Osterholz Antriebs Technik GmbH

Pitch Control System

Programming and parameters

Index of issues:

Issue.	Date	Remarks
1.00	15.09.06	Created
1.01	27.09.06	Added some objects, corrected mistakes
1.02	29.09.06	Some additions, corrected max motor speed
1.03	11.10.06	Added encoder code objects, error mask/flags in object 6510h, moved object numbers of the EFC-objects in area 56xx.
1.04	17.10.06	PDO-Mapping complemented, objects with 16bit-access (5xxxh) added
1.05	10.07.07	PMM complemented, PMC reworked, title changed into “programming“
1.06	19.12.07	Newly arranged chapter PMC, texts complemented
1.07	25.03.08	Translated,
1.08	20.01.09	Add ons in control flags, PMM controlword, new test status (3005h), Objects 68F4h, 6865h, 6866h, 3068h, 60BAh, 60BBh
1.09	27.05.09	bit 5 in test status and bit 2 in PMM_controlword new, object 30E3 new.

2st edition: January 2009 / English
This documentation applies to devices of series PMM, PMC, PBS.

Target group:
Technicians, control systems planning engineers and all specialists familiar with the fundamental principle of operation of automation systems in an industrial environment.

Suggestions and reference to errors are always welcome.

This manual has been compiled with utmost care and exhaustively checked.
Nevertheless, the possibility of errors remaining cannot be excluded entirely. OAT cannot accept any legal responsibility or other liability for any remaining errors and their consequences.

Further details are available from OAT on request if the information provided proves insufficient or if you encounter special problems which are not covered in sufficient depth in this documentation.

OAT Osterholz AntriebsTechnik GmbH

Stiftstrasse 38 / Office: Stohlmanns Heide 15
D-32278 Kirchlengern
Phone: +49 0 5223 / 18309-0
Fax: +49 0 5223 / 18309-15
E-mail: info@oat-gmbh.com
Internet: www.oat-gmbh.com

Table of Contents

1 Introduction

The Pitch-Control-System is composed of two units, Pitch Management Module (PMM) and Pitch Motion Controller (PMC).

The PMM is responsible for battery management and provides the link as well as some special functions.

The PMC is an axial controller that can access DC-motors as well as AC-Synchronic and AC-Asynchronic motors.

This documentation at hand will explain the interface of these units in respect to the field bus activation. This interface consists of a so called object dictionary, which defines all parameters accessible by the user.

Independent from the used field bus activation, the object dictionary builds upon the following specifications of CAN in automation (CiA):

CANopen Application Layer and Communication Profile, CiA DS 301.
CANopen Device Profile Drives and Motion Control, CiA DS402.

This provides a uniform interface for every field bus interface.

The CANopen interface and those system functions both units have in common can be found in the following documentations:

Pitch-Control-System, CANopen-Interface.

Pitch-Control-System, System functions.

2 The Object Dictionary

2.1 Basic Structure

The object dictionary controls all functions of the Pitch Control System. The object dictionary is a grouping of objects that can be accessed via the field bus network in a defined process. These objects represent the parameters accessible by the user and contain information on access rights, file types, entities and default values. Each object is addressed through a 16 bit index.

The following table shows the underlying layout of the standard object dictionary:

Index(hex)	**Object**
0000	Not utilized
0001-001F	Static file types
0020-003F	Complex file types
0040-005F	Manufacturer specific file types
0060-007F	Device profile specific static file types
0080-009F	Device profile specific complex file types
00A0-0FFF	Reserved for future use
1000-1FFF	Communication profile area
2000-5FFF	Manufacturer specific area
6000-9FFF	Standardized device profile area
A000-FFFF	Reserved for future use

Table 1: Object dictionary structure

The standard object dictionary can contain a maximum of 65535 entries which are addressed through the 16 bit index.

The file types (Indexes 0001_h to $009F_h$) contain the type definitions. These entries are only listed for reference purposes and can be neither read nor written. (for more information, please refer to CiA DS301, chapter “Dictionary Structure and Entries.)

The static file types (01_h to $1F_h$, 60_h to $7F_h$) are simple types (e.g. integer, floating point, string, boo-lean). These can be saved in a simple variable.

The complex file types (20_h to $5F_h$, 80_h to $9F_h$) are structures that are composed of simple file types (records or arrays).

The communications profile area (1000_h to $1FFF_h$) contains communications-specific objects according to DS301, like e.g. the definition of cyclical transferred objects, error memory, node address and so on.

The manufacturer specific area (2000_h to $5FFF_h$) is not standardized and can be optionally defined by each manufacturer.

The standardized device profile area (6000_h to $9FFF_h$) is defined in the respective device profile. The profile “Drives and Motion Control”, DS402, is used for the PMC.

2.1.1 Use of Index and Subindex

In order to address an entry in the object dictionary, a 16 bit wide index is utilized. In case of a simple variable, a value can be accessed directly. In case of a complex object dictionary entry like record or array, the index addresses the whole file structure. In order to permit access to single elements of the structure, a subindex was defined. In case of simple file types, this subindex is always Zero. In case of a structure, the subindex addresses the single variables inside this structure. The first variable always stands at subindex 1. The subindex 0 always contains the number of elements of a structure.

Example: The object $608F_h$, *position encoder resolution*, is a fraction that can be described with an array of two integer values:

$$position\_encoder\_resolution = \frac{encoder\_increments}{motor\_revolutions}$$

The corresponding object can be accessed thus:

Index	Subindex	Variable	File type
$648F_h$	0	Number of elements	unsigned8
	1	*encoder increments*	unsigned32
	2	*motor revolutions*	unsigned32

Table 2: Use of Index and Subindex

2.2 Object Dictionary Structure

The object dictionary of PMC and PMM is structured according to the following system:

Index area	Utilization
1000_h - $1FFF_h$	System area, according to DS301 (PMM and PMC)
2000_h - $2FFF_h$	System area, manufacturer specific, file transfer objects (PMM and PMC)
3000_h - $3FFF_h$	Parameters, status message and actual value PMM
5000_h - $57FF_h$	Axis, 16 bit access to some 32 bit objects, manufacturer specific objects (PMC, with PMM manual function only)
5800_h - $5FFF_h$	Blade encoder, 16 bit access to some 32 bit objects (PMC)
6000_h - $67FF_h$	Axis, according to DS402 (PMC, with PMM *error code* only)
6800_h - $6FFF_h$	Blade encoder, as 2nd axis according to DS402 (PMC)

Table 3: Object dictionary Structure of PMC and PMM

In the area of 1000_h to $1FFF_h$ objects from the communication profile DS301 can be found, if they are utilized by the PMC.

The area 3000_h to $3FFF_h$ contains objects for the alignment and control of the PMM's functions.

In the area 5000_h to $5FFF_h$ the 16 bit access to some 32 bit objects in the area 6000_h to $6FFF_h$ is rendered possible. With this, the control unit can link space saving objects to the IO area, if the value area can be displayed with 16 bit. Further manufacturer specific objects are stored here, as well.

The area 6000_h to $67FF_h$ contains objects for alignment and control of the axis according to DS402.

Ex index 6800_h objects needed for the blade encoder are addressed. The blade encoder is considered 2nd axis pursuant to DS402.

2.3 Objects Description

Meaning of table columns:	
Index	The 16 bit index in the object dictionary that represents a variable or structure respectively
Subindex	The 8 bit subindex, that represents a single variable within a complex object
Name	Name of the object
Object	Object type that represents the data, e.g. VAR, ARRAY, RECORD, etc.
Data Type	Data type that represents the information, e.g. unsigned32, unsigned8, String, etc. unsigned: no algebraic sign, positive values only integer: with algebraic sign, negative and positive values The number behind indicates the number of bits.
Attr	Access authorization ro read only wo write only rw read and write
Map	Identifies the nature of the PDO mapping for the object N mapping not allowed Y mapping allowed
Default Value	Preset object value after initializations.
Mem.	N Object will not be saved (set points and actual values, status messages) Y Object can be saved (parameters or remanent data)

Table 4: Meaning of the table columns in the object descriptions

Objects of the object type VAR will always be addressed with subindex 0, therefore, object type and subindex are not specifically mentioned in the object tables for those objects.

The objects which are being saved as parameter in the machine are highlighted in grey. These objects should not be modified by the control unit under normal circumstances.

3 System Area and Shared Objects

3.1 Communication Objects

The objects for the communication are described in the documentation of the respective field bus interface.

3.2 Error Messages

3.2.1 Objects

Index	Object	Name	Type	Access	Mem.
1001_h	VAR	*error register*	unsigned8	ro	N
1003_h	ARRAY	*pre defined error field*	unsigned32	ro	N
2030_h	ARRAY	*error field time stamp*	unsigned32	ro	N
$603F_h$	VAR	*error code*	unsigned16	ro	N

Table 5: Error message objects

The objects 1001h, *error register*, 1003h, *predefined error field* and 2030h, *error field time stamp* are described in the documentation Pitch Control System, CANopen interface.

3.2.1.1 Object $603F_h$: *error code*

In *error code,* error and warning codes are registered. Those entries can also be found in the lower word of object 1003_h, *pre defined error field*, in case of an error code.

At the PCM, the Bit 3 (*fault*) or Bit 7 (*warning*) is set in *Statusword* (6041_h).

At the PMM, the Bit 2 (*fault*) or Bit 1 (*warning*) is set in *PMM_Status* (3000_h).

In case of the PMM, reset of an error occurs by the setting of Bit 7 in the *PMM controlword* (Object 2040_h) or by the NMT command *Reset* on the CAN bus.

At the PMC an error will be reset by setting of Bit 7 in the *controlword* (Object 6040_h).

Index	Name	Type	Attr	Map	Default-Value	Description
$603F_h$	*error code*	unsigned16	ro	Y	0	Error code, 0..FFF_h

3.2.1.2 Meaning of the Error Codes

error code (hex)	Meaning	PMM	PMC
0000_h	No error	X	X
2210_h	Over current 24V feed-in	X	
2250_h	Short circuit charger (fault output of the power stage)	X	
2260_h	Short circuit booster (fault output of the power stage)	X	

***error code* (hex)**	**Meaning**	**PMM**	**PMC**
2310_h	Over current at controller output		X
2320_h	Short circuit, or ground fault, motor side		X
3110_h	Grid over voltage	X	
3120_h	Grid under voltage	X	
3130_h	Phase error	X	
3210_h	Over voltage in the DC link, PMC		X
3211_h	Over voltage in the DC link, PMM, booster	X	
3220_h	Under voltage in the DC link		X
3221_h	DC link PMM-PMC interrupted	X	
3280_h	Battery voltage too low	X	
$328n_h$	Voltage at battery bloc n too low	X	
4110_h	Temperature Sensor 1 too high	X	
4111_h	Temperature Sensor 2 too high	X	
4120_h	Temperature Sensor 1 too low	X	
4121_h	Temperature Sensor 2 too low	X	
4210_h	Unit temperature (heat sink) too high	X	X
4220_h	Unit temperature (heat sink) too low	X	
4310_h	Motor temperature too high		X
4410_h	Battery temperature too high	X	
4420_h	Battery temperature too low	X	
5100_h	Over voltage 24V feed-in	X	
5112_h	Under voltage 24V feed-in	X	
5120_h	Error in power stage of supply controller	X	
5420_h	Chopper error (fault output power stage)	X	
5510_h	Hardware error RAM	X	X
5530_h	Error programming the EEPROM	X	X
6010_h	Watchdog-Reset	X	X
6080_h	Software trap, Firmware error		X
6181_h	Analogue-Digital-Converter error	X	
6320_h	Wrong parameter		X
6380_h	No power stage identifier		X
7300_h	Encoder error (general)		X
7305_h	Encoder error (motor), sine/cosine		X
7320_h	Encoder error (motor), data channel		X
7321_h	2nd Encoder error (blade encoder)		X
7510_h	Communication error RS485 interface	X	
7520_h	Communication error battery interface	X	
8100_h	CAN communication, general	X	X
8110_h	CAN overflow, telegrams lost	X	X
8120_h	CAN in „error passive“	X	X
8130_h	Heartbeat or Node guarding failed	X	X
8140_h	CAN was „bus off“	X	X

error code (hex)	Meaning	PMM	PMC
8230_h	MPDO target object not found	X	X
8400_h	Velocity controller, maximum speed frequency exceeded		X
8611_h	following error too large		X
8620_h	Difference motor encoder – blade encoder too large		X
$FFFn_h$	Fault input n activated (e.g. $FFF1_h$ lightning protection)	X	

Table 6: Error Codes of the Pitch Control System

3.3 Manual Functions

The PMM can be operated by a selector switch and key button in order to run service functions. Those functions can also be controlled through objects, e.g. in order to plug in an external operation terminal.

3.3.1 Objects

Index	Object	Name	Type	Ac-cess	Mem.
5040_h	VAR	*manual control*	unsigned8	rw	N
5041_h	VAR	*manual status*	unsigned8	ro	N
5042_h	VAR	*manual control out*	unsigned8	ro	N

Table 7: Objects for External Manual Functions

3.3.1.1 Object 5040_h: *manual control*

Index	Name	Type	Attr	Map	Default-Value	Description
5040_h	*manual control*	unsigned8	rw	Y	0	Command for manual control, takes over operation of mode switch and key buttons.

Bit	Meaning
0..3	Select manual function like mode switch
4	Manual function key button, 1 = button pressed
5	1 = manual function is controlled by this object
6	reserved
7	0->1 fault reset

Table 8: Bits in the *manual control* and *manual control out,* respectively.

3.3.1.2 Object 5041$_h$: *manual status*

Index	Name	Type	Attr	Map	Default-Value	Description
5041$_h$	*manual status*	unsigned8	rw	Y	0	Manual operation feedback

Bit	Meaning
0..3	Feedback mode (=selector switch)
4	1 = function active (Busy)
5..7	Reserved (0)

Table 9: Bits in *manual status*.

3.3.1.3 Object 5042$_h$: *manual control out*

This object only exists in the PMM.

Index	Name	Type	Attr	Map	Default-Value	Description
5042$_h$	*manual control out*	unsigned8	ro	Y	0	Here, the operating value that the PMM sends to the related PMC can be monitored.

Configuration like *manual control*. The content comes from either the mode switch or the *manual control*.

4 The Pitch Management Module (PMM)

This unit supplies the PCM with the direct current link voltage and provides the brake chopper. Furthermore, it contains the battery management system and a boost converter that raises the battery voltage to intermediate circuit level in case of power failure. The unit operates mostly self sufficient, the behavior is adjusted through parameters in the object dictionary.

Furthermore, programmable temperature controller and other special functions are available.

The unit is composed of the following components:

- Battery management
- Boost converter (Booster)
- Chopper
- Special functions, e.g. temperature controller

4.1 Status Messages and Control Word

Each functional group has its own status object that gives information on the status. A control word monitors the unit in order to e.g. start the loading cycle manually or reset an error message.

4.1.1 Objects

Index	Object	Name	Type	Access	Mem.
3000_h	VAR	*PMM status*	unsigned8	ro	N
3001_h	VAR	*charger status*	unsigned8	ro	N
3002_h	VAR	*booster status*	unsigned8	ro	N
3003_h	VAR	*chopper status*	unsigned8	ro	N
3004_h	VAR	*24V supply status*	unsigned8	ro	N
3040_h	VAR	*PMM controlword*	unsigned16	rw	N

Table 10: PMM Status Messages

PMM status serves as overriding status message that displays errors and warning. The respective error code is displayed in the object *error code* ($603F_h$). Furthermore, if an error occurs, an alarm message is sent to the CAN bus (Emergency, identifier 80_h + node address).

The other status messages display the status of the respective function and can be displayed for control purposes. They are not essential for error response.

With these status objects, bit 0 always displays whether the specific function is ok.

4.1.1.1 Object 3000_h: *PMM status*

The object *PMM status* provides information on the status of the PMM; it is here that errors and warnings are reported. These objects should be queried cyclically.

The PMM fault reset is done by setting bit 7 in the *PMM controlword* (Object 2040_h) or by the NMT command *Reset* on the CAN bus.

Index	Name	Type	Attr	Map	Default-Value	Description
3000_h	*PMM status*	unsigned8	ro	Y		Collective status message of the PMM

Bit	Meaning
0	1 = OK (no fault, no warning, functions normally)
1	1 = Warning (→ error code in object $603F_h$)
2	1 = Fault (→ error code in object $603F_h$)
3	1 = Battery fault
4	1 = Switch off counter runs down
5	1 = Mains under voltage, runs on battery
6	1 = Mains over voltage
7	1 = Mains – phase failure

Table 11: Bits in the *PMM status*.

4.1.1.2 Object 3001_h: *charger status*

Here, it can be inquired which loading phase is active and if a charger error occurred. Bit 1 (charger active) becomes 1 in cases of mains-, intermediate- and battery voltage. In cases of power failure and switch over to booster operation, the charger is turned off, bit 1 is set to 0.

Index	Name	Type	Attr	Map	Default-Value	Description
3001_h	*charger status*	unsigned8	ro	Y		Condition of the charge controller

Bit	Meaning	Error code
0	1 = OK	
1	1 = charger active	
2	1 = Battery charge with current limiting (1^{st} stage)	
3	1 = Battery charge at threshold voltage (2^{nd} stage)	
4	1 = Short circuit charger, fault output at the power stage	2250_h
5	free	
6	free	
7	1 = Battery communication error	7520_h

Table 12: Bits in the *charger status*.

4.1.1.3 Object 3002_h: *booster status*

This is the status of the boost converter. It becomes active (bit 1 = 1) in case of power failure and dropping of the direct current link voltage below initiation voltage (see Chapter 4.4, Page 32).

Index	Name	Type	Attr	Map	Default-Value	Description
3002_h	*booster status*	unsigned8	ro	Y		Boost converter status

Bit	Meaning	Error code
0	1 = OK	
1	1 = Booster active	
2	1 = Booster at current limiting	
3	1 = Booster at threshold voltage	
4	1 = Short circuit booster, fault output at power stage	2260_h
5	1 = Over voltage at booster	3211_h
6	free	
7	free	

Table 13: Bits in the *booster status*.

4.1.1.4 Object 3003_h: *chopper status*

In case of an increase in direct current link voltage, the chopper switches the braking resistance to the link via an adjustable level, in order to use up the feed-back brake energy until the voltage is below the shut down threshold (see Chapter 4.5, Page 33).

Index	Name	Type	Attr	Map	Default-Value	Description
3003_h	*chopper status*	unsigned8	ro	Y		Brake chopper status

Bit	Meaning	Error code
0	1 = OK	
1	1 = Chopper active	
2	free	
3	free	
4	1 = Short circuit chopper, fault output at power stage active	5420_h
5	free	
6	free	
7	free	

Table 14: Bits in the *chopper status*.

4.1.1.5 Object 3004_h: *24V supply status*

Here, the status of the 24V feed-in can be inquired after. It will be displayed whether an external 24V feed-in is connected and if a programmable voltage level was exceeded. The latter can monitor e.g. The function of the motor stop brake. The voltage threshold is displayed in the object *supervisor current level* ($30C0_h$).

Index	Name	Type	Attr	Map	Default-Value	Description
3003_h	*24V supply status*	unsigned8	ro	Y		24V feed-in status

Bit	Meaning	Error code
0	1 = OK	
1	1 = External 24V feed-in fitted	
2	1 = Voltage level of object $30C0_h$ exceeded	
3	free	
4	1 = Overload (I > 8A)	2210_h
5	1 = Under voltage (U < 18V)	5112_h
6	1 = Over voltage (U > 28V)	5100_h
7	free	

Table 15: Bits in the *24V supply status*.

4.1.1.6 Object 3005_h: *test status*

This object shows the state and the results of the automatic battery test. The test is started automatically after switch on and then once every week. It can also be started with bit 1 in the *PMM controlword*. The test will not be carried out until the batteries are charged and the motors don't move.

After setting bit 1 in the *PMM controlword*, bit 2 in *test status* is set. Reset bit 1 in *PMM controlword* then to start the test.

Index	Name	Typ	Attr	Map	Default-Wert	Beschreibung
3005_h	*Test status*	unsigned8	ro	J		Status of the battery test

Bit	Meaning	Error code
0	1 = OK	
1	1 = Test finished	
2	1 = Test started	
3	1 = Test active	
4	1 = Error during test (Battery voltage below minimum)	
5	1 = Test not possible, because batteries not charged or axes moving	
6		
7		

Table 16: Bits in the *test status*.

4.1.1.7 Object 3040_h: *PMM controlword*

In normal operation, the PMM does not need control commands, it automatically operates after switch-on and read in of the set parameters from the EEPROM.

The PMM controlword provides various control options, e.g. fault reset or manual start-up of the battery charger. It can be expanded by additional control options in the future, therefore, not all used bits should be specified 0.

Index	Name	Type	Attr	Map	Default-Value	Description
3040_h	*PMM controlword*	unsigned16	rw	Y		PMM control word

Bit	Meaning	Command
0	0->1 = Manual start-up of the battery charger	0001_h
1	0->1 = Manual start of the battery test. Test starts after resetting to 0	0002_h
2	0->1 = Manual start of booster for 10s, 1->0 stops it early	0004_h
3..6	Not used, always write 0	
7	0->1 = fault reset	0080_h
8..15	Not used, always write 0	

Table 17: Bits in the *PMM controlword*.

In order to start a function, the bit must switch from 0 to 1. If it is already set at 1, 0 must be written previously.

4.2 Voltage Monitoring

The unit continuously monitors the current mains voltage, the direct current link voltage, the battery voltage and the 24V feed-in. The current measured values are provided in objects. Furthermore, a monitoring threshold and respective status messages exist.

The time for bridging a power failure and shut down after power failure and emergency mode can be set.

4.2.1 Mains Voltage

Index	Object	Name	Type	Access	Mem.
3062_h	VAR	*actual mains voltage*	unsigned16	ro	N
$30B2_h$	VAR	*remaining time to backup*	unsigned16	ro	N
$30B3_h$	VAR	*remaining time to power down*	unsigned16	ro	N
$30E0_h$	VAR	*max time of power failure*	unsigned16	rw	Y
$30E1_h$	VAR	*max time to power down*	unsigned16	rw	Y

Table 18: Objects to monitor mains voltage

4.2.1.1 Object 3062h: *actual mains voltage*

The mains voltage is measured directly at the mains connection clamp.

Index	Name	Type	Attr	Map	Default-Value	Description
3062_h	*actual mains voltage*	unsigned16	ro	Y		actual mains voltage in V

4.2.1.2 Object 30B2$_h$: *remaining time to backup*

Continuously displays the remaining time until emergency mode.

The dead time is set in *max time of power failure*, object 30E0$_h$.

Index	Name	Type	Attr	Map	Default-Value	Description
30B2$_h$	*remaining time to backup*	unsigned16	ro	Y		Time remaining (in ms) until emergency stop after power failure

4.2.1.3 Object 30B3$_h$: *remaining time to power down*

This object continuously monitors the time remaining until shut-down.

The dead time is set in *max time of power down*, object 30E1$_h$.

Index	Name	Type	Attr	Map	Default-Value	Description
30B3$_h$	*remaining time to power down*	unsigned16	ro	Y		Time remaining (in seconds) until shut-down

4.2.1.4 Object 30E0$_h$: *max time of power failure*

Here, the dead time until start of the emergency mode is set. It also specifies the maximum duration of a power failure. This duration should be at least 3 seconds. This covers the mains connection rule of electricity providers in case of power failure (e.g. 3 seconds at E.ON).

Index	Name	Type	Attr	Map	Default-Value	Description
30E0$_h$	*max time of power failure*	unsigned16	rw	N	3000	Dead time (in ms) until emergency stop after power failure.

4.2.1.5 Object 30E1$_h$: *max time to shutdown*

Here, the dead time until shut down after power failure can be set.

