Brief Instructions - CANopen Interface

1. Introduction

The servo amplifier ECOSTEP®x00-Ax allows you to parameterize and move a motor by means of a superimposed control via the CANopen bus interface, in which all operating modes of the servo amplifier are available.
The advantages of this control design are:

- Reasonably priced servo amplifier with digital speed and position control.
- Modest wiring overheads between control and servo amplifier.
- Fast and reliable data exchange via CANopen.
- Reasonably priced bus interface.

2. Properties

ECOSTEP® can be operated as a slave in CANopen networks (compare „CiA Draft Standard 301“) and is conforming to the „CANopen Device Profile for Drives and Motion Control“ (compare „CiA Draft Standard Proposal 402“). Additional functions are implemented by using the „Manufacturer Specific Data“ area. The operation of the device is based on the so-called "Object Dictionary". All parameters, values and functions are accessible by means of addresses provided with index and sub-index.
The specific properties of the CANopen bus and the data exchange are not described here. If the controls used comply with the above-mentioned standards, it is not necessary to detail this part. The correct design of the corresponding bus connection, however, is important.

3. Hardware

The ECOSTEP® - CANopen connection is realized by means of a 9-pin D-Sub connector with the following pin configuration:

Pin	Signal	Designation
1	n.c.	
2	CAN_L	CAN data L
3	CAN_GND	Reference potential to CAN data
4	n.c.	
5	n.c.	
6	GND	Reference potential to CAN_V+
7	CAN_H	CAN data H
8	n.c.	
9	CAN V+	+8..+18V DC, max. 50mA

The CAN interface of ECOSTEP® must be supplied externally due to its galvanic isolation from GND and the ECOSTEP® housing. According to the CAN standard, both ends of the bus have to be provided with a terminating resistor (120 ohms) each.

4. Baud Rates

The baud rates are set by means of the objects 2F81,00 and 2F82,00. In ECOSTEP® , a baud rate of 1Mbit/s is preset.

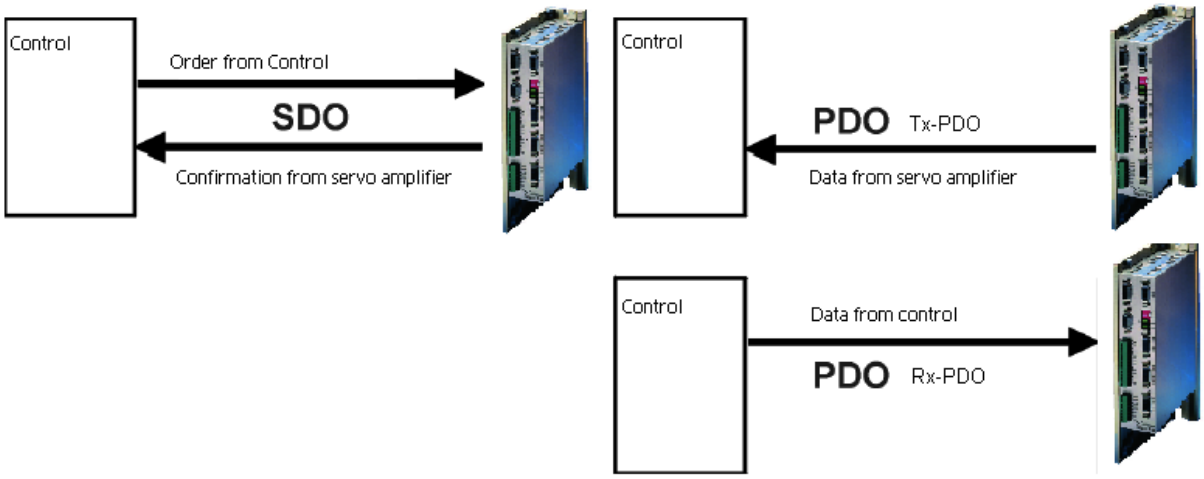
Baud rates	Max. cable length	Value in object 2F81,00	Value in object 2F82,00
1M baud	40m	0h	14h
500 kbauds	130m	0h	1Ch
250 kbauds	270m	1h	1Ch
125 kbauds	530m	3h	1Ch
50 kbauds	1.3km	47h	2Fh
20 kbauds	3.3km	53h	2Fh

