

# Manual PD4-C/-CB

Fieldbus: USB

## For use with the following devices:

- PD4-C5918M4204-E-01
- PD4-C6018L4204-E-01
- PD4-CB59M024035-E-01

# Contents

<b>1 Editorial.....</b>	<b>7</b>
<b>2 Safety instructions and warnings.....</b>	<b>8</b>
2.1 Important information.....	8
2.2 Personnel qualifications.....	8
2.3 Danger and warning signs.....	8
2.4 Other information.....	9
<b>3 About this manual.....</b>	<b>10</b>
3.1 Introduction.....	10
3.2 Numerical values.....	10
3.3 Bits.....	10
3.4 Counting direction (arrows).....	10
3.5 Release notes.....	10
<b>4 Technical data and pin configuration.....</b>	<b>12</b>
4.1 Dimensioned drawings.....	12
4.2 Electrical properties.....	12
4.3 LED signaling.....	13
4.4 Pin configuration.....	14
<b>5 Configuration.....</b>	<b>17</b>
5.1 General information.....	17
5.2 DIP switches.....	17
5.3 USB port.....	18
5.4 Configuration file.....	18
5.5 VMM program.....	21
<b>6 Commissioning.....</b>	<b>22</b>
6.1 Safety instructions.....	22
6.2 Preparation.....	22
<b>7 Operating modes.....</b>	<b>24</b>
7.1 Profile Position.....	24
7.2 Velocity.....	32
7.3 Profile velocity.....	33
7.4 Profile torque.....	36
7.5 Homing.....	37
<b>8 General concepts.....</b>	<b>45</b>
8.1 DS402 Power State machine.....	45
8.2 User-defined units.....	49
<b>9 Special functions.....</b>	<b>52</b>

9.1 Digital inputs and outputs.....	52
9.2 I <sup>2</sup> t motor overload protection.....	54
<b>10 Programming with NanoJ.....</b>	<b>56</b>
10.1 Introduction.....	56
10.2 Available computing time.....	56
10.3 Interaction of the user program with the motor controller.....	56
10.4 OD entries for controlling and configuring the VMM.....	57
10.5 NanoJEasyV2.....	58
10.6 System calls.....	60
<b>11 Object directory description.....</b>	<b>63</b>
11.1 Overview.....	63
11.2 Structure of the object description.....	63
11.3 Object description.....	63
11.4 Value description.....	64
11.5 Description.....	65
1000h Device Type.....	66
1001h Error Register.....	67
1003h Pre-defined Error Field.....	67
1008h Manufacturer Device Name.....	71
1009h Manufacturer Hardware Version.....	71
100Ah Manufacturer Software Version.....	72
1010h Store Parameter.....	72
1011h Restore Default Parameter.....	74
1018h Identity Object.....	74
1600h Drive Control.....	76
1601h Positioning Control.....	78
1602h Velocity Control.....	81
1603h Output Control.....	83
1A00h Drive Status.....	86
1A01h Positioning Status.....	88
1A02h Velocity Status.....	90
1A03h Input Status.....	93
2010h IP-Configuration.....	95
2011h Static-IP-Address.....	96
2012h Static-IP-Subnet-Mask.....	97
2018h Current-IP-Address.....	98
2019h Current-IP-Subnet-Mask.....	99
2020h ApplInfo-Static-IP-Address.....	100
2021h ApplInfo-Static-IP-Subnet-Mask.....	101
2022h Drive Serial Number.....	102
2030h Pole Pair Count.....	103
2031h Peak Current.....	103
2032h Maximum Speed.....	103
2033h Plunger Block.....	104
2034h Upper Voltage Warning Level.....	105
2035h Lower Voltage Warning Level.....	105
2036h Open Loop Current Reduction Idle Time.....	106
2037h Open Loop Current Reduction Value/factor.....	106
2038h Brake Controller Timing.....	107
2039h Motor Currents.....	108
203Ah Homing On Block Configuration.....	110
<sup>2</sup> 203Bh I2t Parameters.....	111
2050h Encoder Alignment.....	113
2051h Encoder Optimization.....	114
2052h Encoder Resolution.....	115

2053h Index Polarity.....	115
2054h Index Width.....	116
2056h Limit Switch Tolerance Band.....	116
2057h Clock Direction Multiplier.....	117
2058h Clock Direction Divider.....	117
2059h Encoder Configuration.....	118
2060h Compensate Polepair Count.....	118
2061h Velocity Numerator.....	119
2062h Velocity Denominator.....	119
2063h Acceleration Numerator.....	120
2064h Acceleration Denominator.....	120
2065h Jerk Numerator.....	121
2066h Jerk Denominator.....	121
2084h Bootup Delay.....	122
2101h Fieldbus Module.....	122
2200h Sampler Control.....	123
2201h Sampler Status.....	124
2202h Sample Data Selection.....	124
2203h Sampler Buffer Information.....	127
2204h Sample Time In Ms.....	128
2300h VMM Control.....	128
2301h VMM Status.....	129
2302h VMM Error Code.....	130
2303h Number Of Active User Program.....	131
2304h Table Of Available User Programs.....	132
2310h VMM Input Data Selection.....	134
2320h VMM Output Data Selection.....	138
2330h VMM In/output Data Selection.....	142
2400h VMM Inputs.....	146
2500h VMM Outputs.....	153
2600h VMM Debug Output.....	160
2700h User Storage Area.....	173
2800h Bootloader Setting.....	178
3202h Motor Drive Submode Select.....	179
3208h Motor Drive Sensor Source Open Loop.....	180
3209h Motor Drive Sensor Source Closed Loop.....	181
320Ah Motor Drive Sensor Display Open Loop.....	183
320Bh Motor Drive Sensor Display Closed Loop.....	184
3210h Motor Drive Parameter Set.....	186
3220h Analog Inputs.....	189
3221h Analogue Inputs Control.....	190
3240h Digital Inputs Control.....	190
3250h Digital Outputs Control.....	192
3700h Following Error Option Code.....	194
603Fh Error Code.....	195
6040h Controlword.....	195
6041h Statusword.....	196
6042h VI Target Velocity.....	197
6043h VI Velocity Demand.....	198
6044h VI Velocity Actual Value.....	198
6046h VI Velocity Min Max Amount.....	199
6048h VI Velocity Acceleration.....	200
6049h VI Velocity Deceleration.....	201
604Ah VI Velocity Quick Stop.....	202
604Ch VI Dimension Factor.....	203
605Ah Quick Stop Option Code.....	204
605Bh Shutdown Option Code.....	204
605Ch Disable Option Code.....	205
605Dh Halt Option Code.....	206

605Eh Fault Option Code.....	206
6060h Modes Of Operation.....	207
6061h Modes Of Operation Display.....	208
6062h Position Demand Value.....	208
6063h Position Actual Internal Value.....	209
6064h Position Actual Value.....	209
6065h Following Error Window.....	210
6066h Following Error Time Out.....	210
6067h Position Window.....	211
6068h Position Window Time.....	211
606Bh Velocity Demand Value.....	212
606Ch Velocity Actual Value.....	212
606Dh Velocity Window.....	213
606Eh Velocity Window Time.....	213
6071h Target Torque.....	214
6072h Max Torque.....	214
6074h Torque Demand.....	214
607Ah Target Position.....	215
607Bh Position Range Limit.....	215
607Ch Home Offset.....	216
607Dh Software Position Limit.....	217
607Eh Polarity.....	218
6081h Profile Velocity.....	219
6082h End Velocity.....	219
6083h Profile Acceleration.....	219
6084h Profile Deceleration.....	220
6085h Quick Stop Deceleration.....	220
6086h Motion Profile Type.....	221
6087h Torque Slope.....	221
608Fh Position Encoder Resolution.....	222
6091h Gear Ratio.....	223
6092h Feed Constant.....	224
6098h Homing Method.....	225
6099h Homing Speed.....	225
609Ah Homing Acceleration.....	226
60A4h Profile Jerk.....	227
60C2h Interpolation Time Period.....	228
60C5h Max Acceleration.....	229
60C6h Max Deceleration.....	230
60F4h Following Error Actual Value.....	230
60FDh Digital Inputs.....	230
60FEh Digital Outputs.....	231
60FFh Target Velocity.....	232
6402h Read Analogue Input.....	233
6431h Analogue Input Offset.....	234
6432h Analogue Input Pre-scaling.....	235
6502h Supported Drive Modes.....	236
6505h Http Drive Catalogue Address.....	237

<b>12 Copyright notice.....</b>	<b>238</b>
12.1 Introduction.....	238
12.2 AES.....	238
12.3 Arcfour (RC4).....	238
12.4 MD5.....	239
12.5 uIP.....	239
12.6 DHCP.....	239
12.7 CMSIS DSP Software Library.....	240
12.8 FatFs.....	240

12.9 Protothreads..... 240

# 1 Editorial

Copyright © 2014 Nanotec Electronic GmbH & Co. KG. All rights reserved.

For copyright information on the software components integrated from external sources: See the "**Copyrights**" section.

Nanotec<sup>®</sup> Electronic GmbH & Co. KG

Kapellenstraße 6

85622 Feldkirchen/Munich, Germany

Tel.: +49 (0)89-900 686-0

Fax: +49 (0)89-900 686-50

Internet: [www.nanotec.com](http://www.nanotec.com)

All rights reserved!

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

**Original programming manual**

## 2 Safety instructions and warnings

### 2.1 Important information

This technical manual must be carefully read before installation and commissioning of the controller.

Nanotec® reserves the right to make technical alterations and further develop hardware and software in the interests of its customers to improve the function of this product without prior notice.

This manual was created with due care. It is exclusively intended as a technical description of the product and as commissioning instructions. The warranty is exclusively for repair or replacement of defective equipment, according to our general terms and conditions; liability for subsequent damage or errors is excluded. Applicable standards and regulations must be complied with during installation of the device.

For criticisms, proposals, and suggestions for improvement, please contact the above address or send an email to: [info@nanotec.com](mailto:info@nanotec.com)

### 2.2 Personnel qualifications

Work on and with this product may only be carried out by skilled workers


- who are familiar with and have understood the contents of this manual
- who have completed a training course or have the corresponding experience to be able to estimate, predict, or identify any dangers that may arise from using the motor controller
- who are familiar with all applicable standards, legal provisions, and accident-prevention regulations that have to be complied with when working on and with the product
- who are able to ensure personal safety when using the motor controller in an overall system


Operation may only be carried out when the specified cables and corresponding accessories are used. Use only original accessories and original spare parts.

### 2.3 Danger and warning signs

All signs listed in this documentation are printed in a standardized form. A hazardous situation is categorized according to the classes below depending on the level of hazard to the user or motor controller.


The DANGER sign indicates an immediately hazardous situation that, when the instruction is neglected, will <b>unavoidably</b> cause a serious or fatal accident.


The WARNING sign indicates a potentially hazardous situation that, when the instruction is neglected, may <b>possibly</b> cause a serious or fatal accident or damage to this device or other devices.


The CAUTION sign indicates a potentially hazardous situation that, when the instruction is neglected, may <b>possibly</b> cause an accident or damage to this device or other devices.



## CAUTION

The CAUTION sign without the warning symbol indicates a possibly hazardous situation that, when the instruction is neglected, may **possibly** cause an accident or damage to this device or other devices.

### 2.4 Other information

The following additional information panels are used in this documentation:

**Tip** This panel indicates a possibility for simplifying work.

#### Note

This panel indicates possible error sources or risks of confusion.

---

#### Example

This panel contains an example.

---

## 3 About this manual

### 3.1 Introduction

This manual is directed toward programmers intending to program a motor controller using the motor controller from nanotec<sup>®</sup>.

### 3.2 Numerical values

Numerical values are always presented in decimal notation. If hexadecimal notation must be used, this is indicated by a subscript "h" at the end of the number.

The objects in the object directory are noted as follows with an index and subindex:  
<Index>:<Subindex>

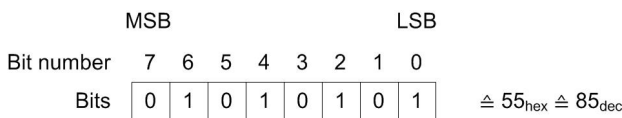
Both the index and subindex are in hexadecimal notation. Subindex 0 is in force when no subindex is noted.

Example: Subindex 5 of object 1003<sub>h</sub> is addressed with "1003<sub>h</sub>:5<sub>h</sub>", subindex 0 of object 6040<sub>h</sub> with "6040<sub>h</sub>".

In the last section of the manual, all objects are listed in full, and the references in the running text and tables are set in bold, e.g. **6040<sub>h</sub>**.

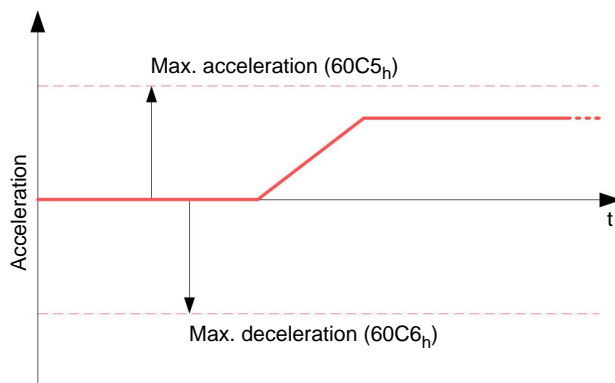
### 3.3 Bits

The individual bits of an object are always numbered beginning with 0 at the LSB. See the following figure, which uses the "UNSIGNED8" data type as an example.



### 3.4 Counting direction (arrows)

In drawings, the counting direction is always in the direction of the arrow. The objects 60C5<sub>h</sub> and 60C6<sub>h</sub> shown in the following figure are both positive.



### 3.5 Release notes

Version	Datum	Änderungen
1.0.0	03.03.2014	First release

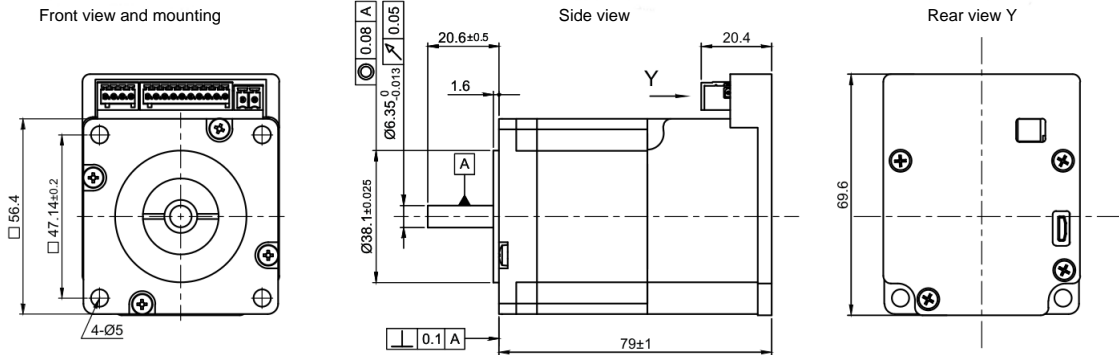
---

Version	Datum	Änderungen
1.0.3	12.05.2014	Minor corrections, field "Specified Value" in object dictionary description now used

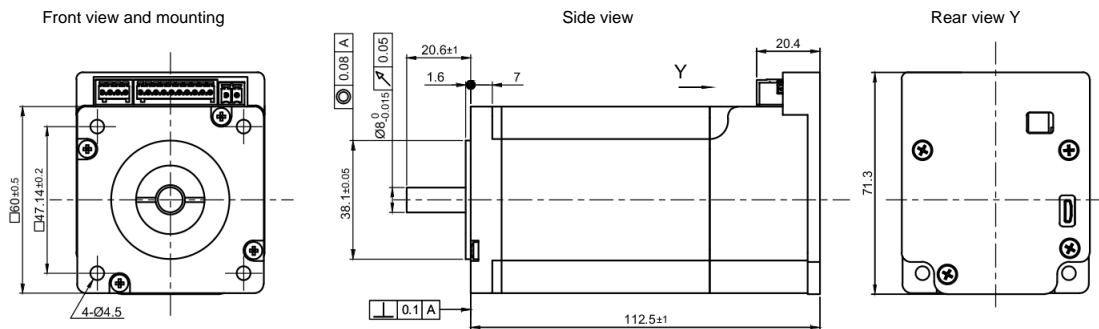
## 4 Technical data and pin configuration

### 4.1 Dimensioned drawings

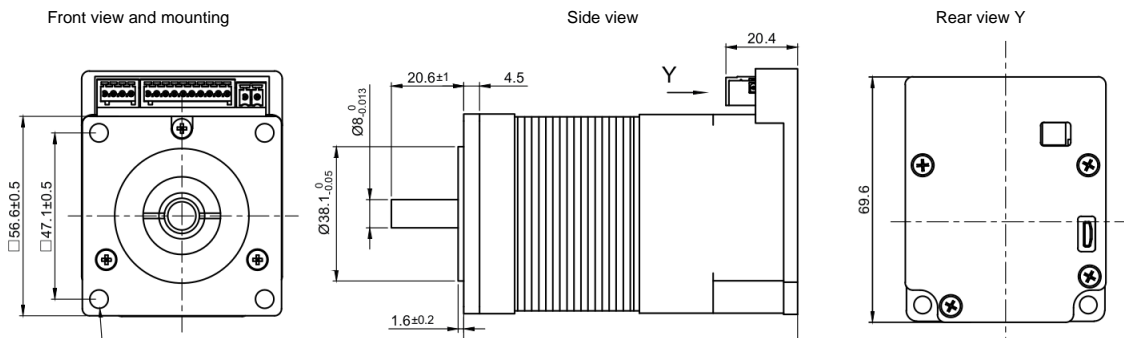
#### 4.1.1 PD4-C5918M4204-E-01



#### 4.1.2 PD4-C6018L4204-E-01



#### 4.1.3 PD4-CB59M024035-E-01



## 4.2 Electrical properties

### 4.2.1 Technical data of motor

	PD4-C	PD4-CB
Type	High-pole DC servo (stepper motor)	Low-pole DC servo (BLDC)
Operating voltage	12 V to 48 V	12 V to 24 V
Phase current eff.	4.2 A	8 A

	PD4-C	PD4-CB
RMS for 1s	Max. 6.3 A	Max. 20 A

#### 4.2.2 Technical data of I/O

Version	USB
Operating modes	Torque, speed, position, homing
Setpoint setting/ programming	Clock-direction, analog input, NanoJ V2, USB
Inputs	Single/differential clock/direction/enable (+5V / +24 V) 3 digital inputs (+24 V) 1 analog input (0 V to 10 V)
Outputs	1 output, max. 0.5 A, open drain
Integrated encoder	Single turn, magnetic absolute encoder, 1024 pulses/rev.

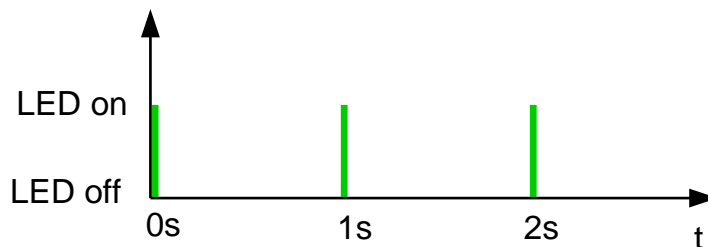
#### 4.2.3 Ratings

Type	Holding torque Ncm	Nom./ Peak Torque Ncm	Nominal Speed (rpm)	Length mm	Weight kg
PD4-C5918M4204- E-01/-08	110			79	0.8
PD4-C6018L4204- E-01/-08	350			112.5	1.5
PD4-CB59M024035- E-01	370	37 (110)	3500	95	0.9

### 4.3 LED signaling

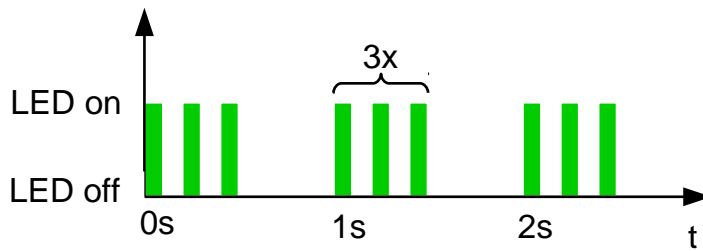
#### 4.3.1 Normal operation

In normal operation the green operating LED flashes very briefly once per second.



#### 4.3.2 Error

Should there be an error, an error number is indicated by the LED within one second. In the following illustration, the error is signaled with the number 3.



The meaning of the error number is printed in the following table.

Amount Flash	Error
1	General
2	Voltage
3	Temperature
4	Overcurrent
5	Control

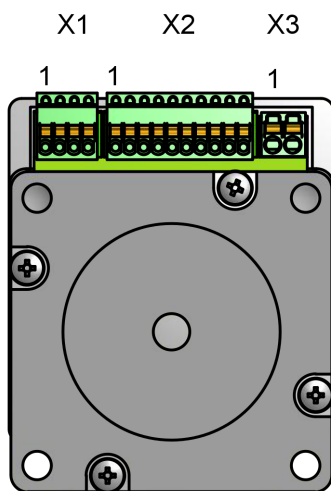
#### Note

A considerably more exact error code is stored in object **1003<sub>h</sub>** for every error that has occurred.

## 4.4 Pin configuration

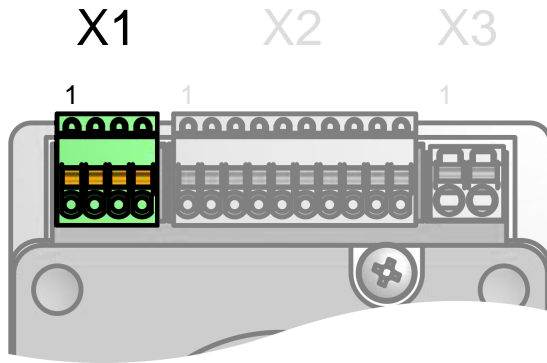
### 4.4.1 Overview

The following illustration shows the motor controller with a view of the shaft.



### 4.4.2 Analog input (connector X1)

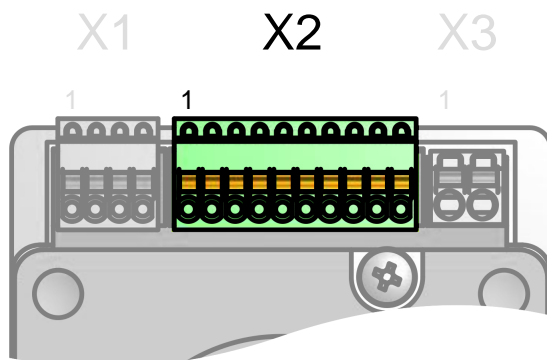
Connections for analog mode



Pin	Description
1	GND
2	Analog input (0 V - 10 V)
3	Output (open drain)
4	+12 V (voltage output, max. 100 mA)

#### 4.4.3 Clock-direction inputs (connector X2)


Connections for the analog inputs as well as inputs for the clock-direction motor controller.



Pin	Description
1	Input1 (+24 V)
2	Input2 (+24 V)
3	Input3 (+24 V)
4	-Enable +5 V / +24 V
5	Enable +5 V / +24 V
6	-Direction +5 V / +24 V
7	Direction +5 V / +24 V
8	-Clock +5 V / +24 V
9	Clock +5 V / +24 V
10	GND

#### 4.4.4 Voltage supply (connector X3)

##### Safety instruction


**CAUTION**

**Danger of electrical overvoltage!**

- Connect a charging capacitor with at least 4700 µF!
- An operating voltage higher than the admissible operating voltage (see the "**Motor technical data**" section) destroys the output stage!
- Mixing up the connections can destroy the output stage!
- Never connect or disconnect lines when live!
- The supply voltage must be selected so that it never exceeds the admissible operating voltage of the motor. In particular, interferences by other consumers or by voltages induced by the motor must be considered, and if necessary a voltage must be selected that offers an adequately high safety reserve.

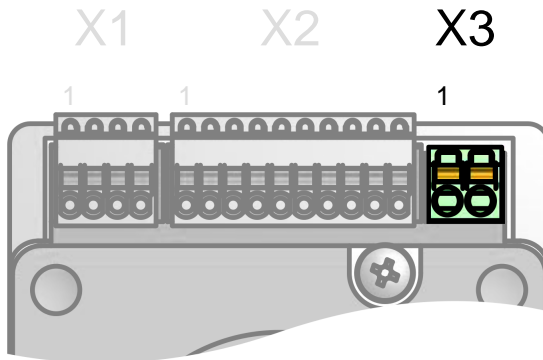
##### Voltage source

The operating or supply voltage is delivered by a battery (low voltage 12 V – 24 V), a transformer with rectification and filtering, or better a switch-mode power supply.

Interference suppression and protection measures are required when a DC power supply line with a length of >30 m is used or the motor is used on a DC bus. An EMI filter is to be installed in the DC supply cable with a small as possible distance to the motor controller/motor.

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

##### Connections



Pin	Description
1	+Vcc
2	GND

##### Permissible operating voltage

The maximum operating voltage for each motor type is:

- 12 V to 24 V for BLDC motors
- 12 V to 48 V for stepper motors

A charging capacitor of at least 4700 µF/ 50 V must be connected to the supply voltage to ensure that the permissible operating voltage is not exceeded (e.g. during braking).



## 5 Configuration

### 5.1 General information

The following options exist for configuring the motor controller:

#### DIP switches

Four DIP switches are fitted on the rear. Read the "**DIP-switch**" section for further details.

#### Configuration file

This file can be stored on the motor controller by using the USB port. Read the "**USB port**" and "**Configuration file**" sections for details.

#### VMM program

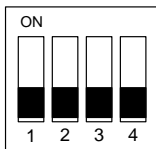
This program can be programmed, compiled, and then transferred over USB to the motor controller with NanoJEasy. Read the "**USB port**" and "**VMM program**" sections for details.

After it has been connected to a voltage supply, the motor controller reads out the configuration in the following sequence:

1. Configuration file is read out and processed.
2. The DIP switches are read out and applied as configuration.
3. The VMM program is launched

### 5.2 DIP switches

The motor controller can be configured with DIP switches on the rear. The base setting when delivered is shown in the following illustration.



A switch pushed up is in the "On" position. A switch pushed down is in the "Off" position.

Switch configurations:

Switch	On	Off
1	Analog speed mode	Clock/Direction mode; full steps
2	<ul style="list-style-type: none"> <li>• In clock/direction mode: Manual move at 30 rpm</li> <li>• In analog mode: Offset 5 V (joystick mode)</li> </ul>	<ul style="list-style-type: none"> <li>• In clock/direction mode: Standard clock/direction mode</li> <li>• In analog mode: Direction over direction input</li> </ul>
3	<ul style="list-style-type: none"> <li>• In clock/direction mode: Manual move to the left</li> <li>• In analog mode: Max speed: 100 rpm</li> </ul>	<ul style="list-style-type: none"> <li>• In clock/direction mode: Manual move to the right</li> <li>• In analog mode: Max speed: 1000 rpm</li> </ul>
4	Closed loop mode	Open loop mode

## 5.3 USB port

### CAUTION

- Use only a **standardized micro-USB cable**. Never use a USB cable that manufacturers of cell phones enclose with their products. These USB cable may have a different connector form or pin assignment.
- Do **not** save files on the motor controller other than those listed below:
  1. `pd4cfg.txt`
  2. `vmmcode.usr`
  3. `info.bin`
  4. `reset.txt`
  5. `firmware.bin`

**All other files are deleted** when the voltage supply for the motor controller is switched on!

### Note

- The motor is brought to idling when the USB cable is connected. The "Switched On" mode is set for this (see the "**DS402 Power State machine**" section for details)
- The voltage supply for the motor controller must also be switched on for USB operation.

If a USB cable is used for connecting the motor controller to a PC, the motor controller behaves like a removable storage medium. You can therefore store the configuration file or VMM program on the motor controller. All changes to files are only applied after the motor controller has been restarted (for example by short disconnection from the voltage supply).

**Tip** A frequent occurrence during set up and installation is that a file is updated and then copied back to the motor controller, it is therefore advisable to use a script file that does this work

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

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

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

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

## 5.4 Configuration file

### 5.4.1 General information

Read the **USB port** section beforehand, if you have not yet done so.

The configuration file `pd4cfg.txt` has the purpose of preassigning values for the object directory to a specific value at startup. This file is kept in a special syntax to keep access to objects in the object directory as simple as possible. The motor controller evaluates all assignments in the file from the top downwards.

### Note

Should you delete the configuration file, the file is recreated (without content) at the next motor controller restart.

#### 5.4.2 Reading and writing the file

To access to the file:

1. Connect the voltage supply to connection X3 (see the "**Pin configuration X3**" section) and switch on the voltage supply.
2. Connect the motor controller to your PC by using the USB cable.
3. After the PC has recognized the device as a removable storage medium, navigate with Explorer or a comparable editor to the directory for the motor controller. This is where file "pd4cfg.txt" is stored.
4. Open this file with a simple text editor, such as Notepad or Vi. Do not use any programs that use text styles (LibreOffice or suchlike).

After you have made changes to the file, take the following action to apply the changes:

1. Save the file if you have not already done this.
2. Disconnect the USB cable from the motor controller.
3. Disconnect the voltage supply from the motor controller for approx. 1 second.
4. Reconnect the voltage supply. At the next motor controller startup, the new values in the configuration file are read out and applied.

**Tip** You can also copy an empty file `reset.txt` to the motor controller in order to restart the motor controller.

This restarts the motor controller. The file `reset.txt` is deleted at the restart.

#### 5.4.3 Syntax

##### Comments

Lines that start with a semicolon are ignored by the motor controller.

##### Example

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

##### Assignments

### CAUTION

**Before** you set a value, find out about its data type (see the **Object directory** section)! The motor controller does **not** validate any entries for logic errors!

Values in the object directory 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 have four digits.

**<SubIndex>**

This value corresponds to the subindex of the object and is interpreted as a hexadecimal number. The value must always have two digits.

**<Value>**

The value that is to be written into the object is interpreted as a decimal number. A "0x" is to be added to the front for hexadecimal numbers.

**Note**

- There are not to be any empty spaces to the left and right of the equals sign. The following assignments are **incorrect**:
  - 6040:00 =5
  - 6040:00= 5
  - 6040:00 = 5
- The number of digits may not be changed. The index must have four digits, the subindex two. The following assignments are **incorrect**:
  - 6040:0=6
  - 6040=6
- Empty spaces at the beginning of the line are not admissible.

**Example**

Setting object **6040<sub>h</sub>:00** to the value "6":

```
6040:00=0006
```

**5.4.4 Short-circuit evaluation**

DIP switches can only be used for executing certain assignments. The following syntax is used for short-circuit execution:

```
#<No>:<Assignment>
```

**<No>**

The number of the DIP switch printed on the switch is given here. Permissible values are 1 to 4

**<Assignment>**

The assignment is given here that is described in the "**Assignments**" section.

**Example**

The following code sets object **2057<sub>h</sub>:00<sub>h</sub>** "Clock Direction Multiplier":

- to 1 if DIP switch 1 is switched to "Off".
- to 2, if the DIP switch is switched to "On" (the previous value is overwritten).

```
2057:00=00000001
#1:2057:00=00000002
```

## 5.5 VMM program

A VMM program may be executed on the motor controller. Carry out the following steps to load and launch a program on the motor controller:

1. Write and compile your program as described in the "**Virtual Machine Monitor (VMM)**" section.
2. Connect the voltage supply to connection X3 (see the "**Pin configuration X3**" section) and switch on the voltage supply.
3. Connect the motor controller to your PC by using the USB cable.
4. After the PC has recognized the device as a removable storage medium, open a file explorer and delete file "vmmcode.usr" on the motor controller
5. Use the explorer to navigate to the directory with your program. The compiled file has the same name as the source code file, only with the file name extension ".usr". Rename this file to "vmmcode.usr".
6. Now copy file "vmmcode.usr" to the motor controller.
7. Disconnect the voltage supply from the motor controller for approx. 1 second.
8. Reconnect the voltage supply. At the next start of the motor controller, the new VMM program is read-in and launched.

**Tip** You can also copy an empty file `reset.txt` to the motor controller in order to restart the motor controller.

This restarts the motor controller. The file `reset.txt` is deleted at the restart.

### Note

- The VMM program on the motor controller must have the file name "vmmcode.usr".
- If the VMM program was deleted, an empty file with name "vmmcode.usr" is created at the next startup.

**Tip** Deletion of the old VMM program and copying of the new one can be automated with a script file.

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

```
Copy <SOURCE PATH>\<OUTPUT>.usr <TARGET> :
  \vmmcode.usr
```

. For example:


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

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

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

## 6 Commissioning

### 6.1 Safety instructions

 <b>CAUTION</b>
<p><b>Alternating electromagnetic fields!</b></p> <p>Alternating electromagnetic fields around current-carrying cables, especially around the supply and motor cables, can cause interference in the motor and other devices.</p> <ul style="list-style-type: none"> <li>• Shield the cables. Run the shield connection on one side or both sides to a short earth.</li> <li>• Use twisted pair cables.</li> <li>• Keep power supply and motor cables as short as possible.</li> <li>• Ground the motor housing large-area to a short earth.</li> <li>• Run supply, motor and control cables separately.</li> </ul>

### 6.2 Preparation

The following components are required for set up and installation:

- Motor controller PD4C
- Voltage supply in accordance with the data sheet

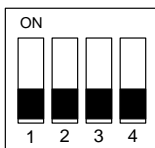
Corresponding to the mode to be used:

- For analog mode: An additional voltage source 0 V to 10 V
- For clock-direction mode: Clock generator

#### 6.2.1 Clock Direction mode

Read the **Configuration** section for the motor controller if you have not yet done so

1. Switch off any connected voltage supply.
2. Set all DIP switches to the "Off" state (corresponds the base setting when delivered):



3. Connect the voltage supply to connection X3 of the motor controller (see "**Voltage supply (connector X3)**").
4. Connect the clock generator to connection X2 (see "**Analog inputs and outputs (connector X2)**")

The motor must change the speed of rotation when the clock generator frequency changes.

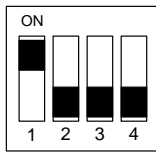
#### 6.2.2 Analog mode

<b>CAUTION</b>
<p>Make sure that the voltage at the analog input does not exceed the value of 10 V.</p>

Read the **Configuration** section for the motor controller if you have not yet done so.

1. Switch off any connected voltage supply.

2. All DIP switches must - except for switch 1 - be at the "Off" state:



3. Connect the voltage supply to connection X3 of the motor controller (see "**Voltage supply (connector X3)**").
4. Connect the adjustable voltage source to connection X1 pin 2 (see "**Analog input (connector X1)**")

The motor must change the speed when the voltage is changed at the input for the analog input.

## 7 Operating modes

### 7.1 Profile Position

#### 7.1.1 Overview

##### Description

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

##### Activation

To activate the mode, the value "1" must be set in the "modes of operation" object **6060<sub>h</sub>** (see "DS402 Power State machine").

##### Control word

The following bits in object **6040<sub>h</sub>** have a special function:

- Bit 4 starts a travel order. This is carried out on a transition of "0" to "1".
- Bit 5: If this bit is set to "1", a travel order triggered by bit 4 is immediately carried out. If it is set to "0", the travel order just being carried out is completed and only then is the next travel order started.
- Bit 6: If "0", the target position (**607A<sub>h</sub>**) is absolute and if "1", the target position is relative to the actual position.
- Bit 9: If this bit is set, the speed is not changed until the first target position is changed. This means that braking is not performed before the first destination is reached as the motor should not stop at this position.

Controlword 6040 <sub>h</sub>		
Bit 9	Bit 5	Definition
X	1	The new target position is moved to immediately.
0	0	The positioning is not completed until the next target position is being moved to with the new limitations.
1	0	The current speed is held until the current target position is reached, and only then is the new target position moved to with the new values.

See also the diagram in "Setting move commands".

##### Status word

The following bits in object **6041<sub>h</sub>** (status word) have a special function:

- Bit 10: Target reached: This bit is set to "1" when the last target was reached and the motor is idling for a specified time (**6068<sub>h</sub>**) within a tolerance window (**6067<sub>h</sub>**).
- Bit 12 (set-point acknowledge): This bit confirms the receipt of a new and valid time. It is synchronously set and reset with the "New set-point" bit in the control word.

