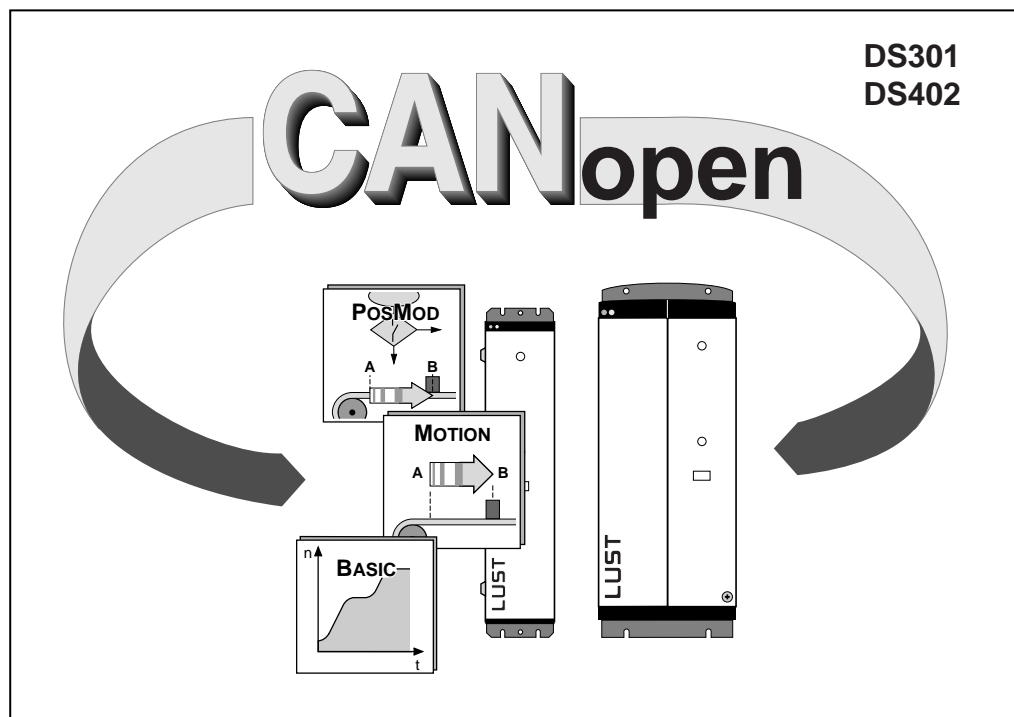


# Interconnection of servocontrollers

via CANopen



Data transfer protocol

## **CANopen-data transfer protocol**

applicable to Servocontrollers of series  
MASTERDRIVE 7000, BASIC, C15  
MASTERDRIVE 7000, MOTION, C15  
MASTERDRIVE 7000, PosMOD, C15

Software version: V4.0 or higher

Date: June 1999

ID no.: 0808.44B.0-00

We reserve the right to make technical changes.

## Table of contents

<b>1</b>	<b>Introduction: CANopen.....</b>	<b>7</b>
<b>2</b>	<b>Overview: Profile support .....</b>	<b>8</b>
2.1	Overview: Object directory DS301 / DS 402.....	8
2.2	Parameter-setting channel (ServiceDataObjects) .....	9
2.3	Process channel (ProcessDataObjects).....	10
2.3.1	Profile-oriented RXPDOs .....	10
2.3.2	Profile-oriented TXPDOs .....	11
2.3.3	Manufacturer-specific RXPDO .....	11
2.3.4	PDO transmission types.....	12
2.4	DS301 boot-up.....	12
2.5	Sync object.....	14
2.6	Emergency object.....	15
2.7	Node guarding.....	15
<b>3</b>	<b>Installation.....</b>	<b>16</b>
3.1	Electrical connection.....	16
3.2	Bus interface and address assignment by connector coding.....	17
3.3	LED status display.....	18
<b>4</b>	<b>Commissioning and configuration .....</b>	<b>19</b>
4.1	Commissioning sequence.....	19
4.2	Key parameter settings .....	19
4.2.1	Assignment of device address (NodeID).....	19
4.2.2	Baud rate setting.....	20
4.3	Connection test.....	20
4.3.1	Bus message after system start.....	20
4.3.2	SDO transfer: Reading manufacturer hardware version.....	21
4.3.3	SDO transfer: Writing and reading device parameters .....	21
4.3.3.1	Where can I find the device parameters? .....	21

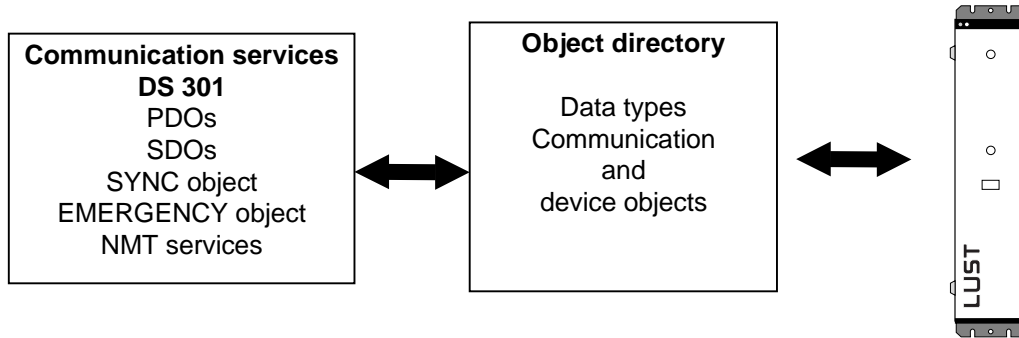
4.3.3.2	Description of data types.....	22
<b>5</b>	<b>Control and reference input.....</b>	<b>25</b>
<b>5.1</b>	<b>Setup of control location and reference transfer .....</b>	<b>25</b>
5.1.1	MC7000 parameters for bus operation.....	25
5.1.1.1	402-CLSEL - Control location .....	25
5.1.1.2	419-RSSL3 - Reference selector .....	25
5.1.1.3	491-CACTR - Control word.....	26
5.1.1.4	490-CASTA - Status word.....	26
5.1.1.5	497-RCAN - Reference from CAN bus .....	26
5.1.1.6	400-ACTV - Actual value .....	26
<b>5.2</b>	<b>DriveCom state machine to DS402.....</b>	<b>27</b>
5.2.1	Cross-mode information .....	27
5.2.1.1	Control and reference input via PDO channel .....	28
5.2.1.2	Status and actual value calculation via PDO channel.....	28
<b>5.3</b>	<b>Control of MC7000 in BASIC modes .....</b>	<b>28</b>
5.3.1	Structure of the control word in BASIC modes .....	28
5.3.2	Structure of the status word in BASIC mode .....	29
<b>5.4</b>	<b>Control of MC7000 in Electronic Gearing mode.....</b>	<b>30</b>
5.4.1	Structure of the control word in Electronic Gearing mode .....	30
5.4.2	Structure of the status word in Electronic Gearing mode .....	30
5.4.3	Example: Activation in Electronic Gearing mode.....	31
<b>5.5</b>	<b>Control of MC7000 in PosMOD mode.....</b>	<b>32</b>
5.5.1	Structure of the control word in POSMOD mode .....	32
5.5.2	Structure of the status word in POSMOD mode.....	32
5.5.3	Example: POSMOD activation.....	33
<b>6</b>	<b>Profile support in detail.....</b>	<b>35</b>
<b>6.1</b>	<b>Telegram structure .....</b>	<b>35</b>
<b>6.2</b>	<b>Default setting of communication objects.....</b>	<b>36</b>
<b>6.3</b>	<b>Default setting of DS301/402 objects .....</b>	<b>37</b>
<b>6.4</b>	<b>Response of the servo in a network.....</b>	<b>38</b>
<b>6.5</b>	<b>Setting up application-specific PDOs .....</b>	<b>38</b>
6.5.1	Changing the PDO communication parameters.....	38
6.5.2	Changing the PDO mapping .....	40

6.5.2.1	Setting up a user-specific RXPDO.....	41
6.5.2.2	Setting up a user-specific TXPDO.....	43
<b>6.6</b>	<b>Monitoring by Node/Life Guarding .....</b>	<b>45</b>
<b>6.7</b>	<b>EDS device file .....</b>	<b>46</b>
6.7.1	How do I create the EDS device file ?.....	46
<b>6.8</b>	<b>Saving the CANopen settings .....</b>	<b>46</b>
<b>6.9</b>	<b>Restoring factory defaults.....</b>	<b>47</b>
<b>6.10</b>	<b>Time response.....</b>	<b>48</b>
<b>7</b>	<b>Fault rectification .....</b>	<b>49</b>
<b>7.1</b>	<b>Troubleshooting.....</b>	<b>49</b>
<b>7.2</b>	<b>Resetting an error .....</b>	<b>49</b>
7.2.1	Error acknowledgment via bus system .....	49
7.2.2	Error acknowledgment, general .....	50
<b>7.3</b>	<b>Table: Emergency error codes.....</b>	<b>50</b>
<b>8</b>	<b>Appendix .....</b>	<b>52</b>
<b>8.1</b>	<b>Downloading a positioning program from the positioning and sequence control.....</b>	<b>52</b>

# 1 Introduction: CANopen

The CANopen profile was defined by the CIA manufacturers' confederation with the aim of combining the products of different manufacturers in one standardized automation network. The CIA DS301 profile provides a collection of CAN communication services without recording the application precisely. Based on these communication services, the **application-specific drive profile DS402** was created.

In addition to the functions defined in the profiles there are more detailed manufacturer-specific add-ons.



*Diagram: The communication services of the profile provide access to the object directory of the MC7000*

In the MC7000 servocontrollers the DS301 profile is implemented. DS402 supports the obligatory elements such as the control word, status word and operation modes. The MC7000 parameters are a manufacturer-specific add-on.

The following sections will provide you with an overview of the CANopen functionality integrated into the MC7000. There then follows the information necessary for commissioning.

In the following sections we assume that you have a commonly available CANopen setup program and a CANopen interface driver. For the precise protocol definitions refer to the CAL specification, or contact Lust Antriebstechnik GmbH.

## 2 Overview: Profile support

### 2.1 Overview: Object directory DS301 / DS 402

As in other field bus protocols, the central instance of all CANopen nodes is the so-called object directory. Each CANopen device must know the so-called object directory. In addition to the standard entries, it must also be possible to set up the objects relevant to the device in this directory.

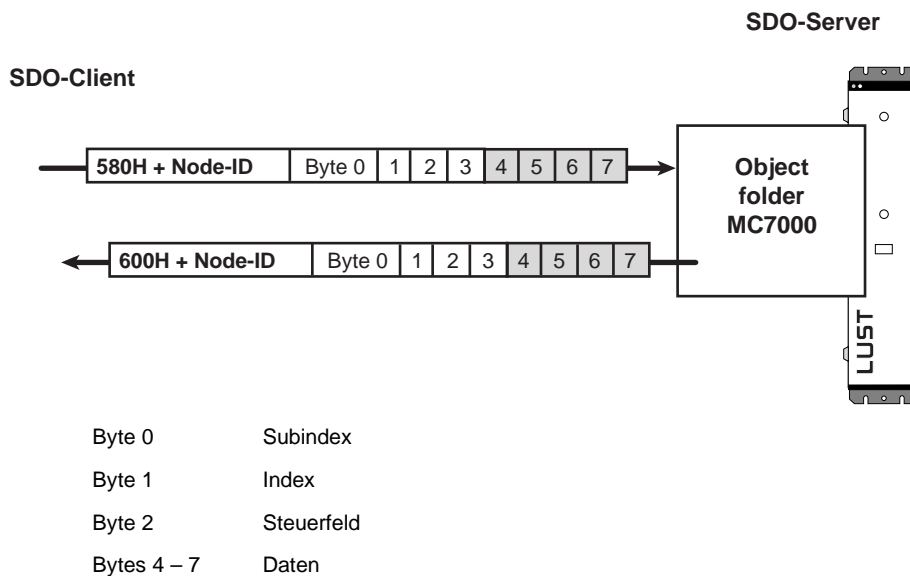
The following table gives an overview of the profile support of the MC7000.