The minimum time for the power failure hold-up plus the maximum time which the blades need to drive into the off position (feather position).

Index	Name	Type	Attr	Map	Default-Value	Description
30E1$_h$	*max time of power down*	unsigned16	rw	N	30	Dead time (in seconds) until shut down after power failure.

4.2.2 Direct Current Link Voltage

The following table shows an overview of the parameters of the direct current link voltage in order of voltage values. This order of magnitude must be kept at set up.

The minimum voltage in the PMC must be below the battery voltage for the axis to run on battery voltage (manual and emergency operation).

Intermediate circuit parameter	Default Value	Name	Object	Page
Minimum voltage (PMC)	100V	*drive data.min dc-link voltage*	6510_h.5	49
Minimum voltage (PMM)	400V	*dc-link voltage.minimal value*	$30FA_h$.3	21
Initial voltage booster	420V	*booster parameter set.switch-on voltage*	$30FB_h$.1	32
Nominal voltage	560V	*dc-link voltage.nominal value*	$30FA_h$.1	21
Control voltage booster	600V	*booster parameter set.target voltage*	$30FB_h$.2	32
Shut down voltage chopper	650V	*brake chopper.switch-off voltage*	$30FC_h$.2	34
Switch-on voltage chopper	670V	*brake chopper.switch-on voltage*	$30FC_h$.1	34
Maximum voltage (PMM)	750V	*dc-link voltage.maximal value*	$30FA_h$.2	21
Maximum voltage (PMC)	750V	*drive data.max dc-link voltage*	6510_h.4	49

Table 19: Parameter overview of direct current link voltage

Index	Object	Name	Type	Access	Mem.
3063_h	VAR	*actual dc-link voltage*	unsigned16	ro	N
$30FA_h$	ARRAY	*dc-link voltage*	unsigned16	rw	Y

Table 20: Direct current link voltage objects.

4.2.2.1 Object 3063_h: *actual dc-link voltage*

Index	Name	Type	Attr	Map	Default-Value	Description
3063_h	*actual dc.link voltage*	unsigned16	ro	Y		actual direct current link voltage in Volt

4.2.2.2 Object $30FA_h$: *dc-link voltage*

Here, the thresholds for the intermediate current monitoring are set.

Index	Name	Object				
$30FA_h$	*dc-link voltage*	ARRAY	Direct current link voltage threshold			
Sub-Index	**Name**	**Type**	**Attr**	**Map**	**Default-Value**	**Description**
0	*number of elements*	unsigned8	ro	N	3	Number of elements
1	*nominal value*	unsigned16	rw	N	560	Nominal voltage in Volt
2	*maximal value*	unsigned16	rw	N	750	Maximum voltage in Volt
3	*minimal value*	unsigned16	rw	N	400	Minimum voltage in Volt

The nominal voltage for the intermediate current results from $U_{mains} * \sqrt{2}$ at tri-phase mains voltage.

Falling below minimum voltage will result in the error message 3220_h.

Exceeding maximum voltage will result in the error message 3211_h.

Normally, the booster and the brake chopper make sure that the direct current link voltage remains within the required limits.

The booster voltage is set in object $30FB_h$.7, the initial voltage in object 30FBh.8. Please refer to chapter 4.4.1.1, page 32.

The switch-on and the shut down voltage of the chopper are set in object $60FC_h$. Please refer to chapter 4.5.1.1, page 34.

4.2.3 24V-Versorgung

Index	Object	Name	Type	Access	Mem.
3065_h	VAR	*actual supply voltage*	unsigned16	ro	N
3066_h	VAR	*actual supply voltage external*	unsigned16	ro	N
3075_h	VAR	*actual supply current*	unsigned16	ro	N
$30C0_h$	VAR	*supervisor current limit*	unsigned16	rw	Y

Table 21: 24V feed-in objects

4.2.3.1 Object 3065_h: *actual supply voltage*

This object displays the 24V supply voltage as measured by the PMM. The voltage is measured at clamp X20.4. This is either the value of internally produced voltage or, in case of external supply, the value of the supplied 24V feed-in minus the flow voltage of a diode (about 0.7V).

Index	Name	Type	Attr	Map	Default-Value	Description
3065_h	*actual supply voltage*	unsigned16	ro	Y		Current 24V feed-in voltage in mV

4.2.3.2 Object 3066_h: *actual supply voltage external*

Displays the 24V supply voltage measured by the PMM at clamp X20.6. This voltage is decoupled from the internal voltage by a diode.

Index	Name	Type	Attr	Map	Default-Value	Description
3066_h	*actual supply voltage external*	unsigned16	ro	Y		Current external 24V supply voltage in mV

4.2.3.3 Object 3075_h: *actual supply current*

The electricity measured in the 24V feed-in. The total current of own consumption and output current is measured.

Index	Name	Type	Attr	Map	Default-Value	Description
3075_h	*actual supply current*	unsigned16	ro	Y		Current electricity of the 24V feed-in in mA

4.2.3.4 Object $30C0_h$: *supervisor current level*

This object can be used to set an electricity threshold from which the status bit 2 in object 3004_h will be set so that it can e.g. be monitored if the brake operated. For this purpose, the usual consumption of the system is calculated. A safety margin is added and entered here as level. In case of unfixed brake, bit 2 should be set.

Index	Name	Type	Attr	Map	Default-Value	Description
$30C0_h$	*supervisor current level*	unsigned16	rw	N	1000	Electricity level at which the bit 2 in *24V supply status* (object 3004_h) is set.

4.3 The Battery Management

The PMM's battery management monitors the status of the battery, gives information on charge state, on the respective bloc voltages and bloc temperatures, the current battery voltage and the electricity.

The batter is charged according to set battery type, including the withdrawn and supplied charge.

Determining the internal resistance will give information on the state of the battery.

4.3.1 The Battery Parameters

Index	Object	Name	Type	Access	Mem.
3020_h	RECORD	*battery parameter*	-	rw	Y

Table 22: Objects with the battery parameters.

4.3.1.1 Object 3020$_h$: *battery parameter*

This object summarizes the battery parameters. The battery is composed of eight blocs that are connected in series. Each bloc contains an interface for temperature and bloc voltage read out.

Index	Name	Object				
3020$_h$	*battery parameter*	RECORD	Battery parameter			
Sub-Index	**Name**	**Type**	**Attr**	**Map**	**Default-Value**	**Description**
0	*number of elements*	unsigned8	ro	N	8	Number of entries
1	*type*	unsigned8	rw	N	A1$_h$	Battery type, see table below
2	*nominal capacity*	unsigned16	rw	N	3000	Nominal capacity of the battery in mAh
3	*charging current*	unsigned16	rw	N	1000	Charge current in mA
4	*number of blocs*	unsigned16	rw	N	6	Number of battery blocs
5	*nominal bloc voltage*	unsigned16	rw	N	36000	Nominal bloc voltage in mV
6	*maximum bloc voltage*	unsigned16	rw	N	41000	Maximum bloc voltage in mV
7	*lead resistance*	unsigned16	rw	N	100	Lead resistance to the batteries in mOhm
8	*internal resistance*	unsigned16	rw	N	150	Nominal internal resistance of the battery in mOhm

Battery type

The battery type is composed of a chemical code and a sub type description. The encoding occurs pursuant to CiA DS418, appendix A.

For the PMM applicable types are:

Value	Battery type
A1$_h$	Lithium ionic, Type 1
18$_h$	Lead AGM
19$_h$	Sealed Lead
1A$_h$	Lead, hybrid

Table 23: encoding of battery types.

At this time (status July 2007) only Lithium ionic batteries (type A1) are supported.

maximum bloc voltage

If one of the battery blocs reaches this voltage upon charging, it will be charge up to the voltage threshold until the minimum bloc voltage is reached. The extent of this voltage influences the battery's durability and the capacity. A higher charge voltage results in a higher capacity but a shorter durability.

lead resistance

The lead resistance of the battery lead including the transfer resistance has to be calculated in order to identify the factual internal resistance of the battery.

internal resistance

The internal resistance of the battery has to be entered here. Through comparison of the measured value, the durability of the battery is identified.

4.3.2 Battery Condition

Index	Object	Name	Type	Access	Mem.
3010_h	VAR	*actual battery temperature*	integer16	ro	N
$301F_h$	ARRAY	*battery temperature.bloc*	integer16	ro	N
3050_h	VAR	*cumulative total Ah charge*	unsigned32	rw	Y
3051_h	VAR	*Ah expended since last charge*	integer16	ro	Y
3052_h	VAR	*Ah returned during last charge*	integer16	ro	Y
3053_h	VAR	*actual battery capacity*	unsigned16	ro	N
3060_h	VAR	*actual battery voltage*	unsigned16	ro	N
3061_h	ARRAY	*battery voltage.bloc*	unsigned16	ro	N
3067_h	VAR	*battery off-load voltage*	unsigned16	ro	N
3070_h	VAR	*actual battery current*	integer32	ro	N
3080_h	VAR	*battery state of charge*	unsigned8	ro	N
3081_h	VAR	*battery state of health*	unsigned8	ro	N
$30B4_h$	VAR	*battery internal resistance*	unsigned16	ro	N
$30E3_h$	VAR	*selftest time*	unsigned16	rw	Y

Table 24: Battery condition objects.

4.3.2.1 Object 3010_h: *actual battery temperature*

This displays the factual battery temperature. It is calculated on the basis of the temperatures measured in the blocs. If the lowest temperature is below 10° C, the minimum bloc temperature is displayed here, otherwise the maximum temperature is displayed. This value is utilized by the temperature controller of the battery box.

Index	Name	Type	Attr	Map	Default-Value	Description
3010_h	*actual battery temperature*	integer16	ro	Y		Measured battery temperature in 0,1℃

4.3.2.2 Object 301F$_h$: *battery temperature.bloc*

Index	**Name**	**Object**				
301F$_h$	*battery temperature*	ARRAY	Temperatures in the different battery blocs			
Sub-Index	**Name**	**Type**	**Attr**	**Map**	**Default-Value**	**Description**
0	*number of elements*	unsigned8	ro	N	8	Number of elements
1	*bloc 1*	integer16	ro	Y		Temperature in 1st bloc in 0,1°C
...	...	...	...	...	...	...
8	*bloc 8*	integer16	ro	Y		Temperature in 8th bloc in 0,1°C

4.3.2.3 Object 3050$_h$: *cumulative total Ah charge*

In this object, the voltage fed to the battery is added during each charging of the battery. On switch-off of the machine, the value is preserved.

After a battery change, the value has to be reset to Zero, which is why write access is allowed.

Index	**Name**	**Type**	**Attr**	**Map**	**Default-Value**	**Description**
3050$_h$	*cumulative total Ah charge*	unsigned32	rw	Y		Cumulative battery charge in mAh

4.3.2.4 Object 3051$_h$: *Ah expended since last charge*

Index	**Name**	**Type**	**Attr**	**Map**	**Default-Value**	**Description**
3051$_h$	*Ah expended since last charge*	integer16	ro	Y		Charge expended (in mAh) since last charge

On machine switch-off, the value is preserved. It will be reset to Zero at the next charging cycle.

4.3.2.5 Object 3052$_h$: *Ah returned during last charge*

Index	**Name**	**Type**	**Attr**	**Map**	**Default-Value**	**Description**
3052$_h$	*Ah returned during last charge*	integer16	ro	Y		Charge returned (in mAh) during a charging cycle

On machine switch-off, the value is preserved. It will be reset to Zero at the next charging cycle.

4.3.2.6 Object 3053$_h$: *actual battery capacity*

The actual battery capacity is calculated during off-load time. During the charging and discharging phase, the expended and returned charge is counted in.

Index	Name	Type	Attr	Map	Default-Value	Description
3053_h	*actual battery capacity*	unsigned16	ro	Y		Actual battery capacity in mAh

4.3.2.7 Object 3060_h: *actual battery voltage*

This object covers the battery voltage at clamps X19.

Index	Name	Type	Attr	Map	Default-Value	Description
3060_h	*actual battery voltage*	unsigned16	ro	Y		Measured battery voltage in Volt

4.3.2.8 Object 3061_h: *battery voltage.bloc*

The single voltages are measured by interfaces to the batteries. It contains only as many measured values as battery blocs are connected.

Index	Name	Object				
3061_h	*battery voltage*	ARRAY	Voltage in the respective battery blocs			
Sub-Index	**Name**	**Type**	**Attr**	**Map**	**Default-Value**	**Description**
0	*number of elements*	unsigned8	ro	N	8	Number of elements
1	*bloc 1*	unsigned16	ro	Y		Voltage in 1st bloc in mV
...	...	...	...	...	...	...
8	*bloc 8*	unsigned16	ro	Y		Voltage in 8th bloc in mV

4.3.2.9 Object 3067_h: *battery off-load voltage*

The off-load voltage updates as long as no battery current is detected.

Index	Name	Type	Attr	Map	Default-Value	Description
3067_h	*actual off-load voltage*	unsigned16	ro	Y		Off-load voltage of the battery in Volt

4.3.2.10 Objekt 3068_h: *battery test voltage*

The minimal battery voltage during the automatic battery test.

Index	Name	Type	Attr	Map	Default-Value	Description
3068_h	*battery test voltage*	unsigned16	ro	J		minimal test voltage of the battery in Volt

4.3.2.11 Object 3070$_h$: *actual battery current*

Positive battery current means: electricity is used, discharged, negative values: the electricity flows into the battery (charging)

Since the differences between the charge current and the discharge current is very high, two different measuring points are used. During charge the current has a resolving power of 2,5mA; during discharge the resolving power is 400mA.

Index	Name	Type	Attr	Map	Default-Value	Description
3070$_h$	*actual battery current*	integer32	ro	Y		Battery current in mA

4.3.2.12 Object 3080$_h$: *battery state of charge*

The battery state of charge is identified during idle-time. During the charging and discharging phase, the expended and returned charge is counted in.

Index	Name	Type	Attr	Map	Default-Value	Description
3080$_h$	*battery state of charge*	unsigned8	ro	Y		Battery's state of charge in %

4.3.2.13 Object 3081$_h$: *battery state of health*

The state of health of a Lithium ionic battery can be identified from the measured internal resistance of a new cell. The actual internal resistance is identified during discharge. A significant value is available afterwards only.

Index	Name	Type	Attr	Map	Default-Value	Description
3081$_h$	*battery state of health*	unsigned8	ro	Y		Battery's state of health in %

4.3.2.14 Object 30B4$_h$: *battery internal resistance*

The state of health of a Lithium ionic battery can be identified by comparing the measured internal resistance to the internal resistance of a new cell. The actual internal resistance is identified during discharge. The nominal internal resistance is entered in the battery parameters (object 3020$_h$, page 24).

Index	Name	Type	Attr	Map	Default-Value	Description
30B4$_h$	*battery internal resistance*	unsigned16	ro	Y		Measured battery internal resistance in mOhm

4.3.2.15 Object 30E3$_h$: *selftest time*

The batteries of the Pitch System can be testet automatically with the booster and the internal brake chopper. The result is seen in *test status* and *battery test voltage*. The test can be started via the PMM_controlword or automatically after the time given in this object. With zero the test will not start automatically.

Index	Name	Type	Attr	Map	Default-Value	Description
$30E3_h$	*selftest time*	unsigned16	rw	N	0	Cycle time for automatic battery test in minutes. 0 switches atomatic test of. minimum time 60min.

4.3.3 Battery charger Parameters

The battery charger consists of two stages: a „regulated mains adapter" composed of current and voltage controller which regulates the supply voltage that feeds the internal 24V mains adapter and the battery charger. The batter charger controls this "mains adapter" in order to run the characteristic charge curve.

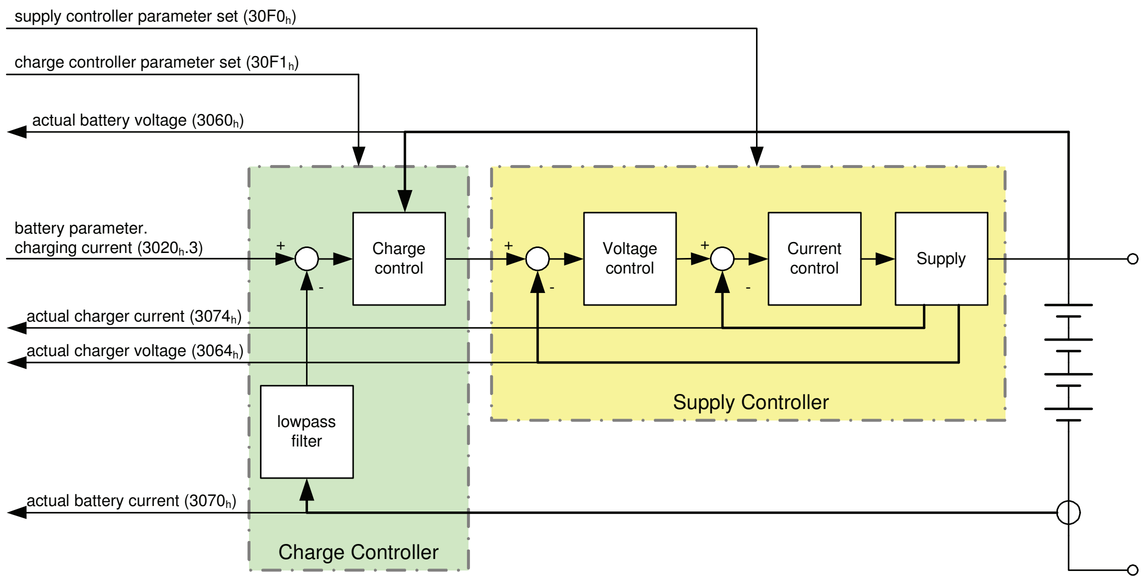

Figure 1: Battery charger structure

Index	Object	Name	Type	Access	Mem.
$30F0_h$	ARRAY	*supply controller parameter set*	unsigned16	rw	Y
$30F1_h$	ARRAY	*charge controller parameter set*	unsigned16	rw	Y
3064_h	VAR	*actual charger voltage*	unsigned16	ro	N
3074_h	VAR	*actual charger current*	integer16	ro	N

Table 25: Battery charger objects.

4.3.3.1 Object $30F0_h$: *supply controller parameter set*

The supply controller is a cascaded PI controller with the respective parameters. The controller parameters are factory preset and must not be changed. They can vary depending on the machine type. Write access for these parameters is not allowed in normal operation.

The parameters described herein must only be changed by qualified specialized staff. Incorrectly set parameters can cause machine damage.

Index	Name	Object				
30F0$_h$	*supply controller parameter set*	ARRAY	Supply controller parameters			
Sub-Index	**Name**	**Type**	**Attr**	**Map**	**Default-Value**	**Description**
0	*number of elements*	unsigned8	ro	N	6	Number of elements
1	*voltage gain*	unsigned16	r(w)	N	2000	Voltage controller P-ratio
2	*voltage integration time*	unsigned16	r(w)	N	4000	Voltage controller I-ratio
3	*current limit*	unsigned16	r(w)	N	1500	Current limiting in mA
4	*current gain*	unsigned16	r(w)	N	64	Voltage controller P-ratio
5	*current integration time*	unsigned16	r(w)	N	1500	Voltage controller I-ratio
6	*PWM limit*	unsigned16	r(w)	N	70	PWM adjustment value limiting in %

4.3.3.2 Object 30F1$_h$: *charge controller parameter set*

The battery charger contains a PI controller and a low pass filter with the respective parameters. The controller parameters (sub index 1 to 3) are factory preset and must not be changed. They can vary depending on the machine type. Write access for these parameters is not allowed in normal operation.

The parameters described herein must only be changed by qualified specialized staff. Incorrectly set parameters can cause machine damage.

Charging voltage and starting voltage as well as cut-off current must be adapted to the battery type and bloc number.

Index	Name	Object				
30F1$_h$	*charge controller parameter set*	ARRAY	Battery charger parameter			
Sub-Index	**Name**	**Type**	**Attr**	**Map**	**Default-Value**	**Description**
0	*number of elements*	unsigned8	ro	N	6	Number of elements
1	*current gain*	unsigned16	r(w)	N	512	Voltage controller P-ratio
2	*current integration time*	unsigned16	r(w)	N	500	Voltage controller I-ratio
3	*filter time*	unsigned16	r(w)	N	500ms	Filter time of the PT-element in ms
4	*charging voltage*	unsigned16	rw	N	246V	Charging voltage in Volt
5	*threshold voltage*	unsigned16	rw	N	241V	Starting voltage of the charger in Volt
6	*switch-off current*	unsigned16	rw	N	300mA	Closing voltage current in mA

charging voltage

Should be set to maximum bloc voltage * number of blocs. In case of 6 blocs Lithium ionic type A1 this means: 41V * 6 = 246V.

threshold voltage

Here the voltage at which the battery has to be recharged is set.

switch-off current

Falling below this charge current will stop the charge. It should be at 1/10 of the nominal capacity.

4.3.3.3 Object 3064$_h$: *actual charger voltage*

This object contains the output voltage of the charger. As long as the battery is being charged, it is about 1 Volt above the battery voltage. If the battery is not charged, the voltage regulates to about 10 Volt below off-load voltage of the battery.

Index	Name	Type	Attr	Map	Default-Value	Description
3064$_h$	*actual charger voltage*	unsigned16	ro	Y		Charger voltage in Volt

4.3.3.4 Object 3074$_h$: *actual charger current*

This is the output current of the charge controller. It consists of the battery's charge current and the input current of the 24V mains adapter.

Index	Name	Type	Attr	Map	Default-Value	Description
3074$_h$	*actual charger current*	integer16	ro	Y		Charger current in mA

4.4 The Boost Converter

In case of a power failure, the boost converter raises the battery voltage to intermediate circuit level. It automatically activates if the intermediate circuit falls below the starting voltage. It is a cascading PI-controller that controls the output voltage and passes into current control when a current limiting value is reached.

In order to protect the cells, the current is limited in such a way that the battery voltage does not drop below a minimum voltage.

4.4.1 Parameter

Index	Object	Name	Type	Access	Mem.
$30FB_h$	ARRAY	*booster parameter set*	unsigned16	rw	Y

Table 26: Boost converter parameters

4.4.1.1 Object $30FB_h$: *booster parameter set*

The controller parameters are factory preset and must not be changed. They can vary depending on the machine type. Write access for these parameters is not allowed in normal operation.

The parameters described herein must only be changed by qualified specialized staff. Incorrectly set parameters can cause machine damage.

Index	**Name**	**Object**				
$30FB_h$	*booster parameter set*	ARRAY	Boost converter parameters			
Sub-Index	**Name**	**Type**	**Attr**	**Map**	**Default-Value**	**Description**
0	*number of elements*	unsigned8	ro	N	9	Number of elements
1	*voltage gain*	unsigned16	r(w)	N	128	Voltage controller P-ratio
2	*voltage integration time*	unsigned16	r(w)	N	2000	Voltage controller I-ratio
3	*current limit*	unsigned16	r(w)	N	90A	Current limiting in A
4	*current gain*	unsigned16	r(w)	N	64	Current controller P-ratio
5	*current integration time*	unsigned16	r(w)	N	5000	Current controller I-ratio
6	*PWM limit*	unsigned16	r(w)	N	80%	PWM adjustment value limiting in %
7	*target voltage*	unsigned16	rw	N	600V	Control voltage in Volt
8	*switch-on voltage*	unsigned16	rw	N	420V	Starting voltage in Volt
9	*inductance*	unsigned16	rw	N	180µH	Booster reactor inductance in µH

target voltage

The control voltage of the boost converter has to be higher than the nominal dc link voltage in order for no inrush current to flow on re-starting of the mains voltage.

switch-on voltage

If the direct current link voltage falls below this value during a power failure, the boost converter activates and regulates the current link voltage at the value in *target voltage*.

inductance

The inductance of the booster reactor is needed in order to correctly calculate the booster current. The value can be gathered from the reactor's type label.

