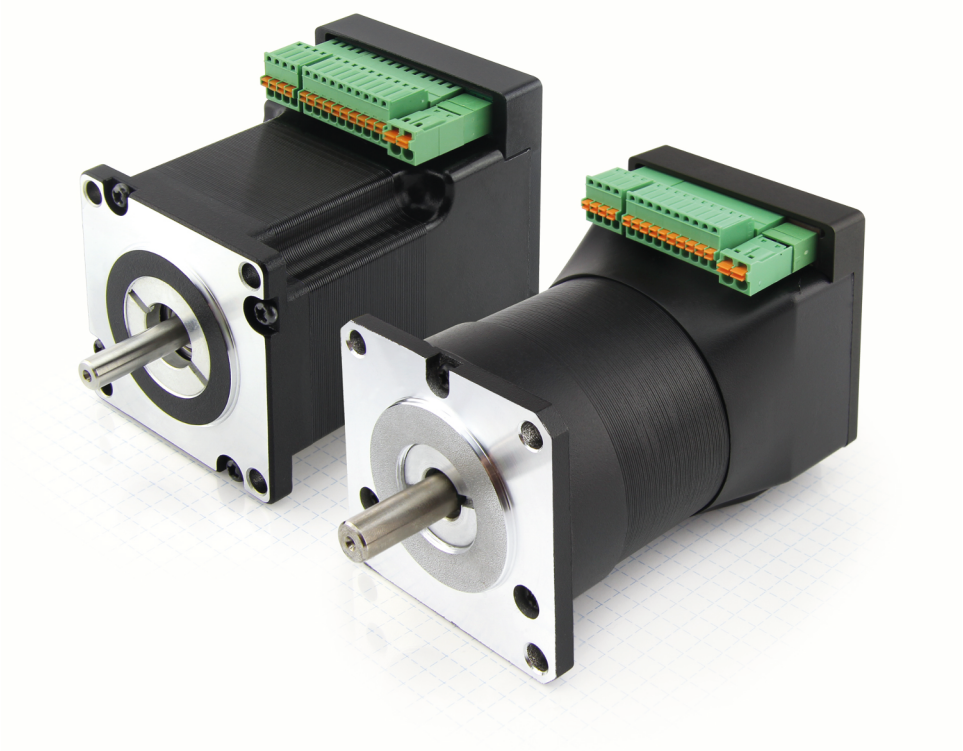
Manual PD4-C

Fieldbus: USB

For use with the following devices:

- PD4-C5918M4204-E-01
- PD4-C5918L4204-E-01
- PD4-C6018L4204-E-01
- PD4-CB59M024035-E-01



Valid with firmware version FIR-v1626
and since hardware version W005

Manual Version: 1.4.1

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

Translation of original manual

2 Release notes

Version Manual	Version Frimware	Date	Changes
1.0.0	FIR-v1403	03.03.2014	First release
1.0.3	FIR-v1419	12.05.2014	Minor corrections, field "Specified Value" in object dictionary description now used

1.1.0	FIR-v1426	23.07.2014	• Added chapter "**Save Objects**", added "Persistent" to the object description
			• The following objects has been moved
			  • "Read Analogue 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

1.1.7	FIR-v1436	10.09.2014	Corrections
1.1.15	FIR-v1446	18.11.2014	• Corrections • The object "Mode of modulo operation" at 2070_h has been replaced with object "Positioning option code" at **60F2**$_h$
1.2.0	FIR-v1504	11.03.2015	New chapter: • **Clock/direction mode** • **Analogue Mode**
1.2.1	FIR-v1512	24.04.2015	• Corrections • New chapter **Input Routing**
1.3.0	FIR-v1540	02.10.2015	• Corrections • New chapter **Overtemperature protection** • New chapter **Output Routing** • New section **Possible move command combinations** • Connection data added • Switching limits for digital inputs added
1.4.0	FIR-v1614	08.04.2016	Corrections
1.4.1	FIR-v1626	22.07.2016	Additions and corrections

3 Safety instructions and warnings

3.1 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.

Nanotec produces component parts that are deployed in a variety of industrial applications. The selection and application of Nanotec products is the responsibility of the system constructor or end user (see **Personnel qualifications**). Nanotec accepts no responsibility for the integration of the products in the end system.

Under no circumstances may a Nanotec product be integrated as the sole safety control in a product or construction. All positioning controls without exception must be designed so that errors are detected dynamically and fail- safe under all circumstances. All products which contain a component part manufactured by Nanotec must show appropriate warning notices and instructions for a safe application and safe operation on delivery to the end user. All warning notices provided by Nanotec must be immediately passed on to the end user.

Nanotec only accepts an express warranty for the quality of its own products in agreement with the standards and specifications as they appear in the Nanotec manual. ALL OTHER IMPLICIT and EXPLICIT WARRANTIES ARE EXCLUDED. Nanotec assumes no liability for injuries, product damage, loss or claims that arise through incorrect application of the products.

Nanotec does not take over any responsibilities for results caused by any unauthorized product changes. Nanotec does not take over any liabilities for damages or disruptions caused by any unauthorized product changes.

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

3.2 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.

3.3 General danger and warning notes

⚠ WARNING

Only use this controller with the - by connectors and power - intended 2-phase stepper and BLDC motors. Otherwise there is a danger of fire or malfunction.

Do not use the product in areas where they are exposed to water, corrosive gases, flammable or explosive gases or fuel. Otherwise there is a danger of electric shock, fire or explosion.

3.4 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.

3.5 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.

4 About this manual

4.1 Introduction

This manual is directed toward programmers intending to program a motor controller using the motor controller from Nanotec®.

4.2 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 sub-index: <Index>:<Sub-index>

Both the index and sub-index are in hexadecimal notation. Sub-index 0 is in force when no sub-index is noted.

Example: Sub-index 5 of object 1003_h is addressed with "1003_h:05_h", sub-index 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**.

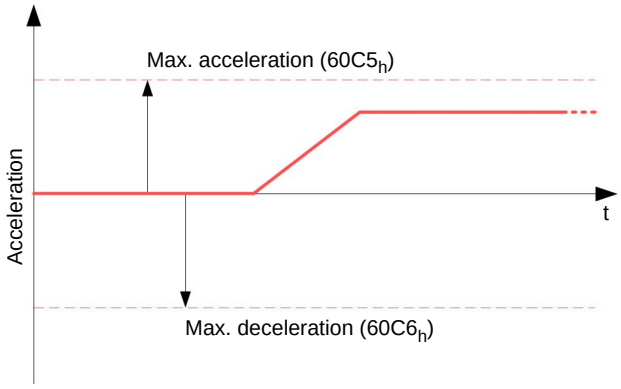
4.3 Bits

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	$\triangleq 55_{hex} \triangleq 85_{dec}$

4.4 Counting direction (arrows)

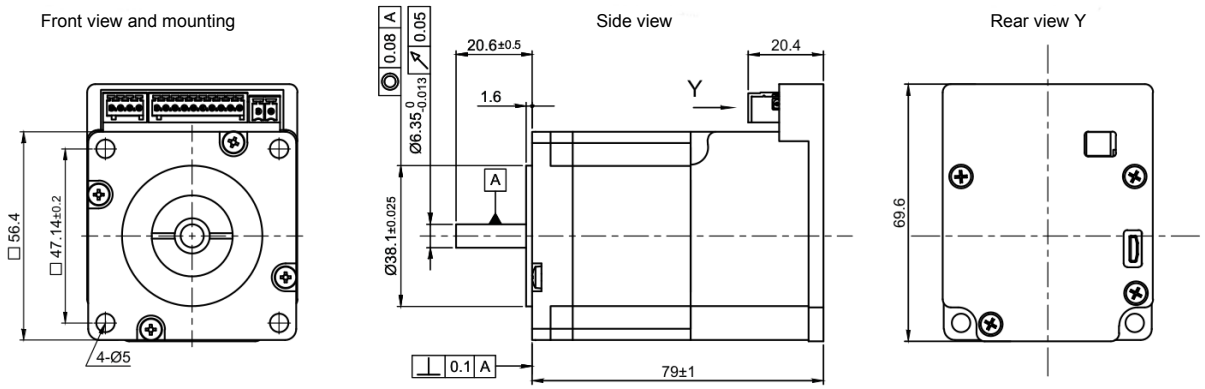
In drawings, the counting direction is always in the direction of the arrow. The objects $60C5_h$ and $60C6_h$ shown in the following figure are both positive.



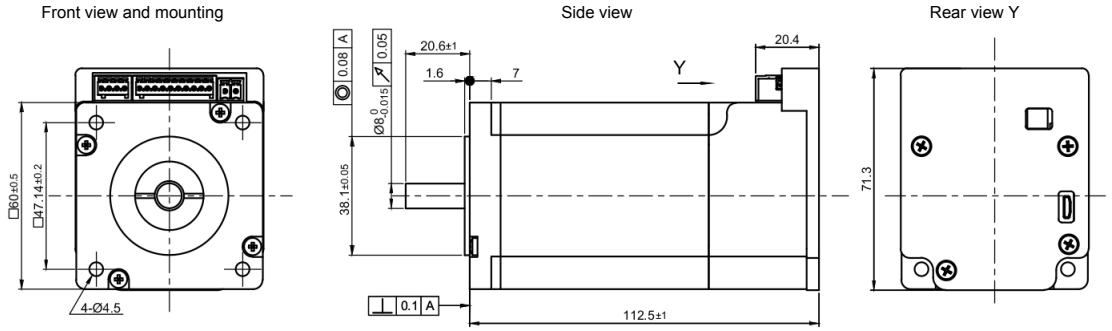
5 Technical data and pin configuration

5.1 Dimensioned drawings

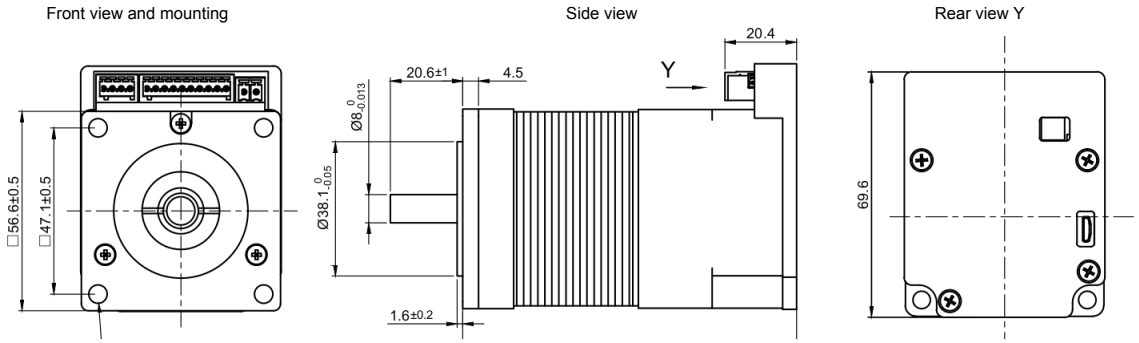
5.1.1 PD4-C5918M4204-E-01



5.1.2 PD4-C6018L4204-E-01



5.1.3 PD4-CB59M024035-E-01



5.2 Electrical properties

5.2.1 Technical data of motor

	PD4-C	PD4-CB
Type	High-pole DC servo (stepper motor)	Low-pole DC servo (BLDC)

	PD4-C	PD4-CB
Operating voltage	12 V to 48 V	12 V to 24 V
Phase current eff.	4.2 A	8 A
RMS for 1s	Max. 6.3 A	Max. 20 A

5.2.2 Technical data of I/O

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

Protection circuit	• Over and under voltage protection
	• Over temperature: protective circuit at temperature > 75° Celsius
	• 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. The interrupting rating of the fuse depends on the application and has to be
	  • greater than the maximum controllers current consumption
	  • and smaller than the maximum current of the voltage supply.
	  In case the fuse being close to the current consumption of the controller, the fuse characteristic should be slow-blow.

5.2.3 Ratings

Type	Holding torque Ncm	Nom./ Peak Torque Ncm	Nominal Speed (rpm)	Length mm	Weight kg
PD4-C5918M4204-E-01/-08	110	n/a	n/a	81	0.8
PD4-C6018L4204-E-01/-08	350	n/a	n/a	111	1.5
PD4-CB59M024035-E-01/-08	n/a	37 / 110	3500	89	0.9

5.3 Overtemperature protection

At a temperature of approx. 80 °C on the board (corresponds to 65-72 °C outside on the rear cover), the power drive of the controller is switched off and the error bit is set (see object **1001**$_h$ and **1003**$_h$). After the controller is cooled and the error is confirmed (see **table for the controlword**, "Fault reset"), it operates normally again.

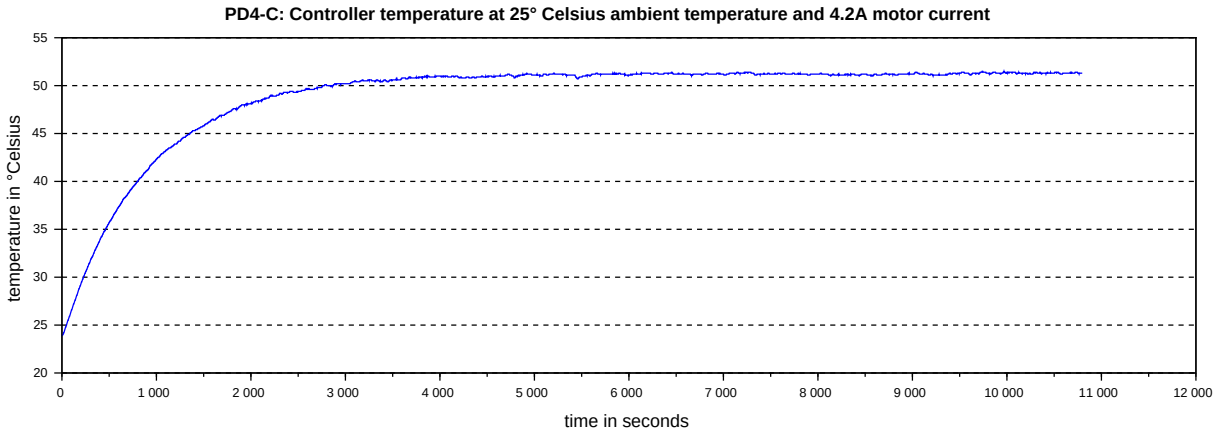
The following temperature test results illustrate the temperature behavior of this controller. However, the specific temperature behavior depends not only on the motor but also largely on the flanging and the heat transition at the flange, as well as on the convection in the machine. Therefore, we

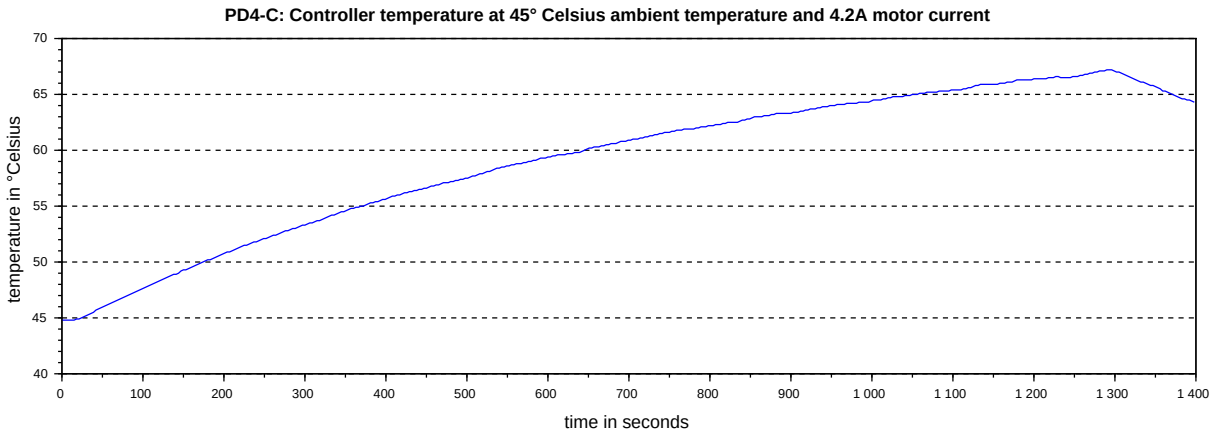
recommend always performing a long-term test in a realistic environment for applications with problematic levels of current and ambient temperature.

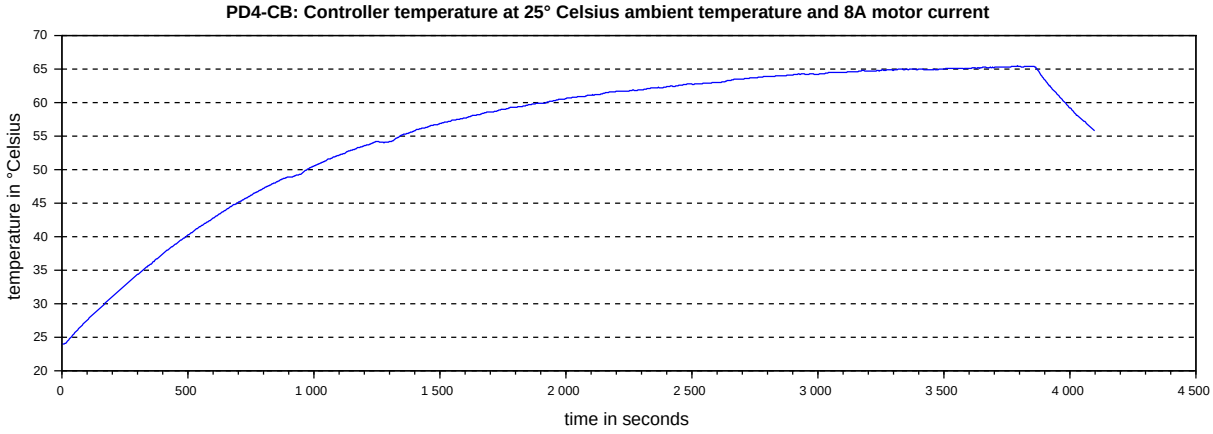
Temperature tests were performed under the following conditions:

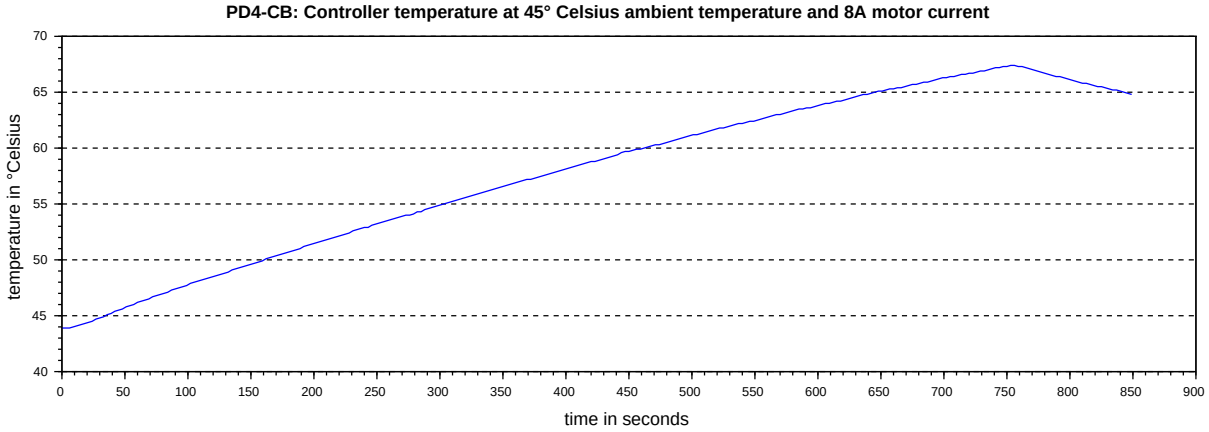
- Operating voltage: 24 V / 48 V DC
- Motor current: nominal current
- Operation mode: Full step speed mode, 30 rpm
- Operating environment: Thermo TEC
- Ambient temperature: 25 or 45 °C
- Measurement point: rear at power transistors, on the outside of the housing

The following graphics show the temperature test results:







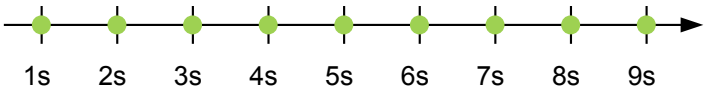


5.4 LED signaling

5.4.1 Operating LED

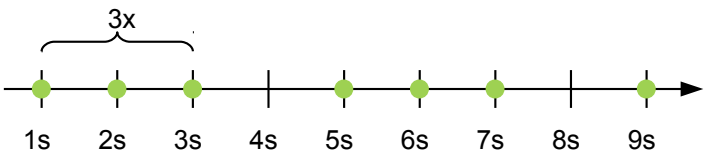
Normal operation

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



Error

Should there be an error, an error number is indicated by the LED, which will switch to a red color. In the following illustration, the error is signaled with the number 3.



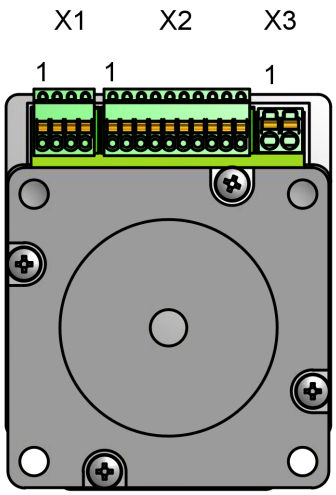
The meaning of the error number is printed in the following table.

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.

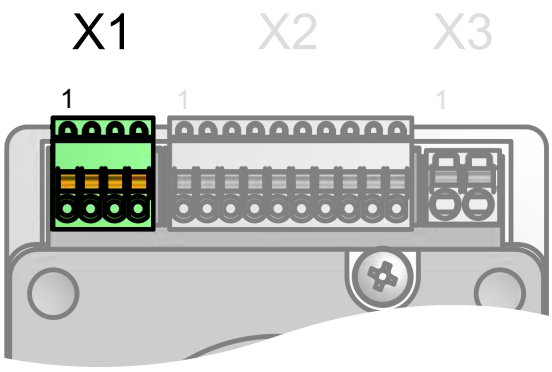
5.5 Pin configuration

5.5.1 Overview



5.5.2 Analogue input (connector X1)

Connections for analogue mode



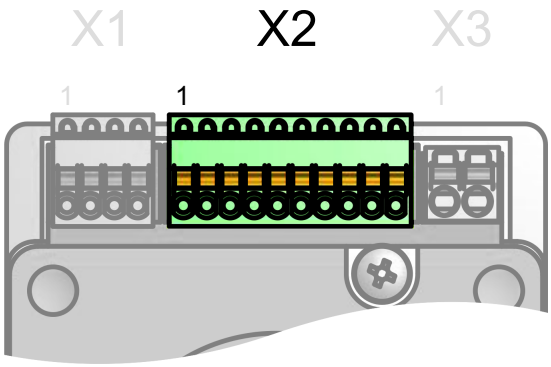
Pin	Function	Remark
1	GND	
2	Analogue input	10 bit, 0 - 10 V
3	Digital output	Open-Drain, max. 24 V/100 mA
4	Voltage output	+12 V, max. 100 mA

Connection data	min	max
Conductor cross section solid min.	0.14 mm^2	0.5 mm^2
Conductor cross section flexible min.	0.14 mm^2	0.5 mm^2
Conductor cross section flexible, with ferrule without plastic sleeve min.	0.25 mm^2	0.5 mm^2
Conductor cross section AWG min.	26	20
Minimum AWG according to UL/CUL	28	20

5.5.3 Clock-direction inputs (connector X2)

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

Note
The inputs for "clock/direction" can get switched between 5 V or 24 V alltogether.



Pin	Function	Remark
1	Digital Input 1	0/+24 V
2	Digital Input 2	0/+24 V
3	Digital Input 3	0/+24 V
4	-Enable	5 V / 24 V signal, all pins together switchable via software with object **3240**$_h$, max. 1 MHz
5	+Enable	5 V / 24 V signal, all pins together switchable via software with object **3240**$_h$, max. 1 MHz
6	-Direction	5 V / 24 V signal, all pins together switchable via software with object **3240**$_h$, max. 1 MHz
7	+Direction	5 V / 24 V signal, all pins together switchable via software with object **3240**$_h$, max. 1 MHz
8	-Clock	5 V / 24 V signal, all pins together switchable via software with object **3240**$_h$, max. 1 MHz
9	+Clock	
10	GND	

For input 1 to 3 the following switching thresholds are defined:

Switching threshold	
On	**Off**
> approx. 16 V	< approx. 4 V

Connection data	min	max
Conductor cross section solid min.	0.14 mm^2	0.5 mm^2
Conductor cross section flexible min.	0.14 mm^2	0.5 mm^2

Connection data	min	max
Conductor cross section flexible, with ferrule without plastic sleeve min.	0.25 mm^2	0.5 mm^2
Conductor cross section AWG min.	26	20
Minimum AWG according to UL/CUL	28	20

5.5.4 Voltage supply (connector X3)

Safety instruction

⚠CAUTION

Danger of electrical overvoltage!

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

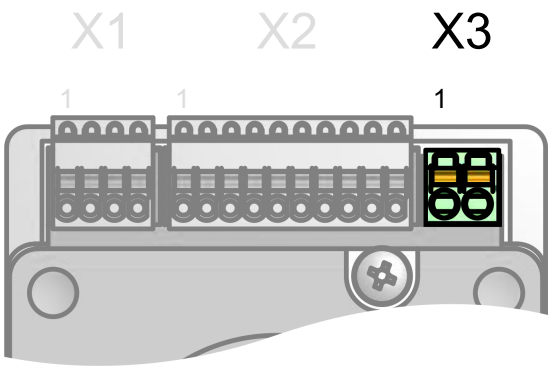
Voltage source

The operating or supply voltage is delivered by a battery, a transformer with rectification and filtering, or better a switch-mode power supply.

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

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

Connections



Pin	Function	Remark
1	+Vcc	• PD4-C: 12-48 V • PD4-CB: 12-24 V
2	GND	

Connection data	min	max
Conductor cross section solid min.	0.2 mm^2	1.5 mm^2

Connection data	min	max
Conductor cross section flexible min.	0.2 mm^2	1.5 mm^2
Conductor cross section flexible, with ferrule without plastic sleeve min.	0.25 mm^2	1.5 mm^2
Conductor cross section flexible, with ferrule with plastic sleeve min.	0.25 mm^2	0.75 mm^2
Conductor cross section AWG min.	24	16
Minimum AWG according to UL/CUL	24	16

Permissible operating voltage

The maximum operating voltage for each motor type is:

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

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

6 Configuration

6.1 General information

The following options exist for configuring the motor controller:

DIP switches

Four DIP switches are fitted on the rear. 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 and "**NanoJ program**"**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

6.2 DIP switches

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

Note
A change of one or more DIP switches will only take effect after a restart of the controller.

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

Switch configurations:

1	2	3	Modus		
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	analogue speed	Direction set by "direction" input	Maximum revolution speed is 1000 rpm

1	2	3	Modus		
On	Off	On	analogue speed	Direction set by "direction" input	Maximum revolution speed is 100 rpm
On	On	Off	analogue speed	Offset 5 V (joystick mode)	Maximum revolution speed is 1000 rpm
On	On	On	analogue speed	Offset 5 V (joystick mode)	Maximum revolution speed is 100 rpm

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

The notation is:

clock/direction

Activates the clock/direction mode, therefore the pins "enable", "clock" and "direction" need to be connected (see chapter " **Analogue input (connector X1)**").

