Data exchange with PowerCube

Description of PowerCube communication interfaces

Directory of Revisions			
Authorized by: Roland Tschakarow			
File name: Data exchange with PowerCube.doc			
No.	Description	Revision Index	Date of Change
001	1. Version	1.0	31.01.2001
002	Used C Datatypes	1.1	10.01.2002
003	Revision	1.2	30.07.2004
004	Added new functions	1.3	15.07.2007

1 PowerCube communication interfaces

The Operating System (OS) of each PowerCube drive module fulfills several tasks:
- Real time control based on the encoder information.
- Observation of Motor current, temperature and limit switches.
- Execution of motion commands received from the host computer.
- Controlling the brake (as far as a brake has been installed).

This document describes the functions in detail. This concerns information about data exchange between master and slave as wells as diagnostics and trouble shooting with PowerCube.

2 Communication Devices

The PowerCubes use a serial data interface for reading and writing motion commands, parameters and motor data.
There are three types of communication devices available (only one actve at a time)
- RS232,
- CAN bus (according to ISO/DIS 11898),
- Profibus-DP.

The PowerCube CAN-Bus interface supports two protocol types:
- CAN data protocol according to Amtec specifications
- CAN data protocol according to CANopen standard DS402 (subset, see document "CANopen and PowerCube" for further information).

3 Data types

To establish a data connection to the PowerCube it is necessary to specify the data types used:

Data type	Description	Data width	Range (decimal)
char	Byte	8 Bit	-128 to 127
usigned char	Byte, unsigned	8 Bit	0 to 255
sort	Word	16 Bit	-32768 to 32767
usigned short	Word, unsigned	16 Bit	0 to 65535
long	Double word	32 Bit	-2147483848 to 2147483847
unsigned long	Double word, unsigned	32 Bit	0 to 4294967295
float	Floating point value	32 Bit	3.4 e-038 to 3.4 e+038

Please note the storage format of the data described in the section following.

3.1 Storage format

The PowerCubes expect data transfered in Intel format (Little-Endian resp. reverse byte ordering). This storage method stores the least significant Byte of a number on the first position. The hexadecimal number 0x12345678 will be stored like this:

0x12345678	0x78	0x56	0x34	0x12

This is important especially when using Motorola- or Sparc-Prozessors saving their data in Big-Endian-Format (PLC-Programs).

4 Communication protocol

The protocol used to exchange data between master and PowerCube is unified and independent from the bus interface used. The module receives serial data, interpretes it and acknowledges the command. The time necessary for this transfer depends on the bus interface and its parameters.

A PowerCube™ command has this structure:

Identifier	CommandID	ParameterID resp. MotionID	Data 0 ... Data 7

Command ID's, Parameter ID's and following data bytes are identical for all communication interfaces (RS232, CAN, Profibus-DP). Only the Identifiers vary in dependency of the bus interface used.

The PowerCube acknowledges all commands received (as far as not configured in another way). The acknowledge data frame has the same structure as the command.

4.1 PowerCube™-Identifiers for CAN-Bus (Basic CAN)

This section describes the colored part of the data frame:

Identifier	CommandID	ParameterID resp. MotionID	Data 0 ... Data 7

A CAN-Bus-Identifier according to CAN-Specification 2.0 Part A (11-bit-ID) has this structure:

CAN-Bus Identifier according to Specification 2.0 Part A															
Byte 1								Byte 0							
11-bit Message Identifier											RTR-Flag	Data Length Code			
ID.10	ID.9	ID.8	ID.7	ID.6	ID.5	ID.4	ID.3	ID2	ID.1	ID.0	RTR	DLC.3	DLC.2	DLC.1	DLC.0

RTR = Remote-Transmission-Request
DLC = Data Length Code
ID = Identifier

4.1.1 Addressing single Modules

The Identifiers for addressing single PowerCubes have this structure:

CAN-Identifier for PowerCube	ID. 10	ID. 9	ID. 8	ID. 7	ID. 6	ID. 5	ID. 4	ID. 3	ID. 2	ID. 1	ID. 0	RTR	DLC .3	DLC .2	DLC .1	DLC .0
CANID_CMDACK	0	0	0	1	0	1	M	M	M	M	M	0	L	L	L	L
CANID_CMDGET	0	0	0	1	1	0	M	M	M	M	M	0	L	L	L	L
CANID_CMDPUT	0	0	0	1	1	1	M	M	M	M	M	0	L	L	L	L

M = Address of PowerCube module ($1 \le M \le 31$)
L = Number of data bytes following

The PowerCube-Identifier for CAN-Bus can be calculated like this:
CANID_CMDACK = 0x0a0 + Module address + Number of data bytes ($1 \le L \le 8$)
CANID_CMDGET = 0x0c0 + Module address + Number of data bytes ($L = 2$)
CANID_CMDPUT = 0x0e0 + Module address + Number of data bytes ($1 \le L \le 8$)

4.1.2 Addressing all Modules (Broadcasting)

A special CAN-Identifier enables the user to send broadcast messages to all connected modules:

CAN-Identifier for PowerCube-Broadcasts	ID. 10	ID. 9	ID. 8	ID. 7	ID. 6	ID. 5	ID. 4	ID. 3	ID. 2	ID. 1	ID. 0	RTR	DLC .3	DLC .2	DLC .1	DLC .0
CANID_CMDALL	0	0	1	0	0	0	0	0	0	0	0	0	L	L	L	L

L = Number of data bytes following

The PowerCube-Broadcast-Identifier can be calculated like this:
CANID_CMDALL = 0x100 + Number of data bytes ($1 \le L \le 8$)

Using the CAN-Identifier CANID_CMDALL it is possible to send the messages described in the table below in only one command. These commands are processed with priority. The modules do not acknowledge them.

CANID_CMDALL (0x100)	Byte 0 ComandID	Byte1 Param.ID	Description	Comment
	0x00	-	Reset	Reset-command to all
	0x01	-	Home	Home-Kommando to all
	0x02	-	Halt	Halt-Kommando to all
	0x07	-	Watchdog-Refresh	Watchdog-Refresh to all
	0x09	Baudrate	Change Baudrate	Changes the baud rate
	0x0e	-	Save position	Stores the actual position
	0x0f	-	Synchronize Motion	Releases the last motion command

Baudrate: 1 = 250 kbit/s, 2 = 500 kbit/s, 3 = 1Mbit/s.

