# CAN-STEPCON-1H

DS-402 Implementation
for Stepper Controller

Software-Manual

| | |
|---|---|
| **Manual File:** | I:\texte\Doku\MANUALS\CAN\STEPCON-1H\Englisch\hcstep_14s.en9 |
| **Manual Order no.:** | C.2090.21 |
| **Date of Print:** | 18.11.2003 |

| | |
|---|---|
| **Described Firmware Version:** | V1.MFH/CanOpen |

**Changes in the Software and/or Documentation**

| Alterations in this manual versus previous version | Alterations in software | Alterations in documentation |
|---|---|---|
| Corrections or changes in object: | | |
| $1008_h$ | - | x |
| $6062_h$ | - | x |
| $6069_h$ | - | x |
| $606A_h$ | - | x |
| $606B_h$ | - | x |
| $606C_h$ | - | x |
| $607F_h$ | - | x |
| $6099_h$ | - | x |
| $60FF_h$ | - | x |
| $2880_h$ | - | x |
| $2881_h$ | - | x |
| New objects: | | |
| $60FE_h$ | x | x |
| ($2890_h$ ) | x | x |
| ($2891_h$ ) | x | x |
| ($2892_h$ ) | x | x |

Technical details are subject to change without notice.

N O T E

The information in this document has been carefully checked and is believed to be entirely reliable. **esd** makes no warranty of any kind with regard to the material in this document, and assumes no responsibility for any errors that may appear in this document. **esd** reserves the right to make changes without notice to this, or any of its products, to improve reliability, performance or design.

**esd** assumes no responsibility for the use of any circuitry other than circuitry which is part of a product of **esd** gmbh.

**esd** does not convey to the purchaser of the product described herein any license under the patent rights of **esd** gmbh nor the rights of others.

**esd electronic system design gmbh**
Vahrenwalder Str. 207
30165 Hannover
Germany

Phone: +49-511-372 98-0
Fax: +49-511-372 98-68
E-mail: info@esd-electronics.com
Internet: www.esd-electronics.com

**USA / Canada**
esd
PMB 292
20423 State Road 7 #F6
Boca Raton, Florida 33498-6797
USA

Phone: +1-800-732-8006
Fax: +1-800-732-8093
E-mail: sales@esd-electronics.com

## Content

Page

# 1. Introduction

## 1.1 About this Document

This document describes the implementation of the DS-402, the device profile for drives and motion control, especially for micro stepper control motors.
This document does not describe the basic organisation and function of the DS-402. This information can be taken from the first chapters of the referred document [1].

You find information about the implemented functions and the predefined DS-402 objects that are supported in this document. Additionally the manufacturer specific objects that are implemented by esd are described.

## 1.2 Reference

| | | |
|---|---|---|
| [1] CiA DSP-402 | CANopen Device Profile for Drives and Motion Control, | V1.2 (20.06.2000) |
| [2] CiA DS-301 | CANopen Application Layer and Communication Profile, | V4.01 (01.06.2000) |
| [3] CiA DS-401 | CANopen Device Profile for I/O Modules, | V1.4 (12.1996) |

## 1.3 Supported Operation Modes

The implementation covers three of the six operation modes, that are described in the DS-402:
- Homing Mode
- Profile Position Mode
- Profile Velocity Mode

**Note:** The change of the operation modes is only allowed if the motor is stopped!

## 1.4 Supported Axles

The implementation supports only a one axles device.

## 1.5 Error Messages

The error messages defined in the DS-402 are not supported.

## 1.6 Abbreviations

| | |
|---|---|
| n.a. | not applicable |
| rw | read and write access supported |
| ro | read only access supported |
| const. | constant value |
| hex, ... $_h$ | hexadecimal value |
| dec | decimal value |

# 2. DS-301 CANopen Objects

## 2.1 Communication Profile

### 2.1.1 Overview (Communication Profile Area)

The format of the communication parameters can be taken from the CiA DS-301 (Table 9.1). Special functions are described in this manual.

| Index [hex] | Name | Subindex | Type | Access | Default |
|---|---|---|---|---|---|
| 1000 | Device Type | - | Unsigned32 | ro | 00 04 01 $92_h$ |
| 1001 | Error Register | - | Unsigned8 | rw | 0 |
| 1002 | Manufacturer Status | - | Unsigned32 | ro | 00 00 00 $D1_h$ |
| 1003 | Predef'd Error... (Error History) | not supported | | | |
| 1004 | Number of total PDOs | 0 | Unsigned32 | ro | 00 04 00 $04_h$ |
| | Number of sync PDOs | 1 | Unsigned32 | ro | 00 04 00 $04_h$ |
| | Number of async PDOs | 2 | Unsigned32 | ro | 00 04 00 $04_h$ |
| 1005 | COB-ID of Sync Message | - | Unsigned32 | rw | 00 00 00 $80_h$ |
| 1006 | Communication Cycle Period | - | Unsigned32 | rw | 0 |
| 1007 | SYNC_Window_Length | not supported | | | |
| 1008 | Manufacturer's Device Name | - | Visible String | ro | esd_step1 |
| 1009 | Manufacturer's Hardware Version | - | Visible String | ro | (*1) |
| 100A | Manufacturer's Software Version | - | Visible String | ro | (*1) |
| 100B | Node-ID | - | Unsigned32 | ro | DIP-switch setting (see Hardware-Manual) |
| 100C | Guard Time | not supported | | | |
| 100D | Life Time Factor | not supported | | | |
| 100E | Node Guarding ID | - | Unsigned32 | ro | $700_h$ +ID |
| 100F | No of SDOs | - | Unsigned32 | ro | 1 |
| 1010 | Store Parameters | 0,1 | Unsigned32 | rw | -- |
| 1011 | Restore Default Parameters | 0,1 | Unsigned32 | rw | -- |
| 1016 | Consumer Heartbeat Time | 0,1 ... 16 | Unsigned32 | rw | -- |
| 1017 | Producer Heartbeat Time | 0 | Unsigned16 | rw | 10.000d |
| 1018 | Identity Object | 0,1 ... 4 | Unsigned32 | ro | see page 20 |
| 1020 | Verify Configuration | 0,1,2 | Unsigned32 | ro | -- |
| 1029 | Error Behaviour Object | 0,1 | Unsigned8 | rw | -- |
| 1400 - 1403 | Receive PDO Communication Parameter | 0 ... 4 | PDOCommPar | ro | ref. chapter 3 |
| 1600 - 1603 | Receive PDO Mapping Parameter | 0 ... 2 | PDO Mapping | ro | ref. chapter 3 |
| 1800 - 1803 | Transmit PDO Communication Parameter | 0 ... 4 | PDOCommPar | ro | ref. chapter 3 |
| 1A00 - 1A03 | Transmit PDO Mapping Parameter | 0 ... 2 | PDO Mapping | ro | ref. chapter 3 |

ro - read only, rw - read/write, n.a. - not applicable
(*1) depends on rev. level of hard- and software

### 2.1.2 Device Type $1000_h$

The device type is defined by the DS-402.

| **Index** | **$1000_h$** |
|---|---|
| Name | device type |
| Data Type | Unsigned8 |
| Default Value | 00 04 01 $92_h$ |

MSB LSB

| *additional information* | | *device profile no* |
|---|---|---|
| *mode bits* | *type* | |

31 24 23 16 15 0

The following bits of the error register are supported at the moment:

| ***device*** | ***additional information*** | | | | | | | | | | | | | | | | ***device profile number*** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ***mode bits*** | | | | | | | | ***type*** | | | | | | | | |
| | **31** | **30** | **29** | **28** | **27** | **26** | **25** | **24** | **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** | **15 - 0** |
| servo drive | * | * | * | * | * | * | * | * | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | $0192_h$ |
| **stepper motor** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **1** | **0** | **0** | **$0192_h$** |
| multiple device module | * | * | * | * | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | $0192_h$ |

### 2.1.3 Error Register $1001_h$

| **Index** | **$1001_h$** |
|---|---|
| Name | error register |
| Data Type | Unsigned8 |
| Default Value | No |

At the moment the following bits are supported:

| Bit | Meaning |
|---|---|
| 0 | generic |
| 1 | - |
| 2 | - |
| 3 | - |
| 4 | - |
| 5 | - |
| 6 | - |
| 7 | manufacturer-specific error |

Unsupported bits return the value ‘0’.

The following error messages are implemented:

$00_h$ - no error
$81_h$ - any manufacturer-specific error has appeared

### 2.1.4 Manufacturer Status Register $1002_h$

| **Index** | **$1002_h$** |
|---|---|
| Name | manufacturer status register |
| Data Type | Unsigned32 |
| Default Value | $D1_h$ |

Assignment of the status register's bits:

| **Register-Bit (Assembler)** | **Description** | **Level Assignment** | |
|---|---|---|---|
| D31 ... D8 | This bits are reserved for future use. | 0 | reserved, may be returned as '0' or '1' |
| | | 1 | |
| D7 | *New on Bus* | 0 | reserved, may be returned as '0' or '1' |
| | | 1 | |
| D6 | *Default wake up* | 0 | normal start |
| | | 1 | module is started with default parameters |
| D5 | *I²C busy* | 0 | no access to EEPROM |
| | | 1 | local SW accesses I²C-EEPROM |
| D4 | *manufacturer-specific error (corresponds bit 7 of object $1001_h$ )* | 0 | no error |
| | | 1 | manufacturer-specific error is detected |
| D3 | *I²C-Error* | 0 | no I²C-error |
| | | 1 | I²C-error |
| D2 | *Error on CAN* | 0 | no CAN-bus error detected |
| | | 1 | CAN-bus error |
| D1 | *Suspend-Bit* | 0 | module is in state 'operational' |
| | | 1 | module is in state 'preoperational' or 'stopped' |
| D0 | *Power-Up-Reset* | 0 | last reset is not generated by power-up |
| | | 1 | last reset is generated by power-up |

### 2.1.5 COB-ID of SYNC-Message $1005_h$

| **Index** | **$1005_h$** |
|---|---|
| Name | COB-ID SYNC message |
| Data Type | Unsigned32 |
| Default Value | $80_h$ |

Parameter Structure:

| Bit no. | Value | Save to EEPROM | Description |
|---|---|---|---|
| 31 (MSB) | 0/1 | no | 0: object enabled<br>1: object disabled |
| 30 | 0/1 | yes | 0: module is SYNC consumer<br>1: module is SYNC generator |
| 29 | 0 | yes | always 0, because only 11-bit IDs are supported by the module |
| 28...11 | 0 | yes | always 0, because no 29-bit IDs are supported by the module |
| 10...0 (LSB) | x | no | bit 0...10 of SYNC-COB-ID |

### 2.1.6 Communication Cycle Period $1006_h$

| **Index** | **$1006_h$** |
|---|---|
| Name | communication cycle period |
| Data Type | Unsigned32 |
| Default Value | 0 µs |

This object defines the cycle time of the SYNC frame:

| Index | Sub-index | PDO-mappable | Save to EEPROM | Access Mode | Value Range | Default Value | Name/Description |
|---|---|---|---|---|---|---|---|
| $1006_h$ | $0_h$ | no | yes | rw | unsigned32 | 0 µs | *communication cycle period* |

*communication cycle period* SYNC frame's cycle time in [µs].
If set to '0', no SYNC messages are send, if bit 30 (SYNC_COB_Id) = '1' (SYNC generator).

### 2.1.7 Manufacturer's Device Name $1008_h$

| **Index** | **$1008_h$** |
|---|---|
| Name | manufacturer's device name |
| Data Type | visible string |
| Default Value | string: 'esd_step1' |

The data transfer for uploading the string has to be executed as shown in the following table (all values given hexadecimal): The commands always has to be 8 bytes long.

| | Byte 0 | Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | Byte 6 | Byte 7 | Note |
|---|---|---|---|---|---|---|---|---|---|
| Client: | **40** | **08** | **10** | 00 | 00 | 00 | 00 | 00 | upload request index $1008_h$ , subindex = 0 |
| STEPCON: | **41** | **08** | **10** | **00** | **09** | 00 | 00 | 00 | upload ackn., len = 9 |
| Client: | **60** | 00 | 00 | 00 | 00 | 00 | 00 | 00 | segment upload request |
| STEPCON: | **00** | **65** | **73** | **64** | **5F** | **73** | **74** | **65** | 'esd_ste', len = 7 |
| Client: | **70** | 00 | 00 | 00 | 00 | 00 | 00 | 00 | - |
| STEPCON: | **1B** | **70** | **31** | 00 | 00 | 00 | 00 | 00 | 'p1', len = 2, ended |

A detailed description of the domain upload can be taken from the CiA DS-202-2 (CMS-Protocol Specification).

### 2.1.8 Manufacturer's Hardware Version $1009_h$

| **Index** | **$1009_h$** |
|---|---|
| Name | manufacturer's hardware version |
| Data Type | visible string |
| Default Value | depends on actual PCB version number |

This parameter contains the actual PCB version number.

Value range: ASCII-string: 0, 1, 2, 3, ...

### 2.1.9 Manufacturer's Software Version $100A_h$

| **Index** | **$100A_h$** |
|---|---|
| Name | manufacturer's software version |
| Data Type | visible string |
| Default Value | string: 'V....' (depends on actual software version) |

The reading of the software version has to be executed similar to the reading of the device name by the domain upload protocol. A detailed description of the domain upload can be taken from the CiA DS-202-2 (CMS-Protocol Specification).

### 2.1.10 Store Parameters $1010_h$

This command stores the parameter in the EPROM. Only the command 'Save *all* Parameters' is supported.

With the write access the below shown byte order has to be send.

A read access returns information about the implemented store function. In this case a read access to the object $1010_h$/subindex 1 always returns the value $00000001_h$, that indicates 'save all parameters' (for further information see CiA DS-301).

| **Index** | **$1010_h$** |
|---|---|
| Name | store parameters |
| Data Type | Unsigned32 |

| Index | Sub-index | PDO-mappable | Save to EEPROM | Access Mode | Value Range | Default Value | Name/Description |
|---|---|---|---|---|---|---|---|
| $1010_h$ | $0_h$ | no | no | const. | unsigned 8 | $1_h$ | number of entries |
| | $1_h$ | no | no | rw | unsigned 32 | no default, write: 65 76 61 $73_h$ (= ASCII: 'e' 'v' 'a' 's') | save all parameters |

### 2.1.11 Restore Default Parameters $1011_h$

This command activates the default parameters of the module. All individual setting, that have been stored in the EEPROM are deleted. Only the command 'Restore *all* Parameters' is supported.

With the write access the below shown byte order has to be send.

A read access returns information about the implemented restore function. In this case a read access to the object $1011_h$/subindex 1 always returns the value $00000001_h$, that indicates 'restore all default parameters' (for further information see CiA DS-301).

| **Index** | **$1011_h$** |
|---|---|
| Name | restore default parameters |
| Data Type | Unsigned32 |

| Index | Sub-index | PDO-mappable | Save to EEPROM | Access Mode | Value Range | Default Value | Name/Description |
|---|---|---|---|---|---|---|---|
| $1011_h$ | 0 | no | no | const. | unsigned 8 | $1_h$ | number of entries |
| | 1 | no | no | rw | unsigned 32 | no default, write: 64 61 65 $6C_h$ (= ASCII: 'd' 'a' 'o' 'l') | load all default parameters |

### 2.1.12 Consumer Heartbeat Time $1016_h$

| **Index** | **$1016_h$** |
|---|---|
| Name | consumer heartbeat time |
| Data Type | Unsigned32 |
| Default Value | No |

**Heartbeat Function**

The heartbeat protocol defines an error control service without need for remote frames.

A heartbeat *producer* transmits a heartbeat message cyclically. One or more heartbeat *consumer* receive the indication. The relationship between *producer* and *consumer* is configurable via the object dictionary.
The heartbeat *consumer* guards the reception of the heartbeat within the heartbeat consumer time. The guarding of the heartbeat *consumer* starts after the reception of the first heartbeat-CAN frame with data(0) ≠ '0'.
If the heartbeat is not received within the heartbeat consumer time a heartbeat event will be generated. At the STEPCON-1H module the heartbeat event generates a heartbeat error.

If the heartbeat *producer* time is set to a value unequal '0' (object $1017_h$ ) on a running device the heartbeat *producer* at that device starts immediately to send heartbeat frames.
If a device starts, that has already set the value of the heartbeat producer time unequal '0' in the EEPROM, the heartbeat protocol starts on the state transition from INITIALISING to PRE-OPERATIONAL.

The STEPCON-$1_h$ module supports the monitoring of up to 16 heartbeat producers.

| Index [hex] | Sub-index [dec] | Description | Value Range [hex] | Default [dec] | Data Type | Save to EEPROM | Access Mode |
|---|---|---|---|---|---|---|---|
| $1016_h$ | 0 | *number_of_entries* | 0...10 | 16 | unsigned 8 | no | ro |
| | 1 | *consumer-heartbeat_time_1* | 0... 00 7F FF $FF_h$ | 0 | unsigned 32 | yes | rw |
| | 2 | *consumer-heartbeat_time_2* | 0... 00 7F FF $FF_h$ | 0 | unsigned 32 | yes | rw |
| | : | *:* | : | : | : | yes | rw |
| | 16 | *consumer-heartbeat_time_16* | 0... 00 7F FF $FF_h$ | 0 | unsigned 32 | yes | rw |

Description of the parameter *consumer-heartbeat_time_x*:

| | *consumer-heartbeat_time_x* | | |
|---|---|---|---|
| Bit | 31 ... ...24 | 23 ... ...16 | 15 ... ...0 |
| Description | reserved (always '0') | *Node-ID* (unsigned 8) | *heartbeat_time* (unsigned 16) |

*Node-ID* Node-Id of the heartbeat producer module that has to be monitored.

*heartbeat_time* If the heartbeat producer does not send a message at the Node Guarding-ID within the time *heartbeat_time*, a heartbeat event will be generated

The consumer *heartbeat_time* has to be higher than the corresponding producer *heartbeat_time* configured on the device producing this heartbeat.

### 2.1.13 Producer Heartbeat Time $1017_h$

| **Index** | **$1017_h$** |
|---|---|
| Name | producer heartbeat time |
| Data Type | Unsigned16 |
| Default Value | 10.000 ms |

The *producer* heartbeat time defines the cycle time of the heartbeat. The time has to be a multiple of 1 ms. The *producer* heartbeat time has to be set to '0' if it is not used.

If the heartbeat *producer* time is set to a value unequal '0' (object $1017_h$ ) on a running device the heartbeat *producer* at that device starts immediately to send heartbeat frames.
If a device starts, that has already set the value of the heartbeat producer time unequal '0' in the EEPROM, the heartbeat protocol starts on the state transition from INITIALISING to PRE-OPERATIONAL.

The heartbeat function is described at page 17.

| Index | Sub-index | PDO-mappable | Save to EEPROM | Access Mode | Value Range | Default Value | Name/Description |
|---|---|---|---|---|---|---|---|
| $1017_h$ | $0_h$ | no | yes | rw | unsigned 16 ($0...FFFF_h$ ) | 10.000 ms | *producer-heartbeat_time* |

*producer-heartbeat_time* Cycle time of the heartbeat producer to send the heartbeat at the node guarding-ID (see object $100E_h$ ).

The consumer *heartbeat_time* has to be higher than the corresponding producer *heartbeat_time* configured on the device producing this heartbeat.

