Manual C5

Fieldbus: USB

Valid with firmware version FIR-v1426
and hardware version W004

NANOTEC ELECTRONIC GmbH & Co. KG
Kapellenstraße 6
85622 Feldkirchen/Munich, Germany

Tel. +49 (0)89-900 686-0
Fax +49 (0)89 900 686-50
info@nanotec.com

Contents

1 Editorial

The firmware in our motor controllers may contain software components produced by third parties. The licensing conditions and copyrights of these code components can be found in the "**Copyright notice**" section.

Nanotec ®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

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

Translation of original manual

2 Safety instructions and warnings

2.1 Important information

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

To submit criticism, proposals and suggestions for improvement, please contact the above address or send an email to: **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.

⚠ DANGER

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

⚠ WARNING

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

⚠ CAUTION

The CAUTION sign indicates a potentially hazardous situation that, when the instruction is neglected, may **possibly** 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®.

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_h is addressed with "1003_h:05_h", subindex 0 of object 6040_h with "6040_h".

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_h**.

3.3 Bits

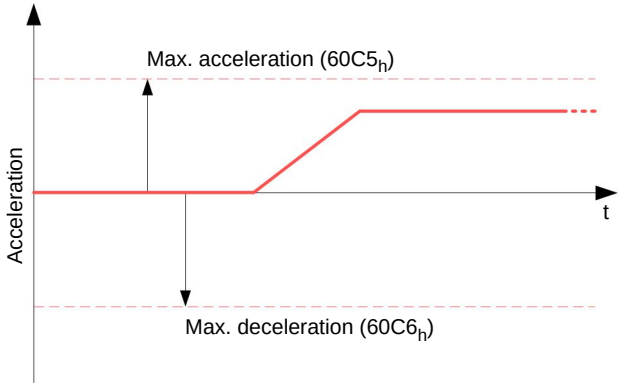
The individual bits of an object are always numbered beginning with 0 at the LSB. See the following figure, which uses the "UNSIGNED8" data type as an example.

	MSB							LSB
Bit Nummer	7	6	5	4	3	2	1	0
Bits	0	1	0	1	0	1	0	1

≙ 55_{hex} ≙ 85_{dec}

3.4 Counting direction (arrows)

In drawings, the counting direction is always in the direction of the arrow. The objects $60C5_h$ and $60C6_h$ shown in the following figure are both positive.



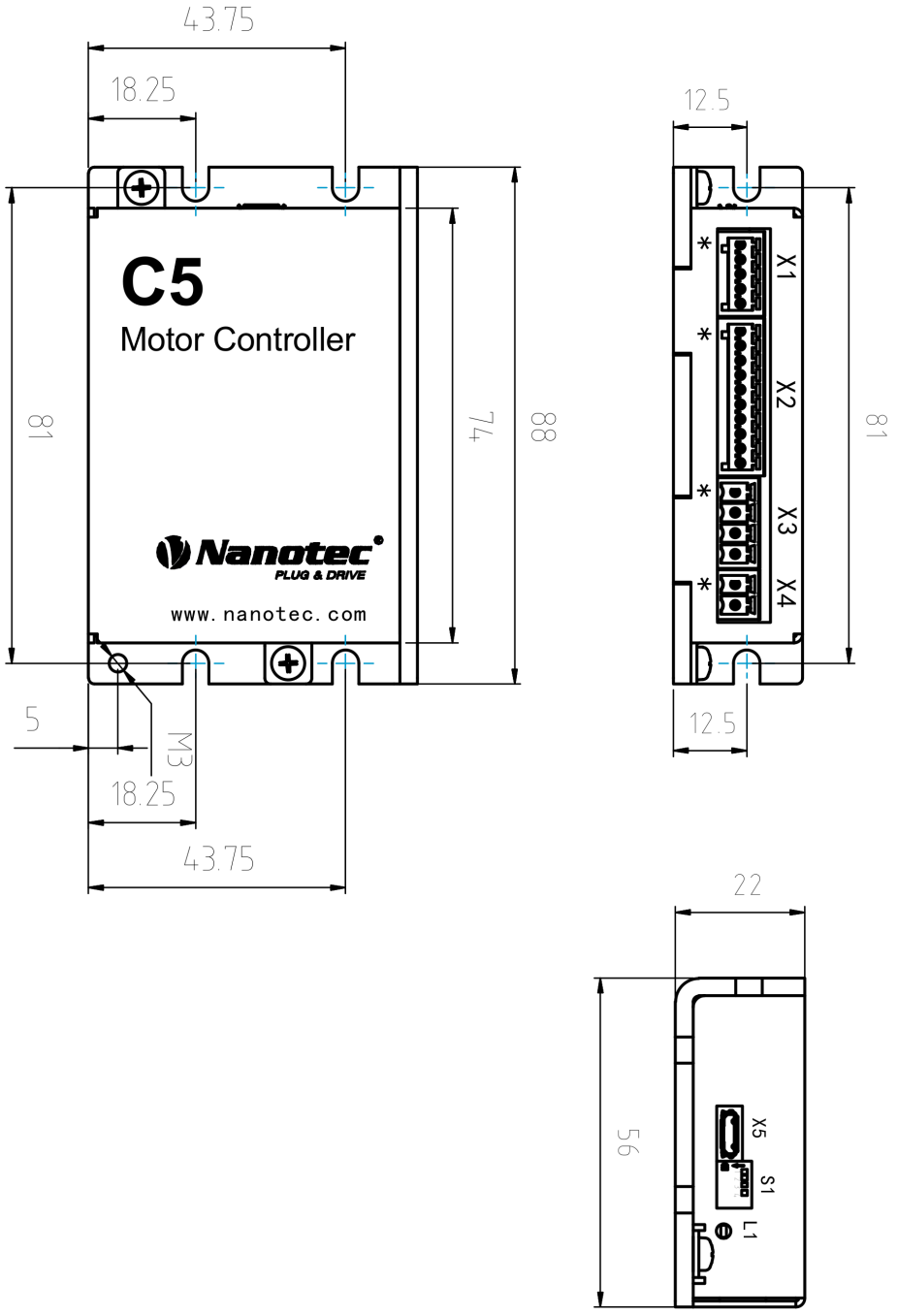
3.5 Version notes

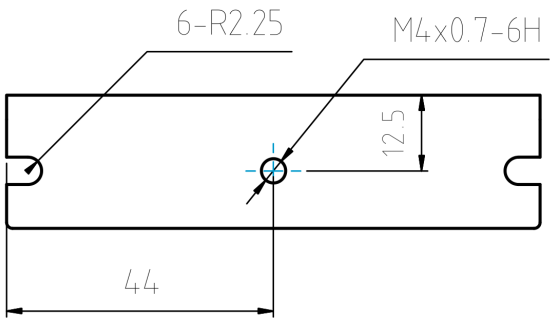
Version manual	Version firmware	Date	Changes
1.0.0		12.05.2014	Published
1.0.3		12.05.2014	Small corrections, field "Specified Value" in object dictionary description now used

1.1.0		23.07.2014.	• Added chapter "**Save Objects**", added "Persistent" to the object description
			• The following objects has been moved
			  • "Read Analog Input": from 6402_h to 3320_h
			  • "Analogue Input Offset": from 6431_h to 3321_h
			  • "Analogue Input Pre-scaling": from 6432_h to 3322_h

4 Technical data and pin configuration

4.1 Dimensioned drawings





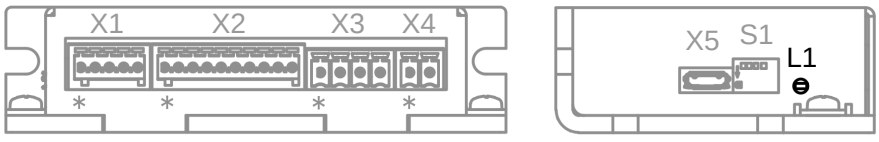
4.2 Electrical properties

Operating voltage	12 V DC to 48 V DC
Peak current	6 A effective
Commutation	Stepper motor open loop, BLDC
Operating mode	• Profile Position • Velocity • Profile velocity • Profile torque • Homing • Application program (NanoJ)
Parameterization	Clock-direction, analog, NanoJ
Field bus interfaces	None
Other interfaces	USB
Inputs	• 3 inputs 24 V (input 1 to 3) • 3 inputs, switchable 5 V/24 V, single ended or differential (inputs 4 to 6) • 1 analog input, switchable 0 V – 10 V or 0 mA – 20 mA
Outputs	2 outputs, (open drain, 0 switching, max. 24 V/100 mA)
Other	• Overvoltage and undervoltage: protective circuit at voltage > 50.5 V or < 9 V • Overtemperature: protective circuit at temperature > 70 °C • Polarity reversal protection: in case of polarity reversal, short circuit between supply voltage and GND via PIN diode, therefore cable protection device (fuse) required in supply cable.

4.3 LED signaling

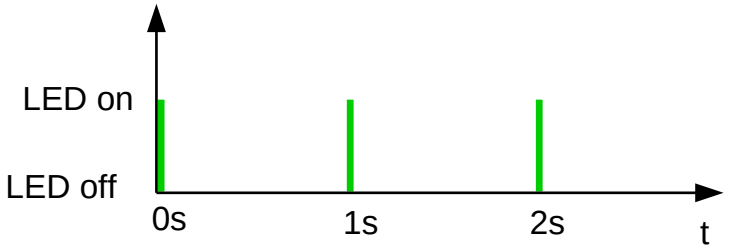
4.3.1 Position

The operating LED is located on the side with the connectors at the bottom right (labeled with "L1" in the figure).



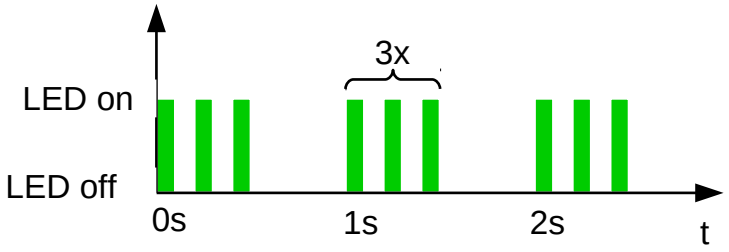
4.3.2 Normal operation

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



4.3.3 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.



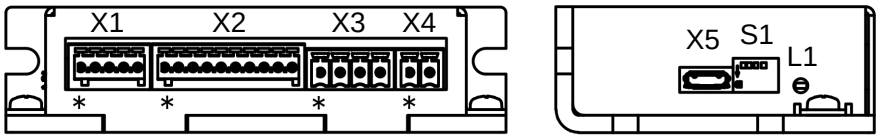
The meaning of the error number is printed in the following table.

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

Note
A considerably more exact error code is stored in object **1003**$_h$ for every error that has occurred.

4.4 Pin configuration

4.4.1 Overview

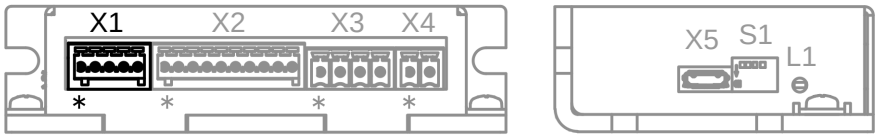


Connector	Function
X1	Analog input and digital outputs
X2	Digital inputs
X3	Motor
X4	Supply voltage

Connector	Function
X5	USB port
S1	DIP switch
L1	Operating LED

4.4.2 Analog input and digital outputs (connector X1)

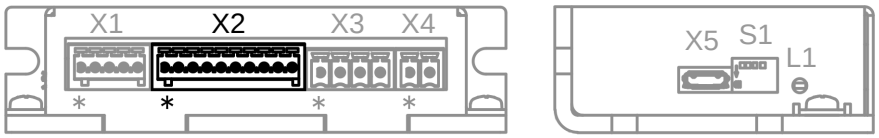
Pin 1 is marked with an asterisk "*".



PIN	Function	Remark
1	GND	
2	Analog input	
3	Digital output 1	
4	Digital output 2	
5	10 V	Maximum load: 150 mA

4.4.3 Digital inputs (connector X2)

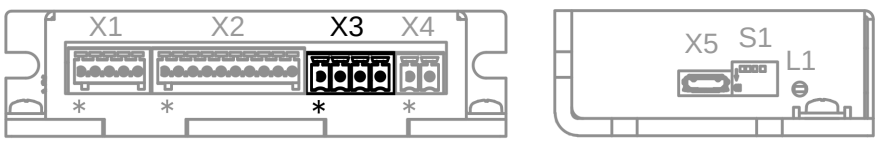
Pin 1 and Pin 2 are marked in the diagram with an asterisk "*".



PIN	Function	Remark
1	Input 1	
2	Input 2	
3	Input 3	
4	- Enable	The default settting for this combination of inputs is "single ended", therefore the input "-Enable" is deactivated, only "Enable" is measured against GND.
5	Enable	
6	- Direction	The default settting for this combination of inputs is "single ended", therefore the input "-Direction" is deactivated, only "Direction" is measured against GND.
7	Direction	
8	- Clock	The default settting for this combination of inputs is "single ended", therefore the input "-Clock" is deactivated, only "Clock" is measured against GND.
9	Clock	
10	GND	

4.4.4 Motor connection (connector X3)

Pin 1 is marked with an asterisk "*".



PIN	Stepper motors	BLDC motors
1	A	U
2	A\	V
3	B	W
4	B\	N.C.

4.4.5 Motor controller voltage supply (connector X4)

Safety instruction

⚠ CAUTION

Danger of electrical overvoltage!

- An operating voltage that is higher than the values specified above will destroy 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. Specifically, interference from other consumers or interference caused by the voltages induced by the motor must be taken into consideration here, and a voltage may need to be selected that offers sufficient safety reserves.

Voltage source

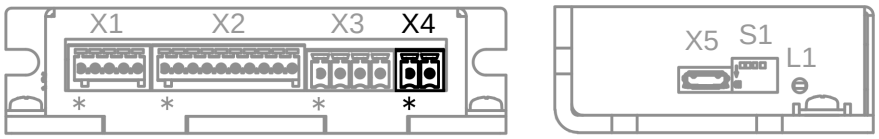
The operating or supply voltage is delivered by a battery (low voltage 12 V – 48 V), by a transformer with rectification and filtering, or better yet, by a switch-mode power supply.

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 must be added to the DC supply cable as close to the motor controller/motor as possible.

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

Connections

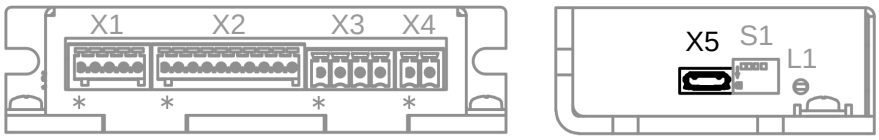
Pin 1 is marked with an asterisk "*".



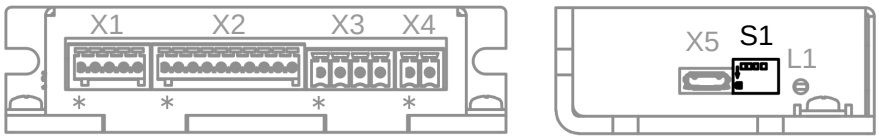
PIN	Function	Remark
1	+VB	12 V – 48 V DC
2	GND	

4.4.6 USB (connector X5)

A cable of type "Micro USB" is required for this USB port.



4.4.7 DIP switch (connector S1)



5 Configuration

5.1 General information

The following options exist for configuring the motor controller:

DIP switches

Four DIP switches are fitted on the side. More information can be found in the section " **DIP switches**".

Configuration file

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

NanoJ program

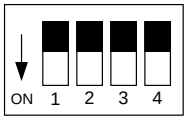
This program can be programmed, compiled, and then transferred over USB to the motor controller with NanoJ Easy. Read the sections " **USB port**" and " **Programming with NanoJ**".

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

Switch configurations:(switch 4 has no function):

1	2	3	mode		
Off	Off	Off	Clock/Direction mode		
Off	Off	On	Clock/Direction mode		
Off	On	Off	clock/direction	Automatic engine run with 30 rpm	Direction of rotation is right
Off	On	On	clock/direction	Automatic engine run with 30 rpm	Direction of rotation is left
On	Off	Off	analog speed	Direction set by "direction" input	Maximum revolution speed is 1000 rpm
On	Off	On	analog speed	Direction set by "direction" input	Maximum revolution speed is 100 rpm
On	On	Off	analog speed	Offset 5 V (joystick mode)	Maximum revolution speed is 1000 rpm
On	On	On	analog speed	Offset 5 V (joystick mode)	Maximum revolution speed is 100 rpm

Switch 4 alternates between Open-Loop (On) and Closed-Loop (Off).

The notation is:

clock/direction

Activates the clock/direction mode, therefore the pins "enable", "clock" and "direction" need to be connected (see chapter " **Digital inputs (connector X2)**").

Analog speed

Activates the analog mode, therefore the "enable" input (see chapter " **Digital inputs (connector X2)**") and the analog input (see chapter " **Analog input and digital outputs (connector X1)**") needs to be connected.

Automatic engine run with 30 rpm

The motor turns with 30 rpm if the "enable" input is set (see chapter " **Digital inputs (connector X2)**").

Direction set by "direction" input

In this mode the "direction" input determines the direction of rotation left/right, the analoge voltage determines the revolution speed.

Offset 5 V (joystick mode)

If the switch is set in analoge mode, the analoge input is split into two locial halfs: from 0 V to 5 V the direction of rotation is left and at 5 V to 10 V the direction of rotation is right. At 5 V the motor is stopped, the more away the voltage from 5 V, the higher the revolution speed is. The maximum revolution speed at 0 V and 10 V is determined by switch 3.

Maximum revolution speed is NNN rpm

In analog speed mode this switch determines the maximum revolution speed which is attained at maximum or minum analoge input voltage.

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

  All other filesare **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 (see the " **DS402 Power State machine**" section).
- 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 NanoJ 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 first if you have not already done so.

The configuration file `cfg.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 connector X4 (see the " **Motor controller voltage supply (connector X4)**" 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 the Explorer to the directory for the motor controller. The file " `cfg.txt`" is stored there.
4. Open this file with a simple text editor, such as Notepad or Vi. Do not use any programs that use 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

Beforeyou set a value, find out about its data type (see the " **Object directory description**" 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** $_h$: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 specified here is described in the " **Assignments**" section.

Example

The following code is set by the object **2057** $_h$:00 $_h$"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 NanoJ program

A NanoJ 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 " **Programming with NanoJ**" section.
2. Connect the voltage supply to connector X4 (see the " **Motor controller voltage supply (connector X4)**" 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 NanoJ 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 NanoJ program on the motor controller must have the file name " `vmmcode.usr`".
- If the NanoJ program was deleted, an empty file with name " `vmmcode.usr`" is created at the next startup.

Tip Deletion of the old NanoJ 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

⚠CAUTION

This product is causing high frequenty disturbances, arrangements for disturbance suppression may be necessary in a living environment .

⚠CAUTION

Alternating electromagnetic fields!

Alternating electromagnetic fields around current-carrying cables, especially around the supply and motor cables, can cause interference in the motor and other devices.

- Shield the cables. Run the shield connection on one side or both sides to a short earth.
- Use twisted pair cables.
- Keep power supply and motor cables as short as possible.
- Ground the motor housing large-area to a short earth.
- Run supply, motor and control cables separately.

6.2 Preparation

The following components are required for setup and installation:

- C5 motor controller
- Voltage supply in accordance with the data sheet
- Additional voltage source unit for "enable" input

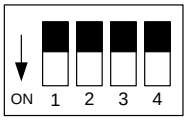
Corresponding to the mode to be used:

- 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 X4 of the motor controller (see "**Motor controller voltage supply (connector X4)**").
4. Connect the clock generator to "clock" pin (X2, pin 9, see "**Digital inputs (connector X2)**")
5. Connect the additional voltage sourcre to the "enable" pin (X2 pin 5, see "**Digital inputs (connector X2)**")
6. Switch on the voltage supply on connection X4.
7. Switch on the voltage supply on the "enable"-pin.

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

6.2.2 Analog mode

CAUTION

Make sure that the voltage at the analog input does not exceed the value of 10 V.

Read the "**Configuration**" section on the motor controller if you have not already done so.

1. Switch off any connected voltage supply.
2. All DIP switches must - except for switch 1 - be at the "Off" state:

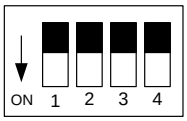
3. Connect the voltage supply to connection X4 of the motor controller (see "**Motor controller voltage supply (connector X4)**").
4. Connect the adjustable voltage source supply to connector X1 pin 2 (see "**Analog input and digital outputs (connector X1)**")

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

6.2.3 Clock Direction mode

Read the "**Configuration**" section on the motor controller if you have not already 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 connector X4 of the motor controller (see "**Motor controller voltage supply (connector X4)**").
4. Connect the clock generator to connector X2 (see "**Digital inputs (connector X2)**")
5. Switch on the voltage supply on connection X4.
6. Switch on the voltage supply on the "enable"-pin.

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

7 Operating modes

7.1 Profile Position

7.1.1 Special feature C5 USB

Note
Because this motor controller does not contain a field bus, the following operating mode is only used with the NanoJ program. Further information on programming and use of a NanoJ program can be found in the "**Programming with NanoJ**" section.

7.1.2 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 object **6060**$_h$ (Modes Of Operation) (see "**DS402 Power State machine**").

Control word

The following bits in object **6040**$_h$ (control word) 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**$_h$) 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$_h$		
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 the figure in "**Setting move commands**".

Status word

The following bits in object **6041**$_h$ (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**$_h$) within a tolerance window (**6067**$_h$).
- 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**$_h$ (Software Position Limit)).

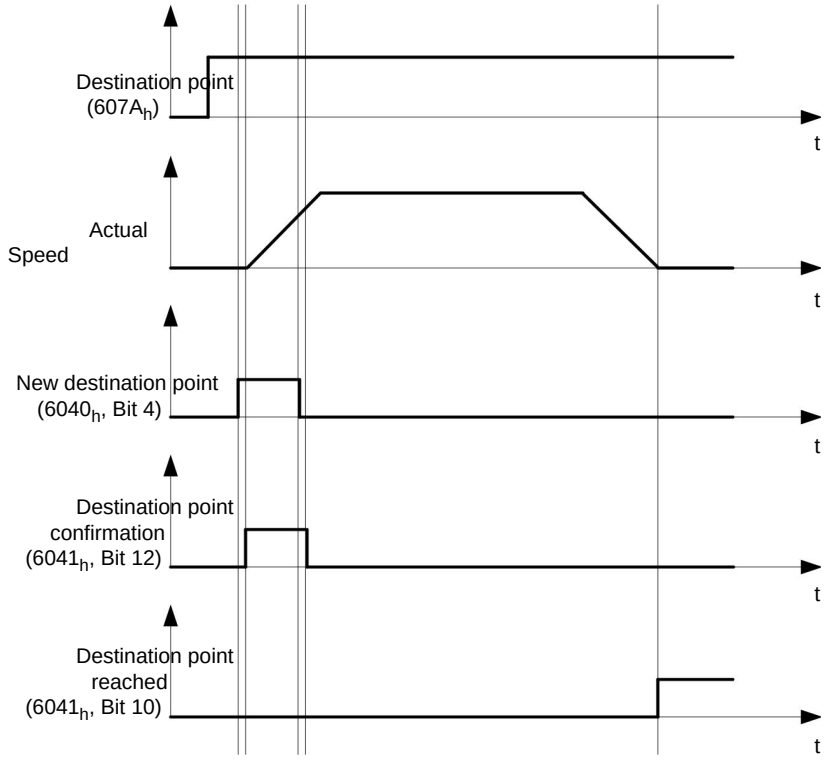
- Bit 13 (Following Error): This bit is set in closed loop mode if the following error is greater than the set limits is (**6065**$_h$ (Following Error Window) and **6066**$_h$ (Following Error Time Out)).