An exception is if a new travel is started when another travel has not yet been completed, and the next travel should only be carried out after the end of the first travel. In this case, the bit is only reset when the command has been accepted and the motor controller is ready to carry out new move commands. If a new travel order is sent although this bit is still set, the latest travel order is ignored.

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

- The new target position can no longer be reached if the marginal conditions are adhered to.



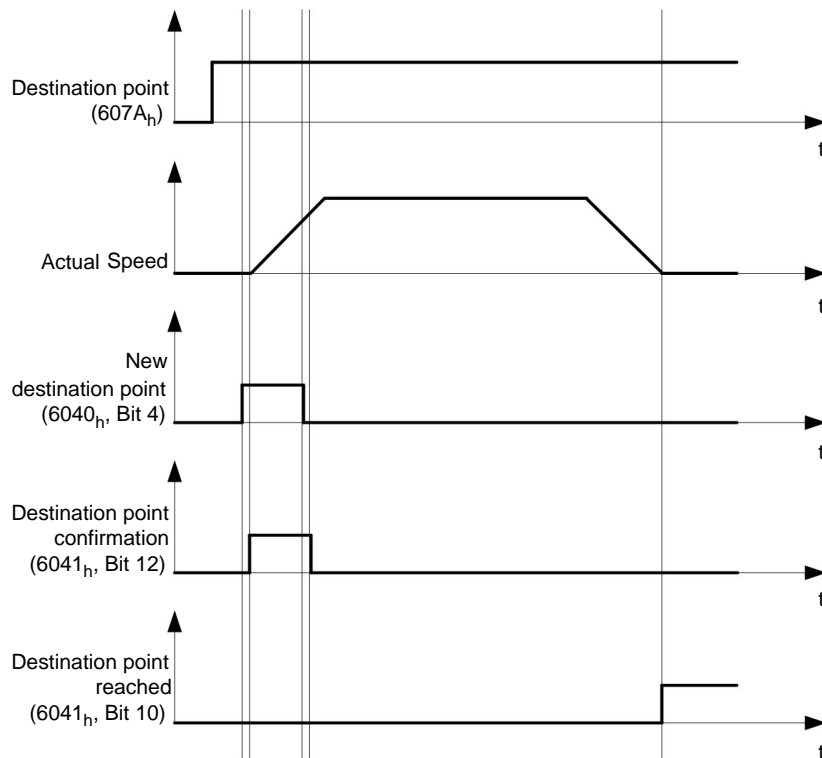
- A target position has already been moved to and a target position has already been specified. A new target position cannot be specified until the current positioning has been completed.
- The new position is outside of the valid range (**607D<sub>h</sub>** (Software Position Limit)).
- Bit 13 (Following Error): This bit is set in closed loop mode if the following error is greater than the set limits is (**6065<sub>h</sub>** (Following Error Window) and **6066<sub>h</sub>** (Following Error Time Out)).

### 7.1.2 Setting move commands

#### Move command

The new target position is specified in user units in object **607A<sub>h</sub>** (Target Position) (see **User-defined units**). Afterwards, the move command is triggered when bit 4 is set in object **6040<sub>h</sub>** (control word). If the target position is valid, the motor controller responds with bit 12 in object **6041<sub>h</sub>** (status word) and begins the positioning run. As soon as the position is reached, bit 10 is set to "1" in the status word.

#### Profile of the move command

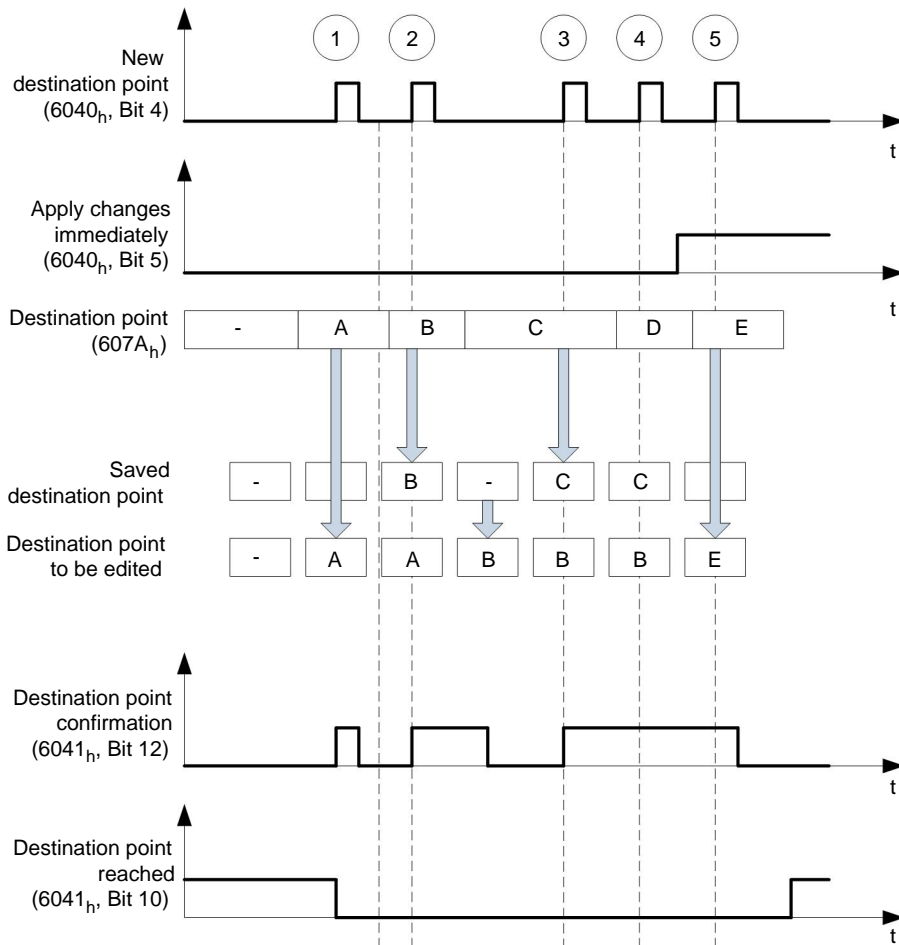


#### Further move commands

Bit 12 in object **6041<sub>h</sub>** (status word, set-point acknowledge) changes to "0" another move command can be buffered (see time 1 in the following diagram). While a target position is being moved to, a second target position can be transferred to the motor controller in preparation. All parameters – such as speed, acceleration, deceleration, etc. – can be reset (time 2). After the buffer is empty again, the next time can be added to the sequence (time 3).

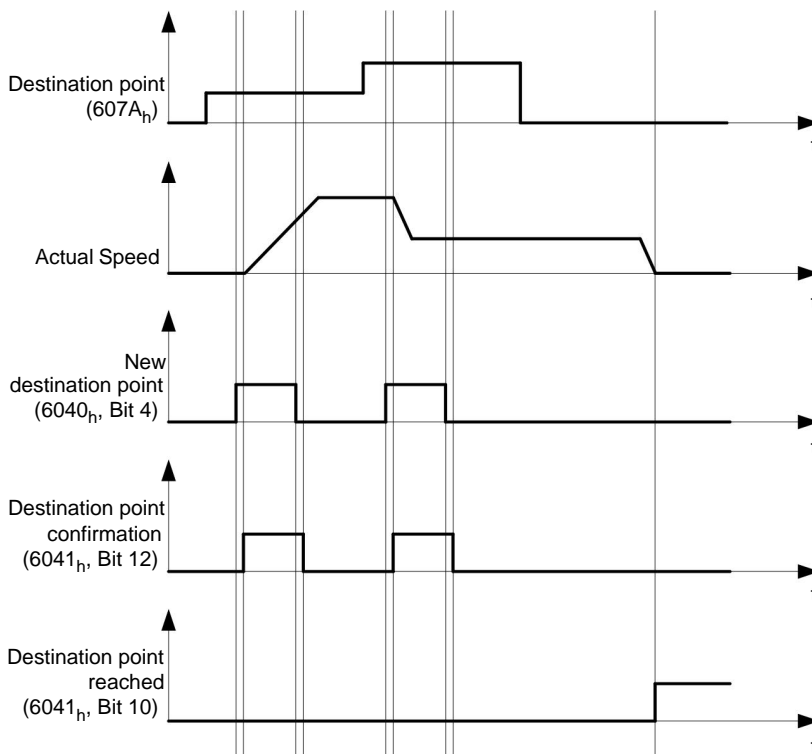
If the buffer is already full, a new time is ignored (time 4). If bit 5 in object **6040<sub>h</sub>** (control word, bit: "Change Set-Point Immediately") is set, the motor controller operates without the buffer and new move commands are implemented directly (time 5).

#### Times



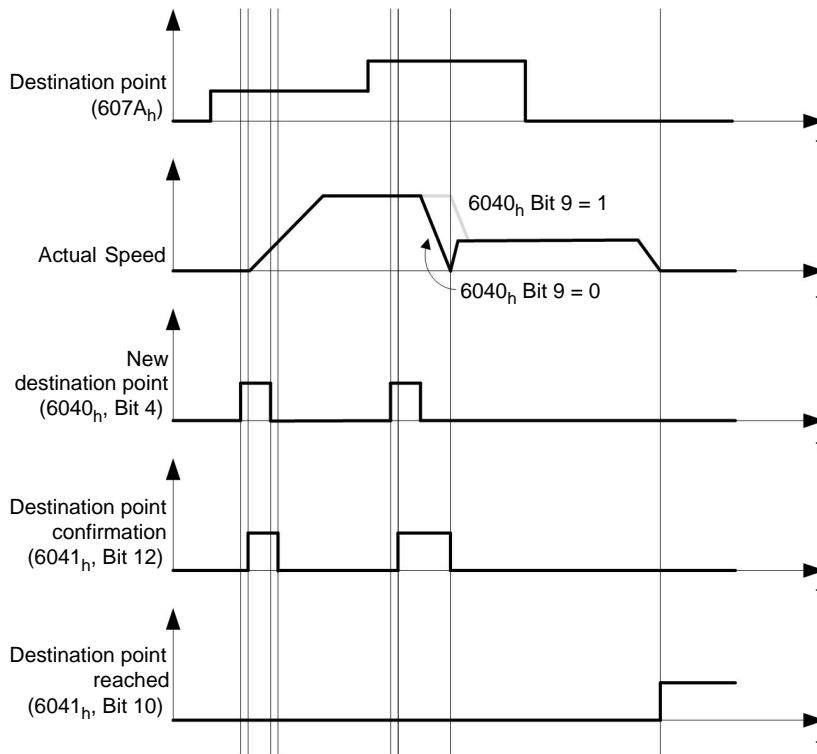
### Transition procedure for second target position

The following graphic shows the transition procedure for the second target position while the first target position is being moved to. In this figure, bit 5 of object **6040<sub>h</sub>** (control word) is set to "1" and the new target value is adopted immediately.



### Options for moving to a target position

Bit 9 in object **6040<sub>h</sub>** (control word) is "0", the actual target position is first moved to completely. In this example, the end speed (**6082<sub>h</sub>**) of the first target position is zero. If bit 9 is set to "1", the end speed is held until the target position is reached; only then do the new marginal conditions apply.



### 7.1.3 Marginal conditions for a positioning run

#### Object entries

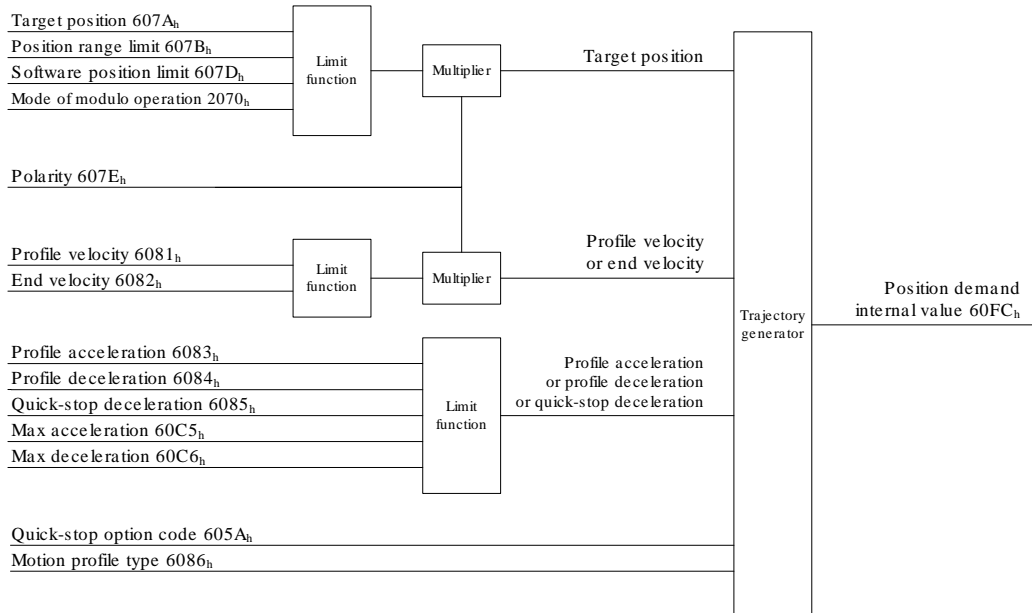
The marginal conditions for the position to which the run is made can be set in the following entries of the object directory:

- **6064<sub>h</sub>** (Position Actual Value): Actual position of the motor
- **607A<sub>h</sub>** (Target Position): Intended target position
- **607B<sub>h</sub>** (Position Range Limit): Definition of the limit stops (see the section further down)
- **607C<sub>h</sub>** (Home Offset): Move of the machine zero point (**see "Homing mode"**)
- **607D<sub>h</sub>** (Software Position Limit): Limits of a modulo operation for replication of a continuous rotational axis (**see "Axis scaling"**)
- **607E<sub>h</sub>** (Polarity): Direction of rotation
- **2070<sub>h</sub>** (Mode Of Modulo Operation): Determination of the direction of rotation if a rotational axis is replicated in the software
- **6081<sub>h</sub>** (Profile Velocity): Maximum speed with which the position should be moved to
- **6082<sub>h</sub>** (End Velocity): Speed when reaching the target position
- **6083<sub>h</sub>** (Profile Acceleration): Required acceleration
- **6084<sub>h</sub>** (Profile deceleration): Required deceleration
- **6085<sub>h</sub>** (Quick Stop Deceleration): Emergency stop deceleration in case of the "Quick stop active" state of the "DS402 Power State machine"
- **6086<sub>h</sub>** (Motion Profile Type): Type of ramp to be moved to; if the value is "0", jerk is not limited, if value is "3", the values from 60A4<sub>h</sub>:1<sub>h</sub>- 4<sub>h</sub> are set as jerk limitations.
- **60C5<sub>h</sub>** (Max Acceleration): The maximum acceleration that may not be exceeded when moving to the end position
- **60C6<sub>h</sub>** (Max Deceleration): The maximum deceleration that may not be exceeded when moving to the end position

- **60A4<sub>h</sub>** (Profile Jerk), Subindex 01<sub>h</sub> to 04<sub>h</sub>: Objects for describing the limit values for the jerk

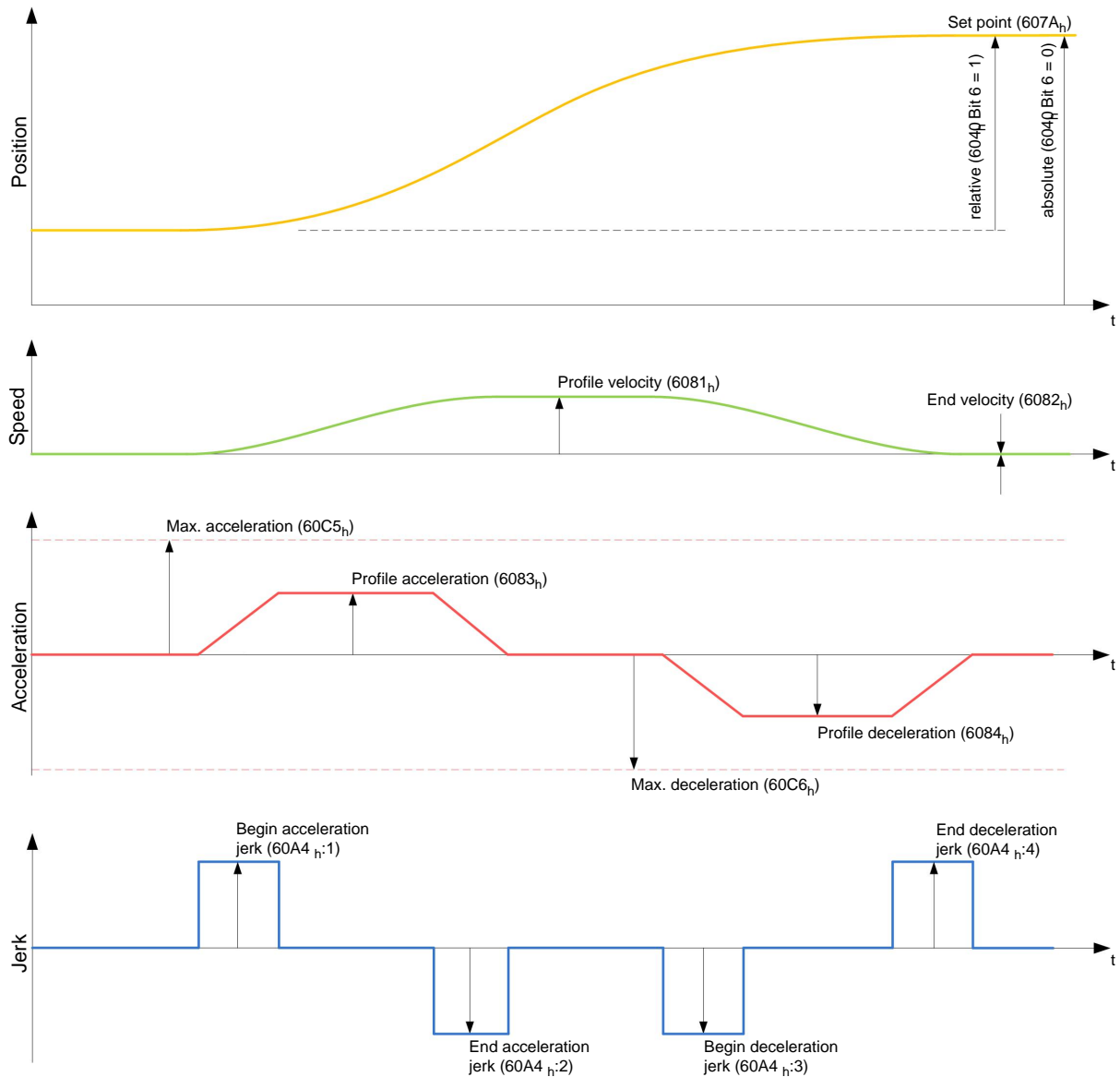
### Objects for the positioning run

The following graphic shows the objects involved for the marginal conditions for the positioning run.



### Parameters for the target position

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



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

#### Description

Two basic modes exist: the "jerk-limited" and "non-jerk-limited" mode.

#### Jerk-limited mode

A jerk-limited positioning is achieved by setting object **6086<sub>h</sub>** to "3". This causes the entries for the jerks in object **60A4<sub>h</sub>:1<sub>h</sub>-4<sub>h</sub>** to become valid.

#### Non-jerk-limited mode

A "0" in an entry means that there is no jerk limitation at the particular point in the profile.

If all four entries of object **60A4<sub>h</sub>** are set to "0", a non-jerk-limited ramp is traveled.

A "non-jerk-limited" ramp is traveled in two ways: Either all values of the jerk in entries **60A4<sub>h</sub>:1<sub>h</sub>** to **60A4<sub>h</sub>:4<sub>h</sub>** are set to "0" and object **6086<sub>h</sub>** is set to "3", or the entry in object **6086<sub>h</sub>** is set to "0".

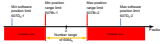
## 7.1.5 Axis scaling

### Software limit switch

The motor controller takes the software limit switch into account (**607D<sub>h</sub>** (Software Position Limit)). Move commands beyond the limit switch are rejected. If the motor is outside of the admissible range when the limit switch is set up, only the move commands in the direction of the admissible range are accepted.

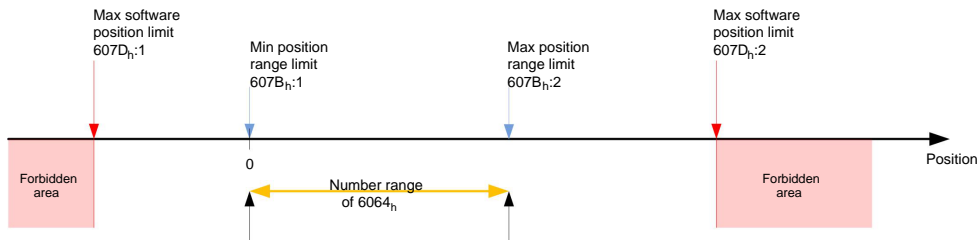
### Linear axis

For a linear axis, the software limit switches **607D<sub>h</sub>:1<sub>h</sub>** and **607D<sub>h</sub>:2<sub>h</sub>** must be outside of the value range of the limit stops.



### Rotational axis

For a rotational axis, a software overflow (e.g. modulo operation) can be set. The specification of the position in object **6064<sub>h</sub>** then begins at the opposite end of the range in each case when the limit is exceeded or undercut. The specification of the target position in object **607A<sub>h</sub>** can continue to take place on the absolute number line for the position, and only the specification of the position in object **6064<sub>h</sub>** is affected.



### Direction of rotation

In this mode, the resulting rotation direction is not always clearly defined. For this reason, modulo operation **2070<sub>h</sub>** (Mode Of Modulo Operation) can be used to specify how the motor controller is to behave:

- 0: Include overflow in count
- 1: Always rotate in the positive direction
- 2: Always rotate in the negative direction
- 3: Always move the shorter distance (standard setting)
- 4: Always move the longer distance

### Include overrun in count

In the normal modulo mode, it is not possible to move one full revolution or more. In the "Include overrun in count" (object **2070<sub>h</sub>** set to "0"), these overruns are counted.

Example:

The motor is currently in position 1000, the underrun is at 0 and the overrun is at 2000. A move command for a full revolution to position 3000 is usually ignored because  $(3000 \bmod 2000) = 1000$  and the motor is already at this position. If the value "0" is entered in object **2070<sub>h</sub>**, the move commands are computed in absolute terms, but the output of object **6040<sub>h</sub>** still takes place with overruns and underruns.



### Positive direction of rotation

If object **2070<sub>h</sub>** is set to "1", the full revolutions (wrap arounds) are not counted and the motor always rotates in the positive direction, even if this may be the longer distance.

Example:

An absolute move command to 500 (or the equivalent of 500 units) from position 1000 means that the motor will take the "long way" by wrapping around, as this is the positive direction of rotation.



### Negative direction of rotation

This mode is the same as the "Positive direction of rotation" mode (see "**Positive direction of rotation**"), only that the direction of rotation is always negative (object **2070<sub>h</sub>** set to "2").

### Shortest distance

If object **2070<sub>h</sub>** is set to "3", the motor is always turned in the direction of the shortest distance. Any arising wrap-arounds ( see "**Include overrun in count**") are ignored.

Example:

An absolute move command to 1500 from position 250 (or the equivalent of 1250 units) causes the motor to move counterclockwise. This corresponds to a (shorter) distance of 750 units, while the distance in a clockwise direction would be 1250 units.



### Longest distance

This mode is identical to the "Shortest distance" mode (see "**Shortest distance**"), only that the direction of the longest distance is always selected (object **2070<sub>h</sub>** set to "4").

## 7.2 Velocity

### 7.2.1 Description

This mode operates the motor with a specified target in a manner similar to a frequency converter. In contrast to profile velocity mode, this mode operates without a speed monitor and does not permit jerk-limited ramps to be selected.

### 7.2.2 Activation

To activate the mode, the value "2" must be set in the "modes of operation" object **6060<sub>h</sub>** (see "DS402 Power State machine").

### 7.2.3 Control word

The following bits in object **6040<sub>h</sub>** have a special function:

- Bit 2 is used to trigger an emergency stop. If it is set to "0", the motor carries out a quick stop with the quick stop ramp set in object **604A<sub>h</sub>**. Then the motor controller changes to the "Switch on disabled" state (see **6040<sub>h</sub>**).
- Bit 8 (Stop): On a transition of "0" to "1" the motor accelerates up to the target speed with the set acceleration ramp. On a transition of "0" to "1", the motor brakes according to the brake ramp and comes to a stop.

### 7.2.4 Status word

The following bits in object **6041<sub>h</sub>** (status word) have a special function:

- Bit 11: Limit exceeded: The target speed exceeds or undercuts the entered limit values.

### 7.2.5 Object entries

The following objects are required to control this mode:

- **604C<sub>h</sub>** (Dimension Factor):

The unit for the speed specifications for the following objects are defined here. If subindices 1 and 2 are set to value "1", the speed is indicated in revolutions per minute.

Otherwise, subindex 1 contains the multiplier and subindex 2 the divisor with which the speed specifications are computed. The result is interpreted as revolutions per second, at object **2060<sub>h</sub>** the selection is made of whether these are electrical (**2060<sub>h</sub> = 0**) or mechanical (**2060<sub>h</sub> = 1**) revolutions per second.

The target speed is set in user units here.

- **6042<sub>h</sub>**: Target Velocity
- **6048<sub>h</sub>**: Velocity Acceleration

This object defines the start acceleration. Subindex 1 contains the speed change, and subindex 2 the associated time in seconds. Both together are computed as the acceleration:

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

- **6049<sub>h</sub>** (Velocity Deceleration):

This object defines the deceleration. The subindices are structured as described in object **6048<sub>h</sub>**, and the speed difference must be indicated by a positive sign.

- **6085<sub>h</sub>** (Quick Stop Deceleration):

This object defines the quick stop deceleration. The subindices are structured as described in object **6048<sub>h</sub>**, and the speed difference must be indicated by a positive sign.

- **6046<sub>h</sub>** (Velocity Min Max Amount):

In this object the limitations to target speeds are specified.



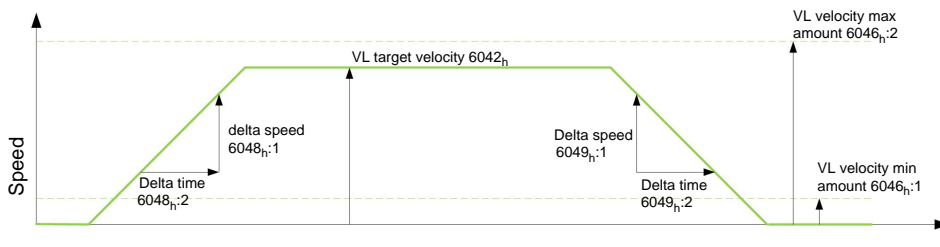
The minimum speed is set in **6046<sub>h</sub>:1<sub>h</sub>**. If the target speed (**6042<sub>h</sub>**) is lower than the minimum speed, the value is set to the minimum speed **6046<sub>h</sub>:1<sub>h</sub>**.

The maximum speed is set in **6046<sub>h</sub>:2<sub>h</sub>**. If the target speed (**6042<sub>h</sub>**) exceeds the maximum speed, the value is limited to the maximum speed **6046<sub>h</sub>:2<sub>h</sub>**.

- **604A<sub>h</sub>** (Velocity Quick Stop):

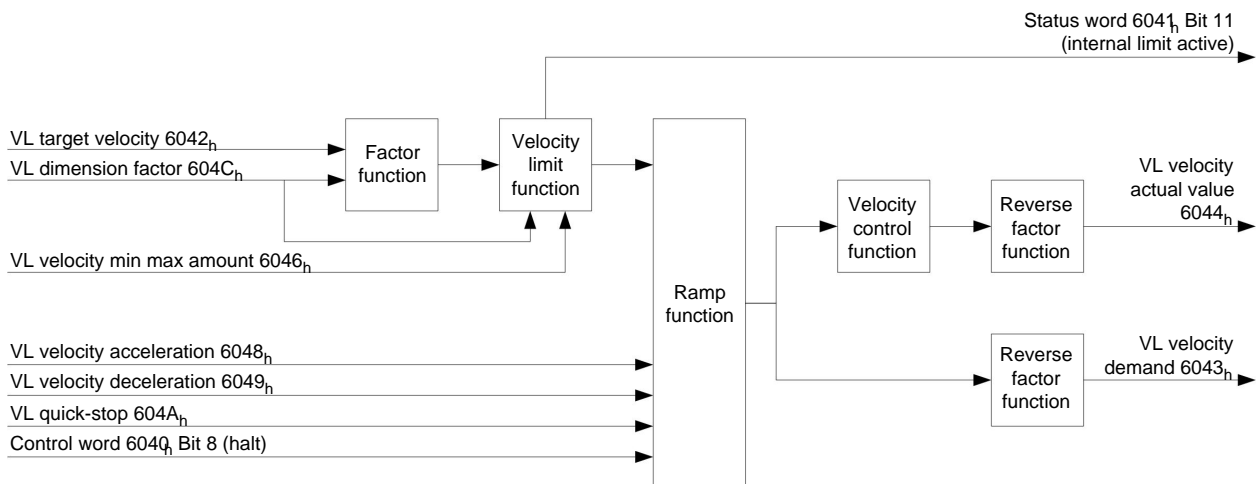
The quick stop ramp can be set with this object. Subindices 1 and 2 are the same as specified for object **6048<sub>h</sub>**.

### Speeds in Velocity Mode



### Objects for the Velocity Mode

The ramp generator follows the target speed while adhering to the set speed and acceleration limits. Bit 11 is set in object **6041<sub>h</sub>** (internal limit active) when a limitation is active.



## 7.3 Profile velocity

### 7.3.1 Description

This mode operates the motor in Velocity Mode with expanded ramps. Unlike velocity mode (see "**Velocity**"), the actual speed can be monitored via an external encoder in this mode.

### 7.3.2 Activation

To activate the mode, the value "3" must be set in the "modes of operation" object **6060<sub>h</sub>** (see "**DS402 Power State machine**").

### 7.3.3 Control word

The following bits in object **6040<sub>h</sub>** have a special function:

- Bit 2 is used to trigger an emergency stop. If it is set to "0", the motor carries out a quick stop with the ramp set in object **6085<sub>h</sub>**. Then the motor controller changes to the "Switch on disabled" state (see **6040<sub>h</sub>**).

- Bit 8 (Stop): On a transition of "0" to "1", the motor accelerates up to the target speed with the set starting ramp. On a transition of "0" to "1", the motor brakes and comes to a stop.

### 7.3.4 Status word

The following bits in object **6041<sub>h</sub>** (status word) have a special function:

- Bit 10 (target speed reached; Target Reached: This bit in combination with bit 8 in the control word indicates whether or not the target speed has been reached, the motor is braking, or the motor is idling (see table).

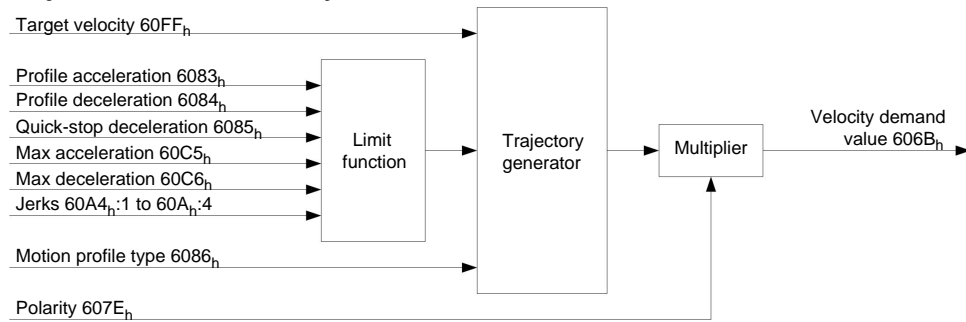
<b>6041<sub>h</sub></b> <b>Bit 10</b>	<b>6040<sub>h</sub></b> <b>Bit 8</b>	<b>Description</b>
0	0	Target speed attained
0	1	Axis is braking
1	0	Target speed within target window (defined in <b>606D<sub>h</sub></b> and <b>606E<sub>h</sub></b> )
1	1	Speed of axis is 0

### 7.3.5 Object entries

The following objects are required to control this mode:

- **606B<sub>h</sub>** (Velocity Demand Value):  
This object contains the output of the ramp generator which is the specified value for the speed controller at the same time.
- **606C<sub>h</sub>** (Velocity Actual Value):  
Specifies the current actual speed.
- **606D<sub>h</sub>** (Velocity Window):  
This value specifies by how much the actual speed may vary from the set speed for bit 10 (target speed reached; Target Reached) in object **6041<sub>h</sub>** (status word) is to be set to "1".
- **606E<sub>h</sub>** (Velocity Window Time):  
This object specifies how long the actual speed and the set speed must be near each other in magnitude (see **606D<sub>h</sub>**) for bit 10 "Target reached" in object **6041<sub>h</sub>** (status word) to be set to "1".
- **607E<sub>h</sub>** (Polarity):  
If bit 6 is set to "1", the sign (plus/minus) of the target speed is reversed.
- **6083<sub>h</sub>** (Profile acceleration):  
Sets the value for the acceleration ramp in velocity mode.
- **6084<sub>h</sub>** (Profile Deceleration):  
Sets the value for the braking ramp in velocity mode.
- **6085<sub>h</sub>** (Quick Stop Deceleration):  
Sets the value for the braking ramp for the quick stop in velocity mode.
- **6086<sub>h</sub>** (Motion Profile Type):  
Here the ramp type can be selected (0 = trapezoid ramp, 3 = jerk-limited ramp).
- **604A<sub>h</sub>** (Velocity Quick Stop), subindex 01<sub>h</sub> to 04<sub>h</sub>:  
The four jerk values are specified here if a jerk-limited ramp is set.
- **60FF<sub>h</sub>** (Target Velocity):  
Specifies the target speed to be attained.
- **2031<sub>h</sub>** (Peak Current):  
Maximum current in mA

## Objects in Profile Velocity Mode

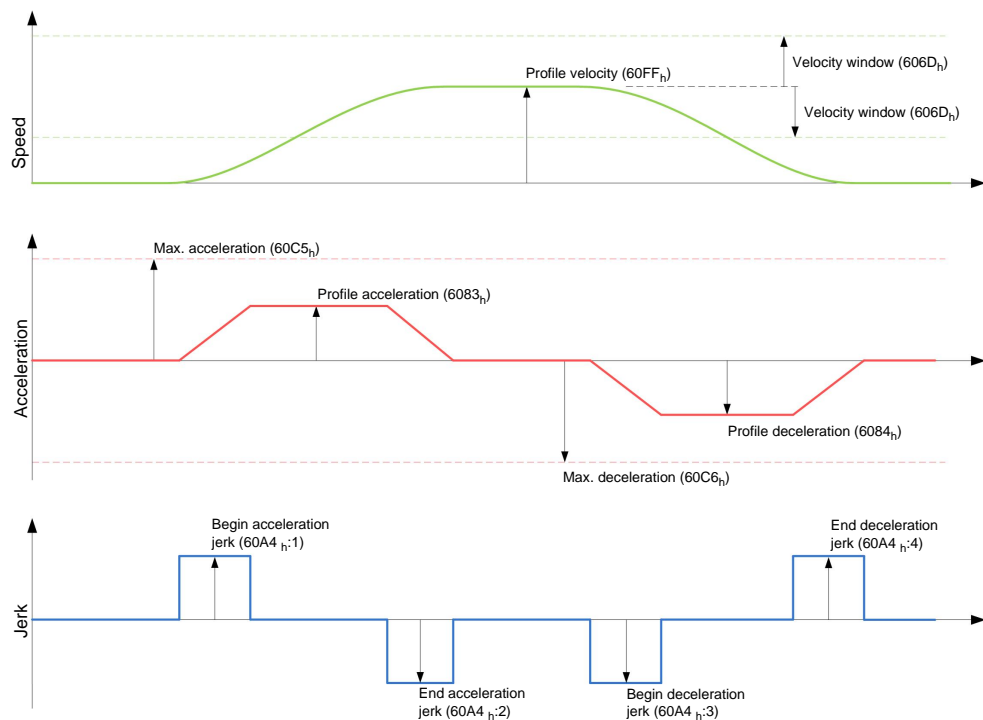


## Mode activation

After the mode was selected in object **6060<sub>h</sub>** (Modes Of Operation) and the "power state machine" (see "**DS402 Power State machine**") was switched to "Operation enabled", the motor is accelerated to the target speed in object **60FF<sub>h</sub>** (see the following diagrams). The speed, the acceleration and, in the case of jerk-limited ramps, the jerk limited values are taken into account.

