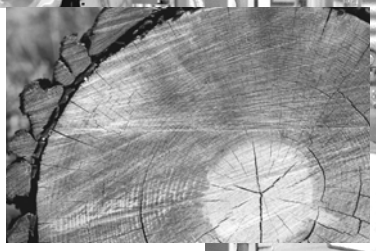
SEW EURODRIVE

MOVIDRIVE® MDX60B/61B
Communication and Fieldbus Unit Profile

Edition 04/2009
11264926 / EN

Manual



SEW EURODRIVE

1 General Information

1.1 How to use the documentation

The documentation is an integral part of the product and contain important information on operation and service. The documentation is written for all employees who assemble, install, startup, and service this product.

1.2 Structure of the safety notes

The safety notes in this documentation are structured as follows:

Pictogram	SIGNAL WORD
	Type and source of danger. Possible consequence(s) if disregarded. • Measure(s) to prevent the danger.

Pictogram	Signal word	Meaning	Consequences if disregarded
Example:  General danger  Specific danger, e.g. electric shock	DANGER	Imminent danger	Severe or fatal injuries
	WARNING	Possible dangerous situation	Severe or fatal injuries
	CAUTION	Possible dangerous situation	Minor injuries
	NOTICE	Possible damage to property	Damage to the drive system or its environment
	TIP	Useful information or tip. Simplifies the handling of the drive system.	

1.3 Rights to claim under limited warranty

A requirement of fault-free operation and fulfillment of any rights to claim under limited warranty is that you adhere to the information in the documentation. Read the documentation before you start working with the unit!

Make sure that the documentation is available to persons responsible for the system and its operation as well as to persons who work independently on the unit. You must also ensure that the documentation is legible.

1.4 Exclusion of liability

You must observe this documentation and the documentation of the connected units from SEW-EURODRIVE to ensure safe operation and to achieve the specified product characteristics and performance requirements. SEW-EURODRIVE assumes no liability for injury to persons or damage to equipment or property resulting from non-observance of the operating instructions. In such cases, any liability for defects is excluded.

1.5 Copyright

2 Safety Notes

2.1 Other applicable documentation

Only electrical specialists are allowed to perform installation and startup observing relevant accident prevention regulations and the MOVIDRIVE® MDX60B/61B operating instructions.

Read through these documents carefully before you commence installation and startup of the communication interfaces of MOVIDRIVE® B.

As a prerequisite to fault-free operation and fulfillment of warranty claims, you must adhere to the information in the documentation.

2.2 General notes on bus systems

MOVIDRIVE® B has communication interfaces that make it possible to adapt the MOVIDRIVE® B inverter to the particulars of the machinery within wide limits. As with all bus systems, there is a danger of invisible, external (as far as the inverter is concerned) modifications to the parameters which give rise to changes in the unit behavior. This may result in unexpected (not uncontrolled) system behavior.

2.3 Safety functions

The MOVIDRIVE® MDX60B/61B inverters may not perform safety functions without higher-level safety systems. Use higher-level safety systems to ensure protection of equipment and personnel. For safety applications, ensure that the information in the following publications is observed: "Safe Disconnection for MOVIDRIVE® MDX60B/61B".

2.4 Hoist applications

MOVIDRIVE® MDX60B/61B is not designed for use as a safety device in hoist applications.

Use monitoring systems or mechanical protection devices as safety equipment to avoid possible damage to property or injury to people.

2.5 Waste disposal

Observe the applicable national regulations.

Dispose of the following materials separately in accordance with the country-specific regulations in force, as:

- Electronics scrap
- Plastic
- Sheet metal
- Copper

3 Introduction

3.1 Content of the manual

The manual describes the communication interfaces of the MOVIDRIVE® MDX60B/61B inverter:

- 2 serial interfaces
- 2 CAN interfaces, one via DFC11B option card
- Fieldbus or Ethernet interface depending on the installed option card

The manual provides a description of the connection, configuration parameters, as well as of the process and parameter data exchange via the communication interfaces of MOVIDRIVE® B.

The manual also describes how MOVITOOLS® MotionStudio communicates with MOVIDRIVE® B using the communication interfaces, and how programmable logic - controllers (PLC) can control MOVIDRIVE® B via the communication interfaces.

3.2 Additional documentation

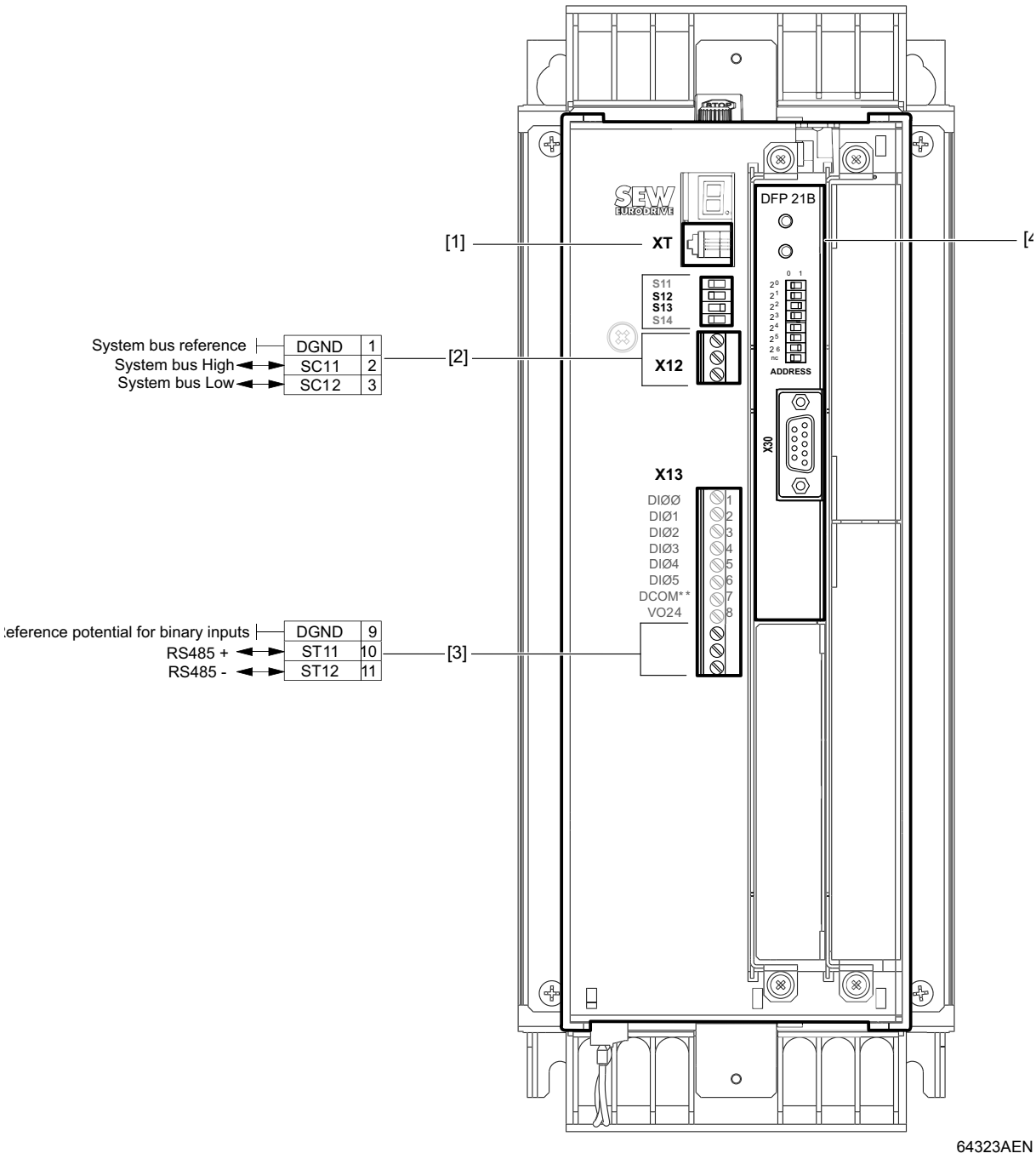
For simple connection of MOVIDRIVE® B to fieldbus systems, have the following documents at hand in addition to the manual:

- MOVIDRIVE® MDX60/61B system manual
- MOVIDRIVE® B list of parameters

  The MOVIDRIVE® B list of parameters describes the device parameters for every device firmware and includes a list of all device indices and error codes. It supplements the system manual as the latter is not rewritten for every firmware version.
- The manual of the fieldbus option used (e.g. DFP21B)

  The manuals on the fieldbus options and the manual at hand describe the access to process and parameter data in general and do not include a detailed description of all the possible control concepts.

3.3 Communication interfaces of MOVIDRIVE® B



[1] Terminal XT
[2] Terminal X12:SBus 1 (CAN)
[3] Terminal X13:10 / X13:11 (RS485)
[4] Fieldbus port

[1] Terminal XT:

RS485 interface for point-to-point connection of a keypad (e.g. DBG60B or DOP11B) or an interface adapter, such as USB11A or UWS21B for connection to an engineering PC.

[2] X12: SBus 1 (CAN) for connection

- directly to controllers (CANopen or MOVILINK® protocol)
- via an SEW fieldbus gateway to fieldbus systems, such as PROFIBUS, DeviceNet, etc.
- to an engineering PC via PC via PC CAN interface or SEW fieldbus gateway

[3] Terminal X13:10 / X13:11

RS485 interface for networking up to 32 devices, for example for connecting MOVIDRIVE® B to a DOP11B operator panel or for connection to an engeering PC via interface adapter (e.g. UWS11A, COM server, or similar)

[4] Fieldbus port

- for installing the DFC11B option card for the SBus 2 (CAN) interface with the same functionality as X12 (SBus1), or
- for installing a fieldbus option, for example PROFIBUS DFP21B, DeviceNet DFD11B, etc. for direct connection to the relevant fieldbus system for exchanging process and parameter data.

3.3.1 Overview of communication interfaces

	Serial interfaces		CAN interfaces		Fieldbus
Terminal / socket	Socket XT [1]	X13:10 / X13:11 [3]	Terminal X12 [2]	Fieldbus port [4]	
Type	RS485		CAN1	CAN2 or fieldbus option	
Profile	MOVILINK®		MOVILINK® or CANopen		PROFIBUS DP, DeviceNet, INTERBUS, etc.
Baud rate	9.6 / 57.6 kBaud (via S13)	9.6 kBaud	1000, 500, 250, 125 kBaud (with P884)	1000, 500, 250, 125 kBaud (with P894)	Depending on option card
Electrical isolation	No	No	No	Yes, on the DFC11B option	Yes
Connector	RJ10	Terminal	Terminal	Terminal and Sub D9 (according to CiA)	Depending on option card
Bus termination	Point-to-point	Dynamic	DIP switch S12	DIP switch R	Depending on option card
Control/setpoint source P100/P101	RS485		SBus 1	SBus 2	Fieldbus
Timeout monitoring	Shared monitoring via P812, P833		Via P883 and P836	Via P893 and P837	Via P819 and P831
Configuration of the interface (address, baud rate, etc.)	P810, P811		P88x	P89x	Depending on the option card via DIP switch or with P78x
Process data	Configuration using P870 - P876				
Master/slave	No	Yes	Yes	No	No
Manual operation (MOVITOOLS®)	Yes		No		
IPOS$^{plus®}$ bus type	1	2	5	8	3

4 Serial Interfaces of MOVIDRIVE® B

As standard, MOVIDRIVE® B is equipped with two separate, serial RS485 interfaces:

- Socket XT
- Terminals X13:10 and X13:11

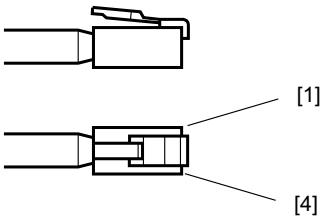
Telegrams received via a serial interface of MOVIDRIVE® B are **not** passed on via the other serial interface.

4.1 Connecting and installing RS485 interfaces

4.1.1 Connection using socket XT

The "XT socket" serial interface is designed as RJ10 plug connector (see following figure).