Profile	Index	Object	Name	Type	Acc.	M/O
Draft 301	<b>1000H</b>	VAR	<b>device type</b>	Unsigned32	ro	M
Draft 301	<b>1001H</b>	VAR	<b>error register</b>	Unsigned8	ro	M
Draft 301	<b>1003H</b>	ARRAY	<b>predefined error field</b>	Unsigned32	ro	O
Draft 301	<b>1004H</b>	ARRAY	<b>number of PDOs supported</b>	Unsigned32	ro	O
Draft 301	<b>1005H</b>	VAR	<b>COB-ID SYNC message</b>	Unsigned32	rw	O
Draft 301	<b>1006H</b>	VAR	<b>Communication cycle period</b>	Unsigned32	rw	O
Draft 301	<b>1007H</b>	VAR	<b>synchronous window length</b>	Unsigned32	rw	O
Draft 301	<b>1008H</b>	VAR	<b>manufacturer device name</b>	Vis-String	ro	O
Draft 301	<b>1009H</b>	VAR	<b>manufacturer hardware version</b>	Vis-String	ro	O
Draft 301	<b>100AH</b>	VAR	<b>manufacturer software version</b>	Vis-String	ro	O
Draft 301	<b>100BH</b>	VAR	<b>Node-Id</b>	Unsigned32	ro	O
Draft 301	<b>100CH</b>	VAR	<b>guard time</b>	Unsigned32	rw	O
Draft 301	<b>100DH</b>	VAR	<b>life time factor</b>	Unsigned32	rw	O
Draft 301	<b>100EH</b>	VAR	<b>node guarding identifier</b>	Unsigned32	rw	O
Draft 301	<b>100FH</b>	VAR	<b>number of SDOs supported</b>	Unsigned32	ro	O
Draft 301	<b>1010H</b>	ARRAY	<b>Store parameters</b>	Unsigned32	rw	O
Draft 301	<b>1011H</b>	VAR	<b>restore default parameters</b>	Unsigned32	rw	O
Draft 301	<b>1014H</b>	VAR	<b>COB-ID emergency message</b>	Unsigned32	rw	O
Draft 301	<b>1200H</b>	RECORD	<b>Server SDO Parameter</b>	SDO-Parameter	rw	O
Draft 301	<b>1400H</b>	RECORD	<b>Receive PDO Comm.-Parameter</b>	PDCommPar	rw	O
Draft 301	<b>1600H</b>	ARRAY	<b>Receive PDO Mapping-Parameter</b>	PDMapping	rw	O
Draft 301	<b>1800H</b>	RECORD	<b>Transmit PDO Comm-Parameter</b>	PDCommPar	rw	O
Draft 301	<b>1A00H</b>	ARRAY	<b>Transmit PDO Mapping-Parameter</b>	PDMapping	rw	O
manufacturer specific	<b>2000H - 23E8H</b>		<b>Servo Parameter</b>			
Draft 402	<b>6040H</b>	VAR	<b>controlword</b>	Integer16	rw	M
Draft 402	<b>6041H</b>	VAR	<b>statusword</b>	Unsigned16	ro	M
Draft 402	<b>6060H</b>	VAR	<b>modes_of_operation</b>	Integer8	wo	M
Draft 402	<b>6061H</b>	VAR	<b>modes_of_operation_display</b>	Integer8	ro	M

With the aid of these objects it is possible to configure the actual CANopen communication very flexibly and adapt it to the specific needs of the user. For a detailed description of the individual functions refer to the appendix.

## 2.2 Parameter-setting channel (ServiceDataObjects)

The Service Data Object (SDO) permits read and write access to the object directory. This SDO is implemented to CAN specification by the Multiplexed Domain CMS object. The protocol is designed for the transfer of data of any length. A so-called Client-SDO is integrated into the device for the SDO transfer. The communication is by way of two reserved identifiers.



In the CAN specification a basic distinction is made between three protocol services:

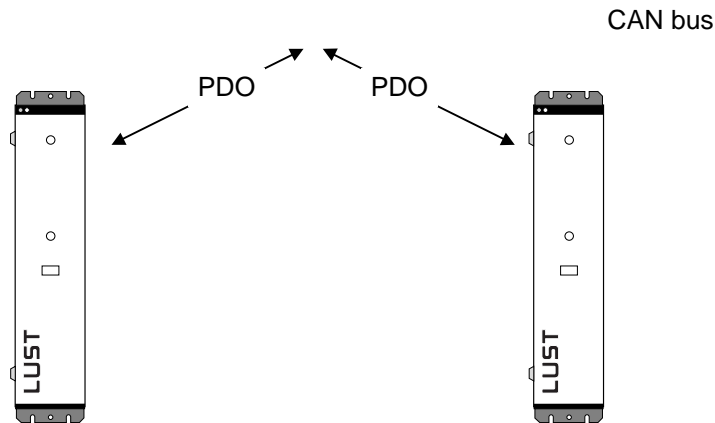
- Download protocol (write)
- Upload protocol (read)
- Abort protocol (error)

The upload and download protocols are additionally differentiated by Expedited (single) and Non-Expedited (segmented) modes.



## 2.3 Process channel (ProcessDataObjects)

The real-time data exchange between CANopen nodes is effected by high-priority PDO transfer. This is a purely CAN communication with no protocol overhead, in which the broadcast properties are retained in full. In contrast to SDO transfer, data may be exchanged between two slaves without any request from the master.



The PDOs may optionally be transferred event-controlled or synchronous. These communication properties can be set in the relevant PDO communication parameters. In the following the static - non-changeable - and variable freely definable PDOs of the servo are listed.

### 2.3.1 Profile-oriented RXPDOs

With the aid of the RXPDO1 it is possible to control the DriveCom state machine of the servo and to make reference inputs. For more information refer to the section headed 5.2.1.1 "Control and reference input via PDO channel".

RX PDO1	Type: static		COB-ID: 0x200 +		NodeID		
Byte:0	1	2	3	4	5	6	7
LB control word	HB control word	LW LB reference	LW HB reference	HW LB reference	HW HB reference	X	X

You can use the variable RXPDO2 for user-specific communication. You can compile the configuration - or rather the mapping - for your specific application, and so control selected variables in the MC7000. For more information refer to the section 6.5.2.1 "Setting up a user-specific RXPDO".

RX PDO2	Type: variable		COB-ID: 0x300 +		Node-ID		
Byte:0	1	2	3	4	5	6	7
variable	variable	variable	variable	variable	variable	variable	variable

### 2.3.2 Profile-oriented TXPDOs

You can get status information on the device by way of the TXPDO1. The first two bytes of the TXPDO indicate the status of the DriveCom state machine; the next four bytes contain the current actual value of the servo. For more information refer to the section headed 5.2.1.2 "Status and actual value calculation via PDO channel".

TXPDO1	Type	static	COB-ID:	0x180 +	Node-ID		
Byte:0	1	2	3	4	5	6	7
LB status word	HB status word	LW LB actual value	LW HB actual value	HW LB actual value	HW HB actual value	X	X

Information from the device which you need for your application can be mapped into the TXPDO2 in the form of parameter objects. For more information refer to the section headed 6.5.2.2 "Setting up a user-specific TXPDO".

TXPDO2	Type	variable	COB-ID	0x280 +	Node-ID		
Byte:0	1	2	3	4	5	6	7
variable	variable	variable	variable	variable	variable	variable	variable

### 2.3.3 Manufacturer-specific RXPDO

This manufacturer-specific RXPDO21 was set up specially for communication with the positioning and sequence control mode (PosMOD). This makes it possible to describe the key parameters for the PosMOD functionality 528-POVAR (variables) and 529-POMER (flags) over the fast PDO channel.

RXPDO21	Type	static	COB-ID:	0x400 +	Node-ID		
Byte:0	1	2	3	4	5	6	7
Mode	Field no.	LW LB	LW HB	HW LB	HW LB	Field no.	Data

Explanation:

This PDO is only used to describe the PosMOD parameters 528-POVAR and 529-POMER.

Mode :            F0H            Write POVAR  
                   0FH            Write POMER  
                   FFH            Write POVAR + POMER

Field no.:            The field number must be included in the transfer for the so-called field parameters.  
                           Value range: 0-63H

### 2.3.4 PDO transmission types

In connection with the PDO transfer, various transmission types are defined in CANopen profile DS301. The servo supports the following transmission types:

- **acyclic synchronous**      **Type No. 0**

Meaning: The "acyclic synchronous" transmission type represents the transfer of a PDO in conjunction with a Sync object; that is, RXPDOs are only evaluated on receipt of a SYNC object in the device and TXPDOs are only sent on receipt.

- **cyclic synchronous**      **Type No. 1-240**

Meaning: The difference between this transmission type and the "acyclic synchronous" type is that RXPDOs are only evaluated on receipt of 1-240 SYNC objects and TXPDOs are only sent every 1-240 SYNC objects.

- **asynchronous**      **Type No. 254**

Meaning: RXPDOs are evaluated immediately on receipt, TXPDOs are transmitted by a device-specific event. The SYNC object is irrelevant to this mode of transfer.

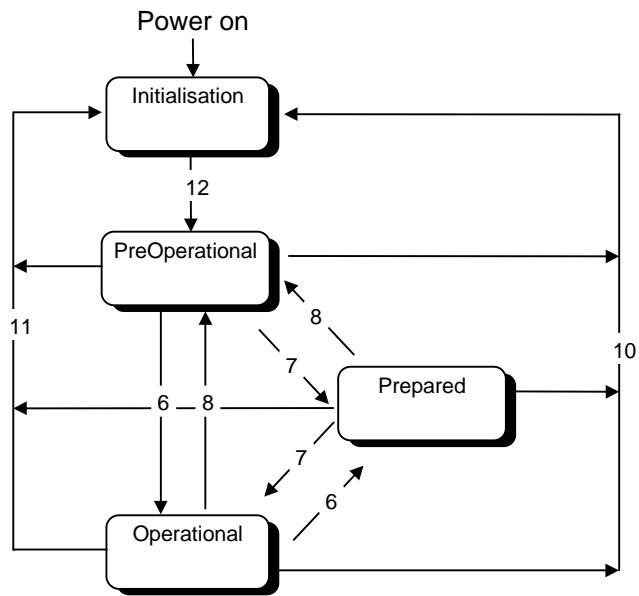
For further information on adapting the transmission type refer to the section headed 6.5.1 "Changing the PDO communication parameters".

## 2.4 DS301 boot-up

The servo supports Minimum Capability device response in a network. After boot-up the servo reaches the **PreOperational** parameter-setting state. In this state the master can communicate with the MC7000 by SDOs only. With appropriate parameter-setting of the MC7000, the network status has a direct influence on the DriveCom state machine described subsequently; that is to say, the device cannot be enabled (i.e. the drive cannot be started) in the **PreOperational** state.

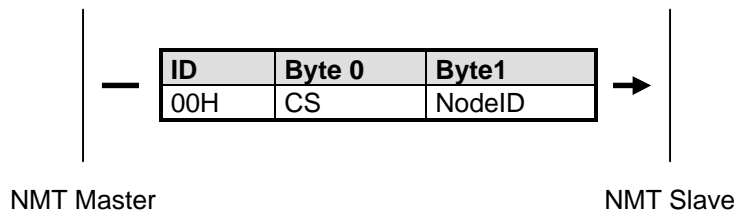
**Note:**                      Fundamentally, the PreOperational state is a **parameter-setting state** and the Operational state is an **operating state**.

