Object Description and Operation Modes CANopen MovingCap

Version 1.1

Date: 2013-09-02 / EN 2019-05-02

Company: Fullmo GmbH

Author: Daniel Wetzel

Fullmo GmbH
Robert-Bosch-Str. 12
88677 Markdorf

49-(0)7 54 4 / 30 73 39-0
Fax: +49-(0)7 54 4 / 30 73 39-9
E-mai: info@fullmo.de

Table of Contents

List of Figures

List of Tables

Definitions, acronyms, abbreviations

References

1 Controlling the power drive system

1.1 Overview

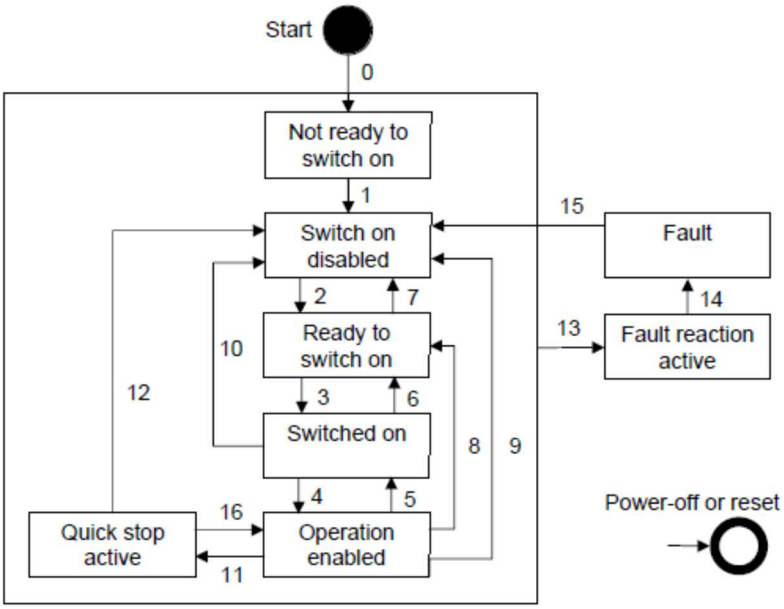


Abbildung 1: Controlling the power drive system [1]

Implemented states:

- Not ready to switch on
- Switch on disabled
- Ready to switch on
- Switched on
- Operation enabled
- Quick stop active
- digestion
- Fault reaction active

Implemented transitions:

- 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16

1.2 Implemented Objects

1.2.1 Object 6040h: Controlword

The object was implemented for the following operating modes:

- Profile Position Mode

- Profiles Velocity Mode
- Homing Mode

No manufacturer-specific adjustments were made.

1.2.2 Object 6041h: Statusword

The object was implemented for the following operating modes:

- Profile Position Mode
- Profiles Velocity Mode
- Homing Mode

No manufacturer-specific adjustments were made.

1.2.3 Object 605Ah: Quick stop option code

The object supports the Codes 0 - 8 option.

No manufacturer specific option codes have been created.

1.2.4 Object 605Bh: Shutdown option code

The object supports the Codes 0 - 1 option.

No manufacturer specific option codes have been created.

1.2.5 Object 605Ch: Disable operation option code

The object was supported the option Codes 0 - 1.

No manufacturer specific option codes have been created.

1.2.6 Object 605Dh: Stop option code

The object supports the option Codes 0 - 4.

No manufacturer specific option codes have been created.

1.2.7 Object 605Eh: Fault reaction option code

The object supports the option Codes 0 - 4.

No manufacturer specific option codes have been created.