Assignment of XT connector (RJ10)



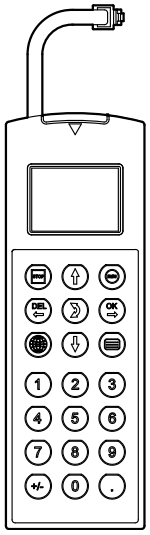
64788AXX

[1] DC 5 V (from electronics supply)
[2] RS485 + (Rx/Tx)
[3] RS485 - (Rx/Tx)
[4] GND (electronics ground)

Connection options

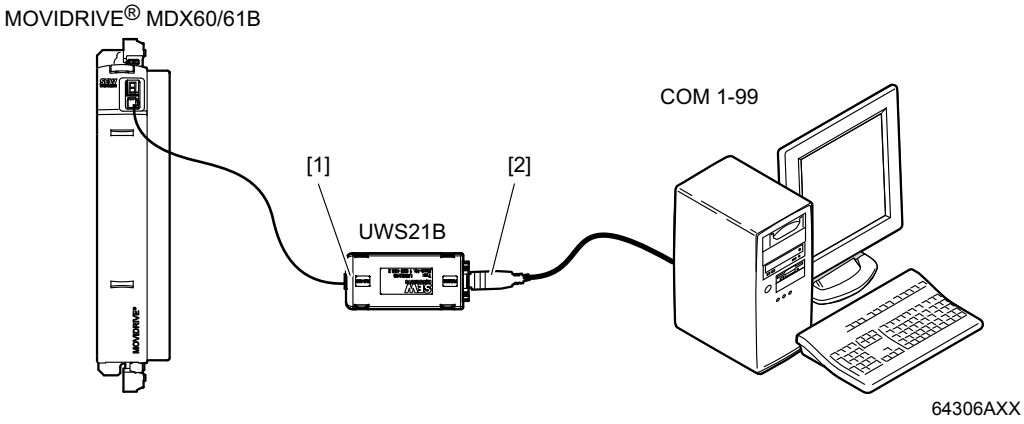
You can connect one of the following SEW options to the XT socket:

- DBG60B keypad



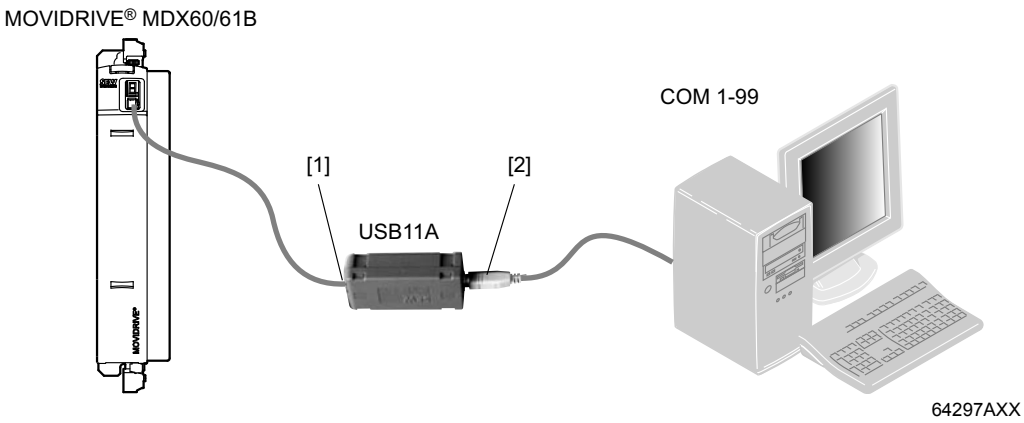
64252AXX

- UWS21B interface adapter (RS485 signals [1] to RS232 signals [2])



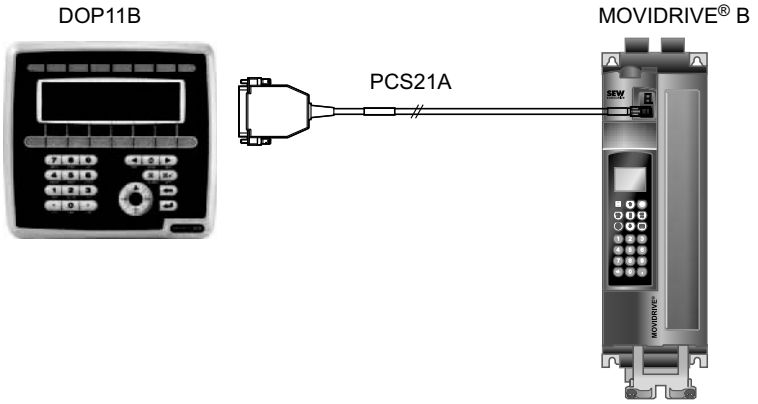
The UWS21B option is used to equip a MOVIDRIVE® B with a potential-free RS232 interface. The RS232 interface is designed as a 9-pole sub-D socket (EIA standard). A Sub D9 extension cable (1:1 connection) is supplied for connection to the PC.

- USB11A interface adapter (RS485 signals [1] to USB signals [2])



You can use the USB11A option for potential-free connection of a MOVIDRIVE® B to an engineering PC via USB. When installing the USB driver, a virtual COM port is created in the PC for communication with MOVIDRIVE® B.

- DOP11B operator terminal



64282AXX

	TIP
	The DBG60B, UWS21B and USB11A options are connected to the XT socket. The options cannot be used at the same time.

Electrical isolation

- The serial interface XT is not electrically isolated. It must be used for point-to-point connections only.

Terminating resistor

- Matching terminating resistors are integrated in all SEW components.

Cable length

- Maximum cable length: 3 m (5 m for shielded cables)

Baud rate

- The baud rate for RS485 communication is set using DIP switch S13 (on the front of MOVIDRIVE® B beneath the XT socket).

Baud rate	DIP switch S13
9.6 kBaud	ON
57.6 kBaud[1)]	OFF[1)]

1) Factory setting

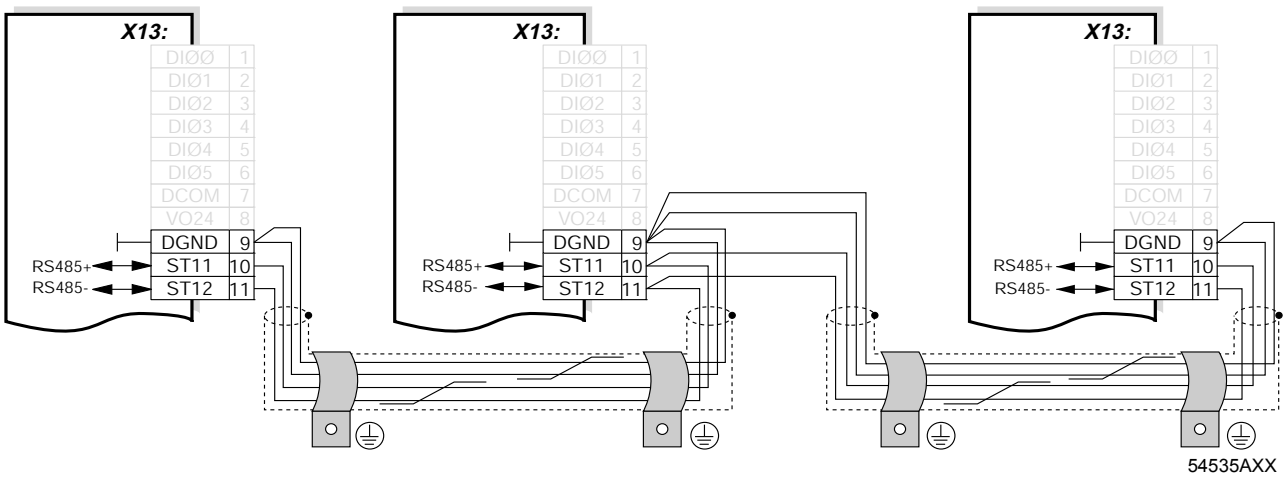
The set baud rate takes effect once the DIP switch position has been changed.

