# Technical Manual N6

## Fieldbus: CANopen

### For use with the following variants:

N6-1-2-1, N6-1-2-1-S, N6-2-2-1, N6-2-2-1-S

Valid with firmware version FIR-v2542
and since hardware version 02

Technical Manual Version: 1.2.4

# Contents


1 Introduction....10
1.1 Product documentation....10
1.2 Version information....10
1.3 Copyright, marking and contact....11
1.4 Intended use....11
1.5 Target group and qualification....12
1.6 Warranty and disclaimer....12
1.7 Other applicable regulations....12
1.8 EU directives for product safety....12
1.9 Used icons....12
1.10 Emphasis in the text....13
1.11 Numerical values....13
1.12 Bits....14
1.13 Counting direction (arrows)....14

2 Safety and warning notices....15

3 Technical details and pin assignment....16
3.1 Environmental conditions....16
3.2 Dimensioned drawings and installation options....17
3.3 Electrical properties and technical data....17
3.4 Overtemperature protection....18
3.5 LED signaling....19
3.5.1 Power LED....19
3.6 Pin assignment....19
3.6.1 X1 – motor connection....19
3.6.2 X2 — voltage supply....20
3.6.3 X3 - STO (*Safe Torque Off*)....22
3.6.4 X4 – inputs and outputs....23
3.6.5 X5 – SSI encoder....24
3.6.6 X6 – incremental encoder/Hall sensors....25
3.6.7 X7 – CANopen IN/OUT....26
3.6.8 S1 – 120 ohm termination resistor....26
3.6.9 S2 – CANopen node-ID and baud rate....27
3.6.10 S3 – CANopen node-ID and baud rate....27
3.6.11 Micro USB....27

4 Commissioning....28
4.1 Configuration via USB....29
4.1.1 General....29
4.1.2 USB connection....29
4.1.3 Configuration file....31
4.1.4 NanoJ program....33
4.2 Configuration via CANopen....33
4.2.1 Communication settings....33
4.2.2 Establishing communication....35
4.3 Setting the motor data....35
4.4 Connecting the motor....36
4.5 Auto setup....37

4.5.1 Parameter determination ... 37
4.5.2 Execution ... 38
4.5.3 Parameter memory ... 39
4.6 Configuring the sensors ... 39
4.7 Test run ... 41

**5 General concepts ... 42**
5.1 Control modes ... 42
5.1.1 General ... 42
5.1.2 Open Loop ... 43
5.1.3 Closed Loop ... 44
5.2 CiA 402 Power State Machine ... 51
5.2.1 State machine ... 51
5.2.2 Behavior upon exiting the *Operation enabled* state ... 53
5.3 User-defined units ... 56
5.3.1 Units ... 57
5.3.2 Encoder resolution ... 58
5.3.3 Gear ratio ... 58
5.3.4 Feed constant ... 59
5.3.5 Calculation formulas for user units ... 59
5.4 Limitation of the range of motion ... 60
5.4.1 Behavior upon reaching the limit switch ... 61
5.4.2 Software limit switches ... 62
5.5 Cycle times ... 62

**6 operating modes ... 63**
6.1 Profile Position ... 63
6.1.1 Overview ... 63
6.1.2 Setting travel commands ... 64
6.1.3 Loss of accuracy for relative movements ... 68
6.1.4 Boundary conditions for a positioning move ... 69
6.1.5 Jerk-limited mode and non-jerk-limited mode ... 70
6.2 Profile Velocity ... 71
6.2.1 Description ... 71
6.2.2 Activation ... 71
6.2.3 Controlword ... 71
6.2.4 Statusword ... 71
6.2.5 Object entries ... 71
6.3 Profile Torque ... 73
6.3.1 Description ... 73
6.3.2 Activation ... 74
6.3.3 Controlword ... 74
6.3.4 Statusword ... 74
6.3.5 Object entries ... 74
6.4 Homing ... 75
6.4.1 Overview ... 75
6.4.2 Homing method ... 77
6.5 Interpolated Position Mode ... 82
6.5.1 Overview ... 82
6.5.2 Activation ... 83
6.5.3 Controlword ... 83
6.5.4 Statusword ... 83
6.5.5 Use ... 83
6.5.6 Setup ... 83
6.5.7 Operation ... 84
6.6 Cyclic Synchronous Position ... 84
6.6.1 Overview ... 84

6.6.2 Object entries....85
6.7 Cyclic Synchronous Velocity....85
6.7.1 Overview....85
6.7.2 Object entries....86
6.8 Cyclic Synchronous Torque....86
6.8.1 Overview....86
6.8.2 Object entries....87
6.9 Clock-direction mode....88
6.9.1 Description....88
6.9.2 Activation....88
6.9.3 General....88
6.9.4 Statusword....89
6.9.5 Subtypes of the clock-direction mode....89
6.10 Auto setup....90
6.10.1 Description....90
6.10.2 Activation....90
6.10.3 Controlword....90
6.10.4 Statusword....90

**7 Special functions....91**
7.1 Digital inputs and outputs....91
7.1.1 Digital inputs....91
7.1.2 Digital outputs....94
7.1.3 Pulse generator....97
7.2 Analog inputs....97
7.2.1 Object entries....97
7.2.2 Scale analog value....98
7.3 Automatic brake control....98
7.3.1 Description....98
7.3.2 Activation and connection....98
7.3.3 Brake control....98
7.3.4 Brake PWM....99
7.4 External brake chopper (ballast)....100
7.4.1 Control of the ballast resistor....100
7.4.2 Activating the brake chopper....100
7.4.3 Ballast monitoring....100
7.5 $I^2t$ Motor overload protection....101
7.5.1 Description....101
7.5.2 Object entries....102
7.5.3 Activation....102
7.5.4 Function of $I^2t$....102
7.6 Saving objects....103
7.6.1 General....103
7.6.2 Category: Communication....103
7.6.3 Category: Application....104
7.6.4 Category: User....106
7.6.5 Category: Motor....106
7.6.6 Category: Sensor....106
7.6.7 Category: CANopen....106
7.6.8 Starting the save process....107
7.6.9 Discarding the saved data....107
7.6.10 Verifying the configuration....108
7.7 Touch Probe (capture)....108

**8 CANopen....109**
8.1 General....109
8.1.1 CAN message....109

8.2 CANopen services ... 109
8.2.1 Network Management (NMT) ... 110
8.2.2 Synchronization object (SYNC) ... 111
8.2.3 Emergency Object (EMCY) ... 112
8.2.4 Service Data Object (SDO) ... 113
8.2.5 Process Data Object (PDO) ... 119
8.2.6 Boot-Up Protocol ... 124
8.2.7 Heartbeat and Nodeguarding ... 125
8.3 LSS protocol ... 126
8.3.1 General ... 126
8.3.2 LSS message ... 126
8.3.3 LSS services ... 127
8.3.4 Example ... 138

**9 Programming with NanoJ ... 139**
9.1 NanoJ program ... 139
9.2 Mapping in the NanoJ program ... 142
9.3 NanoJ functions in the NanoJ program ... 144
9.4 Restrictions and possible problems ... 145

**10 Description of the object dictionary ... 146**
10.1 Overview ... 146
10.2 Structure of the object description ... 146
10.3 Object description ... 146
10.4 Value description ... 147
10.5 Description ... 148
1000h Device Type ... 149
1001h Error Register ... 150
1003h Pre-defined Error Field ... 151
1005h COB-ID Sync ... 155
1006h Communication Cycle Period ... 156
1007h Synchronous Window Length ... 156
1008h Manufacturer Device Name ... 157
1009h Manufacturer Hardware Version ... 157
100Ah Manufacturer Software Version ... 158
100Ch Guard Time ... 158
100Dh Live Time Factor ... 159
1010h Store Parameters ... 159
1011h Restore Default Parameters ... 163
1014h COB-ID EMCY ... 166
1016h Consumer Heartbeat Time ... 167
1017h Producer Heartbeat Time ... 168
1018h Identity Object ... 168
1019h Synchronous Counter Overflow Value ... 170
1020h Verify Configuration ... 170
1029h Error Behavior ... 172
1400h Receive PDO 1 Communication Parameter ... 173
1401h Receive PDO 2 Communication Parameter ... 174
1402h Receive PDO 3 Communication Parameter ... 175
1403h Receive PDO 4 Communication Parameter ... 176
1404h Receive PDO 5 Communication Parameter ... 177
1405h Receive PDO 6 Communication Parameter ... 178
1406h Receive PDO 7 Communication Parameter ... 180
1407h Receive PDO 8 Communication Parameter ... 181
1600h Receive PDO 1 Mapping Parameter ... 182
1601h Receive PDO 2 Mapping Parameter ... 184
1602h Receive PDO 3 Mapping Parameter ... 187

1603h Receive PDO 4 Mapping Parameter....189
1604h Receive PDO 5 Mapping Parameter....191
1605h Receive PDO 6 Mapping Parameter....193
1606h Receive PDO 7 Mapping Parameter....195
1607h Receive PDO 8 Mapping Parameter....198
1800h Transmit PDO 1 Communication Parameter....200
1801h Transmit PDO 2 Communication Parameter....202
1802h Transmit PDO 3 Communication Parameter....204
1803h Transmit PDO 4 Communication Parameter....206
1804h Transmit PDO 5 Communication Parameter....208
1805h Transmit PDO 6 Communication Parameter....210
1806h Transmit PDO 7 Communication Parameter....212
1807h Transmit PDO 8 Communication Parameter....215
1A00h Transmit PDO 1 Mapping Parameter....217
1A01h Transmit PDO 2 Mapping Parameter....219
1A02h Transmit PDO 3 Mapping Parameter....222
1A03h Transmit PDO 4 Mapping Parameter....224
1A04h Transmit PDO 5 Mapping Parameter....227
1A05h Transmit PDO 6 Mapping Parameter....229
1A06h Transmit PDO 7 Mapping Parameter....232
1A07h Transmit PDO 8 Mapping Parameter....234
1C00h Sync Manager Communication Type....237
1F50h Program Data....239
1F51h Program Control....240
1F57h Program Status....241
1F80h NMT Startup....242
2005h CANopen Baudrate....242
2007h CANopen Config....243
2009h CANopen NodeID....244
2030h Pole Pair Count....245
2034h Upper Voltage Error Level [mV]....245
2035h Lower Voltage Error Level [mV]....246
2038h Brake Controller Timing....246
2039h Motor Currents....248
203Dh Torque Window....250
203Eh Torque Window Time Out....251
203Fh Max Slippage Time Out....251
2057h Clock Direction Multiplier....252
2058h Clock Direction Divider....252
205Bh Clock/Direction Or Clockwise/Counter-Clockwise Mode....253
205Ch Pulse Generator Configuration....253
2101h Fieldbus Module Availability....255
2102h Fieldbus Module Control....256
2103h Fieldbus Module Status....258
2290h PDI Control....260
2291h PDI Input....261
2292h PDI Output....262
2300h NanoJ Control....263
2301h NanoJ Status....264
2302h NanoJ Error Code....265
2303h NanoJ Metadata....266
230Eh Timer....268
230Fh Uptime Seconds....269
2310h NanoJ Input Data Selection....269
2320h NanoJ Output Data Selection....271
2330h NanoJ In/output Data Selection....272
2400h NanoJ Inputs....274
2500h NanoJ Outputs....275
2701h Customer Storage Area....275

2800h Bootloader And Reboot Settings....276
306Ch Velocity Actual Internal Value....278
30C2h Cut-off Frequency [mHz] To Filter The Velocity Demand Value Derived From Position Change....278
3202h Motor Drive Submode Select....279
3203h Feedback Selection....280
3204h Feedback Mapping....282
3205h Feedback Use....283
320Dh Moment Of Inertia....285
3211h Motor Drive Set Point....286
3219h Current Configuration....288
321Ah Current Controller Parameters....289
321Bh Velocity Controller Parameters....291
321Ch Position Controller Parameters....292
321Dh Feedforward....293
321Eh Voltage Limit....295
3220h Analog Input Digits....295
323Ah User Pin Settings....296
3241h Digital Input Position Capture....298
3242h Digital Input Routing....301
324Ah Inputs....308
3250h Digital Outputs Control....309
3252h Digital Output Routing....310
325Ah Outputs....312
3260h Pwm Output 0....313
3261h Pwm Output 1....314
3262h Pwm Output 2....315
3273h Generic SPI Hardware Configuration....316
3274h Generic SPI Mosi Data....317
3275h Generic SPI Miso Data....318
3320h Analog Input Values....319
3321h Analog Input Offsets....320
3322h Analog Input Numerators....321
3323h Analog Input Denominators....322
3380h Feedback Sensorless....323
3390h Feedback Hall....324
33A0h Feedback Incremental A/B/I 1....328
33B0h Feedback SSI 1....329
3700h Deviation Error Option Code....334
3701h Limit Switch Error Option Code....335
4014h Operating Conditions....335
4015h Special Drive Modes....337
4021h Ballast Configuration....339
4040h Drive Serial Number....341
4041h Device Id....341
4042h Bootloader Infos....342
6007h Abort Connection Option Code....343
603Fh Error Code....344
6040h Controlword....345
6041h Statusword....346
605Ah Quick Stop Option Code....347
605Bh Shutdown Option Code....348
605Ch Disable Operation Option Code....349
605Dh Halt Option Code....349
605Eh Fault Reaction Option Code....350
6060h Modes Of Operation....351
6061h Modes Of Operation Display....351
6062h Position Demand Value....352
6063h Position Actual Internal Value....352
6064h Position Actual Value....353

6065h Following Error Window....353
6066h Following Error Time Out....354
6067h Position Window....354
6068h Position Window Time....355
606Bh Velocity Demand Value....356
606Ch Velocity Actual Value....356
606Dh Velocity Window....356
606Eh Velocity Window Time....357
606Fh Velocity Threshold....358
6070h Velocity Threshold Time....358
6071h Target Torque....359
6072h Max Torque....359
6073h Max Current....360
6074h Torque Demand....361
6075h Motor Rated Current....361
6077h Torque Actual Value....362
607Ah Target Position....362
607Bh Position Range Limit....363
607Ch Home Offset....364
607Dh Software Position Limit....364
607Eh Polarity....365
607Fh Max Profile Velocity....366
6080h Max Motor Speed....367
6081h Profile Velocity....367
6082h End Velocity....368
6083h Profile Acceleration....368
6084h Profile Deceleration....369
6085h Quick Stop Deceleration....369
6086h Motion Profile Type....369
6087h Torque Slope....370
6091h Gear Ratio....370
6092h Feed Constant....372
6096h Velocity Factor....373
6097h Acceleration Factor....374
6098h Homing Method....375
6099h Homing Speed....375
609Ah Homing Acceleration....376
60A2h Jerk Factor....377
60A4h Profile Jerk....378
60A8h SI Unit Position....380
60A9h SI Unit Velocity....380
60B0h Position Offset....381
60B1h Velocity Offset....381
60B2h Torque Offset....382
60C1h Interpolation Data Record....382
60C2h Interpolation Time Period....383
60C4h Interpolation Data Configuration....384
60C5h Max Acceleration....387
60C6h Max Deceleration....387
60E4h Additional Position Actual Value....387
60E5h Additional Velocity Actual Value....389
60E6h Additional Position Encoder Resolution - Encoder Increments....390
60EBh Additional Position Encoder Resolution - Motor Revolutions....391
60F1h Additional Encoder Inversion....392
60F2h Position Option Code....393
60F4h Following Error Actual Value....395
60F8h Max Slippage....395
60FAh Control Effort....396
60FCh Position Demand Internal Value....396

60FDh Digital Inputs..... 397
60FEh Digital Outputs..... 398
60FFh Target Velocity..... 399
6502h Supported Drive Modes..... 399
6503h Drive Catalogue Number..... 400
6505h Http Drive Catalogue Address..... 401

**11 Copyrights..... 402**
11.1 Introduction..... 402
11.2 AES..... 402
11.3 MD5..... 402
11.4 DHCP..... 403
11.5 CMSIS DSP Software Library..... 403
11.6 Protothreads..... 403
11.7 lwIP..... 404
11.8 littlefs..... 404

# 1 Introduction

The *N6* is a controller for the *open-loop* or *closed loop* operation of stepper motors and the *closed loop* operation of BLDC motors.

This manual describes the functions of the controller and the available operating modes. It also shows how you can address and program the controller via the communication interface.

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

## 1.1 Product documentation

All product documentation is available on the respective product page at us.nanotec.com.

| Document | Content |
|---|---|
| Technical manual (this document) | Installation, commissioning, parameterization/programming, description of the functions of the motor controller |
| Function description STO | Description of the integrated *STO* safety function for the product variants that include this function (N6-...-**S**) |

You can read out the hardware version from object $1009_h$. You can find details in the technical manual of your variant.

## 1.2 Version information

| Manual version | Date | Changes | Firmware version |
|---|---|---|---|
| 1.0.0 | 01/2025 | Edition | FIR-v2451 |
| 1.1.0 | 04/2025 | Corrections<br>USB mass storage replaced with MTP device, PowerShell and Linux scripts can be found in chapter Configuration via USB, | FIR-v2508 |
| 1.1.1 | 06/2025 | Pin assignment corrected for CANopen: CAN_L and CAN_H were reversed. | FIR-v2508 |
| 1.2.0 | 09/2025 | Pins for clock and direction in clock / direction mode changed in Pin assignment.<br>Small modifications and additions in chapter Configuration via USB.<br>Correction: The hardware revision is "0".<br>Small modifications and additions in chapter External brake chopper (ballast).<br>New chapter Touch Probe (capture). | FIR-v2529 |
| 1.2.1 | 12/2025 | New Firmware version: minor adjustments and corrections.<br>Homing: Method 37 replaces 35. Notice regarding methods −17 and −18 added. | FIR-v2542 |
| 1.2.2 | 05/2026 | Correction: Voltaga of the brake output corresponds to the UB_Logic and is not switchable.<br>Corrections / additions in the description and example for the SSI encoder setup in 33B0h Feedback SSI 1. | FIR-v2542 |

| Manual version | Date | Changes | Firmware version |
|---|---|---|---|
| | | Correction: The bootloader does not send a boot-up message, Subindex $01_h$ in 2007h CANopen Config has no function. | |
| 1.2.4 | 07/2026 | Notes on electrical safety updated after UL certification:<br>■ Use in Pollution Degree 2 and Overvoltage Category II environments.<br>■ The device may only be used in circuits with a limited short-circuit current (≤ 5 kA) and the required fuse.<br>■ For the supply use only cables with copper conductors that are rated for a temperature range of at least 60° / 75°.<br>■ No Thermal Memory retention. | FIR-v2542 |

## 1.3 Copyright, marking and contact

© 2013 – 2026 Nanotec Electronic GmbH & Co. KG. All rights reserved.

CE

Nanotec Electronic GmbH & Co. KG

Kapellenstraße 6

85622 Feldkirchen (Germany)

Germany

Phone: +49 89 900 686-0

Fax: +49 (89) 900 686-50

us.nanotec.com

Microsoft® Windows® 98/NT/ME/2000/XP/7/10/11 and PowerShell® are registered trademarks of the Microsoft Corporation.

## 1.4 Intended use

The *N6* serves to control stepper motors and BLDC motors and is used as a component in drive systems in a wide range of industrial applications.

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

With the exception of the  product variants, which support the *STO* safety function (Safe Torque Off)), this Nanotec product may under no circumstances be integrated as a safety component in a product or system.

All products containing a component manufactured by Nanotec must, upon delivery to the end user, be provided with corresponding warning notices including instructions for safe use and safe operation. All warning notices provided by Nanotec must be passed on directly to the end user.

## 1.5 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 controller,
- are familiar with and understand the content of this technical manual,
- know the applicable regulations.

## 1.6 Warranty and disclaimer

Nanotec shall not be liable for damage and malfunctions attributable to installation errors, failure to observe this document or improper repair. The plant engineer, operating company and user shall be responsible for the selection, operation and use of our products. Nanotec shall not take responsibility for integration of the product in the end system. The general terms and conditions listed at www.nanotec.de shall apply.
**Comment:** Modifications/changes to the product as well as opening the product are prohibited.

## 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 EU directives for product safety

The following EU directives were observed:

- RoHS directive (2011/65/EU, 2015/863/EU)
- EMC directive (2014/30/EU)

## 1.9 Used icons

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

**CAUTION!**

**The CAUTION notice indicates a possibly dangerous situation.**

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

► Describes how you can avoid the dangerous situation.

**NOTICE**

**Indicates a possible incorrect operation of the product.**

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

► Describes how you can avoid the incorrect operation.

**TIP**

Shows a tip for the application or task.

**STO-specific notices**

The following points apply only to the products with the STO (*Safe Torque Off*) safety function.

**CAUTION**

**The CAUTION note with the STO icon indicates a possibly dangerous situation that can arise from the loss of the safety function.**

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

► Describes how you can avoid the dangerous situation.

**Note**

**The note with the STO icon indicates a possible incorrect operation of the product with STO function.**

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

► Describes how you can avoid the incorrect operation.

## 1.10 Emphasis in the text

The following conventions are used in the document:

Underlined text indicates cross references and hyperlinks:

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

Text set in *italics* marks named objects:

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

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

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

A text in "quotation marks" marks user input:

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

## 1.11 Numerical values

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

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

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

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

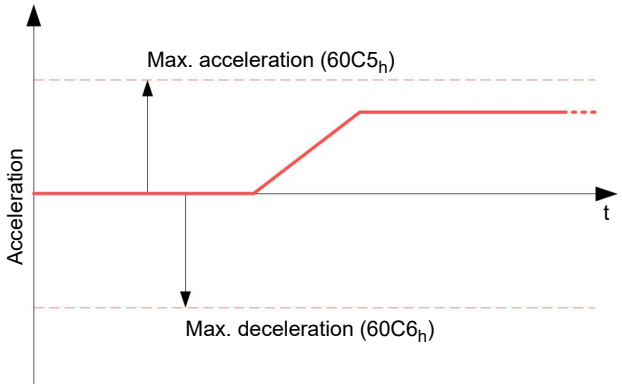
## 1.12 Bits

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

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

## 1.13 Counting direction (arrows)

In figures, the counting direction is always in the direction of an arrow. Objects $60C5_h$ and $60C6_h$ depicted as examples in the following figure are both specified as positive.



# 2 Safety and warning notices

**CAUTION**

**The loss of the safety function may lead to injuries!**

► Observe the conditions for use for the STO function and for the product, as described in the document *Function description STO*

► Do not bypass the STO function.

► Do not use the STO function until all protective measures are set up and the validation of the machine/system is concluded after commissioning.

**CAUTION**

**The loss of the safety function may lead to injuries!**

If the supply voltage exceeds 60 V, the STO function cannot be ensured.

► For the power supply of the product variants with STO function, use only PELV circuits acc. to IEC 60204-1 that permit an absolute maximum voltage of 60 V (even in case of an error).

► If batteries are used, observe a maximum rated voltage of 48 V DC (with a tolerance of +20%), compliant with DVC A or B as defined in IEC 61800-5-1.

**CAUTION**

**The loss of the safety function may lead to injuries!**

Excessively high temperature can damage the electronics of the controller and of the STO. This may result in loss of the STO function.

► Observe the approved environmental conditions and the permissible power values.

► Install the product so that clean and dry cooling air is available for cooling and that this air can circulate around the product.

**NOTICE**

**Damage to the controller!**

Changing the wiring during operation may damage the controller.

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

**NOTICE**

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

Voltage peaks during operation may damage the controller.

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

**NOTICE**

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

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

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

# 3 Technical details and pin assignment

## 3.1 Environmental conditions

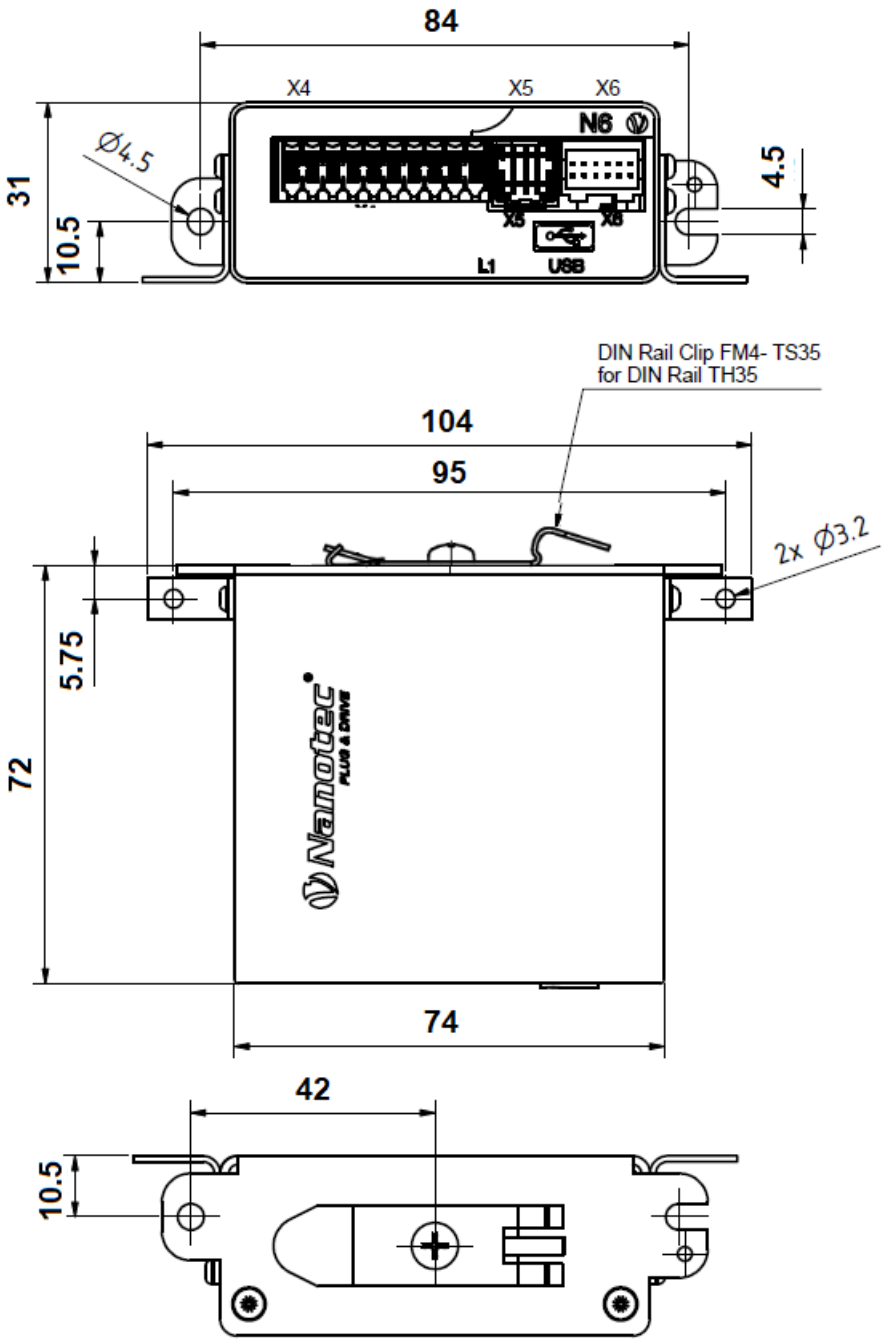
**NOTICE**

For use in Pollution Degree 2 and Overvoltage Category II environments only.

- Only non-conductive contaminants are permitted (e.g., dust). If contamination is more severe, install the device in a suitable enclosure or control cabinet.
- Direct feed-in from the building's electrical system and higher overvoltage categories are not allowed.

| Environmental condition | Value |
|---|---|
| Protection class | IP20 |
| Pollution degree | 2 |
| Ambient temperature (operation) | -10 … +40°C |
| Ambient temperature (storage and transport) | -20 … +85°C |
| Relative humidity (operation), non-condensing | 0 … 95 % |
| Relative humidity (storage and transport), non-condensing | 0 … 90 % |
| Absolute humidity (storage and transport), non-condensing | 30 g/m$^3$ |
| Max. altitude of site above *sea level* | 2000 m (drop in performance above 1000 m: -1%/100 m) |
| Max. altitude of site above *sea level* (storage and transport) | 3000 m |

## 3.2 Dimensioned drawings and installation options




You can secure the controller by its side tabs to a flat mounting surface using screws or mount it on a TH35 DIN rail in your switch cabinet using the supplied DIN rail clip.

## 3.3 Electrical properties and technical data

| Property | Description / value |
|---|---|
| Operating voltage | 12 V -5%…57.6 V DC |
| Rated current | 6 $A_{rms}$ |
| Peak current | N6-1-… (*low current*): 6 $A_{rms}$ |

| Property | Description / value |
|---|---|
| | N6-2-… (*high current*): 18 $A_{rms}$ for 5 seconds |
| Commutation | Stepper motor – open loop, stepper motor – closed loop with encoder, BLDC motor – closed loop with Hall sensor, and BLDC motor – closed loop with encoder |
| Operating modes | *Profile Position Mode*, *Profile Velocity Mode*, *Profile Torque Mode*, *Homing Mode*, *Interpolated Position Mode*, *Cyclic Sync Position Mode*, *Cyclic Sync Velocity Mode*, *Cyclic Synchronous Torque Mode*, *Clock-Direction Mode* |
| Set value setting / programming | *CANopen*, *Clock-direction*, *analog*, *NanoJ program* |
| Interfaces | CANopen, USB |
| Inputs | ■ 6 inputs, 5 V/24 V (=UB_Logic), switchable by means of software, factory setting: 5 V<br>■ 2 analog inputs 0 to +24 V, 12-bit resolution |
| Outputs | 3 outputs, 5 V/24 V (=UB_Logic), switchable by means of software, 100 mA |
| Brake connection | 1 PWM output, max. 1.5 A, 20 kHz |
| Sensor inputs | 1 incremental encoder (5 V, differential), 3 Hall sensors (5 V), 1 SSI encoder (10 V) |
| Protection circuit | Overvoltage and undervoltage protection<br>Overtemperature protection (> 75° Celsius on the power board)<br>Polarity reversal protection |

**NOTICE**

Suitable For Use On A Circuit Capable Of Delivering Not More Than 5000 rms Symmetrical Amperes, 57.6 Volts Maximum. When Protected by CC Class Fuses, rated max. 10 A, min. 150 Vdc.

- The device may only be used in circuits with a limited short-circuit current (≤ 5 kA)
- and the required fuse.

## 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 $1001_h$ and $1003_h$). After cooling down and confirming the error (see table for the controlword, "Fault reset"), the controller again functions normally.

**NOTICE**

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

Make certain that heat dissipation through the mounting surface and passive cooling or active ventilation are possible to ensure that the maximum ambient temperature is not exceeded.

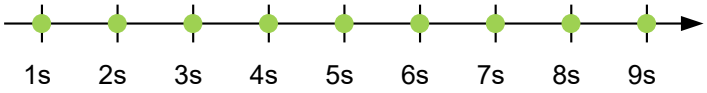
## 3.5 LED signaling

### 3.5.1 Power LED

The power LED indicates the current status.

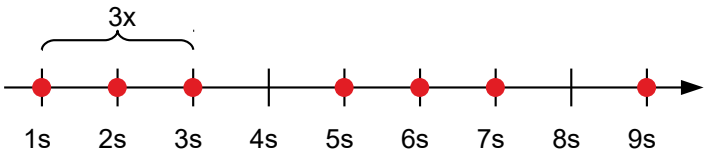
#### 3.5.1.1 Normal operation

In normal operation, the green power LED L1 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 / STO active |
| 5 | Controller |
| 6 | Watchdog-Reset |

**NOTICE**

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

**TIP**

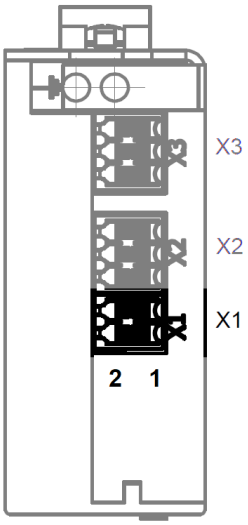
You can switch off the operating LEDs via $3250_h:01_h$.

## 3.6 Pin assignment

### 3.6.1 X1 – motor connection

- Type: Degson 15EDGRHD-3.5-04P-04
- Mating connector (included in the scope of delivery): Degson 15EDGKNHBM-3.5-04P (or equivalent)
- Conductor cross section, min.: 1 $mm^2$ (AWG 17)

Pins 1 and 2 are marked below.




| Pin | Function (stepper motor) | Function (BLDC motor) | Note |
|---|---|---|---|
| 1 | A | U | |
| 2 | A\ | V | |
| 3 | B | W | |
| 4 | B\ | Not used | |

**NOTICE**

Motor cables are to be routed through ferrites (74271222 from Würth or equivalent).

### 3.6.2 X2 — voltage supply

- Type: Degson 15EDGRHD-3.5-06P
- Mating connector (included in the scope of delivery): Degson 15EDGKNHB-3.5-06P (or equivalent)

**NOTICE**

Use only cables with copper conductors that are rated for a temperature range of at least 60° / 75°.

- □ Conductor cross section, min.:
  - 1 $mm^2$ (AWG 17) for main supply UB
  - 0.8 $mm^2$ (AWG 18) for UB_Logic

#### 3.6.2.1 Voltage source

The operating or supply voltage supplies a battery, a transformer with rectification and filtering, or a switching power supply.

**CAUTION**

**The loss of the safety function may lead to injuries!**

If the supply voltage exceeds 60 V, the STO function cannot be ensured.

► For the power supply of the product variants with STO function, use only PELV circuits acc. to IEC 60204-1 that permit an absolute maximum voltage of 60 V (even in case of an error).

► If batteries are used, observe a maximum rated voltage of 48 V DC (with a tolerance of +20%), compliant with DVC A or B as defined in IEC 61800-5-1.

**NOTICE**

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

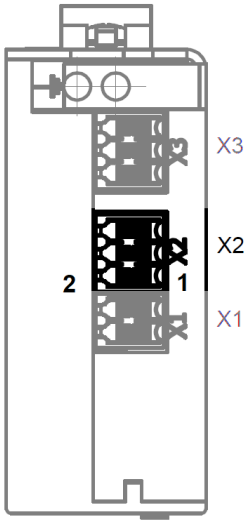
► It is recommended that an EMI filter (810911010 from Würth or equivalent) be inserted in the DC supply line(s) as close as possible to the controller/motor.

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

► Motor cables are to be routed through ferrites (74271222 from Würth or equivalent).

### 3.6.2.2 Connections

Pins 1 and 2 are marked below.




| Pin | Function | Note |
|---|---|---|
| 1 | GND_L | GND for the logic supply UB_Logic |
| 2 | +UB_Logic | Input voltage for the logic supply 12 V - 30 V DC, |
| 3 | Ballast- | Connection for optional external brake chopper, max. 6.5 A for 1 s, afterward switch off for min. 1 s |
| 4 | Ballast+ | Connection for optional external brake chopper, max. 6.5 A for 1 s, afterward switch off for min. 1 s |
| 5 | GND_P | GND for the main supply UB |
| 6 | +UB | Input voltage for the main supply (power part), 12 V - 57.6 V DC |

**NOTICE**

The logic supply powers the electronics, the sensors and the communication interface.

The windings of the motor are not supplied by the logic supply.

### 3.6.2.3 Permissible operating voltage

The maximum operating voltage is 57.6 V DC. If the input voltage of the controller exceeds the threshold value set in $2034_h$, the motor is switched off and an error triggered. Above the response threshold set in $4021_h$:$02_h$, the optional external ballast resistor is also switched on.

The minimum operating voltage is 12 V DC. If the input voltage of the controller falls below 10 V, the motor is switched off and an error triggered.

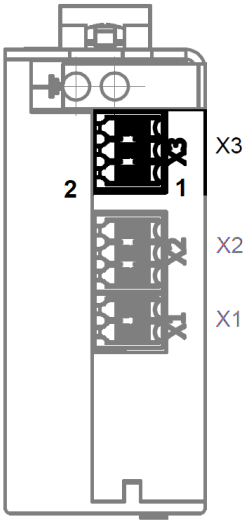
A charging capacitor of at least 4700 µF / 50 V (approx. 1000 µF per ampere rated current) must be connected to the supply voltage to avoid exceeding the permissible operating voltage (e.g., during braking). It is recommended that an EMI filter (810911010 from Würth or equivalent) be inserted in the DC supply line(s) as close as possible to the controller/motor.

## 3.6.3 X3 - STO (*Safe Torque Off*)

Connection for the STO signals. This connection and the corresponding integrated *STO* safety function are available only with the STO variants of the product (N6-x-x-x-**S**).

- Type: Degson 15EDGRHD-3.5-06P-04
- Mating connector (included in the scope of delivery): Degson 15EDGKNHBM-3.5 (or equivalent)

Pins 1 and 2 are marked below.




**CAUTION**

STO

**The loss of the safety function due to electromagnetic interference may lead to injuries!**

External interference may affect and result in the loss of the safety function.

► Observe the maximum permissible cable length of 30 m for all STO signals. Longer cables reduce the interference immunity (EMC) and require additional interference-suppression and protection measures.

► Use shielded cables for the STO signals.

► Lay supply, signal and control cables physically separate from one another.

**CAUTION**

**The loss of the safety function due to incorrect wiring may lead to injuries!**

Incorrect wiring or the use of unsuitable external components may lead to the loss of the safety function. This could result in injuries.

► Only use components that correspond to the safety category of the application.

► Check the electrical installation (wiring, pin assignment) and validate the STO function prior to the initial commissioning and after every intervention in the wiring and each time components/equipment are replaced.

► Do not bypass the STO function. If the wiring for the initial commissioning does not correspond to the required safety category of the application, remove it immediately after the initial commissioning.

| Pin | Function | Note |
|---|---|---|
| 1 | STOA_IN | STO channel A, PWM signal suppression<br>Rated voltage 24 V DC, max. 60 V DC |
| 2 | STOA_RTN | GND connection for STO channel A |
| 3 | STO_FBH | STO feedback, high |
| 4 | STO_FBL | STO feedback, low |
| 5 | STOB_IN | STO channel B, gate driver switch<br>Rated voltage 24 V DC, max. 60 V DC |
| 6 | STOB_RTN | GND connection for STO channel B |

**STO function**

If the voltage at one of the two STO inputs (channel A and B) drops below 5 V, the STO function is triggered and the motor can no longer generate torque.

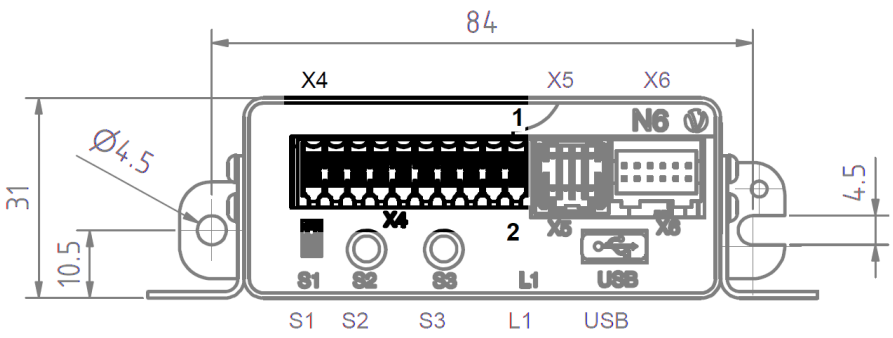
The feedback output is used for monitoring the status of the STO function. If the STO function is active, the contact between *STO_FBH* and *STO_FBL* closes.

A detailed description of the STO function can be found in the document *Function description STO*.

## 3.6.4 X4 – inputs and outputs

- Type: Degson 15EDGRHD-3.5-20P-14
- Mating connector (included in the scope of delivery): Degson 15EDGKNHB-3.5-20P (or equivalent)

Pins 1 and 2 are marked below.




| Pin | Function | Note |
|---|---|---|
| 1 | +10 V | +10 V DC, output voltage, max. 350 mA |

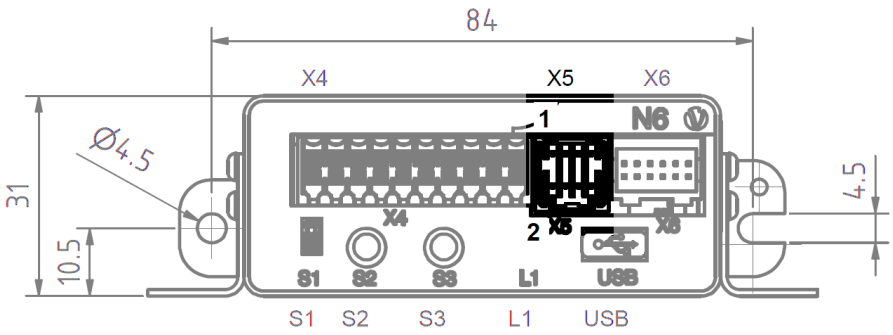
| Pin | Function | Note |
|---|---|---|
| 2 | GNDD | GND for digital inputs/outputs |
| 3 | +5 V | +5 V DC, output voltage, max. 350 mA |
| 4 | GNDD | GND for digital inputs/outputs |
| 5 | Output 1 | 5 V / 24 V digital output (UB_Logic), switchable by means of software with object 323A$_h$, max. 100 mA |
| 6 | Output 2 | 5 V / 24 V digital output (UB_Logic), switchable by means of software with object 323A$_h$, max. 100 mA |
| 7 | Output 3 | 5 V / 24 V digital output (UB_Logic), switchable by means of software with object 323A$_h$, max. 100 mA |
| 8 | GNDD | GND for digital inputs/outputs |
| 9 | Input 1 | 5 V / 24 V digital input, switchable by means of software with object 323A$_h$, direction input in clock-direction mode |
| 10 | Input 2 | 5 V / 24 V digital input, switchable by means of software with object 323A$_h$, clock input in clock-direction mode |
| 11 | Input 3 | 5 V / 24 V digital input, switchable by means of software with object 323A$_h$, max. 1 MHz |
| 12 | Input 4 | 5 V / 24 V digital input, switchable by means of software with object 323A$_h$, max. 1 MHz |
| 13 | Input 5 | 5 V to 24 V digital input, switchable by means of software with object 323A$_h$ |
| 14 | Input 6 | 5 V to 24 V digital input, switchable by means of software with object 323A$_h$ |
| 15 | GNDA | GND for analog input |
| 16 | Analog input 1 | 0 V…+24 V, 12-bit resolution |
| 17 | GNDA | GND for analog input |
| 18 | Analog input 2 | 0 V…+24 V, 12-bit resolution |
| 19 | Brake - | GND for brake |
| 20 | Brake + | PWM-controlled output, max. UB_Logic / 1500 mA, up to 20 KHz |

### 3.6.5 X5 – SSI encoder

Connection for SSI encoder

- Type: Molex MicroClasp 55959-0830
- Mating connector (not included in scope of delivery): Molex 51353-0800 (or equivalent)

Pins 1 and 2 are marked below.




| Pin | Function | Note |
|---|---|---|
| 1 | GND | |
| 2 | SHIELD | Connection for the shielding |

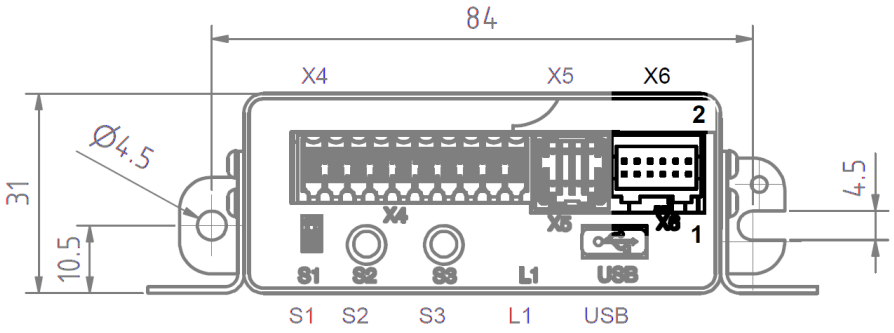
| Pin | Function | Note |
|---|---|---|
| 3 | n.c. | reserved |
| 4 | DATA B | |
| 5 | DATA A | |
| 6 | CLCK B | up to 3.8 MHz |
| 7 | CLCK A | up to 3.8 MHz |
| 8 | Vcc | +10…10.5 V DC±5%, output and supply voltage for SSI encoder, max. 350 mA |

The clock signal and the binary-coded data are transferred as a differential signal (RS-422). A 120-ohm terminating resistor between *DATA A* and *DATA B* is provided internally. You must configure the encoder parameters in the object 33B0h Feedback SSI 1 according to the data sheet so that the encoder can be detected.

### 3.6.6 X6 – incremental encoder/Hall sensors

- Type: JST S12B-PADSS-1
- Mating connector (not included in scope of delivery):
  - Housing: JST PADP-12V-1-S (or equivalent)
  - Socket contacts: JST SPH-001T-P0.5L (or equivalent)
- Suitable Nanotec cables (not included in the scope of delivery):
  - ZK-PADP-12-500-S
  - ZK-M12-8-2M-2-PADP
  - ZK-M12-12-2M-2-PADP
  - ZK-NTO3-10-500-PADP / ZK-NTO3-10-1000-PADP
  - ZK-NOE-10-500-S-PADP
  - ZK-WEDL-500-S-PADP

Pin 1 and pin 2 are marked in the figure.



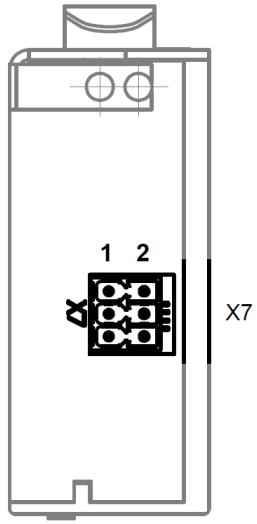

| Pin | Function | Note |
|---|---|---|
| 1 | GND | |
| 2 | Vcc | 5 V DC, output and supply voltage for encoder / Hall sensor; max. 350 mA |
| 3 | A | 5 V signal, max. 1 MHz |
| 4 | B | 5 V signal, max. 1 MHz |
| 5 | A\ | 5 V signal, max. 1 MHz |
| 6 | B\ | 5 V signal, max. 1 MHz |
| 7 | I | 5 V signal, max. 1 MHz |
| 8 | I\ | 5 V signal, max. 1 MHz |
| 9 | Hall 1 | 5 V signal |

| Pin | Function | Note |
|---|---|---|
| 10 | Hall 2 | 5 V signal |
| 11 | Hall 3 | 5 V signal |
| 12 | Shielding | Shielding |

### 3.6.7 X7 – CANopen IN/OUT

- Type: Amphenol NR061A510000H
- Mating connector (included in the scope of delivery): NL06105xxxxxH (or equivalent)

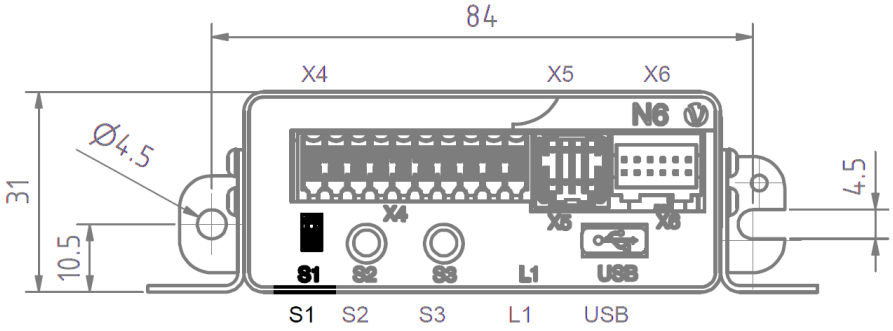
Pins 1 and 2 are marked below.




| Pin | Function | Note |
|---|---|---|
| 1 | GND | |
| 2 | GND | |
| 3 | CAN_L | CAN-Low |
| 4 | CAN_L | CAN-Low |
| 5 | CAN_H | CAN-High |
| 6 | CAN_H | CAN-High |

### 3.6.8 S1 – 120 ohm termination resistor

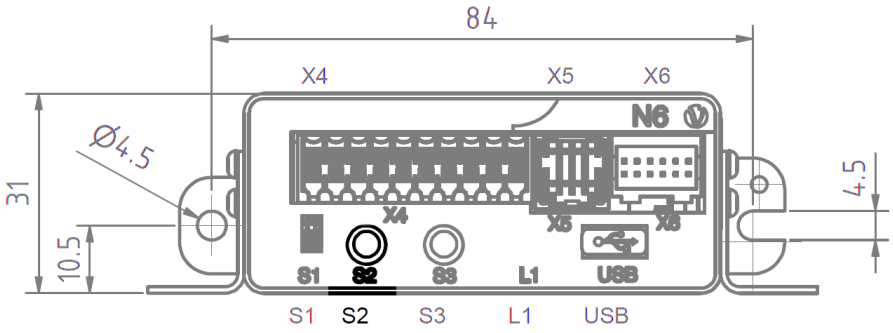
These DIP switches switch the termination of 120 Ω between CAN_H and CAN_L of the CAN bus on (switch up) or off.



### 3.6.9 S2 – CANopen node-ID and baud rate

Hex coding switch for setting the CANopen node-ID and baud rate. See chapter Setting node-ID and baud rate.

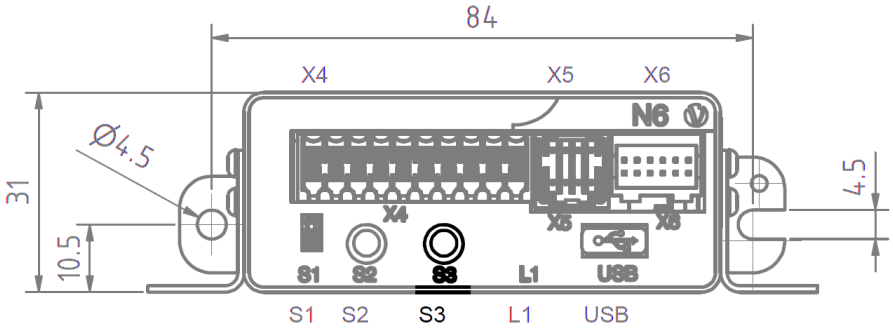
The value of this switch is multiplied by 16 and added to the value of switch S3; this switch thereby sets the 16s place.




### 3.6.10 S3 – CANopen node-ID and baud rate

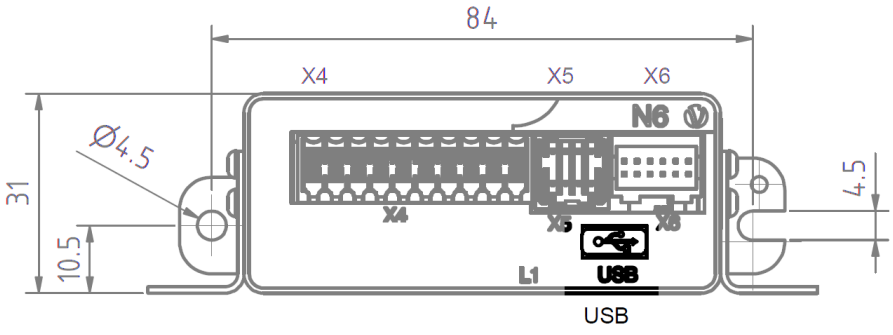
Hex coding switch for setting the CANopen node-ID and baud rate. See chapter Setting node-ID and baud rate.

The value of this switch is added to the value of switch S2; this switch thereby sets the 1s place.




### 3.6.11 Micro USB

Micro USB connection for configuration/parameterization.



# 4 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. You can configure the controller via USB or via the CANopen bus.

The *Plug & Drive Studio 3* software offers you an option for performing the configuration and adapting the controller to the connected motor. You can find further information in document *Plug & Drive Studio: User Manual* at us.nanotec.com.

**CAUTION**

**The loss of the safety function due to incorrect wiring may lead to injuries!**

Incorrect wiring or the use of unsuitable external components may lead to the loss of the safety function. This could result in injuries.

► Only use components that correspond to the safety category of the application.

► Check the electrical installation (wiring, pin assignment) and validate the STO function prior to the initial commissioning and after every intervention in the wiring and each time components/equipment are replaced.

► Do not bypass the STO function. If the wiring for the initial commissioning does not correspond to the required safety category of the application, remove it immediately after the initial commissioning.

**CAUTION**

**The loss of the safety function due to electromagnetic interference may lead to injuries!**

External interference may affect and result in the loss of the safety function.

► Observe the maximum permissible cable length of 30 m for all STO signals. Longer cables reduce the interference immunity (EMC) and require additional interference-suppression and protection measures.

► Use shielded cables for the STO signals.

► Lay supply, signal and control cables physically separate from one another.

**CAUTION**

**Injuries resulting from unexpected and unintended movements of the drive during installation and maintenance work!**

If the STO function is missing or not activated, the drive may start unexpectedly.

► Switch off the power supply before beginning the work.

► Secure all power supplies against accidentally being switched back on again.

**NOTICE**

No Thermal Memory retention: The device does not save any temperature values and can be restarted immediately after power-off, without cooling down first and regardless of the motor temperature. When the power board reaches the maximal permissible temperature, the power part of the controller switches off.

**NOTICE**

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

Suitable measures may be:

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

## 4.1 Configuration via USB

### 4.1.1 General

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

**Configuration file**

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

**NanoJ program**

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

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

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

### 4.1.2 USB connection

**NOTICE**

**Damage to the product and/or external hardware caused by differences in potential at the USB.**

Connecting the USB cable while the electronics are being supplied (hot plugging) may result in damage.

- ► Connect USB before switching on the supply voltage.
- ► If possible, equalize differences in potential between PC and product or use USB isolator.
- ► First connect the USB cable to the product, then to the PC.

If the controller is connected to a PC via a USB cable, an MTP device with the exact *device name* of the product variant is created in the Windows file explorer that contains a data carrier.

Up to three files are displayed, the configuration file (`cfg*.txt`), the firmware (`*.fw`) and the *NanoJ program* (`nanoj*.usr`), which is stored internally immediately after restarting the controller and is no longer displayed.

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

**NOTICE**

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

  1. `cfg*.txt`
  2. `nanoj*.usr`
  3. `reset.txt`

  Other files are not accepted by the controller.

### PowerShell scripts

In the download area of the product website at nanotec.com, you can find file *mtp.ps1* with examples for the Windows PowerShell console. Based on the exact *device name* (see 1008$_h$), you can use this to:

- List all stored files:
  `.\mtp.ps1 -Action list -DeviceName N6-1-1-1`
- Copy a file from the current script folder to the device:
  `.\mtp.ps1 -Action copy -DeviceName N6-1-1-1 -FileName cfg_Example.txt`
- Delete a file:
  `.\mtp.ps1 -Action delete -DeviceName N6-1-1-1 -FileName cfg_Example.txt`

### MTP in Linux

Nanotec recommends the following: MTP Tools. If the *GVfs* often included in Linux distributions is already installed, uninstall it first:

```
sudo apt remove gvfs
```

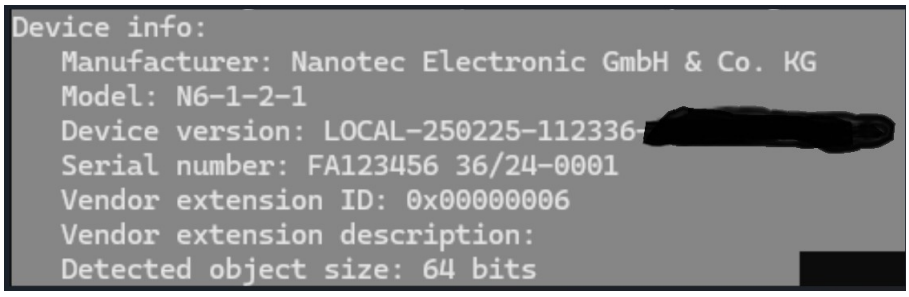
Then install *MPT Tools* and *libmtp* and restart:

```
sudo apt-get install mtp-tools
sudo apt-get install libmtp-runtime
reboot
```

After connecting the USB, check whether the device is detected:

```
mtp-detect
```

The device responds to this with its properties (manufacturer, product name, serial number, etc.), similar to the following figure.




You can now:

- List all stored files with their ID:
  `mtp-files`
- Copy a file from the current script folder to the device (the name must be identical):
  `mtp-sendfile cfg_Out.txt cfg_Out.txt`
- Copy a file from the device (file ID listed as with *mtp-files*):
  `mtp-getfile 1072922872 cfg_Or.txt`

- Delete a file:
  `mtp-delfile -f cfg_Example.txt`

### 4.1.3 Configuration file

#### 4.1.3.1 General

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

**NOTICE**

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

#### 4.1.3.2 Reading and writing the file

How to access the file:

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

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

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

**TIP**

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

#### 4.1.3.3 Structure of the configuration file

##### Comments

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

**Example**

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

### Assignments

**NOTICE**

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

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

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

**<Index>**

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

**<Subindex>**

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

**<Value>**

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

You can also set individual bits:

**Set bit**

```
3202:00.01=1
```

**Reset bit**

```
3202:00.01=0
```

**Bitwise OR**

```
3202:00|=0x01
```

**Bitwise AND**

```
3202:00&=0x01
```

#### Example

Set object $6075_h$:00 (rated current) to the value "600" (mA):

```
6075:00=600
```

Set object $3202_h$:00 to the value "1" (activate *closed-loop* mode):

```
3202:00=1
```

or only set bit 0

```
3202:00.00=1
```

**NOTICE**

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

### 4.1.4 NanoJ program

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

1. Write and compile your program as described in chapter Programming with NanoJ.
2. Connect the voltage supply to the controller and switch on the voltage supply.
3. Connect the controller to your PC using the USB cable.
4. After the PC has detected the device as a removable storage device, open an Explorer window and delete file `nanoj*.usr` on the controller.
5. Navigate in the Explorer to the directory with your program. The compiled file has the same name as the source code file, only with file extension `.usr`. Rename this file `nanoj*.usr`.
6. Copy file `nanoj*.usr` to the controller.
7. Disconnect the voltage supply from the controller for approx. 1 second until the power LEDs stop flashing.
8. Reconnect the voltage supply. When the controller now starts, the new *NanoJ program* is read in and started.

**TIP**

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

**NOTICE**

- The *NanoJ program* on the controller must have file name `nanoj*.usr`.

## 4.2 Configuration via CANopen

### 4.2.1 Communication settings

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

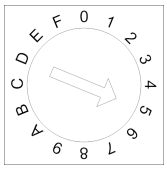
The controller is configured per default for node-ID 1 and a baud rate of 1 MBd.

#### 4.2.1.1 Setting node-ID and baud rate

Node-ID and baud rate are dependent on the position of the S2 and S3 *rotary switches* and, if applicable, on objects 2005h CANopen Baudrate, and 2009h CANopen NodeID.

### Rotary switch

The *N6* is equipped with two hex coding switches – similar to that shown in the following figure.




You can set the source for the CANopen node-ID and the baud rate using the combination of numbers from both rotary switches.

The following applies here: The combination of numbers is formed using rotary switches S2 and S3, where S2 represents the higher-value nibble and S3 the lower-value nibble.

**Example**

Switch S2 is set to the value "$0_h$", switch S3 to the value "$F_h$"; the result is the value "$0F_h$"="$15_d$".

Switch S2 is set to the value "$A_h$", switch S3 to the value "$1_h$"; the result is the value "$A1_h$"="$161_d$".

### General

- If the rotary switch is set to the value "1", the node-ID is set to "1" and the baud rate is permanently set to 1 MBd. In the event of problems with the configuration, communication can thereby always be established with the controller and any errors reversed.
- The changes in objects $2005_h$ and $2009_h$ must be stored by writing value "$65766173_h$" in object $1010_h$:$0A_h$.
- The changes are not accepted until either
  - the voltage supply is briefly disconnected or
  - the controller is restarted via $2800_h$ or
  - the CANopen message "RESET COMM" (NMT) is sent to the motor.
- The load sequence for objects $2005_h$ and $2009_h$ is as follows (each successive value overwrites the previous):
  1. The default value is loaded.
  2. A stored value – if present – is loaded.
  3. The configuration file is used if the controller has one.
  4. The settings of the rotary switches are taken over.

### Node-ID and baud rate

The following table shows the possibilities that arise for the rotary switches.

| Number combinations of the rotary switches | | Node-ID | Baud rate |
|---|---|---|---|
| dec | hex | | |
| 0 | 0 | Object $2009_h$ | 1 MBd fixed |
| 1-127 | 1-7F | Number of the rotary switches | 1 MBd fixed |
| 128 | 80 | Object $2009_h$ | Object $2005_h$ |

| Number combinations of the rotary switches | | Node-ID | Baud rate |
|---|---|---|---|
| dec | hex | | |
| 129-255 | 81-FF | (Number of the rotary switches) −128 | Object $2005_h$ |

You can find the value of $2005_h$ in the following table.

| Value in $2005_h$ | | Baud rate in kBd |
|---|---|---|
| dec | hex | |
| 120 | 78 | 100 |
| 129 | 81 | 10 |
| 130 | 82 | 20 |
| 131 | 83 | 50 |
| 132 | 84 | 125 |
| 133 | 85 | 250 |
| 134 | 86 | 500 |
| 135 | 87 | 1000 |

### 4.2.2 Establishing communication

#### 4.2.2.1 CANopen

Before starting commissioning, we recommend reading chapters Pin assignment and CANopen configuration.

1. Connect the CANopen master to the controller via the CAN_L, CAN_H cables. Check the connection of your CAN-GND and that the necessary termination resistor is present between CAN_H and CAN_L.
2. Supply the controller with voltage (+UB Logic).
3. Change the configuration values if necessary, see configuration CANopen.
4. To test the interface, send bytes `40 41 60 00 00 00 00 00` to the controller.
   Statusword ($6041_h$) was read; you receive this response: `4B 41 60 00 XX XX 00 00`.

## 4.3 Setting the motor data

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

**NOTICE**

The controller offers no protection for the motor against overload or overtemperature. For critical applications, perform an endurance test in the actual environment and, if necessary, ensure sufficient cooling of the motor.

- Number of pole pairs: Object $2030_h$:$00_h$ (pole pair count) The number of motor pole pairs is to be entered here. With a stepper motor, the number of pole pairs is calculated using the step angle, e.g., 1.8° = 50 pole pairs, 0.9° = 100 pole pairs (see step angle in motor data sheet). With BLDC motors, the number of pole pairs is specified directly in the motor data sheet.
- Object $6075_h$:$00_h$: rated current of the motor in mA (see motor data sheet

- Object $6073_h$:$00_h$: maximum current (for a stepper motor, generally corresponds to the rated current, bipolar) in tenths of a percent of the set rated current (see motor data sheet). Factory settings: "1000", which corresponds to 100% of the value in $6075_h$.
- Object **$3219_h$**:$01_h$ Maximum duration of the maximum current ($6073_h$) in ms (for initial commissioning, Nanotec recommends a value of 100 ms; this value is to be adapted later to the specific application).
- Setting the motor type:
  - Stepper motor:
    - Object $3202_h$ (Motor Drive Submode Select): Set bit 6 to "0" for stepper motor and use bit 0 to select between open- and closed-loop: value $0_h$ or $1_h$.See also chapter Commissioning *open-loop*.
    - Object **$3219_h$**:$03_h$ (Open Loop Idle State Current): Specify the root mean square in tenths of a percent to which the rated current is to be reduced if current reduction is activated in *open-loop*.
  - BLDC motor:
    - Object $3202_h$ (Motor Drive Submode Select): Bit 6 for BLDC, bit 0 for the recommended *closed-loop*: $00000041_h$
- Motor with encoder without index: You must set the encoder parameters after the Auto setup, see chapter Configuring the sensors.
- Motor with brake: Object $3202_h$:$00_h$ (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 activated: $00000004_h$
  - BLDC motor, brake control activated, *closed-loop*: $00000045_h$

**NOTICE**

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

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

## 4.4 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.

**NOTICE**

**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 connection X1 – motor connection
- Connect encoders / Hall sensors:
  - to connection X6 – incremental encoder/Hall sensors
  - or to X5 – SSI encoder
- Connect the brake:
  - to connection X4 – inputs and outputs

  How the automatic brake control can be activated is described in chapter Automatic brake control.