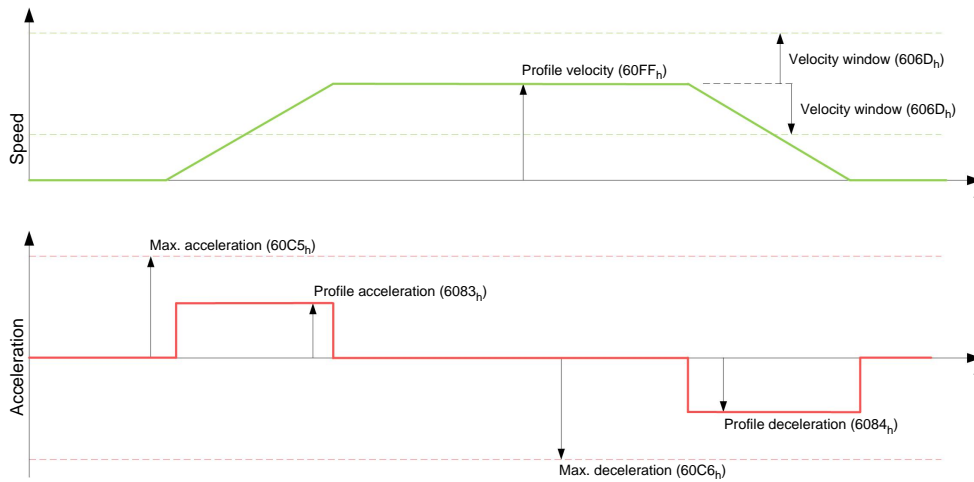
## Limitations in the jerk-limited case

The following diagram shows the adjustable limitations in the jerk-limited case (**6086<sub>h</sub> = 3**).



## Limitations in trapezoid case

This diagram shows the adjustable limitations for the trapezoid case (**6086<sub>h</sub> = 0**).



## 7.4 Profile torque

### 7.4.1 Description

In this mode, the torque is specified as the set point and is moved to via a ramp function.

### 7.4.2 Activation

To activate the mode, the value "4" must be set in the "modes of operation" object **6060<sub>h</sub>** (see "DS402 Power State machine").

### 7.4.3 Control word

The following bits in object **6040<sub>h</sub>** have a special function:

- Bit 8 (Stop): If this bit is set to "0", the motor is started according to the specifications. When set to "1", the motor is brought to idling according to the specified values.

### 7.4.4 Status word

The following bits in object **6041<sub>h</sub>** (status word) have a special function:

- Bit 10 (Target Reached): This bit in combination with bit 8 of object **6040<sub>h</sub>** (control word) indicates whether or not the specified torque has been reached (see the following table).

<b>6040<sub>h</sub></b>	<b>6041<sub>h</sub></b>	<b>Description</b>
<b>Bit 8</b>	<b>Bit 10</b>	
0	0	Specified torque not attained
0	1	Specified torque attained
1	0	Axis accelerated
1	1	Speed of axis is 0

### 7.4.5 Object entries

All values of the following entries in the object directory must be specified as one thousandth of the maximum torque, which corresponds to the maximum current (**2031<sub>h</sub>**). This includes the following objects:

- **6071<sub>h</sub>** (Target Torque):  
Target value of the torque
- **6072<sub>h</sub>** (Max Torque):

Maximum torque during the entire ramp (acceleration, hold torque, brake)

- **6074<sub>h</sub>** (Torque Demand):

Current output value of the ramp generator (torque) for the control

- **6087<sub>h</sub>** (Torque Slope):

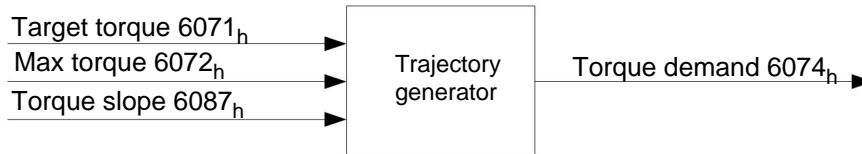
Maximum change of the torque per second

- **3202<sub>h</sub>** Bit 5 (Motor Drive Submode Select):

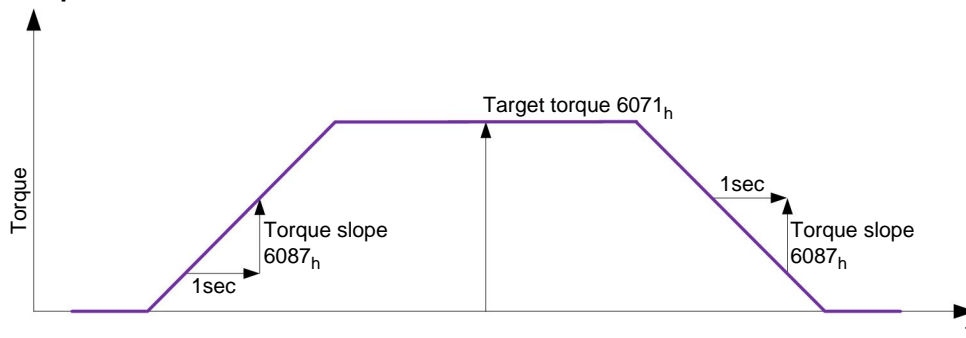
If this bit is set to "0", the drive control is operated in torque-limited velocity mode, i.e. the maximum speed can be limited in object **2032<sub>h</sub>** and the control can work in field weakening mode.

If this bit is set to "1", the control works in torque mode, the maximum speed cannot be limited here and field weakening mode is not possible.

### Objects of the ramp generator



### Torque curve



## 7.5 Homing

### 7.5.1 Overview

#### Description

The purpose of the reference run (homing method) is to synchronize the motor controller with the encoder index of the motor or position switch in a system.

#### Activation

To activate the mode, the value "6" must be set in the "modes of operation" object **6060<sub>h</sub>** (see "**DS402 Power State machine**").

If a reference and/or limit switch is used, these special functions first need to be activated in the I/O configuration (see "**Digital inputs and outputs**").

#### Control word

The following bits in object **6040<sub>h</sub>** have a special function:

- Bit 2 is used to trigger an emergency stop. If it is set to "0", the motor carries out a quick stop with the ramp set in object **6085<sub>h</sub>**. After this, the motor enters into the "Switch on disabled" state (see "**DS402 Power State machine**").

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

### Status word

The following bits in object **6041<sub>h</sub>** (status word) have a special function:

Bit 13	Bit 12	Bit 10	Description
0	0	0	Reference run is performed
0	0	1	Reference run is interrupted or not started
0	1	0	Reference run is started but the target has not yet been reached
0	1	1	Reference run has been fully completed
1	0	0	Errors during the reference run, the motor is still rotating
1	0	1	Error during the reference run, the motor is idling

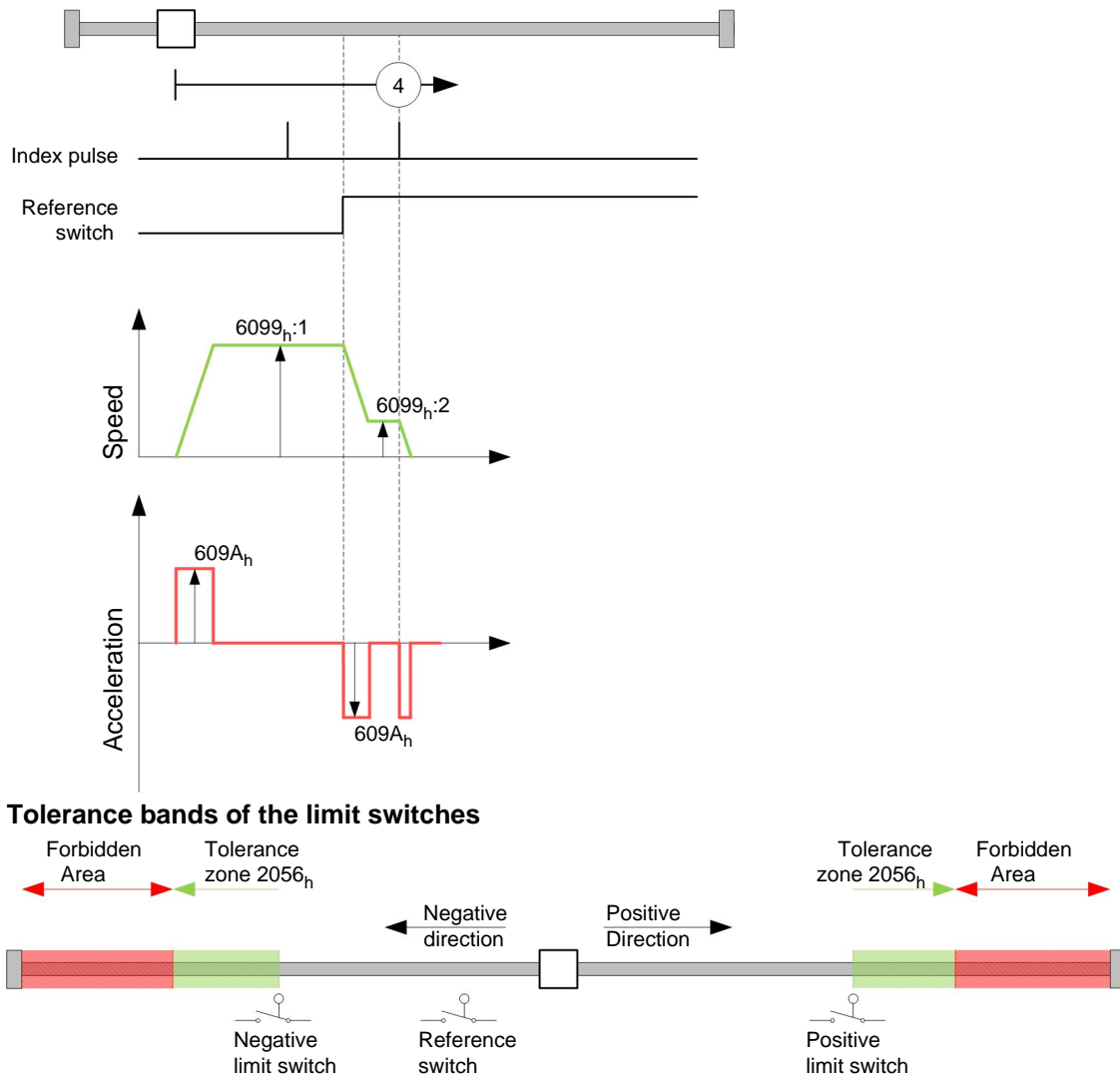
### Object entries

The following objects are required to control this mode:

- **6098<sub>h</sub>** (Homing Method):  
Method to be used for referencing (see "Reference run method")
- **6099<sub>h</sub>:01<sub>h</sub>** (Speed During Search For Switch):  
The speed for the search for the switch
- **6099<sub>h</sub>:02<sub>h</sub>** (Speed During Search For Zero):  
The speed for the search for the index
- **609A<sub>h</sub>** (Homing Acceleration):  
Acceleration and deceleration for the reference run
- **2056<sub>h</sub>** (Limit Switch Tolerance Band):  
After moving to the positive or negative limit switch, the motor controller permits a tolerance range that the motor may not further travel. If this tolerance range is exceeded, the motor stops and the motor controller changes to the "Fault" state. If limit switches can be activated during the reference run, the tolerance range selected should be sufficiently large so that the motor does not leave the tolerance range when braking. Otherwise, the reference run cannot be completed successfully. After completion of the reference run, the tolerance range can be set back to "0" if this is required by the application.
- **203A<sub>h</sub>:1<sub>h</sub>** (Minimum Current For Block Detection):  
Minimum current threshold that, when exceeded, detects blocking of the motor at a block.
- **203A<sub>h</sub>:2<sub>h</sub>** (Period Of Blocking):  
Specifies the time in ms that the motor is nevertheless still to travel against the block after block detection.
- **203A<sub>h</sub>:3<sub>h</sub>** (Block Detection Time)  
Specifies the time in ms that the current has to be at least above the minimum current threshold in order to detect a block

### Speeds of the reference run

The figure shows the speeds of the reference run using method 4 as an example:



## 7.5.2 Reference run method

### Description

The reference run method is written into object **6098<sub>h</sub>** as a number and defines whether referencing should be performed on a switch flank (rising/falling), a current threshold for block detection or an index pulse is referenced, or in which direction the reference run should start. Methods that use the index pulse of the encoder are within the number range 1 to 14, 33 and 34. Methods that reference a limit switch are between 17 and 30, but their travel profiles are identical with those of the methods 1 to 14. These numbers are shown in circles in the following figures. Methods that do not use a limit switch, and instead travel against a block is to be detected, must be called up with a minus in front of the method number.

For the following diagrams, the negative movement direction is to the left. The limit switch is located in front of the mechanical block in each case, and the reference switch (home switch) is between the two limit switches. The index pulses come from the encoder, which is connected with the motor shaft and motor controller.

For methods that use homing on block, the same illustrations apply as for the methods with limit switch. New illustrations are not shown as nothing changes except for the missing limit switches. In this case, the limit switches have to be replaced by a mechanical block in the illustrations.

## Homing on block

Homing on block functions perfectly only in closed loop mode at the moment. The finer points that have to be observed for homing on block in closed loop mode, for instance, are given in detail in the section on controls.

For certain applications it is appropriate to travel against the block for a specific time after a block has been detected. This time can be set in object **203A<sub>h</sub>:2<sub>h</sub>** in ms.

To ensure very precise detection of the block, the block should be traveled against with a very low speed (**6099<sub>h</sub>:1<sub>h</sub>**), high current limit (**203A<sub>h</sub>:1<sub>h</sub>**), and high homing acceleration (**609A<sub>h</sub>**). Additionally, detection can be refined by the block detection time (**203A<sub>h</sub>:3<sub>h</sub>**).

## Methods overview

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

Methods 17 to 32 are identical with the methods 1 to 14 with the exception that referencing is only performed on the limit or home switch and not on the index pulse.

- Methods 1 to 14 contain an index pulse
- Methods 15 and 16 do not exist
- Methods 17 to 30 do not have an index pulse
- Methods 31 and 32 do not exist
- Methods 33 and 34 reference only to the next index pulse
- Method 35 references to the actual position

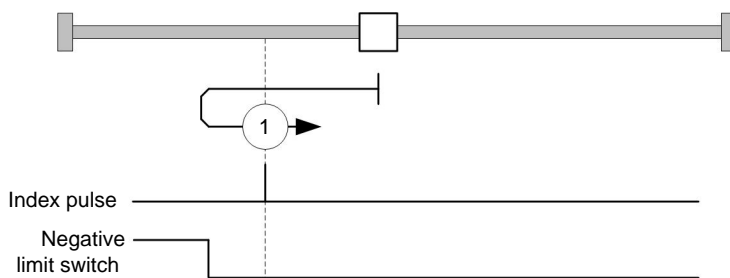
The following methods can be used for homing on block:

- Methods -1 to -2 and -7 to -14 contain an index pulse
- Methods -17 to -18 and -23 to -30 do not have an index pulse

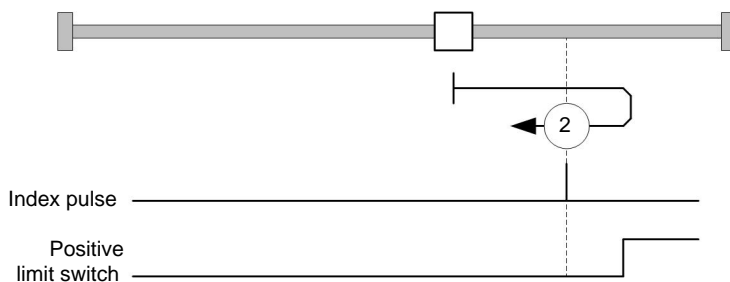
## Methods 1 and 2

Reference the limit switch and index pulse.

Method 1 references a negative limit switch and index pulse:



Method 2 references a positive limit switch and index pulse:

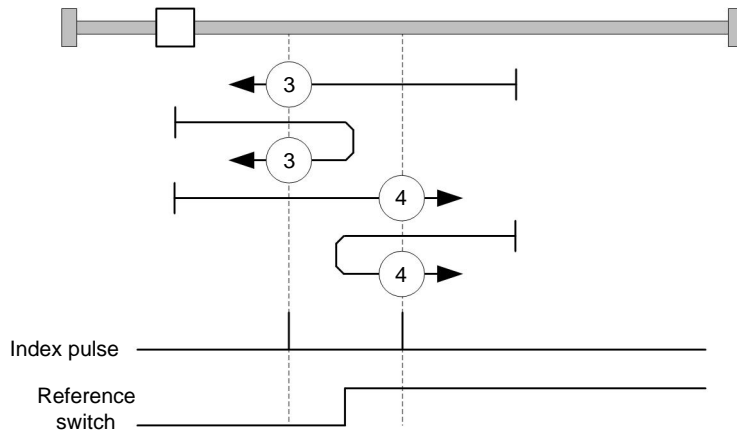


## Methods 3 to 6

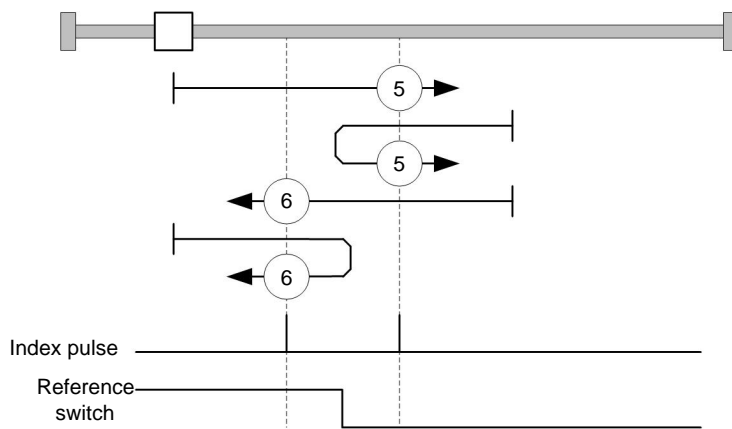
These methods reference the switch flank of the reference switch and index pulse.

In the methods 3 and 4, the left switch flank of the reference switch is used as a reference:





In the methods 5 and 6, the right switch flank of the reference switch is used as a reference:

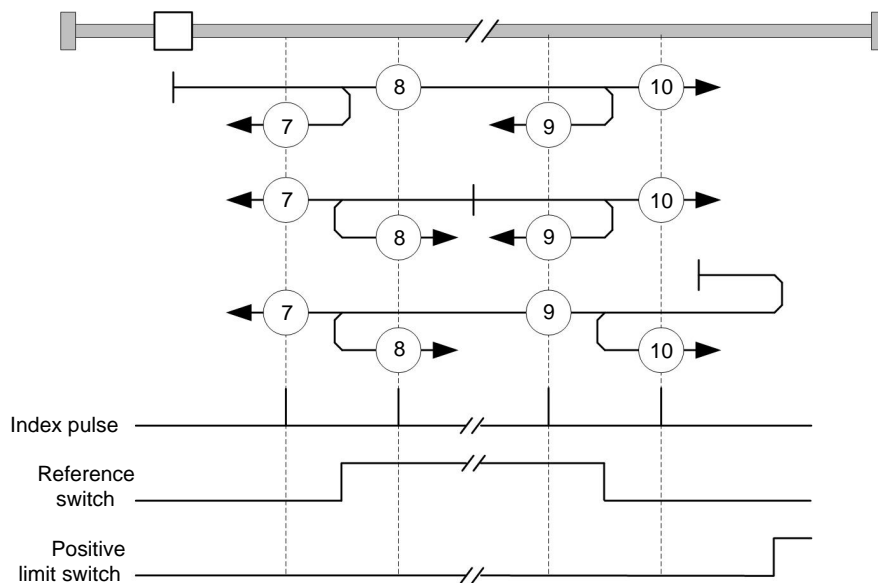


### Methods 7 to 14

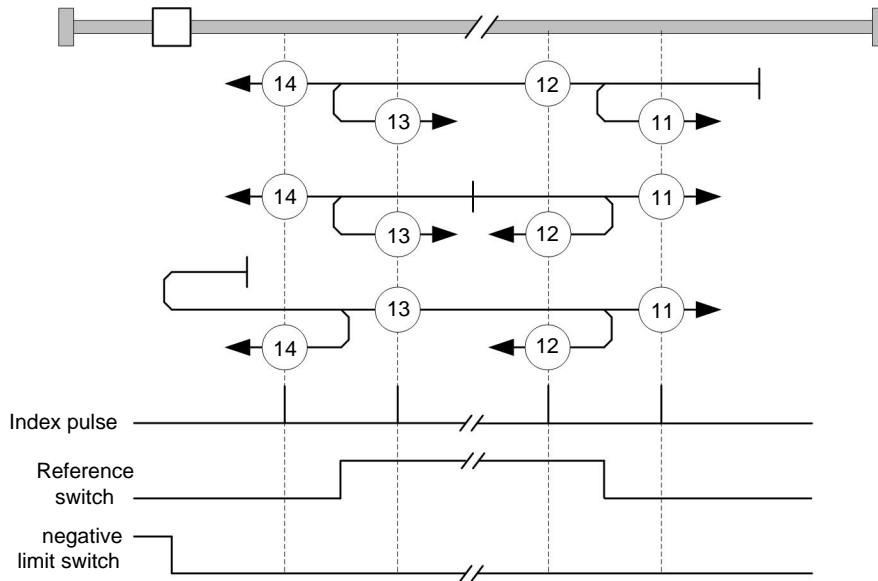
These methods reference the home switch and index pulse (with limit switches).

For these methods, the actual position relative to the reference switch is unimportant. With method 10, referencing is for instance always to the index pulse on the right next to the right flank of the reference switch.

The methods 7 to 10 take the positive limit switch into account:



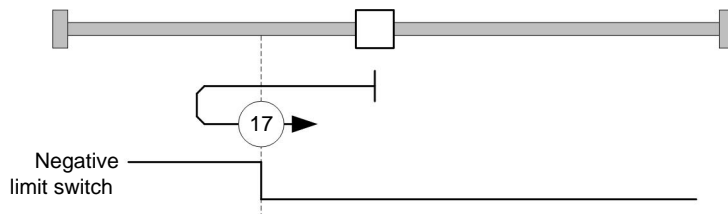
The methods 11 to 14 take the negative limit switch into account:



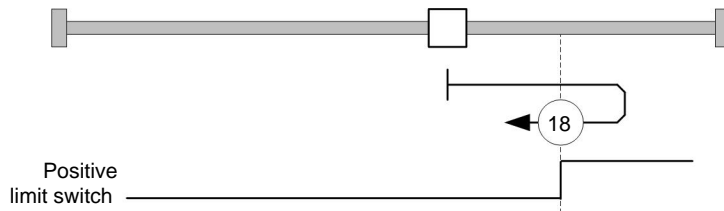
### Methods 17 and 18

These methods reference the limit switch without the index pulse.

Method 17 references the negative limit switch:



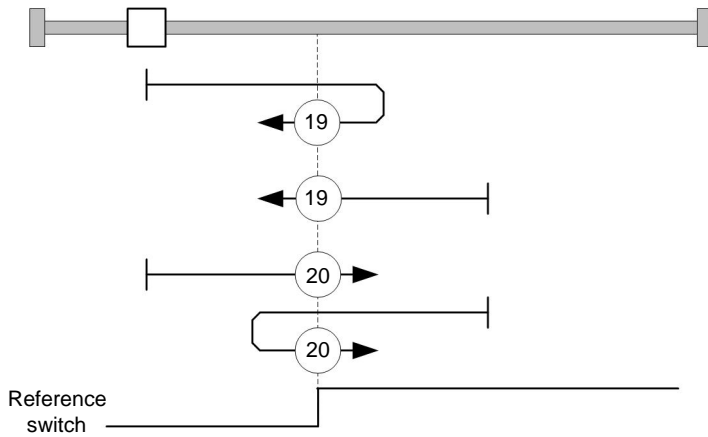
Method 18 references the positive limit switch:



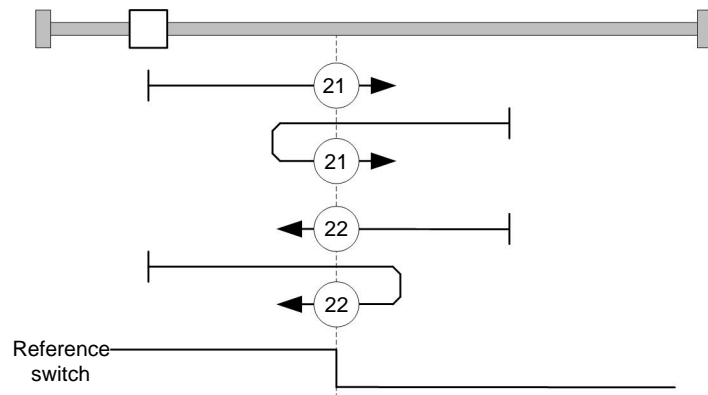
### Methods 19 to 22

These methods reference the switch flank of the reference switch without the index pulse.

In the methods 19 and 20 (equivalent to methods 3 and 4), the left switch flank of the reference switch is used as a reference:



In the methods 21 and 22 (equivalent to methods 5 and 6), the right switch flank of the reference switch is used as a reference:

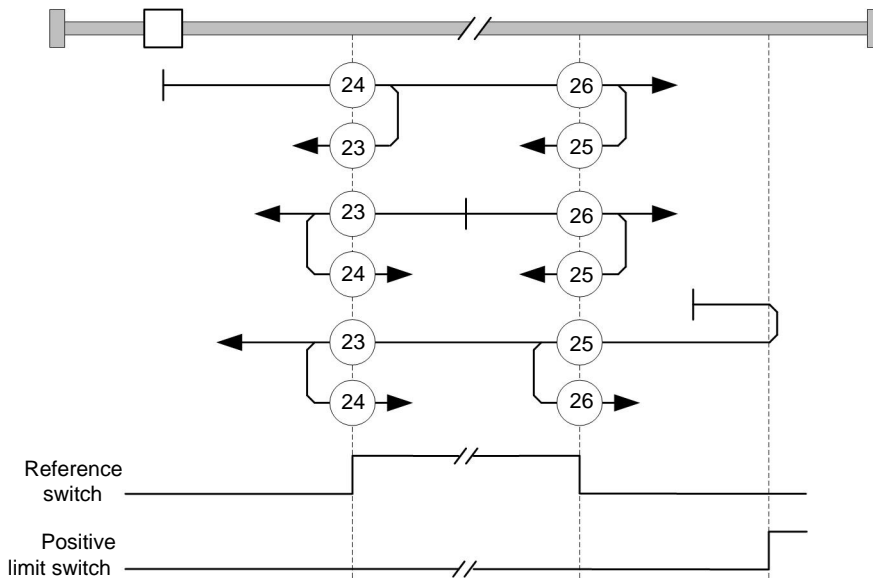


### Methods 23 to 30

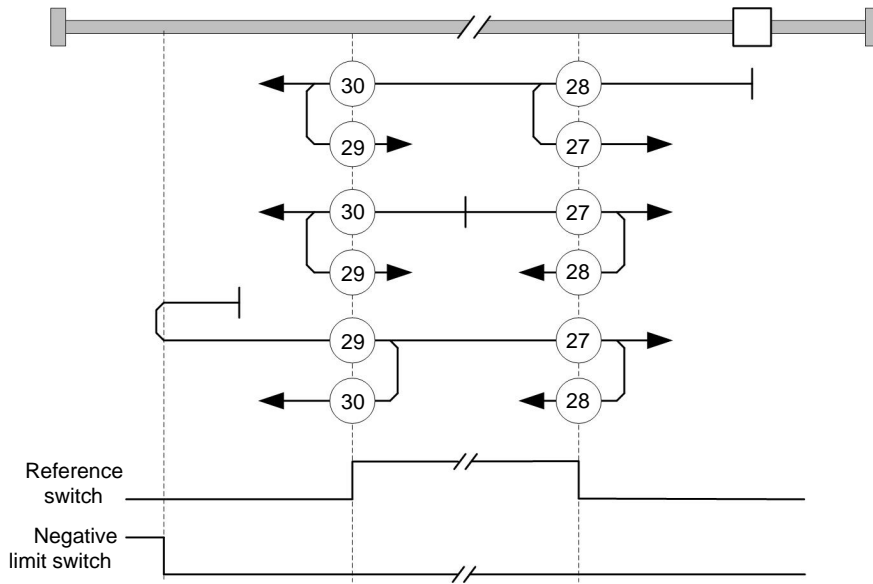
These methods reference the home switch without the index pulse (with limit switches).

For these methods, the actual position relative to the reference switch is unimportant. With method 26, referencing is for instance always to the index pulse on the right next to the right flank of the reference switch.

The methods 23 to 26 take the positive limit switch into account:



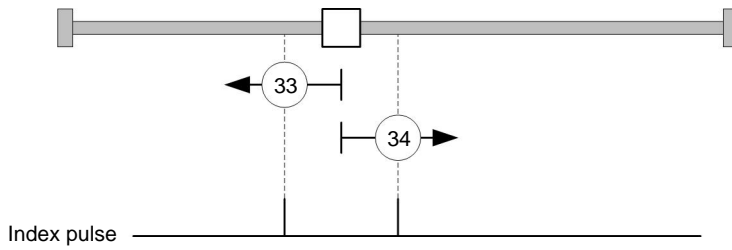
The methods 27 to 30 take the negative limit switch into account:



**Methods 33 and 34**

Reference the next index pulse.

For these methods, referencing is only to respective next index pulse:



**Method 35**

References to the actual position.

## 8 General concepts

### 8.1 DS402 Power State machine

#### 8.1.1 State machine

##### CANopen DS402

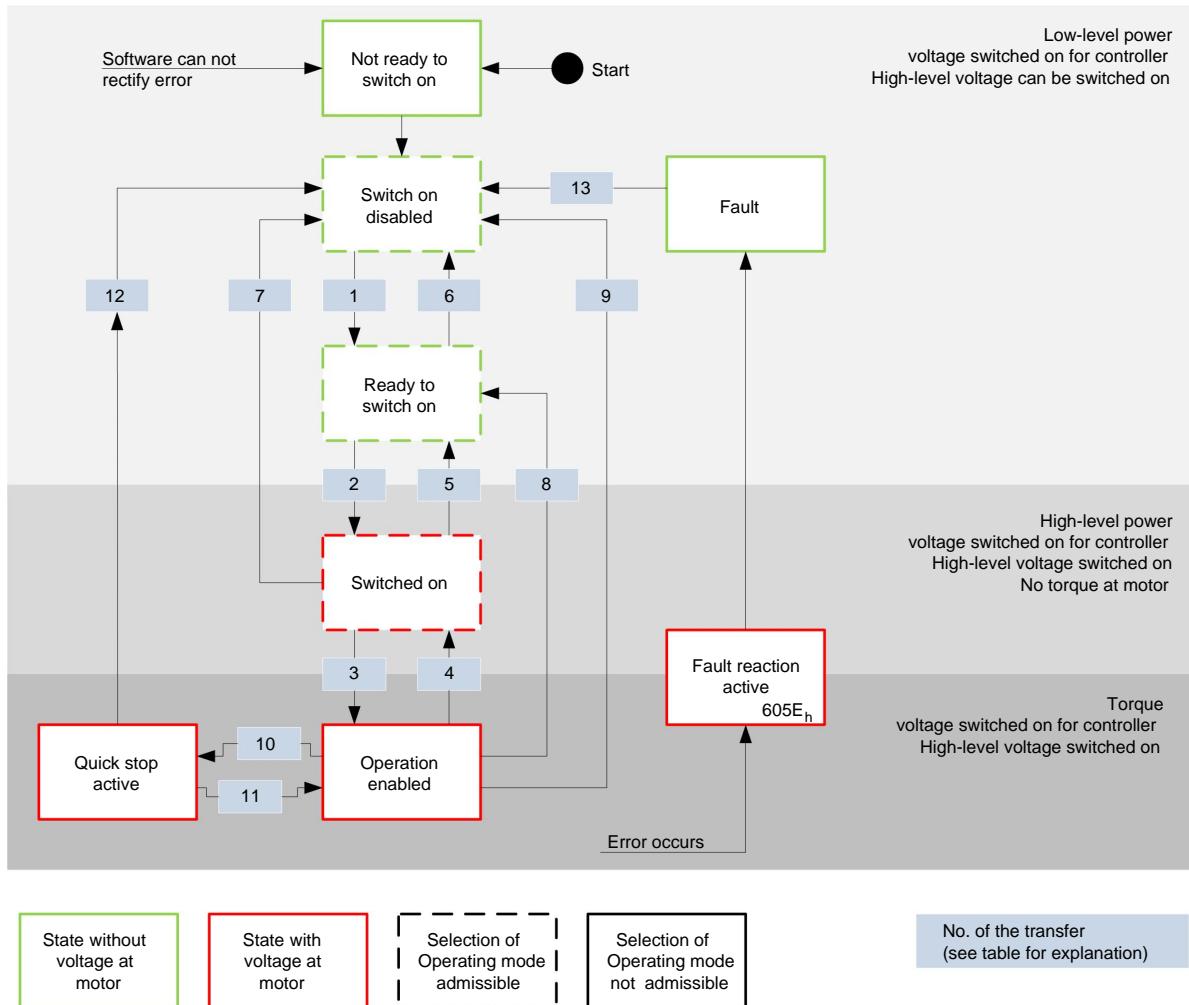
To switch the motor controller to an operational state, a state machine must be run through. This is defined in CANopen standard DS402. State changes are requested in object **6040<sub>h</sub>** (control word). The actual state of the state machine can be read out from object **6041<sub>h</sub>** (status word).

##### Control word


State changes are requested via object **6040<sub>h</sub>** (control word). The following **table** lists the bit combinations that lead to the corresponding state transitions.

##### State transitions

The diagram shows the possible state transitions.



The following **table** lists the bit combinations for the control word that lead to the corresponding state transitions. An X corresponds to a bit state that is no longer to be considered. The single exception is the fault reset: The transition is only requested by the rising flank of the bit.

Command	Bit in object 6040 <sub>h</sub>					Transition
	Bit 7	Bit 3	Bit 2	Bit 1	Bit 0	
Shutdown	0	X	1	1	0	1, 5, 8
Switch on	0	0	1	1	1	2
Disable voltage	0	X	X	0	X	6, 7, 9, 12
Quick stop	0	X	0	1	X	10
Disable operation	0	0	1	1	1	4
Enable operation	0	1	1	1	1	3, 11
Fault reset		X	X	X	X	13

### Status word

The following table lists the bit masks that describe the state of the motor controller.

Status word (6041 <sub>h</sub> )	State
xxxx xxxx x0xx 0000	Not ready to switch on
xxxx xxxx x1xx 0000	Switch on disabled
xxxx xxxx x01x 0001	Ready to switch on
xxxx xxxx x01x 0011	Switched on
xxxx xxxx x01x 0111	Operation enabled
xxxx xxxx x00x 0111	Quick stop active
xxxx xxxx x0xx 1111	Fault reaction active
xxxx xxxx x0xx 1000	Fault

The motor controller reaches the "Switch on" state after it is switched on and the self-test is successful.

### Operating mode

The set operating mode (6060<sub>h</sub>) becomes active in the "Operation enabled" state. The operating mode can only be set or changed in the following states (see states enclosed by a dashed line in the diagram):

- Switch on disabled
- Ready to switch on
- Switched on

During operation ("Operation enabled"), it is not possible to change the operating mode. The "Fault" state is left when bit 7 in object 6040<sub>h</sub> (control word) is set from "0" to "1" (rising flank).

