following error message is sent:

Name	Length	Example value
Slave address	1 byte	
Function code	1 byte	2B <sub>h</sub> +80 <sub>h</sub> (171 <sub>d</sub> = 43 <sub>d</sub> + 128 <sub>d</sub> ) (indicates error)
Modbus exception code	1 byte	FF <sub>h</sub> ("extended exception")
Extended exception length	2 bytes	6
MEI type	1 byte	0D <sub>h</sub>
Exception code	1 byte	CE <sub>h</sub>
Error code	4 bytes	CANopen error code, see following table
CRC	2 bytes	

CANopen error code	Description
FFFF0000 <sub>h</sub>	<i>Abort no error</i>
FFFF1003 <sub>h</sub>	Service is not supported
FFFF1004 <sub>h</sub>	Gap in counter byte of the <i>Protocol control</i> field
FFFF0003 <sub>h</sub>	Unknown or invalid command
FFFF0008 <sub>h</sub>	Access to the object is not supported
FFFF000E <sub>h</sub>	General error in the parameter
FFFF0011 <sub>h</sub>	Length of parameter incorrect
FFFF0012 <sub>h</sub>	Parameter too long
FFFF0013 <sub>h</sub>	Parameter too short
FFFF0015 <sub>h</sub>	Parameter data outside of the permissible value range (for write commands)
FFFF0016 <sub>h</sub>	Parameter data exceed the permissible value range (for write commands)
FFFF0017 <sub>h</sub>	Parameter data below the permissible value range (for write commands)
FFFF0018 <sub>h</sub>	Maximum entered values less than minimum values
FFFF0019 <sub>h</sub>	General error
FFFF001E <sub>h</sub>	Requested object is too large for single message
FFFF1004 <sub>h</sub>	Invalid sequence of messages (e. g., if the value of the <i>counter byte</i> 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
Slave address	1 byte	
Function code	1 byte	2B <sub>h</sub> + 80 <sub>h</sub> (171 <sub>d</sub> = 43 <sub>d</sub> + 128 <sub>d</sub> ) (indicates error)
Modbus exception code	1 byte	FF <sub>h</sub> ("extended exception")
Extended exception length	2 bytes	2 + length of "supported protocol control"
MEI type	1 byte	0D <sub>h</sub>
Exception code	1 byte	AE <sub>h</sub>
Supported protocol control	1 or 2 bytes	See following table
CRC	2 bytes	

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<sub>h</sub>:00 with a length of 2 bytes, but the object has a size of just 1 byte:

#### Request

SA	FC	Data	CRC
05	2B	0D 00 00 01 60 60 00 00 00 00 02	79 8D

#### Response

SA	FC	Data	CRC
05	2B	FF 00 06 0D CE 12 00 07 06	AC 3C

### 9.6.9 FC 102 (66<sub>h</sub>) 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<sub>h</sub> must be sent. This code resets reading out on the object with subindex 00<sub>h</sub>. All subsequent requests must then contain subfunction code AA<sub>h</sub>. 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
Slave address	1 byte	
Function code	1 byte	66 <sub>h</sub>
Subfunction code	1 byte	55 <sub>h</sub> or AA <sub>h</sub>
Length of the data	1 byte	00 <sub>h</sub>
Index of the array to be read	2 bytes	
CRC	2 bytes	

#### Response:

Name	Length	Value / note
Slave address	1 byte	65 <sub>h</sub>
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	Length	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	

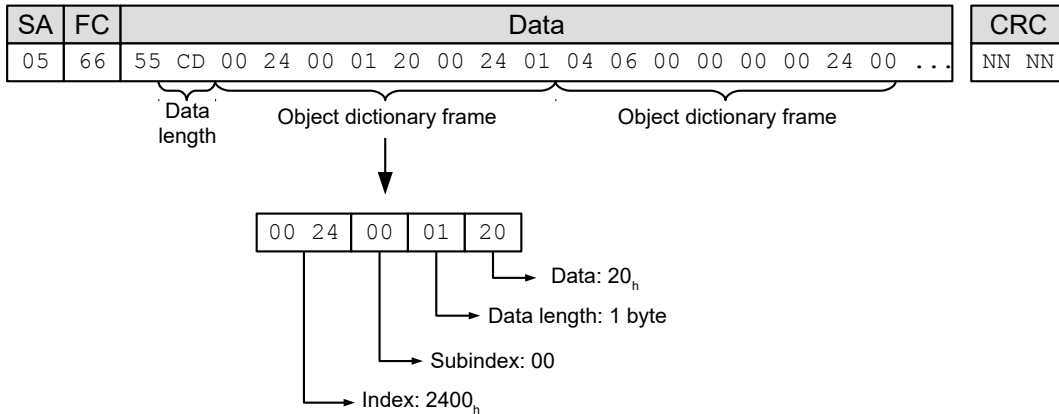
#### Example

All of the following numerical values are in hexadecimal format; the index of the object that is to be read is 2400<sub>h</sub>. The address of the slave is "5"<sub>h</sub>.

Start reading of the array with request:

SA	FC	Data	CRC
05	66	55 00 24 00	02 8A

The response is:



### 9.6.9.1 Error reaction

In the event of an error, the following error message is sent:

Name	Length	Example value
Slave address	1 byte	
Function code	1 byte	2B <sub>h</sub> +80 <sub>h</sub> (171 <sub>d</sub> = 43 <sub>d</sub> + 128 <sub>d</sub> ) (indicates error)
Modbus exception code	1 byte	FF <sub>h</sub> ("extended exception")
Extended exception length	2 bytes	6
MEI type	1 byte	0D <sub>h</sub>
Exception code	1 byte	CE <sub>h</sub>
Error code	4 bytes	CANopen error code, see following table
CRC	2 bytes	

CANopen error code	Description
FFFF0000 <sub>h</sub>	<i>Abort no error</i>
FFFF1003 <sub>h</sub>	Service is not supported
FFFF1004 <sub>h</sub>	Gap in counter byte of the <i>Protocol control</i> field
FFFF0003 <sub>h</sub>	Unknown or invalid command
FFFF0008 <sub>h</sub>	Access to the object is not supported
FFFF000E <sub>h</sub>	General error in the parameter
FFFF0011 <sub>h</sub>	Length of parameter incorrect
FFFF0012 <sub>h</sub>	Parameter too long
FFFF0013 <sub>h</sub>	Parameter too short
FFFF0015 <sub>h</sub>	Parameter data outside of the permissible value range (for write commands)
FFFF0016 <sub>h</sub>	Parameter data exceed the permissible value range (for write commands)
FFFF0017 <sub>h</sub>	Parameter data below the permissible value range (for write commands)
FFFF0018 <sub>h</sub>	Maximum entered values less than minimum values
FFFF0019 <sub>h</sub>	General error
FFFF001E <sub>h</sub>	Requested object is too large for single message

CANopen error code	Description
FFFF1004 <sub>h</sub>	Invalid sequence of messages (e. g., if the value of the <i>counter byte</i> 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
Slave address	1 byte	
Function code	1 byte	2B <sub>h</sub> + 80 <sub>h</sub> (171 <sub>d</sub> = 43 <sub>d</sub> + 128 <sub>d</sub> ) (indicates error)
Modbus exception code	1 byte	FF <sub>h</sub> ("extended exception")
Extended exception length	2 bytes	2 + length of "supported protocol control"
MEI type	1 byte	0D <sub>h</sub>
Exception code	1 byte	AE <sub>h</sub>
Supported protocol control	1 or 2 bytes	See following table
CRC	2 bytes	

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<sub>h</sub>:00 with a length of 2 bytes, but the object has a size of just 1 byte:

**Request**

SA	FC	Data	CRC
05	2B	0D 00 00 01 60 60 00 00 00 00 02	79 8D

## Response

SA	FC	Data	CRC
05	2B	FF 00 06 0D CE 12 00 07 06	AC 3C

### 9.6.10 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*.

## 9.7 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.

### 9.7.1 Configuration

The configuration of the image is referred to as "mapping" and is written in the following objects:

- 3502<sub>h</sub> for the Modbus Rx (master → slave) PDO mapping
- 3602<sub>h</sub> 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<sub>h</sub> and 3602<sub>h</sub> can be written with messages with Modbus function code 2B<sub>h</sub>.

### 9.7.2 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 <sub>h</sub>	Signed integer (8 bit)
0003 <sub>h</sub>	Signed integer (16 bit)
0004 <sub>h</sub>	Signed integer (32 bit)
0005 <sub>h</sub>	Unsigned integer (8 bit)
0006 <sub>h</sub>	Unsigned integer (16 bit)
0007 <sub>h</sub>	Unsigned integer (32 bit)

Mapping is as follows:



- The PDO RX image begins at Modbus register address 6000<sub>d</sub> (1770<sub>h</sub>).
- The PDO TX image begins at Modbus register address 5000<sub>d</sub> (1388<sub>h</sub>).

Read/write access can be performed simultaneously with function code 17<sub>h</sub> or with the 03<sub>h</sub>, 04<sub>h</sub>, 06<sub>h</sub>, 10<sub>h</sub> commands on the respective RX/TX images.

**Note**



To be able to change the mapping, you must first deactivate it by setting the corresponding subindex 0<sub>h</sub> to "0".

After writing the objects to the respective subindices, enter the number of mapped objects in subindex 0<sub>h</sub>.

**Example**

The following objects are to be set in the mapping:

- 3602<sub>h</sub>:00<sub>h</sub> = "0<sub>h</sub>" (mapping is deactivated)
- 3602<sub>h</sub>:01<sub>h</sub> = "60410010<sub>h</sub>" (object 6041<sub>h</sub>:00<sub>h</sub>, length 16 bits is mapped)
- 3602<sub>h</sub>:02<sub>h</sub> = "00050008<sub>h</sub>" (dummy object 0005<sub>h</sub>:00<sub>h</sub>, length 8 bits is mapped)
- 3602<sub>h</sub>:03<sub>h</sub> = "60610008<sub>h</sub>" (object 6061<sub>h</sub>:00<sub>h</sub>, length 8 bits is mapped)
- 3602<sub>h</sub>:04<sub>h</sub> = "60640020<sub>h</sub>" (object 6064<sub>h</sub>:00<sub>h</sub>, length 32 bits is mapped)
- 3602<sub>h</sub>:05<sub>h</sub> = "60440010<sub>h</sub>" (object 6044<sub>h</sub>:00<sub>h</sub>, length 16 bits is mapped)
- 3602<sub>h</sub>:06<sub>h</sub> = "60FD0020<sub>h</sub>" (object 60FD<sub>h</sub>:00<sub>h</sub>, length 32 bits is mapped)
- 3602<sub>h</sub>:00<sub>h</sub> = "6<sub>h</sub>" (6 values are mapped)

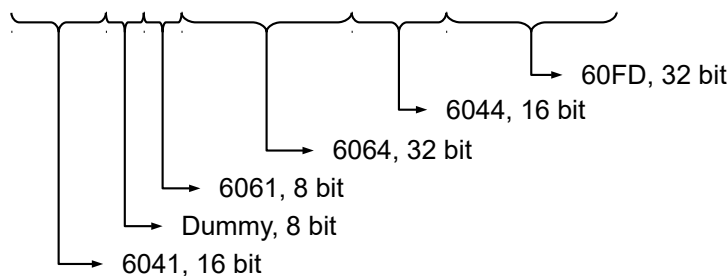
After the mapping for object 6061<sub>h</sub>:00<sub>h</sub>, a dummy object is inserted so that the next object 6064<sub>h</sub>:00<sub>h</sub> can be aligned to 32 bit.

**Rx message:** The master sends the slave the following message:

SA	FC	Data	CRC
05	04	13 88 00 07	34 E2

**Tx message:** The slave sends following response to the master:

SA	FC	Data	CRC
05	04	0E 06 40 00 02 00 00 00 00 00 00 00 00 00	1C 98



**9.8 NanoJ objects**

NanoJ objects 2400<sub>h</sub> NanoJ Input and 2500<sub>h</sub> (NanoJ Output) are, like the process image, mapped to the Modbus register:

- 2500<sub>h</sub> with 32 x 32 bit values is mapped to the Modbus register address beginning with 2000<sub>d</sub> (BB8<sub>h</sub>) and can only be read in this way.
- 2400<sub>h</sub> with 32 x 32 bit values is mapped to the Modbus register address beginning with 3000<sub>d</sub> (7D0<sub>h</sub>) and can only be written in this way.

To access, commands with function codes 03<sub>h</sub>, 04<sub>h</sub>, 10<sub>h</sub> and 17<sub>h</sub> 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:

SA	FC	Data	CRC
05	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	41 21

**Reply:** The slave sends the master the following response:

SA	FC	Data	CRC
05	17	10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	50 9D

## 10 Programming with *NanoJ*

*NanoJ* is a programming language similar to C or C++. *NanoJ* is integrated in the *Plug & Drive Studio* software. You can find further information in document *Plug & Drive Studio: Quick Start Guide* at [us.nanotec.com](http://us.nanotec.com).

### 10.1 *NanoJ* program

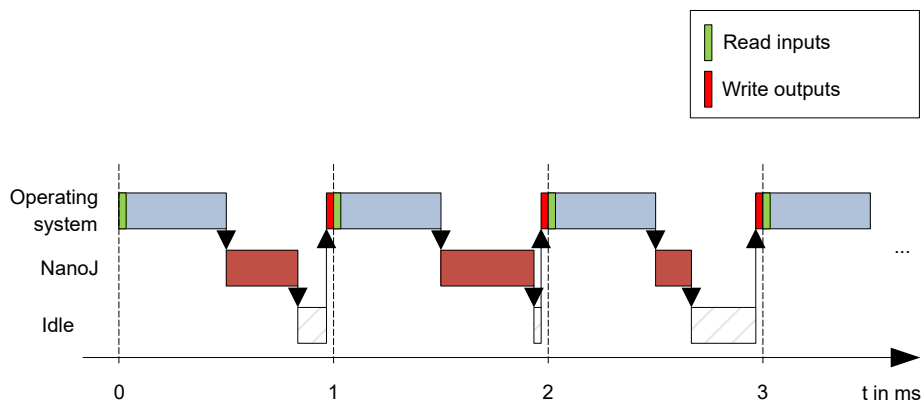
A *NanoJ* program makes a protected runtime environment available within the firmware. Here, the user can create his own processes. These can then trigger functions in the controller by, for example, reading or writing entries in the object dictionary.

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 `2300h` 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.

### Note



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<sub>h</sub>; in the error register for object 2302<sub>h</sub>, 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<sub>h</sub>](#). 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<sub>h</sub>, 2320<sub>h</sub>, and 2330<sub>h</sub>. 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<sub>h</sub>.
- [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<sub>h</sub> to 2330<sub>h</sub> (see [2300h NanoJ Control](#)).

OD-Index	Name and description
2300 <sub>h</sub>	<a href="#">2300h NanoJ Control</a>
2301 <sub>h</sub>	<a href="#">2301h NanoJ Status</a>
2302 <sub>h</sub>	<a href="#">2302h NanoJ Error Code</a>
2310 <sub>h</sub>	<a href="#">2310h NanoJ Input Data Selection</a>
2320 <sub>h</sub>	<a href="#">2320h NanoJ Output Data Selection</a>
2330 <sub>h</sub>	<a href="#">2330h NanoJ In/output Data Selection</a>

#### Example:

To start the *TEST1.USR* user program, the following sequence can, for example, be used:

- Check entry 2302<sub>h</sub> for error code.
- If no error:  
Start the *NanoJ program* by writing object 2300<sub>h</sub>, bit 0 = "1" or by restarting the controller.

#### Note



It can take up to 200 ms for the NanoJ program to start.

- Check entry 2302<sub>h</sub> for error code and object 2301<sub>h</sub>, bit 0 = "1".

To stop a running program: write entry 2300<sub>h</sub> 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.

### Note



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.

### Note



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 is not correct:

```
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 `2500h:01h`.

```
// 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](http://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<sub>h</sub> or *statusword* 6041<sub>h</sub>.
- 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.