Analogue speed

Activates the analogue mode, therefore the "enable" input (see chapter " **Clock-direction inputs (connector X2)**") and the analogue input (see chapter " **Analogue input (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 " **Clock-direction 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 analogue speed mode this switch determines the maximum revolution speed which is attained at maximum or minum analoge input voltage.

Deactivation of the DIP Switches: If you want to deactivate the DIP switch configuration at all, you need to add the command

```
dd4c=1
```

in the `pd4cfg.txt` file.

6.3 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`

⚠CAUTION

3. `info.bin`
4. `reset.txt`
5. `firmware.bin`

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

Note

- The controller behaves like a mass storage device ("USB flash drive"), no further drivers are necessary.
- 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>
```

6.4 Configuration file

6.4.1 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 start up. 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.

6.4.2 Reading and writing the file

To access to the file:

1. Connect the voltage supply 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`" (in case of a PD4-C the file is named "`pd4ccfg.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.

6.4.3 Structure of configuration file

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 sub-index 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`

Note

- The number of digits may not be changed. The index must have four digits, the sub-index 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
```

6.4.4 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
```

6.5 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 the controller 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:

```
#!/bin/bash
cp <SOURCE PATH>/<OUTPUT>.usr <TARGET PATH>/vmmcode.usr
```

7 Setup and commissioning

7.1 Safety instructions

⚠ WARNING

Operation:

- Never touch rotating parts of the motor while the motor is running. Otherwise there is a danger of injury.
- The motor shaft must be disconnected from the machine in order to prevent unpredictable accidents. Otherwise there is a danger of injury.
- Before starting operation with connected machine, change the settings while taking into account the para- meters of the machine. If you start operation without taking the correct settings into account, the machine can run out of control or cause malfunctions.
- Before starting operation with connected machine, make sure that an emergency-stop can be carried out at any time. Otherwise there is a danger of injury.
- Do not touch any of the motors during operation. Otherwise the high temperatures can cause panic reactions.

Maintenance and Inspection:

- Do not attempt to disconnect the motor from the positioning control while the voltage is switched on. There is a danger of electric shock and destruction of the positioning control.
- After switching off the voltage supply wait until the back-up capacitors have discharged. There is a danger of electric shock from the residual voltage.
- Do not disassemble the stepper motor or the positioning control. The components are maintenance-free and opening them will invalidate any claim under guarantee. Otherwise there is a danger of electric shock or injury.
- Do not attempt to change the wiring while the voltage supply is switched on. Otherwise there is a danger of electric shock or injury.

In residential areas, this product may cause high-frequency interference and may require interference suppression measures.

⚠ 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.

7.2 General

Before starting the set up and commissioning read the chapter "**Technical data and pin configuration**" and "**Configuration**".

The following components are required for set up and installation:

- Motor controller PD4-C

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

Corresponding to the mode to be used:

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

7.2.1 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 connector X3 of the motor controller (see "**Voltage supply (connector X3)**").
4. Connect the clock generator to connector X2 (see "**Clock-direction inputs (connector X2)**")

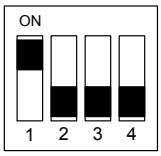
The motor must change the speed of rotation when the clock generator frequency changes.

7.2.2 Analogue mode

CAUTION
Make sure that the voltage at the analogue 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 connector X3 of the motor controller (see "**Voltage supply (connector X3)**").
4. Connect the adjustable voltage source supply to connector X1 pin 2 (see "**Analogue input (connector X1)**")

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

7.2.3 Auto-Setup

To determine some parameter referring the motor and the connected controller an auto-setup is performed.

CAUTION
Prerequisites for performing the auto setup are: • The motor must be load-free. • The motor must not be touched. • The motor must be able to rotate freely in any direction.

CAUTION
Complex computations are performed during auto setup. Often this results in a lack of sufficient computing power to operate the field buses in a timely manner. The buses may be impaired during auto setup.

8 General concepts

8.1 DS402 Power State machine

8.1.1 State machine

CANopen DS402

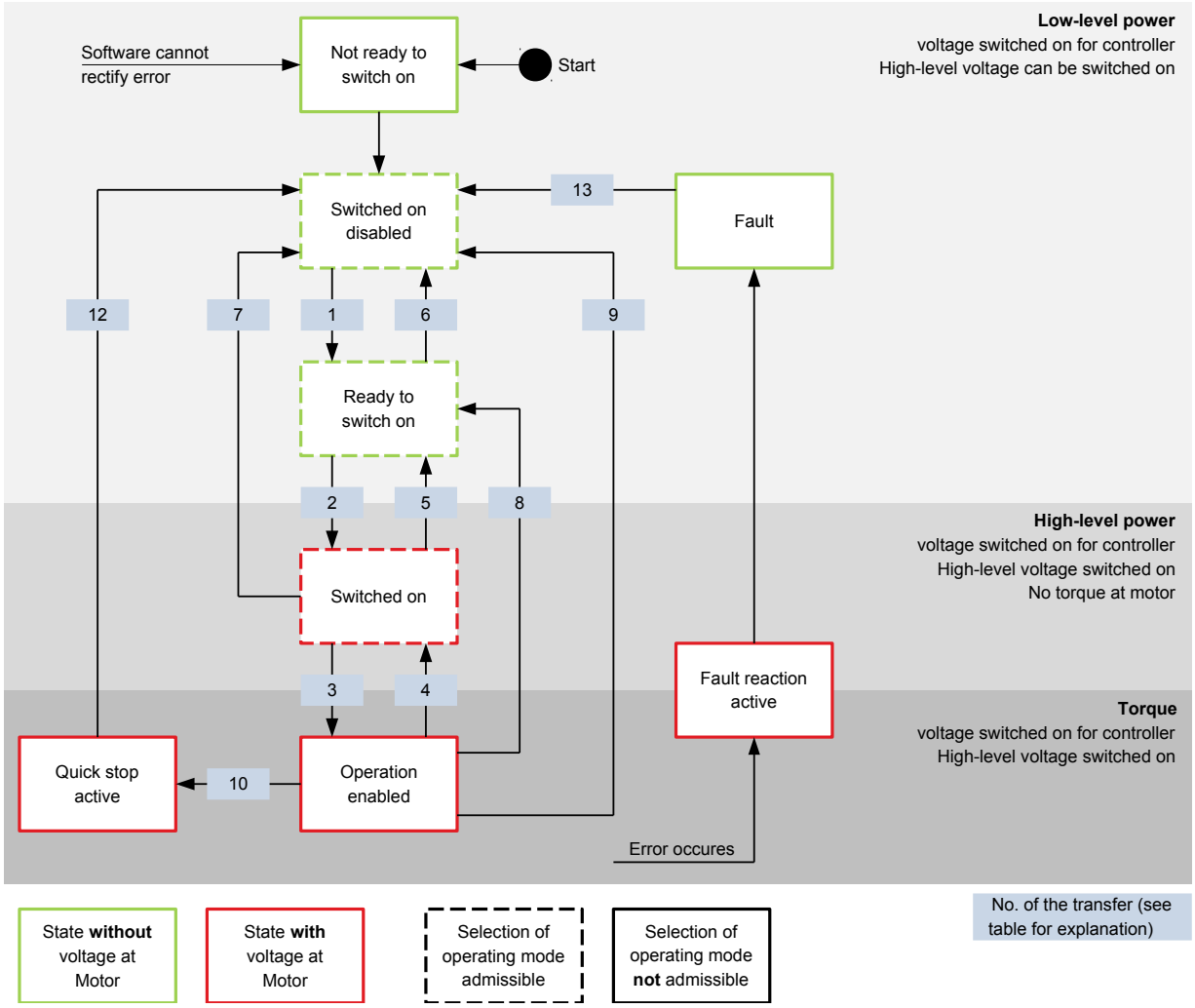
To switch the motor controller to an operational state, a state machine must be run through. This is defined in CANopen standard DS402. State changes are requested in object **6040**$_h$ (control word). The actual state of the state machine can be read out from object **6041**$_h$ (status word).

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
Fault reset	↑	X	X	X	X	13

Holding torque in state "Switched On"

In the state "Switched On" no holding torque is build up by default. If a holding torque is necessary, the value "1" needs to be written in the object 3212_h:01_h.

CAUTION

If the option "Holding torque in state Switched on" is enabled, it is possible that the motor gives a start when switching the operating modes.

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 actually active operation mode is displayed in **6061_h**.

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 includes an encoder error (e.g. due to missing shielding, cable breakage)

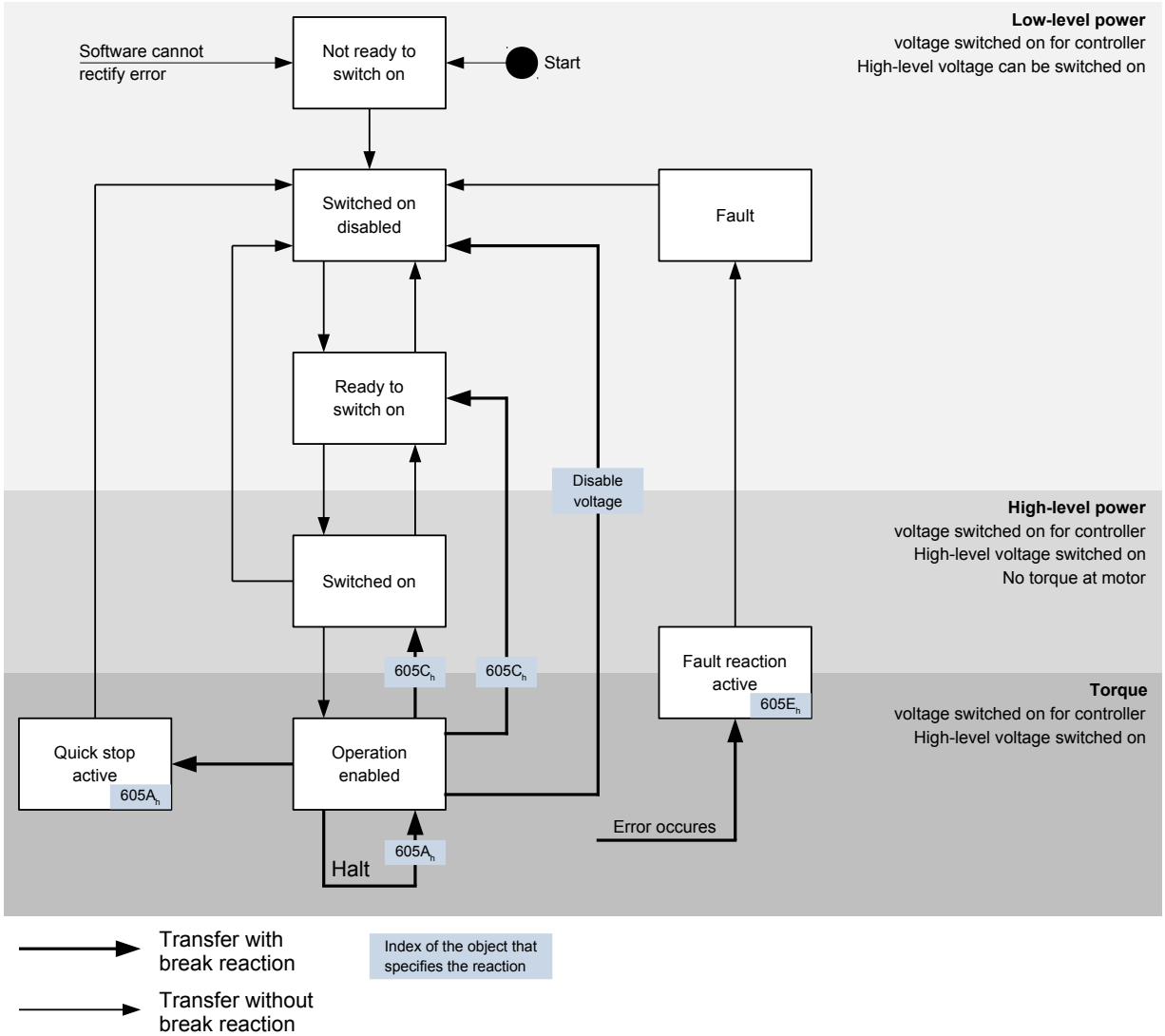
8.1.2 Behavior after the "Operation enabled" state is left

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

Value in object $605A_h$	Description
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

Halt

This bit is available in the following modes:

- **Profile Position**
- **Velocity**
- **Profile Velocity**
- **Profile Torque**

When bit 8 is set in object **6040_h** (control word), the response stored in **$605D_h$** is executed (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

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

Contouring Error

If a contouring error should occur, the motor is braked as stored in object **3700**$_h$.

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

To deactivate the check for the contouring error, set the object **6065**$_h$ to the value "-1" (FFFFFFFF$_h$).

8.2 User-defined units

8.2.1 Overview

Settings

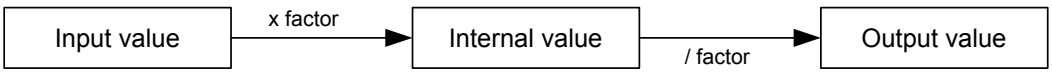
The motor controller supports the possibility of setting user-defined units. In this way, the corresponding parameters can be set and read out directly in degrees, mm, etc.

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.

Factor

There is a factor for the units velocity, acceleration and jerk. It gets either multiplied by the input value or divided by the output value.



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:

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

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:

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

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 unit is rounds per second.

The factor n for the speed is calculated by the factor for the numerator (**2061**$_h$) divided by the factor for the denominator (**2062**$_h$).

$$n_{velocity} = \frac{2061_h}{2062_h}$$

For the input of values it is applied: internal value = $n_{velocity}$ x input value

For the output values it is applied: output value = internal value / $n_{velocity}$

Example

2061$_h$ is set to the value "1", **2062**$_h$ to the value "60". Therefore the user unit is "Rounds per minute" and $n_{velocity}$ = 1/60.

When the **60FF**$_h$ is set to the value "300", the internal value will get set to 300 r/min x 1/60 = 5 r/s.

When the motor turns with a internal speed of 5 r/s the object **606B**$_h$ will get filled with a speed of 5 / 1/60 = 300 r/min.

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 unit for acceleration is rounds per second^2.

The factor n for the acceleration is calculated by the factor for the numerator (**2063**$_h$) divided by the factor for the denominator (**2064**$_h$).

$$n_{Acceleration} = \frac{2063_h}{2064_h}$$

For the input of values it is applied: internal value = $n_{acceleration}$ x input value

Example

2063$_h$ is set to the value "1", **2064**$_h$ to the value "60". Therefor the user unit is "rounds per second per minute" and $n_{acceleration}$ = 1/60.

When the **60C5**$_h$ gets set to the value "600", the internal value will get set to 600 r/(s*min) x 1/60 = 10 r/s^2.

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 **60A4**$_h$:1$_h$ to **60A4**$_h$:4$_h$ can be specified in user units. These objects only affect the Profile Position Mode and Profile Velocity Mode.

The internal unit for jerk is rounds per second^3.

The factor n for the acceleration is calculated by the factor for the numerator (**2065**$_h$) divided by the factor for the denominator (**2066**$_h$).

$$n_{Jerk} = \frac{2065_h}{2066_h}$$

For the input of values it is applied: internal value = n_{jerk} x input value

Example

2063$_h$ is set to the value "1", **2064**$_h$ to the value "60". Therefore the user unit is set to "rounds per seconds^2 per minute" and n_{Jerk} = 1/60.

When the **60A4**$_h$ is set to "500", the internal value will get set to 500 r/(min * s^2) x 1/60 = 8,3 r/s^3.

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:

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

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 Operating modes

9.1 Profile Position

9.1.1 Special feature PD4C USB

Note

Because this motor controller is not fitted with a field bus, the following operating mode is only usable with a NanoJ program.

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

9.1.2 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.

Note

The limit switches - and therefore the tolerance bands - are active in this mode. See chapter "**Tolerance bands of the limit switches**" for further information about the limit switches.

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. The reference position is depending on the Bits 0 and 1 of the object **60F2**$_h$.
- Bit 8 (Halt): At the transition from "1" to "0" the motor accelerates with the given start ramp to the target velocity. At the transition from "0" to "1" the motor decelerates and will come to a halt. The deceleration is depending on the settings in the "Halt Option Code" in object **605D**$_h$.