**Note:** If an error that cannot be corrected occurs, the motor controller changes to the "Not ready to switch on" state and stays there. These errors include:

- Encoder error (e.g. due to missing shielding, cable breakage)

This state can also be reached through a bus error with the EtherCAT field bus type. In this case, the system automatically changes back to the "Switch on disabled" state after the bus error is eliminated.

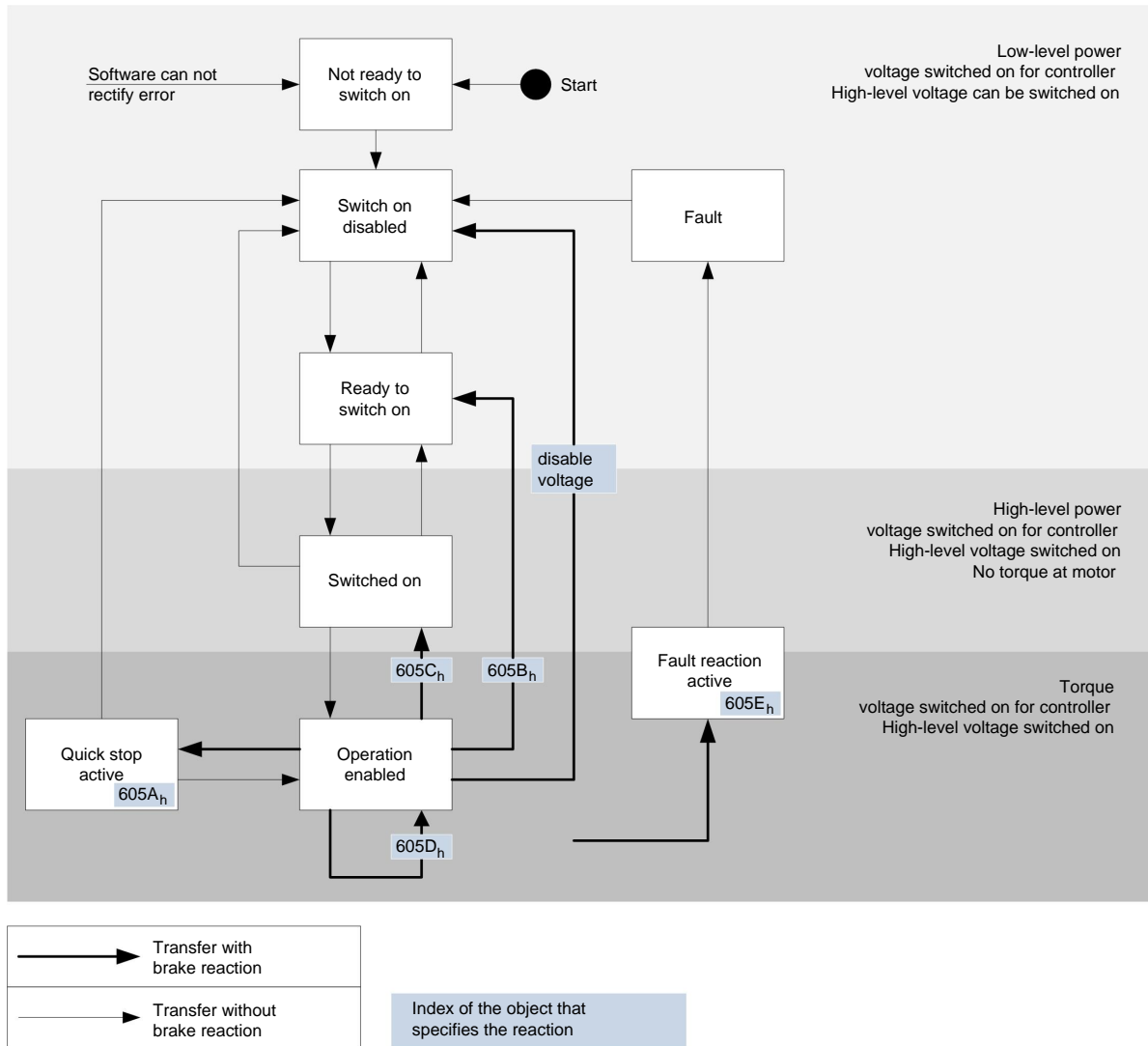
### 8.1.2 Behavior after the "Operation enabled" state is left

#### Brake reactions

Different brake reactions can be programmed when leaving the "Operation enabled" state.

These include the transitions described below.

The following diagram shows an overview of the brake reactions.



#### Quick stop active

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

In this case, the action stored in object **605A<sub>h</sub>** is executed (see the following table).

Value in object <b>605A<sub>h</sub></b>	Description
-32768 to -1	Reserved
0	Immediate stop with short-circuit braking
1	Braking with "slow down ramp" (deceleration 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"

Value in object 605A <sub>h</sub>	Description
3 to 32767	Reserved

### Ready to switch on

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

In this case, the action stored in object **605B<sub>h</sub>** is executed (see the following table).

Value in object 605B <sub>h</sub>	Description
-32768 to -1	Reserved
0	Immediate stop with short-circuit braking
1	Braking with "slow down ramp" (deceleration depending on operating mode) and subsequent state change to "Switch on disabled"
2 to 32767	Reserved

### Switched on

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

In this case, the action stored in object **605C<sub>h</sub>** is executed (see the following table).

Value in object 605C <sub>h</sub>	Description
-32768 to -1	Reserved
0	Immediate stop with short-circuit braking
1	Braking with "slow down ramp" (deceleration depending on operating mode) and subsequent state change to "Switch on disabled"
2 to 32767	Reserved

### Stop

Stop:

When bit 8 is set in object **6040<sub>h</sub>** (control word), the response stored in **605D<sub>h</sub>** is executed in velocity mode and profile velocity mode (see the following table).

Value in object 605D <sub>h</sub>	Description
-32768 to 0	Reserved
1	Braking with "slow down ramp" (deceleration depending on operating mode)
2	Braking with "quick stop ramp" (deceleration depending on operating mode)
3 to 32767	Reserved

### Fault

Fault:

If an error should occur, the motor is braked as stored in object **605E<sub>h</sub>**.



Value in object 605E <sub>h</sub>	Description
-32768 to -1	Reserved
0	Immediate stop with short-circuit braking
1	Braking with "slow down ramp" (deceleration depending on operating mode)
2	Braking with "quick stop ramp" (deceleration depending on operating mode)
3 to 32767	Reserved

## 8.2 User-defined units

### 8.2.1 Overview

#### Settings

The motor controller supports the possibility of setting user-defined units. In this way, the corresponding parameters can be set and read out directly in degrees, mm, etc.

#### Pole pair count compensation

Differences in the pole pair counts of motors can be compensated. For this purpose, the value in object **2060<sub>h</sub>** must be set to "1". Then the pole pair count automatically enters the subsequent computation so that different motors can be operated on the motor controller without requiring a new configuration.

### 8.2.2 Computation formulas for user units

#### Gear ratio

The gear ratio is calculated from the motor revolutions (**6091<sub>h:1</sub>** (Motor Revolutions)) per shaft revolution (**6091<sub>h:2</sub>** (Shaft Revolutions)) as follows:

$$\text{Gear ratio} = \frac{\text{Motor revolution (6091}_{h:1}\text{)}}{\text{Shaft revolution (6091}_{h:2}\text{)}}$$

If object **6091<sub>h:1</sub>** or object **6091<sub>h:2</sub>** are set to "0", the firmware sets the value to "1".

#### Feed constant

The feed constant is calculated from the feed (**6092<sub>h:1</sub>** (Feed Constant) per revolution of the drive axis (**6092<sub>h:2</sub>** (Shaft Revolutions)) as follows:

$$\text{Feed rate} = \frac{\text{Feed (6092}_{h:1}\text{)}}{\text{Revolution of the drive axis (6092}_{h:2}\text{)}}$$

This is useful to indicate the spindle pitch of a linear axis.

If object **6092<sub>h:1</sub>** or object **6092<sub>h:2</sub>** are set to "0", the firmware sets the value to "1".

#### Position

The current position in user units (**6064<sub>h</sub>**) is calculated as follows:

$$\text{Actual position} = \frac{\text{Internal position} \times \text{feed rate}}{\text{Encoder resolution} \times \text{gear ratio}}$$

## Speed

The speed specifications of the following objects can likewise be specified in user units:

Object	Mode	Meaning
<b>606B<sub>h</sub></b>	Profile Velocity Mode	Output value of ramp generator
<b>60FF<sub>h</sub></b>	Profile Velocity Mode	Speed specification
<b>6099<sub>h</sub></b>	Homing Mode	Speed for searching the index/switch
<b>6081<sub>h</sub></b>	Profile Position Mode	Target speed
<b>6082<sub>h</sub></b>	Profile Position Mode	End speed

The internal speed in mechanical revolutions per second is multiplied by a factor for numerator (**2061<sub>h</sub>**) and denominator (**2062<sub>h</sub>**). The speed in user units is computed from

$$\text{Speed} = \frac{\text{Internal speed} \times \text{numerator factor (2061}_h)}{\text{Denominator factor (2062}_h)}$$

If object **2061<sub>h</sub>** or **2062<sub>h</sub>** is to be set to "0", the firmware sets the value to "1".

## Acceleration

The acceleration can also be output in user units:

Object	Mode	Meaning
<b>609A<sub>h</sub></b>	Homing Mode	Acceleration
<b>6083<sub>h</sub></b>	Profile Position Mode	Acceleration
<b>6084<sub>h</sub></b>	Profile Position Mode	Deceleration
<b>60C5<sub>h</sub></b>	Profile Velocity Mode	Acceleration
<b>60C6<sub>h</sub></b>	Profile Position Mode	Deceleration
<b>6085<sub>h</sub></b>	"Quick stop active" state ( <b>DS402 Power State machine</b> )	Deceleration

The internal acceleration in mechanical revolutions per second squared is multiplied by a factor consisting of a numerator (**2063<sub>h</sub>**) and denominator (**2064<sub>h</sub>**).

$$\text{Acceleration} = \frac{\text{Internal acceleration} \times \text{numerator factor (2063}_h)}{\text{Denominator factor (2064}_h)}$$

If object **2063<sub>h</sub>** or **2064<sub>h</sub>** is to be set to "0", the firmware sets the value to "1".

## Jerk

For the jerk, objects **604A<sub>h</sub>:1<sub>h</sub>** to **604A<sub>h</sub>:4<sub>h</sub>** can be specified in user units. These objects only affect the Profile Position Mode and Profile Velocity Mode.

The objects **2065<sub>h</sub>** for the numerator and **2066<sub>h</sub>** for the denominator are available. The values of objects **604A<sub>h</sub>:1<sub>h</sub>** to **4<sub>h</sub>** are computed from the mechanical revolutions per second to the power of three multiplied by a numerator and denominator:

$$\text{Jerk} = \frac{\text{Internal value} \times \text{numerator factor (2065}_h)}{\text{Denominator factor (2066}_h)}$$

If object **2065<sub>h</sub>** or **2066<sub>h</sub>** is to be set to "0", the firmware sets the value to "1".

## Positional data

All positional values in the open loop and closed loop mode are specified in the resolution of the virtual position encoder. This is calculated from the encoder cycles (**608F<sub>h</sub>:1<sub>h</sub>** (encoder cycles)) per motor revolutions (**608F<sub>h</sub>:2<sub>h</sub>** (Motor Revolutions)) multiplied by the polarity of the axis in object **607E<sub>h</sub>** bit 0. If bit 0 in object **607E<sub>h</sub>** is set to the value "1", this corresponds to a polarity reversal, or the value "-1" in the formula:

$$\text{Resolution of the position encoder} = \text{polarity (607E}_h \text{ Bit 0)} \times \frac{\text{Encoder cycles (608F}_h:1)}{\text{Motor revolutions (608F}_h:2)}$$

If the value of **608F<sub>h</sub>:1<sub>h</sub>** or **608F<sub>h</sub>:2<sub>h</sub>** are set to "0", the motor controller continues computing internally with a "1". The factory settings are:

- Encoder cycles **608F<sub>h</sub>:1<sub>h</sub>** = "2000"
- Motor revolutions **608F<sub>h</sub>:2<sub>h</sub>** = "1"
- Polarity **607E<sub>h</sub>** bit 0 = "0" (does not correspond to a polarity reversal)

The resolution of the connected position encoder is set in object **2052<sub>h</sub>**.

## 9 Special functions

### 9.1 Digital inputs and outputs

The motor controller has digital inputs and outputs.

#### 9.1.1 Digital inputs

##### Two groups

The inputs are subdivided into two groups:

- Inputs 1 to 6 are designed for voltages of 0 V to 24 V with the switching thresholds at 11 V (on) and 9.5 V (off)
- Inputs 7 and 8 are wide-range inputs for voltages 0 V to 5 V- 24 V. The shift thresholds are at 3.0 V (on) and 2.5 V (off).

The state of an input is stored in entry **60FD<sub>h</sub>** as a bit value. Each input is assigned a bit position: Input 1 corresponds to bit 16 of object **60FD<sub>h</sub>**, input 2 corresponds to bit 17 of object **60FD<sub>h</sub>**, etc. (see the following table).

The inputs are evaluated every millisecond; reactions to changes to the inputs can occur in <2 ms.

Object <b>60FD<sub>h</sub></b>	Input / special function
Bit 0	Negative limit switch
Bit 1	Positive limit switch
Bit 2	Reference switch
Bit 16	Input 1
Bit 17	Input 2
Bit 18	Input 3
Bit 19	Input 4

##### Object entries

The following OD settings can be used to manipulate the value of an input, in which case only the bit that corresponds to that input will have an effect:

- **3240<sub>h</sub>:1<sub>h</sub>**

This bit is used to switch the special functions of an input on (value "0") or off (value "1"). If input 1 is not to be used as a negative limit switch, for example, the special function must be switched off so that the signal encoder is not erroneously responded to. The object has no effects on bits 16 to 31.

The firmware evaluates the following bits during a reference run (homing method):

- Bit 0: negative limit switch
- Bit 1: positive limit switch
- Bit 2: reference switch

- **3240<sub>h</sub>:2<sub>h</sub>**

This bit changes from closer logic (a logical high level at the pin yields the value of "1" in object **60FD<sub>h</sub>**) to opener logic (the logical high level at the pin yields the value of "0"). This applies to the special functions (except the clock and directional inputs) and for the normal input pins.

- **3240<sub>h</sub>:3<sub>h</sub>**

This bit switches on software simulation of the input values when it is set to "1". In this case, the actual values are no longer used; instead, the values set in object **3240<sub>h</sub>:4<sub>h</sub>** for the respective input are used.

- **3240<sub>h</sub>:4<sub>h</sub>**

This bit specifies the value to be read in as input value if the same bit was set in object **3240<sub>h</sub>:3<sub>h</sub>**.

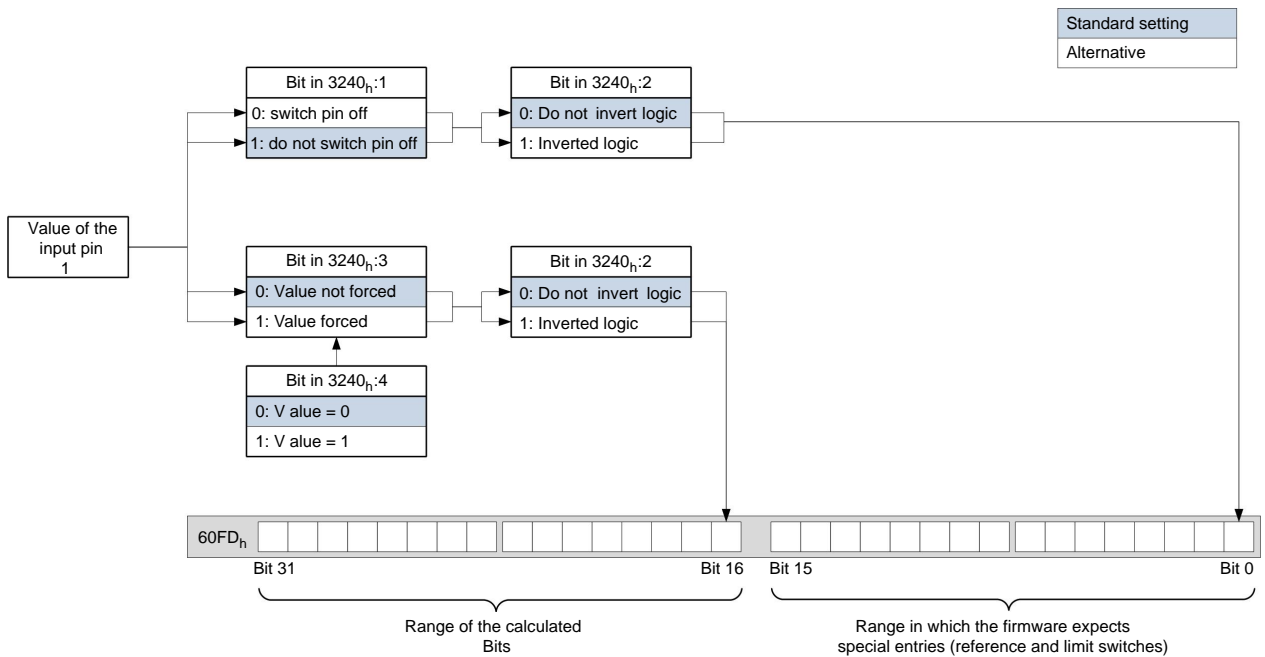
- **3240<sub>h</sub>:5<sub>h</sub>**

This object contains the unmodified input value.

### Computation of the input pin

Computation of the input pin using pin 1 as an example:

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



## 9.1.2 Digital outputs

### Outputs

The outputs are actuated via object **60FE<sub>h</sub>**. Output 1 corresponds to bit 16 in object **60FE<sub>h</sub>**, output 2 corresponds to bit 17, etc., as with the inputs. The outputs with special functions are again entered in the firmware in the lower bits 0 to 15. Currently only bit 0 is assigned that controls the motor brake.

### Object entries

Additional OD entries exist for manipulating the value of the outputs (see the following example for details). Similar to the inputs, only the bit at the corresponding position always has an effect on the respective output:

- **3250<sub>h</sub>:2<sub>h</sub>**

This can be used to change the logic from "closer" to "opener". When configured as a "closer", the pin outputs a logical high level if the bit is "1". When configured as an "opener", the pin outputs a logical low level if there is a "1" in object **60FE<sub>h</sub>**.

- **3250<sub>h</sub>:3<sub>h</sub>**

If a bit is set in **3250<sub>h</sub>**, the output is manually controlled. The value for the output is then contained in object **3250<sub>h</sub>:4<sub>h</sub>**, which is also possible for the brake output.

- **3250<sub>h</sub>:4<sub>h</sub>**

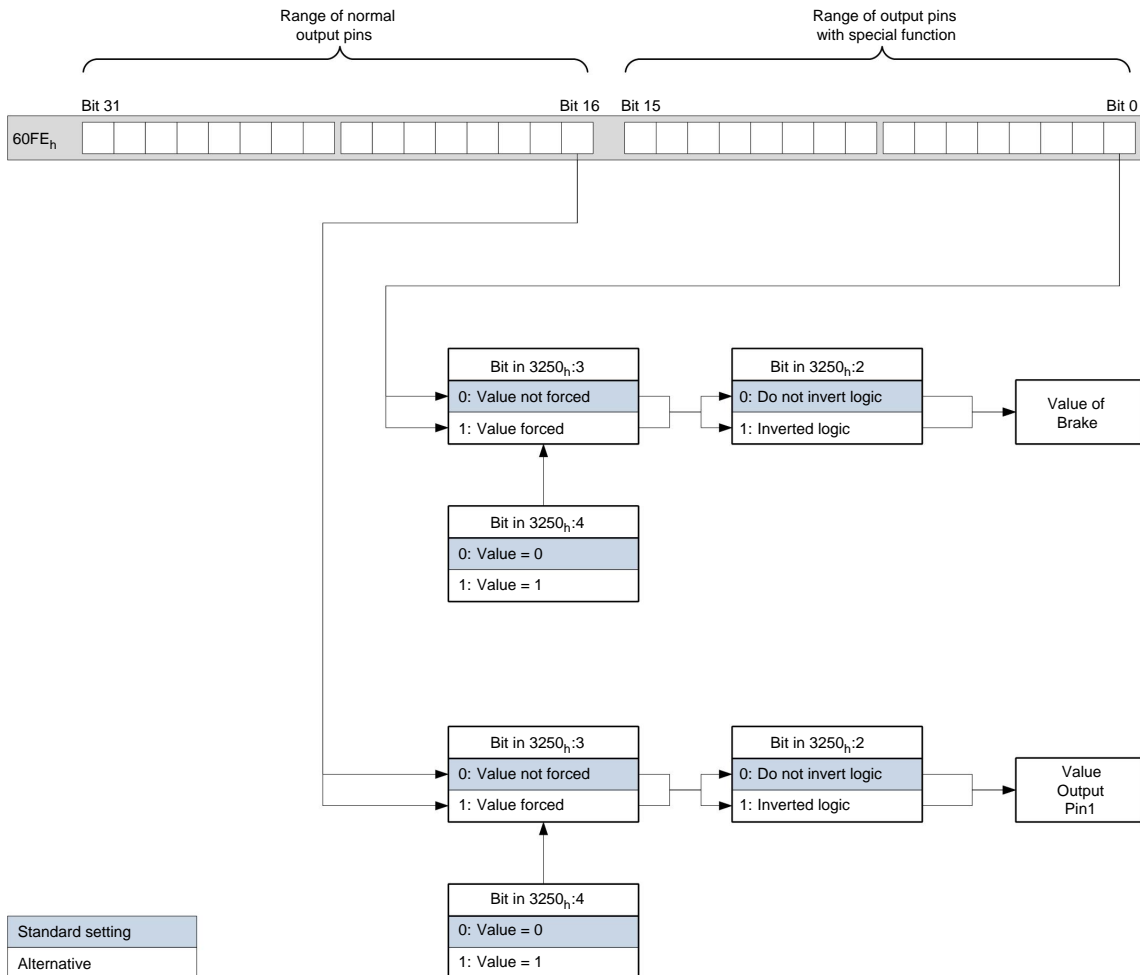
The bits in this object specify the output value that is to be applied to the output pin when the manual motor controller of the output pin is activated by object **3250<sub>h</sub>:3<sub>h</sub>**.

- **3250<sub>h</sub>:5<sub>h</sub>**

This object does not have any function and is included for compatibility reasons.

### Bits of the output pin

Example of the computation of the bits for the output pin:



## 9.2 I<sup>2</sup>t motor overload protection

### 9.2.1 Description

I<sup>2</sup>t motor overload protection has the goal of preventing damage to the motor and of simultaneously operating it normally at its thermal limit.

The function is only available when the motor controller is in closed loop operating mode (bit 0 of object **3202<sub>h</sub>** set to "1") and the motor is **not** in profile torque mode or cycle synchronous torque.

There is a single exception: If I<sup>2</sup>t is activated open loop mode, the current is limited to the set nominal current even when the set maximum current is greater. This feature was implemented for safety reasons, so that it is also possible to switch out of closed loop mode with a very high short-time maximum current into open loop mode without damaging the motor.

### 9.2.2 Object entries

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

- **2031<sub>h</sub>**: Peak Current - specifies the maximum current in mA.
- **203B<sub>h</sub>:1<sub>h</sub>** Nominal Current - specifies the nominal current in mA.
- **203B<sub>h</sub>:2<sub>h</sub>** Maximum Duration Of Peak Current - specifies the maximum time period of the maximum current in ms.

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

- **203B<sub>h</sub>:3<sub>h</sub>** Threshold - specifies the limit in mA, from which is determined whether switching is to the maximum current or nominal current.
- **203B<sub>h</sub>:4<sub>h</sub>** CalcValue - specifies the calculated value that is compared to the threshold in order to set the current.
- **203B<sub>h</sub>:5<sub>h</sub>** LimitedCurrent - shows the actual current value that was set by  $I^2t$ .
- **203B<sub>h</sub>:6<sub>h</sub>** Status:
  - Value = "0":  $I^2t$  deactivated
  - Value = "1":  $I^2t$  activated

### 9.2.3 Activation

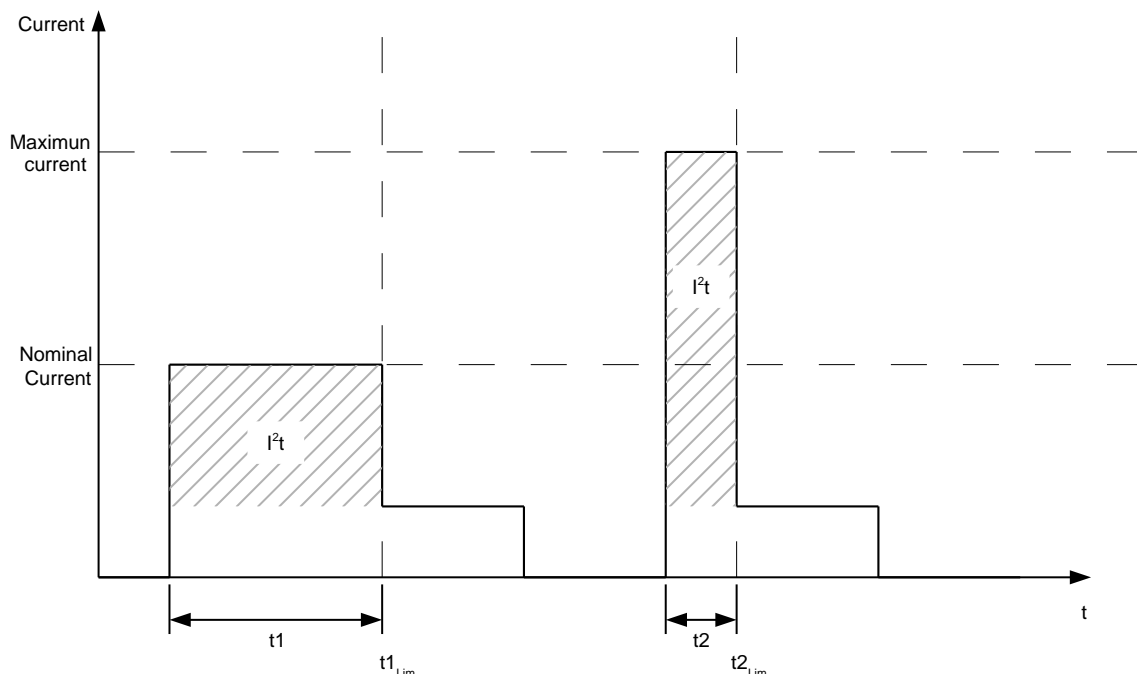
The three object entries above must have been appropriately specified to activate the mode. This means that the maximum current must be greater than the nominal current, and a time value must be entered for the maximum time of the maximum current.  $I^2t$  operability remains deactivated when these conditions are not satisfied.

### 9.2.4 Function of $I^2t$

A  $I^2T_{Lim}$  is calculated by specifying the nominal current, maximum current, and maximum time period for the maximum current.

The motor can run with maximum current until the calculated  $I^2T_{Lim}$  is reached. The current is then immediately reduced to the nominal current.

The following diagram again shows the interactions.



In the first section  $t_1$ , the current value is higher than the nominal current. At time  $t_{1Lim}$ ,  $I^2t_{Lim}$  is reached and the current is limited to the nominal current. During the following time period  $t_2$ , a current comes that corresponds to the maximum current. Accordingly, the value for  $I^2t_{Lim}$  is reached faster than in time period  $t_1$ .

## 10 Programming with NanoJ

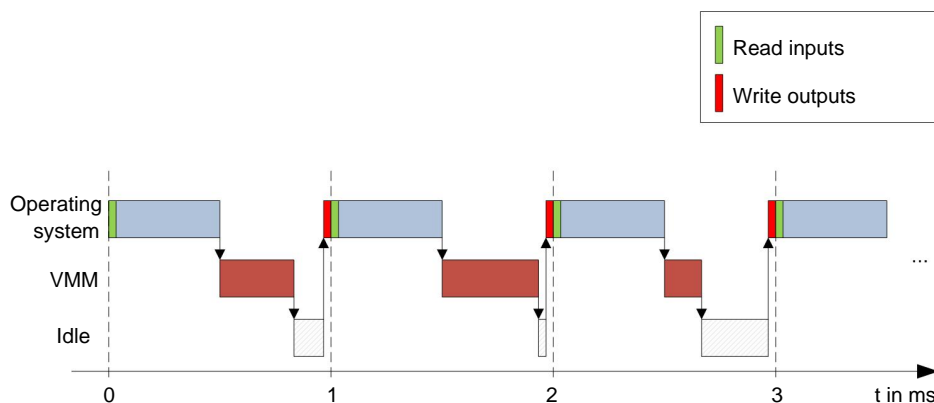
### 10.1 Introduction

The VMM (Virtual Machine Monitor) is a protected execution environment within the firmware. The user can load his or her own programs ("User Program") in this environment. These can trigger functions in the motor controller, for example by reading or writing entries in the object directory.

The use of protective mechanisms makes it impossible for the user programs to cause the actual firmware to crash. In the worst case, the user program alone is aborted with an error code stored in the object directory.

### 10.2 Available computing time

A user program receives computing time in a 1-ms cycle (see also the following diagram). Because the firmware loses computing time due to interrupts and system functions, only about 30% - 50% of this time is available to the user program (depending on the operating mode and application case). During this time, the user program must have completed its operations and must either have closed or have yielded the computing time with the `yield()` function. In the first case, the user program is started again when the next 1-ms cycle begins; in the second case, the program is continued at the command following the `yield()` in the next 1-ms cycle.



If the system detects that the user program requires more than the time assigned to it, it is closed and an error code is entered in the object directory. When developing user programs, therefore, the runtime behavior of the program must be carefully checked, especially in the case of time-intensive tasks. Therefore, it is advisable to use tables, instead of calculating a sinus value from a `sin` function.

#### Note

If the VMM program should not return the computing time for an excessive time, it is ended by the operating system. In this case, the number "4" is entered in the status word at object **2301<sub>h</sub>** of the VMM, the number "5" (timeout) is noted in the VMM error register at object **2302<sub>h</sub>**.

### 10.3 Interaction of the user program with the motor controller

#### 10.3.1 Communication options

A user program has numerous options for communicating with the motor controller:

- Reading and writing of OD values per PDO mapping
- Direct reading and writing of OD values via system calls
- Calling up of other system calls (e.g. write debug output)



Via a PDO mapping, OD values in the form of variables are made available to the user program. Before a user program receives its 1-ms time slot, the firmware transfer the values for this from the OD to the variables of the user program. When the user program now receives computing time, it can manipulate these variables like the usual C variables. At the end of the time slot, the new values are automatically copied into the respective OD entries by the firmware.

To optimize the performance, 3 types of mappings are defined: Input, output and input/output (In, Out, InOut). Input mappings can only be read and are not transferred back into the OD. Output mappings can only be written. Input/Output Mappings, on the other hand, permit reading and writing.

The set mappings can be read out and checked via the web interface at objects **2310<sub>h</sub>**, **2320<sub>h</sub>**, and **2330<sub>h</sub>**. For each mapping, a maximum of 16 entries is allowed.

The specification of the linker section is used to control in NanoJEasy whether a variable is stored the under input, output, or data range.

### 10.3.2 Execution of a VMM cycle

In summary, the procedure for the execution of a VMM cycle with respect to the PDO mapping consists of the following three steps:

1. Read values from the object directory and copy them into the Inputs and Outputs areas.
2. Execute the user program.
3. Copy values from the Outputs and Outputs areas back to the object directory.

The configuration of the copy procedures is in line with the CANopen standard.

In addition, it is also possible to access system calls via the object directory. In general, this is considerably slower and therefore mappings should be given preference. However, the number of mappings is limited (16 entries each in In/Out/InOut). Therefore, it is advisable to map frequently-used and changed OD values and to access less frequently used OD entries by system call. A list of available system calls can be found in the "**System calls**" section.

#### Note

It is strongly advised to access one single OD value **either** by mapping **or** system call with `od_write()`. If both are used at the same time, system call will not have any effect.

## 10.4 OD entries for controlling and configuring the VMM

### 10.4.1 OD entries

The VMM is controlled and configured by means of OD entries in the object range **2300<sub>h</sub>** to **2330<sub>h</sub>**. The web interface handles a large portion of the task, which is why it is usually not necessary for the user to directly access the entries.

OD Index	Name
<b>2300<sub>h</sub></b>	VMM Control (Read/write)
<b>2301<sub>h</sub></b>	VMM Status (Read only)
<b>2302<sub>h</sub></b>	VMM Error Code (Read only)
<b>2303<sub>h</sub></b>	Number Of Active User Program (Read/Write)

OD Index	Name
2304 <sub>h</sub>	Table of available user programs
2310 <sub>h</sub>	VMM Input Data Selection
2320 <sub>h</sub>	VMM Output Data Selection
2330 <sub>h</sub>	VMM In/output Data Selection

### 10.4.2 Example

To select and start the "TEST1.USR" user program, the following sequence can be used for instance:

- Upload the program file into the "config" directory via the Ethernet connection
- Write the value "54453554<sub>h</sub>" in object 2304<sub>h</sub>:01<sub>h</sub>.
- Write the value "31000000<sub>h</sub>" in object 2304<sub>h</sub>:02<sub>h</sub>.
- Write the value "1<sub>h</sub>" in object 2303<sub>h</sub>.
- Check the entry 2302<sub>h</sub> for error codes.
- If no error:

Launch the program by writing object 2300<sub>h</sub>, bit 0 = "1".

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

To stop a running program: Write the bit 0-value = "0" to the entry 2300<sub>h</sub>.

## 10.5 NanoJEasyV2

### 10.5.1 Installation and use

#### Introduction

As an alternative to NanoIP, a user program can also be programmed, uploaded, and controlled with the NanoJEasyV2 software.

#### Installation

Proceed as follows for installation:

1. Unpack "NanoJEasyV2.zip" into a directory of your choice.
2. Launch the program with the file "NanoJEasy.exe".

### 10.5.2 Programming of user programs

#### User program structure

A user program consists of at least two instructions:

1. The `#include "wrapper.h"` preprocessor instruction
2. The `void user(){}` function

The code to be executed can then be stored in the `void user()` function.

The file names of the user programs must not be longer than eight characters and contains three characters in the extension; for example, "main.cpp" is admissible while "alongerfilename.cpp" is not.

#### Example

Programming a rectangular output in object 2500<sub>h</sub>:01<sub>h</sub>

1. Copy the following text to the NanoJEasy editor and store this file under the name "main.cpp".

```
// 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
```

2. When the program has been properly translated:

Rename the output file "main.usr" to "vmmcode.usr".

3. Use USB to copy the file to the motor controller (see the "**USB port**" section). The motor controller must be restarted to launch the program, please read the "**VMM program**" section starting at step 2 for details.

### 10.5.3 Structure of a mapping

#### Introduction

This method can be used to directly link a variable in the VMM program with an entry in the object directory. The mapping must be created at the beginning of the file - before the #include "wrapper.h" instruction. Only a comment above the mapping is allowed.

**Tip** Use mapping if you frequently need access to an object in the object directory, such as the control word **6040<sub>h</sub>** or status word **6041<sub>h</sub>**.

The `od_write()` and `od_read()` functions are better suited for single access to objects (see the "**Access to the object directory**" section)

#### Declaration of the mapping

The declaration of the mapping is structured as follows:

```
map <TYPE> <NAME> as <input|output|inout> <INDEX>:<SUBINDEX>
```

The following applies:

- <TYPE>  
The data type of the variable, i.e. U32, U16, U08, S32, S16 or S08.
- <NAME>  
The name of the variable that is later used in the user program.
- <input|output|inout>

The write and read authorization of a variable: A variable can either be declared as in, out, or inout. This defines whether a variable is readable (input), writable (output) or both (inout) and the structure by which it needs to be addressed in the program.

- <INDEX> : <SUBINDEX>

Index and subindex of the object being mapped in the object directory.