7.1.3 Setting move commands

Move command

In object **607A**$_h$ (Target Position), the new target position is specified in user units (see "**User-defined units**"). Afterwards, the move command is triggered when bit 4 is set in object **6040**$_h$ (control word). If the target position is valid, the motor controller responds with bit 12 in object **6041**$_h$ (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

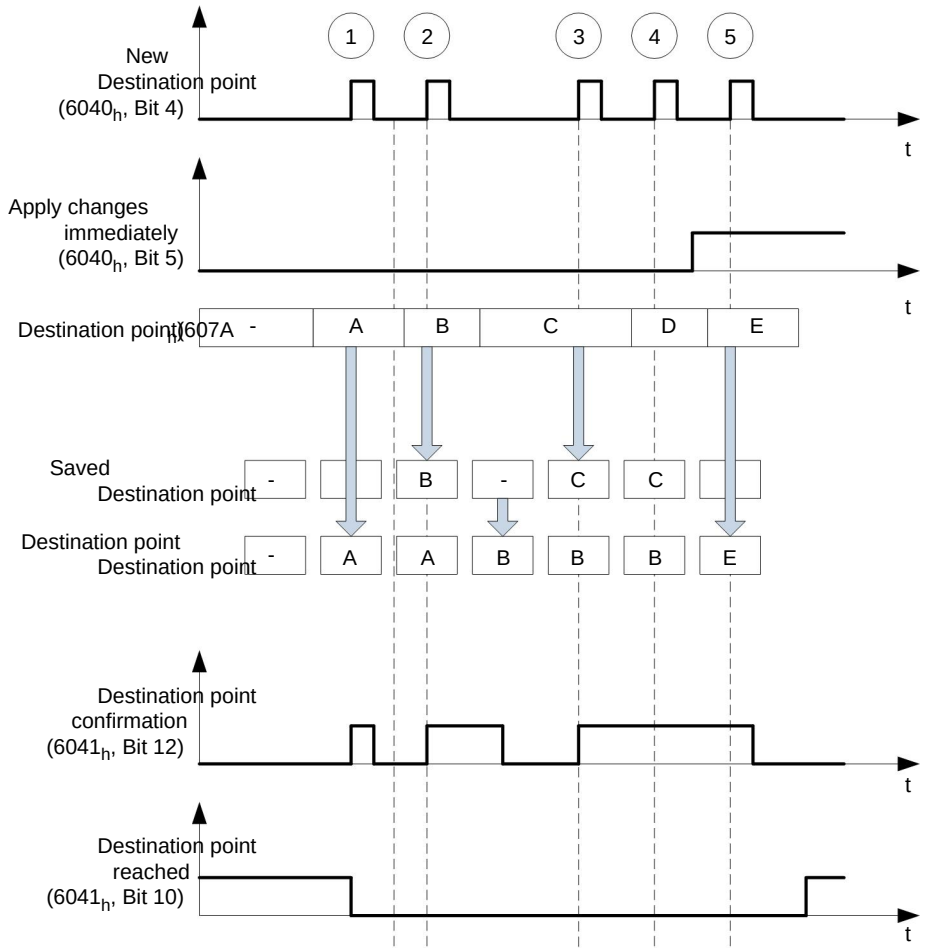


Further move commands

Bit 12 in object **6041**$_h$ (status word, set-point acknowledge) changes to "0" if 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).

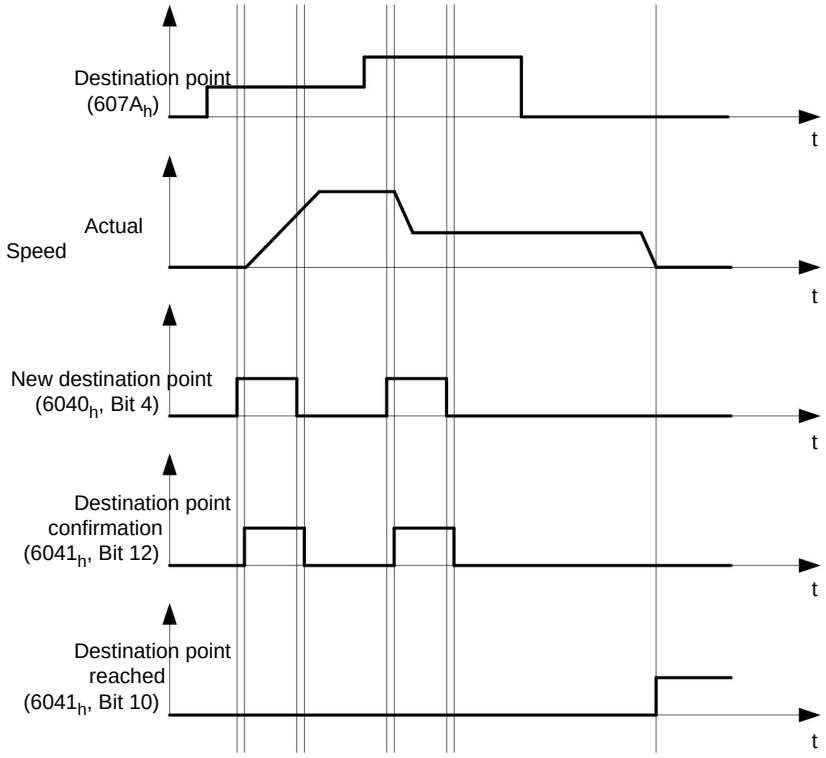
If the buffer is already full, a new time is ignored (time 4). If bit 5 in object **6040**$_h$ (control word, bit: "Change Set-Point Immediately") is set, the motor controller operates without the buffer and new move commands are implemented directly (time 5).

Times



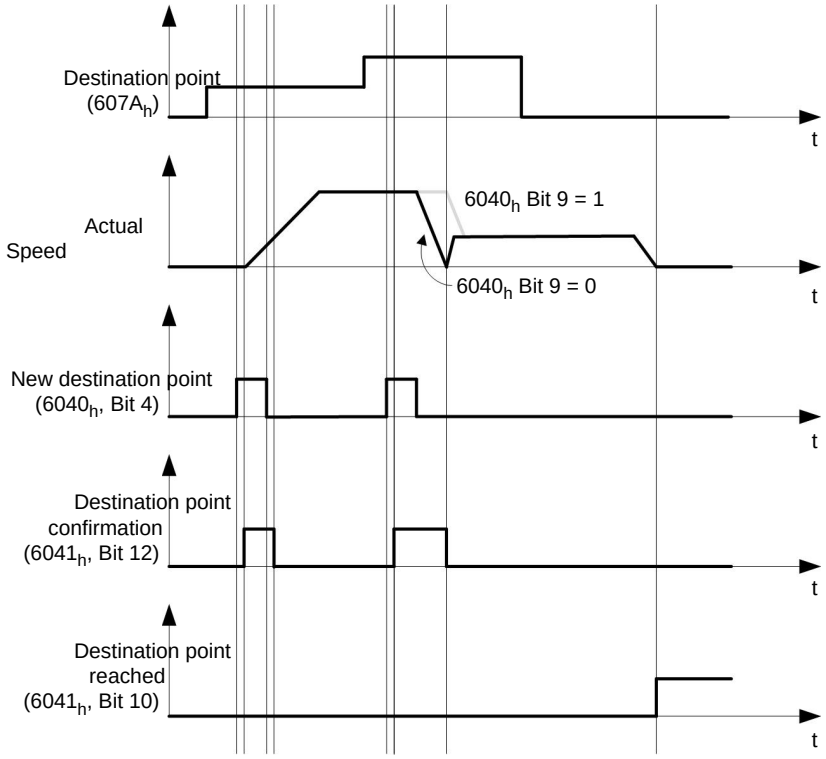
Transition procedure for second target position

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**$_h$ (control word) is set to "1" and the new target value is adopted immediately.



Options for moving to a target position

If bit 9 in object **6040**$_h$ (control word) is "0", the actual target position is first moved to completely. In this example, the end speed (**6082**$_h$) 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.



7.1.4 Marginal conditions for a positioning run

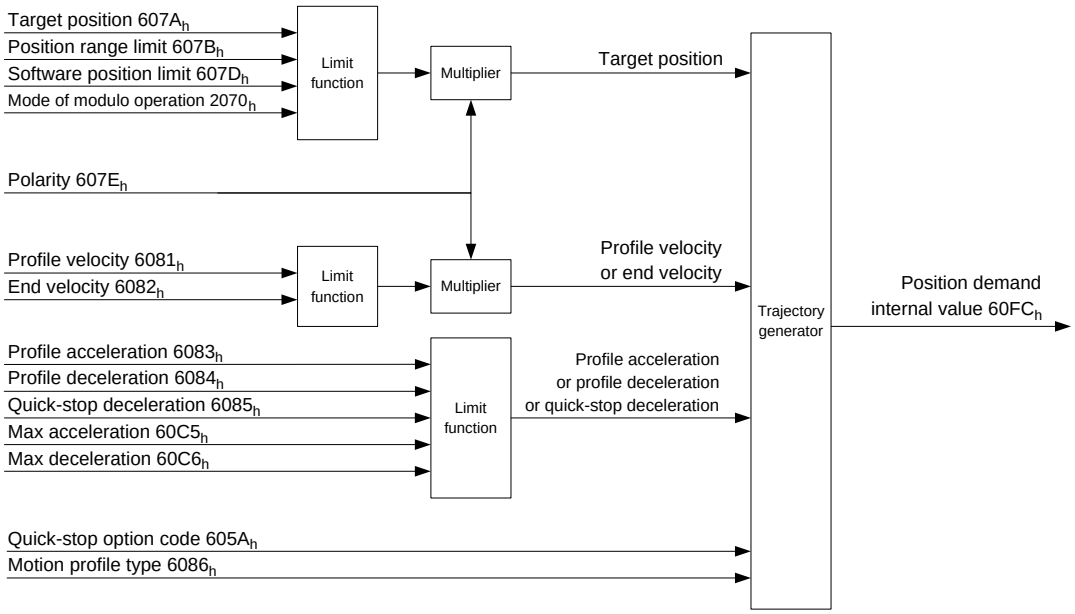
Object entries

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

- **6064**$_h$ (Position Actual Value): Actual position of the motor
- **607A**$_h$ (Target Position): Planned target position
- **607B**$_h$ (Position Range Limit): Definition of the limit stops (see the section below)
- **607C**$_h$ (Home Offset): Shifting of the machine zero point (see "**Homing**")
- **607D**$_h$ (Software Position Limit): Limits of a modulo operation for emulating an endless rotational axis (see "**#id127GE0Z0JNW**")
- **607E**$_h$ (Polarity): Direction of rotation
- **6081**$_h$ (Profile Velocity): Maximum speed with which the position should be moved to
- **6082**$_h$ (End Velocity): Speed when reaching the target position
- **6083**$_h$ (Profile Acceleration): Required acceleration
- **6084**$_h$ (Profile deceleration): Required deceleration
- **6085**$_h$ (Quick Stop Deceleration): Emergency stop deceleration in case of the "Quick stop active" state of the "DS402 Power State machine"
- **6086**$_h$ (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$_h$:1$_h$ - 4$_h$ are set as jerk limitations.
- **60C5**$_h$ (Max Acceleration): The maximum acceleration that may not be exceeded when moving to the end position.
- **60C6**$_h$ (Max Deceleration): The maximum deceleration that may not be exceeded when moving to the end position
- **60A4**$_h$ (Profile Jerk), subindex 01$_h$ to 04$_h$: Objects for describing the limit values for the jerk

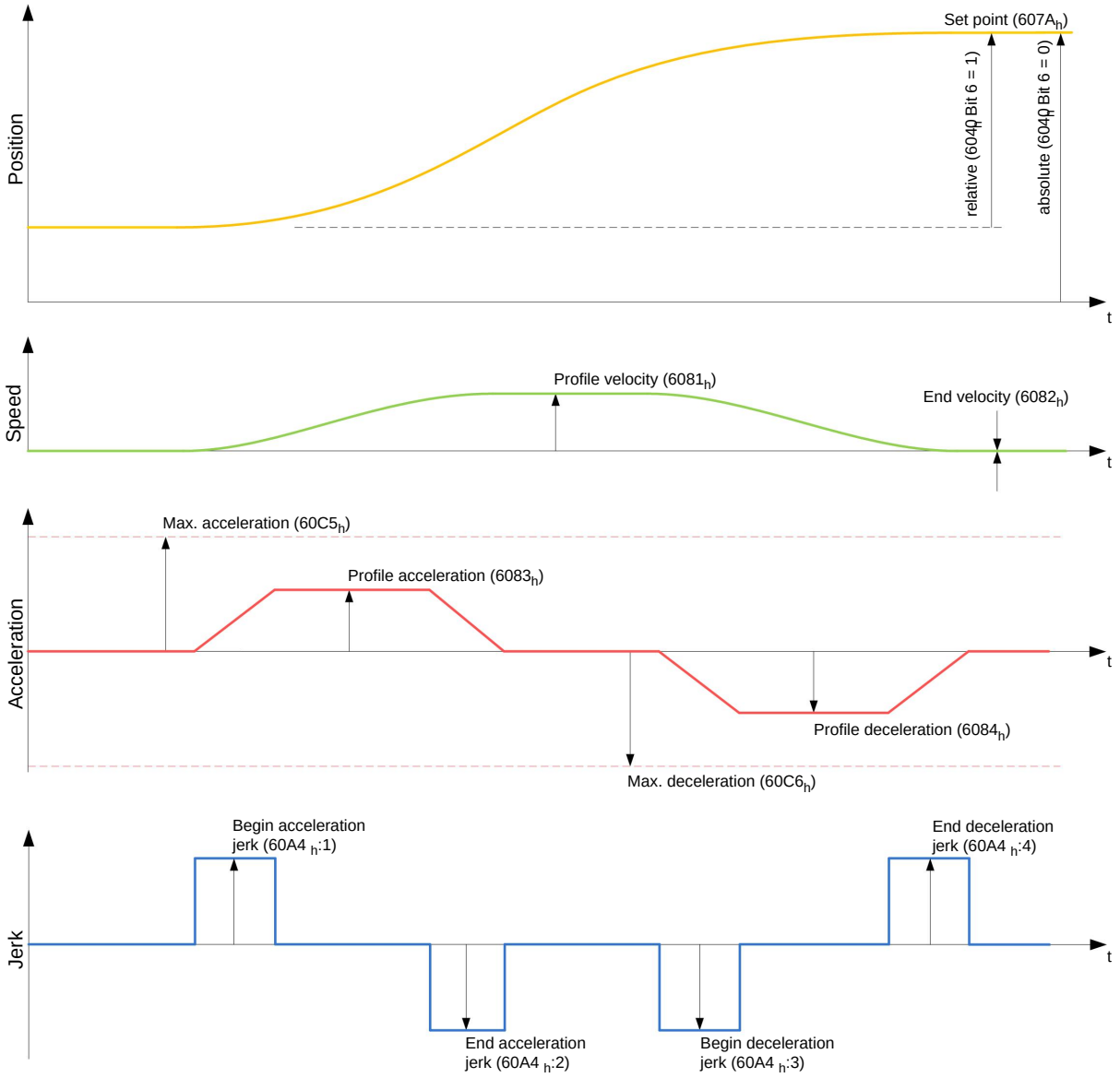
Objects for the positioning run

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).



7.1.5 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**$_h$ to "3". This causes the entries for the jerks in object **60A4**$_h$:01$_h$- 04$_h$ 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**$_h$ 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 the entries **60A4**$_h$:01$_h$ to **60A4**$_h$:04$_h$ are set to "0" and the object **6086**$_h$ is set to "3", or the entry in the object **6086**$_h$ is set to "0".

7.2 Velocity

7.2.1 Special feature C5 USB

Note

Because this motor controller does not contain a field bus, the following operating mode is only used with the NanoJ program.

Further information on programming and use of a NanoJ program can be found in the "**Programming with NanoJ**" section.

7.2.2 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.

7.2.3 Activation

To activate the mode, the value "2" must be set in object **6060**$_h$ (Modes Of Operation) (see "**DS402 Power State machine**").

7.2.4 Control word

The following bits in object **6040**$_h$ (control word) 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 **604A**$_h$. Then the motor controller changes to the "Switch on disabled" state (see **6040**$_h$).
- 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.

7.2.5 Status word

The following bits in object **6041**$_h$ (status word) have a special function:

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

7.2.6 Object entries

The following objects are required to control this mode:

- **604C**$_h$(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**$_h$, the selection is made of whether these are electrical (**2060**$_h$= 0) or mechanical (**2060**$_h$= 1) revolutions per second.

  The target speed is set in user units here.
- **6042**$_h$: Target Velocity
- **6048**$_h$: 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**$_h$ (Velocity Deceleration):

  This object defines the deceleration. The subindices are structured as described in object **6048**$_h$, and the speed difference must be indicated by a positive sign.
- **6085**$_h$ (Quick Stop Deceleration):

  This object defines the quick stop deceleration. The subindices are structured as described in object **6048**$_h$, and the speed difference must be indicated by a positive sign.
- **6046**$_h$ (Velocity Min Max Amount):

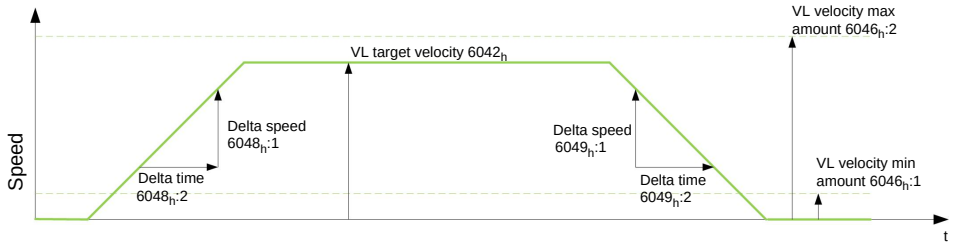
  In this object the limitations to target speeds are specified.

  The minimum speed is set in **6046**$_h$:01$_h$. If the target speed (**6042**$_h$) drops below the minimum speed, the value is limited to the minimum speed **6046**$_h$:01$_h$.

  The maximum speed is set in **6046**$_h$:02$_h$. If the target speed (**6042**$_h$) exceeds the maximum speed, the value is limited to the maximum speed **6046**$_h$:02$_h$.
- **604A**$_h$ (Velocity Quick Stop):

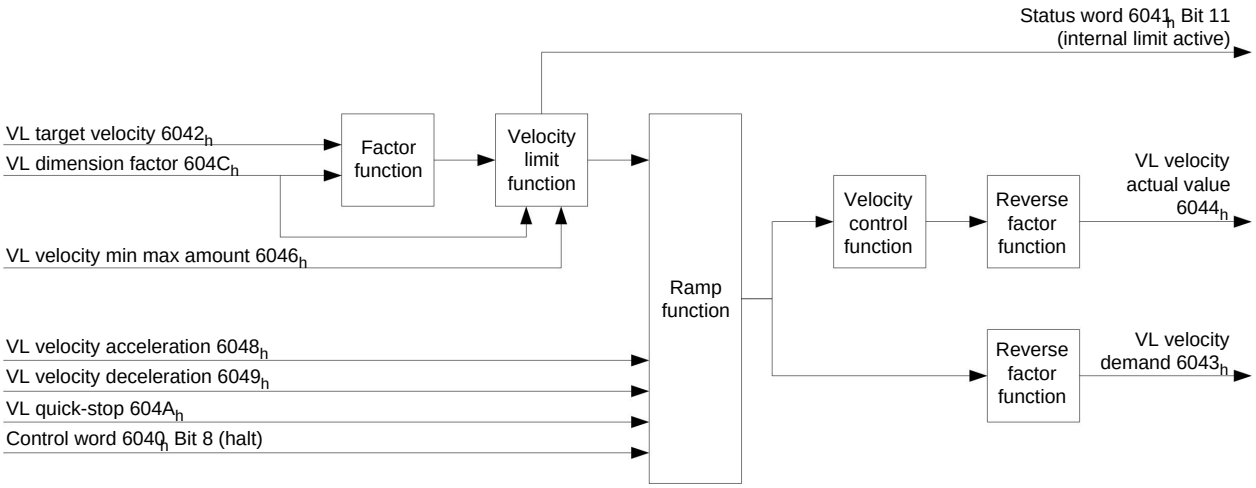
  The quick stop ramp can be set with this object. Subindices 1 and 2 are the same as specified for object **6048**$_h$.

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**$_h$ (internal limit active) when a limitation is active.



7.3 Profile Velocity

7.3.1 Special feature C5 USB

Note

Because this motor controller does not contain a field bus, the following operating mode is only used with the NanoJ program.

Note
Further information on programming and use of a NanoJ program can be found in the "**Programming with NanoJ**" section.

7.3.2 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.

7.3.3 Activation

To activate the mode, the value "3" must be set in object **6060**$_h$ (Modes Of Operation) (see "**DS402 Power State machine**").

7.3.4 Control word

The following bits in object **6040**$_h$ (control word) 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**$_h$. Then the motor controller changes to the "Switch on disabled" state (**6040**$_h$).
- 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.

7.3.5 Status word

The following bits in object **6041**$_h$(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).

6041$_h$ Bit 10	**6040$_h$ Bit 8**	**Description**
0	0	Target speed attained
0	1	Axis is braking
1	0	The target speed within the target window (defined in **606D**$_h$ and **606E**$_h$)
1	1	Speed of axis is 0

7.3.6 Object entries

The following objects are required to control this mode:

- **606B**$_h$(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**$_h$(Velocity Actual Value):

  Specifies the current actual speed.
- **606D**$_h$(Velocity Window):

  This value specifies by how much the actual speed may vary from the set speed for bit 10 (target speed reached; Target Reached) in object **6041**$_h$(status word) is to be set to "1".
- **606E**$_h$(Velocity Window Time):

  This object indicates how long the actual speed and the set speed must be near each other in magnitude (see **606D**$_h$"Velocity Window") for bit 10 "Target Reached" in object **6041**$_h$(status word) to be set to "1".
- **607E**$_h$(Polarity):

If bit 6 is set to "1", the sign (plus/minus) of the target speed is reversed.

