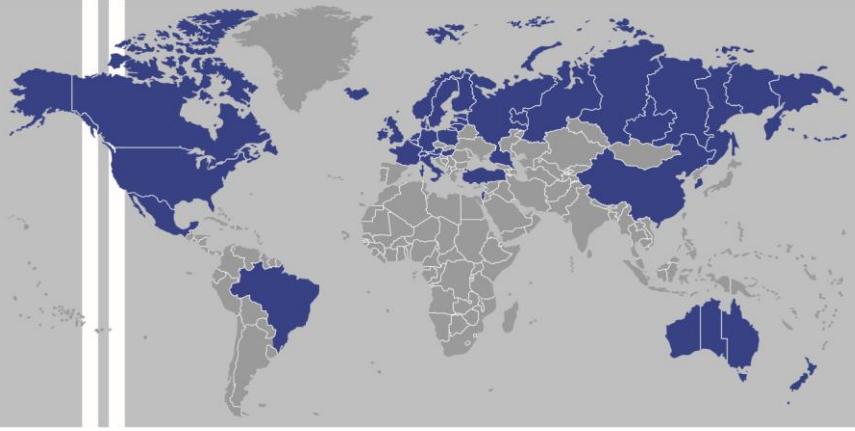

dunkermotoren
advanced motion solutions

DME 230 / 400

CAN Open Interface
for servo drives series
- BN6773
- BN6783

Typ:	Part No:
DME 230x4-CO	81703.00100
DME 230x4-I/O	81703.00101
DME 400x8-CO	81703.00110

Publication Ref: 160616

Dunkermotoren GmbH | Allmendstraße 11 | D-79848 Bonndorf/ Schwarzwald
Phone +49 (0) 7703 930-0 | Fax +49 (0) 7703 930-210/ 212 | info@dunkermotoren.com

AMETEK®
PRECISION MOTION CONTROL

TrioDrive D/CS / MidiDrive D/CS

Digital Servo Drives
for Direct Mains Connection

CANopen Interface

Operating Instructions 6745.205, V 1.0

These operating instructions apply to

- TrioDrive D/CS servo drives, compact design BN 6756 to BN 6758 with built-in power supply unit for single-phase AC voltage connection and integrated safety system
- MidiDrive D/CS servo drives, compact design BN 6745 to BN 6749 with built-in power supply unit for three-phase AC connection and integrated safety system

These operating instructions are applicable together with

- Operating Instructions 6710.201 (Functions and Parameters)
- Operating Instructions 6755.202 or 6745.202 (Connection and Commissioning)
- Operating Instructions 6710.207 (SPP Windows Command and Commissioning Software)

ESR Pollmeier GmbH
Lindenstr. 20

64372 Ober-Ramstadt

Federal Republic of Germany

Phone +49 6167 9306-0
Fax +49 6167 9306-77

E-mail info@esr-pollmeier.de
www.esr-pollmeier.de

Versions of the Document

2006-04-07	V 0.1, KS/Ri	new for TrioDrive D/CS and MidiDrive D/CS on the basis of 6710.205
2009-05-07	V 1.0, KS	pin numbers terminating connector corrected; standards updated; variables added

O:\!DB\6\7\4\6745_205_10.wpd

These operating instructions have been prepared with care. However, ESR Pollmeier GmbH can accept no liability for any errors in these operating instructions or possible consequences. Neither can any liability be accepted for direct or indirect damage resulting from abuse of the device.

The relevant regulations concerning safety technology and electromagnetic compatibility must be complied with when using the device.

Subject to alteration.

Contents

Also see the *keyword index* starting on page 62.

Appendix

Figures

1 Preliminary Remarks

1.1 About this Description

These Operating Instructions 6745.205 explain the CANopen interface of the TrioDrive D/CS (BN 6756 to BN 6758) and MidiDrive D/CS (BN 6745 to BN 6749) digital servo drives. It particularly deals with access to the servo drive functions via CANopen communication interface.

The CANopen interface is usually used for connecting the servo drives to a higher-level controller. Function blocks described in detail in section 1.2 (page 8) are available for an integration of the servo drives in automation systems.

With the integrated part program (optional), servo drives combined with an operating terminal or with I/O extensions can also be used in applications for which a higher-level controller is not required. For further information, please contact ESR.

These Operating Instructions 6745.205 are applicable together with

- Operating Instructions “Connection and Commissioning” of the servo drive (included in the scope of supply of the servo drive)
  - Operating Instructions 6755.202 (TrioDrive D/CS),
  - Operating Instructions 6745.202 (MidiDrive D/CS)
- Operating Instructions “Functions and Parameters” of the servo drive (included in the scope of delivery of the servo drive)
  - Operating Instructions 6710.201
- Operating Instructions “SPP Windows Command and Commissioning Software” (delivered with the optional SPP Windows command and commissioning software)
  - Operating Instructions 6710.207

The following documents are additionally required for working with the CANopen interface:

- CANopen Communication Profile for Industrial Systems CiA 301 (short: CANopen Communication Profile CiA 301 or CiA 301)

  These operating instructions are based on CANopen Communication Profile CiA 301, V 4.0.2, dated February 13, 2002. For the application of a higher version, please contact ESR.
- Description(s) of the CAN interface module installed in the controller and the software used there.

Parts of the CANopen Device Profile Drives and Motion Control CiA 402 (short: CANopen Drives Profile CiA 402 or CiA 402) are implemented with the CANopen interface. The implemented parts of CiA 402 are described completely in

these operating instructions so that this document is not required for working with servo drives with CANopen interface.

The functions of the servo drives are accessed via the CANopen interface using so-called variables. For the purposes of these operating instructions, it is useful to differ between two types of variables, the drive variables and the CANopen variables:

- Drive variables

The drive functions of the servo drive are accessed independent of the communication interface (serial interface, CANopen, or another field bus interface) via drive variables (e.g. *Axis Operating Mode*, *Target Position*, *Actual Velocity*). Drive functions and drive variables of the servo drives are described in Operating Instructions 6710.201 “Functions and Parameters”. Section “Descriptions of Variables” and appendix “List of Variables” of those operating instructions include all information on the variables required for access via one of the communication interfaces.

You can try out these drive functions and drive variables of the servo drives without editing a program for your computer or controller or putting the CAN-bus into operation. For doing so, use a PC with the SPP Windows command and commissioning software. You should not edit programs accessing these variables via the CANopen interface until you have learned about the functions of the servo drive using the PC and know which functions are carried out or queried by which variable access.

- CANopen variables

  In addition to above-mentioned drive variables, further variables are used for managing the CANopen interface (CANopen variables, e. g. *Node-ID*, *Guard Time*). The CANopen variables of the servo drives are described in these operating instructions. Section 14 (Descriptions of Variables) and Appendix B (List of Variables) of these operating instructions contain all information on the variables required for access via the CANopen interface.

Before acccessing the servo drives via the CANopen interface according to these operating instructions, the servo drive (servo drive and servo motor) should have been put into operation. A PC with SPP Windows command and commissioning software is required for putting the servo drives into operation. Please check whether or not these prerequisites are met.

1.2 Function Blocks

Function blocks are available for an easy integration of the servo drives into automation systems.

These are available for various controllers according to IEC 61131-3. For further information, please contact ESR.

Communication is made in the form of PDO/SDO communication via CANopen.

Supported functions:

- parameterization of the servo drives by the controller (e. g. after switch-on)
- triggering of movements (relative/absolute positioning, going to home position, speed setting, ...)
- influencing the drive-integrated positioning control (part program)
- input and output of binary signals (software inputs/outputs)
- example programs for using the function library as a basis for the development of own programs

The function blocks are based on the PLCopen specification “Function Blocks for Motion Control” which is based on IEC 61131-3.

For further information see data sheet 6710.260.

The function blocks simplify the use of the functions described in these operating instructions. The parameterization steps you have to carry out yourself are described in the operating instructions of the function blocks.

2 Safety Instructions

In any case, observe the safety instructions as well as the warnings and hints in the margins of the respective Operating Instruction “Connection and Commissioning” (6755.202 or 6745.202).

Access to the servo drives via the CANopen interface may trigger drive movements. If drive and/or machine have not been set up and secured properly, health and life of persons may be endangered.

Therefore, access via the CANopen interface is prohibited until the requirements of the machine directive have been met.

In bus systems, a bus participant can be influenced invisibly from outside. This can lead to an unexpected (uncontrollable) system behavior. Do not put the bus into operation unless you have made sure that all participants are properly connected and configured.

2.1 Type of Instructions

The warnings and hints in the margin must be observed under any circumstances:

- **Danger** to health and life due to electrical shock or motion of the drive.

- **Caution**: Noncompliance violates the safety regulations or statutory provisions and can lead to personal injury or material damage.

- **Check**: Prior to commissioning and in case of failures or problems, check these items first.

- **Tip**, useful hint.

3 Technical Specifications

The CANopen interface is installed in the servo drive as a module. It consists of two RJ 45 connectors located at the front side of the device. Connector pin assignment and signal levels correspond to CANopen CiA DR 303-1 (Cabling and Connector Pin Assignment) or ISO 11898. The bus connection is separated galvanically from the CAN controller.

Supported transmission rates

- 1000, 800, 500, 250, 125, 50, 20, 10 kBit/s

Functions of the CANopen interface according to CANopen Communication Profile CiA 301 V 4.0.2:

- NMT Master: no (master functions on request)
- NMT Slave: yes
- Extended boot-up: no
- Minimum boot-up: yes
- COB-ID assignment: SDO, default
- Node-ID assignment: software
- No. of PDOs: 4 Rx/4 Tx
- PDO modes: synchronous (cyclic, acyclic), asynchronous
- Variable PDO mapping: yes
- Emergency message: yes
- Life guarding: yes
- No. of SDOs: 2
- Device profile: DRIVECOM 22 with manufacturer-specific extensions and parts of CiA 402

4 CANopen Introduction

To understand these operating instructions, the reader needs to be familiar with the CANopen terms, especially the terms of CANopen Communication Profile CiA 301. This section gives a short overview of the major terms and abbreviations. It cannot replace original documents and a corresponding training, if required.

4.1 Terms and Abbreviations

CAL (CAN Application Layer)

Communication layer above the actual CAN bus which was created for CAN bus applications in open communication systems. It consists of the elements NMT, DBT, LMT, and CMS. Due to the fact that CAL is very extensive and can be used very flexibly, a series of CAL functions was defined particularly for automation applications in CANopen communication profile CiA 301.

CAN (Controller Area Network)

Serial bus system (also called CAN bus) originally developed for an application in automobiles which is meanwhile also used in automation systems. CANopen extends the protocols of the CAN bus by additional layers.

CAN Controller

Electronic component which implements the CAN protocol.