4.1.2 Connection using terminals X13:10 and X13:11

Another RS485 interface is provided via terminals X13:10 and X13:11. This RS485 interface can be used to interconnect several MOVIDRIVE® B units and connect the other options at the same time.

- Interface adapter UWS11A (RS232 signals to RS485 signals)
- DOP11B operator terminal
- Other SEW drives, such as MOVIMOT®
- Other interface adapters, such as COM server or other RS485 master devices

Wiring diagram of the RS485 interface (X13)



Electrical isolation

- The RS485 interface X13 is not electrically isolated. Do not connect more than 32 MOVIDRIVE® B units with one another,

Cable specification

- Use a 4 core twisted pair and shielded copper cable (data transmission cable with braided copper shield). The cable must meet the following specifications:
  - Cable cross section 0.25 - 0.75 mm^2 (AWG 23 - AWG 19)
  - Cable resistance 100 - 150 Ω at 1 MHz
  - Capacitance per unit length ≤ 40 pF/m at 1 kHz

Cable length

- The permitted total cable length is 200 m (656 ft).

Shielding

- Connect the shield to the electronics shield clamp on the inverter or higher-level controller and make sure it is connected over a wide area at both ends.

Baud rate

- The baud rate is set to 9.6 baud by default.

Terminating resistor

- Dynamic terminating resistors are installed. **Do not connect any external terminating resistors.**

	TIPS
	• When interconnecting the units, make sure that always only one master (e.g. DOP11B, engineering PC) is connected and active. • Operating several masters on an RS485 network with SEW drives is not permitted (see chapter "MOVILINK® via RS485"). • There must not be any potential displacement between the units connected via the RS485. This may affect the functionality of the units. Take suitable measures to avoid potential displacement, such as connecting the unit ground connectors using a separate cable.

4.1.3 Shielding and routing cables

Correct shielding of the bus cable attenuates electrical interference that can occur in industrial environments. The following measures ensure the best possible shielding:

- Manually tighten the mounting screws on the connectors, modules, and equipotential bonding conductors.
- Apply the shielding of the bus cable on both ends over a large area.
- Route signal and bus cables in separate cable ducts. Do not route them parallel to power cables (motor leads).
- Use metallic, grounded cable racks in industrial environments.
- Route the signal cable and the corresponding equipotential bonding close to each other using the shortest possible route.
- Avoid using plug connectors to extend bus cables.
- Route the bus cables closely along existing grounding surfaces.

	CAUTION
	In case of fluctuations in the ground potential, a compensating current may flow via the bilaterally connected shield that is also connected to the protective earth (PE). Make sure you supply adequate equipotential bonding according in accordance with relevant VDE regulations in such a case.

4.2 Configuration parameters of the serial interfaces

The following parameters are used to set communication via the **two** serial interfaces. The factory setting of the individual parameters is underlined.

Parameter			
No.	**Name**	**Setting**	**Meaning**
100	Setpoint source	TERMINALS RS485 FIELDBUS SBus	This parameter is used to set the setpoint source for the inverter.
101	Control signal source	TERMINALS RS485 FIELDBUS SBus	This parameter is used to set the source of the control signals for the inverter (CONTROLLER INHIBIT, ENABLE, CW, CCW, ...). Control via IPOS$^{plus®}$ and terminal is taken into account disregarding of P101.
750	Slave setpoint		The setpoint to be transferred to the master is set on the master. The "MASTER-SLAVE OFF" setting must be retained on the slave.
810	RS485 Address	0 ... 99	P810 is used to set the address by means of which communication can take place with MOVIDRIVE® via the serial interfaces. **Note:** MOVIDRIVE® B units are always set to the address 0 on delivery. To avoid problems during data exchange in serial communication with several inverters, we recommend that you do not use address 0.
811	RS485 group address	100 ... 199	P811 allows for grouping several MOVIDRIVE® B units in one group for communication via the serial interface. For example, the RS485 group address allows for sending setpoint selections to a group of MOVIDRIVE® B inverters simultaneously. Group address 100 means that the inverter is not assigned to a group.
812	RS485 Timeout interval	0 ... 650 s	P811 sets the monitoring time for data transmission via the serial interface. No monitoring of serial data transmission takes place when P812 is set to 0. Monitoring is activated with the first cyclical data exchange.
833	Response to RS485 timeout	RAPID STOP/WARN.	P833 programs the fault response that is triggered by the RS485 timeout monitoring.
870 **871** **872**	Setpoint description PO1 Setpoint description PO2 Setpoint description PO3	Factory set to: CONTROL WORD 1 SPEED NO FUNCTION	P870/P871/P872 define the content of the process output data words PO1/PO2/PO3.
873 **874** **875** **876**	Actual value description PI1 Actual value description PI2 Actual value description PI3 Enable PO data	Factory set to: STATUS WORD 1 SPEED NO FUNCTION ON	The content of process input data words PI1/PI2/PI3 is defined.

	TIP
	Refer to the MOVIDRIVE® MDX60B/61B system manual for a detailed description of the parameters.