- 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**".

CAUTION
The bit 9 in the controlword will get ignored, when the velocity will fell below in the target set point. In this case the controller would need to back up and take a run-up to reach the target.

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 target position. 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.

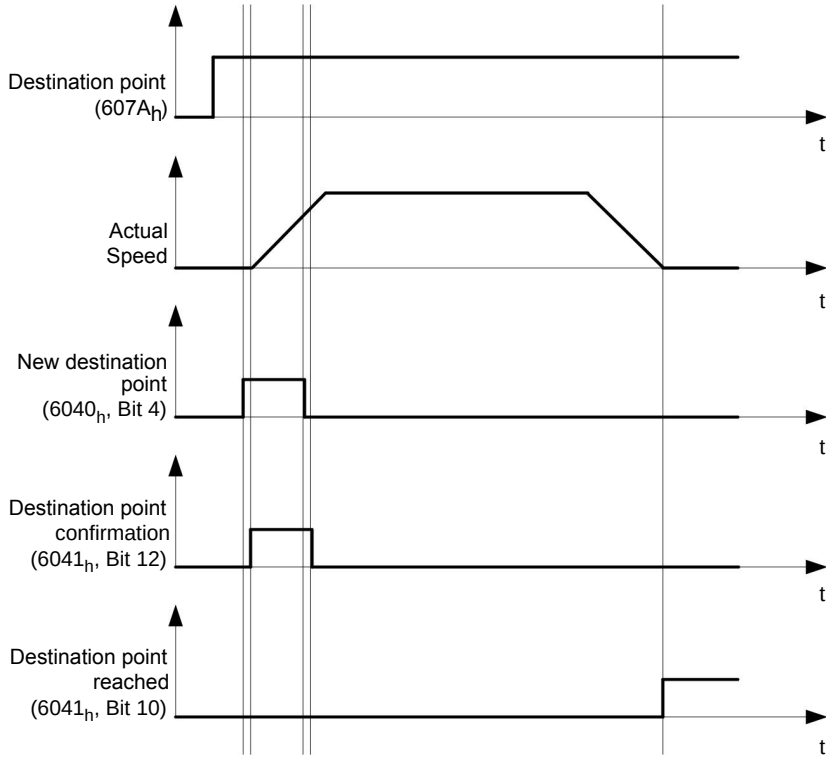
  The bit is not set if one of the following conditions occurs:

  - 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)).

9.1.3 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.



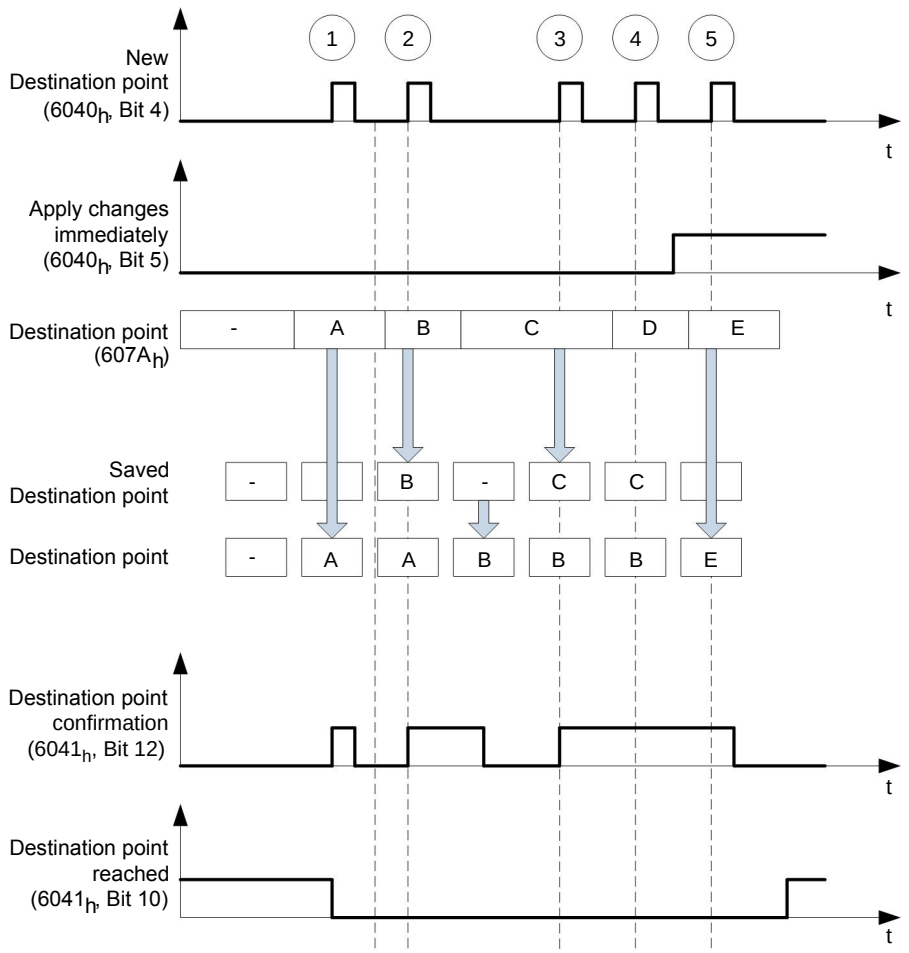
The controller is able to release bit 4 in the object **6040**$_h$ (Controlword) autonomously. This is controlled with bits 4 and 5 of the object **60F2**$_h$.

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).

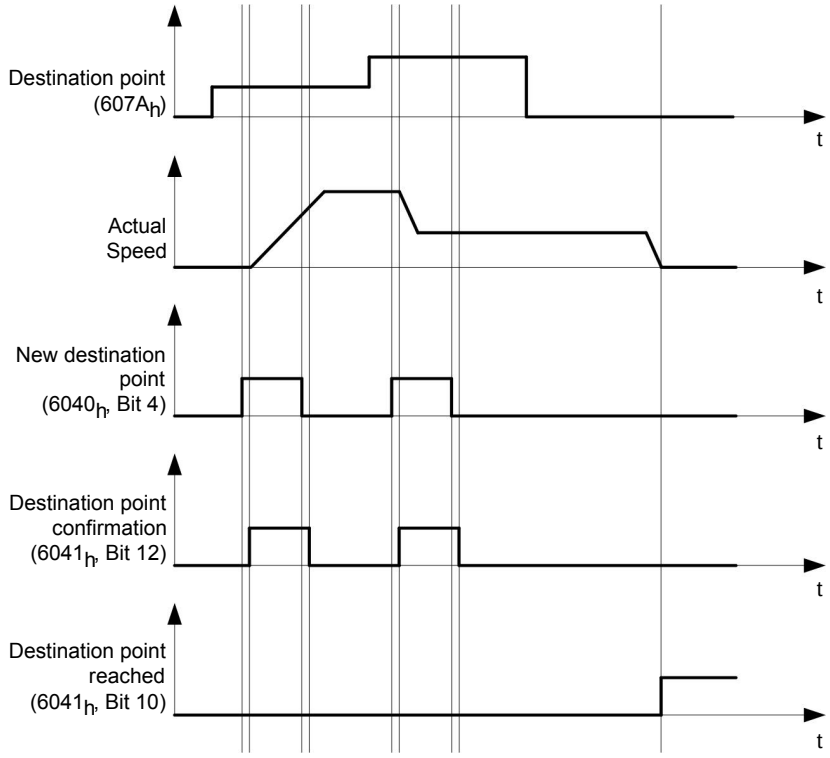
If the buffer is already full, a new time is ignored (time 4). If bit 5 in object **6040**$_h$ (control word, bit: "Change Set-Point Immediately") is set, the motor controller operates without the buffer and new move commands are implemented directly (time 5).

Times



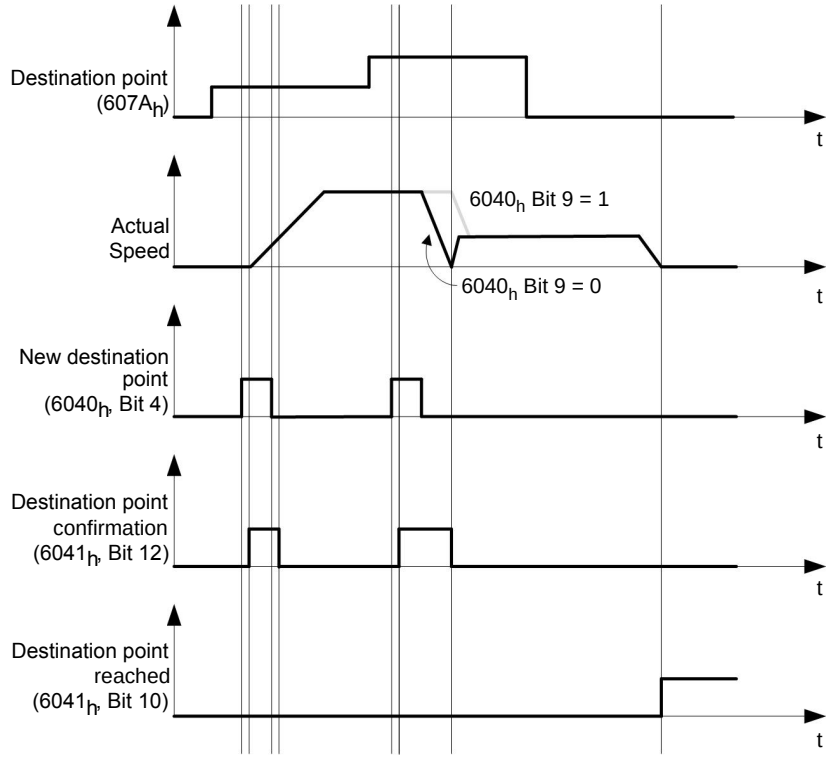
Transition procedure for second target position

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.

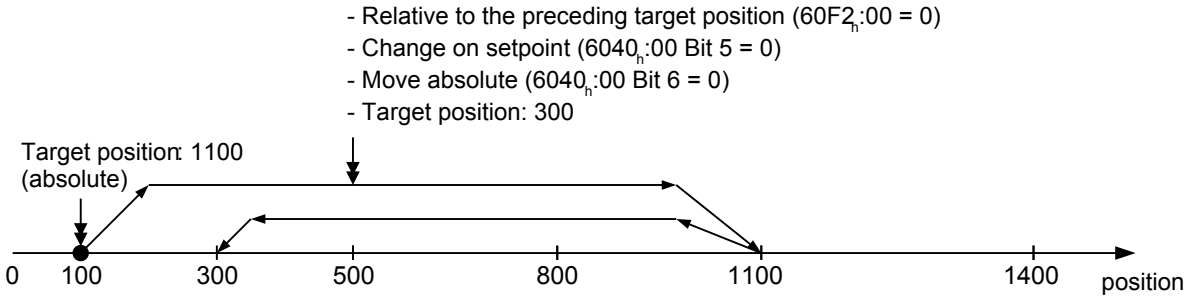


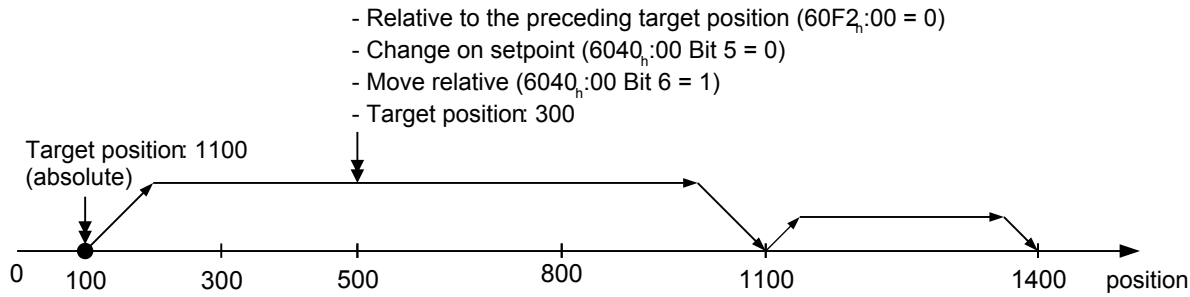
Possible move command combinations

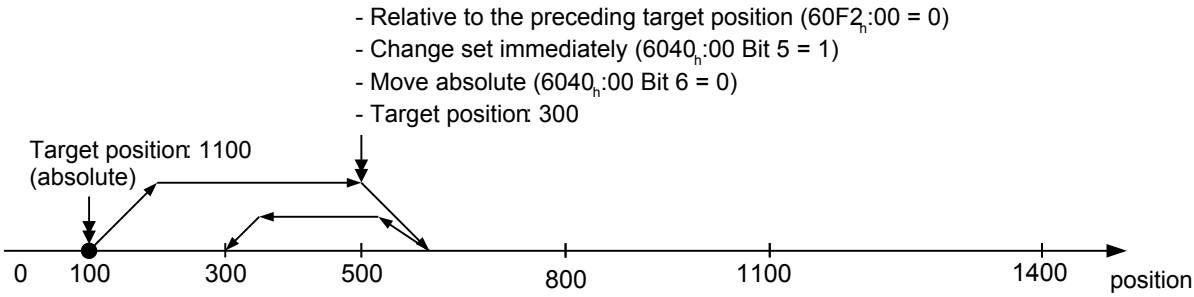
In this chapter combinations of move commands are listened and pictured in order to get a better overview of the move commands.

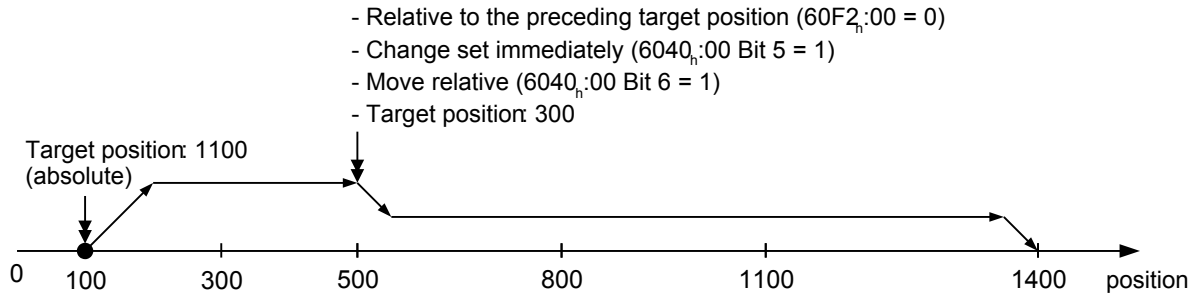
The following images assume:

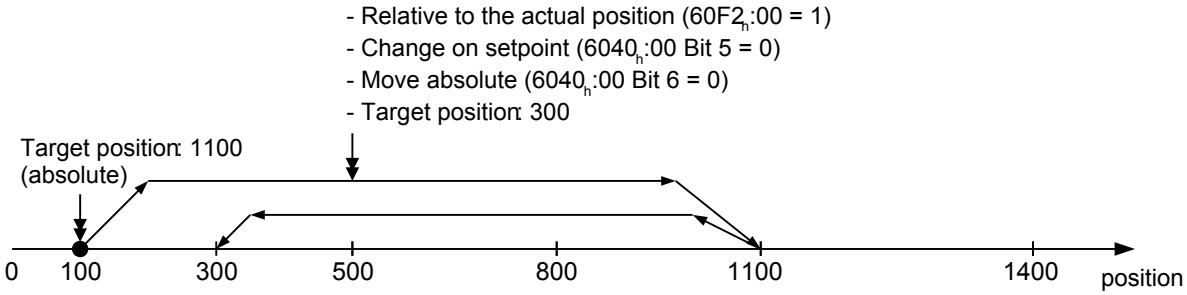
- A double arrow marks a new positioning command.
- The first command from start is always an absolute positioning move to the position 1100.
- The second move is done with a slower speed in order to get a clearly represented graph.

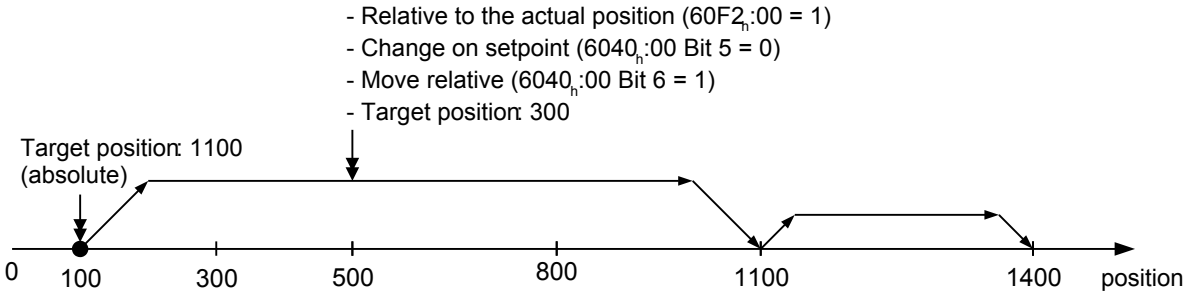


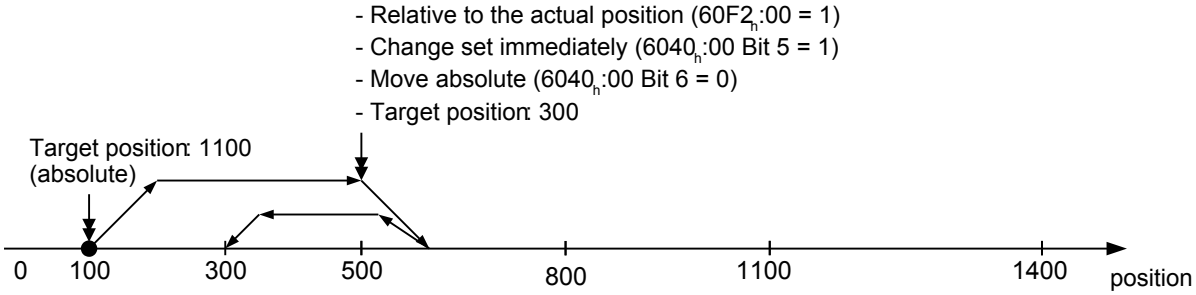


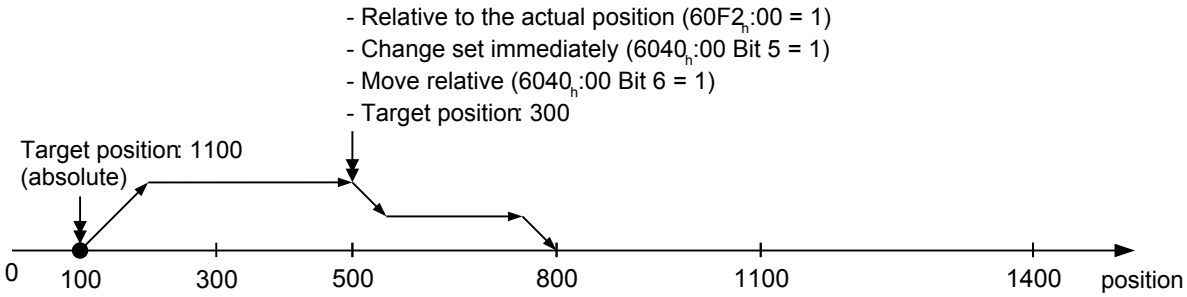












9.1.4 Marginal conditions for a positioning run

Object entries

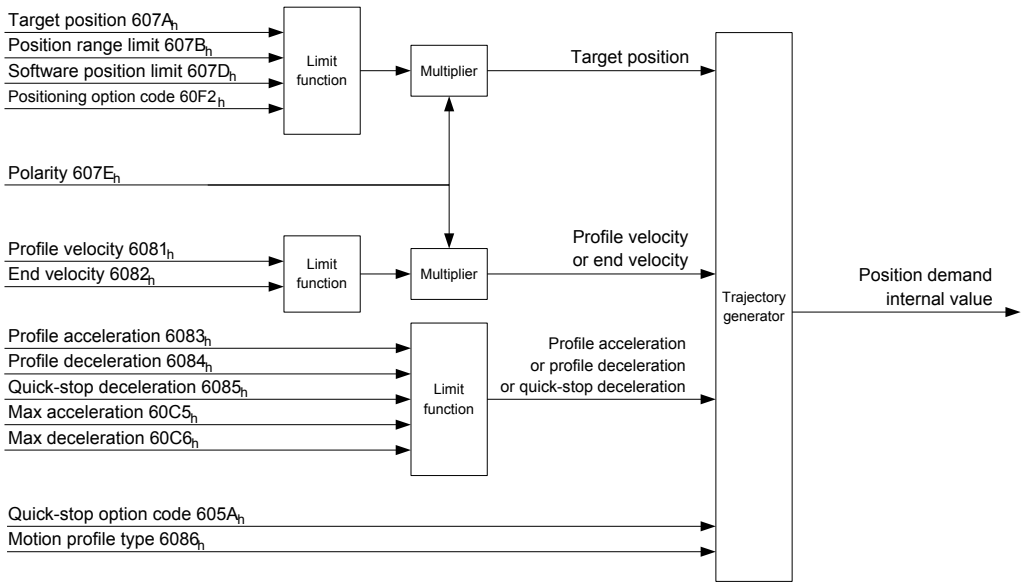
The marginal conditions for the position to which the run is made can be set in the following entries of the object directory:

- **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**")

- **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), sub-index 01$_h$ to 04$_h$: Objects for describing the limit values for the jerk. This value will be clipped by the "Real Jerk Limit" (see **2067**$_h$ for further information).
- **2067**$_h$ (Jerk Limit (internal)): object for the jerk limit.

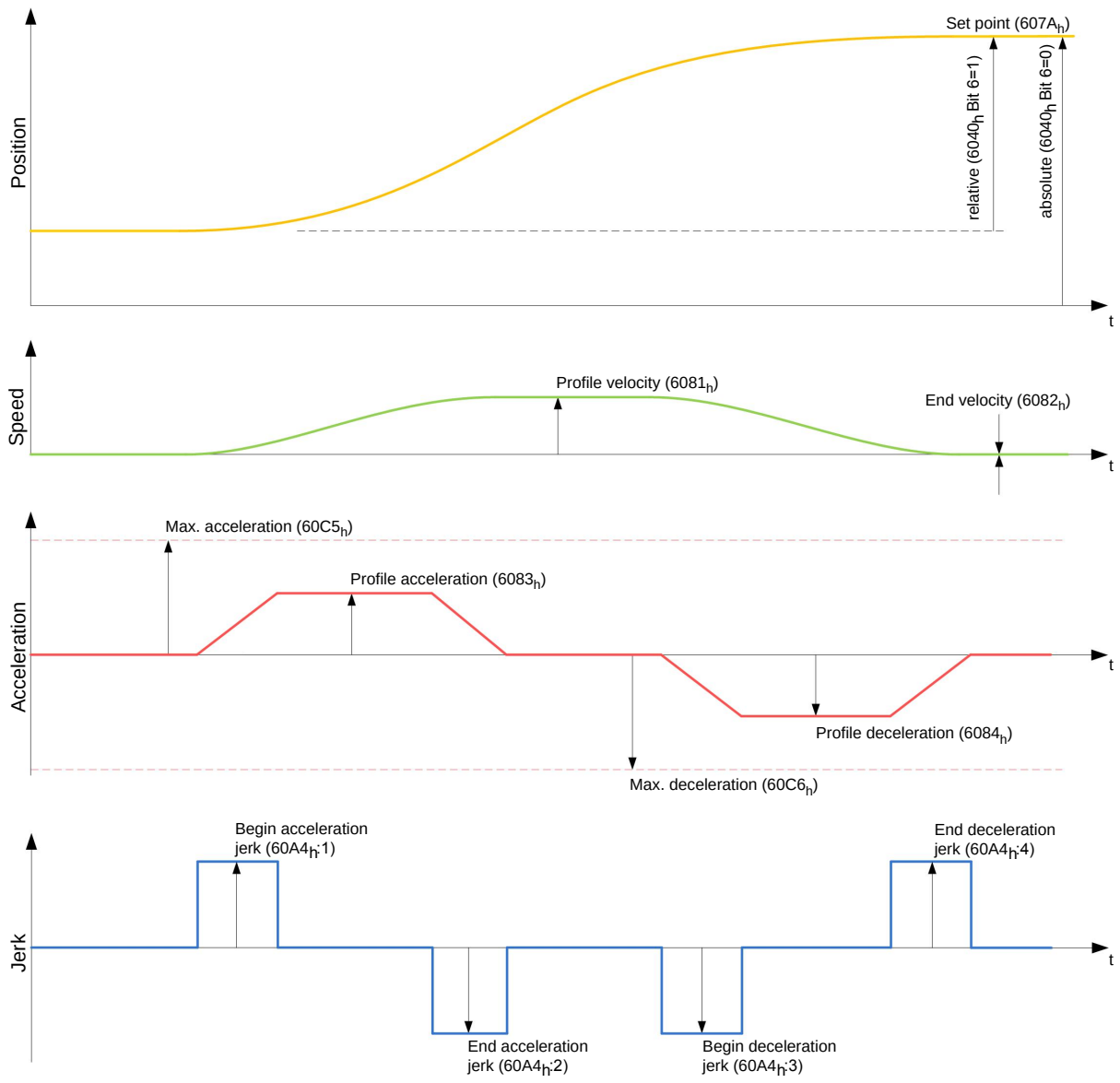
Objects for the positioning run

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



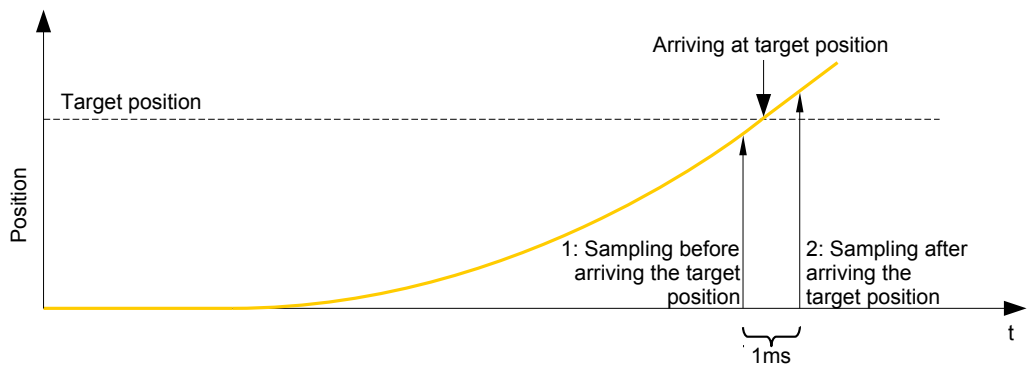
Parameters for the target position

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



9.1.5 Precision loss with relative movements

Chaining relative move commands can lead to loss of precision, when having the "end velocity" set to a non zero value. The following image shows, why.



The current position is sampled once a millisecond. It is possible, that the target position is arrived between two samples. In case the end velocity is not zero the sample of the position value after arriving

the target position will count as the new offset for the next movement. Therefore the next movement *may* be bit longer than expected.

9.1.6 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".

9.2 Velocity

9.2.1 Special feature PD4C USB

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

9.2.2 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.

Note
The limit switches - and therefore the tolerance bands - are active in this mode. See chapter "**Tolerance bands of the limit switches**" for further information about the limit switches.

9.2.3 Activation

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

9.2.4 Control word

The following bits in object **6040**$_h$ (control word) have a special function:

- Bit 2 is used to trigger an quick 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 "1" to "0" 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.

9.2.5 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.

9.2.6 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, sub-index 1 contains the multiplier and sub-index 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. Sub-index 1 contains the speed change, and sub-index 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):

  Emergency stop deceleration in case of the "Quick stop active" state of the "DS402 Power State machine"
- **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$_h$.

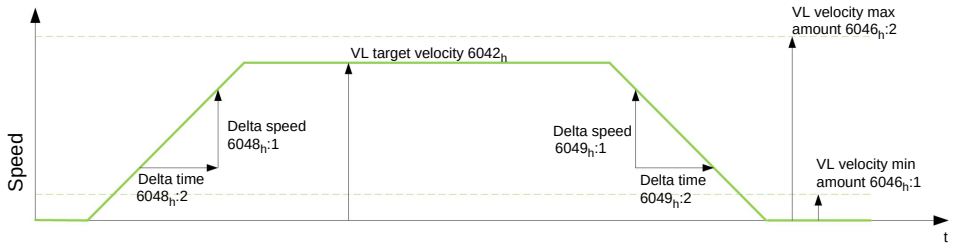
  The maximum speed is set in **6046**$_h$:02$_h$. If the target speed (**6042**$_h$) exceeds the maximum speed, the value is limited to the maximum speed **6046**$_h$:02$_h$.
- **604A**$_h$ (Velocity Quick Stop):

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

The following object can be used for controlling the function:

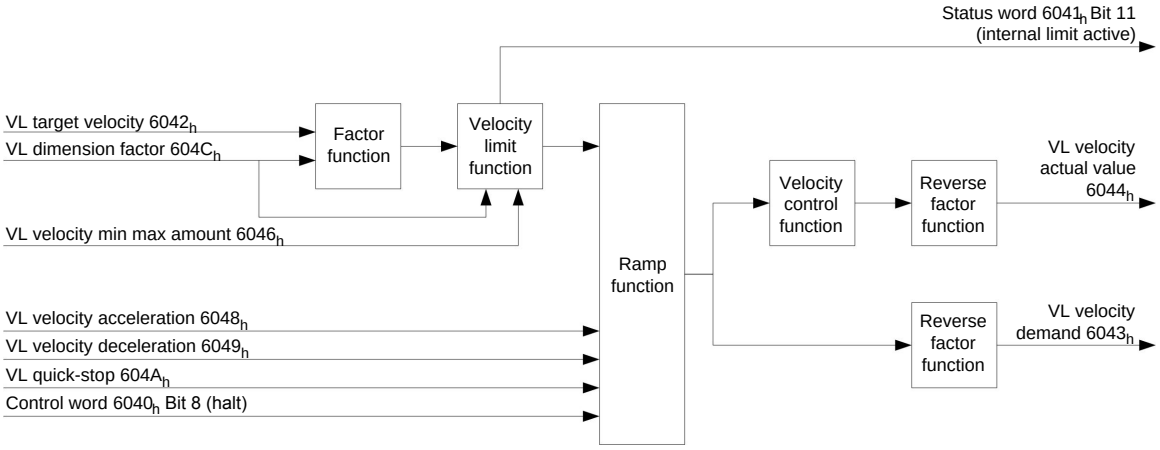
- **6043**$_h$ (Vl Velocity Demand)
- **6044**$_h$ (Vl Velocity Actual Value)

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.



9.3 Profile Velocity

9.3.1 Special feature PD4C USB

Note

Because this motor controller is not fitted with a field bus, the following operating mode is only usable with a NanoJ program.

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

9.3.2 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.

Note

The limit switches - and therefore the tolerance bands - are active in this mode. See chapter "**Tolerance bands of the limit switches**" for further information about the limit switches.

9.3.3 Activation

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

9.3.4 Control word

The following bits in object **6040**$_h$ (control word) have a special function:

- Bit 2 is used to trigger an quick 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 "1" to "0", the motor brakes and comes to a stop.

9.3.5 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 not 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

9.3.6 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), sub-index 01$_h$ to 02$_h$:

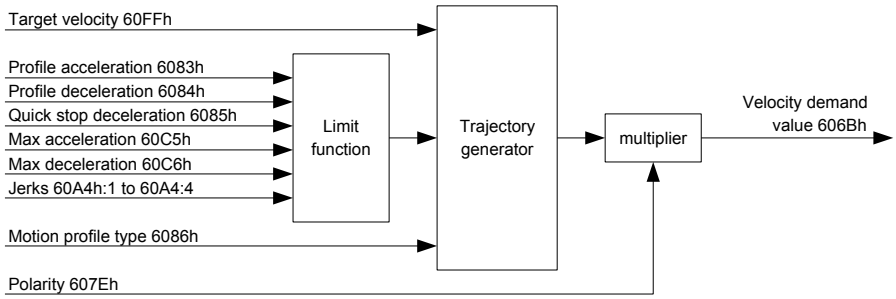
The four jerk values are specified here if a jerk-limited ramp is set.

- **60FF**$_h$(Target Velocity):

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

  Maximum current in mA

Objects in Profile Velocity Mode

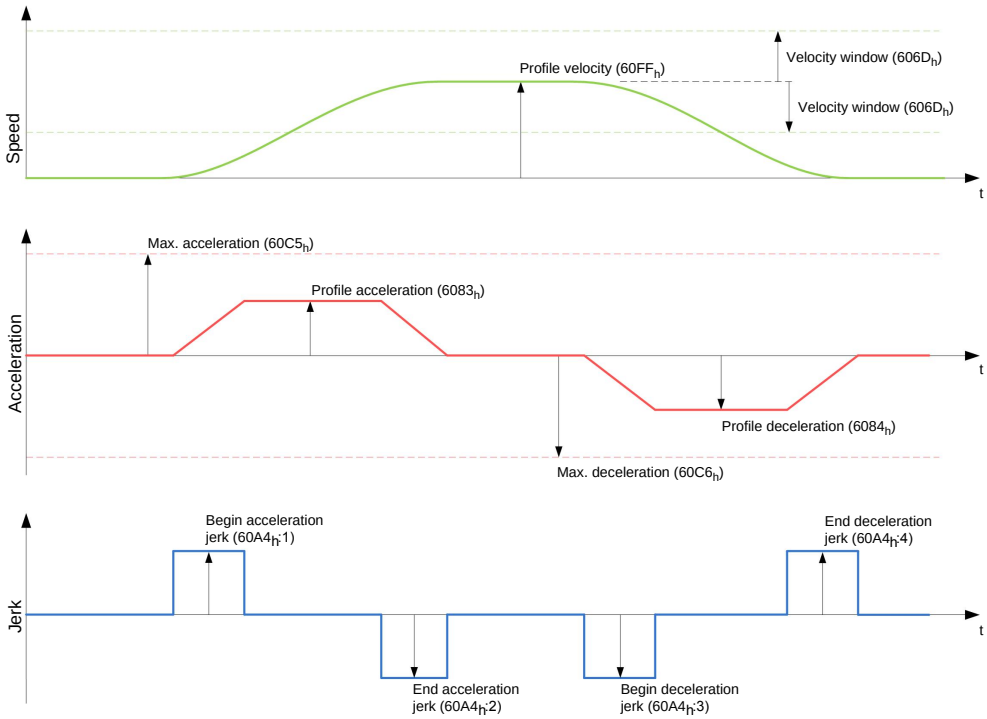


Mode activation

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.

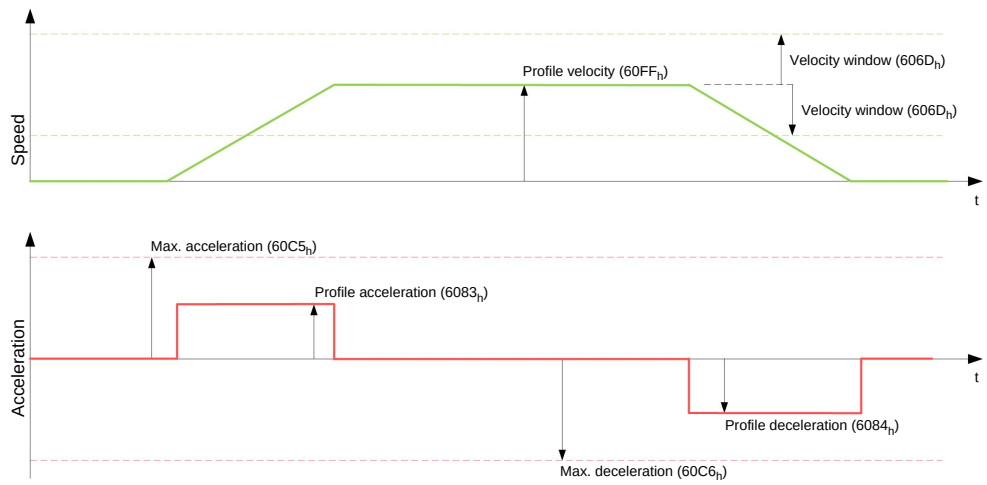
Limitations in the jerk-limited case

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).



9.4 Profile Torque

9.4.1 Special feature PD4C USB

Note

Because this motor controller is not fitted with a field bus, the following operating mode is only usable with a NanoJ program.

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

9.4.2 Description

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

CAUTION

This mode works in Closed-Loop only. Run over the index of the encoder at minimum once before using this function. Otherwise a switch to the state "Operational Enabled" is not possible.

Note

The limit switches - and therefore the tolerance bands - are active in this mode. See chapter "**Tolerance bands of the limit switches**" for further information about the limit switches.

9.4.3 Activation

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

9.4.4 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.

9.4.5 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

- Bit 11 (Torque limit active): In case the maximum torque and target torque values exceed the peak current required to achieve the desired torque (i.e., the required torque exceeds the maximum torque that can be generated) this bit will be set.

9.4.6 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 nominal current (**$203B_h$**:01_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.

Note

The values are not limited to 100% of the nominal current (**$203B_h$**:01_h). Torque values higher than nominal torque (generated by nominal current) can be achieved by setting maximum duration of peak current (**$203B_h$**:02_h, see **I2t motor overload protection** for details). All torque objects are limited by the peak current value.

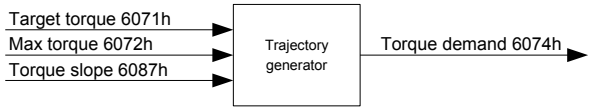
The following objects are also needed for this operation mode:

- **3202_h** Bit 5 (Motor Drive Submode Select):

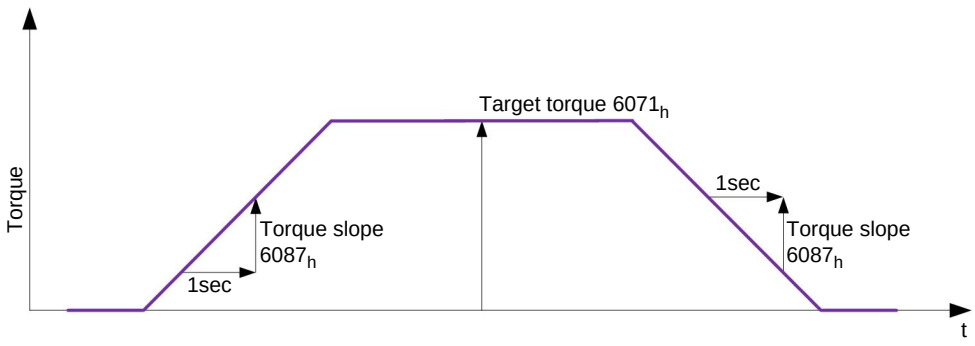
  If this bit is set to "0", the drive control is operated in torque-limited velocity mode, i.e. the maximum speed can be limited in object **2032_h** and the control can work in field weakening mode.

  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



9.5 Homing

9.5.1 Special feature PD4C USB

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

9.5.2 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**").

When using the motor in "Open Loop" mode the object **320A**$_h$:04 needs to be set to the value "1" before starting the homing mode.

Control word

The following bits in object **6040**$_h$(control word) have a special function:

- Bit 2 is used to trigger an quick 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

Bit 13	Bit 12	Bit 10	Description
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
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:

- **607C**$_h$ (Home Offset): Specifies the difference between the zero position of the application and the reference point of the machine.
- **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):

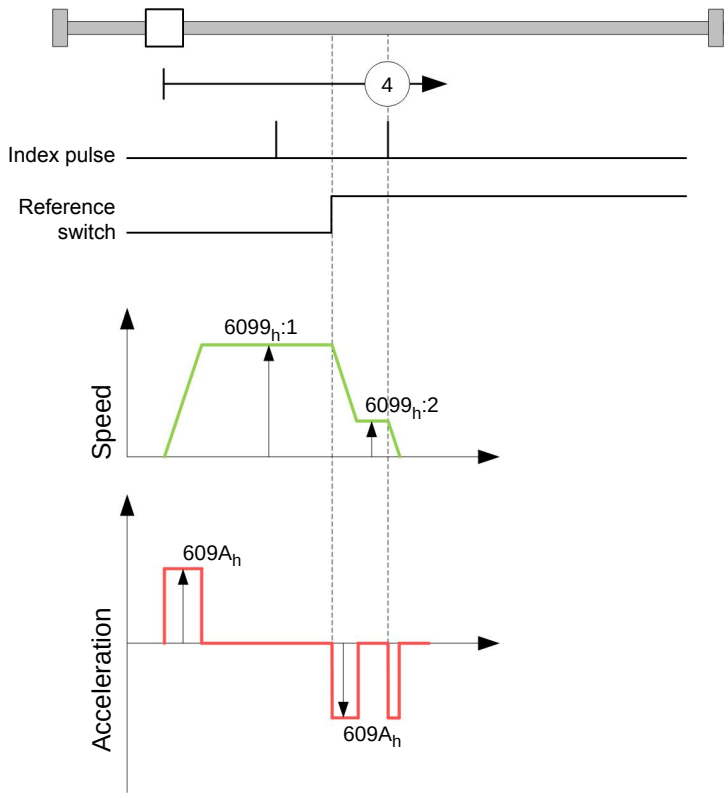
  Minimum current threshold that, when exceeded, detects blocking of the motor at a block.
- **203A**$_h$:02$_h$ (Period Of Blocking):

  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)

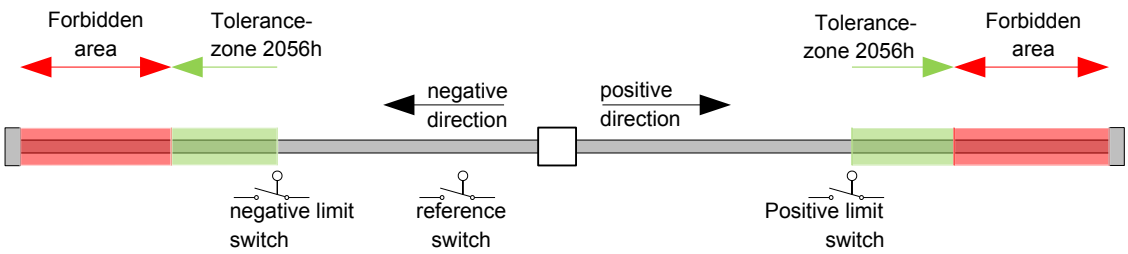
  Specifies the time in ms that the current has to be at least above the minimum current threshold in order to detect a block

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



The previous image displays the setup of the tolerance areas next to the limit switches:

- The tolerance zone begins immediately after the limit switch. It is possible to drive free in this area. The length of the zone can be set in the object **2056**$_h$.
- If the controller drives into the forbidden area, it will execute a deceleration with a quick stop ramp and set an error.

9.5.3 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 do not use the index-pulse of the encoder 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.

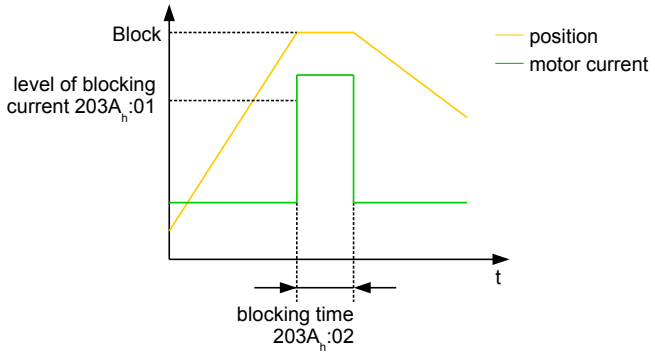
For methods that use homing on block, the same illustrations apply as for the methods with limit switch. New illustrations are not shown as nothing changes except for the missing limit switches. In this case, the limit switches have to be replaced by a mechanical block in the illustrations.

Homing on block

Homing on block is currently working in Closed-Loop only.

"Homing on block" works like every other homing method except for positioning, a block (mechanical end stop) is used instead of a limit switch. Two settings need to be set:

1. Level of blocking current: in object **203A**$_h$:01 the current is defined, on which a drive on the block is detected.
2. Blocking time: in object **203A**$_h$:02 the duration of blocking - in which to motor drives against the block - is defined.



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

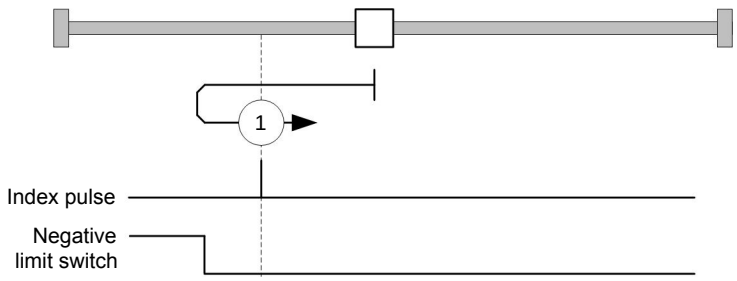
The following methods can be used for homing on block:

- 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

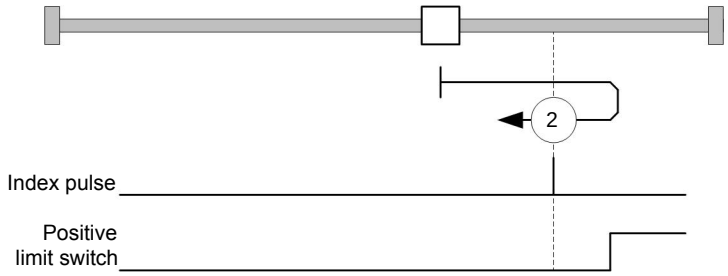
Methods 1 and 2

Reference the limit switch and index pulse.

Method 1 references a negative limit switch and index pulse:



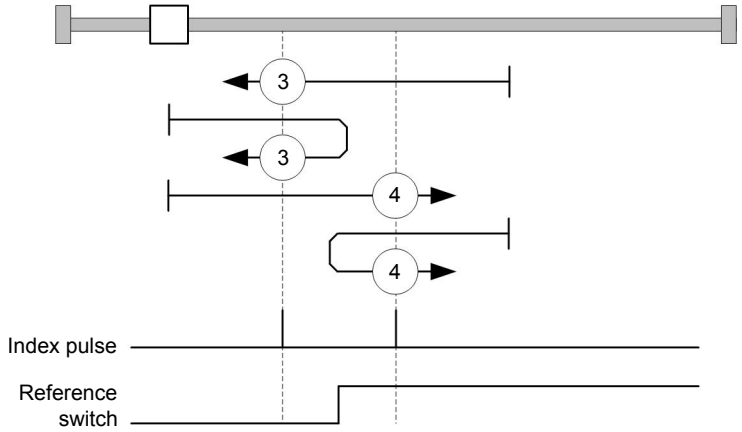
Method 2 references a positive limit switch and index pulse:



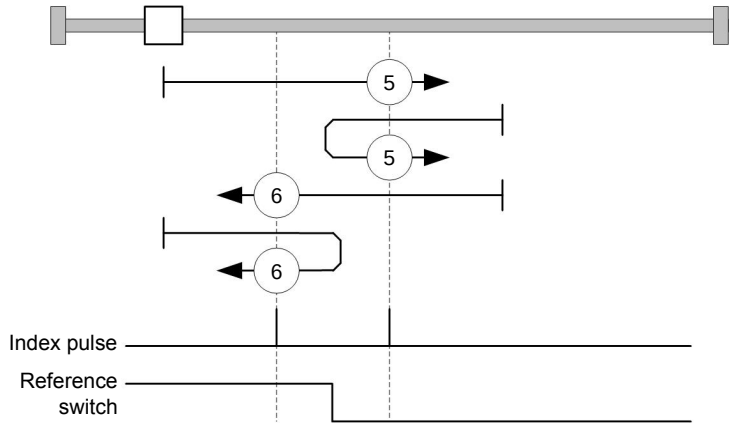
Methods 3 to 6

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:

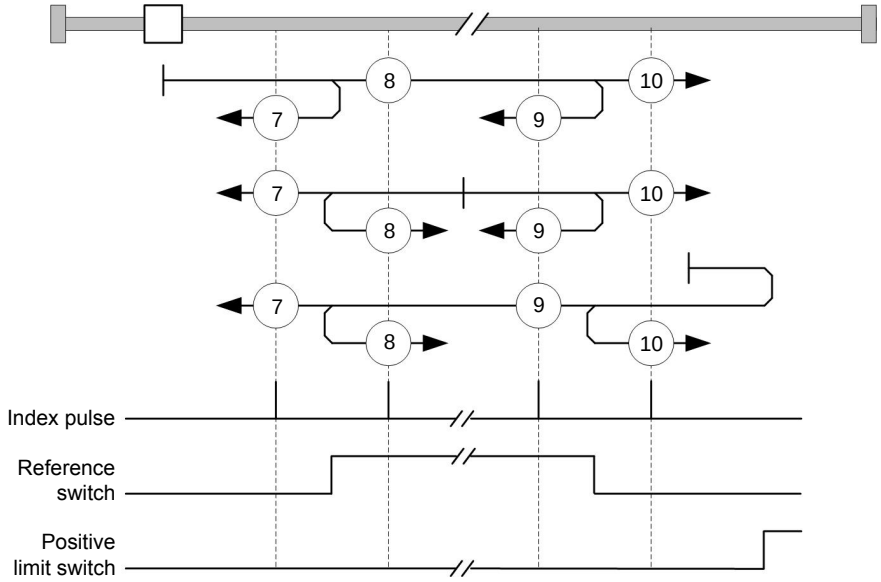


Methods 7 to 14

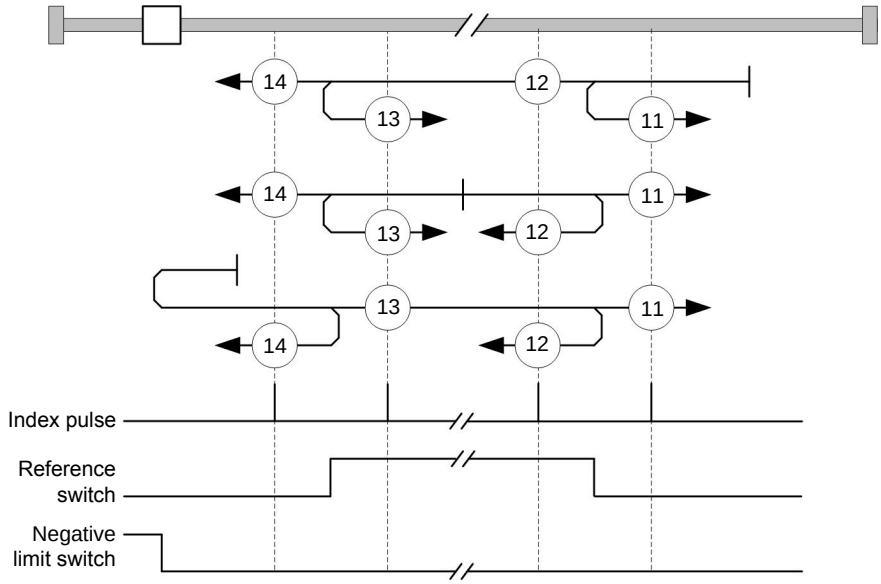
These methods reference the home switch and index pulse (with limit switches).

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:



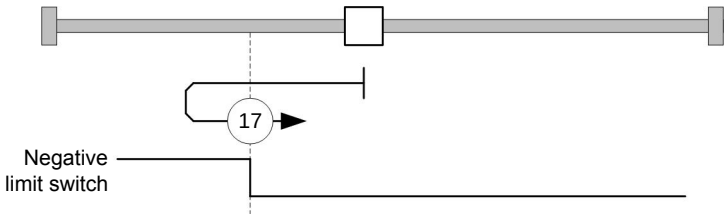
The methods 11 to 14 take the negative limit switch into account:



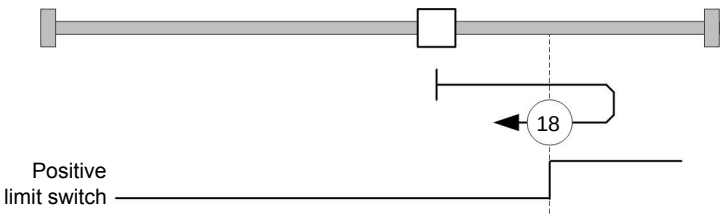
Methods 17 and 18

These methods reference the limit switch without the index pulse.

Method 17 references the negative limit switch:



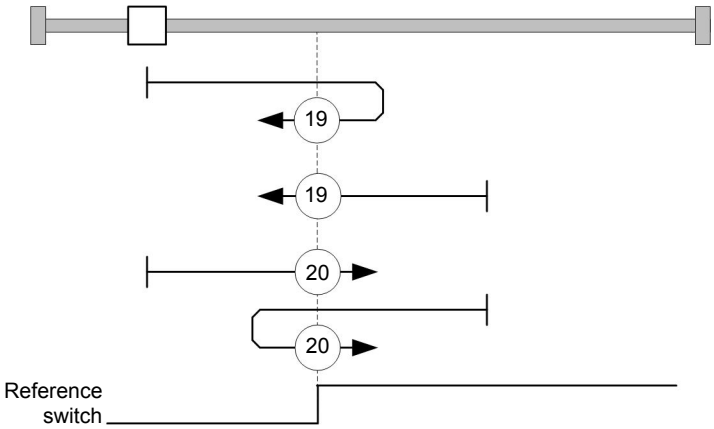
Method 18 references the positive limit switch:



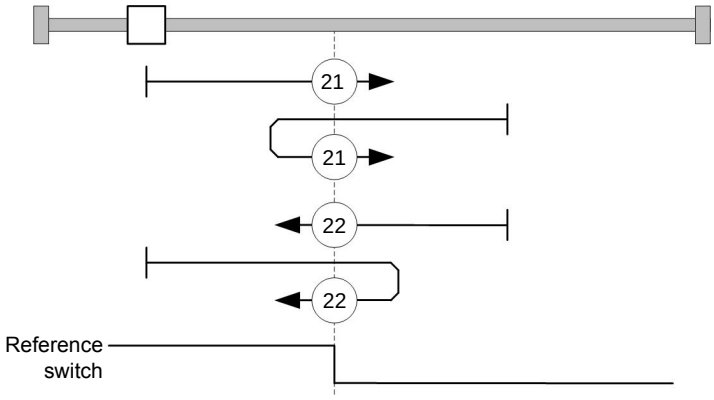
Methods 19 to 22

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:

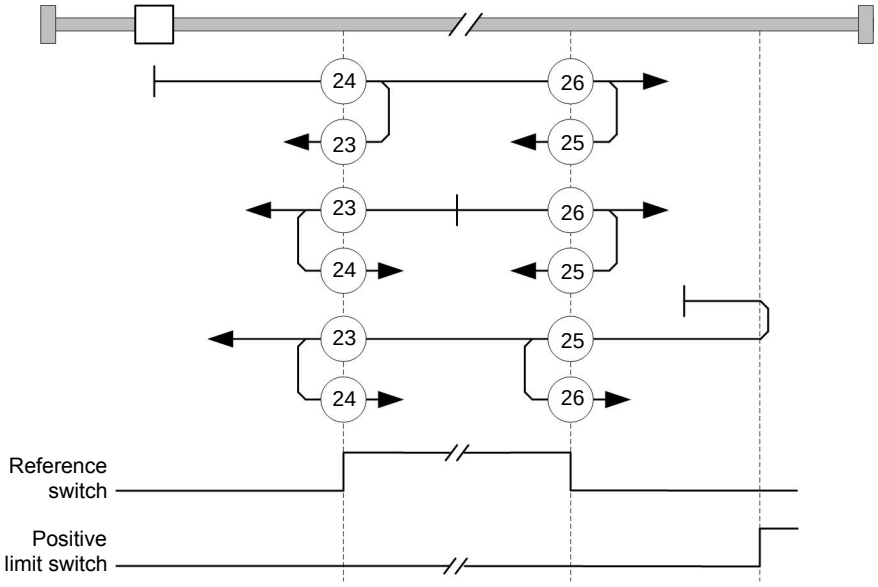


Methods 23 to 30

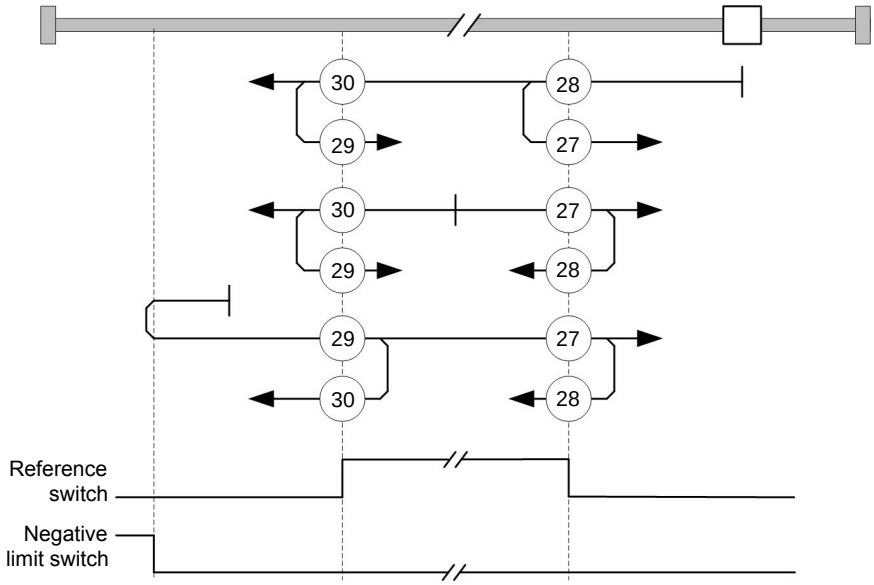
These methods reference the home switch without the index pulse (with limit switches).

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:



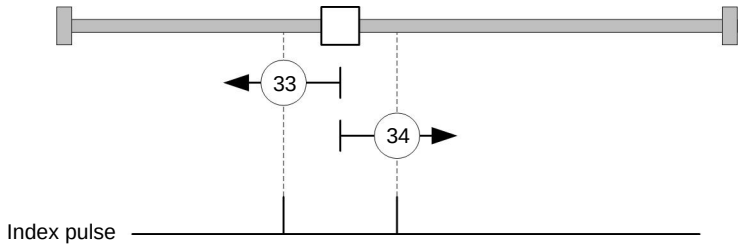
The methods 27 to 30 take the negative limit switch into account:



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.

Note
For the mode 35 it is not necessary to switch the DS402 Power state machine up to the status "Operation Enabled", the status "Switched on" is sufficient.

9.6 Clock/direction mode

9.6.1 Description

This mode is equivalent to the velocity mode but uses pules of two input pins as target.

The analogue mode is only controlled by the "enable"-input: as long as the "enable"-input is not set to logical "high" level the motor won't start driving.

Note
The limit switches - and therefore the tolerance bands - are active in this mode. See chapter "**Tolerance bands of the limit switches**" for further information about the limit switches.

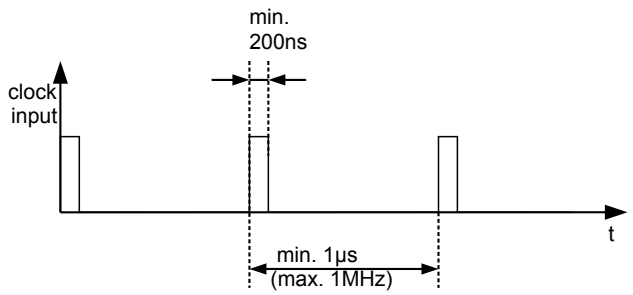
9.6.2 Activation

The activation is not done via the object dictionary but a DIP-switch. For the settings of the switches see chapter "" "**DIP switches**".

9.6.3 General

The following data apply to all sub modes of the clock/direction mode:

- The maximum frequency of the input pulses are limited at 1MHz, the ON-part is not allowed to get shorter than 200 ns.



- The scaling of the steps is done via the objects **2057**$_h$ and **2058**$_h$. The following formula is applied:

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

By default the value for "step width per pulse" is "512", which is equal a full step per pulse. A half step is the value "256", a quarter step accordingly "128" and so on.

Note
During a change of direction it is necessary to let the time of 35µs lapse away before applying a new clock.

9.6.4 Status word

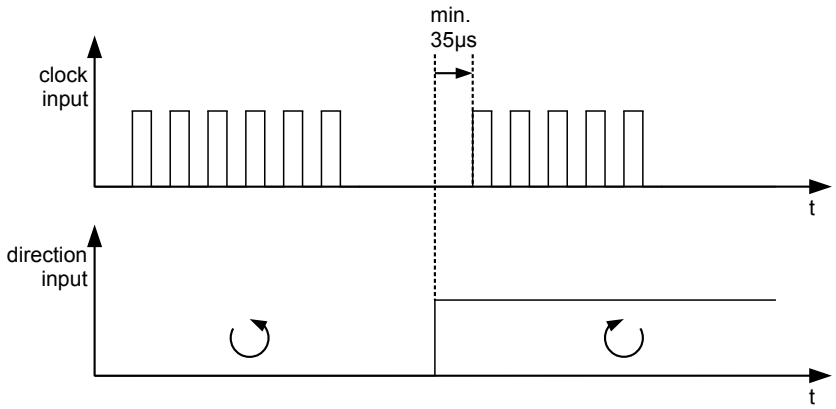
The following bits in object **6041**$_h$ (status word) have a special function:

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

9.6.5 Sub modes of the clock/direction mode

Clock/direction-mode (CD mode)

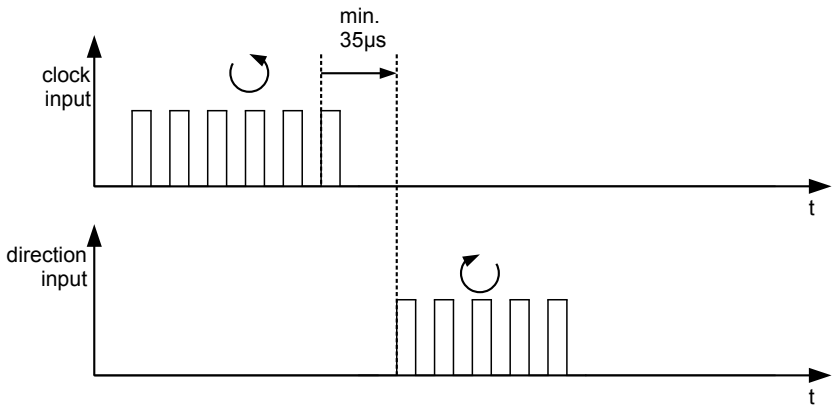
In this mode the clock pulses have to be put to the "clock input", the direction signal is affection the direction (see following image).



Clockwise/counterclockwise mode (CW/CCW mode)

The object **205B**$_h$ needs to be set to "1" in order to activate this mode.

In this sub mode the used input determines the direction of rotation (see following image).



9.7 Analogue Mode

9.7.1 Description

This mode is an equivalent to the velocity mode but uses the height of a voltage of an analogue input as target. This values gets sampled once in a millisecond.

The analogue mode is only controlled by the "enable"-input: as long as the "enable"-input is not set to logical "high" level the motor won't start driving.

The maximal analogue input value is denoted in the following as U_{max}.

9.7.2 Activation

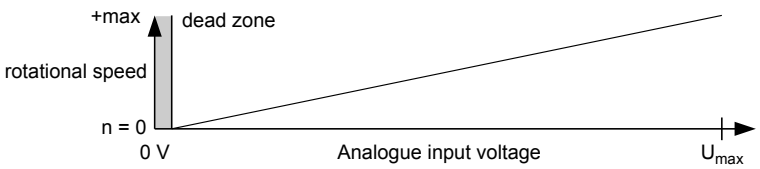
The activation is not done via the object dictionary but a DIP-switch. For the settings of the switches see chapter "**DIP switches**".

9.7.3 Accounting of analogue voltages

There are two modes of accounting of the analogue voltages. This mode is selected by the DIP switch number 2 (see chapter "**DIP switches**").

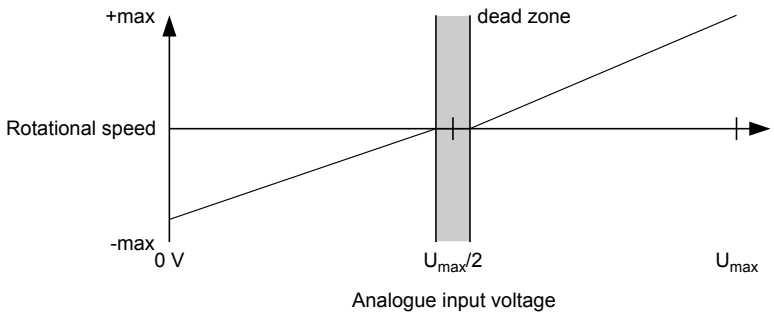
Normal mode

DIP switch 2 "OFF": The maximum of the analogue voltage corresponds the maximal rotational speed. The direction is set by the "direction" input. There is a dead-zone from 0 V to 20 mV in which the motor doesn't drive.



Joystick mode

DIP switch 2 "ON": The half of the maximum analogue input voltage corresponds to the rotational speed zero. If the input voltage falls below the half input voltage, the rotational speed increases in negative direction. Accordingly if the input voltage increases over the half of the input voltage the rotational speed increases in positive direction. The dead zone reaches from n $U_{max}/2 \pm 20$ mV.



9.7.4 Maximum speed of rotation

The maximum speed of rotation can be switched with DIP-switch number 3 (see chapter "**DIP switches**") between 100 rev/min and 1000 rev/min. In case another rotational speed is necessary the maximums rotational speed can set in the configuration file and the object **604C**$_h$ sub-index 01_h and 02_h.

10 Special functions

10.1 Digital inputs and outputs

The motor controller has digital inputs and outputs.

10.1.1 Bit assignment

The software in the controller assigns to bits to every output and input:

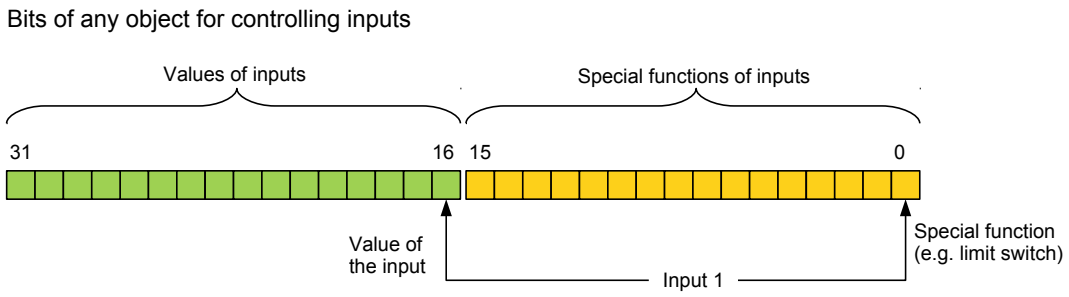
1. The first bit corresponds to a special function of an input or output. These functions are always accessible at bit 0 to bit 15 inclusive. This includes limit switches and clock/direction inputs for the inputs and the brake control for the outputs.
2. The output and the input as a level, these are accessible at bit 16 to 31.

Example

Bit 16 has to be used always to manipulate the value of output 1.

To manipulate the special function "negative limit switch" the bit 0 has to be used.

The assignment is displayed once more in the following image.



10.1.2 Digital inputs

Bit assignment

The software in the controller assigns to bits to every output and input:

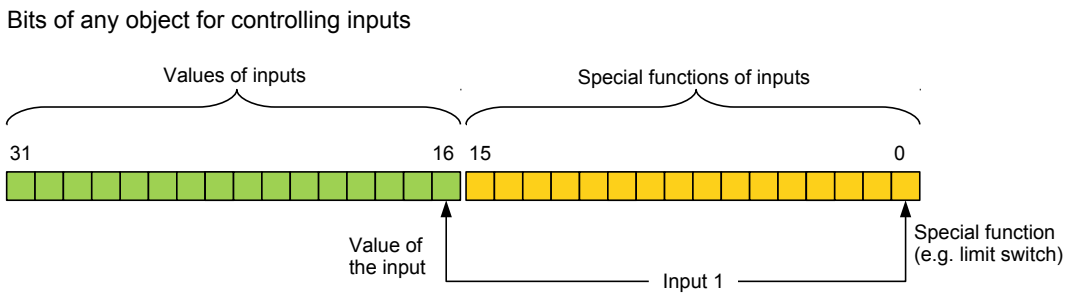
1. The first bit corresponds to a special function of an input or output. These functions are always accessible at bit 0 to bit 15 inclusive. This includes limit switches and clock/direction inputs for the inputs and the brake control for the outputs.
2. The output and the input as a level, these are accessible at bit 16 to 31.

Example

Bit 16 has to be used always to manipulate the value of output 1.

To manipulate the special function "negative limit switch" the bit 0 has to be used.

The assignment is displayed once more in the following image.



Overview

Note
The digital inputs are sampled only once a millisecond. Changes in the input signal shorter than one millisecond cannot be processed.

The following inputs are available:

Input	Special Function	Signal threshold switchable	Differential / single ended
1	Negative limit switch	no, 24 V fix	single ended
2	Positive limit switch	no, 24 V fix	single ended
3	Limit switch	no, 24 V fix	single ended
4	-Enable	The inputs regarding "Enable", "Direction" and "Clock" can only be switch all at once between 5 V or 24 V (see **3240**$_h$:06$_h$)	The inputs regarding "Enable", "Direction" and "Clock" can only be switch all at once. Set to "single ended" the negative input is deactivated (e.g. "-Enable") (see **3240**$_h$:07$_h$)
5	+Enable		
6	-Direction		
7	+Direction		
8	-Clock		
9	+Clock		

The limit switches are described in chapter **Tolerance bands of the limit switches**.

Object entries

The following object dictionary 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$_h$.

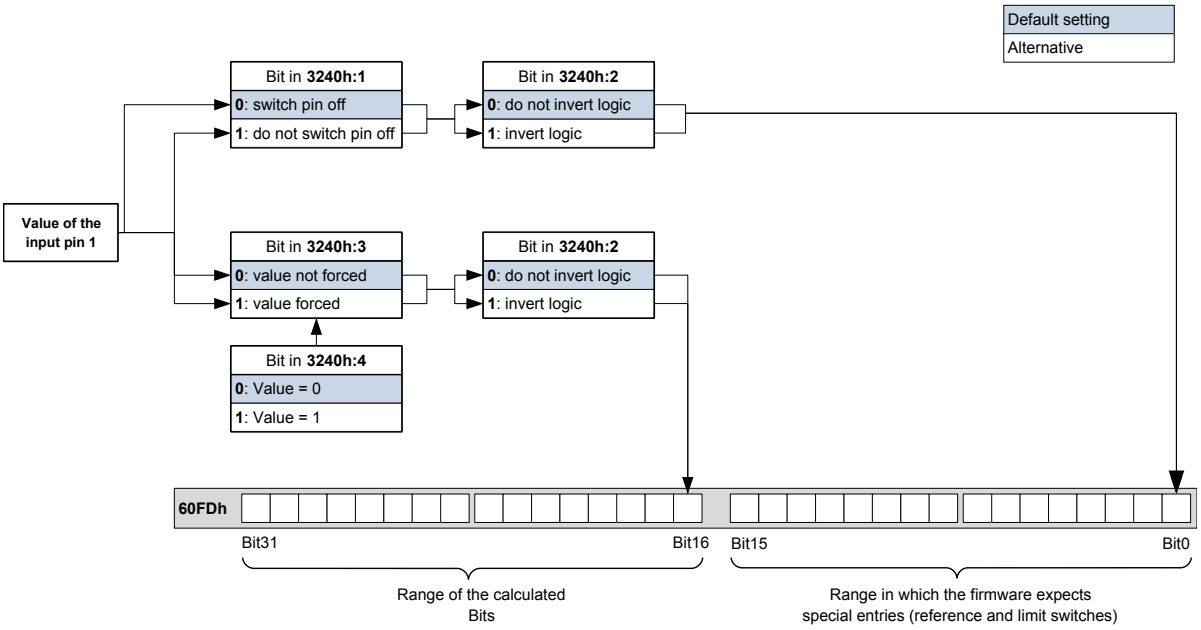
- **3240**$_h$:05$_h$

  This object contains the unmodified input value.

Computation of the inputs

Computation of the input using input 1 as an example:

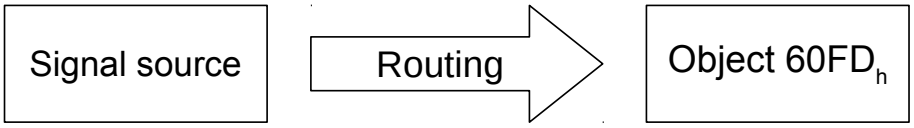
The value of bit 0 of object **60FD**$_h$ is interpreted by the firmware as a negative limitation switch, and the result of the complete computation is stored in bit 16.



Input Routing

Principle

In order to deal with a more flexible input assignment there is a mode called "Input Routing Mode". This mode assigns a signal from a source to a bit in the object **60FD**$_h$.



Activation

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

Note
The entries **3240**$_h$:01$_h$ to **3240**:04$_h$ will have **no** function anymore until the output routing is switched off again.

Note
With activating the input routing the values in the object **3242**$_h$ get changed. These values corresponds to the function of the inputs without the input routing. The controller's inputs are behaving identically with activating the input routing. Therefore you should not switch between normal mode and input routing back and forth.

Routing

The object **3242**$_h$ determines, which signal source will get routed to which bit in **60FD**$_h$. The sub-index 01_h of **3242**$_h$ determines bit 0, sub-index 02_h bit 1, and so on. The signal sources and their numbers are listed in the following tables

Number		
dec	**hex**	**Signal source**
00	00	Signal is always "0"
01	01	Physical input 1
02	02	Physical input 2
03	03	Physical input 3
04	04	Physical input 4
05	05	Physical input 5
06	06	Physical input 6
07	07	Physical input 7
08	08	Physical input 8
09	09	Physical input 9
10	0A	Physical input 10
11	0B	Physical input 11
12	0C	Physical input 12
13	0D	Physical input 13
14	0E	Physical input 14
15	0F	Physical input 15
16	10	Physical input 16
65	41	Hall input "U"
66	42	Hall input "V"
67	43	Hall input "W"
68	44	Encoder input "A"
69	45	Encoder input "B"
70	46	Encoder input "Index"
71	47	USB power signal
72	48	Status ethernet active
73	49	DIP switch 1
74	4A	DIP switch 2
75	4B	DIP switch 3
76	4C	DIP switch 4
77	4D	DIP switch 5
78	4E	DIP switch 6
79	4F	DIP switch 7
80	50	DIP switch 8

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

Number		
dec	**hex**	**Signal source**
128	80	Signal is always "1"
129	81	Inverted physical input 1
130	82	Inverted physical input 2
131	83	Inverted physical input 3
132	84	Inverted physical input 4
133	85	Inverted physical input 5
134	86	Inverted physical input 6
135	87	Inverted physical input 7
136	88	Inverted physical input 8
137	89	Inverted physical input 9
138	8A	Inverted physical input 10
139	8B	Inverted physical input 11
140	8C	Inverted physical input 12
141	8D	Inverted physical input 13
142	8E	Inverted physical input 14
143	8F	Inverted physical input 15