### 2.1.14 Identity Object $1018_h$

| **Index** | **$1018_h$** |
|---|---|
| Name | identity object |
| Data Type | Unsigned32 |
| Default Value | no |

The Identity Object returns general information about the CAN module.

| Index [hex] | Subindex [dec] | Description | Value Range [hex] | Default Value [hex] | Data Type | Access Mode |
|---|---|---|---|---|---|---|
| $1018_h$ | 0 | *no_of_entries* | 4 | 4 | unsigned 8 | ro |
| | 1 | *vendor_id* | 0 ... FF FF FF $FF_h$ | 00 00 00 $17_h$ | unsigned 32 | ro |
| | 2 | *product_code* | 0 ... FF FF FF $FF_h$ | 22 09 00 $10_h$ | : | ro |
| | 3 | *revision_no* | 0 ... FF FF FF $FF_h$ | software-rev. | : | ro |
| | 4 | *serial_no* | 0 ... FF FF FF $FF_h$ | hardware serial number | unsigned 32 | rw |

Parameter Description:

*vendor_id* This parameter returns the esd-vendor ID. The value of the vendor ID is constant 00 00 00 $17_h$ .

*product_code* This parameter returns the esd-order number of the module.
Example:
In the value '2**2 09 00 10**$_h$' the order number 'C.2090.10' is coded.

*revision_no* This parameter returns the software version. The upper two bytes return the revision numbers of the major changes according to DS-301 and the lower two bytes return the revision number of minor changes. The software version contains of the following sections:

- Kernel Software (described by the parameters '*lev*' (level) and '*rev*' (revision))
- Application Software (described by the parameter '*ext*' (extension))
- Protocol Software (described by the parameter '*plev*' (protocol level))

The returned software version is coded for 'major' and for 'minor'-revision number as follows

*revision_no* =xxyy
and is determined by the number s of the ASCII-code of the characters
with xx = (*lev* - '0') · 26 + (*rev* - 'A')
yy = (*plev* - 'H') · 16 + (*ext* - 'A')

Example:
If the actual software has the revision:
Kernel: *lev* = '1' *rev* = 'M'
Application: *ext* = 'A'
Protocol: *plev* = 'H'

the returned value for *revision_no* will be 26 00 00 $00_h$.

*serial_no*

This parameter returns the serial number code of the PCB-hardware.
The higher two bytes of *serial_no* contain the letters that code the production lot. They return the ASCII-code of the letters with the most significant bit set to '1', to distinguish letters from numbers:
(ASCII-code) + $80_h$ = returned bytes

The following numbers code the number of each module as BCD-value.

Example:
If the value 'C1 C1 12 $34_h$ ' is returned, this will mean the hardware serial number code 'AA 1234'. This value has to match with the value labelled at the module.

### 2.1.15 Verify Configuration $1020_h$

| **Index** | **$1020_h$** |
|---|---|
| Name | verify configuration |
| Data Type | Unsigned32 |
| Default Value | no |

In this parameter the date and the time of the last configuration can be stored. This can be used to check if this is the correct configuration version.

After Default-Reset the value is ‘0’.

| Index [hex] | Subindex [dec] | Description | Value Range [hex] | Default Value [hex] | Data Type | Sve to EEPROM | Access Mode |
|---|---|---|---|---|---|---|---|
| $1020_h$ | 0 | *no_of_entries* | 2 | 2 | unsigned 8 | no | ro |
| | 1 | *configuration_date* | 0... FF FF FF $FF_h$ | 0 | unsigned 32 | yes | rw |
| | 2 | *configuration_time* | 0... FF FF FF $FF_h$ | 0 | unsigned 32 | yes | rw |

Parameter Description:

*configuration_date* Date of the last configuration.
The value returns the number of days since the 01.01.1984.

*configuration_time* Time in ms since midnight at the day of the last configuration

### 2.1.16 Error Behaviour Object $1029_h$

| **Index** | **$1029_h$** |
|---|---|
| Name | error behaviour object |
| Data Type | Unsigned8 |
| Default Value | no |

If an error event appears (e.g. heartbeat error), the module will switch to the state that is defined in the parameter *communication_error.*

| Index [hex] | Subindex [dec] | Description | Value Range [hex] | Default Value [dec] | Data Type | Save to EEPROM | Access Mode |
|---|---|---|---|---|---|---|---|
| $1029_h$ | 0 | *no_of_error_classes* | 1 | 1 | unsigned 8 | no | ro |
| | 1 | *communication_error* | 0, 1, 2 | 0 | unsigned 8 | yes | rw |

Parameter Description:

*no_of_error_classes* Number of error classes (here always ‘1’)

*communication_error* 0 - pre-operational (only, if actual state is operational)
1 - no state change
2 - stopped

### 2.1.17 Supported Transmission Types acc. to DS-301, Table 55

(Valid from software version ‘1mfh’.)

| Transmission Type | PDO transmission | | | | | Supported by esd-module |
|---|---|---|---|---|---|---|
| | cyclic | acyclic | synchronous | asynchronous | RTR only | |
| 0 | | X | X | | | YES |
| 1...240 | X | | X | | | YES |
| 241... 251 | reserved | | | | | n.a. |
| 252 | | | X | | X | YES |
| 253 | | | | X | X | NO |
| 254 | | | | X | | YES |
| 255 | | | | X | | YES |

## 2.2 Table of Important Identifier and Messages of CANopen

| CAN Identifier [hex] | Name of Object | Length | Data [hex] | Note |
|---|---|---|---|---|
| 0 | NMT | 2 | 02 xx | module changes to state ‘Stopped’ |
| 0 | NMT | 2 | 01 xx | Start (module changes to state‘ Operational’) |
| 0 | NMT | 2 | 80 xx | module changes to state ‘Preoperational’ |
| 0 | NMT | 2 | 81 xx | reset module |
| 0 | NMT | 2 | 82 xx | reset communication (here same function as 81xx) |
| $700_h$ + Node-ID | Heartbeat | 1 byte | state | NMT error control identifier (see DS-301) |
| $580_h$ + Node-ID | SDO | 8 bytes | parameter | acknowledge of the modules communication parameters |
| $600_h$ + Node-ID | SDO | 8 bytes | parameter | transfer of the communication parameters to the module (Rx) |

xx = Node-ID
Node-ID. = $1...7F_h$

# 3. Motion Control PDO Mapping

## 3.1 Receive PDOs

With the object '**Receive PDO Communication Parameter $1400_h$ - $1403_h$**' the properties of a receive PDO are defined. The subindex '1' defines the assignment of PDO/CAN-ID and the subindex '2' defines the transmission type of the PDO.
The Receive PDO Communication Parameters of the module are defined by the DS-402.

| PDO No. | Mapping Object Index | Mapping Object Name | M/O | Comment |
|---|---|---|---|---|
| 1 | $6040_h$ | *controlword* | M | controls the state machine |
| 2 | $6040_h$<br>$6060_h$ | *controlword*<br>*modes_of_operation* | O | controls the state machine and mode of operation |
| 3 | $6040_h$<br>$607A_h$ | *controlword*<br>*target_position* | O | controls the state machine and the target position (pp) |
| 4 | $6040_h$<br>$60FF_h$ | *controlword*<br>*target_velocity* | O | controls the state machine and the target velocity (pv) |

### 3.1.1 1st Receive PDO

| Index | Subindex | Comment | Save to EEPROM | Default Value |
|---|---|---|---|---|
| $1400_h$ | 0 | number of entries | no | 3 |
| | 1 | COB-ID used by PDO | yes | $200_h$ + Node-ID |
| | 2 | transmission type | no | 255 |
| | 3 | inhibit time | no | 0 |

| Index | Subindex | Comment | Save to EEPROM | Default Value |
|---|---|---|---|---|
| $1600_h$ | 0 | number of mapped objects | no | 1 |
| | 1 | *controlword* | no | 60 40 00 $10_h$ |

### 3.1.2 2nd Receive PDO

| Index | Subindex | Comment | Save to EEPROM | Default Value |
|---|---|---|---|---|
| $1401_h$ | 0 | number of entries | no | 3 |
| | 1 | COB-ID used by PDO | yes | $300_h$ + Node-ID |
| | 2 | transmission type | no | 255 |
| | 3 | inhibit time | no | 0 |

| Index | Subindex | Comment | Save to EEPROM | Default Value |
|---|---|---|---|---|
| $1601_h$ | 0 | number of mapped objects | no | 2 |
| | 1 | *controlword* | no | 60 40 00 $10_h$ |
| | 2 | *modes_of_operation* | no | 60 60 00 $08_h$ |

### 3.1.3 3rd Receive PDO

| Index | Subindex | Comment | Save to EEPROM | Default Value |
|---|---|---|---|---|
| $1402_h$ | 0 | number of entries | no | 3 |
| | 1 | COB-ID used by PDO | yes | $400_h$ + Node-ID |
| | 2 | transmission type | no | 254 |
| | 3 | inhibit time | no | 0 |

| Index | Subindex | Comment | Save to EEPROM | Default Value |
|---|---|---|---|---|
| $1602_h$ | 0 | number of mapped objects | no | 2 |
| | 1 | *controlword* | no | 60 40 00 $10_h$ |
| | 2 | *target_position* | no | 60 7A 00 $20_h$ |

### 3.1.4 4th Receive PDO

| Index | Subindex | Comment | Save to EEPROM | Default Value |
|---|---|---|---|---|
| $1403_h$ | 0 | number of entries | no | 3 |
| | 1 | COB-ID used by PDO | yes | $500_h$ + Node-ID |
| | 2 | transmission type | no | 254 |
| | 3 | inhibit time | no | 0 |

| Index | Subindex | Comment | Save to EEPROM | Default Value |
|---|---|---|---|---|
| $1603_h$ | 0 | number of mapped objects | no | 2 |
| | 1 | *controlword* | no | 60 40 00 $10_h$ |
| | 2 | *profile_velocity* | no | 60 FF 00 $20_h$ |

## 3.2 Transmit PDOs

With the object '**Transmit PDO Communication Parameter $1800_h$ - $1803_h$**' the properties of a receive PDO are defined. The subindex '1' defines the assignment of PDO/CAN-ID and the subindex '2' defines the transmission type of the PDO.
The Transmit PDO Communication Parameters of the module are defined by the DS-402.

| PDO No. | Mapping Object Index | Mapping Object Name | M/O | Comment |
|---|---|---|---|---|
| 1 | $6041_h$ | *statusword* | M | shows status |
| 2 | $6041_h$<br>$6061_h$ | *statusword*<br>*modes_of_operation_display* | O | shows status and the actual mode of operation |
| 3 | $6041_h$<br>$6064_h$ | *statusword*<br>*position_actual_value* | O | shows the status and the actual position (pp) |
| 4 | $6041_h$<br>$606C_h$ | *statusword*<br>*velocity_actual_value* | O | shows the status and the actual velocity (pv) |

### 3.2.1 1st Transmit PDO

| Index | Subindex | Comment | Save to EEPROM | Default Value |
|---|---|---|---|---|
| $1800_h$ | 0 | number of entries | no | 5 |
| | 1 | COB-ID used by PDO | yes | $180_h$ + Node-ID |
| | 2 | transmission type | yes | 255 |
| | 3 | inhibit time | no | 0 |
| | 4 | CMS priority group | no | 6 |
| | 5 | T_Cycle | yes | 0 |

| Index | Subindex | Comment | Save to EEPROM | Default Value |
|---|---|---|---|---|
| $1A00_h$ | 0 | number of mapped objects | no | 1 |
| | 1 | *statusword* | no | 60 41 00 $10_h$ |

### 3.2.2 2nd Transmit PDO

| Index | Subindex | Comment | Save to EEPROM | Default Value |
|---|---|---|---|---|
| $1801_h$ | 0 | number of entries | no | 5 |
| | 1 | COB-ID used by PDO | yes | $280_h$ + Node-ID |
| | 2 | transmission type | yes | 255 |
| | 3 | inhibit time | no | 0 |
| | 4 | CMS priority group | no | 6 |
| | 5 | T_Cycle | yes | 0 |

| Index | Subindex | Comment | Save to EEPROM | Default Value |
|---|---|---|---|---|
| $1A01_h$ | 0 | number of mapped objects | no | 2 |
| | 1 | *statusword* | no | 60 41 00 $10_h$ |
| | 2 | *modes_of_operation_display* | no | 60 61 00 $08_h$ |

### 3.2.3 3rd Transmit PDO

| Index | Subindex | Comment | Save to EEPROM | Default Value |
|---|---|---|---|---|
| $1802_h$ | 0 | number of entries | no | 5 |
| | 1 | COB-ID used by PDO | yes | $380_h$ + Node-ID |
| | 2 | transmission type | yes | 254 |
| | 3 | inhibit time | no | 0 |
| | 4 | CMS priority group | no | 6 |
| | 5 | T_Cycle | yes | 100 |

| Index | Subindex | Comment | Save to EEPROM | Default Value |
|---|---|---|---|---|
| $1A02_h$ | 0 | number of mapped objects | no | 2 |
| | 1 | *statusword* | no | 60 41 00 $10_h$ |
| | 2 | *position_actual_value* | no | 60 64 00 $20_h$ |

### 3.2.4 4th Transmit PDO

| Index | Subindex | Comment | Save to EEPROM | Default Value |
|---|---|---|---|---|
| $1803_h$ | 0 | number of entries | no | 5 |
| | 1 | COB-ID used by PDO | yes | $480_h$ + Node-ID |
| | 2 | transmission type | yes | 254 |
| | 3 | inhibit time | no | 0 |
| | 4 | CMS priority group | no | 6 |
| | 5 | T_Cycle | yes | 100 |

| Index | Subindex | Comment | Save to EEPROM | Default Value |
|---|---|---|---|---|
| $1A03_h$ | 0 | number of mapped objects | no | 2 |
| | 1 | *statusword* | no | 60 41 00 $10_h$ |
| | 2 | *velocity_actual_value* | no | 60 6C 00 $20_h$ |

## 3.3 PDO-Mapping Summary

| **Rx-PDO1** | Data Bytes | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| (default-setting) | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| $201_h$ | *controlword* ($6040_h$) LSB | MSB | - | - | - | - | - | - |

| **Rx-PDO2** | Data Bytes | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| (default-setting) | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| $301_h$ | *controlword* ($6040_h$) LSB | MSB | *modes_of_operation* ($6060_h$) | - | - | - | - | - |

| **Rx-PDO3** | Data Bytes | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| (default-setting) | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| $401_h$ | *controlword* ($6040_h$) | | *target_position* ($607A_h$) | | | | - | - |
| | LSB | MSB | LLB | LUB | ULB | UUB | | |

| **Rx-PDO4** | Data Bytes | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| (default-setting) | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| $501_h$ | *controlword* ($6040_h$) | | *profile_velocity* ($60FF_h$) | | | | - | - |
| | LSB | MSB | LLB | LUB | ULB | UUB | | |

| **Tx-PDO1** | Data Bytes | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| (default-setting) | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| $181_h$ | *statusword* ($6041_h$) LSB | MSB | - | - | - | - | - | - |

| **Tx-PDO2** | Data Bytes | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| (default-setting) | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| $281_h$ | *statusword* ($6041_h$) LSB | MSB | *modes_of_operation_display* ($6061_h$) | - | - | - | - | - |

| **Tx-PDO3** | Data Bytes | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| (default-setting) | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| $381_h$ | *statusword* ($6041_h$) | | *position_actual_value* ($6064_h$) | | | | - | - |
| | LSB | MSB | LLB | LUB | ULB | UUB | | |

| **Tx-PDO4** | Data Bytes | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| (default-setting) | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| $481_h$ | *statusword* ($6041_h$) | | *velocity_actual_value* ($606C_h$) | | | | - | - |
| | LSB | MSB | LLB | LUB | ULB | UUB | | |

-...not determined

# 4. Motion Control Objects

## 4.1 Implemented DS-402 Objects

| Operation Mode | Index | Object | Name | Type | Attr. | Save to EEPROM |
|---|---|---|---|---|---|---|
| Common Entries | $6007_h$ | VAR | *abort_connection_option_code* | Integer16 | rw | yes |
| | $6402_h$ | VAR | *motor_type* | Unsigned16 | ro | no |
| | $6502_h$ | VAR | *supported_drive_modes* | Unsigned32 | ro | no |
| | $65FE_h$ | ARRAY | *digital_outputs* | Unsigned32 | rw | no |
| Device Control | $6040_h$ | VAR | *controlword* | Unsigned16 | rw | no |
| | $6041_h$ | VAR | *statusword* | Unsigned16 | ro | no |
| | $605A_h$ | VAR | *quick_stop_option_code* | Integer16 | rw | yes |
| | $605B_h$ | VAR | *shutdown_option_code* | Integer16 | rw | yes |
| | $605C_h$ | VAR | *disable_operation_option_code* | Integer16 | rw | yes |
| | $605D_h$ | VAR | *halt_option_code* | Integer16 | rw | yes |
| | $605E_h$ | VAR | *fault_reaction_option_code* | Integer16 | rw | yes |
| | $6060_h$ | VAR | *modes_of_operation* | Integer8 | rw | no |
| | $6061_h$ | VAR | *modes_of_operation_display* | Integer8 | ro | no |
| Profile Position Mode | ($6062_h$) | VAR | *position_demand_value* | Integer32 | ro | yes |
| | $6063_h$ | VAR | *position_actual_value** | Integer32 | ro | yes |
| | $6064_h$ | VAR | *position_actual_value* | Integer32 | ro | yes |
| | $607A_h$ | VAR | *target_position* | Integer32 | rw | no |
| | $607F_h$ | VAR | *max_profile_velocity* | Unsigned32 | rw | no |
| | $6081_h$ | VAR | *profile_velocity* | Unsigned32 | rw | no |
| | $6083_h$ | VAR | *profile_acceleration* | Unsigned32 | rw | yes |
| | $6084_h$ | VAR | *profile_deceleration* | Unsigned32 | rw | yes |
| | $6085_h$ | VAR | *quick_stop_deceleration* | Unsigned32 | rw | yes |
| | $6086_h$ | VAR | *motion_profile_type* | Integer16 | ro | no |
| Homing Mode | $607C_h$ | VAR | *home_offset* | Integer32 | rw | yes |
| | $6098_h$ | VAR | *homing_method* | Integer8 | rw | yes |
| | $6099_h$ | ARRAY | *homing_speeds* | Unsigned32 | rw | yes |
| | $609A_h$ | VAR | *homing_acceleration* | Unsigned32 | rw | yes |
| Profile Velocity Mode | $6069_h$ | VAR | *velocity_sensor_actual_value* | Integer32 | ro | no |
| | $606A_h$ | VAR | *sensor_selection_code* | Integer16 | ro | no |
| | $606B_h$ | VAR | *velocity_demand_value* | Integer32 | ro | no |
| | $606C_h$ | VAR | *velocity_actual_value* | Integer32 | ro | no |
| | $60FF_h$ | VAR | *target_velocity* | Integer32 | rw | no |

(Objects shown in brackets are not yet supported by the described firmware version.)