JAT JENAER ANTRIEBSTECHNIK

Jenaer Antriebstechnik GmbH
Buchaer Straße 1 · D - 07745 Jena
Tel. +49 (0) 36 41 63376.0
Fax +49 (0) 36 41 63376.26
e-mail:info@jat-gmbh.de

5. Data Protocol

The CANopen bus provides two important forms of data exchange: In the case of **S**ervice **D**ata **O**bjects, SDO, designed according to the DS301 standard, data exchange is confirmed, and in the case of **P**rocess **D**ata **O**bjects, the PDOs, data exchange is not confirmed.



In addition to that, other kinds of messages are defined for further applications, sent either from the servo amplifier or from the superimposed control:

SDO	**S**ervice **D**ata **O**bject	Used for normal parameterization of the servo amplifier.
PDO	**P**rocess **D**ata **O**bject	Fast process data exchange possible (e.g. actual position).
EMCY	**Em**ergen**cy** Message	Transfer of error messages.
SYNC	**Sync**hronization Message	Synchronization of several CAN nodes.
NMT	**N**etwork **M**anagemen**t**	Network service: It is possible, for example, to act on all CAN nodes at the same time.
NODE-GUARDING	Node Guarding	Supervision of the communication participants by regular messages.

5.1 Definition of Identifiers

All messages (Communication Object, **COB)** are sent from the host to the slaves and back by means of identifiers (COB-ID). The message with the lowest COB-ID has the highest priority on the bus.
The table below shows the most important COB-IDs with the proposed base offsets. In the servo amplifier, the object Node_Offset (2F80,00) is additionally added to the node address (**DIP switch position** on the servo amplifier). This is by default = 0 and is used only if there are more than 15 participants in the network, because only a maximum of 15 devices can be addressed via the DIP switch position.

COB-IDs from servo amplifier's view

Object	**Resulting COB-IDs**	**Communication object**
NMT	0	-
SYNC	80h	1005h
Emergency	80h+ Node address [1)]	1014h
Tx-PDO1	180h+ Node address [1)]	1800h
Rx-PDO1	200h+ Node address [1)]	1400h
Tx-PDO2	280h+ Node address [1)]	1801h
Rx-PDO2	300h+ Node address [1)]	1401h
Tx-PDO3	380h+ Node address [1)]	1802h
Rx-PDO3	400h+ Node address [1)]	1402h
Tx-PDO4	480h+ Node address [1)]	1803h
Rx-PDO4	500h+ Node address [1)]	1403h
Tx-SDO	580h+ Node address	1200h
Rx-SDO	600h+ Node address	1200h

[1)] The base address can be changed in this area.

5.2 SDO Access

The SDOs allow direct access to the object dictionary of the servo amplifier. This access is simple and well arranged. An SDO access always starts from the superimposed control (host), which sends to the servo amplifier either a write instruction in order to change a parameter of the object dictionary, or a read instruction in order to read out a parameter. The host receives an answer for each instruction. This answer either contains the readout value or – in the case of a write instruction – serves as a confirmation. The identification of the message for the servo amplifier is effected by means of the COB-ID. The structure of the instructions or the answers depends on the data type of the object to be read or written, because 1, 2, or 4 data bytes have to be sent or received.

Write Access, Data Transfer from Host to Slave

Each access to the object dictionary is checked by the slave for validity. Any write access to non-existing objects, to read-only objects or with a non-corresponding data format are rejected and answered with a corresponding error message.