### Note



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<sub>h</sub>:00<sub>h</sub>.
2. At the end of the 1 ms cycle, the mapping is written that also specifies object 6040<sub>h</sub>:00<sub>h</sub>, 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



### Note



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

### Note



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)
```

#### Note



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<sub>h</sub> 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.



#### Note

Debug outputs may therefore only be used during the test phase when developing a user program.



#### Note

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 <i>Plug &amp; Drive Studio</i> .	Instead use <code>od_read/od_write</code> to access the object.
If an object was mapped as output and the value of the object was never defined before starting the <i>NanoJ program</i> , 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 <code>constant array</code> instead.
<code>float</code> must not be used with comparison operators.	Use <code>int</code> instead.
<code>double</code> 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.	
<code>math</code> or <code>cmath</code> 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



### Note

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<sub>b</sub>: The description here applies if bit 2 and bit 3 are "0".
- Value 01<sub>b</sub>: The description here applies if bit 2 is "0" and bit 3 is "1".
- Value 10<sub>b</sub>: The description here applies if bit 2 is "1" and bit 3 is "0".
- Value 11<sub>b</sub>: 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**

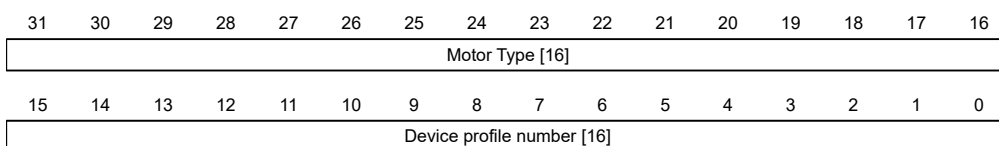
Description of bit A; bits with a gray background are not used.

**1000h Device Type****Function**

Describes the controller type.

**Object description**

Index	1000 <sub>h</sub>
Object name	Device Type
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	00060192 <sub>h</sub>
Firmware version	FIR-v1426
Change history	

**Description****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<sub>h</sub> or 0402<sub>d</sub> (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.



#### Note

For each error that occurs, a more precise error code is stored in object 1003<sub>h</sub>.

### Object description

Index	1001 <sub>h</sub>
Object name	Error Register
Object Code	VARIABLE
Data type	UNSIGNED8
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00 <sub>h</sub>
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 <sub>h</sub>
Object name	Pre-defined Error Field
Object Code	ARRAY
Data type	UNSIGNED32
Savable	no
Firmware version	FIR-v1426
Change history	

**Value description**

Subindex	00 <sub>h</sub>
Name	Number Of Errors
Data type	UNSIGNED8
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00 <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	Standard Error Field
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>

Subindex	02 <sub>h</sub>
Name	Standard Error Field
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	

Preset value	00000000 <sub>h</sub>
Subindex	03 <sub>h</sub>
Name	Standard Error Field
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>
Subindex	04 <sub>h</sub>
Name	Standard Error Field
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>
Subindex	05 <sub>h</sub>
Name	Standard Error Field
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>
Subindex	06 <sub>h</sub>
Name	Standard Error Field
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>
Subindex	07 <sub>h</sub>
Name	Standard Error Field
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>
Subindex	08 <sub>h</sub>

Name	Standard Error Field
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>

## 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<sub>h</sub>) 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 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 <sub>h</sub> ): Error through electrical fault or defective hardware
8	Sensor 2 (see 3204 <sub>h</sub> ): Error through electrical fault or defective hardware
9	Sensor 3 (see 3204 <sub>h</sub> ): 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 <sub>h</sub> (Following Error Window) and object 6066 <sub>h</sub> (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 <a href="#">3204<sub>h</sub></a> ), 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 (203B <sub>h</sub> :01 <sub>h</sub> /6075 <sub>h</sub> )
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
33	Undervoltage due to voltage connected with reverse polarity
39	Error in the ballast configuration: Invalid/unrealistic parameters entered (see <a href="#">Ballast monitoring</a> )
40	Warning: Ballast resistor thermally overloaded
46	Interlock error: Bit 3 in 60FD <sub>h</sub> is set to "0", the motor may not start (see the section <i>Interlock function</i> in the chapter <a href="#">Digital inputs</a> )

### Error Class[8]

This byte is identical to object [1001<sub>h</sub>](#)

### Error Code[16]

Refer to the following table for the meaning of the bytes.

Error Code	Description
1000 <sub>h</sub>	General error
2300 <sub>h</sub>	Current at the controller output too large
3100 <sub>h</sub>	Overvoltage/undervoltage at controller input
4200 <sub>h</sub>	Temperature error within the controller
5540 <sub>h</sub>	Interlock error: Bit 3 in 60FD <sub>h</sub> is set to "0", the motor may not start (see the section <i>Interlock function</i> in the chapter <a href="#">Digital inputs</a> )
6010 <sub>h</sub>	Software reset (watchdog)
6100 <sub>h</sub>	Internal software error, generic
6320 <sub>h</sub>	Rated current must be set (203B <sub>h</sub> :01 <sub>h</sub> /6075 <sub>h</sub> )
7110 <sub>h</sub>	Error in the ballast configuration: Invalid/unrealistic parameters entered (see <a href="#">Ballast monitoring</a> )
7113 <sub>h</sub>	Warning: Ballast resistor thermally overloaded
7121 <sub>h</sub>	Motor blocked
7200 <sub>h</sub>	Internal error: Correction factor for reference voltage missing in the OTP
7305 <sub>h</sub>	Sensor 1 (see <a href="#">3204<sub>h</sub></a> ) faulty
7306 <sub>h</sub>	Sensor 2 (see <a href="#">3204<sub>h</sub></a> ) faulty

Error Code	Description
7307 <sub>h</sub>	Sensor n (see 3204 <sub>h</sub> ), where n is greater than 2
7600 <sub>h</sub>	Warning: Nonvolatile memory full or corrupt; restart the controller for cleanup work
8003 <sub>h</sub>	Error in the ballast configuration: Invalid/unrealistic parameters entered (see <a href="#">Ballast monitoring</a> )
8100 <sub>h</sub>	Error during fieldbus monitoring
8130 <sub>h</sub>	CANopen only: "Life Guard" error or "Heartbeat" error
8200 <sub>h</sub>	CANopen only: Slave took too long to send PDO messages.
8210 <sub>h</sub>	CANopen only: PDO was not processed due to a length error
8220 <sub>h</sub>	CANopen only: PDO length exceeded
8240 <sub>h</sub>	CANopen only: unexpected sync length
8400 <sub>h</sub>	Error in speed monitoring: slippage error too large
8611 <sub>h</sub>	Position monitoring error: Following error too large
8612 <sub>h</sub>	Position monitoring error: Limit switch exceeded

## 1008h Manufacturer Device Name

### Function

Contains the device name as character string.

### Object description

Index	1008 <sub>h</sub>
Object name	Manufacturer Device Name
Object Code	VARIABLE
Data type	VISIBLE_STRING
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	NP5-02
Firmware version	FIR-v1426
Change history	

## 1009h Manufacturer Hardware Version

### Function

This object contains the hardware version as character string.

### Object description

Index	1009 <sub>h</sub>
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 <sub>h</sub>
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-v2039-B807052
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 <sub>h</sub>
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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	0D <sub>h</sub>
Subindex	01 <sub>h</sub>
Name	Save All Parameters To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001 <sub>h</sub>
Subindex	02 <sub>h</sub>
Name	Save Communication Parameters To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001 <sub>h</sub>
Subindex	03 <sub>h</sub>
Name	Save Application Parameters To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001 <sub>h</sub>

Subindex	04 <sub>h</sub>
Name	Save Customer Parameters To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001 <sub>h</sub>
Subindex	05 <sub>h</sub>
Name	Save Drive Parameters To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001 <sub>h</sub>
Subindex	06 <sub>h</sub>
Name	Save Tuning Parameters To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001 <sub>h</sub>
Subindex	07 <sub>h</sub>
Name	Save Miscellaneous Configurations To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001 <sub>h</sub>
Subindex	08 <sub>h</sub>
Name	Save Reserved1 Configurations To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>
Subindex	09 <sub>h</sub>
Name	Save Reserved2 Configurations To Non-volatile Memory
Data type	UNSIGNED32



Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>
<hr/>	
Subindex	0A <sub>h</sub>
Name	Save CANopen Configurations To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001 <sub>h</sub>
<hr/>	
Subindex	0B <sub>h</sub>
Name	Save Modbus RTU Configurations To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001 <sub>h</sub>
<hr/>	
Subindex	0C <sub>h</sub>
Name	Save Ethernet Configurations To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001 <sub>h</sub>
<hr/>	
Subindex	0D <sub>h</sub>
Name	Save Profibus Configurations To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001 <sub>h</sub>

## 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<sub>h</sub>" must be written in the corresponding subindex. This corresponds to the decimal of 1702257011<sub>d</sub> 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 <sub>h</sub>
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	<p>Firmware version FIR-v1436: "Object Name" entry changed from "Restore Default Parameter" to "Restore Default Parameters".</p> <p>Firmware version FIR-v1436: The number of entries was changed from 2 to 4.</p> <p>Firmware version FIR-v1512: The number of entries was changed from 4 to 5.</p> <p>Firmware version FIR-v1512: "Name" entry changed from "Restore The Comm Default Parameters" to "Restore Communication Default Parameters".</p> <p>Firmware version FIR-v1512: "Name" entry changed from "Restore The Application Default Parameters" to "Restore Application Default Parameters".</p> <p>Firmware version FIR-v1540: The number of entries was changed from 5 to 7.</p> <p>Firmware version FIR-v1738-B501312: The number of entries was changed from 7 to 14.</p>

### Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	0D <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	Restore All Default Parameters
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001 <sub>h</sub>
Subindex	02 <sub>h</sub>
Name	Restore Communication Default Parameters
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001 <sub>h</sub>
Subindex	03 <sub>h</sub>
Name	Restore Application Default Parameters
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001 <sub>h</sub>
Subindex	04 <sub>h</sub>
Name	Restore Customer Default Parameters
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001 <sub>h</sub>
Subindex	05 <sub>h</sub>
Name	Restore Drive Default Parameters
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001 <sub>h</sub>
Subindex	06 <sub>h</sub>
Name	Restore Tuning Default Parameters
Data type	UNSIGNED32

Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001 <sub>h</sub>
<hr/>	
Subindex	07 <sub>h</sub>
Name	Restore Miscellaneous Configurations
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001 <sub>h</sub>
<hr/>	
Subindex	08 <sub>h</sub>
Name	Restore Reserved1 Configurations To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>
<hr/>	
Subindex	09 <sub>h</sub>
Name	Restore Reserved2 Configurations To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>
<hr/>	
Subindex	0A <sub>h</sub>
Name	Restore CANopen Configurations To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001 <sub>h</sub>
<hr/>	
Subindex	0B <sub>h</sub>
Name	Restore Modbus RTU Configurations To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	

Preset value	00000001 <sub>h</sub>
Subindex	0C <sub>h</sub>
Name	Restore Ethernet Configurations To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001 <sub>h</sub>
Subindex	0D <sub>h</sub>
Name	Restore Profibus Configurations To Non-volatile Memory
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001 <sub>h</sub>

## Description

If the value 64616F6C<sub>h</sub> (or 1684107116<sub>d</sub> 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 <sub>h</sub>
Object name	Identity Object
Object Code	RECORD
Data type	IDENTITY
Savable	no
Firmware version	FIR-v1426
Change history	

## Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	04 <sub>h</sub>
Subindex	01 <sub>h</sub>
Name	Vendor-ID
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	0000026C <sub>h</sub>
Subindex	02 <sub>h</sub>
Name	Product Code
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000056 <sub>h</sub>
Subindex	03 <sub>h</sub>
Name	Revision Number
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	07F70000 <sub>h</sub>
Subindex	04 <sub>h</sub>
Name	Serial Number
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>

## 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 <sub>h</sub>
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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02 <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	Configuration Date
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>

Subindex	02 <sub>h</sub>
Name	Configuration Time
Data type	UNSIGNED32
Access	read / write
PDO mapping	no

Allowed values	
Preset value	00000000 <sub>h</sub>

---

## Description

Subindex 01<sub>h</sub> (configuration date) is to contain the number of days since 1 January 1984.

Subindex 02<sub>h</sub> (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 <sub>h</sub>
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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02 <sub>h</sub>

---

Subindex	01 <sub>h</sub>
Name	Program Data Bootloader/firmware
Data type	DOMAIN
Access	read / write
PDO mapping	no
Allowed values	
Preset value	0

---



Subindex	02 <sub>h</sub>
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 <sub>h</sub>
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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02 <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	Program Control Bootloader/firmware
Data type	UNSIGNED8
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00 <sub>h</sub>

Subindex	02 <sub>h</sub>
Name	Program Control NanoJ
Data type	UNSIGNED8
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00 <sub>h</sub>

## 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 <sub>h</sub>
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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02 <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	Program Status Bootloader/firmware
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>

Subindex	02 <sub>h</sub>
Name	Program Status NanoJ
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>

## 2028h MODBUS Slave Address

### Function

This object contains the slave address for Modbus. See chapter [Setting slave address and baud rate](#) .

### Object description

Index	2028 <sub>h</sub>
Object name	MODBUS Slave Address
Object Code	VARIABLE
Data type	UNSIGNED8
Savable	yes, category: Modbus RTU
Access	read / write
PDO mapping	no
Allowed values	1-247
Preset value	05 <sub>h</sub>
Firmware version	FIR-v1436
Change history	Firmware version FIR-v1748-B531667: "Savable" entry changed from "yes, category: communication" to "yes, category: Modbus RTU".

## 202Ah MODBUS RTU Baudrate

### Function

This object contains the baud rate of the Modbus in Bd. See chapter [Setting slave address and baud rate](#) .

### Object description

Index	202A <sub>h</sub>
Object name	MODBUS RTU Baudrate
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: Modbus RTU
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00004B00 <sub>h</sub>
Firmware version	FIR-v1436

Change history	Firmware version FIR-v1748-B531667: "Savable" entry changed from "yes, category: communication" to "yes, category: Modbus RTU".
----------------	---

---

## 202Ch MODBUS RTU Stop Bits

### Function

This object contains the number of stop bits of the Modbus.

### Object description

Index	202C <sub>h</sub>
Object name	MODBUS RTU Stop Bits
Object Code	VARIABLE
Data type	UNSIGNED8
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	01 <sub>h</sub>
Firmware version	FIR-v1436
Change history	Firmware version FIR-v1540: "Savable" entry changed from "yes, category: communication" to "no".  Firmware version FIR-v1540: "Access" table entry for subindex 00 changed from "read/write" to "read only".

---

### Description

The number of stop bits is dependent on the parity, which can be set in object [202D<sub>h</sub>](#).

## 202Dh MODBUS RTU Parity

### Function

For Modbus RTU, this object sets the number of parity bits and stop bits.

### Object description

Index	202D <sub>h</sub>
Object name	MODBUS RTU Parity
Object Code	VARIABLE
Data type	UNSIGNED8
Savable	yes, category: Modbus RTU
Access	read / write
PDO mapping	no
Allowed values	
Preset value	04 <sub>h</sub>
Firmware version	FIR-v1540

Change history

Firmware version FIR-v1748-B531667: "Savable" entry changed from "yes, category: communication" to "yes, category: Modbus RTU".

---

## Description

The following values apply:

- Value "0x00": Parity None, Stop Bits 2
- Value "0x04": Parity Even, Stop Bits 1
- Value "0x06": Parity Odd, Stop Bits 1

## 2030h Pole Pair Count

### Function

Contains the number of pole pairs of the connected motor.

### Object description

---

Index	2030 <sub>h</sub>
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 <sub>h</sub>
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 <sub>h</sub>
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 <sub>h</sub>
Firmware version	FIR-v1426
Change history	<p>Firmware version FIR-v1614: "Savable" entry changed from "yes, category: application" to "yes, category: tuning".</p> <p>Firmware version FIR-v1614: "Object Name" entry changed from "Peak Current" to "Max Current".</p> <p>Firmware version FIR-v1748-B538662: "Object Name" entry changed from "Maximum Current" to "Max Motor Current".</p> <p>Firmware version FIR-v1825-B577172: "Object Name" entry changed from "Max Motor Current" to "Maximum Current".</p> <p>Firmware version FIR-v1825-B577172: "Object Name" entry changed from "Maximum Current" to "Max Motor Current".</p> <p>Firmware version FIR-v1825-B577172: "Object Name" entry changed from "Max Motor Current" to "Maximum Current".</p> <p>Firmware version FIR-v1825-B577172: "Object Name" entry changed from "Maximum Current" to "Max Motor Current".</p>

## 2034h Upper Voltage Warning Level

### Function

This object contains the threshold value for the "overvoltage" error in millivolts.

### Object description

Index	2034 <sub>h</sub>
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	0000EF3D <sub>h</sub>
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<sub>h</sub> 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 <sub>h</sub>
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	00002710 <sub>h</sub>
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<sub>h</sub> 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 <sub>h</sub>
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 <sub>h</sub>
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<sub>h</sub> = "1") and the motor is at a standstill.

## Object description

Index	2037 <sub>h</sub>
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 <sub>h</sub>
Firmware version	FIR-v1426
Change history	

## Description

### Value of 2037<sub>h</sub> greater than or equal to 0 and less than value 6075<sub>h</sub>

Current is reduced to the value entered here. The value is in mA and interpreted as root mean square.

### Value of 2037<sub>h</sub> 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<sub>h</sub>. The value in 6075<sub>h</sub> is used for the calculation.

Example: Object 6075<sub>h</sub> has the value 4200 mA. The value -60 in 2037<sub>h</sub> reduces the current by 60% of 6075<sub>h</sub>. The result is a current reduction to a root mean square of 6075<sub>h</sub> \* (2037<sub>h</sub> + 100) / 100 = 1680 mA.

The value -100 in 2037<sub>h</sub> 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 <sub>h</sub>
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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	06 <sub>h</sub>