4.3 MOVILINK® protocol via RS485 transmission method

4.3.1 Transmission method

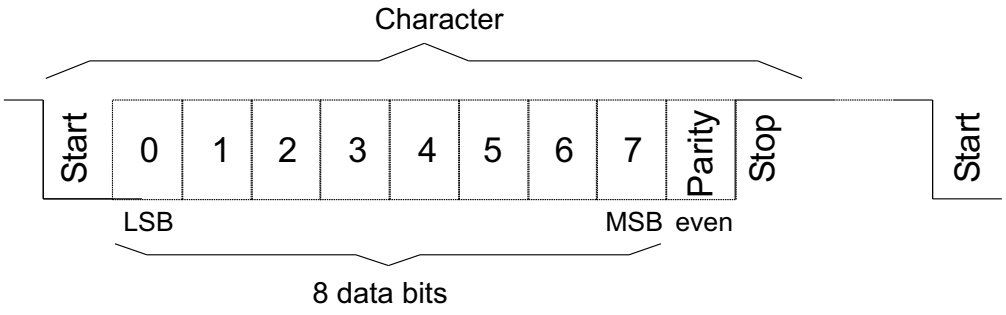
An asynchronous, serial transmission method is used which is supported by the UART modules common in digital technology. This means the MOVILINK® protocol can be implemented in nearly all controllers and master modules.

Characters

Each character in the MOVILINK® protocol consists of 11 bits and is structured as follows:

- 1 start bit
- 8 data bits
- 1 parity bit, completing for even parity (even parity)
- 1 stop bit

Each transmitted character begins with a start bit (always logical 0). The start bit is followed by 8 data bits and the parity bit. The parity bit is set in such a way that the number of logical ones in the data bits including the parity bit is even-numbered. The last bit of a character is a stop bit, which is always set to logical level 1. This level remains on the transmission medium until a new start bit signals the transmission of another character.



64767AEN

Transmission rate and transmission mechanisms

The transmission rate is 9600 baud or 57.6 kBaud (via XT only). The communications link is monitored by the master and the inverter itself. The master monitors the response delay time. The inverter monitors the reception of cyclic request telegrams of the master.

Response delay of the master

A response delay is usually programmed on the higher-level master system. The response delay is the interval between the time when the last character of the request telegram is sent (BCC) and the time when the response telegram is sent (SD2). The maximum permitted response delay interval is 50 ms. A transmission error has occurred if the inverter does not respond within this interval. Check the interface cable and the coding of the sent request telegram. Depending on the application, the request telegram should now be repeated and the next inverter be addressed.

Start delimiter (idle)

To interpret a character as start delimiter (02_{hex} or $1D_{hex}$), it must be preceeded by a pause of at least 3.44 ms.

Character delay

The interval between the time when a character of a telegram is sent must be shorter than the time preceeding the start delimiter (which means max. 3.43 ms). Else, the telegram is invalid.

RS485 timeout interval of the inverter

For MOVIDRIVE®, the maximum permitted time inverval between two cyclic request telegrams is set using parameter *P812 RS485 Timeout interval*. The system must receive a valid request telegram during this time period. Else, the inverter will trigger an RS485 timeout error and execute a defined error response.

After power on or a fault reset, MOVIDRIVE® is maintained in a safe condition until the first request telegram is received. When the inverter is enabled, "t" (= timeout active) appears on the 7-segment display and the enable is ineffective. Only when the first telegram is received, enable will take effect and the drive is set in motion.

If the inverter is controlled via RS485 interface (P100 "Setpoint source" = RS485 / P101 "Control signal source" = RS485) and a fault response with warning was programmed, the last received process data will be active after an RS485 timeout and reestablished communication.

NOTICE

If a timeout is not recognized, the drive will continue to move despite disconnected controller.

Possible consequences: Damage to the system.

Only one of the two RS485 must be used for timeout monitoring.

RS485 timeout is active for both RS485 interfaces. Therefore, timout monitoring for the second interface has no effect with plugged-in DBG60B keypad. The DBG60B keypad permanently sends request telegrams to the inverter and in this way triggers the timeout mechanism.

Processing request/response telegrams

The inverter only processes request telegrams that were received without errors and were correctly addressed. The following reception errors are detected:

- Parity error
- Character frame error
- Exceeded character delay of request telegrams
- Incorrect address
- Incorrect PDU type
- Incorrect BCC
- RS485 timeout (slave)
- Elapsed response time (master)

The inverter does not respond to incorrectly received request telegrams! These reception errors have to be evaluated in the master to ensure correct data transmission.

	TIPS
	If RS485 or RS232 communication is to be transmitted via gateways, COM server, or modem connections, make sure that not only the character (start bit, 8 data bits, 1 stop bit, even parity) is correct but also that start delimiter and character delay time are complied with: • Max. character delay of 3.43 ms between 2 characters of a telegram • Min. 3.44 pause before the start delimiter Else, the individual characters cannot be clearly assigned to the various telegrams.

4.3.2 Telegrams

Telegram transmission

Both cyclic and acyclic data exchange is used in drive engineering. Cyclic telegrams via the serial interface are mainly used for drive control in automation tasks. In this case, the master station has to ensure cyclic data exchange.

Cyclical data exchange

Cyclic data exchange is mainly used for controlling inverters via the serial interface. The master continuously sends telegrams with setpoints (request telegrams) to an inverter (slave) and expects a response telegram with actual values from the inverter. Once the request telegram is sent to an inverter, the master expects the response telegram within a defined time (response delay time). The inverter will only send a response telegram if it has correctly received a request telegram with its slave address. During cyclic data exchange, the inverter monitors data communication and triggers a timeout response if it has not received another request telegram from the master within a specified time.

MOVILINK® allows for performing acyclic service and diagnostic tasks during cyclic communication without having to change the telegram type.

Acyclical data exchange

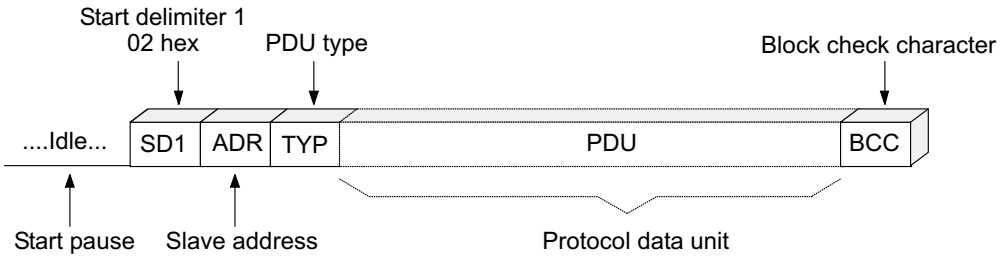
Acyclic data exchange is primarily used for startup and diagnostic purposes. In this case, the inverter does not monitor the communication connection. In acyclic mode, the master can send telegrams to the inverter at irregular intervals.