## 4.2 Manufacturer-Specific Objects

| Index | Object | Name | Type | Attr. | Save to EEPROM | Default |
|---|---|---|---|---|---|---|
| $2800_h$ | VAR | *my_state* | Unsigned8 | ro | no | - |
| $2801_h$ | VAR | *op_mode* | Unsigned8 | ro | no | - |
| $2802_h$ | VAR | *i_ist* | Unsigned8 | ro | no | - |
| $2803_h$ | VAR | *i_nom* | Unsigned8 | rw | yes | $06_h$ |
| $2804_h$ | VAR | *i_low* | Unsigned8 | rw | yes | $86_h$ |
| $2805_h$ | VAR | *brake_on_delay* | Unsigned8 | rw | yes | 0 |
| $2806_h$ | VAR | *brake_off_delay* | Unsigned8 | rw | yes | 0 |
| $2807_h$ | VAR | *t_auto_shutdown* | Unsigned8 | rw | yes | $250_d$ |
| $2808_h$ | VAR | *auto_shut_opt* | Unsigned8 | rw | yes | 3 |
| $2809_h$ | VAR | *min_gear* | Unsigned8 | rw | yes | 1 |
| $280A_h$ | VAR | *max_gear* | Unsigned8 | rw | yes | 2 |
| $280B_h$ | VAR | *es_option* | Unsigned8 | rw | yes | 4 |
| $280C_h$ | VAR | *pdo_pos_mask* | Unsigned8 | rw | yes | 3 |
| $2810_h$ | VAR | *es_lower_opt* | Unsigned16 | rw | yes | 7 |
| $2811_h$ | VAR | *es_upper_opt* | Unsigned16 | rw | yes | 7 |
| $2812_h$ | VAR | *reference_on* | Unsigned16 | rw | yes | 7 |
| $2820_h$ | VAR | *ref_position* | Signed32 | ro | yes | - |
| $2880_h$ | VAR | *temperature* | Signed16 | ro | yes | - |
| $2881_h$ | VAR | *drive_supply_voltage* | Unsigned16 | ro | yes | - |
| ($2890_h$) | VAR | *abs_norm_current* | Unsigned16 | ro | yes | - |
| ($2891_h$) | VAR | *abs_phase_current_a* | Unsigned16 | ro | yes | - |
| ($2892_h$) | VAR | *abs_phase_current_b* | Unsigned16 | ro | yes | - |
| $28FF_h$ | VAR | *esd_mode* | Unsigned8 | rw | yes | - |

(Objects shown in brackets are not yet supported by the described firmware version.)

## 4.3 Object Dictionary

Each drive shares the dictionary entries from $6000_h$ to $63FF_h$. These entries are common to all drive modules and each module implements only the dictionary parts which are relevant for its functions.

| **Meaning of the Table Rows in the Object Description:** | |
|---|---|
| Index | the 16-bit index to the object dictionary used by a module to represent a special function, data or task |
| Name | short description of the usage |
| Object Code | object type which represents the data, e.g. VAR, ARRAY, RECORD, etc. |
| Data Type | data type of the object, e.g. Unsigned32, Unsigned8 etc. |
| Object Class | entries in this row indicates whether an object is mandatory or not:<br>M - this object is mandatory for all drives<br>O - this object is optional dependent on the mode of operation |
| Access | description how the object might be accessed:<br>ro - read only<br>wo - write only<br>rw - read and write |
| PDO Mapping | indicates the manner of PDO mapping for an object.<br>no - mapping is not allowed<br>possible - mapping is allowed for the manufacturer<br>yes - this object is mapped by default |
| Units | physical units of the object value |
| Value Range | the value range allowed and requested for the object |
| Default Value | default value of the object after device initialization |
| Substitute Value | if the object doesn't exist in the object dictionary description, this value will be used for internal calculations |
| **Device Mode Abbreviations:** | |
| pp | mandatory (m), optional (o) or not used (-) for the Profile Position Mode |
| pv | mandatory (m), optional (o) or not used (-) for the Profile Velocity Mode |
| vl | Velocity Mode (not implemented) |
| hm | mandatory (m), optional (o) or not used (-) for the Homing Mode |
| ip | Interpolated Position Mode (not implemented) |
| tq | Profile Torque Mode (not implemented) |
| all | mandatory for all modes |
| **Chapter Titel** | **Abbreviations** |
| ce | Common Entries in the Object Dictionary |
| dc | Device Control |

# 5. Common Entries in the Object Dictionary

## 5.1 General Information

### 5.1.1 Motor Data

The DS-402 defines the objects $6402_h$ to $64FF_h$ to serve as a database for motor parameters.

The objects $6402_h$ to $64\mathbf{0F}_h$ are implemented by esd.

## 5.2 Object Dictionary Entries

### 5.2.1 Objects defined in this Chapter

| Index | Object | Name | Type | Attr. | Save to EEPROM |
|---|---|---|---|---|---|
| $6007_h$ | VAR | *abort_connection_option_code* | Integer16 | rw | yes |
| $6402_h$ | VAR | *motor_type* | Unsigned16 | ro | no |
| $6502_h$ | VAR | *supported_drive_modes* | Unsigned32 | ro | no |
| $60FE_h$ | ARRAY | *digital_outputs* | Unsigned32 | rw | no |

## 5.3 Object Description

### 5.3.1 Object $6007_h$: *abort_connection_option_code*

The content of this object selects the function to be performed when the connection to the network is lost (e.g. heartbeat error event).

| **Index** | **$6007_h$** |
|---|---|
| Name | *abort_connection_option_code* |
| Object Code | VAR |
| Data Type | Integer16 |

**Value Description**

| Object Class | M: - | O: all |
|---|---|---|
| Access | rw | |
| PDO Mapping | no | |
| Units | - | |
| Value Range | -32768...32767 | |
| Mandatory Range | - | |
| Default Value | 0 | |
| Substitute Value | - | |

**Data Description**

| Option Code | Meaning of the Option Code |
|---|---|
| 0 | no action |
| 1 | malfunction (generates RESET) |
| 2 | device control command ‘Disable Voltage’ (Switch_On_Disabled) |
| 3 | device control command ‘Quick Stop’ (Switch_On_Disabled) |
| 4... 32767 | reserved |
| FF $00_h$ | disable drive function |
| FF $01_h$ | slow down on slow down ramp |
| FF $02_h$ | slow down on quick stop ramp |

### 5.3.2 Object $6402_h$: *motor_type*

The type of motor driven by the controller.

| **Index** | **$6402_h$** |
|---|---|
| Name | *motor_type* |
| Object Code | VAR |
| Data Type | Unsigned16 |

**Value Description**

| Object Class | M: - | O: - |
|---|---|---|
| Access | ro | |
| PDO Mapping | no | |
| Units | - | |
| Value Range | 0...65535 | |
| Mandatory Range | - | |
| Default Value | 9 | |
| Substitute Value | - | |

**Data Description**

| **Value** | **Motor Type** |
|---|---|
| : | |
| 9 | Micro-Step Stepper Motor |
| : | |

### 5.3.3 Object $6502_h$: *supported_drive_modes*

A drive can support more then one and several distinct modes of operation. This object gives an overview of the implemented operating modes in the device. This object is read only.

| **Index** | **$6502_h$** |
|---|---|
| Name | *supported_drive_modes* |
| Object Code | VAR |
| Data Type | Unsigned32 |

**Value Description**

| Object Class | M: - | O: - |
|---|---|---|
| Access | ro | |
| PDO Mapping | no | |
| Units | - | |
| Value Range | 0 ... +($2^{32}$-1) | |
| Mandatory Range | - | |
| Default Value | 00 00 00 $25_h$ | |
| Substitute Value | - | |

**Data Description**

| **Bit Number** | **Description** |
|---|---|
| **0** | **Profile Position Mode** |
| 1 | not implemented |
| **2** | **Profile Velocity Mode** |
| 3 | not implemented |
| 4 | reserved |
| **5** | **Homing Mode** |
| 6 | not implemented |
| 7...15 | reserved |
| 16...31 | manufacturer specific (not used) |

### 5.3.4 Object 60FE$_h$: *digital_outputs*

This index defines simple digital outputs for drives.

| **Index** | **60FE$_h$** |
|---|---|
| Name | *digital_outputs* |
| Object Code | ARRAY |
| Number of Elements | 12 |
| Data Type | Unsigned32 |

**Value Description**

| Subindex | 01 $_h$ | |
|---|---|---|
| Description | *physical_outputs* | |
| Object Class | M: all | O:- |
| Access | rw | |
| PDO Mapping | no | |
| Units | - | |
| Value Range | 0000 0000 ... FFFF FFFF | |
| Mandatory Range | - | |
| Default Value | 0 | |
| Substitute Value | 0 | |

| Subindex | 02 $_h$ | |
|---|---|---|
| Description | *bit_mask* | |
| Object Class | M: - | O: all |
| Access | rw | |
| PDO Mapping | no | |
| Units | - | |
| Value Range | 0000 0000 ... FFFF FFFF | |
| Mandatory Range | - | |
| Default Value | 0 | |
| Substitute Value | 0 | |

**Object 60FE$_h$ *digital_outputs* Data Description:**

The first sub-index defines the assigned outputs:

| 31 ... 16 | 15 ... 1 | 0 |
|---|---|---|
| reserved (set always to '0') | reserved (set always to '0') | *set_brake* |
| MSB | | LSB |

*set_brake* = 0000 0000 0000 0000 ... brake inactive
*set_brake* = 0000 0000 0000 0001 ... brake active (stop)

The second sub-index describes a mask to specify which of the outputs shall be used:

| 31 ... 16 | 15 ... 1 | 0 |
|---|---|---|
| reserved (set always to '0') | reserved (set always to '0') | *set_mask* |
| MSB | | LSB |

*set_mask* = 0000 0000 0000 0000 ... brake output setting is ignored
*set_mask* = 0000 0000 0000 0001 ... brake output can be set by *set_brake*

# 6. Device Control

## 6.1 General Information

The device control function block controls all functions of the drive (drive function and power section). It is divided into:

- device control of the state machine
- operation mode function

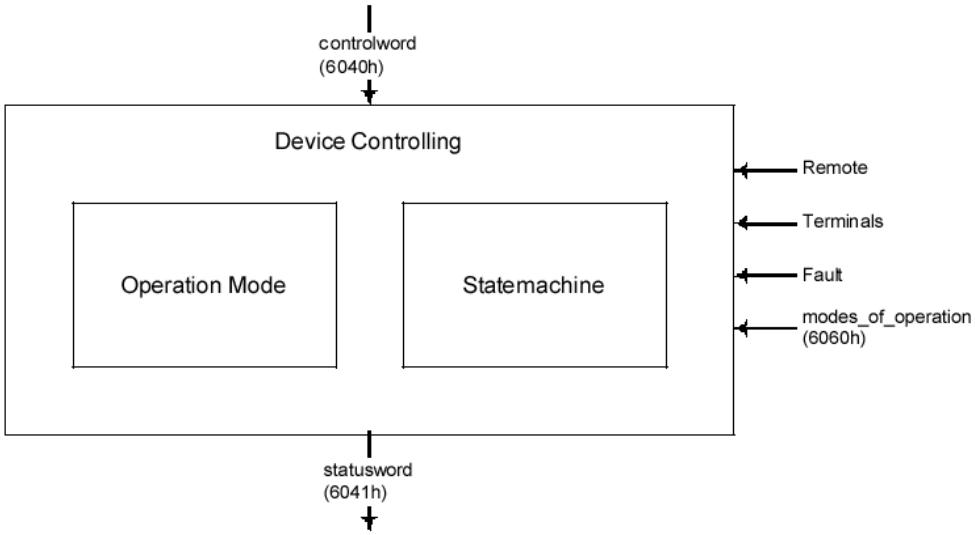


**Figure 6.1.1: Device-Controlling**

The state of the drive can be controlled by the *controlword*.

The state of the drive is shown in the *statusword*.

In remote mode the device is controlled directly from the CAN-network by Process Data Objects (PDOs) and Service Data Objects (SDOs).

The state machine is controlled externally by the *controlword* and external signals. The write access to the *controlword* is controlled by the hardware signal 'Remote' (remote is always active in this implementation). The state machine is also controlled by internal signals like faults and *modes_of_operatio*n.

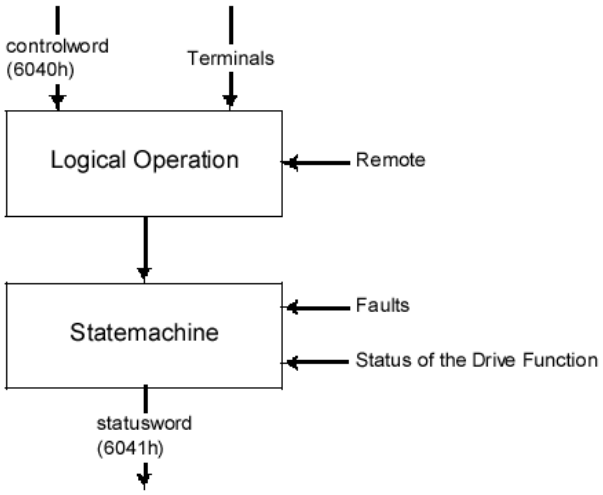


**Figure 6.1.2: Remote Mode**

### 6.1.1 State Machine

The state machine describes the device status and the possible control sequence of the drive. A single state represents a special internal or external behaviour. The state of the drive also determines which commands are accepted. E.g. it is only possible to start a point-to-point move when the drive is in state OPERATION ENABLE.

States may be changed using the *controlword* and/or according to internal events. The current state can be read using the *statusword*.

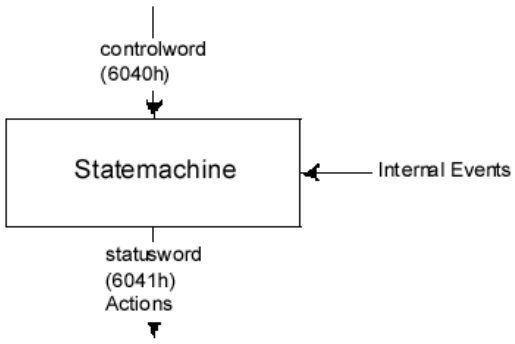


**Figure 6.1.3: State Machine in System Context**

The state machine in Figure 6.1.3 describes the state machine of the device with respect to control of the power electronics as a result of user commands and internal drive faults.

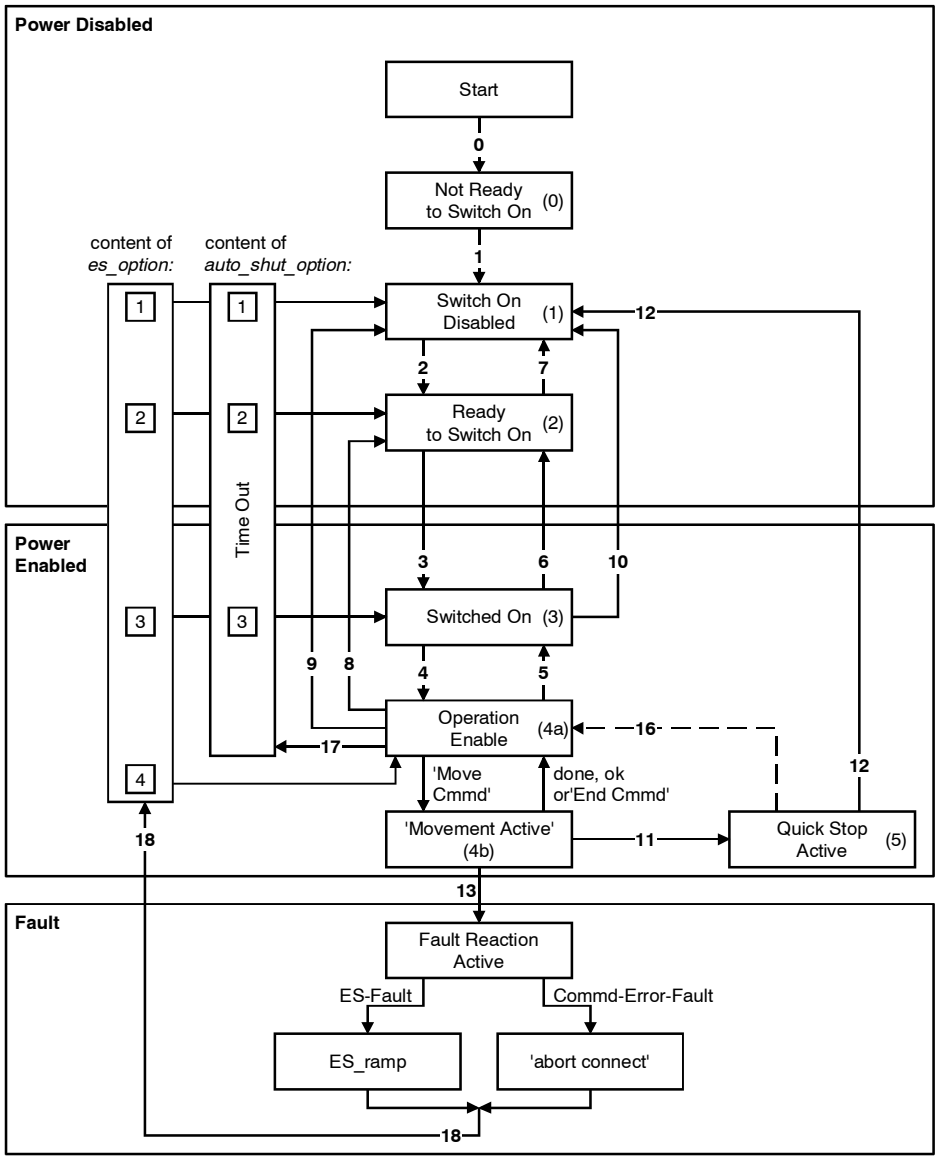


**Figure 6.1.4:** State Machine

The numbers in brackets show the states returned in the manufacturer-specific object *my_state* ($2800_h$).

### 6.1.1.1 Drive States

The drive states may become more evident when considering the following (generic) block diagram of a drive:

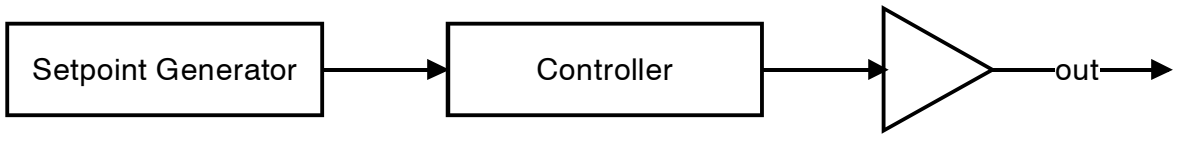


**Figure 6.1.5: Generic Control Block Diagram**

The following states of the device are possible:

| State | Brake | Current |
|---|---|---|
| ○ *NOT READY TO SWITCH ON:*<br>Low level power (e.g. 15V, 5V) has been applied to the drive.<br>The drive is being initialized or is running self test.<br>A brake, if present, has to be applied in this state.<br>The drive function is disabled. | on | 0 |
| ○ *SWITCH ON DISABLED:*<br>Drive initialisation is complete.<br>The drive parameters have been set up.<br>Drive parameters may be changed.<br>High voltage may not be applied to the drive, (e.g. for safety reasons).<br>The drive function is disabled. | on | 0 |
| ○ *READY TO SWITCH ON:*<br>High voltage may be applied to the drive.<br>The drive parameters may be changed.<br>The drive function is disabled. | on | *i_low* |
| ○ *SWITCHED ON:*<br>High voltage has been applied to the drive.<br>The power amplifier is ready.<br>The drive parameters may be changed.<br>The drive function is disabled. | on | *i_nom* |
| ○ *OPERATION ENABLE:*<br>No faults have been detected.<br>The drive function is enabled and power is applied to the motor.<br>The drive parameters may be changed.<br>(This corresponds to normal operation of the drive.) | off | *i_nom* |
| ○ *QUICK STOP ACTIVE:*<br>The drive parameters may be changed.<br>The quick stop function is being executed.<br>The drive function is enabled and power is applied to the motor. | off | *i_nom* |
| ○ *TIME OUT of 'operation enable':*<br>The motor can be switched to one of the another states. The time out can be set by object $2807_h$ (*t_auto_shutdown*) and the state is selected by object $2808_h$ (*auto_shut_opt*). | off | *i_nom* |

#### 6.1.1.2 State Transitions of the Drive Supervisor

State transitions are caused by internal events in the drive or by commands from the host via the controlword.