Subindex	01 <sub>h</sub>
Name	Close Brake Idle Time
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	000003E8 <sub>h</sub>
Subindex	02 <sub>h</sub>
Name	Shutdown Power Idle Time
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	000003E8 <sub>h</sub>
Subindex	03 <sub>h</sub>
Name	Open Brake Delay Time
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	000003E8 <sub>h</sub>
Subindex	04 <sub>h</sub>
Name	Start Operation Delay Time
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>
Subindex	05 <sub>h</sub>

Name	PWM Frequency
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	between 0 and 2000 (7D0 <sub>h</sub> )
Preset value	00000000 <sub>h</sub>

Subindex	06 <sub>h</sub>
Name	PWM Duty Cycle
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	0, between 2 and 100 (64 <sub>h</sub> )
Preset value	00000000 <sub>h</sub>

## Description

The subindices have the following functions:

- 01<sub>h</sub>: Time between motor standstill and the closing of the brake.
- 02<sub>h</sub>: Time between the closing of the brake and the switching off of the motor current.
- 03<sub>h</sub>: Time between the switching on of the motor current and opening of the brake.
- 04<sub>h</sub>: Time between the opening of the brake and when the *Operation enabled* state of the CiA 402 Power State Machine is reached.
- 05<sub>h</sub>: Frequency of the PWM signal in hertz.
- 06<sub>h</sub>: 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 <sub>h</sub>
Object name	Motor Currents
Object Code	ARRAY
Data type	INTEGER32
Savable	no
Firmware version	FIR-v1426
Change history	<p>Firmware version FIR-v1504: "PDO mapping" table entry for subindex 01 changed from "no" to "TX-PDO".</p> <p>Firmware version FIR-v1504: "PDO mapping" table entry for subindex 02 changed from "no" to "TX-PDO".</p> <p>Firmware version FIR-v1504: "PDO mapping" table entry for subindex 03 changed from "no" to "TX-PDO".</p> <p>Firmware version FIR-v1504: "PDO mapping" table entry for subindex 04 changed from "no" to "TX-PDO".</p>

## Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	04 <sub>h</sub>
Subindex	01 <sub>h</sub>
Name	I_d
Data type	INTEGER32
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>
Subindex	02 <sub>h</sub>
Name	I_q
Data type	INTEGER32
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>
Subindex	03 <sub>h</sub>
Name	I_a
Data type	INTEGER32
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>
Subindex	04 <sub>h</sub>
Name	I_b
Data type	INTEGER32
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>

## Description

- 01<sub>h</sub>: Field-forming components of the current
- 02<sub>h</sub>: Torque-forming components of the current
- 03<sub>h</sub>: Phase current in phase A (stepper motor) or U (BLDC motor)
- 04<sub>h</sub>: Phase current in phase B (stepper motor) or W (BLDC motor)

### Note



Motor currents I<sub>d</sub> (subindex 01<sub>h</sub>) and I<sub>q</sub> (subindex 02<sub>h</sub>) 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 <sub>h</sub>
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	<p>Firmware version FIR-v1540: The number of entries was changed from 4 to 3.</p> <p>Firmware version FIR-v1540: "Name" entry changed from "Period Of Blocking" to "Block Detection time".</p> <p>Firmware version FIR-v1614: "Data Type" entry changed from "UNSIGNED32" to "INTEGER32".</p> <p>Firmware version FIR-v1614: "Savable" entry changed from "no" to "yes, category: application".</p> <p>Firmware version FIR-v1614: "Data Type" entry changed from "UNSIGNED32" to "INTEGER32".</p> <p>Firmware version FIR-v1614: "Data Type" entry changed from "UNSIGNED32" to "INTEGER32".</p>

### Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8

Access	read only
PDO mapping	no
Allowed values	
Preset value	02 <sub>h</sub>
<hr/>	
Subindex	01 <sub>h</sub>
Name	Minimum Current For Block Detection
Data type	INTEGER32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	000005DC <sub>h</sub>
<hr/>	
Subindex	02 <sub>h</sub>
Name	Block Detection Time
Data type	INTEGER32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	000000C8 <sub>h</sub>

## Description

The subindices have the following function:

- 01<sub>h</sub>: 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<sub>h</sub>. Example: The value "1000" corresponds to 1000 mA (= 1 A); the value "-70" corresponds to 70% of 2031<sub>h</sub>.
- 02<sub>h</sub>: 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<sup>2</sup>t monitoring.

I<sup>2</sup>t monitoring is activated by entering a value greater than 0 in 203B<sub>h</sub>:01 and 203B<sub>h</sub>:02 and a value greater than 1000 in 6073<sub>h</sub> (see I2t Motor overload protection).

With one exception, I<sup>2</sup>t monitoring can only be used for *closed loop* mode: If I<sup>2</sup>t is activated in *open loop* mode, the current is reduced to the smaller of 203B<sub>h</sub>:01<sub>h</sub>, 6073<sub>h</sub> and 2031<sub>h</sub>.

### Object description

Index	203B <sub>h</sub>
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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	06 <sub>h</sub>
Subindex	01 <sub>h</sub>
Name	Motor Rated Current
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000258 <sub>h</sub>
Subindex	02 <sub>h</sub>
Name	Maximum Duration Of Max Current
Data type	UNSIGNED32
Access	read / write
PDO mapping	no

Allowed values	
Preset value	00000000 <sub>h</sub>
<hr/>	
Subindex	03 <sub>h</sub>
Name	Threshold
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>
<hr/>	
Subindex	04 <sub>h</sub>
Name	CalcValue
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>
<hr/>	
Subindex	05 <sub>h</sub>
Name	LimitedCurrent
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>
<hr/>	
Subindex	06 <sub>h</sub>
Name	Status
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>

## Description

The subindices are divided into two groups: subindex 01<sub>h</sub> and 02<sub>h</sub> contain parameters for the control, subindices 03<sub>h</sub> to 06<sub>h</sub> are status values. The functions are as follows:

- 01<sub>h</sub>: The rated current specified in the motor data sheet is entered here in mA. This must be smaller than the current entered in 2031<sub>h</sub> and 6073<sub>h</sub>, otherwise monitoring is not activated. The specified value is interpreted as root mean square.
- 02<sub>h</sub>: Specifies the maximum duration of the maximum current (6073<sub>h</sub>) in ms.
- 03<sub>h</sub>: Threshold, specifies the limit in mA that determines whether the maximum current or rated current is switched to.

- 04<sub>h</sub>: CalcValue, specifies the calculated value that is compared with the threshold for setting the current.
- 05<sub>h</sub>: LimitedCurrent, contains the momentary current as root mean square set by I<sup>2</sup><sub>t</sub>.
- 06<sub>h</sub>: Current status. If the sub-entry value is "0", I<sup>2</sup><sub>t</sub> is deactivated; if the value is "1", I<sup>2</sup><sub>t</sub> 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 "FFFFFFF"<sub>h</sub>, monitoring is switched off, the "Target reached" bit in object 6041<sub>h</sub> (statusword) is never set.

### Object description

Index	203D <sub>h</sub>
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 <sub>h</sub>
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<sub>h</sub>) for this time (in milliseconds) for the target torque to be considered having been met.

### Object description

Index	203E <sub>h</sub>
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 <sub>h</sub>
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 <sub>h</sub>
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 <sub>h</sub>
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<sub>h</sub> (Max Slippage) is exceeded, bit 13 in object 6041<sub>h</sub> is set. The deviation must last longer than the time in object 203F<sub>h</sub>.

A reaction to the slippage error can be set in object 3700<sub>h</sub>. If a reaction is defined, an error is also entered in object 1003<sub>h</sub>.

## 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 <sub>h</sub>
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 <sub>h</sub>
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 <sub>h</sub>
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 <sub>h</sub>
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 <sub>h</sub>
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 <sub>h</sub>
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 <sub>h</sub>
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 <sub>h</sub>
Firmware version	FIR-v1504
Change history	

## 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 <sub>h</sub>
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 <sub>h</sub>
Firmware version	FIR-v1426
Change history	

## 2101h Fieldbus Module Availability

### Function

Shows the available fieldbuses.

### Object description

Index	2101 <sub>h</sub>
Object name	Fieldbus Module Availability
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	00010002 <sub>h</sub>
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).

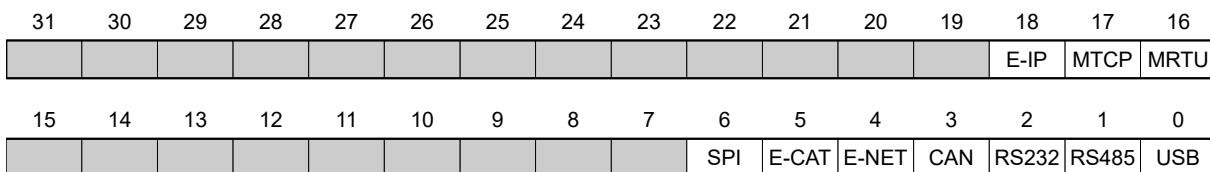
**Object description**

Index	2102 <sub>h</sub>
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	00010002 <sub>h</sub>
Firmware version	FIR-v1540
Change history	Firmware version FIR-v1626: "Savable" entry changed from "yes, category: application" to "yes, category: communication".

**Description**

Object 2103<sub>h</sub>:1<sub>h</sub> contains all physical interfaces/protocols that can be activated/deactivated. These can be switched in this object (2102<sub>h</sub>). The current status of the activated fieldbuses is in object 2103<sub>h</sub>:2<sub>h</sub>.

The following distribution of the bits applies here:



**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 <sub>h</sub>
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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02 <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	Fieldbus Module Disable Mask

Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>

---

Subindex	02 <sub>h</sub>
Name	Fieldbus Module Enabled
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00010002 <sub>h</sub>

### 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 <sub>h</sub>
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 <sub>h</sub>
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 <sub>h</sub>
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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	04 <sub>h</sub>

---

Subindex	01 <sub>h</sub>
Name	PDI Set Value 1
Data type	INTEGER32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>

---

Subindex	02 <sub>h</sub>
Name	PDI Set Value 2
Data type	INTEGER16
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000 <sub>h</sub>

---

Subindex	03 <sub>h</sub>
Name	PDI Set Value 3
Data type	INTEGER8
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00 <sub>h</sub>

---

Subindex	04 <sub>h</sub>
Name	PDI Command
Data type	INTEGER8
Access	read / write

PDO mapping	RX-PDO
Allowed values	
Preset value	00 <sub>h</sub>

## 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 <sub>h</sub>
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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	02 <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	PDI Status
Data type	INTEGER16
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	0000 <sub>h</sub>

Subindex	02 <sub>h</sub>
Name	PDI Return Value

Data type	INTEGER32
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>

## 2300h NanoJ Control

### Function

Controls the execution of a *NanoJ program*.

### Object description

Index	2300 <sub>h</sub>
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 <sub>h</sub>
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.



#### Note

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.



## 2302h NanoJ Error Code

### Function

Indicates which error occurred during the execution of the user program.

### Object description

Index	2302 <sub>h</sub>
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 <sub>h</sub>
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 <sub>h</sub>	Firmware does not support the used function (e.g., <code>sin</code> , <code>cosin</code> , etc.)
0005 <sub>h</sub>	Time Out: Code executed too long without <code>yield()</code> or <code>sleep()</code>
0007 <sub>h</sub>	Too many variables on the stack
0100 <sub>h</sub>	Invalid NanoJ program file
0101 <sub>h</sub>	Invalid NanoJ version of the program file
0102 <sub>h</sub>	CRC error in the NanoJ program file

Error when accessing an object:

Number	Description
1xxxxyy <sub>h</sub>	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 <sub>h</sub>	Invalid mapping in the NanoJ program file: too many variables of type <code>input</code> were declared (see <a href="#">2310h NanoJ Input Data Selection</a> )
3000000 <sub>h</sub>	Invalid mapping in the NanoJ program file: too many variables of type <code>output</code> were declared (see <a href="#">2320h NanoJ Output Data Selection</a> )
4000000 <sub>h</sub>	Invalid mapping in the NanoJ program file: too many variables of type <code>inout</code> were declared (see <a href="#">2330h NanoJ In/output Data Selection</a> )
1000 <sub>h</sub>	Access of a nonexistent object in the object dictionary
1001 <sub>h</sub>	Write access of a write-protected entry in the OD
1002 <sub>h</sub>	An attempt was made to write a value that is too low or too high to an object.

Number	Description
1003 <sub>h</sub>	An attempt was made to read out an object that permits only write access.
1FFF <sub>h</sub>	Unauthorized access of an object

## 230Fh Uptime Seconds

### Function

This object contains the operating time in seconds since the last time the controller was started.



#### Note

This object is not stored; counting begins with "0" again after switching on.

### Object description

Index	230F <sub>h</sub>
Object name	Uptime Seconds
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>
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 <sub>h</sub>
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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	10 <sub>h</sub>

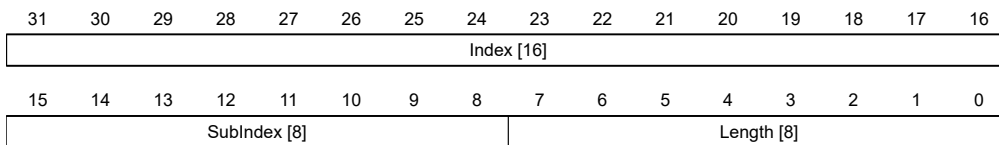
  

Subindex	01 <sub>h</sub> - 10 <sub>h</sub>
Name	Mapping #1 - #16
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>

### 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.



#### 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 <sub>h</sub>
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	<p>Firmware version FIR-v1436: "Object Name" entry changed from "VMM Output Data Selection" to "NanoJ Output Data Selection".</p> <p>Firmware version FIR-v1650-B472161: "Savable" entry changed from "yes, category: application" to "no".</p> <p>Firmware version FIR-v1650-B472161: "Access" table entry for subindex 00 changed from "read/write" to "read only".</p> <p>Firmware version FIR-v1650-B472161: "Access" table entry for subindex 01 changed from "read/write" to "read only".</p>

### Value description

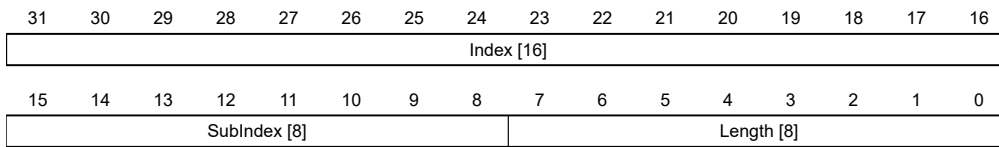
Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	10 <sub>h</sub>
Subindex	01 <sub>h</sub> - 10 <sub>h</sub>
Name	Mapping #1 - #16
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>



## 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.



### 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 <sub>h</sub>
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	<p>Firmware version FIR-v1436: "Object Name" entry changed from "VMM In/output Data Selection" to "NanoJ In/output Data Selection".</p> <p>Firmware version FIR-v1650-B472161: "Savable" entry changed from "yes, category: application" to "no".</p> <p>Firmware version FIR-v1650-B472161: "Access" table entry for subindex 00 changed from "read/write" to "read only".</p> <p>Firmware version FIR-v1650-B472161: "Access" table entry for subindex 01 changed from "read/write" to "read only".</p>

### Value description

Subindex	00 <sub>h</sub>
----------	-----------------

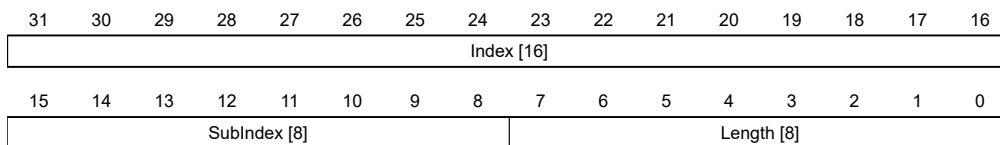
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	10 <sub>h</sub>

Subindex	01 <sub>h</sub> - 10 <sub>h</sub>
Name	Mapping #1 - #16
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>

## 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.



### 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 <sub>h</sub>
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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	20 <sub>h</sub>

Subindex	01 <sub>h</sub> - 20 <sub>h</sub>
Name	NanoJ Input #1 - #32
Data type	INTEGER32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>

## 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<sub>h</sub> with the difference that this object can be stored.

### Object description

Index	2410 <sub>h</sub>
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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	20 <sub>h</sub>

Subindex	01 <sub>h</sub> - 20 <sub>h</sub>
Name	NanoJ Init Parameter #1 - #32
Data type	INTEGER32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>

## 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 <sub>h</sub>
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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	

Preset value	20 <sub>h</sub>
Subindex	01 <sub>h</sub> - 20 <sub>h</sub>
Name	NanoJ Output #1 - #32
Data type	INTEGER32
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>

## 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 <sub>h</sub>
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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00 <sub>h</sub>
Subindex	01 <sub>h</sub> - 40 <sub>h</sub>
Name	Value #1 - #64
Data type	UNSIGNED8
Access	read only
PDO mapping	no

Allowed values	
Preset value	00 <sub>h</sub>

---

## 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 <sub>h</sub>
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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	FE <sub>h</sub>

---

Subindex	01 <sub>h</sub> - FE <sub>h</sub>
Name	Storage #1 - #254
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>

---

## 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 <sub>h</sub>
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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	03 <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	Reboot Command
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>

Subindex	02 <sub>h</sub>
Name	Reboot Delay Time In Ms
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>

Subindex	03 <sub>h</sub>
Name	Bootloader HW Config
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>

### Description

The subindices have the following function:

- 01<sub>h</sub>: If the value "746F6F62<sub>h</sub>" is entered here, the firmware is rebooted.
- 02<sub>h</sub>: Time in milliseconds: delays the reboot of the firmware by the respective time.
- 03<sub>h</sub>: 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

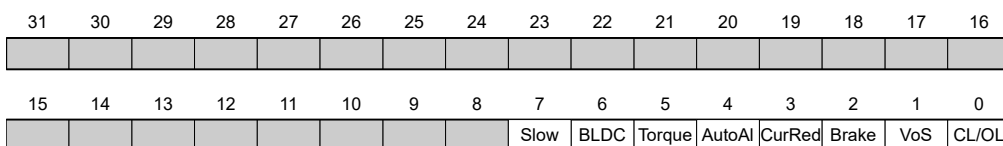
#### Function

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 <sub>h</sub>
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 <sub>h</sub>
Firmware version	FIR-v1426
Change history	<p>Firmware version FIR-v1540: "Savable" entry changed from "yes category: application" to "yes, category: travel".</p> <p>Firmware version FIR-v1540: "Savable" entry changed from "yes category: travel" to "yes, category: movement".</p>

### Description





## 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*

## AutoAI ( *auto alignment* )

For the case that operation in *closed loop* is required (bit 0 in 3202<sub>h</sub> 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<sub>h</sub> 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<sub>h</sub> is ignored; 3210<sub>h</sub>:3 and 3210<sub>h</sub>:4 have no effect on the control.

## BLDC

Value = "1": Motor type "BLDC" (brushless DC motor)

## Slow ( *slow speed* )

Value = "1": The [slow speed](#) mode is activated ( *closed loop* must already be activated)

## 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 <sub>h</sub>
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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	04 <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	1st Feedback Interface
Data type	UNSIGNED8
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00 <sub>h</sub>

Subindex	02 <sub>h</sub>
Name	2nd Feedback Interface
Data type	UNSIGNED8
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00 <sub>h</sub>

Subindex	03 <sub>h</sub>
Name	3rd Feedback Interface
Data type	UNSIGNED8
Access	read / write

PDO mapping	RX-PDO
Allowed values	
Preset value	00 <sub>h</sub>
<hr/>	
Subindex	04 <sub>h</sub>
Name	4th Feedback Interface