- **6083**$_h$(Profile acceleration):

  Sets the value for the acceleration ramp in velocity mode.
- **6084**$_h$(Profile Deceleration):

  Sets the value for the braking ramp in velocity mode.
- **6085**$_h$(Quick Stop Deceleration):

  Sets the value for the braking ramp for the quick stop in velocity mode.
- **6086**$_h$(Motion Profile Type):

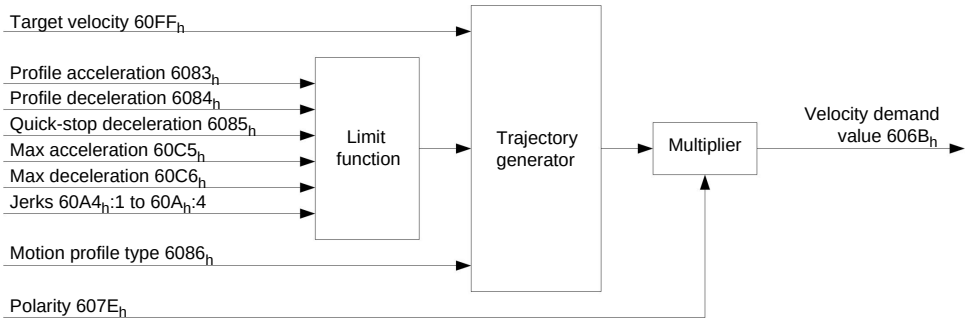
  Here the ramp type can be selected (0 = trapezoid ramp, 3 = jerk-limited ramp).
- **604A**$_h$(Velocity Quick Stop), subindex 01$_h$ to 04$_h$:

  The four jerk values are specified here if a jerk-limited ramp is set.
- **60FF**$_h$(Target Velocity):

  Specifies the target speed to be attained.
- **2031**$_h$(Peak Current):

  Maximum current in mA

Objects in Profile Velocity Mode

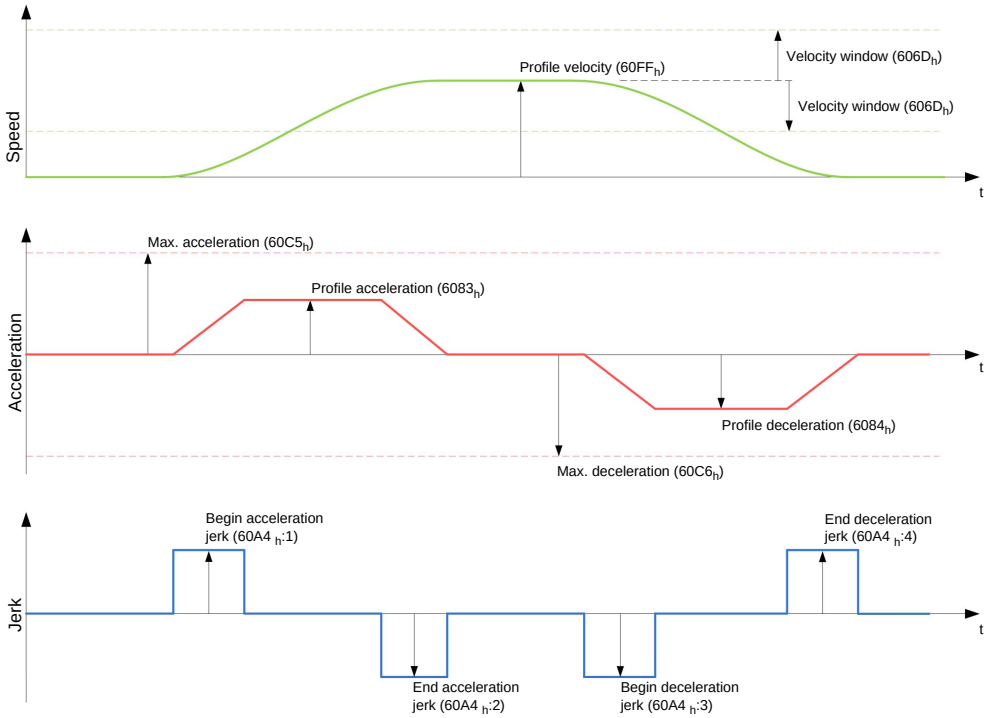


Mode activation

After the mode was selected in object **6060**$_h$(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 **60FF**$_h$(see the following diagrams). The speed, the acceleration and, in the case of jerk-limited ramps, the jerk limited values are taken into account.

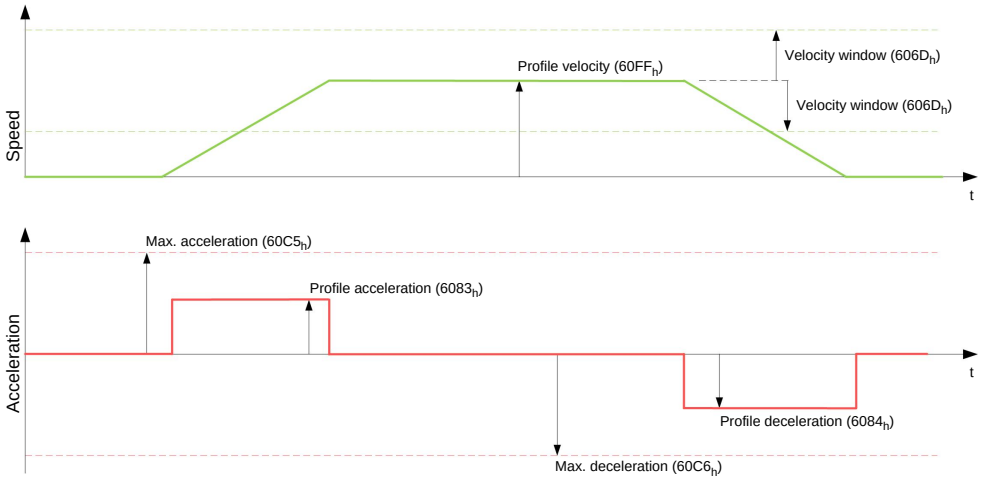
Limitations in the jerk-limited case

The following diagram shows the adjustable limitations in the jerk-limited case (**6086**$_h$=3).



Limitations in trapezoid case

This diagram shows the adjustable limitations for the trapezoid case (**6086**$_h$= 0).



7.4 Profile torque

7.4.1 Special feature C5 USB

Note

Because this motor controller does not contain a field bus, the following operating mode is only used with the NanoJ program.

Further information on programming and use of a NanoJ program can be found in the "**Programming with NanoJ**" section.

7.4.2 Description

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

7.4.3 Activation

To activate the mode, the value "4" must be set in object **6060**$_h$(Modes Of Operation) (see "**DS402 Power State machine**").

7.4.4 Control word

The following bits in object **6040**$_h$(control word) 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.

7.4.5 Status word

The following bits in object **6041**$_h$(status word) have a special function:

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

6040$_h$ Bit 8	**6041$_h$ Bit 10**	**Description**
0	0	Specified torque not attained
0	1	Specified torque attained
1	0	Axis accelerated
1	1	Speed of axis is 0

7.4.6 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**$_h$). This includes the following objects:

- **6071**$_h$(Target Torque):

  Target value of the torque
- **6072**$_h$(Max Torque):

  Maximum torque during the entire ramp (acceleration, hold torque, brake)
- **6074**$_h$(Torque Demand):

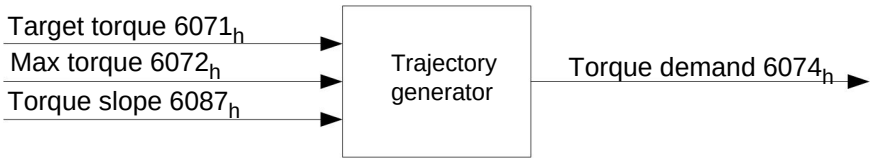
  Current output value of the ramp generator (torque) for the control
- **6087**$_h$(Torque Slope):

  Maximum change of the torque per second
- **3202**$_h$ Bit 5 (Motor Drive Submode Select):

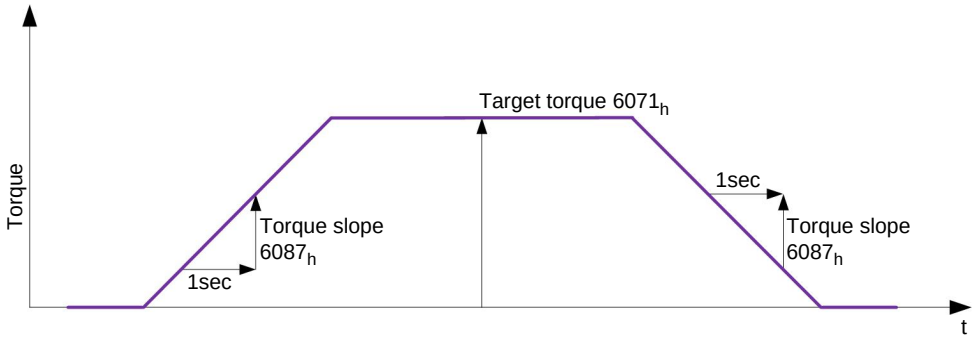
  If this bit is set to "0", the drive control is operated in torque-limited velocity mode, i.e. the maximum speed can be limited in object **2032**$_h$ and the control can work in field weakening mode.

  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 Special feature C5USB

Note
Because this motor controller does not contain a field bus, the following operating mode is only used with the NanoJ program. Further information on programming and use of a NanoJ program can be found in the "**Programming with NanoJ**" section.

7.5.2 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 object **6060**$_h$(Modes Of Operation) (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**$_h$(control word) 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**$_h$. The motor then goes into "Switch on disabled" mode (see the "**DS402 Power State machine**" section).
- 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**$_h$(status word) have a special function:

Bit 13	Bit 12	Bit 10	Description
0	0	0	Homing procedure is in progress
0	0	1	Homing procedure is interrupted or not started
0	1	0	Homing is attained, but target is not reached
0	1	1	Homing procedure is completed successfully

Bit 13	Bit 12	Bit 10	Description
1	0	0	Homing error occurred, velocity is not 0
1	0	1	Homing error occurred, velocity is 0

Object entries

The following objects are required to control this mode:

- **6098**$_h$(Homing Method):

  Method used for referencing (see "**Reference run method**")
- **6099**$_h$:01$_h$ (Speed During Search For Switch):

  The speed for the search for the switch
- **6099**$_h$:02$_h$ (Speed During Search For Zero):

  The speed for the search for the index
- **609A**$_h$(Homing Acceleration):

  Acceleration and deceleration for the reference run
- **2056**$_h$(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**$_h$:01$_h$ (Minimum Current For Block Detection):

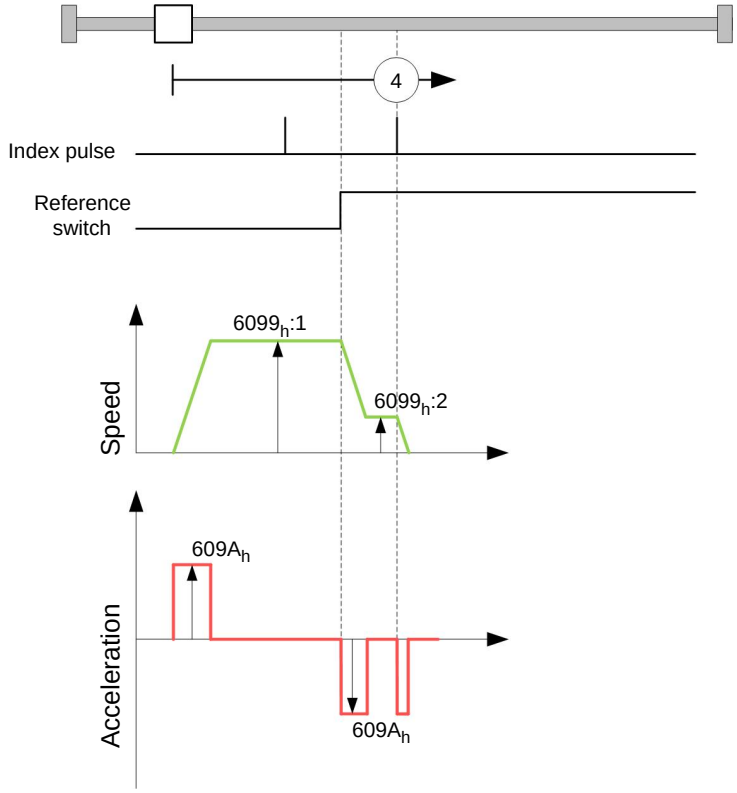
  Minimum current threshold that, when exceeded, detects blocking of the motor at a block.
- **203A**$_h$:02$_h$ (Period Of Blocking):

  Specifies the time in ms that the motor is nevertheless still to travel against the block after block detection.
- **203A**$_h$:03$_h$ (Block Detection Time)

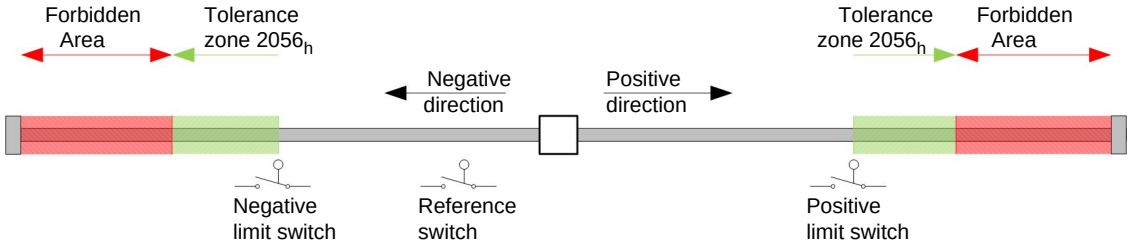
  Specifies the time in ms that the current has to be at least above the minimum current threshold in order to detect a block

Speeds of the reference run

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



Tolerance bands of the limit switches



7.5.3 Reference run method

Description

The reference run method is written into object **6098**$_h$ 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**$_h$:02$_h$ in ms.

To ensure very precise detection of the block, the block should be traveled against with a very low speed (**6099**$_h$:01$_h$), high current limit (**203A**$_h$:01$_h$), and high homing acceleration (**609A**$_h$). Additionally, detection can be refined by the block detection time (**203A**$_h$:03$_h$).

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

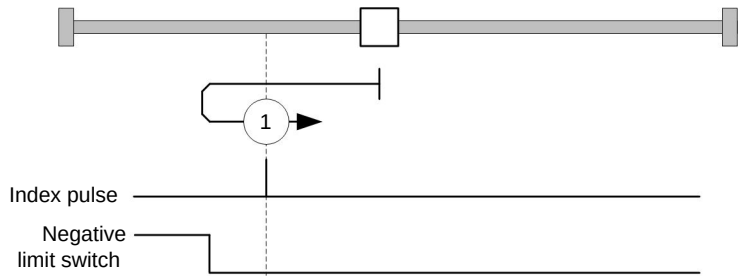
The following methods can be used for homing on block:

- 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

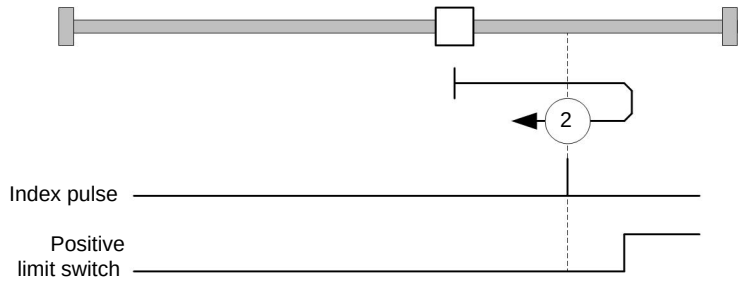
Methods 1 and 2

Reference the limit switch and index pulse.

Method 1 references a negative limit switch and index pulse:



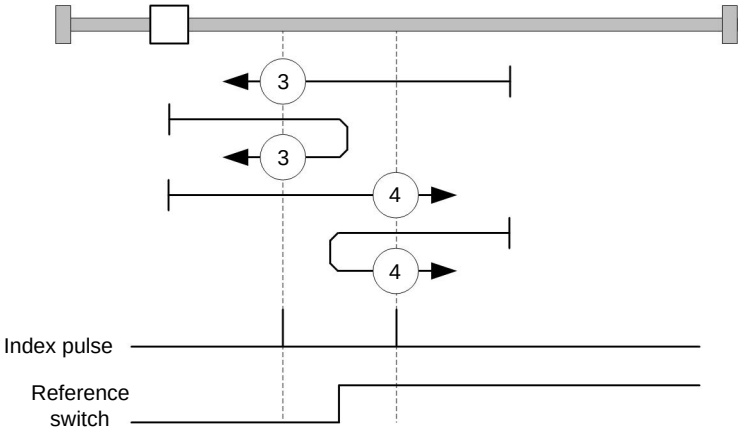
Method 2 references a positive limit switch and index pulse:



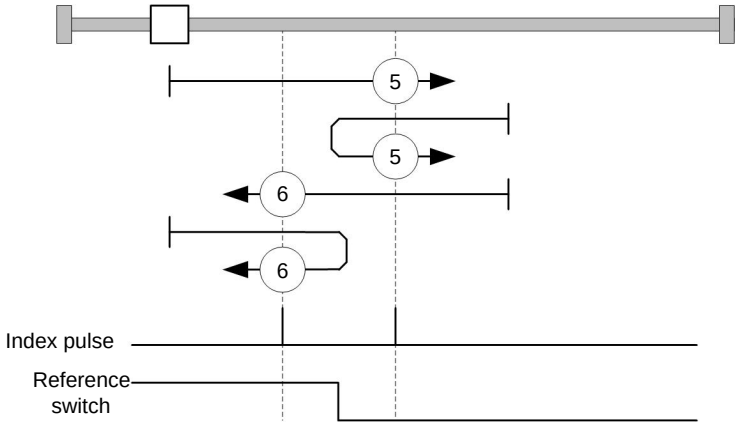
Methods 3 to 6

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:

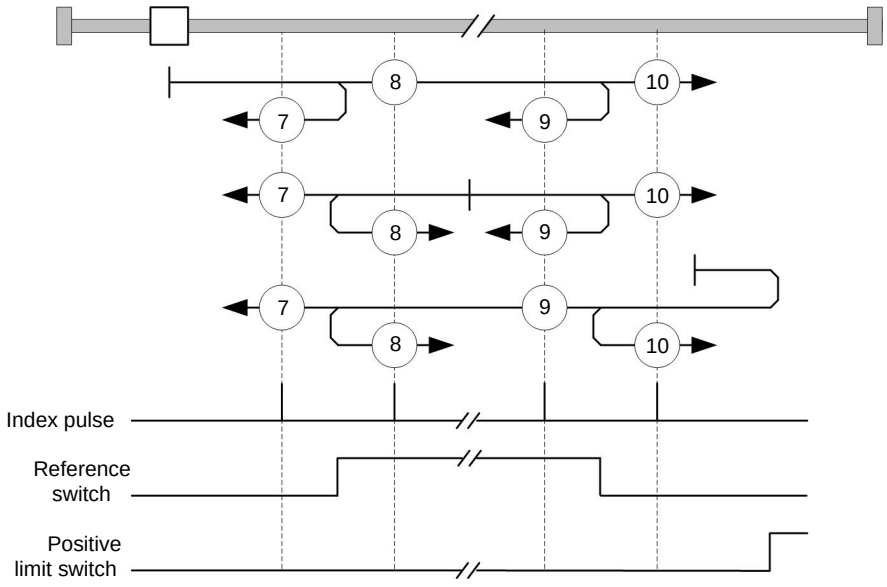


Methods 7 to 14

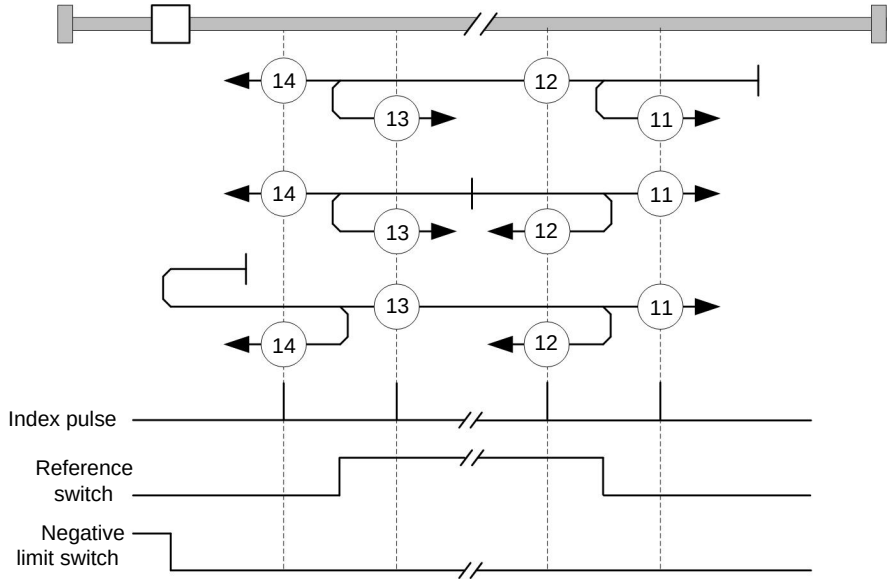
These methods reference the home switch and index pulse (with limit switches).

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:



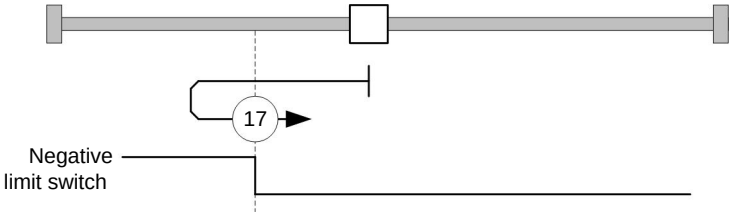
The methods 11 to 14 take the negative limit switch into account:



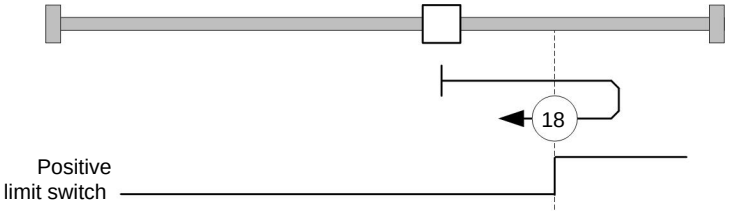
Methods 17 and 18

These methods reference the limit switch without the index pulse.

Method 17 references the negative limit switch:



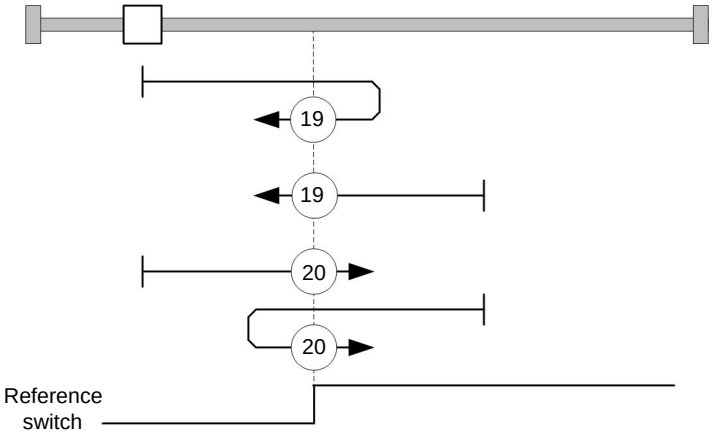
Method 18 references the positive limit switch:



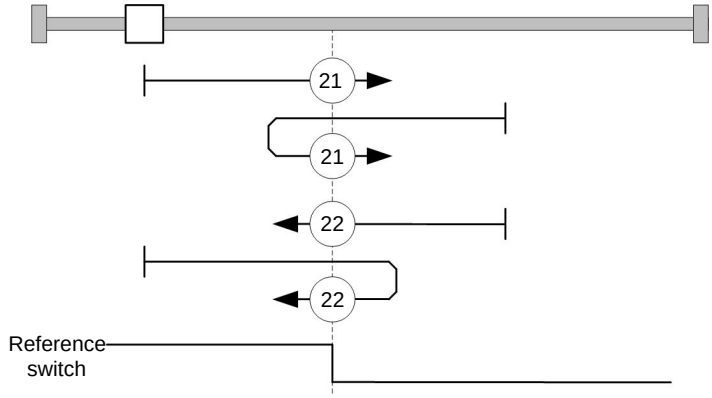
Methods 19 to 22

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:

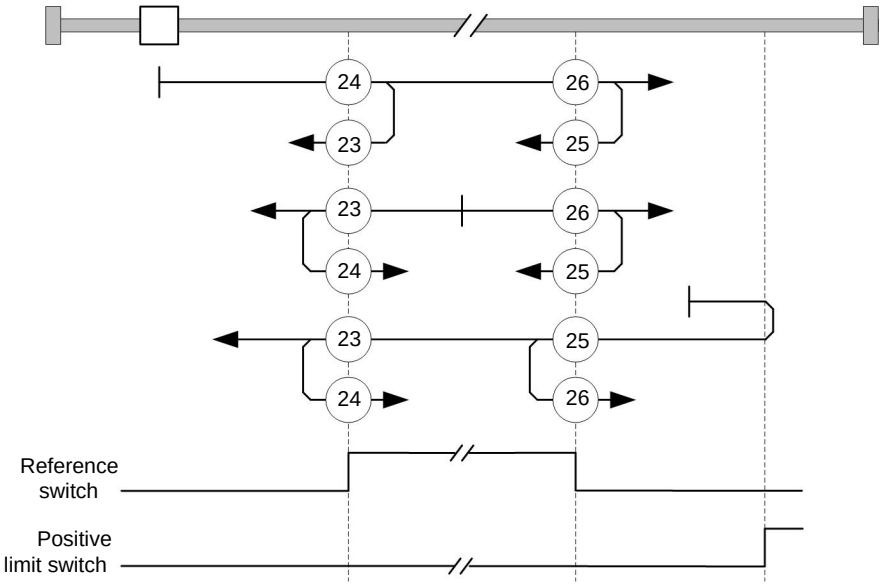


Methods 23 to 30

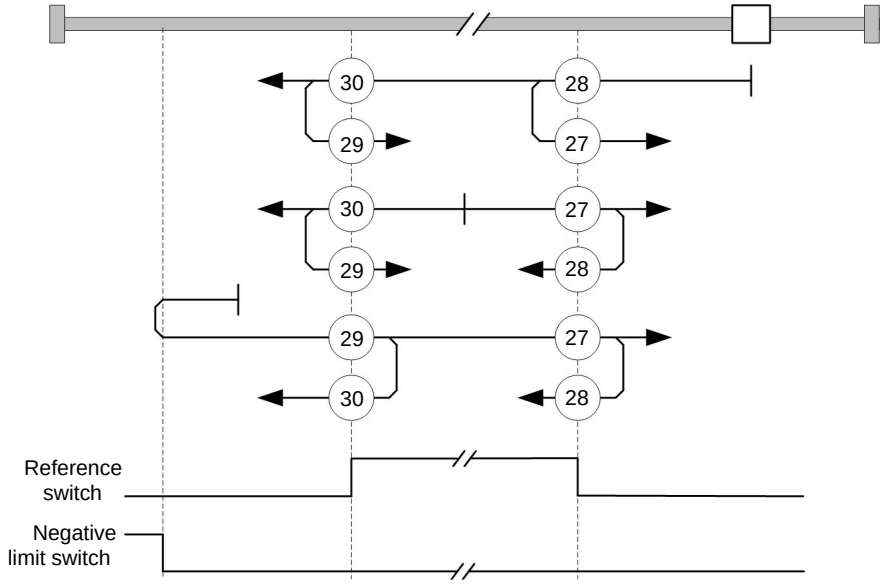
These methods reference the home switch without the index pulse (with limit switches).