1.2.8 Object 6060h: Modes of operation

The object supports the following modes:

- 1: Profile Position Mode
- 3: Profile Velocity Mode
- 6: Homing Mode
- -1: Internal Service Mode (manufacturer-specific)

1.2.9 Object 6061h: Modes of operation display

The object supports the following modes:

- 1: Profile Position Mode
- 3: Profile Velocity Mode
- 6: Homing Mode

- -1: Internal Service Mode (manufacturer-specific)

2 Profile Position Mode

2.1 Overview

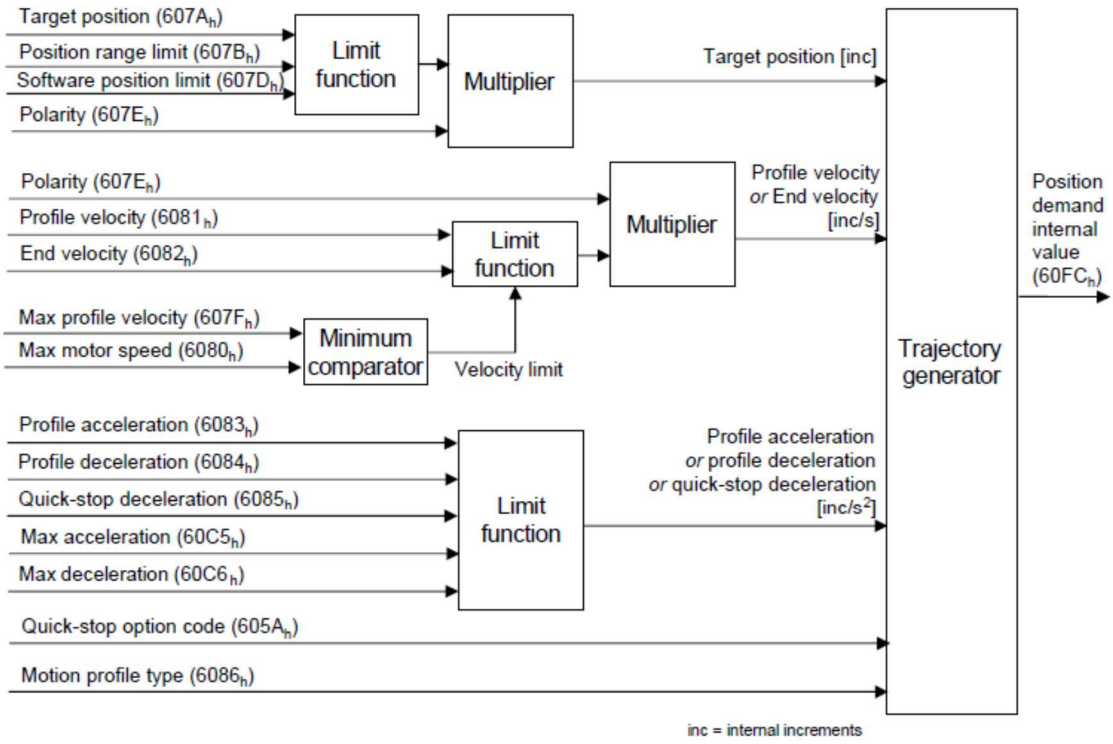


Figure 2: Profile Position Mode

Implemented Objects:

- Target position
- Software position limit
- polarity
- Profiles velocity
- Max profile velocity
- Max motor speed
- acceleration profiles
- Profile deceleration
- Quick stop deceleration
- max acceleration
- Max deceleration
- Quick stop option code

Implemented function blocks:

- Limit function
- Minimum comperator
- Trajectory Generator
- velocity controller

2.2 ControlWord/StatusWord

2.2.1 ControlWord

control	Bit 9	Bit 5	Bit 4	Description of the
1Fh	0	0	1	Single setpoint, each positioning is completed before the next setpoint becomes active. Transmitted setpoints are cached (Set of Setpoints)
3Fh	0	1	1	Transmitted setpoint is activated immediately, Set of Setpoint list is deleted
21Fh	1	0	1	Change on Setpoint

Table 1: PPmode ControlWord

Bit 6 (abs/rel)

0: absolute positioning

1: relative positioning

Bit 8 (Halt)

1: Stop positioning (Halt)