144	90	Inverted physical input 16
193	C1	Inverted hall input "U"
194	C2	Inverted hall input "V"
195	C3	Inverted hall input "W"
196	C4	Inverted encoder input "A"
197	C5	Inverted encoder input "B"
198	C6	Inverted encoder input "Index"
199	C7	Inverted USB power signal
200	C8	Inverted status "Ethernet active"
201	C9	Inverted DIP switch 1
202	CA	Inverted DIP switch 2
203	CB	Inverted DIP switch 3
204	CC	Inverted DIP switch 4
205	CD	Inverted DIP switch 5
206	CE	Inverted DIP switch 6
207	CF	Inverted DIP switch 7
208	D0	Inverted DIP switch 8

Example

The physical input 1 is designated to be route to bit 16 of the object **60FD**$_h$:

The number of the signal source of the physical input 1 is the value "1". The routing for bit 16 will be written in the object 3242_h:11_h.

Therefore the object 3242_h:11_h needs to be set to the value "1".

10.1.3 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.

Circuits

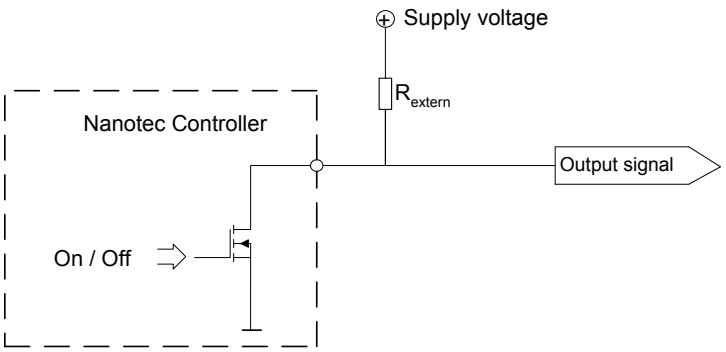
⚠ CAUTION

Take notice of the maximum load of the output (see therefore the chapter "Electrical properties").

The outputs are realized as "Open Drain". Therefore a external supply voltage is needed.

Example

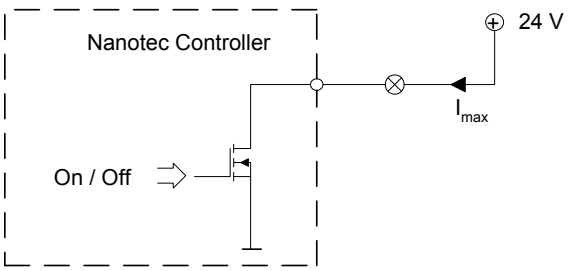
the output signal is supposed to get used further on. For that purpose a circuit like to one shown in the following image is necessary.



At a supply voltage of +24 V a resistor value of R_{extern} of 10 kΩ recommended.

Example

A simple appliance is supposed to get used with the digital output.



Object entries

Additional object dictionary 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$_h$

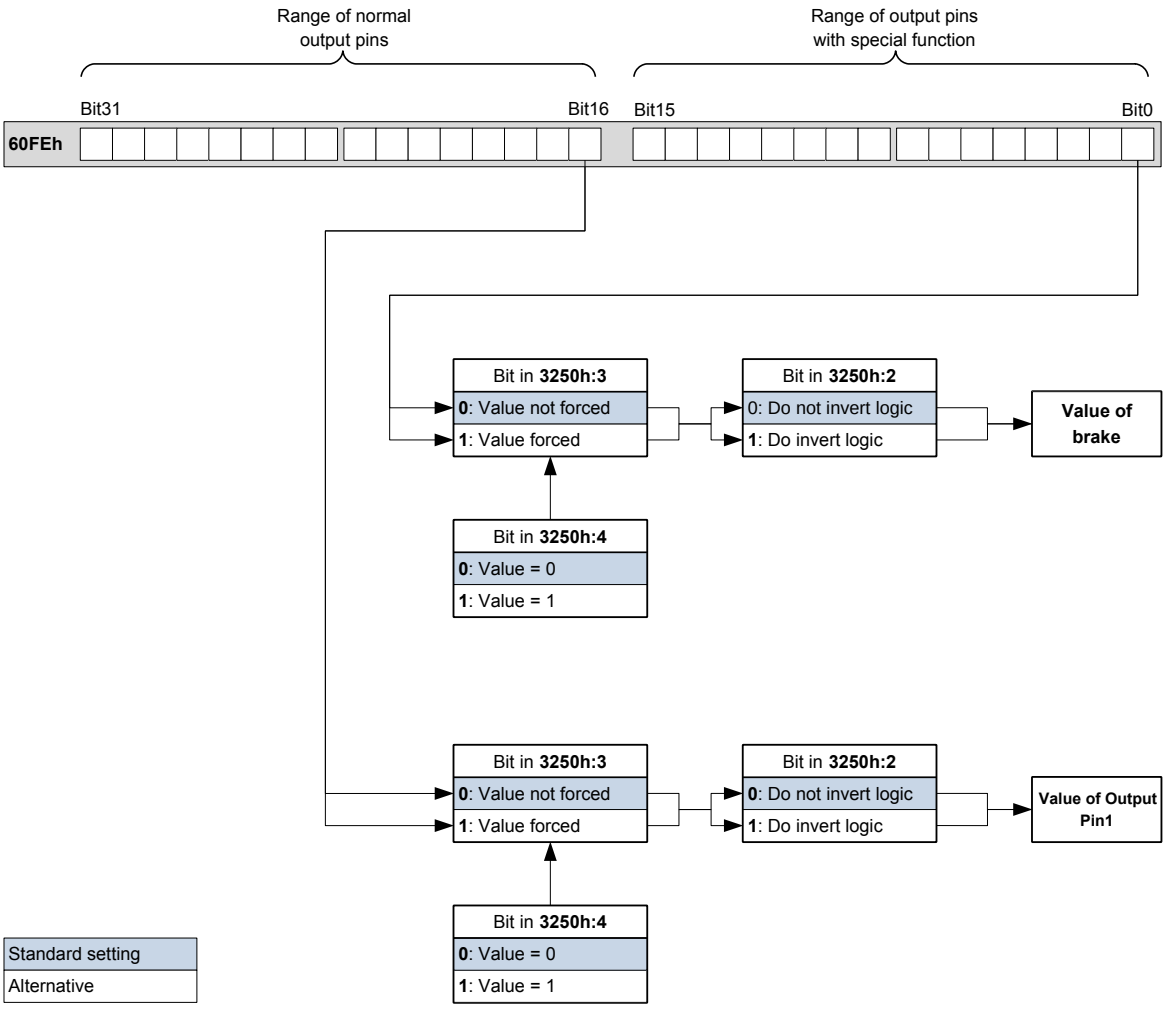
  If a bit is set in **3250**$_h$, the output is manually controlled. The value for the output is then contained in object **3250**$_h$:04$_h$, which is also possible for the brake output.
- **3250**$_h$:04$_h$

  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.

Computation of the outputs

Example of the computation of the bits for the outputs:

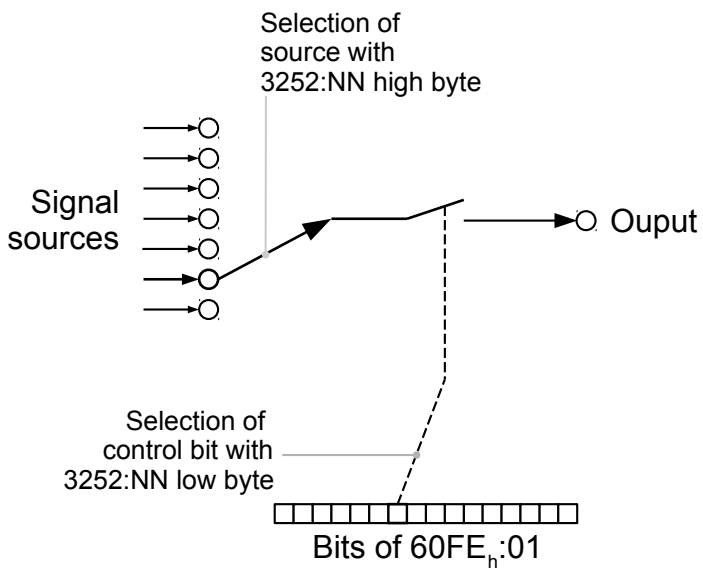


Output Routing

Principle

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

The selection of the source is done with **3250**$_h$:01 to 05 high byte. The "control bit" is selected with **3250**$_h$:01 to 05 low byte (see following figure).



Activation

This mode is activated by setting the object **3250**$_h$:08$_h$ (Routing Enable) to "1".

Note
The entries **3250**$_h$:01$_h$ to **3250**:04$_h$ will have **no** function anymore until the output routing is switched off again.

Routing

The sub-index of object **3252**$_h$ determines, which signal source will get routed to which output. The assignment of sub-index to output pins are listed below:

Sub-index 3252$_h$	Output Pin
01$_h$	Configuration of brake output (if available)
02$_h$	Configuration of output 1
03$_h$	Configuration of output 2 (if available)
04$_h$	Configuration of output 3 (if available)
05$_h$	Configuration of output 4 (if available)

Note
The maximum output frequency of brake output, output 1 and output 2 is 10kHz. All other output pins can only generate up to 500Hz signals.

These subentries **3252**$_h$:01$_h$ to 05$_h$ are 16 bit entries, whereat the high byte (bit 15 to bit 8) indicates the signal source (e.g. the pwm generator) and the low byte (bit 7 to bit 0) determines the control bit in **60FE**$_h$:01.

Bit 7 inverts the control bit taken from **60FE**$_h$:01. Normally the value "1" **60FE**$_h$:01 switches the signal "on", bit with bit 7 set, the value "0" switches the signal on.

Number in 3252:01 to 05	
00XX$_h$	Output is always "1"
01XX$_h$	Output is always "0"
02XX$_h$	Encoder signal with frequency divider 1
03XX$_h$	Encoder signal with frequency divider 2

Number in 3252:01 to 05	
$04XX_h$	Encoder signal with frequency divider 4
$05XX_h$	Encoder signal with frequency divider 8
$06XX_h$	Encoder signal with frequency divider 16
$07XX_h$	Encoder signal with frequency divider 32
$08XX_h$	Encoder signal with frequency divider 64
$09XX_h$	Position Actual Value (**6064_h**) with frequency divider 1
$0AXX_h$	Position Actual Value (**6064_h**) with frequency divider 2
$0BXX_h$	Position Actual Value (**6064_h**) with frequency divider 4
$0CXX_h$	Position Actual Value (**6064_h**) with frequency divider 8
$0DXX_h$	Position Actual Value (**6064_h**) with frequency divider 16
$0EXX_h$	Position Actual Value (**6064_h**) with frequency divider 32
$0FXX_h$	Position Actual Value (**6064_h**) with frequency divider 64
$10XX_h$	brake pwm signal, which is configured with **2038_h**:05_h and 06_h
$11XX_h$	inverted brake pwm signal, which is configured with **2038_h**:05_h and 06_h

Example

The encoder output signal will get routed to output 1 with a frequency divider of "4". The output will get controlled by bit 5 of the object **60FE**:01.

- **3250_h**:08_h = 1 (switch on routing)
- **3252_h**:02_h = 0405_h ($04XX_h$ + 0005_h). The following apply:
- $04XX_h$: Encoder signal with frequency divider 4
- 0005_h: Selection of bit 5 of the object **$60FE_h$**:01

Switching on the output is done with setting the object **60FE**:01 to the value "20_h".

Example

The brake pwm signal will get routed to output 2. The automated brake control is using the bit 0 of **60FE**:01, therefore this bit will be the control bit of the object **60FE**:01

- **3250_h**:08_h = 1 (switch on routing)
- **3252_h**:03_h = 01080_h ($01XX_h$ + 0080_h). The following apply:
  - $01XX_h$: Brake pwm signal
  - 0080_h: Selection of the inverted bit 0 of the object **60FE**:01

10.2 I^2t motor overload protection

10.2.1 Description

⚠ WARNING

For stepper motors only the nominal current but no maximum current is specified. Therefore the usage of I^2t with stepper motors is used at the users own risk

I^2t motor overload protection has the objective of preventing damage to the motor and 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 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.

10.2.2 Object entries

The following objects affect I^2t 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 I^2t:

- **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

10.2.3 Activation

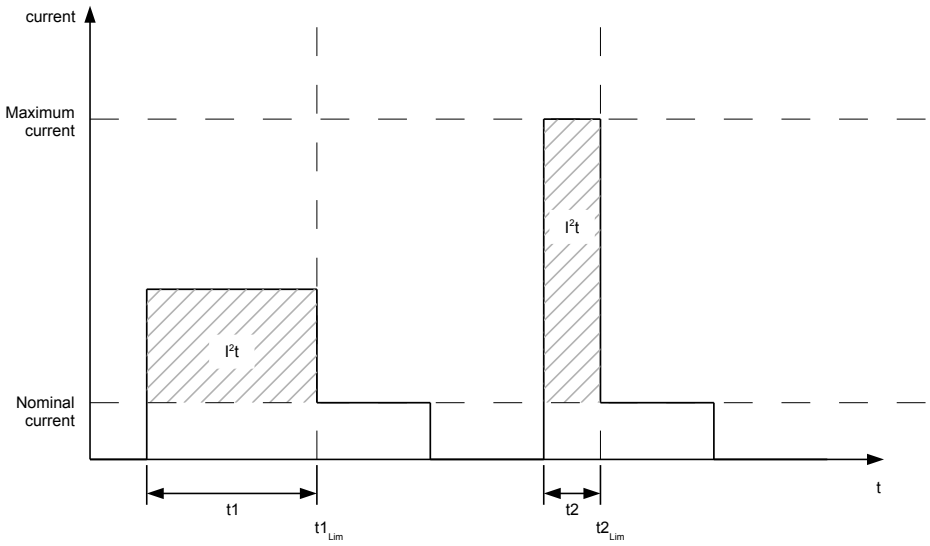
The three object entries above must have been appropriately specified to activate the mode. This means that the maximum current must be greater than the nominal current, and a time value must be entered for the maximum time of the maximum current. I^2t operability remains deactivated when these conditions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

10.2.4 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.

10.3 Save Objects

Objects can only be saved with the file `pd4cfg.txt` , the save mechanism with the object 1011_h and 1010_h is deactivated with this controller. Aside that a NanoJ program can access the object **2700_h** to persist data.

11 Programming with NanoJ

11.1 Introduction

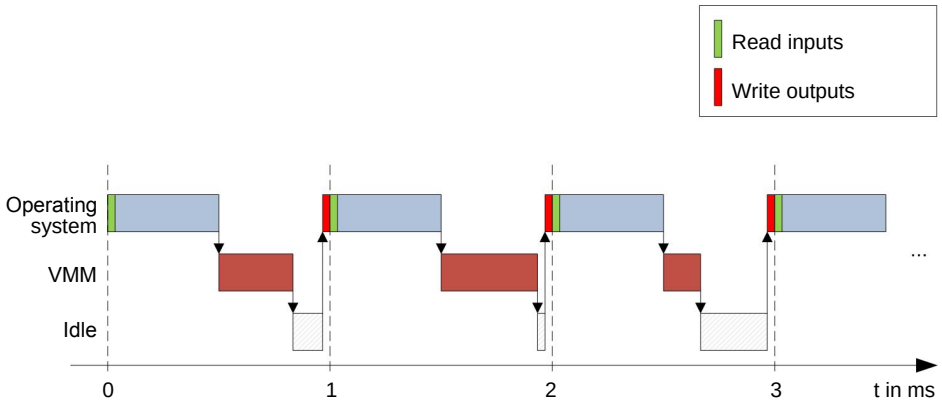
The VMM (Virtual Machine Monitor) is a protected execution environment within the firmware. The user can load his or her own programs ("User Program") in this environment via USB . These can trigger functions in the motor controller, for example by reading or writing entries in the object directory.

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.

Once loaded on the controller, the user program will start autocratically after power up or restart of the controller.

11.2 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 NanoJ 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**$_h$.

11.3 Interaction of the user program with the motor controller

11.3.1 Communication options

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

- Reading and writing of object dictionary values per PDO mapping

- Direct reading and writing of object dictionary values via system calls
- Calling up of other system calls (e.g. write debug output)

Via a PDO mapping, object dictionary 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 object dictionary 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 object dictionary 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.

11.3.2 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 object dictionary values and to access less frequently used object dictionary 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 object dictionary 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.

11.3.3 Saving of data

⚠ **WARNING**
Make sure, that the controller is in state "Switch on disabled" at the start of the saving and during the saving process.

CAUTION
The memory is able to support a certain amount of writing cycles, after that, the memory is damaged. Make sure to keep the amount of writings as low as possible.

For persisting data permanently from within NanoJ there is a special object available: **2700_h**. The subindices have the following meanings:

- 01_h: If the value "1" is written in this sub-index, the subindices 02_h to 09_h are getting saved. When the process has finished, the value "0" is put in this sub-index.
- 02_h to 09_h: In those subindices data can get deposited for saving. At a start or restart of the controller the data will get loaded again.

11.4 Object dictionary entries for controlling and configuring the VMM

11.4.1 Object dictionary entries

The VMM is controlled and configured by means of object dictionary entries in the object range **2300**$_h$ to **2330**$_h$.

object dictionary Index	Name
2300$_h$	NanoJ Control
2301$_h$	NanoJ Status
2302$_h$	NanoJ Error Code
2310$_h$	NanoJ Input Data Selection
2320$_h$	NanoJ Output Data Selection
2330$_h$	NanoJ In/output Data Selection

11.4.2 Example

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

- Rename the file "`TEST1.USR`" to "`vmmcode.usr`".
- Copy the file "`vmmcode.usr`" via USB to the controller.
- Start the NanoJ program with setting the object **2300**$_h$, Bit 0 to "1". or restart the controller.
- Check the object **2302**$_h$ if an error has occurred and check the objects **2301**$_h$, Bit 0 to be set to "1" (vmm is actually running).

Note
Due to limitations in the USB implementation, after a restart of the controller the file "`vmmcode.usr`" will get set to a size of 16kB and the creation date is reset to 13.03.2012.

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

11.5 NanoJ Easy V2

11.5.1 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".

11.5.2 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. The motor controller must be restarted to launch the program; please read the "**NanoJ program**" section starting at step 2 for details.

11.5.3 Structure of a mapping

Introduction

This method can be used to directly link a variable in the NanoJ 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 sub-index 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.

11.6 System calls

11.6.1 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.

11.6.2 Access to the object directory

- void **od_write**(U32 index, U32 sub-index, 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
sub-index	Sub-index 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()`.

- U32 **od_read**(U32 index, U32 sub-index)

  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
sub-index	Sub-index of the object being read in the object directory
Return value	Content of the object dictionary 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();
```