- (6) Start\_Remote\_Node indication
- (7) Stop\_Remote\_Node indication
- (8) Enter\_Pre-Operational\_State indication
- (9) Reset\_Node\_indication
- (10) Reset\_Communication indication
- (11) Initialisation finished-enter Pre-Operational automatically



In the Prepared state a CANopen slave can no longer participate in the communication via SDO or PDO. This state offers CANopen slaves the possibility to opt out of communication in the event of serious errors, without disturbing the network. The response of the MC7000 in a network is detailed in the section headed 6.4 "Response of the servo in a network" .

**Telegram structure of the NMT services:**

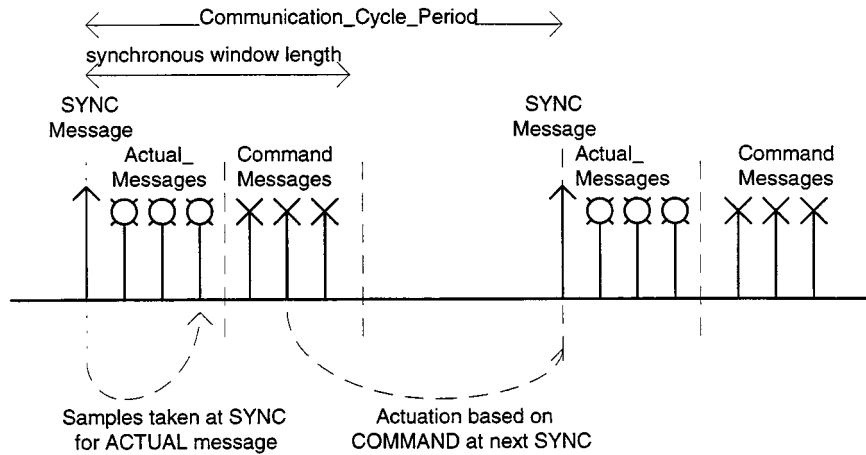


NMT-Service	Command Specifier
Start_Remote_Node indication	01H
Stop_Remote_Node indication	02H
Enter_Pre-Operational_State indication	80H
Reset_Node_indication	81H
Reset_Communication indication	82H

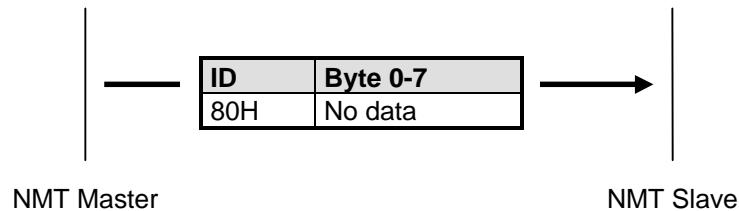
NodeID = 00H all slaves

## 2.5 Sync object

The primary CANopen communication object is the SYNC object. The SYNC object allows the NMT master to synchronize the slaves in the network. With the PDO communication parameters appropriately set, the Sync object can ensure, for example, that the reference value for the MC7000 is adopted quasi synchronously and the actual value is transmitted (see Transmission types of the PDOs).



### Telegram structure of the Sync object:



dlc data length code = 0

The object entries

Draft 301	<b>1006H</b>	VAR	<b>Communication cycle period</b>	Unsigned32	rw	○
Draft 301	<b>1007H</b>	VAR	<b>synchronous window length</b>	Unsigned32	rw	○

are only informative in character. Timeouts do not result in emergency messages.

## 2.6 Emergency object

When an error occurs the MC7000 transmits its errors in the form of an emergency message (profile DS301) with an emergency error code defined in the DS402. In addition, the error code of the MC7000 is entered in the manufacturer-specific data field.

Byte:	0	1	2	3	4	5	6	7
Bit:	0 ... 15	16 ... 23	24 ... 31	32 ... 39	40 ... 47	48 ... 55	56 ... 63	64 ... 71
Profile	Device Profile DS402			MC 7000				
Content:	Emergency error code as per DS402		Error register (Object 1001H)	Error code		Error location	Operating hours on occurrence of the error	

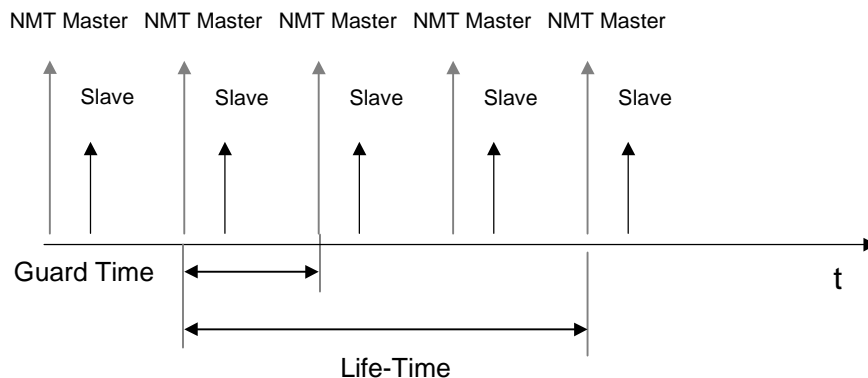
**ID = 80H+ NodeID**

dlc data length code = 8

For further information on error handling, resetting of errors and the meanings of the error codes refer to section 7.

## 2.7 Node guarding

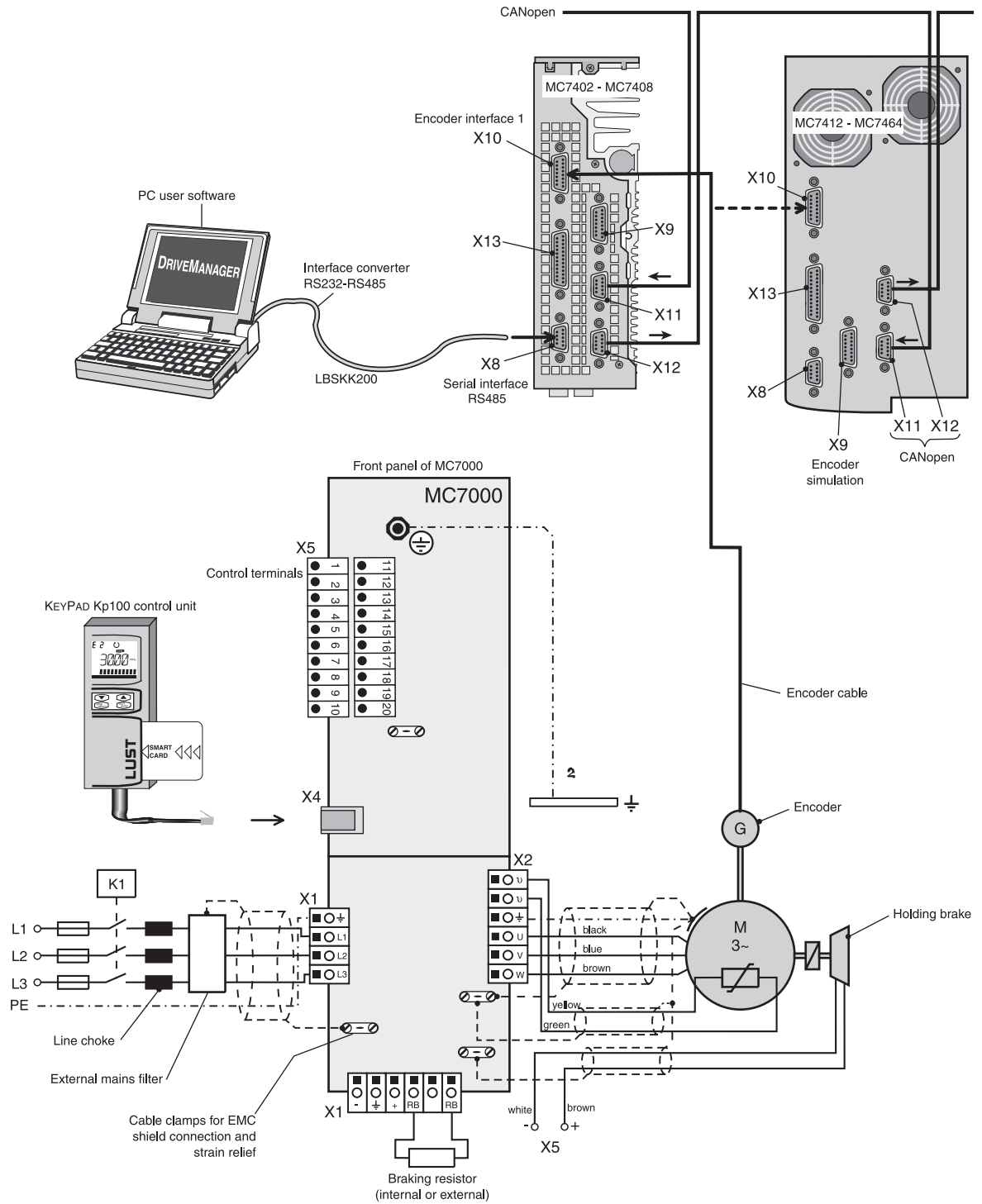
To monitor the slaves the DS301 defines Node or Life guarding. The master polls the devices connected to the bus at a defined sampling time (guard time) using a remote frame. In response to this request the slave sends a telegram containing the internal NMT state (Operational, PreOperational). In this way the master can check whether the slave state matches the master state.



In contrast, the slave can only monitor the master by means of so-called Life Guarding. If the master does not send a remote frame within the preset life time, the slave (MC7000) must assume there is a fault in the network and triggers a Guarding error. For further information on setting up Node Guarding and on the specific response of the servo in case of error, refer to section 6.6.

### 3 Installation

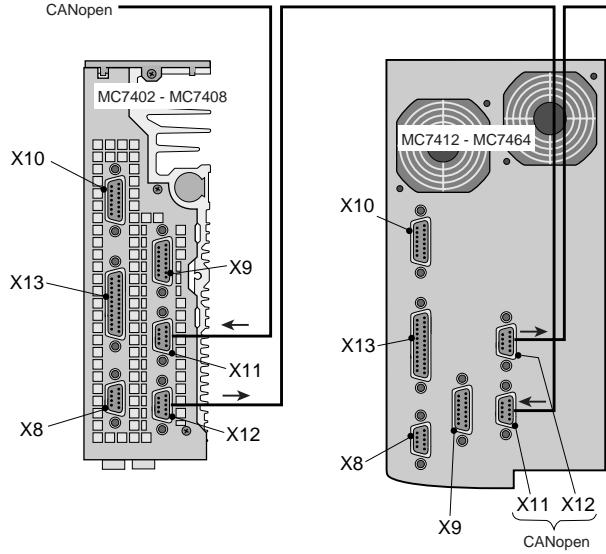
#### 3.1 Electrical connection



### 3.2 Bus interface and address assignment by connector coding

The device is connected to the CAN bus via the D-SUB connector/socket pairing labeled X11 and X12. Both connectors are switched in parallel. The device address in the connector can be coded by jumpers with +5V by way of the pins labeled ADRx.

The address range is 1 to 7, address 0 is not permitted. Alternatively, the addresses can be assigned by way of a parameter - see section 4.2.1.

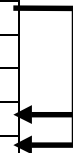


#### X12 = X11

PIN	Assignment
1	+5V
2	CAN_L
3	CAN_GND
4	ADR0
5	ADR1
6	(GND)
7	CAN_H
8	ADR2
9	CAN_V+

#### Example: Device address 3

PIN	Assignment
1	+5V
2	CAN_L
3	CAN_GND
4	ADR0
5	ADR1
6	(GND)
7	CAN_H
8	ADR2
9	CAN_V+



(X12) D-SUB 9-pin, socket, CAN output;  
 (X11) D-SUB 9-pin, connector, CAN output

#### Important!

This terminal setting of the device address is only taken into account when parameter 493-CAADR is set to 0. For more details refer to section 4.2.1.