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:



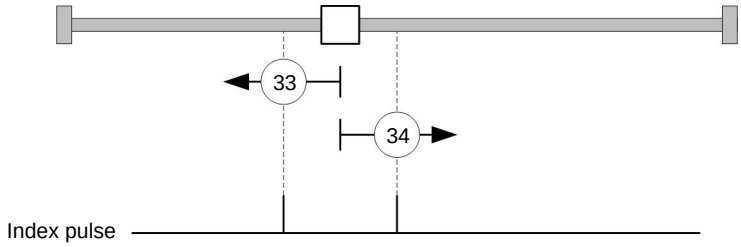
The methods 27 to 30 take the negative limit switch into account:



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

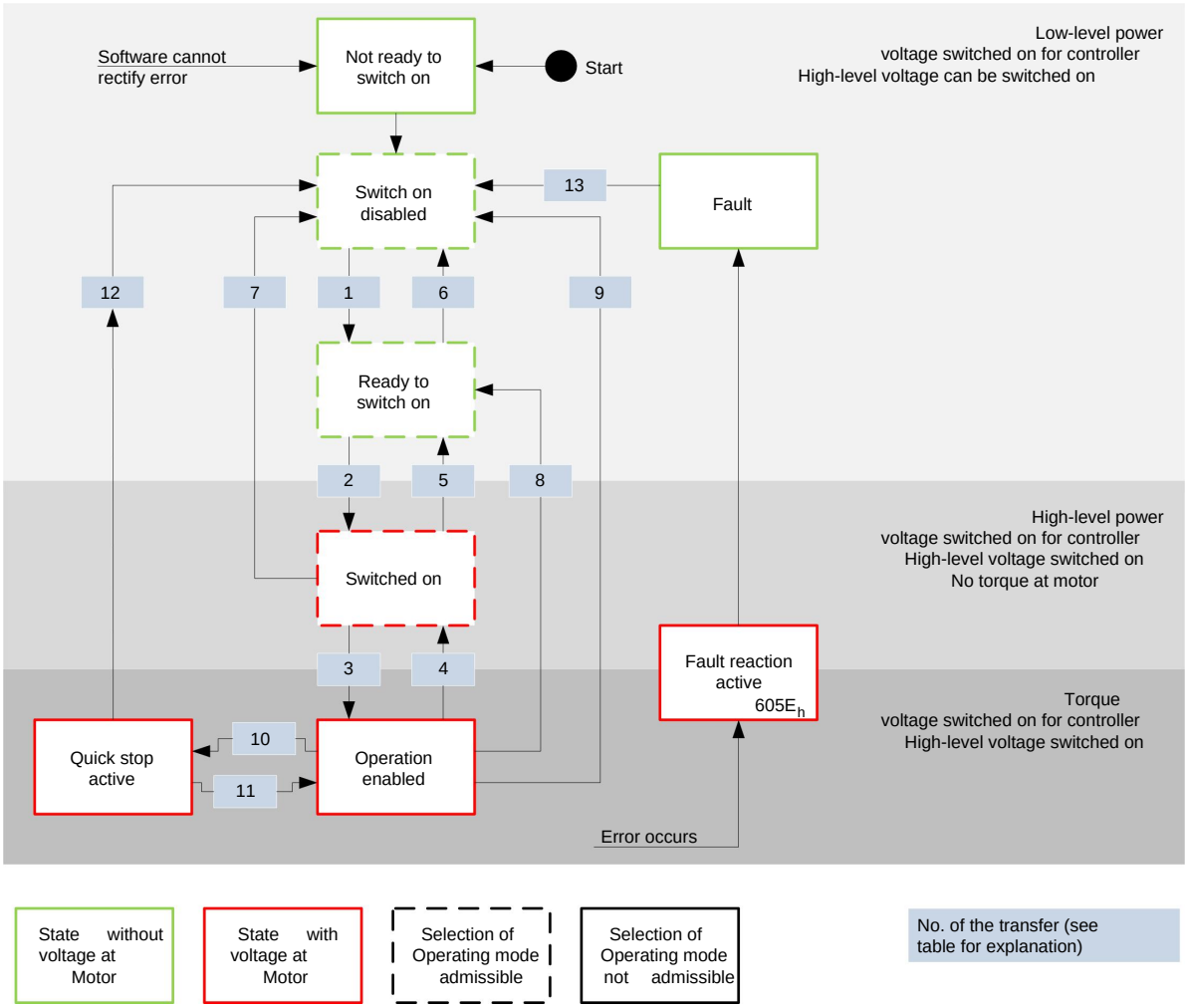
To switch the motor controller to an operational state, a state machine must be run through. This is defined in CANopen standard DS402. State changes are requested in object **6040**$_h$ (control word). The actual state of the state machine can be read out from object **6041**$_h$ (status word).

Control word

State changes are requested via object **6040**$_h$ (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_h					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_h)	State
xxxx xxxx x0xx 0000	Not ready to switch on
xxxx xxxx x1xx 0000	Switch on disabled
xxxx xxxx x01x 0001	Ready to switch on
xxxx xxxx x01x 0011	Switched on
xxxx xxxx x01x 0111	Operation enabled
xxxx xxxx x00x 0111	Quick stop active
xxxx xxxx x0xx 1111	Fault reaction active
xxxx xxxx x0xx 1000	Fault

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_h**) 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_h** (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.

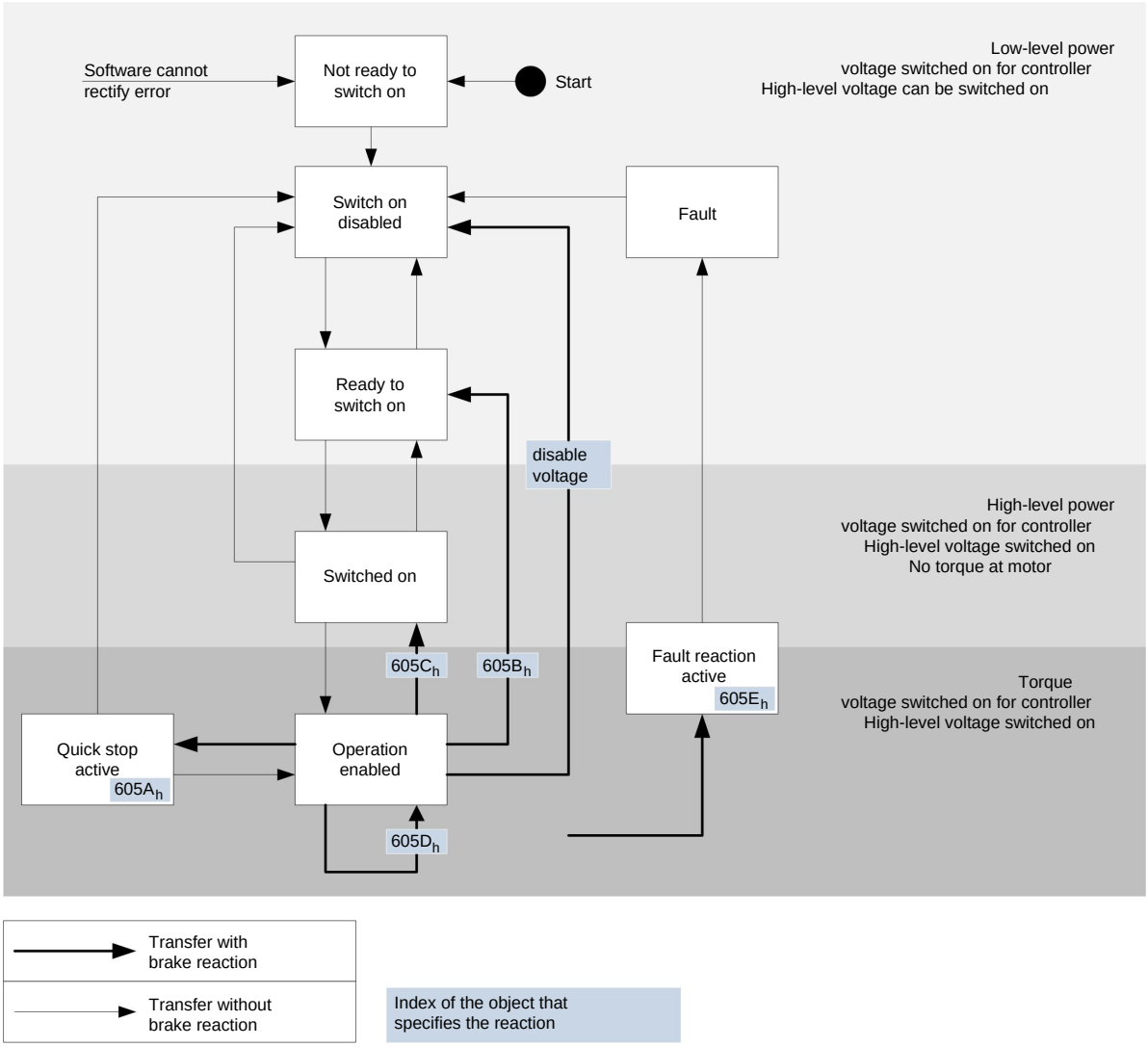
8.1.2 Behavior after the "Operation enabled" state is left

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**$_h$ is executed (see the following table).

Value in object 605A$_h$	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

Ready to switch on

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

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

Value in object $605B_h$	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 **$605C_h$** is executed (see the following table).

Value in object $605C_h$	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_h** (control word), the response stored in **$605D_h$** is executed in velocity mode and profile velocity mode (see the following table).

Value in object $605D_h$	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 **$605E_h$**.

Value in object $605E_h$	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**$_h$ 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**$_h$:1$_h$ (Motor Revolutions)) per shaft revolution (**6091**$_h$:2$_h$ (Shaft Revolutions)) as follows:

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

If object **6091**$_h$:1$_h$ or object **6091**$_h$:2$_h$ are set to "0", the firmware sets the value to "1".

Feed constant

The feed constant is calculated from the feed (**6092**$_h$:1$_h$ (Feed Constant) per revolution of the drive axis (**6092**$_h$:2$_h$ (Shaft Revolutions) as follows:

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

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

If object **6092**$_h$:1$_h$ or object **6092**$_h$:2$_h$ are set to "0", the firmware sets the value to "1".

Position

The current position in user units (**6064**$_h$) is calculated as follows:

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

Speed

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

Object	Mode	Meaning
606B$_h$	Profile Velocity Mode	Output value of ramp generator
60FF$_h$	Profile Velocity Mode	Speed specification
6099$_h$	Homing mode	Speed for searching the index/switch
6081$_h$	Profile Position Mode	Target speed
6082$_h$	Profile Position Mode	End speed

The internal speed in mechanical revolutions per second is multiplied by a factor for numerator (**2061**$_h$) and denominator (**2062**$_h$). The speed in user units is computed from

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

If object **2061**$_h$ or **2062**$_h$ 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
609A$_h$	Homing mode	Acceleration
6083$_h$	Profile Position Mode	Acceleration
6084$_h$	Profile Position Mode	Deceleration
60C5$_h$	Profile Velocity Mode	Acceleration
60C6$_h$	Profile Position Mode	Deceleration
6085$_h$	"Quick stop active" state (**DS402 Power State machine**)	Deceleration

The internal acceleration in mechanical revolutions per second squared is multiplied by a factor consisting of a numerator (**2063**$_h$) and denominator (**2064**$_h$).

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

If object **2063**$_h$ or **2064**$_h$ is to be set to "0", the firmware sets the value to "1".

Jerk

For the jerk, objects **604A**$_h$:1$_h$ to **604A**$_h$:4$_h$ can be specified in user units. These objects only affect the Profile Position Mode and Profile Velocity Mode.

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

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

If object **2065**$_h$ or **2066**$_h$ 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**$_h$:1$_h$ (Encoder Increments)) per motor revolutions (**608F**$_h$:2$_h$ (Motor Revolutions)) multiplied by the polarity of the axis in object **607E**$_h$ bit 0. If bit 0 in object **607E**$_h$ is set to the value "1", this corresponds to a polarity reversal, or the value "-1" in the formula:

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

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

- Encoder increments **608F**$_h$:1$_h$ = "2000"
- Motor revolutions **608F**$_h$:2$_h$ = "1"
- Polarity **607E**$_h$ bit 0 = "0" (does not correspond to a polarity reversal)

The resolution of the connected position encoder is set in object **2052**$_h$.

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:

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

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

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**$_h$:01$_h$

  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**$_h$:02$_h$

  This bit changes from closer logic (a logical high level at the input yields the value of "1" in object **60FD**$_h$) to opener logic (the logical high level at the input yields the value of "0"). This applies to the special functions (except the clock and directional inputs) and for the normal input. The input is set as closer logic if the corresponding bit is "0", it is set to opener logic with the value "1" respectively.
- **3240**$_h$:03$_h$

  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; the values set in object **3240**$_h$:04$_h$ for the respective input are used instead.
- **3240**$_h$:04$_h$

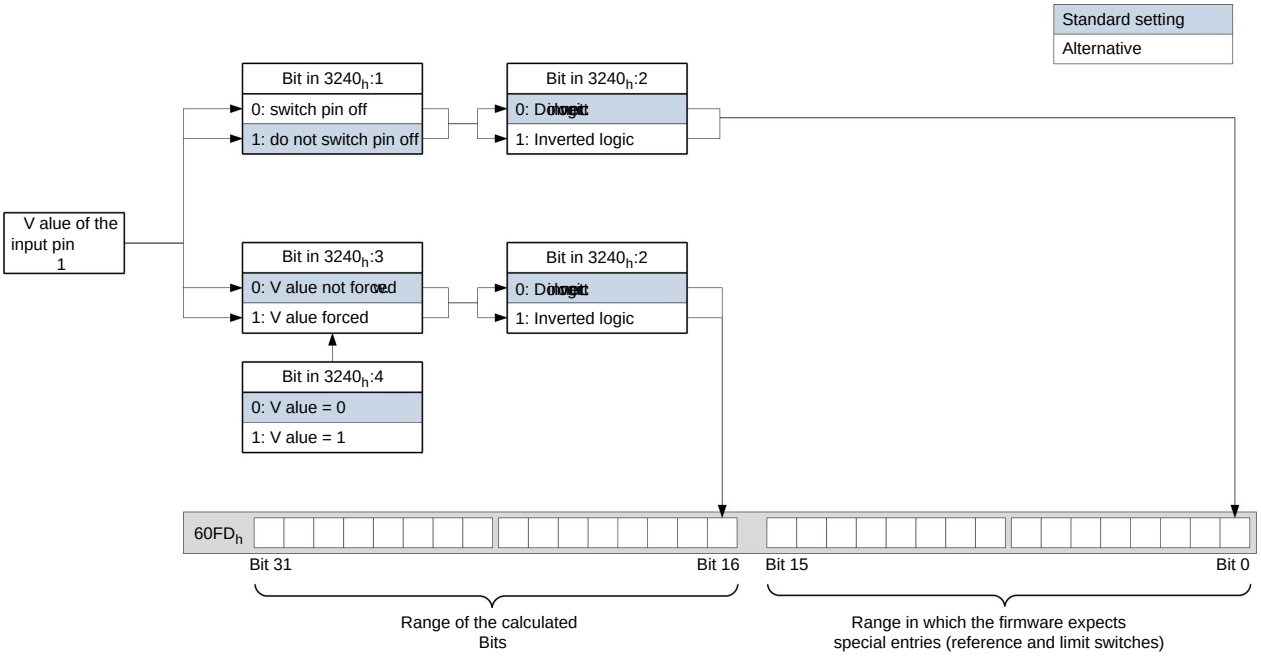
  This bit specifies the value to be read in as the input value if the same bit was set in object **3240**$_h$:03$_h$.
- **3240**$_h$:05$_h$

  This object contains the unmodified input value.

Computation of the inputs

Computation of the input using input 1 as an example:

The value of bit 0 of object **60FD**$_h$ 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 controlled using object **60FE**$_h$. Output 1 corresponds to bit 16 in object **60FE**$_h$, output 2 corresponds to bit 17, etc., as with the inputs. The outputs with special functions are again entered in the firmware in the lower bits 0 to 15. 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**$_h$:02$_h$

  This can be used to change the logic from "closer" to "opener". When configured as a "closer", the outputs a logical high level if the bit is "1". When configured as an "opener", the outputs a logical low level if there is a "1" in object **60FE**$_h$.
- **3250**$_h$:03$_h$

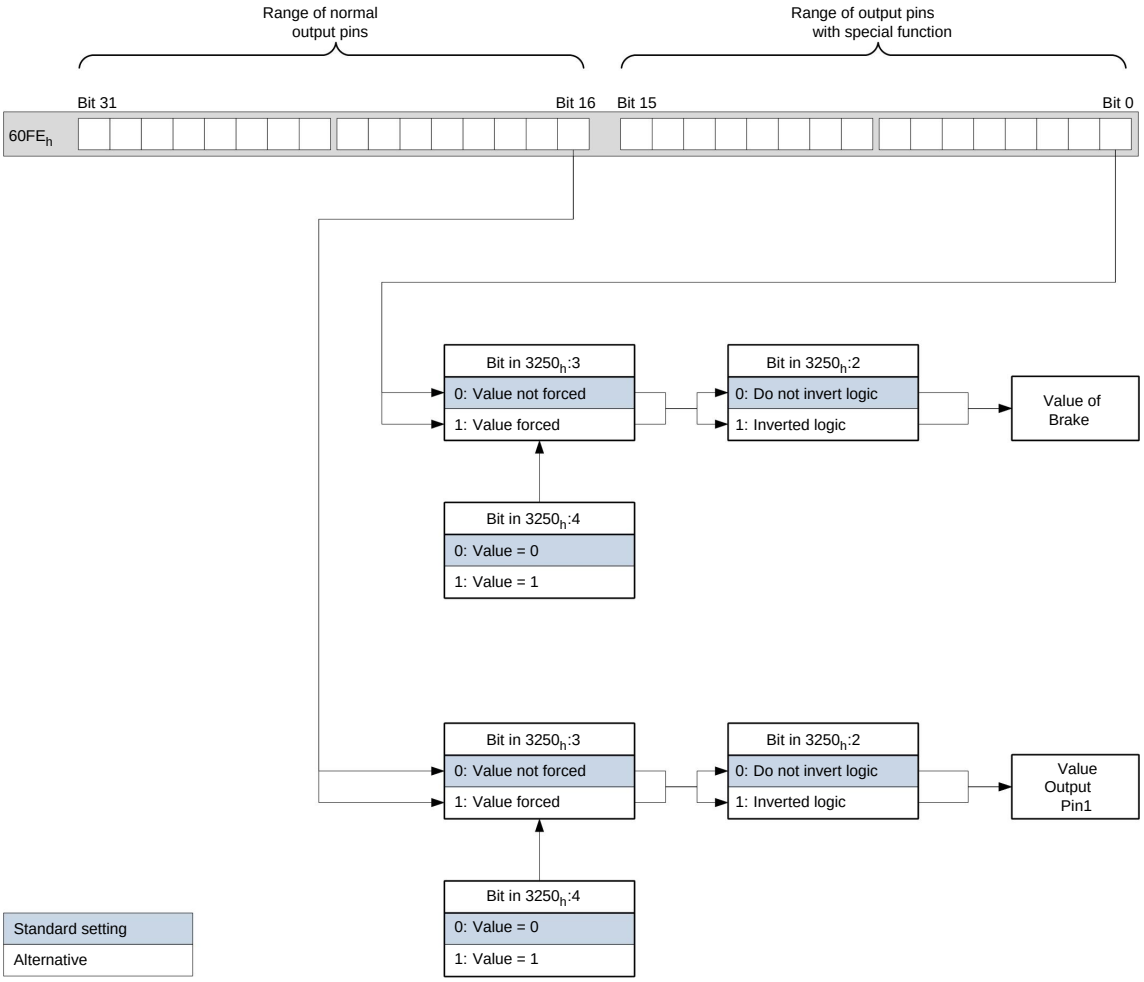
  If a bit is set in **3250**$_h$, the output is manually controlled. The value for the output is then contained in object **3250**$_h$:04$_h$, which is also possible for the brake output.
- **3250**$_h$:04$_h$

  The bits in this object specify the output value that is to be applied to the output when the manual control of the output is activated by object **3250**$_h$:03$_h$.
- **3250**$_h$:05$_h$

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

Bits of the outputs

Example of the computation of the bits for the outputs:



9.2 I²t motor overload protection

9.2.1 Description

I²t 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**$_h$ set to "1") and the motor is **not** in profile torque mode or cycle synchronous torque mode.

There is a single exception: If I²t 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 and into open loop mode with a very high short-time maximum current without damaging the motor.

9.2.2 Object entries

The following objects affect I²t motor overload protection:

- **2031**$_h$: Peak Current - specifies the maximum current in mA.
- **203B**$_h$:1$_h$ Nominal Current - specifies the nominal current in mA.
- **203B**$_h$:2$_h$ Maximum Duration Of Peak Current - specifies the maximum time period of the maximum current in ms.

The following objects indicate the actual state of I²t:

- **203B**$_h$:3$_h$ Threshold - specifies the limit in mA, from which is determined whether switching is to the maximum current or nominal current.