CANopen

Communication model defined by CIA on the basis of CAN bus and CAL. With CANopen Communication Profile CiA 301, a series of CAL functions was defined particularly for automation applications to facilitate the application of devices of different manufacturers in one bus. On this basis, further profiles are defined for certain device types such as e. g. drives; currently, CANopen Drive Profile CiA 402 is available for drives.

CiA (CAN in Automation International Users and Manufacturers Group)

Association of manufacturers and users of devices with CAN interface.

CMS (CAN Message Specification)

Part of CAL, defines various mechanisms for data transmission.

COB (Communication Object)

All CAN bus data are transmitted in packages called COB (another name is CAN message). The devices connected to the bus are able to send and receive COBs.

These COBs must not be mixed up with the communication objects described in these operating instructions. The communication objects described in these operating instructions are variables, not data packages.

COB-ID (COB Identifier)

Each COB is clearly identified by an identifier which is part of the COB. The CAN bus supports up to 2032 COBs identified by identifiers with a length of 11 bits. An extension of CAN also provides identifiers of a length of 29 bit, however, these are not supported by the servo drives. In these operating instructions, COB-IDs are always stated as hexadecimal values.

DBT (Distributor)

Part of CAL, manages the assignment of COB-IDs. Like most CANopen devices, servo drives use an easier way of assigning COB-IDs to a device: they are selected via standard values (standard settings) depending on the Node-ID and can be changed via SDO, if required.

DRIVECOM

Association of drive manufacturers who developed the standards for networking drives (profiles). DRIVECOM Profile 22 for positioning drives was the basis for the development of CANopen Drive Profile CiA 402 by CiA and is implemented in the servo drive.

EDS (Electronic Data Sheet)

Description of the CANopen functions and variables as data file. The EDS can be generated with SPP Windows and provides easy access to a device.

EMCY (Emergency)

Emergency function for transmitting faults including fault codes to the master.

NMT (Network Management)

Part of CAL, responsible for initialization, configuration, and fault handling.

Node-ID (Node Identification)

Identifies a device definitely in the CANopen network, i. e. all devices must have different Node-IDs. The Node-ID of servo drives is set via software. The COB-IDs are derived from the Node-ID. In these operating instructions, Node-IDs are always stated as hexadecimal values.

PDO (Process Data Object)

Is used for fast real-time access to selected data. For certain variables or groups of variables, mappings are pre-configured on certain PDOs.

The SDO is provided for access to all other variables.

Profile

In communication based on bus systems, profiles are documents serving for the standardization of devices. For that, either communication functions (in a communication profile) or device functions (in a device profile) are described from the point of view of the communication interface.

RPDO (Receive Process Data Object)

PDO received by the servo drive (contains e. g. target position).

SDO (Service Data Object = parameter object)

The SDO provides access to all variables in a CANopen device. For servo drives, these are the drive and CANopen variables.

Usually, the SDO is used for configuration. PDOs are used for fast real-time access to selected variables.

SYNC (Synchronization)

PDOs can also be transmitted synchronously using the SYNC message.

TPDO (Transmit Process Data Object)

PDO sent by the servo drive (contains e. g. *actual position*).

Variable

The user can access all drive and CANopen functions via variables. These variables may consist of one single word (e. g. *Position Controller Kp*), but may also be very extensive (e. g. *Part Program* with 71 elements of 128 bytes each). They can be accessed via SDOs or PDOs. In these operating instructions, the CANopen variables for the servo drive are printed in *italics* and described in a separate section.

5 Connection and Commissioning

For connection, basic settings, and status display, the servo drives are equipped with the following elements:

- bus connection
- LEDs

These elements are located at the front side of the TrioDrive D/CS and MidiDrive D/CS servo drives.

Coding switches are not required for TrioDrive D/CS and MidiDrive D/CS servo drives as the settings are done via software.

5.1 Bus Connection

X4.1 CAN in: RJ 45 connector at the front panel
X4.2 CAN out: RJ 45 connector at the front panel

Pin	Signal	Belegung
1	CAN_H	bus line (dominant high)
2	CAN_L	bus line (dominant low)
3	CAN_GND	ground / 0 V / V–
4	–	reserved
5	–	reserved
6	CAN_SHLD	shield connection for special applications (use after consultation with the manufacturer, only)
7	CAN_GND	ground / 0 V / V–
8	–	not assigned
Housg.	Shield	The cable shield must be connected to the connector housing. Use standard CAT5 cables.

The two RJ 45 connectors are assigned according to CiA DR 303-1 (Cabling and Connector Pin Assignment). The signals correspond to standard ISO 11898. The bus connection is galvanically isolated from the CAN controller.

Pin 8 of X4.1 and X4.2 is interconnected for an external supply of the CAN supply voltage in case the master connected to the bus does not provide a voltage supply.

5.1.1 Connectors and Termination

With the CAN in and CAN out connectors, the device can be connected to the bus without T connectors and stub lines.

In case the servo drive is connected to the bus as first or last participant, a proper termination of the bus must be guaranteed. This can be done e. g. by installing a terminating connector on the CAN out connection (terminal resistance of 120 Ω between pin 1 and pin 2).

5.1.2 Bus Lines

Max. permissible bus lengths (total of all bus lengths):

Transmission rate kBit/s	1000	500	125	50
Max. bus length m	40	100	500	1000

Cables corresponding to ISO 11898 part 2 must be used.

5.2 LEDs

The Error, Run, Aux1, and Aux2 state LEDs of the CANopen module are located at the front side of the servo drives below the “Bus” marking.

These LEDs display the states of the CANopen interface (communication).

LD Error (red) displays the send and receive monitoring state of the CAN controller installed in the servo drive:

- off: OK (CAN term: error active)
- flashing, approx. 0.5 Hz: warning (CAN term: error passive)

  sending or receiving faults have occurred frequently, device and bus continue operation
- on: fault (CAN term: bus off)

  sending or receiving faults have occurred too many times, device disconnected from the bus

LED Run (green) displays the state in the NMT node state diagram:

- flashing, approx. 2 Hz: pre-operational
- permanently on: operational

The Aux1 and Aux2 LEDs are currently not used.

5.3 Commissioning

For commissioning, it is essential to observe the safety notes of section 2 (page 10).

The commissioning of a servo drive with CANopen interface must be carried out in 3 steps which are described in the following.

5.3.1 Configuration of the CANopen Interface

Carry out the settings of the CANopen interface using the configuration aid of Appendix C. Copy this configuration aid once for each servo drive. Fill in the sheets, then start setting the device. First of all, the hardware settings have to be done, afterwards, the machine data of the CANopen interface can be set.

Make sure that the settings of the machine data of the CANopen interface are stored on floppy/hard disk and in the servo drive using the CANopen configuration software, see section 12 (page 29) and/or the operating instructions of the applied CANopen configuration software.

The configuration of the CANopen interface with setting of the device can also be carried out by the manufacturer at extra charge. If required, please contact ESR.

5.3.2 Parameterization of the Drive Functions (Servo Drive, Servo Motor)

A PC with command and commissioning software SPP Windows is required for commissioning the ESR servo drives. For further information, see Operating Instructions 6710.201 “Functions and Parameters”.

5.3.3 Commissioning of the CANopen System (Master, Other Devices)

The commissioning of the CANopen system with the master and other devices is the last commissioning step.

For a facilitated commissioning, so-called electronic data sheets (EDS) permitting an easy access to the servo drive can be generated with SPP Windows.

6 Network Management and COB-ID Assignment

6.1 NMT Node State Diagram

The CANopen interface communication behavior of a device is influenced by so-called network management functions (NMT services). The following state diagram represents the communication behavior of the servo drive and possible actions to influence it. For basic information on state machines see Appendix A (page 52).

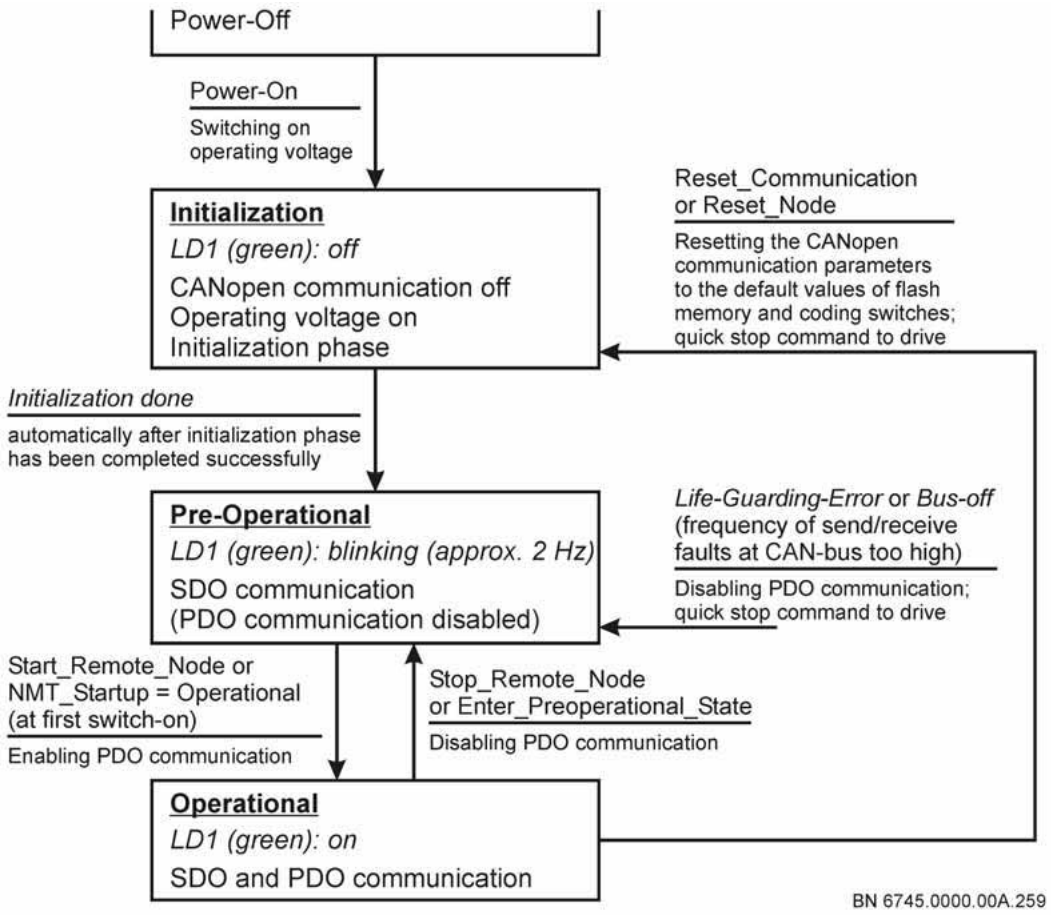

Figure 1: NMT Node State Diagram Servo Drive

The figure shows the states and state transitions of the NMT Node State Diagram of CANopen Communication Profile CiA 301 which are supported by the servo drive.