4.4.2 Actual Values

Index	Object	Name	Type	Access	Mem.
3073_h	VAR	*actual booster current*	integer16	ro	N
$30B0_h$	VAR	*actual load booster*	unsigned8	ro	N

Table 27: Boost converter actual values.

4.4.2.1 Object 3073_h: *actual booster current*

This displays the output current of the boost converter.

Index	Name	Type	Attr	Map	Default-Value	Description
3073_h	*actual booster current*	integer16	ro	Y		Booster current in Ampere

4.4.2.2 Object $30B0_h$: *actual load booster*

This object displays the capacity utilization of the booster output stage, in other words the relation between switch-on and switch-off time.

Index	Name	Type	Attr	Map	Default-Value	Description
$30B0_h$	*actual load booster*	unsigned8	ro	Y		Capacity utilization of the boost converter in %

4.5 The Brake Chopper

The brake chopper ensures that the direct current link voltage does not increase above a specific value due to feed back of a brake application of the axis. When an upper threshold voltage is reached, the brake resistance is cyclically switched to the intermediate circuit. If the voltage drops below the lower threshold, the brake chopper deactivates.

4.5.1 Parameters

Index	Object	Name	Type	Access	Mem.
$30FC_h$	ARRAY	*brake chopper*	unsigned16	rw	Y

Table 28: Brake chopper parameters.

4.5.1.1 Object $30FC_h$: *brake chopper*

This sets the switch-on and switch-off threshold of the brake chopper. The voltage values must not lie below the control voltage (*booster parameter.target voltage*) of the booster.

Index	Name	Object				
$30FC_h$	*chopper parameter set*	ARRAY	Brake chopper parameters			
Sub-Index	**Name**	**Type**	**Attr**	**Map**	**Default-Value**	**Description**
0	*number of elements*	unsigned8	ro	N	2	Number of elements
1	*switch-on voltage*	unsigned16	rw	N	670	Switch-on threshold in Volt
2	*switch-off voltage*	unsigned16	rw	N	650	Switch-off threshold in Volt

4.5.2 Actual Values

Index	Object	Name	Type	Access	Mem.
$30B1_h$	VAR	*actual load chopper*	unsigned8	ro	N

Table 29: Brake chopper actual values.

4.5.2.1 Object $30B1_h$: *actual load chopper*

This object displays the actual capacity utilization of the brake chopper. 100% stand for full capacity utilization with 300W.

Index	Name	Type	Attr	Map	Default-Value	Description
$30B1_h$	*actual load chopper*	unsigned8	ro	Y		Capacity utilization of the brake chopper in %

4.6 Temperature Monitoring and Regulation

The PMM features three temperature controllers in order to actuate external heaters, coolers and fans. Furthermore, four temperatures can be monitored for exceeding or falling below of threshold values and a specific alarm can be activated.

The following temperature measuring points exist:

- Battery temperature
- Heat sink temperature inside the machine
- Two externally lockable KTY110 feeler gauges.

Two relay outputs and the digital 24V outputs are available for actuation of external air conditioning units.

The control- and monitoring modules, are fixedly assigned the temperature ports. The control outputs can be assigned freely.

The temperature control units do not regulate a specific temperature but rather prevent the exceeding or falling below the appointed temperature range:

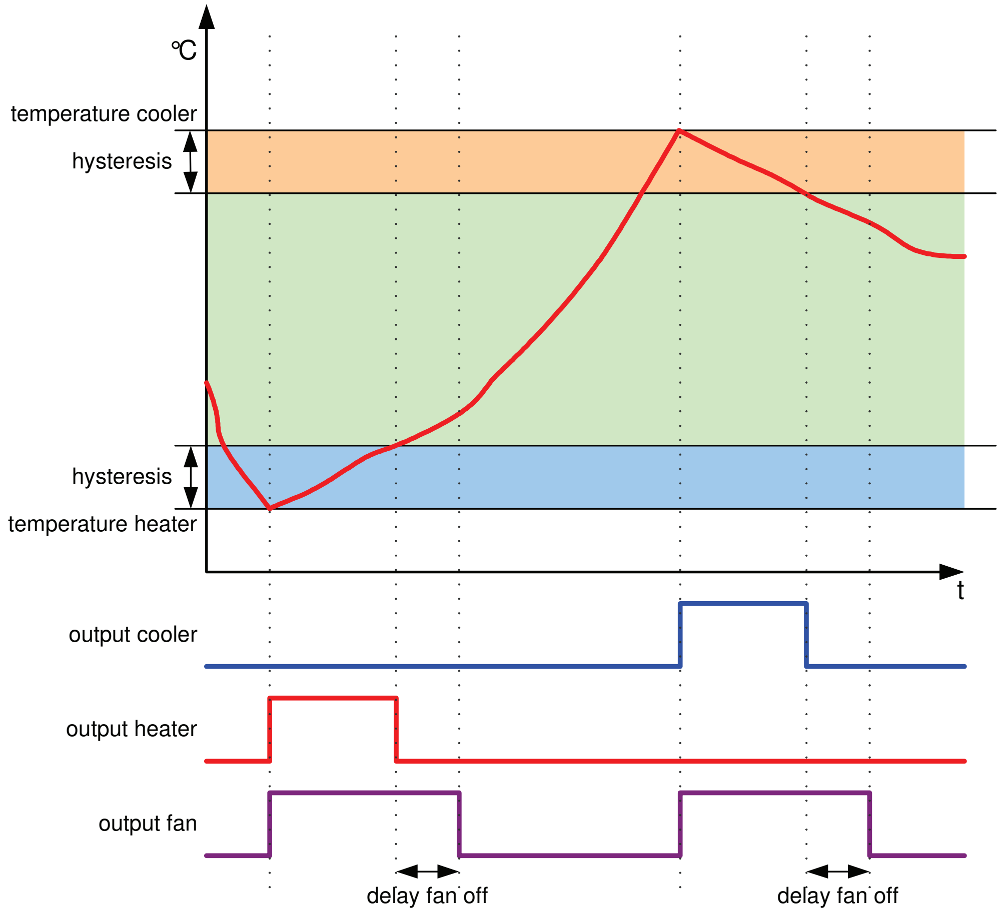

Figure 2: Temperature control unit function

In case of falling below the temperature threshold heating, the heater and fan are activated. If the temperature increases by the hysteresis, the heater deactivates.

In case of exceeding the temperature threshold cooling, cooler and fan are activated. If the temperature drops by the hysteresis, the cooling deactivates.

In both cases the fan will only be deactivated after expiration of the follow-up time.

4.6.1 Parameters

Index	Object	Name	Type	Access	Mem.
3110_h	ARRAY	*temperature supervisor battery*	integer16	rw	Y
3111_h	ARRAY	*temperature supervisor heat sink*	integer16	rw	Y
3112_h	ARRAY	*temperature supervisor sensor1*	integer16	rw	Y
3113_h	ARRAY	*temperature supervisor sensor2*	integer16	rw	Y
3201_h	RECORD	*temperature controller battery*	-	rw	Y
3202_h	RECORD	*temperature controller sensor1*	-	rw	Y
3203_h	RECORD	*temperature controller sensor2*	-	rw	Y
3204_h	RECORD	*motor fan controller*	-	rw	Y

Table 30: Brake chopper parameters.

4.6.1.1 Object 3110_h: *temperature supervisor battery*

Index	Name	Object				
3110_h	*temperature supervisor battery*	ARRAY	Battery temperature monitoring			
Sub-Index	**Name**	**Type**	**Attr**	**Map**	**Default-Value**	**Description**
0	*number of elements*	unsigned8	ro	N	3	Number of elements
1	*warning limit*	integer16	rw	N	50,0°C	Warning limit in 0,1°C
2	*error limit*	integer16	rw	N	55,0°C	Error limit 0,1°C
3	*lower limit*	integer16	rw	N	0,0°C	Lower limit in 0,1°C (warning)

4.6.1.2 Object 3111_h: *temperature supervisor heat sink*

Index	Name	Object				
3111_h	*temperature supervisor heat sink*	ARRAY	Heat sink temperature monitoring			
Sub-Index	**Name**	**Type**	**Attr**	**Map**	**Default-Value**	**Description**
0	*number of elements*	unsigned8	ro	N	3	Number of elements
1	*warning limit*	integer16	rw	N	75,0°C	Warning limit in 0,1°C
2	*error limit*	integer16	rw	N	80,0°C	Error limit in 0,1°C
3	*lower limit*	integer16	rw	N	-20,0°C	Lower limit in 0,1°C (warning)

4.6.1.3 Objects 3112_h an 3113_h: *temperature supervisor sensor 1* and *2*

Index	Name	Object				
3112_h	*temperature supervisor sensor1*	ARRAY	Sensor 1 temperature monitoring, clamp X15.1/X15.2			
3113_h	*temperature supervisor sensor2*	ARRAY	Sensor 2 temperature monitoring, clamp X15.3/X15.4			
Sub-Index	**Name**	**Type**	**Attr**	**Map**	**Default-Value**	**Description**
0	*number of elements*	unsigned8	ro	N	3	Number of elements
1	*warning limit*	integer16	rw	N	45,0°C	Warning limit in 0,1°C
2	*error limit*	integer16	rw	N	50,0°C	Error limit in 0,1°C
3	*lower limit*	integer16	rw	N	-20,0°C	Lower limit in 0,1°C (warning)

4.6.1.4 Object 3201_h: *temperature controller battery*

In the base setting, this controller has the following outputs:

Heating: Clamp X16.1/X16.2 (relay output)
Cooling: not used
Fan: Clamp X14.5 (24V output, 0,5A)

Index	Name	Object				
3201_h	*temperature controller battery*	RECORD	Battery temperature controller			
Sub-Index	**Name**	**Type**	**Attr**	**Map**	**Default-Value**	**Description**
0	*number of elements*	unsigned8	ro	N	7	Number of elements
1	*heater*	integer16	rw	N	0,0°C	Temperature threshold heating in 0,1°C
2	*cooler*	integer16	rw	N	45,0°C	Temperature threshold cooling in 0,1°C
3	*hysteresis*	integer16	rw	N	5,0K	Hysteresis in 0.1K
4	*output heater*	unsigned16	rw	N	0040_h	Mask for heater output
5	*output cooler*	unsigned16	rw	N	0000_h	Mask for cooler output
6	*output fan*	unsigned16	rw	N	0010_h	Mask for fan output
7	*delay fan off*	unsigned16	rw	N	120s	Falling delay fan in seconds

The bit mask for selecting the control output can be found in the table on page 44.

4.6.1.5 Object 3202_h: *temperature controller sensor 1*

This controller is deactivated in the base setting by setting all output masks to 0.

Index	Name	Object				
3202_h	*temperature controller sensor1*	RECORD	Temperature controller sensor 1, clamp X15.1/X15.2			
Sub-Index	**Name**	**Type**	**Attr**	**Map**	**Default-Value**	**Description**
0	*number of elements*	unsigned8	ro	N	7	Number of elements
1	*heater*	integer16	rw	N	0	Temperature threshold heating in 0,1°C
2	*cooler*	integer16	rw	N	45,0°C	Temperature threshold cooling in 0,1°C
3	*hysteresis*	integer16	rw	N	5,0°C	Hysteresis in 0.1K
4	*output heater*	unsigned16	rw	N	0000_h	Mask for heater output
5	*output cooler*	unsigned16	rw	N	0000_h	Mask for cooler output
6	*output fan*	unsigned16	rw	N	0000_h	Mask for fan output
7	*delay fan off*	unsigned16	rw	N	120s	Falling delay fan in seconds

The bit mask for selecting the control output can be found in the table on page 44.

4.6.1.6 Object 3203_h: *temperature controller sensor 2*

This controller's base setting has the following outputs:

Heating: Clamp X16.3/X16.4 (relay output)
Cooling: not used
Fan: Clamp X14.6 (24V output, 0,5A)

Index	Name	Object				
3203_h	*temperature controller sensor2*	RECORD	Temperature controller sensor 2, clamp X15.3/X15.4			
Sub-Index	**Name**	**Type**	**Attr**	**Map**	**Default-Value**	**Description**
0	*number of elements*	unsigned8	ro	N	7	Number of elements
1	*heater*	integer16	rw	N	0,0℃	Temperature threshold heating in 0,1℃
2	*cooler*	integer16	rw	N	45,0℃	Temperature threshold cooling in 0,1℃
3	*hysteresis*	integer16	rw	N	5,0K	Hysteresis in 0.1K
4	*output heater*	unsigned16	rw	N	0080_h	Mask for heater output
5	*output cooler*	unsigned16	rw	N	0000_h	Mask for cooler output
6	*output fan*	unsigned16	rw	N	0020_h	Mask for fan output
7	*delay fan off*	unsigned16	rw	N	120s	Falling delay fan in seconds

The bit mask for selecting the control output can be found in the table on page 44.

4.6.1.7 Object 3204_h: *motor fan controller*

The PMM can also take over control of the motor fan cooler. Motor temperature is queried by the PMC through the CAN bus. Above the temperature threshold the cooler is activated. If the temperature drops by the hysteresis, the cooler is turned-off.

This function is only activated if *output fan* has an output mask.

Index	Name	Object				
3204_h	*motor fan controller*	RECORD	Motor fan controller			
Sub-Index	**Name**	**Type**	**Attr**	**Map**	**Default-Value**	**Description**
0	*number of elements*	unsigned8	ro	N	3	Number of elements
1	*temperature*	integer16	rw	N	80,0℃	Temperature threshold cooling in 0,1℃
2	*hysteresis*	integer16	rw	N	5,0K	Hysteresis in 0.1K
3	*output fan*	unsigned16	rw	N	0000_h	Mask for fan output

The bit mask for selecting the control output can be found in the table on page 44.

4.6.2 Status Message and Actual Values

Index	Object	Name	Type	Access	Mem.
3006_h	VAR	*battery temperature status*	unsigned8	ro	N
3007_h	VAR	*heat sink temperature status*	unsigned8	ro	N
3008_h	VAR	*sensor1 temperature status*	unsigned8	ro	N
3009_h	VAR	*sensor2 temperature status*	unsigned8	ro	N
3010_h	VAR	*actual battery temperature*	integer16	ro	N
3011_h	VAR	*actual heat sink temperature*	integer16	ro	N
3012_h	VAR	*temperature sensor1*	integer16	ro	N
3013_h	VAR	*temperature sensor2*	integer16	ro	N

Table 31: Brake chopper actual value.

4.6.2.1 Objects 3006_h- 3009_h: *temperature status*

Index	Name	Type	Attr	Map	Default-Value	Description
3006_h	*battery temperature status*	unsigned8	ro	Y		Status messages temperature monitoring and controlling of: Battery
3007_h	*heat sink temperature status*	unsigned8	ro	Y		Heat sink
3008_h	*sensor1 temperature status*	unsigned8	ro	Y		Sensor 1, clamp X15.1/X15.2
3009_h	*sensor2 temperature status*	unsigned8	ro	Y		Sensor 2, clamp X15.3/X15.4

The individual bit in the temperature status values have the following meaning:

Bit	Meaning	Error code			
		Battery	Unit	Sensor1	Sensor2
0	1 = Lower temperature threshold undercut	4420_h	4220_h	4120_h	4121_h
1	1 = Fan on				
2	1 = Heater on				
3	1 = Cooler on				
4	1 = Warning limit reached*	4410_h	4210_h	4110_h	4111_h
5	1 = Error limit reached, emergency message occurs simultaneously	4410_h	4210_h	4110_h	4111_h
6	1 = Sensor short circuit				
7	1 = Sensor interrupt				

Table 32: Bits in the *temperature status*.

* In case of a warning, the error code is entered into the object, but no emergency message occurs.

Bits 1 to 3 are always 0 in object 3007_h.

4.6.2.2 Object 3010_h: *actual battery temperature*

This displays the actual battery temperature. It is calculated by the temperature measured inside the blocs. If the lowest temperature is below 10℃, the minimum bloc temperature is displayed here, otherwise the maximum bloc temperature.

Index	Name	Type	Attr	Map	Default-Value	Description
3010_h	*actual battery temperature*	integer16	ro	Y		Battery temperature measured in 0,1℃

4.6.2.3 Object 3011_h: *actual heat sink temperature*

The sensor for this temperature is located directly at the heat sink of the machine.

Index	Name	Type	Attr	Map	Default-Value	Description
3011_h	*actual heat sink temperature*	integer16	ro	Y		Temperature of the heat sink in 0,1℃

4.6.2.4 Objects 3012_h and 3013: *temperature 1* and *2*

The temperature for the externally connected KTY110 sensors is displayed here. Short circuit or wire break, or sensors not connected respectively, are identified and displayed by non-plausible values as well as by bits in the temperature status. An error message does not occur, though.

Index	Name	Type	Attr	Map	Default-Value	Description
3012_h	*temperature 1*	integer16	ro	Y		Temperature at the temperature sensors in 0,1℃.
3013_h	*temperature 2*	integer16	ro	Y		-32768 means short circuit 32767 means wire break

4.7 Inputs and Outputs

The PMM possesses an analogue input for 0-10V, several digital inputs for 24V, ten digital outputs for 24V and two relay outputs for up to 230V. Those inputs and outputs can be used by the controller or linked with internal functions.

4.7.1 Objects

Index	Object	Name	Type	Access	Mem.
$30A0_h$	VAR	*read analog input 1*	unsigned16	ro	N
$30FD_h$	VAR	*digital inputs*	unsigned32	ro	N
$30E2_h$	VAR	*pulse width input check signal*	unsigned16	rw	Y
$30FE_h$	ARRAY	*digital outputs*	unsigned32	rw	N

Table 33: Input and output objects.

4.7.1.1 Object $30A0_h$: *read analog input 1*

The PMM possesses an analog input from 0..10V. The resolution is 10 Bit.

Index	Name	Type	Attr	Map	Default-Value	Description
$30A0_h$	*read analog input 1*	unsigned16	ro	Y		Analog input voltage in mV

4.7.1.2 Object $30FD_h$: *digital inputs*

The PMM's digital inputs can be back-read in this object. Some can also be linked with internal functions, e.g. alert functions. The linkage occurs in the objects of those functions.

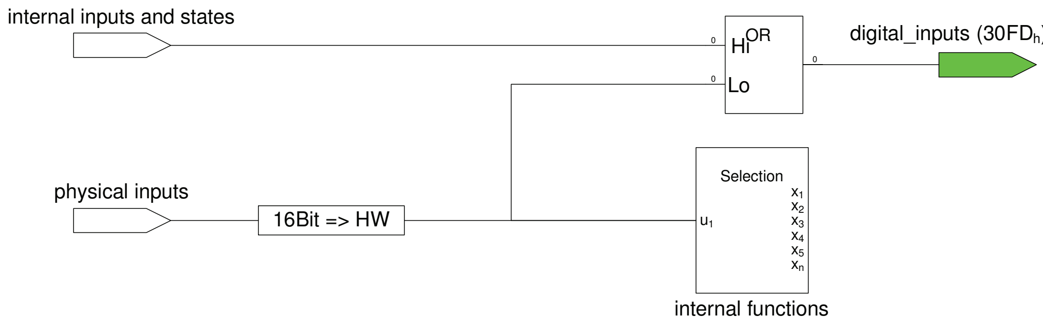

Figure 3: Digital input processing

Index	Name	Type	Attr	Map	Default-Value	Description
$30FD_h$	*digital inputs*	unsigned32	ro	Y		Digital inputs of the PMM

The following table shows the configuration of the individual bits

In the column „bit mask“ those values are displayed hexadecimally, that have to be entered in the respective internal functions for the selection of the input.

Bit	Clamp	Configuration	Bit mask
0	X13.10	Emergency Feather Command (EFC)	0001_h
1		not used, always 0	0002_h
2	X13.8	EFC direct	0004_h
3	X13.7		0008_h
4	X13.6 to X13.5 24V feed-in		0010_h
5		not used, always 0	0020_h
6		not used, always 0	0040_h
7		not used, always 0	0080_h
8	X14.12		0100_h
9	X14.11	1 = Lightning protection OK ($FFF1_h$)	0200_h
10	X14.10		0400_h
11	X14.9		0800_h
12	X14.8 to X14.7 24V feed-in		1000_h
13		not used, always 0	2000_h
14		not used, always 0	4000_h
15		not used, always 0	8000_h
16	X13.10 (feedback signal)		-
17		not used, always 0	-
18	X13.8 (feedback signal)		-
19	X13.7 (feedback signal)		-
20	X13.6 to X13.5 24V feed-in		-
21		not used, always 0	-
22		not used, always 0	-
23		not used, always 0	-
24		not used, always 0	-
25		not used, always 0	-
26		not used, always 0	-
27	Push-button		-
28	Selector switch manual function		-
29	Selector switch manual function		-
30	Selector switch manual function		-
31	Selector switch manual function		-

Table 34: Configuration of inputs and bit mask.

4.7.1.3 Object $30E2_h$: *pulse width input check signal*

Inputs X13.7, X13.8 and X13.10 can be checked for wire break and short circuit, respectively, without activating an emergency stop (EFC), e.g. For this, the controller must set the input for one cycle (**max. 40ms**) to 0. A feedback signal extended by pulse width can then be checked in bits 16, 18 or 19 (see table).

The emergency mode will executed after about 50ms, therefore, the input should be at 0V for a maximum of 40ms for test purposes!

Index	Name	Type	Attr	Map	Default-Value	Description
30E2$_h$	*pulse width input check signal*	unsigned16	rw	N	2000ms	Pulse width time for input monitoring in ms

4.7.1.4 Object 30FE$_h$: *digital outputs*

The PMM provides digital outputs that can be freely programmed in parts. Inputs set by the machine can be read-back. Furthermore, the state of the LED can be read, but not be operated from the controller.

The bit mask assists in selecting whether an output is addressed by the control unit or by the internal function of the PMM.

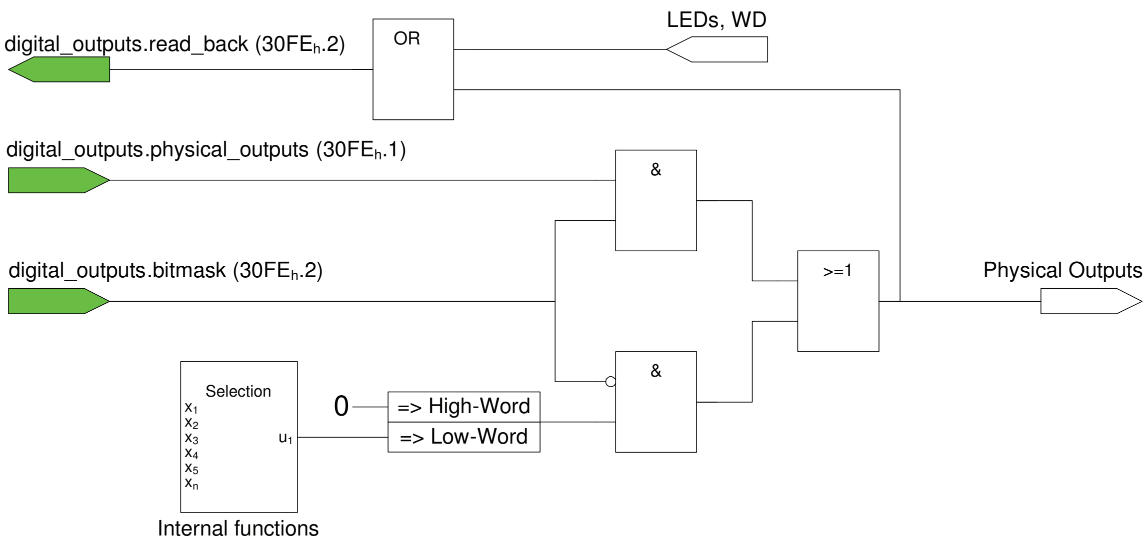

Figure 4: Digital output processing

Index	Name	Object				
30FE$_h$	*digital outputs*	ARRAY	Digital outputs of the PMM			
Sub-Index	**Name**	**Type**	**Attr**	**Map**	**Default-Value**	**Description**
0	*number of elements*	unsigned8	ro	N	3	Number of elements
1	*physical outputs*	unsigned32	rw	Y	0	Output level: 1 = 24V, 0 = 0V
2	*bitmask*	unsigned32	rw	Y	0	Bit mask, only bits with the value 1 can be taken from *physical outputs*.
3	*read back*	unsigned32	ro	Y		Read-back state of the outputs.

