FABRIKAUTOMATION

PEPPERL+FUCHS

Series CVE 10, CVM 10
Absolute Encoders for the CAN-Bus

Singleturn, Multiturn with CAN-Interface according to ISO/DIS 11898.

Date 04.12.1995

Subject to reasonable modifications due to technical advances.

Copyright by Pepperl+Fuchs, Printed in Germany

Pepperl+Fuchs GmbH D-68301 Mannheim Phone (0621) 7 76-0 Fax 0621 7 76-10 00 Internet http://www.pepperl-fuchs.com

Subject to reasonable modifications due to technical advances. Copyright by Pepperl+Fuchs, Printed in Germany

I. General Description

The serial Bus System CAN (Controller Area Network) was developed by BOSCH. It was especially designed to be used in automotive systems, such as lorries, busses or building site machines. The CAN-System is a multi-master network, i.e. more than one master within the network is possible. The CAN-system is standardized in accordance to the ISO/DIS 11898 and meets the 1st and 2nd layer of the ISO/OSI reference model. From the electrical point of view it can be compared with the RS 485 interface, since for the CAN-system, as well as for the RS 485 interface, a differential signal is used and transmitted via a drilled twin-wire cable. However, the RS 485 interface doesn´t support the bus arbitration as used with the CAN-system.

In CAN networks the messages are not addressed to a certain participant. The messages are sent to all participants in the network who then check the comprised identifier. Based on the received identifier the participant then decides whether to read the message or to ignore it. The identifier also comprises the priority of the message. With more than one message at a time to get on the bus the one with the highest priority is allowed to be transmitted first. Other bus-systems, such as the Pepperl+Fuchs SLIN-bus or Profibus DP address their messages directly to a certain participant.

An outstanding characteristic of the CAN-protocol is its high transmission safety with a Hamming distance of 6. In case errors occur anyway the CAN controller recognizes where the error occurs and - if necessary - deactivates the faulty participant.

The maximum data transmission rate is specified with 1 MBaud with a network expansion of max. 40m. With a network expansion of up to 500m the max. transmission rate is specified with 125 KBaud (50 KBaud with up to 1000m).

The so far known CAN variations are BasicCAN and FullCAN (CAN specification 2.0 part A). Recently specified was the ExtendedCAN (CAN specification 2.0 part B). The CAN controller used for the encoders is the SAE 80C91 by Siemens which meets the CAN specification 2.0 part B passive. This means, that the encoders can be used within a network comprising ExtendedCAN participants. The passive components can´t process ExtendedCAN messages, but they don´t reply with an error message.

The theoretically possible number of participants within a FullCAN network is 2032.

Subject to reasonable modifications due to technical advances. Copyright by Pepperl+Fuchs, Printed in Germany

II. Hardware Description

1. Overview

1.1 CE-Sign

The absolute encoders CVE10 and CVM10 meet the specification of the EN 55011/03.91 (EMC disturbance emission) and the EN 50082-2/03.95 (EMC disturbance resistance). Therefore the requirements to carry the CE-sign are fulfilled.

1.2 Block Diagram of a complete System

The absolute encoders give a certain position value with each measuring step. This position value is given to the CAN-bus in binary code. Two operation modes are possible, one is the polling mode, i.e. the position value is requested by a participating master, the other one the cycle mode, i.e. the encoder itself sends its position values on a certain time base (lowest possible time base 1ms).

The micro controller is a 8051 derivate by Intel. It reads and writes the data to and from the EEPROM, processes the data from the optical detection unit and transmits it to the CAN interface.

The required CAN-ID´s and the baud rate are stored in the EEPROM. If required, they can be changed.

The CAN controller meets the specifications of the FullCAN standard (80C91 by Siemens). It comprises 16 storage units that can store one CAN message each. They are used for data transmission as well as for data reception. For data reception a identifier-mask is registered in the storage units. Only messages which exactly match these identifier-masks are accepted. Thus the storage units also perform the filtering of the messages.

Additionally one of the storage units can be used to receive all messages independent from their identifier (BasisCAN characteristic).

The physical data wires are driven with the CAN Tranceiver PCA82C250 by Phillips, which meets the CIA standard ISO/DIS 11898. The proposal for the plug-in connector is a 9-pin sub-D connector with a pinning in accordance to the CIA definition.

Subject to reasonable modifications due to technical advances. Copyright by Pepperl+Fuchs, Printed in Germany

1.3 Installation

The use of high-end micro electronics requires a correct and consequently executed wiring and shielding concept. This especially with increasing performance requirements and higher integration of the electronics in modern machines.