### 3.3 LED status display

For initial system diagnosis during commissioning a red LED is arranged between the two D-SUB connectors. This LED indicates three different bus states:

LED	Bus state
Permanently lit	BUS OFF
Flashing	ERROR
Permanently off	OK

## 4 Commissioning and configuration

### 4.1 Commissioning sequence

This section outlines the steps for initial commissioning of a servodrive.

For more detailed information on optimizing the speed or position control circuit, refer to the device operation manual.

1. Wire up the device, encoder and motor.  
Ensure the motor is connected to the correct phase.
2. Load motor data from the motor database of the DRIVEMANAGER user interface or from a SMARTCARD into the servocontroller.
3. Activate the operation mode you want in the device by way of the DRIVEMANAGER user interface (speed control, position control, ...).
4. Configure the device according to the application requirements.  
(inputs/outputs, encoder simulation, position control, ...).
5. Test the control quality and optimize the controller settings (speed controller) as set out in the device operation manual.
6. Set the CAN-specific parameters; see below.
7. Test higher-order controller.
8. Save device configuration.

### 4.2 Key parameter settings

#### 4.2.1 Assignment of device address (NodeID)

As already mentioned, the device address can be assigned in two ways. The decisive factor is the setting of parameter **493-CAADR**. If the parameter is set to the value 0, the device address is taken from the connector configuration after the system starts.

**Important!** Device address 0 is not permitted if no address is coded in the connector. In such cases the servo transmits an emergency message at 125 KB with NodeID 1.

If the parameter is set to values between 1 and 29, the connector configuration is ignored and the servo starts after the reset with the device address set in CAADR.

## 4.2.2 Baud rate setting

The baud rate is set by parameter **499-COBDR**. 500K, 250K, 125K, 50K, 20K and 10K baud rates are supported.

COBDR	Transmission speed	Comments
2	500 KBaud	Factory setting
3	250 KBaud	
4	125 KBaud	
6	50 KBaud	
7	20 KBaud	
8	10 KBaud	

**Important!** A change of device address or baud rate only takes effect after the next reset (restart) of the servo!

## 4.3 Connection test

### 4.3.1 Bus message after system start

When the device address and baud rate parameters have been set and after the reset, the servo sends a boot-up message over the CAN bus. This boot-up message has the same identifier as the emergency message, but contains no data.

ID	dlc
80 H + NodeID	0

dlc = data length code

### 4.3.2 SDO transfer: Reading manufacturer hardware version

As already described in section 2.2 "Parameter-setting channel (ServiceDataObjects)", the SDO channel is used to read or write device parameters. SDO read access to the **manufacturer hardware version** object with

**Object:**

Draft 301	<b>1009H</b>	VAR	<b>manufacturer hardware version</b>	Vis-String	ro	O
-----------	--------------	-----	--------------------------------------	------------	----	---

**SDO access**

SDO transfer mode	Index	SubIndex
Read	1009H	00H

ID	Data bytes	;Description
SDO(rx)	40 09 10 00 00 00 00 00	; Upload protocol (read) Master->Servo
SDO(tx)	42 09 10 00 37 34 30 34	; Servo->Master

returns a string which indicates the performance class of the device:

For MC7404 ... :       **"7404"**

### 4.3.3 SDO transfer: Writing and reading device parameters

#### 4.3.3.1 Where can I find the device parameters?

All device parameters are addressed by way of a parameter number. In the DRIVEMANAGER PC user software you will find parameter numbers between 1 and 999.

In addition to the standard objects, the CANopen profile additionally provides a range for manufacturer-specific entries. This range is between 2000H and 5FFFH. If you then wish to read or write the parameter 515 POEGW (rapid feed rate) of the device, the object index is formed from 2000H + parameter number (Hex).

In our example:    Index = 2000H + 203H

### 4.3.3.2 Description of data types

The servocontrollers of the MASTERCONTROL series support the following parameter data formats:

Data type	Scaling / Increment	Value range	KP100 display	Data type DS301
PT_USIGN8	1	0 .. 255	0 .. 255	unsigned8
PT_USIGN16	1	0 .. 65535	0 .. 65535	unsigned16
PT_FIXPOINT16	0.05	0.00 .. 3276.80	0.00 .. 999.95	integer16
PT_INTEGER16	1	-32768 .. 32767	-9999 .. 32767	integer16
PT_INTEGER32	1/65536	-32767.99 .. 32766.99	-9999 .. 32767	integer32
PT_FLOAT_IEEE	see IEEE	see IEEE	-99.99E9 .. 99.99E9	float
PT_MC_ERROR	1	0 .. 65535	Plain text e.g. "E-OV"	unsigned16
PT_PASSWORD	1	0 .. 65535	0 .. 65535	unsigned16

#### Notes on the data types:

##### Data type PT\_INTEGER32

This data type is a "signed long" (32-bit) type with scaling 1/65536. It is displayed on the KEYPAD in the same way as a PT\_FLOAT\_IEEE.

##### Data type PT\_FLOAT\_IEEE

This data type corresponds to a 32-bit floating-point number in IEEE format.

##### Data type PT\_MC\_ERROR

This data type is structured and is 32 bits long. Its structure is as follows:

Structure element	Error number	Error location	Error time
Data length	1 byte	1 byte	2 bytes

## Notes on use of the individual data types in C

In the following examples it is assumed that the CAN data block has been imported from the CAN controller into the RAM of the master. The SDO/PDO data start from the memory location *Data byte[ 3]* in the RAM.

If the same physical address is accessed with different data types in C, this is best done by way of "union" structures.

The following example demonstrates data accessing via the "union":

```
/*-----*\
|
|   Example program for parameter data access within
|   the CAN data block
|
\*-----*/

/*-----*\
|
|   Externals
|
\*-----*/
extern unsigned char Data byte[];          /* The Can data block */
/*-----*\
|
|   Definitions
|
\*-----*/
typedef struct                               /* Structure of the error data */
{
    unsigned char Number;
    unsigned char Location;
    unsigned int Time;
}MC_Error;

typedef union                                /* Union for parameter data access */
{
    unsigned char usign8;
    unsigned int usign16;
    signed int int16;
    signed long int32;
    float float32;
    MC_Error err32;
} MC_Data;

/*-----*\
|
|   Function MAIN
|
\*-----*/
void main( void)
{
    /*-----*\
    |
    |   Test variables for data exchange
    |
    \*-----*/
    signed int   TestInt16;
    unsigned char TestUsign8;
    unsigned int TestUsign16;
    float TestFloat32;
    float TestFix16;
    float TestInt32Q16;
    unsigned int ErrorTime;
    unsigned char ErrorLocation;
    unsigned char ErrorNumber;
    MC_Data *CanParaData;

    /*-----*\
    |
    |   Read access to the parameter data
    |
    \*-----*/
    /* Position data pointer */
    CanParaData = &(amp;data byte[ 3]);

    /* Data access PT_USIGN8 */
    TestUsign8 = CanParaData->usign8;

    /* Data access PT_PASSWORD or PT_USIGN16 */
    TestUsign16 = CanParaData->usign16;

    /* Data access PT_INTEGER16 */
    TestInt16 = CanParaData->int16;

    /* Data access PT_INTEGER32 */
    TestInt32Q16 = (float)( CanParaData->int32) / 65536.;
}
```

```

/* Data access PT_FIXPOINT16 */
TestFix16 = (float)( CanParaData->usign16) /20.;

/* Data access PT_MC_ERROR */
ErrorTime = CanParaData->err32.Time;
ErrorNumber = CanParaData->err32.Number;
ErrorLocation = CanParaData->err32.Location;

/* Data access PT_MC_ERROR */
TestFloat32 = CanPara->float32;

/*-----*\
|           Write access to the parameter data
\*-----*/
/* Place the value 3 on a parameter of type PT_USIGN8 */
CanParaData->usign8 = 3;

/* Place the value 3 on PT_USIGN16 or PT_PASSWORD */
CanParaData->usign16 = 3;

/* Place the value -3 on a parameter of type PT_INTEGER16 */
CanParaData->int16 = -3;

/* Place the value 2345.3456 on a parameter of type PT_INTEGER32 */
CanParaData->int32 = (signed long)( 2345.3456 * 65536.);

/* Place the value 23.15 on a parameter of type PT_FIXPOINT16 */
CanParaData->usign16 = ( unsigned int)( 23.15 * 20.);

/* PT_MC_ERROR is read-only */

/* Place the value 2345.1234 on a parameter of type PT_FLOAT_IEEE */
CanPara->float32 = 2345.1234;
}

```

## 5 Control and reference input

### 5.1 Setup of control location and reference transfer

#### 5.1.1 MC7000 parameters for bus operation

##### 5.1.1.1 402-CLSEL - Control location

The control location is selected by way of parameter 402-CLSEL. With CLSEL = OPTN1 the control word for the DriveCom state machine is formed from bytes 0 and 1 of RXPDO1.

Minimum	Maximum	Factory set.	Unit	MODE	SmartCard	Type
See table		TERM	–	R4W4	REFRC	USIGN8

No.	Setting	Designation	Function
1	TERM	Terminal	Control drive via terminal strip (input configured as "Start")
2	KPAD	KEYPAD	Control drive via KEYPAD
3	SIO	Serial Input/Output	Control drive via serial interface (LustBus control word)
4	<b>OPTN1</b>	<b>Option 1</b>	<b>Control drive via module in slot 1 (CANopen)</b>
5	CAN	CAN bus	Control drive via CAN bus (LUST protocol)
6	POMOD	PosMod1	Control drive via position control

##### 5.1.1.2 419-RSSL3 - Reference selector

Set RSSL3 = ROPT1 so the reference value will be formed from bytes 2-5 of RXPDO1.

Minimum	Maximum	Factory set.	Unit	MODE	SmartCard	Type
See table		TERM	–	R4W4	REFRC	USIGN8

No.	Setting	Reference source is:
0	RCON	None (reference channel deactivated [constant =0])
1	RA0	Analog input ISA0
2	RA1	Analog input ISA1
3	RSIO	Serial interface
4	RPOT	Motor operated potentiometer at digital inputs
5	RDIG	Digital reference input (PWM)
6	<b>ROPT1</b>	<b>Module in slot 1 (CANopen)</b>
7	RCAN	CAN bus (LUST protocol)
8	RFIX1	Fixed reference 1
9	RFIX2	Fixed reference 2
10	RFIX3	Fixed reference 3
11	RFIX4	Fixed reference 4
12	RFIX5	Fixed reference 5
13	RFIX6	Fixed reference 6
14	RLIM1	Lower limit of reference value
15	RLIM2	Upper limit of reference value



Note: The reference and control values, or the content of RXPDO1, are only evaluated in the NMT state **Operational**.

### 5.1.1.3 491-CACTR - Control word

The control word received via RXPDO1 is entered in parameter 491-CACTR. During commissioning the parameter can be used to check receipt of the RXPDO1 data.

Minimum	Maximum	Factory set.	Unit	MODE	SmartCard	Type
0000H	FFFFH	0000H		R3W4	ALL	USIGN16

### 5.1.1.4 490-CASTA - Status word

The status of the DriveCom state machine is entered in parameter 490 CASTA. The data content of the parameter corresponds to data bytes 0 and 1 in TXPDO1.

Minimum	Maximum	Factory set.	Unit	MODE	SmartCard	Type
0000H	FFFFH	0000H		R3W15	ALL	USIGN16

### 5.1.1.5 497-RCAN - Reference from CAN bus

The reference value received via RXPDO1 is entered in parameter 497 RCAN. The data content of the parameter corresponds to data bytes 2-5 of RXPDO1. The interpretation of the value depends on the chosen operation mode.