After switch-on of the control voltage and terminated internal initialization of the servo drive firmware, the device automatically reaches state “pre-operational”. Automatic switching to “operational” is possible with variable *NMT Startup*.

The other state transitions are usually triggered by NMT services sent by the NMT master:

- Start_Remote_Node
- Stop_Remote_Node
- Enter_Preoperational_State
- Reset_Communication
- Reset_Node

In state “pre-operational”, only SDO communication is possible, PDO communication is blocked. In state “operational”, PDO communication takes place, as well.

The currently active state is displayed by the green Run LED, see section 5.2 (page 16).

6.2 COB-ID Assignment

The COB-IDs for the CANopen communication objects supported by the device are assigned by the manufacturer depending on the set Node-ID. This assignment is called “Predefined Connection Set”. As far as stated, the COB-IDs can be changed via variables.

COB-ID according to the “Predefined Connection Set” (CANopen COBs)		
CANopen communication object	COB-ID (hex.)	COB-ID can be changed via variable
NMT	0	–
SYNC	80	COB-ID Sync Message
EMCY	80 + Node-ID	–
RPDO 1	200 + Node-ID	Receive PDO 1 Comm. Parameter
RPDO 2	300 + Node-ID	Receive PDO 2 Comm. Parameter
RPDO 3	400 + Node-ID	Receive PDO 3 Comm. Parameter
RPDO 4	500 + Node-ID	Receive PDO 4 Comm. Parameter
TPDO 1	180 + Node-ID	Transmit PDO 1 Comm. Parameter
TPDO 2	280 + Node-ID	Transmit PDO 2 Comm. Parameter
TPDO 3	380 + Node-ID	Transmit PDO 3 Comm. Parameter
TPDO 4	480 + Node-ID	Transmit PDO 4 Comm. Parameter
SDO (Tx)	580 + Node-ID	–
SDO (Rx)	600 + Node-ID	–
Node Guarding	700 + Node-ID	COB-ID Guarding Protocol

7 SDO Communication (Parameters)

All drive variables and CANopen variables of the servo drive can be accessed by SDO communication.

TIP The CANopen variables are described in section 14 (page 30ff) of these operating instructions, the drive variables are described in Operating Instructions 6710.201 “Functions and Parameters”.

In SDO communication, the servo drive is a “server” in the sense of CANopen Communication Profile CiA 301. That means the servo drive permits other devices, called “clients”, reading or writing access to its variables.

? SDO communication is possible in NMT Node State Diagram states “pre-operational” and “operational”. These states are displayed by flashing or lighting of the green Run LED.

The SDO communication is handled with the “multiplexed domain protocol” according to CANopen Communication Profile CiA 301. As described there, the “expedited” variant of the protocol is used for variables of up to 4 byte. The SDO communication uses 2 COB-IDs, one for each transmission direction of the protocol. The actual COB-IDs depend on the Node-ID of the servo drive. The COB-IDs can also be queried via variables *SDO 1 Server Parameter* and *Server SDO 2 Server Parameter*.

TIP The COB-IDs used for SDO communication only depend on the Node-ID which is set via coding switches. They cannot be changed by parameterization. Thus, all variables can be accessed independent of the further configuration of the CANopen interface if just Node-ID of the device and transmission type are known. Therefore, to reach a maximum flexibility in exchanging devices, it is useful to transmit the CANopen machine data to the servo drive via SDO communication after each switch-on. So, only Node-ID and transmission rate have to be set to configure the CANopen interface in case of an exchange of devices. The same applies to the transmission of the remaining device data.

For a description of variables *SDO 1 Server Parameter* and *SDO 2 Server Parameter* used for the SDO communication, see section 14.2 (page 34).

8 PDO Communication (Process Data)

PDO communication is used for a fast real-time access to selected variables.

According to the following table, seven mappings of variables on PDOs are predefined in sending and receiving direction each; however, the PDO mapping can also be configured as desired (variable PDO mapping).

The following table shows the default setting.

PDO Designation	Description				
	Name	Index	Subindex	Length (Byte)	Status
RPDO1	Axis control word	6040	0	2	active
RPDO2	Axis control word	6040	0	2	inactive
	Axis operating mode	6060	0	2	inactive
RPDO3	Axis control word	6040	0	2	inactive
	Target position	607a	0	4	inactive
RPDO4	Axis control word	6040	0	2	inactive
	Target velocity	6081	0	4	inactive
TPDO1	Axis status word	6041	0	2	active
TPDO2	Axis status word	6041	0	2	inactive
	Operating mode selection code	6061	0	2	inactive
TPDO3	Axis status word	6041	0	2	inactive
	Actual position (in PU)	6064	0	4	inactive
TPDO4	Axis status word	6041	0	2	inactive
	Actual velocity	606c	0	4	inactive

The naming of the PDOs corresponds to the conventions of CANopen Drive Profile CiA 402.

PDO communication is possible in NMT Node State Diagram state “operational”, only. This state is displayed by lighting of the green Run LED.

The variables mapped on a PDO as described in the table above can be read via *Receive PDO 1 Mapping* to *Transmit PDO 4 Mapping*. Via *Receive PDO 1 Comm. Parameter* to *Transmit PDO 4 Comm. Parameter*, the behavior of the receive or transmit PDOs can be queried and set.

By default, one PDO is active (valid) in sending and receiving direction each: RPDO1 and TPDO1. All other PDOs are not active. Via communication parameters, each PDO can be activated or deactivated individually.

By default, the COB-IDs of the PDOs are derived from the Node-ID.

The default transmission type of the PDOs is asynchronous. Every time the value of one of the variables mapped on the PDO changes, the servo drive triggers a transmission of the corresponding PDO. If required, a synchronous transmission type can be selected via the communication parameters.

To avoid an overload of the CAN bus by Transmit PDOs with frequently changing values (e. g. actual position), the transmission frequency can be limited by the so-called *Inhibit Time*. The unit of the *Inhibit Time* is 100 µs according to CiA 301 (i. e. 10 ms correspond to a value of 100). The evaluation in the servo drive is carried out at a resolution of approx. 3 ms so that only changes of the *Inhibit Time* setting by 30 or more have an effect. The *Inhibit Time* can be set individually for each *Transmit PDO* via the communication parameters.

For a description of variables *Receive PDO 1* to *4* and *Transmit PDO 1* to *4*, please see section 14.3 (page 35ff).

The standard assignment of the PDOs can be changed using the *PDO Mapping Parameters* (variable PDO mapping). 4 RPDOs and 4 TPDOs with a maximum of 8 entries per PDO are available. The following parameters can be mapped in PDOs:

Index	Object	Type	PDO Direction
1001	Error Register	u8	TPDO
5e93	I2t Load	u16	TPDO
5e95	Position sensor measured value2	i32	TPDO
5e96	Position sensor measured value2 PU	i32	TPDO
5e9f	Position sensor measured value1 PU	i32	TPDO
5f03	Resistor motor temperature sensor	u32	TPDO
5f0b	Heat sink temperature	i16	TPDO
5f0c	Motor temperature	i16	TPDO
5f1c	Position sensor measured value	i32	TPDO
603f	Fault code	u16	TPDO
6041	Status word	u16	TPDO
6061	Modes of operation display	i16	TPDO
6063	Position sensor actual position	i32	TPDO
6064	Actual position	i32	TPDO
606c	Actual velocity	i32	TPDO
6078	Actual current	i16	TPDO
6079	Bus voltage	u16	TPDO

Index	Object	Type	PDO Direction
5f54	Digital outputs	u8 (array)	TPDO, RPDO
5f56	Digital inputs	u8 (array)	TPDO, RPDO
5f5e	Program variables	i32 (array)	TPDO, RPDO
5e9c	Max current amount2	u16	RPDO
5ef4	Acceleration time	u32	RPDO
5ef5	Deceleration time	u32	RPDO
5ef6	Quick stop time	u32	RPDO
5ef7	Velocity reference value	u32	RPDO
5fc1	Alternative interpolation data record	i16/i8	RPDO
6040	Control word	u16	RPDO
6060	Operating mode selection code	i16	RPDO
6071	External torque setpoint	i16	RPDO
6073	Max current amount	u16	RPDO
607a	Target position	i32	RPDO
6081	Target velocity	i32	RPDO
6086	Motion profile type	i16	RPDO
60b1	Velocity offset	i32	RPDO
60c1	Interpolation data record	i32	RPDO
5fc2	Alternative for velocity offset	i16/i8	RPDO

9 Synchronization

9.1 Asynchronous Transmission of PDOs

By default, the PDOs (process data) of the servo drives are set to asynchronous transmission via the corresponding communication parameters (*Transmission Type* = 255). That means:

- *Receive PDOs* (PDOs received by the servo drive) are accepted by the servo drive upon arrival. The sending device determines when the data are sent (typically each time the value of one of the variables mapped on the PDO changes).
- *Transmit PDOs* (PDOs sent by the servo drive) are sent by the servo drive when the value of one of the variables mapped on the PDO has changed. An overload of the bus by frequently changing variables can be inhibited by limiting the transmission frequency with an *Inhibit Time*. Using the *Event Timer*, you can define the point of time a PDO is sent at the latest even if the value of the variable has not changed.

9.2 Synchronous Transmission of PDOs

The PDOs can also be transmitted synchronously to a SYNC message (*Transmission Type* = 1 .. 240). The SYNC message is sent by exactly one of the devices at the CAN bus, the so-called SYNC Master, at defined intervals. If a SYNC master does not exist (no SYNC message), only asynchronous PDOs can be transmitted.

The transmission is carried out cyclically, value n of the *Transmission Type* states the number of SYNC messages between two PDO transmissions.

- *Receive PDOs* (PDOs received by the servo drive) are stored temporarily by the servo drive upon arrival. The data become valid with the arrival of the n^{th} SYNC message, n corresponds to the value of the *Transmission Type* (value range 1 .. 240).
- The data for *Transmit PDOs* (PDOs sent by the servo drive) are acquired and sent upon arrival of the n^{th} SYNC message.

Note: The CANopen time stamp is not supported. *Inhibit Time* and *Event Timer* do not have any meaning for synchronous PDO transmission.

For a description of variable *COB-ID SYNC Message*, see section 14.4 (page 42).

10 Interpolated Position Mode

Axis operating mode “Interpolated Position Mode” serves for operating one or several axes for which a time interpolation of the setpoints is required (e. g. for contouring control). For that, the higher-level controller sends target positions to the drive at defined intervals via an RPDO. In addition to these positions, the fine interpolator of the drive calculates further target positions at the time interval of the position control loop (1 ms). In interpolated position mode, the target positions are transmitted in position sensor steps (PSS), only, not in position units (pU).