Data type	UNSIGNED8
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00 <sub>h</sub>

## Description

The subindices have the following function:

- 00<sub>h</sub>: Value="1" to "n", where "n" is the number of existing feedbacks.
- n<sub>h</sub>:  
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<sub>h</sub> 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.

### Note



If bit 0 in 3202<sub>h</sub> 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 <sub>h</sub>
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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	04 <sub>h</sub>

---

Subindex	01 <sub>h</sub>
Name	Index Of 1st Feedback Interface
Data type	UNSIGNED16
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	3380 <sub>h</sub>

---

Subindex	02 <sub>h</sub>
Name	Index Of 2nd Feedback Interface
Data type	UNSIGNED16
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	3390 <sub>h</sub>

---

Subindex	03 <sub>h</sub>
Name	Index Of 3rd Feedback Interface
Data type	UNSIGNED16
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	33A0 <sub>h</sub>

---

Subindex	04 <sub>h</sub>
Name	Index Of 4th Feedback Interface

Data type	UNSIGNED16
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	33A1 <sub>h</sub>

## Description

The subindices have the following function:

- 00<sub>h</sub>: Value="1" to "n", where "n" is the number of existing feedbacks.
- n<sub>h</sub>:  
 Subindex n refers to the index of the respective object for the configuration of the corresponding feedback.  
 Subindex 01<sub>h</sub> 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](#).

## 320Dh Torque Of Inertia Factor

### Function

This factor is used for calculating the acceleration feed forward (see [320E<sub>h</sub>:08<sub>h</sub>](#)). Default is 0 (feed forward inactive).

Acceleration feed forward applies during deceleration as well.

### Object description

Index	320D <sub>h</sub>
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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02 <sub>h</sub>

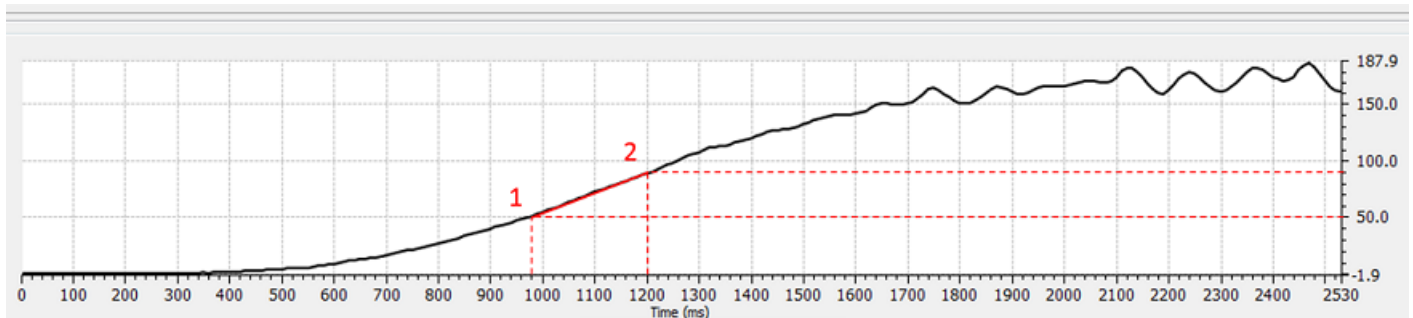
Subindex	01 <sub>h</sub>
Name	Current
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>

Subindex	02 <sub>h</sub>
Name	Acceleration
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>

## 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<sub>h</sub>:01<sub>h</sub>.
3. Record (e. g., in *Plug & Drive Studio*) the current speed (object 606C<sub>h</sub>). Calculate the acceleration in the set *user-defined units* for the speed range, where this is constant. Enter the value in 320D<sub>h</sub>:02<sub>h</sub>.  
Using the speed curve in the following figure as an example:  
 $(90-50)/(1200-980)=182$  rpm/s.



## 320Eh Closed Loop Controller Parameter

### Function

Contains the control parameters for *closed loop*.

#### Note

For firmware versions from FIR-v19xx upwards, the new schema for the *Controller structure* applies.



The old control parameters (object 3210<sub>h</sub>) are activated in the factory settings for compatibility reasons. For new applications, Nanotec recommends using the new control parameters.

To use the new parameters, you must set 3210<sub>h</sub>:07<sub>h</sub> (for *closed loop*) or 3210<sub>h</sub>:09<sub>h</sub> (for *open loop*) to "0". The old values are converted and entered in the new object 320E<sub>h</sub> or 320F<sub>h</sub>. You must save both objects (see *Saving objects*).

## Object description

Index	320E <sub>h</sub>
Object name	Closed Loop Controller Parameter
Object Code	RECORD
Data type	CLOSED_LOOP_CONTROLLER_PARAMETER
Savable	yes, category: drive
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1825-B577172
Change history	<p>Firmware version FIR-v1913-B623284: "Name" entry changed from "PWM Feed Forward" to "Reserved."</p> <p>Firmware version FIR-v2013-B726332: "Name" entry changed from "Max Current Deviation" to "Max Current Deviation [%]".</p> <p>Firmware version FIR-v2013-B726332: "Data type" entry changed from "UNSIGNED16" to "UNSIGNED32".</p> <p>Firmware Version FIR-v2013-B726332: "Name" entry changed from "Max Voltage Via PWM" to "Max Voltage [mV]".</p> <p>Firmware version FIR-v2013-B726332: "Data type" entry changed from "UNSIGNED16" to "UNSIGNED32".</p> <p>Firmware version FIR-v2013-B726332: "Data type" entry changed from "UNSIGNED32" to "UNSIGNED16".</p> <p>Firmware version FIR-v2039-B807052: "Name" entry changed from "Reserved" to "Voltage Feed Forward [%]".</p>

## Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	0F <sub>h</sub>
Subindex	01 <sub>h</sub>
Name	Position Controller Kp [%]
Data type	UNSIGNED16
Access	read / write
PDO mapping	no
Allowed values	
Preset value	0000 <sub>h</sub>

Subindex	02 <sub>h</sub>
Name	Position Controller Tn [μs]
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>
Subindex	03 <sub>h</sub>
Name	Velocity Feed Forward [%]
Data type	UNSIGNED16
Access	read / write
PDO mapping	no
Allowed values	
Preset value	03E8 <sub>h</sub>
Subindex	04 <sub>h</sub>
Name	Max Position Deviation
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>
Subindex	05 <sub>h</sub>
Name	Max Motor Speed
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00007530 <sub>h</sub>
Subindex	06 <sub>h</sub>
Name	Velocity Controller Kp [%]
Data type	UNSIGNED16
Access	read / write
PDO mapping	no
Allowed values	
Preset value	0000 <sub>h</sub>
Subindex	07 <sub>h</sub>
Name	Velocity Controller Tn [μs]
Data type	UNSIGNED32



Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>
<hr/>	
Subindex	08 <sub>h</sub>
Name	Acceleration Feed Forward [%o]
Data type	UNSIGNED16
Access	read / write
PDO mapping	no
Allowed values	
Preset value	03E8 <sub>h</sub>
<hr/>	
Subindex	09 <sub>h</sub>
Name	Max Velocity Deviation
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>
<hr/>	
Subindex	0A <sub>h</sub>
Name	Max Current [%o]
Data type	UNSIGNED16
Access	read / write
PDO mapping	no
Allowed values	
Preset value	03E8 <sub>h</sub>
<hr/>	
Subindex	0B <sub>h</sub>
Name	Current Controller Kp [%o]
Data type	UNSIGNED16
Access	read / write
PDO mapping	no
Allowed values	
Preset value	0000 <sub>h</sub>
<hr/>	
Subindex	0C <sub>h</sub>
Name	Current Controller Tn [μs]
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	

Preset value	00000000 <sub>h</sub>
Subindex	0D <sub>h</sub>
Name	Voltage Feed Forward [%]
Data type	UNSIGNED16
Access	read / write
PDO mapping	no
Allowed values	
Preset value	03E8 <sub>h</sub>
Subindex	0E <sub>h</sub>
Name	Max Current Deviation [%]
Data type	UNSIGNED16
Access	read / write
PDO mapping	no
Allowed values	
Preset value	0000 <sub>h</sub>
Subindex	0F <sub>h</sub>
Name	Max Voltage [mV]
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	000186A0 <sub>h</sub>

## Description

- Subindex 00<sub>h</sub>: Number of entries
- Subindex 01<sub>h</sub>: Gain factor (proportional component) of the position controller in tenths of a percent
- Subindex 02<sub>h</sub>: Reset time (integral component) of the position controller in microseconds
- Subindex 03<sub>h</sub>: Speed feed forward in tenths of a percent. Default is 1000 and, thus, a factor of 1.
- Subindex 04<sub>h</sub>: Maximum control deviation of the position controller in user-defined units
- Subindex 05<sub>h</sub>: Maximum permissible speed of the motor in user-defined units. See 6080<sub>h</sub>.
- Subindex 06<sub>h</sub>: Gain factor (proportional component) of the velocity controller in tenths of a percent
- Subindex 07<sub>h</sub>: Reset time (integral component) of the velocity controller in microseconds
- Subindex 08<sub>h</sub>: Acceleration feed forward in tenths of a percent of the value of 320D<sub>h</sub>
- Subindex 09<sub>h</sub>: Maximum control deviation of the velocity controller in user-defined units
- Subindex 0A<sub>h</sub>: Maximum current in tenths of a percent of the set rated current, see object 6073<sub>h</sub>
- Subindex 0B<sub>h</sub>: Gain factor (proportional component) of the current controller in tenths of a percent
- Subindex 0C<sub>h</sub>: Reset time (integral component) of the current controller in microseconds
- Subindex 0D<sub>h</sub>: Voltage feed forward in tenths of a percent of the voltage that is needed to produce the rated current
- Subindex 0E<sub>h</sub>: Maximum control deviation of the current controller in mA
- Subindex 0F<sub>h</sub>: Maximum permissible PWM voltage (duty cycle). Values ≤ 1000 are interpreted as per mil values (of the available voltage). Values > 1000 as millivolt.

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.
$\geq U_{o\_high}$	Full; the voltage vector describes a square or a hexagon.

### $U_{o\_low}$

The lowest voltage above which overmodulation occurs. Is calculated as follows:

With two-phase stepper motors: operating voltage\*1.063

With three-phase BLDC motors: operating voltage\*0.99

### $U_{o\_high}$

The maximum overmodulation occurs above this voltage. Is calculated as follows:

Operating voltage\*0.9425

## 320Fh Open Loop Controller Parameter

### Function

Contains the control parameters for *open loop*.

#### Note

For firmware versions from FIR-v19xx upwards, the new schema for the Controller structure applies.



The old control parameters (object 3210<sub>h</sub>) are activated in the factory settings for compatibility reasons. For new applications, Nanotec recommends using the new control parameters.

To use the new parameters, you must set 3210<sub>h</sub>:07<sub>h</sub> (for *closed loop*) or 3210<sub>h</sub>:09<sub>h</sub> (for *open loop*) to "0". The old values are converted and entered in the new object 320E<sub>h</sub> or 320F<sub>h</sub>. You must save both objects (see Saving objects).

### Object description

Index	320F <sub>h</sub>
Object name	Open Loop Controller Parameter
Object Code	RECORD
Data type	OPEN_LOOP_CONTROLLER_PARAMETER
Savable	yes, category: drive
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1825-B577172

Change history	<p>Firmware version FIR-v1913-B623284: "Name" entry changed from "PWM Feed Forward" to "Reserved."</p> <p>Firmware Version FIR-v2013-B726332: "Name" entry changed from "Max Voltage Via PWM" to "Max Voltage [mV]".</p> <p>Firmware version FIR-v2013-B726332: "Data type" entry changed from "UNSIGNED16" to "UNSIGNED32".</p>
----------------	--

---

## Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	05 <sub>h</sub>
Subindex	01 <sub>h</sub>
Name	Current Controller Kp [% <sub>o</sub> ]
Data type	UNSIGNED16
Access	read / write
PDO mapping	no
Allowed values	
Preset value	0000 <sub>h</sub>
Subindex	02 <sub>h</sub>
Name	Current Controller Tn [μs]
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>
Subindex	03 <sub>h</sub>
Name	Reserved
Data type	UNSIGNED16
Access	read / write
PDO mapping	no
Allowed values	
Preset value	0000 <sub>h</sub>
Subindex	04 <sub>h</sub>
Name	Max Current Deviation [% <sub>o</sub> ]

Data type	UNSIGNED16
Access	read / write
PDO mapping	no
Allowed values	
Preset value	0000 <sub>h</sub>
<hr/>	
Subindex	05 <sub>h</sub>
Name	Max Voltage [mV]
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	000186A0 <sub>h</sub>

## Description

- Subindex 00<sub>h</sub>: Number of entries
- Subindex 01<sub>h</sub>: Gain factor (proportional component) of the current controller in tenths of a percent
- Subindex 02<sub>h</sub>: Reset time (integral component) of the current controller in microseconds
- Subindex 03<sub>h</sub>: Reserved
- Subindex 04<sub>h</sub>: Maximum control deviation of the current controller in mA
- Subindex 05<sub>h</sub>: Maximum permissible PWM voltage (duty cycle). Values ≤ 1000 are interpreted as per mil values (of the available voltage). Values > 1000 as millivolt.

## 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*.

#### Note

For firmware versions from FIR-v19xx upwards, the new schema for the [Controller structure](#) applies.



The old control parameters (object 3210<sub>h</sub>) are activated in the factory settings for compatibility reasons. For new applications, Nanotec recommends using the new control parameters.

To use the new parameters, you must set 3210<sub>h</sub>:07<sub>h</sub> (for *closed loop*) or 3210<sub>h</sub>:09<sub>h</sub> (for *open loop*) to "0". The old values are converted and entered in the new object 320E<sub>h</sub> or 320F<sub>h</sub>. You must save both objects (see [Saving objects](#)).

## Object description

Index	3210 <sub>h</sub>
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	<p>Firmware version FIR-v1626: "Name" entry changed from "S_P" to "Position Loop, Proportional Gain (closed loop)".</p> <p>Firmware version FIR-v1626: "Name" entry changed from "S_I" to "Position Loop, Integral Gain (closed loop)".</p> <p>Firmware version FIR-v1626: "Name" entry changed from "V_P" to "Velocity Loop, Proportional Gain (closed loop)".</p> <p>Firmware version FIR-v1626: "Name" entry changed from "V_I" to "Velocity Loop, Integral Gain (closed loop)".</p> <p>Firmware version FIR-v1626: "Name" entry changed from "Id_P" to "Flux Current Loop, Proportional Gain (closed loop)".</p> <p>Firmware version FIR-v1626: "Name" entry changed from "Id_I" to "Flux Current Loop, Integral Gain (closed loop)".</p> <p>Firmware version FIR-v1626: "Name" entry changed from "Iq_P" to "Torque Current Loop, Proportional Gain (closed loop)".</p> <p>Firmware version FIR-v1626: "Name" entry changed from "Iq_I" to "Torque Current Loop, Integral Gain (closed loop)".</p> <p>Firmware version FIR-v1626: "Name" entry changed from "I_P" to "Torque Current Loop, Proportional Gain (dspDrive – Stepper Motor, open loop)".</p> <p>Firmware version FIR-v1626: "Name" entry changed from "I_I" to "Torque Current Loop, Integral Gain (dspDrive – Stepper Motor, open loop)".</p> <p>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)".</p> <p>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)".</p> <p>Firmware version FIR-v1650-B472161: "Data type" entry changed from "INTEGER32" to "UNSIGNED32".</p> <p>Firmware version FIR-v1650-B472161: "Data type" entry changed from "INTEGER32" to "UNSIGNED32".</p> <p>Firmware version FIR-v1738-B501312: The number of entries was changed from 11 to 13.</p> <p>Firmware version FIR-v1738-B501312: "PDO mapping" table entry for subindex 00 to 0A changed from "no" to "RX-PDO".</p>

## Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	RX-PDO

Allowed values	
Preset value	0C <sub>h</sub>
<hr/>	
Subindex	01 <sub>h</sub>
Name	Position Loop, Proportional Gain (closed Loop)
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000800 <sub>h</sub>
<hr/>	
Subindex	02 <sub>h</sub>
Name	Position Loop, Integral Gain (closed Loop)
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>
<hr/>	
Subindex	03 <sub>h</sub>
Name	Velocity Loop, Proportional Gain (closed Loop)
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00002EE0 <sub>h</sub>
<hr/>	
Subindex	04 <sub>h</sub>
Name	Velocity Loop, Integral Gain (closed Loop)
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000001E <sub>h</sub>
<hr/>	
Subindex	05 <sub>h</sub>
Name	Flux Current Loop, Proportional Gain (closed Loop)
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00881EE0 <sub>h</sub>

Subindex	06 <sub>h</sub>
Name	Flux Current Loop, Integral Gain (closed Loop)
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0007C740 <sub>h</sub>
Subindex	07 <sub>h</sub>
Name	Torque Current Loop, Proportional Gain (closed Loop)
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00881EE0 <sub>h</sub>
Subindex	08 <sub>h</sub>
Name	Torque Current Loop, Integral Gain (closed Loop)
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0007C740 <sub>h</sub>
Subindex	09 <sub>h</sub>
Name	Torque Current Loop, Proportional Gain (open Loop)
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	004DC880 <sub>h</sub>
Subindex	0A <sub>h</sub>
Name	Torque Current Loop, Integral Gain (open Loop)
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	001D2B30 <sub>h</sub>
Subindex	0B <sub>h</sub>
Name	Velocity Feed Forward Factor In Per Mille
Data type	UNSIGNED32



Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	000003E8 <sub>h</sub>

---

Subindex	0C <sub>h</sub>
Name	Acceleration Feed Forward Factor
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>

## Description

- Subindex 00<sub>h</sub>: Number of entries
- Subindex 01<sub>h</sub>: Proportional component of the S-controller (position)
- Subindex 02<sub>h</sub>: Integral component of the S-controller (position)
- Subindex 03<sub>h</sub>: Proportional component of the V-controller (speed)
- Subindex 04<sub>h</sub>: Integral component of the V-controller (speed)
- Subindex 05<sub>h</sub>: (Closed loop) Proportional component of the current controller of the field-forming component
- Subindex 06<sub>h</sub>: (Closed loop) Integral component of the current controller of the field-forming component
- Subindex 07<sub>h</sub>: (Closed loop) Proportional component of the current controller of the torque-forming component
- Subindex 08<sub>h</sub>: (Closed loop) Integral component of the current controller of the torque-forming component
- Subindex 09<sub>h</sub>: (Open loop) Proportional component of the current controller of the field-building component
- Subindex 0A<sub>h</sub>: (Open loop) Integral component of the current controller of the field-forming component
- Subindex 0B<sub>h</sub>: (Closed loop) Speed feed forward in tenths of a percent. Default is 1000 and, thus, a factor of 1.
- Subindex 0C<sub>h</sub>: (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. The direction of the rotating field can also be changed.

#### Note



Changes in subindex 02<sub>h</sub> do not take effect until after the controller is restarted. Afterwards, Auto setup must again be performed.

## Object description

Index	3212 <sub>h</sub>
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	<p>Firmware version FIR-v1512: The number of entries was changed from 2 to 3.</p> <p>Firmware version FIR-v1738-B501312: "Name" entry changed from "Enable Legacy Power Mode" to "Reserved".</p>

## Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	03 <sub>h</sub>
Subindex	01 <sub>h</sub>
Name	Reserved
Data type	INTEGER8
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00 <sub>h</sub>
Subindex	02 <sub>h</sub>
Name	Override Field Inversion
Data type	INTEGER8
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00 <sub>h</sub>
Subindex	03 <sub>h</sub>
Name	Do Not Touch Controller Settings
Data type	INTEGER8
Access	read / write
PDO mapping	no

Allowed values	
Preset value	00 <sub>h</sub>

---

## Description

Valid values for subindex 02<sub>h</sub>:

- 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<sub>h</sub>:

- 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 values for the controller that were entered in object 3210<sub>h</sub> or 320E<sub>h</sub> before the auto setup; the values in 3210<sub>h</sub> or 320E<sub>h</sub> are not changed.

## 3220h Analog Inputs

### Function

Displays the instantaneous values of the analog inputs in *ADC digits*.

### Object description

---

Index	3220 <sub>h</sub>
Object name	Analog Inputs
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 <sub>h</sub>
Name	Number Of Analogue Inputs
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02 <sub>h</sub>

---



---

Subindex	01 <sub>h</sub>
Name	Analogue Input 1

---

Data type	INTEGER16
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	0000 <sub>h</sub>
<hr/>	
Subindex	02 <sub>h</sub>
Name	Analogue Input 2
Data type	INTEGER16
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	0000 <sub>h</sub>

## Description

Formulas for converting from [digits] to the respective unit:

- Voltage input:  $x \text{ digits} * 3.3 \text{ V} / 1023 \text{ digits}$
- Current input (if configurable):  $x \text{ digits} * 20 \text{ mA} / 1023 \text{ digits}$

## 3225h Analogue Inputs Switches

### Function

This object contains the value of the virtual rotary switch that is used to configure the address of the controller . The switch position is only read during a restart one time.

### Object description

Index	3225 <sub>h</sub>
Object name	Analogue Inputs Switches
Object Code	ARRAY
Data type	UNSIGNED16
Savable	no
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1436
Change history	Firmware version FIR-v1436: "PDO mapping" table entry for subindex 01 changed from "RX-PDO" to "TX-PDO".

### Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8

Access	read only
PDO mapping	no
Allowed values	
Preset value	01 <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	Analogue Input Switch1
Data type	UNSIGNED16
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	0000 <sub>h</sub>

## Description

If the controller is equipped with a rotary switch, the value of the rotary switch is displayed in subindex 01<sub>h</sub>. If the controller is equipped with two rotary switches, the value of the rotary switch is displayed in subindex 01<sub>h</sub>, which consists of switch 1 and 2.