## 4.5 Auto setup

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

**TIP**

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

**NOTICE**

**Note the following prerequisites for performing the auto setup:**

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

**TIP**

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

### 4.5.1 Parameter determination

Auto setup determines the motor type (stepper or BLDC) and 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) | - | ✓ | --- |

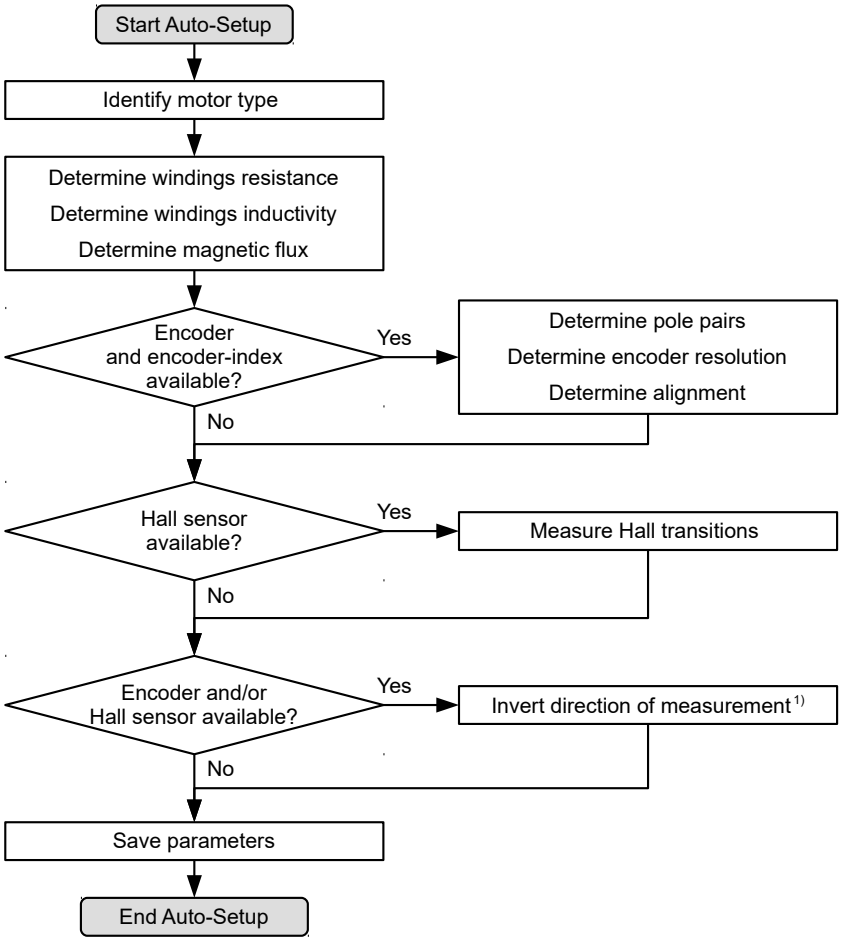
| Parameter | Motor without Hall sensor | Motor with Hall sensor |
|---|---|---|
| Hall transitions | - | ✓ |

### 4.5.2 Execution

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

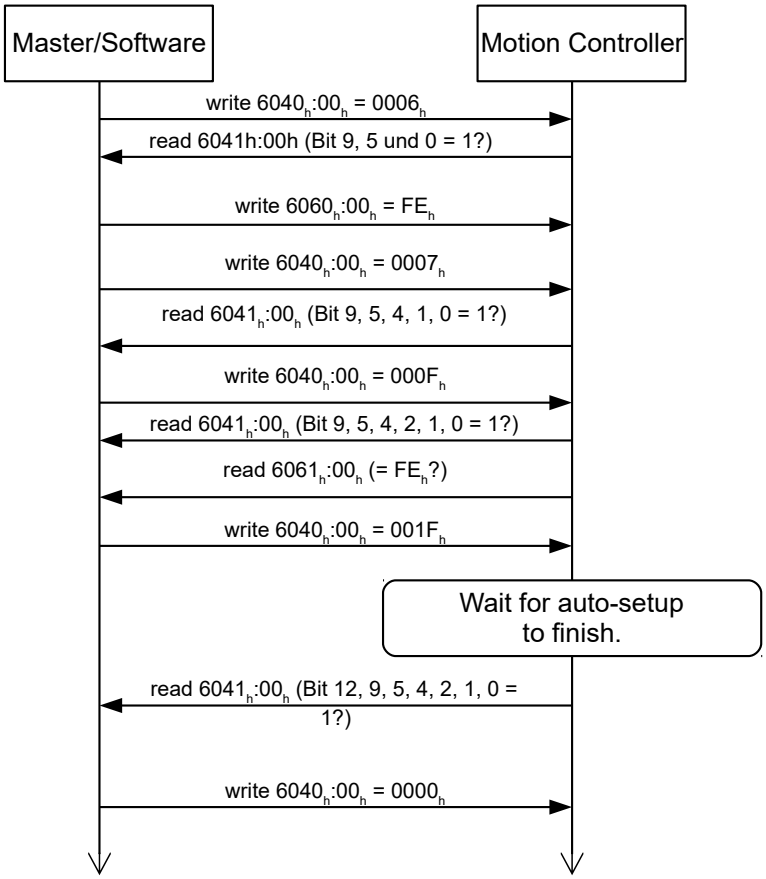
1. To preselect the *auto setup* operating mode, enter the value "-2" (="$FE_h$") in object $6060_h$:$00_h$.
   The *power state machine* must now switch to the *Operation enabled* state, see CiA 402 Power State Machine.
2. Start *auto setup* by setting bit 4 *OMS* in object $6040_h$:$00_h$ (controlword).

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




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

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




### 4.5.3 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. Categoriy *Sensor* $1010_h$:$05_h$ is 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.

## 4.6 Configuring the sensors

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

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

| Object | Feedback | Description |
|---|---|---|
| 33B0$_h$ | SSI encoder 1 | contains configuration values for the first SSI encoder |

**NOTICE**

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

In this case, you must enter and store the parameters in the corresponding objects (see 3204$_h$, 60E6$_h$ and 60EB$_h$) (category *Sensor*, 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 (6091$_h$) and/or the feed constant (6092$_h$) (category *Application*).

**Example**

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

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

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

| Object | Value |
|---|---|
| 60E6h Additional Position Encoder Resolution - Encoder Increments | 1256637 |
| 60EBh Additional Position Encoder Resolution - Motor Revolutions | 20 |
| 6091$_h$:01 (Motor Revolutions) | 4 |
| 6091$_h$:02 (Shaft Revolutions) | 1 |
| 6092$_h$:01 (Feed) | 2513274 incr. (corresponds to 1256.637 mm) |
| 6092$_h$:02 (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$_h$ you can set which of the present feedbacks the controller takes into account for each controller (current controller/commutation, velocity controller, position controller) in *closed-loop* or the determination of the actual position and actual speed in *open-loop*. See also chapter Closed Loop and Assignment of the feedbacks to the control loops.

**NOTICE**

The value "0" in a subindex of the object 60E6$_h$ means that the respective feedback is not connected and is not used. Thus, it is possible, for Example, to switch off the sensorless function to save computing time. This can be helpful if a *NanoJ* program needs the computing time.

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

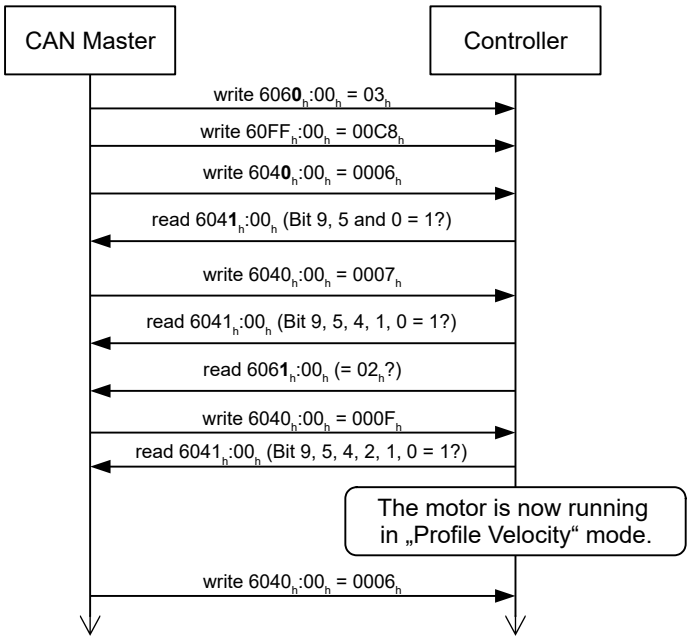
## 4.7 Test run

After configuring and the auto setup, a test run can be performed. As an example, the Profile Velocity operating mode is used.

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

1. Select the mode by setting object 6060$_h$ (Modes Of Operation) to the value "3".
2. Write the desired speed in 60FF$_h$.
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$_h$) to "6".

# 5 General concepts

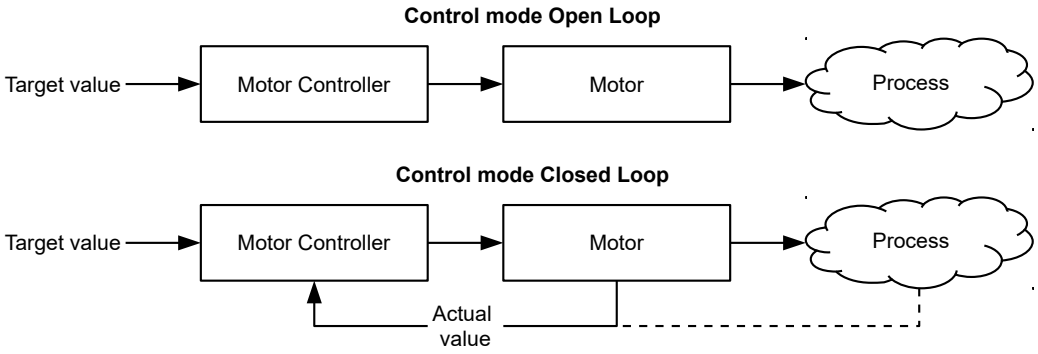
## 5.1 Control modes

### 5.1.1 General

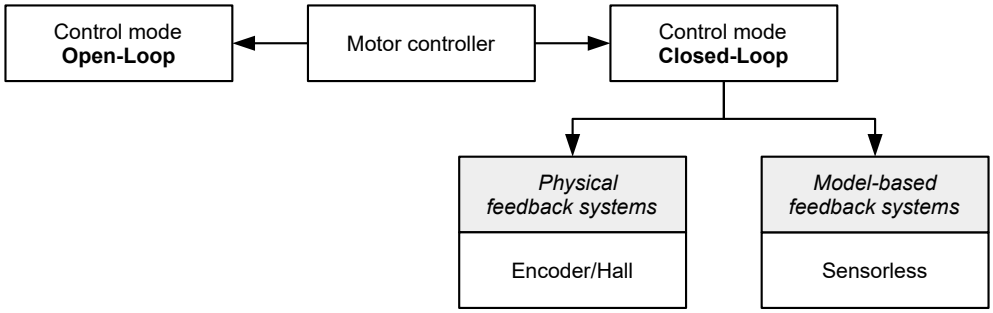
The control mode of systems without feedback is called *open-loop*, the mode with feedback is called *closed-loop*. In the *closed-loop* control mode, it is initially irrelevant whether the fed back signals come from the motor itself or from the influenced process.

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 |

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 |
| Profile Position | yes | yes |
| Profile Velocity | yes | yes |
| Profile Torque | no[1)] | yes |
| Homing | yes[2)] | yes |
| Interpolated Position Mode | yes[3)] | yes |
| Cyclic Synchronous Position | yes[3)] | yes |
| Cyclic Synchronous Velocity | yes[3)] | yes |
| Cyclic Synchronous Torque | no[1)] | yes |
| Clock-direction | 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.

## 5.1.2 Open Loop

### 5.1.2.1 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.

### 5.1.2.2 Commissioning

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

- In object $2030_h$ (Pole Pair Count), enter the number of pole pairs (see motor data sheet: for a stepper motor with 2 phases, a step angle of 1.8° corresponds to 50 pole pairs and 0.9° corresponds to 100 pole pairs).
- In object $6075_h$:$00_h$, enter the rated current of the motor in mA (see motor data sheet).

- In object $6073_h$:$00_h$, enter the maximum current (for a stepper motor, generally corresponds to the rated current, bipolar) in tenths of a percent of the set rated current (see motor data sheet). Factory settings: "1000", which corresponds to 100% of the value in $6075_h$. A value greater than "1000" is limited internally to "1000".
- In object $3202_h$ (Motor Drive Submode Select), set bit 0 (CL/OL) to the value "0".

Nanotec recommends the current reduction on motor standstill in order to reduce the power loss and heat build-up:

- In object $3219_h$:02 (Open Loop Idle State Detection 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 $3219_h$:03 (Open Loop Idle State Current), 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.

#### 5.1.2.3 Optimizations

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

- Reduce or increase current, see objects $6073_h$ and $6075_h$, respectively. An excessive torque reserve promotes resonances.
- Reduce or increase the operating voltage, taking into account the product-specific ranges (with sufficient torque reserve). The permissible operating voltage range can be found in the product data sheet.
- Adjustments to the acceleration, deceleration and/or target speed depending on the selected control mode:

  **Profile Position operating mode**

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

  **Profile Velocity operating mode**

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

  **Homing operating mode**

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

  **Interpolated Position Mode operating mode**

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

  **Cyclic Synchronous Position operating mode**

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

  **Cyclic Synchronous Velocity operating mode**

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

  **Clock-direction operating mode**

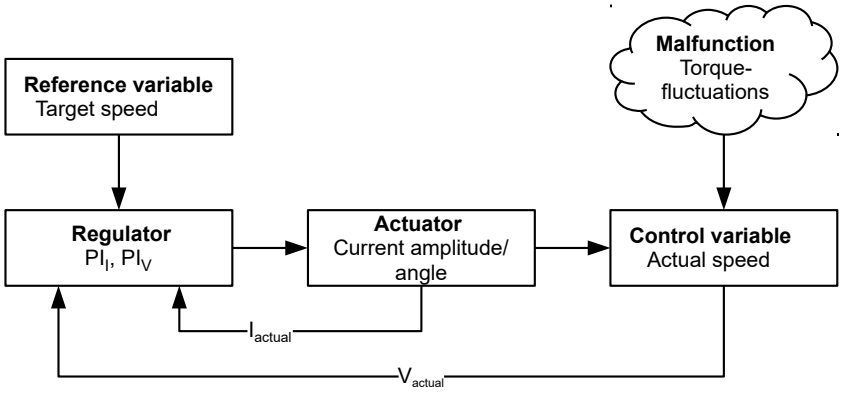
  Change of the step resolution via objects $2057_h$ (Clock Direction Multiplier) and $2058_h$ (Clock Direction Divider). Optimize acceleration / deceleration ramps by adjusting the pulse frequency to pass through the resonance range as quickly as possible.

### 5.1.3 Closed Loop

#### 5.1.3.1 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 rotor position through a software-based model calculation. This requires knowledge of relevant physical motor parameters, in particular the stator inductance, the winding resistance, and the back-EMF. Using a mathematical motor model, a so-called “observer” estimates the rotor position and speed from the measured currents. The sensorless observer thus functions as a “virtual encoder” and, with appropriate parameter settings, can achieve an accuracy comparable to that of optical or magnetic encoders.

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.

### 5.1.3.2 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
- Profile Velocity/Cyclic Synchronous Velocity if bit 1 in $3202_h$ is set to "1"

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

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

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

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

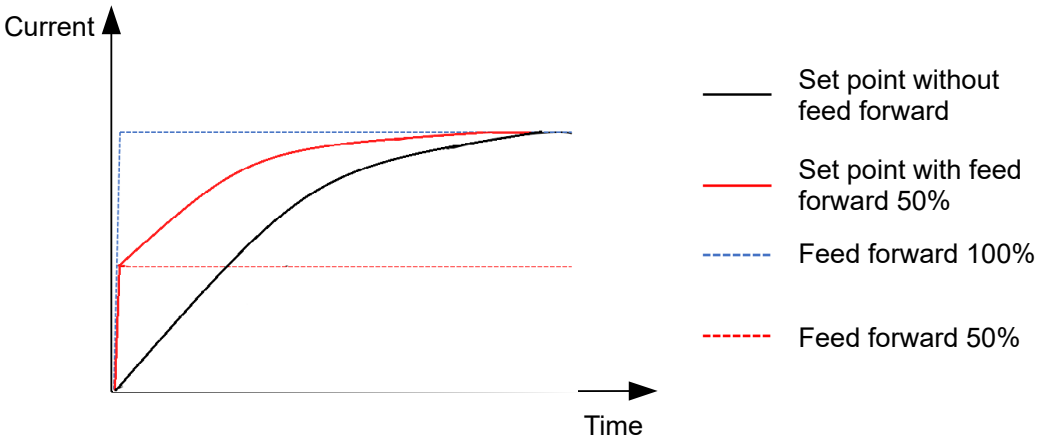
### 5.1.3.3 Feed forward

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

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 and are immediately available. A more dynamic control can thereby be achieved.

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

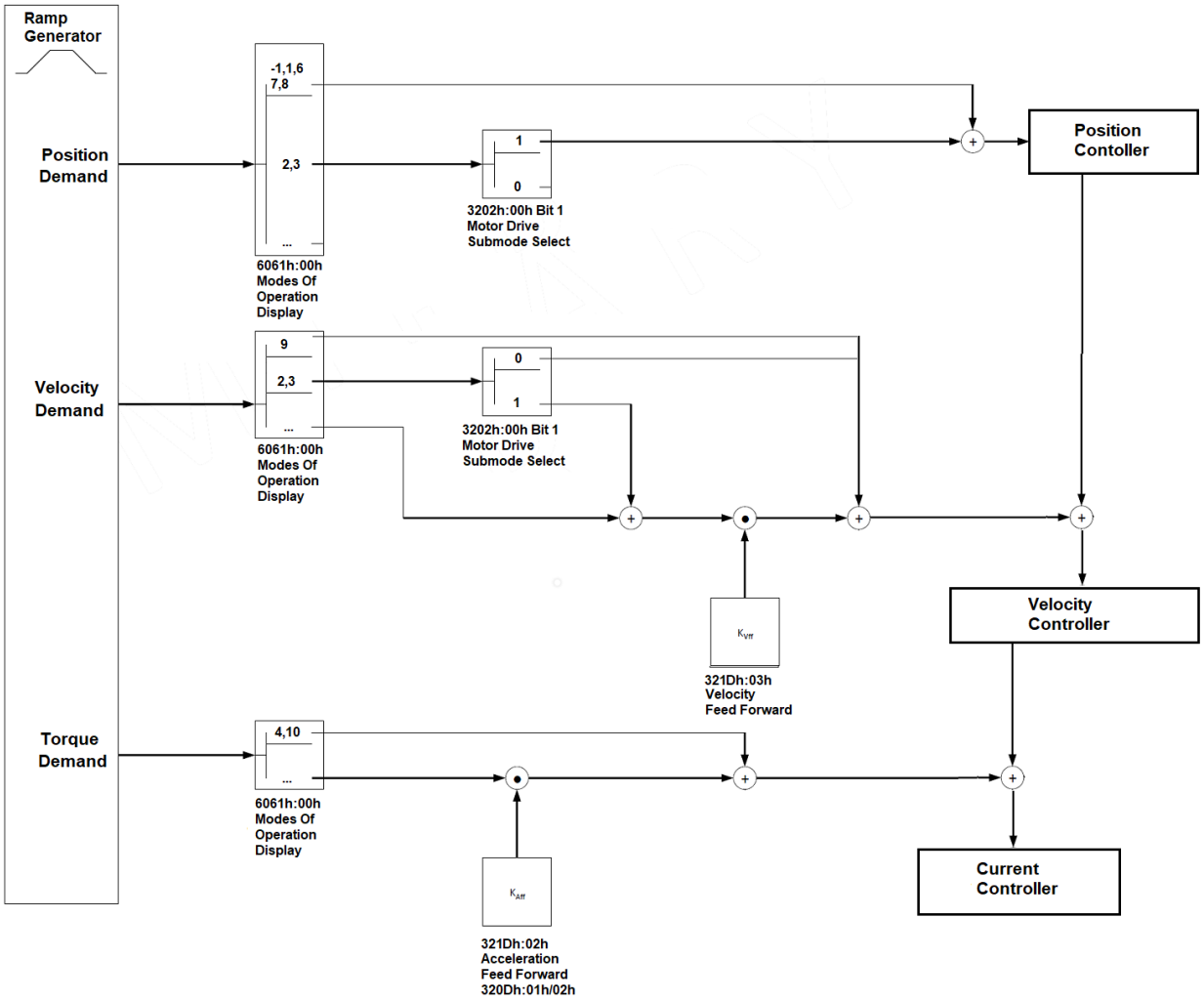



The factor for the *velocity feed forward* is set in object 321D$_h$:03$_h$ in tenths of a percent of the output of the ramp generator (606B$_h$) and added to the output of the position controller before the velocity controller. The *velocity feed forward* is active in all modes with position control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

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

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

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




### 5.1.3.4 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.

Object $3205_h$ shows which sensor is currently used for which control circuit.

**Example**

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

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

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

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

#### Commutation help

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

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

### 5.1.3.5 Commissioning

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

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

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

#### Activation

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

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

If no index is present or if it cannot be used, you can use a second sensor for commutation (see Assignment of the feedbacks to the control loops).

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*).

### 5.1.3.6 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.

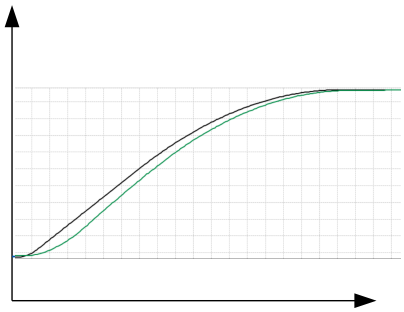
Due to the cascaded Controller structure, it is useful to start the optimization of the inner-most controller (current controller) before the velocity and – if applicable – the position controller are optimized. Each of the three controllers consists of a proportional and an integral component, which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

You can calculate the current controller parameters on the basis of the motor parameters as follows:

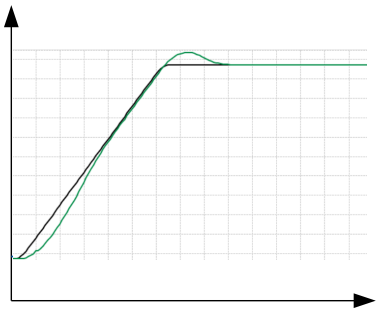
1. $K_p=L/(4*T_c)$, where L is the winding inductance in henrys and $T_c$ is the sampling time (see Cycle times) in seconds.
2. $K_i=L/R$ , where R is the winding resistance in ohm.
3. Convert units and enter values in the corresponding subindices:
   $321A_h:01_h=K_p*10^3$
   $321A_h:02_h=K_1*10^6$

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".

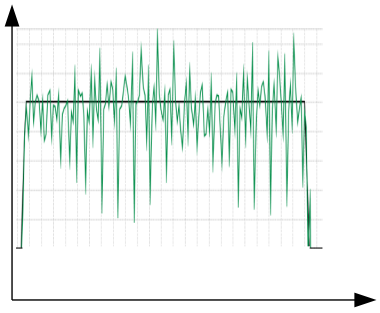


P-part too small

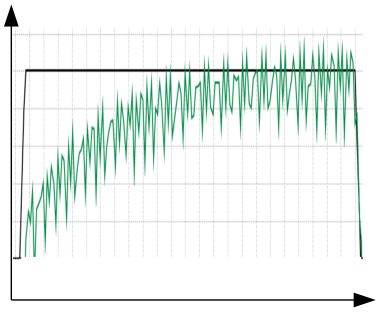


P-part too big

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



Ti too small



Ti too big

**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.

## 5.2 CiA 402 Power State Machine

### 5.2.1 State machine

#### 5.2.1.1 CiA 402

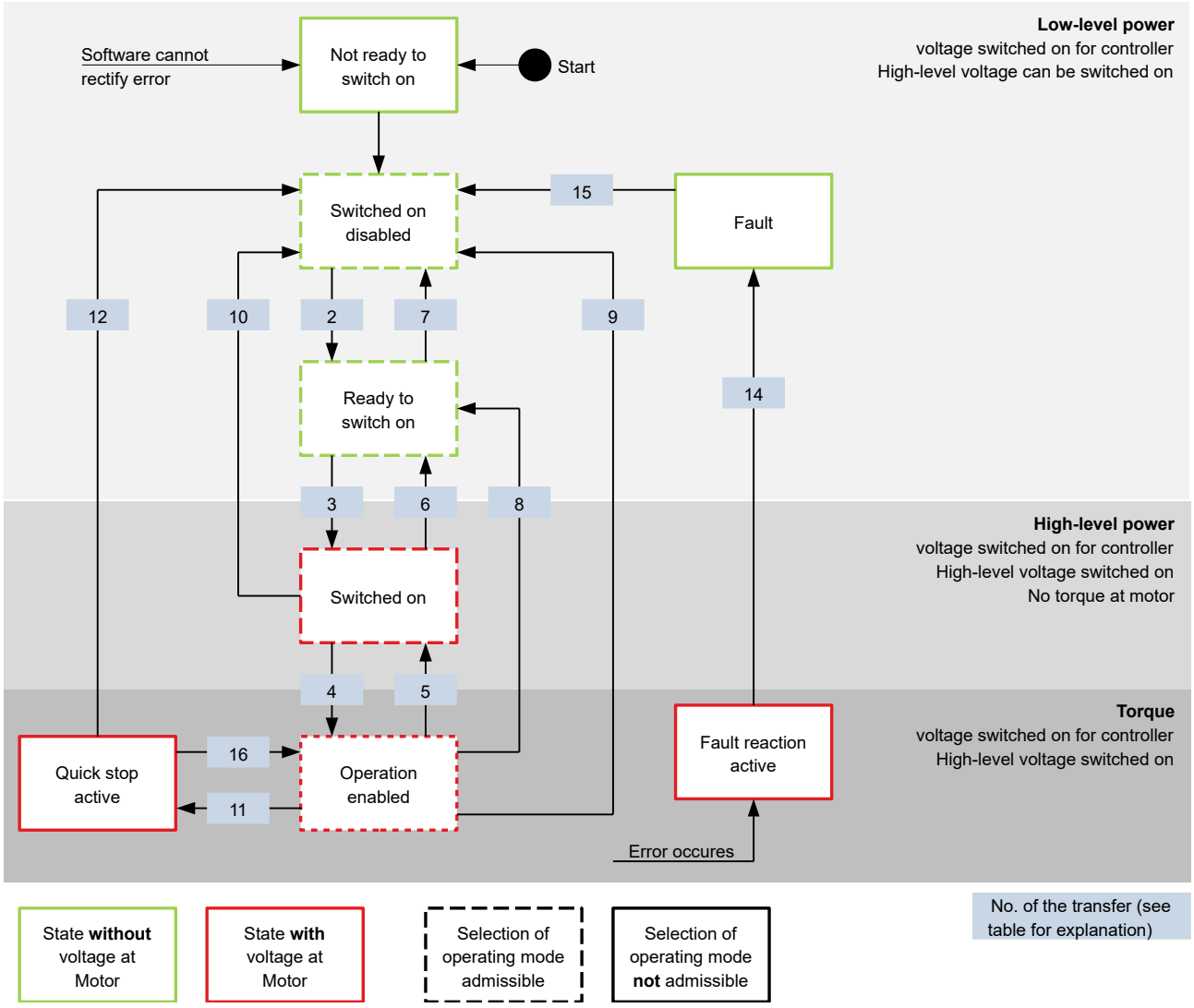
To switch the controller to the ready state, it is necessary to run through a *state machine*. This is defined in *CANopen standard 402*. State changes are requested in object $6040_h$ (controlword). The actual state of the state machine can be found in object $6041_h$ (statusword).

#### 5.2.1.2 Controlword

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

#### State transitions

The diagram shows the possible state transitions.




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

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

| Command | Bit in object $6040_h$ | | | | | Transition |
|---|---|---|---|---|---|---|
| | Bit 7 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | |
| Fault reset | | X | X | X | X | 15 |

### 5.2.1.3 Statusword

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

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

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

### 5.2.1.4 Operating mode

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

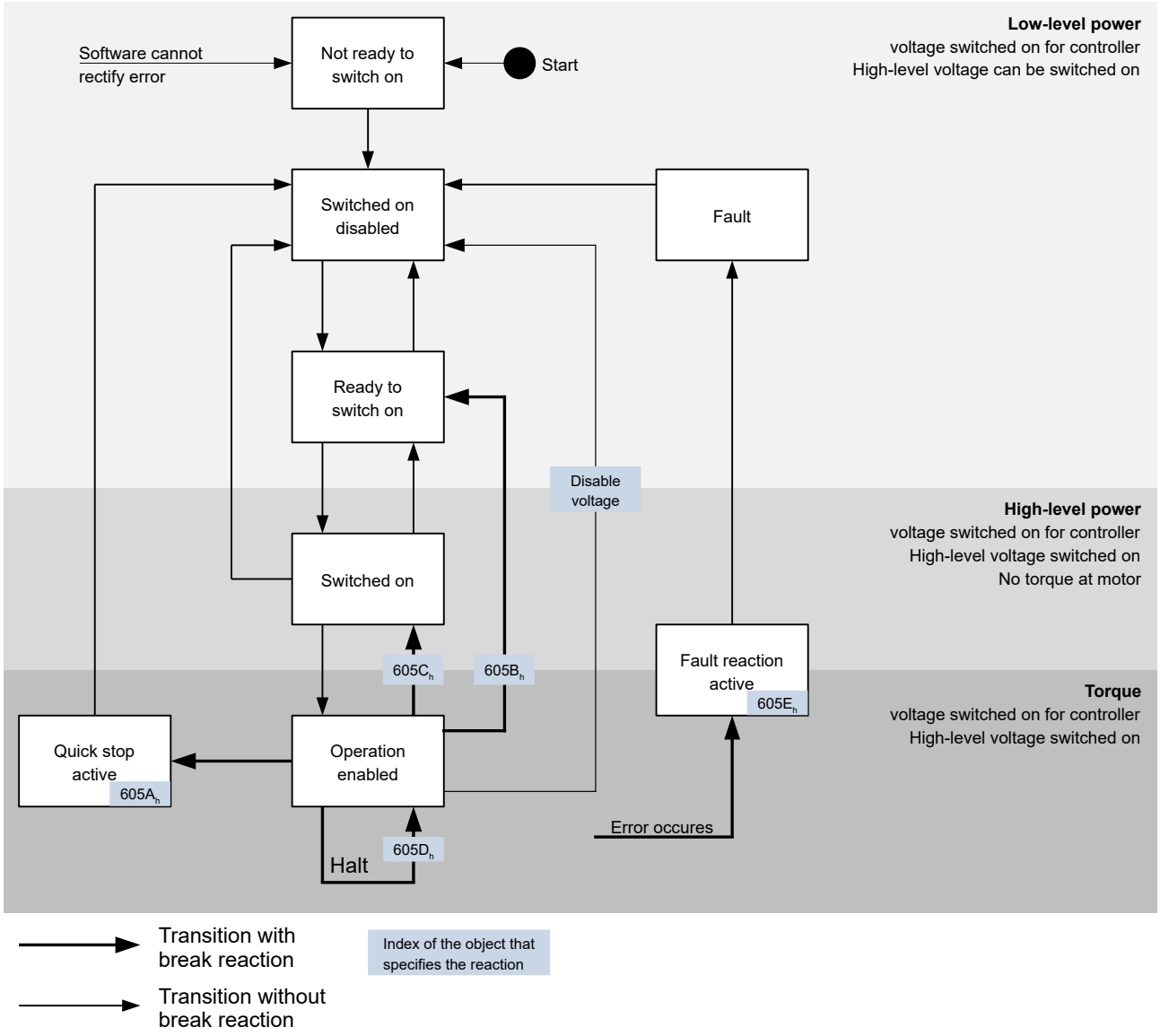
The operating mode can be set or changed at any time.

## 5.2.2 Behavior upon exiting the *Operation enabled* state

### 5.2.2.1 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.




### 5.2.2.2 Quick stop active

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

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

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

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

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

#### 5.2.2.3 Ready to switch on

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

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

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

#### 5.2.2.4 Switched on

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

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

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

#### 5.2.2.5 Halt

The bit is valid in the following modes:

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

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

| Value in object $605D_h$ | Description |
|---|---|
| -32768 … 0 | Reserved |
| 1 | Braking with *slow down ramp* (braking deceleration depending on operating mode) |
| 2 | Braking with *quick stop ramp* ($6085_h$) |
| 3 … 32767 | Reserved |

#### 5.2.2.6 Fault

Case of an error (fault):

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

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

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

#### 5.2.2.7 Following/slippage error

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

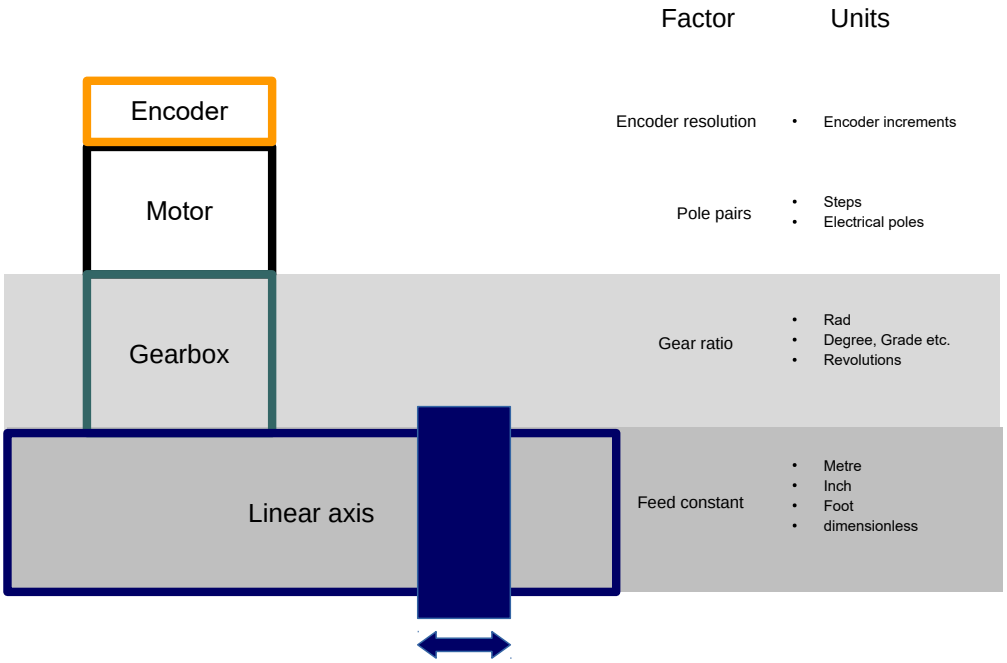
| Value | Description |
|---|---|
| -32768 … -2 | Reserved |
| -1 | no reaction |
| 0 | Switch off driver without deceleration ramp; drive function blocked – motor can turn freely |
| 1 | Braking with *slow down ramp* (braking deceleration depending on operating mode) |
| 2 | Braking with *quick stop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

You can deactivate error monitoring by setting object 6065$_h$ to the value "-1" (FFFFFFFF$_h$) or object 60F8$_h$ to the value "7FFFFFFF$_h$".

## 5.3 User-defined units

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

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




**NOTICE**

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

### 5.3.1 Units

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

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

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

| Name | Unit symbol | Value | 6091$_h$ | 6092$_h$ | Description |
|---|---|---|---|---|---|
| encoder increment | | B5$_h$ | no | no | *Encoder increments.* Dependent on the used sensor (encoder/Hall sensor) and control mode. In *open-loop* and *sensorless* mode and if all other sensors in 3203$_h$ are deactivated, the number of pole pairs (2030$_h$) multiplied by 65536 corresponds to one motor revolution. |
| step | | AC$_h$ | no | no | *Steps.* With 2-phase stepper motors, the number of pole pairs (2030$_h$) multiplied by 4 is equivalent to one revolution. With 3-phase BLDC motors, the number of pole pairs (2030$_h$) multiplied by 6 is equivalent to one revolution. |
| electrical revolution | | C0$_h$ | no | no | *Electrical revolution.* With a stepper motor that has, e.g., 50 pole pairs (2030$_h$), the unit corresponds to 1/50 of a motor revolution. |
| dimensionless | | 00$_h$ | yes | yes | *Dimensionless length unit* |

Listed in the following table are all supported units for the time and their values for 60A9$_h$ (Speed unit):

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

Listed in the following table are the possible exponents and their values for 60A8$_h$ (Position unit) and 60A9$_h$ (Speed unit):

| Factor | Exponent | Value |
|---|---|---|
| $10^6$ | 6 | 06$_h$ |
| $10^5$ | 5 | 05$_h$ |
| ... | ... | ... |
| $10^1$ | 1 | 01$_h$ |
| $10^0$ | 0 | 00$_h$ |
| $10^{-1}$ | -1 | FF$_h$ |
| ... | .. | ... |
| $10^{-5}$ | -5 | FB$_h$ |
| $10^{-6}$ | -6 | FA$_h$ |

### 5.3.2 Encoder resolution

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

### 5.3.3 Gear ratio

The gear ratio is calculated from motor revolutions (6091$_h$:01) per axis rotation (6091$_h$:02).

### 5.3.4 Feed constant

The feed constant is calculated in user-defined position units from the feed ($6092_h$:01) per revolution of the output shaft ($6092_h$:02).

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.

### 5.3.5 Calculation formulas for user units

#### 5.3.5.1 Position unit

Object $60A8_h$ contains:

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

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

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

**Example**

If $60A8_h$ is written with the value "FF410000$_h$" (bits 16-23=$41_h$ and bits 24-31=$FF_h$), the unit is set to *tenths of degree* (factory setting).

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

**Example**

If $60A8_h$ is written with the value "FD010000$_h$" (bits 16-23=$01_h$ and bits 24-31=$FD_h$(=-3)), the unit is set to *millimeter*.

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

#### 5.3.5.2 Speed unit

Object $60A9_h$ contains:

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

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

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

**Example**

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

**Example**

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

### Conversion factor for the speed unit

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

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

#### 5.3.5.3 Acceleration unit

The acceleration unit is speed unit per second.

### Conversion factor for the acceleration unit

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

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

#### 5.3.5.4 Jerk unit

The jerk unit is Acceleration unit per second.

### Conversion factor for jerk

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

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

## 5.4 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.

### 5.4.1 Behavior upon reaching the limit switch

If a limit switch is triggered, the limit switch position is stored internally, bit 7 (*Warning*) in $6041_h$ (*statusword*) is set and the CiA 402 Power State Machine is set to status *Quick Stop Active*. The action stored in object $3701_h$ is thereby executed (see following table).

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

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

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

**NOTICE**

To avoid automatically returning from the *Quick stop active* state to *Operation enabled* when using options "5" or "6" — the quick-stop bit (bit 2) in $6040_h$ is not used upon triggering of the limit switches — a change of the quick-stop bit from "0" to "1" is expected in order to changed back to the *Operation enabled* state (605Ah Quick Stop Option Code must be set to "5" or "6").

#### Discarding the limit switch position

**NOTICE**

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

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

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

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

4. Reset $3701_h$ back to "5" or "6".

### 5.4.2 Software limit switches

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

## 5.5 Cycle times

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

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

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

# 6 operating modes

## 6.1 Profile Position

### 6.1.1 Overview

#### 6.1.1.1 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.

#### 6.1.1.2 Activation

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

#### 6.1.1.3 Controlword

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

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

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

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

**NOTICE**

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

#### 6.1.1.4 Statusword

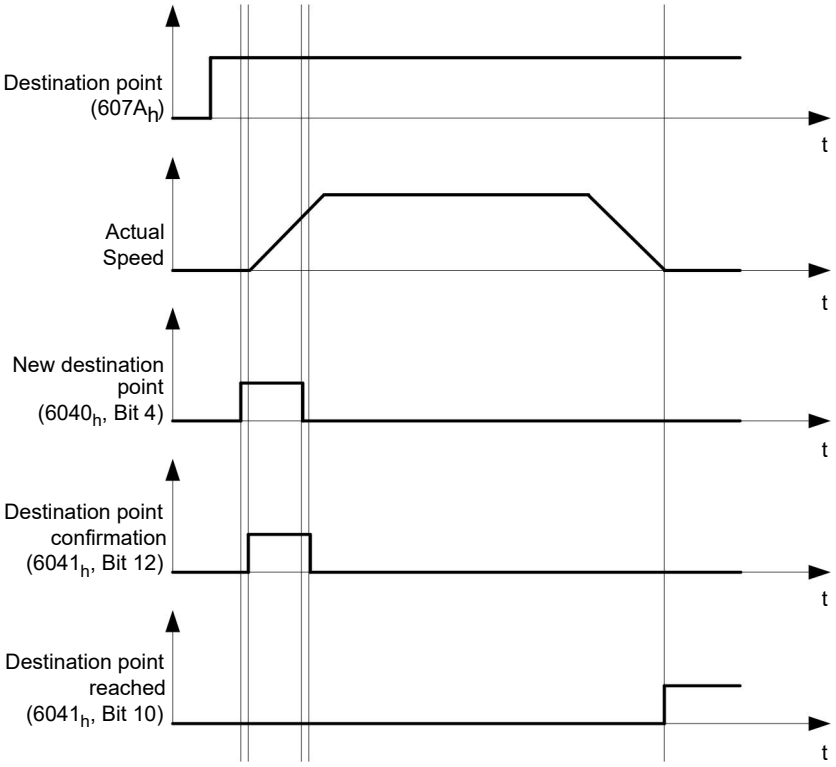
The following bits in object $6041_h$ (statusword) have a special function:

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

## 6.1.2 Setting travel commands

### 6.1.2.1 Travel command

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




The controller can also reset bit 4 in object $6040_h$ (controlword) on its own. This is set with bits 4 and 5 of object $60F2_h$.

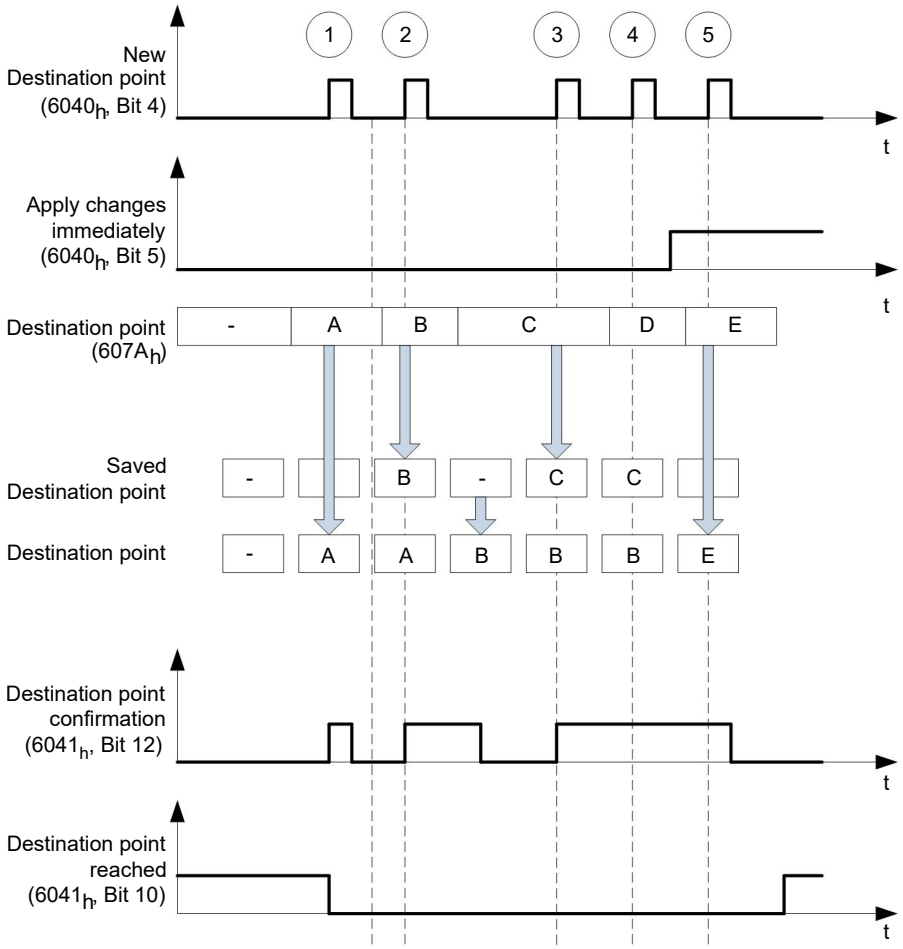
### 6.1.2.2 Other travel commands

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

deceleration, etc. – can thereby be reset (time 2). If the buffer is empty, the next time can be queued up (time 3).

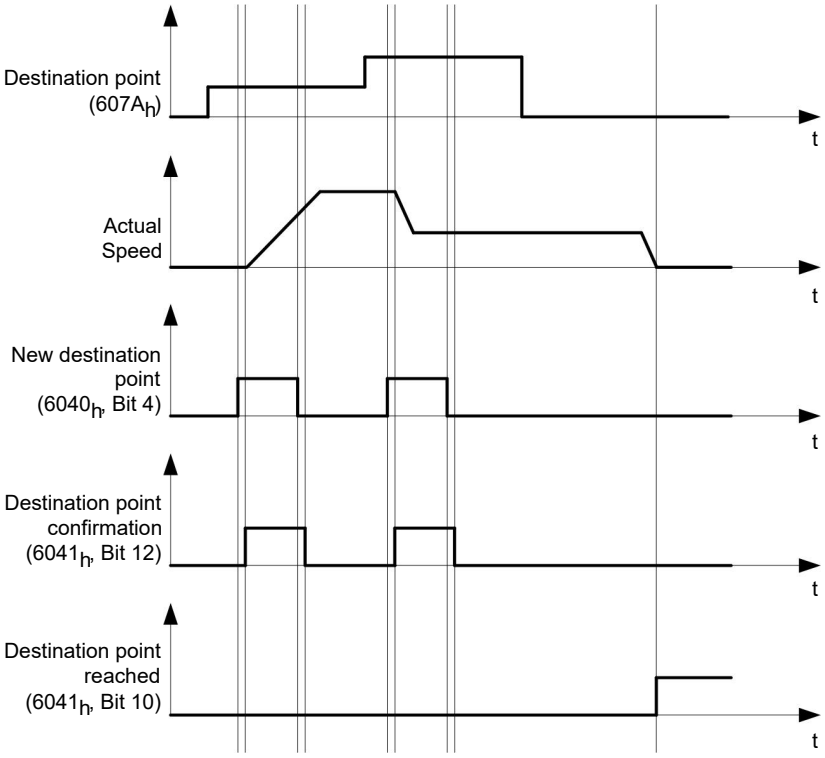
If the buffer is already full, a new set point is ignored (time 4). If bit 5 in object 6040h (controlword, bit: "Change Set-Point Immediately") is set, the controller operates without the buffer; new travel commands are implemented directly (time 5).

### Times



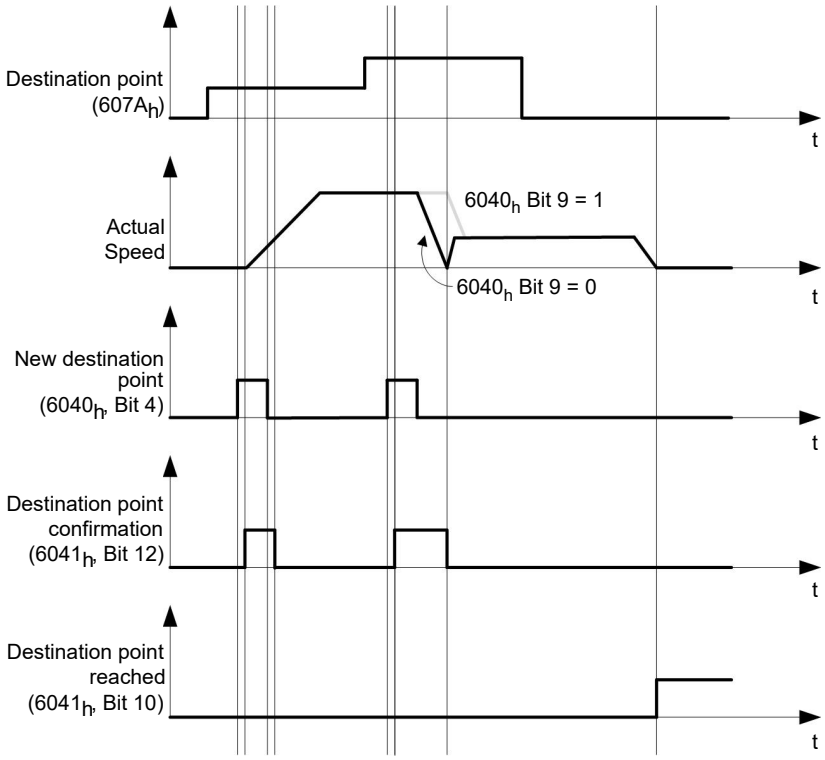

### Transition procedure for second target position

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



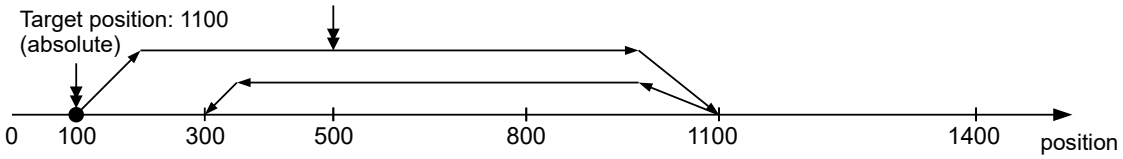

## Possible combinations of travel commands

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

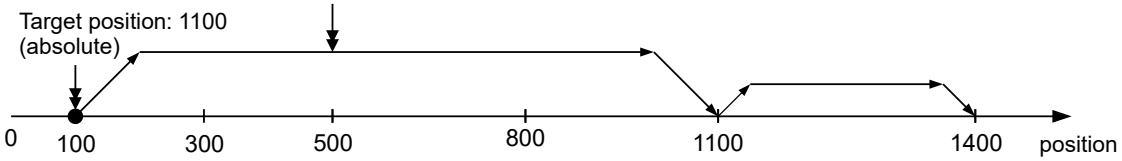
The following applies for the figures below:

- 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.

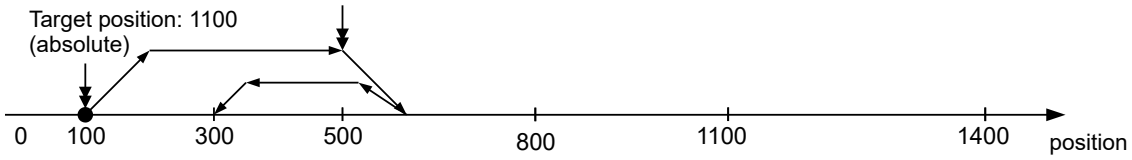
- Change on setpoint ($6040_h$:00 Bit 5 = 0)
- Move absolute ($6040_h$:00 Bit 6 = 0)
- Target position: 300



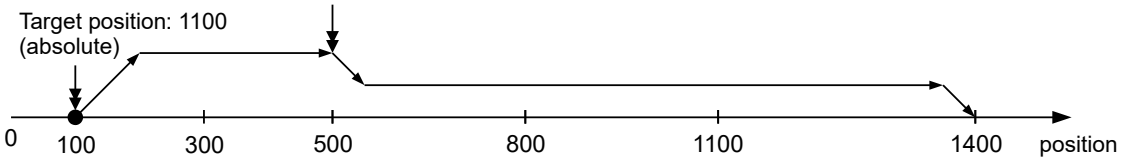

- Relative to the preceding target position ($60F2_h$:00 = 0)
- Change on setpoint ($6040_h$:00 Bit 5 = 0)
- Move relative ($6040_h$:00 Bit 6 = 1)
- Target position: 300

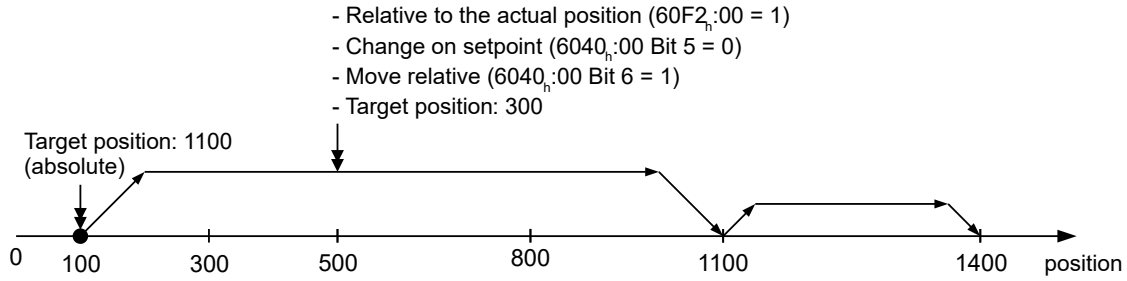


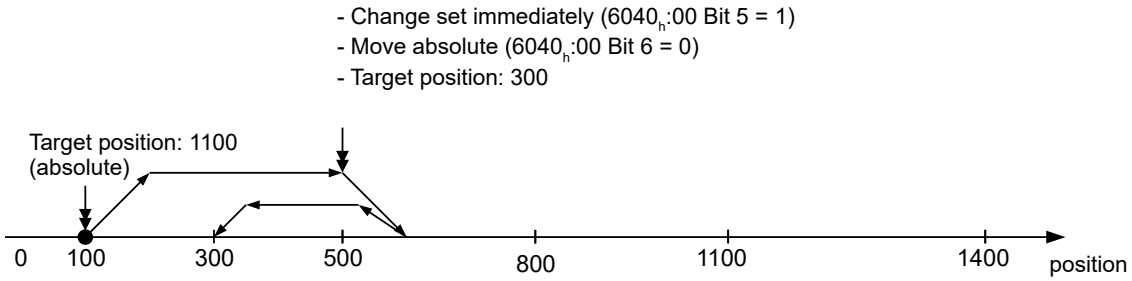

- Change set immediately ($6040_h$:00 Bit 5 = 1)
- Move absolute ($6040_h$:00 Bit 6 = 0)
- Target position: 300

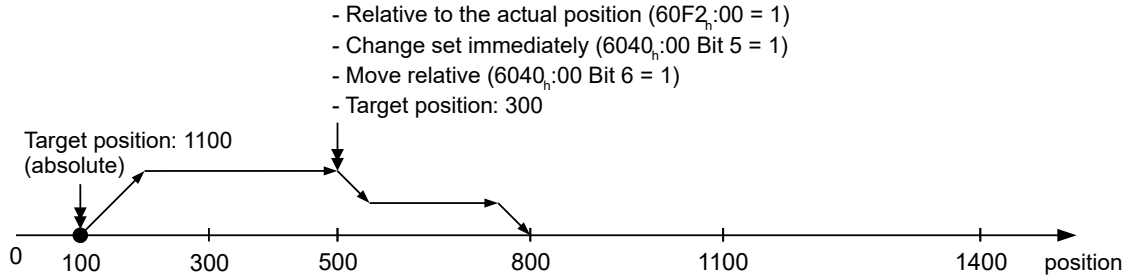



- Relative to the preceding target position ($60F2_h$:00 = 0)
- Change set immediately ($6040_h$:00 Bit 5 = 1)
- Move relative ($6040_h$:00 Bit 6 = 1)
- Target position: 300



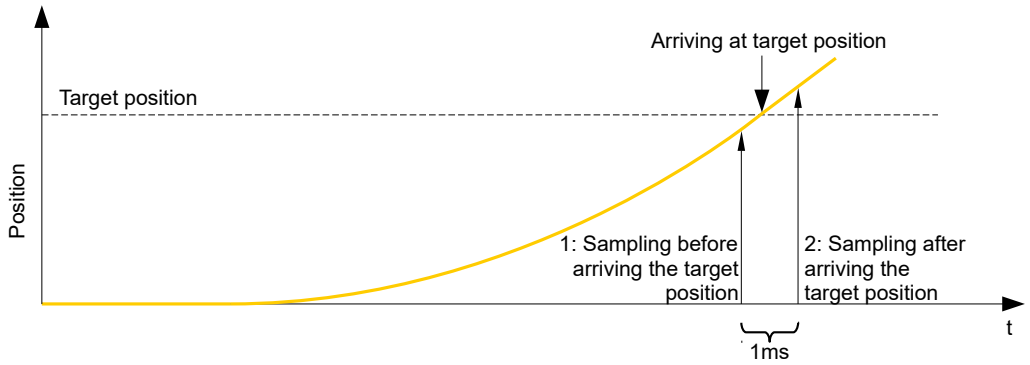



### 6.1.3 Loss of accuracy for relative movements

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.

## 6.1.4 Boundary conditions for a positioning move

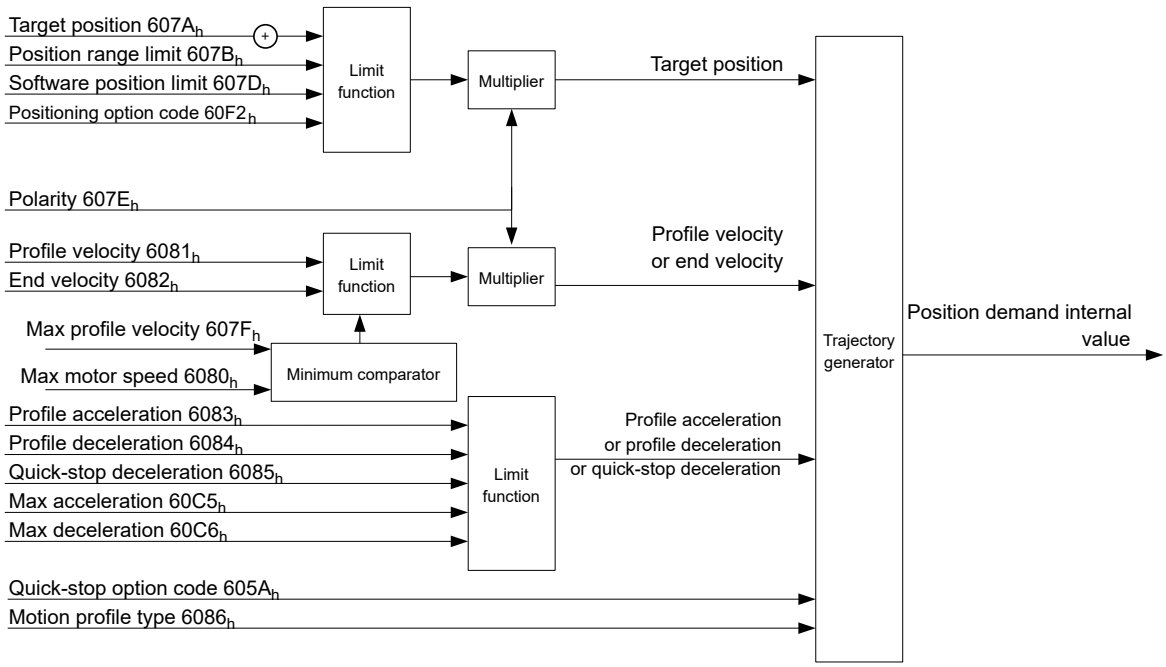
### 6.1.4.1 Object entries

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

- 607A$_h$: (Target Position): Planned target position
- 607D$_h$: (Software Position Limit): Definition of the limit stops (see chapter Software limit switches)
- 607C$_h$ (Home Offset): Specifies the difference between the zero position of the controller and the reference point of the machine in user-defined units. (See "Homing")
- 607B$_h$ (Position Range Limit): Limits of a modulo operation for replicating an endless rotation axis
- 607$_h$ (Polarity): Direction of rotation
- 6081$_h$ (Profile Velocity): Maximum speed with which the position is to be approached
- 6082$_h$ (End Velocity): Speed upon reaching the target position
- 6083$_h$ (Profile Acceleration): Desired starting acceleration
- 6084$_h$ (Profile Deceleration): Desired braking deceleration
- 6085$_h$ (Quick Stop Deceleration): Emergency-stop braking deceleration in case of the "Quick stop active" state of the "CiA 402 Power State Machine"
- 6086$_h$ (Motion Profile Type): Type of ramp to be traveled; if the value is "0", the jerk is not limited; if the value is "3", the values of 60A4$_h$:1$_h$–4$_h$ are set as limits for the jerk.
- 60C5$_h$ (Max Acceleration): The maximum acceleration that may not be exceeded when moving to the end position
- 60C6$_h$ (Max Deceleration): The maximum braking deceleration that may not be exceeded when moving to the end position
- 60A4$_h$ (Profile Jerk), subindex 01$_h$ to 04$_h$: Objects for specifying the limit values for the jerk.
- The speed is is limited by 607F$_h$ (Max Profile Velocity) and 6080$_h$ (Max Motor Speed); the smaller value is used as the limit.
- 60F2$_h$: (Positioning Option Code): Defines the positioning behavior

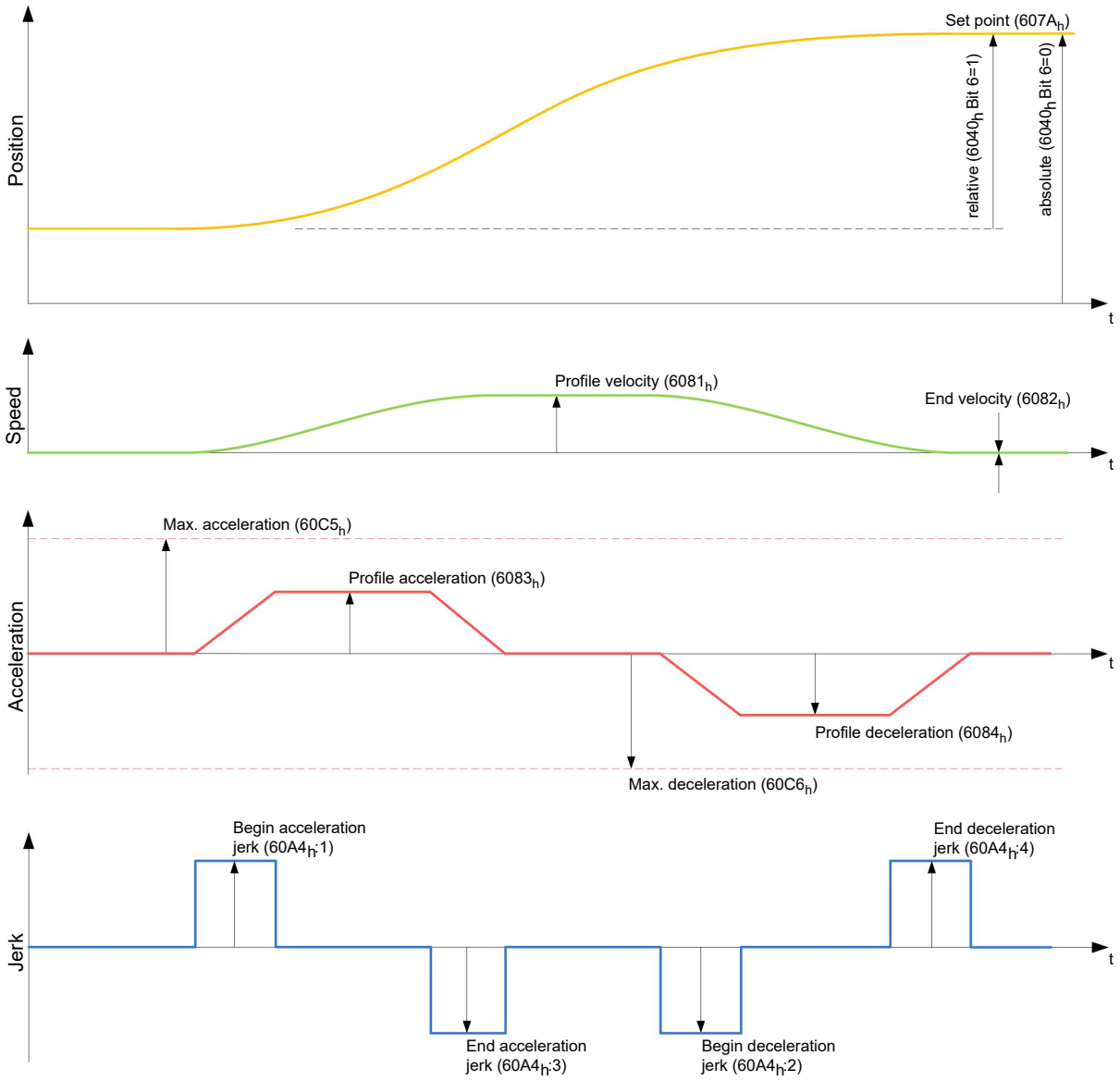
### 6.1.4.2 Objects for the positioning move

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



### 6.1.4.3 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).




## 6.1.5 Jerk-limited mode and non-jerk-limited mode

### 6.1.5.1 Description

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

### 6.1.5.2 Jerk-limited mode

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

### 6.1.5.3 Non-jerk-limited mode

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

## 6.2 Profile Velocity

### 6.2.1 Description

This mode operates the motor in Velocity Mode with extended (jerk-limited) ramps.

### 6.2.2 Activation

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

### 6.2.3 Controlword

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

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

### 6.2.4 Statusword

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

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

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

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

### 6.2.5 Object entries

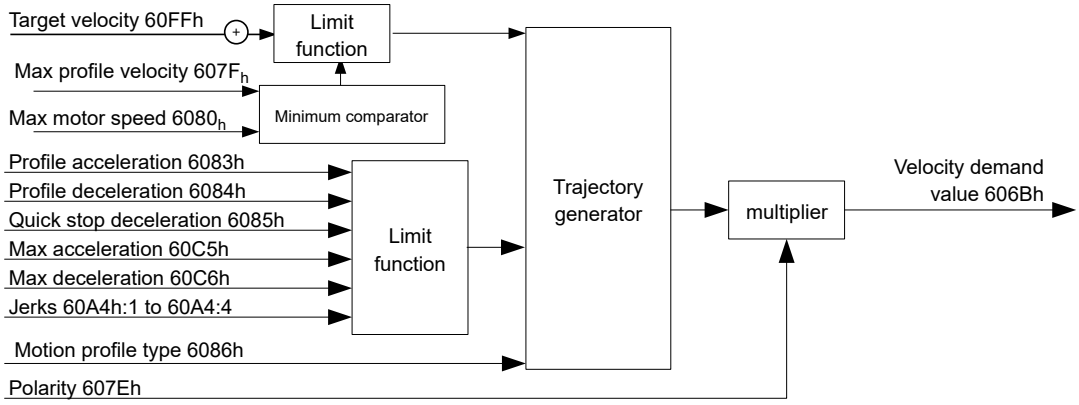
The following objects are necessary for controlling this mode:

- 606B$_h$ (Velocity Demand Value):
  This object contains the output of the ramp generator, which simultaneously serves as the preset value for the velocity controller.
- 606C$_h$ (Velocity Actual Value):
  Indicates the current actual speed.
- 606D$_h$ (Velocity Window):
  This value specifies by how much the actual speed may vary from the set speed for bit 10 (target speed reached; Target Reached") in object 6041$_h$ (statusword) to be set to "1".
- 606E$_h$ (Velocity Window Time):
  This object specifies how long the actual speed and the set speed must be close to one another (see 606D$_h$ "Velocity Window") for bit 10 "Target speed reached" in object 6041$_h$ (statusword) to be set to "1".
- 607E$_h$ (Polarity):
  If bit 6 is set to "1" here, the sign of the target speed is reversed.
- 6083$_h$ (Profile acceleration):
  Sets the value for the acceleration ramp.
- 6084$_h$ (Profile Deceleration):

Sets the value for the deceleration ramp.