The following table represents the output configuration. Bits 16 to 31 are read-back only.

In the column „bit mask“ those values are displayed hexadecimally, that have to be entered in the respective internal functions for the selection of the control output. It Is also possible to parallel outputs for cases of higher current load. For this, the respective values will be added for the mask and entered in the function.

Example: Parallel clamps X14.3 and X14.4: Mask = 0002_h + 0004_h = 0006_h

Bit	Clamp	Default configuration	Bit mask
0	X14.1 (24V, 1A)		0001_h
1	X14.2 (24V, 1A)		0002_h
2	X14.3 (24V, 0,5A)		0004_h
3	X14.4 (24V, 0,5A)		0008_h
4	X14.5 (24V, 0,5A)	Temperature controller battery fan	0010_h
5	X14.6 (24V, 0,5A)	Temperature controller 2 fan	0020_h
6	X16.1 + X16.2 230VAC Relay	Temperature controller battery heater	0040_h
7	X16.3 + X16.4 230VAC Relay	Temperature controller 2 heater	0080_h
8	X13.1 (24V, 0,5A)		0100_h
9	X13.2 (24V, 0,5A)		0200_h
10	X13.3 (24V, 0,5A)	Warning	0400_h
11	X13.4 (24V, 0,5A)	Fault	0800_h
12	not used		1000_h
13	not used		2000_h
14	not used		4000_h
15	not used		8000_h
16	X13.11 + X13.12 24VDC Relay	Safety Chain (WD)	-
17	not used		-
18	not used		-
19	not used		-
20	not used		-
21	not used		-
22	not used		-
23	not used		-
24	LED Status machine		-
25	LED disturbance		-
26	LED Watchdog		-
27	LED Field bus 1		-
28	LED Field bus 2		-
29	LED Field bus 3		-
30	not used		-
31	not used		-

Table 35: Configuration of output and bit mask.

Not used outputs can be set in the object and in the bit mask and also appear in the *read back*. They should be set to 0 in order to be compatible with future hardware expansions.

physical outputs

The controller includes these objects in its I/O image and addresses the outputs, as long as the same bit is set in the object *bitmask*.

bitmask

Only those outputs which have their bit set here can be addressed by the controller. For outputs that are utilized by another function (e.g. temperature controller), the bit must be set to 0.

read back

All outputs, even those operated by internal functions, can be read-back here. However, they will not be read-back physically, that means a level connected to a 24 output on the outside will not be detected.

4.7.2 Programmable Alert Outputs

One output each can be programmed for a warning and error message. The level is 24V in order to parallel the outputs("wired or"). By this, a lamp or a signaling contact, e.g., can be addressed.

Index	Object	Name	Type	Access	Mem.
$30FF_h$	ARRAY	*signaling outputs*	unsigned16	rw	Y

4.7.2.1 Object $30FF_h$: *signaling outputs*

Index	Name	Object				
$30FF_h$	*signaling outputs*	ARRAY	Signaling outputs of the PMM			
Sub-Index	**Name**	**Type**	**Attr**	**Map**	**Default-Value**	**Description**
0	*number of elements*	unsigned8	ro	N	2	Number of elements
1	*fault*	unsigned16	rw	Y	0800_h	Mask for signaling output „fault“
2	*warning*	unsigned16	rw	Y	0400_h	Mask for signaling output „warning“

The bit masks for control output selection can be found on page 44.

4.7.3 Programmable Signaling Inputs

Up to 8 error codes ($FFF1_h$ to $FFF8_h$) can be assigned digital inputs, whereupon the type signal (error or warning) and the release level (24V/0V) can be programmed.

One application is, for example, the signaling contact of a lighting protection module. In case of an error, an alarm message can be automatically sent to the control without querying the input.

Index	Object	Name	Type	Access	Mem.
3300_h	ARRAY	*alarm inputs*	unsigned16	rw	Y

4.7.3.1 Object 3300_h: *alarm inputs*

Index	Name	Object				
3300_h	*alarm inputs*	ARRAY	Alarm inputs of the PMM			
Sub-Index	**Name**	**Type**	**Attr**	**Map**	**Default-Value**	**Description**
0	*number of elements*	unsigned8	ro	N	8	Number of elements
1	*number 1*	unsigned16	rw	N	0000_h	Mask for alarm signal $FFF1_h$
...	...	...	...	...	...	...
8	*number 8*	unsigned16	rw	N	0000_h	Mask for alarm signal $FFF8_h$

The bit masks for the input selection can be found in the following table.

With bit 15 of the mask, the message level is set, bit 14 selects the alarm type error or warning.

Example: Input X14.11 shall report failure of the lighting protection with the error code $FFF1_h$. The message is low-active (wire break safe) and shall generate an error and thereby open the safety chain.

The mask to be programmed is 4200_h and is entered in object 3300_h, sub index 1.

Bit	Clamp	Configuration	Bit mask
0	X13.10	Emergency Feather Command (EFC)	0001_h
1			0002_h
2	X13.8	EFC direct	0004_h
3	X13.7		0008_h
4	X13.6 to X13.5 24V feed-in		0010_h
5			0020_h
6			0040_h
7			0080_h
8	X14.12		0100_h
9	X14.11	1 = Lightning protection OK (Code $FFF1_h$)	0200_h
10	X14.10		0400_h
11	X14.9		0800_h
12	X14.8 to X14.7 24V feed-in		1000_h
13			2000_h
14	Selection of fault / warning*	1= Fault, 0 = Warning	4000_h
15	Selection of level*	1= High-active, 0 = Low-active	8000_h

Table 36: Bitmask for signaling inputs

*Only for alert functions

4.8 Position Monitoring

The PMM offers the option to set a signaling output dependent on the position of the axis. The PMC will report the position via the CAN-bus and compare it with two limits. If the actual position lies in between these limits, the output will be set. The output can be preset.

4.8.1 Objects

Index	Object	Name	Type	Access	Mem.
3220_h	RECORD	*position range enable*	-	rw	Y

4.8.1.1 Object 3220_h: *position range enable*

Index	Name	Object				
3220_h	*position range enable*	RECORD	Position range monitoring			
Sub-Index	**Name**	**Type**	**Attr**	**Map**	**Default-Value**	**Description**
0	*number of elements*	unsigned8	ro	N	3	Number of elements
1	*lower limit*	integer16	rw	N	-8700	Lower limit in 0,01°
2	*upper limit*	integer16	rw	N	8700	Upper limit in 0,01°
3	*output*	unsigned16	rw	N	0000_h	Mask for signaling output

The bit masks for the selection of the control outputs can be found in the table on page 44.

4.9 Direct Current Motor Parameters

A dc motor that is directly connected to the intermediate circuit can be run by a programmable acceleration and deceleration chute.

At this point (status July 2007) this function is not yet available.

4.9.1 Objects

Index	Object	Name	Type	Access	Mem.
$30DC_h$	ARRAY	*dc motor parameter set*	unsigned16	rw	Y

4.9.1.1 Object 30DC$_h$: *dc motor parameter set*

Index	Name	Object				
30DC$_h$	*dc motor parameter set*	ARRAY	Parameter for direct injection of the dc motor			
Sub-Index	**Name**	**Type**	**Attr**	**Map**	**Default-Value**	**Description**
0	*number of elements*	unsigned8	ro	N	4	Number of elements
1	*switch-on delay*	unsigned16	rw	N		Switch-on delay in ms
2	*acceleration*	unsigned16	rw	N		Acceleration chute in V/sec
3	*deceleration*	unsigned16	rw	N		Deceleration chute in V/sec
4	*max voltage*	unsigned16	rw	N		Maximum voltage in Volt

5 The Pitch Motion Controller (PMC)

This device is a drive controller that can drive AC synchronic, asynchronic and DC motors. In addition to the functions of a conventional drive controller, it possesses special functions for the operation for blade adjustment, like, e.g., a process setting for the emergency-stop drive (EFC) or an additional transmitter input for the redundant blade encoder.

There are inputs for limit switch, emergency-stop drive, flag position, referencing, measurement position etc.

There are rely outputs for the holding brake and the safety chain (watchdog relay).

The device has a cascaded closed loop controller which consists of current controller, velocity controller and a position controller. The parameters can be found in chapters 5.2, 5.3 and 5.4.

The controller can be operated in the following modes of operation:

- position controlled positioning (chapter 5.10): The end position will be accessed and controlled by the device via a trajectory generator with set speed and acceleration. The controlword gives the starting command.
- velocity-regulated positioning (chapter 5.11): A target speed will be set and controlled by a trajectory generator with the set acceleration. The controlword gives the starting command.
- Cyclic-synchronous position control (chapter 5.12): The target position is cyclically sent via the bus and accessed by the device with the set speed and acceleration.
- Cyclic-synchronous closed-loop velocity control (chapter 5.13): The device takes over the function of a velocity controller with cyclical speed set points via the field bus.

Activation occurs pursuant to the specification DS402, Drives and Motion Profile of CAN in Automation e.V. According to this, the objects for the parameters, which are discussed in the subsequent chapters, are sorted as well.

5.1 Information on the Drive

5.1.1 Objects

Index	Object	Name	Type	Access	Mem.
6510_h	RECORD	*drive data*	-	ro/rw	Y

Table 37: Objects with information on the drive

5.1.1.1 Object 6510_h: *drive data*

This structure centralizes all data and parameters that are not specified in DS402.

Index	Name	Object				
6510_h	*drive data*	RECORD	Parameter for the controller			
Sub-Index	**Name**	**Type**	**Attr**	**Map**	**Default-Value**	**Description**
0	*number of elements*	unsigned8	ro	N	10	Number of elements
1	*power stage code*	unsigned16	ro	N		Output stage coding
2	*max heat sink temperature*	integer16	rw	N	80,0°C	Maximum heat sink temperature in 0,1°C
3	*heat sink temperature*	integer16	ro	Y		Current heat sink temperature in 0,1°C
4	*min DC link voltage*	unsigned16	rw	N	100V	Minimum direct current link voltage in V
5	*max DC link voltage*	unsigned16	rw	N	750V	Maximum direct current link voltage in V
6	*temperature warning level*	unsigned16	rw	N	75,0°C	Heat sink temperature warning level in 0,1°C
7	*error mask*	unsigned16	rw	N	0xFFFF	Error mask; with the respective bit set to 0, an error can be masked out.
8	*error flags*	unsigned16	ro	Y		Error bits, see Table 38
9	*controller mode*	unsigned16	rw	N	2	0 = voltage controlled 1 = velocity control 2 = position and velocity control 82_h = votage controlled or inverter mode with position control $8F_h$ = motor heating
10	*max PWM frequency*	unsigned16	rw	N	8	Maximum frequency of the PWM output in kHz. Range of values: 6..15

error mask and *error flags*

The function of the individual bits can be gathered from the following table. The error code is contained in the object *error code* and is transferred with the emergency message.

A bit set in *error flags* will lead to an error message and reaction of the respective bit is set in *error mask*. This does not apply to error of priority A, those can't be switched off.

Bit	Description	Prio.	Error code
0	Short circuit or accidental ground, respectively, motor side	A	2320_h
1	free	B	
2	Over voltage in the intermediate circuit, PMC	A	3210_h
3	Under voltage in the intermediate circuit	B	3220_h
4	Error in the shaft encoder (motor), sine/cosine	B	7305_h
5	Unit temperature (heat sink) too high	B	4210_h
6	Wrong parameter	B	6320_h
7	Over current at controller output (motor)	A	2310_h
8	Motor temperature too high	B	4310_h
9	No identification of output stage	B	6380_h
10	Velocity controller, maximum velocity exceeded	B	8400_h
11	Error at the shaft encoder (motor), data channel	B	7320_h
12	Watchdog reset	A	6010_h
13	Error 2. shaft encoder (blade encoder)	B	7321_h
14	Following error too large	B	8611_h
15	Software trap, firmware error	A	6080_h

Table 38: Meaning of the *error flags* (6510_h/7).

5.2 Parameter for Control Loop and Motor

These parameters are taken from the motor database of the service tool. They are calculated for every motor on the test stand at OAT.

The parameters described herein must only be changed by qualified specialized staff. Incorrectly set parameters can cause machine damage.

5.2.1 Objects

Index	Object	Name	Type	Access	Mem.
6402_h	VAR	*motor type*	unsigned16	rw	Y
6410_h	RECORD	*motor data*	-	rw	Y
6073_h	VAR	*max current*	unsigned16	rw	Y
6076_h	VAR	*rated current*	unsigned16	rw	Y
$60F6_h$	RECORD	*torque control parameter set*	-	rw	Y
6078_h	VAR	*current actual value*	integer16	ro	N
6079_h	VAR	*DC link circuit voltage*	unsigned32	ro	N
5079_h	VAR	*DC link circuit voltage 16bit*	unsigned16	ro	N

Table 39: Objects for control loop and motor

5.2.1.1 Object 6402_h: *motor type*

The type of motor is determined here. The PMC can drive asynchronous motors, synchronous servo motors and direct current motors.

Index	Name	Type	Attr	Map	Default-Value	Description
6402_h	*motor type*	unsigned16	rw	N	0	Motor type: 0007_h: Asynchronous (cage rotor) $000A_h$: Synchronous servo $000D_h$: DC Permanent magnet $000E_h$: DC line circuit $000F_h$: DC shunt circuit 0010_h: DC dual circuit

5.2.1.2 Object 6410_h: *motor data*

All manufacturer specific parameters for the motor and the motor control can be found here. All parameters are taken from the motor database.

Index	Name	Object				
6410_h	*motor data*	RECORD	Manufacturer specific motor parameters			
Sub-Index	**Name**	**Type**	**Attr**	**Map**	**Default-Value**	**Description**
0	*number of elements*	unsigned8	ro	N	12	Number of elements
1	*poles*	unsigned16	rw	N	1	Number of pole pairs
2	*rho offset*	unsigned16	rw	N		Installation offset shaft encoder in increments
3	*nominal voltage*	unsigned16	rw	N	400,0V	Nominal voltage in 0,1V
4	*nominal speed*	unsigned16	rw	N	2920	Nominal rotation speed in RPM
5	*exciting voltage*	unsigned16	rw	N	200,0V	Exciting voltage for dc motor in 0,1V
6	*nominal frequency*	unsigned16	rw	N	50,0Hz	Rated frequency of the motor in 0,1Hz
7	*max motor temperature*	unsigned16	rw	N	150,0℃	Maximum motor temperature in 0,1℃
8	*motor temperature warning*	unsigned16	rw	N	130,0℃	Warning level motor temperature in 0,1℃
9	*motor temperature*	integer16	ro	Y		Current motor temperature
10	*cosinus phi*	unsigned16	rw	N	89	Power factor motor
11	*start voltage*	unsigned16	rw	N		Start voltage for F/U operation
12	*IxR factor*	unsigned16	rw	N		IxR factor for F/U operation
13	*heating current*	unsigned16	rw	N	10,0%	Heating current in 0,1% of the nominal current
14	*reserve1*	unsigned16	rw	N		
15	*reserve2*	unsigned16	rw	N		

5.2.1.3 Object 6073_h: *max current*

The maximum current of the controller is given in (0,1%) of the nominal current (*rated current*, object 6076_h). It usually is at double nominal current. It may not be set higher, but can be limited by setting a smaller value.

Index	Name	Type	Attr	Map	Default-Value	Description
6073_h	*max current*	unsigned16	rw	N	200,0%	Maximum motor current in 0,1%

5.2.1.4 Object 6076_h: *rated current*

The motor's nominal current is specified in mA, pursuant to DS402. The value can be taken from the motors type plate.

Index	Name	Type	Attr	Map	Default-Value	Description
6076_h	*rated current*	unsigned32	rw	N	17000	Nominal current motor in mA

5.2.1.5 Object $60F6_h$: *torque control parameter set*

This object explains the manufacturer specific **parameters for the control loop**. The control loop is the internal control loop of the cascaded controller of the PMC, it is a so called PI controller. The P factor is set in the object *torque control parameter set.gain* ($60F6_h$/1) and the I factor is set in the object *torque control parameter set.integration time* ($60F6_h$/2). OAT calculates these parameters for every motor and the parameters are taken from the motor data base.

Index	Name	Object				
$60F6_h$	*torque control parameter set*	RECORD	Control loop parameter			
Sub-Index	**Name**	**Type**	**Attr**	**Map**	**Default-Value**	**Description**
0	*number of elements*	unsigned8	ro	N	3	Number of elements
1	*gain*	unsigned16	rw	N	16	Amplification factor of the control loop.
2	*integration time constant*	unsigned16	rw	N	512	Integral factor of the control loop
3	*peak current time limit*	unsigned16	rw	N	5000ms	Time limit for surge current in ms, after this time nominal current limit applies.

5.2.1.6 Object 6077_h: *torque actual value*

The actual motor torque is specified in 0,1% of the nominal torque. The value is determined from the internal nominal value for the current.

Index	Name	Type	Attr	Map	Default-Value	Description
6077_h	*torque actual value*	integer16	ro	Y		Actual torque in 0,1%

5.2.1.7 Object 6078$_h$: *current actual value*

The actual motor current is specified in 0,1% of the nominal current (*rated current*). It is the actually measured current.

Index	Name	Type	Attr	Map	Default-Value	Description
6078$_h$	*current actual value*	integer16	ro	Y		Actual motor current in 0,1%

5.2.1.8 Object 6079$_h$: *DC link circuit voltage*

The direct current link voltage is specified in mV, pursuant to DS402. It measures the current at the clamps X8.

Index	Name	Type	Attr	Map	Default-Value	Description
6079$_h$	*DC link circuit voltage*	unsigned32	ro	Y		Actual direct current link voltage in **mV**
5079$_h$	*DC link circuit voltage 16bit*	unsigned16	ro	Y		Like above, 16 bit access, but **unit in V**

5.3 Parameters of the Velocity Controller

The parameters for the velocity controller are taken from the motor database of the service tool. They are calculated for every motor on the test stand at OAT.

The parameters described herein must only be changed by qualified specialized staff. Incorrectly set parameters can cause machine damage.

5.3.1 Objects

Index	Object	Name	Type	Access	Mem.
6044$_h$	VAR	*actual motor speed*	integer16	ro	N
606C$_h$	VAR	*actual velocity*	integer32	ro	N
506C$_h$	VAR	*actual velocity 16bit*	integer16	ro	N
6080$_h$	VAR	*max motor speed*	unsigned16	rw	Y
60F9$_h$	RECORD	*velocity control parameter set*	-	rw	Y

Table 40: Velocity controller objects

5.3.1.1 Object 6044$_h$: *actual motor speed*

Index	Name	Type	Attr	Map	Default-Value	Description
6044$_h$	*actual motor speed*	integer16	ro	Y		Current rotation speed of the motor in revolutions per minute (rpm)

5.3.1.2 Object 606C$_h$: *actual velocity*

This controller calculates the velocity of the axis on the basis of conversion factors and writes them into this object.

Index	Name	Type	Attr	Map	Default-Value	Description
606C$_h$	*actual velocity*	integer32	ro	Y		Actual velocity of the axis in position units/s
506C$_h$	*actual velocity 16bit*	integer16	ro	Y		Like above, 16 bit access

5.3.1.3 Object 6080$_h$: *max motor speed*

The initial nominal value of the velocity controller is limited to this value.

Index	Name	Type	Attr	Map	Default-Value	Description
6080$_h$	*max motor speed*	unsigned32	rw	N	3000	Maximum motor speed in revolutions per minute (rpm)

5.3.1.4 Object 60F9$_h$: *velocity control parameter set*

The velocity controller of the PMC is a so called PID controller within a cascade structure. The P ratio is determined by the object *velocity parameter set.gain* (60F9$_h$/1), the I ratio by the object *velocity parameter set.integration time constant* (60F9$_h$/2) and the D ratio by the object *velocity parameter set.differential time constant* (60F9$_h$/3).

All parameters are taken from the motor database. Where necessary, the setting of the velocity controller must be aligned to the machine.

Parameters of the Velocity Controller

Index	Name	Object				
60F9h	*velocity control parameter set*	RECORD	Velocity controller parameters			
Sub-Index	**Name**	**Type**	**Attr**	**Map**	**Default-Value**	**Description**
0	*number of elements*	unsigned8	ro	N	6	Number of elements
1	*gain*	unsigned16	rw	N	100	Proportional amplification of the velocity controller
2	*integration time constant*	unsigned16	rw	N	200	Integral ratio of the velocity controller. The initial value for optimization of the velocity controller should be set to 1/10 of the value in *gain.*
3	*differential time constant*	unsigned16	rw	N	0	Differential ratio of the velocity controller. Normally, this value is set 0. In specific individual cases the differential ratio can steady a natural frequency of the machine.
4	*brake set time*	unsigned16	rw	N	100	Brake dwell time in ms
5	*filter time constant*	unsigned16	rw	N	1	Low pass filter rotation time: $2^{\mathit{filter\ time\ constant}}$ *ms*
6	*min frequency*	unsigned16	rw		200	Minimum frequency for F/U operation in 0,01Hz

The Object *brake set time* contains the brake dwell time. That means, the time between the opening of the brake contact and the moment of the motor current shut down. It must conform to the shut down delay time of the brake lift coil and ensures that suspended axes don't sag. Without holding brake, 0 should be set; with holding brake 50ms should be set, because the incursion time of holding brakes is about 50ms.

The drive system should be slowed down at first. If the axis shall be shut down, it is set rotation speed actual value = 0 and at the same time, the brake contact is opened and after the brake dwell time, which begins to run from that moment on, the drive system is shut down („*disabled*“).

A time for lifting the brakes at start up does not have to be set. For this reason, the axis must be brought into the adjustment setting (“enable”) and only operate afterwards, so the brake has enough time to lift.

5.4 Position Controller Parameters

The PMC's position controller is a proportional controller (P controller). It controls how fast and accurate the axis is positioned. The following objects determine the action of the position controller and give information on the positions.

5.4.1 Objects

Index	Object	Name	Type	Access	Mem.
6064_h	VAR	*position actual value*	integer32	ro	N
5064_h	VAR	*position actual value 16bit*	integer16	ro	N
6065_h	VAR	*following error window*	unsigned32	rw	Y
6066_h	VAR	*following error timeout*	unsigned16	rw	Y
6067_h	VAR	*position window*	unsigned32	rw	Y
6068_h	VAR	*position window time*	unsigned16	rw	Y
$606A_h$	VAR	*sensor selection code*	integer16	rw	Y
$60F4_h$	VAR	*actual following error*	integer32	ro	N
$50F4_h$	VAR	*actual following error 16bit*	integer16	ro	N
$60FB_h$	RECORD	*position control parameter set*	-	rw	Y

Table 41: Position controller objects

5.4.1.1 Object 6064_h: *position actual value*

The controller calculates the actual position of the axis according to conversion factors and writes them into this object.