0: Continue positioning (Release from Halt)

2.2.2 StatusWord

Bit 10 (Target reached)

0: if Halt = 0, then target position not reached
if Halt = 1, drive brakes

1: if Halt = 0, then target position reached
if Halt = 1, drive speed is 0

Bit 12 (SetPoint acknowledge)

0→1: SetPoint accepted or buffered

1→:0 Next SetPoint can be accepted

2.2.3 Single Setpoint

Each setpoint consists of target position and profile speed. For each setpoint you can choose between relative and absolute positioning. The type of relative positioning can be defined via object 60F2h.

2.2.4 Set of Setpoints

The Set of Setpoints function supports the Change on Setpoint option. Currently up to 20 setpoints can be stored. Each setpoint consists of target position and profile speed. For each setpoint you can choose between relative and absolute positioning. The type of relative positioning can be defined via object 60F2h.

NOTE: The maximum number of setpoints could be increased with a minimal firmware change, if required.

2.3 Implemented Objects

2.3.1 Object 607Ah: Target position

The target position is entered in user defined Position Units.

The following applies

$$position\ internal\ value = \frac{target\ position * position\ encoder\ resolution * gear\ ration}{feed\ constant}$$

$$feed\ constant = \frac{6092h{:}\,1}{6092h{:}\,2}$$

$$encoder\ resolution = \frac{608Fh{:}\,1}{608Fh{:}\,2}$$

$$gear\ ratio = \frac{6091h{:}\,1}{6091h{:}\,2}$$

2.3.2 Object 607Dh: Software position limit

The software positions limits are entered in user defined Position Units. Target positions behind the software end positions are limited to the software end positions.

The following applies

$$position\ internal\ value = \frac{software\ position\ limit * position\ encoder\ resolution * gear\ ration}{feed\ constant}$$

$$feed\ constant = \frac{6092h{:}\,1}{6092h{:}\,2}$$

$$encoder\ resolution = \frac{608Fh{:}\,1}{608Fh{:}\,2}$$

$$gear\ ratio = \frac{6091h{:}\,1}{6091h{:}\,2}$$

If the software position limits are changed during active positioning, positioning to the end positions is carried out with the set deceleration, if these are to be overrun. For this, 3 braking points are calculated.

- Braking point right limit position
- Braking point left limit position
- Braking point Target position

2.3.3 Object 607Fh: Max profile velocity

The maximum profile speed is entered in rpm.

2.3.4 Object 6080h: Max motor speed

The maximum motor speed is entered in rpm.

2.3.5 Object 6081h: Profiles velocity

The profile speed is entered in user defined units.

The conversion of the entered unit into the internal units is carried out using objects 6092h, 608Fh and 6091h.

The following applies

$$velocity\ internal\ value = \frac{profile\ velocity * position\ encoder\ resolution * gear\ ration}{feed\ constant}$$

$$feed\ constant = \frac{6092h{:}\,1}{6092h{:}\,2}$$

$$encoder\ resolution = \frac{608Fh{:}\,1}{608Fh{:}\,2}$$

$$gear\ ratio = \frac{6091h{:}\,1}{6091h{:}\,2}$$

2.3.6 Object 6083h: Profile acceleration

The acceleration is entered in 0.01525 U/s^2.

2.3.7 Object 6084h: Profile deceleration

The deceleration is entered in 0.01525 U/s^2.

2.3.8 Object 6085h: Quick stop deceleration

The deceleration is entered in 0.01525 U/s^2.

2.3.9 Object 60C5h: Max acceleration

The maximum acceleration is entered in 0.01525 U/s^2.

2.3.10 Object 60C6h: Max deceleration

The maximum deceleration is entered in 0.01525 U/s^2.