- 6085$_h$ (Quick Stop Deceleration):
  Sets the value for the deceleration ramp for rapid braking.
- 6086$_h$ (Motion Profile Type):
  The ramp type can be selected here ("0" = trapezoidal ramp, "3" = jerk-limited ramp).
- 60FF$_h$ (Target Velocity):
  Specifies the target speed that is to be reached.
- The speed is is limited by 607F$_h$ (Max Profile Velocity) and 6080$_h$ (Max Motor Speed); the smaller value is used as the limit.

### 6.2.5.1 Objects in Profile Velocity Mode

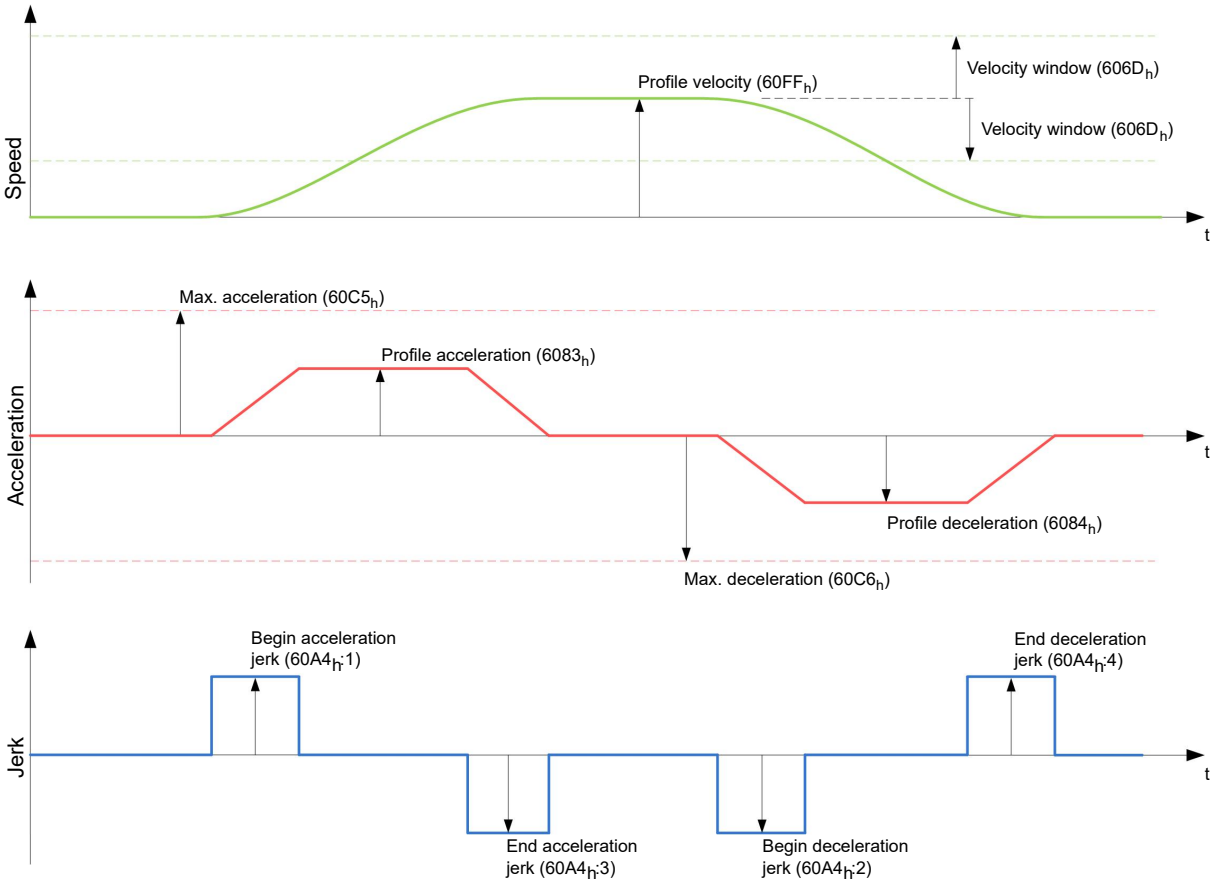



### 6.2.5.2 Activation

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.

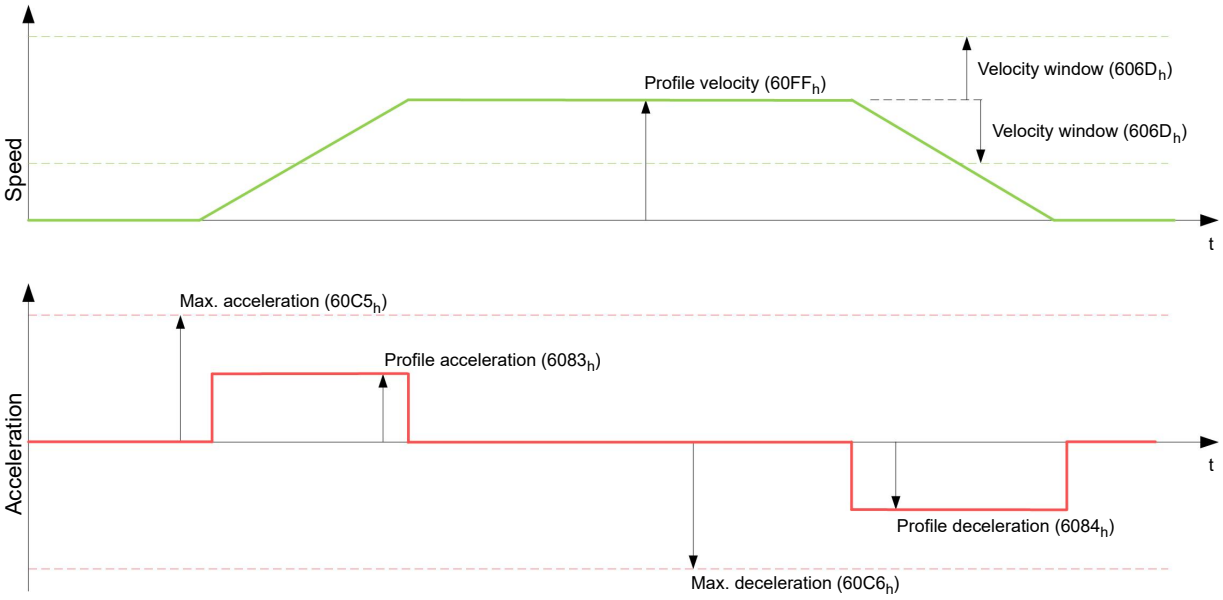
### 6.2.5.3 Limitations in the jerk-limited case

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




### 6.2.5.4 Limitations in the trapezoidal case

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




## 6.3 Profile Torque

### 6.3.1 Description

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

**NOTICE**

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

### 6.3.2 Activation

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

### 6.3.3 Controlword

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

- Bit 8 (Halt): If this bit is set to "1", the motor stops. 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.

### 6.3.4 Statusword

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

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

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

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

### 6.3.5 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 ($6075_h$). 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
- 6077 (Torque Actual Value):
  Current torque value
- $6087_h$ (Torque Slope):
  Max. change in torque per second

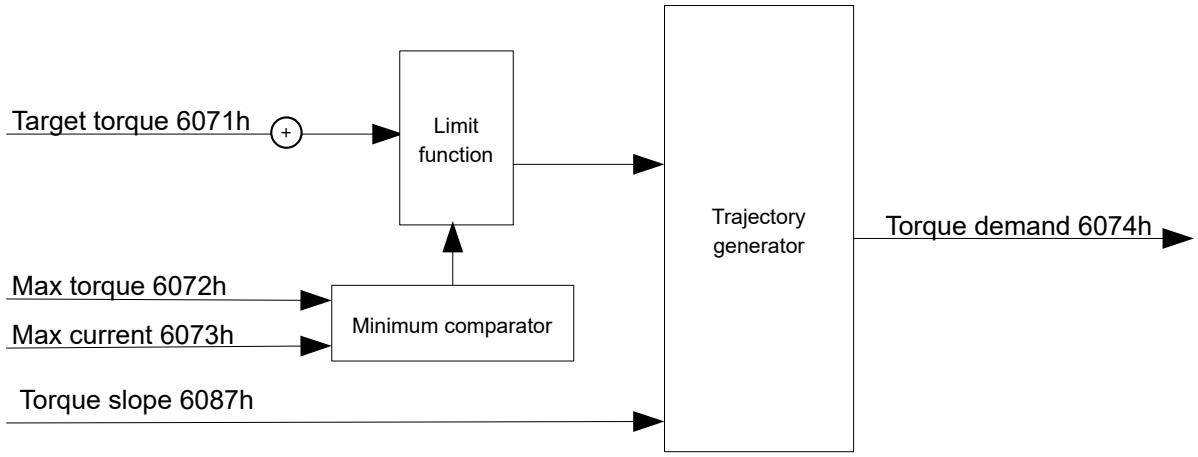
**NOTICE**

These values are not limited to 100% of the rated current ($6075_h$). Torque values greater than the rated torque (generated from the rated current) can be achieved if the maximum duration ($3219_h$:$01_h$) of the maximum current ($6073_h$) is set (see I2t Motor overload protection).

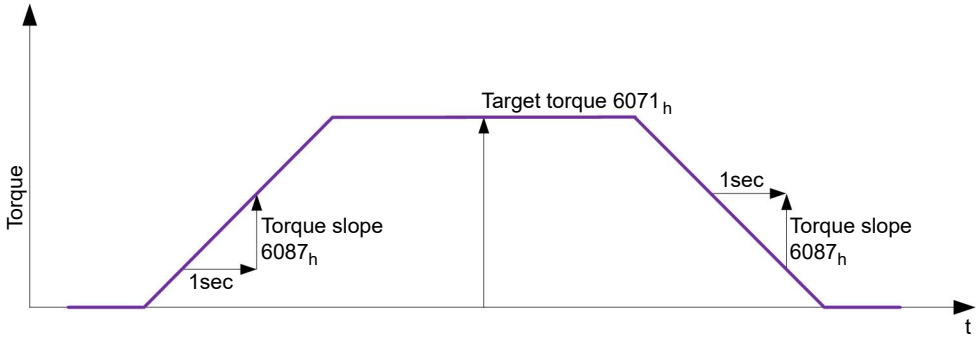
The following objects are also needed for this operating mode:

- 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 objects 607F$_h$ and 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.

#### 6.3.5.1 Objects of the ramp generator




#### 6.3.5.2 Torque curve




## 6.4 Homing

### 6.4.1 Overview

#### 6.4.1.1 Description

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

#### 6.4.1.2 Activation

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

**TIP**

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

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

### 6.4.1.3 Controlword

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

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

### 6.4.1.4 Statusword

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

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

**NOTICE**

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

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

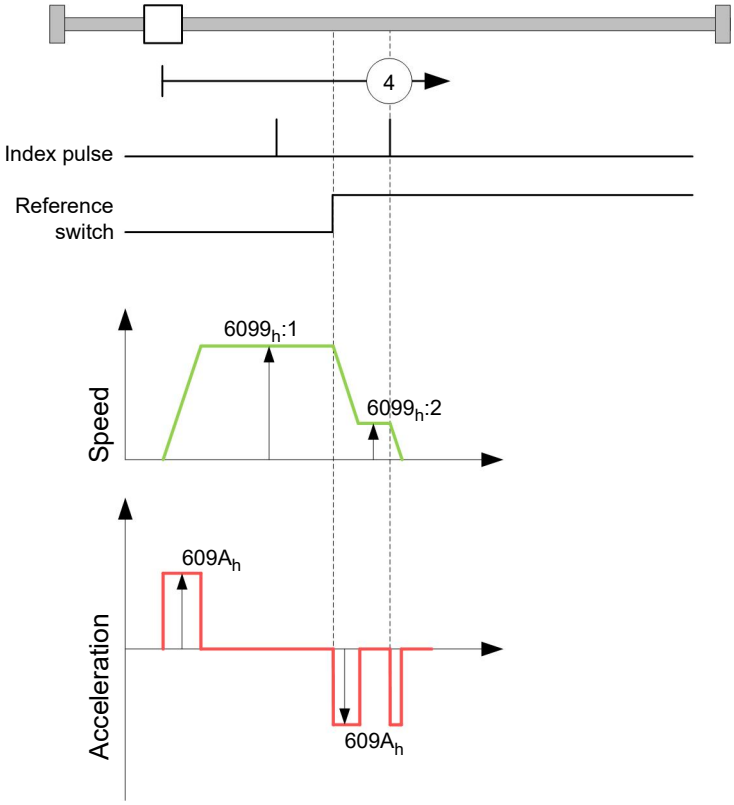
### 6.4.1.5 Object entries

The following objects are necessary for controlling this mode:

- $607C_h$ (Home Offset): Specifies the difference between the zero position of the controller and the reference point of the machine in user-defined units.
- $6098_h$ (Homing Method):
  Method to be used for referencing (see "Homing method")
- $6099_h$:$01_h$ (Speed During Search For Switch):
  Speed for the search of the switch
- $6099_h$:$02_h$ (Speed During Search For Zero):
  Speed for the search of the index
- $6080_h$ (Max Motor Speed): Maximum speed
- $609A_h$ (Homing Acceleration):
  Starting acceleration and braking deceleration for homing
- $3219_h$:$05_h$ (Block Detection Threshold):
  Minimum current threshold which, if exceeded, is to detect the blocking of the motor at a block.
- $3219_h$:$04_h$ (Block Detection Time):
  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:




## 6.4.2 Homing method

### 6.4.2.1 Description

The homing method is written as a number in object 6098h 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.

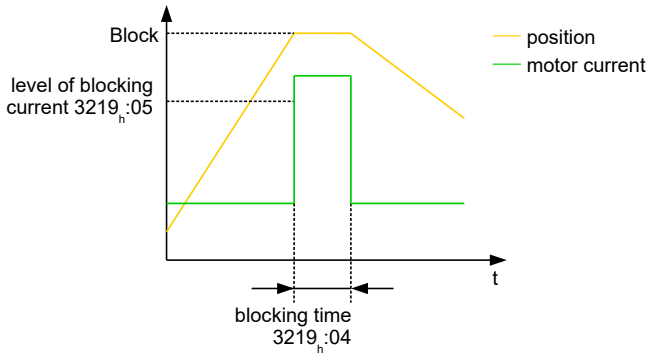
For methods that use homing on block, the same figures apply as for the methods with limit switch. Because nothing is different aside from the missing limit switches, the same figures are used. For the figures here, the limit switches must be replaced with a mechanical block.

### 6.4.2.2 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 3219h:05, the current level is defined above which movement against the block is detected.
2. Blocking duration: In object 3219h:04, the duration during which the motor moves against the block is set.




There is one exception for methods −17 and −18: after reaching the block, the direction is not reversed, but the motor remains energized in its position.

### 6.4.2.3 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 37 references to the current position.

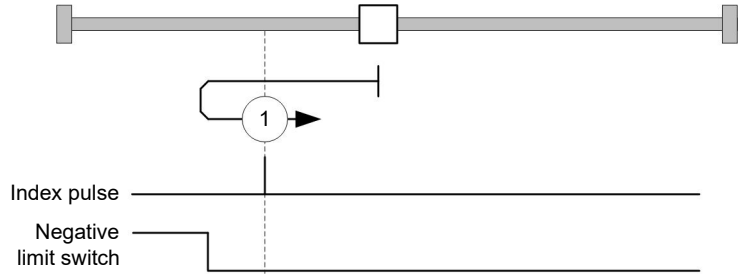
The following methods can be used for homing on block:

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

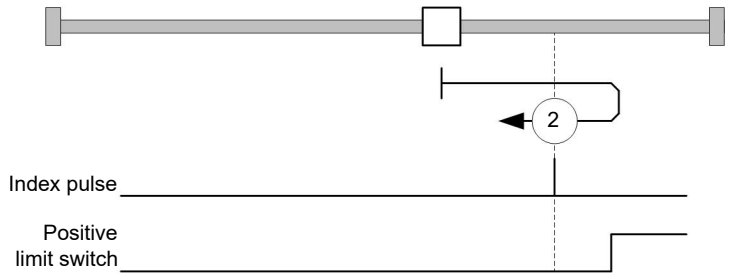
### 6.4.2.4 Methods 1 and 2

Reference to limit switches and index pulse.

Method 1 references to negative limit switch and index pulse:



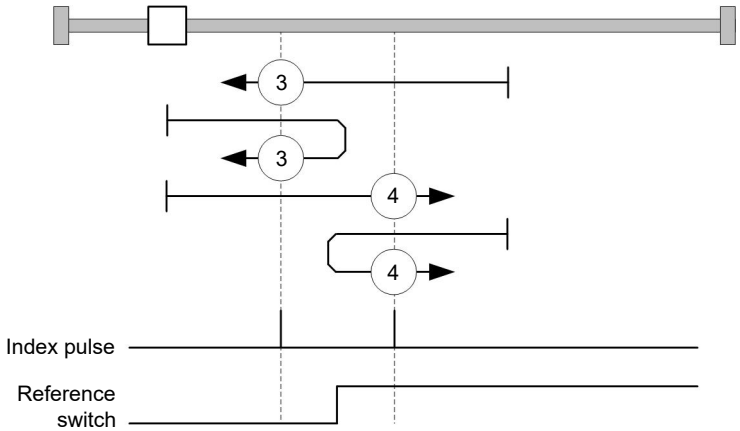

Method 2 references to positive limit switch and index pulse:



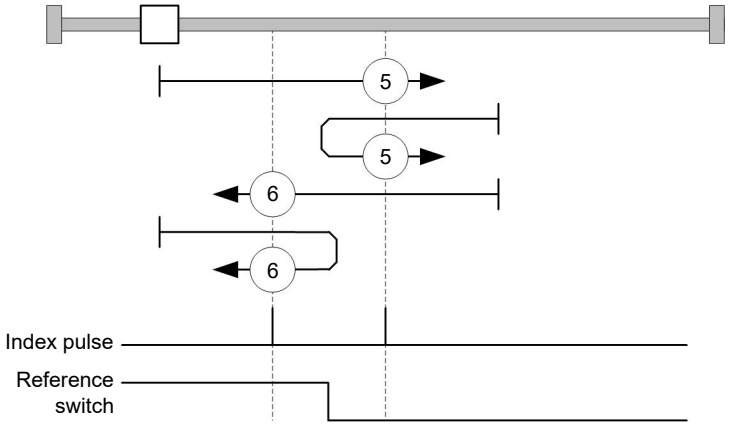

### 6.4.2.5 Methods 3 to 6

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:

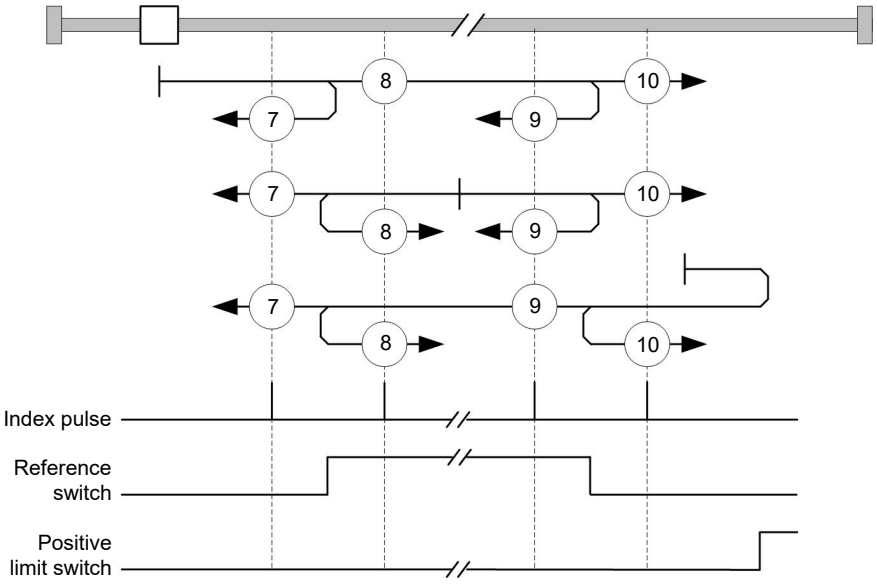



### 6.4.2.6 Methods 7 to 14

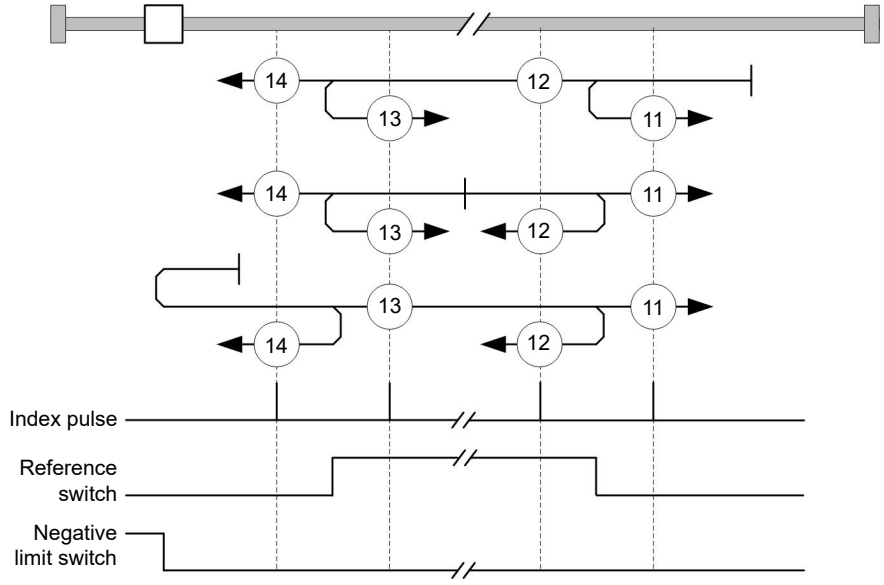
Reference to the home switch and index pulse (with limit switches).

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:



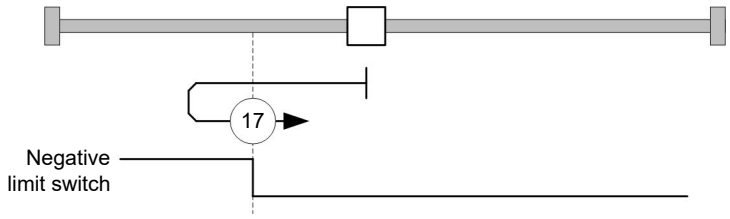

Methods 11 to 14 take the negative limit switch into account:



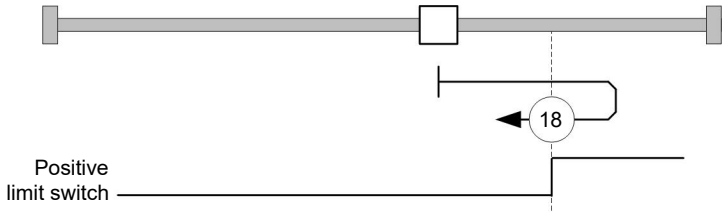

### 6.4.2.7 Methods 17 and 18

Reference to the limit switch without the index pulse.

Method 17 references to the negative limit switch:



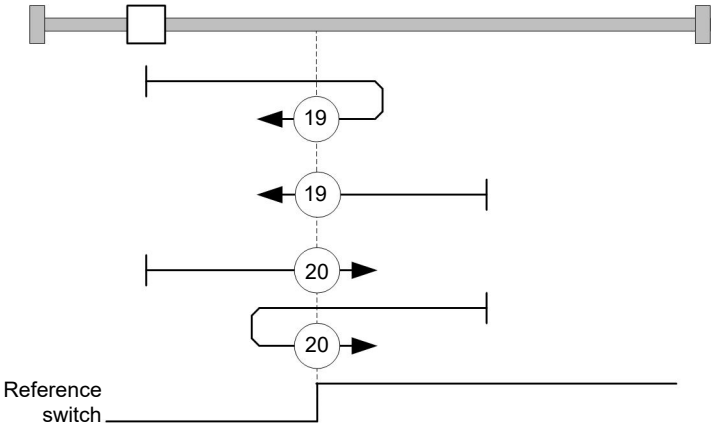

Method 18 references to the positive limit switch:



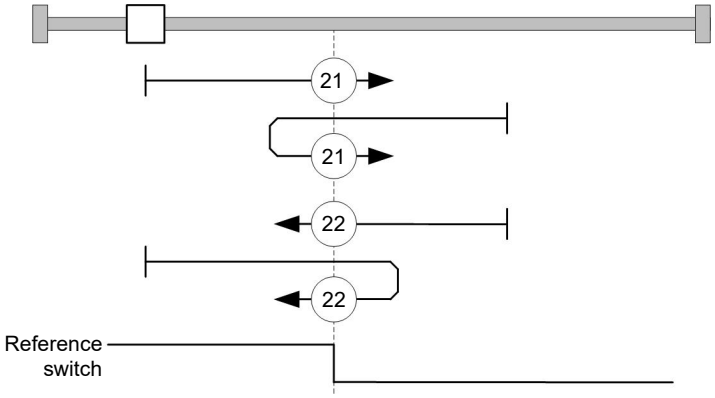

### 6.4.2.8 Methods 19 to 22

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:

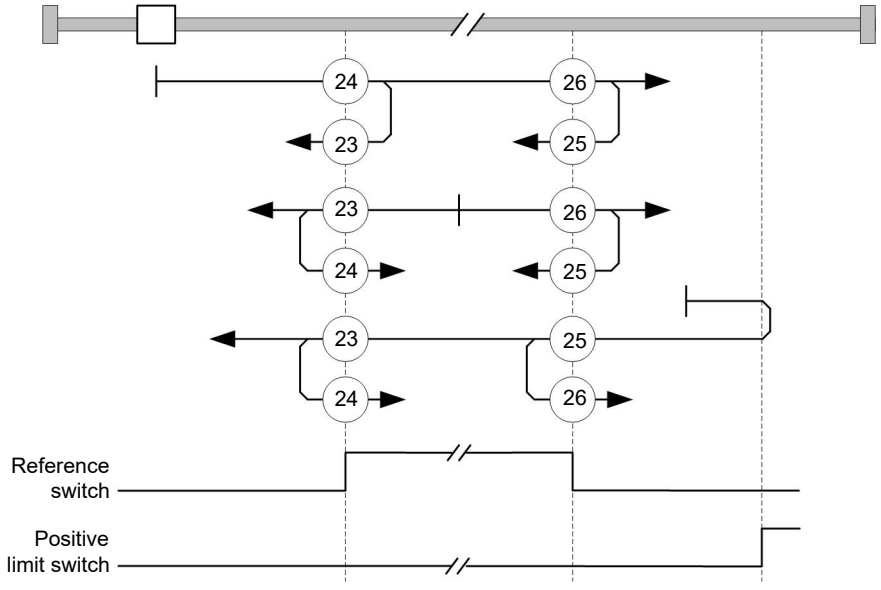



### 6.4.2.9 Methods 23 to 30

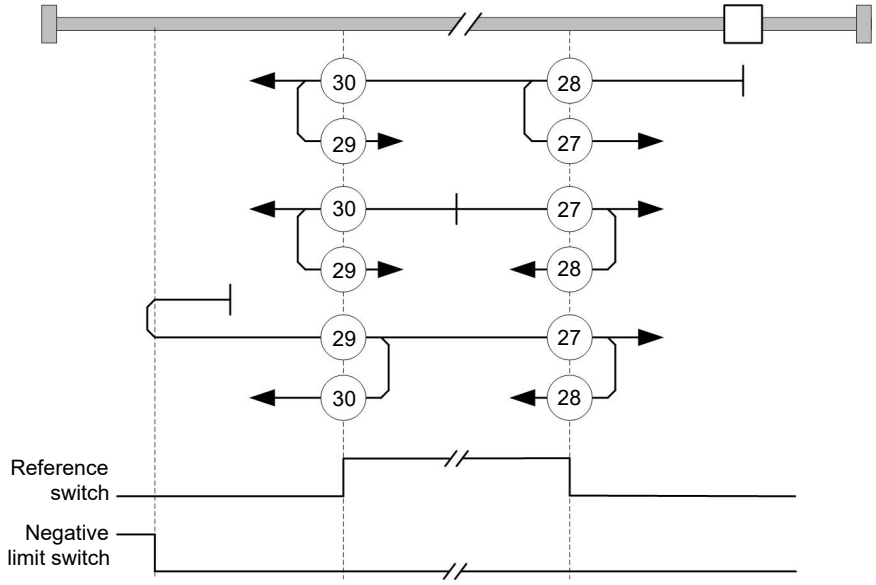
Reference to the home switch without the index pulse (with limit switches).

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:



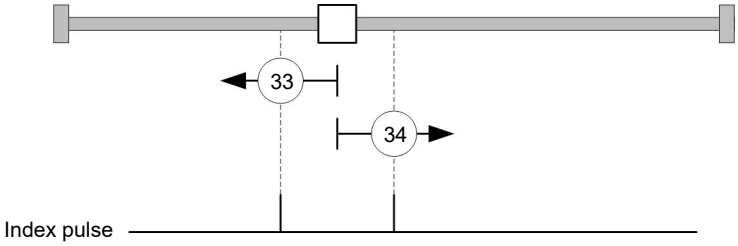

Methods 27 to 30 take the negative home switch into account:




### 6.4.2.10 Methods 33 and 34

Reference to the next index pulse.

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




### 6.4.2.11 Method 37

References to the current position.

**NOTICE**

For homing mode 37, 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 37.

## 6.5 Interpolated Position Mode

### 6.5.1 Overview

#### 6.5.1.1 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.

#### 6.5.1.2 Synchronization with the SYNC object

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

**NOTICE**

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

### 6.5.2 Activation

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

### 6.5.3 Controlword

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

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

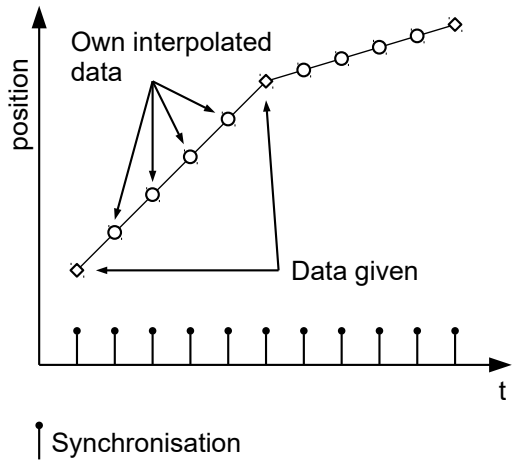
### 6.5.4 Statusword

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

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

### 6.5.5 Use

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




In the current implementation, only

- linear interpolation
- and a target position

are supported.

### 6.5.6 Setup

The following setup is necessary:

- $60C2_h$: Time between two passed target positions (factory settings) in ms.

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

### 6.5.7 Operation

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

## 6.6 Cyclic Synchronous Position

### 6.6.1 Overview

#### 6.6.1.1 Description

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

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

**NOTICE**

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

#### 6.6.1.2 Synchronization with the SYNC object

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

**NOTICE**

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

#### 6.6.1.3 Activation

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

#### 6.6.1.4 Controlword

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

#### 6.6.1.5 Statusword

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

| Bit | Value | Description |
|---|---|---|
| 8 | 0 | The controller is not in sync with the fieldbus |

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

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

### 6.6.2 Object entries

The following objects are necessary for controlling this mode:

- $607A_h$ (Target Position): This object must be written cyclically with the position set value.
- $607B_h$ (Position Range Limit): This object contains the preset for an overrun or underrun of the position specification.
- $607D_h$ (Software Position Limit): This object defines the limitations within which the position specification ($607A_h$) must be located.
- $6065_h$ (Following Error Window): This object specifies a tolerance corridor in both the positive and negative direction from the set specification. If the actual position is outside of this corridor for longer than the specified time ($6066_h$), a following error is reported.
- $6066_h$ (Following Error Time Out): This object specifies the time range in milliseconds. If the actual position is outside of the position corridor ($6065_h$) for longer than this time range, a following error is triggered.
- $6085_h$ (Quick-Stop Deceleration): This object contains the braking deceleration for the case that a quick-stop is triggered.
- $605A_h$ (Quick-Stop Option Code): This object contains the option that is to be executed in the event of a quick-stop.
- Only if closed loop is activated: $6080_h$ (Max Motor Speed): Maximum speed
- $60C2_h$:$01_h$ (Interpolation Time Period): This object specifies the time of a *cycle*; a new set value must be written in $607A_h$ in these time intervals.
  The following applies here: cycle time = value of $60C2_h$:$01_h$ * $10^{\text{value of 60C2:02}}$ seconds.
- $60C2_h$:$02_h$ (Interpolation Time Index): This object specifies the time basis of the cycles. Currently, only value $60C2_h$:$02_h$=-3 is supported; this yields a time basis of 1 millisecond.
- $60B0_h$ (Position Offset): Offset for the position set value in user-defined units
- $60B1_h$ (Velocity Offset): Offset for the speed set value in user-defined units
- $60B2_h$ (Torque Offset): Offset for the torque set value in tenths of a percent

The following objects can be read in this mode:

- $6064_h$ (Position Actual Value)
- $606C_h$ (Velocity Actual Value)
- $60F4_h$ (Following Error Actual Value)

## 6.7 Cyclic Synchronous Velocity

### 6.7.1 Overview

#### 6.7.1.1 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.

#### 6.7.1.2 Activation

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

#### 6.7.1.3 Controlword

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

#### 6.7.1.4 Statusword

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

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

### 6.7.2 Object entries

The following objects are necessary for controlling this mode:

- $60FF_h$ (Target Velocity): This object must be written cyclically with the speed set value.
- $6085_h$ (Quick-Stop Deceleration): This object contains the braking deceleration for the case that a quick-stop is triggered (see "CiA 402 Power State Machine").
- $605A_h$ (Quick-Stop Option Code): This object contains the option that is to be executed in the event of a quick-stop (see "CiA 402 Power State Machine").
- $6080_h$ (Max Motor Speed): Maximum speed
- $60C2_h:01_h$ (Interpolation Time Period): This object specifies the time of a *cycle*; a new set value must be written in $60FF_h$ in these time intervals.
  The following applies here: cycle time = value of $60C2_h:01_h * 10^{\text{value of 60C2:02}}$ seconds.
- $60C2_h:02_h$ (Interpolation Time Index): This object specifies the time basis of the cycles. Currently, only value $60C2_h:02_h$=-3 is supported; this yields a time basis of 1 millisecond.
- $60B1_h$ (Velocity Offset): Offset for the speed set value in user-defined units
- $60B2_h$ (Torque Offset): Offset for the torque set value in tenths of a percent

The following objects can be read in this mode:

- $606C_h$ (Velocity Actual Value)
- $607E_h$ (Polarity)

## 6.8 Cyclic Synchronous Torque

### 6.8.1 Overview

#### 6.8.1.1 Description

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

**NOTICE**

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

#### 6.8.1.2 Activation

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

#### 6.8.1.3 Controlword

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

#### 6.8.1.4 Statusword

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

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

### 6.8.2 Object entries

The following objects are necessary for controlling this mode:

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

The following objects can be read in this mode:

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

## 6.9 Clock-direction mode

### 6.9.1 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.

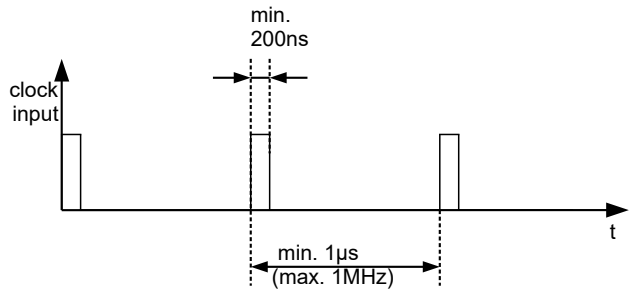
### 6.9.2 Activation

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

### 6.9.3 General

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

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




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

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

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

**NOTICE**

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

**NOTICE**

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

### 6.9.4 Statusword

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

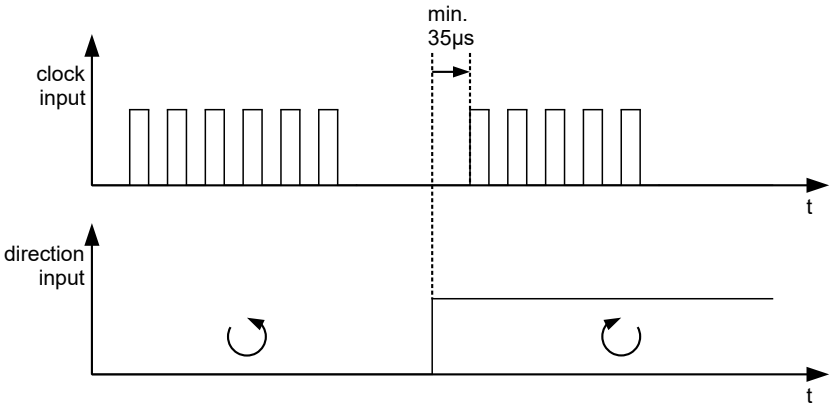
- Bit 13 (Following Error): This bit is set in *closed loop* mode if the following error is greater than the set limits ($6065_h$ (Following Error Window) and $6066_h$ (Following Error Time Out)).

### 6.9.5 Subtypes of the clock-direction mode

#### 6.9.5.1 Clock-direction mode (TR mode)

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

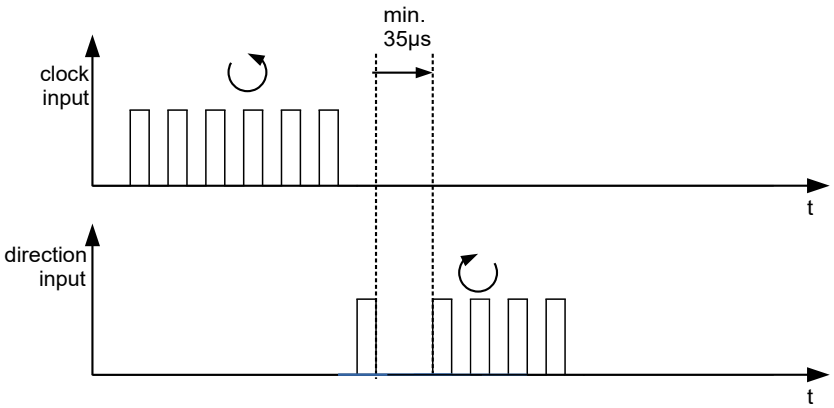
In this mode, the pulses must be preset via the clock input; the signal of the direction input specifies the direction of rotation here (see following graphic).




#### 6.9.5.2 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).




**NOTICE**

To avoid any interference from influencing the clock signal, do not leave the currently unsued pin floating, connect it to GND.

## 6.10 Auto setup

### 6.10.1 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.

### 6.10.2 Activation

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

### 6.10.3 Controlword

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

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

### 6.10.4 Statusword

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

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

# 7 Special functions

## 7.1 Digital inputs and outputs

This product is equipped with digital inputs and outputs. You can find the exact number for the given product variant in chapter Pin assignment.

In 323Ah User Pin Settings, you configure the hardware as follows:

- Subindex $01_h$: Here, you define the level for the inputs/outputs:
  - Value "0": 5 V
  - Value "1": 24 V (inputs) or +UB_Logic (outputs)

**NOTICE**

For the inputs, always use a voltage that is smaller than the operating voltage +UB_Logic.

- Subindex $02_h$: Here, you define the wiring for the digital inputs:
  - Value "0" (Pull-Down): High level at 5/24 V on the pin.
  - Value "1" (Pull-Up): High level without external voltage on the pin.

### 7.1.1 Digital inputs

#### 7.1.1.1 Overview

**NOTICE**

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

**NOTICE**

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

| Input | Special function | Switching threshold switchable (see $323A_h$) | Differential / single-ended |
|---|---|---|---|
| 1 | Negative limit switch | yes, 5 V or 24 V (=UB_Logic) | single-ended |
| 2 | Positive limit switch | yes, 5 V or 24 V (=UB_Logic) | single-ended |
| 3 | Home switch | yes, 5 V or 24 V (=UB_Logic) | single-ended |
| 4 | None | yes, 5 V or 24 V (=UB_Logic) | single-ended |
| 5 | None | yes, 5 V or 24 V (=UB_Logic) | single-ended |
| 6 | None | yes, 5 V or 24 V (=UB_Logic) | single-ended |

#### 7.1.1.2 Computation of the inputs

Object $60FD_h$ (Digital Inputs) contains a summary of the inputs and the special functions. The current status of the inputs is likewise read out from object 324Ah Inputs (including Hall sensors and incremental encoders, if present).

The following table lists the value of the corresponding bit in the respective object for the inputs depending on the configuration in 323Ah User Pin Settings:

| Voltage at pin | Subindex 02 (Pull-Up Enable) | Subindex 01 (Voltage Level Select) | Bit value |
|---|---|---|---|
| n.c | 0 (Pull-Down) | X | 0 |
| GND | 0 (Pull-Down) | X | 0 |
| 5 V | 0 (Pull-Down) | 0 (5 V) | 1 |
| 5 V | 0 (Pull-Down) | 1 (24 V) | 0 |
| 24 V | 0 (Pull-Down) | 1 (24 V) | 1 |
| n.c. | 1 (Pull-Up) | X | 1 |
| GND | 1 (Pull-Up) | X | 0 |
| 5 V | 1 (Pull-Up) | 0 (5 V) | 1 |
| 5 V | 1 (Pull-Up) | 1 (24 V) | 0 |
| 24 V | 1 (Pull-Up) | 1 (24 V) | 1 |

#### 7.1.1.3 Special functions

The firmware evaluates the following bits in 60FD$_h$:

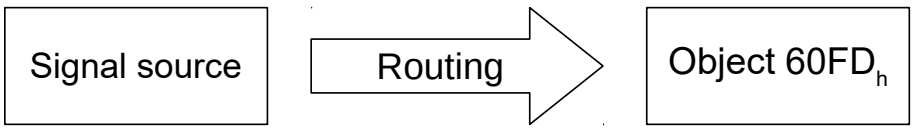
- Bit 0: Negative limit switch (see Limitation of the range of motion)
- Bit 1: Positive limit switch (see Limitation of the range of motion)
- Bit 2: Home switch (see Homing)
- Bit 3: Interlock (see interlock function)

You define the assignment of the bits to the pins with the *Input Routing*.

#### 7.1.1.4 Input Routing

##### Principle

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




##### Routing

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

| 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 |

| Number | | Signal source |
|---|---|---|
| dec | hex | |
| 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" |
| 71 | 47 | USB Power Signal |
| 81 | 51 | Negative block |
| 82 | 52 | Positive block |
| 90 | 5A | Analog input 1 |
| 91 | 5B | Analog input 2 |
| 100 | 64 | STO_IN_A |
| 101 | 65 | STO_IN_B |

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

| Number | | Signal source |
|---|---|---|
| dec | hex | |
| 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 |

| Number | | Signal source |
|---|---|---|
| dec | hex | |
| 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" |
| 199 | C7 | Inverted USB power signal |
| 218 | DA | Inverted analog input 1 |
| 219 | DB | Inverted analog input 2 |
| 228 | E4 | Inverted STO_IN_A |
| 229 | E5 | Inverted STO_IN_B |

**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".

#### 7.1.1.5 Interlock function

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

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

**Example**

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

1. To route input 4 to bit 3, set 3242$_h$:04$_h$ to "4".

### 7.1.2 Digital outputs

#### 7.1.2.1 Outputs

The outputs are controlled via object 60FE$_h$. Here, output 1 corresponds to bit 16 in object 60FE$_h$, output 2 corresponds to bit 17, etc., as with the inputs. The outputs with special functions are 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.

#### 7.1.2.2 Wiring

**NOTICE**

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

**NOTICE**

To use the digital outputs, you must connect a voltage (12…30 V DC) to pins 1 and 2 of X2 — voltage supply (logic supply).

The output voltage then corresponds to

- this logic voltage if you set $323A_{hh}:01_h$ to "1",
- 5 V if you set $323A_h:01_h$ to "0" (factory setting).

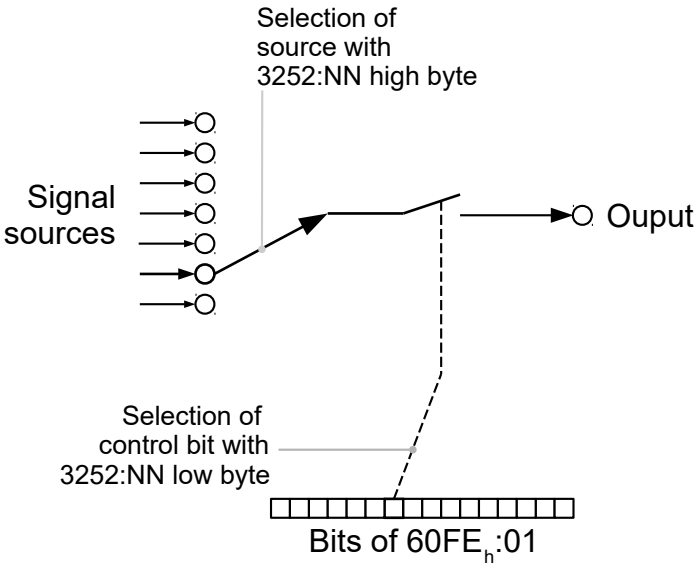
The current should not exceed 100 mA.

### 7.1.2.3 Output Routing

#### Principle

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

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




#### Routing

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

| Subindex $3252_h$ | Output Pin |
|---|---|
| $01_h$ | Configuration of the brake output (PWM output) |
| $02_h$ | Configuration of output 1 |
| $03_h$ | Configuration of output 2 (if available) |
| … | … |
| $0n_h$ | Configuration of output n (if available) |

**NOTICE**

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

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

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

| Number in 3252:01 to 0n | |
|---|---|
| $00XX_h$ | Output is always "1" |
| $01XX_h$ | Output is always "0" |
| $02XX_h$ | Encoder signal ($6063_h$) with frequency divider 1 |
| $03XX_h$ | Encoder signal ($6063_h$) with frequency divider 2 |
| $04XX_h$ | Encoder signal ($6063_h$) with frequency divider 4 |
| $05XX_h$ | Encoder signal ($6063_h$) with frequency divider 8 |
| $06XX_h$ | Encoder signal ($6063_h$) with frequency divider 16 |
| $07XX_h$ | Encoder signal ($6063_h$) with frequency divider 32 |
| $08XX_h$ | Encoder signal ($6063_h$) with frequency divider 64 |
| $09XX_h$ | Position Actual Value ($6064_h$) with frequency divider 1 |
| $0AXX_h$ | Position Actual Value ($6064_h$) with frequency divider 2 |
| $0BXX_h$ | Position Actual Value ($6064_h$) with frequency divider 4 |
| $0CXX_h$ | Position Actual Value ($6064_h$) with frequency divider 8 |
| $0DXX_h$ | Position Actual Value ($6064_h$) with frequency divider 16 |
| $0EXX_h$ | Position Actual Value ($6064_h$) with frequency divider 32 |
| $0FXX_h$ | Position Actual Value ($6064_h$) with frequency divider 64 |
| $10XX_h$ | PWM signal that is configured with object $2038_h$:$05_h$ and $06_h$ for brake output |
| $12XX_h$ | PWM signal that is configured with object $3260_h$ |
| $13XX_h$ | PWM signal that is configured with object $3261_h$ |
| $20XX_h$ | Pulse generator |

**NOTICE**

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

**Example**

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

- $3252_h$:$02_h$ = $0405_h$ ($04XX_h$ + $0005_h$)
- $04XX_h$: Encoder signal with frequency divider 4
- $0005_h$: Selection of bit 5 of 60FE:01

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

**Example**

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

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

### 7.1.3 Pulse generator

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

For example, you can use the signals to monitor position values or feed them into another controller as clock signals. Due to hardware limitations, the signals are not suitable for deriving a speed, as this would result in periodic speed interruptions.

#### Activating the function of the pins

To activate the function, use *Output Routing* and set the source in the corresponding subindices of $3252_h$ to $20XX_h$.

#### Selecting the type of output signals

You can select one of the following types in object $205C_h02_h$:

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

#### Selecting the source of the position data

The position data of one of the existing feedbacks are reproduced via the pulse generator output. To select the source, set bit 3 in the corresponding subindex of object $3203_h$ to "1".

If you do not set a bit, the value from 205C:$01_h$ is used (max. 255 pulses). In this case, the pulses are sent independent of the motor movement.

#### Setting the resolution of the output signals

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

## 7.2 Analog inputs

The controller has 2 analog inputs with 12-bit resolution. They are located on pins 16 and 18 of X4 – inputs and outputs..

You can read out the analog value in a NanoJ program and use it as you like, e. g., to specify the target speed.

### 7.2.1 Object entries

To read out and, if necessary, manipulate the value of the analog input, use the following OD settings:

- $3220_h$ (Analog Inputs):
  This object displays the instantaneous values of the analog inputs in *ADC digits*.
- $3320_h$ (Read Analogue Input):
  This object displays the instantaneous values of the analog inputs in user-defined units.

- $3321_h$ (Analogue Input Offset):
  This is the offset that is added to the read analog value ($3220_h$) before scaling (multiplier from object $3322_h$ and divisor from object $3323_h$).
- $3322_h$(Analogue Input Factor Numerator):
  This is the value by which the read analog value ($3220_h$ + 3321) is multiplied before it is written in object $3320_h$.
- $3323_h$(Analogue Input Factor Denominator):
  This is the value by which the read analog value ($3220_h$ + $3321_h$) is divided before it is written in object $3320_h$.

### 7.2.2 Scale analog value

You read the value in object $3320_h$ (Read Analogue Input): This object displays the instantaneous values of the analog inputs in user-defined units.

The user-defined units are made up of offset ($3321_h$) and scaling value ($3322_h$/ $3323_h$). If both are still set to the default values, the value in $3320_h$ is specified in the *millivolt* unit.

## 7.3 Automatic brake control

### 7.3.1 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.

### 7.3.2 Activation and connection

The brake can be controlled either automatically or manually:

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

**NOTICE**

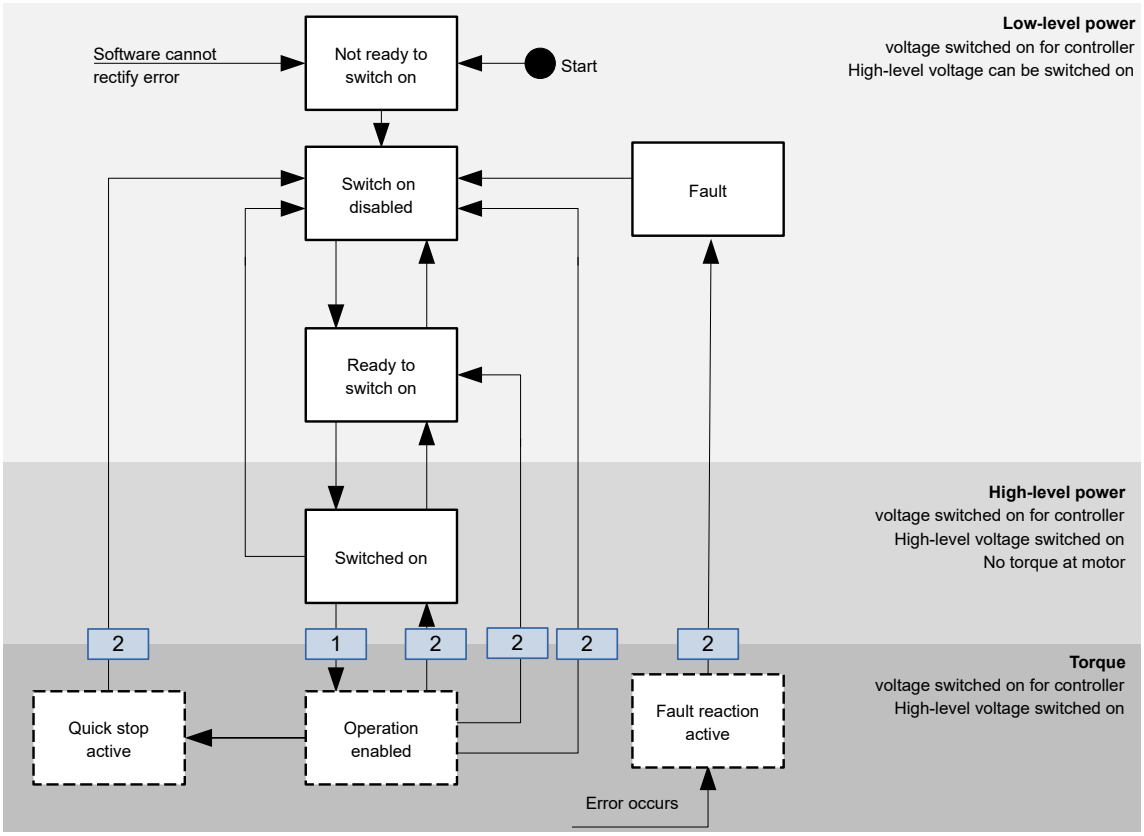
Firmware Version v2602: Default value of 60FEh Digital Outputs Subindex $01_h$: changed from "0" to "1", external holding brake is mechanically closed.

#### 7.3.2.1 Connection

The brake output is located at X4 – inputs and outputs.

### 7.3.3 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.

## 7.3.4 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.he maximum voltage at the *Brake +* pin equals the UB_Logic you have connected and should not be below 17 V.

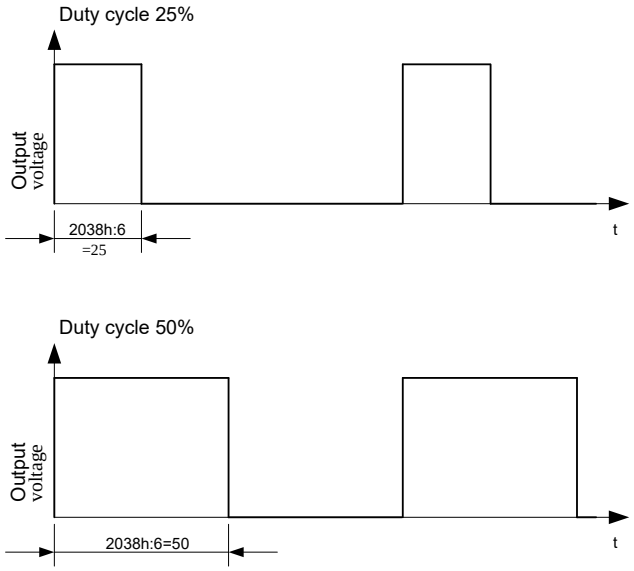
### 7.3.4.1 Frequency

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

### 7.3.4.2 Duty cycle

The duty cycle – the ratio of pulse to period duration – is set in $2038_h$:$6_h$. The value is a percentage and can be selected between 1 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.




## 7.4 External brake chopper (ballast)

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 a connection at pins 3 / 4 of X2 — voltage supply for an external brake chopper.

### 7.4.1 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.

### 7.4.2 Activating the brake chopper

To activate the brake chopper, set bit 0 in $4021_h$:$01_h$ to "1". To use the external brake chopper, set bit 2 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 $4021_h$:$03_h$.

If, in spite of the activation, the brake chopper 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.

### 7.4.3 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 for the external resistance, you must ascertain or determine the following resistor parameters from the data sheet of the ballast resistor and enter them in the corresponding subindex of $4021_h$:

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

Rated value of the ballast resistor

**Cooling Power $P_{Stat_TA_Max}$, [mW]**

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

$$P_{Stat_TA_Max} = (T_{Ballast_Max} - TA_{Max}) / R_{th,A}$$

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

**Short Term Energy Limit $E_{ST_25°C}$, [mWs]**

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

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

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

**Long Term Energy Limit $E_{LT_TA_Max}$, [mWs]**

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

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

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

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

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

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

## 7.5 $I^2t$ Motor overload protection

### 7.5.1 Description

**NOTICE**

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

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

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

## 7.5.2 Object entries

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

- $6075_h$ Motor Rated Current – specifies the rated current in mA.
- $6073_h$ Max Current – specifies the maximum current in tenths of a percent of the set rated current.
- $3219_h$:$01_h$ $I^2t$ Duration Of Max Current - specifies the maximum duration of the maximum current in ms.

## 7.5.3 Activation

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

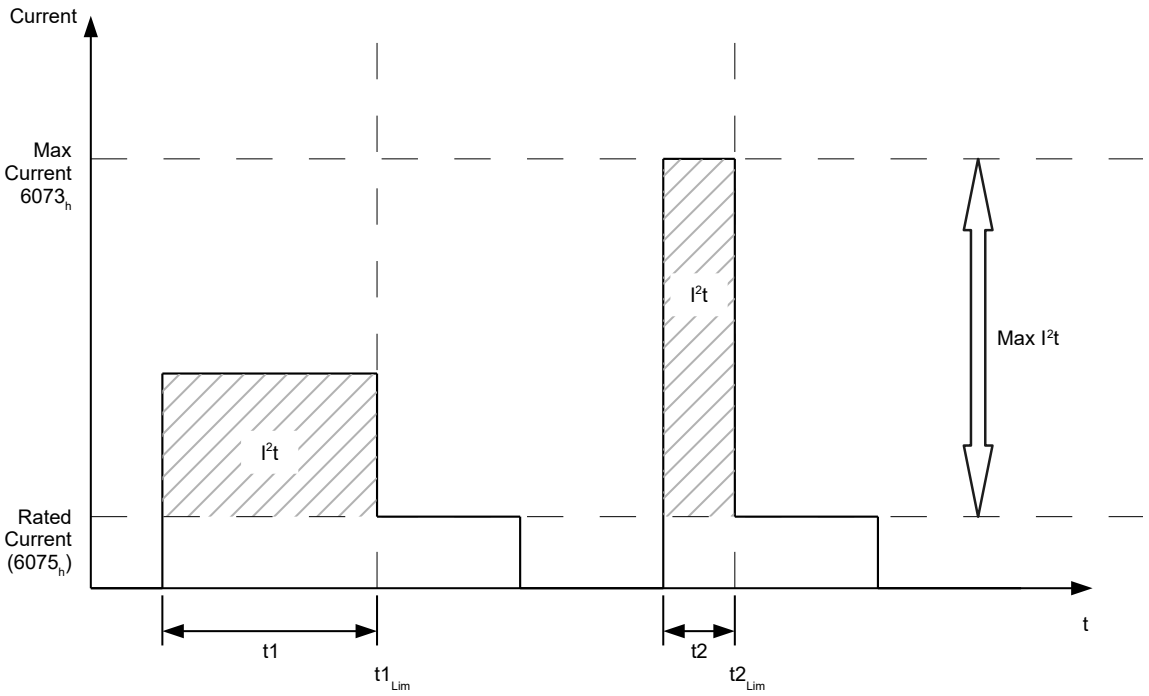
To activate the mode, you must describe the three object entries mentioned above in a meaningful way. This means that the maximum current must be greater than the rated current and a time value for the maximum duration of the maximum current must be entered. If these conditions are not met, the $I^2t$ functionality remains deactivated.

## 7.5.4 Function of $I^2t$

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

The motor can run with maximum current until the calculated $I^2t_{Lim}$ is reached. The current is then immediately reduced to the rated current. The maximum current is limited by the maximum motor current ($6073_h$).

The relationships are illustrated again in the following diagrams.




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

## 7.6 Saving objects

**NOTICE**

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

**NOTICE**

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

### 7.6.1 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.
- User: Parameters that are written and read by the customer/user only and are ignored by the controller firmware.
- Motor: Parameters related to the motor (BLDC/stepper, motor current, *closed/open-loop*...). Some are set and saved by auto setup.
- Sensor: Parameters related to sensor/encoder that are set either by auto setup or that can be found in the data sheets.
- CANopen: Parameters related to CANopen 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.

### 7.6.2 Category: Communication

- $1005_h$: COB-ID Sync
- $1006_h$: Communication Cycle Period
- $1007_h$: Synchronous Window Length
- $100C_h$: Guard Time
- $100D_h$: Live Time Factor
- $1014_h$: COB-ID EMCY
- $1016_h$: Consumer Heartbeat Time
- $1017_h$: Producer Heartbeat Time
- $1019_h$: Synchronous Counter Overflow Value
- $1029_h$: Error Behavior
- $1400_h$: Receive PDO 1 Communication Parameter
- $1401_h$: Receive PDO 2 Communication Parameter
- $1402_h$: Receive PDO 3 Communication Parameter
- $1403_h$: Receive PDO 4 Communication Parameter
- $1404_h$: Receive PDO 5 Communication Parameter

- 1405$_h$: Receive PDO 6 Communication Parameter
- 1406$_h$: Receive PDO 7 Communication Parameter
- 1407$_h$: Receive PDO 8 Communication Parameter
- 1600$_h$: Receive PDO 1 Mapping Parameter
- 1601$_h$: Receive PDO 2 Mapping Parameter
- 1602$_h$: Receive PDO 3 Mapping Parameter
- 1603$_h$: Receive PDO 4 Mapping Parameter
- 1604$_h$: Receive PDO 5 Mapping Parameter
- 1605$_h$: Receive PDO 6 Mapping Parameter
- 1606$_h$: Receive PDO 7 Mapping Parameter
- 1607$_h$: Receive PDO 8 Mapping Parameter
- 1800$_h$: Transmit PDO 1 Communication Parameter
- 1801$_h$: Transmit PDO 2 Communication Parameter
- 1802$_h$: Transmit PDO 3 Communication Parameter
- 1803$_h$: Transmit PDO 4 Communication Parameter
- 1804$_h$: Transmit PDO 5 Communication Parameter
- 1805$_h$: Transmit PDO 6 Communication Parameter
- 1806$_h$: Transmit PDO 7 Communication Parameter
- 1807$_h$: Transmit PDO 8 Communication Parameter
- 1A00$_h$: Transmit PDO 1 Mapping Parameter
- 1A01$_h$: Transmit PDO 2 Mapping Parameter
- 1A02$_h$: Transmit PDO 3 Mapping Parameter
- 1A03$_h$: Transmit PDO 4 Mapping Parameter
- 1A04$_h$: Transmit PDO 5 Mapping Parameter
- 1A05$_h$: Transmit PDO 6 Mapping Parameter
- 1A06$_h$: Transmit PDO 7 Mapping Parameter
- 1A07$_h$: Transmit PDO 8 Mapping Parameter
- 1F80$_h$: NMT Startup
- 2102$_h$: Fieldbus Module Control

### 7.6.3 Category: Application

- 2034$_h$: Upper Voltage Error Level [mV]
- 2035$_h$: Lower Voltage Error Level [mV]
- 2038$_h$: Brake Controller Timing
- 203D$_h$: Torque Window
- 203E$_h$: Torque Window Time Out
- 203F$_h$: Max Slippage Time Out
- 2057$_h$: Clock Direction Multiplier
- 2058$_h$: Clock Direction Divider
- 205B$_h$: Clock/Direction Or Clockwise/Counter-Clockwise Mode
- 205C$_h$: Pulse Generator Configuration
- 2290$_h$: PDI Control
- 2300$_h$: NanoJ Control
- 2800$_h$: Bootloader And Reboot Settings
- 30C2$_h$: Cut-off Frequency [mHz] To Filter The Velocity Demand Value Derived From Position Change
- 320D$_h$: Moment Of Inertia
- 3219$_h$: Current Configuration
- 321A$_h$: Current Controller Parameters
- 321B$_h$: Velocity Controller Parameters
- 321C$_h$: Position Controller Parameters
- 321D$_h$: Feedforward
- 321E$_h$: Voltage Limit

- 323A$_h$: User Pin Settings
- 3241$_h$: Digital Input Position Capture
- 3242$_h$: Digital Input Routing
- 3250$_h$: Digital Outputs Control
- 3252$_h$: Digital Output Routing
- 325A$_h$: Outputs
- 3260$_h$: Pwm Output 0
- 3261$_h$: Pwm Output 1
- 3262$_h$: Pwm Output 2
- 3273$_h$: Generic SPI Hardware Configuration
- 3274$_h$: Generic SPI Mosi Data
- 3321$_h$: Analog Input Offsets
- 3322$_h$: Analog Input Numerators
- 3323$_h$: Analog Input Denominators
- 3700$_h$: Deviation Error Option Code
- 3701$_h$: Limit Switch Error Option Code
- 4015$_h$: Special Drive Modes
- 4021$_h$: Ballast Configuration
- 6007$_h$: Abort Connection Option Code
- 6040$_h$: Controlword
- 605A$_h$: Quick Stop Option Code
- 605B$_h$: Shutdown Option Code
- 605C$_h$: Disable Operation Option Code
- 605D$_h$: Halt Option Code
- 605E$_h$: Fault Reaction Option Code
- 6060$_h$: Modes Of Operation
- 6065$_h$: Following Error Window
- 6066$_h$: Following Error Time Out
- 6067$_h$: Position Window
- 6068$_h$: Position Window Time
- 606D$_h$: Velocity Window
- 606E$_h$: Velocity Window Time
- 606F$_h$: Velocity Threshold
- 6070$_h$: Velocity Threshold Time
- 6071$_h$: Target Torque
- 6072$_h$: Max Torque
- 607A$_h$: Target Position
- 607B$_h$: Position Range Limit
- 607C$_h$: Home Offset
- 607D$_h$: Software Position Limit
- 607E$_h$: Polarity
- 607F$_h$: Max Profile Velocity
- 6080$_h$: Max Motor Speed
- 6081$_h$: Profile Velocity
- 6082$_h$: End Velocity
- 6083$_h$: Profile Acceleration
- 6084$_h$: Profile Deceleration
- 6085$_h$: Quick Stop Deceleration
- 6086$_h$: Motion Profile Type
- 6087$_h$: Torque Slope
- 6091$_h$: Gear Ratio
- 6092$_h$: Feed Constant
- 6096$_h$: Velocity Factor

- 6097$_h$: Acceleration Factor
- 6098$_h$: Homing Method
- 6099$_h$: Homing Speed
- 609A$_h$: Homing Acceleration
- 60A2$_h$: Jerk Factor
- 60A4$_h$: Profile Jerk
- 60A8$_h$: SI Unit Position
- 60A9$_h$: SI Unit Velocity
- 60B0$_h$: Position Offset
- 60B1$_h$: Velocity Offset
- 60B2$_h$: Torque Offset
- 60C1$_h$: Interpolation Data Record
- 60C2$_h$: Interpolation Time Period
- 60C4$_h$: Interpolation Data Configuration
- 60C5$_h$: Max Acceleration
- 60C6$_h$: Max Deceleration
- 60F2$_h$: Position Option Code
- 60F8$_h$: Max Slippage
- 60FE$_h$: Digital Outputs
- 60FF$_h$: Target Velocity

### 7.6.4 Category: User

- 2701$_h$: Customer Storage Area

### 7.6.5 Category: Motor

- 2030$_h$: Pole Pair Count
- 3202$_h$: Motor Drive Submode Select
- 6073$_h$: Max Current
- 6075$_h$: Motor Rated Current

### 7.6.6 Category: Sensor

- 3203$_h$: Feedback Selection
- 3380$_h$: Feedback Sensorless
- 3390$_h$: Feedback Hall
- 33A0$_h$: Feedback Incremental A/B/I 1
- 33B0$_h$: Feedback SSI 1
- 60E6$_h$: Additional Position Encoder Resolution - Encoder Increments
- 60EB$_h$: Additional Position Encoder Resolution - Motor Revolutions
- 60F1$_h$: Additional Encoder Inversion

### 7.6.7 Category: CANopen

- 2005$_h$: CANopen Baudrate
- 2007$_h$: CANopen Config
- 2009$_h$: CANopen NodeID

### 7.6.8 Starting the save process

**CAUTION!**

**Uncontrolled motor movements!**

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

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

**NOTICE**

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

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

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

| Subindex | Category |
|---|---|
| $01_h$ | All categories with the exception of $0A_h$ (CANopen) |
| $02_h$ | Communication |
| $03_h$ | Application |
| $04_h$ | User |
| $05_h$ | Motor |
| $06_h$ | Sensor |
| $0A_h$ | CANopen |

### 7.6.9 Discarding the saved data

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

The following subindices correspond to a *category* here:

| Subindex | Category |
|---|---|
| $01_h$ | All categories (reset to factory settings) with the exception of $06_h$ (Sensor) and $0A_h$ (CANopen) |
| $02_h$ | Communication |
| $03_h$ | Application |
| $04_h$ | User |
| $05_h$ | Motor |

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

| Subindex | Category |
|---|---|
| $06_h$ | Sensor |
| $0A_h$ | CANopen |

The saved objects are subsequently discarded; the change does not take effect until after the controller is restarted. You can restart the controller by entering the value "$746F6F62_h$" in $2800_h$:$01_h$.