Minimum	Maximum	Factory set.	Unit	MODE	SmartCard	Type
-32764	32764	0		R4W14	ALL	INT32Q16

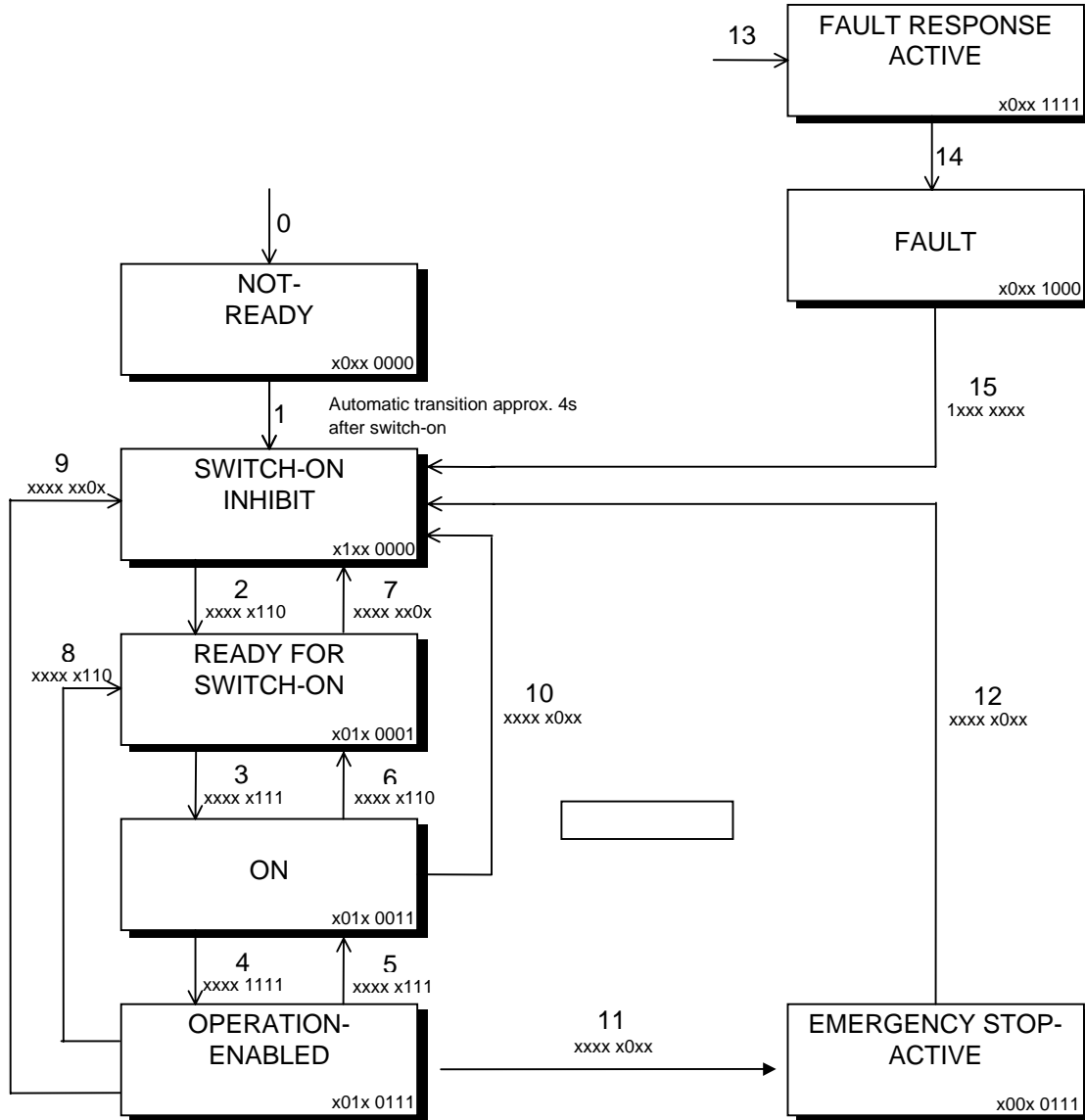
### 5.1.1.6 400-ACTV - Actual value

The current actual value is shown in parameter 400 ACTV and corresponds to data bytes 2-5 in TXPO1. The interpretation of the value depends on the chosen operation mode.

Minimum	Maximum	Factory set.	Unit	MODE	SmartCard	Type
-32764	32764	0		R1W15	ALL	INT32Q16

## 5.2 DriveCom state machine to DS402

### 5.2.1 Cross-mode information



### 5.2.1.1 Control and reference input via PDO channel

When the control location parameter is set to 402-CLSEL = OPTN1 (CANopen module C15), the first word of RXPDO1 is always interpreted as the control word. The 16 bits of the control word result from the logical linking of control commands which act on the state machine. Bits 4 to 6 are mode-specific, bits 11 to 15 manufacturer-specific.

If the parameter RSSL3 is additionally set to ROPT1 (CANopen module C15), the reference value is formed from bytes 2-5 of RXPDO1. The interpretation of the reference value depends on the control mode (speed-controlled operation bytes 2-5 = reference speed in rpm).

### 5.2.1.2 Status and actual value calculation via PDO channel

The first word of TXPDO1 always contains the status word. In the status word the current state of the device and additional messages are displayed. Bits 12 to 15 offer room for mode-dependent or manufacturer-specific displays.

When RSSL3 is set to ROPT1 bytes 2-5 contain the current actual value. The actual value is oriented to the chosen control mode (speed-controlled operation bytes 2-5 = actual speed in rpm).

## 5.3 Control of MC7000 in BASIC modes

### 5.3.1 Structure of the control word in BASIC modes

Bit	Name
0	Switch-on
1	Disable power
2	Emergency stop
3	Enable operation
4	Mode-dependent
5	Mode-dependent
6	Mode-dependent
7	Reset fault
8	Reserve
9	Reserve
10	Reserve
11	Manufacturer-specific
12	Manufacturer-specific
13	Manufacturer-specific
14	Manufacturer-specific, Reference state output OS00
15	Manufacturer-specific, Reference state output OS01

The reserve bits 9 and 10 are reserved for profile expansions, and must always be set to 0.

## Device control commands

The following bit combinations of the control bits 0-3 and 7 form the device control command for the state transitions of the state machine (cf. section 5.2.1):

Command:	Control bit					Transitions:
	7	3	2	1	0	
SHUTDOWN	X	X	1	1	0	2, 6, 8
POWER-UP	X	X	1	1	1	3
DISABLE POWER	X	X	X	0	X	7, 9, 10, 12
EMERGENCY STOP	X	X	0	1	X	7, 10, 11
DISABLE OPERATION	X	0	1	1	1	5
ENABLE OPERATION	X	1	1	1	1	4
RESET FAULT	0⇒1	X	X	X	X	15

### 5.3.2 Structure of the status word in BASIC mode

Bit	Name
0	Ready for start
1	On
2	Operation enabled
3	Fault
4	Power disabled
5	Emergency stop
6	Switch-on inhibit
7	Warning
8	Manufacturer-specific
9	Remote
10	don't care
11	Limit value
12	Mode-dependent
13	Mode-dependent
14	Manufacturer-specific
15	Manufacturer-specific

State:	Status bit					
	6	5	3	2	1	0
NOT READY	0	X	0	0	0	0
SWITCH-ON INHIBIT	1	X	0	0	0	0
READY	0	1	0	0	0	1
ON	0	1	0	0	1	1
OPERATION ENABLED	0	1	0	1	1	1
FAULT	0	X	1	0	0	0
FAULT RESPONSE ACTIVE	0	X	1	1	1	1
EMERGENCY STOP ACTIVE	0	0	0	1	1	1

## 5.4 Control of MC7000 in Electronic Gearing mode

### 5.4.1 Structure of the control word in Electronic Gearing mode

Bit	Name	Comments
0	Switch-on	DRIVECOM state machine for controller enable
1	Disable power	
2	Emergency stop	
3	Enable operation	
4	Manufacturer-specific	
5	Manufacturer-specific	
6	Manufacturer-specific	
7	Reset fault	
8	Reserve	
9	Reserve	
10	Reserve	
11	Manufacturer-specific	
12	Manufacturer-specific	
13	Manufacturer-specific	
14	<b>Engage/disengage</b>	"Electronic gearing" mode, "Stepper motor interface" only
15	<b>CamCircuit</b>	"Electronic gearing" mode, "Stepper motor interface" only

### 5.4.2 Structure of the status word in Electronic Gearing mode

Bit	Name	Comments
0	Ready for start	DRIVECOM state machine
1	On	
2	Operation enabled	
3	Fault	
4	Power disabled	
5	Emergency stop	
6	Switch-on inhibit	
7	Warning	
8	vacant	
9	Remote, always 1	
10	Mode-dependent	
11	Limit value	
12	Mode-dependent, <b>Engaged</b>	Electronic gearing and stepper motor interface only
13	Mode-dependent	
14	Manufacturer-specific	
15	Manufacturer-specific	

### 5.4.3 Example: Activation in Electronic Gearing mode

Presets:

1. Load motor data set via DRIVEMANAGER user interface
2. Activate Electronic Gearing mode via DRIVEMANAGER user interface
3. Set following parameters in Parameter Editor:
  - 402-CLSEL = OPT1
  - 489-COBDR = 500                      Set baud rate
  - 493-CAADR = 1                         Device address
4. Mains reset to activate changed settings

The drive can now be started with input ENPO set, with the following control sequence:

ID	Data bytes	;Description
NMT	01	;GoOperational
RXPDO1	06 00 00 00 00 00 00 00	;Shut down DriveCom
RXPDO1	0F 00 00 00 00 00 00 00	;Switch on Drivecom
RXPDO1	2F 00 00 00 00 00 00 00	;Request reference run (not engaged)
RXPDO1	0F 40 00 00 00 00 00 00	;Engage EGear
SDO(rx)	2f 83 21 00 01 00 00 00	;Write SDO <i>VRNOM-183h</i> = 1 (object 2183h)
SDO(tx)	60 83 21 00 00 00 00 00	
SDO(rx)	2f 84 21 0A 00 00 00 00	;Write SDO <i>VRDEN-184h</i> = 10(object 2184h)
SDO(tx)	60 84 21 00 00 00 00 00	;
SDO(rx)	2f 83 21 00 0A 00 00 00	;Write SDO <i>VRNOM-183h</i> = 10 (object 2183h)
SDO(tx)	60 83 21 00 00 00 00 00	;
SDO(rx)	2f 84 21 01 00 00 00 00	;Write SDO <i>VRDEN-184h</i> = 1 (object 2184h)
SDO(tx)	60 84 21 00 00 00 00 00	;
RXPDO1	0F 00 00 00 00 00 00 00	;Disengage EGear
RXPDO1	06 00 00 00 00 00 00 00	;Shut down DriveCom
NMT	80	;GoPreOperational

Explanation:

Parameter	387-VRNOM	Numerator for transmission ratio of electronic gearing	183 Hex
	388-VRDEN	Denominator for transmission ratio of electronic gearing	184 Hex

## 5.5 Control of MC7000 in PosMOD mode

### 5.5.1 Structure of the control word in PosMOD mode

Bit	Name
0	Switch-on
1	Disable power
2	Emergency stop
3	Enable operation
4	Mode-dependent, <b>Auto</b>
5	Mode-dependent, <b>Start</b>
6	Mode-dependent, <b>Feed hold</b>
7	Reset fault
8	Reserve
9	Reserve
10	Reserve
11	Manufacturer-specific, <b>Update</b>
12	Manufacturer-specific, <b>Jog+</b>
13	Manufacturer-specific, <b>Jog-</b>
14	Manufacturer-specific, Reference state output OS00
15	Manufacturer-specific, Reference state output OS01

### 5.5.2 Structure of the status word in PosMOD mode

Bit	Name
0	Ready for start
1	On
2	Operation enabled
3	Fault
4	Power disabled (low-active)
5	Emergency stop (low-active)
6	Switch-on inhibit
7	Warning
8	Manufacturer-specific, <b>No tracking error</b>
9	Remote
10	don't care
11	Limit value
12	Mode-dependent, <b>Axle in position</b>
13	Mode-dependent, <b>Ref.pt. defined</b>
14	Manufacturer-specific, <b>Prog. end</b>
15	Manufacturer-specific, <b>Tracking error</b>