- **203B**$_h$:4$_h$ CalcValue - specifies the calculated value that is compared to the threshold in order to set the current.
- **203B**$_h$:5$_h$ LimitedCurrent - shows the actual current value that was set by I^2t.
- **203B**$_h$:6$_h$ Status:
  - Value = "0": I^2t deactivated
  - Value = "1": I^2t activated

9.2.3 Activation

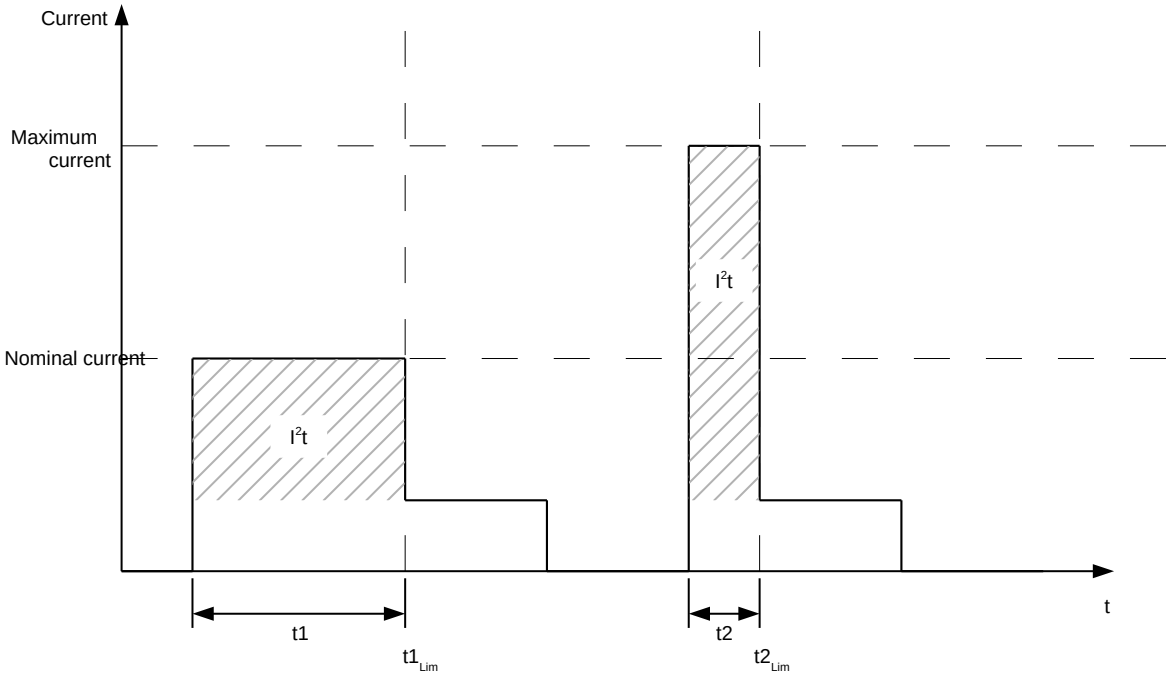
The three object entries above must have been appropriately specified to activate the mode. This means that the maximum current must be greater than the nominal current, and a time value must be entered for the maximum time of the maximum current. I^2t operability remains deactivated when these conditions 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 t1, the current value is higher than the nominal current. At time $t1_{Lim}$, I^2t_{Lim} is reached and the current is limited to the nominal current. During the following time period t2, a current comes that corresponds to the maximum current. Accordingly, the value for I^2t_{Lim} is reached faster than in time period t1.

9.3 Save Objects

⚠ CAUTION

The incorrect use of this feature may cause the controller not be able to start again. Therefore read the chapter completely before using this feature.

9.3.1 General information

A subset of objects can be stored and will get loaded automatically with the next start. Moreover the values will be kept during a firmware update.

Only whole sets of objects (called “categories”) can be stored, it is not possible to store single objects.

An object belongs to one of the following categories:

1. The object is not savable.
2. The object is related to the communication (e.g. fieldbus) so it belongs to the category “communication”.
3. The object holds general user information so it belongs to the category “user”

In chapter “ **object directory description**” - with the list of all objects - the ability to get saved is described for each object.

9.3.2 Category: not savable

The not savable objects are ignored in the process of saving. All status and control words and all objects depending on the status of the controller are ranked among this category.

9.3.3 Category: communication objects

All objects controlling the fieldbus are ranked among this category.

The following objects are considered to be a communication object:

- **2010**$_h$: IP-Configuration
- **2011**$_h$: Static-IP-Address
- **2012**$_h$: Static-IP-Subnet-Mask

9.3.4 Category: user object

The following objects are considered to be a user object:

- **2031**$_h$: Peak Current
- **2032**$_h$: Maximum Speed
- **2033**$_h$: Plunger Block
- **2034**$_h$: Upper Voltage Warning Level
- **2035**$_h$: Lower Voltage Warning Level
- **2036**$_h$: Open Loop Current Reduction Idle Time
- **2037**$_h$: Open Loop Current Reduction Value/factor
- **2038**$_h$: Brake Controller Timing
- **2056**$_h$: Limit Switch Tolerance Band
- **2057**$_h$: Clock Direction Multiplier
- **2058**$_h$: Clock Direction Divider
- **2059**$_h$: Encoder Configuration
- **2060**$_h$: Compensate Polepair Count
- **2061**$_h$: Velocity Numerator
- **2062**$_h$: Velocity Denominator
- **2063**$_h$: Acceleration Numerator
- **2064**$_h$: Acceleration Denominator
- **2065**$_h$: Jerk Numerator
- **2066**$_h$: Jerk Denominator
- **2084**$_h$: Bootup Delay
- **2300**$_h$: VMM Control
- **2303**$_h$: Number Of Active User Program
- **2304**$_h$: Table Of Available User Programs
- **2310**$_h$: VMM Input Data Selection
- **2320**$_h$: VMM Output Data Selection

- **2330**$_h$: VMM In/output Data Selection
- **3202**$_h$: Motor Drive Submode Select
- **320A**$_h$: Motor Drive Sensor Display Open Loop
- **320B**$_h$: Motor Drive Sensor Display Closed Loop
- **3210**$_h$: Motor Drive Parameter Set
- **3221**$_h$: Analogue Inputs Control
- **3240**$_h$: Digital Inputs Control
- **3250**$_h$: Digital Outputs Control
- **3321**$_h$: Analogue Input Offset
- **3322**$_h$: Analogue Input Pre-scaling
- **3700**$_h$: Following Error Option Code
- **6046**$_h$: Vl Velocity Min Max Amount
- **6048**$_h$: Vl Velocity Acceleration
- **6049**$_h$: Vl Velocity Deceleration
- **604A**$_h$: Vl Velocity Quick Stop
- **604C**$_h$: Vl Dimension Factor
- **605A**$_h$: Quick Stop Option Code
- **605B**$_h$: Shutdown Option Code
- **605C**$_h$: Disable Option Code
- **605D**$_h$: Halt Option Code
- **605E**$_h$: Fault Option Code
- **6072**$_h$: Max Torque
- **607B**$_h$: Position Range Limit
- **607C**$_h$: Home Offset
- **607D**$_h$: Software Position Limit
- **607E**$_h$: Polarity
- **6081**$_h$: Profile Velocity
- **6082**$_h$: End Velocity
- **6083**$_h$: Profile Acceleration
- **6084**$_h$: Profile Deceleration
- **6085**$_h$: Quick Stop Deceleration
- **6086**$_h$: Motion Profile Type
- **6087**$_h$: Torque Slope
- **608F**$_h$: Position Encoder Resolution
- **6091**$_h$: Gear Ratio
- **6092**$_h$: Feed Constant
- **6098**$_h$: Homing Method
- **6099**$_h$: Homing Speed
- **609A**$_h$: Homing Acceleration
- **60A4**$_h$: Profile Jerk
- **60C2**$_h$: Interpolation Time Period
- **60C5**$_h$: Max Acceleration
- **60C6**$_h$: Max Deceleration

9.3.5 Starting save process

⚠ CAUTION

- The motor has to stand still during the process of saving and is not allowed to get approached while saving.
- While saving the function of the fieldbus may be affected.
- The process of saving may need – depending on the type of controller – up to ten seconds. Never disconnect the power supply during this period.

⚠ CAUTION

The nonobservance may led to an broken file system, as a result the controller gets unusable.

- Therefore always wait for the controller to signal the successful process of saving with the value “1” in the correspondent subindex in object **1010**$_h$.

For each category there is a subindex in object **1010**$_h$. The only thing to do to save all objects of that category is to write the value `65766173`$_h$ [1] to the subindex. The end of the save process will be signalled by the controller by writing the value “1” to the subindex.

The following table lists, which subentry of the object **1010**$_h$ is related to which category.

Subindex	Category
01$_h$	all categroies
02$_h$	communication

9.3.6 Drop saved values

⚠ CAUTION

After deleting the saved values the controller will reboot.

To drop all saved values of all categories at once the value `64616F6C`$_h$ needs to be written to the object **1011**$_h$:01$_h$ [2]. Thereupon all saved values will get deleted, thus the controller is resetted to factory default. The controller will reboot after deleting all values.

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

10 Programming with NanoJ

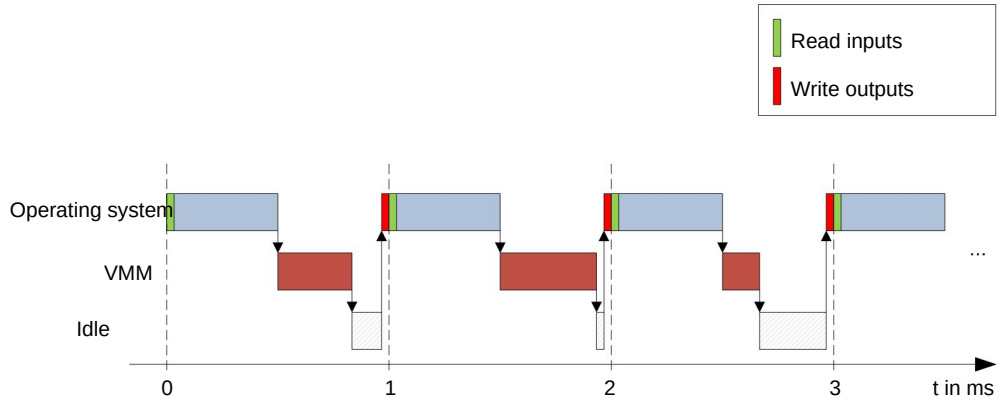
10.1 Introduction

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 via usb. 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**$_h$ of the VMM; the number "5" (timeout) is noted in the VMM error register at object **2302**$_h$.

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**$_h$, **2320**$_h$, and **2330**$_h$. For each mapping, a maximum of 16 entries is allowed.

The specification of the linker section is used to control in NanoJ Easy 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 Inputs 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**$_h$ to **2330**$_h$. 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
2300$_h$	VMM Control (Read/write)
2301$_h$	VMM Status (Read only)
2302$_h$	VMM Error Code (Read only)
2303$_h$	Number Of Active User Program (Read/Write)

OD Index	Name
2304$_h$	Table of available user programs
2310$_h$	VMM Input Data Selection
2320$_h$	VMM Output Data Selection
2330$_h$	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:

- Copy the compiled file via USB
- Write the value "54453554_h " in object **2304**$_h$:01_h.
- Write the value "31000000_h " in object **2304**$_h$:02_h.
- Write the value "1_h " in object **2303**$_h$.
- Check the entry **2302**$_h$ for error codes.
- If no error:

  Launch the program by writing object **2300**$_h$, bit 0 = "1".
- Check the entry **2302**$_h$ for error codes and the object **2301**$_h$, bit 0 = "1".

To stop a running program: Write the bit 0-value = "0" to the entry **2300**$_h$.

10.5 NanoJ Easy V2

10.5.1 Installation and use

Introduction

As an alternative to NanoIP, a user program can also be programmed, uploaded, and controlled with the NanoJ Easy V2 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 square wave signal in the object **2500**$_h$:01_h

1. Copy the following text to the NanoJ Easy 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-Anschluss**" section). The motor controller must be restarted to launch the program; please read the "**NanoJ-Programm**" 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**$_h$ or status word **6041**$_h$.

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

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**$_h$:00$_h$.
- At the end of the 1-ms cycle, the mapping is written that also specifies object **6040**$_h$:00$_h$, 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**$_h$ 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 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 – which specifies the quantity of valid subentries – and the subentries themselves from index 1. The data type in an array never changes, which means that subentry 1 and all following entries always have the same data type.
- RECORD: These objects always consist of one subentry with subindex 0 – which specifies the quantity of valid subentries – and the subentries themselves from index 1. As opposed to

an ARRAY, the data type of subentries may vary, meaning, 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 her 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:

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 apples 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_b: The description at this position applies when bit 2 and bit 3 are at "0".
- Value 01_b: The description at this position applies when bit 2 is at "0" and bit 3 at "1".
- Value 10_b: The description at this position applies when bit 2 is at "1" and bit 3 at "0".
- Value 11_b: 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

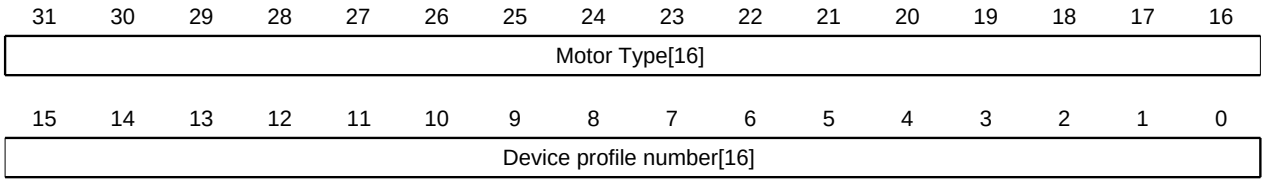
Function

Describes the motor controller type.

Object description

Index	1000_h
Object Name	Device Type
Object Code	VARIABLE
Data type	UNSIGNED32
Persistent	No
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00040192_h
Firmware Version	FIR-v1426
Change History	

Description



Motor Type[16]

Describes the supported motor type.

Device profile number[16]

Describes the supported CANopen standard.

Values:

0129_h (specified value): The DS402 standard is supported.

1001h Error Register

Function

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

Object description

Index	1001_h
Object Name	Error Register
Object Code	VARIABLE
Data type	UNSIGNED8
Persistent	No
Access	Read only
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00_h
Firmware Version	FIR-v1426
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_h
Object Name	Pre-defined Error Field
Object Code	ARRAY
Data type	UNSIGNED32
Persistent	No
Firmware Version	FIR-v1426
Change History	Amount of subentries has changed from 2 to 9

Value description

Subindex	00_h
Name	Number Of Errors
Data type	UNSIGNED8
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	01_h
Name	Standard Error Field
Data type	UNSIGNED32
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	02_h
Name	Standard Error Field
Data type	UNSIGNED32
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	03_h
Name	Standard Error Field
Data type	UNSIGNED32
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	04_h
Name	Standard Error Field
Data type	UNSIGNED32
Access	Read only
PDO Mapping	No

Admissible Values	
Specified Value	00000000_h

Subindex	05_h
Name	Standard Error Field
Data type	UNSIGNED32
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	06_h
Name	Standard Error Field
Data type	UNSIGNED32
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	07_h
Name	Standard Error Field
Data type	UNSIGNED32
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	08_h
Name	Standard Error Field
Data type	UNSIGNED32
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

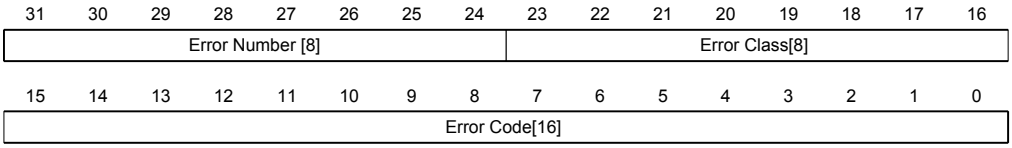
Description

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 **6065**$_h$(Following Error Window) and object **6066**$_h$(Following Error Time Out) have been exceeded, and a fault has been output. This fault must be activated with bit 7 in object **3202**$_h$.

Error Class[8]

This byte is identical to object **1001**$_h$

Error Code[16]

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

Error Code	Description
1000$_h$	General error
2310$_h$	Voltage at output of motor controller too high
3210$_h$	Overvoltage/undervoltage at motor controller input
3220$_h$	Undervoltage at input of motor controller
4200$_h$	Temperature error in motor controller
7305$_h$	Incremental sensor 1 faulty
8000$_h$	Field bus monitoring error
8130$_h$	Only CANopen: Life guard error or heartbeat - error
8611$_h$	Position monitoring error: Subsequent error too large
8612$_h$	Position monitoring error: Reference limit

1008h Manufacturer Device Name

Function

Contains the device name as a string.

Object description

Index	1008$_h$

Object Name	Manufacturer Device Name
Object Code	VARIABLE
Data type	VISIBLE_STRING
Persistent	No
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	
Firmware Version	FIR-v1426
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_h
Object Name	Manufacturer Hardware Version
Object Code	VARIABLE
Data type	VISIBLE_STRING
Persistent	No
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	
Firmware Version	FIR-v1426
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	$100A_h$
Object Name	Manufacturer Software Version
Object Code	VARIABLE
Data type	VISIBLE_STRING

Persistent	No
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	DEV-B50371
Firmware Version	FIR-v1426
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**$_h$ and the baud rate setting from object **2005**$_h$ so they are again available after a restart or power failure. The numerical value `65766173`$_h$ must be written is 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
- Sending of a "Reset Node" CANopen message

Object description

Index	1010$_h$
Object Name	Store Parameter
Object Code	ARRAY
Data type	UNSIGNED32
Persistent	No
Firmware Version	FIR-v1426
Change History	

Value description

Subindex	00$_h$
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	02$_h$

Subindex	01$_h$
Name	Save All The Parameters To Non-volatile Memory
Data type	UNSIGNED32
Access	Read/write

PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	02_h
Name	Save The Comm Parameters To Non-volatile Memory
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Description

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

The value 65766173_h must be written in subindex 2 in order to start the store process. This corresponds to decimal 1702257011_d 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_h
Object Name	Restore Default Parameter
Object Code	ARRAY
Data type	UNSIGNED32
Persistent	No
Firmware Version	FIR-v1426
Change History	

Value description

Subindex	00_h
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	01_h

Subindex	01_h
Name	Restore All Default Parameters
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Description

When value 64616F6C$_h$ 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$_h$
Object Name	Identity Object
Object Code	RECORD
Data type	IDENTITY
Persistent	No
Firmware Version	FIR-v1426
Change History	

Value description

Subindex	00$_h$
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	04$_h$

Subindex	01$_h$
Name	Vendor-ID
Data type	UNSIGNED32
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	0000026C$_h$

Subindex	02$_h$
Name	Product Code
Data type	UNSIGNED32
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00000003$_h$

Subindex	03$_h$

Name	Revision Number
Data type	UNSIGNED32
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00000001_h

Subindex	04_h
Name	Serial Number
Data type	UNSIGNED32
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00000001_h

2010h IP-Configuration

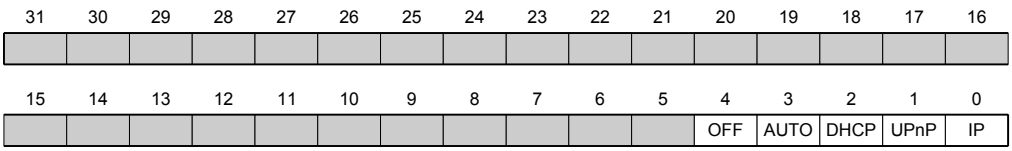
Function

This object is used to set the network configuration.

Object description

Index	2010_h
Object Name	IP-Configuration
Object Code	VARIABLE
Data type	UNSIGNED32
Persistent	yes, category: communication
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h
Firmware Version	FIR-v1426
Change History	

Description



IP

Value = "1": A static IP address is used from object **2011_h** and the network mask from object **2012_h** 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

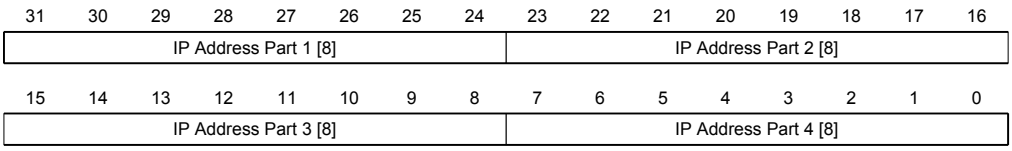
Function

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

Object description

Index	2011_h
Object Name	Static-IP-Address
Object Code	VARIABLE
Data type	UNSIGNED32
Persistent	yes, category: communication
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h
Firmware Version	FIR-v1426
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 => $C0_h$

168 => $A8_h$

2 => 02_h

0 => 0

The associated setting value is then C0A80200$_h$.

2012h Static-IP-Subnet-Mask

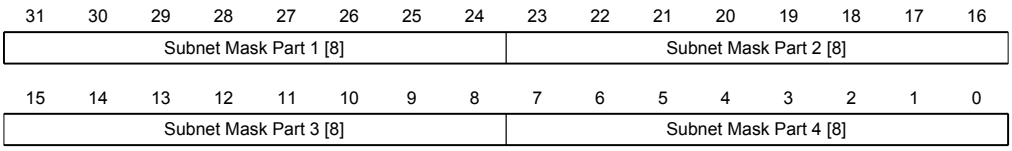
Function

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

Object description

Index	2012$_h$
Object Name	Static-IP-Subnet-Mask
Object Code	VARIABLE
Data type	UNSIGNED32
Persistent	yes, category: communication
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000$_h$
Firmware Version	FIR-v1426
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$_h$

0 => 0

The associated setting value is then FFFFFF00$_h$.

2018h Current-IP-Address

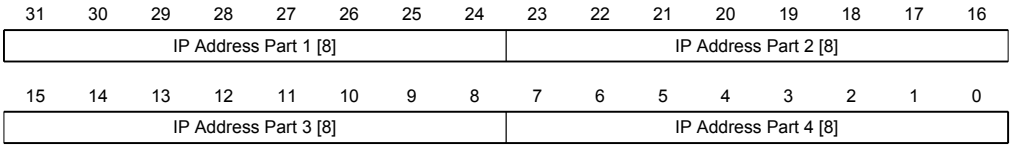
Function

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

Object description

Index	2018_h
Object Name	Current-IP-Address
Object Code	VARIABLE
Data type	UNSIGNED32
Persistent	No
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h
Firmware Version	FIR-v1426
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 => $C0_h$

168 => $A8_h$

2 => 02_h

0 => 0

The associated setting value is then $C0A80200_h$.

2019h Current-IP-Subnet-Mask

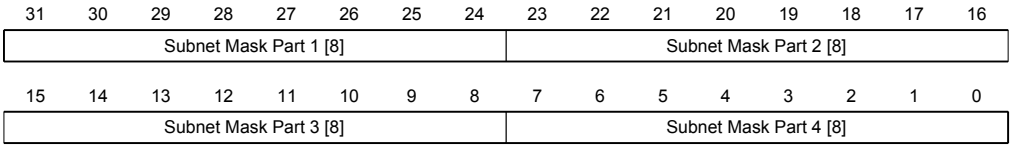
Function

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

Object description

Index	2019_h
Object Name	Current-IP-Subnet-Mask
Object Code	VARIABLE
Data type	UNSIGNED32
Persistent	No
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h
Firmware Version	FIR-v1426
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_h

0 => 0

The associated setting value is then `FFFFFF00`$_h$.

2020h AppInfo-Static-IP-Address

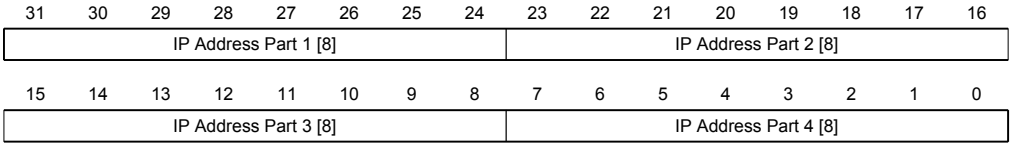
Function

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

Object description

Index	2020_h
Object Name	AppInfo-Static-IP-Address
Object Code	VARIABLE
Data type	UNSIGNED32
Persistent	No
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h
Firmware Version	FIR-v1426
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 => $C0_h$

168 => $A8_h$

2 => 02_h

0 => 0

The associated setting value is then $C0A80200_h$.