Index	Name	Type	Attr	Map	Default-Value	Description
6064_h	*position actual value*	integer32	ro	Y		Actual position of the axis in position units
5064_h	*position actual value 16bit*	integer16	ro	Y		Like above, 16 bit access

The identification of the actual position depends on the object *sensor selection code* ($606A_h$, Page 59).

sensor selection code = 0: Actual position is identified via motor feedback (motor encoder). See chapter 5.5, page 60.

sensor selection code = -1: Actual position is identified by the additional sensor. See chapter 5.6, page 65.

5.4.1.2 Object 6065_h: *following error window*

Index	Name	Type	Attr	Map	Default-Value	Description
6065_h	*following error window*	unsigned32	rw	N	100	Maximum allowable following error for closed-loop position controlled positioning in position units.

An error message is activated (see chapter 3.2), if the difference between reference and nominal value of the axis exceeds this value for the time specified in *following error timeout* (6066_h).

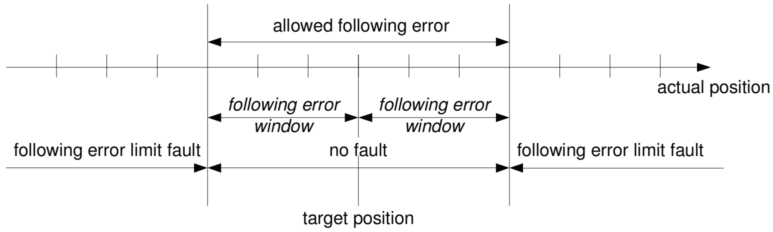

Figure. 5: Definition of *following error window*

The monitoring can be shut down by the value 4294967295 (=FFFFFFFF$_h$) . The reaction to this error can be set in *fault reaction option code* (605E$_h$).

The theoretical maximum following error can be calculated thus: $s_{max} = \frac{V_{max} \cdot 1000}{p\_gain}$

with V_{max} = maximum speed,
s_{max} = maximum following error,
p gain = proportional amplification of the position control, object 60FB$_h$/1,

Due to the control path, the actual maximum following error can be greater, which has to be considered in order to avoid false alarms when setting this parameter.

5.4.1.3 Object 6066$_h$: *following error timeout*

Index	Name	Type	Attr	Map	Default-Value	Description
6066$_h$	*following error timeout*	unsigned16	rw	N	10ms	Time frame for maximum following error in ms.

If the maximum following error overruns the time specified in *following error timeout,* an error message is activated. Quod vide object 6065$_h$, *following error window*.

5.4.1.4 Object 6067$_h$: *position window*

Index	Name	Type	Attr	Map	Default-Value	Description
6067$_h$	*position window*	unsigned32	rw	N	10	Position window of the position control in position units.

The *position window* defines a symmetrical range of positions in relation to the *target position*, the control window.

5.4.1.5 Object 6068$_h$: *position window time*

Index	Name	Type	Attr	Map	Default-Value	Description
6068$_h$	*position window time*	unsigned16	rw	N	10ms	Time frame for target position area in ms.

As soon as the axis arrived at the control window, the axis signals that it has reached its target (bit 10 "*target reached*" is set in *statusword*). The control accuracy is not influenced. The control window is considered to be reached when the nominal value of the axis is in the control window of the *position window time*. In case of minimum time Zero, “target reached” will be reported even if the axis overreached the target (overshoot].

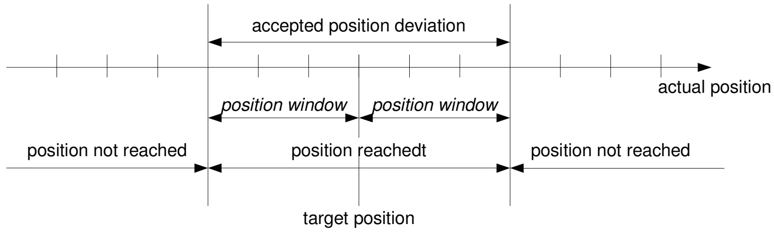

Figure. 6: Definition of *position window*

5.4.1.6 Object 606A$_h$: *sensor selection code*

Index	Name	Type	Attr	Map	Default-Value	Description
606A$_h$	*sensor selection code*	integer16	rw	N	0	Selection of position sensor: 0 = motor encoder, jack X5 -1 = 2. sensor, jack X3

The identification of the nominal value of the axis can either occur through the shaft encoder attached to the motor or through an external shaft encoder. A switch between the two sensors can only take place of the axis is idle.

5.4.1.7 Object 60F4$_h$: *actual following error*

Index	Name	Type	Attr	Map	Default-Value	Description
60F4$_h$	*actual following error*	integer32	ro	Y		Actual deviation between target value and actual position (following error) in position units.
50F4$_h$	*actual following error 16bit*	integer16	ro	Y		Like above, 16 bit access

Quod vide objects 6065$_h$ and 6066$_h$.

5.4.1.8 Object 60FB$_h$: *position control parameter set*

This object contains all manufacturer specific parameters of the position control.

Index	Name	Object				
60FB$_h$	*position control parameter set*	RECORD	Position control parameters			
Sub-Index	**Name**	**Type**	**Attr**	**Map**	**Default-Value**	**Description**
0	*number of elements*	unsigned8	ro	N	4	Number of elements
1	*p gain*	unsigned16	rw	N	17	Proportional amplification ratio of the position control in 1/s
2	*precontrol*	unsigned16	rw	N	100,0%	Speed pre-control in 0,1%
3	*control flags*	unsigned16	rw	N	1	Settings for the position control
4	*reserve*	unsigned16	rw	N	0	

p gain

The position control works with the following equations:

$$V = p\_gain \cdot s \qquad\qquad s = \frac{V}{p\_gain}$$

with V = speed in position unit/s,
s = following error in position unit.

precontrol

With this parameter, the effect of the pre-control can be reduced. It is specified in per mil of the automatically calculated pre-control.

The theoretical following error is calculated by:

$$s = \frac{V}{p\_gain} \cdot \left(1 - \frac{precontrol}{1000}\right)$$

with V = speed in position unit/s,
s = following error in position unit.

control flags

The behavior of the motor can be set via the *control flags*.

Bit		**+hex**
0	1 = Enable autostart, state machine can immediately switch to „Switched On“ without command “shut down” after power on.	0001_h
1..7	Not yet used	
8	1 = Emergency Feather Command (EFC): 90°-Limit switch must be engaged in feather position, otherwise the axis starts again and moves to the limit switch	0100_h
9	1 = EFC moves with velocity control to the 90°-Limit switch (or Quickstop+)	0200_h
10..14	Not yet used	
14	1 = axes can move beyond the limit switches if the device is not in state Operational	4000_h
15	1 = Simulation mode, axes are not moved	8000_h

Table 42: Meaning of the *control flags* ($60FB_h$/3).

5.5 Motor Encoder Settings

A high resolution sine-cosine-encoder (single or multi turn) with Endat or Hyperface interfaces is employed for position and speed detection.

The conversion of the physical values (e.g. encoder increments) into internal values is defined by another group of parameters that are designed as fraction in order to get a conversion as exact as possible. The relation between encoder increments and position values can, of course, only be attributed a single factor, in which the user has to allow for all influencing values like gearing ration, increment per revolution etc. The system provides parameters for every conversion, so that the user can enter values like sensor increments per motor revolution or the gear ration separately and the system does the conversion.

The following figure gives a schematic overview of the usage of the individual conversion factors.

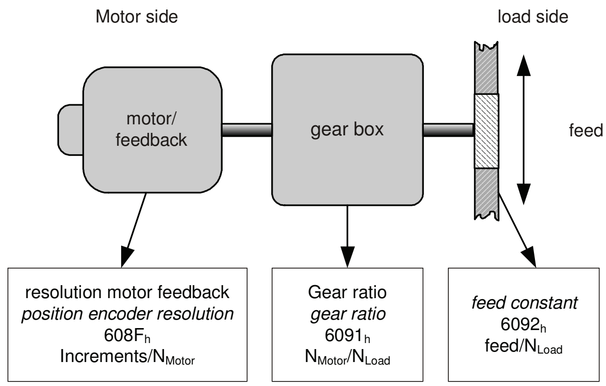

Figure. 7: Position detection with motor feed back

Settings for position detection with motor feedback:

1. Setting "position control motor encoder" (see chapter 5.4.1.6):
   sensor selection code ($606A_h$) = 0;
2. Increments of the motor encoder per number of revolutions:
   position encoder resolution.encoder increments ($608F_h$/1): = 512, 1024 or 2048
   The value can be found on the type plate of the encoder
   position encoder resolution.motor revolutions ($608F_h$/2) = 1
3. Gear ratio between motor side and load side:
   gear ratio.motor revolutions (6091_h/1) = Number of motor revolutions
   gear ratio.shaft revolutions (6091_h/2) = Number of load shaft revolutions
   In case of gears connected in series, the conversions are multiplied.
4. Enter feed load side per number of revolutions of the load shaft (e.g. spindle pitch):
   feed constant.feed (6092_h/1) = Way
   feed constant.shaft revolutions (6092_h/2) = Load shaft number of revolutions
   1.

The formula to convert internal units into position units is:

$$\frac{position\_actual\_value}{position\_internal\_value} = \frac{feed\_constant}{position\_encoder\_resolution \times gear\_ratio}$$

The individual factors specified as fraction.

Pitch system example:

Encoder increments = 1024/revolutions
Gear factor = 183
Number teeth tooth rim = 111
Number teeth bevel = 12
Position unit = 0,01 °

position encoder resolution.encoder increments = 1024
position encoder resolution.motor revolutions = 1

gear ratio.motor revolutions = 183 x 111 = 20313
gear ratio.shaft revolutions = 12

feed constant.feed = 36000 [unit 0,01 °]
feed constant.shaft revolutions = 1

5.5.1 Motor Encoder Parameters

Index	Object	Name	Type	Access	Mem.
$607C_h$	VAR	*home offset*	unsigned32	rw	Y
$608F_h$	ARRAY	*position encoder resolution*	unsigned32	rw	Y
6091_h	ARRAY	*gear ratio*	unsigned32	rw	Y
6092_h	ARRAY	*feed constant*	unsigned32	rw	Y
$607E_h$	VAR	*polarity*	unsigned8	rw	Y
5094_h	VAR	*encoder code*	unsigned16	rw	Y

Table 43: Objects for position detection with motor encoder

5.5.1.1 Object $607C_h$: *home offset*

Index	Name	Type	Attr	Map	Default-Value	Description
$607C_h$	*home offset*	Integer32	rw	N	0	Zero point shift in position units

The object *home offset* is the difference between the zero position for the application (*position actual value*) and the referential position of the machine (encoder value = 0). The *home offset* is added to this position and thereby the actual position for the application is set. All absolute movements are in relation to the new zero position.

This object can only be changed in the *Homing Mode*. It is changed by the automatic zero position setting at the device.

This object only applies to the position detection via primary encoders (motor feed back). If the position detection occurs via an additional encoder, object $687C_h$ is applied. Quod vide *sensor selection code* in chapter 5.4.1.6.

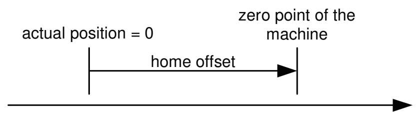

Figure. 8: *home offset* definition

Actual position = zero position + *home offset*.

5.5.1.2 Object $608F_h$: *position encoder resolution*

The relation of encoder increments per motor revolution is set in *position encoder resolution.*

$$position\_encoder\_resolution = \frac{encoder\_incremets}{motor\_revolutions}$$

Index	Name	Object				
$608F_h$	*position encoder resolution*	ARRAY	Encoder resolution			
Sub-Index	**Name**	**Type**	**Attr**	**Map**	**Default-Value**	**Description**
0	*number of elements*	Unsigned8	ro	N	2	Number of elements
1	*encoder increments*	Unsigned32	rw	N	512	Bar number shaft encoder
2	*motor revolutions*	Unsigned32	rw	N	1	Number motor revolutions

This object applies to the primary travel sensor.

It depends on the type of the connected encoder which bar number has to be entered, see encoder type plate (bar number)
Common bar numbers are: 512, 1024, 2048

5.5.1.3 Object 6091_h: *gear ratio*

The object *gear ratio* is the gear factor, thus the relation between motor revolutions to drive shaft revolutions.

$$gear\_ratio = \frac{motor\_revolutions}{shaft\_revolutions}$$

Index	Name	Object				
6091_h	*gear ratio*	ARRAY	Gear ratio			
Sub-Index	**Name**	**Type**	**Attr**	**Map**	**Default-Value**	**Description**
0	*number of elements*	Unsigned8	ro	N	2	Number of elements
1	*motor revolutions*	Unsigned32	rw	N	1	Number motor revolutions, max. 32767
2	*shaft revolutions*	Unsigned32	rw	N	1	Number resolutions at gear shaft, max. 32767

If the position detection shall occur via an external encoder, the object *ae gear ratio,* 6891_h has to be used, as well.

5.5.1.4 Object 6092_h: *feed constant*

The object *feed constant* defines the relation of position units per drive shaft revolutions.

$$feed\_constant = \frac{feed[\text{position\_units}]}{shaft\_revolutions}$$

Index	Name	Object				
6092_h	*feed constant*	ARRAY	Feed per shaft revolution			
Sub-Index	**Name**	**Type**	**Attr**	**Map**	**Default-Value**	**Description**
0	*number of elements*	Unsigned8	ro	N	2	Number of elements
1	*feed*	Unsigned32	rw	N	36000	Travel, degree etc. in position units
2	*shaft revolutions*	Unsigned32	rw	N	1	Number of revolutions at the gear shaft, max. 32767

You can enter the desired travel per shaft revolutions here.

This object is only applied if *sensor selection code* = 0. In case of position detection with additional encoders, see *ae feed constant*, chapter 0.

Example:

Travel / shaft revolution = 1,25°:
feed constant = 125° / 100 revolutions
feed = 125
shaft revolutions = 100

5.5.1.5 Object $607E_h$: *polarity*

The reference and actual positions are multiplied with either factor 1 or -1, depending on the polarity bit in object *polarity*. This can determine the rotating direction of the axis.

Index	Name	Type	Attr	Map	Default-Value	Description
$607E_h$	*polarity*	unsigned8	rw	N	0	00_h: rotating direction positive $C0_h$: rotating direction negative

Bit 6 and 7 = 0: positive counting means that in case of increasing position values the axis turns clockwise (front view on motor shaft).

Bit 6 und 7 = 1: ($C0_h$) negative counting means that in case of increasing position values the axis turns counter clockwise (front view motor shaft).

Since the speed is also detected via the position encoder, bit 6 should always have the same value as bit 7 for compatibility reasons. The encoder does not evaluate bit 6, though.

Bit 0 to 5 are not used currently.

For position control via an external encoder, the counting direction must be considered in order to avoid a positive feedback of the position control. See objects *ae polarity*, $687E_h$ (chapter)

5.5.1.6 Object 5094_h: *encoder code*

Index	Name	Type	Attr	Map	Default-Value	Description
5094_h	*encoder code*	unsigned16	rw	N	0	Encoder settings, ENDAT: 0 HyperFace: 2

In this objects, special settings for the angle encoder can be specified, if they cannot be defined automatically.

5.6 Settings for the Additional Encoder (Blade Encoder)

The PMC offers the possibility to detect and control the position with an additional shaft encoder. If this is the case, the primary encoder (motor feedback) will only be used for speed detection. This operational mode can be selected though *sensor selection code* ($606A_h$) = -1 (see chapter 5.4.1.6).

The objects for the additional encoder comply with their motor encoder counterpart. They are prefixed with the code *ae* for *additional encoder.* The index is higher by the offset 800_h.

The position is available via the object *ae position actual value* (Object 6864_h) in any case.

With Asynchronic or DC motors the position control is also possible without primary encoder for the motor feedback. The control quality will then be less accurate due to higher gear tolerance per resolution.

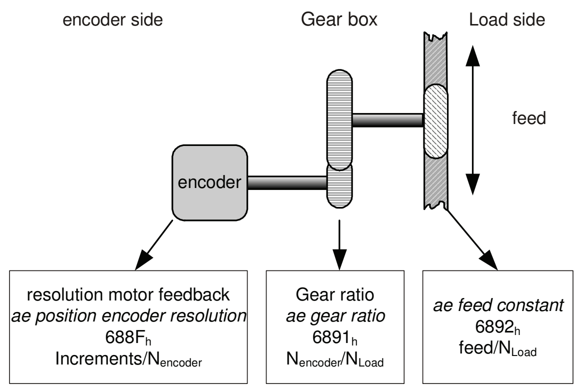

Figure. 9: Position detection with external encoder

Settings for position detection with additional encoder:

1. Encoder increments per number of revolutions:
   ae resolution.encoder increments ($688F_h$/1) = 512, 1024 or 2048;
   The value can be found at the encoder's type plate.
   ae resolution.encoder revolutions ($688F_h$/2) = 1;
2. Gear ratio between encoder and load side:
   ae gear ratio.encoder revolutions (6891_h/1) = Number of encoder revolutions;
   ae gear ratio.shaft revolutions (6891_h/2) = Number of revolutions of the load shaft;
3. Enter feed load side per number of revolutions of the load shaft (e.g. spindle pitch):
   ae feed constant.feed (6892_h/1) = Travel;
   ae feed constant.shaft revolutions (6892_h/2) = Number of revolutions of the load shaft.

Settings for position control on the additional encoder:

1. Set factors for motor feedback correctly, see chapter 5.5.
2. Set polarity according to fitting position: *ae polarity* (Object $687E_h$)
3. Set "position control to external encoder" (see chapter 5.4.1.6):
   sensor selection code ($606A_h$) = -1;

Examples for a pitch system:

Encoder increments = 1024/revolution
Number teeth gear rim = 111
Number teeth bevel = 15
Position unit = 0,01°

ae position encoder resolution.encoder increments = 1024
ae position encoder resolution.motor revolutions = 1

ae gear ratio.encoder revolutions = 111
ae gear ratio.shaft revolutions = 12

ae feed constant.feed = 36000 [value 0,01 °]
ae feed constant.shaft revolutions = 1

5.6.1 Parameters for the Additional Encoder

Index	Object	Name	Type	access	Mem.
6864_h	VAR	*ae position actual value*	integer32	ro	N
5864_h	VAR	*ae position actual value 16bit*	integer16	ro	N
$68F4_h$	VAR	*ae actual position difference*	integer32	ro	N
$58F4_h$	VAR	*ae actual position difference 16bit*	integer16	ro	N
$687C_h$	VAR	*ae home offset*	unsigned32	rw	Y
$688F_h$	ARRAY	*ae resolution*	unsigned32	rw	Y
6891_h	ARRAY	*ae gear ratio*	unsigned32	rw	Y
6892_h	ARRAY	*ae feed constant*	unsigned32	rw	Y
$687E_h$	VAR	*ae polarity*	unsigned8	rw	Y
5894_h	VAR	*ae encoder code*	unsigned16	rw	Y
6865h	VAR	*difference error window*	unsigned32	rw	J
6866h	VAR	*difference error time out*	unsigned16	rw	J

Table 44: Additional encoder objects.

5.6.1.1 Object 6864_h: *ae position actual value*

The controller calculates the actual position of the additional encoder on the basis of conversion factors and writes them into this object.

Index	Name	Type	Attr	Map	Default-Value	Description
6864_h	*ae position actual value*	integer32	ro	Y		Actual position of additional encoder in position units
5864_h	*ae position actual value 16bit*	integer16	ro	Y		Like above, 16 bit access

This always shows the position of the additional encoder, independent of the settings in *sensor selection code*.

5.6.1.2 Object $687C_h$: *ae home offset*

Index	Name	Type	Attr	Map	Default-Value	Description
$687C_h$	*ae home offset*	Integer32	rw	N	0	Zero point offset 2. encoder in position units

The object *ae home offset* is the difference between the zero position for the application (*ae position actual value*) and the reference position of the machine (encoder value = 0). The *ae home offset* is added to this position and thus the actual position for the application is set..

This object can only be modified in the *Homing Mode*. It is modified by the device's automatic zero point setting.

This object is only valid for the position detection via the external encoder.

Actual position = Zero point + *ae home offset*.

5.6.1.3 Object $688F_h$: *ae resolution*

In *ae resolution (additional encoder resolution)* the relation between encoder increments per encoder revolutions is specified.

$$ae\_resolution = \frac{encoder\_incremets}{encoder\_revolutions}$$

Index	Name	Object				
$688F_h$	*ae position encoder resolution*	ARRAY	Resolution 2. encoder			
Sub-Index	**Name**	**Type**	**Attr**	**Map**	**Default-Value**	**Description**
0	*number of elements*	Unsigned8	ro	N	2	Number of elements
1	*encoder increments*	Unsigned32	rw	N	512	Shaft encoder bar number
2	*encoder revolutions*	Unsigned32	rw	N	1	Shaft encoder number of revolu-tions

In *encoder revolutions* 1 is set for standard encoder, since those always provide an integer number increments per revolution. In case of linear position measuring systems it might be necessary to set a value greater than 1.

It depends on the type of the connected encoder, which bar number must be entered, see encoder type plate (bar number)
Common bar numbers are: 512, 1024, 2048

5.6.1.4 Object 6891_h: *ae gear ratio*

The object *ae gear ratio* is the gear factor of a gear between encoder and load side, in other words the relation between encoder revolutions to drive shaft revolutions.

$$ae\_gear\_ratio = \frac{encoder\_revolutions}{shaft\_revolutions}$$

Index	Name	Object				
6891_h	*ae gear ratio*	ARRAY				
Sub-Index	**Name**	**Type**	**Attr**	**Map**	**Default-Value**	**Description**
0	*number of elements*	Unsigned8	ro	N	2	Number of elements
1	*encoder revolutions*	Unsigned32	rw	N	1	Number of revolutions shaft en-coder
2	*shaft revolutions*	Unsigned32	rw	N	1	Number of revolutions at the gear output

5.6.1.5 Object 6892$_h$: *ae feed constant*

The object *ae feed constant* defines the relation of position units per drive shaft revolutions at position control at the external encoder.

$$ae\_feed\_constant = \frac{feed[\text{position\_units}]}{shaft\_revolutions}$$

Index	**Name**	**Object**				
6892$_h$	*ae feed constant*	ARRAY				
Sub-Index	**Name**	**Type**	**Attr**	**Map**	**Default-Value**	**Description**
0	*number of elements*	Unsigned8	ro	N	2	Number of elements
1	*feed*	Unsigned32	rw	N	36000	Travel, degree etc. in position units
2	*shaft revolutions*	Unsigned32	rw	N	1	Number of revolutions at gear output

The desired travel per drive shaft is entered here. The more revolutions are specified, the higher the degree of accuracy of the conversion.

5.6.1.6 Object 687E$_h$: *ae polarity*

The actual position of the additional encoder is multiplied by the factor 1 or -1, depending on the polarity bit in the object *ae polarity*. The counting direction of the encoder can be adjusted to the fitting position.

Index	**Name**	**Type**	**Attr**	**Map**	**Default-Value**	**Description**
687E$_h$	*ae polarity*	unsigned8	rw	N	0	00$_h$: Rotating direction positive C0$_h$: Rotating direction negative

Bit 6 and 7 = 0: positive counting means that in case of increasing position values the encoder turns clockwise (front view on encoder shaft).

Bit 6 und 7 = 1: (C0$_h$) negative counting means that in case of increasing position values the encoder turns counter clockwise (front view encoder shaft).

Since the speed is also detected via the position encoder, bit 6 should always have the same value as bit 7 for compatibility reasons. The encoder does not evaluate bit 6, though.

Bit 0 to 5 are not used currently.

5.6.1.7 Object 5894_h: *ae encoder code*

Index	Name	Type	Attr	Map	Default-Value	Description
5894_h	*ae encoder code*	unsigned16	rw	N	0	Settings to the additional encoder, ENDAT: 0 SSI: 3 Simulation: FF_h

In this object, special settings for the angle encoder can be specified, if they cannot be defined automatically. With code FF_h the position of the main encoder is copied into the position of the additional encoder.