11.6.3 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

11.6.4 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 float val)

Note
The debug outputs are first written to a separate area of the object dictionary and are read out from there by the web interface. This object dictionary entry has the index **2600**$_h$ and is 64 characters long. The sub-index 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 sub-index 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.

12 Object directory description

12.1 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

12.2 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.

12.3 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 sub-index 0.
- ARRAY: This objects always consist of one sub-index 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 sub-index 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. The length of the string appears in sub-index 0 of this object. The individual characters are contained as of sub-index 1. 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.

Saveable

Designates if a object can be saved and - if so - in which category

Firmware Version

The firmware version of the first occurence of the object is entered here.

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.

12.4 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 sub-index 1 or higher are listed in the table with the heading "Value description". The table contains the following entries:

Sub-index

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.

12.5 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
Saveable	No
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	• : 00000000_h • PD4-C5918M4204-E-01: 00040192_h • PD4-C6018L4204-E-01: 00040192_h • PD4-CB59M024035-E-01: 00020192_h • PD4-C5918L4204-E-01: 00040192_h • PD4-C5918M4204-KSAR2: 00040192_h
Firmware Version	FIR-v1426
Change History	

Description

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Motor Type [16]															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Device profile number [16]															

Motor Type[16]

Describes the supported motor type. The following values are possible:

- Bit 23 to Bit 16: value "1": Servo drive
- Bit 23 to Bit 16: value "2": Stepper motor
- Bit 31 to Bit 16: vlaue "255": Multiple device module

Device profile number[16]

Describes the supported CANopen standard.

Values:

0192_h resp. 0402_d (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
Saveable	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
MAN	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"

MAN
Manufacturer specific: engine turned in the wrong direction.

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
Saveable	No
Firmware Version	FIR-v1426
Change History	Amount of subentries has changed from 2 to 9

Value description

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

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

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

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

Sub-index	04_h
Name	Standard Error Field
Data type	UNSIGNED32
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

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

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

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

Sub-index	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 sub-index 1. The existing entries in the subindices 1 to 7 are shifted back by one. The error at sub-index 7 is removed.

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

Bit description

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Error Number [8]								Error Class [8]							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Error Code [16]															

Error Number [8]

This 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**.
14	Non-volatile storage full, restart of controller for cleanup necessary
15	Motor blocked
16	Non-volatile storage corruped, restart of controller for cleanup necessary
17	Slave needed too much time for sending PDO-tickets
18	Hall Sensor faulty
19	PDO not processed due to length error
20	PDO length exceeded

Error number	Description
21	Non-volatile storage full, restart of controller for cleanup necessary
22	Nominal current must be set (203B:01)

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
2300_h	Current at output of motor controller too high
3100_h	Overvoltage/undervoltage at motor controller input
4200_h	Temperature error in motor controller
6320_h	Nominal Current must be set.
7121_h	Motor blocked
7305_h	Incremental or hall sensor faulty
7600_h	Flash storage full or corrupted, restart controller for a cleanup.
8000_h	Field bus error
8130_h	Only CANopen: Life guard error or heartbeat error
8200_h	Only CANopen: Slave needed too much time for sending a pdo message
8210_h	Only CANopen: PDO not processed due to length error
8220_h	Only CANopen: PDO length exceeded
8611_h	Position monitoring error: following error
8612_h	Position monitoring error: Reference switch hit and entered forbidden area

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
Saveable	No
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	• : • PD4-C5918M4204-E-01: PD4-C5918M4204-E-01 • PD4-C6018L4204-E-01: PD4-C6018L4204-E-01 • PD4-CB59M024035-E-01: PD4-CB59M024035-E-01

	• PD4-C5918L4204-E-01: PD4-C5918L4204-E-01 • PD4-C5918M4204-KSAR2: PD4-C5918M4204-KSAR2
Firmware Version	FIR-v1426
Change History	

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
Saveable	No
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	
Firmware Version	FIR-v1426
Change History	

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
Saveable	No
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	FIR-v1626-B336847
Firmware Version	FIR-v1426
Change History	

1010h Store Parameters

Function

This object has no significance for this motor controller.

Object description

Index	1010_h
Object Name	Store Parameters
Object Code	ARRAY
Data type	UNSIGNED32
Saveable	No
Firmware Version	FIR-v1426
Change History	Firmware Version FIR-v1436: Entry "Object Name" modified from "Store Parameter" to "Store Parameters". Firmware Version FIR-v1436: Amount of subentries has changed from 3 to 4. Firmware Version FIR-v1512: Amount of subentries has changed from 4 to 5. Firmware Version FIR-v1540: Amount of subentries has changed from 5 to 7.

Value description

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

Sub-index	01_h
Name	Save All Parameters To Non-volatile Memory
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000001_h

Sub-index	02_h
Name	Save Communication Parameters To Non-volatile Memory
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000001_h

Sub-index	03_h
Name	Save Application Parameters To Non-volatile Memory
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	

Specified Value	00000001_h

Sub-index	04_h
Name	Save Customer Parameters To Non-volatile Memory
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000001_h

Sub-index	05_h
Name	Save Drive Parameters To Non-volatile Memory
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000001_h

Sub-index	06_h
Name	Save Tuning Parameters To Non-volatile Memory
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000001_h

1011h Restore Default Parameters

Function

This object has no significance for this motor controller.

Object description

Index	1011_h
Object Name	Restore Default Parameters
Object Code	ARRAY
Data type	UNSIGNED32
Saveable	No
Firmware Version	FIR-v1426
Change History	Firmware Version FIR-v1436: Entry "Object Name" modified from "Restore Default Parameter" to "Restore Default Parameters". Firmware Version FIR-v1436: Amount of subentries has changed from 2 to 4. Firmware Version FIR-v1512: Amount of subentries has changed from 4 to 5. Firmware Version FIR-v1512: Entry "Name" modified from "Restore The Comm Default Parameters" to "Restore Communication Default Parameters".

	Firmware Version FIR-v1512: Entry "Name" modified from "Restore The Application Default Parameters" to "Restore Application Default Parameters". Firmware Version FIR-v1540: Amount of subentries has changed from 5 to 7.

Value description

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

Sub-index	01_h
Name	Restore All Default Parameters
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000001_h

Sub-index	02_h
Name	Restore Communication Default Parameters
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000001_h

Sub-index	03_h
Name	Restore Application Default Parameters
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000001_h

Sub-index	04_h
Name	Restore Customer Default Parameters
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000001_h

Sub-index	05_h

Name	Restore Drive Default Parameters
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000001_h

Sub-index	06_h
Name	Restore Tuning Default Parameters
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

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
Saveable	No
Firmware Version	FIR-v1426
Change History	

Value description

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

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

Sub-index	02_h
Name	Product Code
Data type	UNSIGNED32
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	• : 00000000_h • PD4-C5918M4204-E-01: 00000001_h • PD4-C6018L4204-E-01: 00000002_h • PD4-CB59M024035-E-01: 00000003_h • PD4-C5918L4204-E-01: 00000017_h • PD4-C5918M4204-KSAR2: 00000027_h

Sub-index	03_h
Name	Revision Number
Data type	UNSIGNED32
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	$065A0000_h$

Sub-index	04_h
Name	Serial Number
Data type	UNSIGNED32
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

1020h Verify Configuration

Function

This object has no significance for this controller.

Object description

Index	1020_h
Object Name	Verify Configuration
Object Code	ARRAY
Data type	UNSIGNED32
Saveable	yes, category: verify
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	
Firmware Version	FIR-v1540
Change History	

Value description

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

Sub-index	01_h
Name	Configuration Date
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Sub-index	02_h
Name	Configuration Time
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

2028h MODBUS Slave Address

Funktion

This object contains the slave address for modbus interface.

Object description

Index	2028_h
Object Name	MODBUS Slave Address
Object Code	VARIABLE
Data type	UNSIGNED8
Saveable	yes, category: communication
Access	Read/write
PDO Mapping	No
Admissible Values	1-247
Specified Value	05_h
Firmware Version	FIR-v1436
Change History	

202Ah MODBUS RTU Baudrate

Funktion

This object contains the baudrate of modbus in Bd.

Object description

Index	$202A_h$
Object Name	MODBUS RTU Baudrate
Object Code	VARIABLE
Data type	UNSIGNED32
Saveable	yes, category: communication
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	$00004B00_h$
Firmware Version	FIR-v1436
Change History	

202Ch MODBUS RTU Stop Bits

Funktion

This object contains the number of stop-bits of modbus interface.

Object description

Index	$202C_h$
Object Name	MODBUS RTU Stop Bits
Object Code	VARIABLE
Data type	UNSIGNED8
Saveable	No
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	01_h
Firmware Version	FIR-v1436
Change History	Firmware Version FIR-v1540: Entry "Saveable" modified from "yes, category: communication" to "No". Firmware Version FIR-v1540: Table entry "Access" at sub-index 00 modified from "Read/write" to "Read only".

Description

Number of Stopbits	Value in object $202C_h$
1	0
2	2

202Dh MODBUS RTU Parity

Function

This object configures the parity and stop bits for MODBUS RTU.

Object description

Index	$202D_h$
Object Name	MODBUS RTU Parity
Object Code	VARIABLE
Data type	UNSIGNED8
Saveable	yes, category: communication
Savable	yes, category: communication
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	04_h
Firmware Version	FIR-v1540
Change History	

Description

The following values apply:

- Value "0x00": Party none, stop bits 2
- Value "0x04": Pairty Even, stop bits 1
- Value "0x06": Pairty odd, stop bits 1

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
Saveable	yes, category: tuning
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	• : 00000032_h • PD4-C5918M4204-E-01: 00000032_h • PD4-C6018L4204-E-01: 00000032_h • PD4-CB59M024035-E-01: 00000003_h • PD4-C5918L4204-E-01: 00000032_h • PD4-C5918M4204-KSAR2: 00000032_h

Firmware Version	FIR-v1426
Change History	Version FIR-v1422-B36464: Name entry changed from "Pole pair count" to "Pole Pair Count" Firmware Version FIR-v1540: Entry "Saveable" modified from "No" to "yes, category: tuning".

2031h Maximum Current

Function

If open-loop mode is selected and I^2t monitoring is not activated, the rated current from the motor datasheet should be entered here. (in mA as root mean square (RMS)). Is closed-loop mode selected and I^2t monitoring has been activated, it defines the peak current as RMS in mA.

Object description

Index	2031_h
Object Name	Maximum Current
Object Code	VARIABLE
Data type	UNSIGNED32
Saveable	yes, category: tuning
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	• : 00000000_h • PD4-C5918M4204-E-01: 00000834_h • PD4-C6018L4204-E-01: 00000834_h • PD4-CB59M024035-E-01: $00001F40_h$ • PD4-C5918L4204-E-01: 00000834_h • PD4-C5918M4204-KSAR2: 00000834_h
Firmware Version	FIR-v1426
Change History	Firmware Version FIR-v1614: Entry "Saveable" modified from "yes, category: application" to "yes, category: tuning". Firmware Version FIR-v1614: Entry "Object Name" modified from "Peak Current" to "Max Current".

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

Saveable	yes, category: tuning
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	• : $00030D40_h$ • PD4-C5918M4204-E-01: $00030D40_h$ • PD4-C6018L4204-E-01: $00030D40_h$ • PD4-CB59M024035-E-01: 00001770_h • PD4-C5918L4204-E-01: $00030D40_h$ • PD4-C5918M4204-KSAR2: $00030D40_h$
Firmware Version	FIR-v1426
Change History	Firmware Version FIR-v1614: Entry "Saveable" modified from "yes, category: application" to "yes, category: tuning".

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
Saveable	yes, category: application
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
Saveable	yes, category: application
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	• : 00006590_h • PD4-C5918M4204-E-01: $0000C92C_h$ • PD4-C6018L4204-E-01: $0000C92C_h$ • PD4-CB59M024035-E-01: 00007530_h • PD4-C5918L4204-E-01: $0000C92C_h$ • PD4-C5918M4204-KSAR2: $0000C92C_h$
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 **2034_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
Saveable	yes, category: application
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00002710_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
Saveable	yes, category: application
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 as root mean square (RMS) to which the current must be reduced when current reduction is activated in open loop and the motor is idling.

Object description

Index	2037$_h$
Object Name	Open Loop Current Reduction Value/factor
Object Code	VARIABLE
Data type	INTEGER32
Saveable	yes, category: application
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	FFFFFFCE$_h$
Firmware Version	FIR-v1426
Change History	

Description

Value of 2037_h greater/equal 0 and smaller than value in 2031_h

The current is reduced to the value entered in object **2037_h**. The value is in mA and used as root mean square (RMS).

Value of 2037_h is between -1 and -100

The value in **2037_h** is interpreted as the percentage reduction factor relative to **2031_h**.

Example: Object **2031_h** was set to 4200 mA. The value -60 in object **2037_h** will reduce the current as root means square (RMS) up to 60 % in comparison to **2031_h**. As a result you'll get a current reduction down to **2031_h** * (**2037_h** + 100) / 100 = 1680 mA.

The value "-100" in **2037_h** as a further example will reduce the current reduction to 0 mA.

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
Saveable	No
Firmware Version	FIR-v1426
Change History	Firmware Version FIR-v1504: Table entry "PDO Mapping" at sub-index 01 modified from "No" to "TX - PDO". Firmware Version FIR-v1504: Table entry "PDO Mapping" at sub-index 02 modified from "No" to "TX - PDO". Firmware Version FIR-v1504: Table entry "PDO Mapping" at sub-index 03 modified from "No" to "TX - PDO". Firmware Version FIR-v1504: Table entry "PDO Mapping" at sub-index 04 modified from "No" to "TX - PDO".

Value description

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

Sub-index	01_h
Name	I_d
Data type	INTEGER32
Access	Read only

PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h

Sub-index	02_h
Name	I_q
Data type	INTEGER32
Access	Read only
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h

Sub-index	03_h
Name	I_a
Data type	INTEGER32
Access	Read only
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h

Sub-index	04_h
Name	I_b
Data type	INTEGER32
Access	Read only
PDO Mapping	TX - PDO
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	INTEGER32
Saveable	yes, category: application
Access	
PDO Mapping	
Admissible Values	
Specified Value	
Firmware Version	FIR-v1426
Change History	Firmware Version FIR-v1540: Amount of subentries has changed from 4 to 3.

	Firmware Version FIR-v1540: Entry "Name" modified from "Period Of Blocking" to "Block Detection Time". Firmware Version FIR-v1614: Entry "Data Type" modified from "UNSIGNED32" to "INTEGER32". Firmware Version FIR-v1614: Entry "Saveable" modified from "No" to "yes, category: application". Firmware Version FIR-v1614: Entry "Data type" modified from "UNSIGNED32" to "INTEGER32". Firmware Version FIR-v1614: Entry "Data type" modified from "UNSIGNED32" to "INTEGER32".

Value description

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

Sub-index	01_h
Name	Minimum Current For Block Detection
Data type	INTEGER32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	• : $FFFFFFBA_h$ • PD4-C5918M4204-E-01: $000004EC_h$ • PD4-C6018L4204-E-01: $000004EC_h$ • PD4-CB59M024035-E-01: $000015E0_h$ • PD4-C5918L4204-E-01: $000004EC_h$ • PD4-C5918M4204-KSAR2: $000004EC_h$

Sub-index	02_h
Name	Block Detection Time
Data type	INTEGER32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	$000000C8_h$

Description

The subentries have the following function:

- 01_h: Specifies the current limit value from which blocking is to be detected. Positive number values setting the current limit in mA, negative number values a percentage of object **2031**$_h$:01_h. Example: The value "1000" is equal to 1000 mA (=1 A), the value "-70" is equal to 70% of **2031**$_h$.
- 02_h: Specifies the time in ms that the motor is nevertheless still to travel against the block after block detection.

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**$_h$:01$_h$ and **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
Saveable	yes, category: tuning
Firmware Version	FIR-v1426
Change History	Firmware Version FIR-v1512: Entry "Savable" modified from "No" to "yes, category: application". Firmware Version FIR-v1614: Entry "Saveable" modified from "yes, category: application" to "yes, category: tuning".

Value description

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

Sub-index	01$_h$
Name	Nominal Current
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	• : 00000000$_h$ • PD4-C5918M4204-E-01: 00000000$_h$ • PD4-C6018L4204-E-01: 00000000$_h$ • PD4-CB59M024035-E-01: 00000FA0$_h$ • PD4-C5918L4204-E-01: 00000000$_h$ • PD4-C5918M4204-KSAR2: 00000000$_h$

Sub-index	02_h
Name	Maximum Duration Of Peak Current
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	• : 00000000_h • PD4-C5918M4204-E-01: 00000000_h • PD4-C6018L4204-E-01: 00000000_h • PD4-CB59M024035-E-01: $000003E8_h$ • PD4-C5918L4204-E-01: 00000000_h • PD4-C5918M4204-KSAR2: 00000000_h

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

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

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

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

Sub-index	07_h
Name	ActualResistance
Data type	UNSIGNED32

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

Description

The subentries are divided into two groups: sub-index 01_h and 02_h holding parameter for controlling, sub-index 03_h to 06_h are status values. The functions are as follows:

- 01_h: Specifies the nominal current as root mean square (RMS) in mA, must be smaller than the peak 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 as root mean square (RMS) 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

203Dh Torque Window

Function

Specifies a symmetrical range relative to the target torque within which the target is considered to be reached.

If the value of the torque window is $FFFFFFFF_h$, the torque window control is switched off, the bit "target reached" in object **6041**$_h$ (Statusword) will never get set..

Object description

Index	$203D_h$
Object Name	Torque Window
Object Code	VARIABLE
Data type	UNSIGNED16
Saveable	yes, category: application
Savable	No
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	0000_h
Firmware Version	FIR-v1540
Change History	Firmware Version FIR-v1614: Entry "Saveable" modified from "No" to "yes, category: application".

203Eh Torque Window Time

Function

For this time period in milliseconds, the actual torque must be within the "Torque Window" (**203D**$_h$) for the target velocity to be considered as reached.

Object description

Index	$203E_h$
Object Name	Torque Window Time
Object Code	VARIABLE
Data type	UNSIGNED16
Saveable	yes, category: application
Savable	No
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	0000_h
Firmware Version	FIR-v1540
Change History	Firmware Version FIR-v1614: Entry "Saveable" modified from "No" to "yes, category: application".

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
Saveable	yes, category: tuning
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h
Firmware Version	FIR-v1426
Change History	Firmware Version FIR-v1540: Entry "Saveable" modified from "No" to "yes, category: tuning".

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
Saveable	yes, category: tuning
Firmware Version	FIR-v1426
Change History	Firmware Version FIR-v1540: Entry "Saveable" modified from "No" to "yes, category: tuning".

Value description

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

Sub-index	01_h
Name	Parameter 1
Data type	INTEGER32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

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

Sub-index	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
Saveable	yes, category: tuning
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	• : 00000000_h • PD4-C5918M4204-E-01: 00001000_h • PD4-C6018L4204-E-01: 00001000_h • PD4-CB59M024035-E-01: 00001000_h • PD4-C5918L4204-E-01: 00001000_h • PD4-C5918M4204-KSAR2: 00001000_h
Firmware Version	FIR-v1426
Change History	Firmware Version FIR-v1540: Entry "Saveable" modified from "No" to "yes, category: tuning".

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.

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
Saveable	yes, category: application
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
Saveable	yes, category: application
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
Saveable	yes, category: application
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
Saveable	yes, category: tuning
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h
Firmware Version	FIR-v1426
Change History	Firmware Version FIR-v1614: Entry "Saveable" modified from "yes, category: application" to "yes, category: tuning".

Description

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

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

VOLT

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

TYPE

Defines the type of the encoder. This bit has to be set to the value "0" if a differential encoder is used. For a single ended encoder, the bit has to be set to the value "1".

205Ah Encoder Boot Value

Function

This object has only a functional purpose, if an absolute encoder is used: From this object, the encoder position (in user units) can be read, which was read from the absolute encoder initially at power up of the controller. This object is always 0 if no absolute encoder is used.

Object description

Index	$205A_h$
Object Name	Encoder Boot Value
Object Code	VARIABLE
Data type	UNSIGNED32

Saveable	No
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h
Firmware Version	FIR-v1446
Change History	Firmware Version FIR-v1512: Table entry "Access" at sub-index 00 modified from "Read/write" to "Read only".

205Bh Clock Direction Or Clockwise/Counter Clockwise Mode

Function

With this object the clock/direction-mode (value = "0") can be switched to clockwise/counterclockwise mode (value = "1").

Object description

Index	$205B_h$
Object Name	Clock Direction Or Clockwise/Counter Clockwise Mode
Object Code	VARIABLE
Data type	UNSIGNED32
Saveable	yes, category: application
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h
Firmware Version	FIR-v1504
Change History	

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
Saveable	yes, category: application
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
Saveable	yes, category: application
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
Saveable	yes, category: application
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
Saveable	yes, category: application
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
Saveable	yes, category: application
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
Saveable	yes, category: application
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
Saveable	yes, category: application

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.

2067h Jerk Limit (internal)

Function

This object controls the calculation for the real jerk limit, this limit is deactivated if this object is set to the value "0".

The following formula describes the calculation of the real jerk limit:

$$\text{Jerk Limit Real} = \frac{\text{Jerk limit internal } (2067_h) \text{ x Jerk denominator } (2066_h)}{2048 \text{ x Jerk numerator } (2065_h) \text{ x Pole Pair Count } (2030_h)}$$

Object description

Index	2067_h
Object Name	Jerk Limit (internal)
Object Code	VARIABLE
Data type	UNSIGNED32
Saveable	yes, category: application
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00989680_h
Firmware Version	FIR-v1450
Change History	

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
Saveable	yes, category: application
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h
Firmware Version	FIR-v1426
Change History	

2101h Fieldbus Module Availability

Function

Shows the type of mounted field bus module.

Object description

Index	2101_h
Object Name	Fieldbus Module Availability
Object Code	VARIABLE
Data type	UNSIGNED32
Saveable	No
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00190001_h
Firmware Version	FIR-v1426
Change History	Firmware Version FIR-v1426: Entry "Data Type" modified from "INTEGER32" to "UNSIGNED32" Firmware Version FIR-v1626: Entry "Object Name" modified from "Fieldbus Module" to "Fieldbus Module Availability".

Description

The bits 0 to 15 are representing the physical interface, the bits 16 to 31 the used protocol (if necessary).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
													E-IP	MTCP	MRTU

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									SPI	E-CAT	E-NET	CAN	RS232	RS485	USB

USB

Value = "1": An USB interface is available.

RS-485

Value = "1": A RS-485 interface is available.

RS-232

Value = "1": A RS-232 interface is available.

CAN
Value = "1": A CANopen interface is available.

E-NET
Value = "1": An EtherNET interface is available.

E-CAT
Value = "1": An EtherCAT interface is available.

SPI
Value = "1": A SPI interface is available.

MRTU
Value = "1": The protocol used is Modbus RTU.

MTCP
Value = "1": The protocol used is Modbus TCP

E-IP
Value = "1": The protocol used is EtherNet/IP™

2102h Fieldbus Module Control

Function

With this object certain field busses (physical interfaces and protocols) can get activated or deactivated.

Object description

Index	2102_h
Object Name	Fieldbus Module Control
Object Code	VARIABLE
Data type	UNSIGNED32
Saveable	yes, category: communication
Savable	yes, category: application
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00090001_h
Firmware Version	FIR-v1540
Change History	Firmware Version FIR-v1626: Entry "Saveable" modified from "yes, category: application" to "yes, category: communication".

Description

In object **2103**$_h$:1_h all physical interfaces and protocols are listed, which are able to get activated or deactivated. These object can get switched here. The actual state of the activated field busses are punt into **2103**$_h$:2_h.

Thereby the following distribution of the bits apply:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
													E-IP	MTCP	MRTU

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									SPI	E-CAT	E-NET	CAN	RS232	RS485	USB

USB
USB interface

RS-485
RS-485 interface

RS-232
RS-232 interface

CAN
CANopen interface

E-NET
EtherNET interface

E-CAT
EtherCAT interface

SPI
SPI interface

MRTU
Modbus RTU protocol

MTCP
Modbus TCP protocol

E-IP
EtherNet/IP™ protocol

2103h Fieldbus Module Status

Function

Displays the active field busses.

Object description

Index	2103_h
Object Name	Fieldbus Module Status
Object Code	ARRAY
Data type	UNSIGNED32
Saveable	No
Savable	No
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	

Firmware Version	FIR-v1540
Change History	

Value description

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

Sub-index	01_h
Name	Fieldbus Module Disable Mask
Data type	UNSIGNED32
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Sub-index	02_h
Name	Fieldbus Module Enabled
Data type	UNSIGNED32
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Description

Sub-index 1 (Fieldbus Module Disable Mask): in this entry all physically interfaces and protocols are shown able to get activated or deactivated. The value "1" means, that this fieldbus is able to get activated or deactivated.

Sub-index 2 (Fieldbus Module Enabled): This object lists all currently active physically interfaces and protocols. The value "1" means, that this fieldbus is active.

The following distribution of the bits for sub-index 1 and 2 apply:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
													E-IP	MTCP	MRTU

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									SPI	E-CAT	E-NET	CAN	RS232	RS485	USB

USB

USB interface

RS-485

RS-485 interface

RS-232
RS-232 interface

CAN
CANopen interface

E-NET
EtherNET interface

E-CAT
EtherCAT interface

SPI
SPI interface

MRTU
Modbus RTU protocol

MTCP
Modbus TCP protocol

E-IP
EtherNet/IP™ protocol

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
Saveable	yes, category: application
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h
Firmware Version	FIR-v1426
Change History	Firmware Version FIR-v1626: Entry "Saveable" modified from "No" to "yes, category: application".