In interpolated position mode, the PDOs are transmitted synchronously to a SYNC message. In case of a failure or a breakdown of the SYNC transmission, the drive keeps the last target position until it receives a new SYNC message.

Monitoring of hardware limit switches and function “measure position” are not active in operating mode interpolated position mode.

In the axis status word, operating mode-related bits do not exist for the interpolated position mode.

In the axis control word, bit 4 has the following special meaning:

- enable interpolation

  The interpolation is started by a rising edge (0 to 1) in bit “enable interpolation”.

The axis operating mode can only be changed when the state of the axis state machine is “switch on disabled”, “ready to switch on”, or “switched on”.

Only linear axes are supported in the interpolated position mode, an operation as circular axis (endless axis) is not possible.

In operating mode “Interpolated Position Mode”, the CANopen communication is influenced by the following parameters:

- Mapping PDOs
  - serve for the transmission of axis status word and axis control word as well as target positions (in *Interpolation Data Record*) and actual positions.

  The mapping PDOs are described in section 14.3.1 (page 35ff).
- Machine data PDOs (comm. parameters)
  - control the synchronous transmission of above-mentioned PDOs.

  The machine data PDOs are described in section 14.3.2 (page 38ff).
- Machine data interpolated position mode with *Interpolation Time Period*
  - determine the time interval for setting the target positions in milliseconds.

For a description of the machine data interpolated position mode, see section 14.5.2 (page 44ff).

Due to the variable PDO mapping, PDOs for the interpolated position mode are no longer defined. For mapping, please note that the objects *GeraetSteuer* (index 6040, subindex 0) and *IpolDataRecord* (index 60c1, subindex 1) must be mapped in an RPDO, objects *GeraetStatus* (index 6041, subindex 0) and *PosIstLg* (index 6063, subindex 1) must be mapped in a TPDO. They are transmitted synchronously, i. e. value n of the *Transmission Type* = 1.

In axis operating mode Interpolated Position Mode, the fine interpolator acts as setpoint generator together with the position controller, speed and current controller are subordinate to the position controller.

In axis operating mode Interpolated Position Mode, the servo drives can be operated with controllers of different manufacturers. If required, please contact ESR.

10.1 Activating the Interpolated Position Mode

To control the axis state machine, the corresponding PDOs must be set to “valid”.

1. If not set via machine data *Axis operating mode at startup*, select axis operating mode Interpolated Position Mode.
2. With enabling the PDO communication by the NMT master, the state machine of the NMT services is switched to “operational”.
3. For the initialization of the interpolated position mode, read out the current actual position and write it into the *Interpolation Data Record*.
4. Switch the axis state machine to “operation enabled”.
5. Activate the interpolation via “enable interpolation” (bit 4 in the axis control word).

10.2 Deactivating the Interpolated Position Mode

For deactivating the interpolated position mode, the axis state machine has to be switched to state “switch on disabled”, “ready to switch on”, or “switched on”. If required, switch the NMT state machine to “pre-operational”.

After that, the axis operating mode can be changed.

11 Monitoring Mechanisms

11.1 Fault Handling (Error, Emergency)

Each fault is transmitted to the master by an emergency telegram. This emergency telegram has the following structure:

- Byte 0, 1: Pre-defined error field, subindex 1 (fault code)
- Byte 2: Error register
- Byte 3 .. 7: not used (0)

Bytes 0 und 1 contain the fault code as reported in CANopen variable *Pre-Defined Error Field*, subindex 1.

Byte 2 contains the contents of CANopen variable *Error Register.*

Bytes 3 to 7 of the emergency telegram are not used.

For a description of the variables *Error Register*, *Pre-Defined Error Field*, and *COB-ID Emergency Message* which are used for monitoring, see section 14.6.2 (page 46ff).

11.2 Node Guarding, Life Guarding

The servo drive supports node guarding/life guarding as defined in CANopen Communication Profile CiA 301.

When the monitoring of the master is enabled (CANopen variables *Guard Time* and *Life Time Factor* are not zero), the servo drive reacts with a fault reaction if the master has not sent a guarding message within a certain period of time (life time):

- if the *abort connection option code* is set to 1, the drive is stopped with a quick-stop command, a possibly running part program is stopped,
- fault code 8130 is displayed,
- a transition into state “pre-operational” is carried out in the NMT node state diagram.

For a description of the variables *Guard Time* and *Life Time Factor* which are used for monitoring, see section 14.6.3 (page 47).

11.3 Heartbeat

The servo drive supports the heartbeat producer procedure defined in CANopen communication profile CiA 301.

When the heartbeat producer is released (CANopen variable *Producer heartbeat time* unequal 0), a heartbeat telegram with COB-ID 700 + Node-ID and the following contents is sent cyclically according to the specified time:

- Byte 0: 127 (pre-operational) or 5 (operational)

For a description of variable *Producer heartbeat time*, see section 14.6.3 (page 48).

11.4 Send and Receive Monitoring

Each CAN controller automatically monitors the sending and receiving data for faults. Sending or receiving faults are displayed in two levels:

- When sending and receiving faults occur frequently, fault code 8100 is displayed as a warning (CAN term: error passive). The devices continues operation as before the warning.
- When the number of sending and receiving faults increases and exceeds a certain limit, the CAN controller disconnects itself from the bus (CAN term: bus off). Related with that is a fault reaction:
  - the drive is stopped with a quick-stop command, a possibly running part program is stopped,
  - fault code 8180 is displayed,
  - a transition into state pre-operational is carried out in the NMT node state diagram.

The warnings and faults described above are displayed by flashing or lighting of the red Error LED.

In case of the warnings and faults described above, also check the connections of all other devices to the CAN bus when searching for the fault.

12 Memory Functions

With the memory functions, CANopen machine data (and other device data) can be stored in the non-volatile memory of the servo drive (servo EPROM).

To reach a maximum flexibility for the exchange of devices, it is useful to send the CANopen machine data to the servo drive by the CANopen master via SDO communication after each switch-on of the system.

For a description of the variables *Store Parameters* and *Restore Default Parameters* which are used for the memory functions, see section 14.7 (page 48ff).

12.1 Memory Functions Command and Commissioning Software SPP Windows

The CANopen machine data are usually stored in the non-volatile memory of the servo drive (servo EPROM) using variable *Store Parameters*. As soon as the CANopen machine data have been stored once in the servo EPROM, they are stored additionally from the RAM of the servo drive into the servo EPROM via menu item “Communication/Save in device” of command and commissioning software SPP Windows.

13 Device Information

Information on connected devices such as device type and manufacturer are also included in variables.

For a description of the variables *Device Type*, *Manufacturer Device Name*, *Manufacturer Hardware Version*, *Manufacturer Software Version*, and *Identity* which are used for device information, see section 14.8 (page 49ff).

14 Description of Variables

In the following sections, you will find information on the CANopen variables described in these operating instructions. In this part, the variable descriptions are sorted according to functions. Access according to the name is possible via the keyword index (page 62). In Appendix B, all CANopen variables are listed according to the index.

The description of variables is structured according to a standardized scheme. With corresponding adaptations, this scheme is also used for the structure of the list of variables in Appendix B (page 53) as well as for the description of variables in Operating Instructions 6710.201 “Functions and Parameters”.

Basic information on a variable is summarized in a table of the following form. The values stated in this table are always default values.

Name	Index: 1234, Short name: VarName
SPP Windows	...
Type	...
Access	...
Var. type	...
Unit	...
Default value	...

Name

The variable name is printed in bold type at the left top of each variable description. In the describing text, the variable name is printed in italics.

Index

The index is a hexadecimal number by which the variable can be accessed in SDO communication via the CAN bus.

Short name

This is a short name intended for a future use with special high-level language driver programs.

SPP Windows

In the SPP Windows command and commissioning software, the variable is displayed as stated in these operating instructions. As an example, this line contains the default factory setting (underlined) which can be restored via

access to variable *Restore Default Parameters* at a later time. If a unit is assigned to this value, it is stated together with the value.

Type

This line is composed of various information:

- Object code:
  - Simple variable

    The simple variable contains one element of the stated data type each. In case of access to the simple variable, subindex 0 must always be stated.
  - Array

    The array contains several elements of the stated data type. In case of access to the array, the subindex of the desired element (1 or higher) has to be stated. Especially with CANopen variables, the number of elements of the array can be read out via subindex 0.
  - Record

    A record contains several elements of different data types. In case of access to the record, the subindex of the desired element (1 or higher) has to be stated. Especially with CANopen variables, the number of elements of the record can be read out via subindex 0.
- Data type:

  Possible values:
  - Boolean (Bool)
  - Integer8 (i8) = byte in double complement representation
  - Integer16 (i16) = word in double complement representation
  - Integer32 (i32) = double word in double complement representation
  - Unsigned8 (u8) = byte, without sign
  - Unsigned16 (u16) = word, without sign
  - Unsigned32 (u32) = double word, without sign
  - Visible string (VisStr) with length = a series of bytes containing text
  - Octet string (OctStr) with length = a series of bytes containing binary coded information
  - PDO CommPar (PDOCoP), see section 14.3.2 (page 38ff)PDO Mapping (PDOMap), see section 14.3.1 (page 35ff)
  - SDO Parameter (SDOPar), see section 14.2 (page 34)

Access, R/W

Possible values:

– read and write, short R W

– read only, short R

Var. type

Possible values (with characters as abbreviations):

– constant (variable value does not change), short “F”

– machine data CANopen (all machine data described in these operating instructions), short “MC “

– machine data, short “M”

– status information, short “S”

– control information, short “S”

– variable value, short “V”

Array element n

In case the individual elements of the array have certain names, units, and/or default values, these are stated in this line.

Record element n

In case the individual elements of the record have certain names, units, and/or default values, these are stated in this line.

Description

Depending on the variable, a short describing text appears before and/or after the table.

Example of a variable description:

Receive PDO 1 Mapping	Index: 1600, Short name: RPDO1Map
SPP Windows	RPDO 1 Mapping 1 60400010
Type	record, 1 element (subindex 0 .. 1), type PDO Mapping
Record element 1	default: axis control word (index 6040, subindex 0, length 16 Bit)
Access	read only
Var. type	constant (variable value does not change)

This variable shows the variable mapped on *Receive PDO 1*.

14.1 Description of Variables Network Management

Node-ID	Index: 5fb4, Short name: CANNodeId
SPP Windows	Node-Id SDO Server 1 01 Node-Id SDO Server 2 00
Type	array, 2 elements (subindex 0 .. 2), U8
Array element 1	Node-ID SDO Server 1
Array element 2	Node-ID SDO Server 2
Access	read only
Var. type	changing value

The set Node-ID can be read via this variable. In these operating instructions, Node-IDs are always stated as hexadecimal values. If *Node-ID SDO Server 1* is written with bit 7 set, the value of the „Predefined Connection Set“ is preset for this node-ID.