4.1.3 Reserved CAN-Identifiers

The CAN-Identifiers 0x3E9 through 0x7E5 are reserved for a maximum of 51 I/O-Modules, manufactured by EMS Thomas Wünsche GmbH.

The CAN-Identifiers 0x580 through 0x680 are reserved for a maximum of 127 force measurement cells of Type MP55, manufactured by Hottinger Baldwin Messtechnik GmbH.

4.2 PowerCube™-Identifiers for Profibus-DP

This section describes the colored part of the data frame:

Identifier	CommandID	ParameterID resp. MotionID	Data 0 ... Data 7

Addressing Profibus slaves is part of the physical layer of Profibus. Therefore no separate Identifier is required for this purpose.
The PowerCube Module have been certified by the PNO (Profibus users organisation) and thus are able to operate with all certified Profibus-DP Masters. At the same time they can be combined with any third party Profibus certified device on the bus. The only requirement for this is a correct configuration of the DP master. Please use the GSD file supplied on the CDRom (Cube1641.gsd) in order to configure the master. A PowerCube has 8 output bytes and 16 input bytes.

When using Profibus communication there is one important issue for error free operation:
In order to enable a module to answer a new command (e.g. Polling of module state or position), at least one of the 8 output bytes has to change its value. This can be done easily by incrementing the normally unused byte 7 (count starting from zero).

4.2.1 Organisation of the 16 Input Bytes

The lower 8 input bytes always incorporate the modules answer to the last command. Their detailed description follows in the next chapters.
The upper 8 input bytes are Profibus specific and contain this information with each answer:

IN Byte 8	IN Byte 9	IN Byte 10	IN Byte 11	IN Byte 12	IN Byte 13	IN Byte 14	IN Byte 15
-	Actual position in Increments (Int32 = 4 Byte)				Short state (UInt16 = 2 Byte)		ms (0..255)

The Profibus-DP short state is defined like this:

Value	Name of Flag	Flag according to full module state
0x0001	STATE_ERRORM4	STATE_ERROR
0x0004	STATE_READYTOPOS	STATE_HOME_OK
0x0010	STATE_MOVING	STATE_MOTION
0x0020	STATE_BRAKED	STATE_BRAKEACTIVE
0x0040	STATE_CURRLIMIT	STATE_CURLIMIT
0x0080	STATE_ENDMOTION	STATE_RAMP_END

Value	Name of Flag	Flag according to full module state
0x0100	STATE_END0	STATE_SW1
0x0200	STATE_END1	STATE_SW2
0x0400	STATE_SYNC	STATE_SWR
0x1000	POSTIME_INPROGRESS_MASK	STATE_INPROGRESS
0x2000	POSTIME_RECEIVED_MASK	STATE_FULLBUFFER
0x8000	STATE_HALTM4	STATE_HALTED

If your application keeps tracking the upper 8 input bytes, there is no necessity to separately poll module state and actual position. On top of this the permanently changing ms-Information (Milliseconds) can be used for monitoring a "life sign" (watchdog).

4.3 PowerCube™-Identifier for RS232 communication

This section describes the colored part of the data frame:

Identifier	CommandID	ParameterID resp. MotionID	Data 0 ... Data 7

The data transfer via RS232 communication unlike industrial field bus systems like CAN and Profibus is not supported by a hardware protocol layer taking care of error free transmission. For this reason Amtec has designed an own data protocol for secure data transfer with RS232 comunication interfaces. A data stream for RS232 has this structure:

STX	TELID	TELID	Data0	Data1	Data2	Data3	Data4	Data5	Data6	Data7	BCC	ETX
0x02	TX or RX		Data								BCC	0x03

BCC = Block Check Charakter (Check sum)
STX = Start of Text
STX = End of Text
TELID = Identifier

A frame is always starting with the character STX (02h) and finishes with the character ETX (03h). If these characters occur within the data stream they will be replaced using a combination of two characters: DLE (10h) and (80h + character to replace). The character DLE (10h) used in this case will thus be replaced in the same manner:

- 0x02 is replaced by 0x10 0x82
- 0x03 is replaced by 0x10 0x83
- 0x10 is replaced by 0x10 0x90.

This DLE correction is used for the complete data stream (including the Identifier). The maximum length of the data stream therefore is 24 Byte.
The Block Check Character is named BCC. This is a checksum over the complete net data frame (without DLE correction). Using this method data integrity is secured to about 95%.

```
BCC = (TELID + Command + Data);
BCC = BCC + (BCC >> 8);
```

The 16-Bit Identifier TELID differs between commands sent to the module (TELID_SENDDAT) and answers from the module (TELID_RECVDAT). This is the TELID structure:

	1. Byte								2. Byte							
Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
TELID_SENDDAT	0	0	0	0	0	1	M	M	M	M	M	0	L	L	L	L
TELID_RECVDAT	0	0	0	0	1	0	M	M	M	M	M	0	L	L	L	L

- M: Module address ($1 \leq M \leq 31$)
- L: number of bytes following (net, without DLE correction)

A command to the module will be interpreted if :
- TELID is coded as SEND,

- the module address matches,
- the number of received net data bytes matches the coded length.

The Block Check Character BCC is not checked because of compatibility reasons.

4.4 PowerCube Command ID's

This section describes the colored part of the data frame:

Identifier	**CommandID**	ParameterID resp. MotionID	Data 0 ... Data 7

The data transfered first of all states the command to be processed by the module. These commands are available:

Command	**Command ID**	**Length**	**Meaning**
Reset	0x00	1 Byte	Clear error state
Home	0x01	1 Byte	Start Homing procedure
Halt	0x02	1 Byte	Stop immediately
RecalcPIDParam	0x09	1 Byte	Recalculate the PID loop parameters
SetExtended	0x08	3-6 Byte	Set parameter
GetExtended	0x0a	2 Byte	Fetch parameter
SetMotion	0x0b	6-8 Byte	Set Motion command
SetIStep	0x0d	7 Byte	Motion command in Step mode
ResetTime	0x12	1 Byte	Reset internal clock to zero

The data sent to the module will be immediately acknowledged (if not configured otherwise in the configuration word). The data flow looks like this:

Command	**Send**		**Acknowledge**	
Reset	[0x00]	len = 1	[0x00]	len = 1
Home	[0x01]	len = 1	[0x01]	len = 1
Halt	[0x02]	len = 1	[0x02]	len = 1
RecalcPIDParam	[0x09]	len = 1	[0x09]	len = 1
SetExtended	[0x08] [paramID] [data] ...	len = 3 .. 6	[0x08] [paramID] [0x64]	len = 3
GetExtended	[0x0a] [paramID]	len = 2	[0x0a] [paramID] [data] ...	len = 3 .. 6
SetMotion	[0x0b] [motionID] [data] ...	len = 6 .. 8	[0x0b] [motionID] [data] ...	len = 3 .. 8
SetIStep	[0x0d] [data] ...	len = 7	[0x0d]	len = 1
ResetTime	[0x12]	len = 1	[0x12]	len = 1

When sending the command „Fetch parameter“ (GetExtended) or „Set parameter“ (SetExtended) the data byte following the CommandID specifies the parameter index. The amount of data bytes to follow depends on the parameter type. The list of available parameters is described in the next chapter.
The command „SetMotion“ expects the MotionID as the data byte following the CommandID. The amount of data bytes sent with the command depends on the motion mode chosen. The next section describes motion modes and MotionIDs in detail.

4.5 PowerCube Motion ID's

This section describes the colored part of the data frame:

Identifier	CommandID	ParameterID resp. **MotionID**	Data 0 ... Data 7

Using the MotionID the user can chose one of the motion modes decribed in the table below. Depending on the MotionID, the data bytes following are interpreted in a different way.

Motion mode	MotionID	Parameters	Datatype	Remark	Acknowledge
FRAMP_MODE	4	Target position	float	rad resp. m	Motion commands with these MotionIDs use a simple acknowledge: [0x0b][motionID][0x64]
FSTEP_MODE	6	Target position Time	float UInt16	rad resp. m ms	
FVEL_MODE	7	Velocity	float	rad/s resp. m/s.	
FCUR_MODE	8	Sollstrom	float	A.	
IRAMP_MODE	9	Target position	Int32	Encoder ticks.	
ISTEP_MODE	11	Target position Time	Int32 UInt16	Encoder ticks. ms.	
IVEL_MODE	12	Velocity	Int32	Encoder ticks/s.	
ICUR_MODE	13	Current	Int16	Digits.	
FCOSLOOP*	24	Target position Time	Float UInt16	rad/s resp. m/s. ms	
FRAMPLOOP*	25	Target position	float	rad/s resp. m/s.	
FRAMP_ACK	14	Target position	float	rad resp. m.	Motion commands with these MotionIDs use an extended acknowledge: [0x0b][motionID] [position][state][dio] PowerCube-OS versions higher than 2.5.16 resp. 3.5.13
FSTEP_ACK	16	Zielposition Zeitvorgabe	float UInt16	rad resp. m. ms.	
FVEL_ACK	17	Velocity	float	rad/s resp. m/s.	
FCUR_ACK	18	Current	float	A.	
IRAMP_ACK	19	Target position	Int32	Encoder ticks.	
ISTEP_ACK	21	Target position Time	Int32 UInt16	Encoder ticks. ms.	
IVEL_ACK	22	Velocity	Int32	Encoder ticks/s.	
ICUR_ACK	23	Current	Int16	Digits.	

position Actual position. Unit and data type according to the commanded values (rad resp. m or Encoder ticks. Length 4 Byte).
state Short module state (Length 1 Byte)
dio Digital IO state (Length 1 Byte)

[*from Version v4630]

4.5.1 Short state in Acknowledge

The short module state transfered in the acknowledge of a motion command is based on the full module state word.

Value	Define	Corresponding flag in long status word
0x01	SHORT_NOT_OK	(STATE_ERROR) OR (NOT STATE_HOME_OK) OR (STATE_HALTED)
0x02	SHORT_SWR	STATE_SWR
0x04	SHORT_SW1	STATE_SW1
0x08	SHORT_SW2	STATE_SW2
0x10	SHORT_MOTION	STATE_MOTION
0x20	SHORT_RAMP_END	STATE_RAMP_END
0x40	SHORT_INPROGRESS	STATE_INPROGRESS
0x80	SHORT_FULLBUFFER	STATE_FULLBUFFER

More details on the full module state word you can find in the section "Observing the module state".

4.5.2 Digital IO state in Acknowledge

The last data byte of the acknowledge to a motion command contains the digital IO state. The byte is organised like this:

Value	Define	Remark
0x01	INBIT0	State of input 0
0x02	INBIT1	State of input 1
0x04	INBIT2	State of input 2
0x08	INBIT3	State of input 3

Value	Define	Remark
0x10	OUTBIT0	State of output 0
0x20	OUTBIT1	State of output 1
0x40	OUTBIT2	State of output 2
0x80	OUTBIT3	State of output 3

4.6 PowerCube Parameter ID's

This section describes the colored part of the data frame:

Identifier	CommandID	**ParameterID** resp. MotionID	Data 0 ... Data 7

This table contains all module parameters available to the user:

Parameter	ParameterID (Dec/Hex)		Remark	Data type	Read	Write
DefHomeOffset	0	0x00	Offset to the Home position (Default value). By means of this parameter teh user can adjust an offset to the physical zero position (home) of the drive.	float	x	
DefGearRatio	1	0x01	Gear ratio (Default value)	float	x	
DefLinRatio	2	0x02	Transmission factor for transforming rotary in linear motion (Default value)	float	x	
DefMinPos	3	0x03	Minimum drive position (Default value)	float	x	
DefMaxPos	4	0x04	Maximum drive position (Default value)	float	x	
DefMaxDeltaPos	5	0x05	Maximum following error (tow) for the digital servo filter (Default value)	float	x	
DefMaxDeltaVel	6	0x06	Maximum following error for velocity control (Default value)	float	x	
DefTorqueRatio	7	0x07	Transmission factor for transforming current to torque (Default value)	float	x	
DefCurRatio	8	0x08	Transmission factor for current measurement (Default value)	float	x	
DefMinVel	9	0x09	Minimum velocity (Default value)	float	x	
DefMaxVel	10	0x0a	Maximum velocity (Default value)	float	x	
DefMinAcc	11	0x0b	Minimum acceleration (Default value)	float	x	
DefMaxAcc	12	0x0c	Maximum acceleration (Default value)	float	x	
DefMinCur	13	0x0d	Minimum current (Default value)	float	x	
DefMaxCur	14	0x0e	Maximum current (Default value)	float	x	
DefHomeVel	15	0x0f	Homing velocity. This value is signed and thereby specifies the homing direction. (Default value)	float	x	
DefHomeAcc	16	0x10	Homing acceleration (Default value)	float	x	
DefCubeSerial	26	0x1a	Serial number of the PowerCube (Default value)	UInt32	x	
DefConfig	27	0x1b	Config word (Default value)	UInt32	x	
DefPulsesPerTurn	28	0x1c	Number of Encoder ticks per revolution (Default value)	UInt32	x	
DefCubeVersion	29	0x1d	Version information (Default value)	UInt16	x	
DefServiceInterval	30	0x1e	Service interval (Default value)	UInt16	x	
DefBrakeTimeOut	31	0x1f	Delay for releasing the brake in ms (Default value)	UInt16	x	
DefAddress	32	0x20	Module bus address [1...31] (Default value)	UInt8	x	
DefPrimBaud	34	0x22	Primary Baud rate setting (Default value)	UInt8	x	
DefScndBaud	35	0x23	Secondary Baud rate setting (Default value)	UInt8	x	
PosCount	36	0x24	Absolute Counter value (Actual value)	Int32	x	
RefPosCount	37	0x25	Absolute Counter value at Homing position (Actual value)	Int32	x	
DioSetup	38	0x26	Digital IO word	UInt 32	x	x
CubeState	39	0x27	Module State word (Actual value)	UInt32	x	
TargetPosInc	40	0x28	Target position in Encoder ticks (Target value)	UInt32	x	x
TargetVelInc	41	0x29	Target velocity in Encoder ticks/s (Target value)	UInt32	x	x
TargetAccInc	42	0x2a	Target accedleration in Encoder ticks/s² (Target value)	UInt32	x	x
StepInc	43	0x2b	Step mode target position in Encoder ticks (Actual value)	UInt32	x	

Parameter	ParameterID (Dec/Hex)		Remark	Data type	Read	Write
HomeOffsetInc	44	0x2c	Home offset in Encoder ticks (Actual value)	Int32	x	
RawCur	53	0x35	Commanded Current in Digits [-500...+500] (Actual value)	Int16	x	x
HomeToZeroInc	54	0x36	Number of Encoder ticks between home switch and Encoder index (Actual value)	Int32	x	
Config	57	0x39	Config word (Vorgabe)	UInt32	x	x
MoveMode	58	0x3a	Motion mode (Actual value)	UInt8	x	
IncRatio	59	0x3b	Ration of Encoder ticks and unit, rad resp. m (Actual value)	float	x	
ActPos	60	0x3c	Actual position in rad resp. m (Actual value)	float	x	
ActPos_	61	0x3d	Previous position in rad resp. m (Actual value)	float	x	
IPolPos	62	0x3e	Actual interpolated position (Actual value)	float	x	
DeltaPos	63	0x3f	Actual following error (Actual value)	float	x	
MaxDeltaPos	64	0x40	Maximum following error (Limit)	float	x	x
ActVel	65	0x41	Actual velocity in units/s (Actual value)	float	x	
IPolVel	66	0x42	Actual interpolated velocity in units/s (Actual value)	float	x	
MinPos	69	0x45	Minimum position (Limit)	float	x	x
MaxPos	70	0x46	Maximum position (Limit)	float	x	x
MaxVel	72	0x48	Maximum velocity in units/s (Limit)	float	x	x
MaxAcc	74	0x4a	Maximum acceleration in units/s² (Limit)	float	x	x
MaxCur	76	0x4c	Maximum Current (Limit)	float	x	x
Cur	77	0x4d	Actual current (Actual value)	float	x	x
TargetPos	78	0x4e	Target position in units/s (Target value)	float		x
TargetVel	79	0x4f	Target velocity in units/s (Target value)	float		x
TargetAcc	80	0x50	Target acceleration in units/s² (Target value)	float		x
DefC0	81	0x51	Servo loop gain C0 (Default value)	Int16	x	
DefDamp	82	0x52	Servo loop damping (Default value)	Int16	x	
DefA0	83	0x53	Servo loop parameter A0 (Default value)	Int16		
ActC0	84	0x54	Servo loop gain C0 (Actual value)	Int16	x	x
ActDamp	85	0x85	Servo loop damping (Actual value)	Int16	x	x
ActA0	86	0x86	Servo loop parameter A0 (Actual value)	Int16	x	x
DefBurnCount	87	0x87	Number of flash downloads (Default value)	UInt8	x	
Setup	88	0x88	Setup word (Default value)	UInt32	x	
HomeOffset	89	0x89	Home offset (Actual value)	float	x	x
ActIPos	90	0x5a	Actual position in in encoder ticks	Int32	x	
ActIMaxDeltaPos	91	0x5b	Maximum allowed difference between commanded and actual position, in encoder ticks = tow distance	Int32	x	x
ActIMinPos	92	0x5c	Minimum allowed Position in encoder ticks	Int32	x	x
ActIMaxPos	93	0x5d	Maximum allowed Position in encoder ticks	Int32	x	x
ActIMaxVel	94	0x5e	Maximum allowed speed in encoder ticks/s	Int32	x	x
ActIMaxAcc	95	0x5f	Maximum allowed acceleration in encoder ticks/s²	Int32	x	x
ActIVel	96	0x60	Actual speed in encoder ticks/s	Int32	x	
ActIDeltaPos	97	0x61	Actual tow distance in encoder ticks	Int32	x	
ActFPosStateDio	98	0x62	Returns 3 Values: Actual Position, Short state, I/O-State	float/UInt8	x	
ActFSavedPos	99	0x63	Last upon broadcast saved position	float	x	
ActFHomeVel	100	0x64	Actual homing speed	float	x	
ActIHomeVel	101	0x65	Actual homing speed in encoder ticks/s	Int32	x	
ActSyncTime	102	0x66	Actual SYNC time	UInt16	x	x
ActIIPolVel	106	0x6A	Actual calculated speed in encoder ticks/s	Int32	x	
ActRawMotorCurrent	108	0x6C	Actual DC Bus Current in Digits	UInt16	x	
ActRawMotorSupply	109	0x6D	Actual DC Bus Voltage in Digits	UInt16	x	
ActRawTemp	110	0x6E	Actual temperature inside housing in Digits	Int16	x	
ActRawLogicSupply	111	0x6F	Actual Logiv voltage in Digits	UInt16	x	
ActFMotorCurrent	112	0x70	Actual DC Bus Current in A	float	x	
ActFMotorSupply	113	0x71	Actual DC Bus Voltage in V	float	x	
ActFTemp	114	0x72	Actual temperature in housing in °C	float	x	
ActFLogicSupply	115	0x73	Actual logic supply voltage in V	float	x	