Telegram structure

Data exchange is carried out with only two telegram types. The master sends a request with data in the form of a request telegram to the inverter. The inverter responds with a response telegram. In case of word information (16 bit) within user data, always the high byte will be sent first followed by the low byte. In case of double-word information (32 bit), always the high byte will be sent first, followed by the low word. The protocol does not include coding of the user data. The content of user data is explained in detail in the "SEW unit profile" chapter.

Structure of the request telegram

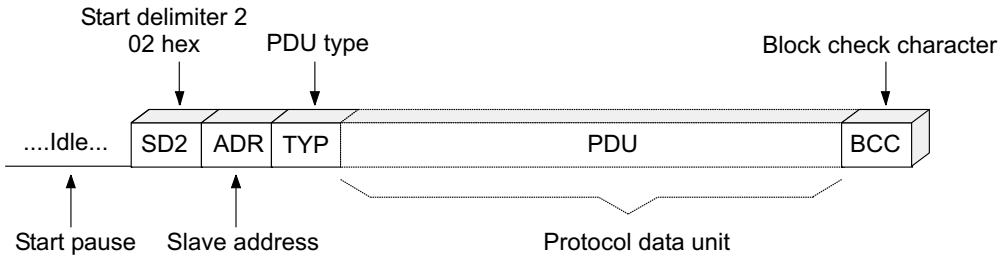
The following figure shows the structure of the request telegram, which the master sends to the inverter. Each telegram starts with an idle time on the bus, the so-called start pause, followed by a start delimiter. Different start characters are used to clearly distinguish between request and response telegrams. The request telegram begins with the start character SD1 = 02hex, followed by the slave address and PDU type.



01485BEN

Structure of the response telegram

The following figure shows the structure of the response telegram, which the inverter (slave) sends as response to a request from the master. Each response telegram begins with a start pause, followed by a start delimiter. To clearly distinguish request and response telegrams, the response telegram begins with the start character SD2 = 1Dhex, followed by the slave address and PDU type.



01487BEN

Start characters (SD1 / SD2)

The start character identifies the beginning and direction of data of a new telegram. The following table depicts the assignment of start character to direction of data.

SD1	02_{hex}	Request telegram	Master → inverter
SD2	$1D_{hex}$	Response telegram	Inverter → master

4.3.3 Addressing and transmission method

Address byte (ADR)

The address byte indicates the slave address independent of the direction of data. This means the ADR character in a request telegram specifies the address of the inverter that is to receive the request. In opposite direction, the master recognizes the inverter that has sent the response telegram. The master is not addressed because the system is generally a single-master system. The MOVILINK® protocol offers other addressing variants in addition to single addressing. The table below shows the address ranges and what they mean.

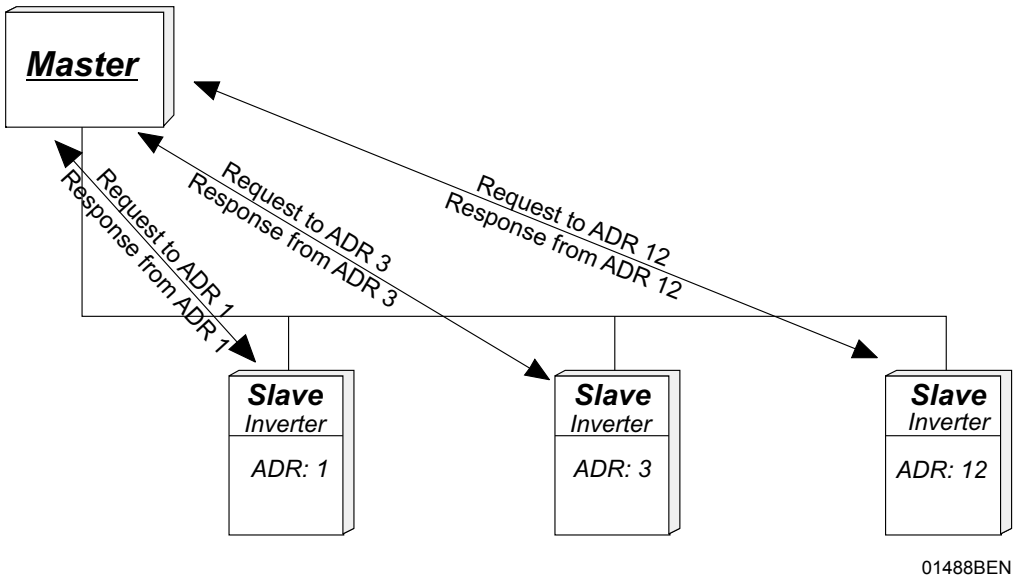
ADR	Meaning
0 - 99	Single addressing within the RS485 bus.
100 - 199	Group addressing (multicast) Special case group address 100: "Meaning not assigned to any group", i.e. not effective.
253	Local address: Only effective in conjunction with IPOS$^{plus®}$ as master and the MOVILINK® command. For unit internal communication
254	Universal address for point-to-point communication.
255	Broadcast address. No response is sent.

TIP

MOVIDRIVE® basically is a slave unit. However, master functions are also available using IPOS$^{plus®}$, the MOVILINK® command, and the master/slave function.

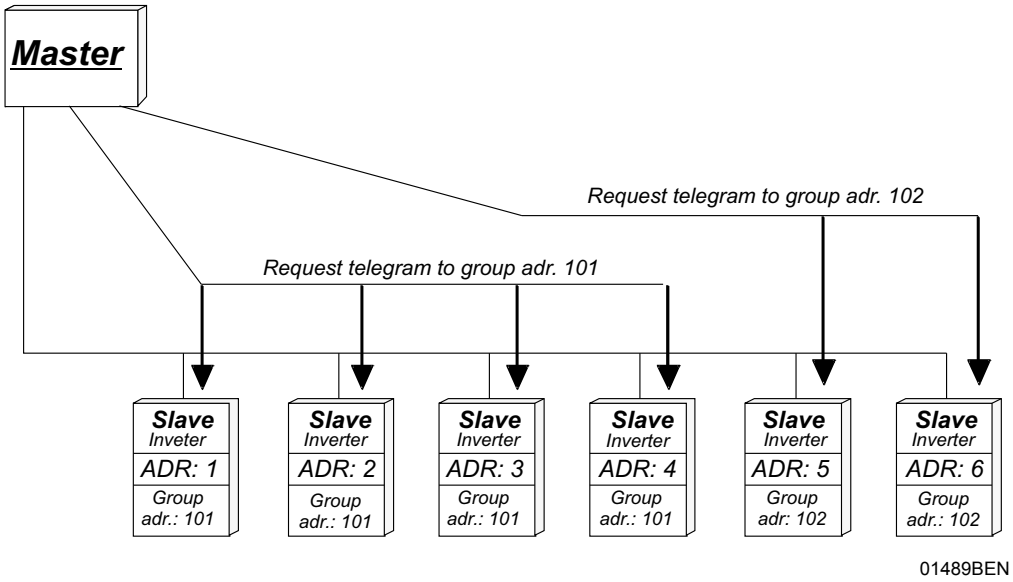
Single addressing

Every inverter can be directly addressed using addresses 0 - 99. The inverter responds to every request telegram from the master with a response telegram.