Description

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

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

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

Description

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

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

ACT

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

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

Saveable	yes, category: application
Firmware Version	FIR-v1426
Change History	Firmware Version FIR-v1626: Entry "Saveable" modified from "No" to "yes, category: application".

Value description

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

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

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

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

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

Sub-index	05_h
Name	Sample Value #5

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

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

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

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

Description

Each sub-index (1-8) describes a mapped object.

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Index [16]															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SubIndex [8]								Length [8]							

Index [16]

Contains the index of the object to be mapped

Sub-index [8]

Contains the sub-index 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
Saveable	No
Firmware Version	FIR-v1426
Change History	Firmware Version FIR-v1626: Table entry "Access" at subindex 01 modified from "Read/write" to "Read only". Firmware Version FIR-v1626: Table entry "Access" at subindex 02 modified from "Read/write" to "Read only". Firmware Version FIR-v1626: Table entry "Access" at subindex 03 modified from "Read/write" to "Read only".

Value description

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

Sub-index	01_h
Name	Sample Buffer Size
Data type	UNSIGNED32
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Sub-index	02_h
Name	Sample Buffer Watermark
Data type	UNSIGNED32
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Sub-index	03_h
Name	Sample Tick

Data type	UNSIGNED32
Access	Read only
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
Saveable	yes, category: application
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000001_h
Firmware Version	FIR-v1426
Change History	Firmware Version FIR-v1626: Entry "Saveable" modified from "No" to "yes, category: application".

2300h NanoJ Control

Function

Controls the execution of a user program.

Object description

Index	2300_h
Object Name	NanoJ Control
Object Code	VARIABLE
Data type	UNSIGNED32
Saveable	yes, category: application
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	

Specified Value	00000000_h
Firmware Version	FIR-v1426
Change History	Firmware Version FIR-v1436: Entry "Object Name" modified from "VMM Control" to "NanoJ Control".

Description

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

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

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.

Note
The launch of the NanoJ program might be delayed up to 200ms.

TIM

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

2301h NanoJ Status

Function

Shows the operating state of the user program.

Object description

Index	2301_h
Object Name	NanoJ Status
Object Code	VARIABLE
Data type	UNSIGNED32
Saveable	No
Access	Read only
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h
Firmware Version	FIR-v1426
Change History	Firmware Version FIR-v1436: Entry "Object Name" modified from "VMM Status" to "NanoJ Status".

Description

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
													ERR	RES	RUN

Status [3]

Specifies the actual status of the VMM.

RUN

Value = "0": Program has been stopped, value = "1": Program is running

RES

Reserved.

ERR

Program was closed with an error. The cause of the error can be read out in object **2302**$_h$.

2302h NanoJ Error Code

Function

Indicates which error occurred when the user program was executed.

Object description

Index	2302_h
Object Name	NanoJ Error Code
Object Code	VARIABLE
Data type	UNSIGNED32
Saveable	No
Access	Read only
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h
Firmware Version	FIR-v1426
Change History	Firmware Version FIR-v1436: Entry "Object Name" modified from "VMM Error Code" to "NanoJ Error Code".

Description

Error codes during program execution:

Number	Description
0000_h	No error
0001_h	Firmware doesn't support the called function (yet)
0002_h	Not or wrong initialized pointer
0003_h	Illegal access on system resource
0004_h	Hardfault (internal error)
0005_h	Code is executed too long without a yield() or sleep() call

Number	Description
0006_h	Illegal access on system resource
0007_h	Too many variables on the stack (stack overflow)
0100_h	Bad NanoJ program file

Error codes regarding the access of objects:

Number	Description
1000_h	Access to an object that doesn't exist in object dictionary
1001_h	Write access to a read only (write protected) object

Object access errors:

Number	Description
1000_h	Access to an object that doesn't exist in object dictionary
1001_h	Write access to a read only (write protected) object
1002_h	Internal file system fault

File system error codes when loading the user program:

Number	Description
10002_h	Internal file system fault
10003_h	Storage medium not ready
10004_h	File not found
10005_h	Directory not found
10006_h	Invalid file name/directory name
10008_h	Access to file not possible
10009_h	Invalid file/directory object
$1000A_h$	Storage medium is write protected
$1000B_h$	Invalid drive number
$1000C_h$	Working range of drive is invalid
$1000D_h$	No valid file system on drive
$1000E_h$	Creation of the file system has failed
$1000F_h$	Access not possible within required time
10010_h	Access was rejected

230Fh Uptime Seconds

Function

This object contains the power-on hours since the last power-on in seconds.

Note
This object won't get saved, the counting will start with "0" after power-on.

Object description

Index	$230F_h$
Object Name	Uptime Seconds
Object Code	VARIABLE
Data type	UNSIGNED32

Saveable	No
Access	Read only
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h
Firmware Version	FIR-v1436
Change History	

2310h NanoJ 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	NanoJ Input Data Selection
Object Code	RECORD
Data type	PDO_MAPPING
Saveable	yes, category: application
Firmware Version	FIR-v1426
Change History	Amount of subentries has changed from 2 to 17 Firmware Version FIR-v1436: Entry "Object Name" modified from "VMM Input Data Selection" to "NanoJ Input Data Selection".

Value description

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

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

Sub-index	02_h
Name	Mapping #2
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No

Admissible Values	
Specified Value	00000000_h

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

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

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

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

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

Sub-index	08_h
Name	Mapping #8
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	

Specified Value	00000000_h

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

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

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

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

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

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

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

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

Description

Each sub-index (1-16) describes a mapped object.

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Index [16]															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SubIndex [8]								Length [8]							

Index [16]

Contains the index of the object to be mapped

Sub-index [8]

Contains the sub-index of the object to be mapped

Length [8]

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

2320h NanoJ 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	NanoJ Output Data Selection
Object Code	RECORD
Data type	PDO_MAPPING
Saveable	yes, category: application
Firmware Version	FIR-v1426
Change History	Amount of subentries has changed from 2 to 17

	Firmware Version FIR-v1436: Entry "Object Name" modified from "VMM Output Data Selection" to "NanoJ Output Data Selection".

Value description

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

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

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

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

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

Sub-index	05_h
Name	Mapping #5
Data type	UNSIGNED32
Access	Read/write

PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

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

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

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

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

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

Sub-index	$0B_h$
Name	Mapping #11
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No

Admissible Values	
Specified Value	00000000_h

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

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

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

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

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

Description

Each sub-index (1-16) describes a mapped object.

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Index [16]															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SubIndex [8]								Length [8]							

Index [16]

Contains the index of the object to be mapped

Sub-index [8]

Contains the sub-index of the object to be mapped

Length [8]

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

2330h NanoJ 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	NanoJ In/output Data Selection
Object Code	RECORD
Data type	PDO_MAPPING
Saveable	yes, category: application
Firmware Version	FIR-v1426
Change History	Amount of subentries has changed from 2 to 17 Firmware Version FIR-v1436: Entry "Object Name" modified from "VMM In/output Data Selection" to "NanoJ In/output Data Selection".

Value description

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

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

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

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

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

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

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

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

Sub-index	08_h

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

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

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

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

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

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

Sub-index	$0E_h$
Name	Mapping #14

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

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

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

Description

Each sub-index (1-16) describes a mapped object.

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Index [16]															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SubIndex [8]								Length [8]							

Index [16]

Contains the index of the object to be mapped

Sub-index [8]

Contains the sub-index of the object to be mapped

Length [8]

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

2400h NanoJ 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	NanoJ Inputs

Object Code	ARRAY
Data type	INTEGER32
Saveable	No
Firmware Version	FIR-v1426
Change History	Amount of subentries has changed from 2 to 33 Firmware Version FIR-v1436: Entry "Object Name" modified from "VMM Inputs" to "NanoJ Inputs".

Value description

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

Sub-index	01_h - 20_h
Name	NanoJ Input #1 - #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.

2410h NanoJ Init Parameters

Function

This object works exactly like the object **2400_h**, except that this object is persistent.

Object description

Index	2410_h
Object Name	NanoJ Init Parameters
Object Code	ARRAY
Data type	INTEGER32
Saveable	yes, category: application
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	
Firmware Version	FIR-v1450
Change History	

Value description

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

Sub-index	01_h - 20_h
Name	NanoJ Init Parameter #1 - #32
Data type	INTEGER32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

2500h NanoJ 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	NanoJ Outputs
Object Code	ARRAY
Data type	INTEGER32
Saveable	No
Firmware Version	FIR-v1426
Change History	Amount of subentries has changed from 2 to 33 Firmware Version FIR-v1436: Entry "Object Name" modified from "VMM Outputs" to "NanoJ Outputs".

Value description

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

Sub-index	01_h - 20_h
Name	NanoJ Output #1 - #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 NanoJ Debug Output

Function

This object contains debug outputs for a user program.

Object description

Index	2600_h
Object Name	NanoJ Debug Output
Object Code	ARRAY
Data type	UNSIGNED8
Saveable	No
Firmware Version	FIR-v1426
Change History	Amount of subentries has changed from 2 to 65 Firmware Version FIR-v1436: Entry "Object Name" modified from "VMM Debug Output" to "NanoJ Debug Output".

Value description

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

Sub-index	01_h - 40_h
Name	Value #1 - #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.

2700h User Storage Area

Function

DANGER

The motor has to stand still during the process of saving and is not allowed to get started while saving.

In this object up to 8 16 Bit values from NanoJ program can be stored permanently. These data are available even after a restart of the controller.

Set sub-index 1 to the value "1", the data will get stored and reloaded after power-up.

Object description

Index	2700_h
Object Name	User Storage Area
Object Code	RECORD
Data type	USER_STORAGE_AREA
Saveable	No
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	
Firmware Version	FIR-v1426
Change History	Firmware Version FIR-v1426: Amount of subentries has changed from 22 to 10. Firmware Version FIR-v1446: Entry "Name" modified from "Storage Control Word" to "Highest Sub-index Supported". Firmware Version FIR-v1446: Table entry "Access" at sub-index 00 modified from "Read/write" to "Read only".

Value description

Sub-index	00_h
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	09_h

Sub-index	01_h
Name	Storage Control Word
Data type	UNSIGNED8
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Sub-index	02_h
Name	Storage #1
Data type	UNSIGNED16
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	0000_h

Sub-index	03_h
Name	Storage #2
Data type	UNSIGNED16
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	0000_h

Sub-index	04_h
Name	Storage #3
Data type	UNSIGNED16
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	0000_h

Sub-index	05_h
Name	Storage #4
Data type	UNSIGNED16
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	0000_h

Sub-index	06_h
Name	Storage #5
Data type	UNSIGNED16
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	0000_h

Sub-index	07_h
Name	Storage #6
Data type	UNSIGNED16
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	0000_h

Sub-index	08_h

Name	Storage #7
Data type	UNSIGNED16
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	0000_h

Sub-index	09_h
Name	Storage #8
Data type	UNSIGNED16
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	0000_h

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
Saveable	yes, category: drive
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	• : 00000000_h • PD4-C5918M4204-E-01: 00000000_h • PD4-C6018L4204-E-01: 00000000_h • PD4-CB59M024035-E-01: 00000040_h • PD4-C5918L4204-E-01: 00000000_h • PD4-C5918M4204-KSAR2: 00000000_h
Firmware Version	FIR-v1426
Change History	Firmware Version FIR-v1540: Entry "Saveable" modified from "yes, category: application" to "yes, category: drive".

Description

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
									BLDC	Torque		CurRed	Brake	VoS	CL/OL

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)

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
Saveable	yes, category: application
Firmware Version	FIR-v1426
Change History	

Value description

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

Sub-index	01$_h$
Name	Commutation
Data type	INTEGER32

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

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

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

Sub-index	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**

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
Saveable	yes, category: application
Firmware Version	FIR-v1426
Change History	

Value description

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

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

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

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

Sub-index	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
Saveable	yes, category: application
Firmware Version	FIR-v1426
Change History	Amount of subentries has changed from 9 to 11
	Firmware Version FIR-v1626: Entry "Name" modified from "S_P" to "Position Loop, Proportional Gain (closed Loop)".
	Firmware Version FIR-v1626: Entry "Name" modified from "S_I" to "Position Loop, Integral Gain (closed Loop)".
	Firmware Version FIR-v1626: Entry "Name" modified from "V_P" to "Velocity Loop, Proportional Gain (closed Loop)".
	Firmware Version FIR-v1626: Entry "Name" modified from "V_I" to "Velocity Loop, Integral Gain (closed Loop)".
	Firmware Version FIR-v1626: Entry "Name" modified from "Id_P" to "Flux Current Loop, Proportional Gain (closed Loop)".
	Firmware Version FIR-v1626: Entry "Name" modified from "Id_I" to "Flux Current Loop, Integral Gain (closed Loop)".
	Firmware Version FIR-v1626: Entry "Name" modified from "Iq_P" to "Torque Current Loop, Proportional Gain (closed Loop)".
	Firmware Version FIR-v1626: Entry "Name" modified from "Iq_I" to "Torque Current Loop, Integral Gain (closed Loop)".

	Firmware Version FIR-v1626: Entry "Name" modified from "I_P" to "Torque Current Loop, Proportional Gain (dspDrive - Stepper Motor, Open Loop)". Firmware Version FIR-v1626: Entry "Name" modified from "I_I" to "Torque Current Loop, Integral Gain (dspDrive - Stepper Motor, Open Loop)".

Value description

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

Sub-index	01_h
Name	Position Loop, Proportional Gain (closed Loop)
Data type	INTEGER32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	• : 00000800_h • PD4-C5918M4204-E-01: 00002710_h • PD4-C6018L4204-E-01: 00000800_h • PD4-CB59M024035-E-01: 00007530_h • PD4-C5918L4204-E-01: 00002710_h • PD4-C5918M4204-KSAR2: 00002710_h

Sub-index	02_h
Name	Position Loop, Integral Gain (closed Loop)
Data type	INTEGER32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Sub-index	03_h
Name	Velocity Loop, Proportional Gain (closed Loop)
Data type	INTEGER32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	• : $00002EE0_h$ • PD4-C5918M4204-E-01: $00004E20_h$ • PD4-C6018L4204-E-01: $00001B58_h$ • PD4-CB59M024035-E-01: $0000EA60_h$ • PD4-C5918L4204-E-01: $00004E20_h$

	• PD4-C5918M4204-KSAR2: $00004E20_h$

Sub-index	04_h
Name	Velocity Loop, Integral Gain (closed Loop)
Data type	INTEGER32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	• : $0000001E_h$ • PD4-C5918M4204-E-01: 00000064_h • PD4-C6018L4204-E-01: 00000004_h • PD4-CB59M024035-E-01: $000001F4_h$ • PD4-C5918L4204-E-01: 00000064_h • PD4-C5918M4204-KSAR2: 00000064_h

Sub-index	05_h
Name	Flux Current Loop, Proportional Gain (closed Loop)
Data type	INTEGER32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	• : $000668A0_h$ • PD4-C5918M4204-E-01: $0007A120_h$ • PD4-C6018L4204-E-01: $000668A0_h$ • PD4-CB59M024035-E-01: $000061A8_h$ • PD4-C5918L4204-E-01: $0007A120_h$ • PD4-C5918M4204-KSAR2: $0007A120_h$

Sub-index	06_h
Name	Flux Current Loop, Integral Gain (closed Loop)
Data type	INTEGER32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	• : $00002EE0_h$ • PD4-C5918M4204-E-01: 00001388_h • PD4-C6018L4204-E-01: $00002EE0_h$ • PD4-CB59M024035-E-01: $00000BB8_h$ • PD4-C5918L4204-E-01: 00001388_h • PD4-C5918M4204-KSAR2: 00001388_h

Sub-index	07_h
Name	Torque Current Loop, Proportional Gain (closed Loop)
Data type	INTEGER32
Access	Read/write
PDO Mapping	No
Admissible Values	

Specified Value	• : $000668A0_h$ • PD4-C5918M4204-E-01: $0007A120_h$ • PD4-C6018L4204-E-01: $000668A0_h$ • PD4-CB59M024035-E-01: $000061A8_h$ • PD4-C5918L4204-E-01: $0007A120_h$ • PD4-C5918M4204-KSAR2: $0007A120_h$

Sub-index	08_h
Name	Torque Current Loop, Integral Gain (closed Loop)
Data type	INTEGER32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	• : $00002EE0_h$ • PD4-C5918M4204-E-01: 00001388_h • PD4-C6018L4204-E-01: $00002EE0_h$ • PD4-CB59M024035-E-01: $00000BB8_h$ • PD4-C5918L4204-E-01: 00001388_h • PD4-C5918M4204-KSAR2: 00001388_h

Sub-index	09_h
Name	Torque Current Loop, Proportional Gain (dspDrive - Stepper Motor, Open Loop)
Data type	INTEGER32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	• : $0003A980_h$ • PD4-C5918M4204-E-01: 00027100_h • PD4-C6018L4204-E-01: 00027100_h • PD4-CB59M024035-E-01: 00000000_h • PD4-C5918L4204-E-01: 00027100_h • PD4-C5918M4204-KSAR2: 00027100_h

Sub-index	$0A_h$
Name	Torque Current Loop, Integral Gain (dspDrive - Stepper Motor, Open Loop)
Data type	INTEGER32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	• : $0000AFC8_h$ • PD4-C5918M4204-E-01: $000055F0_h$ • PD4-C6018L4204-E-01: 00002710_h • PD4-CB59M024035-E-01: 00000000_h • PD4-C5918L4204-E-01: $000055F0_h$ • PD4-C5918M4204-KSAR2: $000055F0_h$

Description

- Sub-index 00_h: Number of entries
- Sub-index 01_h: Proportional value of the S (position) control
- Sub-index 02_h: Integral value of the S (position) control
- Sub-index 03_h: Proportional value of the V (velocity) control
- Sub-index 04_h: Integral value of the V (velocity) control
- Sub-index 05_h: (Closed Loop) Proportional value of the current controller for the field-forming component
- Sub-index 06_h: (Closed Loop) Integral value of the current controller for the field-forming component
- Sub-index 07_h: (Closed Loop) Proportional value of the current controller for the torque-forming component
- Sub-index 08_h: (Closed Loop) Integral value of the current controller for the torque-forming component
- Sub-index 09_h: (Open Loop) Proportional value of the current controller for the torque-forming component
- Sub-index $0A_h$: (Open Loop) Integral value of the current controller for the torque-forming component

3212h Motor Drive Flags

Function

This object determines, whether the output voltage for the motor is active in state "switched on" of the DS 402 Statemachine or not. Furthermore the direction of rotation of the rotating field can get changed.

Note
Changes in sub-index 02 are valid after a reset of the controller.

Object description

Index	3212_h
Object Name	Motor Drive Flags
Object Code	ARRAY
Data type	INTEGER8
Saveable	yes, category: application
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	
Firmware Version	FIR-v1450
Change History	Firmware Version FIR-v1512: Amount of subentries has changed from 2 to 3. Firmware Version FIR-v1540: Amount of subentries has changed from 3 to 4.

Value description

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

PDO Mapping	No
Admissible Values	
Specified Value	03_h

Sub-index	01_h
Name	Enable Legacy Power Mode
Data type	INTEGER8
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Sub-index	02_h
Name	Override Field Inversion
Data type	INTEGER8
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Sub-index	03_h
Name	Do Not Touch Controller Settings
Data type	INTEGER8
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Description

For the sub-index 01_h are valid values:

- Value = "0": The output voltage for the motor (PWM) at state "Switched On" of " **DS402 Power State machine**" is set fix to 50%, no holding torque is build up.
- Value = "1": The output voltage for the motor (PWM) at state "Switched On" of " **DS402 Power State machine**" is controlled active by the controller, holding torque is build up. The motor will kept still.

For the sub-index 02_h are valid values:

- Value = "0": default values of the firmware will be used
- Value = "1": force a non inverting of the rotating field
- Value = "-1": force a inverting of the rotating field

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
Saveable	No
Firmware Version	FIR-v1426
Change History	

Value description

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

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

Sub-index	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
Saveable	yes, category: application
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h
Firmware Version	FIR-v1426
Change History	

Description

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
														AC2	AC1

In general: If a bit is set to 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

3225h Analogue Inputs Switches

Function

This object contains either the adjusted CANopen nodeId of the rotary switch or the positions of the DIP switches

Object description

Index	3225_h
Object Name	Analogue Inputs Switches
Object Code	ARRAY
Data type	UNSIGNED16
Saveable	No
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	
Firmware Version	FIR-v1436
Change History	Firmware Version FIR-v1436: Table entry "PDO Mapping" at sub-index 01 modified from "RX - PDO" to "TX - PDO".

Value description

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

Sub-index	01_h
Name	Analogue Input Switch1
Data type	UNSIGNED16
Access	Read only
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	0000_h

Description

In sub-index 1 the nodeId of the rotary switch(es) is registered if a CANopen interface is available to the controller.

When a DIP-Switch is fitted to the controller, the positions of the DIP switches are stored in sub-index 1. Bit 0 accords the switch 1, the value of the bit is "1" if the switch is set to "on".

3240h Digital Inputs Control

Function

This object can be used to manipulate the digital inputs as described in the " **Digital inputs and outputs**" section.

It applies for all the subindices:

- Bit 0 to 15 control the special functions.
- Bit 16 to 31 control the level of the outputs.

Object description

Index	3240_h
Object Name	Digital Inputs Control
Object Code	ARRAY
Data type	UNSIGNED32
Saveable	yes, category: application
Firmware Version	FIR-v1426
Change History	Version 1.0.3: Sub-index 01_h: The "Name" entry was changed from "Special Function Disable" to "Special Function Enable" Firmware Version FIR-v1512: Amount of subentries has changed from 8 to 9.

Value description

Sub-index	00_h

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

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

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

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

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

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

Sub-index	06_h
Name	Input Range Select

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

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

Sub-index	08_h
Name	Routing Enable
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 sub-index switches on the special functions of the respective input if the bit has the value "1".
- 02_h: This sub-index inverts the logic of an input if the respective input has the value "1".
- 03_h: This sub-index 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 sub-index specifies the input value to be forced.
- 05_h: This sub-index always contains the read, unmodified input value.
- 06_h: This sub-index switches the switching thresholds between 5 V (value "0" in the sub-index) and 24 V (value "1" in the sub-index) for all inputs at once.
- 07_h: This sub-index switches the inputs from a single ended (value "0" in the sub-index) to differential (value "1" in the sub-index) for all inputs at once.
- 08_h: This sub-index enables "Input Routing" when the value "1" is written to it.

3241h Digital Input Capture

Function

With this object the encoder position can autocratically get captured when a change of level at digital input 2 occurs.

Object description

Index	3241_h
Object Name	Digital Input Capture
Object Code	ARRAY
Data type	UNSIGNED32

Saveable	yes, category: application
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	
Firmware Version	FIR-v1446
Change History	

Value description

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

Sub-index	01_h
Name	Control
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Sub-index	02_h
Name	Capture Count
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Sub-index	03_h
Name	Value
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Sub-index	04_h
Name	Encoder Raw Value
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Description

- Sub-index 01_h: the type of level change can get selected:
  - Function deactivated: value "0"
  - On rising edge: value "1"
  - On falling edge: value "2"
  - On both edges: value "3"
- Sub-index 02_h: describes the number of counted level changes since the last start of the function; gets recessed to 0 when sub-index 01_h gets set to 1,2 or 3.
- Sub-index 03_h: Encoder position of the level change (in absolute user units taken from **6064_h**)
- Sub-index 04_h: Encoder value of the level change

3242h Digital Input Routing

Function

This object determines the sources of the input routing which will end up in the object **$60FD_h$**.

Object description

Index	3242_h
Object Name	Digital Input Routing
Object Code	ARRAY
Data type	UNSIGNED8
Saveable	yes, category: application
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	
Firmware Version	FIR-v1504
Change History	

Value description

Sub-index	00_h
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	24_h

Sub-index	01_h - 24_h
Name	Input Source #1 - #36
Data type	UNSIGNED8
Access	Read/write
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00_h

Description

The sub-index 01_h holds the source for the bit 0 of the object **60FD**. The sub-index 02_h holds the soure for bit 2 of the object **60FD** and so on.

The number written to a sub-index determines the source for the corresponding bit. The the follwoing table for all the possible sources.

Nummer		
dec	**hex**	**Signal source**
00	00	Signal is always "0"
01	01	Physical input 1
02	02	Physical input 2
03	03	Physical input 3
04	04	Physical input 4
05	05	Physical input 5
06	06	Physical input 6
07	07	Physical input 7
08	08	Physical input 8
09	09	Physical input 9
10	0A	Physical input 10
11	0B	Physical input 11
12	0C	Physical input 12
13	0D	Physical input 13
14	0E	Physical input 14
15	0F	Physical input 15
16	10	Physical input 16
65	41	Hall input "U"
66	42	Hall input "V"
67	43	Hall input "W"
68	44	Encoder input "A"
69	45	Encoder input "B"
70	46	Encoder input "Index"
71	47	USB power signal
72	48	Status ethernet active
73	49	DIP-Switch 1
74	4A	DIP-Switch 2
75	4B	DIP-Switch 3
76	4C	DIP-Switch 4
77	4D	DIP-Switch 5
78	4E	DIP-Switch 6
79	4F	DIP-Switch 7
80	50	DIP-Switch 8
128	80	Signal is always "1"
129	81	Inverted physical input 1
130	82	Inverted physical input 2
131	83	Inverted physical input 3
132	84	Inverted physical input 4
133	85	Inverted physical input 5
134	86	Inverted physical input 6

Nummer		
dec	**hex**	**Signal source**
135	87	Inverted physical input 7
136	88	Inverted physical input 8
137	89	Inverted physical input 9
138	8A	Inverted physical input 10
139	8B	Inverted physical input 11
140	8C	Inverted physical input 12
141	8D	Inverted physical input 13
142	8E	Inverted physical input 14
143	8F	Inverted physical input 15
144	90	Inverted physical input 16
193	C1	Inverted hall input "U"
194	C2	Inverted hall input "V"
195	C3	Inverted hall input "W"
196	C4	Inverted encoder input "A"
197	C5	Inverted encoder input "B"
198	C6	Inverted encoder input "Index"
199	C7	Inverted USB power signal
200	C8	Inverted status "Ethernet active"
201	C9	Inverted DIP-Switch 1
202	CA	Inverted DIP-Switch 2
203	CB	Inverted DIP-Switch 3
204	CC	Inverted DIP-Switch 4
205	CD	Inverted DIP-Switch 5
206	CE	Inverted DIP-Switch 6
207	CF	Inverted DIP-Switch 7
208	D0	Inverted DIP-Switch 8

3250h Digital Outputs Control

Function

This object can be used to control the digital outputs as described in the " **Digital inputs and outputs**" section.

It applies for all the subindices:

- Bit 0 to 15 control the special functions.
- Bit 16 to 31 control the level of the outputs.

Object description

Index	3250_h
Object Name	Digital Outputs Control
Object Code	ARRAY
Data type	UNSIGNED32
Saveable	yes, category: application
Firmware Version	FIR-v1426

Change History	Firmware Version FIR-v1426: Sub-index 01_h: Entry "Name" changed from "Special Function Disable" auf "Special Function Enable" Firmware Version FIR-v1446: Entry "Name" modified from "Special Function Enable" to "No Function". Firmware Version FIR-v1512: Amount of subentries has changed from 6 to 9.