Parameter	ParameterID (Dec/Hex)		Remark	Data type	Read	Write
ActMinLogicSupply	116	0x74	Minmum allowed logic voltage in V	float	x	x
ActMaxLogicSupply	117	0x75	Maximum allowed logic voltage in V	float	x	x
ActMinMotorSupply	118	0x76	Minimum allowed motor voltage in V	float	x	x
ActMaxMotorSupply	119	0x77	Maximum allowed motor voltage in V	float	x	x
ActLogicUndershootTime	122	0x7A	Maximum allowed undershoot time for logic voltage in ms	UInt32	x	x
ActLogicOvershootTime	123	0x7B	Maximum allowed overshoot time for logic voltage in ms	UInt32	x	x
ActMotorUndershootTime	124	0x7C	Maximum allowed undershoot time for motor voltage in ms	UInt32	x	x
ActMotorOvershootTime	125	0x7D	Maximum allowed overshoot time for motor voltage in ms	UInt32	x	x
ActMaxTemp	132	0x84	Maximum allowed temperature inside housing	float	x	x
ActMinTemp	133	0x85	Minimum allowed temperature inside housing	float	x	x
ActFPosTime	161	0xA1	Returns 2 Values: Actual position and time stamp [ms]	float/UInt16	x	
ActFScanPosFallEdge	162	0xA2	Last upon rising edge on input SW3 saved Position	float	x	
ActFScanPosRisgEdge	163	0xA3	Last upon falling edge on input SW3 saved Position	float	x	

[From Version v351D]
[From Version v4634]

Default value: These parameters are stored in the modules ROM (Read Only Memory) and are used to initialize the modules variables.

Limit: These parameters can be set for the duration of power on. After power has been cut off, the changes are lost and have to be redone after power on.

Target: These are the target values for motion commands. They only concern target position, velocity and acceleration.

The 32-Bit configuration words Config and Setup contain the settings for brake control, communication, feedback and limit switches. Setup is read only, while Config can be read and written. The section "module configuration" describes the flags of the Config word.

5 Observing the module state (ParameterID 39)

The module state can be permanently observed using the "CubeState" word. By means of the state word the user receives error messages as well as information on the execution state of motion commands. It is recommended to poll the state word on a regular basis. The application controls the frequency for updating the data. The state word is an unsigned integer value where each bit is treated as a flag. More than one flag can be set at a time:

Flag	Bit	Value	Meaning
STATE_HOME_OK	1	0x00000002	This flag is set after a successful homing procedure. It means that the drive has successfully found its zero position. All limitations for the operation range are valid now. If the user sends another Home-Command the flag will be reset until the homing procedure has been finished successfully.
STATE_HALTED	2	0x00000004	This flag is set in conjunction with an emergency stop. It means that the cube is in a secure state, not moving and not accepting motion commands. Only after a reset command which resets this flag, the module will return to the normal operation mode. An emergency stop can be caused automatically by the module in case of an error or by the user when sending a Halt command.
STATE_SWR	6	0x00000040	This flag shows the state of the home switch. Flag set means home switch is active, This is no error flag.
STATE_SW1	7	0x00000080	This flag shows the state of the Limit switch 1. Flag set means limit switch 1 is active. This is no error flag.
STATE_SW2	8	0x00000100	This flag shows the state of the Limit switch 2. Flag set means limit switch 2 is active. This is no error flag.
STATE_BRAKEACTIVE	9	0x00000200	This flag shows the state of the brake. Flag set means brake is active and servo loop is open. This is no error flag and it is used only if a brake is installed.
STATE_CURLIMIT	10	0x00000400	This flag is a warning of the servo loop. It has reached the maximum current output. The drive is working at its limits. This flag can be reset by the Reset command. It is no error flag

Flag	Bit	Value	Meaning
STATE_MOTION	11	0x00000800	This flag indicates the drive is in motion. It is set and reset automatically.
STATE_RAMP_ACC	12	0x00001000	This flag indicates the drive is in acceleration when controlled by ramp motion commands. It is automatically reset when the ramp motion profile has ended.
STATE_RAMP_STEADY	13	0x00002000	This flag indicates the drive is moving at constant speed when controlled by ramp motion commands. It is automatically reset when the ramp motion profile has ended.
STATE_RAMP_DEC	14	0x00004000	This flag indicates the drive is in deceleration when controlled by ramp motion commands. It is automatically reset when the ramp motion profile has ended.
STATE_RAMP_END	15	0x00008000	This flag indicates the end of a ramp motion profile. The drive is not moving.
STATE_INPROGRESS	16	0x00010000	This flag is only used in Step motion control. It indicates a Step motion command is in progress.
STATE_FULLBUFFER	17	0x00020000	This flag is only used in Step motion control. It indicates a Step motion command was pushed to the command stack. This happens when the module receives a Step motion command while STATE_INPROGRESS is set. Upon completion of the currently executed step command, the buffered one will automatically be executed.
STATE_ERROR	0	0x00000001	An error occured. The module stop immediately and does not accept motion commands anymore. The reason for the error state can be found reading the error flags. In many cases the error state can be reset by the user sending a Reset command. After a successful Reset the module is ready again to accept motion commands.
STATE_POWERFAULT	3	0x00000008	This flag defines an error of the servo amplifier. This flag si set in conjuction with STATE_ERROR. In most cases the module needs to be switched off to reset this error. One of the flags 18 through 23 will be set to explain the cause.
STATE_TOW_ERROR	4	0x00000010	Tow error: The servo loop was not able to follow the target position within the given limit. The maximum tow can be adjsuted using the parameter „MaxDeltaPos". Check if the module was overloaded.
STATE_COMM_ERROR	5	0x00000020	This error flag is raised if the watchdog has been enabled only. When enabled the watchdog must be refreshed in a given period of time by the external control. If the external control fails to do so, the drive will follow the emergency Stopp routine and enter an error state.
STATE_POW_VOLT_ERR	18	0x00040000	This flag is set in conjunction with STATE_POWERFAULT. It indicates a voltage drop or an overvoltage occurred in the motor supply. This error can be reset after the normal voltage level has been restored. Check your power supply.
STATE_POW_FET_TEMP	19	0x00080000	This flag is set in conjunction with STATE_POWERFAULT. The power transistors have overheated and the servo loop has been disabled. Power must be switched of to reset this error. It is due to overload or too high ambient temperature.
STATE_POW_INTEGRAL-ERR	23	0x00800000	This flag is set in conjunction with STATE_POWERFAULT. The drive has been overloaded and the servo loop has been disabled. Power must be switched off to reset this error. Check your apllication and the load situations of the drive.
STATE_BEYOND_HARD	25	0x02000000	This flag indicates the module has reached the hard limit. An emergency stop has been executed automatically. To remove the module from this position you need to follow the procedure described in "PowerCube™ Operation System: Disorder".
STATE_BEYOND_SOFT	26	0x04000000	This flag indicates the module has reached the soft limit. An emergency stop has been executed automatically. This flag can be reset by a Reset command.
STATE_LOGIC_VOLT	27	0x08000000	The voltage of the logic power supply has either dropped or an overvoltage occured. The drive will be disabled. This error can be reset.
STATE_POW_WDG_TEMP	20	0x00100000	This flag is set in conjunction with STATE_POWERFAULT. The motor has overheated and the servo loop has been disabled. Power must be switched of to reset this error. It is due to overload or too high ambient temperature.
STATE_POW_SHORTCUR	21	0x00200000	This flag is set in conjunction with STATE_POWERFAULT. A short curcuit occured. The servo loop has been disabled. The power must be switched of to reset this error. The module has been overlaoded. If this error cannot be reset consult your service partner.
STATE_POW_HALLERR	22	0x00400000	This flag is set in conjunction with STATE_POWERFAULT. An error occured in reading the hall effect sensors of the motor. The motor has been overheated. Power must be switched off to reset this error.
STATE_CPU_OVERLOAD	24	0x01000000	Communication breakdown between CPU and current controller. Power must be switched off. Please consult your service partner.
STATE_POW_SETUP_ERR	27	0x08000000	Error in initializing the current controller. Module settings disaccord with controller configuration (5A/10A types). Power must be switched off. Please consult your service partner. Available from version 3.5.14 through 3.5.1D.