- *State Transition 0: START _ NOT READY TO SWITCH ON*
  Event: Reset.
  Action: The drive self-tests and/or self-initialises.

- *State Transition 1: NOT READY TO SWITCH ON _ SWITCH ON DISABLED*
  Event: The drive has self-tested and/or initialised successfully.
  Action: Activate communication and process data monitoring.

- *State Transition 2: SWITCH ON DISABLED _ READY TO SWITCH ON*
  Event: 'Shutdown' command received from host.
  Action: None

- *State Transition 3: READY TO SWITCH ON _ SWITCHED ON*
  Event: 'Switch On' command received from host.
  Action: The power section is switched on if it is not already switched on.

- *State Transition 4: SWITCHED ON _ OPERATION ENABLE*
  Event: 'Enable Operation' command received from host.
  Action: The drive function is enabled.

- *State Transition 5: OPERATION ENABLE _ SWITCHED ON*
  Event: 'Disable Operation' command received from host.
  Action: The drive operation will be disabled.

- *State Transition 6: SWITCHED ON _ READY TO SWITCH ON*
  Event: 'Shutdown' command received from host.
  Action: The power section is switched off.

- *State Transition 7: READY TO SWITCH ON _ SWITCH ON DISABLED*
  Event: 'Quick Stop' and ‘Disable Voltage’ command received from host.
  Action: None

- *State Transition 8: OPERATION ENABLE _ READY TO SWITCH ON*
  Event: 'Shutdown' command received from host.
  Action: The power section is switched off immediately, and the motor is free to rotate if unbraked.

- *State Transition 9: OPERATION ENABLE _ SWITCH ON DISABLED*
  Event: 'Disable Voltage' command received from host.
  Action: The power section is switched off immediately, and the motor is free to rotate if unbraked.

- *State Transition 10: SWITCHED ON _ SWITCH ON DISABLED*
  Event: 'Disable Voltage' or 'Quick Stop' command received from host.
  Action: The power section is switched off immediately, and the motor is free to rotate if unbraked.

- *State Transition 11: OPERATION ENABLE _ QUICK STOP ACTIVE*
  Event: 'Quick Stop' command received from host.
  Action: The quick stop function is executed.

- *State Transition 12: QUICK STOP ACTIVE _ SWITCH ON DISABLED*
  Event: 'Quick Stop' is completed or 'Disable Voltage' command received from host.
  This transition is possible, if the Quick-Stop-Option-Code is different 5 (stay in the state ‘Quick Stop Active’).
  Action: The power section is switched off.

- *State Transition 13: All states _ FAULT REACTION ACTIVE*
  A fatal fault has occurred in the drive.
  Action: Execute appropriate fault reaction.

- *State Transition 14: FAULT REACTION ACTIVE _ FAULT*
  Event: The fault reaction is completed.
  Action: The drive function is disabled. The power section may be switched off.

- *State Transition 15: FAULT _ SWITCH ON DISABLED*
  Event: 'Fault Reset' command received from host.
  Action: A reset of the fault condition is carried out if no fault exists currently on the drive.
  After leaving the state Fault the Bit 'Fault Reset' of the *controlword* has to be cleared by the host.

- *State Transition 16: QUICK STOP ACTIVE _ OPERATION ENABLE*
  Event: 'Enable Operation' command received from host. This transition is possible if the Quick-Stop-Option-Code is 5 or 6.
  Action: The drive function is enabled.

Implementation-specific:

- *State Transition 17: OPERATION ENABLE_SWITCHED_ON*
  *OPERATION ENABLE_READY TO SWITCH*
  *OPERATION ENABLE_SWITCH ON DISABLED*
  Event: Time out time is expired.
  Action: One of the three possible states is activated.

- *State Transition 18: EVENT FAULT ACTION IS DONE*

**Notes:**
If a command is received which causes a change of state, this command must be processed completely and the new state attained before the next command can be processed.

| | |
|---|---|
| 'Drive function is disabled' | implies no energy is supplied to the motor. This may be achieved by different manufacturers in different ways. Reference values are not processed. |
| 'Drive function is enabled' | implies that energy can be supplied to the motor. The reference values (torque, velocity, position) are processed. |
| 'Fault occurred' | implies that a fault in the drive has occurred. In this case there is a transition to the state FAULT REACTION ACTIVE. In this state the device will execute a special fault reaction. After the execution of this fault reaction the device will switch to the state FAULT. This state can only be left by the command 'Fault Reset', but only if the fault is not active any more. |

## 6.2 Object Dictionary Entries

### 6.2.1 Objects defined in this Chapter

| Index | Object | Name | Type | Attr. | Save to EERPOM |
|---|---|---|---|---|---|
| $6040_h$ | VAR | *controlword* | Unsigned16 | rw | no |
| $6041_h$ | VAR | *statusword* | Unsigned16 | ro | no |
| $605A_h$ | VAR | *quick_stop_option_code* | Integer16 | rw | yes |
| $605B_h$ | VAR | *shutdown_option_code* | Integer16 | rw | yes |
| $605C_h$ | VAR | *disable_operation_option_code* | Integer16 | rw | yes |
| $605D_h$ | VAR | *halt_option_code* | Integer16 | rw | yes |
| $605E_h$ | VAR | *fault_reaction_option_code* | Integer16 | rw | yes |
| $6060_h$ | VAR | *modes_of_operation* | Integer8 | rw | no |
| $6061_h$ | VAR | *modes_of_operation_display* | Integer8 | ro | no |

## 6.3 Object Description

### 6.3.1 Object $6040_h$: *controlword*

The *controlword* consist of bits for:

- the controlling of the state,
- the controlling of operating modes and
- manufacturer specific options.

| **Index** | **$6040_h$** |
|---|---|
| Name | *controlword* |
| Object Code | VAR |
| Data Type | Unsigned16 |

**Value Description**

| Object Class | M: all | O: -l |
|---|---|---|
| Access | rw | |
| PDO Mapping | yes | |
| Units | - | |
| Value Range | 0...65535 | |
| Mandatory Range | - | |
| Default Value | - | |
| Substitute Value | - | |

**Data Description**

| MSB | | | | | | | | | | | | | | | LSB |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| High-Byte | | | | | | | | Low-Byte | | | | | | | |

The bits of the *controlword* are defined as follows:

| **Bit of the *controlword*** | | **Mandatory** |
|---|---|---|
| **Number** | **Name** | |
| 0 | switch_on | yes |
| 1 | enable_voltage | yes |
| 2 | quick_stop | yes |
| 3 | enable_operation | yes |
| 4 ... 6 | operation mode specific | |
| 7 | reset_fault | yes |
| 8 | halt | |
| 9, 10 | reserved | |
| 11 ... 15 | manufacturer specific (not used) | |

**Table 6.3.1: Bits in the *controlword***

**Bits 0 ... 3 and 7:**
Device control commands are triggered by the following bit patterns in the *controlword*:

| Command | Bit of the *controlword* | | | | | Transitions |
|---|---|---|---|---|---|---|
| | Bit 7 fault_ reset | Bit 3 enable_ operation | Bit 2 quick_ stop | Bit 1 enable_ voltage | Bit 0 switch_ on | |
| Shutdown | 0 | x | 1 | 1 | 0 | 6, 8 |
| Switch On | 0 | x | 1 | 1 | 1 | |
| Disable Voltage | 0 | x | x | 0 | x | 7, 9, 10, 12 |
| Quick Stop | 0 | x | 0 | 1 | x | 7, 10, 11 |
| Disable Operation | 0 | 0 | 1 | 1 | 1 | 5 |
| Enable Operation | 0 | 1 | 1 | 1 | 1 | 4, 16 |
| Fault Reset | rising edge | x | x | x | x | 15 |

**Table 6.3.2: Device Control Commands** (bits marked x are irrelevant)

**Bits 4, 5, 6 and 8:**
These bits are operation mode specific. The description is situated in the chapter of the special mode. The following table gives an overview:

| Bit | Velocity Mode (not supported) | **Profile Position Mode** | **Operation Profile Velocity Mode** | Mode Profile Torque Mode (not supported) | **Homing Mode** | Interpol. Position Mode (not supported) |
|---|---|---|---|---|---|---|
| 4 | (rfg_enable) | new_setpoint | reserved | reserved | homing operation start | (enable_ip_ mode) |
| 5 | (rfg_unlock) | change_set_ immediately | reserved | reserved | reserved | reserved |
| 6 | (rfg_use_ref) | abs/rel | reserved | reserved | reserved | reserved |
| 8 | (halt) | halt | halt | (halt) | halt | (halt) |

**Table 6.3.3: Mode Specific Bits in the *controlword***

rfg: running up frequency generator
halt: interrupts the move of a drive and wait for release to continue

**Bits 9, 10:**
These bits are reserved for further use. They are inactive by setting to zero. They have no special function, therefore they must be set to zero.

**Bits 11, 12, 13, 14 and 15:**
These bits are manufacturer specific, but are not used in this implementation.

### 6.3.2 Object $6041_h$ : *statusword*

The *statusword* indicates the current state of the drive. No bits are latched. The *statusword* consist of bits for:

- the current state of the drive,
- the operating state of the mode and
- manufacturer specific options.

| **Index** | **$6041_h$** |
|---|---|
| Name | *statusword* |
| Object Code | VAR |
| Data Type | Unsigned16 |

**Value Description**

| Object Class | M: all | O: - |
|---|---|---|
| Access | ro | |
| PDO Mapping | yes | |
| Units | - | |
| Value Range | 0 ... 65535 | |
| Mandatory Range | - | |
| Default Value | - | |
| Substitute Value | - | |

**Data Description**

| MSB | | | | | | | | | | | | | | | LSB |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| High-Byte | | | | | | | | Low-Byte | | | | | | | |

The following bits are defined in the *statusword*:

| Bit of the *statusword* | | Mandatory |
|---|---|---|
| **Number** | **Name** | |
| 0 | ready_to_switch_on | yes |
| 1 | switched_on | yes |
| 2 | operation_enabled | yes |
| 3 | fault | yes |
| 4 | voltage_enabled | yes |
| 5 | quick_stop | yes |
| 6 | switch_on_disabled | yes |
| 7 | warning | |
| 8 | reference active | |
| 9 | remote | yes |
| 10 | target_reached | yes |
| 11 | internal_limit_active | yes |
| 12, 13 | operation mode specific | |
| 14 | lower limit switch active | |
| 15 | upper limit switch active | |

**Table 6.3.4: Bits in the *statusword***

**Bits 0 ... 3, 5 and 6:**
The following bits indicate the status of the device:

| State | Bit of the *statusword* | | | | | |
|---|---|---|---|---|---|---|
| | **Bit 6** | **Bit 5** | **Bit 3** | **Bit 2** | **Bit 1** | **Bit 0** |
| | **switch_on_ disable** | **quick_ stop** | **fault** | **operation_ enable** | **switched_ on** | **ready_to_ switch_on** |
| NOT READY TO SWITCH ON | 0 | x | 0 | 0 | 0 | 0 |
| SWITCH ON DISABLED | 1 | x | 0 | 0 | 0 | 0 |
| READY TO SWITCH ON | 0 | 1 | 0 | 0 | 0 | 1 |
| SWITCHED ON | 0 | 1 | 0 | 0 | 1 | 1 |
| OPERATION ENABLED | 0 | 1 | 0 | 1 | 1 | 1 |
| QUICK STOP ACTIVE | 0 | 0 | 0 | 1 | 1 | 1 |
| FAULT REACTION ACTIVE | 0 | x | 1 | 1 | 1 | 1 |
| FAULT | 0 | x | 1 | 1 | 1 | 1 |

**Table 6.3.5: Device State Bits** (x ... irrelevant for this state)

**Bit 4: voltage_enabled**
High voltage is applied to the drive when this bit is set to 1.

**Bit 5: quick_stop**
When reset, this bit indicates that the drive is reacting on a quick stop request. Bits 0, 1 and 2 of the *statusword* must be set to 1 to indicate that the drive is capable to regenerate. The setting of the other bits indicates the status of the drive (e.g. the drive is performing a quick stop as result of a reaction to a non-fatal fault. The fault bit is set as well as bits 0, 1 and 2).

**Bit 7: warning**
A drive warning is present if bit 7 is set. The cause means no error but a state that has to be mentioned, e.g. temperature limit, job refused. The status of the drive does not change. The cause of this warning may be found by reading the fault code parameter. The bit is set and reset by the device.

**Bit 8: reference active**
This bit indicates the state of the reference:
‘0’ reference inactive
‘1’ reference active

**Bit 9: remote**

If bit 9 is set, then parameters may be modified via the CAN-network, and the drive executes the content of a command message. If the bit remote is reset, then the drive is in local mode and will not execute the command message. The drive may transmit messages containing valid actual values like a *position_actual_valu*e, depending on the actual drive configuration. The drive will accept accesses via service data objects (SDOs) in local mode.

**Bit 10: target_reached**

If bit 10 is set by the drive, then a setpoint has been reached. The setpoint is depended on the operating mode. The description is situated in the chapter of the special mode. The change of a target value by software alters this bit.

If *quick_stop_option_code* is 5 or 6 this bit must be set, when the quick stop operation is finished and the drive is halted.

If halt occurred and the drive has halted then this bit is set too.

**Bit 11: internal_limit_active**

This bit is set by the drive to indicate, that an internal limitation is active (e.g. *position_range_limit*).

**Bit 12 and 13:**

These bits are operation mode specific. The description is situated in the chapter of the special mode. The following table gives an overview:

| Bit | Operation Mode | | | | | |
|---|---|---|---|---|---|---|
| | Velocity Mode (not supported) | **Profile Position Mode** | **Profile Velocity Mode** | Profile Torque Mode (not supported) | **Homing Mode** | Interpol. Position Mode (not supported) |
| 12 | reserved | setpoint_ acknowledge | speed | reserved | homing_ attained | ip_mode_active |
| 13 | reserved | following_ error | max_slippage_ error | reserved | homing_error | reserved |

**Table 6.3.6: Mode Specific Bits in the *statusword***

**Bit 14: lower limit switch**

This bit indicates the state of the lower limit switch:

‘0’ limit switch inactive
‘1’ limit switch active

**Bit 15: upper limit switch**

This bit indicates the state of the upper limit switch:

‘0’ limit switch inactive
‘1’ limit switch active

### 6.3.3 Object 605B$_h$: *shutdown_option_code*

The parameter *shutdown_option_code* determines what action should be taken if there is a transition 'OPERATION ENABLE' -> 'READY TO SWITCH ON'.

| **Index** | **605B$_h$** |
|---|---|
| Name | *shutdown_option_code* |
| Object Code | VAR |
| Data Type | Integer16 |

**Value Description**

| | | |
|---|---|---|
| Object Class | M: - | O: all |
| Access | rw | |
| PDO Mapping | no | |
| Units | - | |
| Value Range | -32768 ... +32767 | |
| Mandatory Range | profile specific code | |
| Default Value | 0 | |
| Substitute Value | - | |

**Data Description**

| **Value of *shutdown_option_code*** | **Action** |
|---|---|
| -32768 ... -1 | manufacturer specific (not used) |
| 0 | disable drive function |
| 1 | slow down with slow down ramp and then disabling of the drive function |
| 2 ... 32767 | reserved |

### 6.3.4 Object 605C $_h$: *disable_operation_option_code*

The parameter *disable_operation_option_code* determines what action should be taken if there is a transition 'OPERATION ENABLE' -> 'SWITCHED ON'.

| **Index** | **605C $_h$** |
|---|---|
| Name | *disable_operation_option_code* |
| Object Code | VAR |
| Data Type | Integer16 |

**Value Description**

| | | |
|---|---|---|
| Object Class | M: - | O: all |
| Access | rw | |
| PDO Mapping | no | |
| Units | - | |
| Value Range | -32768 ... +32767 | |
| Mandatory Range | - | |
| Default Value | 1 | |
| Substitute Value | - | |

**Data Description**

| **Value of *disable_operation_option_code*** | **Action** |
|---|---|
| -32768 ... -1 | manufacturer specific (not used) |
| 0 | disable drive function |
| 1 | slow down with slow down ramp and then disabling of the drive function |
| 2 ... 32767 | reserved |

### 6.3.5 Object 605A$_h$: *quick_stop_option_code*

The parameter *quick_stop_option_code* determines what action should be taken if the Quick Stop Function is executed.

| **Index** | **605A$_h$** |
|---|---|
| Name | *quick_stop_option_code* |
| Object Code | VAR |
| Data Type | Integer16 |

**Value Description**

| | | |
|---|---|---|
| Object Class | M: - | O: all |
| Access | rw | |
| PDO Mapping | no | |
| Units | - | |
| Value Range | -32768 ... +32767 | |
| Mandatory Range | - | |
| Default Value | 2 | |
| Substitute Value | - | |

**Data Description**

| **Value of *quick_stop_option_code*** | **Action** |
|---|---|
| -32768 ... -1 | manufacturer specific (not used) |
| 0 | disable drive function |
| 1 | slow down on slow down ramp |
| 2 | slow down on quick stop ramp |
| 3 | n.i. |
| 4 | n.i. |
| 5 | slow down on slow down ramp and stay in QUICK STOP |
| 6 | slow down on quick stop ramp and stay in QUICK STOP |
| 7 | n.i. |
| 8 | n.i. |
| 9 ... 32767 | reserved |

n.i. ... not implemented

### 6.3.6 Object 605D$_h$: *halt_option_code*

The parameter *halt_option_code* determines what action should be taken if the bit 8 (halt) in the *controlword* is active.

| **Index** | **605D$_h$** |
|---|---|
| Name | *halt_option_code* |
| Object Code | VAR |
| Data Type | Integer16 |

**Value Description**

| Object Class | M: - | O: all |
|---|---|---|
| Access | rw | |
| PDO Mapping | no | |
| Units | - | |
| Value Range | -32768 ... +32767 | |
| Mandatory Range | - | |
| Default Value | 1 | |
| Substitute Value | - | |

**Data Description**

| **Value of *halt_option_code*** | **Action** |
|---|---|
| -32768 ... -1 | manufacturer specific (not used) |
| 0 | disable drive, motor is free to rotate |
| 1 | slow down on slow down ramp |
| 2 | slow down on quick stop ramp |
| 3 | n.i. |
| 4 | n.i. |
| 5 ... 32767 | reserved |

n.i. ... not implemented

### 6.3.7 Object 605E $_h$: *fault_reaction_option_code*

The parameter *fault_reaction_option_code* determines what action should be taken if a fault occurs in the drive (Communication Error).

| **Index** | **605E $_h$** |
|---|---|
| Name | *fault_reaction_option_code* |
| Object Code | VAR |
| Data Type | Integer16 |

**Value Description**

| | | |
|---|---|---|
| Object Class | M: - | O: all |
| Access | rw | |
| PDO Mapping | no | |
| Units | - | |
| Value Range | -32768 ... +32767 | |
| Mandatory Range | - | |
| Default Value | 2 | |
| Substitute Value | - | |

**Data Description**

| **Value of *fault_reaction_option_code*** | **Action** |
|---|---|
| -32768 ... -1 | manufacturer specific (not used) |
| 0 | disable drive, motor is free to rotate |
| 1 | slow down on slow down ramp |
| 2 | slow down on quick stop ramp |
| 3 | n.i. |
| 4 | n.i. |
| 5 ... 32767 | reserved |

n.i. ... not implemented

### 6.3.8 Object 6060 $_h$: *modes_of_operation*

The parameter *modes_of_operation* switches the actually chosen operation-mode.

| **Index** | **6060 $_h$** |
|---|---|
| Name | *modes_of_operation* |
| Object Code | VAR |
| Data Type | Integer8 |

**Value Description**

| Object Class | M: all | O: - |
|---|---|---|
| Access | rw | |
| PDO Mapping | yes | |
| Units | - | |
| Value Range | -128 ... +127 | |
| Mandatory Range | - | |
| Default Value | 0 | |
| Substitute Value | - | |

**Data Description**

| **Value of *modes_of_operation*** | **Action** |
|---|---|
| -1 ... -128 | manufacturer specific modes of operation (not used) |
| 0 | reserved |
| 1 | Profile Position Mode |
| 2 | n.i. |
| 3 | Profile Velocity Mode |
| 4 | n.i. |
| 5 | reserved |
| 6 | Homing Mode |
| 7 | n.i. |
| 8 ... 127 | reserved |

n.i. ... not implemented

**Note:**
A read of *modes_of_operation* shows only the value of *modes_of_operatio*n. The actual mode of the drive is reflected in the object *modes_of_operation_displa*y. It may be changed by writing to *modes_of_operation.*

### 6.3.9 Object 6061 $_h$: *modes_of_operation_display*

The *modes_of_operation_display* shows the current mode of operation. The meaning of the returned value corresponds to that of the *modes_of_operation* option code (index 6060$_h$ ).

| **Index** | **6061 $_h$** |
|---|---|
| Name | *modes_of_operation_display* |
| Object Code | VAR |
| Data Type | Integer8 |

**Value Description**

| | | |
|---|---|---|
| Object Class | M: all | O: - |
| Access | ro | |
| PDO Mapping | yes | |
| Units | - | |
| Value Range | -128 ... +127 | |
| Mandatory Range | - | |
| Default Value | - | |
| Substitute Value | - | |

**Data Description**
Same as for object 6060$_h$ *modes_of_operatio*n.

**Note:**
The actual mode is reflected in the *modes_of_operation_display* (index 6061$_h$ ), and not in the *modes_of_operation* (index 6060$_h$ ).