## 3231h Flex IO Configuration

### Function

Defines how the pins (inputs/outputs 1 ... 4) of the device are used.

Each bit in the respective subindex corresponds to a pin, whereby the first pin configures DIO1, the second pin DIO2, etc.

- Pin 1: DIO1\_IO\_CS
- Pin 2: DIO2\_CD\_CLK
- Pin 3: DIO3\_CD\_DIR
- Pin 4: DIO4\_IO\_MOSI

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
													...	Pin 2	Pin 1

- Subindex 03<sub>h</sub> *Alternate Function Mask*: This bit mask defines whether the alternative function of the pin is to be activated.

To activate an alternative function, set the corresponding bit to "1":

Bit	Pin	Alternative function
1	DIO2_CD_CLK	Clock input in clock-direction mode
2	DIO3_CD_DIR	Direction input in clock-direction mode

### Note



If you do not activate the alternative function, you can no longer use the corresponding pin as a normal input/output.

If the Clock-direction mode is activated, the pins for clock and direction are automatically configured accordingly and the settings in 3231<sub>h</sub> are overwritten.

- Subindex 01<sub>h</sub> *Output Mask*: This bit mask defines whether the pin is used as input or output (depending on whether an alternative function was activated for the pin in subindex 03<sub>h</sub>):

- Bit = "0": Pin is input (default)
- Bit = "1": Pin is output
- Subindex 02<sub>h</sub> *Pullup Mask*: This bit mask defines whether the pin is a *pullup* or *pulldown*:
  - Bit = "0": Pin is *pulldown* (default)
  - Bit = "1": Pin is *pullup*

Subindex 02<sub>h</sub> is only active for the pin if it is defined as an input.

Example for subindex 01<sub>h</sub>: Pin 2 and pin 3 are to be outputs, value = "6" (=0110<sub>b</sub>)

## Object description

Index	3231 <sub>h</sub>
Object name	Flex IO Configuration
Object Code	ARRAY
Data type	UNSIGNED16
Savable	yes, category: application
Access	read only
PDO mapping	no
Allowed values	
Preset value	
Firmware version	FIR-v1650-B472161
Change history	

## Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	03 <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	Output Mask
Data type	UNSIGNED16
Access	read / write
PDO mapping	no
Allowed values	
Preset value	0000 <sub>h</sub>

Subindex	02 <sub>h</sub>
Name	Pullup Mask
Data type	UNSIGNED16
Access	read / write
PDO mapping	no

Allowed values	
Preset value	0000 <sub>h</sub>
<hr/>	
Subindex	03 <sub>h</sub>
Name	Alternate Function Mask
Data type	UNSIGNED16
Access	read / write
PDO mapping	no
Allowed values	
Preset value	0000 <sub>h</sub>

## 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 <sub>h</sub>
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 <sub>h</sub> : "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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	08 <sub>h</sub>
<hr/>	
Subindex	01 <sub>h</sub>
Name	Special Function Enable
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO

Allowed values	
Preset value	00000000 <sub>h</sub>
<hr/>	
Subindex	02 <sub>h</sub>
Name	Function Inverted
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>
<hr/>	
Subindex	03 <sub>h</sub>
Name	Force Enable
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>
<hr/>	
Subindex	04 <sub>h</sub>
Name	Force Value
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>
<hr/>	
Subindex	05 <sub>h</sub>
Name	Raw Value
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>
<hr/>	
Subindex	06 <sub>h</sub>
Name	Input Range Select
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>
<hr/>	



Subindex	07 <sub>h</sub>
Name	Differential Select
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>

Subindex	08 <sub>h</sub>
Name	Routing Enable
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>

## Description

The subindices have the following function:

- 3240<sub>h</sub>:01<sub>h</sub> (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<sub>h</sub>:01<sub>h</sub> must be set to "1".

- 3240<sub>h</sub>:02<sub>h</sub> (Function Inverted): This subindex switches from normally open logic (a logical high level at the input yields the value "1" in object 60FD<sub>h</sub>) 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<sub>h</sub>:03<sub>h</sub> (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<sub>h</sub>:04<sub>h</sub>, but rather the set values for the respective input. Bit 0 corresponds to input 1 here, bit 1 to input 2, etc.
- 3240<sub>h</sub>:04<sub>h</sub> (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<sub>h</sub>:03<sub>h</sub>.
- 3240<sub>h</sub>:05<sub>h</sub> (Raw Value): This object contains the unmodified input value.
- 3240<sub>h</sub>:07<sub>h</sub> (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<sub>h</sub>:08<sub>h</sub> (Routing Enable): The value "1" in this subindex activates Input Routing.

## 3242h Digital Input Routing

### Function

This object determines the source of the input routing that ends in 60FD<sub>h</sub>.

### Object description

Index	3242 <sub>h</sub>
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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	24 <sub>h</sub>

Subindex	01 <sub>h</sub> - 24 <sub>h</sub>
Name	Input Source #1 - #36
Data type	UNSIGNED8
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	00 <sub>h</sub>

### Description

Subindex 01<sub>h</sub> contains the source for bit 0 of object 60FD. Subindex 02<sub>h</sub> contains the source for bit 1 of object 60FD and so on.

The number that is written in a subindex determines the source for the corresponding bit. The following table lists all possible signal sources.

Number		Signal source
dec	hex	
00	00	Signal is always 0
01	01	Physical input 1
02	02	Physical input 2
03	03	Physical input 3
04	04	Physical input 4
05	05	Physical input 5
06	06	Physical input 6
07	07	Physical input 7
08	08	Physical input 8
09	09	Physical input 9
10	0A	Physical input 10
11	0B	Physical input 11
12	0C	Physical input 12
13	0D	Physical input 13
14	0E	Physical input 14
15	0F	Physical input 15
16	10	Physical input 16
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"
128	80	Signal is always 1
129	81	Inverted physical input 1
130	82	Inverted physical input 2
131	83	Inverted physical input 3
132	84	Inverted physical input 4
133	85	Inverted physical input 5
134	86	Inverted physical input 6
135	87	Inverted physical input 7
136	88	Inverted physical input 8
137	89	Inverted physical input 9
138	8A	Inverted physical input 10
139	8B	Inverted physical input 11
140	8C	Inverted physical input 12
141	8D	Inverted physical input 13
142	8E	Inverted physical input 14
143	8F	Inverted physical input 15
144	90	Inverted physical input 16
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"

Number		
dec	hex	Signal source
198	C6	Inverted encoder input "Index"

## 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.

#### Note



Do not use this function in combination with a homing operation. The homing operation cannot otherwise be successfully completed.

### Object description

Index	3243 <sub>h</sub>
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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	04 <sub>h</sub>
Subindex	01 <sub>h</sub>
Name	Control
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	

Preset value	00000000 <sub>h</sub>
Subindex	02 <sub>h</sub>
Name	Capture Count
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>
Subindex	03 <sub>h</sub>
Name	Value
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>
Subindex	04 <sub>h</sub>
Name	Sensor Raw Value
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>

## Description

- Subindex 01<sub>h</sub>: 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<sub>h</sub>: Specifies the number of the noted level changes since the time the function was started; is reset to 0 if subindex 01<sub>h</sub> is set to 1,2 or 3
- Subindex 03<sub>h</sub>: Encoder position of the level change (in absolute user units from 6064<sub>h</sub>)
- Subindex 04<sub>h</sub>: 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 <sub>h</sub>
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	<p>Firmware version FIR-v1426: Subindex 01<sub>h</sub>: "Name" entry changed from "Special Function Disable" to "Special Function Enable"</p> <p>Firmware version FIR-v1446: "Name" entry changed from "Special Function Enable" to "No Function".</p> <p>Firmware version FIR-v1512: The number of entries was changed from 6 to 9.</p> <p>Firmware version FIR-v2039: Subindex 09 added</p>

## Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	09 <sub>h</sub>
Subindex	01 <sub>h</sub>
Name	No Function
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>
Subindex	02 <sub>h</sub>
Name	Function Inverted
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	

Preset value	00000000 <sub>h</sub>
Subindex	03 <sub>h</sub>
Name	Force Enable
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>
Subindex	04 <sub>h</sub>
Name	Force Value
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>
Subindex	05 <sub>h</sub>
Name	Raw Value
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>
Subindex	06 <sub>h</sub>
Name	Reserved1
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>
Subindex	07 <sub>h</sub>
Name	Reserved2
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>
Subindex	08 <sub>h</sub>

Name	Routing Enable
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>

---

Subindex	09 <sub>h</sub>
Name	Enable Mask [Bit0=StatusLed, Bit1=ErrorLed]
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	FFFFFFFF <sub>h</sub>

## Description

The subindices have the following function:

- 01<sub>h</sub>: No function.
- 02<sub>h</sub>: This subindex is used to invert the logic (from normally closed logic to normally open logic).
- 03<sub>h</sub>: 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<sub>h</sub>.
- 04<sub>h</sub>: 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<sub>h</sub>: The bit combination applied to the outputs is stored in this subindex.
- 08<sub>h</sub>: If the subindex is set to "1", *Output Routing* is activated.



### Note

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

- 09<sub>h</sub>: 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<sub>h</sub>.

### Object description

Index	3252 <sub>h</sub>
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	

---

## Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	05 <sub>h</sub>

---

Subindex	01 <sub>h</sub>
Name	Output Control #1
Data type	UNSIGNED16
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	1080 <sub>h</sub>

---

Subindex	02 <sub>h</sub>
Name	Output Control #2
Data type	UNSIGNED16
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	0090 <sub>h</sub>

---

Subindex	03 <sub>h</sub>
Name	Output Control #3
Data type	UNSIGNED16
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	0091 <sub>h</sub>

---

Subindex	04 <sub>h</sub>
Name	Output Control #4
Data type	UNSIGNED16
Access	read / write
PDO mapping	TX-PDO

Allowed values	
Preset value	0092 <sub>h</sub>
<hr/>	
Subindex	05 <sub>h</sub>
Name	Output Control #5
Data type	UNSIGNED16
Access	read / write
PDO mapping	TX-PDO
Allowed values	
Preset value	0093 <sub>h</sub>

## 3320h Read Analogue Input

### Function

This object displays the instantaneous values of the analog inputs in user-defined units.

### Object description

Index	3320 <sub>h</sub>
Object name	Read Analogue Input
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 <sub>h</sub>
Name	Number Of Analogue Inputs
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02 <sub>h</sub>
<hr/>	
Subindex	01 <sub>h</sub>
Name	Analogue Input 1
Data type	INTEGER32
Access	read only

PDO mapping	TX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>
<hr/>	
Subindex	02 <sub>h</sub>
Name	Analogue Input 2
Data type	INTEGER32
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>

## Description

The user-defined units are made up of offset (3321<sub>h</sub>) and scaling value (3322<sub>h</sub>/ 3323<sub>h</sub>). If both are still set to the default values, the value in 3320<sub>h</sub> is specified in the *ADC Digits* unit.

Formula for converting from digits to the respective unit:

- Voltage input: x digits \* 3.3 V / 1023 digits
- Current input (if configurable): x digits \* 20 mA / 1023 digits

The following applies for the sub-entries:

- Subindex 00<sub>h</sub>: Number of analog inputs
- Subindex 01<sub>h</sub>: Analog value 1
- Subindex 02<sub>h</sub>: Analog value 2 (if present)

## 3321h Analogue Input Offset

### Function

Offset that is added to the read analog value (3220<sub>h</sub>) before scaling (multiplier from object 3322 and divisor from object 3323<sub>h</sub>).

### Object description

Index	3321 <sub>h</sub>
Object name	Analogue Input Offset
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 <sub>h</sub>
Name	Number Of Analogue Inputs
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02 <sub>h</sub>
Subindex	01 <sub>h</sub>
Name	Analogue Input 1
Data type	INTEGER16
Access	read / write
PDO mapping	no
Allowed values	
Preset value	0000 <sub>h</sub>
Subindex	02 <sub>h</sub>
Name	Analogue Input 2
Data type	INTEGER16
Access	read / write
PDO mapping	no
Allowed values	
Preset value	0000 <sub>h</sub>

## Description

- Subindex 00<sub>h</sub>: Number of offsets
- Subindex 01<sub>h</sub>: Offset for analog input 1
- Subindex 02<sub>h</sub>: Offset for analog input 2 (if present)

## 3322h Analogue Input Factor Numerator

### Function

Value by which the read analog value (3220<sub>h</sub>, 3321<sub>h</sub>) is multiplied before it is written in object 3320<sub>h</sub>.

### Object description

Index	3322 <sub>h</sub>
Object name	Analogue Input Factor Numerator
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	Firmware version FIR-v2039-B807052: entry "Object Name" changed from "Analog Input Factor Numerator" to "Analogue Input Factor Numerator".

## Value description

Subindex	00 <sub>h</sub>
Name	Number Of Analogue Inputs
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02 <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	Analogue Input 1
Data type	INTEGER16
Access	read / write
PDO mapping	no
Allowed values	
Preset value	0001 <sub>h</sub>

Subindex	02 <sub>h</sub>
Name	Analogue Input 2
Data type	INTEGER16
Access	read / write
PDO mapping	no
Allowed values	
Preset value	0001 <sub>h</sub>

## Description

The subindices contain:

- Subindex 01<sub>h</sub>: Multiplier for analog input 1
- Subindex 02<sub>h</sub>: Multiplier for analog input 2 (if present)

## 3323h Analogue Input Factor Denominator

### Function

Value by which the read analog value (3320<sub>h</sub>+ 3321<sub>h</sub>) is divided before it is written in object 3320<sub>h</sub>.

## Object description

Index	3323 <sub>h</sub>
Object name	Analogue Input Factor Denominator
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 <sub>h</sub>
Name	Number Of Analogue Inputs
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02 <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	Analogue Input 1
Data type	INTEGER16
Access	read / write
PDO mapping	no
Allowed values	
Preset value	0001 <sub>h</sub>

Subindex	02 <sub>h</sub>
Name	Analogue Input 2
Data type	INTEGER16
Access	read / write
PDO mapping	no
Allowed values	
Preset value	0001 <sub>h</sub>

## Description

The subindices contain:

- Subindex 01<sub>h</sub>: Divisor for analog input 1
- Subindex 02<sub>h</sub>: 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 <sub>h</sub>
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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	05 <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	Resistance [Ohm]
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>

Subindex	02 <sub>h</sub>
Name	Inductance [H]
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	

Preset value	00000000 <sub>h</sub>
Subindex	03 <sub>h</sub>
Name	Magnetic Flux [Vs]
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>
Subindex	04 <sub>h</sub>
Name	Switch On Speed [rpm]
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000078 <sub>h</sub>
Subindex	05 <sub>h</sub>
Name	Switch Off Speed [rpm]
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000064 <sub>h</sub>

## Description

The subindices have the following function:

- 01<sub>h</sub>: Winding resistance. Float value, shown here as UNSIGNED32. Is determined by [Auto setup](#).
- 02<sub>h</sub>: Winding inductance. Float value, shown here as UNSIGNED32. Is determined by [Auto setup](#).
- 03<sub>h</sub>: Interlinking flux. Float value, shown here as UNSIGNED32. Is determined by [Auto setup](#).
- 04<sub>h</sub>: Switch-on speed in RPM. *Closed loop ( sensorless)* is activated above this speed if no sensors were detected by [Auto setup](#).
- 05<sub>h</sub>: 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 <sub>h</sub>
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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	0C <sub>h</sub>

---

Subindex	01 <sub>h</sub>
Name	1st Alignment
Data type	UNSIGNED16
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000 <sub>h</sub>

---

Subindex	02 <sub>h</sub>
Name	2nd Alignment
Data type	UNSIGNED16
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000 <sub>h</sub>

---

Subindex	03 <sub>h</sub>
Name	3rd Alignment
Data type	UNSIGNED16
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000 <sub>h</sub>

---

Subindex	04 <sub>h</sub>
Name	4th Alignment
Data type	UNSIGNED16
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000 <sub>h</sub>

Subindex	05 <sub>h</sub>
Name	5th Alignment
Data type	UNSIGNED16
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000 <sub>h</sub>

Subindex	06 <sub>h</sub>
Name	6th Alignment
Data type	UNSIGNED16
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000 <sub>h</sub>

Subindex	07 <sub>h</sub>
Name	7th Alignment
Data type	UNSIGNED16
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000 <sub>h</sub>

Subindex	08 <sub>h</sub>
Name	8th Alignment
Data type	UNSIGNED16
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000 <sub>h</sub>

Subindex	09 <sub>h</sub>
Name	9th Alignment
Data type	UNSIGNED16

Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000 <sub>h</sub>

---

Subindex	0A <sub>h</sub>
Name	10th Alignment
Data type	UNSIGNED16
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000 <sub>h</sub>

---

Subindex	0B <sub>h</sub>
Name	11th Alignment
Data type	UNSIGNED16
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000 <sub>h</sub>

---

Subindex	0C <sub>h</sub>
Name	12th Alignment
Data type	UNSIGNED16
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000 <sub>h</sub>

---

## 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 <sub>h</sub>
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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	02 <sub>h</sub>

---

Subindex	01 <sub>h</sub>
Name	Configuration
Data type	UNSIGNED16
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000 <sub>h</sub>

---

Subindex	02 <sub>h</sub>
Name	Alignment
Data type	UNSIGNED16
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000 <sub>h</sub>

---

## Description

The subindices have the following function:

- 01<sub>h</sub> (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<sub>h</sub> (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.

## 33A1h Feedback Incremental A/B/I 2

### Function

Contains configuration values for the second incremental encoder. The values are determined by the [Auto setup](#).

## Object description

Index	33A1 <sub>h</sub>
Object name	Feedback Incremental A/B/I 2
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-B533384
Change history	

## Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	02 <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	Configuration
Data type	UNSIGNED16
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000 <sub>h</sub>

Subindex	02 <sub>h</sub>
Name	Alignment
Data type	UNSIGNED16
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000 <sub>h</sub>

## Description

### Description

The subindices have the following function:

- 01<sub>h</sub> (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<sub>h</sub> (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.

## 3502h MODBUS Rx PDO Mapping

### Function

The objects for RX mapping can be written in this object.

#### Note



To be able to change the mapping, you must first deactivate it by setting subindex 0<sub>h</sub> to "0".

After writing the objects to the respective subindices, enter the number of mapped objects in subindex 0<sub>h</sub>.

### Object description

Index	3502 <sub>h</sub>
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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read / write
PDO mapping	no
Allowed values	
Preset value	08 <sub>h</sub>
Subindex	01 <sub>h</sub>
Name	1st Object To Be Mapped