Changes in variables *Node-ID SDO Server 1* and *CAN Baudrate* become effective with the next switch-on of the device, *Node-ID Server 2* is immediately effective.

Value 0 in Node-ID2 deactivates server 2.

NodeID	Index: 100b, Short name: NodeId
SPP Windows	Node-Id SDO Server 1 01 Node-Id SDO Server 2 00
Type	simple variable 32 bit
Access	read only
Var. type	changing value

This variable is only available for reasons of compatibility with the old DS 301 version 3.0 and CodeSys. Its function corresponds to the one of *Node-ID*, index 5fb4.

CAN Baudrate	Index: 5fb5, Short name: CANBaudrate
SPP Windows	CAN Baudrate 500 Kbps
Type	simple variable, U8
Access	read and write
Var. type	changing value

The baud rate can be set via this variable.

NMT Startup	Index: 1f80, Short name: NMTStartup
SPP Windows	NMT Startup Pre-Operational
Type	simple variable, U32
Access	read and write
Var. type	changing value

This variable is used to define to which NMT Node State Diagram state the servo drive has to change after switch-on.

Permissible values are

Value	State
12	pre-operational
8	operational

Changes in this variable become effective with the next switch-on of the device.

14.2 Description of Variables SDO Communication

Server SDO 1 Parameter	Index: 1200, Short name: SSDO1
SPP Windows	–
Type	record, 2 elements (subindex 0 .. 2), type SDO Parameter
Record element 1	COB-ID client → server, 600 + Node-ID, U32
Record element 2	COB-ID server → client, 580 + Node-ID, U32
Access	read only
Var. type	variable value

Server SDO 2 Parameter	Index: 1201, Short name: SSDO2
SPP Windows	–
Type	record, 3 elements (subindex 0 .. 3), type SDO Parameter
Record element 1	COB-ID client → server, 600 + Node-ID, U32
Record element 2	COB-ID server → client, 580 + Node-ID, U32
Record element 3	SDO2 Node-ID, U8
Accessf	read/write
Var. type	variable value

Server SDO 1 Parameter and *Server SDO 2 Parameter* show the COB-IDs for server 1 and server 2 which are used for SDO communication.

The values depend on the setting of the Node-ID via coding switches S1 and S2. The two record elements have the following structure:

Record element 1, 2 (u32)	bit 31:	0 = SDO valid
	bit 30 .. 12:	reserved (always 0)
	bit 11 .. 0:	COB-ID SDO per direction

In these operating instructions, COB-IDs are always stated as hexadecimal values.

14.3 Description of Variables PDO Communication

14.3.1 PDOs Mapping

14.3.1.1 Receive PDOs Mapping

The PDOs described in the following can be found in SPP Windows in window “Parameterization” on “CANopen/Receive PDO Mapping”.

The variable description of *Receive PDO...Mapping* shows which function is accessed via the respective PDO (e. g. *Axis Control Word*). These variables are records with a type unsigned32 element for each variable mapped on the PDO. The following is entered for each mapped variable:

Record element x (u32)	bit 31 .. 16:	index (16 bit)
	bit 15 .. 8:	subindex (8 bit)
	bit 7 .. 0:	length in bit (8 bit)

The number of mapped variables (corresponds to the number of elements of the record) can be read via subindex 0.

Receive PDO 1 Mapping	Index: 1600, Short name: RPDO1Mapping
SPP Windows	RPDO1 Mapping 1 60400010
Type	record, max. 8 elements (subindex 0 .. 8), type PDO Mapping
Record element 0	number of elements (0–8), default = 1
Record element 1	default: Axis Control Word (index 6040, subindex 0, length 16 bit)
Access	read/write
Var. type	CANopen machine data

Receive PDO 2 Mapping	Index: 1601, Short name: RPDO2Mapping
SPP Windows	RPDO2 Mapping 1 60400010 RPDO2 Mapping 2 60600010
Type	record, max. 8 elements (subindex 0 .. 8), type PDO-Mapping
Record element 0	number of elements (0–8), default = 2
Record element 1	default: Axis Control Word (index 6040, subindex 0, length 16 bit)
Record element 2	default: Axis Operating Mode (index 6060, subindex 0, length 16 bit)
Access	read/write
Var. type	CANopen machine data

Receive PDO 3 Mapping	Index: 1602, Short name: RPDO3Mapping
SPP Windows	RPDO3 Mapping 1 60400010 RPDO3 Mapping 2 607A0020
Type	record, max. 8 elements (subindex 0 .. 8), type PDO Mapping
Record element 0	number of elements (0–8), default = 2
Record element 1	default: Axis Control Word (index 6040, subindex 0, length 16 bit)
Record element 2	default: Target Position (index 607A, subindex 0, length 32 bit)
Access	read/write
Var. type	CANopen machine data

Receive PDO 4 Mapping	Index: 1603, Short name: RPDO4Mapping
SPP Windows	RPDO4 Mapping 1 60400010 RPDO4 Mapping 1 60810020
Type	record, 8 elements (subindex 0 .. 8), type PDO Mapping
Record element 0	number of elements (0–8), default = 2
Record element 1	default: Axis Control Word (index 6040, subindex 0, length 16 bit)
Record element 2	default: Target Velocity (index 6081, subindex 0, length 32 bit)
Access	read/write
Var. type	CANopen machine data

The standard mapping described above can be overwritten (variable mapping). To do so, reset the corresponding record element as described above. To change the mapping, set the number of mapped variables (subindex 0) to 0. After having changed the mapping, the number of variables must be set to the corresponding number of mapped elements (0–8).

14.3.1.2 Transmit PDOs Mapping

The PDOs described in the following can be found in SPP Windows in window “Parameterization” on “CANopen/Transmit PDO Mapping”.

The variable description of *Transmit PDO...Mapping* shows which function is accessed via the respective PDO. These variables are records with a type unsigned32 element for each variable mapped on the PDO. The following is entered for each mapped variable:

Record element x (u32)	bit 31 .. 16:	index (16 Bit)
	bit 15 .. 8:	subindex (8 Bit)
	bit 7 .. 0:	length in bit (8 Bit)

The number of mapped variables (corresponds to the number of elements of the record) can be read via subindex 0.

Transmit PDO 1 Mapping	Index: 1a00, Short name: TPDO1Mapping
SPP Windows	TPDO1 Mapping 1 60410010
Type	record, max. 8 elements (subindex 0 .. 8), type PDO Mapping
Record element 0	number of elements (0–8), default = 1
Record element 1	default: Axis Status Word (index 6041, subindex 0, length 16 Bit)
Access	read/write
Var. type	CANopen machine data

Transmit PDO 2 Mapping	Index: 1a01, Short name: TPDO2Mapping
SPP Windows	TPDO2 Mapping 1 60410010 TPDO2 Mapping 2 60610010
Type	record, max. 8 elements (subindex 0 .. 8), type PDO Mapping
Record element 0	number of elements (0–8), default = 2
Record element 1	default: Axis Status Word (index 6041, subindex 0, length 16 Bit)
Record element 2	default: Operating Mode Selection Code (index 6061, subindex 0, length 16 Bit)
Access	read/write
Var. type	CANopen machine data

Transmit PDO 3 Mapping	index: 1a02, Short name: TPDO3Mapping
SPP Windows	TPDO3 Mapping 1 60640020 TPDO3 Mapping 2 606C0020
Type	number of elements (0–8), default = 2
Record element 0	number of elements (0–8), default = 2
Record element 1	default: Axis Status Word (index 6041, subindex 0, length 16 bit)
Record element 2	default: Actual Position (index 6064, subindex 0, length 32 Bit)
Access	read/write
Var. type	CANopen machine data

Transmit PDO 4 Mapping	index: 1a03, Short name: TPDO4Mapping
SPP Windows	TPDO4 Mapping 1 60410010 TPDO4 Mapping 2 606C0020
Type	record, max. 8 elements (subindex 0 .. 8), type PDO Mapping
Record element 0	number of elements (0–8), default = 2
Record element 1	default: Axis Status Word (index 6041, subindex 0, length 16 bit) default: Actual Velocity (index 606c, subindes 0, length 32 bit)
Access	read/write
Var. type	CANopen machine data

The standard mapping described above can be overwritten (variable mapping). To do so, reset the corresponding record element as described above. To change the mapping, set the number of mapped variables (subindex 0) to 0. After having changed the mapping, the number of variables must be set to the corresponding number of mapped elements (0–8).

14.3.2 Machine Data PDOs (Comm. Parameter)

14.3.2.1 Receive PDOs Comm. Parameter

The PDOs described in the following can be found in SPP Windows in window “Parameterization” on “CANopen/Receive PDOs”.

The behavior of the *Receive PDOs* can be queried and set via variables *Receive PDO...Comm. Parameter*. The variables contain the following elements:

Record element 1 (u32)	bit 31:	1 = PDO not valid 0 = PDO valid
	bit 30:	0 = RTR allowed
	bit 20 .. 12:	reserved (always 0)
	bit 11 .. 0:	COB-ID of the PDO
Record element 2 (u8)	transmission type:	0 = synchronous, acyclic
		1 .. 240 = synchronous, cyclic
		255 = asynchronous

With bit 31 of record element 1, the PDO can be activated (valid) or deactivated (not valid). If a *Receive PDO* is deactivated, the COBs (CAN messages) transmitted to the COB-ID of the PDO do not have any effect on the mapped variables.

Bit 30 of record element 1 shows whether or not a remote transmission request (RTR), i. e. a query of the PDO by sending a certain COB is allowed. By default, it is set to “RTR allowed”.

The COB-ID of the PDO is set by bit 11 .. 0 of record element 1. Generally, the default value which depends on the set Node-ID should not be changed. For further information, please see section 6.2 (page 19). In these operating instructions, COB-IDs are always stated as hexadecimal values. The COB-ID must not be changed unless the PDO is deactivated (bit 31 = 1).

The transmission type is set with record element 2. By default, *Transmission Type* 255 (asynchronous PDO, profile-specific event) is set for all *Receive PDOs*. The event depends on the sending device.

For *Receive PDOs*, the setting of an *Inhibit Time* via record element 3 is not required as the sender of the PDO is responsible for setting this time.