3 Profiles Velocity Mode

3.1 Overview

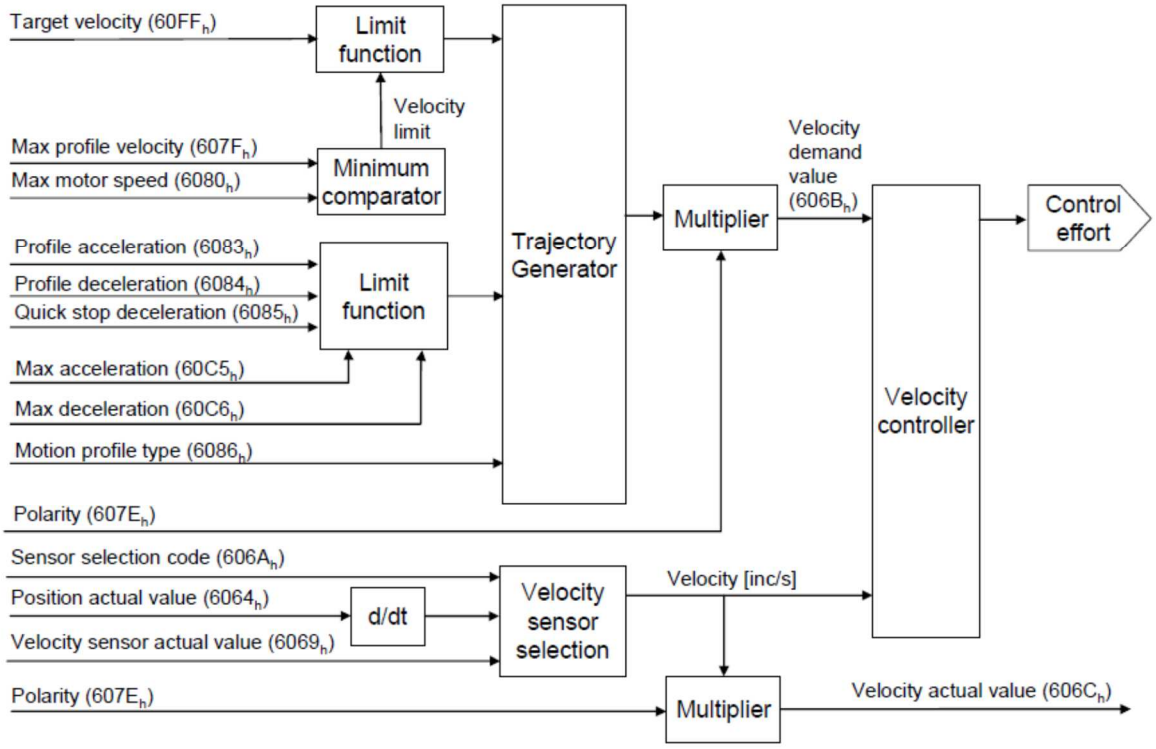


Figure 3: Profile Velocity Mode

Implemented Objects:

- target velocity
- Max profile velocity
- Max motor speed
- acceleration profiles
- Profile deceleration
- Quick stop deceleration
- max acceleration
- Max deceleration
- polarity
- Position actual value

Implemented function blocks:

- Limit function
- Minimum comperator
- Trajectory Generator
- velocity controller

3.2 ControlWord/StatusWord

3.2.1 ControlWord

Bit 8 (Halt)

1: Stop positioning (Halt)

0: Continue positioning (Release from Halt)

3.2.2 StatusWord

Bit 10 (Target reached)

0: if Halt = 0, nominal speed not reached
if Halt = 1, drive brakes

1: if Halt = 0, target speed reached
if Halt = 1, drive speed is 0

Bit 12 (SetPoint acknowledge)

0: actual speed is not equal to 0

1: actual speed equal to 0

3.3 Implemented Objects

3.3.1 Object 606Bh: Velocity demand value

Output of the trajectory generator. The value is expressed in internal speed units is specified.