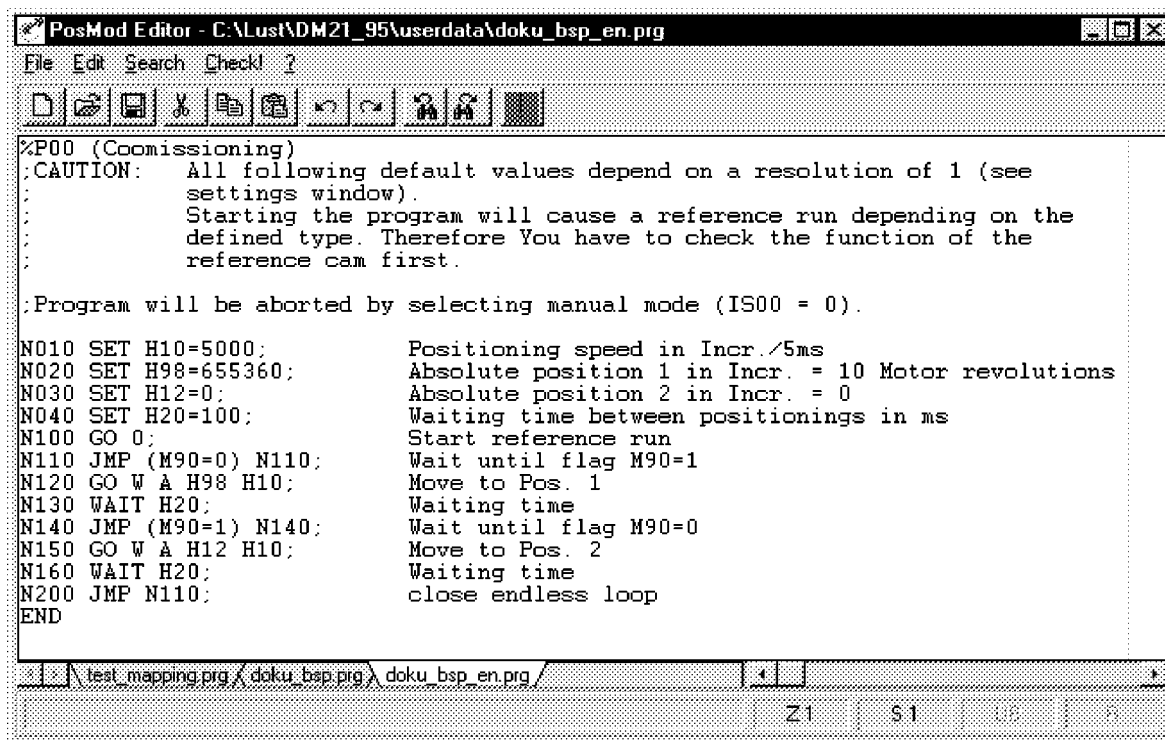
### 5.5.3 Example: PosMOD activation

Task:

- Load a sequence program into the servo axle and activate it over CAN.
- In this process, the positioning is to be controlled by the status of a flag between the absolute position 0 and a freely adjustable position.

Presets:

1. Load motor data set via DRIVEMANAGER user interface
2. Activate positioning and sequence control mode via DRIVEMANAGER user interface
3. Set following parameters in Parameter Editor:  
402-CLSEL = OPTN1  
489-COBDR = 500           Set baud rate  
493-CAADR = 1             Device address
4. Load sequence program into servocontroller
5. Mains reset to activate changed settings
6. Wire control contact hardware enable ENPO



The screenshot shows a window titled "PosMod Editor - C:\Lust\DM21\_95\userdata\doku\_bsp\_en.prg". The window contains a sequence program with the following text:

```
%P00 (Coommissioning)
;CAUTION: All following default values depend on a resolution of 1 (see
; settings window).
; Starting the program will cause a reference run depending on the
; defined type. Therefore You have to check the function of the
; reference cam first.

;Program will be aborted by selecting manual mode (IS00 = 0).

N010 SET H10=5000;           Positioning speed in Incr./5ms
N020 SET H98=655360;        Absolute position 1 in Incr. = 10 Motor revolutions
N030 SET H12=0;             Absolute position 2 in Incr. = 0
N040 SET H20=100;          Waiting time between positionings in ms
N100 GO 0;                  Start reference run
N110 JMP (M90=0) N110;      Wait until flag M90=1
N120 GO W A H98 H10;        Move to Pos. 1
N130 WAIT H20;             Waiting time
N140 JMP (M90=1) N140;      Wait until flag M90=1
N150 GO W A H12 H10;        Move to Pos. 2
N160 WAIT H20;             Waiting time
N200 JMP N110;             close endless loop
END
```

The window also shows a toolbar with various icons and a status bar at the bottom with indicators for Z1, S1, and other parameters.

Notes:

- Flag M90 triggers the positioning operations with an edge change.
- Variable H98 contains the freely selectable reference position.  
Unit = increments



The drive can now be started with input ENPO set, with the following control sequence:

ID	Data bytes	Comments
NMT	01	;GoOperational
RXPDO1	06 00 00 00 00 00 00 00	;Shut down Drivecom (0->6)
RXPDO1	4F 08 00 00 00 00 00 00	;Switch on Drivecom (6->F) ;Feed hold (4) always set ;Update (8) always set
RXPDO1	5F 08 00 00 00 00 00 00	;Automatic enable (5) <sup>1)</sup>
RXPDO1	7F 08 00 00 00 00 00 00	;Start enable, sequence program started (7)
RXPDO21	FF 62 00 00 00 0A 5A 00	;POMER[90] = 0, ;POVAR[98] = 655360 incr., ;Destination position = 10 motor revs (absolute)
RXPDO21	FF 62 00 00 00 0A 5A 01	;POMER[90] = 1, Start positioning ;POVAR[98] = 655360 incr.,
RXPDO21	FF 62 64 00 00 00 5A 00	;POMER[90] = 0, Trigger positioning to pos. 0 ;POVAR[98] = 100 incr.,
RXPDO21	FF 62 64 00 00 00 5A 01	;POMER[90] = 1, Trigger positioning ;POVAR[98] = 100,
.....		
RXPDO1	4F 08 00 00 00 00 00 00	;Switch on Drivecom (6->F) ;Feed hold (4) set ;Update (8) set ;Axle in manual mode, axle can be moved ;by "Jog" function. ;Parameter settings and download of sequence prog. ;possible
RXPDO1	06 00 00 00 00 00 00 00	;Shut down Drivecom (0->6), power stage off
NMT	80	;GoPreOperational

<sup>1)</sup> A delay of 10 ms is required between Automatic and Start Enable

Explanation:

Parameter	528-POVAR	PosMOD variables	210 Hex
	529-POMER	PosMOD flags	211 Hex

## 6 Profile support in detail

### 6.1 Telegram structure

The CAN bus is extremely well suited to the transfer of small data volumes and control sequences. On the CAN bus up to eight data bytes can be transmitted in one transfer.

Structure of a data frame:

Number of bits	1 bit	11 bits	1 bit	6 bits	0...8 bytes	15 bits	1 bit	1 bit	1 bit	7 bits	> 3 bits
	A	B	C	D	E	F	G	H	I	J	K

- A START OF FRAME
- B IDENTIFIER (Arbitration)
- C REMOTE TRANSMISSION REQUEST BIT (Arbitration)
- D CONTROL
- E DATA
- F CYCLIC REDUNDANCY CODE
- G CYCLIC REDUNDANCY CODE DELIMITER
- H ACKNOWLEDGE SLOT
- I ACKNOWLEDGE DELIMITER
- J END OF FRAME
- K INTERFRAME SPACE

Apart from ranges B, C and E, the CAN controller independently controls the status of the bits. The ranges B, C and E are determined by the user protocol.

B contains the transmission identifier.

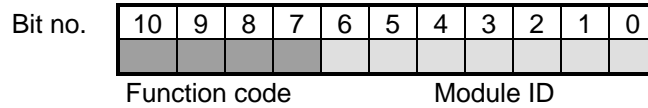
With flag C an automatic checkback of the transferred protocol can be requested on the bus. This mode is not applied in our devices, and must be 0.

All other protocol definitions relate to range E.

## 6.2 Default setting of communication objects

The settings of the standardized objects of the CANopen profile (COB-ID of emergency, Sync, transmission types of PDOs etc.) correspond to a so-called Predefined Connection Set. This a pre-configuration intended to provide the user with a fast commissioning process. The following table shows the delivery condition ex factory.

### COB identifier



Object	Function code	COB-ID	Communication parameters as from index
NMT	0000	0	-
Sync	0001	128	1005H
Emergency	0001	129-255	-
PDO1(tx)	0011	385-511	1800H
PDO1(rx)	0100	513-639	1400H
PDO2(tx)	0101	641-767	1801H
PDO2(rx)	0110	769-895	1401H
SDO(tx)	1011	1409-1535	-
SDO(rx)	1100	1537-1663	-
NodeGuard	1110	1793-1919	100EH



## 6.4 Response of the servo in a network

When the control location parameter 402-CSEL = OPTN1 (CANopen module C15) is set, the network status has a direct influence on the DriveCom state machine. Only by enabling the ENPO and by **transition to Operational mode** is it possible with the aid of RXPDO1 to quit the DriveCom state 2, Ready for start (see Control of POSMOD to DS402). The transition of the network status while the drive is running in the PreOperational state results in a return to the "Ready for start" state by way of an emergency stop.

## 6.5 Setting up application-specific PDOs

### 6.5.1 Changing the PDO communication parameters

For each PDO a so-called Communications Parameter Area is reserved in the object directory.

Draft 301	<b>1400H</b>	RECORD	<b>Receive PDO1 Comm.-Parameter</b>	PDOCommPar	rw	○
Draft 301	<b>1401H</b>	RECORD	<b>Receive PDO2 Comm.-Parameter</b>	PDOCommPar	rw	○

Draft 301	<b>1800H</b>	RECORD	<b>Transmit PDO1 Comm-Parameter</b>	PDOCommPar	rw	○
Draft 301	<b>1801H</b>	RECORD	<b>Transmit PDO2 Comm-Parameter</b>	PDOCommPar	rw	○

This includes the following entries.

SubIndex	Data type	Meaning
0	Unsigned8	Number of entries
1	Unsigned32	COB-ID
2	Unsigned8	Transfer mode
3	Unsigned16	Inhibit time
4	Unsigned8	CMS priority

The decisive factor for adaptation of the PDO communication parameters to your application is entry 2, Transfer mode. By SDO write access you can assign the following transfer modes to the PDO:

- **acyclic synchronous**                      **Value: 00H**
- **cyclic synchronous**                        **Value: 1-F0H**
- **asynchronous**                               **Value FEH**

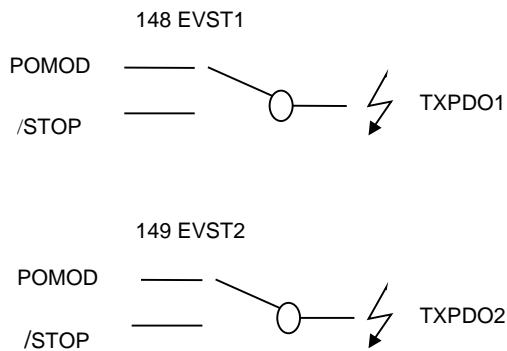
Meanings of the transfer types based on the example of the static RXPDO1 and TXPDO1:

Value	RXPDO1	TXPDO1
00H	When the RXPDO1 is received the MC7000 only adopts the control word, and reference value, after the next SYNC object (response time approx. 1ms).	The current status word and actual value are transmitted in the form of the TXPDO1 only after receipt of the next SYNC object.
1-F0H	When the RXPDO1 is received the MC7000 only adopts the control word, and reference value, after 1-F0H SYNC objects (response time approx. 1ms).	The current status word and actual value are transmitted in the form of the TXPDO1 only after receipt of 1-F0H SYNC objects.
FEH	When the RXPDO1 is received the MC7000 adopts the control word, and reference value, immediately (response time approx. 1ms).	1)

1) By selecting the asynchronous transmission mode (FEH) the transmission of the TXPDO can be triggered by an internal event. For each TXPDO the event can be defined via the parameters 148-EVST1(TXPDO1) resp. 149-EVST2 (TXPDO2).