Data type	UNSIGNED32

Access	read / write
PDO mapping	no
Allowed values	
Preset value	60400010 <sub>h</sub>
<hr/>	
Subindex	02 <sub>h</sub>
Name	2nd Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00050008 <sub>h</sub>
<hr/>	
Subindex	03 <sub>h</sub>
Name	3rd Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	60600008 <sub>h</sub>
<hr/>	
Subindex	04 <sub>h</sub>
Name	4th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	32020020 <sub>h</sub>
<hr/>	
Subindex	05 <sub>h</sub>
Name	5th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	607A0020 <sub>h</sub>
<hr/>	
Subindex	06 <sub>h</sub>
Name	6th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	

Preset value	60810020 <sub>h</sub>
Subindex	07 <sub>h</sub>
Name	7th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	60420010 <sub>h</sub>
Subindex	08 <sub>h</sub>
Name	8th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	60FE0120 <sub>h</sub>
Subindex	09 <sub>h</sub>
Name	9th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>
Subindex	0A <sub>h</sub>
Name	10th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>
Subindex	0B <sub>h</sub>
Name	11th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>
Subindex	0C <sub>h</sub>



Name	12th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>

Subindex	0D <sub>h</sub>
Name	13th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>

Subindex	0E <sub>h</sub>
Name	14th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>

Subindex	0F <sub>h</sub>
Name	15th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>

Subindex	10 <sub>h</sub>
Name	16th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>

## 3602h MODBUS Tx PDO Mapping

### Function

The objects for TX mapping can be written in this object.

<b>Note</b>
-------------



To be able to change the mapping, you must first deactivate it by setting subindex 0<sub>h</sub> to "0".

After writing the objects to the respective subindices, enter the number of mapped objects in subindex 0<sub>h</sub>.

## Object description

Index	3602 <sub>h</sub>
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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read / write
PDO mapping	no
Allowed values	
Preset value	06 <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	1st Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	60410010 <sub>h</sub>

Subindex	02 <sub>h</sub>
Name	2nd Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no

Allowed values	
Preset value	00050008 <sub>h</sub>
<hr/>	
Subindex	03 <sub>h</sub>
Name	3rd Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	60610008 <sub>h</sub>
<hr/>	
Subindex	04 <sub>h</sub>
Name	4th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	60640020 <sub>h</sub>
<hr/>	
Subindex	05 <sub>h</sub>
Name	5th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	60440010 <sub>h</sub>
<hr/>	
Subindex	06 <sub>h</sub>
Name	6th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	60FD0020 <sub>h</sub>
<hr/>	
Subindex	07 <sub>h</sub>
Name	7th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>

Subindex	08 <sub>h</sub>
Name	8th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>
Subindex	09 <sub>h</sub>
Name	9th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>
Subindex	0A <sub>h</sub>
Name	10th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>
Subindex	0B <sub>h</sub>
Name	11th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>
Subindex	0C <sub>h</sub>
Name	12th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>
Subindex	0D <sub>h</sub>
Name	13th Object To Be Mapped
Data type	UNSIGNED32

Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>
<hr/>	
Subindex	0E <sub>h</sub>
Name	14th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>
<hr/>	
Subindex	0F <sub>h</sub>
Name	15th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>
<hr/>	
Subindex	10 <sub>h</sub>
Name	16th Object To Be Mapped
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>

## 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 <sub>h</sub>
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 <sub>h</sub>
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 ... -1	Reserved
0	Blocking of the drive function – motor can turn freely
1	Braking with <i>slow down ramp</i> (braking deceleration depending on operating mode)
2	Braking with <i>quick stop ramp</i> (6085 <sub>h</sub> )
3 ... 32767	Reserved

## 3701h Limit Switch Error Option Code

### Function

If a limit switch is passed over, bit 7 ( *Warning*) is set in 6041<sub>h</sub> ( *statusword*) and the action that is stored in this object executed.

### Object description

Index	3701 <sub>h</sub>
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 <sub>h</sub>
Firmware version	FIR-v1748-B538662
Change history	

## Description

Value in object 3701 <sub>h</sub>	Description
-1	No reaction (e. g., to execute a homing operation)
1	Braking with <i>slow down ramp</i> (deceleration ramp depending on operating mode) and subsequent state change to <i>Switch on disabled</i>
2	Braking with <i>quick stop ramp</i> and subsequent state change to <i>Switch on disabled</i>
5	Braking with <i>slow down ramp</i> (deceleration ramp depending on operating mode) and subsequent state change to <i>Quick stop active</i> ;

Value in object 3701 <sub>h</sub>	Description
6	control does not switch off and the motor remains energized. You can switch back to the <i>Operation enabled</i> state. Braking with <i>quick stop ramp</i> and subsequent state change to <i>Quick Stop Active</i> ; control does not switch off and the motor remains energized. You can switch back to the <i>Operation enabled</i> state.

#### Note



The quick-stop bit (bit 2) in 6040<sub>h</sub> is not automatically set to "0" when the state changes to *Quick stop active*.

► If you want to change the *state machine* back to the *Operation enabled* state, you must set the bit to "0" and then to "1" again.

## 4012h HW Information

### Function

This object contains information about the hardware.

### Object description

Index	4012 <sub>h</sub>
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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	01 <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	EEPROM Size In Bytes
Data type	UNSIGNED32

Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>

---

## 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 <sub>h</sub>
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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	01 <sub>h</sub>

---



---

Subindex	01 <sub>h</sub>
Name	HW Configuration #1
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>

---



## Description

reserved

## 4014h Operating Conditions

### Function

This object is used to read out the current environment values for the controller.

### Object description

Index	4014 <sub>h</sub>
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	<p>Firmware version FIR-v1650-B472161: "Access" table entry for subindex 01 changed from "read/write" to "read only".</p> <p>Firmware version FIR-v1650-B472161: "Access" table entry for subindex 02 changed from "read/write" to "read only".</p> <p>Firmware version FIR-v1650-B472161: "Name" entry changed from "Temperature PCB [d?C]" to "Temperature PCB [Celsius * 10]".</p> <p>Firmware version FIR-v1650-B472161: "Access" table entry for subindex 03 changed from "read/write" to "read only".</p> <p>Firmware version FIR-v1738-B501312: The number of entries was changed from 4 to 6.</p>

### Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	05 <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	Voltage UB Power [mV]
Data type	INTEGER32
Access	read only
PDO mapping	TX-PDO

Allowed values	
Preset value	00000000 <sub>h</sub>
<hr/>	
Subindex	02 <sub>h</sub>
Name	Voltage UB Logic [mV]
Data type	INTEGER32
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>
<hr/>	
Subindex	03 <sub>h</sub>
Name	Temperature PCB [Celsius * 10]
Data type	INTEGER32
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>
<hr/>	
Subindex	04 <sub>h</sub>
Name	Temperature Motor [Celsius * 10]
Data type	INTEGER32
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>
<hr/>	
Subindex	05 <sub>h</sub>
Name	Temperature Microcontroller Chip [Celsius * 10]
Data type	INTEGER32
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>

## Description

The subindices contain:

- 01<sub>h</sub>: Current voltage supply voltage in [mV]
- 02<sub>h</sub>: Current logic voltage in [mV]
- 03<sub>h</sub>: Current temperature of the control board in [d°C] (tenths of degree)
- 04<sub>h</sub>: Reserves
- 05<sub>h</sub>: Reserves

## 4021h Ballast Configuration

### Function

With this object, you switch the ballast circuit on or off and determine its response threshold. Furthermore, you configure the ballast monitoring. You can find details in chapter [External ballast circuit](#).

### Object description

Index	4021 <sub>h</sub>
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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	08 <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	Settings [Bit0: On/Off, Bit1: Polarity]
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>

Subindex	02 <sub>h</sub>
Name	UB Power Limit [mV]
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	0000ED49 <sub>h</sub>

Subindex	03 <sub>h</sub>
Name	UB Power Hysteresis [mV]
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	000001F4 <sub>h</sub>
Subindex	04 <sub>h</sub>
Name	Nominal Resistance [mOhm]
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001 <sub>h</sub>
Subindex	05 <sub>h</sub>
Name	Long Term Energy Limit [mWs]
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001 <sub>h</sub>
Subindex	06 <sub>h</sub>
Name	Long Term Reference Time [ms]
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>
Subindex	07 <sub>h</sub>
Name	Short Term Energy Limit [mWs]
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>
Subindex	08 <sub>h</sub>
Name	Cooling Power [mW]
Data type	UNSIGNED32

Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>

---

## Description

The subindices have the following function:

- 01<sub>h</sub>:
  - Bit 0: Switches the ballast on (value = "1") or off (value = "0")
  - Bit 1: Inverts (value = "1") the polarity of the pins for controlling the external ballast circuit (factory setting: *active high*)
- 02<sub>h</sub>: Response threshold (switch on/off) of the ballast circuit
- 03<sub>h</sub>: Hysteresis for the response threshold (switch on/off)
- 04<sub>h</sub>: Rated value of the ballast resistor
- 05<sub>h</sub>: Amount of energy that can be supplied to the resistor within the *Long Term Reference Time* (subindex 06<sub>h</sub>) without overloading it.
- 06<sub>h</sub>: The reference time for the *Long Term Energy Limit* (subindex 05<sub>h</sub>) (typically between 1 and 5 seconds)
- 07<sub>h</sub>: Amount of energy that can be supplied to the resistor within a short load surge (<1 second) without overloading it.
- 08<sub>h</sub>: The amount of heat that the resistor can/may constantly discharge to its surroundings.

## 4040h Drive Serial Number

### Function

This object contains the serial number of the controller.

### Object description

---

Index	4040 <sub>h</sub>
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 <sub>h</sub>
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 <sub>h</sub>
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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	03 <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	Bootloader Version
Data type	UNSIGNED32
Access	read only
PDO mapping	no

Allowed values	
Preset value	00000000 <sub>h</sub>
<hr/>	
Subindex	02 <sub>h</sub>
Name	Bootloader Supported Fieldbus
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>
<hr/>	
Subindex	03 <sub>h</sub>
Name	Bootloader Hw-group
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000000 <sub>h</sub>

## Description

The subindices have the following functions:

- 01<sub>h</sub>: 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<sub>h</sub>
- 02<sub>h</sub>: 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<sub>h</sub>](#). For the description of the error codes, refer to object [1003<sub>h</sub>](#).

### Object description

Index	603F <sub>h</sub>
Object name	Error Code
Object Code	VARIABLE
Data type	UNSIGNED16
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	0000 <sub>h</sub>
Firmware version	FIR-v1426

Change history
 

---

## Description

For the meaning of the error, see object [1003<sub>h</sub>](#) (Pre-defined Error Field).

## 6040h Controlword

### Function

This object controls the [CiA 402 Power State Machine](#).

### Object description

Index	6040 <sub>h</sub>
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 <sub>h</sub>
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 (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 <sub>h</sub>
Object name	Statusword
Object Code	VARIABLE
Data type	UNSIGNED16
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	0000 <sub>h</sub>
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<sub>h</sub>](#))

#### 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 <sub>h</sub> )	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 VI Target Velocity****Function**

Specifies the target speed in user-defined units for Velocity mode.

**Object description**

Index	6042 <sub>h</sub>
Object name	VI Target Velocity
Object Code	VARIABLE
Data type	INTEGER16
Savable	yes, category: application

Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00C8 <sub>h</sub>
Firmware version	FIR-v1426
Change history	Firmware version FIR-v1626: "Savable" entry changed from "no" to "yes, category: application".

---

## 6043h VI Velocity Demand

### Function

Speed specification in user-defined units for the controller in Velocity mode.

### Object description

Index	6043 <sub>h</sub>
Object name	VI Velocity Demand
Object Code	VARIABLE
Data type	INTEGER16
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	0000 <sub>h</sub>
Firmware version	FIR-v1426
Change history	

---

## 6044h VI Velocity Actual Value

### Function

Specifies the current actual speed in user-defined units in Velocity mode.

### Object description

Index	6044 <sub>h</sub>
Object name	VI Velocity Actual Value
Object Code	VARIABLE
Data type	INTEGER16
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	0000 <sub>h</sub>
Firmware version	FIR-v1426
Change history	

---

## 6046h VI 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 <sub>h</sub>
Object name	VI Velocity Min Max Amount
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Firmware version	FIR-v1426
Change history	

### Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02 <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	MinAmount
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>

Subindex	02 <sub>h</sub>
Name	MaxAmount
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00007530 <sub>h</sub>

### Description

Subindex 1 contains the minimum speed.

Subindex 2 contains the maximum speed.

If the value of the target speed (object [6042<sub>h</sub>](#)) specified here is less than the minimum speed, the minimum speed applies and bit 11 (Internal Limit Reached) in [6041<sub>h</sub> Statusword<sub>h</sub>](#) is set.

A target speed greater than the maximum speed sets the speed to the maximum speed and bit 11 (Internal Limit Reached) in [6041<sub>h</sub> Statusword<sub>h</sub>](#) is set.

## 6048h VI Velocity Acceleration

### Function

Sets the acceleration ramp in Velocity Mode (see [Velocity](#)).

### Object description

Index	6048 <sub>h</sub>
Object name	VI Velocity Acceleration
Object Code	RECORD
Data type	VELOCITY_ACCELERATION_DECELERATION
Savable	yes, category: application
Firmware version	FIR-v1426
Change history	

### Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02 <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	DeltaSpeed
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	000001F4 <sub>h</sub>

Subindex	02 <sub>h</sub>
Name	DeltaTime
Data type	UNSIGNED16
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0001 <sub>h</sub>

## Description

The acceleration is specified as a fraction in user-defined units:

Speed change per change in time.

Subindex 01<sub>h</sub>: Contains the change in speed.

Subindex 02<sub>h</sub>: Contains the change in time.

## 6049h VI Velocity Deceleration

### Function

Sets the deceleration (deceleration ramp) in Velocity Mode (see [Velocity](#)).

### Object description

Index	6049 <sub>h</sub>
Object name	VI Velocity Deceleration
Object Code	RECORD
Data type	VELOCITY_ACCELERATION_DECELERATION
Savable	yes, category: application
Firmware version	FIR-v1426
Change history	

### Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02 <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	DeltaSpeed
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	000001F4 <sub>h</sub>

Subindex	02 <sub>h</sub>
Name	DeltaTime
Data type	UNSIGNED16
Access	read / write
PDO mapping	RX-PDO
Allowed values	

Preset value	0001 <sub>h</sub>
--------------	-------------------

## Description

The deceleration is specified as a fraction in user-defined units:

Speed change per change in time.

Subindex 01<sub>h</sub>: Contains the change in speed.

Subindex 02<sub>h</sub>: 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 <sub>h</sub>
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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02 <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	DeltaSpeed
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00001388 <sub>h</sub>

Subindex	02 <sub>h</sub>
Name	DeltaTime
Data type	UNSIGNED16

Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0001 <sub>h</sub>

---

## Description

The deceleration is specified as a fraction in user-defined units:

Speed change per change in time.

Subindex 01<sub>h</sub>: Contains the change in speed.

Subindex 02<sub>h</sub>: Contains the change in time.

## 604Ch VI Dimension Factor

### Function

The unit for speed values is defined here for the objects associated with velocity mode.

### Object description

Index	604C <sub>h</sub>
Object name	VI Dimension Factor
Object Code	ARRAY
Data type	INTEGER32
Savable	yes, category: application
Firmware version	FIR-v1426
Change history	

---

### Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02 <sub>h</sub>

---

Subindex	01 <sub>h</sub>
Name	VI Dimension Factor Numerator
Data type	INTEGER32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001 <sub>h</sub>

---



Subindex	02 <sub>h</sub>
Name	VI Dimension Factor Denominator
Data type	INTEGER32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001 <sub>h</sub>

## 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 <sub>h</sub>
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 <sub>h</sub>
Firmware version	FIR-v1426
Change history	

### Description

Value in object 605A <sub>h</sub>	Description
0	Immediate stop with subsequent state change to <i>Switch on disabled</i>
1	Braking with <i>slow down ramp</i> (deceleration ramp depending on operating mode) and subsequent state change to <i>Switch on disabled</i>
2	Braking with <i>quick stop ramp</i> (6085 <sub>h</sub> ) and subsequent state change to <i>Switch on disabled</i>
5	Braking with <i>slow down ramp</i> (deceleration ramp depending on operating mode) and subsequent state change to <i>Quick stop active</i> ; control does not switch off and the motor remains energized. You can switch back to the <i>Operation enabled</i> state.
6	Braking with <i>quick stop ramp</i> (6085 <sub>h</sub> ) and subsequent state change to <i>Quick Stop Active</i> ; control does not switch off and the motor

Value in object 605A <sub>h</sub>	Description
	remains energized. You can switch back to the <i>Operation enabled</i> 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 <sub>h</sub>
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 <sub>h</sub>
Firmware version	FIR-v1426
Change history	

### Description

Value in object 605B <sub>h</sub>	Description
-32768 ... -1	Reserved
0	Blocking of the drive function – motor can turn freely
1	Braking with <i>slow down ramp</i> (braking deceleration depending on operating mode) and subsequent state change to <i>Switch on disabled</i>
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 <sub>h</sub>
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 <sub>h</sub>
Firmware version	FIR-v1426
Change history	

## Description

Value in object 605C <sub>h</sub>	Description
-32768 ... -1	Reserved
0	Blocking of the drive function – motor can turn freely
1	Braking with <i>slow down ramp</i> (braking deceleration depending on operating mode) and subsequent state change to <i>Switch on disabled</i>
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<sub>h</sub>.