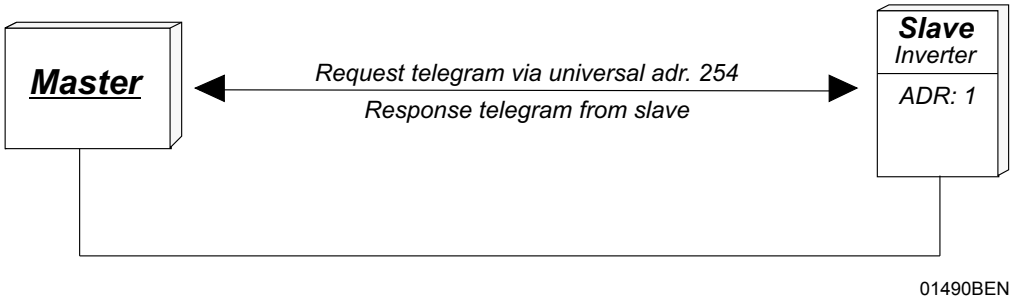
Group addressing (multicast)

Every inverter has a setable group address in addition to its individual address. In this way, the user can group various participants and address the individual participants of a group simultaneously using group addressing. With group addressing, the master does not receive a response telegram. This means that no data can be requested from the inverter. And there is no response when writing data. A maximum of 99 groups can be set up.



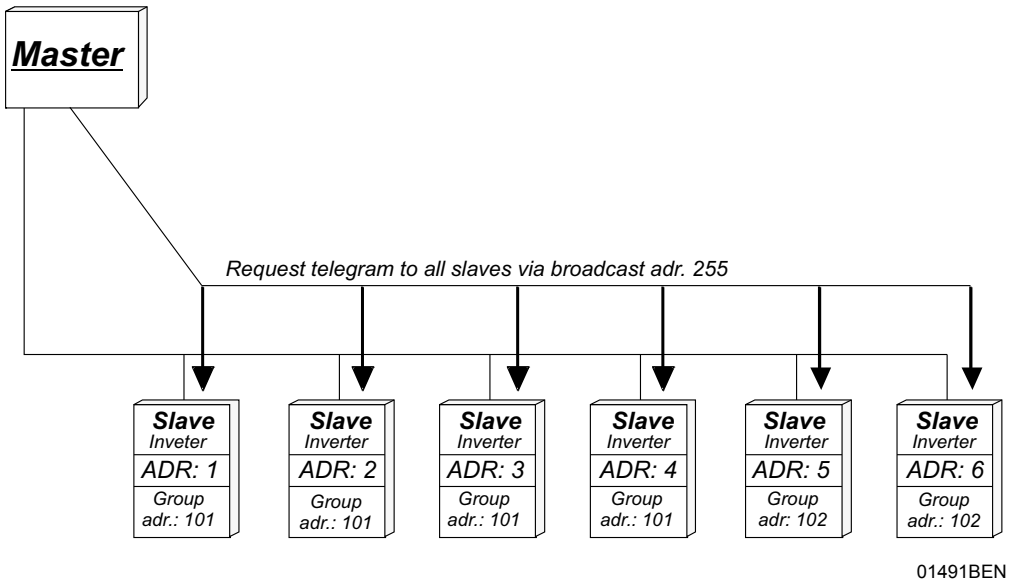
Universal addressing for point-to-point communication

Basically, every inverter can be addressed using universal address 254 independent of the set single address. The advantage of this variant is that point-to-point connections can be established without having to know the individual address. As each inverter of the group is addressed using this universal address, it must not be used for multi-point connections (e.g. RS485 bus). Else, data collisions would occur on the bus because every inverter would send a response telegram once it receives a request telegram.



Broadcast address

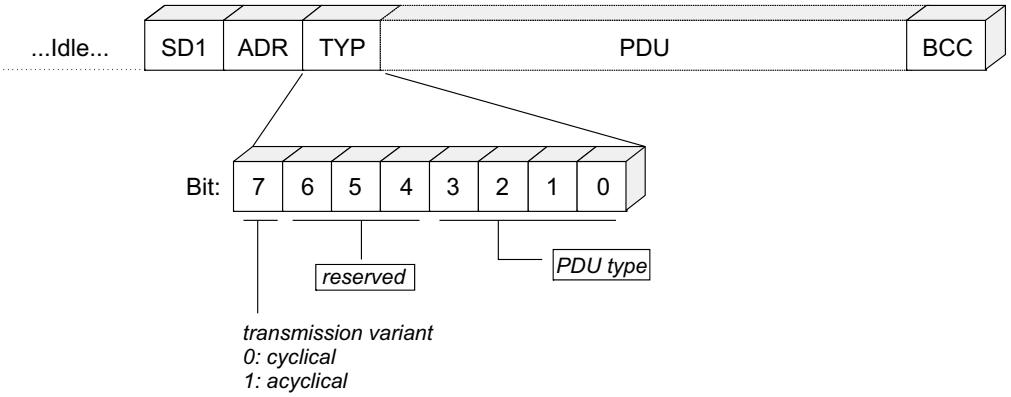
Broadcast address 255 can be used to address all inverter stations. The request telegram with broadcast address 255 sent by the master is received by all inverters but is not responded to. This means, this addressing method is mainly used to transmit setpoints. The master can send broadcast telegrams at a minimum interval of every 25 ms. Consequently, an idle time of at least 25 ms is mandatory between the last sent character of a request telegram (BCC) and the start of a new request telegram (BCC).



4.3.4 Structure and length of user data

PDU type (TYPE)

The TYPE byte describes the structure and length of subsequent user data (protocol data unit (PDU)). The figure below shows the structure of the TYPE byte.