Receive PDO 1 Comm. Parameter	Index: 1400, Short name: RPDO1ComParm
SPP Windows	RPDO1 COB-ID 00000200 + Node-ID RPDO1 Transmission Type 255
Type	record, 2 elements (subindex 0 .. 2), type PDO CommPar
Record element 1	default: valid (0) / RTR (0) / COB-ID = 200 + Node-ID
Record element 2	default: transmission type = asynchronous (255)
Access	read/write
Var. type	CANopen machine data

Receive PDO 2 Comm. Parameter	Index: 1401, Short name: RPDO2ComParm
SPP Windows	RPDO2 COB-ID 80000300 + Node-ID RPDO2 Transmission Type 255
Type	record, 2 elements (subindex 0 .. 2), type PDO CommPar
Record element 1	default: not valid (1) / RTR (0) / COB-ID = 300 + Node-ID
Record element 2	default: transmission type = asynchronous (255)
Access	read/write
Var. type	CANopen machine data

Receive PDO 3 Comm. Parameter	Index: 1402, Short name: RPDO3ComParm
SPP Windows	RPDO3 COB-ID 80000400 + Node-ID RPDO3 Transmission Type 255
Type	record, 2 elements (subindex 0 .. 2), type PDO CommPar
Record element 1	default: not valid (1) / RTR (0) / COB-ID = 400 + Node-ID
Record element 2	default: transmission type = asynchronous (255)
Access	read/write
Var. type	CANopen machine data

Receive PDO 4 Comm. Parameter		Index: 1403, Short name: RPDO4ComParm
SPP Windows	RPDO4 COB-ID RPDO4 Transmission Type	80000500 + Node-ID 255
Type	record, 2 elements (subindex 0 .. 2), type PDO CommPar	
Record element 1	default: not valid (1) / RTR (0) / COB-ID = 500 + Node-ID	
Record element 2	default: transmission type = asynchronous (255)	
Access	read/write	
Var. type	CANopen machine data	

14.3.2.2 Transmit PDOs Comm. Parameter

The PDOs described in the following can be found in SPP Windows in window “Parameterization” on “CANopen/Transmit PDOs”.

The behavior of the *Transmit PDOs* can be queried and set via variables *Transmit PDO...Comm. Parameter*. The variables contain the following elements:

Record element 1 (u32)	bit 31:	1 = PDO not valid 0 = PDO valid
	bit 30:	0 = RTR allowed
	bit 20 .. 12:	reserved (always 0)
	bit 11 .. 0:	COB-ID of the PDO
Record element 2 (u8)	transmission type:	0 = synchronous, acyclic
		1 .. 240 = synchronous, cyclic
		255 = asynchronous
Record element 3 (u16)	inhibit time (in ms):	0 = no inhibit time
Record element 4	not implemented	–
Record element 5 (u16)	event timer (in ms):	0 = no event timer

With bit 31 of record element 1, the PDO can be activated (valid) or deactivated (not valid). If a *Transmit PDO* is activated, the related COB (the related CAN message) is sent to the CAN bus upon certain events and/or at certain times. The point of time of sending depends on the setting of *Transmission Type* and, if applicable the *Inhibit Time* and the *Event Timer* of the PDO.

Bit 30 of record element 1 shows whether or not a remote transmission request (RTR), i. e. a query of the PDO by sending a certain COB is allowed. By default, it is set to “RTR allowed”.

The COB-ID of the PDO is set by bit 11 .. 0 of record element 1. Generally, the default value which depends on the set Node-ID should not be changed. For further information, please see section 6.2 (page 19). In these operating instructions, COB-IDs are always stated as hexadecimal values.

The transmission type is set with record element 2. By default, *Transmission Type* 255 (asynchronous PDO, profile-specific event) is set for all *Transmit PDOs*. The event triggering a transmission of a *Transmit PDO* by the servo drive is a change of a value of one of the variables mapped on the PDO.

With *Transmit PDOs*, variables with frequently changing values may overload the CAN bus. To avoid this, the transmission frequency can be limited via the *Inhibit Time* of record element 3. The unit of the *Inhibit Time* is 100 µs according to CiA 301 (i. e. 10 ms correspond to a value of 100). The evaluation in the servo drive is carried out at a resolution of approx. 3 ms so that only changes in the *Inhibit Time* setting by 30 or more have an effect.

For asynchronous PDOs, the *Event Timer* of record element 5 is used to set the time (in ms) after which the PDO is sent at the latest.

Transmit PDO 1 Comm. Parameter		Index: 1800, Short name: TPDO1ComParm
SPP Windows	TPDO1 COB-ID TPDO1 Transmission Typee TPDO1 Inhibit Time TPDO1 Event Timer	00000180 + Node-ID 255 0,0 ms 0 ms
Type	record, 5 elements (subindex 0 .. 5), type PDO CommPar	
Record element 1	default: valid (0), RTR (0), 180 + Node-ID	
Record element 2	default: transmission typee = asynchron (255)	
Record element 3	default: inhibit time = 0 ms	
Record element 4	not implemented	
Record element 5	default: event timer = 0 ms	
Access	read/write	
Var. type	CANopen machine data	

Transmit PDO 2 Comm. Parameter		Index: 1801, Short name: TPDO2ComParm
SPP Windows	TPDO2 COB-ID TPDO2 Transmission Typee TPDO2 Inhibit Time TPDO2 Event Timer	80000280 + Node-ID 255 0,0 ms 0 ms
Type	record, 5 elements (subindex 0 .. 5), type PDO CommPar	
Record element 1	default: not valid (1), RTR (0), 280 + Node-ID	
Record element 2	default: transmission typee = asynchron (255)	
Record element 3	default: inhibit time = 0 ms	
Record element 4	not implemented	
Record element 5	default: event timer = 0 ms	
Access	read/write	
Var. type	CANopen machine data	

Transmit PDO 3 Comm. Parameter	Index: 1802, Short name: TPDO3ComParm
SPP Windows	TPDO3 COB-ID 80000380 + Node-ID TPDO3 Transmission Typee 255 TPDO3 Inhibit Time 0,0 ms TPDO3 Event Timer 0 ms
Type	record, 5 elements (subindex 0 .. 5), type PDO CommPar
Record element 1	default: not valid (1), RTR (0), 380 + Node-ID
Record element 2	default: transmission typee = asynchron (255)
Record element 3	default: inhibit time = 0 ms
Record element 4	not implemented
Record element 5	default: event timer = 0 ms
Access	read/write
Var. type	CANopen machine data

Transmit PDO 4 Comm. Parameter	Index: 1803, Short name: TPDO4ComParm
SPP Windows	TPDO4 COB-ID 80000480 + Node-ID TPDO4 Transmission Typee 255 TPDO4 Inhibit Time 0,0 ms TPDO4 Event Timer 0 ms
Type	record, 5 elements (subindex 0 .. 5), type PDO CommPar
Record element 1	default: not valid (1), RTR (0), 480 + Node-ID
Record element 2	default: transmission typee = asynchron (255)
Record element 3	default: inhibit time = 0 ms
Record element 4	not implemented
Record element 5	default: event timer = 0 ms
Access	read/write
Var. type	CANopen machine data

14.4 Description of Variables Synchronization

14.4.1 Machine Data Synchronization

This variable can be found in SPP Windows in window “Parameterization” on “CANopen/Synchronization”.

COB-ID SYNC Message	Index: 1005, Short name: IdSync
SPP Windows	COB-ID SYNC Message 00000080
Type	simple variable, unsigned32
Access	read/write
Var. type	CANopen machine data
Default value	80

This variable is used for querying and setting the COB-ID for the SYNC message.

In these operating instructions, COB-IDs are stated as hexadecimal values.

14.5 Description of Variables Interpolated Position Mode

14.5.1 Process Data Mapping Interpolated Position Mode

Interpolation Data Record	Index: 60c1, Short name: IpolDataRecord
SPP Windows	–
Type	record, 1 element (subindex 0 .. 1), integer32
Record element 1	target position, interpolated position mode (index 60c1, subindex 1, length 32 bit)
Access	read/write
Value range	$-2^{24}..2^{24}-1$ PSS

Via this variable, the target position (in position sensor units) is transmitted.

Alternat. Interpolation Data Record	Index: 5fc1, Short name: AltIpolDataRecord
SPP Windows	–
Type	record, 2 elements (subindex 0 .. 2)
Record element 1	interpolation data record bit 0 .. 15, integer16
Record element 2	interpolation data record bit 16 .. 23, integer8
Access	read/write
Value range	$-2^{24}..2^{24}-1$ PSS

This variable can be used as alternative for the 32-bit *Interpolation Data Record* to map this record with 24 bit, only.

Together with *Alternative für Velocity Offset*, the values for *Interpolation Data Record*, *Velocity Offset*, and *Control Word can be mapped in an 8-byte PDO.*

Velocity Offset	Index: 60b1, Short name: VelocityOffset
SPP Windows	–
Type	simple variable, integer32
Access	read/write
Unit	0.25 r.p.m.
Standard value	...

If this variable is mapped in interpolated position mode, velocity feedforward is taken from this value, the master has do deliver valid values cyclically using *IpolDataRecord .*

In other operating modes, this variable does not have any effect.

If this value is not mapped in interpolated position mode, velocity feedforward is calculated internally.

Alternative for Velocity Offset	Index: 6fc2, Kurzname: AltVelocityOffset
SPP Windows	–
Type	record, 2 elements (subindex 0 .. 2)
Record element 1	velocity offset bit 0 .. 15, integer16
Record element 2	velocity offset bit 16 .. 23, integer8
Access	read/write
Unit	0.25 r.p.m.
Standard value	...

This variable can be used as an alternative for the 32-bit *Velocity Offset* to map this record with 24 bit, only.

Together with *Alternative Interpolation Data Record,* the values for *Interpolation Data Record*, *Velocity Offset*, and *Control Word* can be mapped in an 8-byte PDO.

14.5.2 Machine Data Interpolated Position Mode

The parameters described in the following can be found in SPP Windows in window “Parameterization” on “Axis data/Interpolated Position Mode”.

Interpolation Sub Mode Select	Index: 60c0, Short name: IpolSubmodeAuswahl
SPP Windows	Interpolation Submode 0
Type	simple variable, type integer16
Access	read only, writing possible but without effects
Var. type	constant (value does not change)
Default value	0 (linear interpolation)

This variable determines the interpolation mode. Only linear interpolation is supported.

Interpolation Time Period	Index: 60c2, Short name: IpolTimePeriod
SPP Windows	Interpolation Time Units 4 10^{idx} s Interpolation Time Index –3
Type	record, 2 elements (subindex 0 .. 2)
Record element 1	interpolation time units (2 .. 10), unsigned8
Record element 2	interpolation time index (–3), integer8
Access	record element 1: read and write record element 2: read only, writing possible but without effects
Var. type	CANopen machine data, interpolated position mode

The time interval at which the target positions are transmitted in interpolation position mode are determined via the *Interpolation Time Unit*. The *Interpolation Time Index* determines the unit for record element 1 (–3 for 10^{-3} s = 1 ms).