Every 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.

### Example of a mapping

Example of a mapping and the associated variable access methods:

```
map U16 controlWord as output 0x6040:00
map U08 statusWord as input 0x6041:00
map U08 modeOfOperation as inout 0x6060:00

#include "wrapper.h"

void user()
{
[...]
```

Out.controlWord = 1;

U08 tmpVar = In.statusword;

InOut.modeOfOperation = tmpVar;

```
[...]
}
```

### Potential error source

A potential source of error is a write access by means of the `od_write()` function on an object in the object directory that was also created as a mapping. The code shown below is **faulty**:

```
map U16 controlWord as output 0x6040:00
#include " wrapper.h"
void user()
{
[...]
```

Out.controlWord = 1;

```
[...]
od_write(0x6040, 0x00, 5 ); // the value is overwritten by the mapping
[...]
```

The line with the command `od_write(0x6040, 0x00, 5 );` is without effect. As described in the introduction, all mappings are copied into the object directory at the end of each millisecond.

The following procedure is therefore derived:

- The function `od_write` writes the value "5" in object **6040<sub>h</sub>:00<sub>h</sub>**.
- At the end of the 1-ms cycle the mapping is written that also specifies object **6040<sub>h</sub>:00<sub>h</sub>**, though with the value "1".
- This means - from the user's perspective - the `od_write` command is without effect.

## 10.6 System calls

### 10.6.1 Introduction

With system calls, it is possible to call up functions integrated in the firmware directly in a user program. Because a direct code execution is only possible in the protected area of the sandbox, this is implemented via so-called Cortex-Supervisor-Calls (Svc Calls). An interrupt is triggered when the function is called and the firmware thus has the possibility of temporarily allowing a code execution

outside of the sandbox. Developers of user programs do not need to worry about this mechanism. For them, the system calls can be called up like normal C functions. Only the "wrapper.h" file must be integrated as usual.

### 10.6.2 Access to the object directory

- void **od\_write** (U32 index, U32 subindex, U32 value)

This function writes the transferred value to the specified point in the object directory.

index	Index of the object being written in the object directory
subindex	Subindex of the object being written in the object directory
value	Value to be written

#### Note

It is strongly advised, to generate processor time with `yield()` after a `od_write()` has been called up. The value is immediately written to the OD. However, to enable the firmware to trigger dependent actions, it must receive computing time and therefore the user program must have been ended or stopped with `yield()`.

- void **od\_read** (U32 index, U32 subindex)

This function reads the value at the specified point in the object directory and returns it.

index	Index of the object being read in the object directory
subindex	Subindex of the object being read in the object directory
Return value	Content of the OD entry

#### Note

Active waiting for a value in the object directory should always be associated with a `yield()`.

#### Example:

```
while (od_read(2400,2) != 0) // wait until 2400:2 is set
{ yield(); }
```

### 10.6.3 Process control

- void **yield**

This function returns the process time to the operating system. The program is resumed in the next time slot at the same location.

- void **sleep**(U32 ms)

This function returns the process time to the operating system for the specified number of milliseconds. The user program is then continued at the location following the call.

ms	Wait time in milliseconds
----	---------------------------

### 10.6.4 Debug output

The following functions output a value in the debug console. They differ only in the data type of the parameter being output.

- bool **VmmDebugOutputString**(const char \*outstring)
- bool **VmmDebugOutputInt**(const U32 val)

- bool **VmmDebugOutputByte**(const U08 val)
- bool **VmmDebugOutputHalfWord**(const U16 val)
- bool **VmmDebugOutputWord**(const U32 val)
- bool **VmmDebugOutputFloat**(const Woat val)

#### Note

The debug outputs are first written to a separate area of the OD and are read out from there by the web interface. This OD entry has the index **2600<sub>h</sub>** and is 64 characters long. The subindex 0 always contains the number of characters already written.

If the buffer is full, `VmmDebugOutputxxx( )` initially fails; execution of the user program is discontinued and it stops at the location of the debug output. The program is not resumed until the web interface has read out the buffer and reset the subindex 0; `VmmDebugOutputxxx( )` returns to the user program.

Debug outputs therefore may only be used during the test phase in the development of a user program.

# 11 Object directory description

## 11.1 Overview

You can find a description of objects in this section of the manual.

Here you will find information on the following:

- Functions
- Object descriptions ("Index")
- Value descriptions ("Subindices")
- Descriptions of bits
- Description of the object

## 11.2 Structure of the object description

The description of object entries is always structured the same and normally consists of the following sections:

### Function

This section briefly describes the function of the object directory.

### Object description

This table gives detailed information on the data type, specified values, and suchlike. A detailed description can be found in the "**Object description**" section

### Value description

This table is only available for the "Array" or "Record" data type and gives detailed information on the subentries. A more detailed description of entries can be found in the "**Value description**" section

### Description

More precise information on the single bits in an entry is given here or any compositions are explained. A more detailed description can be found in the "**Description**" section

## 11.3 Object description

The object description consists of a table that contains the following entries:

### Index

Designates the index of the object 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 one variable that is indexed with subindex 0.
- ARRAY: This objects always consist of one subindex 0 - that specifies the quantity of subentries - and the subentries themselves from index 1. The data type in an array never changes, this means subentry 1 and all following entries always have the same data type.
- RECORD: These objects always consist of one subentry with subindex 0 - that specifies the quantity of subentries - and the subentries themselves from index 1. As opposed to an

ARRAY, the data type of subentries may vary, this means for example, that subentry 1 may have a different data type than subentry 2.

- **VISIBLE\_STRING**: The object specifies a character string encoded in ASCII. These character strings are **not** terminated by a zero string.

### Data type

The size and interpretation of the object are specified here. The following notation applies for the "VARIABLE" object code:

- Distinction is drawn between entries that are signed, this is designated with the prefix "SIGNED". The prefix "UNSIGNED" is used for unsigned entries.
- The size of the variable in bits is added to the prefix and can be either 8, 16 or 32.

### Firmware version

The firmware version is entered here from which the object is available.

### Change history (ChangeLog)

Any changes to the object are noted here.

Additionally, there are the following table entries for the "VARIABLE" data type entries:

#### Access

The access restriction is entered here. The following restrictions are available:

- "read/write": The object can be read and written
- "read only": The object can only be read from the object directory. It is not possible to set a value.

#### PDO mapping

Some bus systems, such as CANopen or EtherCAT, support PDO mapping. This table entry specifies whether the object may be inserted in a mapping, and in which. The following designations are possible:

- "no": The object may not be entered in any mapping.
- "TX-PDO": The object may in be entered in a RX mapping.
- "RX-PDO": The object may in be entered in a TX mapping.

#### Admissible values

In some cases, it is only permitted to write specific values into the object. When this is the case, these values are listed here. The field remains empty when there is no restriction.

#### Specified value

Some objects must be preassigned with values to bring the motor controller into a safe state at switch on. The value written into the object for the motor controller start is noted in this table entry.

## 11.4 Value description

### Note

For reasons of clarity, some subentries have been summarized here when all the entries have the same name.

All data for subentries with subindex 1 or higher are listed in the table with the heading "Value description". The table contains the following entries:



**Subindex**

Number of the currently specified subentry.

**Name**

The name of the subentry.

**Data type**

The size and interpretation of the subentry are specified here. The following notation always applies:

- Distinction is drawn between entries that are signed, this is designated with the prefix "SIGNED". The prefix "UNSIGNED" is used for unsigned entries.
- The size of the variable in bits is added to the prefix and can be either 8, 16 or 32.

**Access**

The access restriction for the subentry is entered here. The following restrictions are available:

- "read/write": The object can be read and written
- "read only": The object can only be read from the object directory. It is not possible to set a value.

**PDO mapping**

Some bus systems, such as CANopen or EtherCAT, support PDO mapping. This table entry specifies whether the subentry may be inserted in a mapping, and in which. The following designations are possible:

- "no": The object may not be entered in any mapping.
- "TX-PDO": The object may in be entered in a RX mapping.
- "RX-PDO": The object may in be entered in a TX mapping.

**Admissible values**

In some cases, it is only permitted to write specific values into the subentry. When this is the case, these values are listed here. The field remains empty when there is no restriction.

**Specified value**

Some objects must be preassigned with subentries to bring the motor controller into a safe state at switch on. The value written into the subentry for the motor controller start is noted in this table entry.

**11.5 Description**

This section can be available when use requires additional information. When single bits of an object or subentry have a different meaning, diagrams are used as shown in the following example.

**Example:** The object is 8-bits large, bit 0 and 1 separately have one function. Bits 2 and 3 have been combined into one function, the same applies for bits 4 to 7.

7	6	5	4	3	2	1	0
Example [4]			Example [2]		B	A	

**Example [4]**

Description of bits 4 to including 7, these bits logically belong together. The 4 in square brackets specifies the number of associated bits. A list of possible values and their description is frequently attached at this position.

### Example [2]

Description of bits 3 and 2, these bits logically belong together. The 2 in square brackets specifies the number of associated bits.

- Value 00<sub>b</sub>: The description at this position applies when bit 2 and bit 3 are at "0".
- Value 01<sub>b</sub>: The description at this position applies when bit 2 is at "0" and bit 3 at "1".
- Value 10<sub>b</sub>: The at this position applies when bit 2 is at "1" and bit 3 at "0".
- Value 11<sub>b</sub>: The description at this position applies when bit 2 and bit 3 are at "1".

### B

Description of bit B, there is no length information for a single bit.

### A

Description of bit A, bits with a gray background remain unused.

## 1000h Device Type

### Function

Describes the motor controller type.

### Object description

Index	1000 <sub>h</sub>
Object Name	Device Type
Object Code	VARIABLE
Data type	UNSIGNED32
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00040192 <sub>h</sub>
Firmware Version	
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.

### Device profile number[16]

Describes the supported CANopen standard.

Values:

0129<sub>h</sub> (specified value): The DS402 standard is supported.

## 1001h Error Register

### Function

Error register: In the event of error the corresponding error bit is set. It is automatically deleted when the error no longer exists.

### Object description

Index	1001 <sub>h</sub>
Object Name	Error Register
Object Code	VARIABLE
Data type	UNSIGNED8
Access	read only
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	00 <sub>h</sub>
Firmware Version	
Change History	

### Description

7	6	5	4	3	2	1	0
RES	RES	PROF	COM	TEMP	VOL	CUR	GEN

#### GEN

General error

#### CUR

Current

#### VOL

Voltage

#### TEMP

Temperature

#### COM

Communication

#### PROF

Pertains to the device profile

#### RES

Reserved, always "0"

## 1003h Pre-defined Error Field

### Function

This object contains an error stack with up to eight entries.

**Object description**

Index	1003 <sub>h</sub>
Object Name	Pre-defined Error Field
Object Code	ARRAY
Data type	UNSIGNED32
Firmware Version	
Change History	Amount of subentries has changed from 2 to 9

**Value description**

Subindex	00 <sub>h</sub>
Name	Number Of Errors
Data type	UNSIGNED8
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	Standard Error Field
Data type	UNSIGNED32
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

Subindex	02
Name	Standard Error Field
Data type	UNSIGNED32
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

Subindex	03
Name	Standard Error Field
Data type	UNSIGNED32
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	04
Name	Standard Error Field
Data type	UNSIGNED32
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	05
Name	Standard Error Field
Data type	UNSIGNED32
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	06
Name	Standard Error Field
Data type	UNSIGNED32
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	07
Name	Standard Error Field
Data type	UNSIGNED32
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	08
Name	Standard Error Field
Data type	UNSIGNED32
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

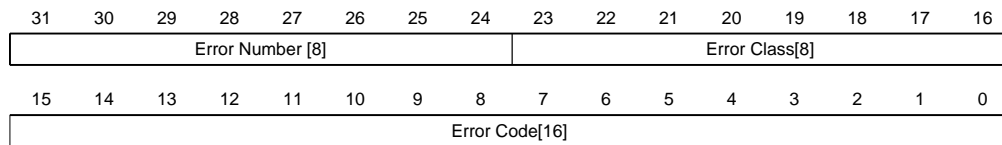
## Description

### General operation

If a new error occurs, it is entered in subindex 1. The existing entries in the subindices 1 to 7 are shifted back by one. The error at subindex 7 is removed.

The number of errors that have occurred can be read from the object with subindex 0. When a "0" is written into this object, counting starts anew.

### Bit description



### Error Number [8]

This allows the reason for the error to be fully narrowed down. The meaning of the number can be found in the following table.

Error number	Description
1	Input voltage too high
2	Output current too high
3	Input voltage too low
4	Field bus error
5	Motor rotating in wrong direction – despite activated block
6	CANopen only: NMT master is taking too long to send Nodeguarding request
7	Encoder error due to electrical fault or faulty hardware
8	Encoder error; index not found during auto setup
9	Error in AB track
10	Positive limit switch and tolerance zone overwritten
11	Negative limit switch and tolerance zone overwritten
12	Device temperature above 80 °C
13	The values of object <b>6065<sub>h</sub></b> (Following Error Window) and object <b>6066<sub>h</sub></b> (Following Error Time Out) have been exceeded, a fault has been output. This fault must be activated with bit 7 in object <b>3202<sub>h</sub></b> .

### Error Class[8]

This byte is identical to object **1001<sub>h</sub>**

### Error Code[16]

The meaning of the two bytes can be seen in the following table.

Error Code	Description
1000 <sub>h</sub>	General error
2310 <sub>h</sub>	Voltage at output of motor controller too high

Error Code	Description
3210 <sub>h</sub>	Overvoltage at input of motor controller
3220 <sub>h</sub>	Undervoltage at input of motor controller
4200 <sub>h</sub>	Temperature error in motor controller
7305 <sub>h</sub>	Incremental sensor 1 faulty
8000 <sub>h</sub>	Field bus monitoring error
8130 <sub>h</sub>	Only CANopen: Life guard error or heartbeat - error
8611 <sub>h</sub>	Position monitoring error: Subsequent error too large
8612 <sub>h</sub>	Position monitoring error: Reference limit

## 1008h Manufacturer Device Name

### Function

Contains the device name as a string.

### Object description

Index	1008 <sub>h</sub>
Object Name	Manufacturer Device Name
Object Code	VARIABLE
Data type	VISIBLE_STRING
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	
Firmware Version	
Change History	

### Description

The length of the string appears in subindex 0 of this object. The individual characters are contained as of subindex 1. The character string is not terminated by a zero string.

## 1009h Manufacturer Hardware Version

### Function

This object contains the hardware version as a string.

### Object description

Index	1009 <sub>h</sub>
Object Name	Manufacturer Hardware Version
Object Code	VARIABLE
Data type	VISIBLE_STRING

Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	
Firmware Version	
Change History	

### Description

The length of the string appears in subindex 0 of this object. The individual characters are contained as of subindex 1. The character string is not terminated by a zero string.

## 100Ah Manufacturer Software Version

### Function

This object contains the software version as a string.

### Object description

Index	100A <sub>h</sub>
Object Name	Manufacturer Software Version
Object Code	VARIABLE
Data type	VISIBLE_STRING
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	DEV-B29286
Firmware Version	
Change History	

### Description

The length of the string appears in subindex 0 of this object. The individual characters are contained as of subindex 1. The character string is not terminated by a zero string.

## 1010h Store Parameter

### Function

Permanently stores the NodeID from object **2009<sub>h</sub>** and the baud rate setting from object **2005<sub>h</sub>**, so they are again available after a restart or power failure. The numerical value 65766173<sub>h</sub> must be written in subindex 1 to store the values.

#### Note

The values are **only stored**. The stored settings only become valid after one of the following conditions:

- Switch off and on of the voltage supply
- Sending of a "Reset Communication" CANopen message



**Note**

- Sending of a "Reset Node" CANopen message

**Attention:**

**Object description**

Index	1010 <sub>h</sub>
Object Name	Store Parameter
Object Code	ARRAY
Data type	UNSIGNED32
Firmware Version	
Change History	

**Value description**

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	02 <sub>h</sub>
Subindex	01 <sub>h</sub>
Name	Save All The Parameters To Non-volatile Memory
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	02 <sub>h</sub>
Name	Save The Comm Parameters To Non-volatile Memory
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

**Description**

This object has the purpose of storing the values from the object directory for the NodeID (object **2005<sub>h</sub>**) and baud rate (object **2009<sub>h</sub>**) for CANopen after a change.

The value 65766173<sub>h</sub> must be written in subindex 2 in order to start the store process. This corresponds to decimal 1702257011<sub>d</sub> or the ASCII string "SAVE".

## 1011h Restore Default Parameter

### Function

This object can be used to reset the entire object directory to the default values.

### Object description

Index	1011 <sub>h</sub>
Object Name	Restore Default Parameter
Object Code	ARRAY
Data type	UNSIGNED32
Firmware Version	
Change History	

### Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	01 <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	Restore All Default Parameters
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

### Description

When value 64616F6C<sub>h</sub> is written into this object, the entire object directory is reset to the default values.

#### Note

For the reset to take effect, the motor controller reboots at the end.

## 1018h Identity Object

### Function

The object contains information on the manufacturer, the product code and the revision and serial numbers.

### Object description

Index	1018 <sub>h</sub>
Object Name	Identity Object
Object Code	RECORD
Data type	IDENTITY
Firmware Version	
Change History	

### Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	04 <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	Vendor-ID
Data type	UNSIGNED32
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	0000026C <sub>h</sub>

Subindex	02 <sub>h</sub>
Name	Product Code
Data type	UNSIGNED32
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00000003 <sub>h</sub>

Subindex	03 <sub>h</sub>
Name	Revision Number
Data type	UNSIGNED32
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00000001 <sub>h</sub>

Subindex	04 <sub>h</sub>
Name	Serial Number
Data type	UNSIGNED32
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00000001 <sub>h</sub>

## 1600h Drive Control

### Function

This object contains an input PDO mapping of the active field bus (EtherCAT or CANopen).

### Object description

Index	1600 <sub>h</sub>
Object Name	Drive Control
Object Code	RECORD
Data type	PDO_MAPPING
Firmware Version	
Change History	

### Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	1st Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

Subindex	02 <sub>h</sub>
Name	2nd Object To Be Mapped
Data type	UNSIGNED32

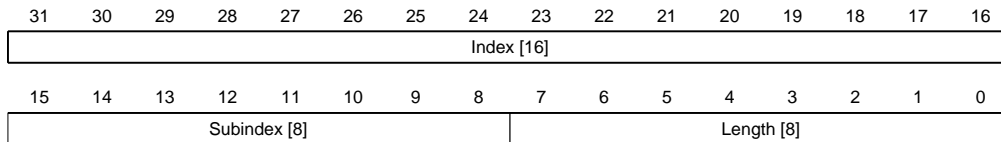
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	03 <sub>h</sub>
Name	3rd Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	04 <sub>h</sub>
Name	4th Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	05 <sub>h</sub>
Name	5th Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	06 <sub>h</sub>
Name	6th Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	07 <sub>h</sub>
Name	7th Object To Be Mapped
Data type	UNSIGNED32
Access	read/write

PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
<hr/>	
Subindex	08 <sub>h</sub>
Name	8th Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

**Description**

Each subindex (1-8) describes a mapped object.

A mapping entry consists of four bytes made up according to the following graphic.



**Index [16]**

Contains the index of the object to be mapped

**Subindex [8]**

Contains the subindex of the object to be mapped

**Length [8]**

Contains the length of the object to be mapped in the bit unit.

**1601h Positioning Control**

**Function**

This object contains an input PDO mapping of the active field bus (EtherCAT or CANopen).

**Object description**

Index	1601 <sub>h</sub>
Object Name	Positioning Control
Object Code	RECORD
Data type	PDO_MAPPING
Firmware Version	
Change History	

**Value description**

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

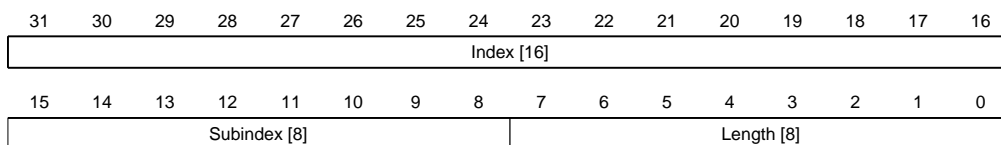
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
Subindex	01 <sub>h</sub>
Name	1st Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	02 <sub>h</sub>
Name	2nd Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	03 <sub>h</sub>
Name	3rd Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	04 <sub>h</sub>
Name	4th Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	05 <sub>h</sub>
Name	5th Object To Be Mapped

Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
<hr/>	
Subindex	06 <sub>h</sub>
Name	6th Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
<hr/>	
Subindex	07 <sub>h</sub>
Name	7th Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
<hr/>	
Subindex	08 <sub>h</sub>
Name	8th Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

**Description**

Each subindex (1-8) describes a mapped object.

A mapping entry consists of four bytes made up according to the following graphic.



**Index [16]**

Contains the index of the object to be mapped

**Subindex [8]**

Contains the subindex of the object to be mapped



**Length [8]**

Contains the length of the object to be mapped in the bit unit.

**1602h Velocity Control**

**Function**

This object contains an input PDO mapping of the active field bus (EtherCAT or CANopen).

**Object description**

Index	1602 <sub>h</sub>
Object Name	Velocity Control
Object Code	RECORD
Data type	PDO_MAPPING
Firmware Version	
Change History	

**Value description**

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	1st Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

Subindex	02 <sub>h</sub>
Name	2nd Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	03 <sub>h</sub>
Name	3rd Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	04 <sub>h</sub>
Name	4th Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	05 <sub>h</sub>
Name	5th Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	06 <sub>h</sub>
Name	6th Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	07 <sub>h</sub>
Name	7th Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	08 <sub>h</sub>
----------	-----------------

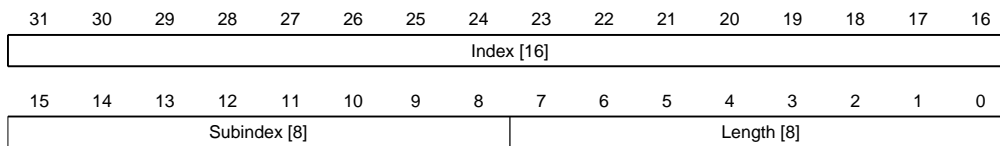
---

Name	8th Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

### Description

Each subindex (1-8) describes a mapped object.

A mapping entry consists of four bytes made up according to the following graphic.



#### Index [16]

Contains the index of the object to be mapped

#### Subindex [8]

Contains the subindex of the object to be mapped

#### Length [8]

Contains the length of the object to be mapped in the bit unit.

## 1603h Output Control

### Function

This object contains an input PDO mapping of the active field bus (EtherCAT or CANopen).

### Object description

Index	1603 <sub>h</sub>
Object Name	Output Control
Object Code	RECORD
Data type	PDO_MAPPING
Firmware Version	
Change History	

### Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read/write
PDO Mapping	no
Admissible Values	

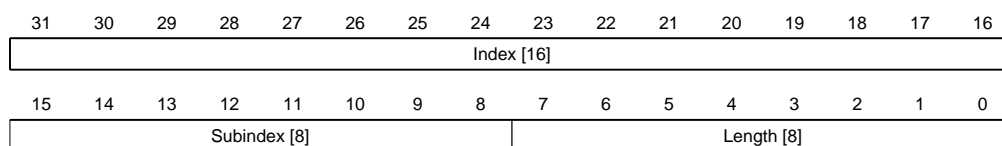
Specified Value	00 <sub>h</sub>
Subindex	01 <sub>h</sub>
Name	1st Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	02 <sub>h</sub>
Name	2nd Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	03 <sub>h</sub>
Name	3rd Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	04 <sub>h</sub>
Name	4th Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	05 <sub>h</sub>
Name	5th Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

Subindex	06 <sub>h</sub>
Name	6th Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	07 <sub>h</sub>
Name	7th Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	08 <sub>h</sub>
Name	8th Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

### Description

Each subindex (1-8) describes a mapped object.

A mapping entry consists of four bytes made up according to the following graphic.



#### Index [16]

Contains the index of the object to be mapped

#### Subindex [8]

Contains the subindex of the object to be mapped

#### Length [8]

Contains the length of the object to be mapped in the bit unit.

## 1A00h Drive Status

### Function

This object specifies an output PDO mapping of the active field bus (EtherCAT or CANopen).

### Object description

Index	1A00h
Object Name	Drive Status
Object Code	RECORD
Data type	PDO_MAPPING
Firmware Version	
Change History	

### Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>

Subindex	01 <sub>h</sub> to 08 <sub>h</sub>
Name	1st Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

Subindex	02 <sub>h</sub>
Name	2nd Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

Subindex	03 <sub>h</sub>
Name	3rd Object To Be Mapped
Data type	UNSIGNED32

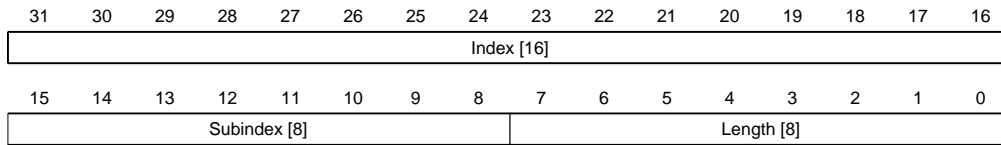
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	04 <sub>h</sub>
Name	4th Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	05 <sub>h</sub>
Name	5th Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	06 <sub>h</sub>
Name	6th Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	07 <sub>h</sub>
Name	7th Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	08 <sub>h</sub>
Name	8th Object To Be Mapped
Data type	UNSIGNED32
Access	read/write

PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

**Description**

Each subindex (1-8) describes a mapped object.

A mapping entry consists of four bytes made up according to the following graphic.



**Index [16]**

Contains the index of the object to be mapped

**Subindex [8]**

Contains the subindex of the object to be mapped

**Length [8]**

Contains the length of the object to be mapped in the bit unit.

**1A01h Positioning Status**

**Function**

This object specifies an output PDO mapping of the active field bus (EtherCAT or CANopen).

**Object description**

Index	1A01h
Object Name	Positioning Status
Object Code	RECORD
Data type	PDO_MAPPING
Firmware Version	
Change History	

**Value description**

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>

---

Subindex	01 <sub>h</sub>
----------	-----------------



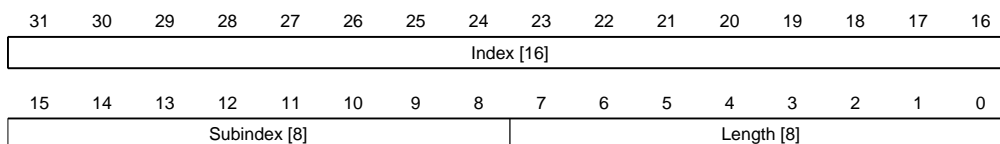
Name	1st Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	02 <sub>h</sub>
Name	2nd Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	03 <sub>h</sub>
Name	3rd Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	04 <sub>h</sub>
Name	4th Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	05 <sub>h</sub>
Name	5th Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	06 <sub>h</sub>
Name	6th Object To Be Mapped

Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
<hr/>	
Subindex	07 <sub>h</sub>
Name	7th Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
<hr/>	
Subindex	08 <sub>h</sub>
Name	8th Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

### Description

Each subindex (1-8) describes a mapped object.

A mapping entry consists of four bytes made up according to the following graphic.



#### Index [16]

Contains the index of the object to be mapped

#### Subindex [8]

Contains the subindex of the object to be mapped

#### Length [8]

Contains the length of the object to be mapped in the bit unit.

## 1A02h Velocity Status

### Function

This object specifies an output PDO mapping of the active field bus (EtherCAT or CANopen).

**Object description**

Index	1A02h
Object Name	Velocity Status
Object Code	RECORD
Data type	PDO_MAPPING
Firmware Version	
Change History	

**Value description**

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	1st Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

Subindex	02 <sub>h</sub>
Name	2nd Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

Subindex	03 <sub>h</sub>
Name	3rd Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	04 <sub>h</sub>
Name	4th Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	05 <sub>h</sub>
Name	5th Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	06 <sub>h</sub>
Name	6th Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	07 <sub>h</sub>
Name	7th Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

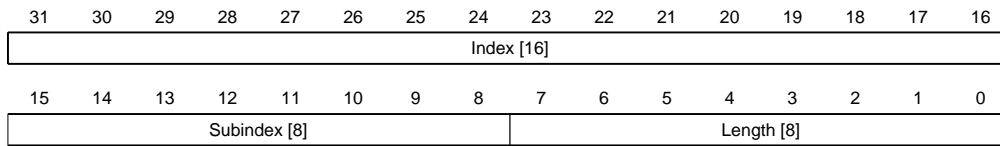
Subindex	08 <sub>h</sub>
Name	8th Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

## Description

Each subindex (1-8) describes a mapped object.

A mapping entry consists of four bytes made up according to the following graphic.



### Index [16]

Contains the index of the object to be mapped

### Subindex [8]

Contains the subindex of the object to be mapped

### Length [8]

Contains the length of the object to be mapped in the bit unit.

## 1A03h Input Status

### Function

This object specifies an output PDO mapping of the active field bus (EtherCAT or CANopen).

### Object description

Index	1A03h
Object Name	Input Status
Object Code	RECORD
Data type	PDO_MAPPING
Firmware Version	
Change History	

### Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
<hr/>	
Subindex	01 <sub>h</sub>
Name	1st Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no

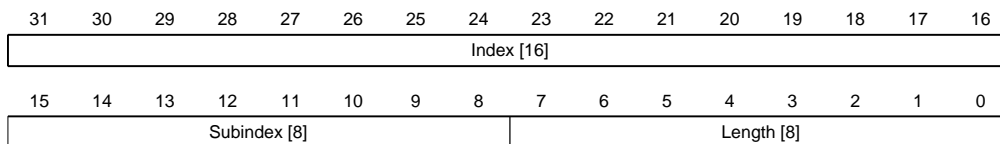
Admissible Values	
Specified Value	00000000 <sub>h</sub>
<hr/>	
Subindex	02 <sub>h</sub>
Name	2nd Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
<hr/>	
Subindex	03 <sub>h</sub>
Name	3rd Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
<hr/>	
Subindex	04 <sub>h</sub>
Name	4th Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
<hr/>	
Subindex	05 <sub>h</sub>
Name	5th Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
<hr/>	
Subindex	06 <sub>h</sub>
Name	6th Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	

Specified Value	00000000 <sub>h</sub>
Subindex	07 <sub>h</sub>
Name	7th Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	08 <sub>h</sub>
Name	8th Object To Be Mapped
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

### Description

Each subindex (1-8) describes a mapped object.

A mapping entry consists of four bytes made up according to the following graphic.



### Index [16]

Contains the index of the object to be mapped

### Subindex [8]

Contains the subindex of the object to be mapped

### Length [8]

Contains the length of the object to be mapped in the bit unit.

## 2010h IP-Configuration

### Function

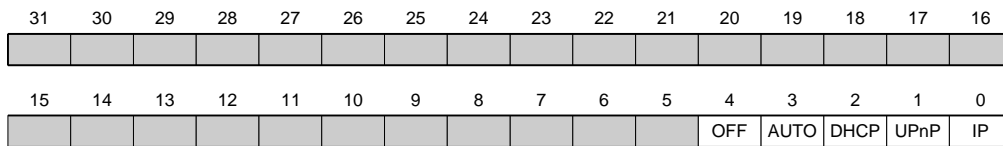
This object is used to set the network configuration.

### Object description

Index	2010 <sub>h</sub>
Object Name	IP-Configuration
Object Code	VARIABLE

Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Firmware Version	
Change History	

### Description



#### IP

Value = "1": A static IP address is used from object **2011<sub>h</sub>** and network mask from object **2012<sub>h</sub>** is used.

#### UPnP

Value = "1": UPnP (Universal Plug and Play) notifications will be activated

#### DHCP

Value = "1": IP address allocation by a DHCP server will be activated

#### AUTO

Value = "1": IP address allocation by the AUTO-IP protocol will be activated

#### OFF

Value = "1": Network interface will be deactivated

## 2011h Static-IP-Address

### Function

Contains the static IPv4 address in the form of a 32-bit word.

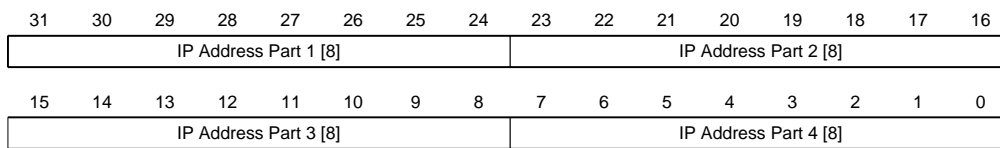
### Object description

Index	2011 <sub>h</sub>
Object Name	Static-IP-Address
Object Code	VARIABLE
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Firmware Version	



## Change History

### Description



#### IP Address Part 1 [8]

Specifies the first part of the IP address

#### IP Address Part 2 [8]

Specifies the second part of the IP address

#### IP Address Part 3 [8]

Specifies the third part of the IP address

#### IP Address Part 4 [8]

Specifies the fourth part of the IP address

### Example

The address 192.168.2.0 is first converted to the hexadecimal notation and then yields the following configuration value:

192 => C0<sub>h</sub>

168 => A8<sub>h</sub>

2 => 02<sub>h</sub>

0 => 0

The corresponding adjustment value is then C0A80200<sub>h</sub>.

## 2012h Static-IP-Subnet-Mask

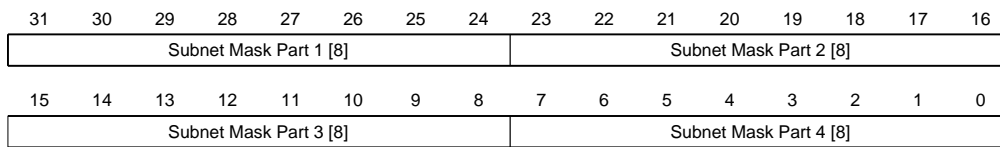
### Function

Contains the subnet mask of the static IP address in the form of a 32-bit word.

### Object description

Index	2012 <sub>h</sub>
Object Name	Static-IP-Subnet-Mask
Object Code	VARIABLE
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Firmware Version	
Change History	

## Description



### Subnet Mask Part 1 [8]

Specifies the first part of the subnet mask

### Subnet Mask Part 2 [8]

Specifies the second part of the subnet mask

### Subnet Mask Part 3 [8]

Specifies the third part of the subnet mask

### Subnet Mask Part 4 [8]

Specifies the fourth part of the subnet mask

### Example

The class C network mask 255 . 255 . 255 . 0 is first converted to the hexadecimal system and then yields the following configuration value:

255 => FF<sub>h</sub>

0 => 0

The corresponding adjustment value is then FFFFFFFF00<sub>h</sub>.

## 2018h Current-IP-Address

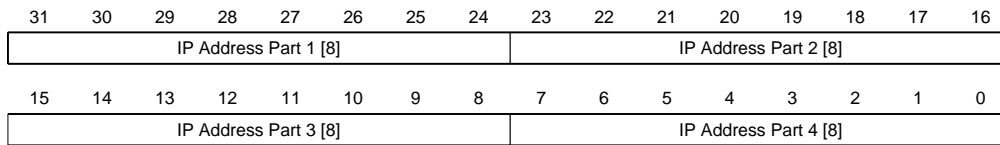
### Function

Contains the currently active IP address in the form of a 32-bit word.

### Object description

Index	2018 <sub>h</sub>
Object Name	Current-IP-Address
Object Code	VARIABLE
Data type	UNSIGNED32
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Firmware Version	
Change History	

## Description



### IP Address Part 1 [8]

Specifies the first part of the IP address

### IP Address Part 2 [8]

Specifies the second part of the IP address

### IP Address Part 3 [8]

Specifies the third part of the IP address

### IP Address Part 4 [8]

Specifies the fourth part of the IP address

### Example

The address 192.168.2.0 is first converted to the hexadecimal notation and then yields the following configuration value:

192 => C0<sub>h</sub>

168 => A8<sub>h</sub>

2 => 02<sub>h</sub>

0 => 0

The corresponding adjustment value is then C0A80200<sub>h</sub>.