[These flags describe an error status]
[These flags provide useful information on the module status]
[These flags are obsolete]

6 Reading and Writing Digital IO (ParameterID 38)

According to the module configuration it is possible to set and read binary signals. The parameter DioSetup (ParameterID 38) is used for this purpose. The unsigned 4 Byte-Integer-value has this structure:

Wert	Define	Remark
0x00000001	DIOID_MOD_INBIT0	State of input 0.
0x00000002	DIOID_MOD_INBIT1	State of input 1.
0x00000004	DIOID_MOD_INBIT2	State of input 2.
0x00000008	DIOID_MOD_INBIT3	State of input 3.
0x00000010	DIOID_MOD_OUTBIT0	State of output 0.
0x00000020	DIOID_MOD_OUTBIT1	State of output 1.
0x00000040	DIOID_MOD_OUTBIT2	State of output 2.
0x00000080	DIOID_MOD_OUTBIT3	State of output 3.
0x00000100	DIOID_MOD_INSWR	State of Home switch (1 = active).
0x00000200	DIOID_MOD_INSW1	State of Limit switch 1 (1 = active).
0x00000400	DIOID_MOD_INSW2	State of Limit switch 2 (1 = active).

All other flags have no meaning and are 0.

7 Altering the module configuration (ParameterID 57)

7.1 Configuration flags in 32-Bit Word Setup (Read Only)

The Config word Setup (ParameterID 88) is read only and used to provide information on the module configuration. If you wish to change the module setup please contact your service partner.

Value	Define	Remark
0x00000001L	SETUPID_MOD_ENCODER_FEEDBACK	not used
0x00000002L	SETUPID_MOD_RESOLVER_FEEDBACK	not used
0x00000004L	SETUPID_MOD_ABSOLUTE_FEEDBACK	not used
0x00000008L	SETUPID_MOD_4IN_4OUT	1 = The 15pole connector is configured for 4 I/O-Signals.
0x00000010L	SETUPID_MOD_3IN_ENCODER_IN	1 = The 15pole connector is configured for Encoder input.
0x00000020L	SETUPID_MOD_3IN_ENCODER_OUT	1 = The 15pole connector is configured for Encoder output.
0x00000040L	SETUPID_MOD_RS232	1 = RS232 communication is active.
0x00000200L	SETUPID_MOD_CAN	1 = CAN communication is active.
0x00000400L	SETUPID_MOD_PROFIBUS	1 = Profibus communication is active.
0x00000800L	SETUPID_MOD_USE_M3ID	1 = CAN identifiers for MoRSE3 are active.
0x00001000L	SETUPID_MOD_USE_M4ID	1 = CAN identifiers for MoRSE4 are active.
0x00002000L	SETUPID_MOD_USE_CANOPEN	1 = The CANopen interface is active.
0x00008000L	SETUPID_MOD_USE_SW2_AS_ENABLE	1 = Input for Limit switch 2 is used as an Enable signal.
0x00010000L	SETUPID_MOD_USE_SW2_AS_BRAKE	1 = Input for Limit switch 2 is used as Release brake signal.
0x00020000L	SETUPID_MOD_ERROR_TO_OUT0	1 = An error will be signalized on output 0.

7.2 Configuration flags in 32-Bit Word Config

The Config word (ParameterID 57) can be read and written. After power on this parameter can be altered to change the module configuration. This parameter is available from module version 3.5.00.