The host sends:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
CMD	Index LSB	MSB	Sub-Index	Data LSB			..MSB

CMD determines the direction of data transfer and the size of the data object:
- 23 hex Sending of 4-byte data (bytes 4..7 contain a 32-bit value)
- 2B hex Sending of 2-byte data (bytes 4, 5 contain a 16-bit value)
- 2F hex Sending of 1-byte data (byte 4 contains an 8-bit value)

Index 16-bit value; index of the object to be written into (in the object dictionary)
Sub-Index 8-bit value; sub-index of the object to be written into (in the object dictionary)
Data 8-bit, 16-bit, or 32-bit value

The slave answers:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
RES	Index LSB	MSB	Sub-Index	Reserved			

RES Response of the slave:
- 60 hex Data sent successfully
- 80 hex Error, bytes 4..7 contain the error code according to the standard

Index 16-bit value; index of the object addressed by the host telegram
Sub-Index 8-bit value; sub-index of the object addressed by the host telegram
Reserved Is not used, or error message (depending on RES)

Example: Writing into the control word (Index 6040, Sub-Index 00) value = 6h (axis off)

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Sending:	2B	40	60	00	06h	00	00	00
Response:	60	40	60	00	00	00	00	00

Read Access, Data Transfer from Slave to Host

Any read access to non-existing objects is answered with an error message.

The host sends:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
CMD	Index		Sub-Index	Reserved			

CMD determines the direction of data transfer:
- 40 hex read access (in any case)

Index 16-bit value; index of the object to be read (in the object dictionary)
Sub-Index 8-bit value; sub-index of the object to be read (in the object dictionary)
Reserved Is not used

The slave answers:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
RES	Index LSB	MSB	Sub-Index	Data LSB			MSB

RES Response of the slave:
- 43 hex Bytes 4..7 contain a 32-bit value
- 4B hex Bytes 4, 5 contain a 16-bit value
- 4F hex Byte 4 contains an 8-bit value
- 80 hex Error, bytes 4..7 contain the error code according to the standard

Index 16-bit value; index of the object addressed by the host telegram
Sub-Index 8-bit value; sub-index of the object addressed by the host telegram
Data Data or error message (depending on RES)

Example: Reading of the status word (Index 6041, Sub-Index 00)

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Sending:	40	41	60	00	00	00	00	00
Response:	4B	41	60	00	37	40	00	00

Value of the status word: 4037h (axis switched on, no error)

5.3 SDO Telegram for Terminals

If the controller provides no possibility for a permanent setting of the identifier and the length of an instruction, see the table below for a complete SDO telegram for sending a 32-bit value to ID 1.

Identifier	Length	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
601	8	2B	40	60	00	6	0	0	0
COB-ID: ID 1	Number of data bytes: 8 bytes	CMD: write 16 bits	Index 6040 Low byte first		Sub-index 0	Data: 6h, low byte first			

The settings for the data exchange with ECOSTEP® are OFF for bits “Extended Frame” and “Remote Request”. The most important objects are quoted in chapter 7.

5.4 PDO Access

The very fast, unconfirmed mechanism to transfer data as PDO has the advantage of an event-driven data transfer. In this process, the PDO transmits one or several predefined parameters. As no confirmation takes place, the recipient has to be able to process possibly arriving PDOs at any time after the PDO has been activated. This kind of transmission offers the advantage that the host computer does not need to poll the parameters transmitted by a PDO. This leads to a considerably reduced utilization of the CAN bus capacity. One PDO allows to transmit data of a maximum of 8 bytes, i.e. 1 to a maximum of 8 objects, depending on the object size. There are receive (Rx) and transmit (Tx) PDOs. Viewed from the servo amplifier, a PDO is defined by specifying the COB-ID, the cycle time, and the data to be transmitted.

EXAMPLE: Check: Positioning from A to B finished?
In the case of SDO access, permanent polling of the object **Statusword** is required. This brings disadvantages for the bus capacity.
In the case of PDO access, with every change of the object **Statusword** the servo amplifier can transmit a PDO that contains the data of the **Statusword**. Thus, the host computer receives automatically a corresponding message as soon as the event has occurred.

The PDOs allow to transfer all objects of the object dictionary with the property „mappable“, i.e. the PDO contains, as data, for example the actual speed value, the actual position value, or the like.
Beforehand, the servo amplifier must be informed of which data are to be transmitted, because the PDO contains merely user data and no information about the parameter type. This way, almost any data telegram can be defined. The settings required therefore are described in the following chapters.
It is interesting that slaves can send and receive PDOs among themselves also without a master, provided the COB-IDs are matched to one another.