## 2019h Current-IP-Subnet-Mask

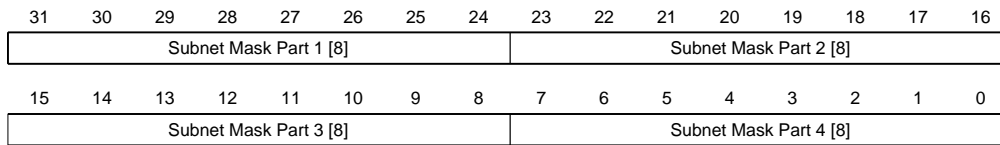
### Function

Contains the currently active subnet mask of the static IP address in the form of a 32-bit word.

### Object description

Index	2019 <sub>h</sub>
Object Name	Current-IP-Subnet-Mask
Object Code	VARIABLE
Data type	UNSIGNED32
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Firmware Version	
Change History	

## Description



### Subnet Mask Part 1 [8]

Specifies the first part of the subnet mask

### Subnet Mask Part 2 [8]

Specifies the second part of the subnet mask

### Subnet Mask Part 3 [8]

Specifies the third part of the subnet mask

### Subnet Mask Part 4 [8]

Specifies the fourth part of the subnet mask

### Example

The class C network mask 255 . 255 . 255 . 0 is first converted to the hexadecimal system and then yields the following configuration value:

255 => FF<sub>h</sub>

0 => 0

The corresponding adjustment value is then FFFFFFFF00<sub>h</sub>.

## 2020h ApplInfo-Static-IP-Address

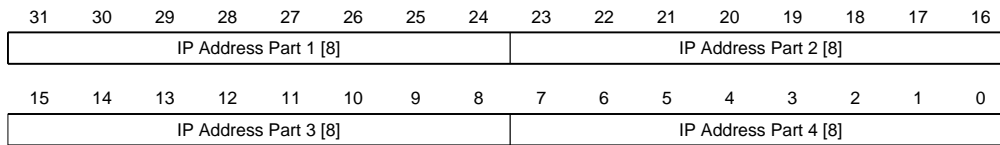
### Function

Contains the IP address specified in Nanoflash in the form of a 32-bit word.

### Object description

Index	2020 <sub>h</sub>
Object Name	AppInfo-Static-IP-Address
Object Code	VARIABLE
Data type	UNSIGNED32
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Firmware Version	
Change History	

## Description



### IP Address Part 1 [8]

Specifies the first part of the IP address

### IP Address Part 2 [8]

Specifies the second part of the IP address

### IP Address Part 3 [8]

Specifies the third part of the IP address

### IP Address Part 4 [8]

Specifies the fourth part of the IP address

### Example

The address 192.168.2.0 is first converted to the hexadecimal notation and then yields the following configuration value:

192 => C0<sub>h</sub>

168 => A8<sub>h</sub>

2 => 02<sub>h</sub>

0 => 0

The corresponding adjustment value is then C0A80200<sub>h</sub>.

## 2021h AppInfo-Static-IP-Subnet-Mask

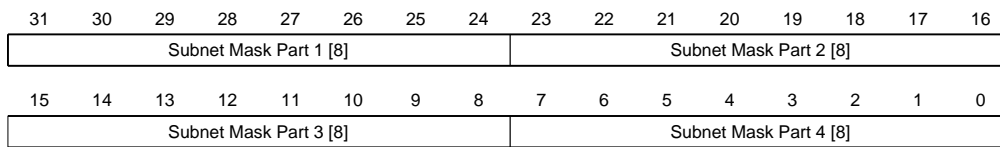
### Function

Contains the subnet mask of the static IP address specified in Nanoflash in the form of a 32-bit word.

### Object description

Index	2021 <sub>h</sub>
Object Name	AppInfo-Static-IP-Subnet-Mask
Object Code	VARIABLE
Data type	UNSIGNED32
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Firmware Version	
Change History	

## Description



### Subnet Mask Part 1 [8]

Specifies the first part of the subnet mask

### Subnet Mask Part 2 [8]

Specifies the second part of the subnet mask

### Subnet Mask Part 3 [8]

Specifies the third part of the subnet mask

### Subnet Mask Part 4 [8]

Specifies the fourth part of the subnet mask

### Example

The class C network mask 255 . 255 . 255 . 0 is first converted to the hexadecimal system and then yields the following configuration value:

255 => FF<sub>h</sub>

0 => 0

The corresponding adjustment value is then FFFFFFFF00<sub>h</sub>.

## 2022h Drive Serial Number

### Function

This object contains the serial number of the motor controller.

### Object description

Index	2022 <sub>h</sub>
Object Name	Drive Serial Number
Object Code	VARIABLE
Data type	VISIBLE_STRING
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	
Firmware Version	
Change History	

## 2030h Pole Pair Count

### Function

Contains the pole pair count of the connected motor.

### Object description

---

Index	2030 <sub>h</sub>
Object Name	Pole Pair Count
Object Code	VARIABLE
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000032 <sub>h</sub>
Firmware Version	
Change History	

---

## 2031h Peak Current

### Function

Specifies the maximum current in mA.

### Object description

---

Index	2031 <sub>h</sub>
Object Name	Peak Current
Object Code	VARIABLE
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	000009C4 <sub>h</sub>
Firmware Version	
Change History	

---

## 2032h Maximum Speed

### Function

Specifies the maximum admissible speed of the v-control in revolutions/s or rpm.

### Object description

---

Index	2032 <sub>h</sub>
-------	-------------------

---

Object Name	Maximum Speed
Object Code	VARIABLE
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00030D40 <sub>h</sub>
Firmware Version	
Change History	

### Description

The conversion is based on the numerator and denominator specified in object **604C<sub>h</sub>**.

## 2033h Plunger Block

### Function

Specifies the maximum positional change in user units (corresponding to Target Position **607A<sub>h</sub>**) that is permitted in the corresponding direction.

### Object description

Index	2033 <sub>h</sub>
Object Name	Plunger Block
Object Code	VARIABLE
Data type	INTEGER32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Firmware Version	
Change History	

### Description

This is used to implement an electronic lock.

The value 0 switches the monitoring of.

For example, the value 100 means that the drive may move in the negative direction by any distance, but as soon as it moves in the positive direction by more than 100 steps the motor is stopped immediately and an error is output.

For example, when winding up threads, this can be used to prevent an accidental unwinding of threads.



## 2034h Upper Voltage Warning Level

### Function

This object holds the threshold level for the "Overvoltage" error in volts.

### Object description

---

Index	2034 <sub>h</sub>
Object Name	Upper Voltage Warning Level
Object Code	VARIABLE
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	0000C92C <sub>h</sub>
Firmware Version	
Change History	

---

### Description

If the input voltage of the motor controller rises above this threshold value, the motor is switched off and an error is output. This error is automatically reset when the input voltage is less than (voltage of the object 2036<sub>h</sub> minus 2 volts).

## 2035h Lower Voltage Warning Level

### Function

This object holds the threshold level for the "Undervoltage" error in volts.

### Object description

---

Index	2035 <sub>h</sub>
Object Name	Lower Voltage Warning Level
Object Code	VARIABLE
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00004650 <sub>h</sub>
Firmware Version	
Change History	

---

### Description

If the input voltage of the motor controller drops below this threshold value, the motor is switched off and an error is output. This error is automatically reset when the input voltage is less than (voltage of the object 2035<sub>h</sub> plus 2 volts).

## 2036h Open Loop Current Reduction Idle Time

### Function

This object specifies the time in milliseconds for which the motor must be idling before the current reduction is activated.

### Object description

---

Index	2036 <sub>h</sub>
Object Name	Open Loop Current Reduction Idle Time
Object Code	VARIABLE
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	000003E8 <sub>h</sub>
Firmware Version	
Change History	

---

## 2037h Open Loop Current Reduction Value/factor

### Function

This object specifies the value to which the current must be reduced when current reduction is activated in open loop.

### Object description

---

Index	2037 <sub>h</sub>
Object Name	Open Loop Current Reduction Value/factor
Object Code	VARIABLE
Data type	INTEGER32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	FFFFFFCE <sub>h</sub>
Firmware Version	
Change History	

---

### Description

If the value is negative between "-100" and "-1", this is interpreted as the percentage reduction factor relative to the maximum current ( **2031<sub>h</sub>**). The value -100 corresponds to 100% of the value in object **2031<sub>h</sub>**, and the value -50 is interpreted as 50% of the value of object **2031<sub>h</sub>**, etc.

If the value is positive, the current is reduced by the value in mA entered in object **2037<sub>h</sub>**.

## 2038h Brake Controller Timing

### Function

This object contains the times for the brake controller in milliseconds as well as the PWM frequency and duty cycle.

### Object description

Index	2038 <sub>h</sub>
Object Name	Brake Controller Timing
Object Code	ARRAY
Data type	UNSIGNED32
Firmware Version	
Change History	

### Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	06 <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	Close Brake Idle Time
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	000003E8 <sub>h</sub>

Subindex	02 <sub>h</sub>
Name	Shutdown Power Idle Time
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	000003E8 <sub>h</sub>

Subindex	03 <sub>h</sub>
Name	Open Brake Delay Time
Data type	UNSIGNED32

Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	000003E8 <sub>h</sub>
Subindex	04 <sub>h</sub>
Name	Start Operation Delay Time
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	000003E8 <sub>h</sub>
Subindex	05 <sub>h</sub>
Name	PWM Frequency
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	between 1 and 2000 (7D0 <sub>h</sub> )
Specified Value	00000000 <sub>h</sub>
Subindex	06 <sub>h</sub>
Name	PWM Duty Cycle
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	between 2 and 100
Specified Value	00000000 <sub>h</sub>

### Description

The subindices have the following functions:

- 01<sub>h</sub>: Time between when the motor begins idle and the brake closes.
- 02<sub>h</sub>: Time between when the brake closes and the current is lowered.
- 03<sub>h</sub>: Time between when a new move command is set and the brake is opened.
- 04<sub>h</sub>: Time between when the brake is opened and the motor starts.
- 05<sub>h</sub>: Frequency of the brake PWM in Herz.
- 06<sub>h</sub>: Duty cycle of the brake PWM in percent.

## 2039h Motor Currents

### Function

This object contains the measured motor currents in mA.

### Object description

Index	2039 <sub>h</sub>
Object Name	Motor Currents
Object Code	ARRAY
Data type	INTEGER32
Firmware Version	
Change History	

### Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	04 <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	I_d
Data type	INTEGER32
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

Subindex	02 <sub>h</sub>
Name	I_q
Data type	INTEGER32
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

Subindex	03 <sub>h</sub>
Name	I_a
Data type	INTEGER32
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

Subindex	04 <sub>h</sub>
Name	I_b
Data type	INTEGER32
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

## 203Ah Homing On Block Configuration

### Function

This object contains the parameters for homing on block (see the " **Homing**" section)

### Object description

Index	203A <sub>h</sub>
Object Name	Homing On Block Configuration
Object Code	ARRAY
Data type	UNSIGNED32
Access	
PDO Mapping	
Admissible Values	
Specified Value	
Firmware Version	
Change History	Amount of subentries has changed from 3 to 4

### Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	03 <sub>h</sub>
Subindex	01 <sub>h</sub>
Name	Minimum Current For Block Detection
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	

Specified Value	000004EC <sub>h</sub>
Subindex	02 <sub>h</sub>
Name	Period Of Blocking
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	03
Name	Block Detection Time
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000005 <sub>h</sub>

### Description

The subentries have the following function:

- 01<sub>h</sub>: Specifies the current limit value from which blocking is to be detected.
- 02<sub>h</sub>: Specifies the time in ms that the motor is nevertheless still to travel against the block after block detection.
- 03<sub>h</sub>: Specifies the time in ms that the current has to be at least above the minimum current threshold in order to detect a block.

## <sup>2</sup>203Bh I<sup>2</sup>t Parameters

### Function

This object contains the parameters for die I<sup>2</sup>t monitoring.

I<sup>2</sup>t monitoring is activated when a value greater than 0 is entered in 203B<sub>h</sub>:2 (see the **Motor overload protection section**).

I<sup>2</sup>t can only be use for closed loop mode with a single exception: If I<sup>2</sup>t is activated open loop mode, the current is limited to the set nominal current even when the set maximum current is greater. This feature was implemented for safety reasons so that it is also possible to switch out of closed loop mode with a very high short-time maximum current into open loop mode without damaging the motor.

### Object description

Index	203B <sub>h</sub>
Object Name	I <sup>2</sup> t Parameters
Object Code	ARRAY
Data type	UNSIGNED32
Firmware Version	
Change History	Amount of subentries has changed from 3 to 7

**Value description**

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	06 <sub>h</sub>
Subindex	01 <sub>h</sub>
Name	Nominal Current
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	02 <sub>h</sub>
Name	Maximum Duration Of Peak Current
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	03
Name	Threshold
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	04
Name	CalcValue
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>



Subindex	05
Name	LimitedCurrent
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	06
Name	Status
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

### Description

The subentries have the following function:

- 01<sub>h</sub>: Specifies the nominal current in mA, must be smaller than the maximum current **2031<sub>h</sub>**, otherwise monitoring will not be activated.
- 02<sub>h</sub>: Specifies the maximum time period of the peak current in ms.
- 03<sub>h</sub>: Threshold, specifies the limit in mA, from which is determined whether switching is to the maximum current or nominal current.
- 04<sub>h</sub>: CalcValue, specifies the calculated value that is compared to the threshold in order to set the current.
- 05<sub>h</sub>: LimitedCurrent, shows the actual current value that was set by I<sup>2</sup>t.
- 06<sub>h</sub>: Actual status. If the subentry value is "0", I<sup>2</sup>t is deactivated; if the value is "1", I<sup>2</sup>t is activated

## 2050h Encoder Alignment

### Function

This value specifies the angle offset between the rotor and the electrical field.

### Object description

Index	2050 <sub>h</sub>
Object Name	Encoder Alignment
Object Code	VARIABLE
Data type	INTEGER32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Firmware Version	

## Change History

---

### Description

The exact determination is only possible via the auto setup. The presence of this value is required for closed loop mode.

## 2051h Encoder Optimization

### Function

Contains compensation values to attain better concentricity in closed loop mode.

### Object description

Index	2051 <sub>h</sub>
Object Name	Encoder Optimization
Object Code	ARRAY
Data type	INTEGER32
Firmware Version	
Change History	

### Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	03 <sub>h</sub>
Subindex	01 <sub>h</sub>
Name	Parameter 1
Data type	INTEGER32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	02 <sub>h</sub>
Name	Parameter 2
Data type	INTEGER32
Access	read/write
PDO Mapping	no
Admissible Values	

Specified Value	00000000 <sub>h</sub>
Subindex	03 <sub>h</sub>
Name	Parameter 3
Data type	INTEGER32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

### Description

The exact determination is only possible via the auto setup.

## 2052h Encoder Resolution

### Function

Contains the resolution of the encoder that is used for electrical commutation.

### Object description

Index	2052 <sub>h</sub>
Object Name	Encoder Resolution
Object Code	VARIABLE
Data type	INTEGER32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00001000 <sub>h</sub>
Firmware Version	
Change History	

### Description

A negative value means that the encoder is operated in the opposite direction to the motor. This can be corrected by changing the poles of the motor winding.

## 2053h Index Polarity

### Function

Specifies the index polarity.

### Object description

Index	2053 <sub>h</sub>
Object Name	Index Polarity
Object Code	VARIABLE

Data type	UNSIGNED8
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
Firmware Version	
Change History	

### Description

The value 0 means that the index is not inverted.

The value 1 means that the index is connected inverted and is inverted in the firmware.

## 2054h Index Width

### Function

Specifies the index width in an internal operand.

### Object description

Index	2054 <sub>h</sub>
Object Name	Index Width
Object Code	VARIABLE
Data type	INTEGER32
Access	read/write
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	FFFFFFFF <sub>h</sub>
Firmware Version	
Change History	

### Description

If this value is not equal to 0, the encoder is monitored for errors.

The value -1 ( FFFFFFFF<sub>h</sub>) deactivates encoder monitoring.

## 2056h Limit Switch Tolerance Band

### Function

Specifies how far positive or negative end switches may be overrun before the motor controller issues an error.

This tolerance range is required, for example, to be able to complete reference runs - in which limit switches can be activated - error-free.

### Object description

Index	2056 <sub>h</sub>
-------	-------------------

Object Name	Limit Switch Tolerance Band
Object Code	VARIABLE
Data type	UNSIGNED32
Access	read/write
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	000001F4 <sub>h</sub>
Firmware Version	
Change History	

## 2057h Clock Direction Multiplier

### Function

The clock counting value in the clock/direction mode is multiplied by this value before it is processed further.

### Object description

---

Index	2057 <sub>h</sub>
Object Name	Clock Direction Multiplier
Object Code	VARIABLE
Data type	INTEGER32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000080 <sub>h</sub>
Firmware Version	
Change History	

---

## 2058h Clock Direction Divider

### Function

The clock counting value in the clock/direction mode is divided by this value before it is processed further.

### Object description

---

Index	2058 <sub>h</sub>
Object Name	Clock Direction Divider
Object Code	VARIABLE
Data type	INTEGER32
Access	read/write
PDO Mapping	no
Admissible Values	

Specified Value	00000001 <sub>h</sub>
Firmware Version	
Change History	

## 2059h Encoder Configuration

### Function

DESCRIPTION!

### Object description

Index	2059 <sub>h</sub>
Object Name	Encoder Configuration
Object Code	VARIABLE
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Firmware Version	
Change History	

## 2060h Compensate Polepair Count

### Function

Makes it possible to order motor-independent motion blocks.

### Object description

Index	2060 <sub>h</sub>
Object Name	Compensate Polepair Count
Object Code	VARIABLE
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000001 <sub>h</sub>
Firmware Version	
Change History	

### Description

If this entry is set to 1, the pole pair count is automatically set for all position, speed, acceleration, and jerk parameters.

If the value is 0, the pole pair count enters into the set values and must be taken into account when the motor is changed, as is the case with conventional stepper motor controllers.

## 2061h Velocity Numerator

### Function

Contains the numerator that is used to convert the speed specifications in profile position mode.

### Object description

Index	2061 <sub>h</sub>
Object Name	Velocity Numerator
Object Code	VARIABLE
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000001 <sub>h</sub>
Firmware Version	
Change History	

### Description

The internal operand pertains to full mechanical ( **2060<sub>h</sub>=1**) or electrical ( **2060<sub>h</sub>=0**) revolutions per second.

Thus, by setting object **2061<sub>h</sub>=1** and object **2062<sub>h</sub>=60**, for example, the speed can be specified in rpm in profile position mode.

## 2062h Velocity Denominator

### Function

Contains the denominator that is used to convert the speed specifications in profile position mode.

### Object description

Index	2062 <sub>h</sub>
Object Name	Velocity Denominator
Object Code	VARIABLE
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	0000003C <sub>h</sub>
Firmware Version	
Change History	

## Description

The internal operand pertains to full mechanical ( **2060<sub>h</sub>**=1) or electrical ( **2060<sub>h</sub>**=0) revolutions per second.

Thus, by setting object **2061<sub>h</sub>**=1 and object **2062<sub>h</sub>**=60, for example, the speed can be specified in revolutions per minute in profile position mode.

## 2063h Acceleration Numerator

### Function

Contains the numerator that is used to convert the acceleration specifications in profile position mode.

### Object description

---

Index	2063 <sub>h</sub>
Object Name	Acceleration Numerator
Object Code	VARIABLE
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000001 <sub>h</sub>
Firmware Version	
Change History	

---

### Description

The internal operand pertains to full mechanical ( **2060<sub>h</sub>**=1) or electrical ( **2060<sub>h</sub>**=0) revolutions per second.

Thus, by setting object **2063<sub>h</sub>**=1 and object **2064<sub>h</sub>**>=60, for example, the acceleration can be specified in (revolutions/min)/s<sup>2</sup> in profile position mode.

## 2064h Acceleration Denominator

### Function

Contains the denominator that is used to convert the acceleration specifications in profile position mode.

### Object description

---

Index	2064 <sub>h</sub>
Object Name	Acceleration Denominator
Object Code	VARIABLE
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	0000003C <sub>h</sub>

---



Firmware Version

Change History

### Description

The internal operand pertains to full mechanical ( **2060<sub>h</sub>**=1) or electrical ( **2060<sub>h</sub>**=0) revolutions per second.

Thus, by setting object **2063<sub>h</sub>**=1 and object **2064<sub>h</sub>**>=60, for example, the acceleration can be specified in (revolutions/min)/s<sup>2</sup> in profile position mode.

## 2065h Jerk Numerator

### Function

Contains the numerator that is used to convert the jerk specifications in profile position mode.

### Object description

Index	2065 <sub>h</sub>
Object Name	Jerk Numerator
Object Code	VARIABLE
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000001 <sub>h</sub>
Firmware Version	
Change History	

### Description

The internal operand pertains to full mechanical ( **2060<sub>h</sub>**=1) or electrical ( **2060<sub>h</sub>**=0) revolutions per second.

Thus, by setting object **2065<sub>h</sub>**=1 and object **2066<sub>h</sub>**>=60, for example, the jerk can be specified in (revolutions/min)/s<sup>2</sup> in profile position mode.

## 2066h Jerk Denominator

### Function

Contains the denominator that is used to convert the jerk specifications in profile position mode.

### Object description

Index	2066 <sub>h</sub>
Object Name	Jerk Denominator
Object Code	VARIABLE
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no

Admissible Values	
Specified Value	0000003C <sub>h</sub>
Firmware Version	
Change History	

### Description

The internal operand pertains to full mechanical ( **2060**<sub>h</sub>=1) or electrical ( **2060**<sub>h</sub>=0) revolutions per second.

Thus, by setting object **2065**<sub>h</sub>=1 and object **2066**<sub>h</sub>>=60, for example, the acceleration can be specified in (revolutions/min)/s<sup>2</sup> in profile position mode.

## 2084h Bootup Delay

### Function

This object allows specification of the time period between when the supply voltage is applied to the motor controller and the provision of operability of the motor controller in milliseconds.

### Object description

Index	2084 <sub>h</sub>
Object Name	Bootup Delay
Object Code	VARIABLE
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Firmware Version	
Change History	

## 2101h Fieldbus Module

### Function

Shows the type of mounted field bus module.

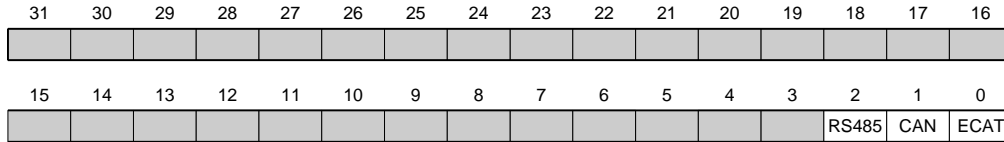
### Object description

Index	2101 <sub>h</sub>
Object Name	Fieldbus Module
Object Code	VARIABLE
Data type	INTEGER32
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

Firmware Version

Change History

**Description**



**ECAT**

Value = "1": The EtherCAT field bus is available

**CAN**

Value = "1": The CANopen field bus is available

**RS485**

Value = "1": A RS485 interface is available

**2200h Sampler Control**

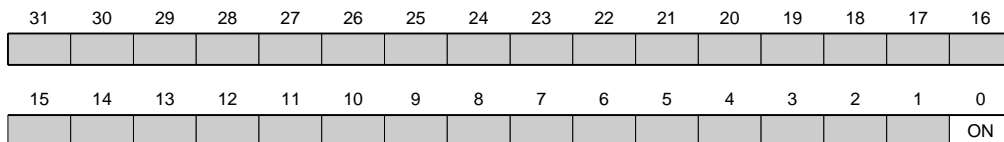
**Function**

Controls the installed sampler used to cyclically record any values from the "Dictionary" object.

**Object description**

Index	2200 <sub>h</sub>
Object Name	Sampler Control
Object Code	VARIABLE
Data type	UNSIGNED32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Firmware Version	
Change History	

**Description**



**ON**

Value = "1": The sampler will be activated

## 2201h Sampler Status

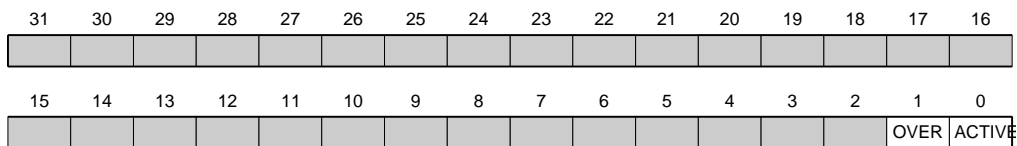
### Function

Shows the operating state of the installed sampler.

### Object description

Index	2201 <sub>h</sub>
Object Name	Sampler Status
Object Code	VARIABLE
Data type	UNSIGNED32
Access	read only
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Firmware Version	
Change History	

### Description



#### ACTIVE

Value = "1": The sampler is active and is recording data.

#### OVER

Value = "1": The recording buffer has not been read out fast enough and data have been lost. The sampler has therefore been stopped and must be restarted by a rising flank in object **2200<sub>h</sub>** bit 0.

## 2202h Sample Data Selection

### Function

As for object **1600<sub>h</sub>** ff., the data collected jointly per scan can be controlled here. In the current firmware, the sample buffer size is 12,000 bytes.

### Object description

Index	2202 <sub>h</sub>
Object Name	Sample Data Selection
Object Code	RECORD
Data type	PDO_MAPPING
Firmware Version	
Change History	

**Value description**

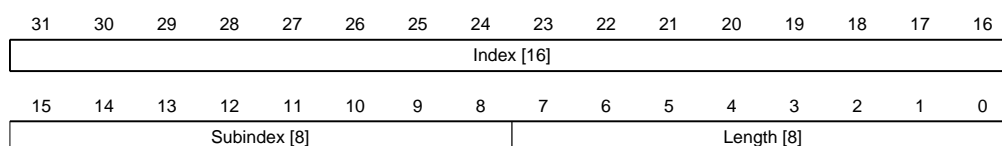
Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	02 <sub>h</sub>
Subindex	01 <sub>h</sub>
Name	Sample Value #1
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	60430010 <sub>h</sub>
Subindex	02 <sub>h</sub>
Name	Sample Value #2
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	22030220 <sub>h</sub>
Subindex	03 <sub>h</sub>
Name	Sample Value #3
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	04 <sub>h</sub>
Name	Sample Value #4
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

Subindex	05 <sub>h</sub>
Name	Sample Value #5
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	06 <sub>h</sub>
Name	Sample Value #6
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	07 <sub>h</sub>
Name	Sample Value #7
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	08 <sub>h</sub>
Name	Sample Value #8
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

### Description

Each subindex (1-8) describes a mapped object.

A mapping entry consists of four bytes made up according to the following graphic.



### Index [16]

Contains the index of the object to be mapped

**Subindex [8]**

Contains the subindex of the object to be mapped

**Length [8]**

Contains the length of the object to be mapped in the bit unit.

**2203h Sampler Buffer Information**

**Function**

This object makes additional information available to the sampler.

**Object description**

---

Index	2203 <sub>h</sub>
Object Name	Sampler Buffer Information
Object Code	ARRAY
Data type	UNSIGNED32
Firmware Version	
Change History	

---

**Value description**

---

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	03 <sub>h</sub>

---

Subindex	01 <sub>h</sub>
Name	Sample Buffer Size
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	02 <sub>h</sub>
Name	Sample Buffer Watermark
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	

---

---

Specified Value	00000000 <sub>h</sub>
<hr/>	
Subindex	03 <sub>h</sub>
Name	Sample Tick
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

### Description

The subindices have the following functions:

- 01<sub>h</sub> specifies the maximum size of the sampler buffer in bytes.
- 02<sub>h</sub> contains the momentary filling level of the sampler buffer in bytes.
- 03<sub>h</sub> contains a numerator that is incremented with each scan.

## 2204h Sample Time In Ms

### Function

This object contains the scan interval of the sampler in milliseconds.

### Object description

---

Index	2204 <sub>h</sub>
Object Name	Sample Time In Ms
Object Code	VARIABLE
Data type	UNSIGNED32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000001 <sub>h</sub>
Firmware Version	
Change History	

---

## 2300h VMM Control

### Function

Controls the execution of a user program.

### Object description

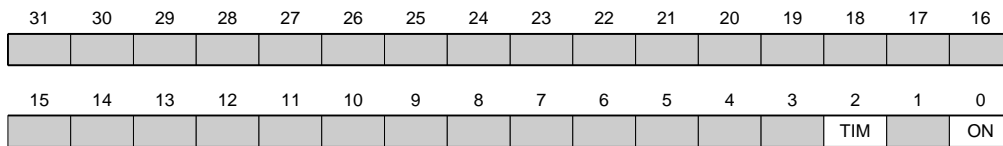
---

Index	2300 <sub>h</sub>
Object Name	VMM Control
Object Code	VARIABLE



Data type	UNSIGNED32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Firmware Version	
Change History	

### Description



#### ON

Switches the VMM on (value = "1") or off (value = "0").

When there is a rising flank in bit 0, the program is first reloaded and the variable range is reset.

#### TIM

Switches the timing control off (value = "1") or on (value = "0").

## 2301h VMM Status

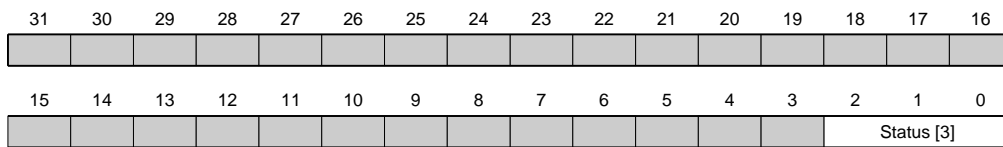
### Function

Shows the operating state of the user program.

### Object description

Index	2301 <sub>h</sub>
Object Name	VMM Status
Object Code	VARIABLE
Data type	UNSIGNED32
Access	read only
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Firmware Version	
Change History	

## Description



### Status [3]

Specifies the actual status of the VMM.

- Value = "0": Program has been stopped
- Value = "1": Program is running
- Value = "4": Program was closed with an error. The cause of the error can be read out in object **2302<sub>h</sub>**.

## 2302h VMM Error Code

### Function

Indicates which error occurred when the user program was executed.

### Object description

Index	2302 <sub>h</sub>
Object Name	VMM Error Code
Object Code	VARIABLE
Data type	UNSIGNED32
Access	read only
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Firmware Version	
Change History	

### Description

Error codes during program execution:

Number	Description
0x0000	No error
0x0001	Invalid service call (Cortex Svc)
0x0002	Memory protection violation (Cortex MPU Fault)
0x0003	Invalid Usage (Cortex error, for example due to an assembler command that is not admissible in the user mode)
0x0004	Hard fault (Cortex error)
0x0005	Timeout of 1-ms cycle
0x0006	Bus fault (Cortex error)
0x0007	Invalid stackpointer

Number	Description
0x0100	Bad program file

File system error codes when loading the user program:

Number	Description
0x10001	Hard fault in disc driver
0x10002	Internal file system fault
0x10003	Storage medium not ready
0x10004	File not found
0x10005	Directory not found
0x10006	Invalid file name/directory name
0x10008	Access to file not possible
0x10009	Invalid file/directory object
0x1000A	Storage medium is write protected
0x1000B	Invalid drive number
0x1000C	Working range of drive is invalid
0x1000D	No valid file system on drive
0x1000E	Creation of the file system has failed
0x1000F	Access not possible within required time
0x10010	Access was rejected

## 2303h Number Of Active User Program

### Function

Selects one of four possible user programs whose file names were stored earlier in object **2304<sub>h</sub>**.

### Object description

Index	2303 <sub>h</sub>
Object Name	Number Of Active User Program
Object Code	VARIABLE
Data type	UNSIGNED8
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00 <sub>h</sub>
Firmware Version	
Change History	

### Description

A change to the entry while a user program is being executed leads to the following procedure:

- The current program is stopped.
- The newly selected program is loaded.
- The newly loaded program is started.

## 2304h Table Of Available User Programs

### Function

The file names of the available user programs are stored here.

### Object description

Index	2304 <sub>h</sub>
Object Name	Table Of Available User Programs
Object Code	ARRAY
Data type	UNSIGNED32
Firmware Version	
Change History	

### Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	08 <sub>h</sub>
Subindex	01 <sub>h</sub>
Name	Name Of User Program 1 UB
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	02 <sub>h</sub>
Name	Name Of User Program 1 LB
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	03 <sub>h</sub>
Name	Name Of User Program 2 UB
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	04 <sub>h</sub>
Name	Name Of User Program 2 LB
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	05 <sub>h</sub>
Name	Name Of User Program 3 UB
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	06 <sub>h</sub>
Name	Name Of User Program 3 LB
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	07 <sub>h</sub>
Name	Name Of User Program 4 UB
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	08 <sub>h</sub>
----------	-----------------

---

Name	Name Of User Program 4 LB
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

### Description

For each user program, the name of the user program is contained in two consecutive subindices in ASCII code.

Program 1: subindices 1 and 2

Program 2: subindices 3 and 4

Program 3: subindices 5 and 6

Program 4: subindices 7 and 8

**Example:** Program 1 with the name `test.usr` is coded as follows:

t = 74<sub>h</sub>

e = 65<sub>h</sub>

s = 73<sub>h</sub>

Thus the two entries in subindices 1 and 2 are:

74657374<sub>h</sub>, 00000000<sub>h</sub>

For each user program, the name of the user program is contained in two consecutive subindices in ASCII code. The subindex with the designation UB (Upper Byte) contains the first four letters of the name, the subindex with LB (Lower Byte) the last four letters. Should the name have less than eight letters, the missing letters must be filled with zeros.

## 2310h VMM Input Data Selection

### Function

Specifies the object dictionary entries that are copied into the input PDO mapping of the VMM program.

### Object description

Index	2310 <sub>h</sub>
Object Name	VMM Input Data Selection
Object Code	RECORD
Data type	PDO_MAPPING
Firmware Version	
Change History	Amount of subentries has changed from 2 to 17

### Value description

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

Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
Subindex	01 <sub>h</sub> -10 <sub>h</sub>
Name	Mapping #1
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	02
Name	Mapping #2
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	03
Name	Mapping #3
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	04
Name	Mapping #4
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	05
Name	Mapping #5
Data type	UNSIGNED32
Access	read/write

---

PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	06
Name	Mapping #6
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	07
Name	Mapping #7
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	08
Name	Mapping #8
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	09
Name	Mapping #9
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	0A
Name	Mapping #10
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no

---



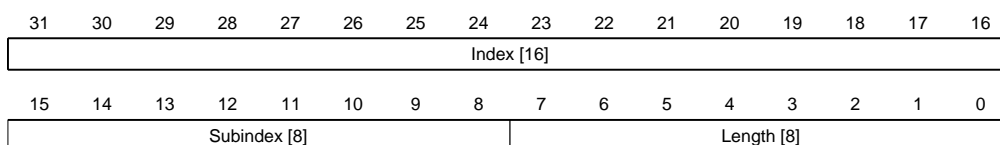
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	
Subindex	0B
Name	Mapping #11
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	
Subindex	0C
Name	Mapping #12
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	
Subindex	0D
Name	Mapping #13
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	
Subindex	0E
Name	Mapping #14
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	
Subindex	0F
Name	Mapping #15
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	

Specified Value	00000000 <sub>h</sub>
Subindex	10
Name	Mapping #16
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

### Description

Each subindex (1-16) describes a mapped object.

A mapping entry consists of four bytes made up according to the following graphic.



#### Index [16]

Contains the index of the object to be mapped

#### Subindex [8]

Contains the subindex of the object to be mapped

#### Length [8]

Contains the length of the object to be mapped in the bit unit.

## 2320h VMM Output Data Selection

### Function

Specifies the object dictionary entries that are copied into the output PDO mapping of the VMM program after it has been executed.