3.3.2 Object 606Ch: Velocity actual value

Current speed in user defined units. The following applies

$$velocity\ actual\ value = \frac{velocity\ internal\ value * feed\ constant}{encoder\ resolution * gear\ ratio}$$

$$feed\ constant = \frac{6092h{:}\,1}{6092h{:}\,2}$$

$$encoder\ resolution = \frac{608Fh{:}\,1}{608Fh{:}\,2}$$

$$gear\ ratio = \frac{6091h{:}\,1}{6091h{:}\,2}$$

3.3.3 Object 606Dh: Velocity window

The speed window specifies a range around the target speed.

If the current actual speed after the run command - over a period as defined in 606Eh - is within the speed window, bit 10 is set in the status word. Bit10 is reset with every new run command.

3.3.4 Object 606Eh: Velocity window time

Time window in ms.

3.3.5 Object 606Fh: Velocity threshold

If the current speed exceeds the value of the object - over a period of time as defined in 6070h, bit12 in the status word is set to 0.

If the current speed falls below the value of the object - over a period of time as defined in 6070h, bit12 in the status word is set to 1.

3.3.6 Object 6070h: Velocity threshold time

Time window in ms.

3.3.7 Object 60FFh: Target velocity

The target speed is entered in user defined units.

The following applies

$$velocity\ internal\ value = \frac{target\ velocity * position\ encoder\ resolution * gear\ ration}{feed\ constant}$$

$$feed\ constant = \frac{6092h{:}\,1}{6092h{:}\,2}$$

$$encoder\ resolution = \frac{608Fh{:}\,1}{608Fh{:}\,2}$$

$$gear\ ratio = \frac{6091h{:}\,1}{6091h{:}\,2}$$

A run command is generated when the setpoint speed is changed in the OperationEnabled state.

4 Homing Mode

4.1 Overview

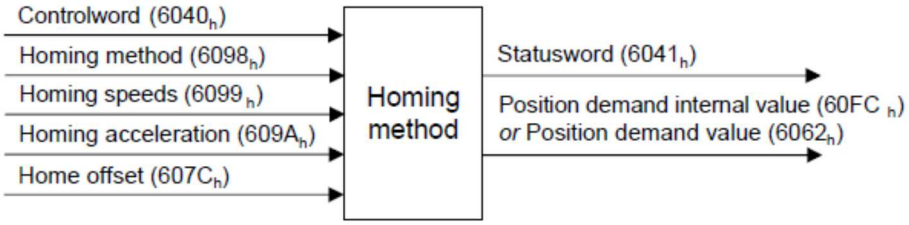


Figure 3: Homing Mode

Implemented Objects:

- controlword
- Homing method
- homing speeds
- homing acceleration
- Home offset
- status word
- Position demand value

Implemented homing methods:

- Method 17
- Method 18
- Method 19
- Method 20
- Method 21
- Method 22
- Method 23
- Method 24
- Method 25
- Method 26
- Method 27
- Method 28
- Method 29
- Method 30
- Method 35
- Methode -18 (Homing on right block)
- Method -19 (Homing on left block)

4.2 ControlWord/StatusWord

4.2.1 ControlWord

Bit 4 (Homing operation start)

0: Do not start reference run

1: Start reference run

Bit 8 (Halt)

1: Stop reference run (Halt)

0: Enable reference run (Release from Halt)

4.2.2 StatusWord

Bit 13	Bit 12	Bit 10	Description
0	0	0	Reference run active
0	0	1	Reference run was interrupted
0	1	0	Not available
0	1	1	Reference run successfully completed
1	0	0	Error during homing
1	0	1	Not available

4.3 Implemented Objects

4.3.1 Object 607Ch: Home offset

If the homing is successful, the negative home offset is assigned to the home position found.

4.3.2 Object 6098h: Homing method

Selection of homing methods.

4.3.3 Object 6099h: Homing speeds

Homing speed. The drive does not automatically return to position 0.

4.3.4 Object 609Ah: homing acceleration

Accelerating.