**NOTICE**

- Objects of *category* $06_h$ (Sensor) are determined by Auto setup and are not reset when resetting to factory settings with subindex $01_h$ (thereby making it unnecessary to again perform an auto setup). You can reset these objects with subindex $06_h$.
- Objects of *category* $0A_h$ (CANopen) are not reset with subindex $01_h$.

### 7.6.10 Verifying the configuration

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

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

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

The following sequence makes verification possible:

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

After the controller is restarted, the master can check the value in $1020_h$:$01_h$ and 1020:$01_h$. If one of the values is "0", the object dictionary was changed after the saved values were loaded. This also applies for changes to the directory following a firmware update, e.g., if a stored object no longer exists. 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.

## 7.7 Touch Probe (capture)

With this function, the sensor position can be recorded and noted automatically if a level change occurs at the digital input of the home switch or at two other inputs. The position is recorded internally by the controller independent of the cycle time of the application or PLC.

In object $3241_h$, you determine for each of the three inputs the source (in absolute user units or in sensor increments) and the time at which the position is recorded:

- on rising edge
- on falling edge
- on both edges

# 8 CANopen

You can address the controller using CANopen. The controller can function in a network as a *slave*.

In this chapter, the services of the CANopen communication structure are described. The CANopen messages are individually broken down.

CANopen references: www.can-cia.org

- *CiA 301 CANopen application layer and communication profile - Application layer and communication profile*, Date: 21.02.2011, Version: 4.2.0
- *CiA 402 Device profile for drives and motion control - Part 1: General definitions*, Date: 14.12.2007, Version: 3.0.0
- *CiA 402 Drives and motion control device profile - Part 2: Operation modes and application data*, Data 14.12.2007, Version: 3.0
- *CiA 402 Drives and motion control device profile - Part 3: PDO mapping*, Date 14.12.2007, Version: 3.0
- *CiA 306 Electronic device description - Part 1: Electronic Data Sheet and Device Configuration File*, Date: 08.02.2012, Version: 1.3.5
- *CiA 305 Layer setting services (LSS) and protocols*, Date: 08.05.2013, Version: 3.0.0

## 8.1 General

**TIP**

- Only 11-bit CAN-IDs are currently supported.
- With CANopen, the data are always sent over the bus in little-endian format.

### 8.1.1 CAN message

CAN messages are described in this chapter; these are written in the following format:

```
583 | 41 09 10 00 1E 00 00 00
```

```
183R | DLC=0
```

The individual messages are written as follows:

- All numbers are written in hexadecimal notation; due to the abbreviated notation, the leading `0x` is omitted.
- Normal data message: The CAN-ID is prefixed to the CAN message; in the above example, the ID `583` (i.e., $583_h$ or $1411_d$). The data and the CAN-ID are separated from the data with a pipe character.
- RTR message (remote transmission request): If an `R` follows the CAN-ID instead of the data, the length of the *DLC* (Download Content) is specified. In the above example, the length of the *DLC* is 0.

## 8.2 CANopen services

The CANopen stack offers the services listed in the following table; more detailed descriptions can be found in the respective chapters.

| Default CAN-ID | Service | Description in |
|---|---|---|
| $000_h$ | Network Management (NMT) | Section Network Management (NMT) |
| $080_h$ | Synchronization Object | Section Synchronization object (SYNC) |
| $080_h$+Node-ID | Emergency | Section Emergency Object (EMCY) |
| $180_h$+Node-ID | TX Process Data Objects (PDO) | Section Process Data Object (PDO) |
| $200_h$+Node-ID | RX Process Data Objects (PDO) | |
| $280_h$+Node-ID | TX Process Data Objects (PDO) | |
| $300_h$+Node-ID | RX Process Data Objects (PDO) | |

| Default CAN-ID | Service | Description in |
|---|---|---|
| $380_h$+Node-ID | TX Process Data Objects (PDO) | |
| $400_h$+Node-ID | RX Process Data Objects (PDO) | |
| $480_h$+Node-ID | TX Process Data Objects (PDO) | |
| $500_h$+Node-ID | RX Process Data Objects (PDO) | |
| $580_h$+Node-ID | TX Service Data Objects (SDO) | Section Service Data Object (SDO) |
| $600_h$+Node-ID | RX Service Data Objects (SDO) | |
| $700_h$+Node-ID | BOOT-UP Protocol | Section Boot-Up Protocol |
| $700_h$+Node-ID | Nodeguarding and Heartbeat | Section Heartbeat and Nodeguarding |

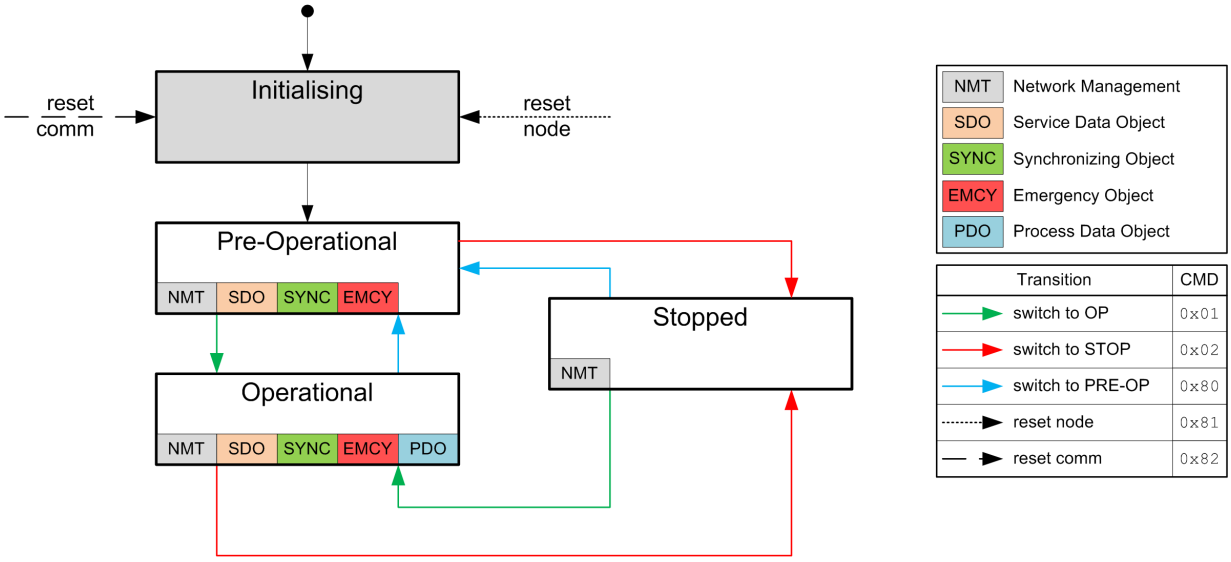
### 8.2.1 Network Management (NMT)

The network management follows a master-slave structure. NMT requires a CANopen device in the network that performs the role of the CANopen master.

All other devices have the role of the NMT slave. Each NMT slave can be addressed via its individual node-ID in the range from [1–127]. NMT services can be used to initiate, start, monitor, reset or stop CANopen devices.

In doing so, the controller follows the state diagram shown in the following figure. The "Initialization" state is only reached after switching on or by sending a "Reset Communication" or "Reset Node" NMT command. The "Pre-Operational" state is automatically activated after initialization.

In object $1F80_h$, you can set whether the "Operational" state is automatically switched to afterwards, thereby allowing you to avoid sending an additional NMT command.

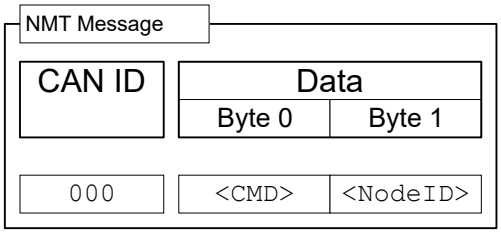



Shown in the following table is an overview of the activity of the services in the respective states.

Note that the *Stopped* state stops communication completely and only permits controller of the NMT state machine.

| Service | Initializing | Pre-Operational | Operational | Stopped |
|---|---|---|---|---|
| PDO | | | Active | |
| SDO | | Active | Active | |
| SYNC | | Active | Active | |
| EMCY | | Active | Active | |
| BOOT-UP | Active | | | |
| NMT | | Active | Active | Active |

The "Network Management" message has CAN-ID 0. A message is always two bytes long and has the following structure:




Here, the <CMD> corresponds to one of the following bytes (see also the legend in the figure of the NMT state diagram):

| <CMD> | Meaning |
|---|---|
| $01_h$ | Switch to the "Operational" state |
| $02_h$ | Switch to the "Stop" state |
| $80_h$ | Switch to the "Pre-Operational" state |
| $81_h$ | Reset Node |
| $82_h$ | Reset Communication |

Completely restart the controller with the "Reset Node" command. Use the "Reset Communication" command to reset the CANopen settings and restart the communication. Wait after every reset for the boot-up message before you send messages to the controller again

The value for <Node-ID> can be $00_h$; in this case, the NMT command applies for all devices on the CAN bus (broadcast). If a number not equal to zero is used, only the device with the corresponding node-ID is addressed.

Example: If all devices on the CAN bus are to be switched to the "Stop" operating state, a broadcast with the "Switch to the Stop state" command can be used. The NMT message is structured as follows:

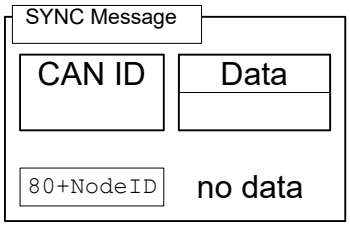
```
000 | 02 00
```

If only the device with node-ID 42 is to be completely restarted, the following CAN message is to be sent:

```
000 | 01 2A
```

### 8.2.2 Synchronization object (SYNC)

The Synchronization object is used to simultaneously validate the time of PDO data for all devices on the bus. The sync message is structured as follows:

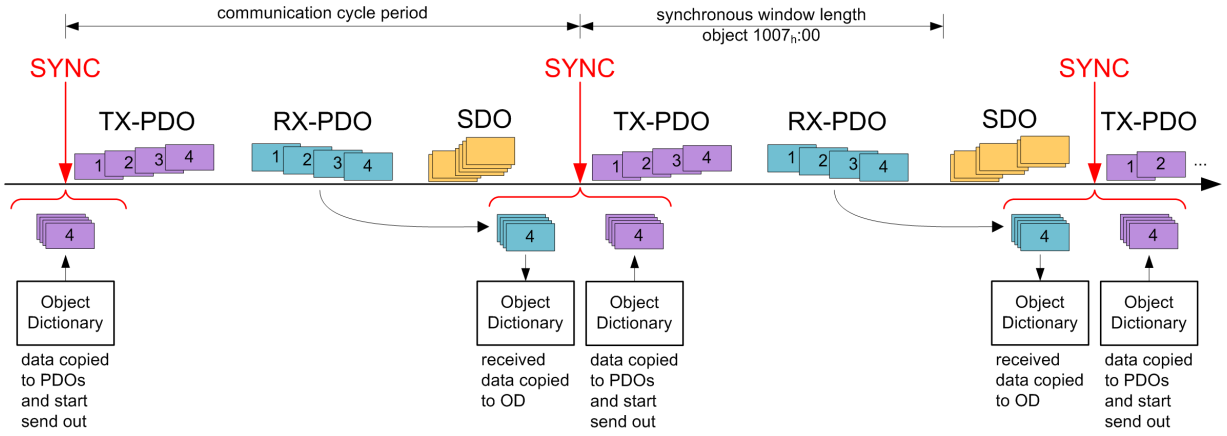



For SYNC operation, transmission mode (Transmission Type) 0 is usually used for the RX-PDOs (data are valid with the next SYNC); for TX-PDOs, a transmission mode between 1 and 240 is selected. (Details: see chapter Process Data Object (PDO)).

After receiving a SYNC message, there is a time window ("synchronous window") within which PDO messages can be sent and received. If the time of the window has elapsed, all devices must stop sending PDOs. The "synchronous window length" can be set in microseconds in object $1007_h$:$00_h$.

A typical CAN-SYNC operation is divided into four phases (see also the following figure):

1. The SYNC message is received. The previously received RX-PDO data are thereby copied to the object dictionary (if present). At that time, the data are also sampled and copied to the TX-PDOs and the sending of these messages initiated.
2. The TX-PDOs are then sent by all slaves on the bus.
3. Afterwards, the PDOs are sent by the CANopen master. After the "synchronous window length" time has elapsed, no further PDOs are permitted.
4. SDO messages can be exchanged at the latest when the "synchronous window" is closed again.




If the *Sync Producer* supports a *Sync Counter*, the sync message receives an additional 1-byte numerical value. This counter is increased by the value "1" per sent sync message and is reset each time the value 1019h Synchronous Counter Overflow Value is reached.

For each TX-PDO, a start value of the *Sync Counter* can be defined in subindex $06_h$ of the corresponding communication parameter (e.g., in $1800_h$:$06_h$) beginning with which the *slave* is to respond to the sync for the first time and send the PDO. The function is not activated until a value greater than 1 is set in $1019_h$.

## 8.2.3 Emergency Object (EMCY)

A message of type "Emergency" is sent whenever an error occurs in the controller that was not caused by an SDO access. This service is unconfirmed and is sent with CAN-ID $80_h$+Node-ID.

The emergency message is structured as follows:

Emergency Message (EMCY)

| CAN ID | Data | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | Byte 6 | Byte 7 |
| 80+NodeID | <EMCY Error Code> | | <E-REG> | <E-Number> | <0> | | | |

A total of three error codes are transferred here:

- the "Emergency Error Code" (`<EMCY Error Code>`)
- the content of the "Error Register" object ($1001_h$), `E-REG`
- the "Error Number" (`E-Number`)

### 8.2.3.1 Error handling

A module for error handling processes all errors that occur internally. Each error is classified into an error class.

Each error that occurs is handled as follows:

1. The bit that belongs to the error in the "Error Register" object ($1001_h$) is set.
2. Three pieces of information are then written together in the "Pre-defined Error Field" object ($1003_h$:01):
   - The *Emergency Error Code*
   - The *Error Register*
   - The manufacturer-specific error code
3. If no further errors are pending, the following message is sent:
   `80 + Node-ID | 00 00 E-REG E-Number 00 00 00 00`

In object $1029_h$, you can configure whether and how the controller is to change its *NMT state* in case of an error.

### 8.2.4 Service Data Object (SDO)

A "Service Data Object" permits read or write access of the object dictionary.

In the following, the owner of the object dictionary is referred to as the "server"; the CAN node – which wants to request or write the data – is referred to as the "client".

An "upload" refers to the reading of a value of an object from the object dictionary; a "download" refers to the writing of a value in the object dictionary. In addition, the following abbreviations are used in the diagrams:

- `<IDX>`: Index of the object that is to be read or written in the object dictionary; the LSB of the index is in byte 1 here. Example: The statusword of the controller has index $6041_h$; byte 1 is then written with $41_h$ and byte 2 with $60_h$. With Expedited Transfer, the SDO answer contains the same index as that of the request.
- `<SUBIDX>`: Subindex of the object in the object dictionary from $00_h$ to $FF_h$. With Expedited Transfer, the answer of the SDO message of the controller also contains the subindex of the request.

Because CAN messages of type SDO contain a large amount of metadata, you should only use SDO messages to configure the controller. Should it be necessary to cyclically exchange data during running operation, use CANopen messages of type PDO (see subsection Process Data Object).

The SDO transfers are divided into three types of access:

- "expedited transfer" for transferring objects with up to four bytes.
- "normal transfer" for transferring any number of bytes, whereby each CAN message is confirmed individually.
- "block transfer" is also for any number of bytes; here, a given block of CAN tickets is confirmed at once.

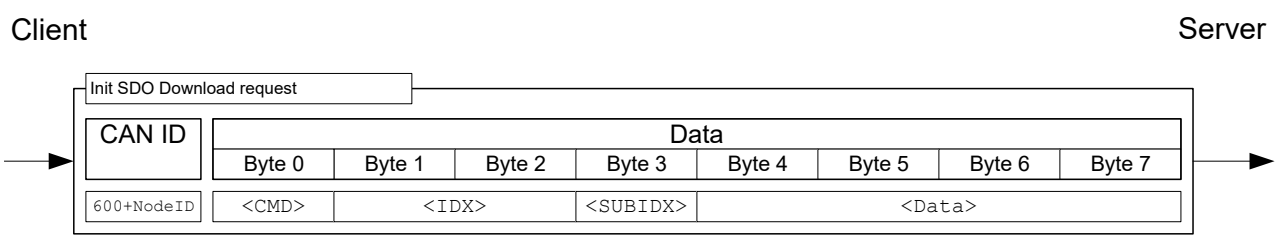
An SDO message is sent to CAN-ID $600_h$ + node-ID, the answer comes from CAN-ID $580_h$ + node-ID.

#### 8.2.4.1 Expedited Transfer

This method is used to write (download) or read (upload) values in objects of type (UN)SIGNED8, INTEGER16 oder INTEGER32 in the object dictionary. This service is confirmed, i.e., each access is answered with data, with a confirmation or with an error message.

**SDO Download**

An expedited SDO message for writing data in the object dictionary of the server is structured as follows:

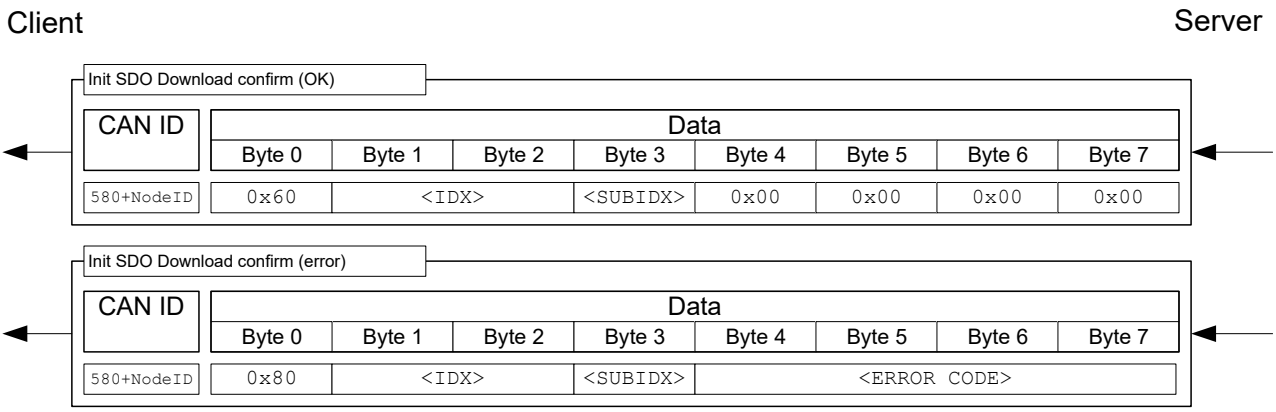


Here, the <CMD> byte is dependent on the length of the data that are to be written. <CMD> can be one of the following values:

- 1 byte data length: $2F_h$
- 2 byte data length: $2B_h$
- 3 byte data length: $27_h$
- 4 byte data length: $23_h$

The <Data> field is written with the data that are to be written; the LSB of the data is entered in byte 4.

The answer from the server is either a confirmation of the write operation or an error message (structure of the messages: see following figure). In the latter case, the reason for the error is also sent with the data (see list of the SDO error messages in section SDO error messages).




**Example**: Set object $607A_h$:$00_h$ (target position, SIGNED32) to value $3E8_h$ (=$1000_d$) of a controller with node-ID 3:

```
603 | 23 7A 60 00 E8 03 00 00
```

Where

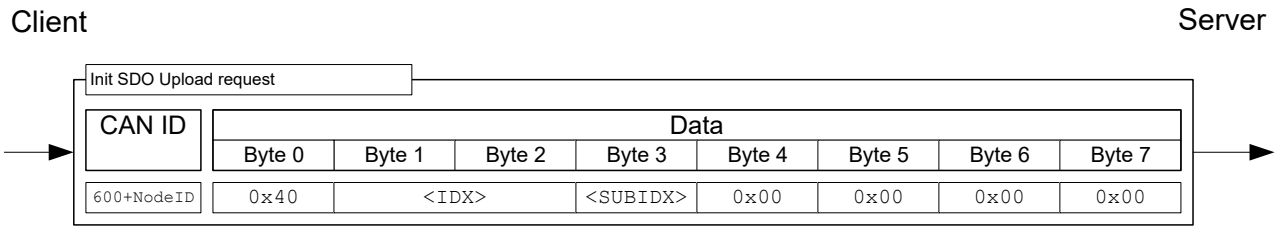
- Byte 1 ($23_h$): SDO expedited download, 4 bytes of data (SIGNED32)
- Bytes 2 and 3 ($7A_h$ $60_h$): index of object is $607A_h$
- Byte 4 ($00_h$): subindex of object is $00_h$
- Bytes 5 to 8 ($E8_h$ $03_h$ $00_h$ $00_h$): value of object: $000003E8_h$

If successful, the controller responds with this message:

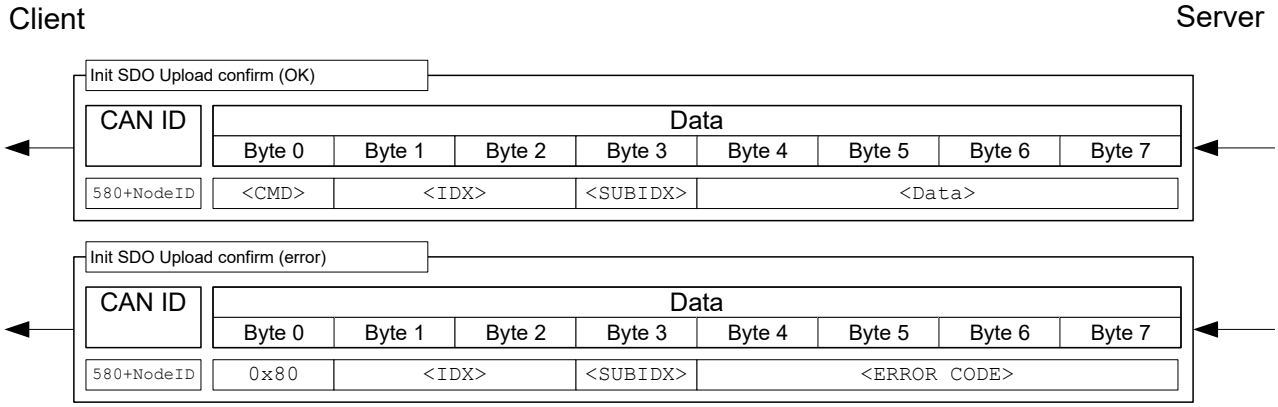
```
583 | 60 7A 60 00 00 00 00 00
```

**SDO upload**

A CAN message for reading an object from the object dictionary has the following structure:




The server responds with one of the following messages.




The length of the data is encrypted in the <CMD> of the answer:

| | |
|---|---|
| 1 byte data length: | $4F_h$ |
| 2 byte data length: | $4B_h$ |
| 3 byte data length: | $47_h$ |
| 4 byte data length: | $43_h$ |

The LSB of the data is again in byte 4 here.

In case of an error, the reason for the error is also specified in the data (see list of SDO error messages in SDO error messages).

**Example**: To read the "statusword" object ($6041_h$:00) from the object dictionary, it suffices to send the following message (always 8 bytes):

```
603 | 40 41 60 00 00 00 00 00
```

The controller generally responds with the following message:

```
583 | 4B 41 60 00 40 02 00 00
```

Where

- Byte 1 ($4B_h$): SDO expedited upload, 2 bytes of data (UNSIGNED16)
- Bytes 2 and 3 ($41_h$ $60_h$): index of object is $6041_h$
- Byte 4 ($00_h$): subindex of object is $00_h$

- Bytes 5 to 6 ($40_h$ $02_h$ ): value of object: $0240_h$
- Bytes 7 to 8 ($00_h$ $2_h$ $h$ $h$): empty. An SDO message always consists of 8 bytes.

### 8.2.4.2 Normal Transfer

Unlike "expedited" CANopen transfer, "normal transfer" is not limited to maximum four bytes. With this type of transfer, the content of multiple messages is grouped together with respect to content; such a block of messages is referred to in the following as a "transfer". Each message within a transfer is confirmed individually here.

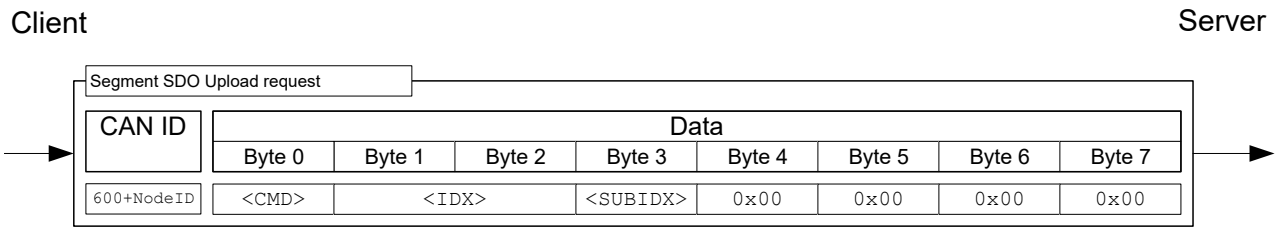
**NOTICE**

If your CANopen master does not support normal transfer, there is another way to access objects of data type String: each string can be read out character by character with an SDO upload to subindex 1 and the following subindicies.

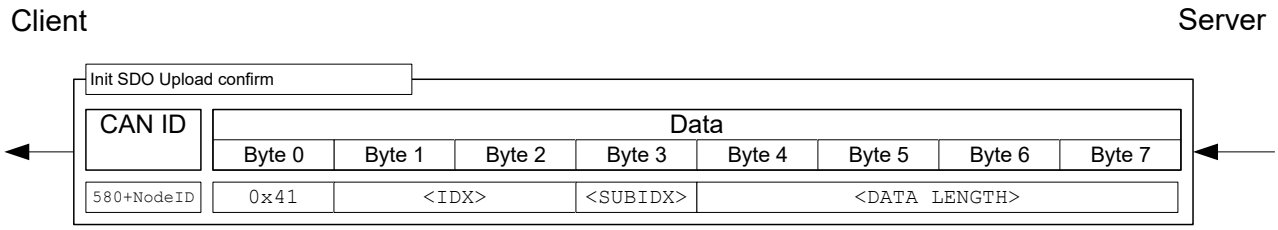
#### SDO upload

Shown in the following figure is the procedure for an "SDO upload" (client requests that the content of an object be sent to it). The transfer is broken down into two phases: an initialization phase and a transfer phase.

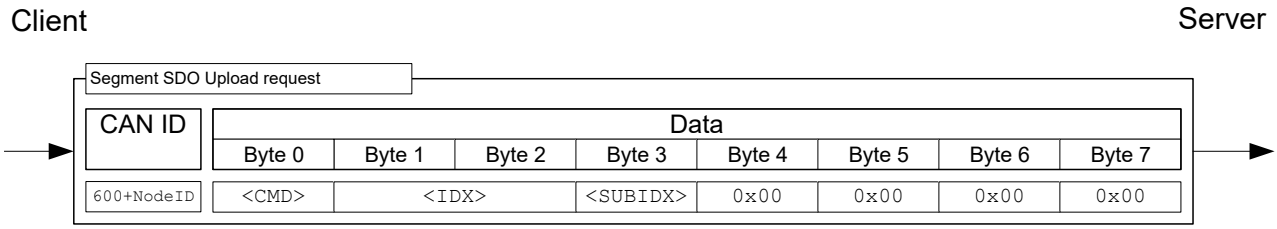
As with an "expedited transfer", the upload begins with the client sending an "Init SDO Update" to the server (see following figure).




The answer for a "normal transfer" does not contain the quantity of bytes to be received encoded in the <CMD>. It is instead entered in the data range as can be seen in the following figure in the <DATA LENGTH> area.



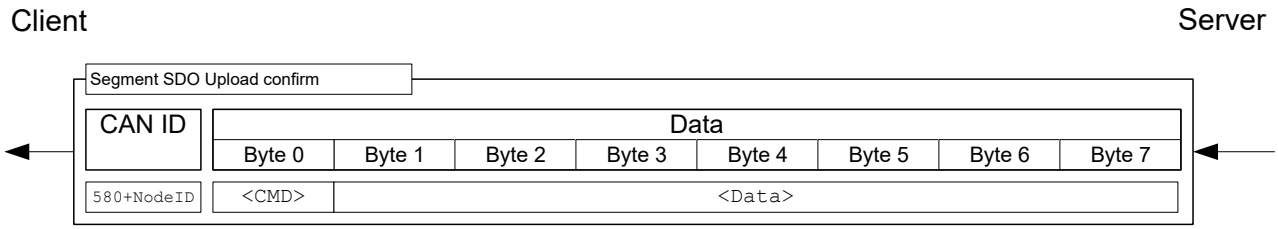
The initialization is thereby concluded; all that remains is the upload of the data. A data packet is requested with the following SDO request:




Byte 0 with command `<CMD>` is structured as follows:

The bit with designation `t` alternates with each request ("toggle bit"). It begins each transfer with 0, even if the previous transfer was aborted.

The controller responds to the above message with the data, whereby the message is structured as follows:




Byte 0 with `<CMD>` is structured as follows:

The bits have the following meaning here:

**`t` (toggle bit)**

The bit alternates with each message sequence; it does not change within a sequence between "request" and "response".

**`n` (number of bytes)**

These three bits specify how many bytes contain *no* data. Example: If bits 2 and 1 are set to 0 and bit 3 is set to 1, then $011_b = 03_d$ bytes are not valid. This, in turn, means that byte 1 to byte 4 contain allowed values and byte 5 to byte 7 should be disregarded.

**`c` (more segments)**

If no further SDO segments are sent and this is the last segment, the bit is set to 1.

**Example**: In this example, the "Manufacturer Software Version" object ($\underline{100A}_h$) is to be read. The node-ID of the node in this example is 3.

The corresponding SDO message sequence is listed in the following table. The string that is to be read out varies from controller to controller.

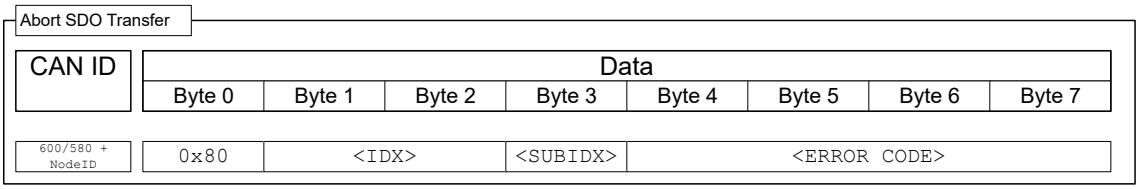
| COB-ID | Data | Description |
|---|---|---|
| $603_h$ | 40 0A 10 00 00 00 00 00 | Init Upload; Index: $100A_h$; Subindex: 00 |
| $583_h$ | 41 0A 10 00 11 00 00 00 | Init Upload; Size: indicated; transfer type: normal; Num of bytes: 17; Index: $100A_h$; Subindex: 00 |
| $603_h$ | 60 0A 10 00 00 00 00 00 | Upload Segment Req.; Toggle bit: not set |
| $583_h$ | 00 46 49 52 2D 76 31 37 | Upload Segment Conf.; More segments: yes; num of bytes: 7; Toggle bit: not set |
| $603_h$ | 70 0A 10 00 00 00 00 00 | Upload Segment Req.; Toggle bit: set |
| $583_h$ | 10 34 38 2D 42 35 33 38 | Upload Segment Conf.; More segments: yes; num of bytes: 7; Toggle bit: set |
| $603_h$ | 60 0A 10 00 00 00 00 00 | Upload Segment Req.; Toggle bit: not set |
| $583_h$ | 09 36 36 32 00 00 00 00 | Upload Segment Conf.; More segments: no (last segment); num of bytes: 3; Toggle bit: not set |

```
46 49 52 2D 76 31 37 34 38 2D 42 35 33 38 36 36 32
```

This corresponds to string: "FIR-v1748-B538662"

#### Aborting the SDO transfer

Both the server and the client are authorized to abort the current transfer. To do this, an "Abort SDO Transfer" must be sent; this is depicted in the following.




After receiving the message, the SDO transfer is considered ended; the service is not confirmed.

A new SDO transfer must then be started from the very beginning. Transfer of the `<ERROR CODE>` is optional; the controller does not evaluate the code.

### 8.2.4.3 SDO error messages

In case of an error, an error number specifying the reason for the error is also sent in the data area.

| Error Code | Description |
|---|---|
| $05030000_h$ | *toggle bit not changed*: Valid only with "normal transfer" or "block transfer". The bit, which is to alternate after each transfer, did not change its state. |
| $05040001_h$ | *command specifier unknown*: Byte 0 of the data block contains a command that is not allowed. |
| $06010000_h$ | *unsupported access*: If "complete access" was requested via CAN over EtherCAT (CoE) (is not supported.) |
| $06010002_h$ | *read only entry*: An attempt was made to write to a constant or read-only object. |
| $06020000_h$ | *object not existing*: An attempt was made to access a non-existing object (index incorrect). |
| $06040041_h$ | *object cannot be pdo mapped*: An attempt was made to map an object in the PDO for which that is not permissible. |

| Error Code | Description |
|---|---|
| $06040042_h$ | *mapped pdo exceed pdo*: If the desired object were to be attached to the PDO mapping, the 8 bytes of the PDO mapping would be exceeded. |
| $06070012_h$ | *parameter length too long*: An attempt was made to write to an object with too much data; for example, with <CMD>=$23_h$ (4 bytes) to an object of type Unsigned8, <CMD>=$2F_h$ would be correct. |
| $06070013_h$ | *parameter length too short*: At attempt was made to write to an object with too little data; for example, with <CMD>=$2F_h$ (1 byte) to an object of type Unsigned32, <CMD>=$23_h$ would be correct. |
| $06090011_h$ | *subindex not existing*: An attempt was made to access an invalid subindex of an object; the index, on the other hand, would exist. |
| $06090031_h$ | *value too great*: Some objects are subject to restrictions in the size of the value; in this case, an attempt was made to write an excessively large value to the object. For example, the "Pre-defined error field: Number of errors" object for $1003_h$:00 may only be set to the value "0"; all other numerical values result in this error. |
| $06090032_h$ | *value too small*: Some objects are subject to restrictions in the size of the value. In this case, an attempt was made to write a value that is too small to the object. |
| $08000000_h$ | *general error*: General error that does not fit in any other category. |
| $08000022_h$ | *data cannot be read or stored in this state*: The parameters of the PDOs may only be changed in the "Stopped" or "Pre-Operational" state. Write access of objects $1400_h$ to $1407_h$, $1600_h$ to $1607_h$, $1800_h$ to $1807_h$ and $1A00_h$ to $1A07_h$ is not permissible in the "Operational" state. |

## 8.2.5 Process Data Object (PDO)

A message that only contains process data is referred to as a "Process Data Object" (PDO). The PDO is intended for data that need to be exchanged cyclically.

The idea behind a PDO message is to remove all additional information (index, subindex and data length) from a CAN message and to only fill the CAN message with data. The source and target information for the PDO are stored separately in the so-called PDO mapping.

PDOs can only be used if the NMT state machine is in the "Operational" state (see section Network Management (NMT)); the PDOs must be configured in the "Pre-Operational" NMT state.

The controller supports a total of 8 independent PDO mappings; each corresponding PDO message can have a maximum of eight bytes (=64 bit) of user data. It is thereby possible to, for example, transfer two UNSIGNED32 values or one UNSIGNED32 and one UNSIGNED08; the message does not need to use all eight data bytes.

The PDOs differ yet again in the configuration in the send and receive configuration. The receive configuration describes the processing for PDO messages that are sent, and the send configuration describes the PDO messages that are to be sent.

### 8.2.5.1 RX configuration

To configure an RX-PDO, you must take into account three object categories in the object dictionary:

- The objects that describe the functionality of the mapping.
- The objects that describe the content of the mapping.
- The objects that are to receive the received data.

#### Configuration of the functionality (communication parameter)

The configuration of the first mapping is stored in the subindices of object $1400_h$. The second mapping is configured in $1401_h$ and so on. In the following, we refer to $140N_h$. Here, the configuration affects the COB-ID of the PDO message and the transfer type.

Objects $140N_h$ have only three subindices:

- Subindex 0 (max. subindex): Total number of subindices
- Subindex 1 (COB-ID): The COB-ID is stored here. For PDO mappings 1–4 ($1600_h$-$1603_h$), the CAN-ID is fixed depending on the node-ID and only the valid bit (bit 31) can be set in the COB-ID. From $1604_h$–$1607_h$, the CAN-ID can be set independently (with the restriction that it not be used by other services, see table at the start of chapter CANopen services) as can the valid bit.

| Mapping | CAN-ID |
|---|---|
| $1600_h$ | $200_h$ + Node-ID |
| $1601_h$ | $300_h$ + Node-ID |
| $1602_h$ | $400_h$ + Node-ID |
| $1603_h$ | $500_h$ + Node-ID |
| $1604_h$ | $xxx_h$ |
| $1605_h$ | $xxx_h$ |
| $1606_h$ | $xxx_h$ |
| $1607_h$ | $xxx_h$ |

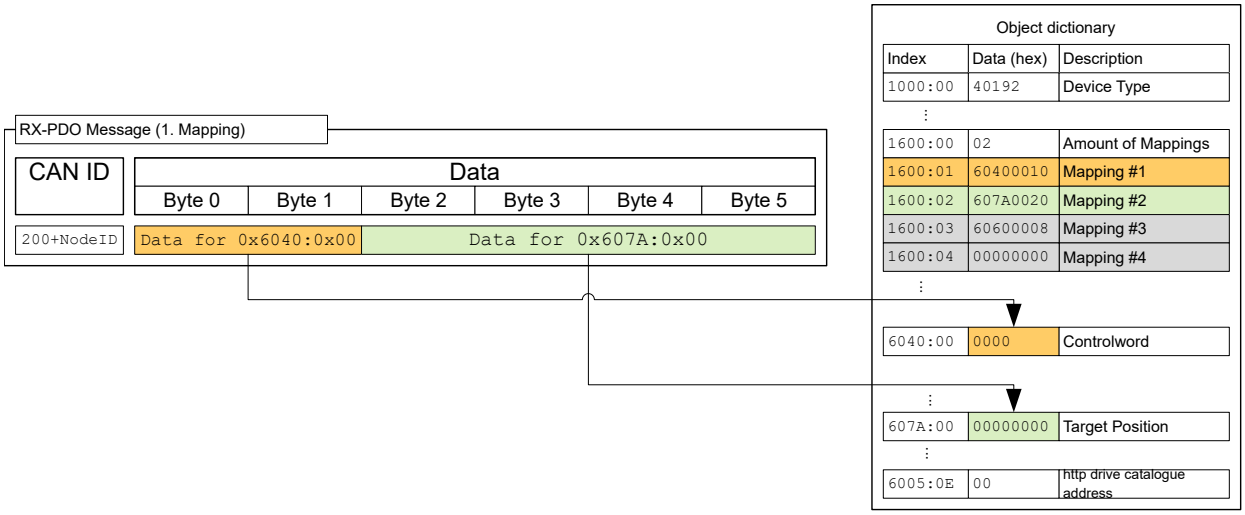
  A COB-ID change does not take effect until *after* the controller or communication is restarted (see Network Management (NMT)).
- Subindex 2 (transmission type): A number is stored in this subindex that defines the time at which the received data become valid. The number and the corresponding meaning can be found in the following table.

| $140N_h:02_h$ | Meaning |
|---|---|
| $00_h$-$F0_h$ | Synchronous: The data are buffered and not until the next SYNC message is received do they become valid and are they taken over into the object dictionary. |
| $F1_h$-$FD_h$ | Reserved |
| $FE_h$, $FF_h$ | Asynchronous: The data become valid when the PDO message is received and are taken over into the object dictionary. |

### Content of a mapping

The configuration of the content of a mapping is structured as follows (see also the following figure as an example):

- All subindices of a configuration object belong together. Thus, $1600_h$ with all subindices describes the first mapping, $1601_h$ the second RX-PDO mapping, etc.
- Subindex $00_h$ specifies how many objects are in a mapping. It simultaneously specifies how many of the subindices are valid. If object $1600_h:00_h$ is set to "0", RX mapping is thereby completely switched off. In the example shown in the following figure, two objects are thus mapped; object $1600_h:03_h$ and $1600_h:04_h$ is, therefore, not active (shown in gray).
- Each subindex from $1600_h:01_h$ to $1600_h:0F_h$ describes one target of the mapping sequentially and without gaps. The index, subindex and bit length are thereby encoded. Example from the following figure: The first two bytes of the message are to be written in object $6040_h:00_h$. In hexadecimal notation, the content of $1600_h:01_h$ then consists of
  `<Index><Subindex><Bit length>`
  or `60400010`. The second mapping ($1600_h:02_h$) contains the entry `607A0020`. Thus, it maps the following four bytes (=$20_h$Bit) in object $607A_h:00_h$




### Dummy objects

You can configure RX-PDOs so that more than one node can respond. In this case, it may be desirable for only part of the data contained in the PDO to be evaluated in one of the devices. For data not used locally, you can include a dummy object of one of the supported data types in the mapping of the PDO:

| Index | Data type |
|---|---|
| $0002_h$ | INTEGER8 |
| $0003_h$ | INTEGER16 |
| $0004_h$ | INTEGER32 |
| $0005_h$ | UNSIGNED08 |
| $0006_h$ | UNSIGNED16 |
| $0007_h$ | UNSIGNED32 |

## 8.2.5.2 TX configuration

To configure a TX-PDO, you must take into account three object categories in the object dictionary:

- The objects that describe the functionality of the mapping.
- The objects that describe the content of the mapping.
- The objects that are to receive the data that are to be sent.

Also note that the time at which the data are copied to the TX-PDO message and the time of sending do not need to be the same (dependent on mode).

### Configuration of the functionality (communication parameter)

The configuration of the functionality of the first mapping is stored in the subindices of object $1800_h$. The second mapping is configured in $1801_h$ and so on. In the following, we refer to $180N_h$. Here, the configuration affects the COB-ID of the PDO message and the transfer type.

Objects $180N_h$ have the following subindices:

- Subindex 0 (max. subindex): Total number of subindices
- Subindex 1 (COB-ID): The COB-ID is stored here. For PDO mappings 1–4 ($1A00_h$–$1A03_h$), the CAN-ID is fixed depending on the node-ID and only the valid bit (bit 31) can be set in the COB-ID. From $1A04_h$–$1A07_h$, the CAN-ID can be set independently (with the restriction that it not be used by other services, see table at the start of chapter CANopen services) as can the valid bit.

| Mapping | CAN-ID |
|---|---|
| $1A00_h$ | $180_h$ + Node-ID |
| $1A01_h$ | $280_h$ + Node-ID |
| $1A02_h$ | $380_h$ + Node-ID |
| $1A03_h$ | $480_h$ + Node-ID |
| $1A04_h$ | $xxx_h$ |
| $1A05_h$ | $xxx_h$ |
| $1A06_h$ | $xxx_h$ |
| $1A07_h$ | $xxx_h$ |

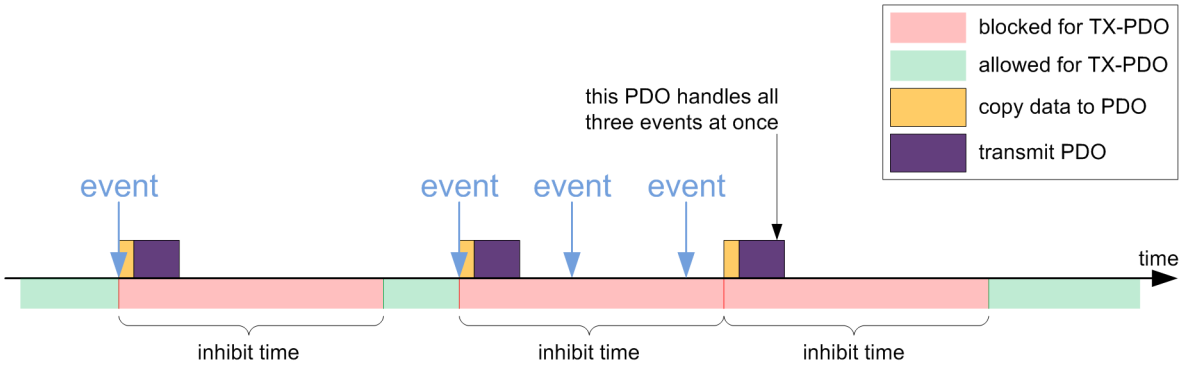
A COB-ID change does not take effect until *after* the controller or communication is restarted (see Network Management (NMT)).

- Subindex 2 (transmission type): A number is stored in this subindex that defines the time at which the data are to be copied into the PDO message and when this is to be sent. The number and the corresponding meaning can be found in the following table. Below, we refer to an *Event* that can trigger the copying and/or sending of the data. This *Event* consists of three events, which can be considered independently of one another:
  - The NMT state machine is switched to "operational".
  - The current data have changed with respect to the last PDO message.
  - The *Event Timer* has expired (see $180N_h$:5).

  If the *Event Timer* is used, it is handled independently of the changes; the *Event Timer* is not restarted until the current event timer expires, not because of another *Event*.

| $180N_h:02_h$ | Meaning |
|---|---|
| 0 | Synchronous (acyclic): The data are copied to the TX-PDO upon arrival of the SYNC but are not sent until the *Event*. |
| $01_h$-$F0_h$ | Synchronous (cyclic): The data are copied upon arrival of the nth SNCY message and are sent immediately thereafter (n corresponds to the number 1 to 240, transmission type "1" sends the new data on each SYNC). |
| $F1_h$-$FB_h$ | Reserved |
| $FC_h$ | RTR-Only (synchronous): The data are copied upon arrival of each SYNC message but are sent only upon request with an RTR message. |
| $FD_h$ | RTR-Only (event-driven): The data are copied to the TX-PDO message upon receipt of an RTR message and sent immediately thereafter. |
| $FE_h$, $FF_h$ | The data are copied upon arrival of the *Event* and sent immediately. |

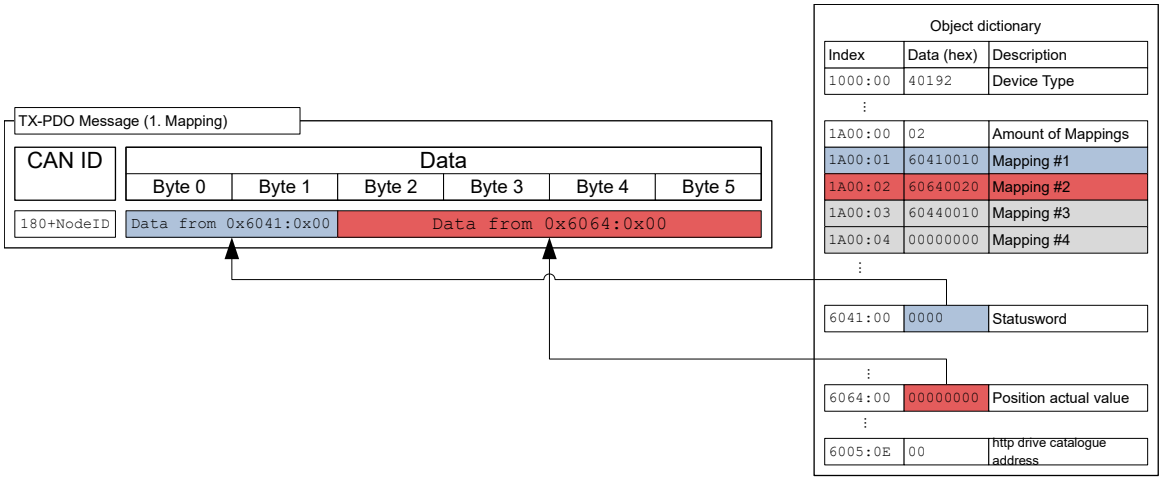
- Subindex 3 (inhibit time): This subindex contains a time lock in 100 µs steps (see following figure). This can be used to set a time that must elapse after the sending of a PDO before the PDO is sent another time. This time only applies for asynchronous PDOs. This is intended to prevent PDOs from being sent continuously if the mapped object constantly changes.
- Subindex 4 (compatibility entry): This subindex has no function and exists only for compatibility reasons.
- Subindex 5 (event timer): This time (in ms) can be used to trigger an *Event* which handles the copying of the data and the sending of the PDO.
- Subindex 6 (sync start value): Here, the start value of the *Sync Counter* is entered beginning with which the *slave* is to initially respond to the sync and send the PDO. Not globally activated until a value greater than 1 is set in 1019h Synchronous Counter Overflow Value.




### Content of a mapping

The configuration of the content of a mapping is structured as follows (see the following figure as an example):

- All subindices of a configuration object belong together. Thus, $1A00_h$ with all subindices describes the first mapping, $1A01_h$ the second TX-PDO mapping, etc.
- Subindex 00 specifies how many objects are in a mapping. It simultaneously specifies how many of the subindices are valid. If object $1A00_h$:$00_h$ is set to "0", TX mapping is thereby completely switched off. In the following example, two objects are thereby mapped in entries $1A00_h$:$01_h$ – $1A00_h$:$02_h$. The objects in entries $1A00_h$:$03_h$ – $1A00_h$:$04_h$ are, thus, not mapped (shown in gray).
- Each subindex from $1A00_h$:$01_h$ to $1A00_h$:$0F_h$ respectively describes sequentially and without gaps (dummy objects can be used for gaps) one source of the mapping. The index, subindex and bit length are thereby encoded. Example from the following figure: The first two bytes of the message are to be read from object $6041_h$:$00_h$. In hexadecimal notation, the content of $1A00_h$:$01_h$ then consists of `<Index><Subindex><Bit Length>`, or `60410010`. The second mapping ($1A00_h$:$02_h$) contains the entry `60640020`. Thus, it maps the following four bytes (corresponds to 32 bits) from object $6064_h$:$00_h$ in the TX-PDO message.




## 8.2.5.3 Presetting

The following configuration is preset:

**RX-PDO**

1. Mapping (CAN-ID: $200_h$ + Node-ID):

- $6040_h$:$00_h$ (controlword)
- $6060_h$:$00_h$ (mode of operation)

2. Mapping (CAN-ID: $300_h$ + Node-ID):

- 607A$_h$:00$_h$ (target position)
- 6081$_h$:00$_h$ (profile velocity)

3. Mapping (CAN-ID: 400$_h$ + Node-ID): object 60FF$_h$:00$_h$ (target velocity)

**TX-PDO**

1. Mapping (CAN-ID: 180$_h$ + Node-ID):

- 6041$_h$:00$_h$ (statusword)
- 6061$_h$:00$_h$ (Modes Of Operation Display)

2. Mapping (CAN-ID: 280$_h$ + Node-ID): 6064$_h$:00$_h$ (Position actual value)

3. Mapping (CAN-ID: 380$_h$ + Node-ID): 606C$_h$:00$_h$ (velocity actual value)

4. Mapping (CAN-ID: 480$_h$ + Node-ID): object 60FD$_h$:00$_h$ (digital inputs)

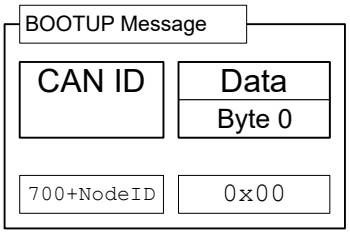
#### 8.2.5.4 Changing PDO mapping

You change the PDO mapping in the "Pre-operational" NMT state as follows:

1. Deactivate the PDO by setting the *Valid Bit* (bit 31) of subindex 01h of the corresponding communication parameter (e.g., 1400$_h$:01$_h$) to "1".
2. Deactivate the mapping by setting subindex 00h of the corresponding mapping parameter (e.g., 1600$_h$:00$_h$) to "0".
3. Change the mapping in the desired subindices (e.g., 1600$_h$:01$_h$).
4. Activate the mapping by writing the number of objects that are to be mapped in subindex 00h of the corresponding mapping parameter (e.g., 1600$_h$:00$_h$).
5. Activate the PDO by setting bit 31 of subindex 01h of the corresponding communication parameter (e.g., 1400$_h$:01$_h$) to "0".
6. Store the configuration by writing the value "65766173h" in 1010$_h$:02$_h$.

### 8.2.6 Boot-Up Protocol

If the CAN slave reaches the "Pre-Operational" NMT state (see following figure), the following message is sent to signal operational readiness.




This service is unconfirmed; there is no response.

### 8.2.7 Heartbeat and Nodeguarding

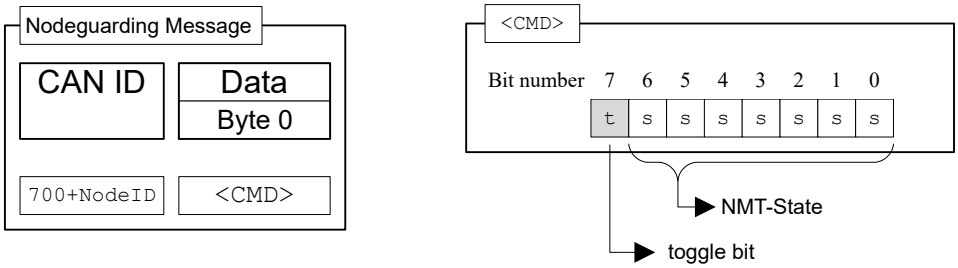
With the "Heartbeat" and "Nodeguarding" services (often also referred to as "Liveguarding"), switched-off or hung devices on the CAN bus can be found. For this purpose, the NMT master cyclically requests a message with the current NMT state of the slave (Nodeguarding).

The alternative is that each slave sends a message unprompted and cyclically (Heartbeat). A combination of Nodeguarding and Heartbeat is not permissible. Furthermore, it is recommended that Heartbeat be given preference over Nodeguarding, as Nodeguarding results in a higher load on the CAN bus.

#### 8.2.7.1 Nodeguarding

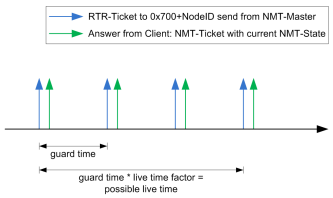
This service is based on the fact that the NMT master sends an RTR message with CAN-ID $700_h$ + node-ID to the respective slave.

The slave must then send a message as response; this message is structured as follows. Bit 7 alternates here on each transfer, thereby allowing one to determine if a message was lost. Entered in bits 6 to 0 is the current NMT status of the slave.




With Nodeguarding, there exist three time intervals (see also the following figure):

- *guard time*: The time between two RTR messages. This can be different for each CAN node and is stored in the slave in object $\underline{100C_h}$:00 (unit: milliseconds)
- *live time factor*: A multiplier for the *guard time*; this is stored in the CAN slave in object $\underline{100D_h}$:00 and can be different for each slave on the CAN bus.
- *possible live time*: The time produced by multiplying *guard time* and *live time factor*.

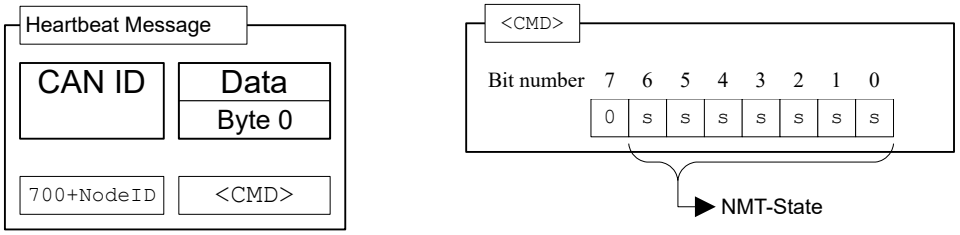



The following conditions are checked during Nodeguarding:

- The NMT master must send the RTR request within the "possible live time".
- The slave must send the response to the RTR request within the "possible live time".
- The slave must respond with its NMT state. In addition, the "toggle bit" must be set correctly.

#### 8.2.7.2 Heartbeat

If Heartbeat is activated, the slave sends its NMT state to the CAN bus unprompted and cyclically. You activate this service by setting the *Producer Heartbeat Time* time in object $1017_h$:$00_h$ to a value other than zero. The *Producer Heartbeat Time* is measured in milliseconds. The message sent by the slave has the form shown below:




The slave must send the Heartbeat message within the *Heartbeat Consumer Time*. This time is known only to the master and is not stored in the controller.

The slave can also monitor a *Heartbeat* from another *producer* (master or another slave). To do this, enter the *Consumer Heartbeat Time* and the node-ID of the *producer* in object $1016_h$.

Errors that occur during this monitoring are reset if either the function is deactivated or the *Heartbeat* is again sent within the correct time.

## 8.3 LSS protocol

The services of the *LSS protocol (Layer Settings Services)* are used to assign the node-ID and/or the baud rate of the controller directly via the CANopen bus. This is especially useful with devices that have no means for the mechanical configuration (e.g., rotary switches) of the parameters.

### 8.3.1 General

The *LSS protocol* requires a CANopen device in the network that performs the role of the *LSS master*. All other devices have the role of the *LSS slave*.

Each *LSS slave* is equipped with a unique *LSS address* that consists of the four 32-bit entries of object 1018h Identity Object.

An *LSS slave* may either be in *configuration mode* or in *wait mode*. The *LSS master* is responsible for switching between the two modes. Some *LSS services* (*Configuration*, *Inquiry*) are only available in *configuration mode*.

### 8.3.2 LSS message

All messages of the *LSS protocol* consist of 8 bytes (*DLC*=8), whereby byte 0 always contains the *Command Specifier (CS)* of the service.

Two CAN IDs are reserved for the *LSS protocol*:

- $7E5_h$: For the messages from the *LSS master* to the *LSS slaves* (request)
- $7E4_h$: For the messages from the *LSS slaves* to the *LSS master* (response)

### 8.3.3 LSS services

Four service categories are supported:

- Switch state services
- Configuration services
- Inquiry services
- Identification services

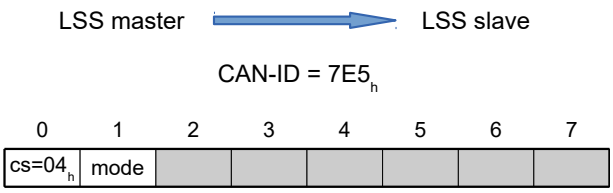
#### 8.3.3.1 Switch state services

With these services, the *LSS master* can switch the *LSS slaves* to *configuration mode* or to *wait mode*.

The node-ID and baud rate can only be changed with the Configuration services and Inquiry services while in *configuration mode*.

#### Switch state global service

With this service, the *LSS master* switches all *LSS slaves* in the network to *configuration mode* or to *wait mode*.




**Byte 0 : CS (Command Specifier)**

Value = "$04_h$"

**Byte 1: mode**

Value = "$00_h$": Switches to *wait mode*

Value = "$01_h$": Switches to *configuration mode*

**Bytes 2-7 :**

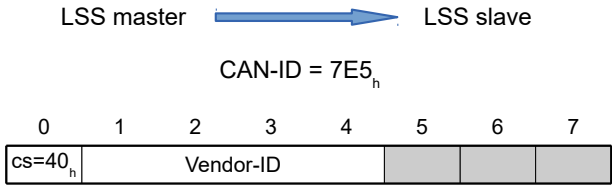
reserved (=$0_h$)

#### Switch state selective service

With this service, the *LSS master* switches the *LSS slaves* with the (or parts of the) corresponding *LSS address to configuration mode*.

The *LSS master* sends four messages, which contain the *LSS address*:

1. The *LSS master* switches the *LSS slaves* with the corresponding *vendor ID* to *configuration mode*:




**Byte 0 : CS (Command Specifier)**

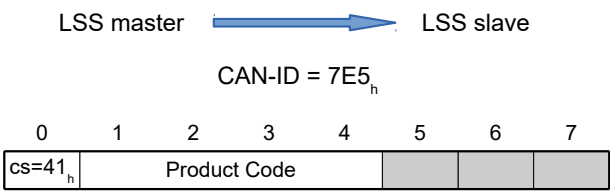
Value = "$40_h$"

**Bytes 1-4: Vendor-ID**

*Vendor-ID*: see $1018_h$:$01_h$

**Bytes 5-7:**
reserved (=$0_h$)

2. The *LSS master* switches the *LSS slaves* with the corresponding *product code* to *configuration mode*:

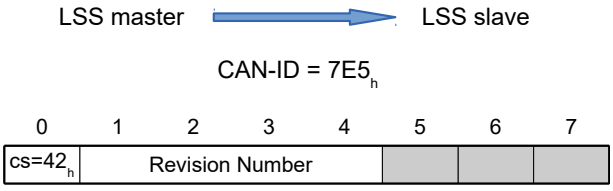



**Byte 0 : CS (Command Specifier)**
Value = "$41_h$"

**Bytes 1-4: Product Code**
*Product code*: see $1018_h$:$02_h$

**Bytes 5-7:**
reserved (=$0_h$)

3. The *LSS master* switches the *LSS slaves* with the corresponding *revision number* to *configuration mode*:

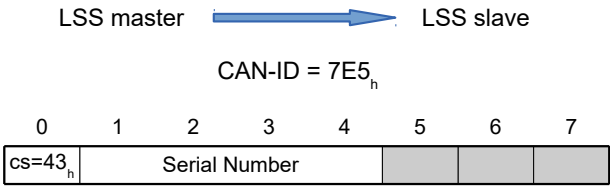



**Byte 0 : CS (Command Specifier)**
Value = "$42_h$"

**Bytes 1-4: Revision Number**
*Revision number*: see $1018_h$:$03_h$

**Bytes 5-7:**
reserved (=$0_h$)

4. The *LSS master* switches the *LSS slaves* with the corresponding *serial number* to *configuration mode*:

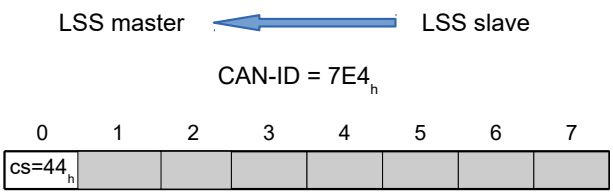



**Byte 0 : CS (Command Specifier)**
Value = "$43_h$"

**Bytes 1-4: mode**
*Serial number*: see $1018_h$:$04_h$

**Bytes 5-7:**
reserved (=$0_h$)

The *LSS slave* with the corresponding *LSS address* was switched to *configuration mode* and sends a confirmation:




**Byte 0 : CS (Command Specifier)**

Value = "$44_h$"

**Bytes 1-7 :**

reserved (=$0_h$)

### 8.3.3.2 Configuration services

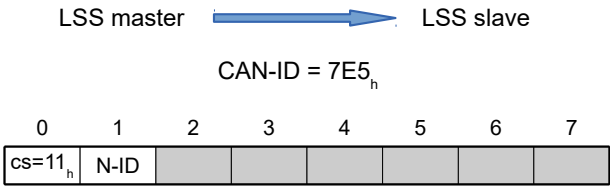
With these services, the *LSS master* can change and, if necessary, store the node-ID or baud rate of the *LSS slaves*.

**NOTICE**

The *LSS slaves* must be in *configuration mode*. See chapter Switch state services.