### Object description

Index	2320 <sub>h</sub>
Object Name	VMM Output Data Selection
Object Code	RECORD
Data type	PDO_MAPPING
Firmware Version	
Change History	Amount of subentries has changed from 2 to 17

### Value description

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

Data type	UNSIGNED8
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
Subindex	01 <sub>h</sub> -10 <sub>h</sub>
Name	Mapping #1
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	02
Name	Mapping #2
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	03
Name	Mapping #3
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	04
Name	Mapping #4
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	05
Name	Mapping #5
Data type	UNSIGNED32

Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	06
Name	Mapping #6
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	07
Name	Mapping #7
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	08
Name	Mapping #8
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	09
Name	Mapping #9
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	0A
Name	Mapping #10
Data type	UNSIGNED32
Access	read/write

---

PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	0B
Name	Mapping #11
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	0C
Name	Mapping #12
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	0D
Name	Mapping #13
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	0E
Name	Mapping #14
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	0F
Name	Mapping #15
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no

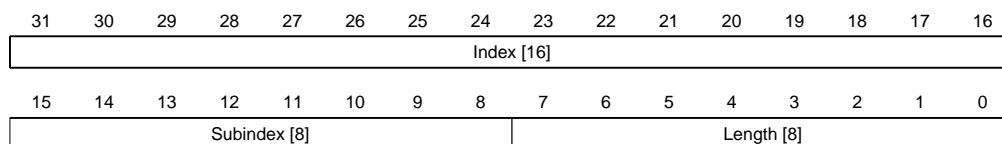
---

Admissible Values	
Specified Value	00000000 <sub>h</sub>
<hr/>	
Subindex	10
Name	Mapping #16
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

### Description

Each subindex (1-16) describes a mapped object.

A mapping entry consists of four bytes made up according to the following graphic.



#### Index [16]

Contains the index of the object to be mapped

#### Subindex [8]

Contains the subindex of the object to be mapped

#### Length [8]

Contains the length of the object to be mapped in the bit unit.

## 2330h VMM In/output Data Selection

### Function

Specifies the object dictionary entries that are copied into the input PDO mapping of the VMM program and after its execution are copied back into the output PDO mapping.

### Object description

Index	2330 <sub>h</sub>
Object Name	VMM In/output Data Selection
Object Code	RECORD
Data type	PDO_MAPPING
Firmware Version	
Change History	Amount of subentries has changed from 2 to 17

### Value description

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

Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
Subindex	01 <sub>h</sub> -10 <sub>h</sub>
Name	Mapping #1
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	02
Name	Mapping #2
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	03
Name	Mapping #3
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	04
Name	Mapping #4
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	05
Name	Mapping #5

Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

Subindex	06
Name	Mapping #6
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

Subindex	07
Name	Mapping #7
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

Subindex	08
Name	Mapping #8
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

Subindex	09
Name	Mapping #9
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

Subindex	0A
Name	Mapping #10
Data type	UNSIGNED32



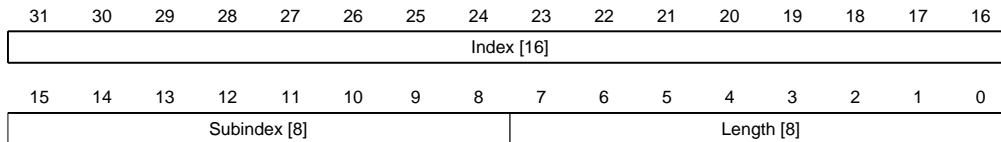
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	0B
Name	Mapping #11
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	0C
Name	Mapping #12
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	0D
Name	Mapping #13
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	0E
Name	Mapping #14
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	0F
Name	Mapping #15
Data type	UNSIGNED32
Access	read/write

PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
<hr/>	
Subindex	10
Name	Mapping #16
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

**Description**

Each subindex (1-16) describes a mapped object.

A mapping entry consists of four bytes made up according to the following graphic.



**Index [16]**

Contains the index of the object to be mapped

**Subindex [8]**

Contains the subindex of the object to be mapped

**Length [8]**

Contains the length of the object to be mapped in the bit unit.

**2400h VMM Inputs**

**Function**

Contains an array with 32 32-bit integer values that is not used within the firmware and is only used for communication with the user program via the field bus.

**Object description**

Index	2400 <sub>h</sub>
Object Name	VMM Inputs
Object Code	ARRAY
Data type	INTEGER32
Firmware Version	
Change History	Amount of subentries has changed from 2 to 33

**Value description**

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	20 <sub>h</sub>
Subindex	01 <sub>h</sub> -20 <sub>h</sub>
Name	VMM Input 1#
Data type	INTEGER32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	02
Name	VMM Input 2#
Data type	INTEGER32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	03
Name	VMM Input 3#
Data type	INTEGER32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	04
Name	VMM Input 4#
Data type	INTEGER32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	05
Name	VMM Input 5#
Data type	INTEGER32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	06
Name	VMM Input 6#
Data type	INTEGER32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	07
Name	VMM Input 7#
Data type	INTEGER32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	08
Name	VMM Input 8#
Data type	INTEGER32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	09
Name	VMM Input 9#
Data type	INTEGER32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	0A
----------	----

---

Name	VMM Input 10#
Data type	INTEGER32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>

Subindex	0B
Name	VMM Input 11#
Data type	INTEGER32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>

Subindex	0C
Name	VMM Input 12#
Data type	INTEGER32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>

Subindex	0D
Name	VMM Input 13#
Data type	INTEGER32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>

Subindex	0E
Name	VMM Input 14#
Data type	INTEGER32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>

Subindex	0F
Name	VMM Input 15#

Data type	INTEGER32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	10
Name	VMM Input 16#
Data type	INTEGER32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	11
Name	VMM Input 17#
Data type	INTEGER32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	12
Name	VMM Input 18#
Data type	INTEGER32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	13
Name	VMM Input 19#
Data type	INTEGER32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	14
Name	VMM Input 20#
Data type	INTEGER32

Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	15
Name	VMM Input 21#
Data type	INTEGER32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	16
Name	VMM Input 22#
Data type	INTEGER32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	17
Name	VMM Input 23#
Data type	INTEGER32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	18
Name	VMM Input 24#
Data type	INTEGER32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	19
Name	VMM Input 25#
Data type	INTEGER32
Access	read/write

PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	1A
Name	VMM Input 26#
Data type	INTEGER32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	1B
Name	VMM Input 27#
Data type	INTEGER32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	1C
Name	VMM Input 28#
Data type	INTEGER32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	1D
Name	VMM Input 29#
Data type	INTEGER32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	1E
Name	VMM Input 30#
Data type	INTEGER32
Access	read/write
PDO Mapping	RX-PDO



Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	
Subindex	1F
Name	VMM Input 31#
Data type	INTEGER32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	
Subindex	20
Name	VMM Input 32#
Data type	INTEGER32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>

### Description

This is where the specified values, for example, can be transferred to the VMM program.

## 2500h VMM Outputs

### Function

Contains an array with 32 32-bit integer values that is not used within the firmware and is only used for communication with the user program via the field bus.

### Object description

Index	2500 <sub>h</sub>
Object Name	VMM Outputs
Object Code	ARRAY
Data type	INTEGER32
Firmware Version	
Change History	Amount of subentries has changed from 2 to 33

### Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO Mapping	no

Admissible Values	
Specified Value	20 <sub>h</sub>
Subindex	
Subindex	01 <sub>h</sub> -20 <sub>h</sub>
Name	VMM Output 1#
Data type	INTEGER32
Access	read/write
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	
Subindex	02
Name	VMM Output 2#
Data type	INTEGER32
Access	read/write
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	
Subindex	03
Name	VMM Output 3#
Data type	INTEGER32
Access	read/write
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	
Subindex	04
Name	VMM Output 4#
Data type	INTEGER32
Access	read/write
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	
Subindex	05
Name	VMM Output 5#
Data type	INTEGER32
Access	read/write
PDO Mapping	TX-PDO
Admissible Values	

Specified Value	00000000 <sub>h</sub>
Subindex	06
Name	VMM Output 6#
Data type	INTEGER32
Access	read/write
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	07
Name	VMM Output 7#
Data type	INTEGER32
Access	read/write
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	08
Name	VMM Output 8#
Data type	INTEGER32
Access	read/write
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	09
Name	VMM Output 9#
Data type	INTEGER32
Access	read/write
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	0A
Name	VMM Output 10#
Data type	INTEGER32
Access	read/write
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	0B
Name	VMM Output 11#
Data type	INTEGER32
Access	read/write
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	0C
Name	VMM Output 12#
Data type	INTEGER32
Access	read/write
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	0D
Name	VMM Output 13#
Data type	INTEGER32
Access	read/write
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	0E
Name	VMM Output 14#
Data type	INTEGER32
Access	read/write
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	0F
Name	VMM Output 15#
Data type	INTEGER32
Access	read/write
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	10
----------	----

---

---

Name	VMM Output 16#
Data type	INTEGER32
Access	read/write
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	11
Name	VMM Output 17#
Data type	INTEGER32
Access	read/write
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	12
Name	VMM Output 18#
Data type	INTEGER32
Access	read/write
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	13
Name	VMM Output 19#
Data type	INTEGER32
Access	read/write
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	14
Name	VMM Output 20#
Data type	INTEGER32
Access	read/write
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	15
Name	VMM Output 21#

Data type	INTEGER32
Access	read/write
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
<hr/>	
Subindex	16
Name	VMM Output 22#
Data type	INTEGER32
Access	read/write
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
<hr/>	
Subindex	17
Name	VMM Output 23#
Data type	INTEGER32
Access	read/write
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
<hr/>	
Subindex	18
Name	VMM Output 24#
Data type	INTEGER32
Access	read/write
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
<hr/>	
Subindex	19
Name	VMM Output 25#
Data type	INTEGER32
Access	read/write
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
<hr/>	
Subindex	1A
Name	VMM Output 26#
Data type	INTEGER32

Access	read/write
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	1B
Name	VMM Output 27#
Data type	INTEGER32
Access	read/write
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	1C
Name	VMM Output 28#
Data type	INTEGER32
Access	read/write
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	1D
Name	VMM Output 29#
Data type	INTEGER32
Access	read/write
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	1E
Name	VMM Output 30#
Data type	INTEGER32
Access	read/write
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	1F
Name	VMM Output 31#
Data type	INTEGER32
Access	read/write

PDO Mapping	TX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	20
Name	VMM Output 32#
Data type	INTEGER32
Access	read/write
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>

### Description

The VMM program can store results here that can then be read out via the field bus.

## 2600h VMM Debug Output

### Function

This object contains debug outputs for a user program.

### Object description

Index	2600 <sub>h</sub>
Object Name	VMM Debug Output
Object Code	ARRAY
Data type	UNSIGNED8
Firmware Version	
Change History	Amount of subentries has changed from 2 to 65

### Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
Subindex	01-40 <sub>h</sub>
Name	Value #1
Data type	UNSIGNED8
Access	read only



PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
<hr/>	
Subindex	02
Name	Value #2
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
<hr/>	
Subindex	03
Name	Value #3
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
<hr/>	
Subindex	04
Name	Value #4
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
<hr/>	
Subindex	05
Name	Value #5
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
<hr/>	
Subindex	06
Name	Value #6
Data type	UNSIGNED8
Access	read only
PDO Mapping	no

Admissible Values	
Specified Value	00 <sub>h</sub>
<hr/>	
Subindex	07
Name	Value #7
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
<hr/>	
Subindex	08
Name	Value #8
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
<hr/>	
Subindex	09
Name	Value #9
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
<hr/>	
Subindex	0A
Name	Value #10
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
<hr/>	
Subindex	0B
Name	Value #11
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	

Specified Value	00 <sub>h</sub>
Subindex	0C
Name	Value #12
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
Subindex	0D
Name	Value #13
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
Subindex	0E
Name	Value #14
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
Subindex	0F
Name	Value #15
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
Subindex	10
Name	Value #16
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>

---

Subindex	11
Name	Value #17
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>

---

Subindex	12
Name	Value #18
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>

---

Subindex	13
Name	Value #19
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>

---

Subindex	14
Name	Value #20
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>

---

Subindex	15
Name	Value #21
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>

---

Subindex	16
----------	----

---

Name	Value #22
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
Subindex	17
Name	Value #23
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
Subindex	18
Name	Value #24
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
Subindex	19
Name	Value #25
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
Subindex	1A
Name	Value #26
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
Subindex	1B
Name	Value #27

Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
Subindex	1C
Name	Value #28
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
Subindex	1D
Name	Value #29
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
Subindex	1E
Name	Value #30
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
Subindex	1F
Name	Value #31
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
Subindex	20
Name	Value #32
Data type	UNSIGNED8

Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
Subindex	21
Name	Value #33
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
Subindex	22
Name	Value #34
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
Subindex	23
Name	Value #35
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
Subindex	24
Name	Value #36
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
Subindex	25
Name	Value #37
Data type	UNSIGNED8
Access	read only

PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
<hr/>	
Subindex	26
Name	Value #38
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
<hr/>	
Subindex	27
Name	Value #39
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
<hr/>	
Subindex	28
Name	Value #40
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
<hr/>	
Subindex	29
Name	Value #41
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
<hr/>	
Subindex	2A
Name	Value #42
Data type	UNSIGNED8
Access	read only
PDO Mapping	no



Admissible Values	
Specified Value	00 <sub>h</sub>
<hr/>	
Subindex	2B
Name	Value #43
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
<hr/>	
Subindex	2C
Name	Value #44
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
<hr/>	
Subindex	2D
Name	Value #45
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
<hr/>	
Subindex	2E
Name	Value #46
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
<hr/>	
Subindex	2F
Name	Value #47
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	

Specified Value	00 <sub>h</sub>
Subindex	30
Name	Value #48
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
Subindex	31
Name	Value #49
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
Subindex	32
Name	Value #50
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
Subindex	33
Name	Value #51
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
Subindex	34
Name	Value #52
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>

---

Subindex	35
Name	Value #53
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>

---

Subindex	36
Name	Value #54
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>

---

Subindex	37
Name	Value #55
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>

---

Subindex	38
Name	Value #56
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>

---

Subindex	39
Name	Value #57
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>

---

Subindex	3A
----------	----

Name	Value #58
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
Subindex	3B
Name	Value #59
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
Subindex	3C
Name	Value #60
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
Subindex	3D
Name	Value #61
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
Subindex	3E
Name	Value #62
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
Subindex	3F
Name	Value #63

Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>

Subindex	40
Name	Value #64
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>

### Description

The VMM program stores the debug outputs here that have been called up with the function `VmmDebugOutputString()`, `VmmDebugOutputInt()`, and suchlike. A more detailed description debug output can be found in the **Debug-output** subsection of the **Programming with NanoJ** section.

## 2700h User Storage Area

### Function

DESCRIPTION!

### Object description

Index	2700 <sub>h</sub>
Object Name	User Storage Area
Object Code	RECORD
Data type	USER_STORAGE_AREA
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	
Firmware Version	
Change History	

### Value description

Subindex	
Name	Storage Control Word
Data type	UNSIGNED8
Access	read/write
PDO Mapping	no

Admissible Values

Specified Value 00<sub>h</sub>

---

Subindex

Name Storage 1#  
Data type UNSIGNED8  
Access read/write  
PDO Mapping no  
Admissible Values  
Specified Value 00<sub>h</sub>

---

Subindex

Name Storage 2#  
Data type UNSIGNED16  
Access read/write  
PDO Mapping no  
Admissible Values  
Specified Value 0000<sub>h</sub>

---

Subindex

Name Storage 3#  
Data type UNSIGNED16  
Access read/write  
PDO Mapping no  
Admissible Values  
Specified Value 0000<sub>h</sub>

---

Subindex

Name Storage 4#  
Data type UNSIGNED16  
Access read/write  
PDO Mapping no  
Admissible Values  
Specified Value 0000<sub>h</sub>

---

Subindex

Name Storage 5#  
Data type UNSIGNED16  
Access read/write  
PDO Mapping no  
Admissible Values

Specified Value	0000 <sub>h</sub>
Subindex	
Name	Storage 6#
Data type	UNSIGNED16
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	0000 <sub>h</sub>
Subindex	
Name	Storage 7#
Data type	UNSIGNED16
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	0000 <sub>h</sub>
Subindex	
Name	Storage 8#
Data type	UNSIGNED16
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	0000 <sub>h</sub>
Subindex	
Name	Storage 9#
Data type	UNSIGNED16
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	0000 <sub>h</sub>
Subindex	
Name	Storage 10#
Data type	UNSIGNED16
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	0000 <sub>h</sub>

---

Subindex	
Name	Storage 11#
Data type	UNSIGNED16
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	0000 <sub>h</sub>

---

Subindex	
Name	Storage 12#
Data type	UNSIGNED16
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	0000 <sub>h</sub>

---

Subindex	
Name	Storage 13#
Data type	UNSIGNED16
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	0000 <sub>h</sub>

---

Subindex	
Name	Storage 14#
Data type	UNSIGNED16
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	0000 <sub>h</sub>

---

Subindex	
Name	Storage 15#
Data type	UNSIGNED16
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	0000 <sub>h</sub>

---

Subindex

---



---

Name	Storage 16#
Data type	UNSIGNED16
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	0000 <sub>h</sub>

---

Subindex

Name	Storage 17#
Data type	UNSIGNED16
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	0000 <sub>h</sub>

---

Subindex

Name	Storage 18#
Data type	UNSIGNED16
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	0000 <sub>h</sub>

---

Subindex

Name	Storage 19#
Data type	UNSIGNED16
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	0000 <sub>h</sub>

---

Subindex

Name	Storage 20#
Data type	UNSIGNED16
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	0000 <sub>h</sub>

---

Subindex

Name	Storage 21#
------	-------------

Data type	UNSIGNED16
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	0000 <sub>h</sub>

## Description

### 2800h Bootloader Setting

#### Function

DESCRIPTION!

#### Object description

Index	2800 <sub>h</sub>
Object Name	Bootloader Setting
Object Code	ARRAY
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	
Firmware Version	
Change History	

#### Value description

Subindex	
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>

Subindex	
Name	Bootloader Password
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

## Description

### 3202h Motor Drive Submode Select

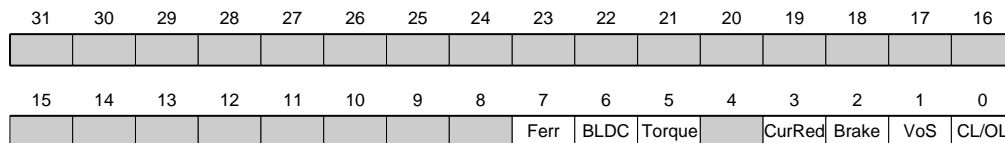
#### Function

Controls the control mode such as the closed loop /open loop changeover and whether velocity mode is simulated via the S control, or whether it operates with a true v control in the closed loop.

#### Object description

Index	3202 <sub>h</sub>
Object Name	Motor Drive Submode Select
Object Code	VARIABLE
Data type	UNSIGNED32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Firmware Version	
Change History	

#### Description



#### CL/OL

Switchover between open loop and closed loop

- Value = "0": Open loop
- Value = "1": Closed loop

#### VoS

Value = "1": Simulate v-control via an S ramp

#### Brake

Value = "1": Switch on the brake controller

#### CurRed (Current Reduction)

Value = "1": Current reduction activated in open loop

#### Torque

Only active in **profile torque mode**

Value = "1": M-control is active, otherwise a V-control is superimposed

**BLDC**

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

**Ferr (Following Error)**

Value = "1": A following error triggers a fault with an associated response (see object **605E<sub>h</sub>**)

**3208h Motor Drive Sensor Source Open Loop**

**Function**

DESCRIPTION!

**Object description**

Index	3208 <sub>h</sub>
Object Name	Motor Drive Sensor Source Open Loop
Object Code	ARRAY
Data type	INTEGER32
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	
Firmware Version	
Change History	

**Value description**

Subindex	
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	04 <sub>h</sub>

Subindex	
Name	Commutation
Data type	INTEGER32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

Subindex	
Name	Torque

---

Data type	INTEGER32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	
Name	Velocity
Data type	INTEGER32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	
Name	Position
Data type	INTEGER32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

**Description**

**3209h Motor Drive Sensor Source Closed Loop**

**Function**

DESCRIPTION!

**Object description**

---

Index	3209 <sub>h</sub>
Object Name	Motor Drive Sensor Source Closed Loop
Object Code	ARRAY
Data type	INTEGER32
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	
Firmware Version	
Change History	

---

**Value description**

Subindex	
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	04 <sub>h</sub>
Subindex	
Name	Commutation
Data type	INTEGER32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	
Name	Torque
Data type	INTEGER32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	
Name	Velocity
Data type	INTEGER32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	
Name	Position
Data type	INTEGER32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

## Description

### 320Ah Motor Drive Sensor Display Open Loop

#### Function

It can be used to change the source for objects **6044<sub>h</sub>** and **6064<sub>h</sub>** in open loop mode.

#### Object description

Index	320A <sub>h</sub>
Object Name	Motor Drive Sensor Display Open Loop
Object Code	ARRAY
Data type	INTEGER32
Firmware Version	
Change History	

#### Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	04 <sub>h</sub>
Subindex	01 <sub>h</sub>
Name	Commutation
Data type	INTEGER32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	02 <sub>h</sub>
Name	Torque
Data type	INTEGER32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	03 <sub>h</sub>
Name	Velocity

Data type	INTEGER32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	FFFFFFFF <sub>h</sub>
Subindex	04 <sub>h</sub>
Name	Position
Data type	INTEGER32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	FFFFFFFF <sub>h</sub>

### Description

The following subindices haven a meaning:

- 01<sub>h</sub>: Unused
- 02<sub>h</sub>: Unused
- 03<sub>h</sub>: Changes the source of object **6044<sub>h</sub>**:
  - Value = "-1": The internally calculated value is entered in object **6044<sub>h</sub>**
  - Value = "0": The value is kept at 0
  - Value = "1": The encoder value is entered in object **6044<sub>h</sub>**
- 04<sub>h</sub>: Changes the source of **6064<sub>h</sub>**:
  - Value = "-1": The internally calculated value is entered in object **6064<sub>h</sub>**
  - Value = "0": The value is kept at 0
  - Value = "1": The encoder value is entered in object **6064<sub>h</sub>**

## 320Bh Motor Drive Sensor Display Closed Loop

### Function

It can be used to change the source for objects **6044<sub>h</sub>** and **6064<sub>h</sub>** in closed loop mode.

### Object description

Index	320B <sub>h</sub>
Object Name	Motor Drive Sensor Display Closed Loop
Object Code	ARRAY
Data type	INTEGER32
Firmware Version	
Change History	

### Value description

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



Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	04 <sub>h</sub>
Subindex	01 <sub>h</sub>
Name	Commutation
Data type	INTEGER32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	02 <sub>h</sub>
Name	Torque
Data type	INTEGER32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	03 <sub>h</sub>
Name	Velocity
Data type	INTEGER32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000001 <sub>h</sub>
Subindex	04 <sub>h</sub>
Name	Position
Data type	INTEGER32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000001 <sub>h</sub>

### Description

The following subindices haven a meaning:

- 01<sub>h</sub>: Unused
- 02<sub>h</sub>: Unused

- 03<sub>h</sub>: Changes the source of object 6044<sub>h</sub>:
  - Value = "-1": The internally calculated value is entered in object 6044<sub>h</sub>
  - Value = "0": The value is kept at 0
  - Value = "1": The encoder value is entered in object 6044<sub>h</sub>
- 04<sub>h</sub>: Changes the source of object 6064<sub>h</sub>:
  - Value = "-1": The internally calculated value is entered in object 6064<sub>h</sub>
  - Value = "0": The value is kept at 0
  - Value = "1": The encoder value is entered in object 6064<sub>h</sub>

## 3210h Motor Drive Parameter Set

### Function

Contains the P and I values of the current, distance and position controllers for the open loop (only the current controller is activated) and closed loop.

### Object description

Index	3210 <sub>h</sub>
Object Name	Motor Drive Parameter Set
Object Code	ARRAY
Data type	INTEGER32
Firmware Version	
Change History	Amount of subentries has changed from 9 to 11

### Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	0A <sub>h</sub>
Subindex	01 <sub>h</sub>
Name	S_P
Data type	INTEGER32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000800 <sub>h</sub>
Subindex	02 <sub>h</sub>
Name	S_I
Data type	INTEGER32

Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	03 <sub>h</sub>
Name	V_P
Data type	INTEGER32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00001B58 <sub>h</sub>
Subindex	04 <sub>h</sub>
Name	V_I
Data type	INTEGER32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000004 <sub>h</sub>
Subindex	05 <sub>h</sub>
Name	Id_P
Data type	INTEGER32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	000668A0 <sub>h</sub>
Subindex	06 <sub>h</sub>
Name	Id_I
Data type	INTEGER32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00002EE0 <sub>h</sub>
Subindex	07 <sub>h</sub>
Name	Iq_P
Data type	INTEGER32
Access	read/write

PDO Mapping	no
Admissible Values	
Specified Value	000668A0 <sub>h</sub>
Subindex	08 <sub>h</sub>
Name	Iq_I
Data type	INTEGER32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00002EE0 <sub>h</sub>
Subindex	09
Name	I_P
Data type	INTEGER32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00027100 <sub>h</sub>
Subindex	0A
Name	I_I
Data type	INTEGER32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	000055F0 <sub>h</sub>

### Description

- Subindex 00<sub>h</sub>: Number of entries
- Subindex 01<sub>h</sub>: Proportional value of the S control
- Subindex 02<sub>h</sub>: Integral value of the S control
- Subindex 03<sub>h</sub>: Proportional value of the V control
- Subindex 04<sub>h</sub>: Integral value of the V control
- Subindex 05<sub>h</sub>: (Closed Loop) Proportional value of the current controller for the field-forming component
- Subindex 06<sub>h</sub>: (Closed Loop) Integral value of the current controller for the field-forming component
- Subindex 07<sub>h</sub>: (Closed Loop) Proportional value of the current controller for the torque-forming component
- Subindex 08<sub>h</sub>: (Closed Loop) Integral value of the current controller for the torque-forming component
- Subindex 09<sub>h</sub>: (Open Loop) Proportional value of the current controller for the torque-forming component
- Subindex 0A<sub>h</sub>: (Open Loop) Integral value of the current controller for the torque-forming component

## 3220h Analog Inputs

### Function

Shows the current values of the analog inputs in [digits].

Object **3221<sub>h</sub>** allows the respective analog input to be configured as a current or voltage input.

### Object description

---

Index	3220 <sub>h</sub>
Object Name	Analog Inputs
Object Code	ARRAY
Data type	INTEGER16
Firmware Version	
Change History	

---

### Value description

---

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	02 <sub>h</sub>

---

Subindex	01 <sub>h</sub>
Name	Analogue Input 1
Data type	INTEGER16
Access	read only
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	0000 <sub>h</sub>

---

Subindex	02 <sub>h</sub>
Name	Analogue Input 2
Data type	INTEGER16
Access	read only
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	0000 <sub>h</sub>

---

### Description

Formulas for conversion of [digits] into the respective unit:

- Voltage input: (x digits - 512 digits) \* 20 V / 1024 digits
- Current input: x digits \* 20 mA / 1024 digits

## 3221h Analogue Inputs Control

### Function

DESCRIPTION!

### Object description

Index	3221 <sub>h</sub>
Object Name	Analogue Inputs Control
Object Code	VARIABLE
Data type	INTEGER32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Firmware Version	
Change History	

### Description

## 3240h Digital Inputs Control

### Function

This object allows manipulation of digital inputs as described in the section **Digital inputs and outputs**. For all the following subindices, bit 0 pertains to digital input 1, bit 1 pertains to input 2, etc.

### Object description

Index	3240 <sub>h</sub>
Object Name	Digital Inputs Control
Object Code	ARRAY
Data type	UNSIGNED32
Firmware Version	
Change History	Version 1.0.3: Subindex 01 <sub>h</sub> :Entry "Name" changed from "Special Function Disable" to "Special Function Enable" Amount of subentries has changed from 6 to 8

### Value description

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

PDO Mapping	no
Admissible Values	
Specified Value	07 <sub>h</sub>
Subindex	01 <sub>h</sub>
Name	Special Function Enable
Data type	UNSIGNED32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	02 <sub>h</sub>
Name	Function Inverted
Data type	UNSIGNED32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	03 <sub>h</sub>
Name	Force Enable
Data type	UNSIGNED32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	04 <sub>h</sub>
Name	Force Value
Data type	UNSIGNED32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	05 <sub>h</sub>
Name	Raw Value
Data type	UNSIGNED32
Access	read/write
PDO Mapping	RX-PDO

Admissible Values	
Specified Value	00000000 <sub>h</sub>
<hr/>	
Subindex	06
Name	Input Range Select
Data type	UNSIGNED32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
<hr/>	
Subindex	07
Name	Differential Select
Data type	UNSIGNED32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>

### Description

The subentries have the following function:

- 01<sub>h</sub>: This subindex switches on the special functions of the respective input if the bit has the value "1".
- 02<sub>h</sub>: This subindex inverts the logic of an input if the respective input has the value "1".
- 03<sub>h</sub>: This subindex forces an input value if the bit has the value "1". An input with a forced value is always set to the value entered in subentry 4<sub>h</sub> regardless of the applied voltage level.
- 04<sub>h</sub>: This subindex specifies the input value to be forced.
- 05<sub>h</sub>: This subindex always contains the read, unmodified input value.

## 3250h Digital Outputs Control

### Function

This object allows manipulation of digital outputs as described in the section **Digital inputs and outputs**. For all the following subindices, bit 0 pertains to digital output 1, bit 1 pertains to output 2, etc.

### Object description

Index	3250 <sub>h</sub>
Object Name	Digital Outputs Control
Object Code	ARRAY
Data type	UNSIGNED32
Firmware Version	
Change History	Version 1.0.3: Subindex 01 <sub>h</sub> : Entry "Name" changed from "Special Function Disable" auf "Special Function Enable"



**Value description**

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	05 <sub>h</sub>
Subindex	01 <sub>h</sub>
Name	Special Function Enable
Data type	UNSIGNED32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	000F0001 <sub>h</sub>
Subindex	02 <sub>h</sub>
Name	Function Inverted
Data type	UNSIGNED32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	03 <sub>h</sub>
Name	Force Enable
Data type	UNSIGNED32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Subindex	04 <sub>h</sub>
Name	Force Value
Data type	UNSIGNED32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

Subindex	05 <sub>h</sub>
Name	Raw Value
Data type	UNSIGNED32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>

---

### Description

The subentries have the following function:

- 01<sub>h</sub>: No function.
- 02<sub>h</sub>: This subindex inverts the logic (from opener logic to closer logic)
- 03<sub>h</sub>: This subindex forces an output value if the bit has the value "1". The level of the output is defined in subindex 4<sub>h</sub>.
- 04<sub>h</sub>: This subindex defines the level to be applied to the output. The value "0" delivers a logical low level at the digital output; value "1" delivers a logical high level.
- 05<sub>h</sub>: In this subindex, the bit combination applied to the outputs is stored.

## 3700h Following Error Option Code

### Function

The object contains the action to be executed if a contouring error is triggered.

### Object description

---

Index	3700 <sub>h</sub>
Object Name	Following Error Option Code
Object Code	VARIABLE
Data type	INTEGER16
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	FFFF <sub>h</sub>
Firmware Version	
Change History	

---

### Description

Wert	Beschreibung
-32768 bis -1	Reserved
0	Immediate stop with short-circuit braking
1	Braking with "slow down ramp" (deceleration depending on operating mode)
2	Braking with "quick stop ramp" (deceleration depending on operating mode)
3 bis 32767	Reserved

## 603Fh Error Code

### Function

Contains the last error that occurred.

### Object description

Index	603F <sub>h</sub>
Object Name	Error Code
Object Code	VARIABLE
Data type	UNSIGNED16
Access	read only
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	0000 <sub>h</sub>
Firmware Version	
Change History	

### Description

For the meaning of the error, see object **1003<sub>h</sub>** (Pre-defined Error Field).

## 6040h Controlword

### Function

The motor is switched on and move commands can be executed with this object.

### Object description

Index	6040 <sub>h</sub>
Object Name	Controlword
Object Code	VARIABLE
Data type	UNSIGNED16
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	0000 <sub>h</sub>
Firmware Version	
Change History	

### Description

This object controls the **DS402 Power State machine**. The function of parts of the object are depending 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": Switch to the "Quick stop" state

**EO (Enable Operation)**

Value = "1": Switch to the "Enable operation" state

**OMS [3], OMS**

Meaning depends on the selected operating mode

**FR (Fault Reset)**

Resets an error (if possible)

**HALT**

Value = "1": Triggers a stop

**6041h Statusword**

**Function**

This object queries whether the state commanded with object **6040<sub>h</sub>** (control word) has been reached.

**Object description**

---

Index	6041 <sub>h</sub>
Object Name	Statusword
Object Code	VARIABLE
Data type	UNSIGNED16
Access	read only
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	0000 <sub>h</sub>
Firmware Version	
Change History	

---

**Description**

This object controls the **DS402 Power State machine**. The function of parts of the object are depending on the currently selected mode.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLA		OMS [2]	ILA	TARG	REM		WARN	SOD	QS	VE	FAULT	OE	SO	RTSO	

**RTSO (Ready To Switch On)**

Value = "1": Motor controller is in the "Ready To Switch On" state

**SO (Switched On)**

Value = "1": Motor controller is in the "Switched On" state

**OE (Operational Enabled)**

Value = "1": Motor controller is in the state "Operational Enabled" state

**FAULT**

Error occurred

**VE (Voltage Enabled)**

Voltage created

**QS (Quick Stop)**

Value = "1": Motor controller is in the "Quick Stop" state

**SOD (Switched On Enabled)**

Value = "1": Motor controller is in the "Switched On Enabled" state

**WARN (Warning)**

Value = "1": Warning

**REM (Remote)**

Remote (value of bit always "1")

**TARG**

Target specification reached

**ILA (Internal Limit Reached)**

Limit exceeded

**OMS (Operation Mode Specific)**

Meaning depends on the selected operating mode

**CLA (Closed Loop Available)**

Value = "1": AutoSetup successful and closed loop possible

**6042h VI Target Velocity**

**Function**

Specifies the target speed in user units.

**Object description**

Index	6042 <sub>h</sub>
Object Name	VI Target Velocity
Object Code	VARIABLE
Data type	INTEGER16
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	

Specified Value	00C8 <sub>h</sub>
Firmware Version	
Change History	

## 6043h VI Velocity Demand

### Function

Specifies the actual target speed in user units.

### Object description

Index	6043 <sub>h</sub>
Object Name	VI Velocity Demand
Object Code	VARIABLE
Data type	INTEGER16
Access	read only
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	0000 <sub>h</sub>
Firmware Version	
Change History	

## 6044h VI Velocity Actual Value

### Function

Specifies the current actual speed in user units.

In open loop mode, the source of this object can be set either to the internal, calculated value or to the encoder with object **320A<sub>h</sub>:03<sub>h</sub>**.

In closed loop mode, the source of this object can be set either to the internal, calculated value or to the encoder with object **320B<sub>h</sub>:03<sub>h</sub>**.

### Object description

Index	6044 <sub>h</sub>
Object Name	VI Velocity Actual Value
Object Code	VARIABLE
Data type	INTEGER16
Access	read only
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	0000 <sub>h</sub>
Firmware Version	
Change History	

## 6046h VI Velocity Min Max Amount

### Function

The minimum speed and maximum speed in user units can be set with this object.

### Object description

Index	6046 <sub>h</sub>
Object Name	VI Velocity Min Max Amount
Object Code	ARRAY
Data type	UNSIGNED32
Firmware Version	
Change History	

### Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	02 <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	MinAmount
Data type	UNSIGNED32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>

Subindex	02 <sub>h</sub>
Name	MaxAmount
Data type	UNSIGNED32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00004E20 <sub>h</sub>

### Description

Subindex 1 contains the minimum speed.

Subindex 2 contains the maximum speed.

**Note**

If the specified target speed (object **6042<sub>h</sub>**) is less than the minimum speed, the minimum speed applies. If the target speed is 0, the motor stops.

A target speed greater than the maximum speed sets the speed to the maximum speed and sets bit 11 "Limit exceeded" in object **6041<sub>h</sub>** (status word).

## 6048h VI Velocity Acceleration

### Function

Sets the acceleration ramp in velocity mode ( see "**Velocity**").