At this time two events can be selected:

- Setting **POMOD**: Transmission of the TXPDO can be triggered by setting flag 98 (TXPDO1) resp. flag 99 (TXPDO2)
- Setting **/STOP**: Transmission of the TXPDOx can be triggered by reaching the standstill window.



The COB ID, inhibit time and CMS priority are defaults defined specially for the servo, and should not be changed.

## 6.5.2 Changing the PDO mapping

The MC7000 has two variable PDOs - that is, the contents of the PDOs can be defined by the users themselves within certain limits. This makes it possible to map parameters of the MC into the PDOs and to inform other stations of their contents over the CAN bus.

A PDO is mapped by way of the relevant mapping table in the object directory.

Draft 301	<b>1601H</b>	ARRAY	<b>Receive PDO2 Mapping-Parameter</b>	PDOMapping	rw	○
Draft 301	<b>1A01H</b>	ARRAY	<b>Transmit PDO2 Mapping-Parameter</b>	PDOMapping	rw	○

### Important!

Since the transmission capacity of a PDO is max. 8 bytes, a maximum of 8 objects of the type UNSIGNED8 can be mapped. Use of other data types reduces the number of maximum mappable objects correspondingly.

SubIndex	Meaning/content	Data type
00H	Number of mapped objects	unsigned8
01H	1st object (Parameter1)	unsigned32
02H	2nd object (Parameter1)	unsigned32
03H	...	unsigned32
04H	...	unsigned32
05H	...	unsigned32
06H	...	unsigned32
07H	...	unsigned32
08H	max. 8 objects (parameters)	unsigned32

PDO Byte	0	1	2	3	4	5	6	7
Content	1st object	2nd object	3rd object	4th object	5th object	6th object	7th object	8th object

### Important!

A Min/Max check of all mapped objects is carried out after receipt of the RXPDO2. If an object is outside the value range, the entire data content of the PDO is rejected.

Errors in the mapping table resulting from a PDO exceeding the transmission capacity or from an attempt to enter a non-mappable parameter generate a corresponding error message on the KEYPAD or trigger an emergency message.

### 6.5.2.1 Setting up a user-specific RXPDO

Task:

- In speed control mode, control and reference input and the facility for torque reduction is to be provided over the CAN bus.
- Control and reference input is by way of the static RXPDO1. Based on parameter 139 SCALE (data type: UNSIGNED8 / value range :1-100%) a mappable object is available for torque reduction. The parameter SCALE is mapped in RXPDO2.

Presets:

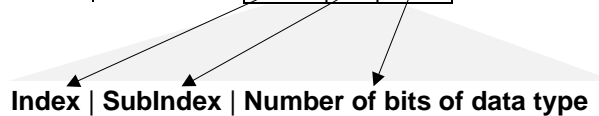
1. Load motor data set via DRIVEMANAGER user interface
2. Activate "Speed control with reference via +/-10V" mode via DRIVEMANAGER user interface
3. Switch to "Free operation mode"
4. Set following parameters in Parameter Editor:
  - 402-CLSEL = OPTN1
  - 489-COBDR = 500                      Set baud rate
  - 493-CAADR = 1                         Device address
  - 148-EVST1 = POMOD
5. Mains reset to activate changed settings
6. Wire control contact hardware enable ENPO

#### 1. Define object index

The parameter 139-SCALE is represented in the object directory by the object Index 208BH, SubIndex 00H.

#### 2. Enter in mapping table of RXPDO2 as 1st object (only in PreOperational state)

SDO transfer mode	Index	SubIndex	Value
Write	1601H	01H	0x208B0008
			208B   00   08



ID	Data bytes	;Description
SDO(rx)	23 01 16 01 08 00 8b 20	;Download protocol (write) Master->Servo
SDO(tx)	60 01 16 01 00 00 00 00	;Servo->Master

#### 3. Define number of mapped objects

SDO transfer mode	Index	SubIndex	Value
Write	1601H	00H	0x01



ID	Data bytes	;Description
SDO(rx)	23 01 16 00 01 00 00 00	;Download protocol (write) Master->Servo
SDO(tx)	60 01 16 00 00 00 00 00	;Servo->Master

#### 4. Define transfer mode of RXPDO1

In communication parameter of RXPDO1 enter transfer type "asynchronous" (0xFEH)

SDO transfer mode	Index	SubIndex	Value
Write	1401H	02H	0xFE

ID	Data bytes	;Description
SDO(rx)	23 01 14 02 FE 00 00 00	;Download protocol (write) Master->Servo
SDO(tx)	60 01 14 00 00 00 00 00	;Servo->Master
NMT	01	;GoOperational

#### 5. Activate ENPO

#### 6. Start drive

ID	Data bytes	;Description
NMT	01	;GoOperational
RXPDO1	06 00 00 00 00 00 00 00	;Shut down Drivecom (0->6)
RXPDO1	07 00 00 00 00 00 00 00	;Switch on Drivecom (6->F)
RXPDO1	0F 00 00 00 00 00 00 00	;Enable operation
RXPDO1	0F 00 00 00 64 00 00 00	;Reference 100 rpm
RXPDO2	3C 00 00 00 00 00 00 00	;Torque reduction to 60%
RXPDO2	64 00 00 00 00 00 00 00	;Torque reduction to 100%
RXPDO1	07 00 00 00 00 00 00 00	;Drivecom ready (F->7)
RXPDO1	06 00 00 00 00 00 00 00	;Shut down Drivecom (7->6)
NMT	80	;GoPreOperational

### 6.5.2.2 Setting up a user-specific TXPDO

Task:

- An external IO node is to be controlled from a PosMOD program. The reference state of the outputs is stored in the variable H01 (parameter 528-POVAR, field index 01). The content of the variable is then to be sent via variable TXPDO2.

Presets:

- Load motor data set via DRIVEMANAGER user interface
- Activate "Positioning and sequence control" mode via DRIVEMANAGER user interface
- Switch to "Free operation mode"
- Set following parameters in Parameter Editor:  
 402-CLSEL = OPTN1  
 489-COBDR = 500                      Set baud rate  
 493-CAADR = 1                        Device address  
 148-EVST1 = POMOD

- Load sequence program into servocontroller
- Mains reset to activate changed settings
- Wire control contact hardware enable ENPO

#### 1. Define object index

You will find the variable 528-POVAR with the field index 1 (= H01 in the sequence program) in the object directory under the object with Index 2210H, SubIndex 01H.

#### 2. Enter in mapping table of TXPDO2 as 1st object (only in PreOperational state)

SDO transfer mode	Index	SubIndex	Value
Write	1A01H	01H	0x22100120
			2210   01   20

Index | SubIndex | Number of bits of data type

ID	Data bytes	;Description
SDO(rx)	23 01 1a 01 20 01 10 22	;Download protocol (write) Master->Servo
SDO(tx)	60 01 1a 00 00 00 00 00	;Servo->Master

### 3. Define number of mapped objects

SDO transfer mode	Index	SubIndex	Value
Write	1A01H	00H	0x01

ID	Data bytes	;Description
SDO(rx)	23 01 1a 00 01 00 00 00	;Download protocol (write) Master->Servo
SDO(tx)	60 01 1a 00 00 00 00 00	;Servo->Master

### 4. Define transfer mode of TXPDO1

In communication parameter of TXPDO1 enter transfer type "asynchronous" (0xFEH)

SDO transfer mode	Index	SubIndex	Value
Write	1801H	02H	0xFE

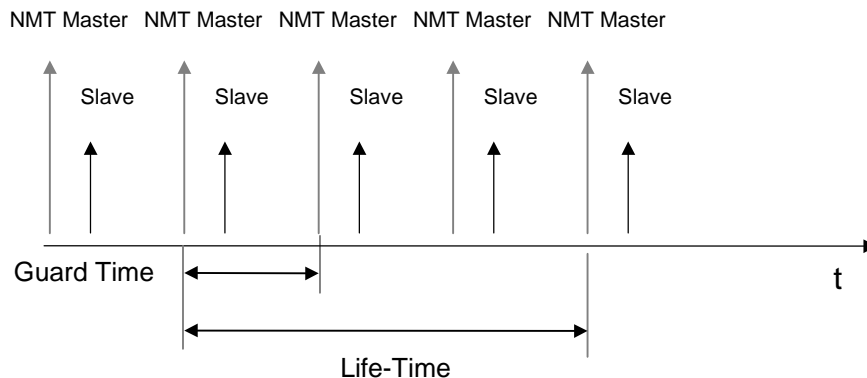
ID	Data bytes	;Description
SDO(rx)	23 01 18 02 FE 00 00 00	;Download protocol (write) Master->Servo
SDO(tx)	60 01 18 00 00 00 00 00	;Servo->Master
NMT	01	;GoOperational

## 5. START PosMOD program

ID	Data bytes	;Description
NMT	01	;GoOperational
RXPDO1	06 00 00 00 00 00 00 00	;Shut down Drivecom (0->6)
RXPDO1	4F 08 00 00 00 00 00 00	;Switch on Drivecom (6->F) ;Feed hold (4) always set ;Update (8) always set
RXPDO1	5F 08 00 00 00 00 00 00	;Automatic enable (5)
RXPDO1	7F 08 00 00 00 00 00 00	;Start enable, sequence program started (7)
TXPDO1	is transmitted	
....		
....		
RXPDO1	4F 08 00 00 00 00 00 00	;Switch on Drivecom (6->F) ;Feed hold (4) set ;Update (8) set ;Axle in manual mode, axle can be moved by "Jog" function. ;Parameter settings and download of sequence prog. possible
RXPDO1	06 00 00 00 00 00 00 00	;Shut down Drivecom (0->6), power stage off
NMT	80	;GoPreOperational

## 6.6 Monitoring by Node/Life Guarding

Two objects in the object directory are responsible for setting up Node/Life guarding.

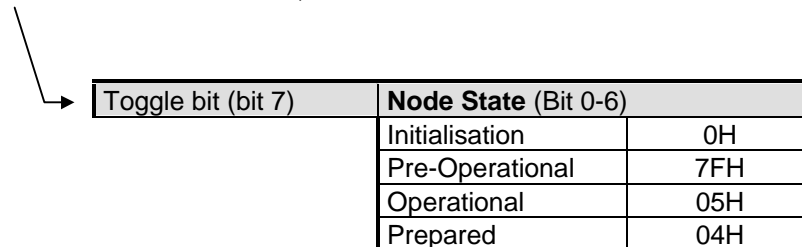


Draft 301	<b>100CH</b>	VAR	<b>guard time</b>	Unsigned32	rw	O
Draft 301	<b>100DH</b>	VAR	<b>life time factor</b>	Unsigned32	rw	O

By writing the object 100C H the NMT master can notify the slave of the so-called **guard time**. The Unsigned32 value of the object is given in ms. The guard time multiplied by the so-called **life time factor** (object 100DH) indicates the max. sampling time (**life time**) for the master and slave.

Node guarding is started by the first remote telegram from the master. The servo responds by transmitting its NMT status and a toggle bit.