2021h AppInfo-Static-IP-Subnet-Mask

Function

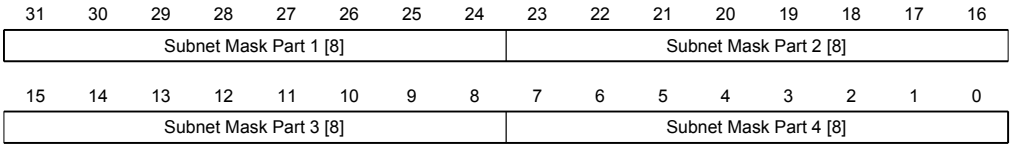
Contains the subnet mask of the static IP address specified in Nanoflash in the form of a 32-bit word.

Object description

Index	2021_h

Object Name	AppInfo-Static-IP-Subnet-Mask
Object Code	VARIABLE
Data type	UNSIGNED32
Persistent	No
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h
Firmware Version	FIR-v1426
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_h

0 => 0

The associated setting value is then `FFFFFF00`$_h$.

2022h Drive Serial Number

Function

This object contains the serial number of the motor controller.

Object description

Index	2022_h
Object Name	Drive Serial Number
Object Code	VARIABLE
Data type	VISIBLE_STRING
Persistent	No
Access	Read only

PDO Mapping	No
Admissible Values	
Specified Value	
Firmware Version	FIR-v1426
Change History	Version FIR-v1422-B36464: Name entry changed from "Drive serial number" to "Drive Serial Number"

2030h Pole Pair Count

Function

Contains the pole pair count of the connected motor.

Object description

Index	2030_h
Object Name	Pole Pair Count
Object Code	VARIABLE
Data type	UNSIGNED32
Persistent	No
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000032_h
Firmware Version	FIR-v1426
Change History	Version FIR-v1422-B36464: Name entry changed from "Pole pair count" to "Pole Pair Count"

2031h Peak Current

Function

Specifies the maximum current in mA.

Object description

Index	2031_h
Object Name	Peak Current
Object Code	VARIABLE
Data type	UNSIGNED32
Persistent	yes, category: user
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	$000009C4_h$
Firmware Version	FIR-v1426
Change History	

2032h Maximum Speed

Function

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

Object description

Index	2032_h
Object Name	Maximum Speed
Object Code	VARIABLE
Data type	UNSIGNED32
Persistent	yes, category: user
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	$00030D40_h$
Firmware Version	FIR-v1426
Change History	

Description

The conversion is based on the numerator and denominator specified in object **$604C_h$**.

2033h Plunger Block

Function

Specifies the maximum positional change in user units (corresponding to Target Position **$607A_h$**) that is permitted in the corresponding direction.

Object description

Index	2033_h
Object Name	Plunger Block
Object Code	VARIABLE
Data type	INTEGER32
Persistent	yes, category: user
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h
Firmware Version	FIR-v1426
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 millivolts.

Object description

Index	2034_h
Object Name	Upper Voltage Warning Level
Object Code	VARIABLE
Data type	UNSIGNED32
Persistent	yes, category: user
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	$0000C92C_h$
Firmware Version	FIR-v1426
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_h** minus 2 volts).

2035h Lower Voltage Warning Level

Function

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

Object description

Index	2035_h
Object Name	Lower Voltage Warning Level
Object Code	VARIABLE
Data type	UNSIGNED32
Persistent	yes, category: user
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00004650_h
Firmware Version	FIR-v1426
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 greater than (voltage of the object **2035_h** 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_h
Object Name	Open Loop Current Reduction Idle Time
Object Code	VARIABLE
Data type	UNSIGNED32
Persistent	yes, category: user
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	$000003E8_h$
Firmware Version	FIR-v1426
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_h
Object Name	Open Loop Current Reduction Value/factor
Object Code	VARIABLE
Data type	INTEGER32
Persistent	yes, category: user
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	$FFFFFFCE_h$
Firmware Version	FIR-v1426
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_h**). The value "-100" corresponds to 100% of the value in object **2031_h**, and the value "-50" is interpreted as 50% of the value in object **2031_h**, etc.

If the value is positive, the current is reduced to the value in mA entered in object **2037_h**.

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_h
Object Name	Brake Controller Timing
Object Code	ARRAY
Data type	UNSIGNED32
Persistent	yes, category: user
Firmware Version	FIR-v1426
Change History	

Value description

Subindex	00_h
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	06_h

Subindex	01_h
Name	Close Brake Idle Time
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	$000003E8_h$

Subindex	02_h
Name	Shutdown Power Idle Time
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	$000003E8_h$

Subindex	03_h
Name	Open Brake Delay Time
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	$000003E8_h$

Subindex	04_h
Name	Start Operation Delay Time
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	$000003E8_h$

Subindex	05_h
Name	PWM Frequency
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	between 1 and 2000 ($7D0_h$)
Specified Value	00000000_h

Subindex	06_h
Name	PWM Duty Cycle
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	between 2 and 100
Specified Value	00000000_h

Description

The subindices have the following functions:

- 01_h: Time between when the motor begins idle and the brake closes.
- 02_h: Time between when the brake closes and the current is lowered.
- 03_h: Time between when a new move command is set and the brake is opened.
- 04_h: Time between when the brake is opened and the motor starts.
- 05_h: Frequency of the brake PWM in hertz.
- 06_h: 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_h
Object Name	Motor Currents
Object Code	ARRAY
Data type	INTEGER32
Persistent	No
Firmware Version	FIR-v1426
Change History	

Value description

Subindex	00_h
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	04_h

Subindex	01_h
Name	I_d
Data type	INTEGER32
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	02_h
Name	I_q
Data type	INTEGER32
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	03_h
Name	I_a
Data type	INTEGER32
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	04_h
Name	I_b
Data type	INTEGER32
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

203Ah Homing On Block Configuration

Function

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

Object description

Index	$203A_h$

Object Name	Homing On Block Configuration
Object Code	ARRAY
Data type	UNSIGNED32
Persistent	No
Access	
PDO Mapping	
Admissible Values	
Specified Value	
Firmware Version	FIR-v1426
Change History	Amount of subentries has changed from 3 to 4

Value description

Subindex	00_h
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	03_h

Subindex	01_h
Name	Minimum Current For Block Detection
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	$000004EC_h$

Subindex	02_h
Name	Period Of Blocking
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	03
Name	Block Detection Time
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000005_h

Description

The subentries have the following function:

- 01_h: Specifies the current limit value from which blocking is to be detected.

- 02_h: Specifies the time in ms that the motor is nevertheless still to travel against the block after block detection.
- 03_h: Specifies the time in ms that the current has to be at least above the minimum current threshold in order to detect a block.

203Bh I2t Parameters

Function

This object contains the parameters for the I^2t monitoring.

The I^2t monitoring is activated when a value greater than 0 is entered in $203B_h:02_h$ (see " **I2t motor overload protection** ").

I^2t can only be use for closed loop mode with a single exception: If I^2t is activated in 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 and into open loop mode with a very high short-time maximum current without damaging the motor.

Object description

Index	$203B_h$
Object Name	I2t Parameters
Object Code	ARRAY
Data type	UNSIGNED32
Persistent	No
Firmware Version	FIR-v1426
Change History	Amount of subentries has changed from 3 to 7

Value description

Subindex	00_h
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	06_h

Subindex	01_h
Name	Nominal Current
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	02_h
Name	Maximum Duration Of Peak Current
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No

Admissible Values	
Specified Value	00000000_h

Subindex	03
Name	Threshold
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	04
Name	CalcValue
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	05
Name	LimitedCurrent
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	06
Name	Status
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Description

The subentries have the following function:

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

2050h Encoder Alignment

Function

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

Object description

Index	2050_h
Object Name	Encoder Alignment
Object Code	VARIABLE
Data type	INTEGER32
Persistent	No
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h
Firmware Version	FIR-v1426
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_h
Object Name	Encoder Optimization
Object Code	ARRAY
Data type	INTEGER32
Persistent	No
Firmware Version	FIR-v1426
Change History	

Value description

Subindex	00_h
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	03_h

Subindex	01_h
Name	Parameter 1

Data type	INTEGER32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	02_h
Name	Parameter 2
Data type	INTEGER32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	03_h
Name	Parameter 3
Data type	INTEGER32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

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_h
Object Name	Encoder Resolution
Object Code	VARIABLE
Data type	INTEGER32
Persistent	No
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00001000_h
Firmware Version	FIR-v1426
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_h
Object Name	Index Polarity
Object Code	VARIABLE
Data type	UNSIGNED8
Persistent	No
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00_h
Firmware Version	FIR-v1426
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_h
Object Name	Index Width
Object Code	VARIABLE
Data type	INTEGER32
Persistent	No
Access	Read/write
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	$FFFFFFFF_h$
Firmware Version	FIR-v1426
Change History	

Description

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

The value -1 ($FFFFFFFF_h$) deactivates encoder monitoring.

2056h Limit Switch Tolerance Band

Function

Specifies how far positive or negative limit 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_h
Object Name	Limit Switch Tolerance Band
Object Code	VARIABLE
Data type	UNSIGNED32
Persistent	yes, category: user
Access	Read/write
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	$000001F4_h$
Firmware Version	FIR-v1426
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_h
Object Name	Clock Direction Multiplier
Object Code	VARIABLE
Data type	INTEGER32
Persistent	yes, category: user
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000080_h
Firmware Version	FIR-v1426
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_h
Object Name	Clock Direction Divider
Object Code	VARIABLE
Data type	INTEGER32
Persistent	yes, category: user
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000001_h
Firmware Version	FIR-v1426
Change History	

2059h Encoder Configuration

Function

This object specifies the supply voltage of the encoder.

Object description

Index	2059_h
Object Name	Encoder Configuration
Object Code	VARIABLE
Data type	UNSIGNED32
Persistent	yes, category: user
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h
Firmware Version	FIR-v1426
Change History	

Description

Setting this object to the value "0" the supply voltage of the encoder is set to 5V. Setting this object to the value "1" the supply voltage of the encoder is set to 24V.

2060h Compensate Polepair Count

Function

Makes it possible to order motor-independent motion blocks.

Object description

Index	2060_h
Object Name	Compensate Polepair Count
Object Code	VARIABLE
Data type	UNSIGNED32
Persistent	yes, category: user
Access	Read/write

PDO Mapping	No
Admissible Values	
Specified Value	00000001_h
Firmware Version	FIR-v1426
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_h
Object Name	Velocity Numerator
Object Code	VARIABLE
Data type	UNSIGNED32
Persistent	yes, category: user
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000001_h
Firmware Version	FIR-v1426
Change History	

Description

The internal operand pertains to full mechanical (**2060_h**=1) or electrical (**2060_h**=0) revolutions per second.

Thus, by setting object **2061_h**=1 and object **2062_h**=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_h
Object Name	Velocity Denominator
Object Code	VARIABLE
Data type	UNSIGNED32
Persistent	yes, category: user

Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	0000003C$_h$
Firmware Version	FIR-v1426
Change History	

Description

The internal operand pertains to full mechanical (**2060**$_h$=1) or electrical (**2060**$_h$=0) revolutions per second.

Thus, by setting object **2061**$_h$=1 and object **2062**$_h$=60, for example, the speed can be specified in rpm 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$_h$
Object Name	Acceleration Numerator
Object Code	VARIABLE
Data type	UNSIGNED32
Persistent	yes, category: user
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000001$_h$
Firmware Version	FIR-v1426
Change History	

Description

The internal operand pertains to full mechanical (**2060**$_h$=1) or electrical (**2060**$_h$=0) revolutions per second.

Thus, by setting object **2063**$_h$=1 and object **2064**$_h$=60, for example, the acceleration can be specified in (revolutions/min)/s ^{2}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$_h$
Object Name	Acceleration Denominator
Object Code	VARIABLE

Data type	UNSIGNED32
Persistent	yes, category: user
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	0000003C$_h$
Firmware Version	FIR-v1426
Change History	

Description

The internal operand pertains to full mechanical (**2060**$_h$=1) or electrical (**2060**$_h$=0) revolutions per second.

Thus, by setting object **2063**$_h$=1 and object **2064**$_h$=60, for example, the acceleration can be specified in (revolutions/min)/s ^{2} 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$_h$
Object Name	Jerk Numerator
Object Code	VARIABLE
Data type	UNSIGNED32
Persistent	yes, category: user
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000001$_h$
Firmware Version	FIR-v1426
Change History	

Description

The internal operand pertains to full mechanical (**2060**$_h$=1) or electrical (**2060**$_h$=0) revolutions per second to the power of 3.

Thus, by setting object **2065**$_h$=1 and object **2066**$_h$=60, for example, the jerk can be specified in (revolutions/min)/s ^{2} 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$_h$
Object Name	Jerk Denominator

Object Code	VARIABLE
Data type	UNSIGNED32
Persistent	yes, category: user
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	$0000003C_h$
Firmware Version	FIR-v1426
Change History	

Description

The internal operand pertains to full mechanical (**2060_h**=1) or electrical (**2060_h**=0) revolutions per second.

Thus, by setting object **2065_h**=1 and object **2066_h**=60, for example, the acceleration can be specified in (revolutions/min)/s ^{2} 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_h
Object Name	Bootup Delay
Object Code	VARIABLE
Data type	UNSIGNED32
Persistent	yes, category: user
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h
Firmware Version	FIR-v1426
Change History	

2101h Fieldbus Module

Function

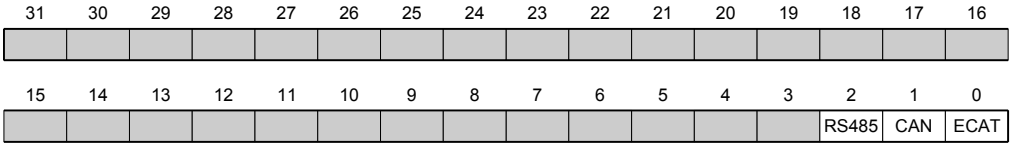
Shows the type of mounted field bus module.

Object description

Index	2101_h
Object Name	Fieldbus Module
Object Code	VARIABLE
Data type	UNSIGNED32
Persistent	No
Access	Read only

PDO Mapping	No
Admissible Values	
Specified Value	00000000_h
Firmware Version	FIR-v1426
Change History	Firmware Version FIR-v1426: Entry "Data Type" modified from "INTEGER32" to "UNSIGNED32"

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

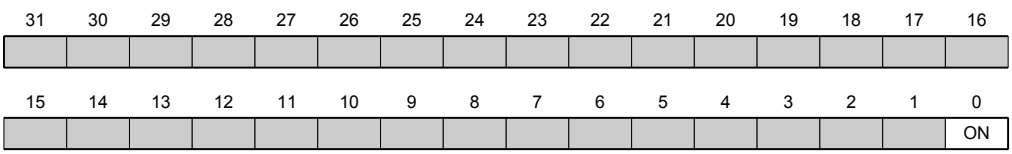
Function

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

Object description

Index	2200_h
Object Name	Sampler Control
Object Code	VARIABLE
Data type	UNSIGNED32
Persistent	No
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h
Firmware Version	FIR-v1426
Change History	

Description



ON

Value = "1": The sampler will be activated

2201h Sampler Status

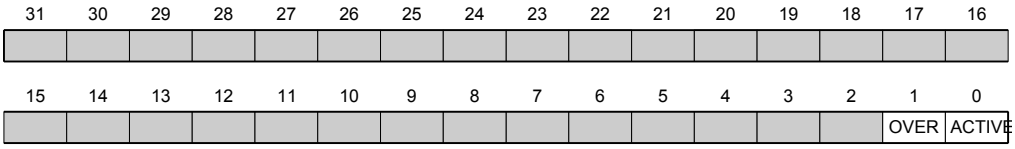
Function

Shows the operating state of the installed sampler.

Object description

Index	2201_h
Object Name	Sampler Status
Object Code	VARIABLE
Data type	UNSIGNED32
Persistent	No
Access	Read only
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h
Firmware Version	FIR-v1426
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_h** bit 0.

2202h Sample Data Selection

Function

As for object **1600_h** 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_h
Object Name	Sample Data Selection
Object Code	RECORD
Data type	PDO_MAPPING
Persistent	No
Firmware Version	FIR-v1426
Change History	

Value description

Subindex	00_h
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	02_h

Subindex	01_h
Name	Sample Value #1
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	60430010_h

Subindex	02_h
Name	Sample Value #2
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	22030220_h

Subindex	03_h
Name	Sample Value #3
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	04_h
Name	Sample Value #4
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	05_h
Name	Sample Value #5
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	06_h
Name	Sample Value #6
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

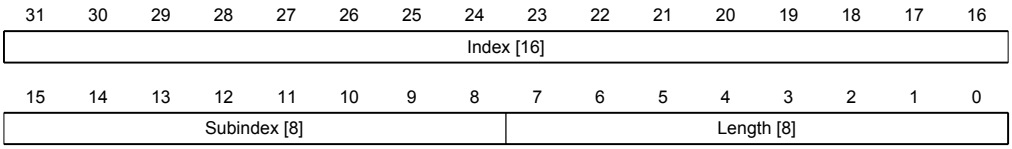
Subindex	07_h
Name	Sample Value #7
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	08_h
Name	Sample Value #8
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

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_h

Object Name	Sampler Buffer Information
Object Code	ARRAY
Data type	UNSIGNED32
Persistent	No
Firmware Version	FIR-v1426
Change History	

Value description

Subindex	00_h
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	03_h

Subindex	01_h
Name	Sample Buffer Size
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	02_h
Name	Sample Buffer Watermark
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	03_h
Name	Sample Tick
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Description

The subindices have the following functions:

- 01_h specifies the maximum size of the sampler buffer in bytes.
- 02_h contains the momentary filling level of the sampler buffer in bytes.
- 03_h 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_h
Object Name	Sample Time In Ms
Object Code	VARIABLE
Data type	UNSIGNED32
Persistent	No
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000001_h
Firmware Version	FIR-v1426
Change History	

2300h VMM Control

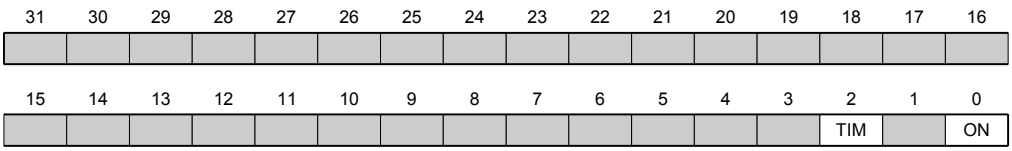
Function

Controls the execution of a user program.

Object description

Index	2300_h
Object Name	VMM Control
Object Code	VARIABLE
Data type	UNSIGNED32
Persistent	yes, category: user
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h
Firmware Version	FIR-v1426
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

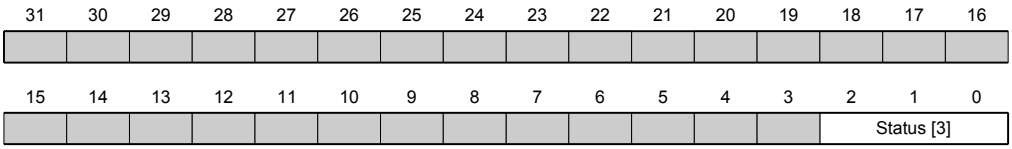
Function

Shows the operating state of the user program.

Object description

Index	2301_h
Object Name	VMM Status
Object Code	VARIABLE
Data type	UNSIGNED32
Persistent	No
Access	Read only
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h
Firmware Version	FIR-v1426
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_h**.

2302h VMM Error Code

Function

Indicates which error occurred when the user program was executed.

Object description

Index	2302_h
Object Name	VMM Error Code
Object Code	VARIABLE
Data type	UNSIGNED32
Persistent	No

Access	Read only
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h
Firmware Version	FIR-v1426
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
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_h**.

Object description

Index	2303_h
Object Name	Number Of Active User Program
Object Code	VARIABLE
Data type	UNSIGNED8
Persistent	yes, category: user
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00_h
Firmware Version	FIR-v1426
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_h
Object Name	Table Of Available User Programs
Object Code	ARRAY
Data type	UNSIGNED32
Persistent	yes, category: user
Firmware Version	FIR-v1426
Change History	

Value description

Subindex	00_h
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	08_h

Subindex	01_h
Name	Name Of User Program 1 UB
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No

Admissible Values	
Specified Value	00000000_h

Subindex	02_h
Name	Name Of User Program 1 LB
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	03_h
Name	Name Of User Program 2 UB
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	04_h
Name	Name Of User Program 2 LB
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	05_h
Name	Name Of User Program 3 UB
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	06_h
Name	Name Of User Program 3 LB
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	07_h
Name	Name Of User Program 4 UB
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	

Specified Value	00000000_h

Subindex	08_h
Name	Name Of User Program 4 LB
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

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_h

e = 65_h

s = 73_h

Thus the two entries in subindices 1 and 2 are:

74657374_h, 00000000_h

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_h
Object Name	VMM Input Data Selection
Object Code	RECORD
Data type	PDO_MAPPING
Persistent	yes, category: user
Firmware Version	FIR-v1426
Change History	Amount of subentries has changed from 2 to 17

Value description

Subindex	00_h
Name	Highest Sub-index Supported

Data type	UNSIGNED8
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	01_h
Name	Mapping #1
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	02_h
Name	Mapping #2
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	03_h
Name	Mapping #3
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	04_h
Name	Mapping #4
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	05_h
Name	Mapping #5
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	06_h
Name	Mapping #6
Data type	UNSIGNED32

Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	07_h
Name	Mapping #7
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	08_h
Name	Mapping #8
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	09_h
Name	Mapping #9
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	$0A_h$
Name	Mapping #10
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	$0B_h$
Name	Mapping #11
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	$0C_h$
Name	Mapping #12
Data type	UNSIGNED32
Access	Read/write

PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	$0D_h$
Name	Mapping #13
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	$0E_h$
Name	Mapping #14
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

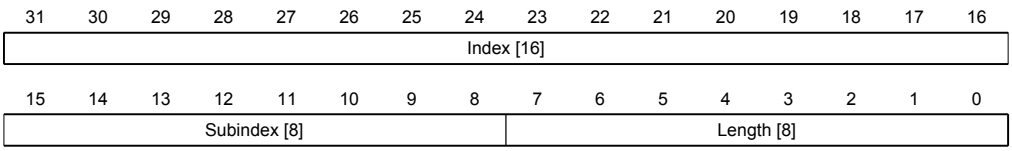
Subindex	$0F_h$
Name	Mapping #15
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	10_h
Name	Mapping #16
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

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_h
Object Name	VMM Output Data Selection
Object Code	RECORD
Data type	PDO_MAPPING
Persistent	yes, category: user
Firmware Version	FIR-v1426
Change History	Amount of subentries has changed from 2 to 17

Value description

Subindex	00_h
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	01_h
Name	Mapping #1
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	02_h
Name	Mapping #2
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	03_h

Name	Mapping #3
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	04_h
Name	Mapping #4
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	05_h
Name	Mapping #5
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	06_h
Name	Mapping #6
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	07_h
Name	Mapping #7
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	08_h
Name	Mapping #8
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	09_h
Name	Mapping #9

Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	$0A_h$
Name	Mapping #10
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	$0B_h$
Name	Mapping #11
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	$0C_h$
Name	Mapping #12
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	$0D_h$
Name	Mapping #13
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	$0E_h$
Name	Mapping #14
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	$0F_h$
Name	Mapping #15
Data type	UNSIGNED32

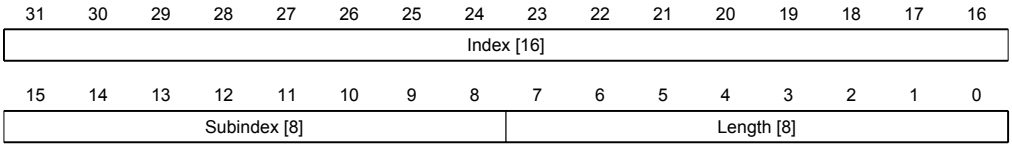
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	10_h
Name	Mapping #16
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

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_h
Object Name	VMM In/output Data Selection
Object Code	RECORD
Data type	PDO_MAPPING
Persistent	yes, category: user
Firmware Version	FIR-v1426
Change History	Amount of subentries has changed from 2 to 17