The following installation tips are suitable for normal industrial conditions. There is unfortunately no general solution for any kind of surrounding condition. Following the tips below the encoders are supposed to work properly:

- Finish the serial transmission cable with a 120 ohms resistor (between CAN+ and CAN-) at the beginning and the end of the serial network (e.g. control unit and last encoder)
- All encoder cables must be installed with as much distance as possible from EMC disturbance causing cables, such as high voltage cables, motor cables etc.
- Data cable cross section minimum 0.14 mm^2.
- Use drilled and shielded twin-wire cables.
- Cross section of the shield not less than 4 mm^2.
- Don´t bend the cable with a radius of less than 10mm. Don´t apply tearing or shearing forces to the cable.
- Best shield performance is only guaranteed when using a PG cable output instead of a plug connector.
- Always use a safety low voltage power supply.

The correct installation of the shield is a decisive factor for a interference free data transmission. Very often installation faults concerning the shield can be seen, e.g. the shield is connected at one end only and then soldered to the earth connection by means of a wire. This is correct when working with low frequencies. When dealing with EMC problems the trouble is caused by high frequency interferences.

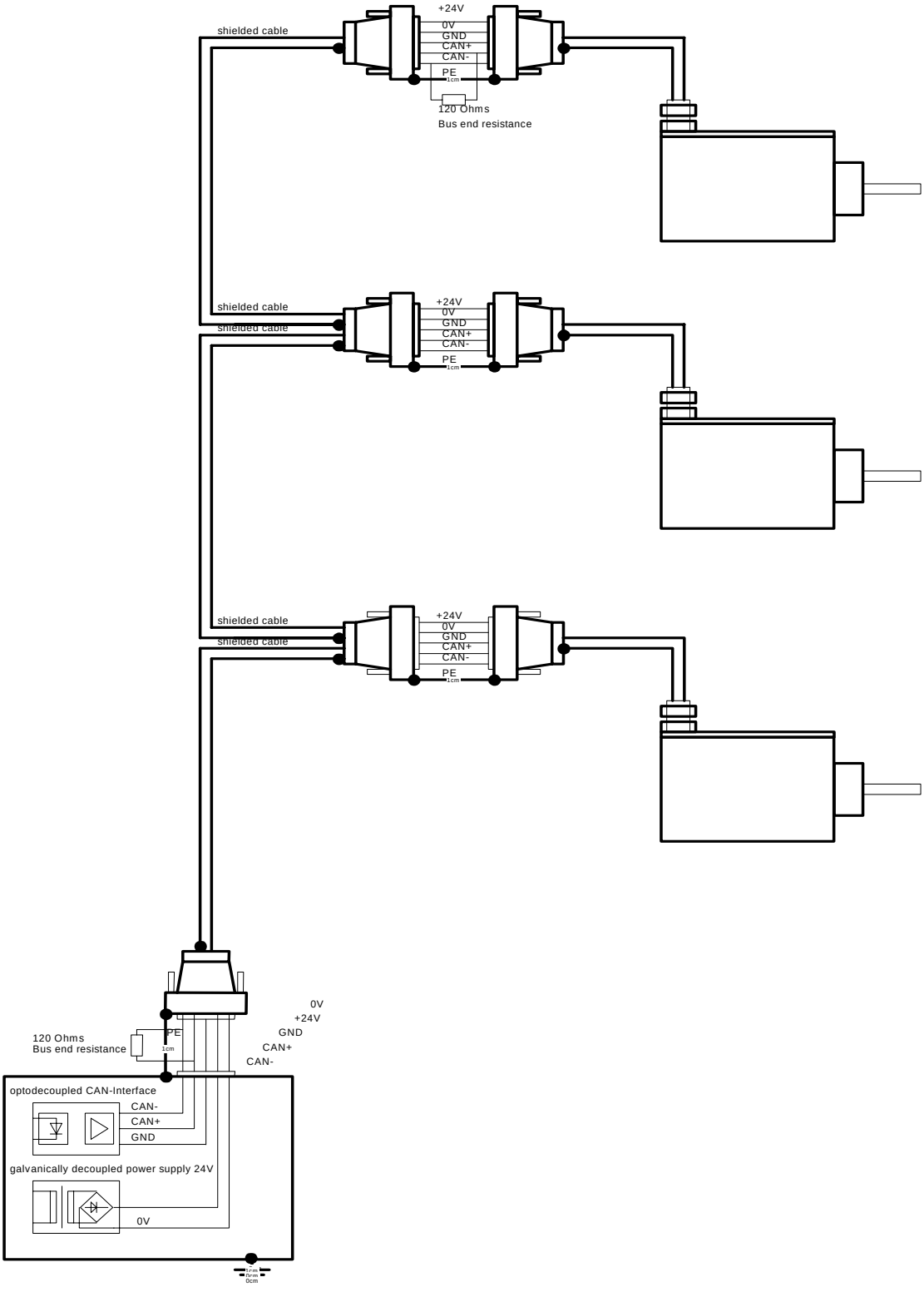
To make sure, that the HF energy will not be discharged into the cable itself, it is required to obtain a very low impedance from the shield to earth. A satisfying impedance can be obtained by connecting metal to metal on a broad area. Best shield performance will be obtained observing the following rules:

- The shield must be connected on both ends, except there is danger of earth-potential-compensation currents on the shield.
- In case there is no satisfying compensation between the earth potentials and no additional compensation wire can be installed from one end to the other the shield must be connected on one side only. However, This procedure will result in a fairly bad shield performance.
- The shield must completely be turn over the insulation and then connected to earth with as much area as possible.

Subject to reasonable modifications due to technical advances. Copyright by Pepperl+Fuchs, Printed in Germany

When using plugs, take care, that only metal or metal covered plugs are used. Take care that the shield has a satisfying connection to the housing.

Subject to reasonable modifications due to technical advances. Copyright by Pepperl+Fuchs, Printed in Germany



Subject to reasonable modifications due to technical advances. Copyright by Pepperl+Fuchs, Printed in Germany

1.4 Plug Pins

The encoder is delivered by with a 9-pin sub-D connector. The pin connection is in accordance with the CIA definition.

Pin	**connection**
1	reserved
2	CAN-
3	GND (Data ground)
4	reserved
5	reserved
6	0 volts power supply
7	CAN+
8	reserved
9	24 volts power supply

Subject to reasonable modifications due to technical advances. Copyright by Pepperl+Fuchs, Printed in Germany

III. Software Description

1. Data Format

1.1 Telegram Bytes

Three reception ID´s and three sending ID´s are assigned to each absolute encoder. The ID´s are permanently stored in the encoder and can be changed by means of special telegrams.

The lengths of the telegrams are defined in the DLC-field of the CAN telegram. The length depends on the type of telegram.

All telegrams are designed to the following scheme:

Byte	0	1	2	3	4	5	6	7
	FSC	D0/ Status	D1	D2	D3	D4	D5	D6

FSC	Function Select Code Different functions within one CAN ID are distinguished.
Status	With the telegrams Reply-Position-Update telegram, Cyclic-Position telegram und Reply-Parameter telegram in the byte D0 a status message is transmitted.
D0..D6	Data bytes

Telegrams to the encoder are allowed to consist of more data bytes then necessary. The additional data bytes are not being processed by the encoder.

Telegrams from the encoder only comprise the required number of data bytes.

Subject to reasonable modifications due to technical advances. Copyright by Pepperl+Fuchs, Printed in Germany

1.2 Status Byte

With the telegrams Reply-Position-Update telegram, Cyclic-Position telegram and Reply-Parameter telegram a status message in the data byte D0 will be transmitted. The meaning of the bits is as follows:

Bit	7	6	5	4	3	2	1	0
	Busy	SyncE	ParamE	PosE	ComE	Default ID	Cyclic Mode	Sync Mode

Busy	= 1	In case the encoder detects an possible error the busy bit is set to a 1. When the encoder data is being recognized as correct again, the busy bit is set back to a 0 automatically.
SyncE	= 1	The synchronisation time has run of without having received a new Sync telegram. The encoder stops the cyclic sending of its position values.
ParamE	= 1	Parameter command error
PosE	= 1	PosUpdate command error
ComE	= 1	Communication error detected, CAN-interface must be initialized.
DefaultID	= 1	Default-ID's and default-baudrate were used
CyclicMode	= 1	Cyclic mode is active
SyncMode	= 1	Sync Mode on

With the Set-Parameter telegram and FSC 02H 'read status' the error bits ComE, PosE, ParamE, SyncE in the status word are set back. In case the error is still active the bits are not set back.

The busy message is set back automatically as soon as the data is valid again.