ID	Data bytes	;Description
NodeGuard	Remote-Frame	;Master->Servo
NodeGuard	05-----	;Servo->Master
NodeGuard	Remote-Frame	;Master->Servo
NodeGuard	85-----	;Servo->Master
NodeGuard	Remote-Frame	;Master->Servo
NodeGuard	05-----	;Servo->Master
NodeGuard	Remote-Frame	;Master->Servo
NodeGuard	85-----	;Servo->Master



Toggle bit (bit 7)	Node State (Bit 0-6)	
	Initialisation	0H
	Pre-Operational	7FH
	Operational	05H
	Prepared	04H

If the life time is exceeded the servo switches to error state and transmits an emergency message with the error location **91 Error Guarding CAN-Master**.

## 6.7 EDS device file

The DS301 defines the content of a so-called EDS (Electronic Data Sheet) file. This text file contains all device-specific data and parameters in terms of their data type, value range and access attributes. A number of setup tools for CANopen networks use this file for graphical visualization of the individual CANopen nodes.

### 6.7.1 How do I create the EDS device file ?

The DRIVEMANAGER setup tool offers a submenu **Create EDS File** from the **Extras** menu. You can then specify again whether you want to create the EDS file from an existing device database stored on hard disk or floppy disk, or from the currently connected device. You can then copy the new EDS file to the relevant directory in your CANopen setup tool (e.g. CAN Analyzer from the Vector Corporation).

## 6.8 Saving the CANopen settings

The user-specific settings made in the object directory (objects 1000H-1A00H) can be permanently stored, and so are available every time the device is started.

The object 1010H with Subindex 0x02 delivers this functionality.

Draft 301	<b>1010H</b>	ARRAY	<b>Store parameters</b>	Unsigned32	rw	O
-----------	--------------	-------	-------------------------	------------	----	---

To prevent unwanted settings from being saved accidentally, the save operation is triggered by a defined code.

Save code                      0x65766173

The save operation can be triggered by the following SDO access:

SDO transfer mode	Index	SubIndex	Value
Write	1010H	02H	0x65766173

ID	Data bytes	;Description
SDO(rx)	23 10 10 02 73 61 76 65	; Download protocol (write) Master->Servo
SDO(tx)	60 10 10 00 00 00 00 00	; Servo->Master

**Important!** The save operation disables the communication capability of the servo for around 0.5 seconds, which means that, with Guarding active, the master would detect an error.

## 6.9 Restoring factory defaults

The factory defaults (Predefined Connection Set) of the above-mentioned communication settings can be restored by SDO write access to object 1011H.

Draft 301	<b>1011H</b>	ARRAY	<b>restore default parameters</b>	Unsigned32	rw	O
-----------	--------------	-------	-----------------------------------	------------	----	---

Here, too, a save code is used for safety.

Save code      0x64616f6c

The factory defaults can be restored by the following SDO access:

SDO transfer mode	Index	SubIndex	Value
Write	1011H	02H	0x64616f6c

ID	Data bytes	;Description
SDO(rx)	23 11 10 02 6c 6f 61 64	;Download protocol (write) Master->Servo
SDO(tx)	60 11 10 00 00 00 00 00	;Servo->Master

**Important!** The save operation disables the communication capability of the servo for around 0.5 seconds, which means that, with Guarding active, the master would detect an error.

## 6.10 Time response

Communication object		Type	Response time (ms)	Comments
PDO		RXPDO	max. 1-2	Time by which the data of the RXPDOs are processed.
		TXPDO	max. 1-2	Time after internal events until PDOs are transmitted.
SDO	*	Download protocol (init download domain.req)	max. 5	Max. time until the reply telegram (init download domain.com) is sent
		Upload protocol (init download domain.req)	max. 5	Time until the reply telegram (init download domain.com) is sent

- \* Segmented data transfer (>4 Byte): Keep a waiting time between the parts of telegrams (upload domain.req/download domein.req) of minimum 5 ms.

## 7 Fault rectification

### 7.1 Troubleshooting

All errors detected by the servo are transmitted over the CAN bus in the form of an emergency message. The cause of the error is additionally indicated on the KEYPAD (red display).

Byte:	0	1	2	3	4	5	6	7
Bit:	0 ... 15	16 ... 23	24 ... 31	32 ... 39	40 ... 47	48 ... 55	56 ... 63	
Profile	Device Profile DS402			MC 7000				
Content:	Emergency error code as per DS402	Error register (Object 1001H)	Error code	Error location	Operating hours on occurrence of the error			

**ID = 80H+ NodeID**

dlc data length code = 8

The decisive factors for rapid localization are the error code and error location. In bytes 3 and 4 of the emergency telegram you will find the error code, which represents a first categorization of the cause of the error. This error code is displayed on the KEYPAD as a 5-character text (see Emergency Codes table). The precise cause is determined by the error location. Bytes 6 and 7 contain the internal operating hours meter of the device (parameter 87 TOP).

CANopen errors - i.e. incorrect configurations, bus disturbances etc. - are indicated by error code 0xFF00. The KEYPAD then displays the text **E-OPT1**. For information on the causes of errors refer to the table which follows.

#### Important!

When an error occurs the servo **automatically** switches to the PreOperational state. The active drive is stopped by an emergency stop, the DriveCom state machine switches to **Ready**.

### 7.2 Resetting an error

The MC7000 offers several different ways of resetting an error.

#### 7.2.1 Error acknowledgment via bus system

Error are reset by transition from the **PreOperational** to the **Operational** state. Resetting of an error is signaled by transmission of the following emergency message:

ID	Data bytes	Description
Emergency	00 00 00 00 00 00 xx xx	;Emergency message acknowledgment error
	xx xx	Operating hours meter

If the cause of the error has not been eliminated, after transmitting another emergency message the MC7000 returns to the NMT state **PreOperational**.



Another possibility is offered by the object **6040H controlword**:

Draft 402	<b>6040H</b>	VAR	<b>controlword</b>	Integer16	rw	M
-----------	--------------	-----	--------------------	-----------	----	---

The error is also reset by assigning the object 6040H the value 0x0080 (SDO download protocol, because PDO transfer is only possible in Operational state).

### 7.2.2 Error acknowledgment, general

- By way of the KEYPAD (see Operation Manual)
- By means of a rising signal edge at input **ENPO**

## 7.3 Table: Emergency error codes

### Standard error messages of the MC7000

Error text	Error code	Emergency error code	Error register	Description
E-CPU	1	0x6010	7/manuf.	Hardware or software error
OFF	2	0x3120	2/voltage	Power failure
E-OC	3	0x2300	1/current	Current overload shut-off
E-OV	4	0x3110	2/voltage	Voltage overload shut-off
E-OLI	5	0x2200	1/current	I*t shut-off
E-OTM	6	0x4310	3/temp.	Motor overheating
E-OTI	7	0x4210	3/temp.	Servo overheating
E-EEP	8	0x6320	7/manuf.	Faulty EEPROM
E-OLM	9	0x2200	1/current	I*I*t shut-off
E-PLS	10	0x6320	7/manuf.	Plausibility error in parameter or program sequence
E-PAR	11	0x6320	7/manuf.	Faulty parameter setting
E-FLT	12	0x6320	7/manuf.	Floating-point error
E-PWR	13	0x5400	7/manuf.	Power pack not recognized
E-EXT	14	0x9000	7/manuf.	External error message (input)
E-ENC	15	0xFF00	7/manuf.	Encoder evaluation defective
<b>E-OP1</b>	<b>16</b>	<b>0xFF00</b>	<b>7/manuf.</b>	<b>Error in module in option slot 1</b>
E-OP2	17	0xFF00	7/manuf.	Error in module in option slot 2
E-TIM	18	0x6010	7/manuf.	Runtime error
E-FLW	19	0x8611	7/manuf.	Tracking error
E-WDG	20	0x8100	4/comm.	SIO watchdog
E-CAN	21	0xFF00	7/manuf.	Error in CAN hardware or software
E-IO1	22	0xFF00	7/manuf.	Input submodule not recognized
E-IO2	23	0xFF00	7/manuf.	Output submodule not recognized
E-VEC	24	0xFF00	7/manuf.	Error initializing VeCon processor
E-BRK	25	0xFF00	7/manuf.	Error at brake output OS03
<b>E-POS</b>	<b>26</b>	<b>0x6100</b>	<b>7/manuf.</b>	<b>Error message from PosMod1</b>
E-FLH	27	0x5530	7/manuf.	Error in FLASH memory
E-END	28	0x8500	7/manuf.	Hardware limit switch tripped
E-EEX	29	0x8500	7/manuf.	Hardware limit switches interchanged

### Specific CANopen error messages (error code = 16)

Error location	Description of error
80	Non-specified error
81	SDO receive queue overflow
82	Overflow in CAN controller
83	BUS-OFF
84	CAN communication
85	reserved
86	SDO transmit queue full
87	PDO receive queue overflow
88	Mapping Parameter Rx PDO 2
89	Mapping Parameter Tx PDO 2
90	Internal error
91	Error Guarding CAN-Master
92	reserved
93	reserved
94	reserved
95	reserved
96	PDO transmit queue full
97	reserved
98	reserved
99	reserved

### PosMOD error messages (error code = 26)

All errors signaled by the PosMOD are displayed with the error text 'E-POS'. Various error locations are used to differentiate between the errors.

Error location	Description of error
210	Positive hardware limit switch approached
211	Negative hardware limit switch approached
212	Positive software limit switch approached
213	Negative software limit switch approached
214	Reference point not defined
215	Addressed hardware not available
216	Selected program not available
217	Jump to non-existent record number
218	Called subroutine not available
219	Destination position outside positioning range
220	Division by zero
221	Max. nesting depth exceeded
222	Timeout in manual mode
223	Destination position not reached
224	No feed hold
225	Automatic mode, referencing or jog mode not possible because DRIVEMANAGER in manual mode
226	Index overflow (indexed addressing)
227	not used
228	not used
229	not used
230	Max. velocity of servo exceeded
231	not used
232	No controller enable
233	not used
234	not used
235	Impermissible command during axle movement

## 8 Appendix

### 8.1 Downloading a positioning program from the positioning and sequence control

A positioning program can be downloaded to the PosMOD software by writing to the string parameter 551-POCMD (PosMOD direct command input) line-by-line in manual mode.

#### Example program:

%P00 (Commissioning)	
N010 SET H10=5000;	Positioning speed in inc/5ms
N020 SET H11=655360;	Absolute position 1 in inc. = 10 motor revs
N030 SET H12=0;	Absolute position 2 in inc.
N040 SET H20=100;	Waiting time between positioning operations in ms
N100 GO 0;	Trigger referencing
N110 WAIT (IE01=1);	Wait until input IE01=1
N120 GO W A H11 H10;	Approach pos. 1
N130 WAIT H20;	Waiting time
N140 WAIT (IE01=0);	Wait until input IE01=0
N150 GO W A H12 H10;	Approach pos. 2
N160 WAIT H20;	Waiting time
N200 JMP N110;	Close endless loop
END	

The above example program is transferred line-by-line as a string via the parameter channel to the servocontroller. The comments separated by semicolons are eliminated in the process. That is to say, the following strings are transmitted as data:

1st string	"%P00 (Commissioning)"	
2nd string	"N010 SET H10=5000"	
3rd string	"N020 SET H11=655360"	
.		
.		
14th string	"END"	Servo detects end of program transmission
15th string	"%SAV"	Back-up program code in Flash. The execution time depends on the length of the program (approx. 100ms).

If a sequence program is to be overwritten, the original must first be deleted from the device memory. To this end the following string is transmitted:

String	"%CLPxx"
	↗ xx = Program number

**Important!** Sequence programs can only be transmitted with the sequence control in manual mode!

We reserve the right to make technical changes.

ID no.: 0808.44B.0-00

**EN** 06/99

---

Lust Antriebstechnik GmbH \* Gewebestr. 5-9 \* D-35633 Lahnau \* Phone +49 64 41 / 966 -0 \* Fax +49 64 41 / 966 -137

Internet: <http://www.lust-tec.de> \* e-mail: [lust@lust-tec.de](mailto:lust@lust-tec.de)