## 6.4 Functional Description

### 6.4.1 Modes of Operation Function

The device behaviour depends on the activated modes of operation.

Different device modes are implemented. Since it is not possible to operate the modes in parallel, the user is able to activate the required function by selecting a mode of operation. An example of exclusive functions are those for position and torque control, which can only control one variable at any one time. The variables can perform at most a limited function. Such hybrids are regarded as the particular characteristics of a mode of operation. Position control operation and encoder profile support can be active at the same time, for example. Consequently encoder profile support is not regarded as a mode of operation.

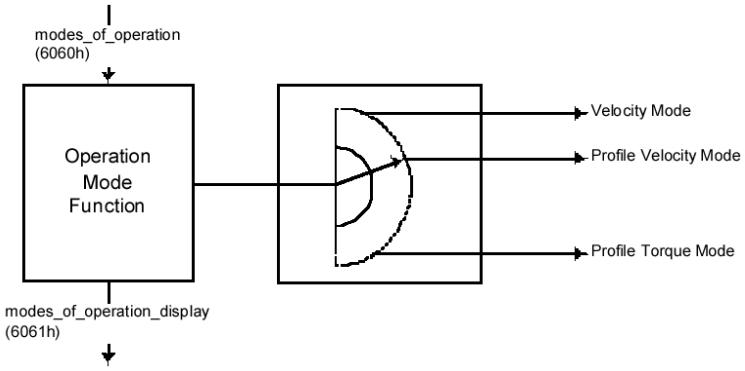


**Figure 6.4.1: Operation Mode Function**

It is possible for the user to switch between the various modes of operation. Switching is only allowed, when then motor is off.

The following modes of operation are implemented by esd:
- Profile Velocity Mode
- Homing Mode
- Profile Position Mode

With the exception of the ‘Homing Mode’, these listed modes of operation can all be put under the heading of 'setpoint setting'.

The option of manufacturer-specific operation modes is not used in this implementation.

The reference operation is regarded as a special form of a program function. The program function allows the user to run complex of time-critical sequence, e. g. tool change or special reference operations, directly in the device.

The switching between the modes of operation listed above should not incur any automatic reconfiguration of the process data channel. Problems which occur through switching of setpoint values during change of operating modes must be monitored by the user. If necessary they can be rectified by prior reconfiguration of the process data channel.

Two objects are defined for management of the modes of operation:
- *modes_of_operation*
- *modes_of_operation_display*

The *statusword* contains bits, whose meaning is dependent on the mode of operation. When switching the mode of operation, the bits changing their meaning need to be monitored.

### 6.4.2 Drive Disabling Function

The drive disabling function defines the behaviour of the drive when transitioning from the state OPERATION ENABLE to the state READY TO SWITCH ON (‘Shutdown’ command) or the state SWITCHED ON (‘Disable Operation’ command).

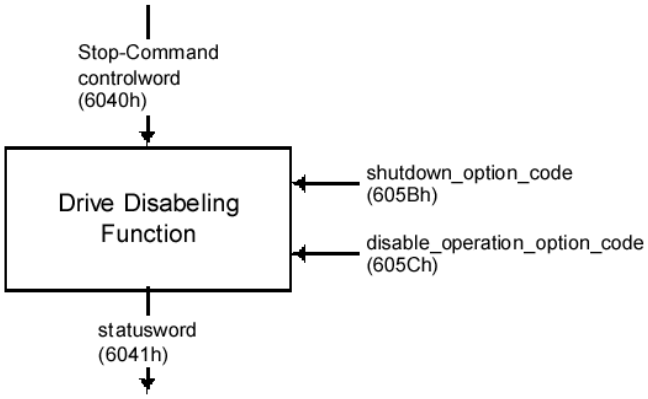


**Figure 6.4.2: Modes of Operation Function**

### 6.4.3 Quick Stop Function

The Quick Stop Function is triggered by the ‘Quick Stop’ command.

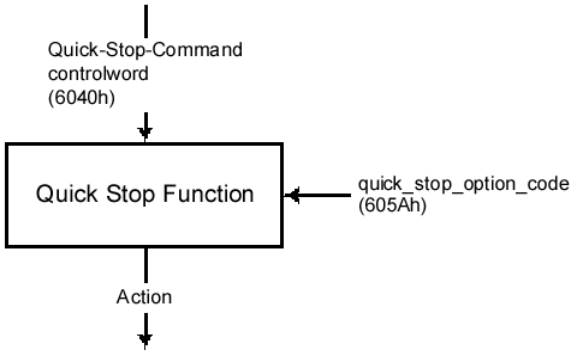


**Figure 6.4.3: Quick-Stop-Function**

### 6.4.4 Stop Function

The Stop Function may be triggered by resetting the bit 'RFG-disable' in the *controlword*.

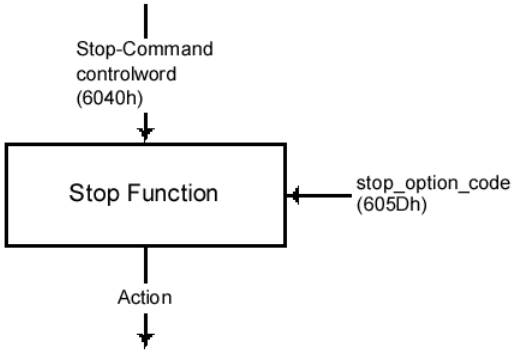


**Figure 6.4.4: Stop-Function**

### 6.4.5 Fault Reaction

Drive faults are classified as fatal or non-fatal faults.

#### 6.4.5.1 Fatal Faults

When a fatal fault occurs, the drive is no longer able to control the motor, so an immediate switch-off of the drive is executed.

#### 6.4.5.2 Non-Fatal Faults

When a non-fatal fault occurs, the drive can run the motor in a controlled fashion. The actions which are executed depend on the *fault_reaction_option_cod*e.

Once a fault occurs the drive will always enter the FAULT state, even if the fault clears before the drive enters the FAULT state. The FAULT state may only be left if the 'Fault Reset' command is received from a host, and no further fault is present in the drive.

# 7. Profile Position Mode

## 7.1 General Information

The overall structure for this mode is shown in the following figure.

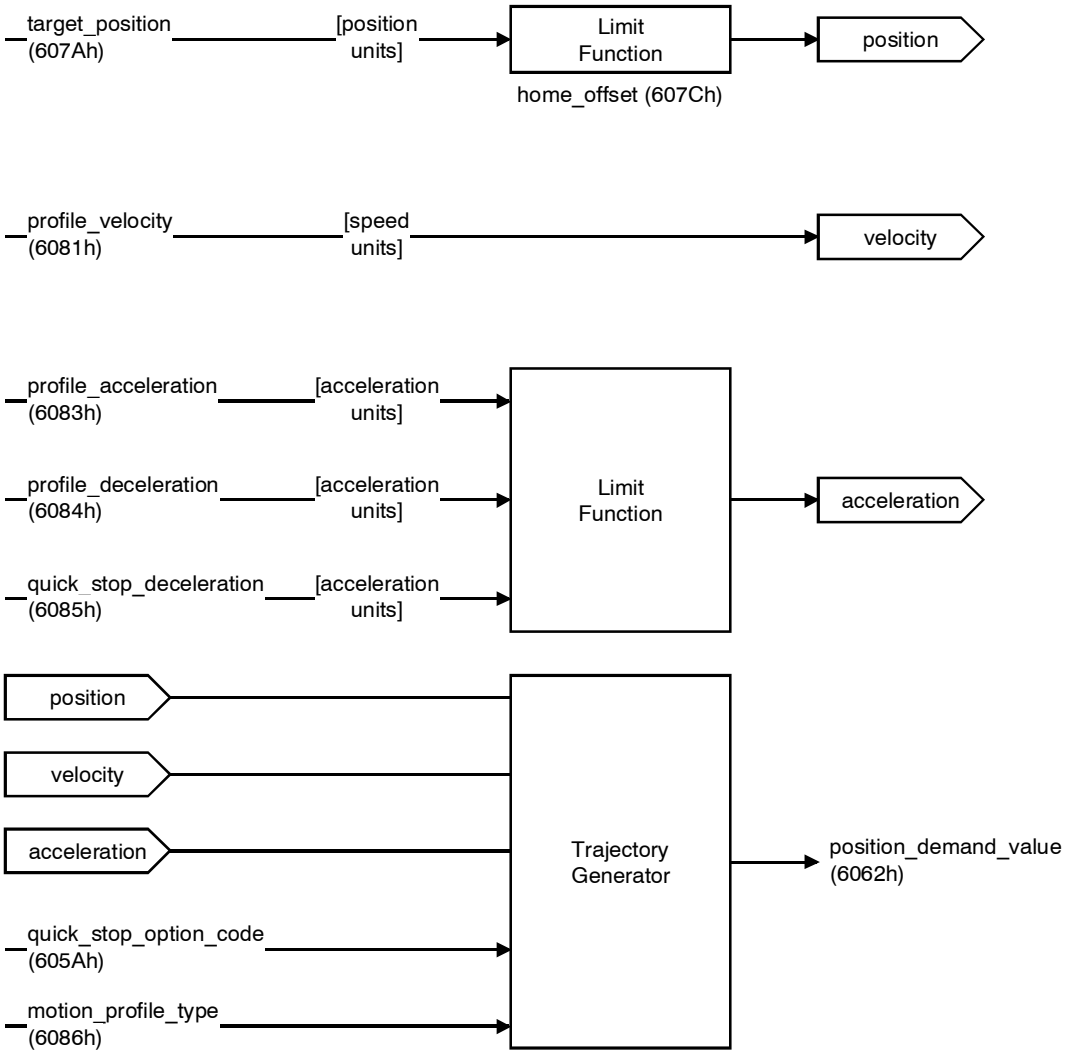


**Figure 7.1.1: Overall Structure for the Profile Position Mode**

### 7.1.1 Input Data Description

| Operating Mode | Input Parameters Used |
|---|---|
| Profile Position Mode | *target_position,*<br>*profile_velocity,*<br>*profile_acceleration, profile_deceleration,*<br>*quick_stop_deceleration,*<br>*quick_stop_option_code,*<br>*motion_profile_type* |

### 7.1.2 Output Data Description

| Operating Mode | Output Parameters Used |
|---|---|
| Profile Position Mode | *position_demand_value** |

### 7.1.3 Internal States

The Profile Position Mode will control by the bits of the *controlword* and *statusword*.

#### 7.1.3.1 *controlword* of Profile Position Mode

| Bit of the *controlword* | | | Meaning |
|---|---|---|---|
| **Number** | **Name** | **Value** | |
| 4 | new_setpoint | 0 | does not assume *target_position* |
| | | 1 | assume *target_position* |
| 5 | change_ set_immediately | 0 | not supported |
| | | 1 | interrupt the actual positioning and start the next positioning |
| 6 | abs/rel | 0 | *target_position* is an absolute value |
| | | 1 | *target_position* is a relative value |
| 8 | halt | 0 | execute positioning |
| | | 1 | stop axle with *profile_acceleration* |

**Table 7.1.1: Profile Position Mode Bits of the *controlword***

#### 7.1.3.2 *statusword* of Profile Position Mode

| Bit of the *statusword* | | | Meaning |
|---|---|---|---|
| **Number** | **Name** | **Value** | |
| 10 | target_reached | 0 | halt = 0: *target_position* not reached<br>halt = 1: axle decelerates |
| | | 1 | halt = 0: *target_position* reached<br>halt = 1: velocity of axle is 0 |
| 12 | setpoint_ acknowledge | 0 | Trajectory Generator has not assumed the positioning values (yet) |
| | | 1 | Trajectory Generator has assumed the positioning values |
| 13 | following_error | 0 | no following error |
| | | 1 | following error |

**Table 7.1.2: Profile Position Mode Bits of the *statusword***

## 7.2 Object Dictionary Entries

### 7.2.1 Implemented Objects defined in this Chapter

| Index | Object | Name | Type | Attr. | Save to EEPROM |
|---|---|---|---|---|---|
| $607A_h$ | VAR | *target_position* | Integer32 | rw | no |
| $607F_h$ | VAR | *max_profile_velocity* | Integer32 | ro | no |
| - | - | - | - | - | |
| $6081_h$ | VAR | *profile_velocity* | Unsigned32 | rw | no |
| $6083_h$ | VAR | *profile_acceleration* | Unsigned32 | rw | yes |
| $6084_h$ | VAR | *profile_deceleration* | Unsigned32 | rw | yes |
| $6085_h$ | VAR | *quick_stop_deceleration* | Unsigned32 | rw | yes |
| $6086_h$ | VAR | *motion_profile_type* | Integer16 | ro | no |

### 7.2.2 Objects defined in other Chapters

| Index | Object | Name | Type | Chapter |
|---|---|---|---|---|
| $6040_h$ | VAR | *controlword* | Integer16 | 66 |
| $6041_h$ | VAR | *statusword* | Unsigned16 | 66 |
| $605A_h$ | VAR | *quick_stop_option_code* | Integer16 | 66 |

## 7.3 Object Description

### 7.3.1 Object 607A$_h$: *target_position*

The *target_position* is the position that the drive should move to in position profile mode using the current settings of motion control parameters such as velocity, acceleration, deceleration, *motion_profile_type* etc. The *target_position* is given in user defined position units. The *target_position* will be interpreted as absolute or relative depending on the 'absolute_relative' flag (bit 6) in the *controlword*.

| **Index** | **607A$_h$** |
|---|---|
| Name | *target_position* |
| Object Code | VAR |
| Data Type | Integer32 |

**Value Description**

| Object Class | M: pp pc | O:- |
|---|---|---|
| Access | rw | |
| PDO Mapping | yes | |
| Units | position units | |
| Value Range | -($2^{31}$) ... +($2^{31}$-1) | |
| Mandatory Range | - | |
| Default Value | - | |
| Substitute Value | - | |

Range: 80 00 00 00$_h$ ... FF FF FF FF$_h$ ...00 00 00 00$_h$ ...7F FF FF FF$_h$

Unit = 1/64 Full_Step

e.g. F_Step_Pos: 1 = 00 00 00 40$_h$
2 = 00 00 00 80$_h$
3 = 00 00 00 C0$_h$
4 = 00 00 01 00$_h$

### 7.3.2 Object 607F$_h$: *max_profile_velocity*

The *max_profile_velocity* depends on the parameter *min_gear*.

| **Index** | **607F $_h$** |
|---|---|
| Name | *max_profile_velocity* |
| Object Code | VAR |
| Data Type | Integer32 |

**Value Description**

| Object Class | M: pv pv | O:- |
|---|---|---|
| Access | ro | |
| PDO Mapping | yes | |
| Units | velocity units | |
| Value Range | 0 ... +($2^{23}$-1) | |
| Mandatory Range | - | |
| Default Value | - | |
| Substitute Value | - | |

| *min_gear* | *max_profile_velocity* [Full-Steps/s] (CPU-Clock = 20 MHz) |
|---|---|
| 0 | 30000.0 |
| 1 | 15000.0 |
| 2 | 7500.0 |
| 3 | 3750.0 |
| 4 | 1875.0 |
| 5 | 937.0 |
| 6 | 468.8 |

**Table 7.3.1:** Values of *max_profile_velocity*

### 7.3.3 Object $6081_h$ : *profile_velocity*

The *profile_velocity* is the velocity normally attained at the end of the acceleration ramp during a profiled move and is valid for both directions of motion.

| **Index** | **$6081_h$** |
|---|---|
| Name | *profile_velocity* |
| Object Code | VAR |
| Data Type | Unsigned32 |

**Value Description**

| | | |
|---|---|---|
| Object Class | M: pp pv | O:- |
| Access | rw | |
| PDO Mapping | yes | |
| Units | velocity units | |
| Value Range | 1... f(min_gear) | |
| Mandatory Range | - | |
| Default Value | - | |
| Substitute Value | - | |

**Velocity units**

Range: $00\ 00\ 00\ 01_h$ ... [function of (min_gear)]

Velocity unit: 1/256 Steps/s

e.g.: 5000 Steps/s -> $00\ 13\ 88\ 00_h$
5000.5 Steps/s -> $00\ 13\ 88\ 80_h$

### 7.3.4 Object $6083_h$: *profile_acceleration*

The *profile_acceleration* is given in user defined acceleration units. It is converted to position increments per second ² using the normalizing factors.

| **Index** | **$6083_h$** |
|---|---|
| Name | *profile_acceleration* |
| Object Code | VAR |
| Data Type | Unsigned32 |

**Value Description**

| Object Class | M: pp pv | O:- |
|---|---|---|
| Access | rw | |
| PDO Mapping | no | |
| Units | acceleration units | |
| Value Range | 0 ... +($2^{16}$-1) | |
| Mandatory Range | - | |
| Default Value | 5000 | |
| Substitute Value | - | |

Acceleration units: 1 Full_Step/s²

Maximum: $2^{16}$ -1 Full_Step/s²

### 7.3.5 Object $6084_h$: *profile_deceleration*

The *profile_deceleration* is given in the same units as *profile_acceleratio*n. If this parameter is not supported, then the *profile_acceleration* value is also used for deceleration.

| |
|---|
| **Note:** This object is only used in the 'Profile Velocity Mode'. |

| **Index** | **$6084_h$** |
|---|---|
| Name | *profile_deceleration* |
| Object Code | VAR |
| Data Type | Unsigned32 |

**Value Description**

| | | |
|---|---|---|
| Object Class | M: pp pv | O:- |
| Access | rw | |
| PDO Mapping | no | |
| Units | acceleration units | |
| Value Range | 0 ... +($2^{16}$-1) | |
| Mandatory Range | - | |
| Default Value | 10000 | |
| Substitute Value | - | |

Acceleration units: 1 Full_Step/s²

Maximum: $2^{16}$ -1 Full_Step/s²

### 7.3.6 Object $6085_h$: *quick_stop_deceleration*

The *quick_stop_deceleration* is the deceleration used to stop the motor if the 'Quick Stop' command (bit 2 of the *controlword*) is given and the *quick_stop_option_code* (605Ah) is set to 2. The *quick_stop_deceleration* is given in the same units as the *profile_acceleratio*n.

| **Index** | **$6085_h$** |
|---|---|
| Name | *quick_stop_deceleration* |
| Object Code | VAR |
| Data Type | Unsigned32 |

**Value Description**

| Object Class | M: - | O: pp pv hm |
|---|---|---|
| Access | rw | |
| PDO Mapping | no | |
| Units | acceleration units | |
| Value Range | 0 ... +($2^{16}$-1) | |
| Mandatory Range | - | |
| Default Value | 10000 | |
| Substitute Value | - | |

Acceleration units: 1 Full_Step/s²

Maximum: $2^{16}$ -1 Full_Step/s²

### 7.3.7 Object $6086_h$: *motion_profile_type*

The *motion_profile_type* is used to select the type of motion profile used to perform a profiled move.

| **Index** | **$6086_h$** |
|---|---|
| Name | *motion_profile_type* |
| Object Code | VAR |
| Data Type | Integer16 |

**Value Description**

| | | |
|---|---|---|
| Object Class | M: pp pv | O:- |
| Access | ro | |
| PDO Mapping | no | |
| Units | none | |
| Value Range | -32768 ... +32767 | |
| Mandatory Range | - | |
| Default Value | 0 | |
| Substitute Value | 0 | |

**Data Description**

| **Profile Code** | **Profile Type** |
|---|---|
| -32768 ... -1 | manufacturer specific (not used) |
| 0 | linear ramp (trapezoidal profile) |
| 1 ... 3 | not implemented |
| 4 ... 32767 | reserved for future profile types |

## 7.4 Functional Description

The *target_positions* are applied to the drive by the 'Single setpoint' mode:
After reaching the *target_position* the drive unit signals this status to a host computer and then receives a new setpoint. After reaching a *target_position* the velocity normally is reduced to zero before starting a move to the next setpoint.