### Object description

Index	605D <sub>h</sub>
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 <sub>h</sub>
Firmware version	FIR-v1426
Change history	

## Description

Value in object 605D <sub>h</sub>	Description
-32768 ... 0	Reserved
1	Braking with <i>slow down ramp</i> (braking deceleration depending on operating mode)
2	Braking with <i>quick stop ramp</i> ( <u>6085<sub>h</sub></u> )
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 <sub>h</sub>
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 <sub>h</sub>
Firmware version	FIR-v1426
Change history	

### Description

Value in object 605E <sub>h</sub>	Description
-32768 ... -1	Reserved
0	Blocking of the drive function – motor can turn freely
1	Braking with <i>slow down ramp</i> (braking deceleration depending on operating mode)
2	Braking with <i>quick stop ramp</i> (6085 <sub>h</sub> )
3 ... 32767	Reserved

## 6060h Modes Of Operation

### Function

The desired operating mode is entered in this object.

### Object description

Index	6060 <sub>h</sub>
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 <sub>h</sub>
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 <sub>h</sub>
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 <sub>h</sub>
Firmware version	FIR-v1426
Change history	

## 6062h Position Demand Value

### Function

Indicates the current demand position in [user-defined units](#).

## Object description

Index	6062 <sub>h</sub>
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 <sub>h</sub>
Firmware version	FIR-v1426
Change history	

## 6063h Position Actual Internal Value

### Function

Contains the current rotary encoder position in increments. Unlike objects [6062<sub>h</sub>](#) and [6064<sub>h</sub>](#), this value is not set to "0" following a [Homing](#) operation.



#### Note

If the encoder resolution in object [608F<sub>h</sub>](#) = zero, the numerical values of this object are invalid.

## Object description

Index	6063 <sub>h</sub>
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 <sub>h</sub>
Firmware version	FIR-v1426
Change history	

## 6064h Position Actual Value

### Function

Contains the current actual position in [user-defined units](#).

## Object description

Index	6064 <sub>h</sub>
-------	-------------------

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 <sub>h</sub>
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 <sub>h</sub>
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 <sub>h</sub>
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<sub>h</sub> is set. The deviation must last longer than the time in object 6066<sub>h</sub>.

If the value of the "Following Error Window" is set to "FFFFFFFF"<sub>h</sub>, following error monitoring is switched off.

A reaction to the following error can be set in object 3700<sub>h</sub>. If a reaction is defined, an error is also entered in object 1003<sub>h</sub>.

## 6066h Following Error Time Out

### Function

Time in milliseconds until a larger following error results in an error message.

### Object description

Index	6066 <sub>h</sub>
-------	-------------------

---

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 <sub>h</sub>
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<sub>h</sub> is exceeded, bit 13 in object 6041<sub>h</sub> 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<sub>h</sub>. If a reaction is defined, an error is also entered in object 1003<sub>h</sub>.

## 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 <sub>h</sub>
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 <sub>h</sub>
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<sub>h</sub> is set. The condition must be satisfied for longer than the time defined in object 6066<sub>h</sub>.

If the value is set to "FFFFFFFF"<sub>h</sub>, monitoring is switched off.



## 6068h Position Window Time

### Function

The current position must be within the "Position Window" (6067<sub>h</sub>) 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 <sub>h</sub>
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 <sub>h</sub>
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<sub>h</sub>, bit 10 in object 6041<sub>h</sub> is set. The condition must be satisfied for longer than the time defined in object 6066<sub>h</sub>.

## 606Bh Velocity Demand Value

### Function

Speed specification in user-defined units for the velocity controller.

### Object description

Index	606B <sub>h</sub>
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 <sub>h</sub>
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 <sub>h</sub>
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 <sub>h</sub>
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 <sub>h</sub>
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 <sub>h</sub>
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 [6041<sub>h</sub>](#) is set. The condition must be satisfied for longer than the time defined in object [606E<sub>h</sub>](#) (see also [statusword in Profile Velocity Mode](#)).

## 606Eh Velocity Window Time

### Function

The current speed must be within the "Velocity Window" ([606D<sub>h</sub>](#)) for this time (in milliseconds) for the target to be considered having been met.

### Object description

---

Index	606E <sub>h</sub>
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	0000 <sub>h</sub>
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 [606D<sub>h</sub>](#), bit 10 in object [6041<sub>h</sub>](#) 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	606F <sub>h</sub>
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 <sub>h</sub>
Firmware version	FIR-v2013-B726332
Change history	

---

## Description

If the actual speed is greater than the value in 606E<sub>h</sub>(Velocity Threshold) for a time of 6070<sub>h</sub>(Velocity Threshold Time), bit 12 in 6041<sub>h</sub>(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 606E<sub>h</sub> in Profile Velocity mode is considered to be nonzero.

### Object description

---

Index	6070 <sub>h</sub>
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 <sub>h</sub>
Firmware version	FIR-v2013-B726332
Change history	

---

## Description

If the actual speed is greater than the value in 606E<sub>h</sub>(Velocity Threshold) for a time of 6070<sub>h</sub>(Velocity Threshold Time), bit 12 in 6041<sub>h</sub>(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 <sub>h</sub>
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 <sub>h</sub>
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<sub>h</sub>:01.

The minimum of 6073<sub>h</sub> and 6072<sub>h</sub> is used as limit for the torque in 6071<sub>h</sub>.

The target torque may not exceed the peak torque (proportional to the maximum motor current in 2031<sub>h</sub>).

## 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 <sub>h</sub>
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 <sub>h</sub>
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<sub>h</sub>:01.

The minimum of 6073<sub>h</sub> and 6072<sub>h</sub> is used as limit for the torque in 6071<sub>h</sub>.

The target torque may not exceed the peak torque (proportional to the maximum motor current in 2031<sub>h</sub>).

## 6073h Max Current

### Function

Contains the maximum current in tenths of a percent of the set rated current entered in 320E<sub>h</sub>:0A<sub>h</sub>. Is limited by the maximum motor current (2031<sub>h</sub>). See also I2t Motor overload protection.

<b>Note</b>
-------------



For stepper motors, only the rated current is specified, not a maximum current. The value of 6073<sub>h</sub> should therefore not exceed the value 1000 (100%).

## Object description

Index	6073 <sub>h</sub>
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 <sub>h</sub>
Firmware version	FIR-v1825-B577172
Change history	

## Description

The maximum current is calculated in tenths of a percent of the rated current as follows:

$$(6073_{\text{h}} * 203\text{B}_{\text{h}} : 01) / 1000$$

The maximum current determines:

- the maximum current for the I<sub>2t</sub> Motor overload protection
- the rated current in *open loop* mode

<b>Note</b>
-------------



The maximum current also affects the control behavior in *closed loop* mode (see Controller structure). If you change the maximum current, you must also proportionally adjust the value of 320E<sub>h</sub>:09<sub>h</sub>.

## 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 <sub>h</sub>
Object name	Torque Demand
Object Code	VARIABLE
Data type	INTEGER16
Savable	no
Access	read only
PDO mapping	TX-PDO

Allowed values	
Preset value	0000 <sub>h</sub>
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<sub>h</sub>:01.

The minimum of 6073<sub>h</sub> and 6072<sub>h</sub> is used as limit for the torque in 6071<sub>h</sub>.

The target torque may not exceed the peak torque (proportional to the maximum motor current in 2031<sub>h</sub>).

## 6075h Motor Rated Current

### Function

Contains the rated current entered in 203B<sub>h</sub>:01<sub>h</sub> 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 <sub>h</sub>
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 <sub>h</sub>
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<sub>h</sub>:01.

The minimum of 6073<sub>h</sub> and 6072<sub>h</sub> is used as limit for the torque in 6071<sub>h</sub>.

The target torque may not exceed the peak torque (proportional to the maximum motor current in 2031<sub>h</sub>).

## 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 <sub>h</sub>
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 <sub>h</sub>
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 <sub>h</sub>
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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02 <sub>h</sub>



Subindex	01 <sub>h</sub>
Name	Min Position Range Limit
Data type	INTEGER32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>

Subindex	02 <sub>h</sub>
Name	Max Position Range Limit
Data type	INTEGER32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>

## 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<sub>h</sub> ("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 <sub>h</sub>
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 <sub>h</sub>
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 <sub>h</sub>
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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02 <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	Min Position Limit
Data type	INTEGER32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>

Subindex	02 <sub>h</sub>
Name	Max Position Limit
Data type	INTEGER32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>

## Description

The absolute target position must lie within the limits set here. The Home Offset (607C<sub>h</sub>) is not taken into account.

## 607Eh Polarity

### Function

With this object, the direction of rotation can be reversed.

## Object description

Index	607E <sub>h</sub>
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 <sub>h</sub>
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<sub>h</sub>:02<sub>h</sub>](#).

## 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 <sub>h</sub>
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 <sub>h</sub>
Firmware version	FIR-v1540
Change history	<p>Firmware version FIR-v1738-B501312: "Object Name" entry changed from "Max profile velocity" to "Max Profile Velocity".</p> <p>Firmware version FIR-v1738-B501312: "Data type" entry changed from "INTEGER16" to "UNSIGNED32".</p> <p>Firmware version FIR-v1738-B501312: "Savable" entry changed from "no" to "yes, category: application".</p> <p>Firmware version FIR-v1738-B501312: "Access" table entry for subindex 00 changed from "read only" to "read/write".</p> <p>Firmware version FIR-v1738-B501312: "PDO mapping" table entry for subindex 00 changed from "TX-PDO" to "RX-PDO".</p>

## 6080h Max Motor Speed

### Function

Contains the maximum permissible speed of the motor in user-defined units entered in 320E<sub>h</sub>:05<sub>h</sub>.

#### Note



The maximum speed also affects the control behavior in *closed loop* mode (see Controller structure). If you change the maximum speed, you must also proportionally adjust the value of 320E<sub>h</sub>:04<sub>h</sub>.

### Object description

Index	6080 <sub>h</sub>
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 <sub>h</sub>
Firmware version	FIR-v1426
Change history	<p>Firmware version FIR-v1614: "Savable" entry changed from "yes, category: application" to "yes, category: tuning".</p> <p>Firmware version FIR-v1738-B501312: "Object Name" entry changed from "Maximum Speed" to "Max Motor Speed".</p>

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 <sub>h</sub>
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 <sub>h</sub>
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 <sub>h</sub>
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 <sub>h</sub>
Firmware version	FIR-v1426
Change history	

## 6083h Profile Acceleration

### Function

Specifies the maximum acceleration in user-defined units.

### Object description

Index	6083 <sub>h</sub>
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 <sub>h</sub>
Firmware version	FIR-v1426
Change history	

## 6084h Profile Deceleration

### Function

Specifies the maximum deceleration (deceleration ramp) in user-defined units. Is limited by 60C6<sub>h</sub>.

### Object description

Index	6084 <sub>h</sub>
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 <sub>h</sub>
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<sub>h</sub> (Max Deceleration) and, if applicable, 60A4<sub>h</sub> (Profile Jerk).

### Object description

Index	6085 <sub>h</sub>
-------	-------------------

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 <sub>h</sub>
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 <sub>h</sub>
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 <sub>h</sub>
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 <sub>h</sub>
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	00000000 <sub>h</sub>
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<sub>h</sub>:01](#).

The minimum of [6073<sub>h</sub>](#) and [6072<sub>h</sub>](#) is used as limit for the torque in [6071<sub>h</sub>](#).

The target torque may not exceed the peak torque (proportional to the maximum motor current in [2031<sub>h</sub>](#)).

## 608F<sub>h</sub> Position Encoder Resolution

### Function

Contains the physical resolution (see objects [60E6<sub>h</sub>](#)/[60EB<sub>h</sub>](#)) of the encoder/sensor that is used for position control (see [3203<sub>h</sub> Feedback Selection](#)).

### Object description

Index	608F <sub>h</sub>
Object name	Position Encoder Resolution
Object Code	ARRAY
Data type	INTEGER32
Savable	yes, category: tuning
Firmware version	FIR-v1426
Change history	<p>Firmware version FIR-v1738-B501312: "Savable" entry changed from "yes, category: application" to "yes, category: tuning".</p> <p>Firmware version FIR-v1738-B501312: "PDO mapping" table entry for subindex 01 changed from "no" to "RX-PDO".</p> <p>Firmware version FIR-v1738-B501312: "PDO mapping" table entry for subindex 02 changed from "no" to "RX-PDO".</p> <p>Firmware version FIR-v1748-B538662: "Data type" entry changed from "UNSIGNED32" to "INTEGER32".</p>

---

### Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02 <sub>h</sub>

---



Subindex	01 <sub>h</sub>
Name	Encoder Increments
Data type	INTEGER32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	000007D0 <sub>h</sub>

Subindex	02 <sub>h</sub>
Name	Motor Revolutions
Data type	INTEGER32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001 <sub>h</sub>

## Description

Position Encoder Resolution = Encoder Increments (608F<sub>h</sub>:01<sub>h</sub>) / Motor Revolutions (608F<sub>h</sub>:02<sub>h</sub>)

## 6090h Velocity Encoder Resolution

### Function

Contains the physical resolution (see objects 60E6<sub>h</sub>/ 60EB<sub>h</sub>) of the encoder/sensor that is used for speed control (see 3203h Feedback Selection).

### Object description

Index	6090 <sub>h</sub>
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	<p>Firmware version FIR-v1748-B538662: "Data type" entry changed from "UNSIGNED32" to "INTEGER32".</p> <p>Firmware version FIR-v1748-B538662: "Data type" entry changed from "UNSIGNED32" to "INTEGER32".</p> <p>Firmware version FIR-v1748-B538662: "Data type" entry changed from "UNSIGNED32" to "INTEGER32".</p> <p>Firmware version FIR-v1825-B577172: "Data type" entry changed from "INTEGER32" to "UNSIGNED32".</p>

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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02 <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	Encoder Increments Per Second
Data type	INTEGER32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>

Subindex	02 <sub>h</sub>
Name	Motor Revolutions Per Second
Data type	INTEGER32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>

## Description

Velocity Encoder Resolution = Encoder Increments per second (6090<sub>h</sub>:01<sub>h</sub>) / Motor Revolutions per second (6090<sub>h</sub>:02<sub>h</sub>)

## 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 <sub>h</sub>
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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02 <sub>h</sub>
Subindex	01 <sub>h</sub>
Name	Motor Revolutions
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001 <sub>h</sub>
Subindex	02 <sub>h</sub>
Name	Shaft Revolutions
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001 <sub>h</sub>

## Description

Gear Ratio = Motor Revolutions (6091<sub>h</sub>:01<sub>h</sub>) / Shaft Revolutions (6091<sub>h</sub>:02<sub>h</sub>)

## 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 <sub>h</sub>
Object name	Feed Constant
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Firmware version	FIR-v1426
Change history	

### Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02 <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	Feed
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001 <sub>h</sub>

Subindex	02 <sub>h</sub>
Name	Shaft Revolutions
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001 <sub>h</sub>

## Description

Feed Constant = Feed (6092<sub>h</sub>:01<sub>h</sub>) / Shaft Revolutions (6092<sub>h</sub>:02<sub>h</sub>)

## 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 <sub>h</sub>
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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02 <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	Numerator
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001 <sub>h</sub>

Subindex	02 <sub>h</sub>
Name	Divisor
Data type	UNSIGNED32
Access	read / write

PDO mapping	RX-PDO
Allowed values	
Preset value	00000001 <sub>h</sub>

---

## Description

The subindices have the following functions:

- 01<sub>h</sub>: Numerator of the factor
- 02<sub>h</sub>: 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 <sub>h</sub>
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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02 <sub>h</sub>

---



---

Subindex	01 <sub>h</sub>
Name	Numerator
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	

Preset value	00000001 <sub>h</sub>
Subindex	02 <sub>h</sub>
Name	Divisor
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001 <sub>h</sub>

## Description

The subindices have the following functions:

- 01<sub>h</sub>: Numerator of the factor
- 02<sub>h</sub>: Denominator of the factor

## 6098h Homing Method

### Function

This object defines the Homing method in Homing mode.

### Object description

Index	6098 <sub>h</sub>
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 <sub>h</sub>
Firmware version	FIR-v1426
Change history	

## 6099h Homing Speed

### Function

Specifies the speeds for homing mode (6098<sub>h</sub>) in user-defined units.

### Object description

Index	6099 <sub>h</sub>
Object name	Homing Speed
Object Code	ARRAY
Data type	UNSIGNED32

Savable	yes, category: application
Firmware version	FIR-v1426
Change history	

## Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02 <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	Speed During Search For Switch
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000032 <sub>h</sub>

Subindex	02 <sub>h</sub>
Name	Speed During Search For Zero
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	0000000A <sub>h</sub>

## 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.

### Note



- 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. The speed in subindex 2 should therefore be less than 1000 steps per second.
- 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 <sub>h</sub>
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 <sub>h</sub>
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 <sub>h</sub>