Subject to reasonable modifications due to technical advances. Copyright by Pepperl+Fuchs, Printed in Germany

2. Operation Modes

2.1 Polled Mode

In the polled mode the position of the encoder is requested by means of a special telegram, the Request-Position-Update telegram. The encoder answers with the Reply-Position-Update telegram.

2.2 Cyclic Mode

By means of the Set-Parameter telegram a certain cycle time base can be set. This results in a cyclic sending of the encoder data depending on the time base. A time base of 1ms up to 65,536 ms is possible.

The cycle time can be stored in the permanent memory (FSC 21H) or in the non-permanent memory (FSC 03H). The time which is set in the permanent memory is valid after switching on the system. If a time is being written into the non-permanent memory this time becomes valid until a reset is done or the system is being switched off.

To avoid, that the drift between the control unit clock and the encoder clock becomes too high it is possible to re-synchronize the encoder clock by means of the Sync telegram in the cycle mode. In case there was no re-synchronization performed by means of the Sync telegram the error bit SyncE will be set when the time SyncTime has run off.

With the command byte the performance of the encoder in the cycle mode can be set. It can be set

- whether the encoder works in the cycle mode or not,
- whether the encoder clock should be re-synchronized or not, and whether the encoder should start in the cyclic mode directly after being switched on,
- or whether the encoder should wait for a Sync telegram and then start with the cyclic mode after the Sync-Offset-Time has run off.

2.3. Parameter Mode

By means of the Set-Parameter telegram the ID number and the data of the other modes can be set. The encoder replies with the Reply-Parameter telegram.

Subject to reasonable modifications due to technical advances. Copyright by Pepperl+Fuchs, Printed in Germany

3. Baud Rates and Types of Telegrams

The CAN absolute encoder can be operated with 9 different baud rates. The CAN controller will be initialized with the baud rate stored in the EEPROM when the system is switched on.

The following baud rates are possible:

Baud rate	Code number	
10 kBit/s	0	not yet possible
20 kBit/s	1	DEFAULT
50 kBit/s	2	
100 kBit/s	3	
125 kBit/s	4	
250 kBit/s	5	
500 kBit/s	6	
800 kBit/s	7	
1 MBit/s	8	

The electrical specification is in accordance to the ISO/DIS 11898.

The CAN absolute encoder supports 3 sending and 3 receiving telegrams.

Telegrams	Default-ID	Sending or receiving telegram
Request-Position-Update telegram	200H	R
Reply-Position-Update telegram	280H	S
Cyclic-Position telegram	180H	S
Sync telegram	100H	R
Set-Parameter telegram	300H	R
Reply-Parameter telegram	380H	S

Subject to reasonable modifications due to technical advances. Copyright by Pepperl+Fuchs, Printed in Germany

4. CAN-Communication

4.1 Request-Position-Update Telegram

The ReqPosUP-ID will be set in the EEPROM, when it is not defined, the default ReqPosUp-ID=200H will be used.

FSC	D0	D1	D2	D3	D4	D5	D6	Function
00H	------	------	------	------	------	------	------	Request position only

4.2 Reply-Position-Update Telegram

The ReplyPosUP-ID will be set in the permanent memory, when it is not defined, the default PosUP-ID=280H will be used.

When the position data requires more than one byte the least significant bits are transmitted in the first byte. The higher significant bits are transmitted in the following bytes.

Example: The position data consists of 4 bytes. Therefore the data byte 0 will be transmitted in D1, data byte 1 in D2, etc.

FSC	D0	D1	D2	D3	D4	D5	D6	Function
00H	Status	Pos 0	Pos 1	Pos 2	Pos 3	-----	-----	Position with status

Subject to reasonable modifications due to technical advances. Copyright by Pepperl+Fuchs, Printed in Germany

4.3 Cyclic-Position Telegram

The encoder cyclically sends its position on a certain time base (1 ms up to 65,536 ms). The performance of the Cyclic-Position telegram is defined by means of the command byte, the cycle time, the sync time and the sync offset time.

- The command byte is explained in chapter 4.5. The cycle time indicates the time base with which the encoder data is sent.
- The sync-time indicates the time after which the encoder expects a sync telegram (refer also to chapter 4.4). When the sync-time has run off and no sync telegram was received the encoder stops the cycle mode and the error bit will be set.
- The sync offset time is used to define the time after which the encoder starts with the cyclic mode after having received a sync telegram.

The CycPos-ID is defined in the EEPROM, if it is not defined, the default CycPos-ID =180H will be used.