The single setpoint mode is controlled by the timing of the bits 'new_setpoint' and 'change_set_immediately' in the *controlword* and 'setpoint_acknowledge' in the *statusword*. These bits allow to set up a request-response mechanism in order to prepare a set of setpoints while another set still is processed in the drive unit. This minimizes reaction times within a control program on a host computer.

**Note:**
The mode 'Set of setpoints' is not supported!

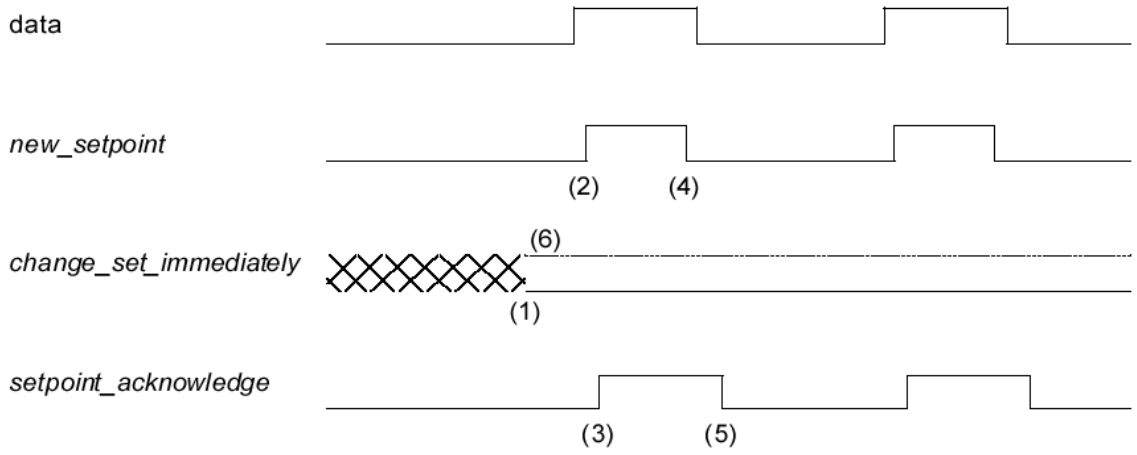


**Figure 7.4.1: Setpoint Transmission from a Host Computer**

The figure above shows the function of the Single setpoint mode. The initial status of the bit 'change_set_immediately' in the *controlword* determines, if the 'Set of setpoint' or the 'Single Setpoint' mode is used.

If the bit 'change_set_immediately' is '0' a single setpoint is expected by the drive(1). After data is applied to the drive, a host signals that the data is valid by changing the bit 'new_setpoint' to '1' in the *controlword* (2). The drive responds with 'setpoint_acknowledge' set to '1' in the *statusword* (3) after it recognized and buffered the new valid data. Now the host may release 'new_setpoint' (4) and afterwards the drive signals with 'setpoint_acknowledge' equal '0' its ability to accept new data again (5). In the following figure this mechanism results in a velocity of zero after ramping down in order to reach a *target_position* $x_1$ at $t_1$. After signalling to the host, that the setpoint is reached like described above, the next *target_position* $x_2$ is processed at $t_2$ and reached at $t_3$.

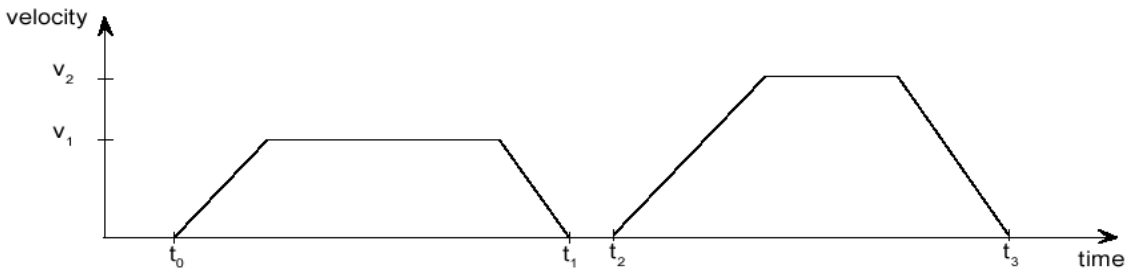


**Figure 7.4.2: Single Setpoint**

# 8. Homing Mode

## 8.1 General Information

This chapter describes the method by which a drive seeks the home position (also called, the datum, reference point or zero point). There are various methods of achieving this using limit switches at the ends of travel or a home switch (zero point switch) in mid-travel, most of the methods also use the index (zero) pulse train from an incremental encoder.

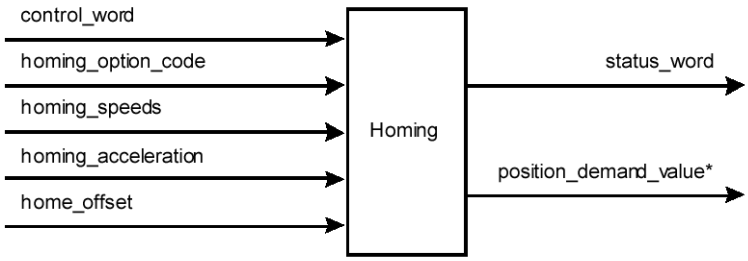


**Figure 8.1.1: The Homing Function**

### 8.1.1 Input Data Description

The user can specify the speeds, acceleration and the method of homing. There is a further object *home_offset* which allows the user to displace zero in the user's coordinate system from the home position.

There are two *homing_speed*s; in a typical cycle the faster speed is used to find the home switch and the slower speed is used to find the index pulse. The manufacturer is allowed some discretion in the use of these speeds as the response to the signals may be dependent upon the hardware used.

### 8.1.2 Output Data Description

There is no output data except for those bits in the *statusword* which return the status or result of the homing process and the demand to the position control loops.

### 8.1.3 Internal States

The Homing Mode will control by the bits of the *controlword* and *statusword*.

#### 8.1.3.1 *controlword* of Homing Mode

| Bit of the *controlword* | | | Meaning |
|---|---|---|---|
| **Number** | **Name** | **Value** | |
| 4 | homing_operation_start | 0 | Homing Mode inactive |
| | | 0 -> 1 | start Homing Mode |
| | | 1 | Homing Mode active |
| | | 1 -> 0 | Interrupt Homing Mode |
| 5 | | | reserved |
| 6 | | | reserved |
| 8 | halt | 0 | execute the instruction of bit 4 |
| | | 1 | stop axle with *homing_acceleration* |

**Table 8.1.1: Homing Mode Bits of the *controlword***

#### 8.1.3.2 *statusword* of Homing Mode

| Bit of the *statusword* | | | Meaning |
|---|---|---|---|
| **Number** | **Name** | **Value** | |
| 10 | target_reached | 0 | halt = 0: home position not reached<br>halt = 1: axle decelerates |
| | | 1 | halt = 0: home position reached<br>halt = 1: axle has velocity 0 |
| 12 | homing_attained | 0 | Homing Mode not yet completed |
| | | 1 | Homing Mode carried out successfully |
| 13 | homing_error | 0 | no homing error |
| | | 1 | Homing error occurred; Homing Mode carried out not successfully; the error cause is found by reading the error code |

**Table 8.1.2: Homing Mode Bits of the *statusword***

## 8.2 Object Dictionary Entries

### 8.2.1 Objects defined in this Chapter

| Index | Object | Name | Type | Attr. | Save to EEPROM |
|---|---|---|---|---|---|
| $607C_h$ | VAR | *home_offset* | Integer32 | rw | yes |
| $6098_h$ | VAR | *homing_method* | Integer8 | rw | yes |
| $6099_h$ | ARRAY | *homing_speeds* | Unsigned32 | rw | yes |
| $609A_h$ | VAR | *homing_acceleration* | Unsigned32 | rw | yes |

### 8.2.2 Objects defined in other Chapters

| Index | Object | Name | Type | Chapter |
|---|---|---|---|---|
| $6040_h$ | VAR | *controlword* | Integer16 | 6 |
| $6041_h$ | VAR | *statusword* | Unsigned16 | 6 |

## 8.3 Object Description

### 8.3.1 Object 607C$_h$: *home_offset*

The *home_offset* object is the difference between the zero position for the application and the machine home position (found during homing). During homing the machine home position is found and once the homing is completed the zero position is offset from the home position by adding the *home_offset* to the home position. All subsequent absolute moves shall be taken relative to this new zero position. This is illustrated in the following diagram.

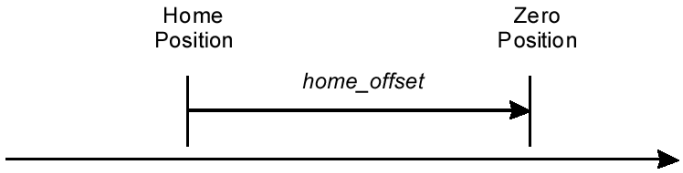


**Figure 8.3.1: Home Offset**

If the *home_offset* is not implemented then it shall be zero.

| **Index** | **607C$_h$** |
|---|---|
| Name | *home_offset* |
| Object Code | VAR |
| Data Type | Integer32 |

**Value Description**

| | | |
|---|---|---|
| Object Class | M: - | O: hm |
| Access | rw | |
| PDO Mapping | no | |
| Units | position units | |
| Value Range | -($2^{31}$) ... +($2^{31}$-1) | |
| Mandatory Range | - | |
| Default Value | 0 | |
| Substitute Value | 0 | |

### 8.3.2 Object $6098_h$: *homing_method*

The *homing_method* object determines the method that will be used during homing.

| **Index** | **$6098_h$** |
|---|---|
| Name | *homing_method* |
| Object Code | VAR |
| Data Type | Integer8 |

**Value Description**

| | | |
|---|---|---|
| Object Class | M: hm | O:- |
| Access | rw | |
| PDO Mapping | no | |
| Units | - | |
| Value Range | -128 ... +127 | |
| Mandatory Range | 0 | |
| Default Value | 17 | |
| Substitute Value | 0 | |

**Data Description**

| **Value of *homing_method*** | **Meaning** |
|---|---|
| -128 ... -1 | manufacturer specific (not used) |
| 0 | no homing operation required |
| 1...16, 23 ... 34 | not implemented homing methods |
| 17... 22, 35 | implemented homing methods (see the functional description) |
| 36 ... 127 | reserved |

### 8.3.3 Object $6099_h$: *homing_speeds*

This entry in the object dictionary defines the speeds used during homing.

| **Index** | $6099_h$ |
|---|---|
| Name | *homing_speeds* |
| Object Code | ARRAY |
| Number of Elements | 2 |
| Data Type | Unsigned32 |

**Value Description**

| Subindex | $01_h$ | |
|---|---|---|
| Description | speed_during_search_for_switch | |
| Object Class | M: hm | O: - |
| Access | rw | |
| PDO Mapping | no | |
| Units | velocity units | |
| Value Range | 0 ... +($2^{23}$-1) | |
| Mandatory Range | - | |
| Default Value | 00 13 88 $88_h$ = 5000.53 Full_Steps/s | |
| Substitute Value | 0 | |

| Subindex | $02_h$ | |
|---|---|---|
| Description | speed_during_search_for_zero | |
| Object Class | M: hm | O: - |
| Access | rw | |
| PDO Mapping | no | |
| Units | velocity units | |
| Value Range | 0 ... +($2^{23}$-1) | |
| Mandatory Range | - | |
| Default Value | 00 05 88 $88_h$ = 1416.53 Full_Steps/s | |
| Substitute Value | 0 | |

Range: 00 00 00 $00_h$ ...00 7F FF $FF_h$

Velocity unit: 1/256 Steps/s

e.g.: 5000 Steps/s -> 00 13 88 $00_h$
5000.5 Steps/s -> 00 13 88 $80_h$

### 8.3.4 Object 609A$_h$: *homing_acceleration*

The *homing_acceleration* establishes the acceleration to be used for all accelerations and decelerations with the standard homing modes.

| **Index** | 609A $_h$ |
|---|---|
| Name | *homing_acceleration* |
| Object Code | VAR |
| Data Type | Unsigned32 |

**Value Description**

| Object Class | M: - | O: hm |
|---|---|---|
| Access | rw | |
| PDO Mapping | no | |
| Units | acceleration units | |
| Value Range | 0 ... +($2^{16}$-1) | |
| Mandatory Range | - | |
| Default Value | 5000 | |
| Substitute Value | - | |

Acceleration units: 1 Full_Step/s²

Maximum: $2^{16}$ -1 Full_Step/s²

## 8.4 Functional Description

By choosing a method of homing by writing a value to *homing_method* will clearly establish

- the homing signal (positive limit switch, negative limit switch, home switch)
- the direction of actuation and where appropriate
- the position of the index pulse.

The home position and the zero position are offset by the *home_offset*, see the definition of *home_offset* for how this offset is used.

Various homing positions are illustrated in the following diagrams. An encircled number indicates the code for selection of this homing position. The direction of movement is also indicated.

There are four sources of homing signal available, these are the negative and positive limit switches, the home switch and the index pulse from an encoder:

In the diagrams of homing sequences shown below, the encoder count increases as the axle`s position moves to the right, in other words the left is the minimum position and the right is the maximum position.

For the operation of positioning drives, an exact knowledge of the absolute position is normally required. Since for cost reasons, drives often do not have an absolute encoder, a homing operation is necessary. There are several, application-specific methods. The *homing_method* is used for selection.

The exact sequence of the homing operation is clearly described by the method. In some circumstances, a device has several methods to choose from, using the homing_method.

### 8.4.1 Homing Methods

The following sub-sections describe the details of how each of the homing modes shall function. Only homing mode without index pulse are supported.

#### 8.4.1.1 Method 17: Homing on the Negative Limit Switch

Using this method the initial direction of movement is leftward if the negative limit switch is inactive (here shown as low). The home position is where the negative limit switch becomes inactive again.

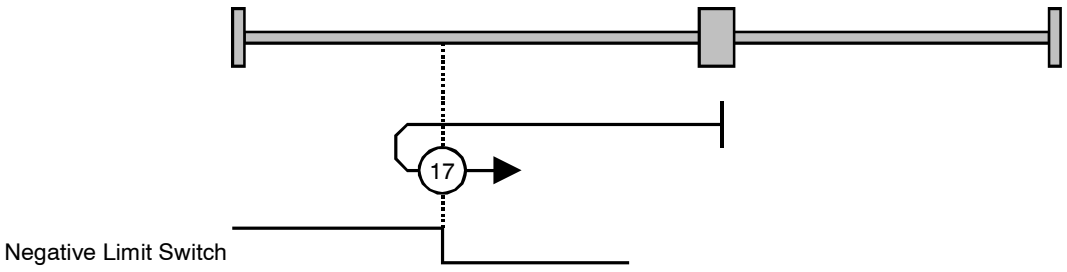


**Figure 8.4.1: Homing on the Negative Limit Switch**

### 8.4.1.2 Method 18: Homing on the Positive Limit Switch

Using this method the initial direction of movement is rightward if the positive limit switch is inactive (here shown as low). The position of home is where the positive limit switch becomes inactive again.

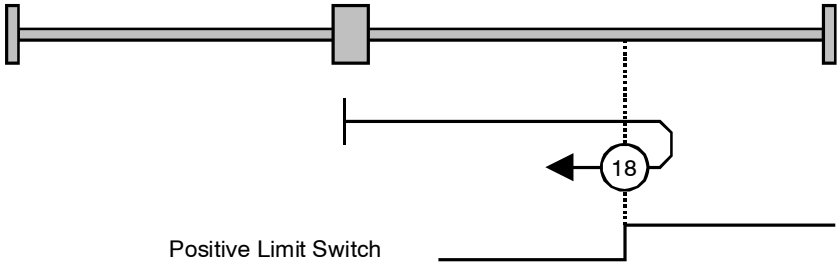


**Figure 8.4.2: Homing on the Positive Limit Switch**

### 8.4.1.3 Methods 19 and 20: Homing on the Positive Home Switch

Using methods 19 or 20 the initial direction of movement is dependent on the state of the home switch.

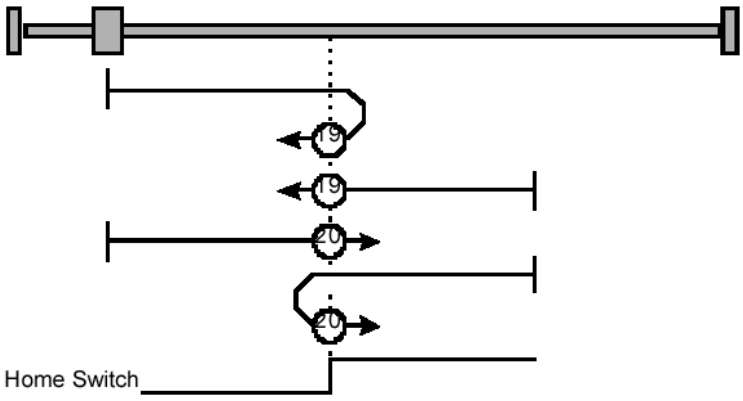


**Figure 8.4.3: Homing on the Positive Home Switch**

### 8.4.1.4 Methods 21 and 22: Homing on the Negative Home Switch

Using methods 21 or 22 the initial direction of movement is dependent on the state of the home switch.