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 <sub>h</sub>
Name	Highest Sub-index Supported

Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02 <sub>h</sub>
<hr/>	
Subindex	01 <sub>h</sub>
Name	Numerator
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001 <sub>h</sub>
<hr/>	
Subindex	02 <sub>h</sub>
Name	Divisor
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001 <sub>h</sub>

## Description

The subindices have the following functions:

- 01<sub>h</sub>: Numerator of the factor
- 02<sub>h</sub>: 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 <sub>h</sub>
Object name	Profile Jerk
Object Code	ARRAY
Data type	UNSIGNED32
Savable	yes, category: application
Firmware version	FIR-v1426
Change history	<p>Firmware version FIR-v1614: "Name" entry changed from "End Acceleration Jerk" to "Begin Deceleration Jerk".</p> <p>Firmware version FIR-v1614: "Name" entry changed from "Begin Deceleration Jerk" to "End Acceleration Jerk".</p>

## Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	04 <sub>h</sub>
Subindex	01 <sub>h</sub>
Name	Begin Acceleration Jerk
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	000003E8 <sub>h</sub>
Subindex	02 <sub>h</sub>
Name	Begin Deceleration Jerk
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	000003E8 <sub>h</sub>
Subindex	03 <sub>h</sub>
Name	End Acceleration Jerk
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	000003E8 <sub>h</sub>
Subindex	04 <sub>h</sub>
Name	End Deceleration Jerk
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	000003E8 <sub>h</sub>

## Description

- Subindex 01<sub>h</sub> ( *Begin Acceleration Jerk*): Initial jerk during acceleration
- Subindex 02<sub>h</sub> ( *Begin Deceleration Jerk*): Initial jerk during braking
- Subindex 03<sub>h</sub> ( *End Acceleration Jerk*): Final jerk during acceleration
- Subindex 04<sub>h</sub> ( *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 <sub>h</sub>
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 <sub>h</sub>
Firmware version	FIR-v1738-B501312
Change history	

### Description

Object 60A8<sub>h</sub> 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 <sub>h</sub>
Object name	SI Unit Velocity
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	yes, category: application

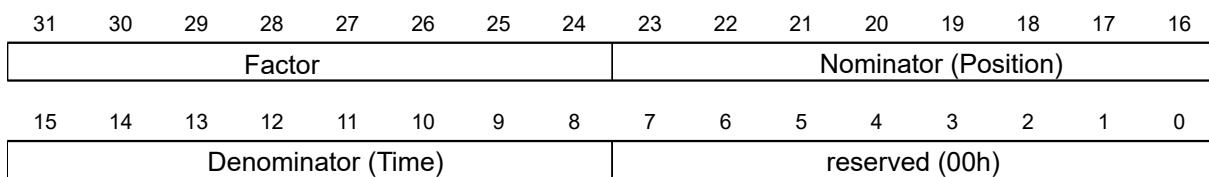
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00B44700 <sub>h</sub>
Firmware version	FIR-v1738-B501312
Change history	

---

## Description

Object 60A9<sub>h</sub> 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](#))



## 60B0h Position Offset

### Function

Offset for the position set value in [user-defined units](#).

### Object description

---

Index	60B0 <sub>h</sub>
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 <sub>h</sub>
Firmware version	FIR-v1738-B505321
Change history	

---

## 60B1h Velocity Offset

### Function

Offset for the speed set value in [user-defined units](#).

### Object description

---

Index	60B1 <sub>h</sub>
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 <sub>h</sub>
Firmware version	FIR-v1738-B505321
Change history	

---

## 60B2h Torque Offset

### Function

Offset for the torque set value in tenths of a percent.

### Object description

Index	60B2 <sub>h</sub>
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 <sub>h</sub>
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 <sub>h</sub>
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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	01 <sub>h</sub>

---

Subindex	01 <sub>h</sub>
Name	1st Set-point
Data type	INTEGER32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>

---

### 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 <sub>h</sub>
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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	02 <sub>h</sub>
Subindex	01 <sub>h</sub>
Name	Interpolation Time Period Value
Data type	UNSIGNED8
Access	read / write
PDO mapping	no
Allowed values	
Preset value	01 <sub>h</sub>
Subindex	02 <sub>h</sub>
Name	Interpolation Time Index
Data type	INTEGER8
Access	read / write
PDO mapping	no
Allowed values	
Preset value	FD <sub>h</sub>

## Description

The subindices have the following functions:

- 01<sub>h</sub>: Interpolation time.
- 02<sub>h</sub>: 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<sub>h</sub>:01<sub>h</sub> \* 10<sup>value of 60C2:02</sup> 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 <sub>h</sub>
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	<p>Firmware version FIR-v1540: "Access" table entry for subindex 05 changed from "read/write" to "write only".</p> <p>Firmware version FIR-v1540: "Access" table entry for subindex 06 changed from "read/write" to "write only".</p> <p>Firmware version FIR-v1626: "Savable" entry changed from "no" to "yes, category: application".</p> <p>Firmware version FIR-v1650-B472161: "Access" table entry for subindex 01 changed from "read/write" to "read only".</p>

## Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	06 <sub>h</sub>
Subindex	01 <sub>h</sub>
Name	MaximumBufferSize
Data type	UNSIGNED32
Access	read only
PDO mapping	no
Allowed values	
Preset value	00000001 <sub>h</sub>
Subindex	02 <sub>h</sub>
Name	ActualBufferSize
Data type	UNSIGNED32
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00000001 <sub>h</sub>
Subindex	03 <sub>h</sub>
Name	BufferOrganization

Data type	UNSIGNED8
Access	read / write
PDO mapping	no
Allowed values	
Preset value	00 <sub>h</sub>
<hr/>	
Subindex	04 <sub>h</sub>
Name	BufferPosition
Data type	UNSIGNED16
Access	read / write
PDO mapping	no
Allowed values	
Preset value	0001 <sub>h</sub>
<hr/>	
Subindex	05 <sub>h</sub>
Name	SizeOfDataRecord
Data type	UNSIGNED8
Access	write only
PDO mapping	no
Allowed values	
Preset value	04 <sub>h</sub>
<hr/>	
Subindex	06 <sub>h</sub>
Name	BufferClear
Data type	UNSIGNED8
Access	write only
PDO mapping	no
Allowed values	
Preset value	00 <sub>h</sub>

## Description

The value of subindex 01<sub>h</sub> contains the maximum possible number of interpolated records.

The value of subindex 02<sub>h</sub> contains the current number of interpolated records.

If subindex 03<sub>h</sub> is "00<sub>h</sub>", this means a FIFO buffer organization; if it is "01<sub>h</sub>", it specifies a ring buffer organization.

The value of subindex 04<sub>h</sub> is unitless and specifies the next free buffer entry point.

The value of subindex 05<sub>h</sub> is specified in units of "byte".

If the value "00<sub>h</sub>" is written in subindex 06<sub>h</sub>, it deletes the received data in the buffer, deactivates access and deletes all interpolated records.

If the value "01<sub>h</sub>" is written in subindex 06<sub>h</sub>, 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 <sub>h</sub>
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 <sub>h</sub>
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 <sub>h</sub>
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 <sub>h</sub>
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 <sub>h</sub>
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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	04 <sub>h</sub>

Subindex	01 <sub>h</sub> - 04 <sub>h</sub>
Name	Additional Position Actual Value #1 - #4
Data type	INTEGER32
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>

## Description

The subindices have the following function:

- 00<sub>h</sub>: Value="1" to "n", where "n" is the number of existing feedbacks.
- n<sub>h</sub>:  
 Subindex n contains the current actual position of the corresponding feedback.  
 Subindex 01<sub>h</sub> 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](#).

## 60E5h Additional Velocity Actual Value

### Function

Contains the current actual speed of all existing feedbacks in user-defined units.

### Object description

Index	60E5 <sub>h</sub>
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	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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	04 <sub>h</sub>
Subindex	01 <sub>h</sub> - 04 <sub>h</sub>
Name	Additional Velocity Actual Value #1 - #4
Data type	INTEGER32
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>

### Description

The subindices have the following function:

- 00<sub>h</sub>: Value="1" to "n", where "n" is the number of existing feedbacks.
- n<sub>h</sub>:  
Subindex n contains the current actual speed of the corresponding feedback.

Subindex 01<sub>h</sub> 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](#).

## 60E6h Additional Position Encoder Resolution - Encoder Increments

### Function

With this object and with 60EB<sub>h</sub>, the resolution of each existing feedback is calculated.

### Object description

Index	60E6 <sub>h</sub>
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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	04 <sub>h</sub>

Subindex	01 <sub>h</sub> - 04 <sub>h</sub>
Name	Additional Position Encoder Resolution - Encoder Increments Feedback Interface #1 - #4
Data type	INTEGER32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>

### Description

The subindices have the following function:

- 00<sub>h</sub>: Value="1" to "n", where "n" is the number of existing feedbacks.
- n<sub>h</sub>:  
Subindex n contains the number of increments of the corresponding feedback.

Subindex 01<sub>h</sub> 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](#).

The resolution of feedback "n" is calculated as follows:

Position Encoder Resolution = Encoder Increments (60E6<sub>h</sub>:01<sub>h</sub>) / Motor Revolutions (60EB<sub>h</sub>:02<sub>h</sub>)

#### Tip



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.

## 60E8h Additional Gear Ratio - Motor Shaft Revolutions

### Function

In this object and in 60ED<sub>h</sub>, you can set the gear ratio of each existing feedback.

### Object description

Index	60E8 <sub>h</sub>
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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	04 <sub>h</sub>

Subindex	01 <sub>h</sub> - 04 <sub>h</sub>
Name	Additional Gear Ratio - Motor Shaft Revolutions Feedback Interface #1 - #4
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO

Allowed values	
Preset value	00000001 <sub>h</sub>

---

## Description

The subindices have the following function:

- 00<sub>h</sub>: Value = "n", where "n" is the number of existing feedbacks.
- n<sub>h</sub>: Subindex "n" contains the number of motor revolutions for the corresponding feedback. Subindex 01<sub>h</sub> 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](#).

The gear ratio of feedback "n" is calculated as follows:

$$\text{Gear Ratio} = \text{Motor Shaft Revolutions (60E8}_{n:n_h}) / \text{Driving Shaft Revolutions (60ED}_{n:n_h})$$

## 60E9<sub>h</sub> Additional Feed Constant - Feed

### Function

In this object and in [60EE<sub>h</sub>](#), you can set a feed constant for each existing feedback.

### Object description

---

Index	60E9 <sub>h</sub>
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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	04 <sub>h</sub>

---



---

Subindex	01 <sub>h</sub> - 04 <sub>h</sub>
Name	Additional Feed Constant - Feed Feedback Interface #1 - #4
Data type	UNSIGNED32
Access	read / write



PDO mapping	RX-PDO
Allowed values	
Preset value	00000001 <sub>h</sub>

## Description

The subindices have the following function:

- 00<sub>h</sub>: Value = "n", where "n" is the number of existing feedbacks.
- n<sub>h</sub>: Subindex "n" contains the feed in user-defined units for the corresponding feedback.  
 Subindex 01<sub>h</sub> 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.

The feed constant of feedback "n" is calculated as follows:

Feed Constant = Feed (60E9<sub>h</sub>:n<sub>h</sub>) / Driving Shaft Revolutions (60EE<sub>h</sub>:n<sub>h</sub>)

## 60EBh Additional Position Encoder Resolution - Motor Revolutions

### Function

With this object and with 60E6<sub>h</sub>, the resolution of each existing feedback is calculated.

### Object description

Index	60EB <sub>h</sub>
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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	04 <sub>h</sub>
Subindex	01 <sub>h</sub> - 04 <sub>h</sub>
Name	Additional Position Encoder Resolution - Motor Revolutions Feedback Interface #1 - #4

Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001 <sub>h</sub>

## Description

The subindices have the following function:

- 00<sub>h</sub>: Value="1" to "n", where "n" is the number of existing feedbacks.
- n<sub>h</sub>:  
 Subindex n contains the number of motor revolutions of the corresponding feedback.  
 Subindex 01<sub>h</sub> 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](#).

The resolution of feedback "n" is calculated as follows:

Position Encoder Resolution = Encoder Increments (60E6<sub>h</sub>:n<sub>h</sub>) / Motor Revolutions (60EB<sub>h</sub>:n<sub>h</sub>)

## 60EDh Additional Gear Ratio - Driving Shaft Revolutions

### Function

In this object and in 60E8<sub>h</sub>, you can set the gear ratio of each existing feedback.

### Object description

Index	60ED <sub>h</sub>
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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	04 <sub>h</sub>

Subindex	01 <sub>h</sub> - 04 <sub>h</sub>
Name	Additional Gear Ratio - Driving Shaft Revolutions Feedback Interface #1 - #4
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001 <sub>h</sub>

## Description

The subindices have the following function:

- 00<sub>h</sub>: Value = "n", where "n" is the number of existing feedbacks.
- n<sub>h</sub>: Subindex "n" contains the number of revolutions of the output shaft for the corresponding feedback. Subindex 01<sub>h</sub> 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](#).

The gear ratio of feedback "n" is calculated as follows:

Gear Ratio = Motor Shaft Revolutions (60E8<sub>h</sub>:n<sub>h</sub>) / Driving Shaft Revolutions (60ED<sub>h</sub>:n<sub>h</sub>)

## 60EEh Additional Feed Constant - Driving Shaft Revolutions

### Function

In this object and in 60E9<sub>h</sub>, you can set a feed constant for each existing feedback.

### Object description

Index	60EE <sub>h</sub>
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 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	RX-PDO
Allowed values	
Preset value	04 <sub>h</sub>

Subindex	01 <sub>h</sub> - 04 <sub>h</sub>
Name	Additional Feed Constant - Driving Shaft Revolutions Feedback Interface #1 - #4
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001 <sub>h</sub>

## Description

The subindices have the following function:

- 00<sub>h</sub>: Value = "n", where "n" is the number of existing feedbacks.
- n<sub>h</sub>: Subindex "n" contains the number of revolutions of the output shaft for the corresponding feedback. Subindex 01<sub>h</sub> 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](#).

The feed constant of feedback "n" is calculated as follows:

Feed Constant = Feed (60E9<sub>h</sub>:n<sub>h</sub>) / Driving Shaft Revolutions (60EE<sub>h</sub>:n<sub>h</sub>)

## 60F2h Positioning Option Code

### Function

The object describes the positioning behavior in [Profile Position](#) mode.

### Object description

Index	60F2 <sub>h</sub>
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 <sub>h</sub>
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  $6041_h$ .



#### Note

These options cause the controller to modify object controlword  $6040_h$ .

Bit 5	Bit 4	Definition
0	0	The functionality is as described under <a href="#">Setting travel commands</a> .
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 as this is possible for the controller.
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 $607D_h:01_h$ .

Bit 7	Bit 6	Definition
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 <sub>h</sub>
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 <sub>h</sub>
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 <sub>h</sub>
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 <sub>h</sub>
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 `6041h` is set. The deviation must last longer than the time in object `203Fh`.

If the value of `60F8h` is set to "7FFFFFFF"<sub>h</sub>, slippage error monitoring is switched off.

A reaction to the slippage error can be set in object `3700h`. If a reaction is defined, an error is also entered in object `1003h`.

## 60FAh Control Effort

### Function

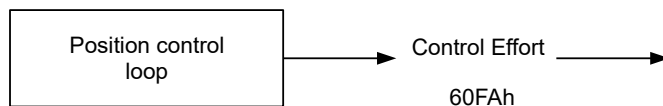
This object contains the correction speed (control variable) in user-defined units that is fed to the velocity controller by the position controller.

### Object description

Index	60FA <sub>h</sub>
Object name	Control Effort
Object Code	VARIABLE
Data type	INTEGER32
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>
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 <sub>h</sub>
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 <sub>h</sub>
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 <sub>h</sub>
Object name	Digital Inputs
Object Code	VARIABLE
Data type	UNSIGNED32
Savable	no
Access	read only
PDO mapping	TX-PDO
Allowed values	
Preset value	00000000 <sub>h</sub>
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 <sub>h</sub>
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".

**Value description**

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO mapping	no
Allowed values	
Preset value	01 <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	Digital Outputs #1
Data type	UNSIGNED32
Access	read / write
PDO mapping	RX-PDO
Allowed values	
Preset value	00000001 <sub>h</sub>

**Description**To write the outputs, the entries in object 3250<sub>h</sub>, subindex 02<sub>h</sub> to 05<sub>h</sub>, 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 <sub>h</sub>
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 <sub>h</sub>
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<sub>h</sub>.

### Object description

Index	6502 <sub>h</sub>
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 <sub>h</sub>
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 <sub>h</sub>
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 <sub>h</sub>
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	<a href="http://www.nanotec.de">http://www.nanotec.de</a>
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

Copyright (c) 2005, Swedish Institute of Computer Science

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. 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.
3. Neither the name of the Institute 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 INSTITUTE 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 INSTITUTE 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.

## 12.5 DHCP

Copyright (c) 2005, Swedish Institute of Computer Science

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. 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.
3. Neither the name of the Institute 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 INSTITUTE 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 INSTITUTE 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.

## 12.6 CMSIS DSP Software Library

Copyright (C) 2010 ARM Limited. All rights reserved.

## 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 terms.

Copyright (C) 2010, ChaN, all right reserved.

The FatFs module is a free software and there is NO WARRANTY.

No restriction on use. You can use, modify and redistribute it for personal, non-profit or commercial product UNDER YOUR RESPONSIBILITY.

Redistributions of source code must retain the above copyright notice.

## 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](http://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

Copyright (c) 2004-2005, Swedish Institute of Computer Science.

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. 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.
3. Neither the name of the Institute 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 Institute 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 Institute 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.

## 12.9 lwIP

Copyright (c) 2001-2004 Swedish Institute of Computer Science.

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. 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.
3. The name of the author may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``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 AUTHOR 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.

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  
 */
```

Copyright (c) 2017, Arm Limited. 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.
- Neither the name of ARM 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 HOLDER 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.