FSC	D0	D1	D2	D3	D4	D5	D6	Function
30H	Status	Pos 0	Pos 1	Pos 2	Pos 3	-----	-----	Position with status

4.4 Sync Telegram

With the sync telegram the encoder clock can be re-adjusted. With the reception of the sync telegram the encoder clock will be adjusted to the next integer millisecond. The sync telegram is a broadcast telegram, e.g. the clocks of more than one encoder can be re-adjusted at the same time.

The SyncID is defined in the permanent memory, if it is not defined, the default SyncID=100H will be used.

FSC	D0	D1	D2	D3	D4	D5	D6	Function
00H	------	------	------	------	------	------	------	Request synchronization

Subject to reasonable modifications due to technical advances. Copyright by Pepperl+Fuchs, Printed in Germany

4.5 Set-Parameter Telegram

The SetParam-ID is defined in the EEPROM, if it is not defined, the default SetParam-ID=300H will be used.

A dynamic reconfiguration of the parameters should be performed in the RAM, since the number of write cycles to the EEPROM is limited.

FSC	D0	D1	D2	D3	D4	D5	D6	Function
00H	------	------	------	------	------	------	------	reserved
01H	------	------	------	------	------	------	------	Software reset
02H	------	------	------	------	------	------	------	Read status
03H	CykTime0	CykTime1	SyncTime1	SyncTime2	SyncOffset0	SyncOffset1	------	Set present cycl time in the RAM
04H	Command [1)]	------	------	------	------	------	------	Write startup-command into the RAM
11H	------	------	------	------	------	------	------	Read cycle time from permanent memory
12H	------	------	------	------	------	------	------	Read start-up command from permanent memory
13H	------	------	------	------	------	------	------	Read Request-Position-Update ID from permanent memory
14H	------	------	------	------	------	------	------	Read Reply-Position-Update ID from permanent memory
15H	------	------	------	------	------	------	------	Read Cyclic-Position ID from permanent memory
16H	------	------	------	------	------	------	------	Read Sync- ID from permanent memory
17H	------	------	------	------	------	------	------	Read Set-ParameterI ID from permanent memory
18H	------	------	------	------	------	------	------	Read Reply-Parameter ID from permanent memory
19H	------	------	------	------	------	------	------	Read baud rate from permanent memory
21H	CykTime0	CykTime1	SyncTime0	SyncTime1	SyncOffset0	SyncOffset1	-------	Write cycle time to permanent memory
22H	Command [1)]	------	------	------	------	------	------	Write start-up command to permanent memory
23H	ReqPosUpd0-ID	ReqPosUpd1-ID	------	------	------	------	------	Write Request-Position-Update ID to permanent memory
24H	ReplyPosUpd 0-ID	ReplyPosUpd 1-ID	------	------	------	------	------	Write Reply-Position-Update ID to permanent memory
25H	CyclicPos0-ID	CyclicPos1-ID	------	------	------	------	------	Write Cyclic-Position ID to permanent memory
26H	Snyc0-ID	Sync1-ID	------	------	------	------	------	Write Sync-ID to permanent memory
27H	SetPara0-ID	SetPara1-ID	------	------	------	------	------	Write Set-Parameter ID to permanent memory
28H	ReplyPara0-ID	ReplyPara1-ID	------	------	------	------	------	Write Reply-Parameter ID to permanent memory
29H	Baud rate	------	------	------	------	------	------	Write baud rate to permanent memory
3FH	------	------	------	------	------	------	------	reserved

[1)] Command: Please refer to the next page

Command Byte

Cyclic-Position-Mode on/off	Bit 0 = 1	Cyclic sending of the encoder data with the cyclic telegram, time base of sending is CycTime (refer to chapter 4.3)
	Bit 0 = 0	No cyclic sending
Sync-Mode on/off	Bit 1 = 1	Within the defined time SyncTime a Sync telegram for clock re-synchronization is expected
	Bit 1 = 0	Encoder does not expect a Sync telegram
Start after Sync telegra m on/off	Bit 2 = 1	Cyclic sending of encoder data starts after having received a Sync telegram and aditional run off of the time SyncOffsetTime
	Bit 2 = 0	Encoder is in the cycle mode directly after system switch on without receiving the Sync telegram. Can´t be used with the command byte FSC 04H, since the data is only stored in the RAM (non-permanent memory).