5.4.1 PDO Communication Parameters

The communication objects have the same structure for all PDOs. The table below shows the communication object for Tx-PDO 1.

Object	Sub-Index	Description	Default
1800h	1	COB-ID for PDO	180_h+ Node address
1800h	2	Transmission Type	FF_h
1800h	3	Inhibit Time [100µs]	1000

5.4.2 PDO Transmission Type

For each PDO the event leading to the transmission (Tx-PDO) or evaluation (Rx-PDO) of a message must be defined. This is done by the object Transmission_Type (communication parameters, object 1800, sub-index 02).

Value		Description
0h – 8Ch	**SYNC Message**	The numerical value indicates after how many SYNC messages the PDO - is sent (Tx-PDO) or - evaluated (Rx-PDO) each time.
FFh	**Change**	The Tx-PDO is sent as soon as at least 1 bit has changed in the PDO data. With **Inhibit_Time**, the minimum interval between two PDOs sent can be defined additionally in 100µs steps.

5.4.3 PDO Mapping

After the identifier and the transmission type have been defined, the servo amplifier is informed about the data to be sent in one PDO or where the received data shall be transferred to. This is called "mapping".
As each PDO can transfer a maximum of 8 bytes, it is possible to assign a maximum of 8 objects with 1 byte each.
In the mapping objects, the number of objects to be transferred is defined first. After that, those objects are entered, the data of which shall be sent or to which the received data shall be assigned.

Mapping object for Tx-PDO 1

Object	Sub-index	Description	Default
1A00h	0	Number of mapping entries	0
1A00h	1	PDO1-Mapping 1	00020008_h (wildcard)
1A00h	2	PDO1-Mapping 2	00020008_h (wildcard)
1A00h	3	PDO1-Mapping 3	00020008_h (wildcard)
1A00h	4	PDO1-Mapping 4	00020008_h (wildcard)
1A00h	5	PDO1-Mapping 5	00020008_h (wildcard)
1A00h	6	PDO1-Mapping 6	00020008_h (wildcard)
1A00h	7	PDO1-Mapping 7	00020008_h (wildcard)
1A00h	8	PDO1-Mapping 8	00020008_h (wildcard)

The mapping entries are structured as follows:

Main index of the object to be mapped (hex)
Sub-index of the object to be mapped (hex)
Length coding of the object in hex code

Consequently, length codings 08h are for 8-bit, 10h for 16-bit, and 20h for 32-bit values.

With the length specification even parts of an object can be written byte by byte, beginning with LSB (e.g. the lower 8 bytes of a 32-bit object), into the PDO.
The length specification of each object is registered in the object dictionary.

5.4.4 Programming Example for Tx-PDO 1

In the example given below, the servo amplifier with ID 1 sends the status word, the actual position and 2 bytes of the digital inputs on the message ID 181 every 10 ms, after programming has been completed.

Index, Sub.	Comment	Default	Example
1800h_01h	COB_ID_used_by_PDO	00000181h	00000181h
1800h_02h	Transmission type	FFh	Feh
1800h_03h	Inhibit time [1/10 milliseconds]	1000	100
1A00h_01h	PDO1-Mapping 1	00020008h	60410010h
1A00h_02h	PDO1-Mapping 2	00020008h	60630020h
1A00h_03h	PDO1-Mapping 3	00020008h	60FD0010h
1A00h_00h	Number of mapping entries	00h	03h

5.4.5 Parameterization of PDOs for the interpolating Mode

One of the operating modes of ECOSTEP® is the interpolating mode. In this mode, the reference and the actual values of an axis are cyclically exchanged with a superimposed control by means of PDOs. All instructions required to secure this data exchange between ECOSTEP® and a superimposed control are listed below.
In addition to that, the host sends a **Sync** message for reading in and synchronizing the axes.