### Configure node-ID service

The *LSS master* sends a message with the new node-ID to an *LSS slave*:




**Byte 0 : CS (Command Specifier)**

Value = "$11_h$"

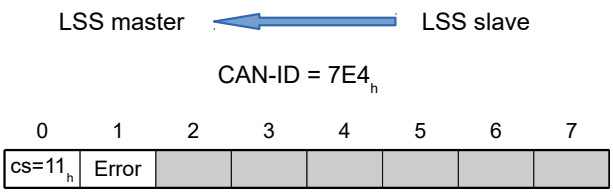
**Byte 1: N-ID (Node-ID)**

Valid node-ID between $01_h$ and $7F_h$

**Bytes 2-7 :**

reserved (=$0_h$)

The *LSS slave* responds with a confirmation/error code:



**Byte 0 : CS (Command Specifier)**

Value = "$11_h$"

**Byte 1: Error Code**

Value = "$00_h$": Not an error

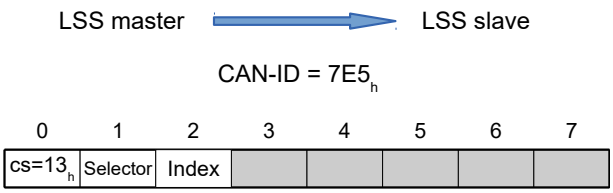
Value = "$01_h$": Invalid node-ID

**Bytes 2-7 :**

reserved (=$0_h$)

### Configure bit timing parameters service

The *LSS master* sends a message with the new baud rate to an *LSS slave*:




**Byte 0 : CS (Command Specifier)**

Value = "$13_h$"

**Byte 1: Table Selector**

Value = "$00_h$": The table for the baud rate from the *CiA 301* standard is used.

**Byte 2: Table Index**

The value for the index is taken from the following table.

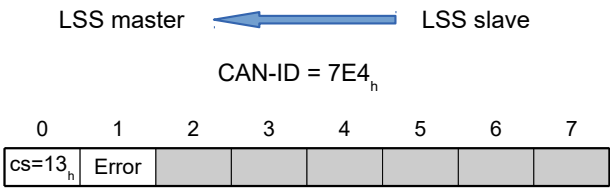
**Bytes 3-7 :**

reserved (=$0_h$)

The following values are supported for the *Table Index*:

| Table Index | Baud rate in kBd |
|---|---|
| 0 | 1000 |
| 2 | 500 |
| 3 | 250 |
| 4 | 125 |
| 6 | 50 |
| 7 | 20 |
| 8 | 10 |

The *LSS slave* responds with a confirmation/error code:




**Byte 0 : CS (Command Specifier)**

Value = "$13_h$"

**Byte 1: Error Code**

Value = "$00_h$": Not an error

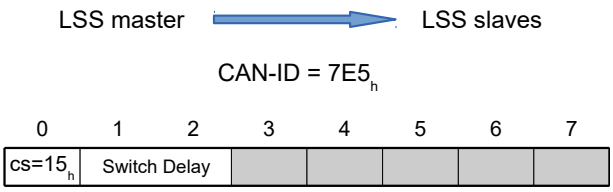
Value = "$01_h$": Invalid *Table Index*/baud rate is not supported

**Bytes 2-7 :**

reserved (=$0_h$)

### Activate bit timing parameters service

The *LSS master* uses this command to activate the set baud rate of all *LSS slaves* in the network simultaneously:




**Byte 0 : CS (Command Specifier)**

Value = "$51_h$"

**Bytes 1-2: Switch Delay**

Delay in ms. It is thereby ensured that all *LSS slaves* in the network have the same baud rate before messages may again be sent.

After receiving this messages from each *LSS slave*, the time that is stored here is allowed to elapse. Only then is the new baud rate accepted.

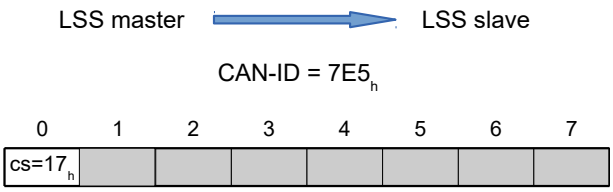
The same time is allowed to elapse a second time; only then may an *LSS slave* send messages again.

**Bytes 3-7 :**

reserved (=$0_h$)

### Store configuration service

With this command, the *LSS master* saves the set node-ID and baud rate of an *LSS slave*. The *LSS master* must ensure that at that moment only one *LSS slave* in the network is in *configuration mode*.



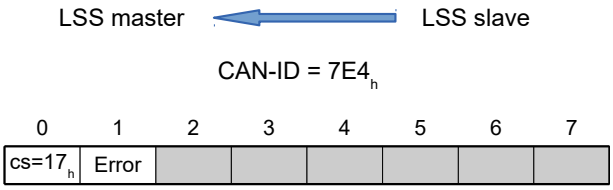

**Byte 0 : CS (Command Specifier)**

Value = "$17_h$"

**Bytes 1-7 :**

reserved (=$0_h$)

The *LSS slave* responds with a confirmation/error code:




**Byte 0 : CS (Command Specifier)**

Value = "$17_h$"

**Byte 1: Error Code**

Value = "$00_h$": Not an error

Value = "$02_h$": Access of non-volatile memory failed

**Bytes 2-7 :**

reserved (=$0_h$)

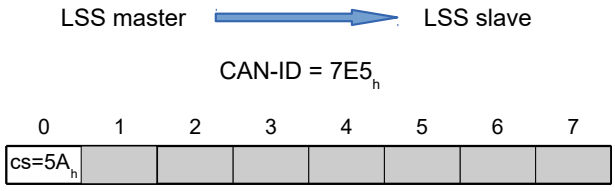
### 8.3.3.3 Inquiry services

With these services, the *LSS master* can query the *LSS address* or the node-ID of an *LSS slave*. The *LSS master* must ensure that only one *LSS slave* in the network is in *configuration mode*.

### Inquire LSS address service

With this service, the *LSS master* queries the *LSS address* of a slave.

1. The *LSS master* queries the *vendor ID*:



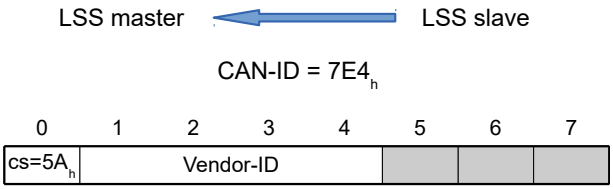

**Byte 0 : CS (Command Specifier)**

Value = "$5A_h$"

**Bytes 1-7 :**

reserved (=$0_h$)

The *LSS slave* returns its *vendor ID*:




**Byte 0 : CS (Command Specifier)**

Value = "$5A_h$"

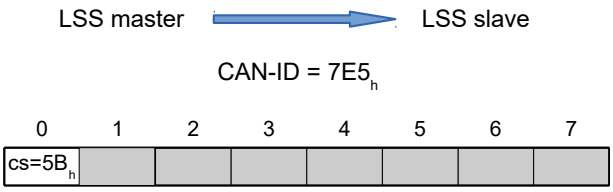
**Bytes 1-4: Vendor-ID**

*Vendor-ID*: see $1018_h$:$01_h$

**Bytes 5-7:**

reserved (=$0_h$)

2. The *LSS master* queries the *product code*:



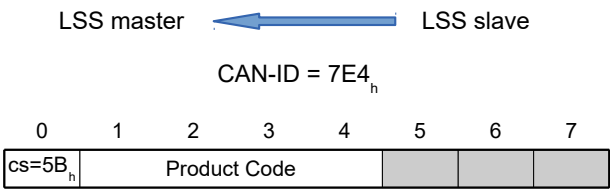

**Byte 0 : CS (Command Specifier)**

Value = "$5B_h$"

**Bytes 1-7 :**

reserved (=$0_h$)

The *LSS slave* returns its *product code*:




**Byte 0 : CS (Command Specifier)**

Value = "$5B_h$"

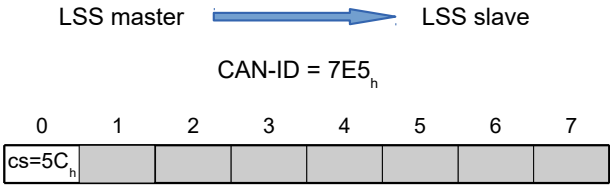
**Bytes 1-4: Produt Code**

*Product code*: see $1018_h$:$02_h$

**Bytes 5-7:**

reserved (=$0_h$)

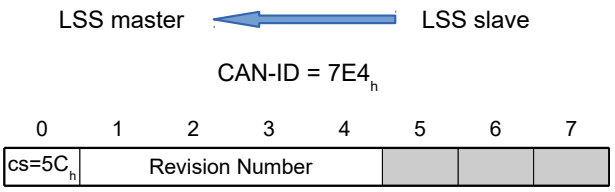
3. The *LSS master* queries the *revision number*:




**Byte 0 : CS (Command Specifier)**

Value = "$5C_h$"

The *LSS slave* returns its *revision number*:




**Byte 0 : CS (Command Specifier)**

Value = "$5C_h$"

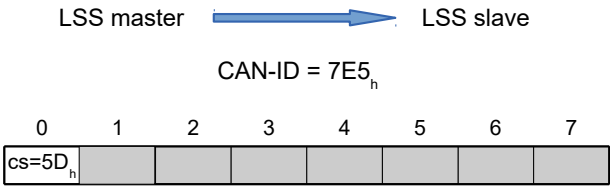
**Bytes 1-4: Revision Number**

*Revision number*: see $\underline{1018_h}$:$03_h$

**Bytes 5-7:**

reserved (=$0_h$)

4. The *LSS master* queries the *serial number*:



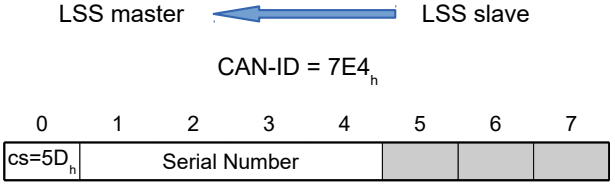

**Byte 0 : CS (Command Specifier)**

Value = "$5D_h$"

**Bytes 1-7 :**

reserved (=$0_h$)

The *LSS slave* returns its *serial number*:




**Byte 0 : CS (Command Specifier)**

Value = "$5D_h$"

**Bytes 1-4: Serial Number**

*Serial number*: see $1018_h$:$04_h$

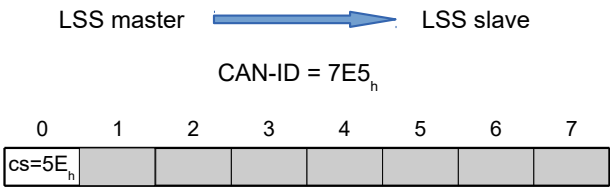
**Bytes 5-7:**

reserved (=$0_h$)

### Inquire node-ID service

With this service, the *LSS master* queries the node-ID of a slave.

The *LSS master* queries the node-ID:



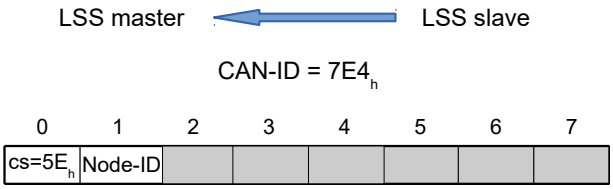

**Byte 0 : CS (Command Specifier)**

Value = "$5E_h$"

**Bytes 1-7 :**

reserved (=$0_h$)

The *LSS slave* responds with its node-ID:




**Byte 0 : CS (Command Specifier)**

Value = "$5E_h$"

**Byte 1: Node-ID**

Node-ID of the *LSS slave*

**Bytes 2-7 :**

reserved (=$0_h$)

## 8.3.3.4 Identification services

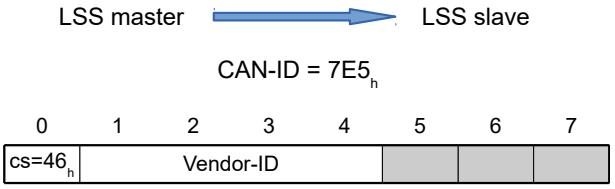
With these services, the *LSS master* can ask the *LSS slaves* to identify themselves based on their *LSS address*.

### LSS identify remote slave service

With this service, the *LSS master* asks the *LSS slaves* to identify themselves with the (or parts of the) corresponding *LSS address* with the LSS identify slave service.

A range can be defined for the *revision number* and the *serial number*. All *LSS slaves* whose numbers are in the corresponding range must identify themselves. It is the task of the *LSS master* to restrict the range so that ultimately only one *LSS slave* responds.

1. The *LSS master* defines the *Vendor-ID* of the *LSS slaves* that are to identify themselves:




**Byte 0 : CS (Command Specifier)**

Value = "$46_h$"

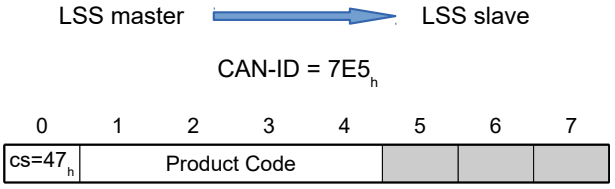
**Bytes 1-4: Vendor-ID**

*Vendor-ID*: see $1018_h$:$01_h$

**Bytes 5-7:**

reserved (=$0_h$)

2. The *LSS master* defines the *product code* of the *LSS slaves* that are to identify themselves.:




**Byte 0 : CS (Command Specifier)**

Value = "$47_h$"

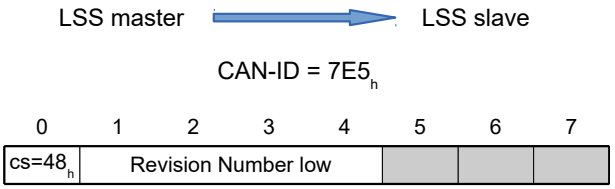
**Bytes 1-4: Product Code**

*Product code*: see $1018_h$:$02_h$

**Bytes 5-7:**

reserved (=$0_h$)

3. The *LSS master* defines the lowest and highest *revision number* of a range. All *LSS slaves* whose *revision number* is within this range are to identify themselves:




**Byte 0 : CS (Command Specifier)**

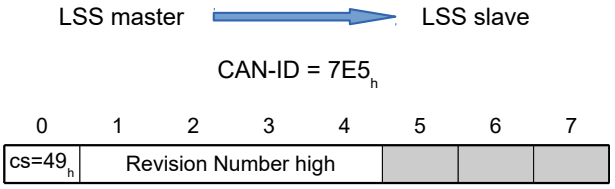
Value = "$48_h$"

**Bytes 1-4: Revision Number low**

Lowest *revision number* of the range: see $1018_h$:$03_h$

**Bytes 5-7:**

reserved (=$0_h$)




**Byte 0 : CS (Command Specifier)**

Value = "$49_h$"

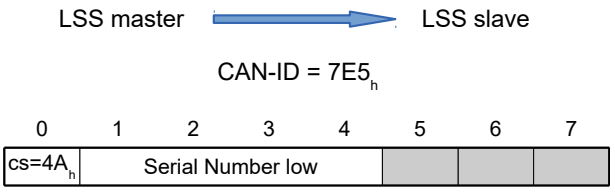
**Bytes 1-4: Revision Number high**

Highest *revision number* of the range: see $1018_h$:$03_h$

**Bytes 5-7:**

reserved (=$0_h$)

4. The *LSS master* defines the lowest and highest *serial number* of a range. All *LSS slaves* whose *serial number* is within this range are to identify themselves:




**Byte 0 : CS (Command Specifier)**

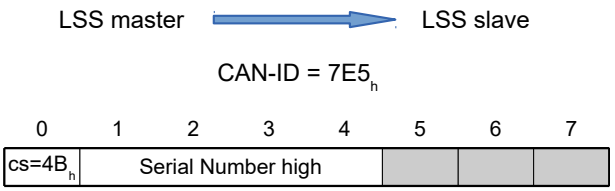
Value = "$4A_h$"

**Bytes 1-4: Serial Number low**

Lowest *serial number* of the range: see $1018_h$:$04_h$

**Bytes 5-7:**

reserved (=$0_h$)




**Byte 0 : CS (Command Specifier)**

Value = "$4B_h$"

**Bytes 1-4: Serial Number high**

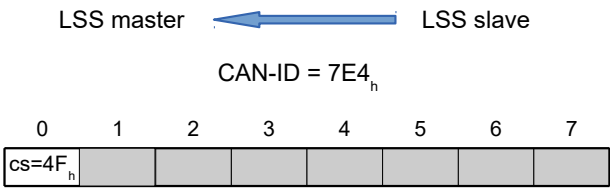
Highest *serial number* of the range: see $1018_h$:$04_h$

**Bytes 5-7:**

reserved (=$0_h$)

## LSS identify slave service

The *LSS slave* whose *LSS address* was defined with the LSS identify remote slave service by the *LSS master* identifies itself:




**Byte 0 : CS (Command Specifier)**

Value = "$4F_h$"

**Bytes 1-7 :**

reserved ($=0_h$)

### 8.3.4 Example

The controller (*LSS slave*) is delivered with the following parameters:

- Node-ID = $7F_h$ ($=127_d$)
- Baud rate = 1000 kBd

The parameters are to be set as follows:

- Node-ID = $05_h$ ($=5_d$)
- Baud rate = 125 kBd

It is assumed that only one *LSS slave* is currently present in the network.

1. The *LSS master* switches the *LSS slave* to *configuration mode* (see Switch state global service):
   `7E5 | 04 01 00 00 00 00 00 00`
2. The *LSS master* queries the node-ID of the *LSS slave* (see Inquire node-ID service):
   `7E5 | 5E 00 00 00 00 00 00 00`
   The *LSS slave* responds with its node-ID:
   `7E4 | 5E 7F 00 00 00 00 00 00`
3. The *LSS master* sets the node-ID to "$05_h$" (see Configure node-ID service):
   `7E5 | 11 05 00 00 00 00 00 00`
   The *LSS slave* confirms (error code=$00_h$):
   `7E4 | 11 00 00 00 00 00 00 00`
4. The *LSS master* sets the baud rate to 125 kBd (*Table Index*=4) (see Configure bit timing parameters service):
   `7E5 | 13 00 04 00 00 00 00 00`
   The *LSS slave* confirms (error code=$00_h$):
   `7E4 | 13 00 00 00 00 00 00 00`
5. The *LSS master* sends the command to save the changes (see Store configuration service):
   `7E5 | 17 00 00 00 00 00 00 00`
6. The *LSS master* switches the *LSS slave* to *wait mode* (see Switch state global service):
   `7E5 | 04 00 00 00 00 00 00 00`
   The *LSS slave* confirms (error code=$00_h$):
   `7E4 | 17 00 00 00 00 00 00 00`
7. The new parameters are accepted after the controller is restarted.

The controller registers with node-ID 5 and baud rate 125 kBd:

`705 | 00`

# 9 Programming with NanoJ

*NanoJ* is a programming language similar to *C* or *C++*. NanoJ is integrated in the *Plug & Drive Studio 3* software. You can find further information in the document *Plug & Drive Studio 3: User Manual* at us.nanotec.com.

## 9.1 NanoJ program

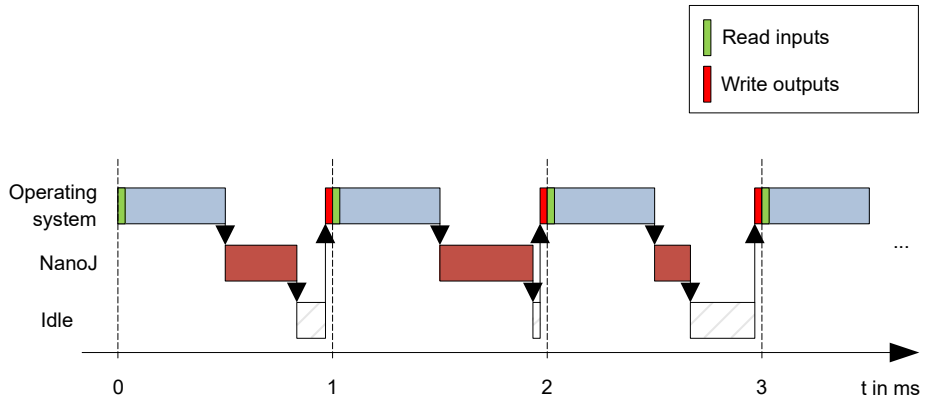
A *NanoJ program* makes a protected runtime environment available within the firmware. Here, the user can create his own processes. These can then trigger functions in the controller by, for example, reading or writing entries in the object dictionary.

Through the use of protective mechanisms, a *NanoJ program* is prevented from crashing the firmware. In the worst case, the execution is interrupted with an error code stored in the object dictionary.

If the *NanoJ program* was loaded on the controller, it is automatically executed after the controller is switched on or restarted, as long as you do not set bit 0 in object $2300_h$ to "0".

### 9.1.1 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 part of computing time is available to the user program (depending on control mode and application). In this time, the user program must run through the cycle and either complete the cycle or yield the computing time by calling the `yield()` function. In the former case, the user program is restarted with the start of the next 1 ms cycle; the latter results in the program being continued on the next 1 ms cycle with the command that follows the `yield()` function.




If the *NanoJ program* needs more time than was allotted, it is ended and an error code set in the object dictionary.

**TIP**

When developing user programs, the runtime behavior must be carefully examined, especially for more time-intensive tasks. For example, it is therefore recommended that tables be used instead of calculating a sine value using a `sin` function.

**NOTICE**

If the *NanoJ program* does not yield the computing time after too long a time, it is ended by the operating system. In this case, the number `4` is entered in the statusword for object $2301_h$; in the error register for object $2302_h$, the number `5` (timeout) is noted, see 2301h NanoJ Status and 2302h NanoJ Error Code.

To keep the *NanoJ program* from stopping, you can activate *AutoYield* mode by writing value "5" in $2300_h$. In *AutoYield* mode, however, the *NanoJ program* is no longer real-time capable and no longer runs every 1 ms.

### 9.1.2 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.

### 9.1.3 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

The OD values of the user program are made available in the form of variables via *PDO mapping*. Before a user program receives the 1 ms time slot, the firmware transfers the values from the object dictionary to the variables of the user program. As soon as the user program receives computing time, it can manipulate these variables as regular C variables. At the end of the time slot, the new values are then automatically copied by the firmware back to the respective OD entries.

To optimize the performance, three types of mapping are defined: input, output, and input/output (In, Out, InOut).

- *Input mappings* can only be read; they are not transferred back to the object dictionary.
- *Output mappings* can only be written.
- *Input/output mappings*, on the other hand, can both be read and written.

The set mappings can be read and checked via the GUI for objects $2310_h$, $2320_h$, and $2330_h$. Up to 16 entries are allowed for each mapping.

Whether a variable is stored in the input, output or data range is controlled in *Plug & Drive Studio* via the specification of the *linker section*.

#### NanoJ inputs and NanoJ outputs

To communicate with the *NanoJ program* via the respective interface, you can use the following objects:

- 2400h NanoJ Inputs: Array with thirty-two S32 values for passing values to the *NanoJ program*
- 2500h NanoJ Outputs: Array with thirty-two S32 values, where the *NanoJ program* can store values that can be read out via the fieldbus

### 9.1.4 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.

### 9.1.5 NanoJ program – OD entries

The *NanoJ program* is controlled and configured in object range $2300_h$ to $2330_h$ (see 2300h NanoJ Control).

| OD-Index | Name and description |
|---|---|
| $2300_h$ | 2300h NanoJ Control |
| $2301_h$ | 2301h NanoJ Status |
| $2302_h$ | 2302h NanoJ Error Code |
| $2310_h$ | 2310h NanoJ Input Data Selection |
| $2320_h$ | 2320h NanoJ Output Data Selection |
| $2330_h$ | 2330h NanoJ In/output Data Selection |

**Example:**

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

- Rename file "`TEST1.USR`" with `nanoj*.usr`.
- Copy file `nanoj*.usr` to the controller via USB and restart the controller.
- Start the NanoJ program by writing object $2300_h$, bit 0 = "1".
- Check entry $2302_h$ for error code and object $2301_h$, bit 0 = "1" (NanoJ program running).

To stop a running program: write entry $2300_h$ with bit 0 value = "0".

### 9.1.6 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 implemented in the `void user()` function.

**NOTICE**

In *NanoJ programs*, global variables may only be initialized within functions. It then follows:

- No `new` operator
- No constructors
- No initialization of global variables outside of functions

**Examples:**

The global variable is to be initialized within the `void user()` function:

```
unsigned int i;
void user(){
 i = 1;
 i += 1;
}
```

The following assignment results in an error during compilation:

```
unsigned int i = 1;
 void user() {
 i += 1;
}
```

### 9.1.7 NanoJ program example

The example shows the programming of a square wave signal in object $2500_h$:$01_h$.

```
// file main.cpp
map S32 outputReg1 as inout 0x2500:1
#include "wrapper.h"

// user program
void user()
{
  U16 counter = 0;
  while( 1 )
  {
    ++counter;

    if( counter < 100 )
     InOut.outputReg1 = 0;
    else if( counter < 200 )
      InOut.outputReg1 = 1;
    else
      counter = 0;

    // yield() 5 times (delay 5ms)
    for(U08 i = 0; i < 5; ++i )
      yield();
  }
}// eof
```

You can find other examples at us.nanotec.com.

## 9.2 Mapping in the NanoJ program

With this method, a variable in the *NanoJ program* is linked directly with an entry in the object dictionary. The creation of the mapping must be located at the start of the file here, even before the `#include "wrapper.h"` instruction.

**TIP**

Nanotec recommends:

- Use mapping if you need to access an object in the object dictionary frequently, e. g., *controlword* $6040_h$ or *statusword* $6041_h$.
- The `od_write()` and `od_read()` functions are better suited for accessing objects a single time, see Accessing the object dictionary.

### 9.2.1 Declaration of the mapping

The declaration of the mapping is structured as follows:

```
map <TYPE> <NAME> as <input|output|inout> <INDEX>:<SUBINDEX>
```

Where:

- `<TYPE>`

  The data type of the variable; U32, U16, U08, S32, S16 or S08.
- `<NAME>`
  The name of the variable as it is used in the user program.
- `<input|output|inout>`

  The read and write permission of a variable: a variable can be declared as an `input`, `output` or `inout`. This defines whether a variable is readable (`input`), writable (`output`) or both (`inout`) and the structure by means of which it must be addressed in the program.
- `<INDEX>:<SUBINDEX>`

  Index and subindex of the object to be mapped in the object dictionary.

Each declared variable is addressed in the user program via one of the three structures: *In*, *Out* or *InOut* depending on the defined write and read direction.

**NOTICE**

A comment is only permitted above the respective mapping declaration in the code, not on the same line.

### 9.2.2 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;
  [...]
}
```

### 9.2.3 Possible error at `od_write()`

A possible source of errors is a write access with the `od_write()` function (see NanoJ functions in the NanoJ program) of an object in the object dictionary that was simultaneously created as mapping. The code listed in the following is incorrect:

```
map U16 controlWord as output 0x6040:00
```

```
#include " wrapper.h"
void user()
{
 [...]
  Out.controlWord = 1;
  [...]
  od_write(0x6040, 0x00, 5 ); // der Wert wird durch das Mapping überschrieben
  [...]
}
```

The line with the `od_write(0x6040, 0x00, 5 );` command has no effect. As described in the introduction, all mappings are copied to the object dictionary at the end of each millisecond.

This results in the following sequence:

1. The `od_write` function writes the value `5` in object $6040_h$:$00_h$.
2. At the end of the 1 ms cycle, the mapping is written that also specifies object $6040_h$:$00_h$, however, with the value `1`.
3. From the perspective of the user, the `od_write` command thus serves no purpose.

## 9.3 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.

### 9.3.1 Accessing the object dictionary

void **od_write** (U32 index, U32 subindex, U32 value)

This function writes the transferred value to the specified location in the object dictionary.

| | |
|---|---|
| index | Index of the object to be written in the object dictionary |
| subindex | Subindex of the object to be written in the object dictionary |
| value | Value to be written |

**NOTICE**

It is highly recommended that the processor time be passed on with `yield()` after calling a `od_write()`. The value is immediately written to the OD. For the firmware to be able to trigger actions that are dependent on this, however, it must receive computing time. This, in turn, means that the user program must either be ended or interrupted with `yield()`.

U32 **od_read** (U32 index, U32 subindex)

This function reads the value at the specified location in the object dictionary and returns it.

| | |
|---|---|
| index | Index of the object to be read in the object dictionary |
| subindex | Subindex of the object to be read in the object dictionary |
| Output value | Content of the OD entry |

**NOTICE**

Active waiting for a value in the object dictionary should always be associated with a `yield()`.

**Example**