Value	Define	Remark

Value	Define	Remark
0x00000008L	CONFIGID_MOD_BRAKE_PRESENT	1 = Brake is present
0x00000010L	CONFIGID_MOD_BRAKE_AT_POWERON	0 = Brake is released on power on
0x00000020L	CONFIGID_MOD_SWR_WITH_ENCODERZERO	1 = Encoderindex signal is used in homing procedure
0x00000040L	CONFIGID_MOD_SWR_AT_FALLING_EDGE	1 = Homing on falling edge of limit switch
0x00000080L	CONFIGID_MOD_CHANGE_SWR_TO_LIMIT	1 = Home switch is limit switch (except during Homing)
0x00000100L	CONFIGID_MOD_SWR_ENABLED	1 = Home switch is enabled
0x00000200L	CONFIGID_MOD_SWR_LOW_ACTIVE	1 = Home switch is low active
0x00000400L	CONFIGID_MOD_SWR_USE_EXTERNAL	1 = External home switch is used
0x00000800L	CONFIGID_MOD_SW1_ENABLED	1 = Limit switch 1 is enabled
0x00001000L	CONFIGID_MOD_SW1_LOW_ACTIVE	1 = Limit switch 1 is low active
0x00002000L	CONFIGID_MOD_SW1_USE_EXTERNAL	1 = External limit switch 1 is used
0x00004000L	CONFIGID_MOD_SW2_ENABLED	1 = Limit switch 2 is enabled
0x00008000L	CONFIGID_MOD_SW2_LOW_ACTIVE	1 = Limit switch 2 is low active
0x00010000L	CONFIGID_MOD_SW2_USE_EXTERNAL	1 = External Limit switch 2 is used
0x00020000L	CONFIGID_MOD_LINEAR	1 = Module is a linear type
0x00080000L	CONFIGID_MOD_ALLOW_FULL_CUR	0 = Commanded current (PCube_moveCur) is limited to nominal current.
0x00100000L	CONFIGID_MOD_M3_COMPATIBLE	1 = Module is M3 compatible. This concerns CAN communication and behaviour at PCube_moveStep. Module does not accept motion commands unless successfully homed.
0x00200000L	CONFIGID_MOD_LINEAR_SCREW	1 = Module is linear, driven by ball screw.
0x00800000L	CONFIGID_MOD_DISABLE_ON_HALT	1 = Motor power disabled In case of error
0x01000000L	CONFIGID_MOD_WATCHDOG_ENABLE	1 = Watchdog is enabled. Activated automatically by the first "life sign" of the host control.
0x02000000L	CONFIGID_MOD_ZERO_MOVE_AFTER_HOK	1 = After Homing the module moves to zero
0x04000000L	CONFIGID_MOD_DISABLE_ACK	1 = All acknowledge messages are disabled. All Get commands will still be acknowledged. Valid only for CAN-Bus.
0x08000000L	CONFIGID_MOD_SYNC_MOTION	1 = Sychronized motion commands enabled. After sending a motion command the drive expects the broadcast PCube_startMotionAll to start motion. Valid only for CAN-Bus.

8 Examples for CAN-Bus communication

A few examples follow to introduce CAN-Bus communication. These commands are sent to the module (the module address used is 17):

- Home.
- Reset.
- Ramp motion profile command with short and extended acknowledge).
- Retrieve position.
- Retrieve module state.

Home	CAN-ID	DLC	Byte 0 ComandID	Byte 1 Param.ID	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Send	0x0F1	1	0x01	-	-	-	-	-	-	-
Ack	0x0B1	1	0x01	-	-	-	-	-	-	-

Reset	CAN-ID	DLC	Byte 0 ComandID	Byte 1 Param.ID	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Send	0x0F1	1	0x00	-	-	-	-	-	-	-
Ack	0x0B1	1	0x00	-	-	-	-	-	-	-

Set Ramp motion acceleration	CAN-ID	DLC	Byte 0 ComandID	Byte 1 Param.ID	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Send	0x0F1	6	0x08	0x50	acc (float)				-	-

Set Ramp motion acceleration	CAN-ID	DLC	Byte 0 ComandID	Byte 1 Param.ID	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Ack	0x0B1	3	0x08	0x50	0x64	-	-	-	-	-

Set Ramp motion velocity	CAN-ID	DLC	Byte 0 ComandID	Byte 1 Param.ID	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Send	0x0F1	6	0x08	0x4f	vel (float)				-	-
Ack	0x0B1	3	0x08	0x4f	0x64	-	-	-	-	-

Ramp motion command	CAN-ID	DLC	Byte 0 ComandID	Byte 1 Param.ID	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Send	0x0F1	6	0x0b	0x04	pos (float)				-	-
Ack (einfach)	0x0B1	3	0x0b	0x04	0x64	-	-	-	-	-

Retrieve module state	CAN-ID	DLC	Byte 0 ComandID	Byte 1 Param.ID	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Send	0x0D1	2	0x0a	0x27	-	-	-	-	-	-
Ack	0x0B1	6	0x0a	0x27	state (UInt32)				-	-

Retrieve module position	CAN-ID	DLC	Byte 0 ComandID	Byte 1 Param.ID	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Send	0x0D1	2	0x0a	0x3c	-	-	-	-	-	-
Ack	0x0B1	6	0x0a	0x3c	pos (float)				-	-

When using the extended acknowledge to motion commands the module will answer the command by sending actual position, module state and digital IO state. Motion command and acknowledge according to the above example would look like this:

Ramp motion command with ext. Acknowledge	CAN-ID	DLC	Byte 0 ComandID	Byte 1 Param.ID	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Send	0x0f1	6	0x0b	0x0e	pos (float)				-	-
Ack (extended)	0x0f1	8	0x0b	0x0e	actPos (float)				State	DIO

9 Examples for RS232 communication

A few examples follow to introduce RS232 communication. These commands are sent to the module (the module address used is 17):

- Home.
- Reset.
- Ramp motion profile command.

Home	STX	TEL-ID	TEL-ID	CmdID	Par-ID	B2	B3	B4	B5	B6	B7	BCC	ETX
Send	0x02	0x06	0x21	0x01	-	-	-	-	-	-	-	0x28	0x03
Ack	0x02	0x0a	0x21	0x01	-	-	-	-	-	-	-	0x2c	0x03

Reset	STX	TEL-ID	TEL-ID	CmdID	Par-ID	B2	B3	B4	B5	B6	B7	BCC	ETX
Send	0x02	0x06	0x21	0x00	-	-	-	-	-	-	-	0x27	0x03
Ack	0x02	0x0a	0x21	0x00	-	-	-	-	-	-	-	0x2b	0x03

Set Ramp motion acceleration	STX	TEL-ID	TEL-ID	CmdID	Par-ID	B2	B3	B4	B5	B6	B7	BCC	ETX
Send	0x02	0x06	0x26	0x08	0x50	0x00	0x00	0x00	0x3f	-	-	0xc3	0x03
Ack	0x02	0x0a	0x26	0x08	0x50	0x64	-	-	-	-	-	0xec	0x03

Set Ramp motion velocity	STX	TEL-ID	TEL-ID	CmdID	Par-ID	B2	B3	B4	B5	B6	B7	BCC	ETX
Send	0x02	0x06	0x26	0x08	0x4f	0x0a	0xd7	0xa3	0x3d	-	-	0x46	0x03
Ack	0x02	0x0a	0x26	0x08	0x4f	0x64	-	-	-	-	-	0xeb	0x03