Interpolation Sync Definition	Index: 60c3, Short name: IpolSyncDefinition
SPP Windows	Synchronize on group 0 ip_sync every n events 1
Type	array, 2 elements (subindex 0 .. 2), unsigned8
Array element 1	synchronize on group
Array element 2	ip sync every n events
Access	read only, writing possible but without effects
Var. type	constant (value does not change)

Interpolation Data Configuration	Index: 60c4, Short name: IpolDataConfig
SPP Windows	–
Type	record, 6 elements (subindex 0 .. 6)
Record element 1	max. buffer size, unsigned32; constant 1
Record element 2	actual size, unsigned32; constant 1
Record element 3	buffer organization, unsigned8; constant 0 (FIFO buffer)
Record element 4	buffer position, unsigned16; constant 0
Record element 5	size of data record, unsigned8; constant 1
Record element 6	buffer clear, unsigned8; constant 1
Access	read only, writing possible but without effects
Var. type	constant (value does not change)

14.6 Description of Variables Monitoring

14.6.1 Fault Handling Status (Error, Emergency)

The variables described in the following are used for describing and transmitting faults.

Error Register	Index: 1001, Short name: ErrorReg
SPP Windows	–
Type	simple variable, type unsigned8
Access	read only
Var. type	status information
Default value	0 or a fault after switch-on

Bits 0 and 5 are used (generic error or profile specific error). In case of a fault, these bits are set and signal that an error code can be read from variable *Pre-Defined Error Field*, Subindex 1, describing the cause of the fault more detailed.

All other bits of variable *Error Register* are always 0.

Pre-Defined Error Field	Index: 1003 Short name: ErrorField
SPP Windows	–
Type	array, 1 element (subindex 0 .. 1), type unsigned32
Array element 1	default error field
Access	read only
Var. type	status information

In this variable, the fault code of the last fault is displayed and stored on subindex 1. In case of a change, the new value is transmitted within the emergency message. For further information on the fault codes see Appendix D (page 60).

14.6.2 Machine Data Fault Handling (Emergency)

COB-ID Emergency Message	Index: 1014, Short name: IdEmcy
SPP Windows	COB-ID Emergency Message 0000008 + Node-ID
Type	simple variable, type unsigned32
Access	read and write
Var. type	CANopen machine data
Default value	80 + Node-ID

With this variable, the COB-ID for the emergency message can be queried and set.

In these operating instructions, COB-IDs are always stated as hexadecimal values.

Inhibit Time Emergency Message	Index: 1015, Short name: InhibitTimeEmcy
SPP Windows	Inhibit Time 0,0 ms
Type	simple variable, type integer16
Access	read/write
Var. type	CANopen machine data
Default value	0

With this variable, the frequency of the transmission of the emergency message can be limited.

Abort Connection Option Code	Index: 6007, Short name: AbortOptionCode
SPP Windows	–
Type	simple variable, integer16
Access	read/write
Var. type	CANopen machine data
Default value	0 = no reaction, 1 = device control in case of node guarding/life guarding faults (default)

This variable can be used for defining reactions on a communication breakdown.

14.6.3 Machine Data Node Guarding, Life Guarding

Guard Time	Index: 100c, Short name: GuardTime
SPP Windows	Guard Time 0 ms
Type	simple variable, type unsigned16
Access	read/write
Var. type	CANopen machine data
Default value	0

With this variable, the monitoring period can be set in milliseconds.

Life Time Factor	Index: 100d, Short name: LifeTimeFac
SPP Windows	Life Time Factor 0
Type	simple variable, type unsigned8
Access	read/write
Var. type	CANopen machine data
Default value	0

This variable serves for indirect setting of the so-called life time, the period of time the master must have sent a message within.

The value of variable *Guard Time* multiplied by this *Life Time Factor* results in the life time.

For further information, please see section 11.2 (page 27).

Producer Heartbeat Time	Index: 1017, Short name:ProducerHeartbeatTime
SPP Windows	Producer heartbeat time 0 ms
Type	simple variable, type unsigned16
Access	read/write
Var. type	CANopen machine data CANopen
Default valuet	0 ms

This variable is used to define the intervals in which the servo drive sends heartbeat telegrams (0 = off).

For further information see section 11.3 (page 28).

14.7 Description of Variables Storage Functions

14.7.1 Storage Functions Control

The storage functions are controlled via the variables described in the following.

Store Parameters	Index: 1010, Short name: StoreParm
SPP Windows	–
Type	array, 4 elements (subindex 0 .. 4), type unsigned32
Array element 1	save all
Array element 2	save communication (not supported)
Array element 3	save application (not supported)
Array element 4	save manufacturer (not supported)
Access	read/write
Var. type	control information

Function “save all” (subindex 1) is supported. By writing a certain code into array element 2 (ASCII: e v a s), all machine data, the part program, and the user variables of the servo drive are stored in the non-volatile memory of the servo drive (servo EPROM). Part of the machine data are the CANopen machine data (var. type = MC) of these operating instructions.

After the CANopen machine data have been stored in the servo EPROM by corresponding entry in this variable, these are accepted in case of

- switch-on of the control voltage of the servo drive
- NMT-Service Reset_Communication (CANopen parameters, only)
- NMT-Service Reset_Node (all parameters)

The data of the predefined connection set (standard or default settings ex works) are not used any longer.

With a certain code described with variable *Restore Default Parameters*, the values of the predefined connection set can be re-activated.

Restore Default Parameters	Index: 1011, Short name: RestoreParm
SPP Windows	–
Type	array, 4 elements (subindex 0 .. 4), type unsigned32
Array element 1	restore all (not supported)
Array element 2	restore communication
Array element 3	restore application (not supported)
Array element 4	restore manufacturer (not supported)
Access	read/write
Var. type	control information

Function “restore communication default” (subindex 2) is supported. By writing a certain code into array element 2 (ASCII: d a o l), the standard settings (default values) of all CANopen variables can be restored.

- For the standard settings (default values) of the COB-IDs, please see table “Predefined Connection Set” in section 6.2 (page 19). They can also be taken from the description of the respective variable.
- For the standard settings (default values) of the other CANopen machine data, please see the description of the respective variable.

NMT service Reset_Node has to be carried out after the entry into the variable to make the standard settings effective.

For a permanent storage of the restored standard settings, they have to be stored by access to variable *Store Parameters*.

14.8 Description of Variables Device Information

14.8.1 General Information Device

Device Type	Index: 1000, Short name: DeviceType
SPP Windows	–
Type	simple variable, unsigned32
Access	read only
Var. type	constant (value does not change)
Default value	device profile: CiA 402, drive type: servo drive (0×00020192)

The device type can be read out via variable *Device Type*.

According to CANopen communication profile CiA 301 and CANopen device profile CiA 402, the hexadecimal value of this variable describes the device type and its function:

Bit	Meaning
1..5	device profile: 192 $_{hex}$ = 402 $_{dec}$ = CiA 402
16..23	drive type: 2 = servo drive
24..31	always 0

Manufacturer Device Name	Index: 1008 Short name: DeviceName
SPP Windows	–
Type	simple variable, visible string, length 16 bit
Access	read only
Var. type	constant (variable value does not change)
Default value	–

The name of the manufacturer can be read via variable *Manufacturer Device Name*.

Manufacturer Hardware Version	Index: 1009, Short name: HWVersion
SPP Windows	–
Type	simple variable, visible string, length 16 bit
Access	read only
Var. type	constant (variable value does not change)
Default value	–

The hardware version of the servo drive can be read via variable *Manufacturer Hardware Version*.

Manufacturer Software Version	Index: 100a, Short name: SWVersion
SPP Windows	–
Type	simple variable, visible string, length 16 bit
Access	read only
Var. type	constant (variable value does not change)
Default value	–

The software version of the servo drive can be read via variable *Manufacturer Software Version*.

Identity	Index: 1018, Short name: Identity
SPP Windows	–
Type	array, 4 elements (subindex 0 .. 4), type unsigned32
Array element 1	Vendor ID (15)
Array element 2	Product code (6755)
Array element 3	Revision number
Array element 4	Serial number
Access	read only
Var. type	constant (variable value does not change)

The manufacturer of the servo drive can be read via this variable.

Appendix

Appendix A State Machines

State machines describe the behavior of systems. The graphical representation of a state machine is called state diagram.

The elements of a state machine are

- states
- transitions
- events
- actions

A representation of the elements of a state machine in the state diagram is shown in the following figure:

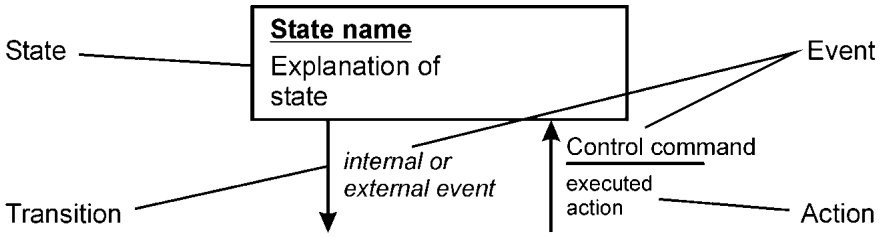

Figure 2: Elements of State Machines

States (shown as a rectangle with the name of the state and further explanations, if required) can be changed by **transitions** (displayed as arrows). A transition is performed when an **event** occurs. In the case of a transition, an **action** is carried out (represented below the event, separated from it by a line). Transmissions for which an action is not carried out are also permitted.

In the application of state machines, the following types of events are differentiated:

- control commands
- internal or external events

Control commands are events the user can trigger e. g. by inducing the NMT master to carry out a NMT service.

Internal or external events (represented in italics) are triggered by the servo drive or a component connected to one of its interfaces. This can be e. g. the switch-on of the control voltage or the termination of the initialization.

Appendix B List of Variables

In this appendix, you can find an overview of all CANopen variables, sorted according to the index.

For explanations regarding the individual columns, please see section 14 (page 30).