```
while (od_read(2400,2) != 0) // wait until 2400:2 is set
{ yield(); }
```

### 9.3.2 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 |

## 9.4 Restrictions and possible problems

Restrictions and possible problems when working with NanoJ are listed below:

| Restriction/problem | Measure |
|---|---|
| If an object is mapped, e. g., 0x6040, the object is reset to its previous value every 1 ms. This makes it impossible to control this object via the fieldbus or the *Plug & Drive Studio*. | Instead use `od_read` / `od_write` to access the object. |
| If an object was mapped as output and the value of the output was never defined before starting the *NanoJ program*, the value of this object may be random. | Initialize the values of the mapped outputs in your NanoJ program to ensure that it behaves deterministically. |
| The array initialization must not be used with more than 16 entries. | Use `constant array` instead. |
| Too many local variables and arrays within functions may result in a stack overflow. | Declare the variables globally. Memory requirements are monitored already during compilation; errors do not occur at runtime. |
| Functions that are too deeply nested may result in a stack overflow. | Observe a maximum nesting depth of 2. |
| `float` must not be used with comparison operators. | Use `int` instead. |
| `double` must not be used. | |
| If a NanoJ program restarts the controller (either directly with an explicit restart or indirectly, e. g., through the use of the Reset function), the controller may fall into a restart loop that can be exited only with difficulty if at all. | |
| `math` or `cmath` cannot be included. | |

# 10 Description of the object dictionary

## 10.1 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

## 10.2 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"

## 10.3 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.

## 10.4 Value description

**NOTICE**

For the sake of clarity, a number of subindices are grouped together if the entries all have the same name.

Listed in the table with the "Value description" heading are all data for sub-entries with subindex 1 or higher. The table contains the following entries:

**Subindex**

Number of the currently written sub-entry.

**Name**

Name of the sub-entry.

**Data type**

The size and interpretation of the sub-entry is specified here. The following notation always applies here:

- A distinction is made between entries that are signed; these are designated with the prefix "SIGNED". For entries that are unsigned, the prefix "UNSIGNED" is used.
- The size of the variable in bits is placed before the prefix and can be 8, 16 or 32.

**Access**

The access restriction for the sub-entry is entered here. The following restrictions are available:

- "read/write": The object can both be read as well as written
- "read only": The object can only be read from the object dictionary. It is not possible to set a value.

**PDO mapping**

Some bus systems, such as CANopen or EtherCAT, support PDO mapping. Described in this table entry is whether the sub-entry can be inserted into a mapping and, if so, into which. The following designations are available here:

- "no": The object may not be entered in a mapping.
- "TX-PDO": The object may be entered in an RX mapping.
- "RX-PDO": The object may be entered in a TX mapping.

**Allowed values**

In some cases, only certain values may be written in the sub-entry. If this is the case, these values are listed here. If there are no restrictions, the field is empty.

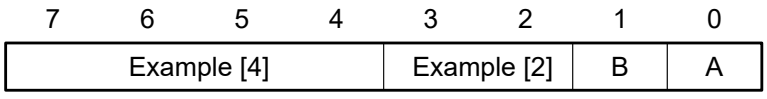
**Preset value**

To bring the controller to a secured state when switching on, it is necessary to preset a number of sub-entries with values. The value that is written in the sub-entry when the controller is started is noted in this table entry.

## 10.5 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.




**Example [4]**

Description of bit 4 up to and including bit 7; these bits are logically related. The 4 in square brackets specifies the number of related bits. A list with possible values and their description is often attached at this point.

**Example [2]**

Description of bits 3 and 2; these bits are logically related. The 2 in square brackets specifies the number of related bits.

- Value $00_b$: The description here applies if bit 2 and bit 3 are "0".
- Value $01_b$: The description here applies if bit 2 is "0" and bit 3 is "1".
- Value $10_b$: The description here applies if bit 2 is "1" and bit 3 is "0".
- Value $11_b$: The description here applies if bit 2 and bit 3 are "1".

**B**

Description of bit B; no length is specified for a single bit.

**A**

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

## 1000h Device Type

### Function

Describes the controller type.

### Object description

| | |
|---|---|
| Index | $1000_h$ |
| Object name | Device Type |
| Object Code | VARIABLE |
| Data type | UNSIGNED32 |
| Savable | no |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00060192_h$ |
| Firmware version | FIR-v1426 |
| Change history | |

### Description

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Motor Type [16] | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Device profile number [16] | | | | | | | | | | | | | | | |

**Motor Type[16]**

Describes the supported motor type. The following values are possible:

- Bit 23 to bit 16: Value "2": BLDC motor
- Bit 23 to bit 16: Value "4": Stepper motor
- Bit 23 to bit 16: Value "6": Stepper motor as well as BLDC motor

**Device profile number[16]**

Describes the supported CANopen standard.

Values:

$0192_h$ or $0402_d$ (preset value): The CiA 402 standard is supported.

## 1001h Error Register

### Function

Error register: In the event of an error, the corresponding error bit(s) is/are set. If the error no longer exists, it is deleted automatically.

**NOTICE**

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

### Object description

| | |
|---|---|
| Index | $1001_h$ |
| Object name | Error Register |
| Object Code | VARIABLE |
| Data type | UNSIGNED8 |
| Savable | no |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00_h$ |
| Firmware version | FIR-v1426 |
| Change history | |

### Description

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| MAN | RES | PROF | COM | TEMP | VOL | CUR | GEN |

**GEN**

General error, always set in the event of an error

**CUR**

Current

**VOL**

Voltage

**TEMP**

Temperature

**COM**

Communication

**PROF**

Relates to the device profile

**RES**

Reserved, always "0"

**MAN**

Manufacturer-specific

## 1003h Pre-defined Error Field

### Function

This object contains an error stack with up to eight entries.

### Object description

| | |
|---|---|
| Index | $1003_h$ |
| Object name | Pre-defined Error Field |
| Object Code | ARRAY |
| Data type | UNSIGNED32 |
| Savable | no |
| Firmware version | FIR-v1426 |
| Change history | |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Errors |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | 1st Standard Error Field |
| Data type | UNSIGNED32 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | 2nd Standard Error Field |
| Data type | UNSIGNED32 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |

| | |
|---|---|
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | 3th Standard Error Field |
| Data type | UNSIGNED32 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | 4th Standard Error Field |
| Data type | UNSIGNED32 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $05_h$ |
| Name | 5th Standard Error Field |
| Data type | UNSIGNED32 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $06_h$ |
| Name | 6th Standard Error Field |
| Data type | UNSIGNED32 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $07_h$ |
| Name | 7th Standard Error Field |
| Data type | UNSIGNED32 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $08_h$ |

| | |
|---|---|
| Name | 8th Standard Error Field |
| Data type | UNSIGNED32 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

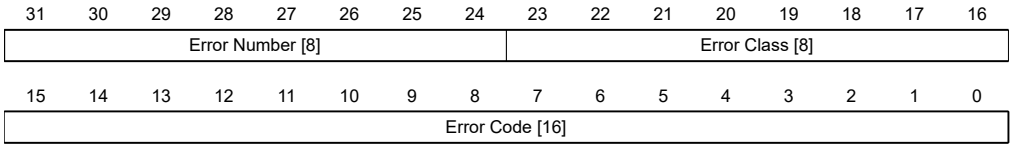
## Description

### General function

If a new error occurs, it is entered in subindex 1. The already existing entries in subindices 1 to 7 are moved back one position. The error in subindex 7 is thereby removed.

The number of errors that have already occurred can be read from the object with subindex 0. If no error is currently entered in the error stack, it is not possible to read one of the eight subindices 1–8 and an error (abort code = $08000024_h$) is sent in response. If a "0" is written in subindex 0, counting starts again from the beginning.

### Bit description




#### Error Number [8]

This can be used to pinpoint the cause of the error. The meaning of the number can be found in the following table.

| Error number | Description |
|---|---|
| 0 | Watchdog-Reset |
| 1 | Input voltage (+Ub) too high |
| 2 | Output current too high |
| 3 | Input voltage (+Ub) too low |
| 4 | Error at fieldbus |
| 5 | Logic voltage too low (< 17 V) to supply the brake. |
| 6 | CANopen only: NMT master takes too long to send Nodeguarding request |
| 7 | Sensor 1 (see $3204_h$): Error through electrical fault or defective hardware |
| 8 | Sensor 2 (see $3204_h$): Error through electrical fault or defective hardware |
| 9 | Sensor 3 (see $3204_h$): Error through electrical fault or defective hardware |
| 10 | Warning: Positive limit switch exceeded |
| 11 | Warning: Negative limit switch exceeded |
| 12 | Overtemperature error |
| 13 | The values of object $6065_h$ (Following Error Window) and object $6066_h$ (Following Error Time Out) were exceeded; a fault was triggered. |
| 14 | Warning: Nonvolatile memory full. The current save process could not be completed; parts of the data of the save process are lost. Controller must be restarted for cleanup work. |
| 15 | Motor blocked |
| 16 | Warning: Nonvolatile memory damaged; controller must be restarted for cleanup work (all saved objects are reset to default). |

| Error number | Description |
|---|---|
| 17 | CANopen only: Slave took too long to send PDO messages. |
| 18 | Sensor n (see $3204_h$), 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 ($6075_h$) |
| 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 |
| 37 | *STO Active*: STO is active, it generates no torque or holding torque. |
| 39 | Error in the ballast configuration: Invalid/unrealistic parameters entered (see Ballast monitoring) |
| 40 | Warning: Ballast resistor thermally overloaded |
| 41 | Only EtherCAT: *Sync Manager Watchdog*: The controller has not received any PDO data for an excessively long period of time; check the software and hardware connections. |
| 43 | Input voltage (+Ub Logic) too high |
| 44 | Input voltage (+Ub Logic) too low |
| 46 | Interlock error: Bit 3 in $60FD_h$ is set to "0", the motor may not start (see the section *Interlock function* in the chapter Digital inputs) |
| 48 | Only CANopen: NMT status has been set to *stopped* |
| 52 | Fieldbus cycle time exceeded. |
| 53 | PROFINET only: wrong stack version |
| 54 | *Sensorless Observer* Error: wrong motor parameter configuration or instability of the observer system |

**Error Class[8]**

This byte is identical to object $1001_h$

**Error Code[16]**

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

| Error Code | Description |
|---|---|
| $1000_h$ | General error |
| $2300_h$ | Current at the controller output too large |
| $3100_h$ | Overvoltage/undervoltage at controller input |
| $3120_h$ | Logic voltage too low (< 17 V) to supply the brake. |
| $4200_h$ | Temperature error within the controller |

| Error Code | Description |
|---|---|
| $5440_h$ | Interlock error: Bit 3 in $60FD_h$ is set to "0", the motor may not start (see the section *Interlock function* in the chapter Digital inputs) |
| $6010_h$ | Software reset (watchdog) |
| $6100_h$ | Internal software error, generic |
| $6320_h$ | Rated current must be set ($6075_h$) |
| $7110_h$ | Error in the ballast configuration: Invalid/unrealistic parameters entered (see Ballast monitoring) |
| $7113_h$ | Warning: Ballast resistor thermally overloaded |
| $7121_h$ | Motor blocked |
| $7200_h$ | Internal error: Correction factor for reference voltage missing in the OTP |
| $7305_h$ | Sensor 1 (see $3204_h$) faulty |
| $7306_h$ | Sensor 2 (see $3204_h$) faulty |
| $7307_h$ | Sensor n (see $3204_h$), where n is greater than 2 |
| $7600_h$ | Warning: Nonvolatile memory full or corrupt; restart the controller for cleanup work |
| $8000_h$ | general error during monitoring of the STO function |
| $8001_h$ | general error during monitoring of the STO function |
| $8002_h$ | general error during monitoring of the STO function |
| $8003_h$ | *STO Self-Test*: Error during self-test of the firmware. Contact Nanotec. |
| $8100_h$ | Error during fieldbus monitoring |
| $8130_h$ | CANopen only: "Life Guard" error or "Heartbeat" error |
| $8200_h$ | CANopen only: Slave took too long to send PDO messages. |
| $8210_h$ | CANopen only: PDO was not processed due to a length error |
| $8220_h$ | CANopen only: PDO length exceeded |
| $8240_h$ | CANopen only: unexpected sync length |
| $8400_h$ | Error in speed monitoring: slippage error too large |
| $8611_h$ | Position monitoring error: Following error too large |
| $8612_h$ | Position monitoring error: Limit switch exceeded |

## 1005h COB-ID Sync

### Function

Defines the COB-ID of the SYNC message for the SYNC protocol. The value must correspond to an 11-bit-long CAN-ID and is evaluated when the controller is restarted or on a Reset Communication command.

**NOTICE**

If the CAN-ID is not to correspond to the default value of $80_h$, it must be ensured that only not-yet unassigned or reserved CAN-IDs are used.

You can activate the generation of sync messages (the controller becomes the *sync master of the network*) by setting bit 30 to "1". Set the cycle time in object $1006_h$.

### Object description

| | |
|---|---|
| Index | $1005_h$ |
| Object name | COB-ID Sync |
| Object Code | VARIABLE |
| Data type | UNSIGNED32 |
| Savable | yes, category: communication |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000080_h$ |
| Firmware version | FIR-v1426 |
| Change history | |

## 1006h Communication Cycle Period

### Function

Contains the cycle time for the generated sync messages (see $1005_h$) in µs. Only multiples of 1000 µs are permitted.

### Object description

| | |
|---|---|
| Index | $1006_h$ |
| Object name | Communication Cycle Period |
| Object Code | VARIABLE |
| Data type | UNSIGNED32 |
| Savable | yes, category: communication |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |
| Firmware version | FIR-v2013-B726332 |
| Change history | |

## 1007h Synchronous Window Length

### Function

This object contains the length of the time window in microseconds for synchronous PDOs. If the synchronous time window has elapsed, all synchronous TxPDOs are rejected. The RxPDOs are also rejected up to the next SYNC message.

The value "0" switches off the time window, thereby allowing the PDOs to be sent at any time.

This object is only available in device variants with CANopen connection.

### Object description

| | |
|---|---|
| Index | $1007_h$ |

| | |
|---|---|
| Object name | Synchronous Window Length |
| Object Code | VARIABLE |
| Data type | UNSIGNED32 |
| Savable | yes, category: communication |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |
| Firmware version | FIR-v1426 |
| Change history | |

## 1008h Manufacturer Device Name

### Function

Contains the device name as character string.

### Object description

| | |
|---|---|
| Index | $1008_h$ |
| Object name | Manufacturer Device Name |
| Object Code | VARIABLE |
| Data type | VISIBLE_STRING |
| Savable | no |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | ■ 0<br>■ N6-1-2-1-S: N6-1-2-1-S<br>■ N6-1-2-1: N6-1-2-1<br>■ N6-2-2-1-S: N6-2-2-1-S<br>■ N6-2-2-1: N6-2-2-1 |
| Firmware version | FIR-v1426 |
| Change history | |

## 1009h Manufacturer Hardware Version

### Function

This object contains the hardware version as character string.

### Object description

| | |
|---|---|
| Index | $1009_h$ |
| Object name | Manufacturer Hardware Version |
| Object Code | VARIABLE |
| Data type | VISIBLE_STRING |
| Savable | no |

| | |
|---|---|
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | 0 |
| Firmware version | FIR-v1426 |
| Change history | |

## 100Ah Manufacturer Software Version

### Function

This object contains the software version as character string.

### Object description

| | |
|---|---|
| Index | $100A_h$ |
| Object name | Manufacturer Software Version |
| Object Code | VARIABLE |
| Data type | VISIBLE_STRING |
| Savable | no |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | FIR-v2542-B1080909 |
| Firmware version | FIR-v1426 |
| Change history | |

## 100Ch Guard Time

### Function

Object $100C_h$ multiplied by object 100Dh Live Time Factor yields the so-called lifetime for the Lifeguarding / Nodeguarding protocol. The value is specified in milliseconds. See also Nodeguarding.

**NOTICE**

The *Heartbeat protocol* has a higher priority than *Nodeguarding*. If both protocols are activated simultaneously, the Node Guarding Timer is suppressed, but no EMCY message is sent either.

### Object description

| | |
|---|---|
| Index | $100C_h$ |
| Object name | Guard Time |
| Object Code | VARIABLE |
| Data type | UNSIGNED16 |
| Savable | yes, category: communication |
| Access | read / write |
| PDO mapping | no |

| | |
|---|---|
| Allowed values | |
| Preset value | $0000_h$ |
| Firmware version | FIR-v1426 |
| Change history | |

## 100Dh Live Time Factor

### Function

This object is a multiplier which, multiplied by object $100C_h$, yields the time window for the *Nodeguarding* protocol in milliseconds. See also Nodeguarding.

**NOTICE**

The *Heartbeat protocol* has a higher priority than *Nodeguarding*. If both protocols are activated simultaneously, the Node Guarding Timer is suppressed, but no EMCY message is sent either.

This object is only available in device variants with CANopen connection.

### Object description

| | |
|---|---|
| Index | $100D_h$ |
| Object name | Live Time Factor |
| Object Code | VARIABLE |
| Data type | UNSIGNED8 |
| Savable | yes, category: communication |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00_h$ |
| Firmware version | FIR-v1426 |
| Change history | |

## 1010h Store Parameters

### Function

This object is used to start the saving of objects. See chapter Saving objects.

### Object description

| | |
|---|---|
| Index | $1010_h$ |
| Object name | Store Parameters |
| Object Code | ARRAY |
| Data type | UNSIGNED32 |
| Savable | no |
| Access | read only |
| PDO mapping | no |
| Allowed values | |

| | |
|---|---|
| Preset value | |
| Firmware version | FIR-v1426 |
| Change history | Firmware version FIR-v1436: "Object name" entry changed from "Store Parameter" to "Store Parameters".<br>Firmware version FIR-v1436: The number of entries was changed from 3 to 4.<br>Firmware version FIR-v1512: The number of entries was changed from 4 to 5.<br>Firmware version FIR-v1540: The number of entries was changed from 5 to 7.<br>Firmware version FIR-v1738-B501312: The number of entries was changed from 7 to 14.<br>Firmware version FIR-v2412-B1057638: "Name" entry changed from "Save Miscellaneous Configurations To Non-volatile Memory" to "Save Reserved0 Configurations To Non-volatile Memory".<br>Firmware version FIR-v2451-B1068465: "Name" entry changed from "Save Drive Parameters To Non-volatile Memory" to "Save Motor Parameters To Non-volatile Memory".<br>Firmware version FIR-v2451-B1068465: "Name" entry changed from "Save Tuning Parameters To Non-volatile Memory" to "Save Sensor Parameters To Non-volatile Memory". |

## Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $0C_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Save All Parameters To Non-volatile Memory |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000001_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Save Communication Parameters To Non-volatile Memory |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |

| | |
|---|---|
| Allowed values | |
| Preset value | $00000001_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | Save Application Parameters To Non-volatile Memory |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000001_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | Save Customer Parameters To Non-volatile Memory |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000001_h$ |

| | |
|---|---|
| Subindex | $05_h$ |
| Name | Save Motor Parameters To Non-volatile Memory |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000001_h$ |

| | |
|---|---|
| Subindex | $06_h$ |
| Name | Save Sensor Parameters To Non-volatile Memory |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000001_h$ |

| | |
|---|---|
| Subindex | $07_h$ |
| Name | Save Reserved0 Configurations To Non-volatile Memory |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $08_h$ |
| Name | Save Reserved1 Configurations To Non-volatile Memory |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $09_h$ |
| Name | Save Reserved2 Configurations To Non-volatile Memory |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $0A_h$ |
| Name | Save CANopen Configurations To Non-volatile Memory |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000001_h$ |

| | |
|---|---|
| Subindex | $0B_h$ |
| Name | Save Modbus RTU Configurations To Non-volatile Memory |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000001_h$ |

| | |
|---|---|
| Subindex | $0C_h$ |
| Name | Save Ethernet Configurations To Non-volatile Memory |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000001_h$ |

## Description

Each subindex of the object stands for a certain memory class. By reading out the entry, it is possible to determine whether (value "1") or not (value="0") this memory category can be saved.

To start the save process of a memory category, value "65766173$_h$" must be written in the corresponding subindex. This corresponds to the decimal of 1702257011$_d$ or the ASCII string `save`. As soon as the saving process is completed, the save command is again overwritten with the value "1", since saving is possible again.

For a detailed description, see chapter Saving objects.

## 1011h Restore Default Parameters

### Function

This object can be used to reset all or part of the object dictionary to the default values. See chapter Saving objects.

### Object description

| | |
|---|---|
| Index | 1011$_h$ |
| Object name | Restore Default Parameters |
| Object Code | ARRAY |
| Data type | UNSIGNED32 |
| Savable | no |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1426 |
| Change history | Firmware version FIR-v1436: "Object Name" entry changed from "Restore Default Parameter" to "Restore Default Parameters".<br>Firmware version FIR-v1436: The number of entries was changed from 2 to 4.<br>Firmware version FIR-v1512: The number of entries was changed from 4 to 5.<br>Firmware version FIR-v1512: "Name" entry changed from "Restore The Comm Default Parameters" to "Restore Communication Default Parameters".<br>Firmware version FIR-v1512: "Name" entry changed from "Restore The Application Default Parameters" to "Restore Application Default Parameters".<br>Firmware version FIR-v1540: The number of entries was changed from 5 to 7.<br>Firmware version FIR-v1738-B501312: The number of entries was changed from 7 to 14.<br>Firmware version FIR-v2412-B1057638: "Name" entry changed from "Restore Miscellaneous Configurations" to "Restore Reserved0 Configurations To Non-volatile Memory".<br>Firmware version FIR-v2451-B1068465: "Name" entry changed from "Restore Drive Default Parameters" to "Restore Motor Default Parameters".<br>Firmware version FIR-v2451-B1068465: "Name" entry changed from "Restore Tuning Default Parameters" to "Restore Sensor Default Parameters". |

## Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $0C_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Restore All Default Parameters |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000001_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Restore Communication Default Parameters |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000001_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | Restore Application Default Parameters |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000001_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | Restore Customer Default Parameters |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000001_h$ |

| | |
|---|---|
| Subindex | $05_h$ |

| | |
|---|---|
| Name | Restore Motor Default Parameters |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000001_h$ |

| | |
|---|---|
| Subindex | $06_h$ |
| Name | Restore Sensor Default Parameters |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000001_h$ |

| | |
|---|---|
| Subindex | $07_h$ |
| Name | Restore Reserved0 Configurations To Non-volatile Memory |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $08_h$ |
| Name | Restore Reserved1 Configurations To Non-volatile Memory |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $09_h$ |
| Name | Restore Reserved2 Configurations To Non-volatile Memory |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $0A_h$ |
| Name | Restore CANopen Configurations To Non-volatile Memory |
| Data type | UNSIGNED32 |
| Access | read / write |

| | |
|---|---|
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000001_h$ |

| | |
|---|---|
| Subindex | $0B_h$ |
| Name | Restore Modbus RTU Configurations To Non-volatile Memory |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000001_h$ |

| | |
|---|---|
| Subindex | $0C_h$ |
| Name | Restore Ethernet Configurations To Non-volatile Memory |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000001_h$ |

### Description

If the value $64616F6C_h$ (or $1684107116_d$ or ASCII `load`) is written in this object, part or all of the object dictionary is reset to the default values. The subindex that is used decides which range is reset.

For a detailed description, see chapter Discarding the saved data.

## 1014h COB-ID EMCY

### Function

This object describes the COB-ID of the "Emergency Service" under CANopen.

With the *Valid Bit* (bit 31) = "1", the Emergency Service can be deactivated; the service is active with the value "0". Every time the controller is restarted, bits 0 to 30 are generated according to the node-ID.

### Object description

| | |
|---|---|
| Index | $1014_h$ |
| Object name | COB-ID EMCY |
| Object Code | VARIABLE |
| Data type | UNSIGNED32 |
| Savable | yes, category: communication |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |
| Firmware version | FIR-v1426 |

| | |
|---|---|
| Change history | Firmware version FIR-v1540: "Access" table entry for subindex 00 changed from "read only" to "read/write". |

## 1016h Consumer Heartbeat Time

### Function

This object defines the cycle time of the *Consumer Heartbeat* of the *Network Management* CANopen service and the Node-ID of the *Producer* of the *Heartbeat*.

If the cycle time or node-ID is set to the value 0, there is no response to the Heartbeat message. See also chapter Heartbeat.

### Object description

| | |
|---|---|
| Index | $1016_h$ |
| Object name | Consumer Heartbeat Time |
| Object Code | ARRAY |
| Data type | UNSIGNED32 |
| Savable | yes, category: communication |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1738-B501312 |
| Change history | |

### Value description

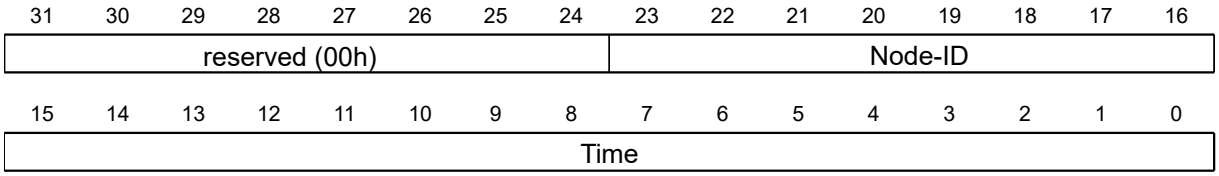
| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $01_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Consumer Heartbeat Time |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

### Description

Subindex $01_h$ contains:

- Bits 0 to 15: The time of the *Consumer Heartbeat* in ms.
- Bits 16 to 23: The node-ID of the *Producer* whose *Heartbeat* is to be monitored.




## 1017h Producer Heartbeat Time

### Function

This object defines the cycle time of the *Heartbeat* of the *Network Management* CANopen service in milliseconds. If the object is set to the value 0, no Heartbeat message is sent. See also Heartbeat.

**NOTICE**

The *Heartbeat protocol* has a higher priority than *Nodeguarding*. If both protocols are activated simultaneously, the Node Guarding Timer is suppressed, but no EMCY message is sent either.

This object is only available in device variants with CANopen connection.

### Object description

| | |
|---|---|
| Index | $1017_h$ |
| Object name | Producer Heartbeat Time |
| Object Code | VARIABLE |
| Data type | UNSIGNED16 |
| Savable | yes, category: communication |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $0000_h$ |
| Firmware version | FIR-v1426 |
| Change history | |

## 1018h Identity Object

### Function

This object returns general information on the device, such as manufacturer, product code, revision and serial number.

**TIP**

Have these values ready in the event of service inquiries.

## Object description

| | |
|---|---|
| Index | $1018_h$ |
| Object name | Identity Object |
| Object Code | RECORD |
| Data type | IDENTITY |
| Savable | no |
| Firmware version | FIR-v1426 |
| Change history | |

## Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $04_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Vendor ID |
| Data type | UNSIGNED32 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $0000026C_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Product Code |
| Data type | UNSIGNED32 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | ■ $00000000_h$<br>■ N6-1-2-1: $00110102_h$<br>■ N6-2-2-1-S: $00120002_h$<br>■ N6-2-2-1: $00120102_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | Revision Number |
| Data type | UNSIGNED32 |
| Access | read only |
| PDO mapping | no |

| | |
|---|---|
| Allowed values | |
| Preset value | $09EE0000_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | Serial Number |
| Data type | UNSIGNED32 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

## 1019h Synchronous Counter Overflow Value

### Function

The value from which the *Sync Counter* is to begin counting anew is entered here. See chapter Synchronization object (SYNC).

### Object description

| | |
|---|---|
| Index | $1019_h$ |
| Object name | Synchronous Counter Overflow Value |
| Object Code | VARIABLE |
| Data type | UNSIGNED8 |
| Savable | yes, category: communication |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00_h$ |
| Firmware version | FIR-v1426 |
| Change history | Firmware Version FIR-v1738-B501312: "Object Name" entry changed from "Synchronous counter overflow value" to "Synchronous Counter Overflow Value".<br>Firmware version FIR-v1738-B501312: "Data type" entry changed from "UNSIGNED16" to "UNSIGNED8". |

### Description

Allowed values: $02_h$ to $F0_h$ and 0. If the value is "0", the sync message is transmitted as a CAN message with a data length of 0.

## 1020h Verify Configuration

### Function

This object indicates the date and time that the configuration was stored.

A configuration tool or a master can use this object to verify the configuration after a reset and, if necessary, perform a new configuration.

The tool must set the date and time before the storage mechanism is started (see chapter Saving objects).

## Object description

| | |
|---|---|
| Index | $1020_h$ |
| Object name | Verify Configuration |
| Object Code | ARRAY |
| Data type | UNSIGNED32 |
| Savable | yes, category: verify |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1540 |
| Change history | |

## Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $02_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Configuration Date |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Configuration Time |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

## Description

Subindex $01_h$ (configuration date) is to contain the number of days since 1 January 1984.

Subindex $02_h$ (configuration time) is to contain the number of milliseconds since midnight.

## 1029h Error Behavior

### Function

This object is used to define what the *NMT state* of the controller should be in case of an error. See also chapter Network Management (NMT).

### Object description

| | |
|---|---|
| Index | $1029_h$ |
| Object name | Error Behavior |
| Object Code | ARRAY |
| Data type | UNSIGNED8 |
| Savable | yes, category: communication |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1738-B501312 |
| Change history | |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $02_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Communication Error |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Internal Device Error |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |

| | |
|---|---|
| Preset value | $01_h$ |

### Description

The subindices have the following function:

- $01_h$: This subindex is used to define how to respond in case of a communication error:
  - Value "00"$_h$: The controller switches to the *Pre-Operational* state (if previously in the *Operational* state).
  - Value "01"$_h$: The controller does not change state.
  - Value "02"$_h$: The controller switches to the *Stopped* state.
- $02_h$: This subindex is used to define how to respond to the remaining errors (except for communication errors):
  - Value "00"$_h$: The controller switches to the *Pre-Operational* state (if previously in the *Operational* state).
  - Value "01"$_h$: The controller does not change state.
  - Value "02"$_h$: The controller switches to the *Stopped* state.

## 1400h Receive PDO 1 Communication Parameter

### Function

Contains the communication parameters for the receiving-side mapping (RX-PDO) in object $1600_h$. See chapter Process Data Object (PDO).

### Object description

| | |
|---|---|
| Index | $1400_h$ |
| Object name | Receive PDO 1 Communication Parameter |
| Object Code | RECORD |
| Data type | PDO_COMMUNICATION_PARAMETER |
| Savable | yes, category: communication |
| Firmware version | FIR-v1426 |
| Change history | |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $02_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | COB-ID |
| Data type | UNSIGNED32 |
| Access | read / write |

| | |
|---|---|
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Transmission Type |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $FF_h$ |

### Description

Subindex $01_h$ (COB-ID): The COB-ID is stored here.

Subindex $02_h$ (transmission type): A number is stored in this subindex that defines the time at which the received data become valid.

For details see chapter on configuring the RX-PDO mapping.

## 1401h Receive PDO 2 Communication Parameter

### Function

Contains the communication parameters for the receiving-side mapping (RX-PDO) in object $1601_h$. See chapter Process Data Object (PDO).

### Object description

| | |
|---|---|
| Index | $1401_h$ |
| Object name | Receive PDO 2 Communication Parameter |
| Object Code | RECORD |
| Data type | PDO_COMMUNICATION_PARAMETER |
| Savable | yes, category: communication |
| Firmware version | FIR-v1426 |
| Change history | |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $02_h$ |

| | |
|---|---|
| Subindex | $01_h$ |

| | |
|---|---|
| Name | COB-ID |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Transmission Type |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $FF_h$ |

### Description

Subindex $01_h$ (COB-ID): The COB-ID is stored here.

Subindex $02_h$ (transmission type): A number is stored in this subindex that defines the time at which the received data become valid.

For details see chapter on configuring the RX-PDO mapping.

## 1402h Receive PDO 3 Communication Parameter

### Function

Contains the communication parameters for the receiving-side mapping (RX-PDO) in object $1602_h$. See chapter Process Data Object (PDO).

### Object description

| | |
|---|---|
| Index | $1402_h$ |
| Object name | Receive PDO 3 Communication Parameter |
| Object Code | RECORD |
| Data type | PDO_COMMUNICATION_PARAMETER |
| Savable | yes, category: communication |
| Firmware version | FIR-v1426 |
| Change history | |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |

| | |
|---|---|
| Preset value | $02_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | COB-ID |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Transmission Type |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $FF_h$ |

### Description

Subindex $01_h$ (COB-ID): The COB-ID is stored here.

Subindex $02_h$ (transmission type): A number is stored in this subindex that defines the time at which the received data become valid.

For details see chapter on configuring the RX-PDO mapping.

## 1403h Receive PDO 4 Communication Parameter

### Function

Contains the communication parameters for the receiving-side mapping (RX-PDO) in object $1603_h$. See chapter Process Data Object (PDO).

### Object description

| | |
|---|---|
| Index | $1403_h$ |
| Object name | Receive PDO 4 Communication Parameter |
| Object Code | RECORD |
| Data type | PDO_COMMUNICATION_PARAMETER |
| Savable | yes, category: communication |
| Firmware version | FIR-v1426 |
| Change history | |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |

| | |
|---|---|
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $02_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | COB-ID |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Transmission Type |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $FF_h$ |

### Description

Subindex $01_h$ (COB-ID): The COB-ID is stored here.

Subindex $02_h$ (transmission type): A number is stored in this subindex that defines the time at which the received data become valid.

For details see chapter on configuring the RX-PDO mapping.

## 1404h Receive PDO 5 Communication Parameter

### Function

Contains the communication parameters for the receiving-side mapping (RX-PDO) in object $1604_h$. See chapter Process Data Object (PDO).

### Object description

| | |
|---|---|
| Index | $1404_h$ |
| Object name | Receive PDO 5 Communication Parameter |
| Object Code | RECORD |
| Data type | PDO_COMMUNICATION_PARAMETER |
| Savable | yes, category: communication |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1614 |

| Change history | |
|---|---|

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $02_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | COB-ID |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $80000000_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Transmission Type |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $FF_h$ |

### Description

Subindex $01_h$ (COB-ID): The COB-ID is stored here.

Subindex $02_h$ (transmission type): A number is stored in this subindex that defines the time at which the received data become valid.

For details see chapter on configuring the RX-PDO mapping.

## 1405h Receive PDO 6 Communication Parameter

### Function

Contains the communication parameters for the receiving-side mapping (RX-PDO) in object $1605_h$. See chapter Process Data Object (PDO).

### Object description

| | |
|---|---|
| Index | $1405_h$ |
| Object name | Receive PDO 6 Communication Parameter |
| Object Code | RECORD |

| | |
|---|---|
| Data type | PDO_COMMUNICATION_PARAMETER |
| Savable | yes, category: communication |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1614 |
| Change history | |

## Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $02_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | COB-ID |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $80000000_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Transmission Type |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $FF_h$ |

## Description

Subindex $01_h$ (COB-ID): The COB-ID is stored here.

Subindex $02_h$ (transmission type): A number is stored in this subindex that defines the time at which the received data become valid.

For details see chapter on configuring the RX-PDO mapping.

## 1406h Receive PDO 7 Communication Parameter

### Function

Contains the communication parameters for the receiving-side mapping (RX-PDO) in object $1606_h$. See chapter Process Data Object (PDO).

### Object description

| | |
|---|---|
| Index | $1406_h$ |
| Object name | Receive PDO 7 Communication Parameter |
| Object Code | RECORD |
| Data type | PDO_COMMUNICATION_PARAMETER |
| Savable | yes, category: communication |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1614 |
| Change history | |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $02_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | COB-ID |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $80000000_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Transmission Type |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $FF_h$ |

### Description

Subindex $01_h$ (COB-ID): The COB-ID is stored here.

Subindex $02_h$ (transmission type): A number is stored in this subindex that defines the time at which the received data become valid.

For details see chapter on configuring the RX-PDO mapping.

## 1407h Receive PDO 8 Communication Parameter

### Function

Contains the communication parameters for the receiving-side mapping (RX-PDO) in object $1607_h$. See chapter Process Data Object (PDO).

### Object description

| | |
|---|---|
| Index | $1407_h$ |
| Object name | Receive PDO 8 Communication Parameter |
| Object Code | RECORD |
| Data type | PDO_COMMUNICATION_PARAMETER |
| Savable | yes, category: communication |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1614 |
| Change history | |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $02_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | COB-ID |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $80000000_h$ |

| | |
|---|---|
| Subindex | $02_h$ |

| | |
|---|---|
| Name | Transmission Type |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $FF_h$ |

### Description

Subindex $01_h$ (COB-ID): The COB-ID is stored here.

Subindex $02_h$ (transmission type): A number is stored in this subindex that defines the time at which the received data become valid.

For details see chapter on configuring the RX-PDO mapping.

## 1600h Receive PDO 1 Mapping Parameter

### Function

This object contains the mapping parameters for PDOs that the controller can receive (RX-PDO 1). The PDO was previously configured via 1400h Receive PDO 1 Communication Parameter. See chapter Process Data Object (PDO).

### Object description

| | |
|---|---|
| Index | $1600_h$ |
| Object name | Receive PDO 1 Mapping Parameter |
| Object Code | RECORD |
| Data type | PDO_MAPPING |
| Savable | yes, category: communication |
| Firmware version | FIR-v1426 |
| Change history | Firmware version FIR-v1426: "Heading" entry changed from "1600h Drive Control" to "1600h Receive PDO 1 Mapping Parameter".<br>Firmware version FIR-v1426: "Object Name" entry changed from "Drive Control" to "Receive PDO 1 Mapping Parameter". |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $02_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | 1st Object To Be Mapped |

| | |
|---|---|
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $60400010_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | 2nd Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $60600008_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | 3rd Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | 4th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $05_h$ |
| Name | 5th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $06_h$ |
| Name | 6th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |

| | |
|---|---|
| Allowed values | |
| Preset value | $00000000_h$ |

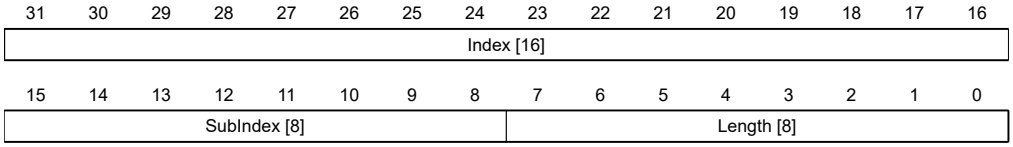
| | |
|---|---|
| Subindex | $07_h$ |
| Name | 7th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $08_h$ |
| Name | 8th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

### Description

Each subindex (1–8) 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.

## 1601h Receive PDO 2 Mapping Parameter

### Function

This object contains the mapping parameters for PDOs that the controller can receive (RX-PDO 2). The PDO was previously configured via 1401h Receive PDO 2 Communication Parameter. See chapter Process Data Object (PDO).

### Object description

| | |
|---|---|
| Index | $1601_h$ |

| | |
|---|---|
| Object name | Receive PDO 2 Mapping Parameter |
| Object Code | RECORD |
| Data type | PDO_MAPPING |
| Savable | yes, category: communication |
| Firmware version | FIR-v1426 |
| Change history | Firmware version FIR-v1426: "Heading" entry changed from "1601h Positioning Control" to "1601h Receive PDO 2 Mapping Parameter".<br>Firmware version FIR-v1426: "Object Name" entry changed from "Positioning Control" to "Receive PDO 2 Mapping Parameter". |

## Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $02_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | 1st Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $607A0020_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | 2nd Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $60810020_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | 3rd Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | 4th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $05_h$ |
| Name | 5th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $06_h$ |
| Name | 6th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

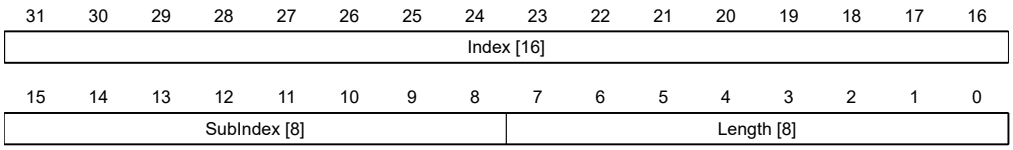
| | |
|---|---|
| Subindex | $07_h$ |
| Name | 7th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $08_h$ |
| Name | 8th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

## Description

Each subindex (1–8) 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.

## 1602h Receive PDO 3 Mapping Parameter

### Function

This object contains the mapping parameters for PDOs that the controller can receive (RX-PDO 3). The PDO was previously configured via 1402h Receive PDO 3 Communication Parameter. See chapter Process Data Object (PDO).

### Object description

| | |
|---|---|
| Index | $1602_h$ |
| Object name | Receive PDO 3 Mapping Parameter |
| Object Code | RECORD |
| Data type | PDO_MAPPING |
| Savable | yes, category: communication |
| Firmware version | FIR-v1426 |
| Change history | Firmware version FIR-v1426: "Heading" entry changed from "1602h Velocity Control" to "1602h Receive PDO 3 Mapping Parameter".<br>Firmware version FIR-v1426: "Object Name" entry changed from "Velocity Control" to "Receive PDO 3 Mapping Parameter". |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $01_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | 1st Object To Be Mapped |
| Data type | UNSIGNED32 |

| | |
|---|---|
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $60FF0020_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | 2nd Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | 3rd Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | 4th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $05_h$ |
| Name | 5th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $06_h$ |
| Name | 6th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |

| | |
|---|---|
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $07_h$ |
| Name | 7th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $08_h$ |
| Name | 8th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

## 1603h Receive PDO 4 Mapping Parameter

### Function

This object contains the mapping parameters for PDOs that the controller can receive (RX-PDO 4). The PDO was previously configured via 1403h Receive PDO 4 Communication Parameter. See chapter Process Data Object (PDO).

### Object description

| | |
|---|---|
| Index | $1603_h$ |
| Object name | Receive PDO 4 Mapping Parameter |
| Object Code | RECORD |
| Data type | PDO_MAPPING |
| Savable | yes, category: communication |
| Firmware version | FIR-v1426 |
| Change history | Firmware version FIR-v1426: "Heading" entry changed from "1603h Output Control" to "1603h Receive PDO 4 Mapping Parameter".<br>Firmware version FIR-v1426: "Object Name" entry changed from "Output Control" to "Receive PDO 4 Mapping Parameter". |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read / write |

| | |
|---|---|
| PDO mapping | no |
| Allowed values | |
| Preset value | $01_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | 1st Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $60FE0120_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | 2nd Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | 3rd Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | 4th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $05_h$ |
| Name | 5th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $06_h$ |
| Name | 6th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $07_h$ |
| Name | 7th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $08_h$ |
| Name | 8th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

## 1604h Receive PDO 5 Mapping Parameter

### Function

This object contains the mapping parameters for PDOs that the controller can receive (RX-PDO 5). The PDO was previously configured via 1404h Receive PDO 5 Communication Parameter. See chapter Process Data Object (PDO).

### Object description

| | |
|---|---|
| Index | $1604_h$ |
| Object name | Receive PDO 5 Mapping Parameter |
| Object Code | RECORD |
| Data type | PDO_MAPPING |
| Savable | yes, category: communication |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1614 |
| Change history | |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | 1st Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | 2nd Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | 3rd Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | 4th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $05_h$ |

| | |
|---|---|
| Name | 5th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $06_h$ |
| Name | 6th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $07_h$ |
| Name | 7th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $08_h$ |
| Name | 8th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

## 1605h Receive PDO 6 Mapping Parameter

### Function

This object contains the mapping parameters for PDOs that the controller can receive (RX-PDO 6). The PDO was previously configured via 1405h Receive PDO 6 Communication Parameter. See chapter Process Data Object (PDO).

### Object description

| | |
|---|---|
| Index | $1605_h$ |
| Object name | Receive PDO 6 Mapping Parameter |
| Object Code | RECORD |
| Data type | PDO_MAPPING |
| Savable | yes, category: communication |

| | |
|---|---|
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1614 |
| Change history | |

## Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | 1st Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | 2nd Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | 3rd Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | 4th Object To Be Mapped |

| | |
|---|---|
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $05_h$ |
| Name | 5th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $06_h$ |
| Name | 6th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $07_h$ |
| Name | 7th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $08_h$ |
| Name | 8th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

## 1606h Receive PDO 7 Mapping Parameter

### Function

This object contains the mapping parameters for PDOs that the controller can receive (RX-PDO 7). The PDO was previously configured via 1406h Receive PDO 7 Communication Parameter. See chapter Process Data Object (PDO).

## Object description

| | |
|---|---|
| Index | $1606_h$ |
| Object name | Receive PDO 7 Mapping Parameter |
| Object Code | RECORD |
| Data type | PDO_MAPPING |
| Savable | yes, category: communication |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1614 |
| Change history | |

## Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | 1st Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | 2nd Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | 3rd Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |

| | |
|---|---|
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | 4th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $05_h$ |
| Name | 5th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $06_h$ |
| Name | 6th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $07_h$ |
| Name | 7th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $08_h$ |
| Name | 8th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

## 1607h Receive PDO 8 Mapping Parameter

### Function

This object contains the mapping parameters for PDOs that the controller can receive (RX-PDO 8). The PDO was previously configured via 1407h Receive PDO 8 Communication Parameter. See chapter Process Data Object (PDO).

### Object description

| | |
|---|---|
| Index | $1607_h$ |
| Object name | Receive PDO 8 Mapping Parameter |
| Object Code | RECORD |
| Data type | PDO_MAPPING |
| Savable | yes, category: communication |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1614 |
| Change history | |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | 1st Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | 2nd Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | 3rd Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | 4th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $05_h$ |
| Name | 5th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $06_h$ |
| Name | 6th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $07_h$ |
| Name | 7th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $08_h$ |
| Name | 8th Object To Be Mapped |
| Data type | UNSIGNED32 |

| | |
|---|---|
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

## 1800h Transmit PDO 1 Communication Parameter

### Function

Contains the communication parameters for the sending-side mapping (TX-PDO) 1. See chapter Process Data Object (PDO).

### Object description

| | |
|---|---|
| Index | $1800_h$ |
| Object name | Transmit PDO 1 Communication Parameter |
| Object Code | RECORD |
| Data type | PDO_COMMUNICATION_PARAMETER |
| Savable | yes, category: communication |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1426 |
| Change history | Firmware version FIR-v1738-B501312: The number of entries was changed from 6 to 7. |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $06_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | COB-ID |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Transmission Type |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $FF_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | Inhibit Time |
| Data type | UNSIGNED16 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $0064_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | Compatibility Entry |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00_h$ |

| | |
|---|---|
| Subindex | $05_h$ |
| Name | Event Timer |
| Data type | UNSIGNED16 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $0000_h$ |

| | |
|---|---|
| Subindex | $06_h$ |
| Name | SYNC Start Value |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00_h$ |

## Description

Subindex $01_h$ (COB-ID): The COB-ID is stored here.

Subindex $02_h$ (transmission type): A number is stored in this subindex that defines the time at which the received data become valid.

Subindex 3 (inhibit time): This can be used to set a time in 100 µs steps that must elapse after the sending of a PDO before the PDO is sent another time. This time only applies for asynchronous PDOs.

Subindex 4 (compatibility entry): This subindex has no function and exists only for compatibility reasons.

Subindex 5 (event timer): This time (in ms) can be used to trigger an *Event* which handles the copying of the data and the sending of the PDO.

Subindex 6 (sync start value): Here, the start value of the *Sync Counter* is entered beginning with which the *slave* is to initially respond to the sync and send the PDO. Thereafter, sending of the PDO is only dependent on *Transmission Type* (subindex $02_h$). Not globally activated until a value greater than 1 is set in 1019h Synchronous Counter Overflow Value.

For details, see chapter on configuring the Tx-PDO mapping.

## 1801h Transmit PDO 2 Communication Parameter

### Function

Contains the communication parameters for the sending-side mapping (TX-PDO) 2. See chapter Process Data Object (PDO).

### Object description

| | |
|---|---|
| Index | $1801_h$ |
| Object name | Transmit PDO 2 Communication Parameter |
| Object Code | RECORD |
| Data type | PDO_COMMUNICATION_PARAMETER |
| Savable | yes, category: communication |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1426 |
| Change history | Firmware version FIR-v1738-B501312: The number of entries was changed from 6 to 7. |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $06_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | COB-ID |

| | |
|---|---|
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Transmission Type |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $FF_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | Inhibit Time |
| Data type | UNSIGNED16 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $0064_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | Compatibility Entry |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00_h$ |

| | |
|---|---|
| Subindex | $05_h$ |
| Name | Event Timer |
| Data type | UNSIGNED16 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $0000_h$ |

| | |
|---|---|
| Subindex | $06_h$ |
| Name | SYNC Start Value |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |

| | |
|---|---|
| Allowed values | |
| Preset value | $00_h$ |

## Description

Subindex $01_h$ (COB-ID): The COB-ID is stored here.

Subindex $02_h$ (transmission type): A number is stored in this subindex that defines the time at which the received data become valid.

Subindex 3 (inhibit time): This can be used to set a time in 100 µs steps that must elapse after the sending of a PDO before the PDO is sent another time. This time only applies for asynchronous PDOs.

Subindex 4 (compatibility entry): This subindex has no function and exists only for compatibility reasons.

Subindex 5 (event timer): This time (in ms) can be used to trigger an *Event* which handles the copying of the data and the sending of the PDO.

Subindex 6 (sync start value): Here, the start value of the *Sync Counter* is entered beginning with which the *slave* is to initially respond to the sync and send the PDO. Thereafter, sending of the PDO is only dependent on *Transmission Type* (subindex $02_h$). Not globally activated until a value greater than 1 is set in 1019h Synchronous Counter Overflow Value.

For details, see chapter on configuring the Tx-PDO mapping.

# 1802h Transmit PDO 3 Communication Parameter

## Function

Contains the communication parameters for the sending-side mapping (TX-PDO) 3. See chapter Process Data Object (PDO).

## Object description

| | |
|---|---|
| Index | $1802_h$ |
| Object name | Transmit PDO 3 Communication Parameter |
| Object Code | RECORD |
| Data type | PDO_COMMUNICATION_PARAMETER |
| Savable | yes, category: communication |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1426 |
| Change history | Firmware version FIR-v1738-B501312: The number of entries was changed from 6 to 7. |

## Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |

| | |
|---|---|
| Allowed values | |
| Preset value | $06_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | COB-ID |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Transmission Type |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $FF_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | Inhibit Time |
| Data type | UNSIGNED16 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $0064_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | Compatibility Entry |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00_h$ |

| | |
|---|---|
| Subindex | $05_h$ |
| Name | Event Timer |
| Data type | UNSIGNED16 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $0000_h$ |

| | |
|---|---|
| Subindex | $06_h$ |
| Name | SYNC Start Value |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00_h$ |

### Description

Subindex $01_h$ (COB-ID): The COB-ID is stored here.

Subindex $02_h$ (transmission type): A number is stored in this subindex that defines the time at which the received data become valid.

Subindex 3 (inhibit time): This can be used to set a time in 100 µs steps that must elapse after the sending of a PDO before the PDO is sent another time. This time only applies for asynchronous PDOs.

Subindex 4 (compatibility entry): This subindex has no function and exists only for compatibility reasons.

Subindex 5 (event timer): This time (in ms) can be used to trigger an *Event* which handles the copying of the data and the sending of the PDO.

Subindex 6 (sync start value): Here, the start value of the *Sync Counter* is entered beginning with which the *slave* is to initially respond to the sync and send the PDO. Thereafter, sending of the PDO is only dependent on *Transmission Type* (subindex $02_h$). Not globally activated until a value greater than 1 is set in 1019h Synchronous Counter Overflow Value.

For details, see chapter on configuring the Tx-PDO mapping.

## 1803h Transmit PDO 4 Communication Parameter

### Function

Contains the communication parameters for the sending-side mapping (TX-PDO) 4. See chapter Process Data Object (PDO).

### Object description

| | |
|---|---|
| Index | $1803_h$ |
| Object name | Transmit PDO 4 Communication Parameter |
| Object Code | RECORD |
| Data type | PDO_COMMUNICATION_PARAMETER |
| Savable | yes, category: communication |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1426 |
| Change history | Firmware version FIR-v1738-B501312: The number of entries was changed from 6 to 7. |

## Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $06_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | COB-ID |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Transmission Type |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $FF_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | Inhibit Time |
| Data type | UNSIGNED16 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $0064_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | Compatibility Entry |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00_h$ |

| | |
|---|---|
| Subindex | $05_h$ |

| | |
|---|---|
| Name | Event Timer |
| Data type | UNSIGNED16 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $0000_h$ |

| | |
|---|---|
| Subindex | $06_h$ |
| Name | SYNC Start Value |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00_h$ |

### Description

Subindex $01_h$ (COB-ID): The COB-ID is stored here.

Subindex $02_h$ (transmission type): A number is stored in this subindex that defines the time at which the received data become valid.

Subindex 3 (inhibit time): This can be used to set a time in 100 µs steps that must elapse after the sending of a PDO before the PDO is sent another time. This time only applies for asynchronous PDOs.

Subindex 4 (compatibility entry): This subindex has no function and exists only for compatibility reasons.

Subindex 5 (event timer): This time (in ms) can be used to trigger an *Event* which handles the copying of the data and the sending of the PDO.

Subindex 6 (sync start value): Here, the start value of the *Sync Counter* is entered beginning with which the *slave* is to initially respond to the sync and send the PDO. Thereafter, sending of the PDO is only dependent on *Transmission Type* (subindex $02_h$). Not globally activated until a value greater than 1 is set in 1019h Synchronous Counter Overflow Value.

For details, see chapter on configuring the Tx-PDO mapping.

## 1804h Transmit PDO 5 Communication Parameter

### Function

Contains the communication parameters for the sending-side mapping (TX-PDO) 5. See chapter Process Data Object (PDO).

### Object description

| | |
|---|---|
| Index | $1804_h$ |
| Object name | Transmit PDO 5 Communication Parameter |
| Object Code | RECORD |
| Data type | PDO_COMMUNICATION_PARAMETER |
| Savable | yes, category: communication |
| Access | read only |
| PDO mapping | no |
| Allowed values | |

| | |
|---|---|
| Preset value | |
| Firmware version | FIR-v1614 |
| Change history | Firmware version FIR-v1738-B501312: The number of entries was changed from 6 to 7. |

## Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $06_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | COB-ID |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $C0000000_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Transmission Type |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $FF_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | Inhibit Time |
| Data type | UNSIGNED16 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $0064_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | Compatibility Entry |
| Data type | UNSIGNED8 |
| Access | read / write |

| | |
|---|---|
| PDO mapping | no |
| Allowed values | |
| Preset value | $00_h$ |

| | |
|---|---|
| Subindex | $05_h$ |
| Name | Event Timer |
| Data type | UNSIGNED16 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $0000_h$ |

| | |
|---|---|
| Subindex | $06_h$ |
| Name | SYNC Start Value |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00_h$ |

### Description

Subindex $01_h$ (COB-ID): The COB-ID is stored here.

Subindex $02_h$ (transmission type): A number is stored in this subindex that defines the time at which the received data become valid.

Subindex 3 (inhibit time): This can be used to set a time in 100 µs steps that must elapse after the sending of a PDO before the PDO is sent another time. This time only applies for asynchronous PDOs.

Subindex 4 (compatibility entry): This subindex has no function and exists only for compatibility reasons.

Subindex 5 (event timer): This time (in ms) can be used to trigger an *Event* which handles the copying of the data and the sending of the PDO.

Subindex 6 (sync start value): Here, the start value of the *Sync Counter* is entered beginning with which the *slave* is to initially respond to the sync and send the PDO. Thereafter, sending of the PDO is only dependent on *Transmission Type* (subindex $02_h$). Not globally activated until a value greater than 1 is set in 1019h Synchronous Counter Overflow Value.

For details, see chapter on configuring the Tx-PDO mapping.

## 1805h Transmit PDO 6 Communication Parameter

### Function

Contains the communication parameters for the sending-side mapping (TX-PDO) 6. See chapter Process Data Object (PDO).

### Object description

| | |
|---|---|
| Index | $1805_h$ |
| Object name | Transmit PDO 6 Communication Parameter |

| | |
|---|---|
| Object Code | RECORD |
| Data type | PDO_COMMUNICATION_PARAMETER |
| Savable | yes, category: communication |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1614 |
| Change history | Firmware version FIR-v1738-B501312: The number of entries was changed from 6 to 7. |

## Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $06_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | COB-ID |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $C0000000_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Transmission Type |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $FF_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | Inhibit Time |
| Data type | UNSIGNED16 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $0064_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | Compatibility Entry |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00_h$ |

| | |
|---|---|
| Subindex | $05_h$ |
| Name | Event Timer |
| Data type | UNSIGNED16 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $0000_h$ |

| | |
|---|---|
| Subindex | $06_h$ |
| Name | SYNC Start Value |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00_h$ |

### Description

Subindex $01_h$ (COB-ID): The COB-ID is stored here.

Subindex $02_h$ (transmission type): A number is stored in this subindex that defines the time at which the received data become valid.

Subindex 3 (inhibit time): This can be used to set a time in 100 µs steps that must elapse after the sending of a PDO before the PDO is sent another time. This time only applies for asynchronous PDOs.

Subindex 4 (compatibility entry): This subindex has no function and exists only for compatibility reasons.

Subindex 5 (event timer): This time (in ms) can be used to trigger an *Event* which handles the copying of the data and the sending of the PDO.

Subindex 6 (sync start value): Here, the start value of the *Sync Counter* is entered beginning with which the *slave* is to initially respond to the sync and send the PDO. Thereafter, sending of the PDO is only dependent on *Transmission Type* (subindex $02_h$). Not globally activated until a value greater than 1 is set in 1019h Synchronous Counter Overflow Value.

For details, see chapter on configuring the Tx-PDO mapping.

## 1806h Transmit PDO 7 Communication Parameter

### Function

Contains the communication parameters for the sending-side mapping (TX-PDO) 7. See chapter Process Data Object (PDO).

## Object description

| | |
|---|---|
| Index | $1806_h$ |
| Object name | Transmit PDO 7 Communication Parameter |
| Object Code | RECORD |
| Data type | PDO_COMMUNICATION_PARAMETER |
| Savable | yes, category: communication |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1614 |
| Change history | Firmware version FIR-v1738-B501312: The number of entries was changed from 6 to 7. |

## Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $06_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | COB-ID |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $C0000000_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Transmission Type |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $FF_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | Inhibit Time |
| Data type | UNSIGNED16 |

| | |
|---|---|
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $0064_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | Compatibility Entry |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00_h$ |

| | |
|---|---|
| Subindex | $05_h$ |
| Name | Event Timer |
| Data type | UNSIGNED16 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $0000_h$ |

| | |
|---|---|
| Subindex | $06_h$ |
| Name | SYNC Start Value |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00_h$ |

## Description

Subindex $01_h$ (COB-ID): The COB-ID is stored here.

Subindex $02_h$ (transmission type): A number is stored in this subindex that defines the time at which the received data become valid.

Subindex 3 (inhibit time): This can be used to set a time in 100 µs steps that must elapse after the sending of a PDO before the PDO is sent another time. This time only applies for asynchronous PDOs.

Subindex 4 (compatibility entry): This subindex has no function and exists only for compatibility reasons.

Subindex 5 (event timer): This time (in ms) can be used to trigger an *Event* which handles the copying of the data and the sending of the PDO.

Subindex 6 (sync start value): Here, the start value of the *Sync Counter* is entered beginning with which the *slave* is to initially respond to the sync and send the PDO. Thereafter, sending of the PDO is only dependent on *Transmission Type* (subindex $02_h$). Not globally activated until a value greater than 1 is set in 1019h Synchronous Counter Overflow Value.

For details, see chapter on configuring the Tx-PDO mapping.

## 1807h Transmit PDO 8 Communication Parameter

### Function

Contains the communication parameters for the sending-side mapping (TX-PDO) 8. See chapter Process Data Object (PDO).

### Object description

| | |
|---|---|
| Index | $1807_h$ |
| Object name | Transmit PDO 8 Communication Parameter |
| Object Code | RECORD |
| Data type | PDO_COMMUNICATION_PARAMETER |
| Savable | yes, category: communication |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1614 |
| Change history | Firmware version FIR-v1738-B501312: The number of entries was changed from 6 to 7. |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $06_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | COB-ID |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $C0000000_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Transmission Type |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |

| | |
|---|---|
| Preset value | $FF_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | Inhibit Time |
| Data type | UNSIGNED16 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $0064_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | Compatibility Entry |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00_h$ |

| | |
|---|---|
| Subindex | $05_h$ |
| Name | Event Timer |
| Data type | UNSIGNED16 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $0000_h$ |

| | |
|---|---|
| Subindex | $06_h$ |
| Name | SYNC Start Value |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00_h$ |

## Description

Subindex $01_h$ (COB-ID): The COB-ID is stored here.

Subindex $02_h$ (transmission type): A number is stored in this subindex that defines the time at which the received data become valid.

Subindex 3 (inhibit time): This can be used to set a time in 100 µs steps that must elapse after the sending of a PDO before the PDO is sent another time. This time only applies for asynchronous PDOs.

Subindex 4 (compatibility entry): This subindex has no function and exists only for compatibility reasons.

Subindex 5 (event timer): This time (in ms) can be used to trigger an *Event* which handles the copying of the data and the sending of the PDO.

Subindex 6 (sync start value): Here, the start value of the *Sync Counter* is entered beginning with which the *slave* is to initially respond to the sync and send the PDO. Thereafter, sending of the PDO is only dependent on *Transmission Type* (subindex $02_h$). Not globally activated until a value greater than 1 is set in 1019h Synchronous Counter Overflow Value.

For details, see chapter on configuring the Tx-PDO mapping.

## 1A00h Transmit PDO 1 Mapping Parameter

### Function

This object contains the mapping parameters for PDOs that the controller can send (TX-PDO 1). The PDO was previously configured via 1800h Transmit PDO 1 Communication Parameter. See chapter Process Data Object (PDO).

### Object description

| | |
|---|---|
| Index | $1A00_h$ |
| Object name | Transmit PDO 1 Mapping Parameter |
| Object Code | RECORD |
| Data type | PDO_MAPPING |
| Savable | yes, category: communication |
| Firmware version | FIR-v1426 |
| Change history | Firmware version FIR-v1426: "Heading" entry changed from "1A00h Drive Status" to "1A00h Transmit PDO 1 Mapping Parameter".<br>Firmware version FIR-v1426: "Object Name" entry changed from "Drive Status" to "Transmit PDO 1 Mapping Parameter". |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $02_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | 1st Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $60410010_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | 2nd Object To Be Mapped |

| | |
|---|---|
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $60610008_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | 3rd Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | 4th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $05_h$ |
| Name | 5th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $06_h$ |
| Name | 6th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $07_h$ |
| Name | 7th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |

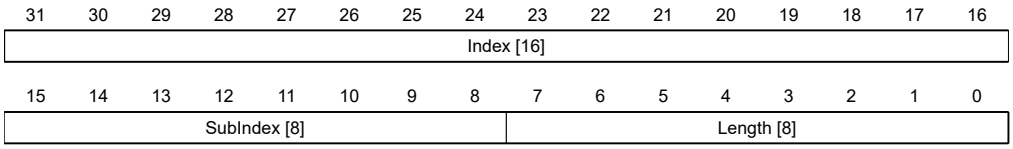
| | |
|---|---|
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $08_h$ |
| Name | 8th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

### Description

Each subindex (1–8) 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.

## 1A01h Transmit PDO 2 Mapping Parameter

### Function

This object contains the mapping parameters for PDOs that the controller can send (TX-PDO 2). The PDO was previously configured via 1801h Transmit PDO 2 Communication Parameter. See chapter Process Data Object (PDO).

### Object description

| | |
|---|---|
| Index | $1A01_h$ |
| Object name | Transmit PDO 2 Mapping Parameter |
| Object Code | RECORD |
| Data type | PDO_MAPPING |
| Savable | yes, category: communication |
| Firmware version | FIR-v1426 |
| Change history | Firmware version FIR-v1426: "Heading" entry changed from "1A01h Positioning Status" to "1A01h Transmit PDO 2 Mapping Parameter". |

| | |
|---|---|
| | Firmware version FIR-v1426: "Object Name" entry changed from "Positioning Status" to "Transmit PDO 2 Mapping Parameter". |

## Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $01_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | 1st Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $60640020_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | 2nd Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | 3rd Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | 4th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |

| | |
|---|---|
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $05_h$ |
| Name | 5th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $06_h$ |
| Name | 6th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

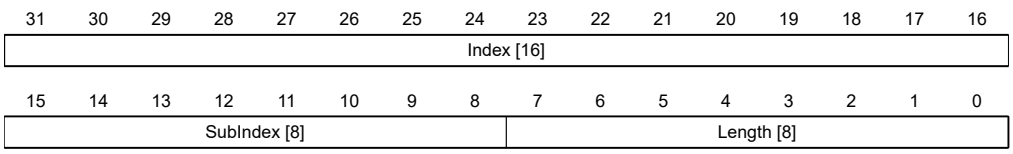
| | |
|---|---|
| Subindex | $07_h$ |
| Name | 7th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $08_h$ |
| Name | 8th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

## Description

Each subindex (1–8) 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.

## 1A02h Transmit PDO 3 Mapping Parameter

### Function

This object contains the mapping parameters for PDOs that the controller can send (TX-PDO 3). The PDO was previously configured via 1802h Transmit PDO 3 Communication Parameter. See chapter Process Data Object (PDO).

### Object description

| | |
|---|---|
| Index | $1A02_h$ |
| Object name | Transmit PDO 3 Mapping Parameter |
| Object Code | RECORD |
| Data type | PDO_MAPPING |
| Savable | yes, category: communication |
| Firmware version | FIR-v1426 |
| Change history | Firmware version FIR-v1426: "Heading" entry changed from "1A02h Velocity Status" to "1A02h Transmit PDO 3 Mapping Parameter".<br>Firmware version FIR-v1426: "Object Name" entry changed from "Velocity Status" to "Transmit PDO 3 Mapping Parameter". |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $01_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | 1st Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $606C0020_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | 2nd Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | 3rd Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | 4th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $05_h$ |
| Name | 5th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $06_h$ |
| Name | 6th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $07_h$ |
| Name | 7th Object To Be Mapped |
| Data type | UNSIGNED32 |

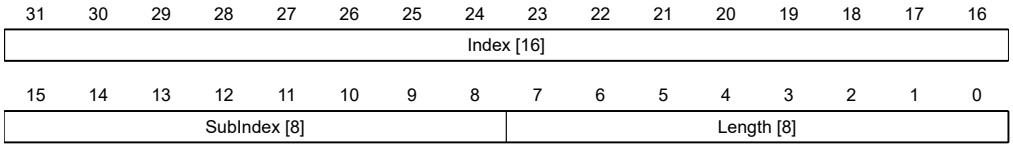
| | |
|---|---|
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $08_h$ |
| Name | 8th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

### Description

Each subindex (1–8) 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.

## 1A03h Transmit PDO 4 Mapping Parameter

### Function

This object contains the mapping parameters for PDOs that the controller can send (TX-PDO 4). The PDO was previously configured via 1803h Transmit PDO 4 Communication Parameter. See chapter Process Data Object (PDO).

### Object description

| | |
|---|---|
| Index | $1A03_h$ |
| Object name | Transmit PDO 4 Mapping Parameter |
| Object Code | RECORD |
| Data type | PDO_MAPPING |
| Savable | yes, category: communication |
| Firmware version | FIR-v1426 |
| Change history | Firmware version FIR-v1426: "Heading" entry changed from "1A03h Input Status" to "1A03h Transmit PDO 4 Mapping Parameter". |

| | |
|---|---|
| | Firmware version FIR-v1426: "Object Name" entry changed from "Input Status" to "Transmit PDO 4 Mapping Parameter". |

## Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $01_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | 1st Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $60FD0020_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | 2nd Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | 3rd Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | 4th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |

| | |
|---|---|
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $05_h$ |
| Name | 5th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $06_h$ |
| Name | 6th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

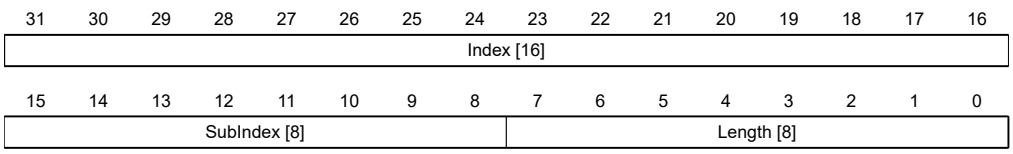
| | |
|---|---|
| Subindex | $07_h$ |
| Name | 7th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $08_h$ |
| Name | 8th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

## Description

Each subindex (1–8) 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.

## 1A04h Transmit PDO 5 Mapping Parameter

### Function

This object contains the mapping parameters for PDOs that the controller can send (TX-PDO 5). The PDO was previously configured via 1804h Transmit PDO 5 Communication Parameter. See chapter Process Data Object (PDO).

### Object description

| | |
|---|---|
| Index | $1A04_h$ |
| Object name | Transmit PDO 5 Mapping Parameter |
| Object Code | RECORD |
| Data type | PDO_MAPPING |
| Savable | yes, category: communication |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1614 |
| Change history | |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | 1st Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | 2nd Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | 3rd Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | 4th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $05_h$ |
| Name | 5th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $06_h$ |
| Name | 6th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $07_h$ |
| Name | 7th Object To Be Mapped |
| Data type | UNSIGNED32 |

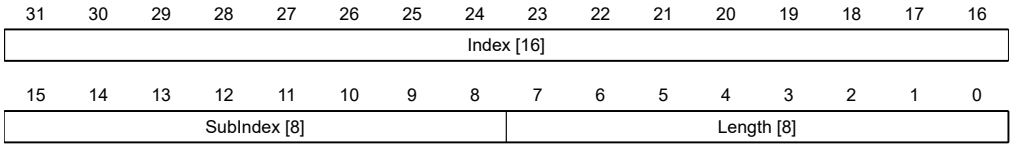
| | |
|---|---|
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $08_h$ |
| Name | 8th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

### Description

Each subindex (1–8) 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.

## 1A05h Transmit PDO 6 Mapping Parameter

### Function

This object contains the mapping parameters for PDOs that the controller can send (TX-PDO 6). The PDO was previously configured via 1805h Transmit PDO 6 Communication Parameter. See chapter Process Data Object (PDO).

### Object description

| | |
|---|---|
| Index | $1A05_h$ |
| Object name | Transmit PDO 6 Mapping Parameter |
| Object Code | RECORD |
| Data type | PDO_MAPPING |
| Savable | yes, category: communication |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |

| | |
|---|---|
| Preset value | |
| Firmware version | FIR-v1614 |
| Change history | |

## Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | 1st Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | 2nd Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | 3rd Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | 4th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |

| | |
|---|---|
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $05_h$ |
| Name | 5th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $06_h$ |
| Name | 6th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

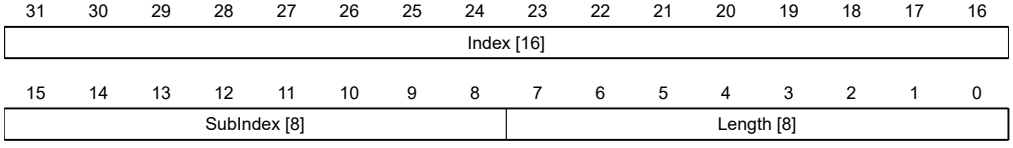
| | |
|---|---|
| Subindex | $07_h$ |
| Name | 7th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $08_h$ |
| Name | 8th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

## Description

Each subindex (1–8) 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.

## 1A06h Transmit PDO 7 Mapping Parameter

### Function

This object contains the mapping parameters for PDOs that the controller can send (TX-PDO 7). The PDO was previously configured via 1806h Transmit PDO 7 Communication Parameter. See chapter Process Data Object (PDO).

### Object description

| | |
|---|---|
| Index | $1A06_h$ |
| Object name | Transmit PDO 7 Mapping Parameter |
| Object Code | RECORD |
| Data type | PDO_MAPPING |
| Savable | yes, category: communication |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1614 |
| Change history | |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | 1st Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | 2nd Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | 3rd Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | 4th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $05_h$ |
| Name | 5th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $06_h$ |
| Name | 6th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $07_h$ |
| Name | 7th Object To Be Mapped |
| Data type | UNSIGNED32 |

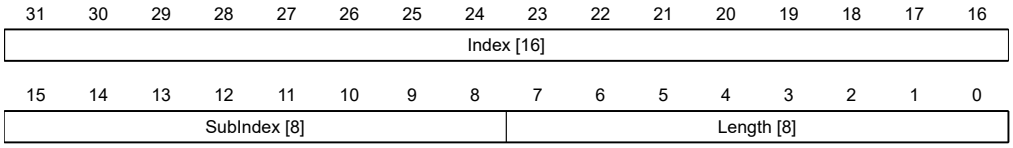
| | |
|---|---|
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $08_h$ |
| Name | 8th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

### Description

Each subindex (1–8) 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.

## 1A07h Transmit PDO 8 Mapping Parameter

### Function

This object contains the mapping parameters for PDOs that the controller can send (TX-PDO 8). The PDO was previously configured via 1807h Transmit PDO 8 Communication Parameter. See chapter Process Data Object (PDO).

### Object description

| | |
|---|---|
| Index | $1A07_h$ |
| Object name | Transmit PDO 8 Mapping Parameter |
| Object Code | RECORD |
| Data type | PDO_MAPPING |
| Savable | yes, category: communication |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |

| | |
|---|---|
| Preset value | |
| Firmware version | FIR-v1614 |
| Change history | |

## Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | 1st Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | 2nd Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | 3rd Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | 4th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |

| | |
|---|---|
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $05_h$ |
| Name | 5th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $06_h$ |
| Name | 6th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

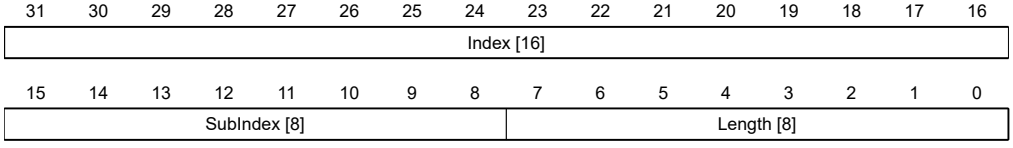
| | |
|---|---|
| Subindex | $07_h$ |
| Name | 7th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $08_h$ |
| Name | 8th Object To Be Mapped |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

## Description

Each subindex (1–8) 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.

## 1C00h Sync Manager Communication Type

### Function

This object indicates the assignment of the four EtherCAT SyncManagers.

### Object description

| | |
|---|---|
| Index | $1C00_h$ |
| Object name | Sync Manager Communication Type |
| Object Code | ARRAY |
| Data type | UNSIGNED8 |
| Savable | no |
| Firmware version | FIR-v1426 |
| Change history | The number of entries was changed from 2 to 5<br>Firmware version FIR-v2115-B1016293: "Name" entry changed from "Sync Manager Communication Type" to "Sync-Manager Type Channel 1: Mailbox Write".<br>Firmware version FIR-v2115-B1016293: "Name" entry changed from "Sync Manager Communication Type" to "Sync-Manager Type Channel 2: Mailbox Read".<br>Firmware version FIR-v2115-B1016293: "Name" entry changed from "Sync Manager Communication Type" to "Sync-Manager Type Channel 3: Process Data Write (Outputs)".<br>Firmware version FIR-v2115-B1016293: "Name" entry changed from "Sync Manager Communication Type" to "Sync-Manager Type Channel 4: Process Data Read (Inputs)". |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $04_h$ |

| | |
|---|---|
| Subindex | $01_h$ |

| | |
|---|---|
| Name | Sync-Manager Type Channel 1: Mailbox Write |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $01_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Sync-Manager Type Channel 2: Mailbox Read |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $02_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | Sync-Manager Type Channel 3: Process Data Write (Outputs) |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $03_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | Sync-Manager Type Channel 4: Process Data Read (Inputs) |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $04_h$ |