5.6.1.8 Object $68F4_h$: *ae actual position difference*

Index	Name	Type	Attr	Map	Default-Value	Description
$68F4_h$	*ae actual position difference*	integer32	ro	J	0	Difference between motor encoder and blade encoder in position units
$58F4_h$	*ae actual position difference 16bit*	integer16	ro	J	0	Like above, 16 bit access

The value is calculated synchronous with the internal sampling cycle as follows:

ae actual position difference = position actual value – ae.position actual value

5.6.1.9 Object 6865_h: *difference error window*

Index	Name	Type	Attr	Map	Default-Value	Description
6865_h	*difference error window*	unsigned32	rw	N	0	Maximal allowed difference between the positions of motor encoder and blade encoder.

Exceeds the difference between motor encoder and blade encoder this value for more than the time given in Object *difference error time out* (6866_h), the error 8620_h is set off.

5.6.1.10 Object 6866_h: *difference error time out*

Index	Name	Type	Attr	Map	Default-Value	Description
6866_h	*difference error time out*	unsigned16	rw	N	0	Maximal allowed time, in which the maximum position difference between motor and blade encoderc is exceeded.

Exceeds the difference in *ae actual position difference* the value in *difference error window* for more than this time, the error 8620_h is set off.

5.7 Error Reaction Settings

The controller reactions can be set for the following errors:

- Field bus communication termination

- Occurrence of an axis error

5.7.1 Objects

Index	Object	Name	Type	Access	Mem.
6007_h	VAR	*abort connection option code*	integer16	rw	Y
$605E_h$	VAR	*fault reaction option code*	integer16	rw	Y

Table 45: Objects for error reaction settings

5.7.1.1 Object 6007_h: *abort connection option code*

This object defines the reaction of the system in case of an interrupted connection to the host.

Index	Name	Type	Attr	Map	Default-Value	Description
6007_h	*abort connection option code*	integer16	rw	N	1	Error reaction in case of communication error on the field bus

Value	DS402	Meaning
0	*No Action*	No action
1	*Malfunction*	Error message only
2	*Disable Voltage*	Turn off axis
3	*Quickstop*	Quick stop axis

Table 46: Error reaction in case of connection interrupt

In the cases 0 and 1, the axis follows through its last task. In case of RPM regulated drive, it drives up to the limit switch. Therefore, these settings may only be used if no damage can occur.

In case of setting 2, the output stage of the axis is turned off.

In case of setting 3, a QUICKSTOP will be executed, that means the axis decelerates at the current limit.

5.7.1.2 Object $605E_h$: *fault reaction option code*

This object specifies the axis' reaction to an error. The settings only have an effect in case of target value-, following error- and encoder faults.

Index	Name	Type	Attr	Map	Default-Value	Description
$605E_h$	*fault reaction option code*	integer16	rw	Y	2	Setting of error reaction to an axis malfunction

Value	Meaning
0	Turn axis off, axis can rotate freely
1	Decelerate at the deceleration chute (*profile deceleration*, 6084_h)
2	Decelerate at quick stop chute (= current limit)
3	Decelerate at current limit
4	Not used
-1	Error message only

Table 47: Error reaction settings

5.7.1.3 Axis Error Reaction

If an emergency stop shall be executed (input EFC or bit EFC in controlword), the flag position is approached if possible.

With each error, the watch dog contact is opened. With suitable wiring of the security chain, an emergency drive (EFC drive) is executed.

Axis error	Error code	Reaction
Over current at encoder output (motor)	2310_h	Turn-off motor
Short circuit/accidental ground motor side	2320_h	Turn-off motor
Over voltage in the intermediate circuit, PMC	3210_h	Turn-off motor
Under voltage in the intermediate circuit	3220_h	Turn-off motor
Machine temperature (heat sink) too high	4210_h	After 30s alert: Turn-off motor
Motor temperature too high	4310_h	After 30s alert: Turn-off motor
Watchdog reset	6010_h	Turn-off motor (because of reset).
Software trap, firmware error	6080_h	Turn-off motor
Wrong parameter	6320_h	No axis command possible
No output stage detection	6380_h	No axis command possible
Error at the shaft encoder (motor), sine / cosine	7305_h	Reaction according to *fault reaction option code*
Error at the shaft encoder (motor), data channel	7320_h	Reaction according to *fault reaction option code*
Error 2. shaft encoder (blade encoder)	7321_h	Reaction according to *reaction option code*
Velocity controller, maximum RPM exceeded	8400_h	Turn-off motor
following error too large	8611_h	Reaction according to *fault reaction option code*

Table 48: Reactions to axis errors

5.8 Drive Control

The behavior of the drive control is defined though a finite automaton. 10 Shows the control of the finite automaton by the *Controlword* or local signals and event in the controller. The status is reported by the *Statusword*. Local signal are, e.g., the EFC signal, quick stop inputs etc. The finite automaton is also influenced by the error detection.

The finite automaton reports the status of the controller and the possible control sequence. A status represents an according internal or external reaction of the drive. The status further defines which command will be accepted. It is only possible, for example, to perform a movement of the controller's status is *operation enable*.

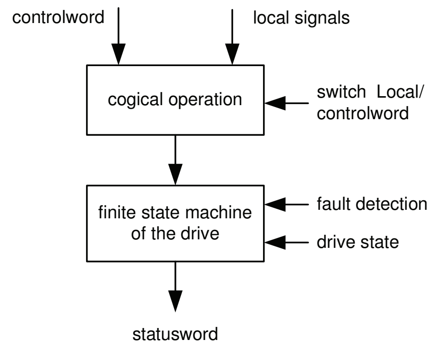

Figure. 10: Finite state automaton control

5.8.1 Finite Automaton of the Controller

11 Represents the finite automaton of the controller. The transitions are set by commands in the *Controlword* and internal events.

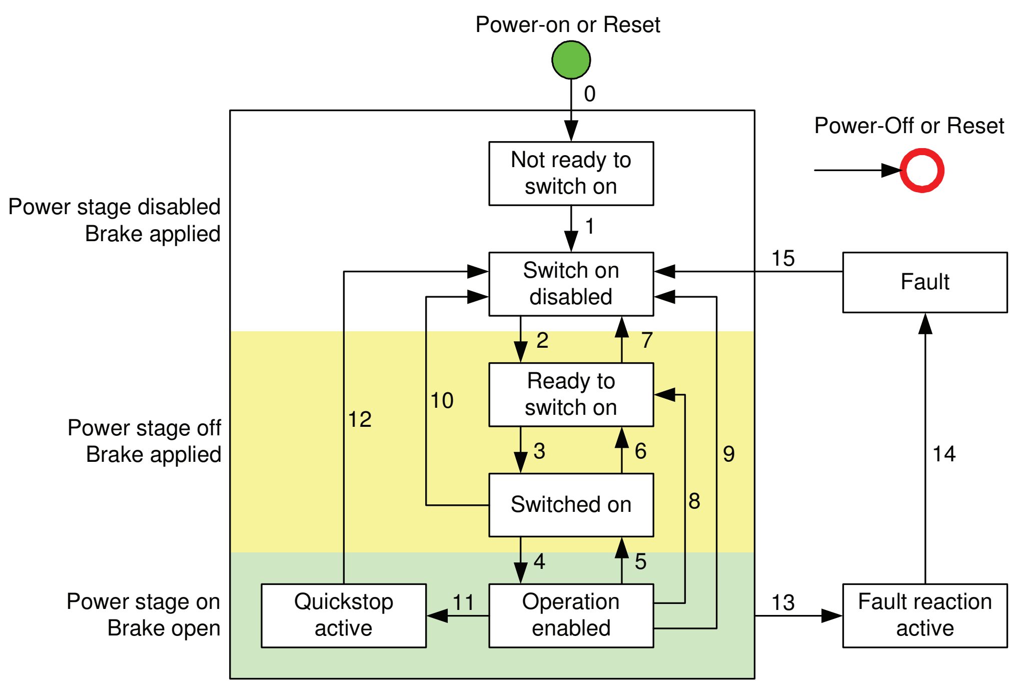

Figure 11: Finite automaton of the controller

The identification of states in the finite state machine are rendered possible by the *Statusword*:

Statusword	Condition	Meaning
xxxx xxxx x0xx 0000_b	Not ready to switch-on	Device actuates, initializing
xxxx xxxx x1xx 0000_b	Switch-on disabled	Device initialized, not enabled for switch-on (Enable input X4.5 off)
xxxx xxxx x01x 0001_b	Ready to Switch-on	Enable input active, output stage not switched on, brake engaged
xxxx xxxx x01x 0011_b	Switched-on	Command "*switch on*" received by *Controlword*, output stage switched on, brake lifted
xxxx xxxx x01x 0111_b	Operation enabled	Output stage switched on, brake lifted, axis can be moved
xxxx xxxx x00x 0111_b	Quick stop active	Quick stop pending, reaction processed
xxxx xxxx x0xx 1111_b	Fault reaction active	Error reaction in process
xxxx xxxx x0xx 1000_b	Fault	Error status, error reaction complete

Table 49: *Statusword* and finite state machine

Transition	Event	Action
0	Automatic transition after power-on or re-set	Device executes self test and initializes
1	Automatic transition	Activate communication Close watchdog contact
2	Enable input activated, Command „*shut-down*“ received	None
3	Command “*switch on*” received or EFC-Signal active	Switch-on output stage, lift brake, initialize internal target values
4	Command “*enable operation*” received or EFC-Signal active	Enable axis operation If EFC signal active, execute EFC drive
5	Command “*disable operation*” received, 90°-limit switch activated	Axis operation disabled, axis stops
6	Command “*shutdown*” received	Allow brake to engage, switch off control after brake dwell time. The axis coasts with-out the brake
7	Enable input deactivated, Command „*dis-able voltage*“ or „*quickstop*“ received	None
8	Command “*shutdown*” received	Allow brake to engage, switch off control after brake dwell time
9	Command “*disable voltage*” received or enable input deactivated	Allow brake to engage, switch off control after brake dwell time
10	Quick stop neg. (0°-limit switch) activated commands „*quickstop*“ or “*disable voltage*” received or enable input deactivated	Allow brake to engage, switch off control after brake dwell time
11	Quick stop neg. (0°- limit switch) activated or command „*quickstop*“ received	Execute quick stop function: brake at current limit, switch off control after brake dwell time
12	Automatic transition of quick stop finished	None
13	Error signal: internal axis error, communi-cation error, quick stop pos. (95°- limit switch)	Error reaction as preset in *abort connection option code* / *fault reaction option code*. Open watch dog contact
14	Automatic transition	Allow brake to engage, switch off control after brake dwell time
15	Command „Fault reset“ received and no error queued, EFC signal active	Close watch dog contact
16	EFC signal active and 95°- or 90°- limit switch not activated.	Switch-on output stage, lift brake, execute EFC drive if possible

Table 50: Status transition events and activities

Transition 3 and 4 can be connected jointly, the control does not need an intermediate step. The same is true for 5 and 6.

Special case EFC signal (*Emergency Feather Command***):** If this request is pending, the controller directly switches to „controller active“, if possible, and executes an EFC drive, that means driving to the 90°-Position. Exception: 90°- or 95°-limit switch are activated. An EFC drive is also executed, if another error is pending as well, insofar that it is possible in that case. In case of an encoder error, for example, it can be switched to the additional encoder or driven to the 90° limit switch with controlled speed (in case of ASM or DC motors).

If a status transfer is requested, the corresponding action is executed prior to giving out the new status.

5.8.2 Objects

Index	Object	Name	Type	Access	Mem.
6040_h	VAR	*controlword*	unsigned16	rw	N
6041_h	VAR	*statusword*	unsigned16	rw	N

5.8.2.1 Object 6040_h: *Controlword*

The drive functions are controlled by the *Controlword*.

Index	Name	Type	Attr	Map	Default-Value	Description
6040_h	*Controlword*	unsigned16	rw	Y		Drive control word

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ms				efc	r	oms	h	fr	oms			eo	qs	ev	so
MSB															LSB

Bit	Code	DS 402	Meaning
0	so	*Switch on*	1 = Output stage can be switched on 1->0 = Engage brake, switch-off output stage
1	ev	*Enable voltage*	1 = Output stage can be switched on 1->0 = Engage brake, switch-off output stage
2	qs	*Quickstop*	1 = normal operation possible 1->0: Stop axis as quickly as possible (current limit) 0 = Quick stop, no motion possible
3	eo	*Enable operation*	1 = Switch on output stage, lift brake 0 = Engage brake, switch off output stage
4	oms	*Operation mode specific*	Only in mode 1: *new set-point* 0->1 = New target position
5	oms	*Operation mode specific*	0
6	oms	*Operation mode specific*	0
7	fr	*Fault reset*	0->1: Reset error
8	h	*Halt*	1 = Stop all motion 0 = Continue or start requested motion, if possible 0->1: Stop axis with normal chute
9	oms	*Operation mode specific*	0
10	r	*Reserved*	0
11	efc	*emergency feather command*	0 = Normal mode of operation 1 = Complete EFC drive
12	ms	*Manufacturer specific*	0
13	ms	*Manufacturer specific*	0
14	ms	*Manufacturer specific*	is reflected by bit 14 in *statusword*
15	ms	*Manufacturer specific*	0

Table 51: Meaning of bits in *controlword* (6040_h)

The *Controlword* can be interpreted as a command. The following table show the possible combinations:

Command	**Bits in the *Controlword***							**Code (Hex)**	**Transition in finite automaton**
	Bit 8 halt	**Bit 7 fr**	**Bit 4 new**	**Bit 3 eo**	**Bit 2 qs**	**Bit 1 ev**	**Bit 0 so**		
Shutdown	X(1)	0	0	X(0)	1	1	0	0106_h	2, 6, 8
Switch on	X(1)	0	0	0	1	1	1	0107_h	3
Switch on + Enable operation	1	0	0	1	1	1	1	$010F_h$	3+4
Disable Voltage	X(1)	0	0	X(1)	X(1)	0	X	$010C_h$	7,9,10,12
Quickstop	X(1)	0	0	1	0	1	1	$010B_h$	7,10,11
Disable operation	1	0	0	0	1	1	1	0107_h	5
Enable operation	1	0	0	1	1	1	1	$010F_h$	4,16
Fault reset	X	0->1	0	X	X	X	X	$0X8X_h$	15
Start	1->0	0	0	1	1	1	1	$000F_h$	-
Start, new position	0	0	0->1	1	1	1	1	$001F_h$	(only Mode 1)
Stop	1	0	0	1	1	1	1	$010F_h$	-

Table 52: Command coding

The commands listen in light gray don't have to be executed by the device.

Bit 11 is manufacturer specific and activates an *Emergency Feather Command* (EFC). The code for this in the *Controlword* is $080F_h$.

5.8.2.2 Object 6041_h: *Statusword*

This object gives information on the status of the controller

Index	**Name**	**Type**	**Attr**	**Map**	**Default-Value**	**Description**
6041_h	*Statusword*	unsigned16	rw	Y		Status information of the drive

Meaning of the single bits:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ms		oms		ila	tr	rm	ms	w	sod	qs	ve	f	oe	so	rtso
MSB															LSB

Bit	Code	DS 402	Meaning
0	rtso	*Ready to switch on*	1 = Ready to switch on
1	so	*Switched on*	1 = Switched on, initialization finished
2	oe	*Operation enabled*	1 = Enable operation
3	f	*Fault*	1 = Error, fault reaction executed, error code in object $603F_h$
4	ve	*Voltage enabled*	1 = Intermediate circuit enabled
5	qs	*Quickstop*	0 = Quick stop executed
6	sod	*Switch on disabled*	1 = Switch on disabled, switch on not possible
7	w	*Warning*	1 = Warning, e.g. temperature warning, code in $603F_h$ No error reaction
8	ms	*Manufacturer specific*	0
9	rm	*Remote*	1 = Control of *Controlword* active, 0 = External signal control, e.g. EFC, *Controlword* is ignored
10	tr	*Target reached*	1 = Target value reached, see below
11	ila	*Internal Limit Active*	1 = Target position limit exceeded
12	oms	*Operation Mode Specific*	see Table 54
13	oms	*Operation Mode Specific*	see Table 54
14	ms	*Manufacturer Specific*	reflects state of bit 14 in *controlword*
15	ms	*Manufacturer Specific*	0

Table 53: Meaning of bits in *Statusword* (6041_h)

Bit 10, *Target reached* is set, if

- The position target value is reached in *profile position mode* (mode 1)
- The speed target value is reached in *profile velocity mode* (mode 3)
- Mode of operation is changed. The bit remains set in the *cyclic synchronous modes* (modes 8 and 9).
- Bit 8 (Stop) was set in *Controlword* and the axis stopped

The bit is reset if

- a new target value is transferred (modes 1 and 3).
- Bit 8 (Stop) was set in *Controlword* and the axis has not yet stopped (all modes).

Operational mode	Mode	Bit 12	Bit 13
profile position mode	1	*Setpoint Acknowledge*, 0 = Waiting for new target position 1 = Target value not yet transferred	*Following Error*, 1 = Following error too large
profile velocity mode	3	0 = Nominal speed not equal Zero 1 = Nominal speed is Zero	0
homing mode	6	0 = Zero point not set 1 = Zero point set	*Homing Error*, 1 = Error
cyclic synchronous position mode	8	0 = Target position ignored 1 = Target position controlled	*Following Error*, 1 = Following error too large
cyclic synchronous velocity mode	9	0 = Target speed ignored 1 = Target speed controlled	0

Table 54: Meaning of bits 12 and 13 in *Statusword*

5.9 Modes of Operation

The behavior of the PMC depends on the set operational mode. The control writes *modes of operation* into the object in order to select an operational mode. The object *modes of operation display* represents the actually set mode of operation.

Some bits in *Controlword* and *Statusword* have different meaning in the modes of operation.

Changing modes of operation can be effected by local signal as well. The EFC signal, for example, automatically switches to „*Profile position mode*". In this case, it is not possible to change the controls mode of operation.

The following operational modes are supported by the PMC

Profile position mode:	Positioning at target position with acceleration and deceleration chutes.
Profile velocity mode:	Setting of target speed with acceleration and deceleration chutes.
Cyclic synchronous position mode:	Position control according to cyclic position setting
Cyclic synchronous velocity mode:	Speed control according to cyclic velocity setting.
homing mode:	Set zero point
motor heating	motor will be heated by current without moving

The object *supported drive modes* lists which modes of operation are supported by the device.

5.9.1 Objects

Index	Object	Name	Type	Access	Mem.
6060_h	VAR	*mode of operation*	integer8	rw	N
6061_h	VAR	*mode of operation display*	integer8	ro	N
6502_h	VAR	*supported drive modes*	unsigned32	ro	N

5.9.1.1 Object 6060$_h$: *modes of operation*

This object requests the respective mode of operation for the controller. The actually set mode of operation is reported in object 6061$_h$.

The following table shows the supported modes of operation. Other modes of operation are not accepted and will not change the set mode of operation.

The operational mode can only be changed when the axis output stage is switched off (bit 3 in *controlword* = 0, bit 2 in *statusword* = 0)).

Mode	Operational mode	Description
0	*No mode*	No change / no operational mode active
1	*Profile position mode*	Positioning to target position with acceleration and deceleration chutes
3	*Profile velocity mode*	Setting of a target velocity with acceleration and deceleration chutes
6	*Homing mode*	Set zero point
8	*Cyclic synchronous position mode*	Position control according to cyclic position setting
9	*Cyclic synchronous velocity mode*	Speed control according to cyclic velocity setting
8F$_h$	*motor heating*	motor will be heated by current without moving

Table 55: Modes of operations supported by the PMC.

Index	Name	Type	Attr	Map	Default-Value	Description
6060$_h$	*modes of operation*	integer8	rw	Y	0	Modes of operation standards of the controller

5.9.1.2 Object 6061$_h$: *modes of operation display*

This object reports the activated mode of operation. Possible values are shown in Table 55.

A change in the operational mode can also take place internally, e.g. by an EFC signal.

Index	Name	Type	Attr	Map	Default-Value	Description
6061$_h$	*modes of operation display*	integer8	ro	Y	0	Set of controller mode of operation.

5.9.1.3 Object 6502$_h$: *supported drive modes*

This object shows which operation modes are supported by the device.

Index	Name	Type	Attr	Map	Default-Value	Description
6502$_h$	*supported drive modes*	unsigned32	ro	N	000001A5$_h$	Modes supported by the drive

Meaning of the individual bits in *supported drive modes* and PMC setting:

Bit	Def.	Description	Meaning
0	**1**	***Profile Position Mode***	Position control with operational profile calculation
1	0	*Velocity Mode*	Absolute rotation speed control
2	**1**	***Profile Velocity Mode***	Rotation speed control with profile calculation
3	0	*Profile Torque Mode*	Torque control
4	0	*reserved*	Reserved for DS402
5	**1**	***Homing Mode***	Referencing
6	0	*Interpolated Position Mode*	Interpolation
7	**1**	***Cyclic Synchronous Position Mode***	Cyclic setting of position target values
8	**1**	***Cyclic Synchronous Velocity Mode***	Cyclic setting of speed target values
9	0	*Cyclic Synchronous Torque Mode*	Cyclic setting of torque target values
10..15	0	*reserved*	Reserved for DS402
16..31	0	*reserved, Manufacturer specific*	Free, manufacturer specific operational modes

Table 56: Bits in object 6502_h, *supported drive modes*.

5.10 Position Controlled Positioning (Profile Position Mode)

A chute generator calculates the acceleration and deceleration chutes and the cyclic position target values for the position control. The following figure shows the structure.

The target values are limited by the respective parameters. This occurs at transfer of the target values from the field bus to the controller.

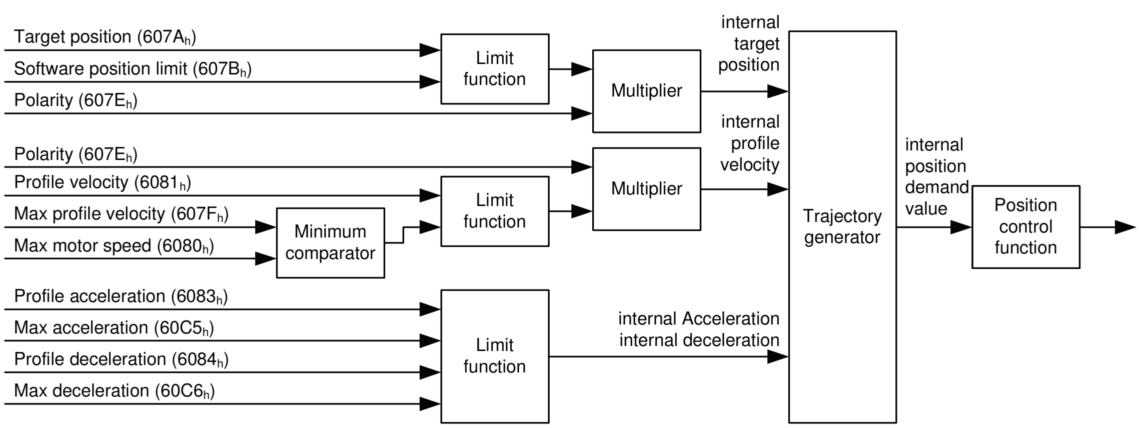

Figure 12: chute generator for *profile position mode*

The transmission of the target values is controlled through the timing of the bits *new set-point* in *controlword* and *set-point acknowledge* in *Statusword.*

After setting the new target values, the control signals the effective target value by setting of *new set-point*. The device verifies by setting *set-point acknowledge*. The control resets *new set-point* and the device deletes *set-point acknowledge*, as soon as it is ready to transfer a new target value.