The following combinations of command bits may be used (other combinations are possible, but do not make sense):

Cyclic-Posbit	Syncbit	Startbit	Description
1	1	1	cyclic sending of encoder data including re-synchronisation of encoder clock start after having received the Sync telegram and SyncOffsetTime run off
1	1	0	cyclic sending of encoder data including re-synchronization of encoder clock start after encoder switch on
1	0	1	cyclic sending of encoder data without re-synchronization of encoder clock start after having rceived the Sync telegram and SyncOffsetTime run off
1	0	0	cyclicc sending of encoder data without re-synchronization of encoder clock start after encoder switch on not possible with cycle times being written into the RAM
0	0	1	single sending of encoder position by means of Cyc-Position telegram after having received the Sync telegram and the SyncOffsetTime has run off
0	0	0	Off

Example:

The following telegrams to set the cycle time are sent:

FSC	D0	D1	D2	D3	D4	D5	D6	Function
03H	CycTime0	CycTime1	SyncTime0	SyncTime1	SyncOffset0	SyncOffset1	------	Set present cycle time
03H	05H	00H	00H	00H	00H	00H	------	set present cycle time

and

FSC	D0	D1	D2	D3	D4	D5	D6	Function
04H	Command	------	------	------	------	------	------	Set cycle mode
04H	01H	------	------	------	------	------	------	Set cycle mode

After reset or system switch on the encoder sends its data with a cycle time base of 5 ms (CycTime 0005H) by means of the Cyc-Position telegram.

Subject to reasonable modifications due to technical advances. Copyright by Pepperl+Fuchs, Printed in Germany

4.6 Reply-Parameter Telegram

The ReplyParam-ID will be defined in the permanent memory, if it is not defined, the default Reply-Param-ID=380H will be used.

When it is switched on the encoder replies twice with the FSC FFH and gives the information with which ID the parameters can be redefined and which software version is installed.

FSC	D0	D1	D2	D3	D4	D5	D6	Function (reply to)
00H	Status	------	------	------	------	---	---	reserved
01H	Status	------	------	------	------	---	---	Sofware reset
02H	Status	------	------	------	------	---	---	Read status
03H	Status	------	------	------	------	---	---	Set present cycle time
04H	Status	------	------	------	------	---	---	Start/stop cycle mode
11H	Status	CycTime0	CycTime1	SyncTime0	SyncTime1	SyncOffset0	SyncOffset1	Read cycle time from permanent memory
12H	Status	Command[1])	------	------	------	---	---	Read start-up command from permanent memory
13H	Status	ReqPosUpd0-ID	ReqPosUpd1-ID	------	------	---	---	Read Request-Position-Update ID from permanent memory.
14H	Status	ReplyPosUpd 0-ID	ReplyPosUpd 1-ID	------	------	---	---	Read Reply-Position-Update ID from permanent memory
15H	Status	CyclicPos0-ID	CyclicPos1-ID	------	------	---	---	Read Cyclic-Position ID from permanent memory
16H	Status	Sync0-ID	Sync0-ID	------	------	---	---	Read Sync- ID from permanent memory
17H	Status	SetPara0-ID	SetPara1-ID	------	------	---	---	read Set-Parameter ID from permanent memory
18H	Status	ReplyPara0-ID	ReplyPara1-ID	------	------	---	---	Read Reply-Parameter ID from permanent memory
19H	Status	Baud rate	------	------	------	---	---	Read baud rate from permanent memory
21H	Status	------	------	------	------	---	---	Write cycle time to permanent memory
22H	Status	------	------	------	------	---	---	Write start-up command to permanent memory
23H	Status	------	------	------	------	---	---	Write Request-Position-Update ID to permanent memory
24H	Status	------	------	------	------	---	---	Write Reply-Position-Update ID to permanent memory
25H	Status	------	------	------	------	---	---	Write Cyclic-Position ID to permanent memory
26H	Status	------	------	------	------	---	---	Write Sync ID to permanent memory
27H	Status	------	------	------	------	---	---	Write Set-ParameterI ID to permanent memory
28H	Status	------	------	------	------	---	---	Write Reply-Parameter ID to permanent memory
29H	Status	------	------	------	------	---	---	Write baud rate to permanent memory
.....	------	------	------	------	------	---	---	reserved
3FH	------	------	------	------	------	---	---	reserved
.....	------	------	------	------	------	---	---	reserved
FFH	Status	SetPara0-ID	SetPara1-ID	VersNr0	VersNr1	---	---	Read Parameter ID and software version from permanent memory

[1]) Command: Please refer to chapter 4.5

Subject to reasonable modifications due to technical advances. Copyright by Pepperl+Fuchs, Printed in Germany