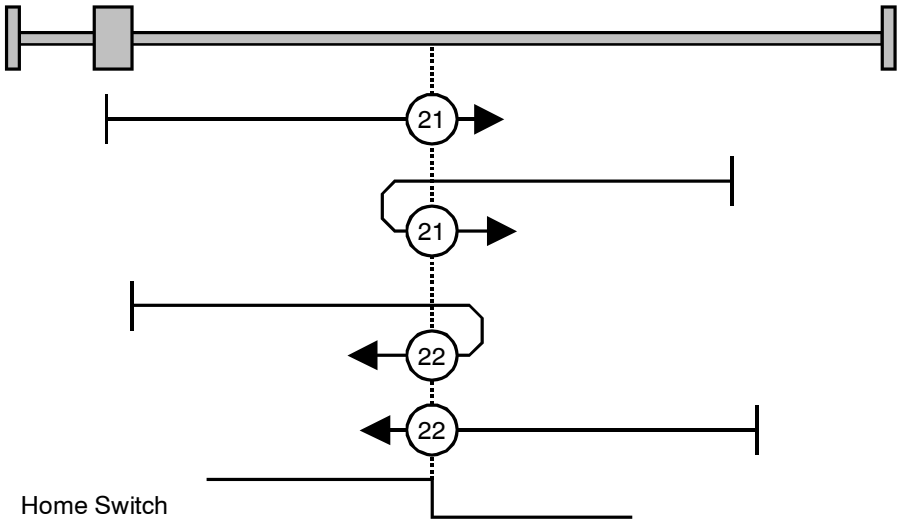


**Figure 8.4.4: Homing on the Negative Home Switch**

### 8.4.1.5 Method 35: Homing on the Current Position

In method 35 the current position is taken to be the home position.

# 9. Profile Velocity Mode

## 9.1 General Information

The Profile Velocity Mode includes the following sub functions:

- demand value input via Trajectory Generator
- velocity capture using velocity sensor
- velocity control function with appropriate input and output signals
- limitation of *torque_demand_value*
- monitoring of the *profile_velocity* using a window-function
- monitoring of *velocity_actual_value* using a threshold

The operation of the reference value generator and its input parameters:

- *profile_velocity*
- *profile_acceleration*
- *profile_deceleration*
- *emergency_stop* and
- *motion_profile_type*

are described in the Profile Position Mode.

Various sensors can be used for velocity capture. In particular the aim is that costs should be reduced and the system should be simplified by evaluating position and velocity using a common sensor, such as is possible using a resolver or an encoder.

The velocity control function is not specified more precisely at this point as it is highly manufacturer specific, but the format and maximum number of control coefficients are established.

Monitoring functions for the *velocity_actual_value* provide status information for super-ordinated systems.

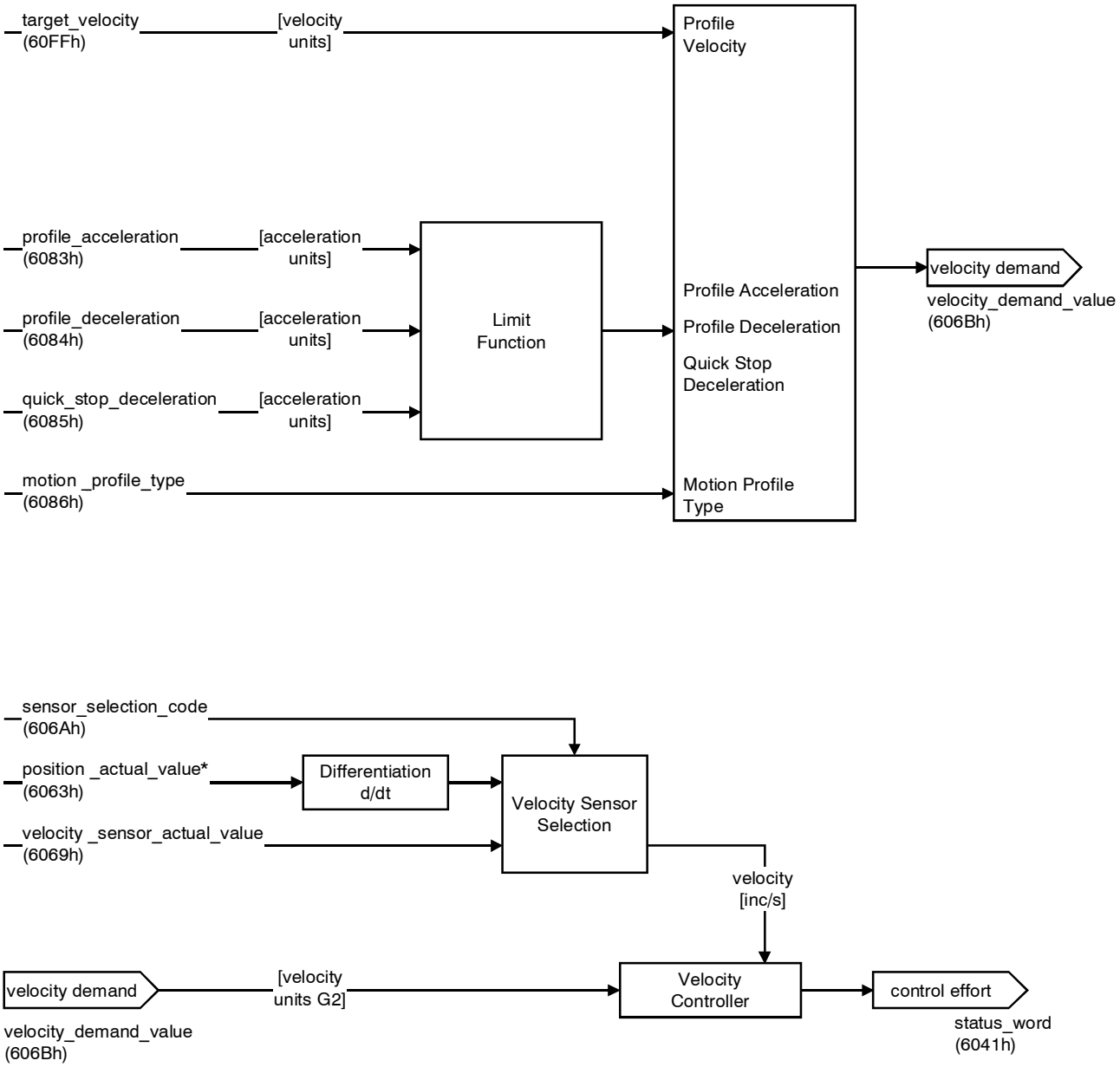


**Figure 8.4.5: Structure of the Profile Velocity Mode**

### 9.1.1 Input Data Description

| Operating Mode | Input Parameters Used |
|---|---|
| Profile Velocity Mode | *target_velocity,*<br>*profile_acceleration,*<br>*profile_deceleration,*<br>*quick_stop_deceleration,*<br>*quick_stop_option_code,*<br>*motion_profile_type,*<br>*position_actual_value** |

### 9.1.2 Output Data Description

| Operation Mode | Output Parameter used |
|---|---|
| Profile Velocity Mode | *velocity_actual_value, velocity_demand_value, statusword* |

### 9.1.3 Internal States

The Profile Velocity Mode will control by the bits of the *controlword* and *statusword*.

#### 9.1.3.1 *controlword* of Profile Velocity Mode

| Bit of the *controlword* | | | Meaning |
|---|---|---|---|
| Number | Name | Value | |
| 4 | | | reserved |
| 5 | | | reserved |
| 6 | | | reserved |
| 8 | halt | 0 | execute motion |
| | | 1 | stop axle with *profile_deceleration* |

**Table 9.1.1: Profile Velocity Bits of the controlword**

### 9.1.3.2 *statusword* of Profile Velocity Mode

| Number | Name | Value | |
|---|---|---|---|
| 10 | target_reached | 0 | halt = 0: *target_velocity* not (yet) reached<br>halt = 1: axle decelerates |
| | | 1 | halt = 0: *target_velocity* reached<br>halt = 1: axle has the velocity 0 |
| 12 | speed | 0 | speed is not equal 0 |
| | | 1 | speed is equal 0 |
| 13 | max_slippage_error | 0 | maximal slippage not reached |
| | | 1 | maximal slippage reached (not implemented) |

**Table 9.1.2: Profile Velocity Mode Bits of the statusword**

## 9.2 Object Dictionary Entries

### 9.2.1 Objects defined in this Chapter

In the DS-402 the objects $6062_h$ , $6063_h$ and $6064_h$ are defined in the chapter 'Position Control Function'. Because these are the only objects of the chapter 'Position Control Function', in this document the objects are described within the chapter Profile Velocity Mode.

| Index | Object | Name | Type | Attr. | Save to EEPROM |
|---|---|---|---|---|---|
| ($6062_h$) | VAR | *position_demand_value* | Integer32 | ro | no |
| $6063_h$ | VAR | *position_actual_value** | Integer32 | ro | no |
| $6064_h$ | VAR | *position_actual_value* | Integer32 | ro | no |
| $6069_h$ | VAR | *velocity_sensor_actual_value* | Integer32 | ro | no |
| $606A_h$ | VAR | *sensor_selection_code* | Integer16 | ro | no |
| $606B_h$ | VAR | *velocity_demand_value* | Integer32 | ro | no |
| $606C_h$ | VAR | *velocity_actual_value* | Integer32 | ro | no |
| - | - | - | - | - | - |
| $60FF_h$ | VAR | *target_velocity* | Integer32 | rw | no |

(Objects shown in brackets are not yet supported by the described firmware version.)

**Note:**
The objects $6069_h$, $606B_h$ and $606C_h$ have the same function in this implementation. They use the same variable '*v_ist*'.

### 9.2.2 Objects defined in other Chapters

| Index | Object | Name | Type | Chapter |
|---|---|---|---|---|
| $6040_h$ | VAR | *controlword* | Integer16 | 6 |
| $6041_h$ | VAR | *statusword* | Unsigned16 | 6 |
| $6083_h$ | VAR | *profile_acceleration* | Unsigned32 | 7 |
| $6084_h$ | VAR | *profile_deceleration* | Unsigned32 | 7 |
| $6085_h$ | VAR | *quick_stop_deceleration* | Unsigned32 | 7 |

## 9.3 Object Description

### 9.3.1 Object 6062 $_h$: *position_demand_value*

**Note:** This object is not yet supported by the described firmware version.

| **Index** | **6062 $_h$** |
|---|---|
| Name | *position_demand_value* |
| Object Code | VAR |
| Data Type | Integer32 |

**Value Description**

| Object Class | M: - | O: pp |
|---|---|---|
| Access | ro | |
| PDO Mapping | no | |
| Units | position units | |
| Value Range | $-(2^{31})$ ... $+(2^{31}-1)$ | |
| Mandatory Range | - | |
| Default Value | - | |
| Substitute Value | - | |

### 9.3.2 Object 6063 $_h$: *position_actual_value**

The actual value of the position measurement device is one of the two input values of the closed loop position control. The data unit is defined as increments.

**Note:**
The objects $6063_h$ and $6064_h$ have the same function in this implementation.

| **Index** | **6063 $_h$** |
|---|---|
| Name | *position_actual_value** |
| Object Code | VAR |
| Data Type | Integer32 |

**Value Description**

| | | |
|---|---|---|
| Object Class | M: - | O: pp, hm |
| Access | ro | |
| PDO Mapping | yes | |
| Units | increments | |
| Value Range | $-(2^{31})$ ... $+(2^{31}-1)$ | |
| Mandatory Range | - | |
| Default Value | - | |
| Substitute Value | - | |

Resolution: 1/64 Full_Step

### 9.3.3 Object $6064_h$: *position_actual_value*

This object represents the actual value of the position measurement device in user defined units.

**Note:** The objects $6063_h$ and $6064_h$ have the same function in this implementation.

| **Index** | **$6064_h$** |
|---|---|
| Name | *position_actual_value* |
| Object Code | VAR |
| Data Type | Integer32 |

**Value Description**

| Object Class | M: - | O: pp, hm |
|---|---|---|
| Access | ro | |
| PDO Mapping | yes | |
| Units | increments | |
| Value Range | $-(2^{31})$ ... $+(2^{31}-1)$ | |
| Mandatory Range | - | |
| Default Value | - | |
| Substitute Value | - | |

Resolution: 1/64 Full_Step

### 9.3.4 Object 6069 $_h$: *velocity_sensor_actual_value*

The *velocity_sensor_actual_value* returned the **real velocity** value that is determined by the software depending on a given timer preload value.

| **Index** | **6069 $_h$** |
|---|---|
| Name | *velocity_sensor_actual_value* |
| Object Code | VAR |
| Data Type | Integer32 |

**Value Description**

| | | |
|---|---|---|
| Object Class | M: pv | O: - |
| Access | ro | |
| PDO Mapping | no | |
| Units | velocity units | |
| Value Range | -($2^{23}$) ... +($2^{23}$-1) | |
| Mandatory Range | - | |
| Default Value | - | |
| Substitute Value | - | |

Range: 00 7F FF FF$_h$ (+max)
:
00 00 00 00$_h$ 0
:
FF 80 00 00$_h$ (-max)

Velocity unit: 1/256 Steps/s

e.g.: 5000 Steps/s -> 00 13 88 00$_h$
5000.5 Steps/s -> 00 13 88 80$_h$

### 9.3.5 Object 606A$_h$: *sensor_selection_code*

The source of the *velocity_sensor_actual_value* can be determined using the *sensor_selection_code.* This determines whether a differentiated position signal or the signal from a separate velocity sensor is to be evaluated.

| **Index** | **606A$_h$** |
|---|---|
| Name | *sensor_selection_code* |
| Object Code | VAR |
| Data Type | Integer16 |

**Value Description**

| | | |
|---|---|---|
| Object Class | M: pv | O:- |
| Access | ro | |
| PDO Mapping | no | |
| Units | - | |
| Value Range | -32768 ... +32767 | |
| Mandatory Range | - | |
| Default Value | 1 | |
| Substitute Value | - | |

**Data Description**

| Selection Code | Meaning of the Selection Function |
|---|---|
| -32768 ... -1 | manufacturer specific (not used) |
| 0 | not implemented |
| 1 | velocity actual value from velocity encoder |
| 2 ... 32767 | reserved for other profiles |

### 9.3.6 Object 606B$_h$: *velocity_demand_value*

The output value of the Trajectory Generator may be corrected by the output value of the Position Control Function. It is then provided as a demand value for the velocity controller.

| **Index** | **606B $_h$** |
|---|---|
| Name | *velocity_demand_value* |
| Object Code | VAR |
| Data Type | Integer32 |

**Value Description**

| | | |
|---|---|---|
| Object Class | M: pv | O:- |
| Access | ro | |
| PDO Mapping | no | |
| Units | velocity units | |
| Value Range | -($2^{23}$) ... +($2^{23}$-1) | |
| Mandatory Range | - | |
| Default Value | - | |
| Substitute Value | - | |

**Note:** The objects 606B$_h$ and 606C$_h$ have the same function in this implementation. They use the same variable '*v_ist*'.

Range: 00 7F FF FF$_h$ (+max)
:
00 00 00 00$_h$ 0
:
FF 80 00 00$_h$ (-max)

Velocity unit: 1/256 Steps/s

e.g.: 5000 Steps/s -> 00 13 88 00$_h$
5000.5 Steps/s -> 00 13 88 80$_h$

### 9.3.7 Object 606C$_h$: *velocity_actual_value*

The *velocity_actual_value* is also represented in velocity units and is coupled to the velocity used as input to the velocity controller.

| **Index** | **606C $_h$** |
|---|---|
| Name | *velocity_actual_value* |
| Object Code | VAR |
| Data Type | Integer32 |

**Value Description**

| | | |
|---|---|---|
| Object Class | M: pv | O:- |
| Access | ro | |
| PDO Mapping | yes | |
| Units | velocity units | |
| Value Range | -($2^{23}$) ... +($2^{23}$-1) | |
| Mandatory Range | - | |
| Default Value | - | |
| Substitute Value | - | |

**Note:** The objects 606B$_h$ and 606C$_h$ have the same function in this implementation. They use the same variable '*v_ist*'.

Range: 00 7F FF FF$_h$ (+max)
:
00 00 00 00$_h$ 0
:
FF 80 00 00$_h$ (-max)

Velocity unit: 1/256 Steps/s

### 9.3.8 Object 60FF$_h$: *target_velocity*

The *target_velocity* is the input for the Trajectory Generator.

| **Index** | **60FF$_h$** |
|---|---|
| Name | *target_velocity* |
| Object Code | VAR |
| Data Type | Integer32 |

**Value Description**

| Object Class | M: pv | O:- |
|---|---|---|
| Access | rw | |
| PDO Mapping | yes (Default: Rx-PDO4) | |
| Units | velocity units | |
| Value Range | -($2^{23}$) ... +($2^{23}$-1) | |
| Mandatory Range | - | |
| Default Value | - | |
| Substitute Value | - | |

Range: 00 7F FF FF$_h$ (+max)
:
00 00 00 00$_h$ 0
:
FF 80 00 00$_h$ (-max)

Velocity unit: 1/256 Steps/s

# 10. Manufacturer Specific Objects

## 10.1 General Information

This chapter describes the objects that have been implemented in addition to the DS-402 objects for motion control.

## 10.2 Object Dictionary Entries

### 10.2.1 Objects defined in this Chapter

| Index | Object | Name | Type | Attr. | Default |
|---|---|---|---|---|---|
| $2800_h$ | VAR | *my_state* | Unsigned8 | ro | - |
| $2801_h$ | VAR | *op_mode* | Unsigned8 | ro | - |
| $2802_h$ | VAR | *i_ist* | Unsigned8 | ro | - |
| $2803_h$ | VAR | *i_nom* | Unsigned8 | rw | $06_h$ |
| $2804_h$ | VAR | *i_low* | Unsigned8 | rw | $86_h$ |
| $2805_h$ | VAR | *brake_on_delay* | Unsigned8 | rw | 0 |
| $2806_h$ | VAR | *brake_off_delay* | Unsigned8 | rw | 0 |
| $2807_h$ | VAR | *t_auto_shutdown* | Unsigned8 | rw | $250_d$ |
| $2808_h$ | VAR | *auto_shut_opt* | Unsigned8 | rw | 3 |
| $2809_h$ | VAR | *min_gear* | Unsigned8 | rw | 1 |
| $280A_h$ | VAR | *max_gear* | Unsigned8 | rw | 2 |
| $280B_h$ | VAR | *es_option* | Unsigned8 | rw | 3 |
| $280C_h$ | VAR | *pdo_pos_mask* | Unsigned8 | rw | 4 |
| $2810_h$ | VAR | *es_lower_opt* | Unsigned16 | rw | 7 |
| $2811_h$ | VAR | *es_upper_opt* | Unsigned16 | rw | 7 |
| $2812_h$ | VAR | *reference_on* | Unsigned16 | rw | 7 |
| $2820_h$ | VAR | *ref_position* | Signed32 | ro | - |
| $2880_h$ | VAR | *temperature* | Signed16 | ro | - |
| $2881_h$ | VAR | *drive_supply_voltage* | Unsigned16 | ro | - |
| ($2890_h$) | VAR | *abs_norm_current* | Unsigned16 | ro | - |
| ($2891_h$) | VAR | *abs_phase_current_a* | Unsigned16 | ro | - |
| ($2892_h$) | VAR | *abs_phase_current_b* | Unsigned16 | ro | - |
| $28FF_h$ | VAR | *esd_mode* | Unsigned8 | rw | - |

(Objects shown in brackets are not yet supported by the described firmware version.)