## Description

The assignment of the SyncManagers is defined by the manufacturer and cannot be changed.

| Subindex / Syncmanager | Function |
|---|---|
| 1 | Receive mailbox messages |
| 2 | Send mailbox messages |
| 3 | Receive cyclical process data |
| 4 | Send cyclical process data |

## 1F50h Program Data

### Function

This object is used to program memory areas of the controller. Each entry stands for a certain memory area.

### Object description

| | |
|---|---|
| Index | $1F50_h$ |
| Object name | Program Data |
| Object Code | ARRAY |
| Data type | DOMAIN |
| Savable | no |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1540 |
| Change history | |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $02_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Program Data Bootloader/firmware |
| Data type | DOMAIN |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | 0 |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Program Data NanoJ |
| Data type | DOMAIN |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | 0 |

## 1F51h Program Control

### Function

This object is used to control the programming of memory areas of the controller. Each entry stands for a certain memory area.

### Object description

| | |
|---|---|
| Index | $1F51_h$ |
| Object name | Program Control |
| Object Code | ARRAY |
| Data type | UNSIGNED8 |
| Savable | no |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1540 |
| Change history | |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $02_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Program Control Bootloader/firmware |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Program Control NanoJ |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00_h$ |

## 1F57h Program Status

### Function

This object indicates the programming status during the programming of memory areas of the controller. Each entry stands for a certain memory area.

### Object description

| | |
|---|---|
| Index | $1F57_h$ |
| Object name | Program Status |
| Object Code | ARRAY |
| Data type | UNSIGNED32 |
| Savable | no |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1540 |
| Change history | |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $02_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Program Status Bootloader/firmware |
| Data type | UNSIGNED32 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Program Status NanoJ |
| Data type | UNSIGNED32 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

## 1F80h NMT Startup

### Function

In this object you can set whether, after starting the controller, the state is automatically switched to the NMT state *Operational*. See also chapter Network Management (NMT).

### Object description

| | |
|---|---|
| Index | $1F80_h$ |
| Object name | NMT Startup |
| Object Code | VARIABLE |
| Data type | UNSIGNED32 |
| Savable | yes, category: communication |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |
| Firmware version | FIR-v1748-B531667 |
| Change history | |

### Description

- Value "$0"_h$: The state of the NMT state machine after initialization is *Pre-Operational*.
- Value "$8"_h$ (bit 3): The state of the NMT state machine after initialization is *Operational*.

## 2005h CANopen Baudrate

### Function

This object contains the baud rate of the CANopen bus.

### Object description

| | |
|---|---|
| Index | $2005_h$ |
| Object name | CANopen Baudrate |
| Object Code | VARIABLE |
| Data type | UNSIGNED8 |
| Savable | yes, category: CANopen |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $88_h$ |
| Firmware version | FIR-v1426 |
| Change history | Firmware version FIR-v1748-B531667: "Savable" entry changed from "yes, category: communication" to "yes, category: CANopen". |

### Description

The baud rates are to be set according to the following table. Each value outside of this table is interpreted as 1000 kBd.

| Value | | Baud rate |
|---|---|---|
| dec | hex | in kBd |
| 129 | 81 | 10 |
| 130 | 82 | 20 |
| 131 | 83 | 50 |
| 132 | 84 | 125 |
| 133 | 85 | 250 |
| 134 | 86 | 500 |
| 136 | 88 | 1000 |

## 2007h CANopen Config

### Function

This object can be used to perform various settings for CANopen.

### Object description

| | |
|---|---|
| Index | $2007_h$ |
| Object name | CANopen Config |
| Object Code | ARRAY |
| Data type | UNSIGNED32 |
| Savable | yes, category: CANopen |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1540 |
| Change history | Firmware version FIR-v1748-B531667: "Savable" entry changed from "yes, category: communication" to "yes, category: CANopen".<br>Firmware version FIR-v2412: subindex $02_h$ added. |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $02_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | BL Config |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Automatic COB-ID |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000001_h$ |

### Description

The subindices have the following functions:

- Subindex 01: reserved
- Subindex 02: If the value is "1" (factory setting), the COB-ID for PDOs 1 to 4 is automatically entered by the firmware. To manually set and save the COB-IDs, set the value to "0".

## 2009h CANopen NodeID

### Function

This object contains the node-ID of the controller. See chapter *Commissioning*.

### Object description

| | |
|---|---|
| Index | $2009_h$ |
| Object name | CANopen NodeID |
| Object Code | VARIABLE |
| Data type | UNSIGNED8 |
| Savable | yes, category: CANopen |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $7F_h$ |
| Firmware version | FIR-v1426 |
| Change history | Firmware version FIR-v1748-B531667: "Savable" entry changed from "yes, category: communication" to "yes, category: CANopen". |

## 2030h Pole Pair Count

### Function

Contains the number of pole pairs of the connected motor.

### Object description

| | |
|---|---|
| Index | $2030_h$ |
| Object name | Pole Pair Count |
| Object Code | VARIABLE |
| Data type | UNSIGNED16 |
| Savable | yes, category: motor |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $0032_h$ |
| Firmware version | FIR-v1426 |
| Change history | Firmware version FIR-v1540: "Savable" entry changed from "no" to "yes, category: tuning".<br>Firmware version FIR-v2315-B1040535: "Data type" entry changed from "UNSIGNED32" to "UNSIGNED16". |

## 2034h Upper Voltage Error Level [mV]

### Function

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

### Object description

| | |
|---|---|
| Index | $2034_h$ |
| Object name | Upper Voltage Error Level [mV] |
| Object Code | VARIABLE |
| Data type | UNSIGNED32 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $0000EA60_h$ |
| Firmware version | FIR-v1426 |
| Change history | Firmware version FIR-v2542-B1080909: entry "Object Name" changed from "Upper Voltage Warning Level" to "Upper Voltage Error Level [mV]". |

### Description

If the input voltage of the controller exceeds this threshold value, the motor is switched off and an error triggered. As soon as the input voltage is again smaller than the voltage of object $2034_h$ minus 2 volt, you can reset the error via the controlword.

## 2035h Lower Voltage Error Level [mV]

### Function

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

### Object description

| | |
|---|---|
| Index | $2035_h$ |
| Object name | Lower Voltage Error Level [mV] |
| Object Code | VARIABLE |
| Data type | UNSIGNED32 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00002710_h$ |
| Firmware version | FIR-v1426 |
| Change history | Firmware version FIR-v2542-B1080909: entry "Object Name" changed from "Lower Voltage Warning Level" to "Lower Voltage Error Level [mV]". |

### Description

If the input voltage of the controller falls below this threshold value, the motor is switched off and an error triggered. As soon as the input voltage is again larger than the voltage of object $2035_h$ plus 1.5 volt, you can reset the error via the controlword.

## 2038h Brake Controller Timing

### Function

This object contains the times for the *brake control* in milliseconds as well as the PWM frequency and the duty cycle.

### Object description

| | |
|---|---|
| Index | $2038_h$ |
| Object name | Brake Controller Timing |
| Object Code | ARRAY |
| Data type | UNSIGNED32 |
| Savable | yes, category: application |
| Firmware version | FIR-v1426 |
| Change history | |

## Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $06_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Close Brake Idle Time |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $000003E8_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Shutdown Power Idle Time |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $000003E8_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | Open Brake Delay Time |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $000003E8_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | Start Operation Delay Time |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $05_h$ |

| | |
|---|---|
| Name | PWM Frequency |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | between 50 and 20000 ($4E20_h$) |
| Preset value | $00004E20_h$ |

| | |
|---|---|
| Subindex | $06_h$ |
| Name | PWM Duty Cycle |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | Between 0 and 100 ($64_h$) |
| Preset value | $00000032_h$ |

### Description

The subindices have the following functions:

- $01_h$: Time between motor standstill and the closing of the brake.
- $02_h$: Time between the closing of the brake and the switching off of the motor current.
- $03_h$: Time between the switching on of the motor current and opening of the brake.
- $04_h$: Time between the opening of the brake and when the *Operation enabled* state of the CiA 402 Power State Machine is reached.
- $05_h$: Frequency of the PWM signal in hertz.
- $06_h$: Duty cycle of the PWM signal in percent. Value "0" switches off the PWM signal.

## 2039h Motor Currents

### Function

This object contains the measured motor currents in mA. All values are peak values, (#2*rms).

### Object description

| | |
|---|---|
| Index | $2039_h$ |
| Object name | Motor Currents |
| Object Code | ARRAY |
| Data type | INTEGER32 |
| Savable | no |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1426 |
| Change history | Firmware version FIR-v1504: "PDO mapping" table entry for subindex 01 changed from "no" to "TX-PDO".<br>Firmware version FIR-v1504: "PDO mapping" table entry for subindex 02 changed from "no" to "TX-PDO". |

| | |
|---|---|
| | Firmware version FIR-v1504: "PDO mapping" table entry for subindex 03 changed from "no" to "TX-PDO".<br>Firmware version FIR-v1504: "PDO mapping" table entry for subindex 04 changed from "no" to "TX-PDO".<br>Firmware version FIR-v2213: subindex $05_h$, "Actual Current" added. Phase currents Ia and Ib changed to Iα and Iβ (Clarke transformation). |

## Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $05_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Id |
| Data type | INTEGER32 |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Iq |
| Data type | INTEGER32 |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | Iα |
| Data type | INTEGER32 |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | Iβ |

| | |
|---|---|
| Data type | INTEGER32 |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $05_h$ |
| Name | Actual Current |
| Data type | INTEGER32 |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |

### Description

- $01_h$: Field-forming components of the current
- $02_h$: Torque-forming components of the current
- $03_h$: Iα
- $04_h$: Iβ
- $05_h$: total current calculated down to a motor phase. In *closed-loop*, the sign of Iq is also used. The current value can then be placed on a scale to compare with the current from $6075_h$.
  *Open-Loop*:
  - Stepper motor: $I = \sqrt{(I\alpha^2 + I\beta^2)} / \sqrt{2}$
  - BLDC motor: $I = \sqrt{(I\alpha^2 + I\beta^2)} / (2/\sqrt{3})$

  *Closed-Loop*:
  - Stepper motor: $I = sgn(Iq) * \sqrt{(I\alpha^2 + I\beta^2)} / \sqrt{2}$
  - BLDC motor: $I = sgn(Iq) * \sqrt{(I\alpha^2 + I\beta^2)} / (2/\sqrt{3})$

## 203Dh Torque Window

### Function

Specifies a symmetrical range relative to the target torque within which the target is considered having been met.

If the value is set to "FFFFFFFF"$_h$, monitoring is switched off, the "Target reached" bit in object $6041_h$ (statusword) is never set.

### Object description

| | |
|---|---|
| Index | $203D_h$ |
| Object name | Torque Window |
| Object Code | VARIABLE |
| Data type | UNSIGNED16 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $0000_h$ |

| | |
|---|---|
| Firmware version | FIR-v1540 |
| Change history | Firmware version FIR-v1614: "Savable" entry changed from "no" to "yes, category: application". |

## 203Eh Torque Window Time Out

### Function

The current torque must be within the "Torque Window" (203D$_h$) for this time (in milliseconds) for the target torque to be considered having been met.

### Object description

| | |
|---|---|
| Index | 203E$_h$ |
| Object name | Torque Window Time Out |
| Object Code | VARIABLE |
| Data type | UNSIGNED16 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | 0000$_h$ |
| Firmware version | FIR-v1540 |
| Change history | Firmware version FIR-v1614: "Savable" entry changed from "no" to "yes, category: application".<br>Firmware version FIR-v1738-B501312: "Object Name" entry changed from "Torque Window Time" to "Torque Window Time Out". |

## 203Fh Max Slippage Time Out

### Function

Time in milliseconds until an excessively large slippage error in Profile Velocity mode results in an error message.

### Object description

| | |
|---|---|
| Index | 203F$_h$ |
| Object name | Max Slippage Time Out |
| Object Code | VARIABLE |
| Data type | UNSIGNED16 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | 0064$_h$ |
| Firmware version | FIR-v1738-B501312 |
| Change history | |

### Description

If the actual speed deviates so much from the set speed that the value (absolute value) of the object 60F8$_h$ (Max Slippage) is exceeded, bit 13 in object 6041$_h$ is set. The deviation must last longer than the time in object 203F$_h$.

A reaction to the slippage error can be set in object 3700$_h$. If a reaction is defined, an error is also entered in object 1003$_h$.

## 2057h Clock Direction Multiplier

### Function

The clock count value in Clock-direction mode is multiplied by this value before it is processed further.

### Object description

| | |
|---|---|
| Index | 2057$_h$ |
| Object name | Clock Direction Multiplier |
| Object Code | VARIABLE |
| Data type | UNSIGNED32 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | 00000080$_h$ |
| Firmware version | FIR-v1426 |
| Change history | Firmware version FIR-v2339-B1048823: "Data type" entry changed from "INTEGER32" to "UNSIGNED32". |

## 2058h Clock Direction Divider

### Function

The clock count value in Clock-direction mode is divided by this value before it is processed further.

### Object description

| | |
|---|---|
| Index | 2058$_h$ |
| Object name | Clock Direction Divider |
| Object Code | VARIABLE |
| Data type | UNSIGNED32 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | 00000001$_h$ |
| Firmware version | FIR-v1426 |
| Change history | Firmware version FIR-v2339-B1048823: "Data type" entry changed from "INTEGER32" to "UNSIGNED32". |

## 205Bh Clock/Direction Or Clockwise/Counter-Clockwise Mode

### Function

This object can be used to switch the clock-direction mode (value = "0") to the right/left rotation mode (value = "1").

### Object description

| | |
|---|---|
| Index | $205B_h$ |
| Object name | Clock/Direction Or Clockwise/Counter-Clockwise Mode |
| Object Code | VARIABLE |
| Data type | UNSIGNED8 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00_h$ |
| Firmware version | FIR-v1504 |
| Change history | Firmware version FIR-v2542-B1079618: entry "Object Name" changed from "Clock Direction Or Clockwise/Counter-Clockwise Mode" to "Clock/Direction Or Clockwise/Counter-Clockwise Mode".<br>Firmware version FIR-v2542-B1079618: entry "Data type" changed from "UNSIGNED32" to "UNSIGNED8". |

## 205Ch Pulse Generator Configuration

### Function

Use this object to configure the virtual encoder output. You can find details in chapter Digital inputs and outputs, pulse generator.

### Object description

| | |
|---|---|
| Index | $205C_h$ |
| Object name | Pulse Generator Configuration |
| Object Code | ARRAY |
| Data type | INTEGER16 |
| Savable | yes, category: application |
| Access | read only |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v2115-B1016293 |
| Change history | Firmware version FIR-v2115-B1016293: "Name" entry changed from "Number Of Increments To Be Sent When No Sensor In 0x3203 Is Choosen" to "Number Of Increments To Be Sent When No Sensor In 0x3203 Is Choosen". |

| | |
|---|---|
| | Firmware version FIR-v2115-B1016293: "Name" entry changed from "Output Signal Mode: 0=AB, 1=ClkDir, 2=CW/CCW" to "Output Signal Mode: 0=AB, 1=ClkDir, 2=CW/CCW".<br>Firmware version FIR-v2115-B1016293: "Name" entry changed from "Numerator For Conversion Of Sensor Increments To Virtual Encoder Increments" to "Numerator For Conversion Of Sensor Increments To Virtual Encoder Increments".<br>Firmware version FIR-v2115-B1016293: "Name" entry changed from "Denominator For Conversion Of Sensor Increments To Virtual Encoder Increments" to "Denominator For Conversion Of Sensor Increments To Virtual Encoder Increments".<br>Firmware version FIR-v2412-B1057638: "Object Name" entry changed from "Virtual Encoder Configuration" to "Pulse Generator Configuration".<br>Firmware version FIR-v2412-B1057638: "Name" entry changed from "Numerator For Conversion Of Sensor Increments To Virtual Encoder Increments" to "Numerator For Conversion Of Sensor Increments To Pulse Generator Increments".<br>Firmware version FIR-v2412-B1057638: "Name" entry changed from "Denominator For Conversion Of Sensor Increments To Virtual Encoder Increments" to "Denominator For Conversion Of Sensor Increments To Pulse Generator Increments".<br>Firmware version FIR-v2542-B1079618: entry "Name" changed from "Number Of Increments To Be Sent When No Sensor In 0x3203 Is Choosen" to "Number Of Increments To Be Sent When No Sensor In 0x3203 Is Chosen". |

## Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $04_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Number Of Increments To Be Sent When No Sensor In 0x3203 Is Chosen |
| Data type | INTEGER16 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $0000_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Output Signal Mode: 0=AB, 1=ClkDir, 2=CW/CCW |

| | |
|---|---|
| Data type | INTEGER16 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $0000_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | Numerator For Conversion Of Sensor Increments To Pulse Generator Increments |
| Data type | INTEGER16 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $0000_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | Denominator For Conversion Of Sensor Increments To Pulse Generator Increments |
| Data type | INTEGER16 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $0001_h$ |

## 2101h Fieldbus Module Availability

### Function

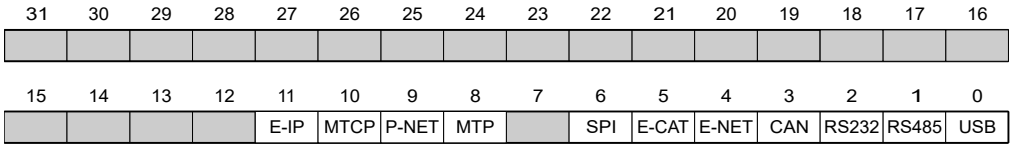
Displays the available interfaces/fieldbuses or protocols.

### Object description

| | |
|---|---|
| Index | $2101_h$ |
| Object name | Fieldbus Module Availability |
| Object Code | VARIABLE |
| Data type | UNSIGNED32 |
| Savable | no |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | ■ $00000000_h$<br>■ N6-1-2-1-S: $00000109_h$<br>■ N6-1-2-1: $00000109_h$<br>■ N6-2-2-1-S: $00000109_h$<br>■ N6-2-2-1: $00000109_h$ |
| Firmware version | FIR-v1426 |

| | |
|---|---|
| Change history | Firmware version FIR-v1626: "Object Name" entry changed from "Fieldbus Module" to "Fieldbus Module Availability".<br>Firmware version FIR-v1529: new bit assignment |

## Description




**USB**

Value = "1": The USB fieldbus (Modbus RTU) is available.

**RS-485**

Value = "1": An RS-485 interface (Modbus RTU) is available.

**RS-232**

Value = "1": An RS-232 interface (Modbus RTU) is available.

**CAN**

Value = "1": The CANopen fieldbus is available.

**E-NET**

Value = "1": An Ethernet interface (REST protocol) is available.

**E-CAT**

Value = "1": An EtherCAT interface is available.

**SPI**

Value = "1": An SPI interface (NanoSPI) is available.

**MTP**

Value = "1": A USB interface (MTP) is available.

**P-NET**

Value = "1": The PROFINET (PROFIdrive) fieldbus is available.

**MTCP**

Value = "1": The used protocol is Modbus TCP

**E-IP**

Value = "1": The used protocol is EtherNet/IP™

# 2102h Fieldbus Module Control

## Function

This object can be used to activate/deactivate certain fieldbuses (physical interfaces and protocols).

## Object description

| | |
|---|---|
| Index | $2102_h$ |
| Object name | Fieldbus Module Control |

| | |
|---|---|
| Object Code | VARIABLE |
| Data type | UNSIGNED32 |
| Savable | yes, category: communication |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | ■ $00000000_h$<br>■ N6-1-2-1-S: $00000109_h$<br>■ N6-1-2-1: $00000109_h$<br>■ N6-2-2-1-S: $00000109_h$<br>■ N6-2-2-1: $00000109_h$ |
| Firmware version | FIR-v1540 |
| Change history | Firmware version FIR-v1626: "Savable" entry changed from "yes, category: application" to "yes, category: communication". |

## Description

Object $\underline{2103_h}:1_h$ contains all physical interfaces/protocols that can be activated/deactivated. These can be switched in this object ($2102_h$). The current status of the activated fieldbuses is in object $\underline{2103_h}:2_h$.

The following distribution of the bits applies here:

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

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | E-IP | MTCP | P-NET | MTP | | SPI | E-CAT | E-NET | CAN | RS232 | RS485 | USB |

**USB**

Value = "1": The USB fieldbus (Modbus RTU) is available.

**RS-485**

Value = "1": An RS-485 interface (Modbus RTU) is available.

**RS-232**

Value = "1": An RS-232 interface (Modbus RTU) is available.

**CAN**

Value = "1": The CANopen fieldbus is available.

**E-NET**

Value = "1": An Ethernet interface (REST protocol) is available.

**E-CAT**

Value = "1": An EtherCAT interface is available.

**SPI**

Value = "1": An SPI interface (NanoSPI) is available.

**MTP**

Value = "1": A USB interface (MTP) is available.

**P-NET**

Value = "1": The PROFINET (PROFIdrive) fieldbus is available.

**MTCP**

Value = "1": The used protocol is Modbus TCP

**E-IP**

Value = "1": The used protocol is EtherNet/IP™

## 2103h Fieldbus Module Status

### Function

Shows the active fieldbuses.

### Object description

| | |
|---|---|
| Index | $2103_h$ |
| Object name | Fieldbus Module Status |
| Object Code | ARRAY |
| Data type | UNSIGNED32 |
| Savable | no |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1540 |
| Change history | |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $02_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Fieldbus Module Disable Mask |
| Data type | UNSIGNED32 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | ■ $00000000_h$<br>■ N6-1-1-1-S: $00000121_h$<br>■ N6-1-1-1: $00000121_h$ |

- N6-2-1-1-S: $00000121_h$
- N6-2-1-1: $00000121_h$
- N6-1-2-1-S: $00000109_h$
- N6-1-2-1: $00000109_h$
- N6-2-2-1-S: $00000109_h$
- N6-2-2-1: $00000109_h$
- N6-1-3-1-S: $00000911_h$
- N6-1-3-1: $00000911_h$
- N6-1-3-2-S: $00000911_h$
- N6-1-3-2: $00000911_h$
- N6-2-3-1-S: $00000911_h$
- N6-2-3-1: $00000911_h$
- N6-2-3-2-S: $00000911_h$
- N6-2-3-2: $00000911_h$
- N6-1-4-1-S: $00000511_h$
- N6-1-4-1: $00000511_h$
- N6-2-4-1-S: $00000511_h$
- N6-2-4-1: $00000511_h$
- N6-1-5-1-S: $00000103_h$
- N6-1-5-1: $00000103_h$
- N6-2-5-1-S: $00000103_h$
- N6-2-5-1: $00000103_h$
- N6-1-7-1: $00000101_h$

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Fieldbus Module Enabled |
| Data type | UNSIGNED32 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

## Description

Subindex 1 (Fieldbus Module Disable Mask): This subindex contains all physical interfaces and protocols that can be activated or deactivated. The set bit means that this fieldbus can be deactivated.

Subindex 2 (Fieldbus Module Enabled): This subindex contains all currently activated physical interfaces and protocols. The set bit means 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | E-IP | MTCP | P-NET | MTP | | SPI | E-CAT | E-NET | CAN | RS232 | RS485 | USB |

**USB**

Value = "1": The USB fieldbus (Modbus RTU) is available.

**RS-485**
Value = "1": An RS-485 interface (Modbus RTU) is available.

**RS-232**
Value = "1": An RS-232 interface (Modbus RTU) is available.

**CAN**
Value = "1": The CANopen fieldbus is available.

**E-NET**
Value = "1": An Ethernet interface (REST protocol) is available.

**E-CAT**
Value = "1": An EtherCAT interface is available.

**SPI**
Value = "1": An SPI interface (NanoSPI) is available.

**MTP**
Value = "1": A USB interface (MTP) is available.

**P-NET**
Value = "1": The PROFINET (PROFIdrive) fieldbus is available.

**MTCP**
Value = "1": The used protocol is Modbus TCP

**E-IP**
Value = "1": The used protocol is EtherNet/IP™

## 2290h PDI Control

### Function

With this object, you can activate the *Plug&Drive interface*. You can find additional information in document *Function description Plug&Drive interface*.

### Object description

| | |
|---|---|
| Index | $2290_h$ |
| Object name | PDI Control |
| Object Code | VARIABLE |
| Data type | UNSIGNED8 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $01_h$ |
| Firmware version | FIR-v1748-B531667 |
| Change history | Firmware version FIR-v1748-B538662: "Access" table entry for subindex 00 changed from "read only" to "read/write". |

### Description

To activate the *Plug&Drive interface*, set bit 0 to "1".

## 2291h PDI Input

### Function

If you use the *Plug&Drive interface*, you can use this object to select and start the operating mode and set the corresponding target values (target position, speed, etc.). You can find additional information in document *Function description Plug&Drive interface*.

### Object description

| | |
|---|---|
| Index | $2291_h$ |
| Object name | PDI Input |
| Object Code | RECORD |
| Data type | PDI_INPUT |
| Savable | no |
| Access | read only |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1748-B531667 |
| Change history | Firmware version FIR-v2013-B726332: "Savable" entry changed from "yes, category: application" to "no".<br>Firmware version FIR-v2315-B1040535: "Data type" entry changed from "INTEGER8" to "UNSIGNED8". |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $04_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | PDI Set Value 1 |
| Data type | INTEGER32 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | PDI Set Value 2 |
| Data type | INTEGER16 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $0000_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | PDI Set Value 3 |
| Data type | INTEGER8 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | PDI Command |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00_h$ |

## 2292h PDI Output

### Function

If you use the *Plug&Drive interface*, you can, in this object, read the status and a return value that is dependent on the used operating mode. You can find additional information in document *Function description Plug&Drive interface*.

### Object description

| | |
|---|---|
| Index | $2292_h$ |
| Object name | PDI Output |
| Object Code | RECORD |
| Data type | PDI_OUTPUT |
| Savable | no |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1748-B531667 |
| Change history | Firmware version FIR-v2315-B1040535: "Data type" entry changed from "INTEGER16" to "UNSIGNED16". |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $02_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | PDI Status |
| Data type | UNSIGNED16 |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $0000_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | PDI Return Value |
| Data type | INTEGER32 |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |

## 2300h NanoJ Control

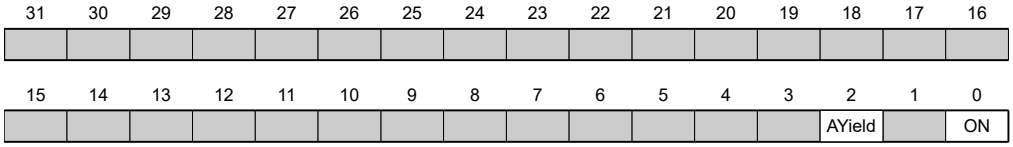
### Function

Controls the execution of a *NanoJ program*.

### Object description

| | |
|---|---|
| Index | $2300_h$ |
| Object name | NanoJ Control |
| Object Code | VARIABLE |
| Data type | UNSIGNED32 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |
| Firmware version | FIR-v1426 |
| Change history | Firmware version FIR-v1436: "Object Name" entry changed from "VMM Control" to "NanoJ Control". |

## Description




**ON**

Switches the *NanoJ program* on (value = "1") or off (value = "0").

With a rising edge in bit 0, the program is first reloaded and the variable range reset.

**NOTICE**

Startup of the *NanoJ program* can take up to 200 ms.

When switching on, a check is performed to determine whether a *NanoJ program* is present. If present, "1" is entered in 2300 and the *NanoJ program* is started.

**AYield (AutoYield)**

If this feature is activated (bit set to "1"), the *NanoJ program* is no longer stopped if it runs longer than it is allowed to. The *NanoJ program* is, thus, no longer real-time capable and no longer runs every 1 ms (see Available computing time).

# 2301h NanoJ Status

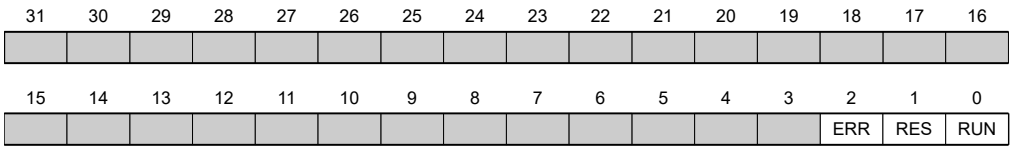
## Function

Indicates the operating state of the user program.

## Object description

| | |
|---|---|
| Index | $2301_h$ |
| Object name | NanoJ Status |
| Object Code | VARIABLE |
| Data type | UNSIGNED32 |
| Savable | no |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |
| Firmware version | FIR-v1426 |
| Change history | Firmware version FIR-v1436: "Object Name" entry changed from "VMM Status" to "NanoJ Status". |

### Description




**RUN**

Value = "0": Program is stopped, value = "1": NanoJ program is running.

**RES**

Reserved.

**ERR**

Program was ended with an error. Cause of the error can be read from object 2302h.

## 2302h NanoJ Error Code

### Function

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

### Object description

| | |
|---|---|
| Index | $2302_h$ |
| Object name | NanoJ Error Code |
| Object Code | VARIABLE |
| Data type | UNSIGNED32 |
| Savable | no |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |
| Firmware version | FIR-v1426 |
| Change history | Firmware version FIR-v1436: "Object Name" entry changed from "VMM Error Code" to "NanoJ Error Code". |

### Description

Error codes during program execution:

| Number | Description |
|---|---|
| $1xxxxyy_h$ | Invalid mapping in the NanoJ program file: The value in "xxxx" specifies the index, the value in "yy" specifies the subindex of the object that should – but cannot – be mapped. |
| $2000000_h$ | Invalid mapping in the NanoJ program file: too many variables of type `input` were declared (see 2310h NanoJ Input Data Selection) |
| $3000000_h$ | Invalid mapping in the NanoJ program file: too many variables of type `output` were declared (see 2320h NanoJ Output Data Selection) |

| Number | Description |
|---|---|
| $4000000_h$ | Invalid mapping in the NanoJ program file: too many variables of type `inout` were declared (see 2330h NanoJ In/output Data Selection) |
| $0002_h$ | Error in memory management |
| $0003_h$ | Usage fault |
| $0004_h$ | Hatd fault |
| $0005_h$ | Time Out: Code executed too long without yield() or sleep() |
| $0006_h$ | Bus fault |
| $1000_h$ | Access of a nonexistent object in the object dictionary |
| $1002_h$ | An attempt was made to write a value that is too low or too high to an object. |
| $1003_h$ | An attempt was made to read out an object that permits only write access. |
| $1FFF_h$ | Unauthorized access of an object |

Error when accessing an object:

| Number | Description |
|---|---|
| $1xxxxyy_h$ | Invalid mapping in the NanoJ program file: The value in "xxxx" specifies the index, the value in "yy" specifies the subindex of the object that should – but cannot – be mapped. |
| $2000000_h$ + number of excess variables | Invalid mapping in the NanoJ program file: too many variables of type `input` were declared (see 2310h NanoJ Input Data Selection) |
| $3000000_h$ + number of excess variables | Invalid mapping in the NanoJ program file: too many variables of type `output` were declared (see 2320h NanoJ Output Data Selection) |
| $4000000_h$ + number of excess variables | Invalid mapping in the NanoJ program file: too many variables of type `inout` were declared (see 2330h NanoJ In/output Data Selection) |
| $1000_h$ | Access of a nonexistent object in the object dictionary |
| $1001_h$ | Write access of a write-protected entry in the OD |
| $1002_h$ | An attempt was made to write a value that is too low or too high to an object. |
| $1003_h$ | An attempt was made to read out an object that permits only write access. |
| $1FFF_h$ | Unauthorized access of an object |

## 2303h NanoJ Metadata

### Function

This object is used to correctly recognize the NanoJ program by external software, e.g., *Plug & Drive Studio*.

### Object description

| | |
|---|---|
| Index | $2303_h$ |
| Object name | NanoJ Metadata |
| Object Code | ARRAY |
| Data type | UNSIGNED32 |
| Savable | no |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |

| | |
|---|---|
| Firmware version | FIR-v2339-B1048823 |
| Change history | Firmware version FIR-v2542-B1079618: entry "Object Name" changed from "NanoJ metadata" to "NanoJ Metadata". |

## Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $06_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Version |
| Data type | UNSIGNED32 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000300_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | FLASH Start Address |
| Data type | UNSIGNED32 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | FLASH End Address |
| Data type | UNSIGNED32 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | RAM Start Address |
| Data type | UNSIGNED32 |
| Access | read only |
| PDO mapping | no |

| | |
|---|---|
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $05_h$ |
| Name | RAM End Address |
| Data type | UNSIGNED32 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $06_h$ |
| Name | Header Size |
| Data type | UNSIGNED32 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

## 230Eh Timer

### Function

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

**NOTICE**

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

### Object description

| | |
|---|---|
| Index | $230E_h$ |
| Object name | Timer |
| Object Code | ARRAY |
| Data type | UNSIGNED32 |
| Savable | no |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v2139-B1020888 |
| Change history | |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $01_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | 1ms Timer |
| Data type | UNSIGNED32 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

## 230Fh Uptime Seconds

### Function

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

**NOTICE**

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

### Object description

| | |
|---|---|
| Index | $230F_h$ |
| Object name | Uptime Seconds |
| Object Code | VARIABLE |
| Data type | UNSIGNED32 |
| Savable | no |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |
| Firmware version | FIR-v1436 |
| Change history | |

## 2310h NanoJ Input Data Selection

### Function

Describes the object dictionary entries that are copied to the PDO mapping input of the NanoJ program.

## Object description

| | |
|---|---|
| Index | $2310_h$ |
| Object name | NanoJ Input Data Selection |
| Object Code | ARRAY |
| Data type | UNSIGNED32 |
| Savable | no |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1650-B472161 |
| Change history | Firmware version FIR-v1436: "Object Name" entry changed from "VMM Input Data Selection" to "NanoJ Input Data Selection".<br>Firmware version FIR-v1650-B472161: "Savable" entry changed from "yes, category: application" to "no".<br>Firmware version FIR-v1650-B472161: "Access" table entry for subindex 00 changed from "read/write" to "read only".<br>Firmware version FIR-v1650-B472161: "Access" table entry for subindex 01 changed from "read/write" to "read only". |

## Value description

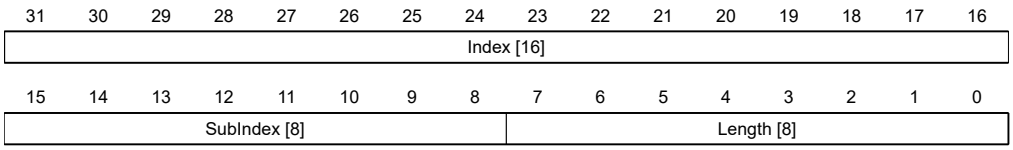
| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $10_h$ |

| | |
|---|---|
| Subindex | $01_h$ - $10_h$ |
| Name | Mapping #1 - #16 |
| Data type | UNSIGNED32 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

## Description

Each subindex (1–16) describes a different mapped object.

A mapping entry consists of four bytes, which are structured according to the following graphic.




**Index [16]**

This contains the index of the object to be mapped

**Subindex [8]**

This contains the subindex of the object to be mapped

**Length [8]**

This contains the length of the object to be mapped in units of bits.

## 2320h NanoJ Output Data Selection

### Function

Describes the object dictionary entries that are copied into the output PDO mapping of the *NanoJ program* after it is executed.

### Object description

| | |
|---|---|
| Index | $2320_h$ |
| Object name | NanoJ Output Data Selection |
| Object Code | ARRAY |
| Data type | UNSIGNED32 |
| Savable | no |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1650-B472161 |
| Change history | Firmware version FIR-v1436: "Object Name" entry changed from "VMM Output Data Selection" to "NanoJ Output Data Selection".<br>Firmware version FIR-v1650-B472161: "Savable" entry changed from "yes, category: application" to "no".<br>Firmware version FIR-v1650-B472161: "Access" table entry for subindex 00 changed from "read/write" to "read only".<br>Firmware version FIR-v1650-B472161: "Access" table entry for subindex 01 changed from "read/write" to "read only". |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |

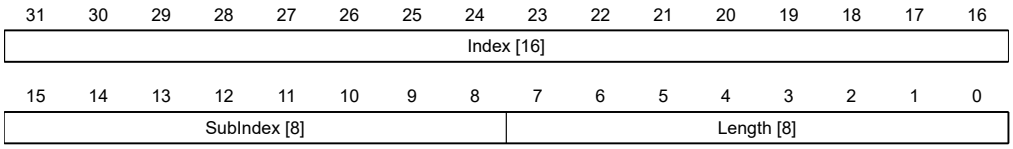
| | |
|---|---|
| Allowed values | |
| Preset value | $10_h$ |

| | |
|---|---|
| Subindex | $01_h$ - $10_h$ |
| Name | Mapping #1 - #16 |
| Data type | UNSIGNED32 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

### Description

Each subindex (1–16) describes a different mapped object.

A mapping entry consists of four bytes, which are structured according to the following graphic.




**Index [16]**

This contains the index of the object to be mapped

**Subindex [8]**

This contains the subindex of the object to be mapped

**Length [8]**

This contains the length of the object to be mapped in units of bits.

## 2330h NanoJ In/output Data Selection

### Function

Describes the object dictionary entries that are first copied to the input PDO mapping of the NanoJ program and, after it is executed, are copied back to the output PDO mapping.

### Object description

| | |
|---|---|
| Index | $2330_h$ |
| Object name | NanoJ In/output Data Selection |
| Object Code | ARRAY |
| Data type | UNSIGNED32 |
| Savable | no |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1650-B472161 |

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

## Value description

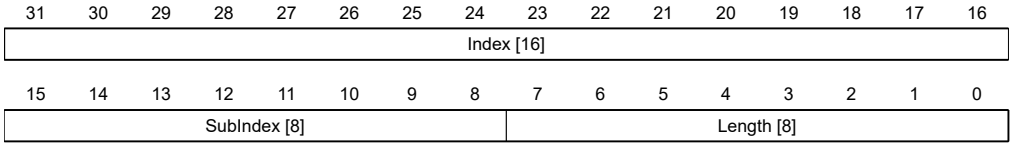
| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $10_h$ |

| | |
|---|---|
| Subindex | $01_h$ - $10_h$ |
| Name | Mapping #1 - #16 |
| Data type | UNSIGNED32 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

## Description

Each subindex (1–16) describes a different mapped object.

A mapping entry consists of four bytes, which are structured according to the following graphic.




**Index [16]**

This contains the index of the object to be mapped

**Subindex [8]**

This contains the subindex of the object to be mapped

**Length [8]**

This contains the length of the object to be mapped in units of bits.

## 2400h NanoJ Inputs

### Function

Located here is an array with 32, 32-bit integer values that is not used within the firmware and serves only for communicating with the user program via the fieldbus.

### Object description

| | |
|---|---|
| Index | $2400_h$ |
| Object name | NanoJ Inputs |
| Object Code | ARRAY |
| Data type | INTEGER32 |
| Savable | no |
| Firmware version | FIR-v1426 |
| Change history | The number of entries was changed from 2 to 33<br>Firmware version FIR-v1436: "Object Name" entry changed from "VMM Inputs" to "NanoJ Inputs".<br>Firmware version FIR-v1436: "Name" entry changed from "VMM Input N#" to "NanoJ Input N#". |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $20_h$ |

| | |
|---|---|
| Subindex | $01_h$ - $20_h$ |
| Name | NanoJ Input #1 - #32 |
| Data type | INTEGER32 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |

### Description

Here, it is possible to pass, e.g., preset values, to the *NanoJ program*.

## 2500h NanoJ Outputs

### Function

Located here is an array with 32, 32-bit integer values that is not used within the firmware and serves only for communicating with the user program via the fieldbus.

### Object description

| | |
|---|---|
| Index | $2500_h$ |
| Object name | NanoJ Outputs |
| Object Code | ARRAY |
| Data type | INTEGER32 |
| Savable | no |
| Firmware version | FIR-v1426 |
| Change history | Firmware version FIR-v1436: "Object Name" entry changed from "VMM Outputs" to "NanoJ Outputs".<br>Firmware version FIR-v1436: "Name" entry changed from "VMM Output N#" to "NanoJ Output N#". |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $20_h$ |

| | |
|---|---|
| Subindex | $01_h$ - $20_h$ |
| Name | NanoJ Output #1 - #32 |
| Data type | INTEGER32 |
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |

### Description

Here, the *NanoJ program* can store results which can then be read out via the fieldbus.

## 2701h Customer Storage Area

### Function

Data can be deposited and stored in this object.

### Object description

| | |
|---|---|
| Index | $2701_h$ |
| Object name | Customer Storage Area |
| Object Code | ARRAY |
| Data type | UNSIGNED32 |
| Savable | yes, category: customer |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1540 |
| Change history | Firmware version FIR-v1540: "Data Type" entry changed from "UNSIGNED32" to "UNSIGNED8". |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $FE_h$ |

| | |
|---|---|
| Subindex | $01_h$ - $FE_h$ |
| Name | Storage #1 - #254 |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

## 2800h Bootloader And Reboot Settings

### Function

With this object, a reboot of the firmware can be triggered.

### Object description

| | |
|---|---|
| Index | $2800_h$ |
| Object name | Bootloader And Reboot Settings |
| Object Code | ARRAY |
| Data type | UNSIGNED32 |
| Savable | yes, category: application |

| | |
|---|---|
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1540 |
| Change history | |

## Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $03_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Reboot Command |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Reboot Delay Time In Ms |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000064_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | Bootloader HW Config |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

## Description

The subindices have the following function:

- $01_h$: If the value "$746F6F62_h$" is entered here, the firmware is rebooted.
- $02_h$: Time in milliseconds: delays the reboot of the firmware by the respective time.
- $03_h$: Reserved

## 306Ch Velocity Actual Internal Value

### Function

Contains the current actual speed in rpm. Compared to 606Ch Velocity Actual Value, the value is more strongly filtered and is used internally by the velocity controller and for the voltage feed forward and dead time compensation.

### Object description

| | |
|---|---|
| Index | $306C_h$ |
| Object name | Velocity Actual Internal Value |
| Object Code | VARIABLE |
| Data type | INTEGER32 |
| Savable | no |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |
| Firmware version | FIR-v2412-B1057638 |
| Change history | |

## 30C2h Cut-off Frequency [mHz] To Filter The Velocity Demand Value Derived From Position Change

### Object description

| | |
|---|---|
| Index | $30C2_h$ |
| Object name | Cut-off Frequency [mHz] To Filter The Velocity Demand Value Derived From Position Change |
| Object Code | VARIABLE |
| Data type | UNSIGNED32 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $FFFFFFFF_h$ |
| Firmware version | FIR-v2542-B1079618 |
| Change history | |

## 3202h Motor Drive Submode Select

### Function

Controls the controller mode, such as the changeover between *closed-loop* / *open-loop* and whether Velocity Mode is simulated via the S-controller or functions with a real V-controller in *closed loop*.

### Object description

| | |
|---|---|
| Index | $3202_h$ |
| Object name | Motor Drive Submode Select |
| Object Code | VARIABLE |
| Data type | UNSIGNED32 |
| Savable | yes, category: motor |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |
| Firmware version | FIR-v1426 |
| Change history | Firmware version FIR-v1540: "Savable" entry changed from "yes category: application" to "yes, category: travel".<br>Firmware version FIR-v1540: "Savable" entry changed from "yes category: travel" to "yes, category: movement".<br>Firmware version FIR-v2451-B1068465: "Savable" entry changed from "yes, category: movement" to "yes, category: motor". |

### 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | BLDC | Torque | | | Brake | VoP | CL/OL |

**CL/OL**

Changeover between *open-loop* and *closed-loop* (see chapter Control modes)

- Value = "0": *open-loop*
- Value = "1": *closed loop*

Toggling is not possible in the *Operation enabled* state.

**VoP**

Value = "1": Simulate V-controller with a P-ramp: simulate the speed modes through continuous position changes

**Brake**

Value = "1": Switch on automatic brake control.

**Torque**

only active in operating modes Profile Torque and Cyclic Synchronous Torque

Value = "1": M-controller is active, otherwise a V-controller is superimposed: no V-controller is used in the torque modes for speed limiting, thus object 6080$_h$ is ignored; the 321Bh Velocity Controller Parameters have no effect on the control.

**BLDC**

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

## 3203h Feedback Selection

### Function

In this object, the sources of the presets are defined for the commutation and the velocity and position control.

A value change in the *Operation enabled* state shows no immediate effect. Value changes in objects are buffered and read out upon changing to the *Operation enabled* state.

### Object description

| | |
|---|---|
| Index | 3203$_h$ |
| Object name | Feedback Selection |
| Object Code | ARRAY |
| Data type | UNSIGNED8 |
| Savable | yes, category: sensors |
| Access | read only |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1748-B538662 |
| Change history | Firmware version FIR-v2451-B1068465: "Savable" entry changed from "yes, category: tuning" to "yes, category: sensors". |

### Value description

| | |
|---|---|
| Subindex | 00$_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | 04$_h$ |

| | |
|---|---|
| Subindex | 01$_h$ |
| Name | 1st Feedback Interface |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |

| | |
|---|---|
| Preset value | $00_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | 2nd Feedback Interface |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | 3rd Feedback Interface |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | 4th Feedback Interface |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00_h$ |

## Description

The subindices have the following function:

- $00_h$: Value="1" to "n", where "n" is the number of existing feedbacks.
- $n_h$:
  Subindex n contains a bit mask for the respective feedback n. The bits have the following meaning here:
- Bit 0: If the bit is set to "1", this sensor is used for position feedback.
- Bit 1: If the bit is set to "1", this sensor is used for velocity feedback.
- Bit 2: If the bit is set to "1", this sensor is used for commutation feedback in Closed Loop.

Subindex $01_h$ always corresponds to the first feedback *Sensorless* . 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 the second sensor and continues in ascending order until all existing sensors have been queried. If a sensor is found whose feedback is set, it is assigned to the corresponding controller and the search ended.

**NOTICE**

If bit 0 in $3202_h$ is set to "0", *closed loop* is deactivated; bit 2 (commutation) then has no meaning. Bit 1 for the velocity and bit 0 for the position in the respective subindices are still used for the display of the actual position and speed values.

## 3204h Feedback Mapping

### Function

This object contains information on the existing feedbacks.

### Object description

| | |
|---|---|
| Index | $3204_h$ |
| Object name | Feedback Mapping |
| Object Code | ARRAY |
| Data type | UNSIGNED16 |
| Savable | no |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1748-B538662 |
| Change history | |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $04_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Index Of 1st Feedback Interface |
| Data type | UNSIGNED16 |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $3380_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Index Of 2nd Feedback Interface |

| | |
|---|---|
| Data type | UNSIGNED16 |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $3390_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | Index Of 3rd Feedback Interface |
| Data type | UNSIGNED16 |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $33A0_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | Index Of 4th Feedback Interface |
| Data type | UNSIGNED16 |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $33B0_h$ |

### Description

The subindices have the following function:

- $00_h$: Value="1" to "n", where "n" is the number of existing feedbacks.
- $n_h$:
  Subindex n refers to the index of the respective object for the configuration of the corresponding feedback.
  Subindex $01_h$ always corresponds to the first feedback *Sensorless* . The order of the remaining feedbacks corresponds to the table in chapter Configuring the sensors.

## 3205h Feedback Use

### Function

This object shows which sensor is currently used for which control circuit.

### Object description

| | |
|---|---|
| Index | $3205_h$ |
| Object name | Feedback Use |
| Object Code | ARRAY |
| Data type | UNSIGNED16 |
| Savable | no |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |

| | |
|---|---|
| Preset value | |
| Firmware version | FIR-v2412-B1057638 |
| Change history | |

## Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $04_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Sensor For Position Control Loop |
| Data type | UNSIGNED16 |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $0000_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Sensor For Velocity Control Loop |
| Data type | UNSIGNED16 |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $0000_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | Sensor For Current Control Loop |
| Data type | UNSIGNED16 |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $0000_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | Sensor For Pulse Generator |
| Data type | UNSIGNED16 |
| Access | read only |
| PDO mapping | TX-PDO |

| | |
|---|---|
| Allowed values | |
| Preset value | $0000_h$ |

## 320Dh Moment Of Inertia

### Function

This factor is used for calculating the acceleration feed forward (see 321D). Default is 0 (feed forward inactive).

Acceleration feed forward applies during deceleration as well.

### Object description

| | |
|---|---|
| Index | $320D_h$ |
| Object name | Moment Of Inertia |
| Object Code | ARRAY |
| Data type | UNSIGNED32 |
| Savable | yes, category: application |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1825-B577172 |
| Change history | Firmware version FIR-v2451-B1068465: "Savable" entry changed from "yes, category: movement" to "yes, category: application".<br>Firmware version FIR-v2508-B1072143: "Object Name" entry changed from "Torque Of Inertia Factor" to "Moment Of Inertia".<br>Firmware version FIR-v2508-B1072143: "Name" entry changed from "Current" to "Torque [‰]". |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $02_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Torque [‰] |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |

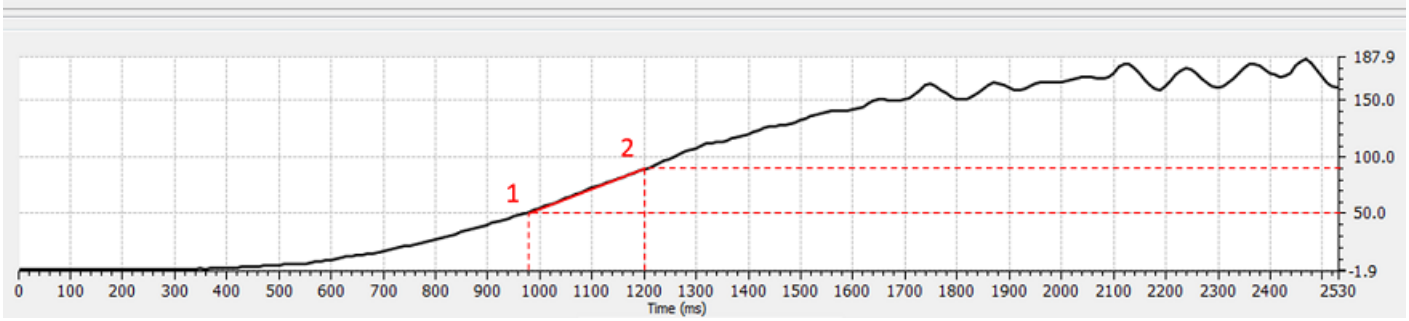
| | |
|---|---|
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Acceleration |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

### Description

The value is dependent on the inertia of the load. To determine the factor:

1. Activate *closed loop* and select the *profile torque* mode.
2. Set a target for the torque and enter the value in $320D_h:01_h$.
3. Record (e. g., in *Plug & Drive Studio*) the current speed (object $606C_h$). Calculate the acceleration in the set user-defined units for the speed range, where this is constant. Enter the value in $320D_h:02_h$.
   Using the speed curve in the following figure as an example:
   (90-50)/(1200-980)=182 rpm/s.




## 3211h Motor Drive Set Point

### Function

For reading out the Id/Iq and Ud/Uq set points,

### Object description

| | |
|---|---|
| Index | $3211_h$ |
| Object name | Motor Drive Set Point |
| Object Code | ARRAY |
| Data type | INTEGER32 |
| Savable | no |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v2451-B1068465 |
| Change history | |

## Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $04_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Id Set Point [mA] |
| Data type | INTEGER32 |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Iq Set Point [mA] |
| Data type | INTEGER32 |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | Ud Set Point [mV] |
| Data type | INTEGER32 |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | Uq Set Point [mV] |
| Data type | INTEGER32 |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |

## 3219h Current Configuration

### Function

Contains various current settings.

### Object description

| | |
|---|---|
| Index | $3219_h$ |
| Object name | Current Configuration |
| Object Code | ARRAY |
| Data type | UNSIGNED16 |
| Savable | yes, category: application |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v2451-B1068465 |
| Change history | |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $05_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | I²t Duration Of Max Current [ms] |
| Data type | UNSIGNED16 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $03E8_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Open Loop Idle State Detection Time [ms] |
| Data type | UNSIGNED16 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $03E8_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | Open Loop Idle State Current [‰] |
| Data type | UNSIGNED16 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $01F4_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | Block Detection Time [ms] |
| Data type | UNSIGNED16 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00C8_h$ |

| | |
|---|---|
| Subindex | $05_h$ |
| Name | Block Detection Threshold [‰] |
| Data type | UNSIGNED16 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $02BC_h$ |

### Description

The functions are as follows:

- Subindex $01_h$: Maximum duration of the maximum current ($6073_h$) in milliseconds. Only relevant if $6073_h$ > 1000‰.
- Subindex $02_h$: Time in milliseconds that the motor must be at a standstill before current reduction is activated.
- Subindex $03_h$: Reduced current as tenths of a percent of the rated current ($6075_h$) if the motor is at a standstill in *open-loop*.
- Subindex $04_h$: Time in milliseconds that the motor is to continue to travel against the block after block detection (see *homing on block* in mode Homing).
- Subindex $05_h$: Current limit value in tenths of a percent of the rated current ($6075_h$) above which blocking is to be detected (see *Homing on block* in mode Homing).

## 321Ah Current Controller Parameters

### Function

Contains the parameters for the current controller (commutation). As a rule, the values for Iq (subindex $01_h$/$02_h$) and Id (subindex $03_h$/$04_h$) should be the same. See chapter Controller structure.

### Object description

| | |
|---|---|
| Index | $321A_h$ |
| Object name | Current Controller Parameters |

| | |
|---|---|
| Object Code | ARRAY |
| Data type | UNSIGNED32 |
| Savable | yes, category: application |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v2213-B1028181 |
| Change history | |

## Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $04_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Proportional Gain Kp For Iq [mV/A] |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00001DB0_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Integrator Time Ti For Iq [µs] |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $000018BD_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | Proportional Gain Kp For Id [mV/A] |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00001DB0_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | Integrator Time Ti For Id [µs] |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $000018BD_h$ |

## 321Bh Velocity Controller Parameters

### Function

Contains the parameters for the velocity controller. See chapter Controller structure.

### Object description

| | |
|---|---|
| Index | $321B_h$ |
| Object name | Velocity Controller Parameters |
| Object Code | ARRAY |
| Data type | UNSIGNED32 |
| Savable | yes, category: application |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v2213-B1028181 |
| Change history | |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $02_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Proportional Gain Kp [mA/Hz] |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000140_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Integrator Time Ti [µs] |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $0000C350_h$ |

## 321Ch Position Controller Parameters

### Function

Contains the parameters for the position controller. See chapter Controller structure.

### Object description

| | |
|---|---|
| Index | $321C_h$ |
| Object name | Position Controller Parameters |
| Object Code | ARRAY |
| Data type | UNSIGNED32 |
| Savable | yes, category: application |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v2213-B1028181 |
| Change history | |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $02_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Proportional Gain Kp [Hz] |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $0000008F_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Integrator Time Ti [µs] |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

## 321Dh Feedforward

### Function

Contains the parameters for the feed forward. See chapter Controller structure.

### Object description

| | |
|---|---|
| Index | $321D_h$ |
| Object name | Feedforward |
| Object Code | ARRAY |
| Data type | UNSIGNED16 |
| Savable | yes, category: application |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v2213-B1028181 |
| Change history | Firmware version FIR-v2315-B1040535: "Data type" entry changed from "UNSIGNED32" to "UNSIGNED16".<br>Firmware version FIR-v2315-B1040535: "Data type" entry changed from "UNSIGNED32" to "UNSIGNED16".<br>Firmware version FIR-v2315-B1040535: "Data type" entry changed from "UNSIGNED32" to "UNSIGNED16".<br>Firmware version FIR-v2315-B1040535: "Data type" entry changed from "UNSIGNED32" to "UNSIGNED16".<br>Firmware version FIR-v2412-B1057638: "Object Name" entry changed from "Pre-control" to "Feedforward".<br>Firmware version FIR-v2412-B1057638: "Name" entry changed from "Voltage Pre-control For Dq-decoupling [‰]" to "Voltage Feedforward For Dq-decoupling [‰]".<br>Firmware version FIR-v2412-B1057638: "Name" entry changed from "Acceleration Pre-control [‰]" to "Acceleration Feedforward [‰]".<br>Firmware version FIR-v2412-B1057638: "Name" entry changed from "Velocity Pre-control [‰]" to "Velocity Feedforward [‰]".<br>Firmware version FIR-v2412-B1057638: "Name" entry changed from "Ohmic Based Voltage Pre-control [‰]" to "Ohmic Based Voltage Feedforward [‰]". |

## Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $04_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Voltage Feedforward For Dq-decoupling [‰] |
| Data type | UNSIGNED16 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $03E8_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Acceleration Feedforward [‰] |
| Data type | UNSIGNED16 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $03E8_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | Velocity Feedforward [‰] |
| Data type | UNSIGNED16 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $03E8_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | Ohmic Based Voltage Feedforward [‰] |
| Data type | UNSIGNED16 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $0000_h$ |

## 321Eh Voltage Limit

### Function

Maximum permissible PWM voltage (duty cycle) in millivolt. See also chapter Controller structure.

### Object description

| | |
|---|---|
| Index | $321E_h$ |
| Object name | Voltage Limit |
| Object Code | VARIABLE |
| Data type | UNSIGNED32 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $000186A0_h$ |
| Firmware version | FIR-v2213-B1028181 |
| Change history | |

### Description

Also dependent on this value is whether the *overmodulation* of the voltage vector is used. If *overmodulation* is used, a higher torque can be achieved. The resulting voltage is no longer sinusoidal, which can result in harmonics and higher losses.

| Value in mV | Overmodulation |
|---|---|
| 1001…$U_{o_low}$ | None; the voltage vector describes a circle. |
| $U_{o_low}$…$U_{o_high}$ | The voltage vector describes a circle that is increasingly flattened on four/six sides in proportion to the set value. |
| ≥$U_{o_high}$ | Full; the voltage vector describes a square or a hexagon. |

**$U_{o_low}$**

The lowest voltage above which overmodulation occurs. Is calculated as follows (small deviations possible depending on the hardware):

Operating voltage*0.95

**$U_{o_high}$**

The maximum overmodulation occurs above this voltage. Is calculated as follows (small deviations possible depending on the hardware):

With two-phase stepper motors: operating voltage*1.128

With three-phase BLDC motors: operating voltage*1.05

## 3220h Analog Input Digits

### Function

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

### Object description

| | |
|---|---|
| Index | $3220_h$ |
| Object name | Analog Input Digits |
| Object Code | ARRAY |
| Data type | INTEGER16 |
| Savable | no |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1426 |
| Change history | |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Analog Input Digits |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $02_h$ |

| | |
|---|---|
| Subindex | $01_h$ - $02_h$ |
| Name | Analog Input #1 - #2 |
| Data type | INTEGER16 |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $0000_h$ |

### Description

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

- Voltage input: x digits * 24 V / 4095 digits
- Current input (if configurable): x digits * 20 mA / 1023 digits

## 323Ah User Pin Settings

### Function

With this object, you can configure the digital inputs/outputs as described in chapter Digital inputs and outputs.

### Object description

| | |
|---|---|
| Index | $323A_h$ |

| | |
|---|---|
| Object name | User Pin Settings |
| Object Code | ARRAY |
| Data type | UNSIGNED8 |
| Savable | yes, category: application |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v2339-B1048823 |
| Change history | |

## Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $02_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Voltage Level Select |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Pull-Up Enable |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00_h$ |

## Description

- Subindex $01_h$: Here, you define the level for the inputs/outputs:
  - Value "0": 5 V
  - Value "1": 24 V (inputs) or +UB _Logic (outputs)

**NOTICE**

For the inputs, always use a voltage that is smaller than the voltage +UB _Logic.

- Subindex $02_h$: Here, you define the type of digital inputs:
  - Value "0" (Pull-Down): High level at 5/24 V on the pin.
  - Value "1" (Pull-Up): High level without external voltage on the pin.

## 3241h Digital Input Position Capture

### Function

With this object, the sensor position can be noted automatically if a level change occurs at the digital input of the home switch or at two other inputs. See chapter Touch Probe (capture).

### Object description

| | |
|---|---|
| Index | $3241_h$ |
| Object name | Digital Input Position Capture |
| Object Code | ARRAY |
| Data type | UNSIGNED32 |
| Savable | yes, category: application |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1446 |
| Change history | Firmware version FIR-v1446: "Data type" entry changed from "UNSIGNED32" to "UNSIGNED8".<br>Firmware version FIR-v1738-B501312: "Name" entry changed from "Encoder Raw Value" to "Sensor Raw Value".<br>Firmware version FIR-v1748-B531667: "PDO mapping" table entry for subindex 00 changed from "no" to "TX-PDO".<br>Firmware version FIR-v1748-B531667: "PDO mapping" table entry for subindex 01 changed from "RX-PDO" to "TX-PDO".<br>Firmware version FIR-v1748-B531667: "PDO mapping" table entry for subindex 02 changed from "RX-PDO" to "TX-PDO".<br>Firmware version FIR-v1748-B531667: "PDO mapping" table entry for subindex 03 changed from "RX-PDO" to "TX-PDO".<br>Firmware version FIR-v1748-B531667: "PDO mapping" table entry for subindex 04 changed from "RX-PDO" to "TX-PDO". |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |

| | |
|---|---|
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $0C_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Control For Capture Of Home Switch |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Capture Count For Capture Of Home Switch |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | Position Value For Capture Of Home Switch |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | Sensor Raw Value For Capture Of Home Switch |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $05_h$ |
| Name | Control For Capture Of Input #3 |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $06_h$ |
| Name | Capture Count For Capture Of Input #3 |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $07_h$ |
| Name | Position Value For Capture Of Input #3 |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $08_h$ |
| Name | Sensor Raw Value For Capture Of Input #3 |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $09_h$ |
| Name | Control For Capture Of Input #4 |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $0A_h$ |
| Name | Capture Count For Capture Of Input #4 |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $0B_h$ |
| Name | Position Value For Capture Of Input #4 |
| Data type | UNSIGNED32 |

| | |
|---|---|
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $0C_h$ |
| Name | Sensor Raw Value For Capture Of Input #4 |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |

### Description

- Subindex $01_h$, $05_h$, $09_h$: This is used to select the type of level change:
  - Deactivate function: Value "0"
  - With rising edge: Value "1"
  - With falling edge: Value "2"
  - Both edges: Value "3"
- Subindex $02_h$. $06_h$, $0A_h$: Specifies the number of the noted level changes since the time the function was started; is reset to 0 if subindex $01_h$ is set to 1, 2 or 3
- Subindex $03_h$, $07_h$, $0B_h$: Encoder position of the level change (in absolute user units as in $6064_h$)
- Subindex $04_h$, $08_h$, $0C_h$: Encoder position of the level change (in sensor increments as in $6063_h$)

## 3242h Digital Input Routing

### Function

This object determines the source of the input routing that ends in $60FD_h$.

### Object description

| | |
|---|---|
| Index | $3242_h$ |
| Object name | Digital Input Routing |
| Object Code | ARRAY |
| Data type | UNSIGNED8 |
| Savable | yes, category: application |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1504 |
| Change history | |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |

| | |
|---|---|
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $20_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Input Source For Bit #0 In 60FDh - Negative Limit Switch |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Input Source For Bit #1 In 60FDh - Positive Limit Switch |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | Input Source For Bit #2 In 60FDh - Home Switch |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | Input Source For Bit #3 In 60FDh - Interlock |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00_h$ |

| | |
|---|---|
| Subindex | $05_h$ |
| Name | Input Source For Bit #4 In 60FDh |
| Data type | UNSIGNED8 |
| Access | read / write |

| | |
|---|---|
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00_h$ |

| | |
|---|---|
| Subindex | $06_h$ |
| Name | Input Source For Bit #5 In 60FDh |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00_h$ |

| | |
|---|---|
| Subindex | $07_h$ |
| Name | Input Source For Bit #6 In 60FDh |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00_h$ |

| | |
|---|---|
| Subindex | $08_h$ |
| Name | Input Source For Bit #7 In 60FDh |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00_h$ |

| | |
|---|---|
| Subindex | $09_h$ |
| Name | Input Source For Bit #8 In 60FDh |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00_h$ |

| | |
|---|---|
| Subindex | $0A_h$ |
| Name | Input Source For Bit #9 In 60FDh |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00_h$ |

| | |
|---|---|
| Subindex | $0B_h$ |
| Name | Input Source For Bit #10 In 60FDh |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00_h$ |

| | |
|---|---|
| Subindex | $0C_h$ |
| Name | Input Source For Bit #11 In 60FDh |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00_h$ |

| | |
|---|---|
| Subindex | $0D_h$ |
| Name | Input Source For Bit #12 In 60FDh |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00_h$ |

| | |
|---|---|
| Subindex | $0E_h$ |
| Name | Input Source For Bit #13 In 60FDh |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00_h$ |

| | |
|---|---|
| Subindex | $0F_h$ |
| Name | Input Source For Bit #14 In 60FDh |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00_h$ |

| | |
|---|---|
| Subindex | $10_h$ |
| Name | Input Source For Bit #15 In 60FDh |
| Data type | UNSIGNED8 |

| | |
|---|---|
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00_h$ |

| | |
|---|---|
| Subindex | $11_h$ |
| Name | Input Source For Bit #16 In 60FDh |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $01_h$ |

| | |
|---|---|
| Subindex | $12_h$ |
| Name | Input Source For Bit #17 In 60FDh |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $02_h$ |

| | |
|---|---|
| Subindex | $13_h$ |
| Name | Input Source For Bit #18 In 60FDh |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $03_h$ |

| | |
|---|---|
| Subindex | $14_h$ |
| Name | Input Source For Bit #19 In 60FDh |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $04_h$ |

| | |
|---|---|
| Subindex | $15_h$ |
| Name | Input Source For Bit #20 In 60FDh |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |

| | |
|---|---|
| Preset value | $05_h$ |

| | |
|---|---|
| Subindex | $16_h$ |
| Name | Input Source For Bit #21 In 60FDh |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $06_h$ |

| | |
|---|---|
| Subindex | $17_h$ |
| Name | Input Source For Bit #22 In 60FDh |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $5A_h$ |

| | |
|---|---|
| Subindex | $18_h$ |
| Name | Input Source For Bit #23 In 60FDh |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $5B_h$ |

| | |
|---|---|
| Subindex | $19_h$ |
| Name | Input Source For Bit #24 In 60FDh |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $07_h$ |

| | |
|---|---|
| Subindex | $1A_h$ |
| Name | Input Source For Bit #25 In 60FDh |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $08_h$ |

| | |
|---|---|
| Subindex | $1B_h$ |

| | |
|---|---|
| Name | Input Source For Bit #26 In 60FDh |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $09_h$ |

| | |
|---|---|
| Subindex | $1C_h$ |
| Name | Input Source For Bit #27 In 60FDh |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $0A_h$ |

| | |
|---|---|
| Subindex | $1D_h$ |
| Name | Input Source For Bit #28 In 60FDh |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $0B_h$ |

| | |
|---|---|
| Subindex | $1E_h$ |
| Name | Input Source For Bit #29 In 60FDh |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $0C_h$ |

| | |
|---|---|
| Subindex | $1F_h$ |
| Name | Input Source For Bit #30 In 60FDh |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | ■ $0D_h$<br>■ N6-1-1-1-S: $E4_h$<br>■ N6-2-1-1-S: $E4_h$<br>■ N6-1-2-1-S: $E4_h$<br>■ N6-2-2-1-S: $E4_h$<br>■ N6-1-3-1-S: $E4_h$<br>■ N6-1-3-2-S: $E4_h$<br>■ N6-2-3-1-S: $E4_h$ |

| | |
|---|---|
| | ■ N6-2-3-2-S: $E4_h$<br>■ N6-1-4-1-S: $E4_h$<br>■ N6-2-4-1-S: $E4_h$<br>■ N6-1-5-1-S: $E4_h$<br>■ N6-2-5-1-S: $E4_h$ |

| | |
|---|---|
| Subindex | $20_h$ |
| Name | Input Source For Bit #31 In 60FDh |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | ■ $0E_h$<br>■ N6-1-1-1-S: $E5_h$<br>■ N6-2-1-1-S: $E5_h$<br>■ N6-1-2-1-S: $E5_h$<br>■ N6-2-2-1-S: $E5_h$<br>■ N6-1-3-1-S: $E5_h$<br>■ N6-1-3-2-S: $E5_h$<br>■ N6-2-3-1-S: $E5_h$<br>■ N6-2-3-2-S: $E5_h$<br>■ N6-1-4-1-S: $E5_h$<br>■ N6-2-4-1-S: $E5_h$<br>■ N6-1-5-1-S: $E5_h$<br>■ N6-2-5-1-S: $E5_h$ |

## 324Ah Inputs

### Function

You read out the current status of the inputs from this object.

### Object description

| | |
|---|---|
| Index | $324A_h$ |
| Object name | Inputs |
| Object Code | ARRAY |
| Data type | UNSIGNED16 |
| Savable | no |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v2339-B1048823 |
| Change history | |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $02_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | User Inputs |
| Data type | UNSIGNED16 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $0000_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Encoder Inputs |
| Data type | UNSIGNED16 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $0000_h$ |

### Description

- Subindex $01_h$: Shows the status of the following inputs:
  - Bits 0 to 5: Digital inputs 1 to 6 (exact number dependent on product)
  - Bits 6 and 7: Analog inputs 1 and 2 as digital input (exact number dependent on product)
- Subindex $02_h$: Shows the status of the following sensor inputs:
  - Bits 0 to 2: Hall sensors 1 to 3 (if present)
  - Bits 3 to 5: Channels A, B, index of the first incremental encoder (if present)
  - Bits 6 to 8: Channels A, B index of the second incremental encoder (if present)

## 3250h Digital Outputs Control

### Function

### Object description

| | |
|---|---|
| Index | $3250_h$ |
| Object name | Digital Outputs Control |
| Object Code | ARRAY |
| Data type | UNSIGNED32 |

| | |
|---|---|
| Savable | yes, category: application |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1426 |
| Change history | Firmware version FIR-v1426: Subindex $01_h$: "Name" entry changed from "Special Function Disable" to "Special Function Enable"<br>Firmware version FIR-v1446: "Name" entry changed from "Special Function Enable" to "No Function".<br>Firmware version FIR-v1512: The number of entries was changed from 6 to 9.<br>Firmware version FIR-v2039: Subindex 09 added |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $01_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Enable Mask [Bit0=StatusLed, Bit1=ErrorLed] |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $FFFFFFFF_h$ |

## 3252h Digital Output Routing

### Function

This object assigns a signal source to an output; this signal source can be controlled with $60FE_h$. You can find details in chapter *Output Routing*.

### Object description

| | |
|---|---|
| Index | $3252_h$ |
| Object name | Digital Output Routing |
| Object Code | ARRAY |
| Data type | UNSIGNED16 |
| Savable | yes, category: application |

| | |
|---|---|
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1540 |
| Change history | |

## Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $04_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Brake Output: Signal In High Byte, Bit Number Of 60FEh:1h In Low Byte |
| Data type | UNSIGNED16 |
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $1080_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Output #1: Signal In High Byte, Bit Number Of 60FEh:1h In Low Byte |
| Data type | UNSIGNED16 |
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $0010_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | Output #2: Signal In High Byte, Bit Number Of 60FEh:1h In Low Byte |
| Data type | UNSIGNED16 |
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $0011_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | Output #3: Signal In High Byte, Bit Number Of 60FEh:1h In Low Byte |

| | |
|---|---|
| Data type | UNSIGNED16 |
| Access | read / write |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $0012_h$ |

## 325Ah Outputs

### Function

Use this object to control the digital outputs (alternative to 60FEh Digital Outputs).

### Object description

| | |
|---|---|
| Index | $325A_h$ |
| Object name | Outputs |
| Object Code | ARRAY |
| Data type | UNSIGNED16 |
| Savable | yes, category: application |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v2339-B1048823 |
| Change history | Firmware version FIR-v2412-B1057638: "Savable" entry changed from "yes, category: communication" to "yes, category: application". |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $01_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | User Outputs |
| Data type | UNSIGNED16 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $0000_h$ |

### Description

- Subindex $01_h$:
  - Bits 0 to 6: Digital outputs 1 to 7 (exact number dependent on product)
  - Bit 7: Brake output (if present)

## 3260h Pwm Output 0

### Function

Use this object to configure the first PWM output. You must define the output as PWM output using *Output Routing*.

### Object description

| | |
|---|---|
| Index | $3260_h$ |
| Object name | Pwm Output 0 |
| Object Code | ARRAY |
| Data type | UNSIGNED32 |
| Savable | yes, category: application |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1939-B682906 |
| Change history | |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $02_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Pwm Frequency |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00002710_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Pwm Duty Cycle |

| | |
|---|---|
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

### Description

The subindices have the following function:

- $01_h$: Frequency of the PWM signal in hertz. 50…20000
- $02_h$: Duty cycle of the PWM signal: 0…100

## 3261h Pwm Output 1

### Function

Use this object to configure the second PWM output. You must define the output as PWM output using *Output Routing*.

### Object description

| | |
|---|---|
| Index | $3261_h$ |
| Object name | Pwm Output 1 |
| Object Code | ARRAY |
| Data type | UNSIGNED32 |
| Savable | yes, category: application |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1939-B682906 |
| Change history | Firmware version FIR-v2542-B1079618: entry "Name" changed from "PWM Frequency" to "Pwm Frequency".<br>Firmware version FIR-v2542-B1079618: entry "Name" changed from "PWM Duty Cycle" to "Pwm Duty Cycle". |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $02_h$ |

| | |
|---|---|
| Subindex | $01_h$ |

| | |
|---|---|
| Name | Pwm Frequency |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00002710_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Pwm Duty Cycle |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

### Description

The subindices have the following function:

- $01_h$: Frequency of the PWM signal in hertz. 50…20000
- $02_h$: Duty cycle of the PWM signal: 0…100

## 3262h Pwm Output 2

### Function

Use this object to configure the third PWM output. You must define the output as PWM output using *Output Routing*.

### Object description

| | |
|---|---|
| Index | $3262_h$ |
| Object name | Pwm Output 2 |
| Object Code | ARRAY |
| Data type | UNSIGNED32 |
| Savable | yes, category: application |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1939-B682906 |
| Change history | |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |

| | |
|---|---|
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $02_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Pwm Frequency |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00002710_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Pwm Duty Cycle |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

### Description

The subindices have the following function:

The subindices have the following function:

- $01_h$: Frequency of the PWM signal in hertz. 50…20000
- $02_h$: Duty cycle of the PWM signal: 0…100

## 3273h Generic SPI Hardware Configuration

### Function

See chapter Generic SPI.

### Object description

| | |
|---|---|
| Index | $3273_h$ |
| Object name | Generic SPI Hardware Configuration |
| Object Code | ARRAY |
| Data type | UNSIGNED32 |
| Savable | yes, category: application |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v2213-B1029645 |
| Change history | |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $01_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Hardware Feature Control |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

## 3274h Generic SPI Mosi Data

### Function

See chapter Generic SPI.

### Object description

| | |
|---|---|
| Index | $3274_h$ |
| Object name | Generic SPI Mosi Data |
| Object Code | ARRAY |
| Data type | UNSIGNED8 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v2213-B1029645 |
| Change history | |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Length Of SPI Message To Be Sent |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |

| | |
|---|---|
| Preset value | $00_h$ |

| | |
|---|---|
| Subindex | $01_h$ - $1F_h$ |
| Name | Generic SPI Mosi Data Byte #1 - #31 |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00_h$ |

## 3275h Generic SPI Miso Data

### Function

See chapter Generic SPI.

### Object description

| | |
|---|---|
| Index | $3275_h$ |
| Object name | Generic SPI Miso Data |
| Object Code | ARRAY |
| Data type | UNSIGNED8 |
| Savable | no |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v2213-B1029645 |
| Change history | |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Length Of Received SPI Message |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00_h$ |

| | |
|---|---|
| Subindex | $01_h$ - $1F_h$ |
| Name | Generic SPI Miso Data Byte #1 - #31 |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |

| | |
|---|---|
| Allowed values | |
| Preset value | $00_h$ |

## 3320h Analog Input Values

### Function

This object displays the instantaneous values of the analog inputs in user-defined units.

### Object description

| | |
|---|---|
| Index | $3320_h$ |
| Object name | Analog Input Values |
| Object Code | ARRAY |
| Data type | INTEGER32 |
| Savable | no |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1426 |
| Change history | |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Analog Input Values |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $02_h$ |

| | |
|---|---|
| Subindex | $01_h$ - $02_h$ |
| Name | Analog Input #1 - #2 |
| Data type | INTEGER32 |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |

### Description

The user-defined units are made up of offset ($3321_h$) and scaling value ($3322_h$/ $3323_h$). If both are still set to the default values, the value in $3320_h$ is specified in the *ADC Digits* unit.

Formula for converting from digits to the respective unit:

- Voltage input: x digits * 24 V / 4095 digits
- Current input (if configurable): x digits * 20 mA / 1023 digits

The following applies for the sub-entries:

- Subindex $00_h$: Number of analog inputs
- Subindex $01_h$: Analog value 1
- Subindex $02_h$: Analog value 2 (if present)

## 3321h Analog Input Offsets

### Function

Offset that is added to the read analog value ($3220_h$) before scaling (multiplier from object 3322 and divisor from object $3323_h$).

### Object description

| | |
|---|---|
| Index | $3321_h$ |
| Object name | Analog Input Offsets |
| Object Code | ARRAY |
| Data type | INTEGER16 |
| Savable | yes, category: application |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v2139-B1022383 |
| Change history | |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Analog Input Offsets |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $02_h$ |

| | |
|---|---|
| Subindex | $01_h$ - $02_h$ |
| Name | Analog Input #1 - #2 |
| Data type | INTEGER16 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $0000_h$ |

## 3322h Analog Input Numerators

### Function

Value by which the read analog value (3220$_h$, 3321$_h$) is multiplied before it is written in object 3320$_h$.

### Object description

| | |
|---|---|
| Index | 3322$_h$ |
| Object name | Analog Input Numerators |
| Object Code | ARRAY |
| Data type | INTEGER16 |
| Savable | yes, category: application |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1426 |
| Change history | |

### Value description

| | |
|---|---|
| Subindex | 00$_h$ |
| Name | Number Of Analog Input Numerators |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | 02$_h$ |

| | |
|---|---|
| Subindex | 01$_h$ - 02$_h$ |
| Name | Analog Input #1 - #2 |
| Data type | INTEGER16 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | 6098$_h$ |

### Description

The subindices contain:

- Subindex 01$_h$: Multiplier for analog input 1
- Subindex 02$_h$: Multiplier for analog input 2 (if present)

## 3323h Analog Input Denominators

### Function

Value by which the read analog value (3220$_h$+ 3321$_h$) is divided before it is written in object 3320$_h$.

### Object description

| | |
|---|---|
| Index | 3323$_h$ |
| Object name | Analog Input Denominators |
| Object Code | ARRAY |
| Data type | INTEGER16 |
| Savable | yes, category: application |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1926-B648637 |
| Change history | |

### Value description

| | |
|---|---|
| Subindex | 00$_h$ |
| Name | Number Of Analog Input Denominators |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | 02$_h$ |

| | |
|---|---|
| Subindex | 01$_h$ - 02$_h$ |
| Name | Analog Input #1 - #2 |
| Data type | INTEGER16 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | 0FFF$_h$ |

### Description

The subindices contain:

- Subindex 01$_h$: Divisor for analog input 1
- Subindex 02$_h$: Divisor for analog input 2 (if present)

## 3380h Feedback Sensorless

### Function

Contains measurement and configuration values that are necessary for the sensorless control in Closed Loop and the current control in *Open Loop*.

### Object description

| | |
|---|---|
| Index | $3380_h$ |
| Object name | Feedback Sensorless |
| Object Code | ARRAY |
| Data type | UNSIGNED32 |
| Savable | yes, category: sensors |
| 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.<br>Firmware version FIR-v2451-B1068465: "Savable" entry changed from "yes, category: tuning" to "yes, category: sensors". |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $05_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Resistance [Ohm] |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Inductance [H] |
| Data type | UNSIGNED32 |
| Access | read / write |

| | |
|---|---|
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | Magnetic Flux [Vs] |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | Switch On Speed [rpm] |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00000078_h$ |

| | |
|---|---|
| Subindex | $05_h$ |
| Name | Switch Off Speed [rpm] |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00000064_h$ |

### Description

The subindices have the following function:

- $01_h$: Winding resistance. Float value, shown here as UNSIGNED32. The value is detected via the Auto setup.
- $02_h$: Winding inductance Ld, absolute. Float value, shown here as UNSIGNED32. The value is detected via the Auto setup.
- $03_h$: Interlinking flux. Float value, shown here as UNSIGNED32. The value is detected via the Auto setup.
- $04_h$: Switch-on speed in RPM. Above this speed, the *Closed Loop* ( *Sensorless*) is activated, when Auto setup has not detected any sensors.