Index	Sub.	Byte	Value (h)	Description
0x1800	1	4	0x181	Setting the COB-ID for the transmit (Tx) PDO 1 to 181_h
0x1800	2	1	0x1	Setting the operating mode for the transmit PDO 1 to synchronous mode
0x1400	1	4	0x201	Setting the ID for the receive (Rx) PDO 1 to 201_h
0x1400	2	1	0x1	Setting the operating mode for the PDO to synchronous mode
0x1600	1	4	0x60400010	Mapping the first two bytes of the receive PDO 1 to the control word of ECOSTEP®
0x1600	2	4	0x607a0020	Mapping the next 4 bytes of the receive PDO 1 to the target position of ECOSTEP®
0x1600	0	1	0x2	Number of the mapped variables of the receive PDO1
0x1A00	1	4	0x60410010	Mapping the status word of ECOSTEP® to the first two bytes of the transmit PDO 1
0x1A00	2	4	0x60630020	Mapping the actual position of ECOSTEP® to the next 4 bytes of the transmit PDO 1
0x1A00	0	1	0x2	Number of the mapped variables of the transmit PDO1

With these few SDO transmission instructions the preconditions are fulfilled to access the agreed objects of ECOSTEP® by means of a transmit and a receive PDO, or to evaluate the actual values of ECOSTEP®. Only the operating mode must still be set to the interpolating mode (6060,00,1 byte, value 7) via SDO access.

Index	Sub.	Byte	Value (h)	Description
0x6040	0	2	0x6	Control word, switching the axis off or ready to start (possible by PDO)
0x6040	0	2	0xf	Control word, switching the axis on (PDO)
0x6060	0	1	0x7	Setting the operating mode to interpolating mode (SDO) (special firmware necessary)
0x6040	0	2	0x1f	Control word, enabling the interpolating mode (PDO)

The synchronous data transfer is always started by the superimposed control.

5.4.6 Parameterization of the Homing Procedure (inside the Servo Amplifier)

A specialty is the homing procedure. As, in the most cases, the index signal of an encoder or sensor does not get to the control via the CAN bus, homing is executed inside the ECOSTEP® ; the control just initiates it. If the control word is already in the 1Fh state, it is only necessary to set the operating mode to homing, starting it immediately with that. The way the homing procedure is carried out depends on the homing method.

Index	Sub.	Byte	Value (h)	Description
0x6040	0	2	0xf	Control word, switching the axis on or ready to start
0x6098	0	1	17 (dec.)	Setting the homing method to 17, traveling to the negative limit position without evaluation of the index signal
0x6060	0	1	0x6	Setting the operating mode to homing
0x6040	0	2	0x1f	Control word, starting the homing procedure

The homing procedure is evaluated by means of the bit "reference found" in the status word (bit 0x8000).
After successful homing, the operating mode can be changed again to the „interpolating mode“.
As not all controls are provided with the homing function in the CNC part, it must be programmed in the PLC part of the user program itself.

5.5 Network Management (NMT-Service)

All CANopen devices can be controlled by means of the network management. A special identifier (000h) is reserved for this purpose.
It is used to send instructions to one or all servo amplifiers. Each instruction is composed of two bytes - the first byte includes the instruction code and the second byte the node address of the operated servo amplifier.
Structure of the message of the superimposed control:

Identifier	Byte0	Byte1
000h	Instruction code	Node address

With the node address zero, all nodes in the network can be activated at the same time. The servo amplifiers do not confirm the NMT instructions.
The „pre-operational" status is provided for the configuration phase of the network. For the work with PDOs, the node must have accepted the communication status „**operational“**. Generally, the network is booted by the host and put in the "operational" status by means of the „NMT-Start“ instruction.

Code	Name	Communication status
01	NMT-Start	operational
02	NMT-Stop	pre-operational
04	Prepare Remote Node	pre-operational
80	pre-operational	pre-operational

5.6 Emergency Message

The identifier of the emergency message is formed by the **Offset 080h + node number** of the servo amplifier.
The emergency message will be sent as soon as a not yet registered error occurs. Permanent errors generate an emergency message only once, or, after each attempted error reset, once more.
It is composed of eight data bytes. The first two bytes contain an **error code**. Further bytes are not used up to now.

080h+node number	**error_code**	sub_error	0	0	0	0	0

The table with the error codes is attached at the end of the document.