## 10.3 Object Description

### 10.3.1 Object $2800_h$: *my_state*

| **Index** | **2800 h** |
|---|---|
| Name | *my_state* |
| Object Code | VAR |
| Data type | Unsigned8 |

**Value Description**

| **Subindex** | **00 h** |
|---|---|
| Description | state of 402-machine |
| Access | ro |
| PDO-Mapping | no |
| Units | - |
| Value Range | 0...5 |
| Mandatory Range | - |
| Default Value | 0 |
| Substitute Value | - |

The different states are displayed in the state diagram at page 44.

### 10.3.2 Object $2801_h$ : *op_mode*

**Note:** This object has the same content as the object $6060_h$ *modes_of_operation*.

| **Index** | **$2801_h$** |
|---|---|
| Name | *op_mode* |
| Object Code | VAR |
| Data type | Unsigned8 |

**Value Description**

| **Subindex** | **$00_h$** |
|---|---|
| Description | internal info only: actual operating mode |
| Access | ro |
| PDO-Mapping | no |
| Units | - |
| Value Range | 0...n |
| Mandatory Range | - |
| Default Value | 0 |
| Substitute Value | - |

### 10.3.3 Object $2802_h$: *i_ist*

| **Index** | **$2802_h$** |
|---|---|
| Name | *i_ist* |
| Object Code | VAR |
| Data type | Unsigned8 |

**Value Description**

| **Subindex** | **$00_h$** |
|---|---|
| Description | internal info only: actual current of stepper |
| Access | ro |
| PDO-Mapping | no |
| Units | (units see page 126) |
| Value Range | (value see current table at page 126) |
| Mandatory Range | - |
| Default Value | n.a. |
| Substitute Value | - |

### 10.3.4 Object $2803_h$: *i_nom*

| **Index** | **$2803_h$** |
|---|---|
| Name | *i_nom* |
| Object Code | VAR |
| Data type | Unsigned8 |

**Value Description**

| **Subindex** | **$00_h$** |
|---|---|
| Description | phase current *I_Phase* in active state (value see current table at page 126) |
| Access | rw |
| PDO-Mapping | no |
| Units | (units see page 126) |
| Value Range | $00_h$... $0F_h$ |
| Mandatory Range | - |
| Default Value | 6 |
| Substitute Value | - |

### 10.3.5 Object $2804_h$: *i_low*

| **Index** | **$2804_h$** |
|---|---|
| Name | *i_low* |
| Object Code | VAR |
| Data type | Unsigned8 |

**Value Description**

| **Subindex** | **$00_h$** |
|---|---|
| Description | phase current in inactive state |
| Access | rw |
| PDO-Mapping | no |
| Units | (units see page 127) |
| Value Range | $y0_h$... $yF_h$ (value description see page 127) |
| Mandatory Range | - |
| Default Value | 86 |
| Substitute Value | - |

**Note:**
The minimum value of *i_low* is (0.52/16) A. If you want to set the current to 0, set the motor to one of the power disabled states.

### 10.3.6 Object $2805_h$: *brake_on_delay*

| **Index** | **$2805_h$** |
|---|---|
| Name | *brake_on_delay* |
| Object Code | VAR |
| Data type | Unsigned8 |

**Value Description**

| **Subindex** | **$00_h$** |
|---|---|
| Description | delay after setting brake ‘on’ |
| Access | rw |
| PDO-Mapping | no |
| Units | msec |
| Value Range | 0 ... +255 |
| Mandatory Range | - |
| Default Value | 0 |
| Substitute Value | - |

### 10.3.7 Object $2806_h$: *brake_off_delay*

| **Index** | **$2806_h$** |
|---|---|
| Name | *brake_off_delay* |
| Object Code | VAR |
| Data type | Unsigned8 |

**Value Description**

| **Subindex** | **$00_h$** |
|---|---|
| Description | delay after setting brake ‘off’ |
| Access | rw |
| PDO-Mapping | no |
| Units | ms |
| Value Range | 0 ... +255 |
| Mandatory Range | - |
| Default Value | 0 |
| Substitute Value | - |

### 10.3.8 Object $2807_h$: *t_auto_shutdown*

| **Index** | **2807 h** |
|---|---|
| Name | *t_auto_shutdown* |
| Object Code | VAR |
| Data type | Unsigned8 |

**Value Description**

| **Subindex** | **00 h** |
|---|---|
| Description | timeout for ‘no movement state’ |
| Access | rw |
| PDO-Mapping | no |
| Units | 0.1 s |
| Value Range | 0 ... +255<br>0: function disabled |
| Mandatory Range | - |
| Default Value | 250 |
| Substitute Value | - |

### 10.3.9 Object $2808_h$: *auto_shut_opt*

| **Index** | **2808 h** |
|---|---|
| Name | *auto_shut_opt* |
| Object Code | VAR |
| Data type | Unsigned8 |

**Value Description**

| **Subindex** | **00 h** |
|---|---|
| Description | if no movement return to state x after ‘T_Auto_Shutdown’ |
| Access | rw |
| PDO-Mapping | no |
| Units | - |
| Value Range | 0 ... 3 |
| Mandatory Range | - |
| Default Value | 3 |
| Substitute Value | - |

### 10.3.10 Object $2809_h$: *min_gear*

| **Index** | **$2809_h$** |
|---|---|
| Name | *min_gear* |
| Object Code | VAR |
| Data type | Unsigned8 |

**Value Description**

| **Subindex** | **$00_h$** |
|---|---|
| Description | step setting for minimum gear |
| Access | rw |
| PDO-Mapping | no |
| Units | - |
| Value Range | 0 ... 6 |
| Mandatory Range | - |
| Default Value | 1 |
| Substitute Value | - |

#### 10.3.10.1 Value of Parameter *min_gear* and *max_gear*

| ***min_gear, max_gear*** | **Description** |
|---|---|
| 0 | Full_Step |
| 1 | Half_Step |
| 2 | Quarter_Step |
| 3 | 1/8_Step |
| 4 | 1/16_Step |
| 5 | 1/32_Step |
| 6 | 1/64_Step |

**Table 10.3.1:** Value of parameters *min_gear* and *max_gear*

### 10.3.11 Object 280A$_h$: *max_gear*

| **Index** | **280A$_h$** |
|---|---|
| Name | *max_gear* |
| Object Code | VAR |
| Data type | Unsigned8 |

**Value Description**

| **Subindex** | **00$_h$** |
|---|---|
| Description | step setting for maximum gear |
| Access | rw |
| PDO-Mapping | no |
| Units | - |
| Value Range | 0 ... 6 (value description see page 111) |
| Mandatory Range | - |
| Default Value | 2 |
| Substitute Value | - |

### 10.3.12 Object 280B$_h$: *es_option*

| **Index** | **280B $_h$** |
|---|---|
| Name | *es_option* |
| Object Code | VAR |
| Data type | Unsigned8 |

**Value Description**

| **Subindex** | **00 $_h$** |
|---|---|
| Description | goto state n after ‘Stop_On_ES’ |
| Access | rw |
| PDO-Mapping | no |
| Units | - |
| Value Range | 0 ... 4 |
| Mandatory Range | - |
| Default Value | 4 |
| Substitute Value | - |

This parameter defines the state the module should enter, after the selected ‘ramp_down’ function is completed. Some stepper motors need a hold current to stay on their position, therefore it may be necessary to provide this current even if the limit switch is reached.

### 10.3.13 Object 280C$_h$: *pdo_pos_mask*

| **Index** | **280C $_h$** |
|---|---|
| Name | *pdo_pos_mask* |
| Object Code | VAR |
| Data type | Unsigned8 |

**Value Description**

| **Subindex** | **00 $_h$** |
|---|---|
| Description | quantifies the position change limit that must be reached to generate an event |
| Access | rw |
| PDO-Mapping | no |
| Units | - |
| Value Range | 0 ... FF$_h$ |
| Mandatory Range | - |
| Default Value | 3 |
| Substitute Value | - |

This parameter quantifies the position change limit that must be reached to generate an event. It is implemented to define a permitted tolerance of the actual position to provide the CAN bus from unnecessary load by too many position control messages.

The lower byte of the position value is compared with the value defined in *pdo_pos_mask:*

If (position_LL AND *pdo_pos_mask*) == ‘00’ THEN generate Event

The following table shows examples of the change limit values:

| *pdo_pos_mask*<br>D7... ...D0 | Event if the following limit is exceeded |
|---|---|
| 0000.0000 | each micro step |
| 0000.0001 | each 1/32 step |
| 0000.0011 | each 1/16 step |
| : | : |
| 0111.1111 | 2 full step |
| 1111.1111 | 4 full step |

**Table 10.3.2:** Values of parameter *pdo_pos_mask*

### 10.3.14 Object $2810_h$: *es_lower_opt*

| **Index** | **$2810_h$** |
|---|---|
| Name | *es_lower_opt* |
| Object Code | VAR |
| Data type | Unsigned16 |

**Value Description**

| **Subindex** | **$00_h$** |
|---|---|
| Description | determination mode of lower limit switch |
| Access | rw |
| PDO-Mapping | no |
| Units | - |
| Value Range | 0 ... $D7_h$ (values description see page 118) |
| Mandatory Range | - |
| Default Value | 7 |
| Substitute Value | - |

### 10.3.15 Object $2811_h$ : *es_upper_opt*

| **Index** | **$2811_h$** |
|---|---|
| Name | *es_upper_opt* |
| Object Code | VAR |
| Data type | Unsigned16 |

**Value Description**

| **Subindex** | **$00_h$** |
|---|---|
| Description | determination mode of upper limit switch |
| Access | rw |
| PDO-Mapping | no |
| Units | - |
| Value Range | 0 ... $D7_h$ (values description see table at page 118) |
| Mandatory Range | - |
| Default Value | 7 |
| Substitute Value | - |

### 10.3.16 Object $2812_h$: *reference_opt*

| **Index** | **$2812_h$** |
|---|---|
| Name | *reference_opt* |
| Object Code | VAR |
| Data type | Unsigned16 |

**Value Description**

| **Subindex** | **$00_h$** |
|---|---|
| Description | determination mode of reference input |
| Access | rw |
| PDO-Mapping | no |
| Units | - |
| Value Range | 0 ... $D7_h$ (values description see table at page 118) |
| Mandatory Range | - |
| Default Value | 7 |
| Substitute Value | - |

### 10.3.16.1 Bits of parameters *es_lower_opt*, *es_upper_opt* and *reference_opt*

| Register Bit | Name | Description | Level Assignment | |
|---|---|---|---|---|
| D7 | L_->_H | latch actual position counter into *Ref_Position* latch at rising edge of switch input | 0 | no latch |
| | | | 1 | latch at rising edge |
| D6 | H_->_L | latch actual position counter into *Ref_Position* at falling edge of switch input | 0 | no latch |
| | | | 1 | latch at falling edge |
| D5 | Trigger | trigger transmit PDO3 on change | 0 | no trigger |
| | | | 1 | trigger transmit |
| D4 | Act_State | definition of active state of limit switch input/reference input | 0 | 1 (5V) is active |
| | | | 1 | 0 (0V) is active |
| D3 | - | reserved | set always to '0' | |
| D2<br>D1<br>D0 | Stop_Mode | Stop Mode acc. to objects $605A_h$ ... $605E_h$ | see table below | |

**Table 10.3.3:** Bits of parameters *ES_lower_Opt*, *ES_upper_Opt* and *Reference_Opt*

### 10.3.16.2 Stop-Modes (acc. to object $605D_h$ )

| Bits of stop_mode | | | Action |
|---|---|---|---|
| D2 | D1 | D0 | |
| 0 | 0 | 0 | Disable drive function, motor is free to rotate |
| 0 | 0 | 1 | Slow down with slow down ramp and then execute the *es_option* (object $280B_h$ ) |
| 0 | 1 | 0 | Slow down on quick stop ramp and then execute the *es_option* (object $280B_h$ ) |
| 0 | 1 | 1 | (not implemented) |
| 1 | 0 | 0 | |
| 1 | 0 | 1 | |
| 1 | 1 | 0 | |
| 1 | 1 | 1 | ignore ES/Ref (i.e. no action if reference or limit switch is entered) |

**Table 10.3.4:** Implemented Stop Modes

### 10.3.17 Object $2820_h$: *ref_position*

| **Index** | **$2820_h$** |
|---|---|
| Name | *ref_position* |
| Object Code | VAR |
| Data type | Signed 32 |

**Value Description**

| **Subindex** | **$00_h$** |
|---|---|
| Description | position latched by edge of limit switches esl / esu or reference switch |
| Access | ro |
| PDO-Mapping | no |
| Units | - |
| Value Range | Pos_Units $-(2^{31})$ ... $+(2^{31}-1)$ |
| Mandatory Range | - |
| Default Value | n.a. |
| Substitute Value | - |

### 10.3.18 Object $2880_h$: *temperature*

| **Index** | **$2880_h$** |
|---|---|
| Name | *temperature* |
| Object Code | VAR |
| Data type | Signed 16 |

**Value Description**

| **Subindex** | **$00_h$** |
|---|---|
| Description | temperature at the STEPCON-1H modules |
| Access | ro |
| PDO-Mapping | no |
| Units | 1/256 °C |
| Value Range | 0000 ... $FFFF_h$ |
| Mandatory Range | - |
| Default Value | n.a. |
| Substitute Value | - |

Temperature unit: *temperature* returned in steps of 1/256 °C

Temperature range:

| | | |
|---|---|---|
| 8000 | ... | -128.0 °C |
| : | | |
| 0 | ... | 0 °C |
| : | | |
| 7FFF | ... | +127.99 °C |

### 10.3.19 Object $2881_h$: *drive_supply_voltage*

| **Index** | **$2881_h$** |
|---|---|
| Name | *drive_supply_voltage* |
| Object Code | VAR |
| Data type | Unsigned16 |

**Value Description**

| **Subindex** | **$00_h$** |
|---|---|
| Description | drive supply voltage in [V] |
| Access | ro |
| PDO-Mapping | no |
| Units | 1/256 V |
| Value Range | 0000 ... $7FFF_h$ |
| Mandatory Range | - |
| Default Value | n.a. |
| Substitute Value | - |

*drive_supply_voltage* voltage in steps of 1/256 V
i.e.
$18\ 26_{hex} = 24_{dez} + (38_{dez} / 256_{dez}) = 24.148$ V

### 10.3.20 Object $2890_h$: *abs_norm_current*

**Note:** This object is not yet supported by the described firmware version.

| **Index** | **$2890_h$** |
|---|---|
| Name | *abs_norm_current* |
| Object Code | VAR |
| Data type | Unsigned16 |

**Value Description**

| **Subindex** | **$00_h$** |
|---|---|
| Description | absolute normal current |
| Access | ro |
| PDO-Mapping | no |
| Units | - |
| Value Range | 0000 ... FFFF |
| Mandatory Range | - |
| Default Value | n.a. |
| Substitute Value | - |

### 10.3.21 Object $2891_h$: *abs_phase_current_a*

**Note:** This object is not yet supported by the described firmware version.

| **Index** | **$2890_h$** |
|---|---|
| Name | *abs_phase_current_a* |
| Object Code | VAR |
| Data type | Unsigned16 |

**Value Description**

| **Subindex** | **$00_h$** |
|---|---|
| Description | absolute phase current A |
| Access | ro |
| PDO-Mapping | no |
| Units | - |
| Value Range | 0000 ... FFFF |
| Mandatory Range | - |
| Default Value | n.a. |
| Substitute Value | - |

### 10.3.22 Object $2892_h$: *abs_phase_current_b*

**Note:** This object is not yet supported by the described firmware version.

| **Index** | **$2890_h$** |
|---|---|
| Name | *abs_phase_current_b* |
| Object Code | VAR |
| Data type | Unsigned16 |

**Value Description**

| **Subindex** | **$00_h$** |
|---|---|
| Description | absolute phase current B |
| Access | ro |
| PDO-Mapping | no |
| Units | - |
| Value Range | 0000 ... FFFF |
| Mandatory Range | - |
| Default Value | n.a. |
| Substitute Value | - |

### 10.3.23 Object 28FF$_h$: *esd_mode*

| **Index** | **28FF $_h$** |
|---|---|
| Name | *esd_mode* |
| Object Code | VAR |
| Data type | Unsigned8 |

**Value Description**

| **Subindex** | **00 $_h$** |
|---|---|
| Description | status / config-bits |
| Access | rw |
| PDO-Mapping | no |
| Units | - |
| Value Range | 00 ... 07$_h$ |
| Mandatory Range | - |
| Default Value | 0 |
| Substitute Value | - |

#### 10.3.23.1 Bits of Parameter *esd_mode*

| Register Bit | Name | Default State after RESET | Description | Level Assignment | |
|---|---|---|---|---|---|
| D7 : D2 | - | 0 | reserved | set always to '0' | |
| D2 | AdcEnable | 0 | enable temperature/VCC measurement | 0<br>1 | disable<br>enable |
| D1 | ValData | 0 | valid data in object *Ref_Position* (must be reset by software) | 0<br>1 | no valid data<br>valid data |
| D0 | LSB_first | 0 | position and velocity values in PDOs are LSB-first (CANopen specific) or MSB-first (esd-mot specific) | 0<br>1 | MSB-first<br>LSB-first |

**Table 10.3.5:** Bits of parameter *esd_mode*

### 10.3.24 Current of Stepper

The value of the stepper's current is determined within a one byte parameter as follows:

**Lower Nibble (all current value objects)**

The lower nibble is determined as the index value in the table for stepper phase current *I_Phase*.

| ***index*** | ***I_Phase*** **[A]** |
|---|---|
| 0 | 0.58 |
| 1 | 0.64 |
| 2 | 0.72 |
| 3 | 0.80 |
| 4 | 0.92 |
| 5 | 1.05 |
| 6 | 1.18 |
| 7 | 1.35 |
| 8 | 1.47 |
| 9 | 1.65 |
| 10 | 1.85 |
| 11 | 2.06 |
| 12 | 2.36 |
| 13 | 2.70 |
| 14 | 3.00 |
| 15 | (3.50) Not in Driver_Spec!! |

Current values have a tolerance of ± 10% !

**Table 10.3.6:** Current setting by parameter *index*

**Upper Nibble**

The upper nibble is only determined in the object 2804.0 *I_Low*. With the upper nibble an additional factor is determined, that is used to reduce the resulting current value. In the object description the upper nibble is named 'y'.

| **Upper Nibble y** | ***factor*** |
|---|---|
| 0 | 1.0 |
| 1...15 | y/16 |

The current is determined according to the following equation:

$$I\_Phase\_low = I\_Phase \cdot factor$$

e.g. *I_Low* = $46_h$:

>> *I_Phase* = 1.2 A (1.18 A ± 10 %)
*factor* = 4/16 = 0.25
>> *I_Phase_low* = 0.3A