Ramp motion command	STX	TEL-ID	TEL-ID	CmdID	Par-ID	B2	B3	B4	B5	B6	B7	BCC	ETX

Ramp motion command	STX	TEL-ID	TEL-ID	CmdID	Par-ID	B2	B3	B4	B5	B6	B7	BCC	ETX
Send	0x02	0x06	0x26	0x0b	0x04	0x00	0x00	0x40	0x3f	-	-	0xba	0x03
Ack	0x02	0x0a	0x26	0x0b	0x04	0x64	-	-	-	-	-	0xa3	0x03

This example shows how the TELID is calculated (based on a SetExtended command):

Example Send (TELID_SENDDAT, Module address = 17)

Command SetExtended	0	0	0	0	0	1	1	0	0	0	1	0	0	1	1	0
	TELID Byte1 = 0x06								TELID Byte2 = 0x26							

Example Receive (TELID_RECVDAT, Module address = 17)

Command SetExtended	0	0	0	0	1	0	1	0	0	0	1	0	0	1	1	0
	TELID Byte1 = 0x0a								TELID Byte2 = 0x26							

The command Halt (CommandID: 0x02) shows how the DLE correction is done:

Send w/o correction:	STX	TELID	TELID	CMD	BCC	ETX
	0x02	0x06	0x21	0x02	0x29	0x03

Senden w/ correction:	STX	TELID	TELID	CMD (w/ DLE)	BCC	ETX
	0x02	0x06	0x21	0x10 0x82	0x29	0x03

10 Examples for Profibus-DP communication

A few examples follow to introduce RS232 communication. These commands are sent to the module:

- Home.
- Reset.
- Ramp motion profile command.

Home	Byte 0 CmdID	Byte 1 ParID	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	
Send	0x01	-	-	-	-	-	-	X	
Ack	0x01	-	-	-	-	-	-	-	Byte 8-15

Reset	Byte 0 CmdID	Byte 1 ParID	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	
Send	0x00	-	-	-	-	-	-	X	
Ack	0x00	-	-	-	-	-	-	-	Byte 8-15

Set Ramp motion acceleration	Byte 0 CmdID	Byte 1 ParID	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	
Send	0x08	0x50	acc (float)				-	X	
Ack	0x08	0x50	0x64	-	-	-	-	-	Byte 8-15

Set Ramp motion velocity	Byte 0 CmdID	Byte 1 ParID	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	
Send	0x08	0x4f	vel (float)				-	X	
Ack	0x08	0x4f	0x64	-	-	-	-	-	Byte 8-15

Ramp motion command	Byte 0 CmdID	Byte 1 ParID	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	
Send	0x0b	0x04	pos (float)				-	X	
Ack	0x0b	0x04	0x64	-	-	-	-	-	Byte 8-15

X: When sending data to the module the value in 7 has to change with every command sent, e. g. an incremental counter. The reason is that only the first of all identical messages received by the module will be interpreted (e.g. retrieving module state)!

Byte 8-15: When using Profibus-DP communication the command passed to the module has a length of 8 Byte while the module always acknowledges 16 Byte to the Master:

Byte 8	Byte 9	Byte 10	Byte 11	Byte 12	Byte 13	Byte 14	Byte 15
-	Actual position in Encoder ticks (Int32 = 4 Byte)				Short state (UInt16 = 2 Byte)		time [ms] (0..255)

11 PowerCube Operation

For a trouble free operation of the PowerCube a few rules should be mentioned. They are especially useful for commanding motion and system supervision. The type of host control depends on the application it is ought to run. You should consider these questions when plannig your application:

- Realtime operation and time critical sections in the process
- Daisy chaining several drices in a kinematic structure
- Security problems in operation

In every case the master control for PowerCube can be optimized for the specific application. Here is some information on this topic.

12 Normal operation

After power on the PowerCube™ is immediately ready for operation. The returned position is the default home offset. The drive can execute motion commands.
The modules are mechatronical systems with grease lubricated mechanical parts. Maximum acceleration and speed are available only after a complete warm up of the module.
Typically the first motion command after power on is a homing procedure. This is necessary to fix the coordinate system of the module. Maximum and minmum position of the module are adjusted in this coordinate system. This means that a module, once it is homed regularly, will only accept target positions within its normal range of operation. Please take care of this:

- Run motion commands only after a successful homing,
- If a homing is not possible, run the module only at slow speed. This enables you to stop the module in case before an emergency situation occures,
- Always move the the module back to its home position before power down.
- The status word CubeState shows the flag STATE_HOME_OK after a successful homing procedure.

Commands sent to the PowerCube™ are checked by the cubes operation system. This means:

- Automatic limitation of target position, velocity, acceleration and current when runnig motion commands. The targets are limited to the preadjusted maximum values (see PowerCube Parameter ID's ": Targets).
- Limitation of commanded values. This concerns all values that can be changed while the cube is online (see PowerCube Parameter ID's": Commands)

13 Using the Modules Watchdog

Every module connected must have the Watchdog enabled in the Config word (Flag CONFIG_WATCHDOG_ENABLE). Send the command CANID_CMDPUT (0x0e0 + Modul-ID) + CommandID 0x07 to each module in order to sitch the watchdog to the online state. Sending this command again will switch the watchdog to the offline state.
In an interval smaller than 50 ms send this broadcast command to refresh the watchdog:

CANID_CMDALL (0x100) + CommandID 0x07

If the watchdog refresh command has not been received within the given interval, all modules will stop and the flags STATE_COMM_ERROR and STATE_HALTED are raised.

To restart operation you can either:
- switch the watchdog offline in each module and send a Reset command or
- restart sending cyclic Refresh commands and broadcast a Reset to all modules connected

14 Trouble shooting

All commands sent to the module are checked by the operation system. This means:
Errors that occur while operation will be displayed as flags in the modules state. There is a difference between resetable and nonresetable errors. Once a nonresetable error occures the power must be switched off.
These errors are not resetable:
- Temperature problems. Caused by overload or too high ambient temperature. Check your application.
- Short in the servo amplifier, problems with hall effect sensors. Total overload. Consult your service partner.

These errors can be reset by command:
- following error (tow).
- running over soft or hard limits of the operation range.
- error in data exchange (communication).
- voltage drop in power supply.