- $05_h$: Switch-off speed in RPM. If the speed falls below this threshold, the *Closed Loop* ( *Sensorless*) is deactivated, when Auto setup has not detected any sensors.

## 3390h Feedback Hall

### Function

Contains configuration values for the Hall sensors. The values are determined by the Auto setup.

## Object description

| | |
|---|---|
| Index | $3390_h$ |
| Object name | Feedback Hall |
| Object Code | ARRAY |
| Data type | UNSIGNED16 |
| Savable | yes, category: sensors |
| Access | read only |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1748-B531667 |
| Change history | Firmware version FIR-v2412-B1057638: "Name" entry changed from "1st Alignment" to "Alignment Of H1 Edge While H3 Low And H2 Low".<br>Firmware version FIR-v2412-B1057638: "Name" entry changed from "2nd Alignment" to "Alignment Of H1 Edge While H3 Low And H2 High".<br>Firmware version FIR-v2412-B1057638: "Name" entry changed from "3rd Alignment" to "Alignment Of H1 Edge While H3 High And H2 Low".<br>Firmware version FIR-v2412-B1057638: "Name" entry changed from "4th Alignment" to "Alignment Of H1 Edge While H3 High And H2 High".<br>Firmware version FIR-v2412-B1057638: "Name" entry changed from "5th Alignment" to "Alignment Of H2 Edge While H3 Low And H1 Low".<br>Firmware version FIR-v2412-B1057638: "Name" entry changed from "6th Alignment" to "Alignment Of H2 Edge While H3 Low And H1 High".<br>Firmware version FIR-v2412-B1057638: "Name" entry changed from "7th Alignment" to "Alignment Of H2 Edge While H3 High And H1 Low".<br>Firmware version FIR-v2412-B1057638: "Name" entry changed from "8th Alignment" to "Alignment Of H2 Edge While H3 High And H1 High".<br>Firmware version FIR-v2412-B1057638: "Name" entry changed from "9th Alignment" to "Alignment Of H3 Edge While H2 Low And H1 Low".<br>Firmware version FIR-v2412-B1057638: "Name" entry changed from "10th Alignment" to "Alignment Of H3 Edge While H2 Low And H1 High".<br>Firmware version FIR-v2412-B1057638: "Name" entry changed from "11th Alignment" to "Alignment Of H3 Edge While H2 High And H1 Low".<br>Firmware version FIR-v2412-B1057638: "Name" entry changed from "12th Alignment" to "Alignment Of H3 Edge While H2 High And H1 High".<br>Firmware version FIR-v2451-B1068465: "Savable" entry changed from "yes, category: tuning" to "yes, category: sensors". |

## Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $0C_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Alignment Of H1 Edge While H3 Low And H2 Low |
| Data type | UNSIGNED16 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $0000_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Alignment Of H1 Edge While H3 Low And H2 High |
| Data type | UNSIGNED16 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $0000_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | Alignment Of H1 Edge While H3 High And H2 Low |
| Data type | UNSIGNED16 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $0000_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | Alignment Of H1 Edge While H3 High And H2 High |
| Data type | UNSIGNED16 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $0000_h$ |

| | |
|---|---|
| Subindex | $05_h$ |

| | |
|---|---|
| Name | Alignment Of H2 Edge While H3 Low And H1 Low |
| Data type | UNSIGNED16 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $0000_h$ |

| | |
|---|---|
| Subindex | $06_h$ |
| Name | Alignment Of H2 Edge While H3 Low And H1 High |
| Data type | UNSIGNED16 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $0000_h$ |

| | |
|---|---|
| Subindex | $07_h$ |
| Name | Alignment Of H2 Edge While H3 High And H1 Low |
| Data type | UNSIGNED16 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $0000_h$ |

| | |
|---|---|
| Subindex | $08_h$ |
| Name | Alignment Of H2 Edge While H3 High And H1 High |
| Data type | UNSIGNED16 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $0000_h$ |

| | |
|---|---|
| Subindex | $09_h$ |
| Name | Alignment Of H3 Edge While H2 Low And H1 Low |
| Data type | UNSIGNED16 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $0000_h$ |

| | |
|---|---|
| Subindex | $0A_h$ |
| Name | Alignment Of H3 Edge While H2 Low And H1 High |
| Data type | UNSIGNED16 |
| Access | read / write |

| | |
|---|---|
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $0000_h$ |

| | |
|---|---|
| Subindex | $0B_h$ |
| Name | Alignment Of H3 Edge While H2 High And H1 Low |
| Data type | UNSIGNED16 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $0000_h$ |

| | |
|---|---|
| Subindex | $0C_h$ |
| Name | Alignment Of H3 Edge While H2 High And H1 High |
| Data type | UNSIGNED16 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $0000_h$ |

## 33A0h Feedback Incremental A/B/I 1

### Function

Contains configuration values for the first incremental encoder. The values are determined by the Auto setup.

### Object description

| | |
|---|---|
| Index | $33A0_h$ |
| Object name | Feedback Incremental A/B/I 1 |
| Object Code | ARRAY |
| Data type | UNSIGNED16 |
| Savable | yes, category: sensors |
| Access | read only |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1738-B501312 |
| Change history | Firmware version FIR-v2451-B1068465: "Savable" entry changed from "yes, category: tuning" to "yes, category: sensors". |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |

| | |
|---|---|
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $03_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Configuration |
| Data type | UNSIGNED16 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $0000_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Alignment |
| Data type | UNSIGNED16 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $0000_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | Latency [ns] |
| Data type | UNSIGNED16 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $0000_h$ |

### Description

The subindices have the following function:

- $01_h$ (Configuration): The following bits have a meaning:
  - Bit 0: Value = "0": The encoder does not have an index. Value = "1": Encoder index exists and is to be used.
- $02_h$ (Alignment): This value specifies the offset between the index of the encoder and the rotor's magnets. The exact determination is possible via auto setup. The presence of this value is necessary for *closed-loop* mode with encoder.
- $03_h$ (Latency): Here you can enter the latency of the used encoder.

## 33B0h Feedback SSI 1

### Function

Contains configuration values for the first SSI encoder.

## Object description

| | |
|---|---|
| Index | $33B0_h$ |
| Object name | Feedback SSI 1 |
| Object Code | RECORD |
| Data type | SSI ENCODER |
| Savable | yes, category: sensors |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1939-B682906 |
| Change history | Firmware version FIR-v2451-B1068465: "Savable" entry changed from "yes, category: tuning" to "yes, category: sensors". |

## Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $0D_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Configuration |
| Data type | UNSIGNED16 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $0000_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Alignment |
| Data type | UNSIGNED16 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $0000_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | Home Position Low |
| Data type | INTEGER32 |

| | |
|---|---|
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | Home Position High |
| Data type | INTEGER32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $05_h$ |
| Name | Number Of Bits For Transfer |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $17_h$ |

| | |
|---|---|
| Subindex | $06_h$ |
| Name | Baud Rate |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $000F4240_h$ |

| | |
|---|---|
| Subindex | $07_h$ |
| Name | Position Bitmask Low |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $007FFFC0_h$ |

| | |
|---|---|
| Subindex | $08_h$ |
| Name | Position Bitmask High |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |

| | |
|---|---|
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $09_h$ |
| Name | Status Bitmask Low |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000001_h$ |

| | |
|---|---|
| Subindex | $0A_h$ |
| Name | Status Bitmask High |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $0B_h$ |
| Name | Status Value Low |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000001_h$ |

| | |
|---|---|
| Subindex | $0C_h$ |
| Name | Status Value High |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $0D_h$ |
| Name | Latency [ns] |
| Data type | INTEGER32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

## Description

The subindices have the following function:

- $01_h$ (Configuration):
  - Bit 0: Value = "0": Alignment has not yet been determined or is not to be used. Value = "1": Alignment exists and is to be used.
- $02_h$ (Alignment): This value specifies the offset between the zero position of the encoder and the rotor's magnets.
  The exact determination is only possible via auto setup. The presence of this value is necessary for *closed-loop* mode with encoder.
- $03_h$ (Home Position Low) and $04_h$ (Home Position High): The absolute encoder position after a homing has been completed is entered in these subindices.
- $05_h$ (Number Of Bits For Transfer): Number of bits in a message (encoder data). Maximum 64 bits.
- $06_h$ (Baud Rate): Baud rate of the interface in hertz. The following frequencies are supported: 3.375 MHz, 1.6875 MHz, 656.25 kHz, 843.75 kHz, 421.875 kHz, 210.9375 kHz, 105.46875 kHz . If the values are different, the valid frequency with the smallest difference is selected.
- $07_h$ (Position Bitmask Low) and $08_h$ (Position Bitmask High): In these subindices, you enter a bitmask that determines which bits of the encoder data contain the position data (see following instructions).
- $09_h$ (Status Bitmask Low) and $0A_h$ (Status Bitmask High): In these subindices, you enter a bitmask that determines which bits of the encoder data contain the status information (see following instructions).
- $0B_h$ (Status Value Low) and $0C_h$ (Status Value High): In these subindices, you enter a bitmask that determines which value the status information bits (subindices $09_h$ and $0A_h$) must have (see following instructions). A different value at this point of the encoder data is interpreted as an error by the controller.
- $0D_h$ (Latency): Latency of the encoder in nanoseconds, specified in the encoder data sheet.

To set the configuration according to your encoder:

1. Set the baud rate in subindex $06_h$ and the number of bits in subindex $05_h$ according to the encoder data sheet.
   When doing so, or when you select the encoder, take into account the pause/dead time to be kept between the data packages and the fact that the controller reads the sensor at 16 kHz (62.5 µs). For example, with an encoder pause time of 40 µs, only 22.5 µs remain for transmission, which corresponds to 18.98 bits (a maximum of 2 bytes) at a baud rate of, say, 843.75 kHz.

**TIP**

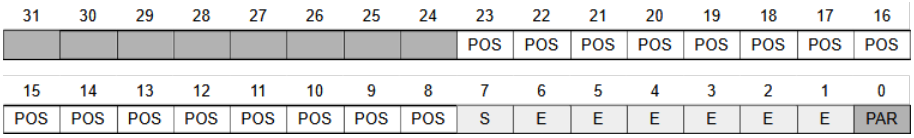
In order to keep the pause time, you can omit unused or unimportant bits at the end of the message when specifying the number of bits in subindex $05_h$, such as the parity bit in the example below.

2. Define which bits the position data should include and set subindices $07_h$ and $08_h$ to the corresponding value.
3. Define which bits the status information (e. g., status, error, etc.) should include and set subindices $09_h$ and $0A_h$ to the corresponding value.
4. Define which value, "0" or "1", the status information bits must have and set the corresponding bits in subindices $09_h$ and $0A_h$ to the value.
5. Store the object by writing the value "$65766173_h$" in $1010_h$:$06_h$ and restart the controller.

**Example**

Bits 8…23 contain the position, Bit 0 is the parity bit, which is ignored by the controller. The status information is divided into the following bits:

- Bits 1…6 are error bits that should always have the value "0".
- Bit 7 is the status bit that has the value "1" if the encoder is ready.




The controller always internally adds the latch bit and rounds up to the nearest multiple of eight. The data is therefore transmitted in a 32-bit message (transmission can only occur byte by byte), if the parity bit is not omitted.

You must enter the following values in the subindices:

- $05_h$ (Number Of Bits For Transfer): $24_h$(with the parity bit)
- $07_h$(Position Bitmask Low) $FFFF00_h$
- $09_h$ (Status Bitmask Low): $FE_h$
- $0B_h$ (Status Value Low): $80_h$

Subindices $08_h$, $0A_h$ and $0C_h$, which would contain the most-significant 32 bits of a 64-bit message, have the value "0".

## 3700h Deviation Error Option Code

### Function

The object contains the action that is to be executed if a following or slippage error is triggered.

### Object description

| | |
|---|---|
| Index | $3700_h$ |
| Object name | Deviation Error Option Code |
| Object Code | VARIABLE |
| Data type | INTEGER16 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $FFFF_h$ |
| Firmware version | FIR-v1426 |
| Change history | Firmware version FIR-v1738-B501312: "Object Name" entry changed from "Following Error Option Code" to "Deviation Error Option Code". |

### Description

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

## 3701h Limit Switch Error Option Code

### Function

If a limit switch is triggered, the limit switch position is stored internally, bit 7 ( *Warning*) in 6041$_h$ ( *statusword*) is set and the CiA 402 Power State Machine is set to status *Quick Stop Active*. The action stored in this object is thereby executed. See chapter Limitation of the range of motion.

### Object description

| | |
|---|---|
| Index | 3701$_h$ |
| Object name | Limit Switch Error Option Code |
| Object Code | VARIABLE |
| Data type | INTEGER16 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | FFFF$_h$ |
| Firmware version | FIR-v1748-B538662 |
| Change history | |

### Description

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

## 4014h Operating Conditions

### Function

This object is used to read out the current environment values for the controller.

## Object description

| | |
|---|---|
| Index | $4014_h$ |
| Object name | Operating Conditions |
| Object Code | ARRAY |
| Data type | INTEGER32 |
| Savable | no |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1540 |
| Change history | Firmware version FIR-v1650-B472161: "Access" table entry for subindex 01 changed from "read/write" to "read only".<br>Firmware version FIR-v1650-B472161: "Access" table entry for subindex 02 changed from "read/write" to "read only".<br>Firmware version FIR-v1650-B472161: "Name" entry changed from "Temperature PCB [d?C]" to "Temperature PCB [Celsius * 10]".<br>Firmware version FIR-v1650-B472161: "Access" table entry for subindex 03 changed from "read/write" to "read only".<br>Firmware version FIR-v1738-B501312: The number of entries was changed from 4 to 6.<br>Firmware version FIR-v2451-B1068465: The number of entries was changed from 6 to 5.<br>Firmware version FIR-v2451-B1068465: "Name" entry changed from "Temperature PCB [Celsius * 10]" to "Temperature PCB [d°C]".<br>Firmware version FIR-v2451-B1068465: "Name" entry changed from "Temperature Motor [Celsius * 10]" to "Temperature Microcontroller Chip [d°c]". |

## Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $04_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Voltage UB Power [mV] |
| Data type | INTEGER32 |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |

| | |
|---|---|
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Voltage UB Logic [mV] |
| Data type | INTEGER32 |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | Temperature PCB [d°C] |
| Data type | INTEGER32 |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | Temperature Microcontroller Chip [d°C] |
| Data type | INTEGER32 |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |

### Description

The subindices contain:

- $01_h$: Current voltage supply voltage in [mV]
- $02_h$: Current logic voltage in [mV]
- $03_h$: Current temperature of the control board in [d°C] (tenths of degree)
- $04_h$: Current temperature of the processor in [d°C] (tenths of degree)

## 4015h Special Drive Modes

### Function

With this object, you can switch the *special drive modes* off or on. See chapter .

### Object description

| | |
|---|---|
| Index | $4015_h$ |
| Object name | Special Drive Modes |
| Object Code | ARRAY |
| Data type | UNSIGNED8 |
| Savable | yes, category: application |

| | |
|---|---|
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1738-B501312 |
| Change history | |

## Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $02_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Special Drive Mode Configuration |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Virtual Config Switch Value |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00_h$ |

## Description

The subindices have the following functions:

- $01_h$:
  - Value ="0"$_h$: The *special drive modes* are switched off
  - Value="2"$_h$: The *special drive modes* are switched on and the mode is set in subindex $02_h$.
- $02_h$: Defines the used mode.

## 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 brake chopper (ballast).

### Object description

| | |
|---|---|
| Index | $4021_h$ |
| Object name | Ballast Configuration |
| Object Code | ARRAY |
| Data type | UNSIGNED32 |
| Savable | yes, category: application |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v2013-B726332 |
| Change history | Firmware version FIR-v2451-B1068465: "Savable" entry changed from "yes, category: tuning" to "yes, category: application". |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $08_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Settings [Bit0: On/Off, Bit1: Polarity, Bit2: 0(Internal)/1(External)] |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000004_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | UB Power Limit [mV] |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |

| | |
|---|---|
| Preset value | 000186A0$_h$ |

| | |
|---|---|
| Subindex | 03$_h$ |
| Name | UB Power Hysteresis [mV] |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | 000001F4$_h$ |

| | |
|---|---|
| Subindex | 04$_h$ |
| Name | Nominal Resistance [mOhm] |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | 00000001$_h$ |

| | |
|---|---|
| Subindex | 05$_h$ |
| Name | Long Term Energy Limit [mWs] |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | 00000001$_h$ |

| | |
|---|---|
| Subindex | 06$_h$ |
| Name | Long Term Reference Time [ms] |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | 00000000$_h$ |

| | |
|---|---|
| Subindex | 07$_h$ |
| Name | Short Term Energy Limit [mWs] |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | 00000000$_h$ |

| | |
|---|---|
| Subindex | 08$_h$ |

| | |
|---|---|
| Name | Cooling Power [mW] |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

### Description

The subindices have the following function:

- $01_h$:
  - Bit 0: Switches the ballast on (value = "1") or off (value = "0")
- $02_h$: Response threshold (switch on/off) of the ballast circuit
- $03_h$: Hysteresis for the response threshold (switch on/off)
- $04_h$: Rated value of the ballast resistor
- $05_h$: Amount of energy that can be supplied to the resistor within the *Long Term Reference Time* (subindex $06_h$) without overloading it.
- $06_h$: The reference time for the *Long Term Energy Limit* (subindex $05_h$) (typically between 1 and 5 seconds)
- $07_h$: Amount of energy that can be supplied to the resistor within a short load surge (<1 second) without overloading it.
- $08_h$: 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 | $4040_h$ |
| Object name | Drive Serial Number |
| Object Code | VARIABLE |
| Data type | VISIBLE_STRING |
| Savable | no |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | 0 |
| Firmware version | FIR-v1450 |
| Change history | |

## 4041h Device Id

### Function

This object contains the ID of the device.

### Object description

| | |
|---|---|
| Index | $4041_h$ |
| Object name | Device Id |
| Object Code | VARIABLE |
| Data type | OCTET_STRING |
| Savable | no |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | 0 |
| Firmware version | FIR-v1540 |
| Change history | |

## 4042h Bootloader Infos

### Object description

| | |
|---|---|
| Index | $4042_h$ |
| Object name | Bootloader Infos |
| Object Code | ARRAY |
| Data type | UNSIGNED32 |
| Savable | no |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v2013-B726332 |
| Change history | |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $03_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Bootloader Version |
| Data type | UNSIGNED32 |
| Access | read only |
| PDO mapping | no |

| | |
|---|---|
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Bootloader Supported Fieldbus |
| Data type | UNSIGNED32 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | Bootloader Hw-group |
| Data type | UNSIGNED32 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

### Description

The subindices have the following functions:

- $01_h$: Version of the boot loader. The 4 most significant bytes contain the main version number; the 4 least significant bytes contain the minor version number. Example for version 4.2: $00040002_h$
- $02_h$: Fieldbuses supported by the boot loader. The bits have the same function as the bits of object 2101h Fieldbus Module Availability.

## 6007h Abort Connection Option Code

### Function

If an error (watchdog, heartbeat, etc.) occurs on the bus, the controller automatically switches to the SAFEOPERATIONAL state. With this object, you can set the reaction.

### Object description

| | |
|---|---|
| Index | $6007_h$ |
| Object name | Abort Connection Option Code |
| Object Code | VARIABLE |
| Data type | INTEGER16 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | ■ $0001_h$<br>■ N6-1-1-1-S: $FFFF_h$<br>■ N6-1-1-1: $FFFF_h$<br>■ N6-2-1-1-S: $FFFF_h$ |

| | |
|---|---|
| | ■ N6-2-1-1: $FFFF_h$ |
| Firmware version | FIR-v2013-B726332 |
| Change history | |

### Description

You can set the following reactions:

| Value | Reaction |
|---|---|
| -1 | The controller (slave) sets all input values (RX-PDO) to the value "0". This also sets the object $6040_h$ (controlword) to "0", which causes the motor to coast to a stop. |
| 0 | no reaction |
| 1 | *Fault*: The action stored in object $605E_h$ is executed. |
| 2 | *Disable voltage*: Transition to the *Switched on disabled* state without halt motion reaction (the motor coasts to a stop) |
| 3 | *Quick stop*: The action stored in object $605A_h$ is executed. |

## 603Fh Error Code

### Function

This object returns the error code of the last error that occurred.

It corresponds to the lower 16 bits of object $1003_h$. For the description of the error codes, refer to object $1003_h$.

### Object description

| | |
|---|---|
| Index | $603F_h$ |
| Object name | Error Code |
| Object Code | VARIABLE |
| Data type | UNSIGNED16 |
| Savable | no |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $0000_h$ |
| Firmware version | FIR-v1426 |
| Change history | |

### Description

For the meaning of the error, see object $1003_h$ (Pre-defined Error Field).

If the error is reset by setting bit 7 in 6040h Controlword, this object is also automatically reset to "0".

## 6040h Controlword

### Function

This object controls the CiA 402 Power State Machine.

### Object description

| | |
|---|---|
| Index | $6040_h$ |
| Object name | Controlword |
| Object Code | VARIABLE |
| Data type | UNSIGNED16 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $0000_h$ |
| Firmware version | FIR-v1426 |
| Change history | Firmware version FIR-v1626: "Savable" entry changed from "no" to "yes, category: application". |

### Description

Parts of the object are, with respect to function, dependent on the currently selected mode.

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | OMS | HALT | FR | OMS [3] | | | EO | QS | EV | SO |

**SO (Switched On)**

Value = "1": Switches to the "Switched on" state

**EV (Enable Voltage)**

Value = "1": Switches to the "Enable voltage" state

**QS (Quick Stop)**

Value = "0": Switches to the "Quick stop" state

**EO (Enable Operation)**

Value = "1": Switches to the "Enable operation" state

**OMS (Operation Mode Specific)**

Meaning is dependent on the selected operating mode

**FR (Fault Reset)**

Resets an error or a warning (if possible)

**HALT**

Value = "1": Triggers a halt; valid in the following modes:

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

# 6041h Statusword

## Function

This object returns information about the status of the CiA 402 Power State Machine.

## Object description

| | |
|---|---|
| Index | $6041_h$ |
| Object name | Statusword |
| Object Code | VARIABLE |
| Data type | UNSIGNED16 |
| Savable | no |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $0000_h$ |
| Firmware version | FIR-v1426 |
| Change history | |

## Description

Parts of the object are, with respect to function, dependent on the currently selected mode. Refer to the corresponding section in chapter operating modes.

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CLA | | OMS [2] | | ILA | TARG | REM | SYNC | WARN | SOD | QS | VE | FAULT | OE | SO | RTSO |

**RTSO (Ready To Switch On)**

Value = "1": Controller is in the "Ready to switch on" state

**SO (Switched On)**

Value = "1": Controller is in the "Switched on" state

**OE (Operation Enabled)**

Value = "1": Controller is in the "Operation enabled" state

**FAULT**

Error occurred (see $1003_h$)

**VE (Voltage Enabled)**

Voltage applied

**QS (Quick Stop)**

Value = "0": Controller is in the "Quick stop" state

**SOD (Switched On Disabled)**

Value = "1": Controller is in the "Switched on disabled" state

**WARN (Warning)**

Value = "1": Warning

**SYNC (synchronization)**

Value = "1": Controller is in sync with the fieldbus; value = "0": Controller is not in sync with the fieldbus

**REM (Remote)**

Remote (value of the bit is always "1" )

**TARG**

Target reached. The value remains at "1" even after an operating mode change until a new target is set.

**ILA (Internal Limit Active)**

Limit exceeded

**OMS (Operation Mode Specific)**

Meaning is dependent on the selected operating mode

**CLA (Closed Loop Active)**

Value = "1": The controller is in the *Operation enabled* state and the Closed Loop is activated.

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

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

## 605Ah Quick Stop Option Code

### Function

The object contains the action that is to be executed on a transition of the CiA 402 Power State Machine to the *Quick Stop active* state.

### Object description

| | |
|---|---|
| Index | $605A_h$ |
| Object name | Quick Stop Option Code |
| Object Code | VARIABLE |
| Data type | INTEGER16 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $0002_h$ |
| Firmware version | FIR-v1426 |

| Change history | |
|---|---|

### Description

| Value in object $605A_h$ | Description |
|---|---|
| 0 | Immediate stop with subsequent state change to *Switch on disabled* |
| 1 | Braking with *slow down ramp* (deceleration ramp depending on operating mode) and subsequent state change to *Switch on disabled* |
| 2 | Braking with *quick stop ramp* ($6085_h$) and subsequent state change to *Switch on disabled* |
| 5 | Braking with *slow down ramp* (deceleration ramp depending on operating mode) and subsequent state change to *Quick stop active*; control does not switch off and the motor remains energized. You can switch back to the *Operation enabled* state. |
| 6 | Braking with *quick stop ramp* ($6085_h$) and subsequent state change to *Quick Stop Active*; control does not switch off and the motor remains energized. You can switch back to the *Operation enabled* state. |

In Homing mode, the deceleration ramp set in $609A_h$ (Homing Acceleration) is used instead of $6085_h$.

## 605Bh Shutdown Option Code

### Function

This object contains the action that is to be executed on a transition of the CiA 402 Power State Machine from the *Operation enabled* state to the *Ready to switch on* state.

### Object description

| | |
|---|---|
| Index | $605B_h$ |
| Object name | Shutdown Option Code |
| Object Code | VARIABLE |
| Data type | INTEGER16 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $0000_h$ |
| Firmware version | FIR-v1426 |
| Change history | |

### Description

| Value in object $605B_h$ | Description |
|---|---|
| -32768 … -1 | Reserved |
| 0 | Blocking of the drive function – motor can turn freely |

| Value in object $605B_h$ | Description |
|---|---|
| 1 | Braking with *slow down ramp* (braking deceleration depending on operating mode) and subsequent state change to *Ready to switch on* |
| 2 … 32767 | Reserved |

## 605Ch Disable Operation Option Code

### Function

This object contains the action that is to be executed on a transition of the CiA 402 Power State Machine from the *Operation enabled* state to the *Switched on* state.

### Object description

| | |
|---|---|
| Index | $605C_h$ |
| Object name | Disable Operation Option Code |
| Object Code | VARIABLE |
| Data type | INTEGER16 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $0001_h$ |
| Firmware version | FIR-v1426 |
| Change history | Firmware version FIR-v2412-B1057638: "Object Name" entry changed from "Disable Option Code" to "Disable Operation Option Code". |

### Description

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

## 605Dh Halt Option Code

### Function

The object contains the action that is to be executed if bit 8 (Halt) is set in controlword $6040_h$.

### Object description

| | |
|---|---|
| Index | $605D_h$ |
| Object name | Halt Option Code |
| Object Code | VARIABLE |

| | |
|---|---|
| Data type | INTEGER16 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $0001_h$ |
| Firmware version | FIR-v1426 |
| Change history | |

### Description

| Value in object $605D_h$ | Description |
|---|---|
| -32768 … 0 | Reserved |
| 1 | Braking with *slow down ramp* (braking deceleration depending on operating mode) |
| 2 | Braking with *quick stop ramp* ($6085_h$) |
| 3 … 32767 | Reserved |

## 605Eh Fault Reaction Option Code

### Function

The object contains the action specifying how the motor is to be brought to a standstill in case of an error.

### Object description

| | |
|---|---|
| Index | $605E_h$ |
| Object name | Fault Reaction Option Code |
| Object Code | VARIABLE |
| Data type | INTEGER16 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $0002_h$ |
| Firmware version | FIR-v1426 |
| Change history | Firmware version FIR-v2412-B1057638: "Object Name" entry changed from "Fault Option Code" to "Fault Reaction Option Code". |

### Description

| Value in object $605E_h$ | Description |
|---|---|
| -32768 … -1 | Reserved |
| 0 | Blocking of the drive function – motor can turn freely |
| 1 | Braking with *slow down ramp* (braking deceleration depending on operating mode) |

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

## 6060h Modes Of Operation

### Function

The desired operating mode is entered in this object.

### Object description

| | |
|---|---|
| Index | $6060_h$ |
| Object name | Modes Of Operation |
| Object Code | VARIABLE |
| Data type | INTEGER8 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00_h$ |
| Firmware version | FIR-v1426 |
| Change history | Firmware version FIR-v1626: "Savable" entry changed from "no" to "yes, category: application". |

### Description

| Mode | Description |
|---|---|
| -2 | Auto setup |
| -1 | Clock-direction mode |
| 0 | No mode change/no mode assigned |
| 1 | Profile Position Mode |
| 3 | Profile Velocity Mode |
| 4 | Profile Torque Mode |
| 5 | Reserved |
| 6 | Homing Mode |
| 7 | Interpolated Position Mode |
| 8 | Cyclic Synchronous Position Mode |
| 9 | Cyclic Synchronous Velocity Mode |
| 10 | Cyclic Synchronous Torque Mode |

## 6061h Modes Of Operation Display

### Function

Indicates the current operating mode. See also 6060h Modes Of Operation.

### Object description

| | |
|---|---|
| Index | $6061_h$ |
| Object name | Modes Of Operation Display |
| Object Code | VARIABLE |
| Data type | INTEGER8 |
| Savable | no |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00_h$ |
| Firmware version | FIR-v1426 |
| Change history | |

## 6062h Position Demand Value

### Function

Indicates the current demand position in user-defined units.

### Object description

| | |
|---|---|
| Index | $6062_h$ |
| Object name | Position Demand Value |
| Object Code | VARIABLE |
| Data type | INTEGER32 |
| Savable | no |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |
| Firmware version | FIR-v1426 |
| Change history | |

## 6063h Position Actual Internal Value

### Function

Contains the current actual position in encoder increments. Unlike objects $6062_h$ and $6064_h$, this value is not set to "0" following a Homing operation. The source is determined in 3203h Feedback Selection.

**NOTICE**

If the encoder resolution in object $60E6_h$ = zero, the numerical values of this object are invalid.

### Object description

| | |
|---|---|
| Index | $6063_h$ |

| | |
|---|---|
| Object name | Position Actual Internal Value |
| Object Code | VARIABLE |
| Data type | INTEGER32 |
| Savable | no |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |
| Firmware version | FIR-v1426 |
| Change history | |

## 6064h Position Actual Value

### Function

Contains the current actual position in user-defined units. The source is determined in 3203h Feedback Selection.

### Object description

| | |
|---|---|
| Index | $6064_h$ |
| Object name | Position Actual Value |
| Object Code | VARIABLE |
| Data type | INTEGER32 |
| Savable | no |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |
| Firmware version | FIR-v1426 |
| Change history | |

## 6065h Following Error Window

### Function

Defines the maximum allowed following error in user-defined units symmetrically to the demand position.

### Object description

| | |
|---|---|
| Index | $6065_h$ |
| Object name | Following Error Window |
| Object Code | VARIABLE |
| Data type | UNSIGNED32 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |

| | |
|---|---|
| Preset value | $00000100_h$ |
| Firmware version | FIR-v1426 |
| Change history | Firmware version FIR-v1504: "Savable" entry changed from "no" to "yes, category: application". |

### Description

If the actual position deviates so much from the demand position that the value of this object is exceeded, bit 13 in object $6041_h$ is set. The deviation must last longer than the time in object $6066_h$.

If the value of the "Following Error Window" is set to "FFFFFFFF"$_h$, following error monitoring is switched off.

A reaction to the following error can be set in object $3700_h$. If a reaction is defined, an error is also entered in object $1003_h$.

## 6066h Following Error Time Out

### Function

Time in milliseconds until a larger following error results in an error message.

### Object description

| | |
|---|---|
| Index | $6066_h$ |
| Object name | Following Error Time Out |
| Object Code | VARIABLE |
| Data type | UNSIGNED16 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $0064_h$ |
| Firmware version | FIR-v1426 |
| Change history | Firmware version FIR-v1504: "Savable" entry changed from "no" to "yes, category: application". |

### Description

If the actual position deviates so much from the demand position that the value of object $6065_h$ is exceeded, bit 13 in object $6041_h$ is set. The deviation must persist for longer than the time defined in this object.

A reaction to the following error can be set in object $3700_h$. If a reaction is defined, an error is also entered in object $1003_h$.

## 6067h Position Window

### Function

Specifies a range symmetrical to the target position within which that target is considered having been met in modes Profile Position and Interpolated Position Mode.

### Object description

| | |
|---|---|
| Index | $6067_h$ |
| Object name | Position Window |
| Object Code | VARIABLE |
| Data type | UNSIGNED32 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $0000000A_h$ |
| Firmware version | FIR-v1426 |
| Change history | Firmware version FIR-v1504: "Savable" entry changed from "no" to "yes, category: application". |

### Description

If the current position deviates from the target position by less than the value of this object, bit 10 in object $6041_h$ is set. The condition must be satisfied for longer than the time defined in object $6068_h$.

If the value is set to "FFFFFFFF"$_h$, monitoring is switched off.

## 6068h Position Window Time

### Function

The current position must be within the "Position Window" ($6067_h$) for this time in milliseconds for the target position to be considered having been met in the Profile Position and Interpolated Position Mode modes.

### Object description

| | |
|---|---|
| Index | $6068_h$ |
| Object name | Position Window Time |
| Object Code | VARIABLE |
| Data type | UNSIGNED16 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $0064_h$ |
| Firmware version | FIR-v1426 |
| Change history | Firmware version FIR-v1504: "Savable" entry changed from "no" to "yes, category: application". |

### Description

If the current position deviates from the target position by less than the value of object $6067_h$, bit 10 in object $6041_h$ is set. The condition must be satisfied for longer than the time defined in object $6068_h$.

## 606Bh Velocity Demand Value

### Function

Speed specification in user-defined units for the velocity controller.

### Object description

| | |
|---|---|
| Index | $606B_h$ |
| Object name | Velocity Demand Value |
| Object Code | VARIABLE |
| Data type | INTEGER32 |
| Savable | no |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |
| Firmware version | FIR-v1426 |
| Change history | |

### Description

This object contains the output of the ramp generator, which simultaneously serves as the preset value for the velocity controller.

## 606Ch Velocity Actual Value

### Function

Current actual speed in user-defined units.

### Object description

| | |
|---|---|
| Index | $606C_h$ |
| Object name | Velocity Actual Value |
| Object Code | VARIABLE |
| Data type | INTEGER32 |
| Savable | no |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |
| Firmware version | FIR-v1426 |
| Change history | |

## 606Dh Velocity Window

### Function

Specifies a symmetrical range relative to the target speed within which the target is considered having been met in the Profile Velocity mode.

### Object description

| | |
|---|---|
| Index | $606D_h$ |
| Object name | Velocity Window |
| Object Code | VARIABLE |
| Data type | UNSIGNED16 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $001E_h$ |
| Firmware version | FIR-v1426 |
| Change history | Firmware version FIR-v1614: "Savable" entry changed from "no" to "yes, category: application". |

### Description

If the current speed deviates from the set speed by less than the value of this object, bit 10 in object $6041_h$ is set. The condition must be satisfied for longer than the time defined in object $606E_h$ (see also statusword in Profile Velocity Mode).

## 606Eh Velocity Window Time

### Function

The current speed must be within the "Velocity Window" ($606D_h$) for this time (in milliseconds) for the target to be considered having been met.

### Object description

| | |
|---|---|
| Index | $606E_h$ |
| 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 | $0000_h$ |
| 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 $606D_h$, bit 10 in object $6041_h$ 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 | $606F_h$ |
| Object name | Velocity Threshold |
| Object Code | VARIABLE |
| Data type | UNSIGNED16 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $0000_h$ |
| Firmware version | FIR-v2013-B726332 |
| Change history | |

### Description

If the actual speed is greater than the value in $606F_h$(Velocity Threshold) for a time of $6070_h$(Velocity Threshold Time), bit 12 in $6041_h$(Statusword) has the value "0". The bit otherwise remains set to "1".

## 6070h Velocity Threshold Time

### Function

Time in milliseconds above which an actual speed greater than the value in $606F_h$ in Profile Velocity mode is considered to be nonzero.

### Object description

| | |
|---|---|
| Index | $6070_h$ |
| Object name | Velocity Threshold Time |
| Object Code | VARIABLE |
| Data type | UNSIGNED16 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $0000_h$ |

| | |
|---|---|
| Firmware version | FIR-v2013-B726332 |
| Change history | |

### Description

If the actual speed is greater than the value in $606F_h$(Velocity Threshold) for a time of $6070_h$(Velocity Threshold Time), bit 12 in $6041_h$(Statusword) has the value "0". The bit otherwise remains set to "1".

## 6071h Target Torque

### Function

This object contains the target torque for the Profile Torque and Cyclic Synchronous Torque modes in tenths of a percent of the rated torque.

### Object description

| | |
|---|---|
| Index | $6071_h$ |
| Object name | Target Torque |
| Object Code | VARIABLE |
| Data type | INTEGER16 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $000A_h$ |
| Firmware version | FIR-v1426 |
| Change history | Firmware version FIR-v1626: "Savable" entry changed from "no" to "yes, category: application". |

### Description

This object is calculated as thousandths of the torque, e.g., the value "500" means "50%" of the rated torque; "1100" is equivalent to 110%. The rated torque corresponds to the rated current in object $6075_h$.

The minimum of $6073_h$ and $6072_h$ is used as limit for the torque in $6071_h$.

## 6072h Max Torque

### Function

The object describes the maximum torque for the Profile Torque and Cyclic Synchronous Torque modes in tenths of a percent of the rated torque.

### Object description

| | |
|---|---|
| Index | $6072_h$ |
| Object name | Max Torque |
| Object Code | VARIABLE |
| Data type | UNSIGNED16 |
| Savable | yes, category: application |

| | |
|---|---|
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $0064_h$ |
| Firmware version | FIR-v1426 |
| Change history | |

### Description

This object is calculated as thousandths of the torque, e.g., the value "500" means "50%" of the rated torque; "1100" is equivalent to 110%. The rated torque corresponds to the rated current in object $6075_h$.

The minimum of $6073_h$ and $6072_h$ is used as limit for the torque in $6071_h$.

## 6073h Max Current

### Function

Contains the maximum current in tenths of a percent of the set rated current. . See also I2t Motor overload protection.

**NOTICE**

For stepper motors, only the rated current is specified, not a maximum current. Therefore, the value of $6073_h$ should generally not exceed the value 1000 (100%).

### Object description

| | |
|---|---|
| Index | $6073_h$ |
| Object name | Max Current |
| Object Code | VARIABLE |
| Data type | UNSIGNED16 |
| Savable | yes, category: motor |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $03E8_h$ |
| Firmware version | FIR-v1825-B577172 |
| Change history | Firmware version FIR-v2451-B1068465: "Savable" entry changed from "yes, category: movement" to "yes, category: motor". |

### Description

The maximum current is calculated in tenths of a percent of the rated current as follows:

$(6073_h*6075_h)/1000$

The unit is mA DC, equivalent to the current for a block commutation, as is typically specified in the motor data sheet.

The maximum current determines:

- the maximum current for the I2t Motor overload protection

- the rated current in *open-loop* mode.

## 6074h Torque Demand

### Function

Current torque set value requested by the ramp generator in tenths of a percent of the rated torque for the internal controller.

### Object description

| | |
|---|---|
| Index | $6074_h$ |
| Object name | Torque Demand |
| Object Code | VARIABLE |
| Data type | INTEGER16 |
| Savable | no |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $0000_h$ |
| Firmware version | FIR-v1426 |
| Change history | |

### Description

This object is calculated as thousandths of the torque, e.g., the value "500" means "50%" of the rated torque; "1100" is equivalent to 110%. The rated torque corresponds to the rated current in object $6075_h$.

The minimum of $6073_h$ and $6072_h$ is used as limit for the torque in $6071_h$.

## 6075h Motor Rated Current

### Function

Contains the rated current in mA DC, equivalent to the current for a block commutation, as is typically specified in the motor data sheet.

### Object description

| | |
|---|---|
| Index | $6075_h$ |
| Object name | Motor Rated Current |
| Object Code | VARIABLE |
| Data type | UNSIGNED32 |
| Savable | yes, category: motor |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000708_h$ |
| Firmware version | FIR-v1738-B501312 |
| Change history | Firmware version FIR-v2451-B1068465: "Savable" entry changed from "yes, category: tuning" to "yes, category: motor". |

## 6077h Torque Actual Value

### Function

This object indicates the current torque value in tenths of a percent of the rated torque for the internal controller.

### Object description

| | |
|---|---|
| Index | $6077_h$ |
| Object name | Torque Actual Value |
| Object Code | VARIABLE |
| Data type | INTEGER16 |
| Savable | no |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $0000_h$ |
| Firmware version | FIR-v1540 |
| Change history | |

### Description

This object is calculated as thousandths of the torque, e.g., the value "500" means "50%" of the rated torque; "1100" is equivalent to 110%. The rated torque corresponds to the rated current in object $6075_h$.

The minimum of $6073_h$ and $6072_h$ is used as limit for the torque in $6071_h$.

## 607Ah Target Position

### Function

This object specifies the target position in user-defined units for the Profile Position and Cyclic Synchronous Position modes.

### Object description

| | |
|---|---|
| Index | $607A_h$ |
| Object name | Target Position |
| Object Code | VARIABLE |
| Data type | INTEGER32 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00000E10_h$ |
| Firmware version | FIR-v1426 |
| Change history | Firmware version FIR-v1626: "Savable" entry changed from "no" to "yes, category: application". |

## 607Bh Position Range Limit

### Function

Contains the minimum and maximum position in user-defined units.

### Object description

| | |
|---|---|
| Index | $607B_h$ |
| Object name | Position Range Limit |
| Object Code | ARRAY |
| Data type | INTEGER32 |
| Savable | yes, category: application |
| Firmware version | FIR-v1426 |
| Change history | |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $02_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Min Position Range Limit |
| Data type | INTEGER32 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Max Position Range Limit |
| Data type | INTEGER32 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |

### Description

If this range is exceeded or not reached, an overflow occurs. To prevent this overflow, limit values for the target position can be set in object $607D_h$ ("Software Position Limit").

## 607Ch Home Offset

### Function

Specifies the difference between the zero position of the controller and the reference point of the machine in user-defined units.

### Object description

| | |
|---|---|
| Index | $607C_h$ |
| Object name | Home Offset |
| Object Code | VARIABLE |
| Data type | INTEGER32 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |
| Firmware version | FIR-v1426 |
| Change history | |

## 607Dh Software Position Limit

### Function

Defines the limit positions relative to the reference point of the application in user-defined units.

### Object description

| | |
|---|---|
| Index | $607D_h$ |
| Object name | Software Position Limit |
| Object Code | ARRAY |
| Data type | INTEGER32 |
| Savable | yes, category: application |
| Firmware version | FIR-v1426 |
| Change history | |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $02_h$ |

| | |
|---|---|
| Subindex | $01_h$ |

| | |
|---|---|
| Name | Min Position Limit |
| Data type | INTEGER32 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Max Position Limit |
| Data type | INTEGER32 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |

### Description

The absolute target position must lie within the limits set here. The Home Offset ($607C_h$) is not taken into account.

## 607Eh Polarity

### Function

With this object, the direction of rotation can be reversed.

### Object description

| | |
|---|---|
| Index | $607E_h$ |
| Object name | Polarity |
| Object Code | VARIABLE |
| Data type | UNSIGNED8 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00_h$ |
| Firmware version | FIR-v1426 |
| Change history | Firmware version FIR-v1738-B501312: "PDO mapping" table entry for subindex 00 changed from "no" to "RX-PDO". |

### Description

The following generally applies for direction reversal: If a bit is set to the value "1", reversal is activated. If the value is "0", the direction of rotation is as described in the respective mode.

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| POS | VEL | | | | | | GLB |

**GLB (Global)**

Direction of rotation reversal in all modes. It is not necessary here to either perform the auto setup again or to adjust the target values. The actual values are adjusted automatically. The change is not taken over if the *state machine* is already in the *Operation Enabled* state.

**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

## 607Fh Max Profile Velocity

### Function

Specifies the maximum speed in user-defined units for which the Mod i Profile Torque, Profile Position and Profile Velocity.

### Object description

| | |
|---|---|
| Index | $607F_h$ |
| Object name | Max Profile Velocity |
| Object Code | VARIABLE |
| Data type | UNSIGNED32 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00007530_h$ |
| Firmware version | FIR-v1540 |
| Change history | Firmware version FIR-v1738-B501312: "Object Name" entry changed from "Max profile velocity" to "Max Profile Velocity".<br>Firmware version FIR-v1738-B501312: "Data type" entry changed from "INTEGER16" to "UNSIGNED32".<br>Firmware version FIR-v1738-B501312: "Savable" entry changed from "no" to "yes, category: application".<br>Firmware version FIR-v1738-B501312: "Access" table entry for subindex 00 changed from "read only" to "read/write".<br>Firmware version FIR-v1738-B501312: "PDO mapping" table entry for subindex 00 changed from "TX-PDO" to "RX-PDO". |

## 6080h Max Motor Speed

### Function

Contains the maximum permissible speed of the motor in user-defined units.

### Object description

| | |
|---|---|
| Index | $6080_h$ |
| Object name | Max Motor Speed |
| Object Code | VARIABLE |
| Data type | UNSIGNED32 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00007530_h$ |
| Firmware version | FIR-v1426 |
| Change history | Firmware version FIR-v1614: "Savable" entry changed from "yes, category: application" to "yes, category: tuning".<br>Firmware version FIR-v1738-B501312: "Object Name" entry changed from "Maximum Speed" to "Max Motor Speed".<br>Firmware version FIR-v1738-B501312: "PDO mapping" table entry for subindex 00 changed from "no" to "RX-PDO".<br>Firmware version FIR-v1748-B538662: "Savable" entry changed from "yes, category: tuning" to "yes, category: movement".<br>Firmware version FIR-v1825-B577172: "Savable" entry changed from "yes, category: movement" to "yes, category: tuning".<br>Firmware version FIR-v1825-B577172: "Savable" entry changed from "yes, category: tuning" to "yes, category: movement".<br>Firmware version FIR-v2451-B1068465: "Savable" entry changed from "yes, category: movement" to "yes, category: application". |

## 6081h Profile Velocity

### Function

Specifies the maximum travel speed in user-defined units.

### Object description

| | |
|---|---|
| Index | $6081_h$ |
| Object name | Profile Velocity |
| Object Code | VARIABLE |
| Data type | UNSIGNED32 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | RX-PDO |

| | |
|---|---|
| Allowed values | |
| Preset value | $000001F4_h$ |
| Firmware version | FIR-v1426 |
| Change history | |

## 6082h End Velocity

### Function

Specifies the speed at the end of the traveled ramp in user-defined units.

### Object description

| | |
|---|---|
| Index | $6082_h$ |
| Object name | End Velocity |
| Object Code | VARIABLE |
| Data type | UNSIGNED32 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |
| Firmware version | FIR-v1426 |
| Change history | |

## 6083h Profile Acceleration

### Function

Specifies the maximum acceleration in user-defined units.

### Object description

| | |
|---|---|
| Index | $6083_h$ |
| Object name | Profile Acceleration |
| Object Code | VARIABLE |
| Data type | UNSIGNED32 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $000001F4_h$ |
| Firmware version | FIR-v1426 |
| Change history | |

## 6084h Profile Deceleration

### Function

Specifies the maximum deceleration (deceleration ramp) in user-defined units. Is limited by $60C6_h$.

### Object description

| | |
|---|---|
| Index | $6084_h$ |
| Object name | Profile Deceleration |
| Object Code | VARIABLE |
| Data type | UNSIGNED32 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $000001F4_h$ |
| Firmware version | FIR-v1426 |
| Change history | |

## 6085h Quick Stop Deceleration

### Function

Specifies the maximum Quick Stop Deceleration in user-defined units. Depending on the operating mode, is limited by $60C6_h$ (Max Deceleration) and, if applicable, $60A4_h$ (Profile Jerk).

### Object description

| | |
|---|---|
| Index | $6085_h$ |
| Object name | Quick Stop Deceleration |
| Object Code | VARIABLE |
| Data type | UNSIGNED32 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00001388_h$ |
| Firmware version | FIR-v1426 |
| Change history | |

## 6086h Motion Profile Type

### Function

Specifies the ramp type for the Profile Position and Profile Velocity modes.

### Object description

| | |
|---|---|
| Index | $6086_h$ |

| | |
|---|---|
| Object name | Motion Profile Type |
| Object Code | VARIABLE |
| Data type | INTEGER16 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $0000_h$ |
| Firmware version | FIR-v1426 |
| Change history | |

### Description

Value = "0": = Trapezoidal ramp

Value = "3": Ramp with limited jerk

## 6087h Torque Slope

### Function

This object contains the slope of the torque in Torque mode.

### Object description

| | |
|---|---|
| Index | $6087_h$ |
| Object name | Torque Slope |
| Object Code | VARIABLE |
| Data type | UNSIGNED32 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $000003E8_h$ |
| Firmware version | FIR-v1426 |
| Change history | |

### Description

This object is calculated as thousandths of the torque, e.g., the value "500" means "50%" of the rated torque; "1100" is equivalent to 110%. The rated torque corresponds to the rated current in object $6075_h$.

The minimum of $6073_h$ and $6072_h$ is used as limit for the torque in $6071_h$.

## 6091h Gear Ratio

### Function

Contains the gear ratio (number of motor revolutions per revolution of the output shaft) of the encoder/sensor that is used for position control (see 3203h Feedback Selection).

## Object description

| | |
|---|---|
| Index | $6091_h$ |
| Object name | Gear Ratio |
| Object Code | ARRAY |
| Data type | UNSIGNED32 |
| Savable | yes, category: application |
| Firmware version | FIR-v1426 |
| Change history | Firmware version FIR-v1738-B501312: "PDO mapping" table entry for subindex 01 changed from "no" to "RX-PDO".<br>Firmware version FIR-v1738-B501312: "PDO mapping" table entry for subindex 02 changed from "no" to "RX-PDO". |

## Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $02_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Motor Revolutions |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00000001_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Shaft Revolutions |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00000001_h$ |

## Description

Gear Ratio = Motor Revolutions ($6091_h$:$01_h$) / Shaft Revolutions ($6091_h$:$02_h$)

## 6092h Feed Constant

### Function

Contains the feed constant (feed in user-defined units per revolution of the output shaft) of the encoder/ sensor that is used for position control (see 3203h Feedback Selection).

### Object description

| | |
|---|---|
| Index | $6092_h$ |
| Object name | Feed Constant |
| Object Code | ARRAY |
| Data type | UNSIGNED32 |
| Savable | yes, category: application |
| Firmware version | FIR-v1426 |
| Change history | |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $02_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Feed |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00000001_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Shaft Revolutions |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00000001_h$ |

### Description

Feed Constant = Feed ($6092_h$:$01_h$) / Shaft Revolutions ($6092_h$:$02_h$)

## 6096h Velocity Factor

### Function

This object contains the factor that is used for converting from user-defined speed units. See chapter User-defined units.

### Object description

| | |
|---|---|
| Index | $6096_h$ |
| Object name | Velocity Factor |
| Object Code | ARRAY |
| Data type | UNSIGNED32 |
| Savable | yes, category: application |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1738-B501312 |
| Change history | |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $02_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Numerator |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00000001_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Divisor |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00000001_h$ |

### Description

The subindices have the following functions:

- $01_h$: Numerator of the factor
- $02_h$: Denominator of the factor

## 6097h Acceleration Factor

### Function

This object contains the factor that is used for converting from user-defined acceleration units. See chapter User-defined units.

### Object description

| | |
|---|---|
| Index | $6097_h$ |
| Object name | Acceleration Factor |
| Object Code | ARRAY |
| Data type | UNSIGNED32 |
| Savable | yes, category: application |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1738-B501312 |
| Change history | |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $02_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Numerator |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00000001_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Divisor |

| | |
|---|---|
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00000001_h$ |

### Description

The subindices have the following functions:

- $01_h$: Numerator of the factor
- $02_h$: Denominator of the factor

## 6098h Homing Method

### Function

This object defines the Homing method in Homing mode.

### Object description

| | |
|---|---|
| Index | $6098_h$ |
| Object name | Homing Method |
| Object Code | VARIABLE |
| Data type | INTEGER8 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $25_h$ |
| Firmware version | FIR-v1426 |
| Change history | |

## 6099h Homing Speed

### Function

Specifies the speeds for homing mode ($6098_h$) in user-defined units.

### Object description

| | |
|---|---|
| Index | $6099_h$ |
| Object name | Homing Speed |
| Object Code | ARRAY |
| Data type | UNSIGNED32 |
| Savable | yes, category: application |
| Firmware version | FIR-v1426 |
| Change history | |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $02_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Speed During Search For Switch |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00000032_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Speed During Search For Zero |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $0000000A_h$ |

### Description

The speed for the search for the switch is specified in subindex 1.

The (lower) speed for the search for the reference position is specified in subindex 2.

**NOTICE**

- The speed in subindex 2 is simultaneously the initial speed when starting the acceleration ramp. If this is set too high, the motor loses steps or fails to turn at all.
  If the setting is too high, the index marking will be overlooked, especially with high-resolution encoders. The minimum detectable width of the index pulse is 31.25 µs.
  .
- The speed in subindex 1 must be greater than the speed in subindex 2.

## 609Ah Homing Acceleration

### Function

Specifies the acceleration ramp for homing mode in user-defined units.

### Object description

| | |
|---|---|
| Index | $609A_h$ |
| Object name | Homing Acceleration |
| Object Code | VARIABLE |
| Data type | UNSIGNED32 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $000001F4_h$ |
| Firmware version | FIR-v1426 |
| Change history | |

### Description

The ramp is only used when starting up. When the switch is reached, the motor immediately switches to the lower speed; when the end position is reached, it immediately stops.

## 60A2h Jerk Factor

### Function

This object contains the factor that is used for converting from user-defined jerk units. See chapter User-defined units.

### Object description

| | |
|---|---|
| Index | $60A2_h$ |
| Object name | Jerk Factor |
| Object Code | ARRAY |
| Data type | UNSIGNED32 |
| Savable | yes, category: application |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1738-B501312 |
| Change history | |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |

| | |
|---|---|
| Preset value | $02_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Numerator |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00000001_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Divisor |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00000001_h$ |

### Description

The subindices have the following functions:

- $01_h$: Numerator of the factor
- $02_h$: Denominator of the factor

## 60A4h Profile Jerk

### Function

In the case of a ramp with limited jerk, the size of the jerk in user-defined units can be entered in this object. An entry with the value "0" means that the jerk is not limited.

### Object description

| | |
|---|---|
| Index | $60A4_h$ |
| Object name | Profile Jerk |
| Object Code | ARRAY |
| Data type | UNSIGNED32 |
| Savable | yes, category: application |
| Firmware version | FIR-v1426 |
| Change history | Firmware version FIR-v1614: "Name" entry changed from "End Acceleration Jerk" to "Begin Deceleration Jerk".<br>Firmware version FIR-v1614: "Name" entry changed from "Begin Deceleration Jerk" to "End Acceleration Jerk". |

## Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $04_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Begin Acceleration Jerk |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $000003E8_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Begin Deceleration Jerk |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $000003E8_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | End Acceleration Jerk |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $000003E8_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | End Deceleration Jerk |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $000003E8_h$ |

### Description

- Subindex $01_h$ ( *Begin Acceleration Jerk*): Initial jerk during acceleration
- Subindex $02_h$ ( *Begin Deceleration Jerk*): Initial jerk during braking
- Subindex $03_h$ ( *End Acceleration Jerk*): Final jerk during acceleration
- Subindex $04_h$ ( *End Deceleration Jerk*): Final jerk during braking

## 60A8h SI Unit Position

### Function

This object contains the position unit. See chapter User-defined units.

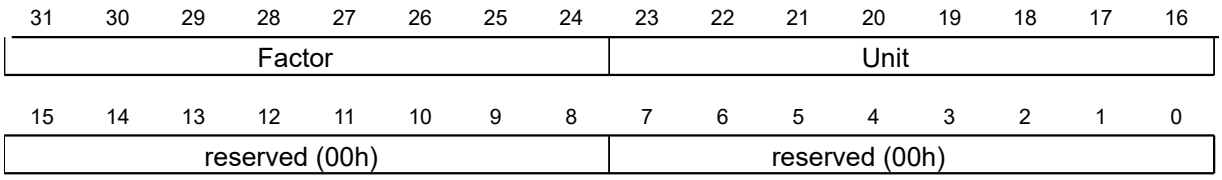
### Object description

| | |
|---|---|
| Index | $60A8_h$ |
| Object name | SI Unit Position |
| Object Code | VARIABLE |
| Data type | UNSIGNED32 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $FF410000_h$ |
| Firmware version | FIR-v1738-B501312 |
| Change history | |

### Description

Object $60A8_h$ contains:

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




## 60A9h SI Unit Velocity

### Function

This object contains the speed unit. See chapter User-defined units.

### Object description

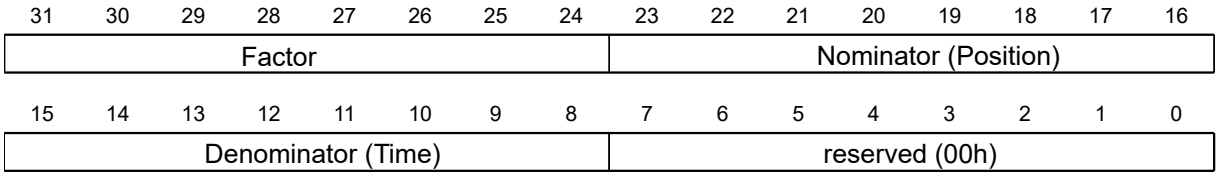
| | |
|---|---|
| Index | $60A9_h$ |
| Object name | SI Unit Velocity |
| Object Code | VARIABLE |
| Data type | UNSIGNED32 |
| Savable | yes, category: application |

| | |
|---|---|
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00B44700_h$ |
| Firmware version | FIR-v1738-B501312 |
| Change history | |

### Description

Object $60A9_h$ contains:

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




## 60B0h Position Offset

### Function

Offset for the position set value in user-defined units. Is taken into account in mode Cyclic Synchronous Position.

### Object description

| | |
|---|---|
| Index | $60B0_h$ |
| Object name | Position Offset |
| Object Code | VARIABLE |
| Data type | INTEGER32 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |
| Firmware version | FIR-v1738-B505321 |
| Change history | |

## 60B1h Velocity Offset

### Function

Offset for the speed set value in user-defined units. Is taken into account in the Cyclic Synchronous Position, Cyclic Synchronous Velocity and Clock-direction mode modes.

### Object description

| | |
|---|---|
| Index | $60B1_h$ |
| Object name | Velocity Offset |
| Object Code | VARIABLE |
| Data type | INTEGER32 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |
| Firmware version | FIR-v1738-B505321 |
| Change history | |

## 60B2h Torque Offset

### Function

Offset for the torque set value in tenths of a percent. Is taken into account in the Cyclic Synchronous Position, Cyclic Synchronous Velocity, Cyclic Synchronous Torque and Clock-direction mode modes.

### Object description

| | |
|---|---|
| Index | $60B2_h$ |
| Object name | Torque Offset |
| Object Code | VARIABLE |
| Data type | INTEGER16 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $0000_h$ |
| Firmware version | FIR-v1738-B505321 |
| Change history | |

## 60C1h Interpolation Data Record

### Function

This object contains the demand position in user-defined units for the interpolation algorithm for the interpolated position operating mode.

### Object description

| | |
|---|---|
| Index | $60C1_h$ |
| Object name | Interpolation Data Record |
| Object Code | ARRAY |
| Data type | INTEGER32 |
| Savable | yes, category: application |

| | |
|---|---|
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1512 |
| Change history | Firmware version FIR-v1626: "Savable" entry changed from "no" to "yes, category: application". |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $01_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | 1st Set-point |
| Data type | INTEGER32 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |

### Description

The value is taken over at the next synchronization time.

## 60C2h Interpolation Time Period

### Function

This object contains the interpolation time.

### Object description

| | |
|---|---|
| Index | $60C2_h$ |
| Object name | Interpolation Time Period |
| Object Code | RECORD |
| Data type | INTERPOLATION_TIME_PERIOD |
| Savable | yes, category: application |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |

| | |
|---|---|
| Firmware version | FIR-v1426 |
| Change history | |

## Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $02_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Interpolation Time Period Value |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $01_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Interpolation Time Index |
| Data type | INTEGER8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $FD_h$ |

## Description

The subindices have the following functions:

- $01_h$: Interpolation time.
- $02_h$: Power of ten of the interpolation time: -3 corresponds to the time basis in milliseconds

The following applies here: cycle time = value of $\underline{60C2_h}:01_h$ * 10 $^{\text{value of 60C2:02}}$ seconds.

# 60C4h Interpolation Data Configuration

## Function

This object offers the maximum buffer size, specifies the configured buffer organization of the interpolated data and offers objects for defining the size of the record and for deleting the buffer.

It is also used to store the position of other data points.

## Object description

| | |
|---|---|
| Index | $60C4_h$ |
| Object name | Interpolation Data Configuration |
| Object Code | RECORD |
| Data type | INTERPOLATION_DATA_CONFIGURATION |
| Savable | yes, category: application |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1512 |
| Change history | Firmware version FIR-v1540: "Access" table entry for subindex 05 changed from "read/write" to "write only".<br>Firmware version FIR-v1540: "Access" table entry for subindex 06 changed from "read/write" to "write only".<br>Firmware version FIR-v1626: "Savable" entry changed from "no" to "yes, category: application".<br>Firmware version FIR-v1650-B472161: "Access" table entry for subindex 01 changed from "read/write" to "read only".<br>Firmware version FIR-v2412-B1057638: "Name" entry changed from "Buffer Organization" to "Buffer Organisation". |

## Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $06_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Maximum Buffer Size |
| Data type | UNSIGNED32 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000001_h$ |

| | |
|---|---|
| Subindex | $02_h$ |
| Name | Actual Buffer Size |
| Data type | UNSIGNED32 |
| Access | read / write |

| | |
|---|---|
| PDO mapping | no |
| Allowed values | |
| Preset value | $00000000_h$ |

| | |
|---|---|
| Subindex | $03_h$ |
| Name | Buffer Organisation |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00_h$ |

| | |
|---|---|
| Subindex | $04_h$ |
| Name | Buffer Position |
| Data type | UNSIGNED16 |
| Access | read / write |
| PDO mapping | no |
| Allowed values | |
| Preset value | $0000_h$ |

| | |
|---|---|
| Subindex | $05_h$ |
| Name | Size Of Data Record |
| Data type | UNSIGNED8 |
| Access | write only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $01_h$ |

| | |
|---|---|
| Subindex | $06_h$ |
| Name | Buffer Clear |
| Data type | UNSIGNED8 |
| Access | write only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $00_h$ |

## Description

The value of subindex $01_h$ contains the maximum possible number of interpolated records.

The value of subindex $02_h$ contains the current number of interpolated records.

If subindex $03_h$ is "$00_h$", this means a FIFO buffer organization; if it is "$01_h$", it specifies a ring buffer organization.

The value of subindex $04_h$ is unitless and specifies the next free buffer entry point.

The value of subindex $05_h$ is specified in units of "byte".

If the value "$00_h$" is written in subindex $06_h$, it deletes the received data in the buffer, deactivates access and deletes all interpolated records.

If the value "$01_h$" is written in subindex $06_h$, it activates access to the input buffer.

## 60C5h Max Acceleration

### Function

This object contains the maximum permissible acceleration for the Profile Position and Profile Velocity modes.

### Object description

| | |
|---|---|
| Index | $60C5_h$ |
| Object name | Max Acceleration |
| Object Code | VARIABLE |
| Data type | UNSIGNED32 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00001388_h$ |
| Firmware version | FIR-v1426 |
| Change history | |

## 60C6h Max Deceleration

### Function

This object contains the maximum permissible deceleration (deceleration ramp) for the Profile Position, Profile Velocity and Interpolated Position Mode operating modes.

### Object description

| | |
|---|---|
| Index | $60C6_h$ |
| Object name | Max Deceleration |
| Object Code | VARIABLE |
| Data type | UNSIGNED32 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00001388_h$ |
| Firmware version | FIR-v1426 |
| Change history | |

## 60E4h Additional Position Actual Value

### Function

Contains the current actual position of all existing feedbacks in user-defined units.

## Object description

| | |
|---|---|
| Index | $60E4_h$ |
| Object name | Additional Position Actual Value |
| Object Code | ARRAY |
| Data type | INTEGER32 |
| Savable | no |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1738-B501312 |
| Change history | Firmware version FIR-v1748-B538662: "Data type" entry changed from "UNSIGNED32" to "INTEGER32".<br>Firmware version FIR-v1748-B538662: "Data type" entry changed from "UNSIGNED32" to "INTEGER32". |

## Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $04_h$ |

| | |
|---|---|
| Subindex | $01_h$ - $04_h$ |
| Name | Additional Position Actual Value #1 - #4 |
| Data type | INTEGER32 |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |

## Description

The subindices have the following function:

- $00_h$: Value="1" to "n", where "n" is the number of existing feedbacks.
- $n_h$:
  Subindex n contains the current actual position of the corresponding feedback.
  Subindex $01_h$ always corresponds to the first feedback *Sensorless* . 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 | $60E5_h$ |
| Object name | Additional Velocity Actual Value |
| Object Code | ARRAY |
| Data type | INTEGER32 |
| Savable | no |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1738-B501312 |
| Change history | |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $04_h$ |

| | |
|---|---|
| Subindex | $01_h$ - $04_h$ |
| Name | Additional Velocity Actual Value #1 - #4 |
| Data type | INTEGER32 |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |

### Description

The subindices have the following function:

- $00_h$: Value="1" to "n", where "n" is the number of existing feedbacks.
- $n_h$:
  Subindex n contains the current actual speed of the corresponding feedback.
  Subindex $01_h$ always corresponds to the first feedback *Sensorless* . 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$_h$, the resolution of each existing feedback is calculated.

### Object description

| | |
|---|---|
| Index | 60E6$_h$ |
| Object name | Additional Position Encoder Resolution - Encoder Increments |
| Object Code | ARRAY |
| Data type | UNSIGNED32 |
| Savable | yes, category: sensors |
| Access | read only |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1748-B538662 |
| Change history | Firmware version FIR-v2451-B1068465: "Data type" entry changed from "INTEGER32" to "UNSIGNED32".<br>Firmware version FIR-v2451-B1068465: "Savable" entry changed from "yes, category: tuning" to "yes, category: sensors".<br>Firmware version FIR-v2451-B1068465: "Data type" entry changed from "INTEGER32" to "UNSIGNED32". |

### Value description

| | |
|---|---|
| Subindex | 00$_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | 04$_h$ |

| | |
|---|---|
| Subindex | 01$_h$ - 04$_h$ |
| Name | Additional Position Encoder Resolution - Encoder Increments Feedback Interface #1 - #4 |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | 00000000$_h$ |

### Description

The subindices have the following function:

- $00_h$: Value="1" to "n", where "n" is the number of existing feedbacks.
- $n_h$:
  Subindex n contains the number of increments of the corresponding feedback.
  Subindex $01_h$ always corresponds to the first feedback *Sensorless* . 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 ($60E6_h$:$01_h$) / Motor Revolutions ($60EB_h$:$02_h$)

**NOTICE**

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

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

## 60EBh Additional Position Encoder Resolution - Motor Revolutions

### Function

With this object and with $60E6_h$, the resolution of each existing feedback is calculated.

### Object description

| | |
|---|---|
| Index | $60EB_h$ |
| Object name | Additional Position Encoder Resolution - Motor Revolutions |
| Object Code | ARRAY |
| Data type | UNSIGNED32 |
| Savable | yes, category: sensors |
| Access | read only |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v1738-B501312 |
| Change history | Firmware version FIR-v2451-B1068465: "Savable" entry changed from "yes, category: tuning" to "yes, category: sensors". |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $04_h$ |

| | |
|---|---|
| Subindex | $01_h$ - $04_h$ |
| 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 | $00000001_h$ |

### Description

The subindices have the following function:

- $00_h$: Value="1" to "n", where "n" is the number of existing feedbacks.
- $n_h$:
  Subindex n contains the number of motor revolutions of the corresponding feedback.
  Subindex $01_h$ always corresponds to the first feedback *Sensorless* . 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 ($60E6_h$:$n_h$) / Motor Revolutions ($60EB_h$:$n_h$)

## 60F1h Additional Encoder Inversion

### Function

The polarity of the sensors is stored here.

### Object description

| | |
|---|---|
| Index | $60F1_h$ |
| Object name | Additional Encoder Inversion |
| Object Code | ARRAY |
| Data type | UNSIGNED8 |
| Savable | yes, category: sensors |
| Access | read only |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | |
| Firmware version | FIR-v2451-B1068465 |
| Change history | |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | RX-PDO |
| Allowed values | |

| | |
|---|---|
| Preset value | $04_h$ |

| | |
|---|---|
| Subindex | $01_h$ - $04_h$ |
| Name | Additional Encoder Inversion - Feedback Interface #1 - #4 |
| Data type | UNSIGNED8 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00_h$ |

## 60F2h Position Option Code

### Function

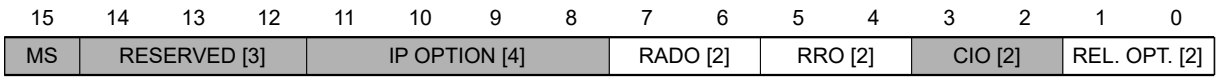
The object describes the positioning behavior in Profile Position mode.

### Object description

| | |
|---|---|
| Index | $60F2_h$ |
| Object name | Position Option Code |
| Object Code | VARIABLE |
| Data type | UNSIGNED16 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $0000_h$ |
| Firmware version | FIR-v1446 |
| Change history | Firmware version FIR-v1614: "Savable" entry changed from "no" to "yes, category: application".<br>Firmware version FIR-v2412-B1057638: "Object Name" entry changed from "Positioning Option Code" to "Position Option Code". |

### Description

Only the following bits are supported at the present time:




**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) |

| Bit 1 | Bit 0 | Definition |
|---|---|---|
| 0 | 1 | Position movements are executed relative to the preset value (or output) of the ramp generator. |
| 1 | 0 | Position movements are performed relative to the current position (object $6064_h$). |
| 1 | 1 | Reserved |

### RRO (Request-Response Option)

These bits determine the behavior when passing controlword $6040_h$ bit 4 ("new setpoint") – in this case, the controller releases the bit itself. This eliminates the need to externally reset the bit to "0" afterwards. After the bit is set to the value "0" by the controller, bit 12 ("setpoint acknowledgment") is also set to the value "0" in statusword $6041_h$.

**NOTICE**

These options cause the controller to modify object controlword $6040_h$.

| Bit 5 | Bit 4 | Definition |
|---|---|---|
| 0 | 0 | The functionality is as described under Setting travel commands. |
| 0 | 1 | The controller releases the "new setpoint" bit as soon as the current targeted movement has reached its target. |
| 1 | 0 | The controller releases the "new setpoint" bit as soon this is possible for the controller. |
| 1 | 1 | Reserved |

### RADO (Rotary Axis Direction Option)

These bits determine the direction of rotation in "profile position" mode.

| Bit 7 | Bit 6 | Definition |
|---|---|---|
| 0 | 0 | Normal positioning similar to a linear axis: If one of the "Position Range Limits" – $607B_h$:$01_h$ and $02_h$ – is reached or exceeded, the preset is automatically transferred to the other end of the limit. Only with this bit combination is a movement greater than the modulo value possible. |
| 0 | 1 | Positioning only in negative direction: If the target position is greater than the current position, the axis moves to the target position via the "Min Position Range Limit" from object $607D_h$:$01_h$. |
| 1 | 0 | Positioning only in positive direction: If the target position is less than the current position, the axis moves to the target position via the "Max Position Range Limit" from object $607D_h$:$01_h$. |
| 1 | 1 | Positioning with the shortest distance to the target position. If the difference between the current position and the target position in a 360° system is less than 180°, the axis moves in the positive direction. |

## 60F4h Following Error Actual Value

### Function

This object contains the current following error in user-defined units.

### Object description

| | |
|---|---|
| Index | $60F4_h$ |
| Object name | Following Error Actual Value |
| Object Code | VARIABLE |
| Data type | INTEGER32 |
| Savable | no |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |
| Firmware version | FIR-v1426 |
| Change history | |

## 60F8h Max Slippage

### Function

Defines the maximum allowed slippage error in user-defined units symmetrically to the set speed in Profile Velocity mode.

### Object description

| | |
|---|---|
| Index | $60F8_h$ |
| Object name | Max Slippage |
| Object Code | VARIABLE |
| Data type | INTEGER32 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00000190_h$ |
| Firmware version | FIR-v1738-B501312 |
| Change history | |

### Description

If the actual speed deviates so much from the set speed that the value (absolute value) of this object is exceeded, bit 13 in object $6041_h$ is set. The deviation must last longer than the time in object $203F_h$.

If the value of $60F8_h$ is set to "7FFFFFFF"$_h$ or "80000000"$_h$, slippage error monitoring is switched off.

A reaction to the slippage error can be set in object $3700_h$. If a reaction is defined, an error is also entered in object $1003_h$.

## 60FAh Control Effort

### Function

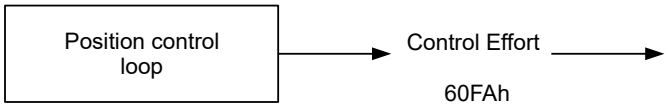
This object contains the correction speed (control variable) in user-defined units that is fed to the velocity controller by the position controller.

### Object description

| | |
|---|---|
| Index | $60FA_h$ |
| Object name | Control Effort |
| Object Code | VARIABLE |
| Data type | INTEGER32 |
| Savable | no |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |
| Firmware version | FIR-v1748-B531667 |
| Change history | |

### Description

The position controller calculates a correction speed (in user-defined units) from the difference between the current position and the demand position which is then passed on to the velocity controller. This correction value is dependent on the proportional component and integral component of the position controller. See also chapter Closed Loop.




## 60FCh Position Demand Internal Value

### Function

### Object description

| | |
|---|---|
| Index | $60FC_h$ |
| Object name | Position Demand Internal Value |
| Object Code | VARIABLE |
| Data type | INTEGER32 |
| Savable | no |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |

| | |
|---|---|
| Preset value | $00000000_h$ |
| Firmware version | FIR-v1738-B501312 |
| Change history | |

## 60FDh Digital Inputs

### Function

With this object, the digital inputs of the motor can be read.

### Object description

| | |
|---|---|
| Index | $60FD_h$ |
| Object name | Digital Inputs |
| Object Code | VARIABLE |
| Data type | UNSIGNED32 |
| Savable | no |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |
| Firmware version | FIR-v1426 |
| Change history | |

### Description

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| STO B | STO A | | | | | | ... | 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.

**STO A**
STO channel A

**STO B**
STO channel B

## 60FEh Digital Outputs

### Function

With this object, the digital outputs of the motor can be written.

### Object description

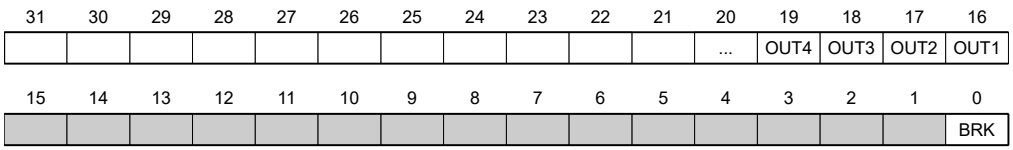
| | |
|---|---|
| Index | $60FE_h$ |
| Object name | Digital Outputs |
| Object Code | ARRAY |
| Data type | UNSIGNED32 |
| Savable | yes, category: application |
| Firmware version | FIR-v1426 |
| Change history | Firmware version FIR-v1626: "Savable" entry changed from "no" to "yes, category: application". |

### Value description

| | |
|---|---|
| Subindex | $00_h$ |
| Name | Number Of Entries |
| Data type | UNSIGNED8 |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | $01_h$ |

| | |
|---|---|
| Subindex | $01_h$ |
| Name | Physical Outputs |
| Data type | UNSIGNED32 |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |

### Description




**BRK (Brake)**

Bit for the brake output (if the controller supports this function):

Value "1" means that the brake is activated (no current can flow between the two pins of the brake connection; the brake is closed).

**OUT n (Output No n)**

Bit for the respective digital output; the exact number of digital outputs is dependent on the controller.

## 60FFh Target Velocity

### Function

In this object, the target speed for the profile velocity and cyclic synchronous velocity modes is entered in user-defined units.

### Object description

| | |
|---|---|
| Index | $60FF_h$ |
| Object name | Target Velocity |
| Object Code | VARIABLE |
| Data type | INTEGER32 |
| Savable | yes, category: application |
| Access | read / write |
| PDO mapping | RX-PDO |
| Allowed values | |
| Preset value | $00000000_h$ |
| Firmware version | FIR-v1426 |
| Change history | Firmware version FIR-v1626: "Savable" entry changed from "no" to "yes, category: application". |

## 6502h Supported Drive Modes

### Function

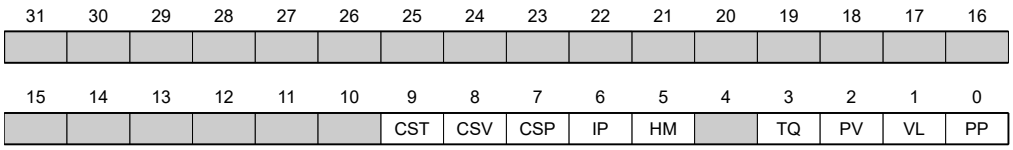
The object describes the supported operating modes in object $6060_h$.

### Object description

| | |
|---|---|
| Index | $6502_h$ |
| Object name | Supported Drive Modes |
| Object Code | VARIABLE |
| Data type | UNSIGNED32 |
| Savable | no |
| Access | read only |
| PDO mapping | TX-PDO |
| Allowed values | |
| Preset value | $000003ED_h$ |
| Firmware version | FIR-v1426 |
| Change history | |

### Description

The set bit specifies whether the respective mode is supported. If the value of the bit is "0", the mode is not supported.




**PP**
Profile Position Mode

**VL**
Velocity Mode

**PV**
Profile Velocity Mode

**TQ**
Torque Mode

**HM**
Homing Mode

**IP**
Interpolated Position Mode

**CSP**
Cyclic Synchronous Position Mode

**CSV**
Cyclic Synchronous Velocity Mode

**CST**
Cyclic Synchronous Torque Mode

## 6503h Drive Catalogue Number

### Function

Contains the device name as character string.

### Object description

| | |
|---|---|
| Index | $6503_h$ |
| Object name | Drive Catalogue Number |
| Object Code | VARIABLE |
| Data type | VISIBLE_STRING |
| Savable | no |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | 0 |
| Firmware version | FIR-v1426 |
| Change history | |

## 6505h Http Drive Catalogue Address

### Function

This object contains the manufacturer's web address as a character string.

### Object description

| | |
|---|---|
| Index | $6505_h$ |
| Object name | Http Drive Catalogue Address |
| Object Code | VARIABLE |
| Data type | VISIBLE_STRING |
| Savable | no |
| Access | read only |
| PDO mapping | no |
| Allowed values | |
| Preset value | 0 |
| Firmware version | FIR-v1426 |
| Change history | |

# 11 Copyrights

## 11.1 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.

## 11.2 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

## 11.3 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.

## 11.4 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.

## 11.5 CMSIS DSP Software Library

Copyright (C) 2010 ARM Limited. All rights reserved.

## 11.6 Protothreads

Protothread class and macros for lightweight, stackless threads in C++.

This was "ported" to C++ from Adam Dunkels' protothreads C library at: http://www.sics.se/~adam/pt/

Originally ported for use by Hamilton Jet (www.hamiltonjet.co.nz) by Ben Hoyt, but stripped down for public release. See his blog entry about it for more information: http://blog.micropledge.com/2008/07/protothreads/

Original BSD-style license

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.

## 11.7 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>

## 11.8 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.