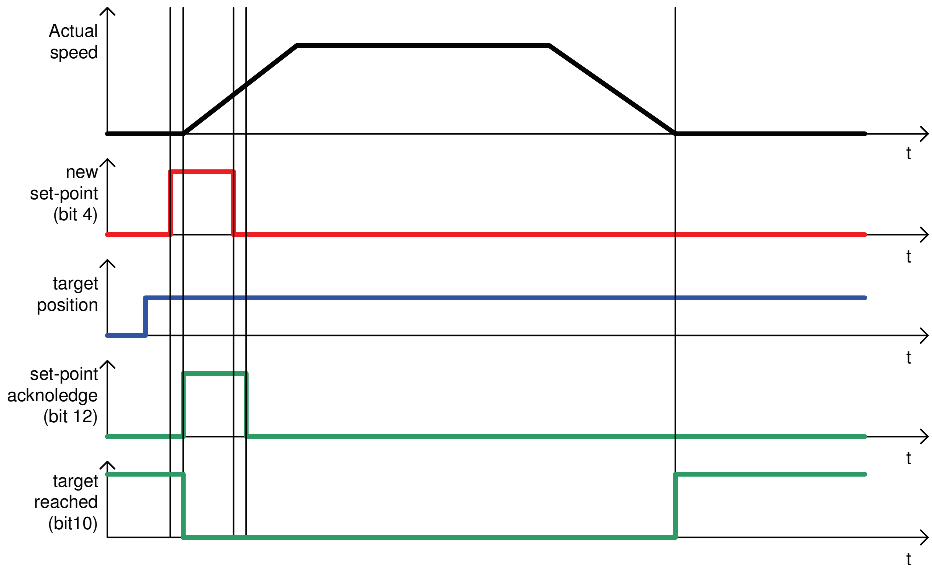

Figure 13: Target value handshake *new set-point*

If a new target value is transferred while the old one is still executed, the bit *set-point acknowledge* remains set until a new one can be transferred. The following figure clarifies the procedure.

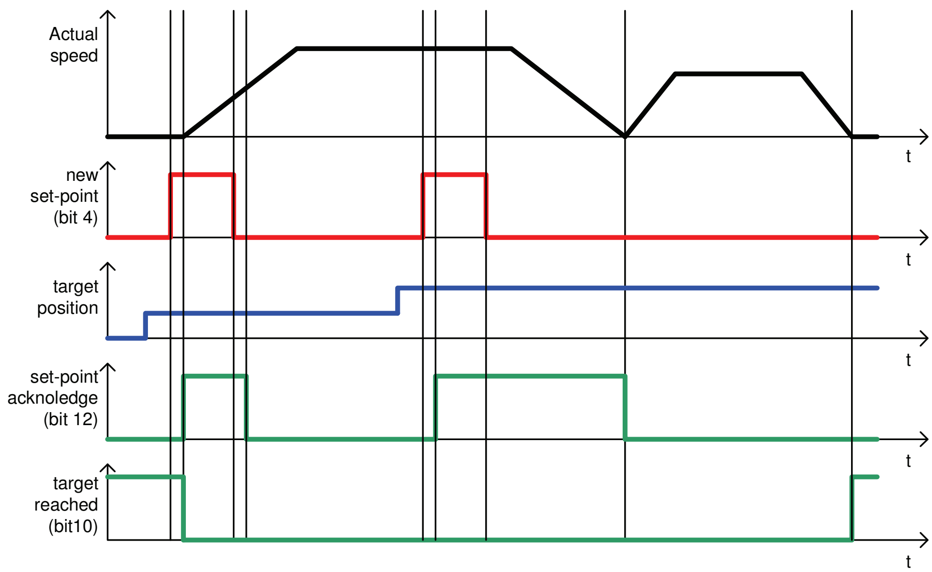

Figure 14: Handshake at target value transfer during drive

Bit 10 (*target reached*) is set after the final motion.

If bit 8 (*halt*) is still set, the procedure looks like this:

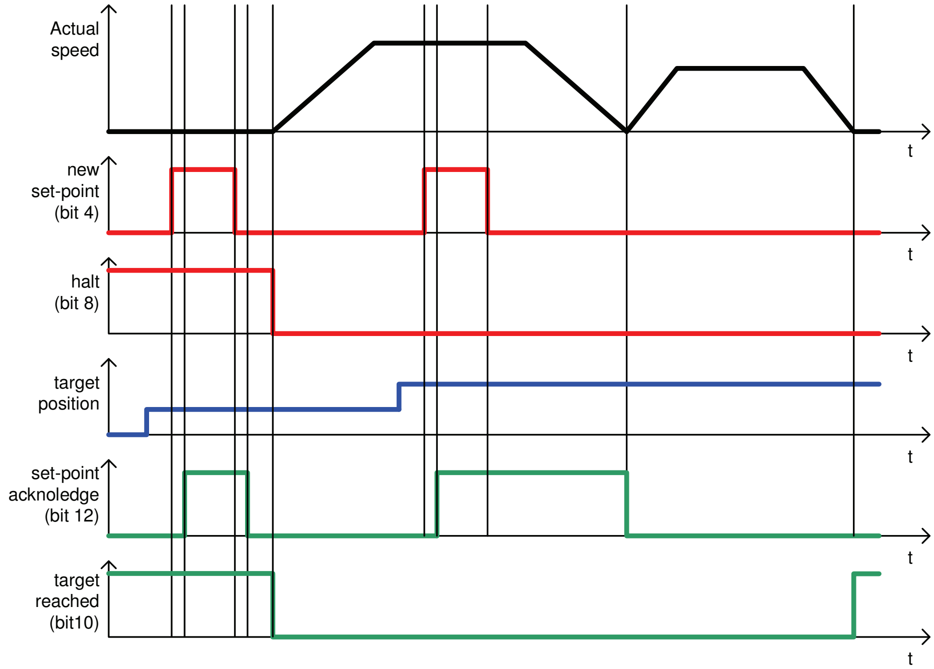

Figure 15: Axis start with bit 8 (*halt*)

5.10.1 Use of *Controlword* and *Statusword*

As shown in the last paragraph, bit 4 of the *Controlword* is used for the transfer of a new target value in the *profile position mode.* The other bits (5, 6 and 9) that rely on the mode of operation must always set 0. At the moment, the PMC neither supports relative positioning nor positioning to a new target position during the drive without stopping.

If no positioning is activated, the rising edge starts the motion from bit 4 (*new set-point*), as long as bit 8 (*halt*) is not set. The following table shows the interrelation.

If bit 8 (*halt*) is set, the axis only starts at the falling edge of bit 8.

Bit 8 halt	**Bit 4 nsp**	**Condition**	**Definition**
0	0->1	Axis stopped	Positioning starting
0	0->1	Axis in motion	Position will finish before the next one starts (see 14)
1	0->1	Axis stopped	New target value transferred, axis stops
1->0	X	Axis stopped	Positioning starts if target value is available
0->1	X	Axis drives	Axis stops with normal brake chute. If target value is not reached, it remains active. A possibly new upcoming target value is only started up after the old one has been reached.

Table 57: Bits in the *Controlword* in the *profile position mode*

Through bits 10 (*target reached*), 12 (*setpoint acknowledge*) and 13 (*following error*) in the *Statusword* the status of the axis can be determined.

Bit	Value	Definition
10	0	Stop = 0: Target position not reached, axis in motion Stop = 1: Axis decelerates
	1	Stop = 0: Target position reached, axis stopped Stop = 1: Axis decelerated, axis stopped
12	0	The last target value was transferred, waiting for new target position
	1	The last target value was not yet transferred
13	0	Target-actual value difference (following error) in accepted range
	1	Target-actual value difference greater than following error window (see object 6065_h)

Table 58: Bits in the *Statusword* in the *profile position mode*

5.10.2 Objects

Index	Object	Name	Type	Access	Mem.
$607A_h$	VAR	*target position*	integer32	rw	N
$507A_h$	VAR	*target position 16bit*	integer16	rw	N
$607D_h$	ARRAY	*software position limit*	integer32	rw	Y
$607F_h$	VAR	*max profile velocity*	unsigned32	rw	Y
6081_h	VAR	*profile velocity*	unsigned32	rw	N
5081_h	VAR	*profile velocity 16bit*	unsigned16	rw	N
$60C5_h$	VAR	*max acceleration*	unsigned32	rw	Y
$60C6_h$	VAR	*max deceleration*	unsigned32	rw	Y
6083_h	VAR	*profile acceleration*	unsigned32	rw	Y
5083_h	VAR	*profile acceleration 16bit*	unsigned16	rw	Y
6084_h	VAR	*profile deceleration*	unsigned32	rw	Y
5084_h	VAR	*profile deceleration 16bit*	unsigned16	rw	Y

Table 59: Objects for closed-loop position controlled positioning

5.10.2.1 Object $607A_h$: *target position*

Index	Name	Type	Attr	Map	Default-Value	Description
$607A_h$	*target position*	integer32	rw	Y	0	Target position for next positioning signal in position units.
$507A_h$	*target position 16bit*	integer16	rw	Y	0	Like above, 16 bit access

The object is employed in the *profile position mode* and in the *cyclic synchronous position mode.*

5.10.2.2 Object 607D$_h$: *software position limit*

Index	Name	Object				
607D$_h$	*software position limit*	ARRAY	Limits for target position in position units.			
Sub-Index	**Name**	**Type**	**Attr**	**Map**	**Default-Value**	**Description**
0	*number of elements*	Unsigned8	ro	N	2	Number of elements
1	*min position limit*	integer32	rw	N	0	Lower target value limit
2	*max position limit*	integer32	rw	N	36000	Upper target value limit

Software *position limit* contains the target position limits in the sub parameters *min position limit* and *max position limit*. These parameters define the absolute position limits.

If it is tried to drive to a target position that lies outside those limits, the target position will be adjusted according to the respective limits and approached.

5.10.2.3 Object 607F$_h$: *max profile velocity*

Index	Name	Type	Attr	Map	Default-Value	Description
607F$_h$	*max profile velocity*	unsigned32	rw	N	20000	Maximum speed in position units/s

The object displays the **maximum profile velocity** of an axis. This is the maximum allowed speed with which the axis positions. It must not be greater than:

$$max\_profile\_velocity \leq \frac{max\_motor\_speed}{60} \cdot \frac{feed\_constant}{gear\_ratio}$$

Since the *feed constant* and *gear ratio* are fractions, the complete formula is:

$$max\_profile\_velocity \leq \frac{max\_motor\_speed}{60} \cdot \frac{feed\_constant.feed}{feed\_constant.shaft\_revolutions} \cdot \frac{gear\_ratio.shaft\_revolutions}{gear\_ratio.motor\_revolutions}$$

Object numbers of the variables:

max motor speed: 6080$_h$, Unit 1/min, therefore factor 60 in the formula.
gear ratio: 6091$_h$, no unit.
In case of position detection with motor encoder (*sensor selection code* = 0):
feed constant: 6092$_h$, positioning units.
In case of position detection with additional encoder (*sensor selection code* = -1):
(ae) feed constant: 6892$_h$, positioning units.

max profile velocity can be set smaller, in order to limit the accepted drive speed.

5.10.2.4 Object 6081$_h$: *profile velocity*

Index	Name	Type	Attr	Map	Default-Value	Description
6081$_h$	*profile velocity*	unsigned32	rw	Y	0	Target speed for the *profile position mode* in position units/s
5081$_h$	*profile velocity 16bit*	unsigned16	rw	Y	0	Like above, 16 bit access

This target velocity only applies for *profile position mode* (*Modes of operation* = 1). If a value greater than that in *max profile velocity* is set, the smaller value will be used for the actual positioning.

5.10.2.5 Object 60C5$_h$: *max acceleration*

Index	Name	Type	Attr	Map	Default-Value	Description
60C5$_h$	*max acceleration*	unsigned32	rw	Y	2000	Maximum acceleration pitch in the *profile position mode* and *profile velocity mode.* Unit: Position units/s²

This value limits the acceleration in *profile acceleration*.

5.10.2.6 Object 60C6$_h$: *max deceleration*

Index	Name	Type	Attr	Map	Default-Value	Description
60C6$_h$	*max deceleration*	unsigned32	rw	Y	0	Maximum deceleration pitch in the *profile position mode* and *profile velocity mode.* Unit: position units/s²

This value limits the deceleration chute in *profile deceleration*.

5.10.2.7 Object 6083$_h$: *profile acceleration*

Index	Name	Type	Attr	Map	Default-Value	Description
6083$_h$	*profile acceleration*	unsigned32	rw	Y	0	Axis acceleration in the *profile position mode* and *profile velocity mode.* Unit: Position units/s²
5083$_h$	*profile acceleration 16bit*	unsigned16	rw	Y	0	Like above, 16 bit access

The acceleration of the axis for closed-loop position controlled or RPM regulated positioning. The value is displayed in acceleration units. Those are differentiated from the units for travel and speed.

If a value greater than that in *max acceleration* is set here, a smaller value will be applied for actual positioning.

In order to identify the run up time (t_{acc})- and the deceleration time (t_{dec}), the following strings apply:

t_{acc} = *profile velocity* / *profile acceleration*

The regulations that can be set depend on the mechanical system and have to be determined for the respective application.

5.10.2.8 Object 6084$_h$: *profile deceleration*

Index	Name	Type	Attr	Map	Default-Value	Description
6084$_h$	*profile deceleration*	unsigned32	rw	Y	0	Deceleration chute of the axis in the *profile position mode* and *profile velocity mode.* Unit: position units/s²
5084$_h$	*profile deceleration 16bit*	unsigned16	rw	Y	0	Like above, 16 bit access

The deceleration of the axis for closed-loop position controlled or RPM regulated positioning. The value is displayed in acceleration units. Those are differentiated from the units for travel and speed.

If a value greater than that in *max deceleration* is set here, a smaller value will be applied for actual positioning. If 0 is set, the value for *profile acceleration* is used.

In order to identify the deceleration time (t_{acc}), the following strings apply:

t_{dec} = *profile velocity* / *profile deceleration*

5.11 *Profile Velocity Mode*

The *profile velocity mode* of the PMC offers only one possibility to accelerate the axis via an acceleration chute and decelerate it again. A shift in speed is also carried out via this chute.

The target values are limited by the respective parameters. This occurs at transfer of target values from field bus to the controller.

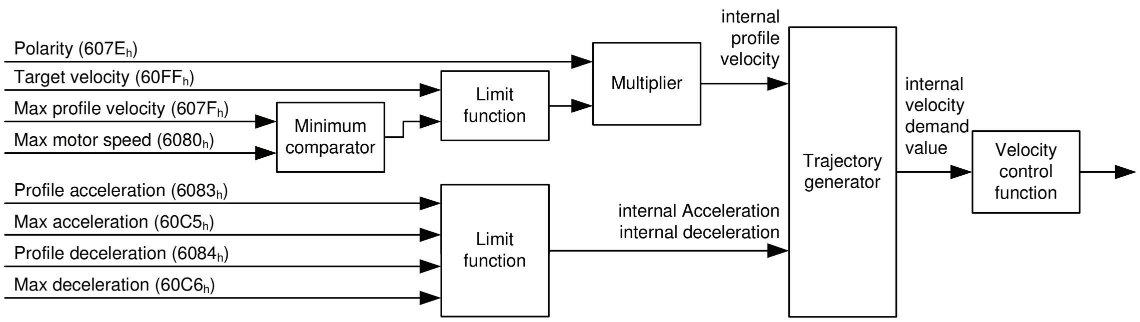

Figure 16: Parameters for *profile velocity mode*

The transfer of target values occurs by specification of object $60FF_h$. If bit 8 in the *Controlword* is set 0, the axis immediately accelerates to the set speed.

5.11.1 Use of *Controlword* and *Statusword*

No mode of operation specific bits are used in the *Controlword*.

If bit 8(*halt*) is set, the axis only starts up with the falling edge of 8 bit.

Bit 8 halt	Definition
0	New speed is set through chute, axis moves if target value <> Zero
1	Axis stops
1->0	Target speed is set through chute
0->1	Axis stops with normal brake chute

Table 60: Bits in the *Controlword* in the *profile velocity mode*

Through the bits 10 (*target reached*), 12 (*speed = 0*) in the *Statusword* the status of the axis can be determined. A monitoring of the set speed does not occur, there bit 13 always remains 0.

Bit	Value	Definition
10	0	Stop = 0: Target speed not achieved, axis accelerates or decelerates Stop = 1: Axis decelerates
	1	Stop = 0: Target speed achieved Stop = 1: Axis decelerated, axis stops
12	0	Speed is not 0, axis drives
	1	Speed is 0, axis stops
13	0	Target-actual value difference in the accepted range

Table 61: Bits in the *Statusword* in the *profile velocity mode*

The decision whether the axis reached target speed or stops, is made by the PMC on basis of the internal target values for the velocity controller in consideration of the possible setting accuracy.

5.11.2 Objects

Index	Object	Name	Type	Access	Mem.
$607F_h$	VAR	*max profile velocity*	unsigned32	rw	Y
$60FF_h$	VAR	*target velocity*	integer32	rw	N
$50FF_h$	VAR	*target velocity 16bit*	integer16	rw	N
6083_h	VAR	*profile acceleration*	unsigned32	rw	Y
5083_h	VAR	*profile acceleration 16bit*	unsigned16	rw	Y
6084_h	VAR	*profile deceleration*	unsigned32	rw	Y
5084_h	VAR	*profile deceleration 16bit*	unsigned16	rw	Y

Table 62: Objects for RPM regulated positioning

The objects $607F_h$, 6083_h and 6084_h, that were already described in chapter 5.10, „ (Profile Position Mode)“, are used for RPM regulated positioning as well.

5.11.2.1 Object $60FF_h$: *target velocity*

Index	Name	Type	Attr	Map	Default-Value	Description
$60FF_h$	*target velocity*	integer32	rw	Y	0	Target velocity for the *profile velocity mode* and the *cyclic synchronous velocity mode* in position units/s
$50FF_h$	*target velocity 16bit*	integer16	rw	Y	0	Like above, 16 bit access

This **target velocity** only applies to the *profile velocity mode* and *cyclic synchronous velocity mode* (*Modes of operation* = 3 or 9). If a value greater than that in *max profile velocity* is set, a smaller value is used for the actual positioning.

5.12 *Cyclic Synchronous Position Mode*

In this mode of operation, the PMC works as position controller with cyclic position setting. The track calculation occurs in the control. The position target value must be sent cyclically and synchronously via the field bus. For this, the synchronous mechanisms of the bus system should be used, like e.g.: synchronous PDOs at CANopen. The cycle time of the supplied target value are transferred to the PMC by the parameters *interpolation time period* (object 60C2h). It has to be a complete multiple of the set synchronous cycle in the bus system. The same parameters for limiting as in *profile position mode* are applicable.

If the superior position controller allows a velocity pre-control, this value can be cyclically transferred into *velocity offset* (object $60B1_h$).

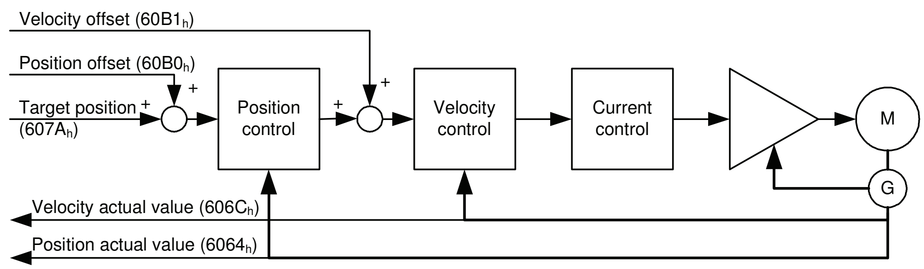

Figure 17: Overview *cyclic synchronous position mode*

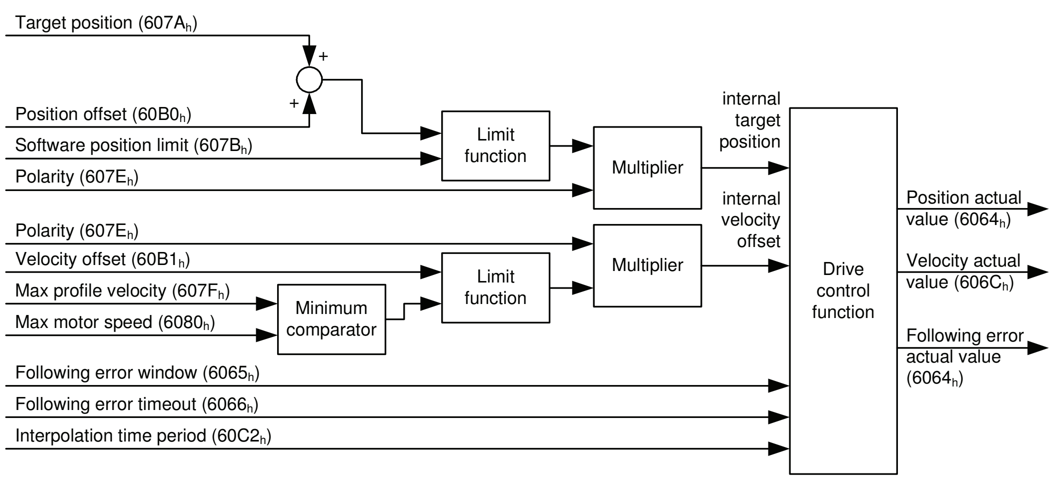

Figure 18: Parameters for the *cyclic synchronous position mode*

5.12.1 Use of *Controlword* and *Statusword*

No mode of operation specific bits are used in *Controlword*.

If bit 8(*halt*) is set, the axis does not follow the cyclic target values. With the falling edge from bit 8, it begins the position control to the cyclic target values. The control is responsible for forcing the correct positions. No acceleration or deceleration chute is driven.

Bit 8 halt	Definition
0	Axis follows the value in *target position*
1	Axis stops
1->0	Axis adjusts to the position in *target position*
0->1	Axis stops with minimal brake chute

Table 63: Bits in the *Controlword* in the *cyclic synchronous position mode*

Through bits 12 (*target position active*) and 13 (*following error*) in the *Statusword,* the status of the axis can be determined. Bit 10 (*target reached*) is not used.

Bit	Value	Definition
10	0	Reserved
	1	Reserved
12	0	Axis stops, the value in *target position* is ignored
	1	Axis follows the value in *target position*
13	0	Target-actual value difference (following error) in accepted range
	1	Target-actual value difference greater than following error window (see object 6065_h)

Table 64: Bits in the *Statusword* in the *cyclic synchronous position mode*

5.12.2 Objects

Index	Object	Name	Type	Access	Mem.
$607A_h$	VAR	*target position*	integer32	rw	N
$507A_h$	VAR	*target position 16bit*	integer16	rw	N
$607D_h$	ARRAY	*software position limit*	integer32	rw	Y
6081_h	VAR	*profile velocity*	unsigned32	rw	N
5081_h	VAR	*profile velocity 16bit*	unsigned16	rw	N
6083_h	VAR	*profile acceleration*	unsigned32	rw	Y
5083_h	VAR	*profile acceleration 16bit*	unsigned16	rw	Y
6084_h	VAR	*profile deceleration*	unsigned32	rw	Y
5084_h	VAR	*profile deceleration 16bit*	unsigned16	rw	Y
$60B0_h$	VAR	*position offset*	integer32	rw	N
$50B0_h$	VAR	*position offset 16bit*	integer16	rw	N
$60B1_h$	VAR	*velocity offset*	integer32	rw	N
$50B1_h$	VAR	*velocity offset 16bit*	integer16	rw	N
$60C2_h$	ARRAY	*interpolation time period*	unsigned8	rw	Y

Table 65: Objects for cyclic synchronous position control

The objects $607A_h$, $607D_h$, 6081_h and 6083_h are described in chapter 5.10.2, ex page 83.