### Object description

Index	6048 <sub>h</sub>
Object Name	VI Velocity Acceleration
Object Code	RECORD
Data type	VELOCITY_ACCELERATION_DECELERATION
Firmware Version	
Change History	

### Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	02 <sub>h</sub>
Subindex	01 <sub>h</sub>
Name	DeltaSpeed
Data type	UNSIGNED32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	000001F4 <sub>h</sub>
Subindex	02 <sub>h</sub>
Name	DeltaTime
Data type	UNSIGNED16
Access	read/write
PDO Mapping	RX-PDO



Admissible Values

Specified Value                      0001<sub>h</sub>

**Description**

The acceleration is specified as a fraction:

Speed change per time change.

Subindex 01<sub>h</sub>: Contains the speed change in steps per second (U32).

Subindex 02<sub>h</sub>: Contains the time change in seconds (U16).

**6049h VI Velocity Deceleration**

**Function**

Sets the braking ramp in velocity mode ( **see "Velocity"**).

**Object description**

Index	6049 <sub>h</sub>
Object Name	VI Velocity Deceleration
Object Code	RECORD
Data type	VELOCITY_ACCELERATION_DECELERATION
Firmware Version	
Change History	

**Value description**

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	02 <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	DeltaSpeed
Data type	UNSIGNED32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	000001F4 <sub>h</sub>

Subindex	02 <sub>h</sub>
Name	DeltaTime
Data type	UNSIGNED16

---

Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	0001 <sub>h</sub>

---

## 604Ah VI Velocity Quick Stop

### Function

This object defines the deceleration if the quick stop state is initiated in velocity mode.

### Object description

---

Index	604A <sub>h</sub>
Object Name	VI Velocity Quick Stop
Object Code	RECORD
Data type	VELOCITY_ACCELERATION_DECELERATION
Firmware Version	
Change History	

---

### Value description

---

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	02 <sub>h</sub>

---



---

Subindex	01 <sub>h</sub>
Name	DeltaSpeed
Data type	UNSIGNED32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00001388 <sub>h</sub>

---



---

Subindex	02 <sub>h</sub>
Name	DeltaTime
Data type	UNSIGNED16
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	

---

Specified Value	0001 <sub>h</sub>
-----------------	-------------------

### Description

Subindex 1 contains the speed change, and subindex 2 contains the associated time in seconds.

Both together are computed as the acceleration:

Velocity Quick Stop = DeltaSpeed ( **604A<sub>h</sub>:01<sub>h</sub>** ) / DeltaTime ( **604A<sub>h</sub>:02<sub>h</sub>** )

## 604Ch VI Dimension Factor

### Function

The unit for the speed specifications for the objects that pertain to the Velocity Mode are defined here.

### Object description

Index	604C <sub>h</sub>
Object Name	VI Dimension Factor
Object Code	ARRAY
Data type	INTEGER32
Firmware Version	
Change History	

### Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	02 <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	VI Dimension Factor Numerator
Data type	INTEGER32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000001 <sub>h</sub>

Subindex	02 <sub>h</sub>
Name	VI Dimension Factor Denominator
Data type	INTEGER32
Access	read/write
PDO Mapping	RX-PDO

Admissible Values

Specified Value 0000003C<sub>h</sub>

**Description**

If subindices 1 and 2 are set to the value "1", the speed is indicated in revolutions per minute.

Otherwise, subindex 1 contains the denominator (multiplier) and subindex 2 the numerator (divisor) with which the speed specifications are computed.

The result is interpreted as revolutions per second, at **2060<sub>h</sub>** the selection is made of whether these are electrical ( **2060<sub>h</sub>** = 0) or mechanical ( **2060<sub>h</sub>** = 1) revolutions per second.

**605Ah Quick Stop Option Code**

**Function**

The object contains the action to be executed when the **DS402 Power State machine** transitions to the quick stop state.

**Object description**

---

Index	605A <sub>h</sub>
Object Name	Quick Stop Option Code
Object Code	VARIABLE
Data type	INTEGER16
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	0001 <sub>h</sub>
Firmware Version	
Change History	

---

**Description**

Value	Description
-32768 to -1	Reserved
0	Immediate stop with short-circuit braking
1	Braking with "slow down ramp" (deceleration 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"
3 to 32767	Reserved

**605Bh Shutdown Option Code**

**Function**

The object contains the action to be executed when the **DS402 Power State machine** transitions from the "Operation enabled" state to the "Ready to switch on" state.

### Object description

Index	605B <sub>h</sub>
Object Name	Shutdown Option Code
Object Code	VARIABLE
Data type	INTEGER16
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	0001 <sub>h</sub>
Firmware Version	
Change History	

### Description

Value	Description
-32768 to -1	Reserved
0	Immediate stop with short-circuit braking
1	Braking with "slow down ramp" (deceleration depending on operating mode) and subsequent state change to "Switch on disabled"
2 to 32767	Reserved

### 605Ch Disable Option Code

#### Function

The object contains the action to be executed when the **DS402 Power State machine** transitions from the "Operation enabled" state to the "Switched on" state.

### Object description

Index	605C <sub>h</sub>
Object Name	Disable Option Code
Object Code	VARIABLE
Data type	INTEGER16
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	0001 <sub>h</sub>
Firmware Version	
Change History	

### Description

Value	Description
-32768 to -1	Reserved

Value	Description
0	Immediate stop with short-circuit braking
1	Braking with "slow down ramp" (deceleration depending on operating mode) and subsequent state change to "Switch on disabled"
2 to 32767	Reserved

## 605Dh Halt Option Code

### Function

The object contains the action to be executed if stop bit 8 is set in control word **6040<sub>h</sub>**.

### Object description

Index	605D <sub>h</sub>
Object Name	Halt Option Code
Object Code	VARIABLE
Data type	INTEGER16
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	0001 <sub>h</sub>
Firmware Version	
Change History	

### Description

Value	Description
-32768 to 0	Reserved
1	Braking with "slow down ramp" (deceleration depending on operating mode)
2	Braking with "quick stop ramp" (deceleration depending on operating mode)
3 to 32767	Reserved

## 605Eh Fault Option Code

### Function

The object contains the action that is to be executed when the motor needs to be brought to idling in case of a fault.

### Object description

Index	605E <sub>h</sub>
Object Name	Fault Option Code
Object Code	VARIABLE
Data type	INTEGER16

Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	0002 <sub>h</sub>
Firmware Version	
Change History	

### Description

Value	Description
-32768 to -1	Reserved
0	Immediate stop with short-circuit braking
1	Braking with "slow down ramp" (deceleration depending on operating mode)
2	Braking with "quick stop ramp" (deceleration depending on operating mode)
3 to 32767	Reserved

## 6060h Modes Of Operation

### Function

The desired operating mode is entered in this object.

### Object description

Index	6060 <sub>h</sub>
Object Name	Modes Of Operation
Object Code	VARIABLE
Data type	INTEGER8
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00 <sub>h</sub>
Firmware Version	
Change History	

### Description

Mode	Description
-128 to -2	Manufacturer-specific operation modes
-1	Clock/direction mode
0	No mode change/no mode assigned
1	Profile Position Mode
2	Velocity Mode
3	Profile Velocity Mode

Mode	Description
4	Profile Torque Mode
5	Reserved
6	Homing Mode
7	Not assigned
8	Cyclic Synchronous Position Mode
9	Cyclic Synchronous Velocity Mode
10	Cyclic Sync Torque Mode
11 to 127	Reserved

## 6061h Modes Of Operation Display

### Function

Contains the current operating mode set in object **6060<sub>h</sub>** ("Modes Of Operation").

### Object description

Index	6061 <sub>h</sub>
Object Name	Modes Of Operation Display
Object Code	VARIABLE
Data type	INTEGER8
Access	read only
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	00 <sub>h</sub>
Firmware Version	
Change History	

## 6062h Position Demand Value

### Function

Specifies the actual set position in user units.

### Object description

Index	6062 <sub>h</sub>
Object Name	Position Demand Value
Object Code	VARIABLE
Data type	INTEGER32
Access	read only
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>



Firmware Version

Change History

## 6063h Position Actual Internal Value

### Function

Contains the actual encoder position in cycles since the drive was switched on.

### Object description

Index	6063 <sub>h</sub>
Object Name	Position Actual Internal Value
Object Code	VARIABLE
Data type	INTEGER32
Access	read only
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Firmware Version	
Change History	

## 6064h Position Actual Value

### Function

Contains the current actual position (encoder position converted according to Feed Constant ( **6092**) and Gear Ratio ( **6091** and reference position)

In open loop mode, the source of this object can be set either to the internal, calculated value or to the encoder with object **320A<sub>h</sub>:04<sub>h</sub>**.

In closed loop mode, the source of this object can be set either to the internal, calculated value or to the encoder with object **320B<sub>h</sub>:04<sub>h</sub>**.

### Object description

Index	6064 <sub>h</sub>
Object Name	Position Actual Value
Object Code	VARIABLE
Data type	INTEGER32
Access	read only
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Firmware Version	
Change History	

## 6065h Following Error Window

### Function

Specifies the maximum following error symmetrically to the demanded position.

### Object description

---

Index	6065 <sub>h</sub>
Object Name	Following Error Window
Object Code	VARIABLE
Data type	UNSIGNED32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000100 <sub>h</sub>
Firmware Version	
Change History	

---

### Description

If the difference between the actual position and the set position is so large that value of this object is exceeded, bit 11 is set for "Limit exceeded" in object **6041<sub>h</sub>** (status word). The deviation must be longer than the time in object **6066<sub>h</sub>**.

To obtain an automatic response to the error, bit 7 must be activated in object **3202<sub>h</sub>**. A fault is generated if the following error originates - and reacts correspondingly to it ( **6041<sub>h</sub>** bit 3 "Error occurred").

## 6066h Following Error Time Out

### Function

Time in milliseconds until too large a following error leads to an error message.

### Object description

---

Index	6066 <sub>h</sub>
Object Name	Following Error Time Out
Object Code	VARIABLE
Data type	UNSIGNED16
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	0064 <sub>h</sub>
Firmware Version	
Change History	

---

## Description

If the difference between the actual position and the set position is so large that the value of object **6065<sub>h</sub>** is exceeded, bit 11 for "Limit exceeded" is set in the status word **6041<sub>h</sub>**. The deviation must be longer than the time in this object.

To obtain an automatic response to the error, bit 7 must be activated in object **3202<sub>h</sub>**. A fault is generated if the following error originates - and reacts correspondingly to it ( **6041<sub>h</sub>** bit 3 "Error occurred").

## 6067h Position Window

### Function

Specifies a symmetrical range relative to the target position within which the target is considered to be reached.

### Object description

---

Index	6067 <sub>h</sub>
Object Name	Position Window
Object Code	VARIABLE
Data type	UNSIGNED32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	0000000A <sub>h</sub>
Firmware Version	
Change History	

---

## 6068h Position Window Time

### Function

For this time period in milliseconds, the actual position must be within the position window ( **6067**) so that the target position is considered to be reached.

### Object description

---

Index	6068 <sub>h</sub>
Object Name	Position Window Time
Object Code	VARIABLE
Data type	UNSIGNED16
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	0064 <sub>h</sub>
Firmware Version	
Change History	

---

## 606Bh Velocity Demand Value

### Function

Speed specification for the control in the Profile Velocity Mode.

This object is computed with user-defined units (see also **User-defined units**). The motor controller is delivered with the units set to rpm.

### Object description

---

Index	606B <sub>h</sub>
Object Name	Velocity Demand Value
Object Code	VARIABLE
Data type	INTEGER32
Access	read only
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Firmware Version	
Change History	

---

### Description

This object contains the output of the ramp generator which is the specified value for the speed controller at the same time.

## 606Ch Velocity Actual Value

### Function

The current actual speed in the profile velocity mode.

### Object description

---

Index	606C <sub>h</sub>
Object Name	Velocity Actual Value
Object Code	VARIABLE
Data type	INTEGER32
Access	read only
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Firmware Version	
Change History	

---

## 606Dh Velocity Window

### Function

Speed window for the Profile Velocity Mode.

### Object description

---

Index	606D <sub>h</sub>
Object Name	Velocity Window
Object Code	VARIABLE
Data type	UNSIGNED16
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	0000 <sub>h</sub>
Firmware Version	
Change History	

---

### Description

This value specifies by how much the actual speed may vary from the set speed for bit 10 "Target reached" in status word ( **6041<sub>h</sub>**) to be set to "1".

## 606Eh Velocity Window Time

### Function

Time window for the Profile Velocity Mode.

### Object description

---

Index	606E <sub>h</sub>
Object Name	Velocity Window Time
Object Code	VARIABLE
Data type	UNSIGNED16
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	0000 <sub>h</sub>
Firmware Version	
Change History	

---

### Description

This object specifies how long the actual speed and the set speed must be near each other in magnitude (see **606D<sub>h</sub>**) for bit 10 "Target reached" in status word ( **6041<sub>h</sub>**) to be set to "1".

## 6071h Target Torque

### Function

This object contains the target torque for the Profile Torque Mode.

### Object description

---

Index	6071 <sub>h</sub>
Object Name	Target Torque
Object Code	VARIABLE
Data type	INTEGER16
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	0000 <sub>h</sub>
Firmware Version	
Change History	

---

## 6072h Max Torque

### Function

The object describes the maximum torque.

### Object description

---

Index	6072 <sub>h</sub>
Object Name	Max Torque
Object Code	VARIABLE
Data type	UNSIGNED16
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	0000 <sub>h</sub>
Firmware Version	
Change History	

---

## 6074h Torque Demand

### Function

Current output value of the ramp generator (torque) for the internal control.

### Object description

---

Index	6074 <sub>h</sub>
-------	-------------------

---

Object Name	Torque Demand
Object Code	VARIABLE
Data type	INTEGER16
Access	read only
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	0000 <sub>h</sub>
Firmware Version	
Change History	

## 607Ah Target Position

### Function

This object specifies the target position.

### Object description

Index	607A <sub>h</sub>
Object Name	Target Position
Object Code	VARIABLE
Data type	INTEGER32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000FA0 <sub>h</sub>
Firmware Version	
Change History	

## 607Bh Position Range Limit

### Function

Contains the minimum and maximum position.

### Object description

Index	607B <sub>h</sub>
Object Name	Position Range Limit
Object Code	ARRAY
Data type	INTEGER32
Firmware Version	
Change History	

### Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	02 <sub>h</sub>
Subindex	01 <sub>h</sub>
Name	Min Position Range Limit
Data type	INTEGER32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	80000001 <sub>h</sub>
Subindex	02 <sub>h</sub>
Name	Max Position Range Limit
Data type	INTEGER32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	7FFFFFFE <sub>h</sub>

### Description

If this range is exceeded or undercut, an overflow occurs. Limit values for the target position can be set in object **607D<sub>h</sub>** ("Software Position Limit") to prevent this overflow.

## 607Ch Home Offset

### Function

Specifies the difference between the zero position of the application and the reference point the machine. This object is computed in the same unit used for calculation for object **607A<sub>h</sub>** (see **User-defined units**).

### Object description

Index	607C <sub>h</sub>
Object Name	Home Offset
Object Code	VARIABLE
Data type	INTEGER32
Access	read/write
PDO Mapping	RX-PDO



Admissible Values	
Specified Value	00000000 <sub>h</sub>
Firmware Version	
Change History	

## 607Dh Software Position Limit

### Function

Limit values for the target position.

### Object description

Index	607D <sub>h</sub>
Object Name	Software Position Limit
Object Code	ARRAY
Data type	INTEGER32
Firmware Version	
Change History	

### Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	02 <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	Min Position Limit
Data type	INTEGER32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	80000000 <sub>h</sub>

Subindex	02 <sub>h</sub>
Name	Max Position Limit
Data type	INTEGER32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	

Specified Value                      7FFFFFFF<sub>h</sub>

**Description**

The target position must lie within the limits set here. Before the check, the home offset ( **607C<sub>h</sub>**) is deducted in each case:

Corrected min position limit = min position limit - home offset

Corrected max position limit = max position limit - home offset.

**607Eh Polarity**

**Function**

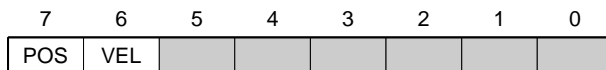
This object can be used to reverse the direction of rotation.

**Object description**

Index	607E <sub>h</sub>
Object Name	Polarity
Object Code	VARIABLE
Data type	UNSIGNED8
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>
Firmware Version	
Change History	

**Description**

The general rule for direction reversal is: Reversal is activated if a bit is set to the value "1". If the value is "0", the direction of rotation is as specified in the respective mode



**VEL (Velocity)**

Reversal of the direction of rotation in the following modes:

- Profile Velocity Mode
- Cyclic Synchronous Velocity Mode
- Velocity Mode

**POS (Position)**

Reversal of the direction of rotation in the following modes:

- Profile Position Mode
- Cyclic Synchronous Position Mode

## 6081h Profile Velocity

### Function

Specifies the maximum traveling speed in revolutions per second.

This object is computed with user-defined units (see **User-defined units**). The motor controller is delivered with the units set to revolutions per minute.

### Object description

---

Index	6081 <sub>h</sub>
Object Name	Profile Velocity
Object Code	VARIABLE
Data type	UNSIGNED32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	000001F4 <sub>h</sub>
Firmware Version	
Change History	

---

## 6082h End Velocity

### Function

Specifies the speed at the end of the traveled ramp.

This object is computed with user-defined units (see **User-defined units**). The motor controller is delivered with the units set to revolutions per minute.

### Object description

---

Index	6082 <sub>h</sub>
Object Name	End Velocity
Object Code	VARIABLE
Data type	UNSIGNED32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Firmware Version	
Change History	

---

## 6083h Profile Acceleration

### Function

Specifies the maximum acceleration in revolutions/s<sup>2</sup>.

### Object description

Index	6083 <sub>h</sub>
Object Name	Profile Acceleration
Object Code	VARIABLE
Data type	UNSIGNED32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	000001F4 <sub>h</sub>
Firmware Version	
Change History	

### 6084h Profile Deceleration

#### Function

Specifies the maximum deceleration in revolutions/s<sup>2</sup>.

#### Object description

Index	6084 <sub>h</sub>
Object Name	Profile Deceleration
Object Code	VARIABLE
Data type	UNSIGNED32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	000001F4 <sub>h</sub>
Firmware Version	
Change History	

### 6085h Quick Stop Deceleration

#### Function

Specifies the maximum quick stop deceleration in revolutions/s<sup>2</sup>.

#### Object description

Index	6085 <sub>h</sub>
Object Name	Quick Stop Deceleration
Object Code	VARIABLE
Data type	UNSIGNED32
Access	read/write
PDO Mapping	RX-PDO

Admissible Values	
Specified Value	00001388 <sub>h</sub>
Firmware Version	
Change History	

## 6086h Motion Profile Type

### Function

Specifies the ramp type.

### Object description

Index	6086 <sub>h</sub>
Object Name	Motion Profile Type
Object Code	VARIABLE
Data type	INTEGER16
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	0000 <sub>h</sub>
Firmware Version	
Change History	

### Description

Value = "0": = trapezoid ramp

Value = "3": Jerk-limited ramp

## 6087h Torque Slope

### Function

This object contains the torque slope in torque mode.

### Object description

Index	6087 <sub>h</sub>
Object Name	Torque Slope
Object Code	VARIABLE
Data type	UNSIGNED32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Firmware Version	
Change History	

## 608Fh Position Encoder Resolution

### Function

Encoder cycles per revolution.

### Object description

Index	608F <sub>h</sub>
Object Name	Position Encoder Resolution
Object Code	ARRAY
Data type	UNSIGNED32
Firmware Version	
Change History	

### Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	02 <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	Encoder Increments
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	000007D0 <sub>h</sub>

Subindex	02 <sub>h</sub>
Name	Motor Revolutions
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000001 <sub>h</sub>

### Description

Position encoder resolution = encoder cycles (608F<sub>h</sub>:01<sub>h</sub>) / motor revolutions (608F<sub>h</sub>:02<sub>h</sub>)

## 6091h Gear Ratio

### Function

Number of motor revolutions per revolution of the output axis.

### Object description

Index	6091 <sub>h</sub>
Object Name	Gear Ratio
Object Code	ARRAY
Data type	UNSIGNED32
Firmware Version	
Change History	

### Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	02 <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	Motor Revolutions
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000001 <sub>h</sub>

Subindex	02 <sub>h</sub>
Name	Shaft Revolutions
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000001 <sub>h</sub>

### Description

Gear ratio = motor revolutions (6091<sub>h</sub>:01<sub>h</sub>) / shaft revolutions (6091<sub>h</sub>:02<sub>h</sub>)

## 6092h Feed Constant

### Function

Feed per revolution for a linear drive.

### Object description

Index	6092 <sub>h</sub>
Object Name	Feed Constant
Object Code	ARRAY
Data type	UNSIGNED32
Firmware Version	
Change History	

### Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	02 <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	Feed
Data type	UNSIGNED32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000001 <sub>h</sub>

Subindex	02 <sub>h</sub>
Name	Shaft Revolutions
Data type	UNSIGNED32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000001 <sub>h</sub>

### Description

Feed Constant = feed (6092<sub>h</sub>:01<sub>h</sub>) / shaft revolutions (6092<sub>h</sub>:02<sub>h</sub>)



## 6098h Homing Method

### Function

This object selects the homing mode.

### Object description

Index	6098 <sub>h</sub>
Object Name	Homing Method
Object Code	VARIABLE
Data type	INTEGER8
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	23 <sub>h</sub>
Firmware Version	
Change History	

## 6099h Homing Speed

### Function

Specifies the speeds for homing mode ( **6098<sub>h</sub>** ) in revolutions per second.

This object is computed with user-defined units (see **User-defined units**). The motor controller is delivered with the units set to revolutions per minute.

### Object description

Index	6099 <sub>h</sub>
Object Name	Homing Speed
Object Code	ARRAY
Data type	UNSIGNED32
Firmware Version	
Change History	

### Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	02 <sub>h</sub>
Subindex	01 <sub>h</sub>

Name	Speed During Search For Switch
Data type	UNSIGNED32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000032 <sub>h</sub>
Subindex	02 <sub>h</sub>
Name	Speed During Search For Zero
Data type	UNSIGNED32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000001 <sub>h</sub>

### Description

This value is computed with the numerator in object **2061<sub>h</sub>** and the denominator in object **2062<sub>h</sub>**.

The speed for the search of the switch is specified in subindex 1.

The (lower) speed for the search for the reference position is specified in Subindex 2.

#### Note

- The speed in Subindex 2 is also the starting speed for starting the acceleration ramp. If this is set too high, the motor loses steps or does not rotate at all. An excessive setting leads to the index marking being overlooked. The speed in subindex 2 should therefore be below 1000 steps per second.
- The speed in subindex 1 must be greater than the speed in subindex 2.

## 609Ah Homing Acceleration

### Function

Specifies the acceleration ramp for homing mode in steps/s<sup>2</sup>.

### Object description

Index	609A <sub>h</sub>
Object Name	Homing Acceleration
Object Code	VARIABLE
Data type	UNSIGNED32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	000001F4 <sub>h</sub>
Firmware Version	

## Change History

---

### Description

The ramp is only used when starting off. When the switch is reached, the unit is automatically switched to the lower speed and is stopped as soon as it reaches the limit position.

## 60A4h Profile Jerk

### Function

In case of a jerk-limited ramp, the magnitude of the jerk can be entered in this object. An entry with the value "0" means that the jerk is not limited.

### Object description

Index	60A4 <sub>h</sub>
Object Name	Profile Jerk
Object Code	ARRAY
Data type	UNSIGNED32
Firmware Version	
Change History	

### Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	04 <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	Begin Acceleration Jerk
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	000003E8 <sub>h</sub>

Subindex	02 <sub>h</sub>
Name	End Acceleration Jerk
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no

Admissible Values	
Specified Value	000003E8 <sub>h</sub>
Subindex	
Subindex	03 <sub>h</sub>
Name	Begin Deceleration Jerk
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	000003E8 <sub>h</sub>
Subindex	
Subindex	04 <sub>h</sub>
Name	End Deceleration Jerk
Data type	UNSIGNED32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	000003E8 <sub>h</sub>

## 60C2h Interpolation Time Period

### Function

DESCRIPTION!

### Object description

Index	60C2 <sub>h</sub>
Object Name	Interpolation Time Period
Object Code	RECORD
Data type	INTERPOLATION_TIME_PERIOD
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	
Firmware Version	
Change History	

### Value description

Subindex	
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only

PDO Mapping	no
Admissible Values	
Specified Value	02 <sub>h</sub>
<hr/>	
Subindex	
Name	Interpolation Time Period Value
Data type	UNSIGNED8
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	01 <sub>h</sub>
<hr/>	
Subindex	
Name	Interpolation Time Index
Data type	INTEGER8
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00 <sub>h</sub>

## Description

### 60C5h Max Acceleration

#### Function

This object contains the maximum admissible acceleration ramp.

For the braking ramp: See object **60C6<sub>h</sub>** "Max Deceleration".

#### Object description

Index	60C5 <sub>h</sub>
Object Name	Max Acceleration
Object Code	VARIABLE
Data type	UNSIGNED32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	FFFFFFFD <sub>h</sub>
Firmware Version	
Change History	

## 60C6h Max Deceleration

### Function

This object contains the maximum admissible braking ramp.

For the acceleration ramp: See object **60C5h** "Max Deceleration".

### Object description

---

Index	60C6 <sub>h</sub>
Object Name	Max Deceleration
Object Code	VARIABLE
Data type	UNSIGNED32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00001388 <sub>h</sub>
Firmware Version	
Change History	

---

## 60F4h Following Error Actual Value

### Function

This object contains the current following error.

### Object description

---

Index	60F4 <sub>h</sub>
Object Name	Following Error Actual Value
Object Code	VARIABLE
Data type	INTEGER32
Access	read only
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Firmware Version	
Change History	

---

### Description

This object is calculated in user-defined units (see **User-defined units**).

## 60FDh Digital Inputs

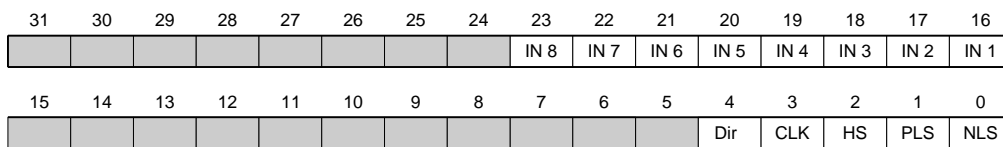
### Function

The digital inputs of the motor can be read with this object.

### Object description

Index	60FD <sub>h</sub>
Object Name	Digital Inputs
Object Code	VARIABLE
Data type	UNSIGNED32
Access	read only
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Firmware Version	
Change History	

### Description



#### NLS (Negative Limit Switch)

Negative limit switch

#### PLS (Positive Limit Switch)

Positive limit switch

#### HS (Home Switch)

Reference switch

#### CLK (Clock)

Clock input

#### DIR (Direction)

Directional input

#### IN n (Input n)

Input n - the number of used bits is depending on the respective motor controller.

## 60FEh Digital Outputs

### Function

The digital outputs of the motor can be written with this object.

### Object description

Index	60FE <sub>h</sub>
Object Name	Digital Outputs
Object Code	ARRAY
Data type	UNSIGNED32

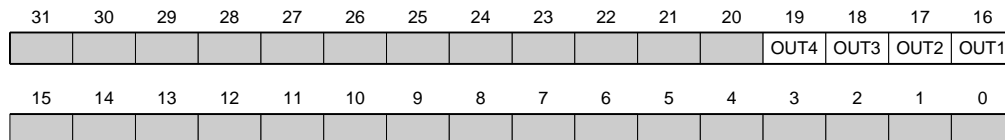
Firmware Version  
Change History

### Value description

Subindex	00 <sub>h</sub>
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	01 <sub>h</sub>
Subindex	01 <sub>h</sub>
Name	Digital Outputs #1
Data type	UNSIGNED32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>

### Description

The entries in object **3250<sub>h</sub>**, subindex 02<sub>h</sub> to 05<sub>h</sub> still have to be taken into account for writing the outputs.



#### OUT n (Output No n)

Bit for the respective digital output, the exact number of digital outputs is dependent on the motor controller.

## 60FFh Target Velocity

### Function

The target speed for the Profile Velocity Mode is entered in this object.

This object is computed with user-defined units (see **User-defined units**). The motor controller is delivered with the units set to revolutions per minute.

### Object description

Index	60FF <sub>h</sub>
Object Name	Target Velocity
Object Code	VARIABLE



Data type	INTEGER32
Access	read/write
PDO Mapping	RX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>
Firmware Version	
Change History	

## 6402h Read Analogue Input

### Function

DESCRIPTION!

### Object description

Index	6402 <sub>h</sub>
Object Name	Read Analogue Input
Object Code	ARRAY
Data type	INTEGER32
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	
Firmware Version	
Change History	

### Value description

Subindex	
Name	Number Of Analogue Inputs
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	02 <sub>h</sub>

Subindex	
Name	Analogue Input 1
Data type	INTEGER32
Access	read only
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>

Subindex	
Name	Analogue Input 2
Data type	INTEGER32
Access	read only
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	00000000 <sub>h</sub>

### Description

## 6431h Analogue Input Offset

### Function

Offset that is added to the read-in analog value ( **6402<sub>h</sub>**) before division with the divider from object **6432<sub>h</sub>** is carried out.

### Object description

Index	6431 <sub>h</sub>
Object Name	Analogue Input Offset
Object Code	ARRAY
Data type	INTEGER32
Firmware Version	
Change History	

### Value description

Subindex	00 <sub>h</sub>
Name	Number Of Analogue Inputs
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	02 <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	Analogue Input 1
Data type	INTEGER32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

Subindex	02 <sub>h</sub>
----------	-----------------

Name	Analogue Input 2
Data type	INTEGER32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000000 <sub>h</sub>

### Description

- Subindex 00<sub>h</sub>: Number of offsets
- Subindex 01<sub>h</sub>: Offset for analog input 1
- Subindex 02<sub>h</sub>: Offset for analog input 2

## 6432h Analogue Input Pre-scaling

### Function

Value with which the read-in analog value ( **6402<sub>h</sub>**, **6431<sub>h</sub>**) is divided before it is written into object **6402<sub>h</sub>**.

### Object description

Index	6432 <sub>h</sub>
Object Name	Analogue Input Pre-scaling
Object Code	ARRAY
Data type	INTEGER32
Firmware Version	
Change History	

### Value description

Subindex	00 <sub>h</sub>
Name	Number Of Analogue Inputs
Data type	UNSIGNED8
Access	read only
PDO Mapping	no
Admissible Values	
Specified Value	02 <sub>h</sub>

Subindex	01 <sub>h</sub>
Name	Analogue Input 1
Data type	INTEGER32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000001 <sub>h</sub>

Subindex	02 <sub>h</sub>
Name	Analogue Input 2
Data type	INTEGER32
Access	read/write
PDO Mapping	no
Admissible Values	
Specified Value	00000001 <sub>h</sub>

**Description**

- Subindex 00<sub>h</sub>: Number of dividers
- Subindex 01<sub>h</sub>: Divider for analog input 1
- Subindex 02<sub>h</sub>: Divider for analog input 2

**6502h Supported Drive Modes**

**Function**

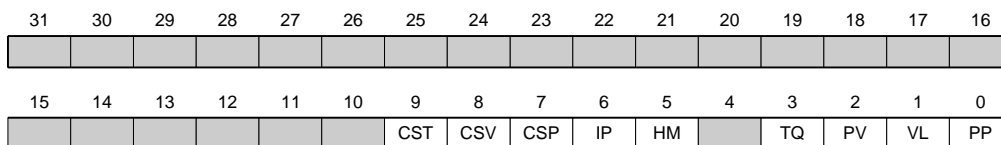
The object specifies the supported drive modes.

**Object description**

Index	6502 <sub>h</sub>
Object Name	Supported Drive Modes
Object Code	VARIABLE
Data type	UNSIGNED32
Access	read only
PDO Mapping	TX-PDO
Admissible Values	
Specified Value	000000AF <sub>h</sub>
Firmware Version	
Change History	

**Description**

A set bit specifies whether the respective mode is supported. The mode is not supported if the value of the bit is "0".



**PP**

Profile Position mode

**VL**

Velocity mode

- PV**  
Profile Velocity mode
- TQ**  
Torque mode
- HM**  
Homing (reference run) mode
- IP**  
Interpolated Position mode
- CSP**  
Cyclic Synchronous Position mode
- CSV**  
Cyclic Synchronous Velocity mode
- CST**  
Cyclic Sync Torque mode

## 6505h Http Drive Catalogue Address

### Function

This object contains the manufacturer web address.

### Object description

---

Index	6505h
Object Name	Http Drive Catalogue Address
Object Code	VARIABLE
Data type	VISIBLE_STRING
Access	read only
PDO Mapping	no
Change History	
Specified Value	<a href="http://www.nanotec.de">http://www.nanotec.de</a>
Firmware Version	
Änderungshistorie	

---

## 12 Copyright notice

### 12.1 Introduction

Components from external software manufacturers are integrated in the Nanotec software. In this section you will find copyright information on the external sources of software components.

### 12.2 AES

FIPS-197 compliant AES implementation

Based on XySSL: Copyright (C) 2006-2008 Christophe Devine

Copyright (C) 2009 Paul Bakker <polarssl\_maintainer at polarssl dot org>

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution; or, the application vendor's website must provide a copy of this notice.
- Neither the names of PolarSSL or XySSL nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The AES block cipher was designed by Vincent Rijmen and Joan Daemen.

<http://csrc.nist.gov/encryption/aes/rijndael/Rijndael.pdf>

<http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>

### 12.3 Arcfour (RC4)

Copyright (c) April 29, 1997 Kalle Kaukonen.

All Rights Reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that this copyright notice and disclaimer are retained.

THIS SOFTWARE IS PROVIDED BY KALLE KAUKONEN 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 KALLE KAUKONEN OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED

AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

## 12.4 MD5

MD5C.C - RSA Data Security, Inc., MD5 message-digest algorithm

Copyright (C) 1991-2, RSA Data Security, Inc. Created 1991. All rights reserved.

License to copy and use this software is granted provided that it is identified as the "RSA Data Security, Inc. MD5 Message-Digest Algorithm" in all material mentioning or referencing this software or this function.

License is also granted to make and use derivative works provided that such works are identified as "derived from the RSA Data Security, Inc. MD5 Message-Digest Algorithm" in all material mentioning or referencing the derived work.

RSA Data Security, Inc. makes no representations concerning either the merchantability of this software or the suitability of this software for any particular purpose. It is provided "as is" without express or implied warranty of any kind.

These notices must be retained in any copies of any part of this documentation and/or software.

## 12.5 uIP

Copyright (c) 2005, Swedish Institute of Computer Science

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the Institute nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE INSTITUTE AND CONTRIBUTORS ``AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE INSTITUTE OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

## 12.6 DHCP

Copyright (c) 2005, Swedish Institute of Computer Science

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the Institute nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE INSTITUTE AND CONTRIBUTORS ``AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE INSTITUTE OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

## 12.7 CMSIS DSP Software Library

Copyright (C) 2010 ARM Limited. All rights reserved.

## 12.8 FatFs

FatFs - FAT file system module include file R0.08 (C)ChaN, 2010

FatFs module is a generic FAT file system module for small embedded systems.

This is a free software that opened for education, research and commercial developments under license policy of following terms.

Copyright (C) 2010, ChaN, all right reserved.

The FatFs module is a free software and there is NO WARRANTY.

No restriction on use. You can use, modify and redistribute it for personal, non-profit or commercial product UNDER YOUR RESPONSIBILITY.

Redistributions of source code must retain the above copyright notice.

## 12.9 Protothreads

Protothread class and macros for lightweight, stackless threads in C++.

This was "ported" to C++ from Adam Dunkels' protothreads C library at: <http://www.sics.se/~adam/pt/>

Originally ported for use by Hamilton Jet ([www.hamiltonjet.co.nz](http://www.hamiltonjet.co.nz)) by Ben Hoyt, but stripped down for public release. See his blog entry about it for more information: <http://blog.micropledge.com/2008/07/protothreads/>

Original BSD-style license

Copyright (c) 2004-2005, Swedish Institute of Computer Science.

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the Institute nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.



This software is provided by the Institute and contributors "as is" and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall the Institute or contributors be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.