6. Objects for Initialization via CAN

The user is free to integrate further objects via SDO access in his application in order to improve ease of operation or simplify trouble shooting. For each servo amplifier to be initialized completely via the CAN bus, the objects specified here are necessary and must be stored in the control.
As an example, the table below contains the data for a motor 23S31-0650-xx3Jx-xx connected to ECOSTEP®100-AA.

Index	Sub	Byte	Value (h)	Description
Commutation parameters				
60F6	1	4	8000	Commutation period
60F6	2	2	50	Pole pairs per commutation period
60F6	3	2	180	Phase lead factor dependant on velocity
60F6	4	2	0	Phase lead factor dependant on torque
60F6	5	2	60	Maximum phase lead value
60F6	6	2	1023	Current for finding commutation
60F6	7	2	500	Delay time for finding commutation
60F6	8	2	0	Damping for finding commutation
60F6	9	2	3 (1)	Method for finding commutation [1=vertical axes]
60F6	10	2	100	Response delay of holding brake [ms]
60F6	13	2	1000	Commutation function, component 1xf
60F6	14	2	0	Commutation function, component 3xf
60F6	15	2	0	Commutation function, component 5xf
60F6	16	2	0	Commutation function, component 7xf
60F6	22	2	0	Current equalization, basic value
60F6	23	2	0	Current equalization, first coefficient
60F6	24	2	0	Current equalization, second coefficient
60F6	25	4	5.000.000	Limit velocity for silent mode
Current controller parameters				
6073	0	2	2047	Max. motor current 2047 = 100%
60F6	11	2	1500	I^2t-supervision current
60F6	12	2	40	I^2t-time constant [s] (0 = function inactive)
Velocity controller parameters				
60F9	1	2	50	p-gain of velocity controller
60F9	2	2	1	i1-gain of velocity controller
60F9	7	2	0	i2-gain [1/256], i_{tot}= i1+i2
60F9	3	2	1000	i-limiting of velocity controller
60F9	4	2	2	Time constant of error filter
60F9	5	2	2	Time constant of output filter
Position controller parameters				
60FB	1	2	1000	p-gain of position controller
60FB	2	4	0	Maximum accelerating power
60FB	3	2	16384	Velocity precontrol 16384=100%
60FB	4	2	0	Acceleration precontrol
60FB	5	2	0	Current precontrol offset
60FB	6	2	0	Current precontrol factor
60FB	7	2	0	Current precontrol limit
607E	0	1	0	Polarity of position controller [0, 80]
Limit values / monitoring				
6065	0	4	2000	Max. admissible following error [inc]
6067	0	4	10	Target position window [inc]
607D	1	4	0	Min. software limit position (both 0 = inactive)
607D	2	4	0	Max. software limit position
607F	0	4	30.000.000	Max. velocity [inc/64s]
608F	1	4	8000	Encoder resolution, encoder monitoring

Homing settings				
6098	0	1	17	Homing method
607C	0	4	0	Relocation zero point [inc]
6099	1	4	1.000.000	Velocity during search for reference switch [inc/64s]
6099	2	4	100.000	Velocity during search for reference set point [inc/64s]
609A	0	4	20.000	Acceleration during homing [16x inc/s^2]
Configuration of digital inputs				
2170	0	1	60	Polarity mask inputs DIN1..8 (DIN 6, 7=openers)
2171	1	1	0	OR-mask, positive limit position
2171	2	1	20	AND-mask, positive limit position (DIN6 active)
2171	3	1	20	Comparison mask, positive limit position
2172	1	1	0	OR-mask, negative limit position
2172	2	1	40	AND-mask, negative limit position (DIN7 active)
2172	3	2	40	Comparison mask, negative limit position
Reference values				
607A	0	4	0	Setpoint position [inc]
6081	0	4	0	Positioning speed [inc/64s]
6083	0	4	20000	Run-up acceleration [16x inc/s^2]
6084	0	4	20000	Braking acceleration [16x inc/s^2]
6085	0	4	20000	Fast stop braking acceleration [16x inc/s^2]
6040	0	2	6	Control word
6060	0	1	1	Operating mode [1,3,4,6,-3,-4,7]
60FF	0	4	0	Nominal speed - speed mode [inc/64s]
Actual values				
6063	0	4		Actual position [inc]
606C	0	4		Actual velocity [inc/64s]
6041	0	2		Status word
6061	0	1		Current operating mode
60FD	0	4		Value of digital inputs
Digital outputs				
60FE	1	4	0	Setting of outputs [0, 10000_h, 20000_h, 30000_h]
Turn-off behavior / Behavior in case of error				
605A	0	2	5	Reaction on quick stop (description in object dictionary)
605B	0	2	1	Reaction on switch-off
605C	0	2	1	Reaction on disable operation
605D	0	2	0	Reaction on stop
605E	0	2	1	Reaction on fault
Fault evaluation (reading)				
2600	2	4		Error display (Object catalogue)
603F	1	2		Last error