5.12.2.1 Object $60B0_h$: *Position offset*

Index	Name	Type	Attr	Map	Default-Value	Description
$60B0_h$	*position offset*	integer32	rw	Y	0	Offset for the position in position units
$50B0_h$	*position offset 16bit*	integer16	rw	Y	0	Like above, 16 bit access

In the *cyclic synchronous position mode* (mode 8) this object includes an offset to the *target position*. This value is cyclically added, the sum is checked for position limits. The object can be used in order to process a position offset from an overlay set control algorithm (e.g. Rotor position dependent positioning).

5.12.2.2 Object 60B1$_h$: *Velocity offset*

Index	Name	Type	Attr	Map	Default-Value	Description
60B1$_h$	*velocity offset*	integer32	rw	Y	0	Offset for the velocity in position units/s
50B1$_h$	*velocity offset 16bit*	integer16	rw	Y	0	Like above, 16 bit access

In the *cyclic synchronous position mode* (mode 8) this object contains the pre-control velocity. In the *cyclic synchronous velocity mode* (mode 9) an offset to the *target velocity* can be transmitted here.

5.12.2.3 Object 60C2$_h$: *Interpolation time period*

Index	Name	Object				
60C2$_h$	*interpolation time period*	RECORD	Cycle time of the interpolation target values			
Sub-Index	**Name**	**Type**	**Attr**	**Map**	**Default-Value**	**Description**
0	*number of elements*	Unsigned8	ro	N	2	Number of elements
1	*time period value*	Unsigned8	rw	N	1	Time between two transmissions of target values
2	*time index*	integer8	rw	N	-3	Is interpreted as $*10^{\text{time index}}$ and represents the unit for the cycle time. Default: -3 -> 10^{-3} -> ms

The *interpolation time period* sets the period of time between two updates of the position and/or velocity target values. It is the basis for internal intermediate interpolation to the internal position control cycle (1ms).

The control system must be able to send the target values in at least this speed. The acceleration and the velocity are limited by the parameters *profile velocity* (6081$_h$), *profile acceleration* (6083$_h$) and *profile deceleration* (6084$_h$).

5.13 *Cyclic Synchronous Velocity Mode*

In this mode of operation, the PMC functions as a velocity controller with cyclic velocity setting. The position calculation takes place in the control. The velocity target value must be send cyclically and synchronously via the field bus. For this, the synchronous mechanisms of the bus system should be used, like e.g.: Synchronous-PDOs at CANopen. The cycle time of the delivered target values are transferred to the PMC through the parameter *interpolation time period* (object 60C2h). It has to be a complete multiple of the set synchronous cycle in the bus system. The same parameters for limiting like in the *profile velocity mode* are valid.

The velocity can be transferred in *target velocity* or *velocity offset*. The sum of these two is checked for limits and transferred to the velocity controller.

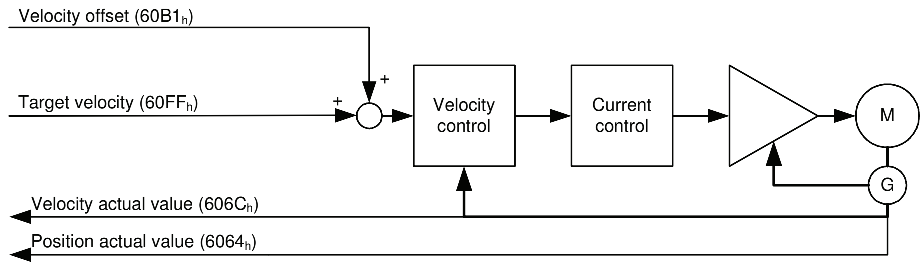

Figure 19: Overview *cyclic synchronous velocity mode*

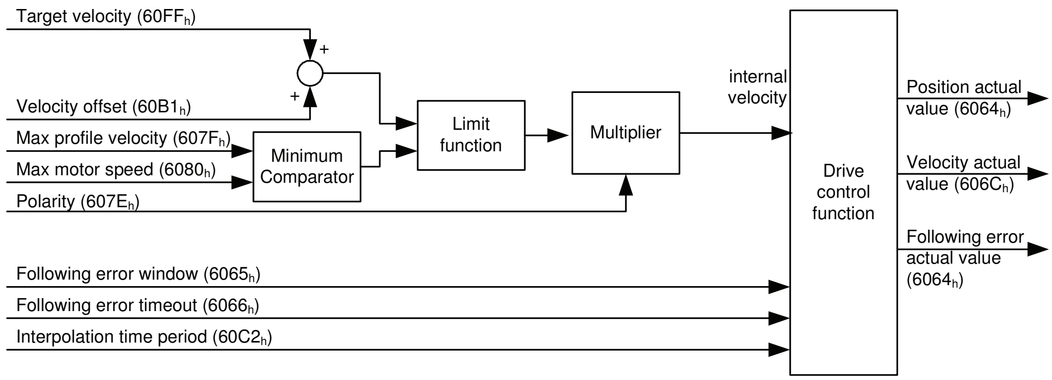

Figure 20: Parameters in the *cyclic synchronous velocity mode*

5.13.1 Use of *Controlword* and *Statusword*

No mode of operation specific bit are used in the *Controlword*.

If bit 8 (*halt*) is set, the axis does not follow the cyclic target values. It begins velocity control to the cyclic target values with the falling edge of bit 8. No acceleration or deceleration chute are driven. The lastly transferred velocity is kept.

Bit 8 halt	Definition
0	Axis follows the value in *target velocity*
1	Axis stops
1->0	Axis adjusts to the velocity in *target velocity*
0->1	Axis stops with maximum brake chute

Table 66: Bits in the *Controlword* in the *cyclic synchronous velocity mode*

With bit 12 (*target velocity active*) in the *Statusword*, the status of the axis can be determined. Bits 10 (*target reached*) and 13 are not used.

Bit	Value	Definition
10	0	Reserved
	1	Reserved
12	0	Axis stops, value in *target velocity* ignored
	1	Axis follows the value in *target velocity*
13	0	Reserved
	1	Reserved

Table 67: Bits in the *Statusword* in the *cyclic synchronous velocity mode*

5.13.2 Objects

Index	Object	Name	Type	Access	Mem.
$607F_h$	VAR	*max profile velocity*	unsigned32	rw	Y
$60FF_h$	VAR	*target velocity*	unsigned32	rw	N
$50FF_h$	VAR	*target velocity 16bit*	unsigned16	rw	N
$60B1_h$	VAR	*velocity offset*	integer32	rw	N
$50B1_h$	VAR	*velocity offset 16bit*	integer16	rw	N
$60C2_h$	ARRAY	*interpolation time period*	unsigned8	rw	Y

Table 68: Objects for cyclic synchronous RPM regulation

These objects were explained in previous chapter.

5.14 Set Zero Position (*Homing Mode*)

This mode of operation is only used to set the zero position of the axis. At the current development status (July 2007) the PMC does only support the *Homing Method* 35, in other words set zero at the current position. The offset calculated in such a way that the position entered in *target position* ($607A_h$) is set.

Furthermore, the objects *home offset* ($607C_h$) and *ae.home offset* ($687C_h$) can only be written in the *Homing Mode.*

5.14.1 Use of *Controlword* and *Statusword*

The axis must stand in *homing method* 35 and be disabled. With the rising edge of Bit 4 (*new setpoint*) the home offset for the current position is calculated. This is confirmed with bit 12 in the *statusword*. In order to permanently save this value, the parameters have to be saved in the flash. This happens by writing of 65766173_h in object 1010_h sub index 3.

Bit	Value	Definition
4	0	Don't start referencing
nsp	1	Start referencing, calculate reference point
8	0	Start axis for referencing (only if motion is required)
halt	1	Axis stops

Table 69: Bits in the *Controlword* in *homing mode*

Bit	Value	Definition
10	0	Command execution in process
	1	Command executed
12	0	Zero point not set
	1	Zero point set
13	0	No error
	1	Referencing error

Table 70: Bits in the *Statusword* in the *homing mode*

5.14.2 Objects

Index	Object	Name	Type	Access	Mem.
6098_h	VAR	*homing method*	integer8	rw	Y
$607C_h$	VAR	*home offset*	integer32	rw	Y
$687C_h$	VAR	*ae.home offset*	integer32	rw	Y

Object *home offset* ($607C_h$) see page 62.

Object ae.*home offset* ($687C_h$) see page 66.

5.14.2.1 Object 6098_h: *homing method*

With this object, the method to determine the zero point of the axis is selected.

Only mode 35 is currently (status July 2007) supported.

Index	Name	Type	Attr	Map	Default-Value	Description
6098_h	*homing method*	integer8	rw	N	35	Method for determining zero point (Referencing)

5.15 *Emergency Feather Command* (EFC)

The „*Feather position*“ is the 90° position of the rotor blade, at which no lifting is produced, in other words the stand-by position of the blade. With the help of the „*Emergency Feather Command*“ the blade shall get there as quickly as possible and under all circumstances.

The EFC drive is an exception of the *profile position mode* at the PMC. It can be triggered by an external input signal (X2.3) or by bit11 in the *Controlword*.

If the drive is triggered by an external signal, bit 9 (*remote*) is reset in the *Statusword* and a control via the *Controlword* is no longer possible, until the EFC input signal becomes inactive. The *Statusword* further shows the status of the axis, as described in chapter 5.10. The mode of operation is automatically switched to *profile position mode*. This mode of operation remains even after the EFC drive.

Position, velocity and acceleration for the EFC drive are taken from the objects saved in the PMC.

5.15.1 Objects

Index	Object	Name	Type	Access	Mem.
$567A_h$	VAR	*EFC target position*	integer32	rw	Y
5681_h	VAR	*EFC profile velocity*	unsigned32	rw	Y
5683_h	VAR	*EFC profile acceleration*	unsigned32	rw	Y
5684_h	VAR	*EFC profile deceleration*	unsigned32	rw	Y

Table 71: Objects for closed-loop position controlled positioning

5.15.1.1 Object $567A_h$: *EFC target position*

Index	Name	Type	Attr	Map	Default-Value	Description
$567A_h$	*EFC target position*	integer32	rw	N	90,00°	Target position for the EFC drive. Unit: Position units

In the EFC case, the object is transferred to the object $607A_h$, *target position*.

5.15.1.2 Object 5681_h: *EFC profile velocity*

Index	Name	Type	Attr	Map	Default-Value	Description
5681_h	*EFC profile velocity*	unsigned32	rw	N	10,00°/s	Target value velocity for the EFC drive. Unit: Position units/s

In the EFC case, the object is transferred to the object 6081_h, *profile velocity*.

5.15.1.3 Object 5683_h: *EFC profile acceleration*

Index	Name	Type	Attr	Map	Default-Value	Description
5683_h	*EFC profile accelera-tion*	unsigned32	rw	N	20,00°/s²	Acceleration chute for the EFC drive. Unit: Position unit/s²

In the EFC case the object is transferred to the object 6083_h, *profile acceleration*.

5.15.1.4 Object 5684_h: *EFC profile deceleration*

Index	Name	Type	Attr	Map	Default-Value	Description
5684_h	*EFC profile decelera-tion*	unsigned32	rw	N	20,00°/s²	Deceleration chute for the EFC drive. Unit: position units/s²

In the EFC case the object is transferred to the object 6084_h, *profile deceleration*.

5.16 Inputs and Outputs

The input and output states can be queried in the objects explained in the following. Furthermore, a measuring input (probe) is available, that will save the position in case of a level change.

5.16.1 Objects

Index	Object	Name	Type	Access	Mem.
$60FD_h$	VAR	*digital inputs*	unsigned32	ro	N
$60FE_h$	VAR	*digital outputs*	ARRAY	rw	N
5664_h	ARRAY	*capture values*	integer32	ro	N

5.16.1.1 Object $60FD_h$: *digital inputs*

This index specifies the digital inputs of the controller. Signals like reference input and limit switch can be found here.

Index	Name	Type	Attr	Map	Default-Value	Description
$60FD_h$	*digital inputs*	unsigned32	ro	Y	0	1 = 24V Input level

The setting of bits 0..15 is specified in DS402, Bit 16..31 are manufacturer dependent.

Bit	Digital input	Clamp
0	Negative quick stop input (0 = 0V, quick stop active!)	X4.9
1	Positive quick stop input (0 = 0V, quick stop active!)	X4.8
2	Referential switch (1 = High level)	X4.6
3	Release input (1 = Enable)	X4.5
4..15	Reserved, always 0	
16	"*feather position*", 90°-switch(1 = High level)	X4.10
17	Measuring input (probe)	X4.7
18	EFC, *Emergency Feather Command* (0 = execute emergency drive)	X2.3
19	Reset input, (0 -> 1 = execute reset)	X2.4
20	Free input	X2.5
21..31	Not used, always 0	

Table 72: Setting of the digital inputs ($60FD_h$). 1 means voltage at input.

5.16.1.2 Object $60FE_h$: *digital outputs*

In the current (July 2007) expansion stage, the PMC has no outputs with external access. Sub index 1 and 2 have no function, in sub index 3 the internal accessed outputs can be back-read.

Index	Name	Object				
$60FE_h$	*digital outputs*	ARRAY	Physical outputs and bit mask			
Sub-Index	**Name**	**Type**	**Attr**	**Map**	**Default-Value**	**Description**
0	*number of elements*	unsigned8	ro	N	3	Number of elements
1	*physical outputs*	unsigned32	rw	N	0	At the PMC, currently unused
2	*bitmask*	unsigned32	rw	N	0	At the PMC, currently unused
3	*read outputs*	unsigned32	ro	N		Back-reading of the outputs

The setting of bits 0..15 is specified in the DS402, Bit 16..31 are manufacturer dependent.

Bit	Digital Output	Clamp
0	Holding brake, 1 = Brake off	X6.3/X6.4
1..15	Reserved, always 0	
16	Watchdog switch, 1 = OK, set internally	X2.1/X2.2
17..23	Not used, always 0	
24	LED Status device	
25	LED Failure	
26	LED Watchdog	
27	LED Field bus 1	
28	LED Field bus 2	
29	LED Field bus 3	
30	Not used, always 0	
31	Not used, always 0	

Table 73: Digital output configuration (60FE$_h$). 1 means: output set.

5.16.1.3 Objects 60BA$_h$ and 60BB$_h$: *touch probe pos 1*

The input X4.7 (probe) can be used as measuring input. With every change at this input, the actual nominal value is saved in the respective objects. This can be used for a mechanical position control through switches, among others.

Index	Name	Type	Attr	Map	Default-Value	Description
60BA$_h$	*touch probe pos 1 pos value*	integer32	ro	N		Position value at a rising edge (0V→24V) at the probe input
60BB$_h$	*touch probe pos 1 neg value*	integer32	ro	N		Position value at a falling edge (24V→0V) at the probe input

The scanning of the input is clocked to one millisecond. The accuracy of the measured position depends on the speed of the axis and can be calculated by dividing the velocity (e.g. in °/s) by 1000.

5.17 Use of PDOs

A detailed description of the PDOs can be found in the documentation "Pitch Control System, CANopen Interface". In the document at hand, only the presetting of the PDO is explained. It is pursuant to the specifications of the CiA, DS402, Drives and Motion Profile.

Only the first PDO is active after switch-on, it only contains the *statusword* or *controlword*.

If further PDOs shall be used, they have to be activated by the control first.

The setting (the „mapping") can be changed and thus adjusted to the requirements.

Manufacturer specific version of the firmware can provide other default values for the mapping. Those can be found in a separate documentation.

5.17.1 Default Settings of Mapping Parameters, Receiving PDO

PDO	COB-ID	active	Index	Sub	Default value	Object
RPDO01	200_h+NodeID	yes	1600_h	0	1	
				1	6040 00 10_h	*controlword*
RPDO02	80000300_h+NodeID	no	1601_h	0	2	
				1	6040 00 10_h	*controlword*
				2	607A 00 20_h	*target position*
RPDO03	80000400_h+NodeID	no	1602_h	0	2	
				1	6040 00 10_h	*controlword*
				2	60FF 00 20_h	*target velocity*
RPDO04	80000500_h+NodeID	no	1603_h	0	5	
				1	607A 00 20_h	*target position*
				2	6081 00 20_h	*profile velocity*

Table 74: PMC: Default mapping of the receiving PDOs

5.17.2 Default Settings of Mapping Parameters, Sending PDO:

PDO	COB-ID	active	Index	Sub	Default value	Object
TPDO01	40000180h+NodeID	yes	$1A00_h$	0	1	
				1	6041 00 10_h	*statusword*
TPDO02	C0000280h+NodeID	no	$1A01_h$	0	2	
				1	6041 00 10_h	*statusword*
				2	6064 00 20_h	*position actual value*
TPDO03	C0000380h+NodeID	no	$1A02_h$	0	2	
				1	6041 00 10_h	*statusword*
				2	606C 00 20_h	*actual velocity*
TPDO04	C0000480h+NodeID	no	$1A03_h$	0	2	
				1	6064 00 20_h	*position actual value*
				2	606C 00 20_h	*actual velocity*

Table 75: PMC: Default mapping of the sending PDO

6 Appendix

6.1 Contents of Figures

6.2 Glossary

Explanation of abbreviations and terms

CiA	CAN in Automation (CAN-user organization)
EPROM	Erasable Programmable Read Only Memory
ID	Identifier
LSB	Least Significant Bit (Byte)
MSB	Most Significant Bit (Byte)
PCS	Pitch Control System
PDO	Process Data Object, CAN-telegram for transmission of process data
PMC	Digital axis regulation controller
PMM	Pitch Management System
PWM	Pulse width modulation, is used for motor controlling
SDO	Service Data Object, CAN-telegram for transmission of service data

Specification Corrosion Cabinets and enclosures

1. PURPOSE

This purchase specification specifies the characteristics of organic coatings concerning their mechanical properties and their corrosion protection.

This specification refers to tests and properties achieved with sample panels regarding the quality of the coating. Real products may be coated differing. Surfaces inside the protected room and surfaces of protected openings may show lower layer thicknesses. Colour and appearance
are no object of this specification.

To verify the corrosion protection, corrosion tests with complete products or scaled models with all critical properties have to be accomplished additional.

2. SCOPE OF VALIDITY

This purchase specification is valid for all surfaces of organic coated parts made of carbon steel, which are lying outside the sealed area.

3. OTHER VALID DOCUMENTS

DIN EN ISO 9227	Corrosion tests in artificial atmospheres - salt spray tests
DIN EN ISO 1519	Mandrel bending test on coatings
DIN EN ISO 1520	Cupping in accordance with Erichsen on paints and similar coatings with visual assessment
DIN EN ISO 2178	Non-magnetic coatings on magnetic substrates - Measurement of coating thickness - Magnetic method
DIN EN ISO 2360	Non-conductive coatings on non-magnetic electrically conductive basis materials - Measurement of coating thickness - Amplitude-sensitive eddy current method
DIN EN ISO 2409	Cross-cut tests of paintwork and similar coatings
DIN EN ISO 4628-2	Paints and varnishes - Evaluation of degradation of coatings - Designation of quantity and size of defects, and of intensity of uniform changes in appearance - Part 2: Assessment of degree of blistering
DIN EN ISO 4628-3	Paints and varnishes - Evaluation of degradation of coatings - Designation of quantity and size of defects, and of intensity of uniform changes in appearance - Part 3: Assessment of degree of rusting
DIN EN ISO 4628-4	Paints and varnishes - Evaluation of degradation of coatings - Designation of quantity and size of defects, and of intensity of uniform changes in appearance - Part 4: Assessment of degree of cracking
DIN EN ISO 4628-5	Paints and varnishes - Evaluation of degradation of coatings - Designation of quantity and size of defects, and of intensity of uniform changes in appearance - Part 5: Assessment of degree of flaking
DIN EN ISO 4628-8	Paints and varnishes - Evaluation of degradation of coatings - Designation of quantity and size of defects, and of intensity of uniform changes in ap-

	pearance - Part 8: Assessment of degree of delamination and corrosion around a scribe
DIN EN ISO 6270-2	Coating materials – determination of the resistance to humidity - Part 2: Procedure for exposing test specimens in condensation water atmosphere
DIN EN ISO 17872	Paints and varnishes - Guidelines for the introduction of scribe marks through coatings on metallic panels for corrosion testing
DIN EN ISO 20567-1	Paints and varnishes - Determination of stone-chip resistance of coatings - Part 1: Multi-impact testing

4. SAMPLES AND PREPARATION OF SAMPLES

4.1 Test panels

The test panels are to be made of easily formable sheet steel, of material from the serial production. The build-up of the coating (cleaning, pre-treatment, grounding, finish) has to be done identical to the serial production process.

4.2 Paint finish

The total coat thickness should be 60 µm - 100 µm on average. Textured surface shall have no single measurement value less than 35 µm or higher than 140 µm.

4.3 Number and dimension of sample panels

The dimensions of the required test panels are 200 x 200 x 1…1.5 mm. To perform the following tests, at least 10 sample panels are required.

5. SPECIFICATION AND REQUIREMENTS

5.1 Corrosion resistance

The coatings of the samples for the corrosion tests have to be scribed according to DIN EN ISO 17872. One single scribe has to be fit lengthwise to one edge in the middle of the sample. The scribe has to reach the substrate, “Van Laar-tool” has to be used.

5.1.1 Salt spray test DIN EN ISO 9227 NSS Test duration 480 h

The specimens have to be positioned in a holder on the ground of the test chamber with an angle of 20° ± 5° with the scribed side facing up, the scribe has to be positioned vertical. The scribed side has to show into direction of the spray nozzle.

5.1.2 Condensation test DIN EN ISO 6270-2 CH Test duration 500 h

The specimens have to be positioned hanging in the test chamber with the scribe in vertical position.

5.1.3 Condensation alternating atmosphere DIN EN ISO 6270-2 AHT Test duration, 20 cycles

The specimens have to be positioned hanging in the test chamber with the scribe in vertical position.

5.1.4 Evaluation

At least two of the three tested samples have to fulfil the following requirements.

DIN EN ISO 4628-8 Degree of corrosion	c < 5mm.
DIN EN ISO 4628-8 Degree of delamination	d < 6mm.

Away from the scribe and the edges :
The surface of the specimen besides an area of 10 mm away from the scribe and the edges or other similar critical areas may not show any deterioration as blistering, rusting, cracking or flaking.

Near the edges :

DIN EN ISO 4628-2 Blistering may not exceed	2(S4)
DIN EN ISO 4628-3 Rusting may not exceed	Ri3
DIN EN ISO 4628-4 Cracks may not exceed	Rating 1
DIN EN ISO 4628-5 Flaking may not exceed	Rating 3

5.2 CORROSION TESTS ON COMPLETE PRODUCTS

After successful completion of the tests on sample panels, complete products or scaled models with all critical properties of the original products have to be tested in salt spray test according to DIN EN ISO 9227 NSS for 480 hours. The products have to be positioned as utilised in service, doors, covers etc. have to be closed during the test.

Fitting parts (screws, hinges, locking rods) have to be judged similar to the painted parts. Locations where corrosion protection is impractical (slide faces etc.) are not to be judged.

Evaluation as stated in 5.1.4 Evaluation.

6. Mechanical properties

6.1.1 Adhesion test in accordance with DIN EN ISO 2409

Evaluation: Set value Gt 0-1

6.1.2 Cupping in accordance with Erichsen to DIN EN ISO 1520 m, tool diameter 20 mm, panel thickness 1 mm

Evaluation: Minimum value 3.5 mm

6.1.3 Mandrel bending test in accordance with DIN EN ISO 1519, mandrel 10 mm

Evaluation: No flaking

6.1.4 Multi-impact test in accordance with DIN EN ISO 20567-1, method A

Evaluation: The specific value may not exceed 1.5.