Value description

Subindex	00_h

Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	01_h
Name	Mapping #1
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	02_h
Name	Mapping #2
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	03_h
Name	Mapping #3
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	04_h
Name	Mapping #4
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	05_h
Name	Mapping #5
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	06_h
Name	Mapping #6

Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	07_h
Name	Mapping #7
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	08_h
Name	Mapping #8
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	09_h
Name	Mapping #9
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	$0A_h$
Name	Mapping #10
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	$0B_h$
Name	Mapping #11
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	$0C_h$
Name	Mapping #12
Data type	UNSIGNED32

Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	$0D_h$
Name	Mapping #13
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	$0E_h$
Name	Mapping #14
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

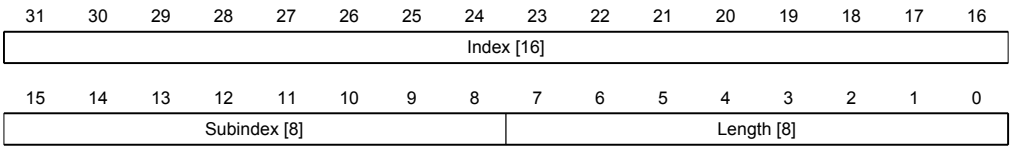
Subindex	$0F_h$
Name	Mapping #15
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	10_h
Name	Mapping #16
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

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_h
Object Name	VMM Inputs
Object Code	ARRAY
Data type	INTEGER32
Persistent	No
Firmware Version	FIR-v1426
Change History	Amount of subentries has changed from 2 to 33

Value description

Subindex	00_h
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	20_h

Subindex	01_h
Name	VMM Input 1#
Data type	INTEGER32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	02_h
Name	VMM Input 2#
Data type	INTEGER32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	03_h

Name	VMM Input 3#
Data type	INTEGER32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	04_h
Name	VMM Input 4#
Data type	INTEGER32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	05_h
Name	VMM Input 5#
Data type	INTEGER32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	06_h
Name	VMM Input 6#
Data type	INTEGER32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	07_h
Name	VMM Input 7#
Data type	INTEGER32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	08_h
Name	VMM Input 8#
Data type	INTEGER32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	09_h
Name	VMM Input 9#

Data type	INTEGER32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	$0A_h$
Name	VMM Input 10#
Data type	INTEGER32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	$0B_h$
Name	VMM Input 11#
Data type	INTEGER32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	$0C_h$
Name	VMM Input 12#
Data type	INTEGER32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	$0D_h$
Name	VMM Input 13#
Data type	INTEGER32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	$0E_h$
Name	VMM Input 14#
Data type	INTEGER32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	$0F_h$
Name	VMM Input 15#
Data type	INTEGER32

Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	10_h
Name	VMM Input 16#
Data type	INTEGER32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	11_h
Name	VMM Input 17#
Data type	INTEGER32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	12_h
Name	VMM Input 18#
Data type	INTEGER32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	13_h
Name	VMM Input 19#
Data type	INTEGER32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	14_h
Name	VMM Input 20#
Data type	INTEGER32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	15_h
Name	VMM Input 21#
Data type	INTEGER32
Access	Read/write

PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	16_h
Name	VMM Input 22#
Data type	INTEGER32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	17_h
Name	VMM Input 23#
Data type	INTEGER32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	18_h
Name	VMM Input 24#
Data type	INTEGER32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	19_h
Name	VMM Input 25#
Data type	INTEGER32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	$1A_h$
Name	VMM Input 26#
Data type	INTEGER32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	$1B_h$
Name	VMM Input 27#
Data type	INTEGER32
Access	Read/write
PDO Mapping	RX - PDO

Admissible Values	
Specified Value	00000000_h

Subindex	$1C_h$
Name	VMM Input 28#
Data type	INTEGER32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	$1D_h$
Name	VMM Input 29#
Data type	INTEGER32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	$1E_h$
Name	VMM Input 30#
Data type	INTEGER32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	$1F_h$
Name	VMM Input 31#
Data type	INTEGER32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	20_h
Name	VMM Input 32#
Data type	INTEGER32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

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_h
Object Name	VMM Outputs
Object Code	ARRAY
Data type	INTEGER32
Persistent	No
Firmware Version	FIR-v1426
Change History	Amount of subentries has changed from 2 to 33

Value description

Subindex	00_h
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	20_h

Subindex	01_h
Name	VMM Output 1#
Data type	INTEGER32
Access	Read/write
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	02_h
Name	VMM Output 2#
Data type	INTEGER32
Access	Read/write
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	03_h
Name	VMM Output 3#
Data type	INTEGER32
Access	Read/write
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	04_h
Name	VMM Output 4#
Data type	INTEGER32
Access	Read/write
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	05_h
Name	VMM Output 5#
Data type	INTEGER32
Access	Read/write
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	06_h
Name	VMM Output 6#
Data type	INTEGER32
Access	Read/write
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	07_h
Name	VMM Output 7#
Data type	INTEGER32
Access	Read/write
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	08_h
Name	VMM Output 8#
Data type	INTEGER32
Access	Read/write
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	09_h
Name	VMM Output 9#
Data type	INTEGER32
Access	Read/write
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	$0A_h$

Name	VMM Output 10#
Data type	INTEGER32
Access	Read/write
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	$0B_h$
Name	VMM Output 11#
Data type	INTEGER32
Access	Read/write
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	$0C_h$
Name	VMM Output 12#
Data type	INTEGER32
Access	Read/write
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	$0D_h$
Name	VMM Output 13#
Data type	INTEGER32
Access	Read/write
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	$0E_h$
Name	VMM Output 14#
Data type	INTEGER32
Access	Read/write
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	$0F_h$
Name	VMM Output 15#
Data type	INTEGER32
Access	Read/write
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	10_h
Name	VMM Output 16#

Data type	INTEGER32
Access	Read/write
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	11_h
Name	VMM Output 17#
Data type	INTEGER32
Access	Read/write
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	12_h
Name	VMM Output 18#
Data type	INTEGER32
Access	Read/write
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	13_h
Name	VMM Output 19#
Data type	INTEGER32
Access	Read/write
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	14_h
Name	VMM Output 20#
Data type	INTEGER32
Access	Read/write
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	15_h
Name	VMM Output 21#
Data type	INTEGER32
Access	Read/write
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	16_h
Name	VMM Output 22#
Data type	INTEGER32

Access	Read/write
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	17_h
Name	VMM Output 23#
Data type	INTEGER32
Access	Read/write
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	18_h
Name	VMM Output 24#
Data type	INTEGER32
Access	Read/write
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	19_h
Name	VMM Output 25#
Data type	INTEGER32
Access	Read/write
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	$1A_h$
Name	VMM Output 26#
Data type	INTEGER32
Access	Read/write
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	$1B_h$
Name	VMM Output 27#
Data type	INTEGER32
Access	Read/write
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	$1C_h$
Name	VMM Output 28#
Data type	INTEGER32
Access	Read/write

PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	$1D_h$
Name	VMM Output 29#
Data type	INTEGER32
Access	Read/write
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	$1E_h$
Name	VMM Output 30#
Data type	INTEGER32
Access	Read/write
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	$1F_h$
Name	VMM Output 31#
Data type	INTEGER32
Access	Read/write
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	20_h
Name	VMM Output 32#
Data type	INTEGER32
Access	Read/write
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h

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_h
Object Name	VMM Debug Output
Object Code	ARRAY

Data type	UNSIGNED8
Persistent	No
Firmware Version	FIR-v1426
Change History	Amount of subentries has changed from 2 to 65

Value description

Subindex	00_h
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	01_h
Name	Value #1
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	02_h
Name	Value #2
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	03_h
Name	Value #3
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	04_h
Name	Value #4
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	05_h
Name	Value #5

Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	06_h
Name	Value #6
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	07_h
Name	Value #7
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	08_h
Name	Value #8
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	09_h
Name	Value #9
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	$0A_h$
Name	Value #10
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	$0B_h$
Name	Value #11
Data type	UNSIGNED8

Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	$0C_h$
Name	Value #12
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	$0D_h$
Name	Value #13
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	$0E_h$
Name	Value #14
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	$0F_h$
Name	Value #15
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	10_h
Name	Value #16
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	11_h
Name	Value #17
Data type	UNSIGNED8
Access	Read only

PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	12_h
Name	Value #18
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	13_h
Name	Value #19
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	14_h
Name	Value #20
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	15_h
Name	Value #21
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	16_h
Name	Value #22
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	17_h
Name	Value #23
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No

Admissible Values	
Specified Value	00_h

Subindex	18_h
Name	Value #24
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	19_h
Name	Value #25
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	$1A_h$
Name	Value #26
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	$1B_h$
Name	Value #27
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	$1C_h$
Name	Value #28
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	$1D_h$
Name	Value #29
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	

Specified Value	00_h

Subindex	$1E_h$
Name	Value #30
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	$1F_h$
Name	Value #31
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	20_h
Name	Value #32
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	21_h
Name	Value #33
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	22_h
Name	Value #34
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	23_h
Name	Value #35
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	24_h
Name	Value #36
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	25_h
Name	Value #37
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	26_h
Name	Value #38
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	27_h
Name	Value #39
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	28_h
Name	Value #40
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	29_h
Name	Value #41
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	$2A_h$

Name	Value #42
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	$2B_h$
Name	Value #43
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	$2C_h$
Name	Value #44
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	$2D_h$
Name	Value #45
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	$2E_h$
Name	Value #46
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	$2F_h$
Name	Value #47
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	30_h
Name	Value #48

Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	31_h
Name	Value #49
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	32_h
Name	Value #50
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	33_h
Name	Value #51
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	34_h
Name	Value #52
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	35_h
Name	Value #53
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	36_h
Name	Value #54
Data type	UNSIGNED8

Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	37_h
Name	Value #55
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	38_h
Name	Value #56
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	39_h
Name	Value #57
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	$3A_h$
Name	Value #58
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	$3B_h$
Name	Value #59
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	$3C_h$
Name	Value #60
Data type	UNSIGNED8
Access	Read only

PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	$3D_h$
Name	Value #61
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	$3E_h$
Name	Value #62
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	$3F_h$
Name	Value #63
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Subindex	40_h
Name	Value #64
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Description

The VMM program stores the debug outputs here that have been called up with the function `VmmDebugOutputString()`, `VmmDebugOutputInt()`, and suchlike. A detailed description of the debug output can be found in the " **Debug output**" sub-section of the " **Programming with NanoJ**" section.

3202h Motor Drive Submode Select

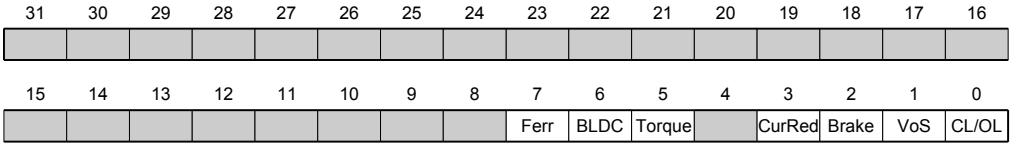
Function

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_h
Object Name	Motor Drive Submode Select
Object Code	VARIABLE
Data type	UNSIGNED32
Persistent	yes, category: user
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h
Firmware Version	FIR-v1426
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 **$605E_h$**)

320Ah Motor Drive Sensor Display Open Loop

Function

It can be used to change the source for objects **6044_h** and **6064_h** in open loop mode.

Object description

Index	$320A_h$
Object Name	Motor Drive Sensor Display Open Loop
Object Code	ARRAY
Data type	INTEGER32
Persistent	yes, category: user
Firmware Version	FIR-v1426
Change History	

Value description

Subindex	00_h
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	04_h

Subindex	01_h
Name	Commutation
Data type	INTEGER32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	02_h
Name	Torque
Data type	INTEGER32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	03_h
Name	Velocity
Data type	INTEGER32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	$FFFFFFFF_h$

Subindex	04_h
Name	Position
Data type	INTEGER32
Access	Read/write
PDO Mapping	No
Admissible Values	

Specified Value	$FFFFFFFF_h$

Description

The following subindices haven a meaning:

- 01_h: Unused
- 02_h: Unused
- 03_h: Changes the source of object **6044_h**:
  - Value = "-1": The internally calculated value is entered in object **6044_h**
  - Value = "0": The value is kept at 0
  - Value = "1": The encoder value is entered in object **6044_h**
- 04_h: Changes the source of object **6064_h**:
  - Value = "-1": The internally calculated value is entered in object **6064_h**
  - Value = "0": The value is kept at 0
  - Value = "1": The encoder value is entered in object **6064_h**

320Bh Motor Drive Sensor Display Closed Loop

Function

It can be used to change the source for objects **6044_h** and **6064_h** in closed loop mode.

Object description

Index	$320B_h$
Object Name	Motor Drive Sensor Display Closed Loop
Object Code	ARRAY
Data type	INTEGER32
Persistent	yes, category: user
Firmware Version	FIR-v1426
Change History	

Value description

Subindex	00_h
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	04_h

Subindex	01_h
Name	Commutation
Data type	INTEGER32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	02_h
Name	Torque

Data type	INTEGER32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	03_h
Name	Velocity
Data type	INTEGER32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000001_h

Subindex	04_h
Name	Position
Data type	INTEGER32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000001_h

Description

The following subindices haven a meaning:

- 01_h: Unused
- 02_h: Unused
- 03_h: Changes the source of object **6044_h**:
  - Value = "-1": The internally calculated value is entered in object **6044_h**
  - Value = "0": The value is kept at 0
  - Value = "1": The encoder value is entered in object **6044_h**
- 04_h: Changes the source of object **6064_h**:
  - Value = "-1": The internally calculated value is entered in object **6064_h**
  - Value = "0": The value is kept at 0
  - Value = "1": The encoder value is entered in object **6064_h**

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_h
Object Name	Motor Drive Parameter Set
Object Code	ARRAY
Data type	INTEGER32
Persistent	yes, category: user
Firmware Version	FIR-v1426

Change History	Amount of subentries has changed from 9 to 11

Value description

Subindex	00_h
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	$0A_h$

Subindex	01_h
Name	S_P
Data type	INTEGER32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000800_h

Subindex	02_h
Name	S_I
Data type	INTEGER32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	03_h
Name	V_P
Data type	INTEGER32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	$00001B58_h$

Subindex	04_h
Name	V_I
Data type	INTEGER32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000004_h

Subindex	05_h
Name	Id_P
Data type	INTEGER32
Access	Read/write
PDO Mapping	No

Admissible Values	
Specified Value	$000668A0_h$

Subindex	06_h
Name	Id_I
Data type	INTEGER32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	$00002EE0_h$

Subindex	07_h
Name	Iq_P
Data type	INTEGER32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	$000668A0_h$

Subindex	08_h
Name	Iq_I
Data type	INTEGER32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	$00002EE0_h$

Subindex	09
Name	I_P
Data type	INTEGER32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00027100_h

Subindex	0A
Name	I_I
Data type	INTEGER32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	$000055F0_h$

Description

- Subindex 00_h: Number of entries
- Subindex 01_h: Proportional value of the S control
- Subindex 02_h: Integral value of the S control
- Subindex 03_h: Proportional value of the V control
- Subindex 04_h: Integral value of the V control

- Subindex 05_h: (Closed Loop) Proportional value of the current controller for the field-forming component
- Subindex 06_h: (Closed Loop) Integral value of the current controller for the field-forming component
- Subindex 07_h: (Closed Loop) Proportional value of the current controller for the torque-forming component
- Subindex 08_h: (Closed Loop) Integral value of the current controller for the torque-forming component
- Subindex 09_h: (Open Loop) Proportional value of the current controller for the torque-forming component
- Subindex $0A_h$: (Open Loop) Integral value of the current controller for the torque-forming component

3220h Analog Inputs

Function

Shows the present values of the analog inputs in [digits].

Object **3221_h** allows the respective analog input to be configured as a current or voltage input.

Object description

Index	3220_h
Object Name	Analog Inputs
Object Code	ARRAY
Data type	INTEGER16
Persistent	No
Firmware Version	FIR-v1426
Change History	

Value description

Subindex	00_h
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	02_h

Subindex	01_h
Name	Analogue Input 1
Data type	INTEGER16
Access	Read only
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	0000_h

Subindex	02_h
Name	Analogue Input 2
Data type	INTEGER16
Access	Read only
PDO Mapping	TX - PDO
Admissible Values	

Specified Value	0000_h

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

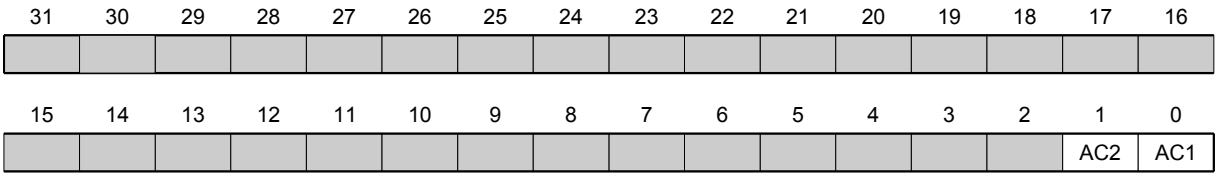
Function

This object can be used to change an analog input from voltage to current measurement.

Object description

Index	3221_h
Object Name	Analogue Inputs Control
Object Code	VARIABLE
Data type	INTEGER32
Persistent	yes, category: user
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h
Firmware Version	FIR-v1426
Change History	

Description



In general: If a bit is set to 0, the analog input measures the voltage; if the bit is set to 1, the current is measured.

AC1

Setting for analog input 1

AC2

Setting for analog input 2

3240h Digital Inputs Control

Function

This object can be used to manipulate the digital inputs as described in the " **Digital inputs and outputs**" section. For all the following subindices, bit 0 pertains to digital input 1, bit 1 pertains to input 2, etc.

Object description

Index	3240_h
Object Name	Digital Inputs Control
Object Code	ARRAY
Data type	UNSIGNED32
Persistent	yes, category: user
Firmware Version	FIR-v1426
Change History	Version 1.0.3: Subindex 01_h: The "Name" entry was changed from "Special Function Disable" to "Special Function Enable"

Value description

Subindex	00_h
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	07_h

Subindex	01_h
Name	Special Function Enable
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	02_h
Name	Function Inverted
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	03_h
Name	Force Enable
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	04_h
Name	Force Value
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	RX - PDO

Admissible Values	
Specified Value	00000000_h

Subindex	05_h
Name	Raw Value
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	06_h
Name	Input Range Select
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	07_h
Name	Differential Select
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Description

The subentries have the following function:

- 01_h: This subindex switches on the special functions of the respective input if the bit has the value "1".
- 02_h: This subindex inverts the logic of an input if the respective input has the value "1".
- 03_h: 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_h regardless of the applied voltage level.
- 04_h: This subindex specifies the input value to be forced.
- 05_h: This subindex always contains the read, unmodified input value.
- 06_h: This subindex switches the switching thresholds between 5 V (value "0") and 24 V (value "1") if the input supports this function.
- 07_h: This subindex switches the inputs from a a single ended (value "0") to differential (value "1") input if the inputs support this function.

3250h Digital Outputs Control

Function

This object can be used to control the digital outputs as described in the " **Digital inputs and outputs**" section. For all the following subindices, bit 0 pertains to digital output 1, bit 1 pertains to output 2, etc.

Object description

Index	3250_h

Object Name	Digital Outputs Control
Object Code	ARRAY
Data type	UNSIGNED32
Persistent	yes, category: user
Firmware Version	FIR-v1426
Change History	Firmware Version FIR-v1426: Subindex 01_h: Entry "Name" changed from "Special Function Disable" auf "Special Function Enable"

Value description

Subindex	00_h
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	05_h

Subindex	01_h
Name	Special Function Enable
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	$000F0001_h$

Subindex	02_h
Name	Function Inverted
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	03_h
Name	Force Enable
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	04_h
Name	Force Value
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	05_h
Name	Raw Value
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Description

The subentries have the following function:

- 01_h: No function.
- 02_h: This subindex inverts the logic (from opener logic to closer logic)
- 03_h: This subindex forces an output value if the bit has the value "1". The level of the output is defined in subindex 4_h.
- 04_h: 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_h: In this subindex, the bit combination applied to the outputs is stored.

3320h Read Analogue Input

Function

DESCRIPTION!

Object description

Index	3320_h
Object Name	Read Analogue Input
Object Code	ARRAY
Data type	INTEGER32
Persistent	No
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	
Firmware Version	FIR-v1426
Change History	

Value description

Subindex	
Name	Number Of Analogue Inputs
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	02_h

Subindex	
Name	Analogue Input 1
Data type	INTEGER32

Access	Read only
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	
Name	Analogue Input 2
Data type	INTEGER32
Access	Read only
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h

Description

3321h Analogue Input Offset

Function

Offset that is added to the read-in analog value (**3320**$_h$) before division with the divider from object **3322**$_h$ is carried out.

Object description

Index	3321_h
Object Name	Analogue Input Offset
Object Code	ARRAY
Data type	INTEGER32
Persistent	yes, category: user
Firmware Version	FIR-v1426
Change History	

Value description

Subindex	00_h
Name	Number Of Analogue Inputs
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	02_h

Subindex	01_h
Name	Analogue Input 1
Data type	INTEGER32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Subindex	02_h

Name	Analogue Input 2
Data type	INTEGER32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Description

- Subindex 00_h: Number of offsets
- Subindex 01_h: Offset for analog input 1
- Subindex 02_h: Offset for analog input 2

3322h Analogue Input Pre-scaling

Function

Value with which the read-in analog value (**3320_h**, **3321_h**) is divided before it is written into object **3320_h**.

Object description

Index	3322_h
Object Name	Analogue Input Pre-scaling
Object Code	ARRAY
Data type	INTEGER32
Persistent	yes, category: user
Firmware Version	FIR-v1426
Change History	

Value description

Subindex	00_h
Name	Number Of Analogue Inputs
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	02_h

Subindex	01_h
Name	Analogue Input 1
Data type	INTEGER32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000001_h

Subindex	02_h
Name	Analogue Input 2
Data type	INTEGER32
Access	Read/write

PDO Mapping	No
Admissible Values	
Specified Value	00000001_h

Description

- Subindex 00_h: Number of dividers
- Subindex 01_h: Divider for analog input 1
- Subindex 02_h: Divider for analog input 2

3700h Following Error Option Code

Function

The object contains the action to be executed if a "following error" is triggered.

Object description

Index	3700_h
Object Name	Following Error Option Code
Object Code	VARIABLE
Data type	INTEGER16
Persistent	yes, category: user
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	$FFFF_h$
Firmware Version	FIR-v1426
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

603Fh Error Code

Function

Contains the last error that occurred.

Object description

Index	$603F_h$
Object Name	Error Code
Object Code	VARIABLE
Data type	UNSIGNED16
Persistent	No

Access	Read only
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	0000_h
Firmware Version	FIR-v1426
Change History	

Description

For the meaning of the error, see object **1003_h** (Pre-defined Error Field).

6040h Controlword

Function

The motor is switched on and travel commands can be carried out with this object.

Object description

Index	6040_h
Object Name	Controlword
Object Code	VARIABLE
Data type	UNSIGNED16
Persistent	No
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	0000_h
Firmware Version	FIR-v1426
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": Switches the "Quick stop" state

EO (Enable Operation)

Value = "1": Switches to the "Enable operation" state

OMS [3] (Operation Mode Specific)

Meaning depends on the selected operating mode

FR (Fault Reset)

Resets an error (if possible)

HALT

Value = "1": Triggers a stop

6041h Statusword

Function

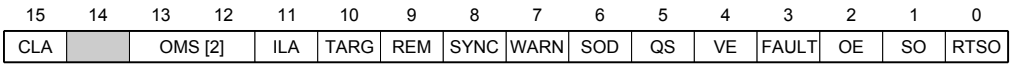
This object queries whether the state commanded with object **6040**$_h$ (control word) has been reached.