4.3.5 Object 60B8h: Touch probe function

Configuration of the sensors:

Bits 0-3: home switch, possible inputs 1-9

Bits 4-7: left limit switch, possible inputs 1-9

Bits 8-11: right limit switch, possible inputs1-9

Bits 12-15: reserved

Value 1 activates input 1 as corresponding sensor.

Value 2 activates input 2 as corresponding sensor.

...

Value 9 activates input 9 as corresponding sensor.

5 I/O functions

5.1.1 Object 0x3611 Output function 1

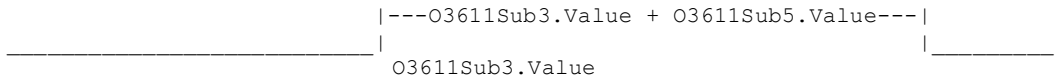
Activation and deactivation of output 1 via position condition.

The pulse duration is given in increments.

index	subindex	Description	default
0x3611	0x01	enable/disable right signal	
	0x02	enable/disable left signal	
	0x03	right position	
	0x04	left position	
	0x05	right position distance	
	0x06	left position distance	

Table 2: Object 0x3611

Example (enable/disable right signal = 1):



5.1.2 Object 0x3612 Output function 2

Output 3 can be assigned the bits of object 0x1002 via object 0x3612 subindex 1.

index	subindex	designation	default
0x3612	0x01	enable/disable output signal	

Table 3: Object 0x3612

Bit Code	Description
0x0000	Disable out
0x0001	Error over volt (Uzk)
0x0002	Error under volt (Uzk)
0x0004	Error Ack
0x0008	Error over temp
0x0010	Error derating
0x0020	Abort connection
0x0040	Error stroke