Value description

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

Sub-index	01_h
Name	No Function
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

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

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

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

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

Sub-index	06_h
Name	Reserved1
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Sub-index	07_h
Name	Reserved2
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Sub-index	08_h
Name	Routing Enable
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 sub-index inverts the logic (from opener logic to closer logic)
- 03_h: This sub-index forces an output value if the bit has the value "1". The level of the output is defined in sub-index 4_h.
- 04_h: This sub-index 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 sub-index, the bit combination applied to the outputs is stored.
- 06_h and 07_h: reseved
- 08_h: This subentry enables "Output Routing" when the value "1" is written to it.

3252h Digital Output Routing

Function

This object assigns a signal source to a digital output and can be controlled with object **60FE**$_h$

Object description

Index	3252_h
Object Name	Digital Output Routing
Object Code	ARRAY
Data type	UNSIGNED16
Saveable	yes, category: application
Savable	yes, category: application
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	
Firmware Version	FIR-v1540
Change History	

Value description

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

Sub-index	01_h - 05_h
Name	Output Control #1 - #5
Data type	UNSIGNED16
Access	Read/write
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	1080_h

3320h Read Analogue Input

Function

Displays the momentary value of the analogue inputs in user units.

Object description

Index	3320_h
Object Name	Read Analogue Input
Object Code	ARRAY
Data type	INTEGER32
Saveable	No
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	

Firmware Version	FIR-v1426
Change History	

Value description

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

Sub-index	01_h
Name	Analogue Input 1
Data type	INTEGER32
Access	Read only
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h

Sub-index	02_h
Name	Analogue Input 2
Data type	INTEGER32
Access	Read only
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	00000000_h

Description

The user units are combined from offset (**3321**$_h$) and pre scaling value (**3322**$_h$). When both objects are left to the default settings, the value in **3320**$_h$ is given in the unint "ADC digits".

Formulars for converting from digits to the particular unig:

Voltage input: x digits * 10 V / 1024 digits

Current input: x digits * 20 mA / 1024 digits

For the sub entries apply:

- Sub-index 00_h: Amount of analogue inputs
- Sub-index 01_h: Analogue value 1
- Sub-index 02_h: Analogue value 2

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
Saveable	yes, category: application
Firmware Version	FIR-v1426
Change History	

Value description

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

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

Sub-index	02_h
Name	Analogue Input 2
Data type	INTEGER32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Description

- Sub-index 00_h: Number of offsets
- Sub-index 01_h: Offset for analog input 1
- Sub-index 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
Saveable	yes, category: application
Firmware Version	FIR-v1426
Change History	

Value description

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

Sub-index	01_h
Name	Analogue Input 1
Data type	INTEGER32
Access	Read/write
PDO Mapping	No
Admissible Values	All values allowed, except 0
Specified Value	00000001_h

Sub-index	02_h
Name	Analogue Input 2
Data type	INTEGER32
Access	Read/write
PDO Mapping	No
Admissible Values	All values allowed, except 0
Specified Value	00000001_h

Description

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

3502h MODBUS Rx PDO Mapping

Function

Objects for the rx mapping can get written in this object.

Object description

Index	3502_h
Object Name	MODBUS Rx PDO Mapping
Object Code	RECORD
Data type	PDO_MAPPING

Saveable	yes, category: communication
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	
Firmware Version	FIR-v1614
Change History	

Value description

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

Sub-index	01_h
Name	1st Object To Be Mapped
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	60400010_h

Sub-index	02_h
Name	2nd Object To Be Mapped
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00050008_h

Sub-index	03_h
Name	3rd Object To Be Mapped
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	60600008_h

Sub-index	04_h
Name	4th Object To Be Mapped
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	32020020_h

Sub-index	05_h
Name	5th Object To Be Mapped
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	$607A0020_h$

Sub-index	06_h
Name	6th Object To Be Mapped
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	60810020_h

Sub-index	07_h
Name	7th Object To Be Mapped
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	60420010_h

Sub-index	08_h
Name	8th Object To Be Mapped
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	$60FE0120_h$

Sub-index	09_h
Name	9th Object To Be Mapped
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Sub-index	$0A_h$
Name	10th Object To Be Mapped
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Sub-index	$0B_h$

Name	11th Object To Be Mapped
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Sub-index	$0C_h$
Name	12th Object To Be Mapped
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Sub-index	$0D_h$
Name	13th Object To Be Mapped
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Sub-index	$0E_h$
Name	14th Object To Be Mapped
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Sub-index	$0F_h$
Name	15th Object To Be Mapped
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Sub-index	10_h
Name	16th Object To Be Mapped
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

3602h MODBUS Tx PDO Mapping

Function

Objects for the tx mapping can get written in this object.

Object description

Index	3602_h
Object Name	MODBUS Tx PDO Mapping
Object Code	RECORD
Data type	PDO_MAPPING
Saveable	yes, category: communication
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	
Firmware Version	FIR-v1614
Change History	

Value description

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

Sub-index	01_h
Name	1st Object To Be Mapped
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	60410010_h

Sub-index	02_h
Name	2nd Object To Be Mapped
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00050008_h

Sub-index	03_h
Name	3rd Object To Be Mapped
Data type	UNSIGNED32
Access	Read/write

PDO Mapping	No
Admissible Values	
Specified Value	60610008_h

Sub-index	04_h
Name	4th Object To Be Mapped
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	60640020_h

Sub-index	05_h
Name	5th Object To Be Mapped
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	60440010_h

Sub-index	06_h
Name	6th Object To Be Mapped
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	$60FD0020_h$

Sub-index	07_h
Name	7th Object To Be Mapped
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Sub-index	08_h
Name	8th Object To Be Mapped
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Sub-index	09_h
Name	9th Object To Be Mapped
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No

Admissible Values	
Specified Value	00000000_h

Sub-index	$0A_h$
Name	10th Object To Be Mapped
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Sub-index	$0B_h$
Name	11th Object To Be Mapped
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Sub-index	$0C_h$
Name	12th Object To Be Mapped
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Sub-index	$0D_h$
Name	13th Object To Be Mapped
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Sub-index	$0E_h$
Name	14th Object To Be Mapped
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Sub-index	$0F_h$
Name	15th Object To Be Mapped
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	

Specified Value	00000000_h

Sub-index	10_h
Name	16th Object To Be Mapped
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

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
Saveable	yes, category: application
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	$FFFF_h$
Firmware Version	FIR-v1426
Change History	

Description

Value	Description
-32768 to -2	Reserved
-1	No reaction.
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

4012h HW Information

Function

This object shows information about the hardware.

Object description

Index	4012_h
Object Name	HW Information

Object Code	ARRAY
Data type	UNSIGNED32
Saveable	No
Savable	No
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	
Firmware Version	FIR-v1540
Change History	

Value description

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

Sub-index	01_h
Name	EEPROM Size In Bytes
Data type	UNSIGNED32
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	00000000_h

Description

Sub-index 01: Specifies the size of the connected EEPROM in byte. The value 0 implies "no EEPROM connected".

4040h Drive Serial Number

Function

This object contains the serial number of the motor controller.

Object description

Index	4040_h
Object Name	Drive Serial Number
Object Code	VARIABLE
Data type	VISIBLE_STRING
Saveable	No
Access	Read only
PDO Mapping	No
Admissible Values	

Specified Value	
Firmware Version	FIR-v1450
Change History	

603Fh Error Code

Function

This object contains the error code of the last error occurred.

This value corresponds to the lower 16 bit of the object **1003**$_h$. For the description of the error code consult the object **1003**$_h$, please.

Object description

Index	603F$_h$
Object Name	Error Code
Object Code	VARIABLE
Data type	UNSIGNED16
Saveable	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
Saveable	yes, category: application
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	0000$_h$
Firmware Version	FIR-v1426
Change History	Firmware Version FIR-v1626: Entry "Saveable" modified from "No" to "yes, category: application".

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 (Operation Mode Specific)

Meaning depends on the selected operating mode

FR (Fault Reset)

Resets an error (if possible)

HALT

Value = "1": Triggers a stop, available in the following modes:

- **Profile Position**
- **Velocity**
- **Profile Velocity**
- **Profile Torque**

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
Saveable	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.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLA		OMS [2]		ILA	TARG	REM	SYNC	WARN	SOD	QS	VE	FAULT	OE	SO	RTSO

RTSO (Ready To Switch On)

Value = "1": 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 = "0": 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
Saveable	yes, category: application
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	$00C8_h$
Firmware Version	FIR-v1426
Change History	Firmware Version FIR-v1626: Entry "Saveable" modified from "No" to "yes, category: application".

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
Saveable	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
Saveable	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
Saveable	yes, category: application
Firmware Version	FIR-v1426
Change History	

Value description

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

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

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

Description

Sub-index 1 contains the minimum speed.

Sub-index 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
Saveable	yes, category: application
Firmware Version	FIR-v1426
Change History	

Value description

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

Sub-index	01$_h$
Name	DeltaSpeed
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	000001F4$_h$

Sub-index	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.

Sub-index 01_h: Contains the speed change in steps per second (U32).

Sub-index 02_h: Contains the time change in seconds (U16).

6049h Vl Velocity Deceleration

Function

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

Object description

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

Value description

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

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

Sub-index	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
Saveable	yes, category: application
Firmware Version	FIR-v1426
Change History	

Value description

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

Sub-index	01_h
Name	DeltaSpeed
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00001388_h

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

Description

Sub-index 1 contains the speed change, and sub-index 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 VI Dimension Factor

Function

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

Object description

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

Value description

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

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

Sub-index	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 sub-index 1 is set to the value "1" and sub-index 2 is set to the value "60", the speed is indicated in revolutions per minute.

Otherwise, sub-index 1 contains the denominator (multiplier) and sub-index 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
Saveable	yes, category: application
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
Saveable	yes, category: application
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
Saveable	yes, category: application
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
Saveable	yes, category: application
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
Saveable	yes, category: application
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
Saveable	yes, category: application
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00_h
Firmware Version	FIR-v1426
Change History	Firmware Version FIR-v1626: Entry "Saveable" modified from "No" to "yes, category: application".

Description

Mode	Description
-128 to -2	Manufacturer-specific operation modes
-1	Clock/direction mode
0	No mode change/no mode assigned
1	Profile Position Mode
2	Velocity Mode
3	Profile Velocity Mode
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.

Object description

Index	6061_h
Object Name	Modes Of Operation Display
Object Code	VARIABLE
Data type	INTEGER8
Saveable	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
Saveable	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.

CAUTION
The value of this object is invalid in case the object for the encoder resolution (object **2052_h**) is set to the value "0".

Object description

Index	6063_h
Object Name	Position Actual Internal Value
Object Code	VARIABLE
Data type	INTEGER32
Saveable	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$_h$.

CAUTION
The value of this object is invalid in case the object for the encoder resolution (object **2052**$_h$) is set to the value "0" and the encoder is selected as source.

Object description

Index	6064$_h$
Object Name	Position Actual Value
Object Code	VARIABLE
Data type	INTEGER32
Saveable	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
Saveable	yes, category: application
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000100$_h$
Firmware Version	FIR-v1426
Change History	Firmware Version FIR-v1504: Entry "Saveable" modified from "No" to "yes, category: application".

Description

If the value of the following error window is set to "FFFFFFFF"$_h$, the following control is switched off.

If the difference between the actual position and the set position is so large that value of this object is exceeded, an error will be noted in the object **1003**$_h$. The deviation must be longer than the time in object **6066**$_h$.

In the object **3700**$_h$ a reaction for the following error can be set.

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
Saveable	yes, category: application
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	0064$_h$
Firmware Version	FIR-v1426
Change History	Firmware Version FIR-v1504: Entry "Saveable" modified from "No" to "yes, category: application".

Description

If the difference between the actual position and the set position is so large that the value of object **6065**$_h$ is exceeded, the bit 11 "Internal Limit Reached" in the object **6041**$_h$ (statusword) is set. The deviation must be longer than the time in this object.

In the object **3700**$_h$ a reaction for the following error can be set.

6067h Position Window

Function

Specifies a symmetrical range relative to the target position within which the target is considered to be reached.

If the value of the position window is FFFFFFFF$_h$, the position window control is switched off.

Object description

Index	6067$_h$
Object Name	Position Window
Object Code	VARIABLE
Data type	UNSIGNED32
Saveable	yes, category: application

Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	$0000000A_h$
Firmware Version	FIR-v1426
Change History	Firmware Version FIR-v1504: Entry "Saveable" modified from "No" to "yes, category: application".

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 as reached.

Object description

Index	6068_h
Object Name	Position Window Time
Object Code	VARIABLE
Data type	UNSIGNED16
Saveable	yes, category: application
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	0064_h
Firmware Version	FIR-v1426
Change History	Firmware Version FIR-v1504: Entry "Saveable" modified from "No" to "yes, category: application".

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

606Dh Velocity Window

Function

Specifies a symmetrical range relative to the target velocity within which the target is considered to be reached.

Object description

Index	$606D_h$
Object Name	Velocity Window
Object Code	VARIABLE
Data type	UNSIGNED16
Saveable	yes, category: application
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	$001E_h$
Firmware Version	FIR-v1426
Change History	Firmware Version FIR-v1614: Entry "Saveable" modified from "No" to "yes, category: application".

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

For this time period in milliseconds, the actual velocity must be within the "Velocity Window" (**606D**$_h$) for the target velocity to be considered as reached.

Object description

Index	606E$_h$
Object Name	Velocity Window Time
Object Code	VARIABLE
Data type	UNSIGNED16
Saveable	yes, category: application
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	0000$_h$
Firmware Version	FIR-v1426
Change History	Firmware Version FIR-v1614: Entry "Saveable" modified from "No" to "yes, category: application".

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" and the "Cyclic Synchronous Torque" mode.

Object description

Index	6071$_h$
Object Name	Target Torque
Object Code	VARIABLE
Data type	INTEGER16
Saveable	yes, category: application
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	0000$_h$
Firmware Version	FIR-v1426

Change History	Firmware Version FIR-v1626: Entry "Saveable" modified from "No" to "yes, category: application".

Description

The nominal current in object **203B**$_h$:01$_h$ is equal to a nominal torque. This object is expressed as per thousand of this nominal torque, e.g. the value "500" means "50%" of the nominal torque, "1100" is equal to 110%. This value cannot exceed the peak torque (generated by peak current in **2031**$_h$).

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
Saveable	yes, category: application
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	0000$_h$
Firmware Version	FIR-v1426
Change History	

Description

The nominal current in object **203B**$_h$:01$_h$ is equal to a nominal torque. This object is expressed as per thousand of this nominal torque, e.g. the value "500" means "50%" of the nominal torque, "1100" is equal to 110%. This value cannot exceed the peak torque (generated by peak current in **2031**$_h$).

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
Saveable	No
Access	Read only
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	0000$_h$

Firmware Version	FIR-v1426
Change History	

Description

The nominal current in object **203B**$_h$:01$_h$ is equal to a nominal torque. This object is expressed as per thousand of this nominal torque, e.g. the value "500" means "50%" of the nominal torque, "1100" is equal to 110%. This value cannot exceed the peak torque (generated by peak current in **2031**$_h$).

6077h Torque Actual Value

Function

This object provides the actual value of the torque.

Object description

Index	6077$_h$
Object Name	Torque Actual Value
Object Code	VARIABLE
Data type	INTEGER16
Saveable	No
Savable	No
Access	Read only
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	0000$_h$
Firmware Version	FIR-v1540
Change History	

Description

The nominal current in object **203B**$_h$:01$_h$ is equal to a nominal torque. This object is expressed as per thousand of this nominal torque, e.g. the value "500" means "50%" of the nominal torque, "1100" is equal to 110%. This value cannot exceed the peak torque (generated by peak current in **2031**$_h$).

607Ah Target Position

Function

This object specifies the target position in the "Profile Position" the "Cyclic Synchronous Position" mode.

Object description

Index	607A$_h$
Object Name	Target Position
Object Code	VARIABLE
Data type	INTEGER32
Saveable	yes, category: application
Access	Read/write
PDO Mapping	RX - PDO

Admissible Values	
Specified Value	$00000FA0_h$
Firmware Version	FIR-v1426
Change History	Firmware Version FIR-v1626: Entry "Saveable" modified from "No" to "yes, category: application".

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
Saveable	yes, category: application
Firmware Version	FIR-v1426
Change History	

Value description

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

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

Sub-index	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
Saveable	yes, category: application
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
Saveable	yes, category: application
Firmware Version	FIR-v1426
Change History	

Value description

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

Sub-index	01_h
Name	Min Position Limit
Data type	INTEGER32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	80000000_h

Sub-index	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
Saveable	yes, category: application
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

7	6	5	4	3	2	1	0
POS	VEL						

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
Saveable	yes, category: application
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
Saveable	yes, category: application

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 within a second.

Object description

Index	6083_h
Object Name	Profile Acceleration
Object Code	VARIABLE
Data type	UNSIGNED32
Saveable	yes, category: application
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
Saveable	yes, category: application
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
Saveable	yes, category: application
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
Saveable	yes, category: application
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
Saveable	yes, category: application
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h
Firmware Version	FIR-v1426
Change History	

Description

The nominal current in object **203B**$_h$:01_h is equal to a nominal torque. This object is expressed as per thousand of this nominal torque, e.g. the value "500" means "50%" of the nominal torque, "1100" is equal to 110%. This value cannot exceed the peak torque (generated by peak current in **2031**$_h$).

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
Saveable	yes, category: application
Firmware Version	FIR-v1426
Change History	

Value description

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

Sub-index	01_h
Name	Encoder Increments
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No

Admissible Values	
Specified Value	$000007D0_h$

Sub-index	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
Saveable	yes, category: application
Firmware Version	FIR-v1426
Change History	

Value description

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

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

Sub-index	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
Saveable	yes, category: application
Firmware Version	FIR-v1426
Change History	

Value description

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

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

Sub-index	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
Saveable	yes, category: application
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
Saveable	yes, category: application
Firmware Version	FIR-v1426
Change History	

Value description

Sub-index	00_h
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No

Admissible Values	
Specified Value	02_h

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

Sub-index	02_h
Name	Speed During Search For Zero
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	$0000000A_h$

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 sub-index 1.

The (lower) speed for the search for the reference position is specified in Sub-index 2.

Note

- The speed in Sub-index 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 sub-index 2 should therefore be below 1000 steps per second.
- The speed in sub-index 1 must be greater than the speed in sub-index 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
Saveable	yes, category: application
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
Saveable	yes, category: application
Firmware Version	FIR-v1426
Change History	Firmware Version FIR-v1614: Entry "Name" modified from "End Acceleration Jerk" to "Begin Deceleration Jerk". Firmware Version FIR-v1614: Entry "Name" modified from "Begin Deceleration Jerk" to "End Acceleration Jerk".

Value description

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

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

Sub-index	02_h
Name	Begin Deceleration Jerk
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	

Specified Value	$000003E8_h$

Sub-index	03_h
Name	End Acceleration Jerk
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	$000003E8_h$

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

60C1h Interpolation Data Record

Function

This object contains the target position for the interpolation algorithm of the operation mode "Interpolated Position".

Object description

Index	$60C1_h$
Object Name	Interpolation Data Record
Object Code	ARRAY
Data type	INTEGER32
Saveable	yes, category: application
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	
Firmware Version	FIR-v1512
Change History	Firmware Version FIR-v1626: Entry "Saveable" modified from "No" to "yes, category: application".

Value description

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

Sub-index	01_h
Name	1st Set-point
Data type	INTEGER32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h

Description

The next target position can be written in the sub-index 1 for the "Interpolated Position". At the next synchronisation point in time (depending of the type of field bus) this value gets valid.

60C2h Interpolation Time Period

Function

This object contains the interpolation time in milliseconds.

Object description

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

Value description

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

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

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

60C4h Interpolation Data Configuration

Function

This object provides the maximum buffer size, indicates the configured buffer organisation of interpolation data, and provides objects to define the size of the data record and to clear the buffers. This object is used to enable the drive device to receive the needed data in advance. It also is used to store the positions and further data sent by the control device.

Object description

Index	$60C4_h$
Object Name	Interpolation Data Configuration
Object Code	RECORD
Data type	INTERPOLATION_DATA_CONFIGURATION
Saveable	yes, category: application
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	
Firmware Version	FIR-v1512
Change History	Firmware Version FIR-v1540: Table entry "Access" at sub-index 05 modified from "Read/write" to "Write only". Firmware Version FIR-v1540: Table entry "Access" at sub-index 06 modified from "Read/write" to "Write only". Firmware Version FIR-v1626: Entry "Saveable" modified from "No" to "yes, category: application".

Value description

Sub-index	00_h
Name	Highest Sub-index Supported
Data type	UNSIGNED8
Access	Read only
PDO Mapping	No

Admissible Values	
Specified Value	06_h

Sub-index	01_h
Name	MaximumBufferSize
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000001_h

Sub-index	02_h
Name	ActualBufferSize
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00000001_h

Sub-index	03_h
Name	BufferOrganization
Data type	UNSIGNED8
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	00_h

Sub-index	04_h
Name	BufferPosition
Data type	UNSIGNED16
Access	Read/write
PDO Mapping	No
Admissible Values	
Specified Value	0001_h

Sub-index	05_h
Name	SizeOfDataRecord
Data type	UNSIGNED8
Access	Write only
PDO Mapping	No
Admissible Values	
Specified Value	04_h

Sub-index	06_h
Name	BufferClear
Data type	UNSIGNED8
Access	Write only
PDO Mapping	No
Admissible Values	

Specified Value	00_h

Description

The value of sub-index 01_h contains the maximum possible number of interpolation data records.

The value of sub-index 02_h contains the actual number of interpolation data records.

If sub-index 03_h is "00" indicates a FIFO buffer organisation, if it is "01" it indicates a ring buffer organisation.

The value of sub-index 04_h is dimensionless indicating the next free buffer entry point.

The value of sub-index 05_h is given in byte. If the value "00" is written to sub-index 06_h this clears the buffer inputs, disables the access, and clears all interpolated data records. If "01" is written to sub-index 06_h, this enables access to the input buffers. All other values are reserved.

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
Saveable	yes, category: application
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
Saveable	yes, category: application
Access	Read/write
PDO Mapping	RX - PDO

Admissible Values	
Specified Value	00001388_h
Firmware Version	FIR-v1426
Change History	

60F2h Positioning Option Code

Function

This object defines the positioning behaviour in "Profile Position" mode.

Object description

Index	$60F2_h$
Object Name	Positioning Option Code
Object Code	VARIABLE
Data type	UNSIGNED16
Saveable	yes, category: application
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	0001_h
Firmware Version	FIR-v1446
Change History	Firmware Version FIR-v1614: Entry "Saveable" modified from "No" to "yes, category: application".

Description

At the moment only the following bits are supported:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MS	RESERVED [3]			IP OPTION [4]				RADO [2]		RRO [2]		CIO [2]		REL. OPT. [2]	

REL. OPT. (Relative Option)

These Bits determine relative rotation behaviour in the "Profile Position" mode when bit 6 of the controlword **6040_h** ="1" is set.

Bit 1	Bit 0	Definition
0	0	Positioning moves is performed relative to the preceding (internal absolute) target position (rsp. relative to 0 if there is no preceding target position).
0	1	Positioning moves is performed relative to the actual position demand value (respectively the output of the trajectory generator).
1	0	Positioning moves is performed relative to the position actual value (object **6064_h**).
1	1	Reserved

RRO (Request-Response Option)

⚠ WARNING

These options make the controller modify the object Controlword **6040**$_h$.

These bits determine the behaviour of the controlword **6040**$_h$ Bit 5 ("new setpoint") - the controller releases the bit itself. Therefore there is no need to set the bit to the value "0" externally. After the bit was set to "0" by the controller, the bit 12 ("setpoint acknowledgement") in the statusword **6041**$_h$ is set to the value "0".

Bit 4	Bit 5	Definition
0	0	The functionality is as described in **Setting move commands**.
0	1	The controller releases autonomously the new setpoint bit as soon as target is reached.
1	0	The controller releases autonomously the new setpoint bit as soon as able to accept new set- point data.
1	1	Reserved

RADO (Rotary Axis Direction Option)

These Bits determine the rotation in the "Profile Position" mode.

Bit 7	Bit 6	Definition
0	0	Normal positioning similar to linear axis; If reaching or exceeding the position range limits (**607B**$_h$) the input value shall wrap automatically to the other end of the range. Positioning can be relative or absolute. Only with this bit combination, the movement greater than a modulo value is possible.
0	1	Positioning only in negative direction; if target position is higher than actual position, axis moves over the min position limit **607D**$_h$:01$_h$ to the target position.
1	0	Positioning only in positive direction; if target position is lower than actual position, axis moves over the max position limit **607D**$_h$:02$_h$ to the target position.
1	1	Positioning with the shortest way to the target position. **Note** If the difference between actual value and target position in a 360° system is 180°, the axis moves in positive direction.

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

Description

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
								IN 8	IN 7	IN 6	IN 5	IN 4	IN 3	IN 2	IN 1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
													HS	PLS	NLS

NLS (Negative Limit Switch)

Negative limit switch

PLS (Positive Limit Switch)

Positive limit switch

HS (Home Switch)

Reference switch

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
Saveable	yes, category: application
Firmware Version	FIR-v1426
Change History	Firmware Version FIR-v1626: Entry "Saveable" modified from "No" to "yes, category: application".

Value description

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

Sub-index	01_h
Name	Digital Outputs #1
Data type	UNSIGNED32
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000001_h

Description

The entries in object **3250**$_h$, sub-index 02_h to 05_h still have to be taken into account for writing the outputs.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
												OUT4	OUT3	OUT2	OUT1

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

BRK (Brake)

Bit brake output if ihe controller supports this functionality.

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" the "Cyclic Synchronous Torque" 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
Saveable	yes, category: application
Access	Read/write
PDO Mapping	RX - PDO
Admissible Values	
Specified Value	00000000_h
Firmware Version	FIR-v1426
Change History	Firmware Version FIR-v1626: Entry "Saveable" modified from "No" to "yes, category: application".

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
Saveable	No
Access	Read only
PDO Mapping	TX - PDO
Admissible Values	
Specified Value	$000003EF_h$
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".

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						CST	CSV	CSP	IP	HM		TQ	PV	VL	PP

PP
Profile Position mode

VL
Velocity mode

PV
Profile Velocity mode

TQ
Torque mode

HM
Homing (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
Saveable	No
Access	Read only
PDO Mapping	No
Admissible Values	
Specified Value	http://www.nanotec.de
Firmware Version	FIR-v1426
Change History	

13 Copyright notice

13.1 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.

13.2 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

13.3 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.

13.4 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.

13.5 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.

13.6 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.

13.7 CMSIS DSP Software Library

Copyright (C) 2010 ARM Limited. All rights reserved.

13.8 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.

13.9 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.

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.

13.10 lwIP

Copyright (c) 2001-2004 Swedish Institute of Computer Science.

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The name of the author may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This file is part of the lwIP TCP/IP stack.

Author: Adam Dunkels <adam@sics.se>