Index	Name	Short name	Obj. code elements	Data type length	R/W	Var. type	Unit
1000	Device Type	DeviceType	Var	u32	R	F	-
1001	Error Register	ErrorReg	Var	u8	R	S	-
1003	Pre-Defined Error Field	ErrorField	Array 1	u32	R	S	-
1005	COB-ID SYNC Message	IdSync	Var	u32	R/W	MC	-
1008	Manufacturer Device Name	Device Name	Var	VisStr 16	R	F	-
1009	Manufacturer Hardware Version	HWVersion	Var	VisStr 16	R	F	-
100a	Manufacturer Software Version	SWVersion	Var	VisStr 16	R	F	-
100b	NodeID	NodeID	Var	32 bit	R	V	-
100c	Guard Time	GuardTime	Var	u16	R/W	MC	ms
100d	Life Time Factor	LifeTimeFac	Var	u8	R/W	MC	-
1010	Store Parameters	StoreParm	Array 4	u32	R/W	S	-
1011	Restore Default Parameters	RestoreParm	Array 4	u32	R/W	S	-
1014	COB-ID Emergency Message	IDEmcy	Var	u32	R/W	MC	-
1015	Inhibit Time Emergency Message	InhibitTimeEmcy	Var	i16	R/W	MC	ms
1017	Producer Heartbeat Time	ProducerHeartbeatTime	Var	u16	R/W	MC	ms
1018	Identity	Identity	Array 4	u32	R	F	-
1200	Server SDI 2 Parameter	SSDO1	Record 2	SDO Par.	R	V	-
1201	Server SDO 2 Parameter	SSDO1	Record 3	SDO Par.	R/W	V	-
1400	Receive PDO 1 Comm. Parameter	RPDO1ComParm	Record 2	PDO CommPar	R/W	MC	-
1401	Receive PDO 2 Comm. Parameter	RPDO2ComParm	Record 2	PDO CommPar	R/W	MC	-
1402	Receive PDO 3 Comm. Parameter	RPDO3ComParm	Record 2	PDO CommPar	R/W	MC	-
1403	Receive PDO 4 Comm. Parameter	RPDO4ComParm	Record 2	PDO CommPar	R/W	MC	-
1600	Receive PDO 1 Mapping	RPDO1Mapping	Record 8	PDOMap	R/W	MC	-
1601	Receive PDO 2 Mapping	RPDO2Mapping	Record 8	PDOMap	R/W	MC	-
1602	Receive PDO 3 Mapping	RPDO3Mapping	Record 8	PDOMap	R/W	MC	-
1603	Receive PDO 4 Mapping	RPDO4Mapping	Record 8	PDOMap	R/W	MC	-
1800	Transmit PDO 1 Comm. Parameter	TPDO1ComParm	Record 5	PDO CommPar	R/W	MC	-
1801	Transmit PDO 2 Comm. Parameter	TPDO2ComParm	Record 5	PDO CommPar	R/W	MC	-
1802	Transmit PDO 3 Comm. Parameter	TPDO3ComParm	Record 5	PDO CommPar	R/W	MC	-
1803	Transmit PDO 4 Comm. Parameter	TPDO4ComParm	Record 5	PDO CommPar	R/W	MC	-
1a00	Transmit PDO 1 Mapping	TPDO1Mapping	Record 8	PDOMap	R/W	MC	-
1a01	Transmit PDO 2 Mapping	TPDO2Mapping	Record 8	PDOMap	R/W	MC	-
1a02	Transmit PDO 3 Mapping	TPDO3Mapping	Record 8	PDOMap	R/W	MC	-
1a03	Transmit PDO 4 Mapping	TPDO4Mapping	Record 8	PDOMap	R/W	MC	-
1f80	NMT Startup	NMTStartup	Var	u32	R/W	V	-
5fb4	Node-ID	CANNodeId	Array 2	u8	R	V	-
5fb5	CAN Baudrate	CANBaudrate	Var	u8	R/W	V	Kbps
5fc1	Alternat. Interpol. Data Record	AltIpolDataRecord	Record 2	i16/i8	R/W	MC	PSS
5fc2	Alternative for Velocity Offset	AltVelocityOffset	Record 2	i16/i8	R/W	MC	-
6007	Abort Connection Option Code	AbortOptionCode	Var	i16	R/W	ME	-

Index	Name	Short name	Obj. code elements	Data type length	R/W	Var. type	Unit
60b1	Velocity Offset	VelocityOffset	Var	i32	R/W	ME	r.p.m.
60c0	Interpolation Sub Mode Select	IpolSubmodeAuswahl	Var	i16	R/W	MC	-
60c1	Interpolation Data Record	IpolDataRecord	Record 1	i32	R/W	MC	PSS
60c2	Interpolation Time Period	IpolTimePeriod	Record 2	IpolTime	R/W	MC	-
60c3	Interpolation Sync Definition	IpolSyncDefinition	Array 2	u8	R/W	MC	-
60c4	Interpolation Data Configuration	IpolDataConfig	Record 6	IpolConf	R/W	MC	-

Appendix C Configuration Aid

This configuration aid serves for setting the CANopen interface. Copy this configuration aid once for each servo drive. Do not start setting the device until you have filled in the sheets. First of all, do the hardware settings, afterwards, the machine data of the CANopen interface can be set.

Make sure to store the settings of the machine data of the CANopen interface on a floppy/ hard disk using the CANopen configuration software, see section 12 (page 29) and/or the operating instructions of the CANopen configuration software used.

Application Data

Designation of the application:_____

(e. g. paletting machine)

Designation of the axis: _____

(e. g. X axis)

Created, changed (name, date):_____

Hardware Settings

Node-ID hexadecimal: _____

The Node-ID is also used for determining the COB-ID. In order to avoid problems within the CAN network, the following COB-IDs must not be changed:

COB-ID NMT (= 0), COB-ID SDO Tx (= 580 + Node-ID), COB-ID SDO Rx (= 600 + Node-ID).

Transmission rate: _____

Permissible values [kBit/s]: 10, 20, 50, 125, 250, 500, 800, 1000. Also observe the max. permissible bus lengths, see section 5.1.2 (page 16).

Machine data PDOs (process data channel): Receive PDOs

Index	Name	Default Value	Value to be set (if not default)
1600	Receive PDO 1 Mapping	No. of entries = 1	
		60400010	
		0	
		0	
		0	
		0	
		0	
		0	
		0	
1601	Receive PDO 2 Mapping	No. of entries = 2	
		60400010	
		60600010	
		0	
		0	
		0	
		0	
		0	
		0	
1602	Receive PDO 3 Mapping	No. of entries = 2	
		60400010	
		607A0020	
		0	
		0	
		0	
		0	
		0	
		0	
1603	Receive PDO 4 Mapping	No. of entries = 2	
		60400010	
		60810020	
		0	
		0	
		0	
		0	
		0	
		0	

Machine data PDOs (process data channel): Transmit PDOs

Index	Name	Default Value	Value to be set (if not default)
1a00	Transmit PDO 1 Mapping	No. of entries = 1	
		60410010	
		0	
		0	
		0	
		0	
		0	
		0	
		0	
1a01	Transmit PDO 2 Mapping	No. of entries = 2	
		60410010	
		60610010	
		0	
		0	
		0	
		0	
		0	
		0	
1a02	Transmit PDO 3 Mapping	No. of entries = 2	
		60410010	
		60640020	
		0	
		0	
		0	
		0	
		0	
		0	
1a03	Transmit PDO 4 Mapping	No. of entries = 2	
		60410010	
		606C020	
		0	
		0	
		0	
		0	
		0	
		0	

Machine data PDOs (process data channel): Comm. Parameter

Index	Name	Default Value	Value to be set (if not default)
1400	Receive PDO 1 Comm. Parameter	valid (0)	
		200 + Node-ID = ____ (hexadecimal)	
		asynchronous (255)	
1401	Receive PDO 2 Comm. Parameter	not valid (1)	
		300 + Node-ID = ____ (hexadecimal)	
		asynchronous (255)	
1402	Receive PDO 3 Comm. Parameter	not valid (1)	
		400 + Node-ID = ____ (hexadecimal)	
		asynchronous (255)	
1403	Receive PDO 4 Comm. Parameter	not valid (1)	
		500 + Node-ID = ____ (hexadecimal)	
		asynchronous (255)	
1800	Transmit PDO 1 Comm. Parameter	valid (0)	
		180 + Node-ID = ____ (hexadecimal)	
		asynchronous (255)	
		Inhibit time = 0 ms	
		Event timer = 0 ms	
1801	Transmit PDO 2 Comm. Parameter	not valid (1)	
		280 + Node-ID = ____ (hexadecimal)	
		asynchronous (255)	
		Inhibit time = 0 ms	
		Event timer = 0 ms	
1802	Transmit PDO 3 Comm. Parameter	not valid (1)	
		380 + Node-ID = ____ (hexadecimal)	
		asynchronous (255)	
		Inhibit time = 0 ms	
		Event timer = 0 ms	
1803	Transmit PDO 4 Comm. Parameter	not valid (1)	
		480 + Node-ID = ____ (hexadecimal)	
		asynchronous (255)	
		Inhibit time = 0 ms	
		Event timer = 0 ms	

Machine Data Synchronization

Index	Name	Default value	Value to be set (if not default)
1005	COB-ID Sync Message	80 (hexadecimal)	

Machine Data Node Guarding, Life Guarding

Index	Name [units]	Default value	Value to be set (if not default)
100c	Guard Time [ms]	switched off (0)	
100d	Life Time Factor [multiple of the Guard Time]	switched off (0)	
1017	Producer Heartbeat Time	0 ms	

Machine Data Interpolated Position Mode

Index	Name	Default value	Value to be set (if not default)
60c2	Interpolation Time Period	Time Units: 4 10^{-3} s	

Appendix D Fault Codes

In case of a fault, the device status (displayed in the axis status word) changes to “Fault”, the corresponding axis fault code is displayed. When the cause of the fault has been removed, the fault status can be reset by a transition of bit “reset fault” from 0 to 1 in the axis control word.

There are two types of faults:

- Faults of the drive which are documented in Operating Instructions 6710.201 “Functions and Parameters” and displayed in drive variable *Axis Fault Code*.

  See operating instructions 6710.201 “Functions and Parameters”, section “Axis Fault Code”.

- Faults of the CANopen interface or the CAN controller

In the following, the possible CAopen fault codes (hexadecimal) are listed with fault designation and a short explanation.

8110 Message loss in the CAN controller

This fault is processed in the CAN system, only, it is not reported to the software of the servo drive.

8120 Fault CAN-Bus, warning limit reached

This fault is processed in the CAN system only, it is not reported to the servo drive software.

8130 Life Guarding / Heartbeat Error

This fault is also reported to the servo drive firmware as a fault code and causes a switch-off of the drive system.

8140 Bus OFF state left

This fault is processed in the CAN system only, it is not reported to the servo drive software.

8150 COB-ID collision

This fault is also reported to the servo drive firmware as a fault code and causes a switch-off of the drive system.

8180 Fault; too many sending and/or receiving faults (CAN term: bus off), device has decoupled from the CAN bus, drive has been stopped by quick-stop command, for further details see section 11.4 (page 28).

8210 Received PDO too short

This fault is processed in the CAN system only, it is not reported to the servo drive software.

8220 Received PDO too long

This fault is processed in the CAN system only, it is not reported to the servo drive software.

Appendix E Keyword Index

In the keyword index, groups were set up to provide a better overview of certain keywords in addition to the direct entry in alphabetical order.

Variables	The variable description of the corresponding variable can be found via the designation of a variable.
Variables short names	Via the short name of a variable, you can find the variable description of the corresponding variable.

Further groups were set up for an illustration of the connections.