0x0080	Error communication
0x0100	error overcurrent

Table 4: Error coding

6 Position control function

6.1 Overview

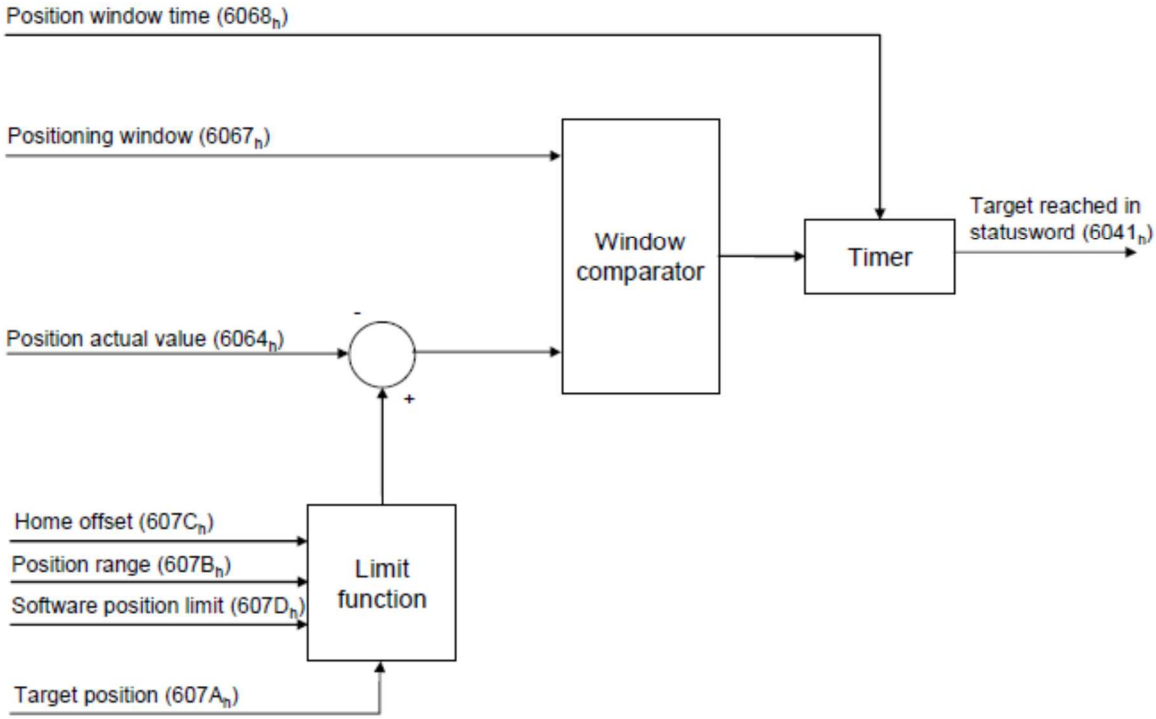


Figure 4: Position reached

Implemented Objects:

- Position window time
- Positioning window
- Position actual value
- Home offset
- Software position limit
- Target position

Implemented function blocks:

- Limit function
- Window comperator
- timer

6.2 Implemented Objects

6.2.1 Object 6063h: Position actual internal value

Current position in internal units.

6.2.2 Object 6064h: Position actual value

Current position in user defined units. The following applies

$$position\ actual\ value = \frac{position\ internal\ value * feed\ constant}{encoder\ resolution * gear\ ratio}$$

$$feed\ constant = \frac{6092h{:}\,1}{6092h{:}\,2}$$

$$encoder\ resolution = \frac{608Fh{:}\,1}{608Fh{:}\,2}$$

$$gear\ ratio = \frac{6091h{:}\,1}{6091h{:}\,2}$$

6.2.3 Object 6067h: Position window

Window around the target position. If the drive is within the target window during time 6068h, the Target Reached bit is set in the status word.

6.2.4 Object 6068h: Position window time

Time window in ms.

6.2.5 Object 60FCh: Position demand internal value

Current target position of the trajectory generator.

6.2.6 Object 60F2h: Positioning option code

Supported options:

Bit 1	Bit 0	Definition
0	0	Positioning moves shall be performed relative to the preceding (internal absolute) target position (rsp. relative to 0 if there is no preceding target position) as described in 10.2
0	1	Positioning moves shall be performed relative to the actual position demand value (object $60FC_h$) – output of the trajectory generator
1	0	Positioning moves shall be performed relative to the position actual value (object 6064_h)
1	1	Reserved

7 Manufacturer specific objects

7.1 Python Interpreter

The interface between the Python interpreter and the motor controller is the object directory. Functions from Python targetlib allow access to individual objects.

The following functions are available for this purpose:

7.1.1 Functions

7.1.1.1 ReadObjectByte(s, m)

```
/*****************************************************************************
**
**    Function      : ReadObjectByte
**
**    Description   : Reads an 8 bit object entry.
**
**    Parameter     : wIndex     - Index
**                    bySubindex - Subindex
**                    byValue    - Value
**
*    * Returnvalues : tCopKernel
**
*****************************************************************************/
```

7.1.1.2 ReadObjectDword(s, m)

```
/*****************************************************************************
**
**    Function      : ReadObjectWord
**
**    Description   : Reads a 16 bit object confirmation entry.
**
**    Parameter     : wIndex     - Index
**                    bySubindex - Subindex
**                    wValue     - value
**
*    * Returnvalues : tCopKernel
```

```
**
*************************************************************************/
```

7.1.1.3 ReadObjectWord(n, m)

```
/*************************************************************************
**
**    Function      : WriteObjectWord
**
**    Description   : Writes a 16 bit object verification entry.
**
**    Parameter     : wIndex     - Index
**                    bySubindex - Subindex
**                    byValue    - Value
**
*   * Returnvalues : tCopKernel
**
*************************************************************************/
```

7.1.1.4 SendEmcyMsg()

```
/*************************************************************************
**
**    Function      : SendEmcyMessage
**
**    Description   : Sends the transferred error number via
**    the emergency channel.
**
**    Parameter     : none
**
*   * Returnvalues : none
**
*************************************************************************/
```