7. Technical Data

Line-Interface	ISO-High-Speed-Driver PCA82C251T Full-CAN-Controller
Baud rate	max. 1000 kB/s
Physical Layer	ISO 11898
Data Link Layer	CAN Specification V2.0A (V2.0B passive)
Protocol	CANopen DS301 and DS402 (see CANopen Manual)
Recommended sync clock of the control	< 4ms
Fine interpolation cycle ECOSTEP®	1ms

9. Error Bits and Codes ECOSTEP®

Error bits in the object 0x2600, 02 (Detected_Faults).
The object 0x603F, 00 contains the last detected fault as a 16-bit value.
Only the two low bytes of the object 603F are sent by means of the emergency message.

Fault	**Bit code in object 2600,02**	**Description**	**Code in object 603f,00**
FAULT_H8SWD_BIT	1h	Software reset triggered	0x00006010
FAULT_REGLERWD_BIT	2h	Synchronization loss	0x00006011
FAULT_ENC_ERROR_BIT	4h	Motor encoder non-equivalence error	0x00007305
FAULT_MOTENCCAP_BIT	8h	Motor encoder counting error	0x00017305
FAULT_MAENCCAP_BIT	10h	Master encoder counting error	0x00017306
FAULT_OVERTEMP_BIT	20h	Servo amplifier temperature > 80°C	0x00004210
FAULT_UVMESS_BIT	40h	Undervoltage logic supply	0x00005112
FAULT_OV_ERROR_BIT	80h	Overvoltage DC link	0x00003210
FAULT_UV_ERROR_BIT	100h	Undervoltage DC link	0x00003220
FAULT_A_ERROR_BIT	200h	Short circuit phase A	0x00012320
FAULT_B_ERROR_BIT	400h	Short circuit phase B	0x00022320
FAULT_OUT_DIAG_BIT	800h	Short circuit or overload of output READY, OUT1, OUT2, or Brake	0x00032320
FAULT_EX_ENABLE_BIT	1000h	External ENABLE on low, with axis on	0x00005441
FAULT_FOLLOWINGERROR_BIT	2000h	Following error too high	0x00008611
FAULT_OVERSPEEDERROR_BIT	4000h	Velocity too high (not used)	0
FAULT_COMMUFINDERROR_BIT	8000h	Commutation not found	0x00008312
FAULT_ABORT_CONNECT_BIT	10000h	CAN communication interrupted in the Node Guarding mode	0x00008120
FAULT_IxIxT_BIT	20000h	i^2*t supervision released	0x00002310
Additional firmware - 800 (interpolation)			
FAULT_NLOCK_BIT	40000h	Min. hardware limit position triggered	0x00008612
FAULT_PLOCK_BIT	80000h	Max. hardware limit position triggered	0x00008613
Extended error codes firmware – 009 (absolute encoder)			
Motortemperatur_error	100.000h	Motor temperature is too high (bridge is missing)	0x00004310
User_data_error	200.000h	No access to saved user data	0x00007600
User_data_valid_error	400.000h	User data invalid	0x00007601
Motor_data_error	800.000h	No access to motor data	0x00007602
Motor_data_valid_error	1.000.000h	Motor data invalid	0x00007602