Object description

Index	6041$_h$
Object Name	Statusword
Object Code	VARIABLE
Data type	UNSIGNED16
Persistent	No
Access	Read only
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	0000$_h$
Firmware Version	FIR-v1426
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.



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 Disabled)

Value = "1": Motor controller is in the "Switched on disabled" state

WARN (Warning)
Value = "1": Warning

REM (Remote)
Remote (value of bit always "1")

TARG (Target Reached)
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 Vl Target Velocity

Function

Specifies the target speed in user units.

Object description

Index	6042_h
Object Name	Vl Target Velocity
Object Code	VARIABLE
Data type	INTEGER16
Persistent	No
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	$00C8_h$
Firmware Version	FIR-v1426
Change History	

6043h Vl Velocity Demand

Function

Specifies the actual target speed in user units.

Object description

Index	6043_h
Object Name	Vl Velocity Demand
Object Code	VARIABLE
Data type	INTEGER16
Persistent	No
Access	Read only
PDO Mapping	TX - PDO

Admissible Values	
Specified Value	0000_h
Firmware Version	FIR-v1426
Change History	

6044h Vl 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 **$320A_h$**:03_h.

In closed loop mode, the source of this object can be set either to the internal, calculated value or to the encoder with object **$320B_h$**:03_h.

Object description

Index	6044_h
Object Name	Vl Velocity Actual Value
Object Code	VARIABLE
Data type	INTEGER16
Persistent	No
Access	Read only
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	0000_h
Firmware Version	FIR-v1426
Change History	

6046h Vl Velocity Min Max Amount

Function

The minimum speed and maximum speed in user units can be set with this object.

Object description

Index	6046_h
Object Name	Vl Velocity Min Max Amount
Object Code	ARRAY
Data type	UNSIGNED32
Persistent	yes, category: user
Firmware Version	FIR-v1426
Change History	

Value description

Subindex	00_h
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	Read only

PDO Mapping	No
Admissible Values	
Specified Value	02_h

Subindex	01_h
Name	MinAmount
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Subindex	02_h
Name	MaxAmount
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	$00004E20_h$

Description

Subindex 1 contains the minimum speed.

Subindex 2 contains the maximum speed.

Note

If the magnitude of the specified target speed (object **6042_h**) 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_h** (status word).

6048h Vl Velocity Acceleration

Function

Sets the acceleration ramp in velocity mode (see " **Velocity**").

Object description

Index	6048_h
Object Name	Vl Velocity Acceleration
Object Code	RECORD
Data type	VELOCITY_ACCELERATION_DECELERATION
Persistent	yes, category: user
Firmware Version	FIR-v1426
Change History	

Value description

Subindex	00_h

Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	02_h

Subindex	01_h
Name	DeltaSpeed
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	$000001F4_h$

Subindex	02_h
Name	DeltaTime
Data type	UNSIGNED16
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	0001_h

Description

The acceleration is specified as a fraction:

Speed change per time change.

Subindex 01_h: Contains the speed change in steps per second (U32).

Subindex 02_h: Contains the time change in seconds (U16).

6049h Vl Velocity Deceleration

Function

Sets the brake ramp in velocity mode (siehe chapter " **Velocity**").

Object description

Index	6049_h
Object Name	Vl Velocity Deceleration
Object Code	RECORD
Data type	VELOCITY_ACCELERATION_DECELERATION
Persistent	yes, category: user
Firmware Version	FIR-v1426
Change History	

Value description

Subindex	00_h
Name	Highest Sub-index Supported
Data type	UNSIGNED8

Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	02_h

Subindex	01_h
Name	DeltaSpeed
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	$000001F4_h$

Subindex	02_h
Name	DeltaTime
Data type	UNSIGNED16
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	0001_h

604Ah Vl Velocity Quick Stop

Function

This object defines the deceleration if the quick stop state is initiated in velocity mode.

Object description

Index	$604A_h$
Object Name	Vl Velocity Quick Stop
Object Code	RECORD
Data type	VELOCITY_ACCELERATION_DECELERATION
Persistent	yes, category: user
Firmware Version	FIR-v1426
Change History	

Value description

Subindex	00_h
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	02_h

Subindex	01_h
Name	DeltaSpeed
Data type	UNSIGNED32
Access	Read/write

PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00001388_h

Subindex	02_h
Name	DeltaTime
Data type	UNSIGNED16
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	0001_h

Description

Subindex 1 contains the speed change, and subindex 2 the associated time in seconds.

Both together are computed as the acceleration:

Velocity Quick Stop = DeltaSpeed (**604A**$_h$:01_h)/DeltaTime (**604A**$_h$:02_h)

604Ch Vl Dimension Factor

Function

The unit for the speed specifications for the objects that pertain to the Velocity Mode are defined here.

Object description

Index	$604C_h$
Object Name	Vl Dimension Factor
Object Code	ARRAY
Data type	INTEGER32
Persistent	yes, category: user
Firmware Version	FIR-v1426
Change History	

Value description

Subindex	00_h
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	02_h

Subindex	01_h
Name	Vl Dimension Factor Numerator
Data type	INTEGER32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000001_h

Subindex	02_h
Name	VI Dimension Factor Denominator
Data type	INTEGER32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	$0000003C_h$

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 object **2060_h**, the selection is made of whether these are electrical (**2060_h** = 0) or mechanical (**2060_h** = 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	$605A_h$
Object Name	Quick Stop Option Code
Object Code	VARIABLE
Data type	INTEGER16
Persistent	yes, category: user
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	0001_h
Firmware Version	FIR-v1426
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	$605B_h$
Object Name	Shutdown Option Code
Object Code	VARIABLE
Data type	INTEGER16
Persistent	yes, category: user
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	0001_h
Firmware Version	FIR-v1426
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	$605C_h$
Object Name	Disable Option Code
Object Code	VARIABLE
Data type	INTEGER16
Persistent	yes, category: user
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	0001_h
Firmware Version	FIR-v1426
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

605Dh Halt Option Code

Function

The object contains the action to be executed if stop bit 8 is set in control word **6040**$_h$.

Object description

Index	605D$_h$
Object Name	Halt Option Code
Object Code	VARIABLE
Data type	INTEGER16
Persistent	yes, category: user
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	0001$_h$
Firmware Version	FIR-v1426
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$_h$
Object Name	Fault Option Code
Object Code	VARIABLE
Data type	INTEGER16
Persistent	yes, category: user

Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	0002_h
Firmware Version	FIR-v1426
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_h
Object Name	Modes Of Operation
Object Code	VARIABLE
Data type	INTEGER8
Persistent	No
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00_h
Firmware Version	FIR-v1426
Change History	

Description

Mode	Description
-1	Clock/direction mode
0	No mode change/no mode assigned
1	Profile Position Mode
2	Velocity Mode
3	Profile Velocity Mode
4	Profile Torque Mode
5	Reserved
6	Homing Mode
7	Not assigned
8 to 127	Reserved

6061h Modes Of Operation Display

Function

Contains the current operating mode set in object **6060**$_h$ ("Modes Of Operation").

Object description

Index	6061$_h$
Object Name	Modes Of Operation Display
Object Code	VARIABLE
Data type	INTEGER8
Persistent	No
Access	Read only
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00$_h$
Firmware Version	FIR-v1426
Change History	

6062h Position Demand Value

Function

Specifies the actual set position in user units.

Object description

Index	6062$_h$
Object Name	Position Demand Value
Object Code	VARIABLE
Data type	INTEGER32
Persistent	No
Access	Read only
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000$_h$
Firmware Version	FIR-v1426
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$_h$
Object Name	Position Actual Internal Value
Object Code	VARIABLE
Data type	INTEGER32

Persistent	No
Access	Read only
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h
Firmware Version	FIR-v1426
Change History	

6064h Position Actual Value

Function

Contains the current actual position (encoder position converted acc. 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**$_h$:04_h.

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**$_h$:04_h.

Object description

Index	6064_h
Object Name	Position Actual Value
Object Code	VARIABLE
Data type	INTEGER32
Persistent	No
Access	Read only
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h
Firmware Version	FIR-v1426
Change History	

6065h Following Error Window

Function

Specifies the maximum following error symmetrically to the demanded position.

Object description

Index	6065_h
Object Name	Following Error Window
Object Code	VARIABLE
Data type	UNSIGNED32
Persistent	No
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000100_h
Firmware Version	FIR-v1426

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**$_h$ (status word). The deviation must be longer than the time in object **6066**$_h$.

To obtain an automatic response to the error, bit 7 must be activated in object **3202**$_h$. A fault is generated if the "following error" is created – and reacts correspondingly to it (**6041**$_h$ 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$_h$
Object Name	Following Error Time Out
Object Code	VARIABLE
Data type	UNSIGNED16
Persistent	No
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	0064$_h$
Firmware Version	FIR-v1426
Change History	

Description

If the difference between the actual position and the set position is so large that the value of object **6065**$_h$ is exceeded, bit 11 for "Limit exceeded" is set in **6041**$_h$ (status word). 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**$_h$. A fault is generated if the "following error" is created – and reacts correspondingly to it (**6041**$_h$ 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$_h$
Object Name	Position Window
Object Code	VARIABLE
Data type	UNSIGNED32
Persistent	No

Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	$0000000A_h$
Firmware Version	FIR-v1426
Change History	

6068h Position Window Time

Function

For this time period in milliseconds, the actual position must be within the "Position Window" (**6067**) for the target position to be considered to have been reached.

Object description

Index	6068_h
Object Name	Position Window Time
Object Code	VARIABLE
Data type	UNSIGNED16
Persistent	No
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	0064_h
Firmware Version	FIR-v1426
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	$606B_h$
Object Name	Velocity Demand Value
Object Code	VARIABLE
Data type	INTEGER32
Persistent	No
Access	Read only
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h
Firmware Version	FIR-v1426
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	$606C_h$
Object Name	Velocity Actual Value
Object Code	VARIABLE
Data type	INTEGER32
Persistent	No
Access	Read only
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h
Firmware Version	FIR-v1426
Change History	

606Dh Velocity Window

Function

Speed window for the Profile Velocity Mode.

Object description

Index	$606D_h$
Object Name	Velocity Window
Object Code	VARIABLE
Data type	UNSIGNED16
Persistent	No
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	0000_h
Firmware Version	FIR-v1426
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_h**) to be set to "1".

606Eh Velocity Window Time

Function

Time window for the Profile Velocity Mode.

Object description

Index	$606E_h$
Object Name	Velocity Window Time
Object Code	VARIABLE
Data type	UNSIGNED16
Persistent	No
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	0000_h
Firmware Version	FIR-v1426
Change History	

Description

This object specifies how long the actual speed and the set speed must be near each other in magnitude (see **606D**$_h$) for bit 10 "Target reached" in status word (**6041**$_h$) to be set to "1".

6071h Target Torque

Function

This object contains the target torque for the Profile Torque Mode.

Object description

Index	6071_h
Object Name	Target Torque
Object Code	VARIABLE
Data type	INTEGER16
Persistent	No
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	0000_h
Firmware Version	FIR-v1426
Change History	

6072h Max Torque

Function

The object describes the maximum torque.

Object description

Index	6072_h
Object Name	Max Torque
Object Code	VARIABLE
Data type	UNSIGNED16
Persistent	yes, category: user
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	0000_h
Firmware Version	FIR-v1426
Change History	

6074h Torque Demand

Function

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

Object description

Index	6074_h
Object Name	Torque Demand
Object Code	VARIABLE
Data type	INTEGER16
Persistent	No
Access	Read only
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	0000_h
Firmware Version	FIR-v1426
Change History	

607Ah Target Position

Function

This object specifies the target position.

Object description

Index	$607A_h$
Object Name	Target Position
Object Code	VARIABLE
Data type	INTEGER32
Persistent	No
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	$00000FA0_h$

Firmware Version	FIR-v1426
Change History	

607Bh Position Range Limit

Function

Contains the minimum and maximum position.

Object description

Index	$607B_h$
Object Name	Position Range Limit
Object Code	ARRAY
Data type	INTEGER32
Persistent	yes, category: user
Firmware Version	FIR-v1426
Change History	

Value description

Subindex	00_h
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	02_h

Subindex	01_h
Name	Min Position Range Limit
Data type	INTEGER32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	80000001_h

Subindex	02_h
Name	Max Position Range Limit
Data type	INTEGER32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	$7FFFFFFE_h$

Description

If this range is exceeded or undercut, an overflow occurs. Limit values for the target position can be set in object **$607D_h$** ("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 of the machine. This object is computed in the same unit used for calculation for object **607A$_h$** (see " **User-defined units**").

Object description

Index	607C$_h$
Object Name	Home Offset
Object Code	VARIABLE
Data type	INTEGER32
Persistent	yes, category: user
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000$_h$
Firmware Version	FIR-v1426
Change History	

607Dh Software Position Limit

Function

Limit values for the target position.

Object description

Index	607D$_h$
Object Name	Software Position Limit
Object Code	ARRAY
Data type	INTEGER32
Persistent	yes, category: user
Firmware Version	FIR-v1426
Change History	

Value description

Subindex	00$_h$
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	02$_h$

Subindex	01$_h$
Name	Min Position Limit
Data type	INTEGER32
Access	Read/write

PDO Mapping	RX - PDO
Admissible Values	
Specified Value	80000000_h

Subindex	02_h
Name	Max Position Limit
Data type	INTEGER32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	$7FFFFFFF_h$

Description

The target position must lie within the limits set here. Before the check, the home offset (**$607C_h$**) 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

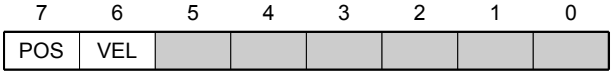
This object can be used to reverse the direction of rotation.

Object description

Index	$607E_h$
Object Name	Polarity
Object Code	VARIABLE
Data type	UNSIGNED8
Persistent	yes, category: user
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00_h
Firmware Version	FIR-v1426
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_h
Object Name	Profile Velocity
Object Code	VARIABLE
Data type	UNSIGNED32
Persistent	yes, category: user
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	$000001F4_h$
Firmware Version	FIR-v1426
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_h
Object Name	End Velocity
Object Code	VARIABLE
Data type	UNSIGNED32
Persistent	yes, category: user
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h
Firmware Version	FIR-v1426

Change History	

6083h Profile Acceleration

Function

Specifies the maximum acceleration in revolutions/s ^{2}.

Object description

Index	6083_h
Object Name	Profile Acceleration
Object Code	VARIABLE
Data type	UNSIGNED32
Persistent	yes, category: user
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	$000001F4_h$
Firmware Version	FIR-v1426
Change History	

6084h Profile Deceleration

Function

Specifies the maximum deceleration in revolutions/s ^{2}.

Object description

Index	6084_h
Object Name	Profile Deceleration
Object Code	VARIABLE
Data type	UNSIGNED32
Persistent	yes, category: user
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	$000001F4_h$
Firmware Version	FIR-v1426
Change History	

6085h Quick Stop Deceleration

Function

Specifies the maximum Quick Stop deceleration in revolutions/s ^{2}.

Object description

Index	6085_h
Object Name	Quick Stop Deceleration

Object Code	VARIABLE
Data type	UNSIGNED32
Persistent	yes, category: user
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00001388_h
Firmware Version	FIR-v1426
Change History	

6086h Motion Profile Type

Function

Specifies the ramp type.

Object description

Index	6086_h
Object Name	Motion Profile Type
Object Code	VARIABLE
Data type	INTEGER16
Persistent	yes, category: user
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	0000_h
Firmware Version	FIR-v1426
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_h
Object Name	Torque Slope
Object Code	VARIABLE
Data type	UNSIGNED32
Persistent	yes, category: user
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	

Specified Value	00000000_h
Firmware Version	FIR-v1426
Change History	

608Fh Position Encoder Resolution

Function

Encoder cycles per revolution.

Object description

Index	$608F_h$
Object Name	Position Encoder Resolution
Object Code	ARRAY
Data type	UNSIGNED32
Persistent	yes, category: user
Firmware Version	FIR-v1426
Change History	

Value description

Subindex	00_h
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	02_h

Subindex	01_h
Name	Encoder Increments
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	$000007D0_h$

Subindex	02_h
Name	Motor Revolutions
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000001_h

Description

Position encoder resolution = encoder cycles ($608F_h$:01_h)/motor revolutions ($608F_h$:02_h)

6091h Gear Ratio

Function

Number of motor revolutions per revolution of the output axis.

Object description

Index	6091_h
Object Name	Gear Ratio
Object Code	ARRAY
Data type	UNSIGNED32
Persistent	yes, category: user
Firmware Version	FIR-v1426
Change History	

Value description

Subindex	00_h
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	02_h

Subindex	01_h
Name	Motor Revolutions
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000001_h

Subindex	02_h
Name	Shaft Revolutions
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000001_h

Description

Gear ratio = motor revolutions (6091_h:01_h) / shaft revolutions (6091_h:02_h)

6092h Feed Constant

Function

Feed per revolution for a linear drive.

Object description

Index	6092_h
Object Name	Feed Constant
Object Code	ARRAY
Data type	UNSIGNED32
Persistent	yes, category: user
Firmware Version	FIR-v1426
Change History	

Value description

Subindex	00_h
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	02_h

Subindex	01_h
Name	Feed
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000001_h

Subindex	02_h
Name	Shaft Revolutions
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000001_h

Description

Feed Constant = Feed (6092_h:01_h)/Shaft Revolutions (6092_h:02_h)

6098h Homing Method

Function

This object selects the homing mode.

Object description

Index	6098_h
Object Name	Homing Method
Object Code	VARIABLE
Data type	INTEGER8

Persistent	yes, category: user
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	23_h
Firmware Version	FIR-v1426
Change History	

6099h Homing Speed

Function

Specifies the speeds for the Homing Mode (**6098_h**) in revolutions/s.

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_h
Object Name	Homing Speed
Object Code	ARRAY
Data type	UNSIGNED32
Persistent	yes, category: user
Firmware Version	FIR-v1426
Change History	

Value description

Subindex	00_h
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	02_h

Subindex	01_h
Name	Speed During Search For Switch
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000032_h

Subindex	02_h
Name	Speed During Search For Zero
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	

Specified Value	00000001_h

Description

This value is computed with the numerator in object **2061_h** and the denominator in object **2062_h**.

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 ^{2}.

Object description

Index	$609A_h$
Object Name	Homing Acceleration
Object Code	VARIABLE
Data type	UNSIGNED32
Persistent	yes, category: user
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	$000001F4_h$
Firmware Version	FIR-v1426
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	$60A4_h$
Object Name	Profile Jerk
Object Code	ARRAY
Data type	UNSIGNED32

Persistent	yes, category: user
Firmware Version	FIR-v1426
Change History	

Value description

Subindex	00_h
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	04_h

Subindex	01_h
Name	Begin Acceleration Jerk
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	$000003E8_h$

Subindex	02_h
Name	End Acceleration Jerk
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	$000003E8_h$

Subindex	03_h
Name	Begin Deceleration Jerk
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	$000003E8_h$

Subindex	04_h
Name	End Deceleration Jerk
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	$000003E8_h$

60C2h Interpolation Time Period

Function

This object contains the interpolation time in milliseconds squared.

Object description

Index	$60C2_h$
Object Name	Interpolation Time Period
Object Code	RECORD
Data type	INTERPOLATION_TIME_PERIOD
Persistent	yes, category: user
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	
Firmware Version	FIR-v1426
Change History	

Value description

Subindex	00_h
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	02_h

Subindex	01_h
Name	Interpolation Time Period Value
Data type	UNSIGNED8
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	01_h

Subindex	02_h
Name	Interpolation Time Index
Data type	INTEGER8
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	FD_h

Description

The subindices have the following functions:

- 01_h: Interpolation time, units: Specifies the interpolation time; at this time, only times that are powers of two are supported, such as 1, 2, 4, 8, 16, etc.

- 02_h: Interpolation time, index: must hold the value of -3 (corresponds to the time basis in milliseconds).

60C5h Max Acceleration

Function

This object contains the maximum admissible acceleration ramp.

For the braking ramp: see object **$60C6_h$** "Max Deceleration".

Object description

Index	$60C5_h$
Object Name	Max Acceleration
Object Code	VARIABLE
Data type	UNSIGNED32
Persistent	yes, category: user
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00001388_h
Firmware Version	FIR-v1426
Change History	

60C6h Max Deceleration

Function

This object contains the maximum admissible braking ramp.

For the acceleration ramp: See object **$60C5_h$** "Max Acceleration".

Object description

Index	$60C6_h$
Object Name	Max Deceleration
Object Code	VARIABLE
Data type	UNSIGNED32
Persistent	yes, category: user
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00001388_h
Firmware Version	FIR-v1426
Change History	

60F4h Following Error Actual Value

Function

This object contains the current following error.

Object description

Index	$60F4_h$
Object Name	Following Error Actual Value
Object Code	VARIABLE
Data type	INTEGER32
Persistent	No
Access	Read only
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h
Firmware Version	FIR-v1426
Change History	

Description

This object is computed with user-defined units (see " **User-defined units**").

60FDh Digital Inputs

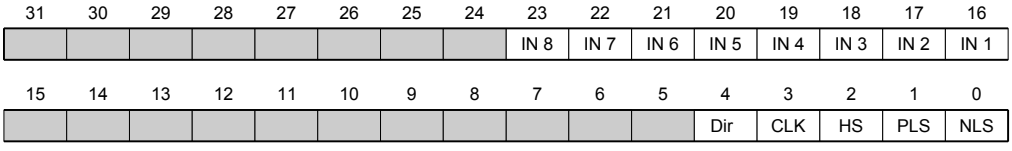
Function

The digital inputs of the motor can be read with this object.

Object description

Index	$60FD_h$
Object Name	Digital Inputs
Object Code	VARIABLE
Data type	UNSIGNED32
Persistent	No
Access	Read only
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h
Firmware Version	FIR-v1426
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	$60FE_h$
Object Name	Digital Outputs
Object Code	ARRAY
Data type	UNSIGNED32
Persistent	No
Firmware Version	FIR-v1426
Change History	

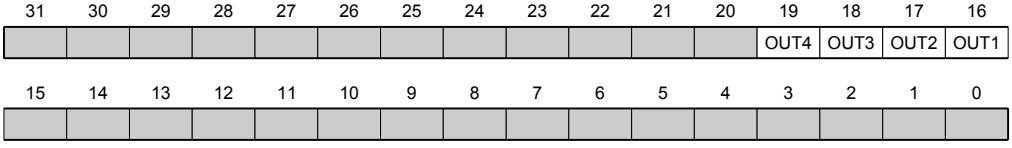
Value description

Subindex	00_h
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	01_h

Subindex	01_h
Name	Digital Outputs #1
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Description

The entries in object **3250**$_h$, subindex 02_h to 05_h 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	$60FF_h$
Object Name	Target Velocity
Object Code	VARIABLE
Data type	INTEGER32
Persistent	No
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h
Firmware Version	FIR-v1426
Change History	

6502h Supported Drive Modes

Function

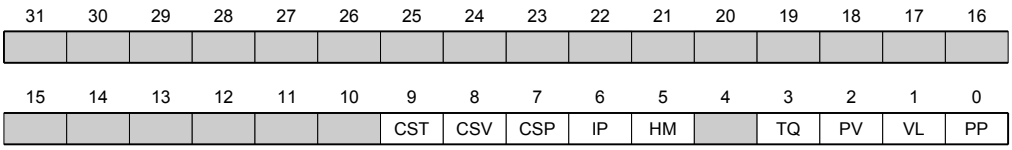
The object specifies the supported drive modes.

Object description

Index	6502_h
Object Name	Supported Drive Modes
Object Code	VARIABLE
Data type	UNSIGNED32
Persistent	No
Access	Read only
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	$000000AF_h$
Firmware Version	FIR-v1426
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 web address of the manufacturer as a string.

Object description

Index	6505_h
Object Name	Http Drive Catalogue Address
Object Code	VARIABLE
Data type	VISIBLE_STRING
Persistent	No
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	http://www.nanotec.de
Firmware Version	FIR-v1426
Change History	

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 trems.

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) 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.