7.1.1.5 WriteObjectByte(n, m, o)

```
/*************************************************************************
**
**    Function      : WriteObjectByte
**
```

```
**    Description    : Writes an 8 bit object confirmation entry.
**
**    Parameter      : wIndex      - Index
**                     bySubindex - Subindex
**                     byValue     - Value
**
*    * Returnvalues : tCopKernel
**
*****************************************************************************/
```

7.1.1.6 WriteObjectDword(n, m, o)

```
/*****************************************************************************
**
**    Function       : WriteObjectDWord
**
**    Description    : Writes a 32 bit object verification entry.
**
**    Parameter      : wIndex      - Index
**                     bySubindex - Subindex
**                     byValue     - Value
**
*    * Returnvalues : tCopKernel
**
*****************************************************************************/
```

7.1.1.7 WriteObjectWord(n, m, o)

```
/*****************************************************************************
**
**    Function       : WriteObjectWord
**
**    Description    : Writes a 16 bit object verification entry.
**
**    Parameter      : wIndex      - Index
**                     bySubindex - Subindex
**                     byValue     - Value
```

```
**
*    * Returnvalues : tCopKernel
**
*************************************************************************/
```

7.1.2 python variables

MovingCap has a set of freely usable objects that can be stored in the permanent memory (store all parameters).

7.1.3 8Bit Variables

Object 0x340B Subindex 1 - 0x0A

Object 0x340C Subindex 1 - 0x0A

7.1.4 16Bit Variables

Object 0x340D Subindex 1 - 0x0A

Object 0x340E Subindex 1 - 0x0A

7.1.5 16Bit Variables

Object 0x340F Subindex 1 - 0x0A

Object 0x3410 Subindex 1 - 0x0A

7.2 Object 0x3401 Control parameter

7.2.1 Block Detection

The number of block events is evaluated for block detection.

A Blocking event occurs when the actual motor phase current is greater than the specified maximum current (object 0x6073) and the actual speed is less than the minimum speed for block speed detection.

One event can be counted per ms. If the number of events is greater than the maximum value of the counter for block detection, Block Detection is reported.

7.2.1.1 Object 0x3401 Subindex 1

Maximum value Counter Block speed detection

7.2.1.2 Object 0x3401 Subindex 2

Minimum speed Block speed detection

7.2.2 Object 0x3401 Subindex 3

positioning window

7.2.3 RS485

MovingCap349 communicates with the motor via an internal RS485 interface.

7.2.3.1 Object 0x3401 Subindex 4

Uart Timeout [ms]

If no message is received from the motor within the specified time, a communication error is reported.

7.2.3.2 Object 0x3401 Subindex 5

Uart Errors

Number of Uart errors since Power Up.

7.2.4 Object 0x3401 Subindex 0x15

acceleration torque

Maximum current in the acceleration ramp in 1/1000 of the rated current.

7.2.5 Object 0x3401 Subindex 0x16

deceleration torque

Maximum current in the deceleration ramp in 1/1000 of the rated current.

7.3 Option Codes

7.3.1 Object 0x6059 Subindex 0x01 Stroke option code

Block Detection Option Code

0: Drive changes to FAULT state

1: Drive stops, sends emergency message, does not change to FAULT state

2: Drive performs **absolute** positioning and sends the corresponding emergency message.

Object **0x6059:0x02**: Target Position [Increments]

Object **0x6059:0x03**: Speed [rpm]

3: Drive performs **relative** positioning and sends the corresponding emergency message.

Object **0x6059:0x02**: Target Position [Increments]

Object **0x6059:0x03**: Speed [rpm]

8 Emergency Messages

error code	Description
3211h	Overvoltage DC link
3221h	Undervoltage DC link
4220h	I2t
4210h	Overtemperature
8160h	CAN
7121h	block detection
7510h	motor communication error
5530h	EEprom

Table 5: Emergency Messages