01492BEN

Besides, bit 7 of the TYPE byte indicates whether the user data are transmitted cyclically or acyclically. A request telegram with cyclic transmission method signals the inverter that the data sent by the master is updated cyclically. Consequently, response monitoring can be triggered in the inverter. This means if the inverter does not receive a new cyclic request telegram within a timeout time, which can be set, a timeout response will be triggered.

The following tables show the PDU types for cyclic and acyclic transmission. The telegram length depends on the PDU type used and is calculated as follows:

Telegram length = PDU length + 4.

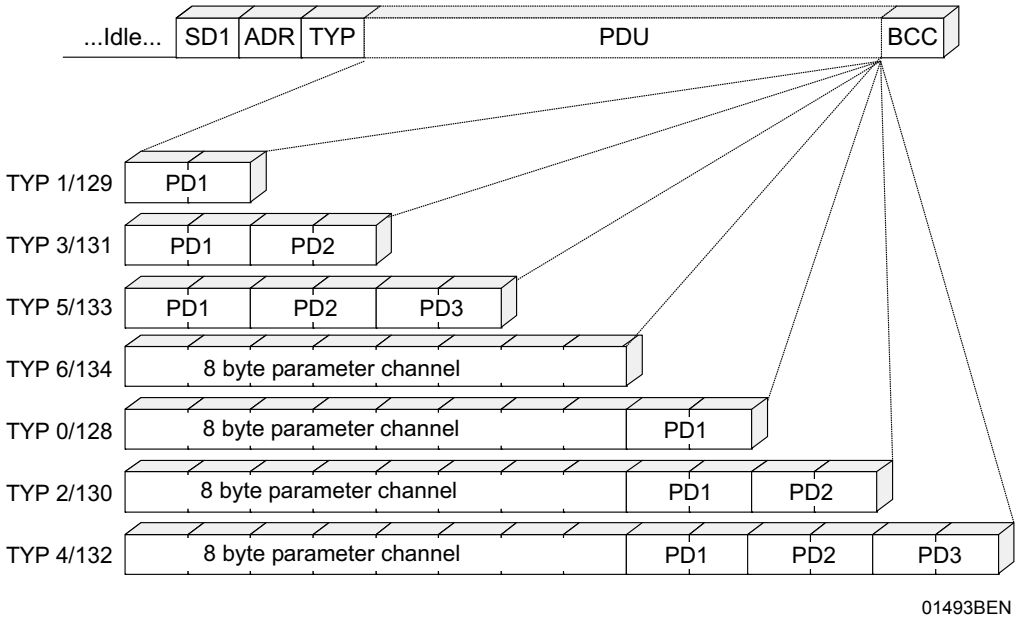
Transmission methods

The following tables show the PDU types for ACYCLIC and CYCLIC transmission methods.

TYPE byte				PDU name	Description	PDU length in bytes	Telegram length in bytes
Cyclic		Acyclic					
00_{hex}	0_{dec}	80_{hex}	128_{dec}	PARAM + 1PD	8 byte parameter channel + 1 process data word	10	14
01_{hex}	1_{dec}	81_{hex}	129_{dec}	1PD	1 process word	2	6
02_{hex}	2_{dec}	82_{hex}	130_{dec}	PARAM + 2PD	8 byte parameter channel + 2 process data words	12	16
03_{hex}	3_{dec}	83_{hex}	131_{dec}	2PD	2 process data words	4	8
04_{hex}	4_{dec}	84_{hex}	132_{dec}	PARAM + 3PD	8 byte parameter channel + 3 process data words	14	18
05_{hex}	5_{dec}	85_{hex}	133_{dec}	3PD	3 process data words	6	10
06_{hex}	6_{dec}	86_{hex}	134_{dec}	PARAM + 0PD	8 byte parameter channel without process data	8	12

The standard PDU types consist of the MOVILINK® parameter channel and a process data channel. For the coding of the parameter channel and process data, refer to chapter "SEW unit profile".

The following figure shows the structure of a request telegram with the standard PDU types. The associated response telegram has the same structure except for the SD2 start character.



Block check character

Transmission reliability

Transmission reliability with the MOVILINK® protocol is increased by a combination of character parity and block parity. The parity bit is set for each character of the telegram in such a way that the number of binary ones including the parity bit is even-numbered.

Block parity provides for additional reliability and means that a block check character (BCC = Block Check Character) is added to the telegram. Each bit of the block check character is set in such a way that the result is an even parity for all information bits of the same value of the telegram character. In programming, the block parity is implemented by EXORing all telegram characters. The result is entered in the BCC at the end of the message. The block check character itself is also ensured with the even character parity.

Creating the block check character

The following figure gives an example of how a block check character is created for a cyclical telegram of type PDU 5 with 3 process data words. The EXOR logic operation of the characters SD1 - $PD3_{low}$ results in the value 57_{hex} as the block check character BCC. This block check character will be sent as the last character of the telegram. The recipient checks the character parity after having received the individual characters. Next, the block check character is created from the received characters SD1 - $PD3_{low}$ according to the pattern described above. The telegram has been correctly transmitted if the calculated and received block check characters are identical and there is no character parity error. Any other result will be displayed as a transmission error.

	Stop	Parity								Start
SD1: 02 hex		1	0	0	0	0	0	0	1	0
						EXOR				
ADR: 01 hex		1	0	0	0	0	0	0	0	1
						EXOR				
TYP: 05 hex		0	0	0	0	0	0	1	0	1
						EXOR				
PD1 high: 00 hex		0	0	0	0	0	0	0	0	0
						EXOR				
PD1 low: 06 hex		0	0	0	0	0	0	1	1	0
						EXOR				
PD2 high: 3A hex		0	0	0	1	1	1	0	1	0
						EXOR				
PD2 low: 98 hex		1	1	0	0	1	1	0	0	0
						EXOR				
PD3 high: 01 hex		1	0	0	0	0	0	0	0	1
						EXOR				
PD3 low: F4 hex		1	1	1	1	1	0	1	0	0
calculated BCC: 57 hex		**1**	**0**	**1**	**0**	**1**	**0**	**1**	**1**	**1**

01494BEN

	TIP
	For a description of process data and the structure of the 8-byte parameter data channel, refer to chapter "SEW unit profile".

4.4 Other unit functions via RS485 interfaces

In addition to process and parameter data exchange between PC, keypad and MOVIDRIVE® B, the RS485 interfaces can also be used for the following functions:

- Master/slave operation
- IPOS$^{plus®}$
- Manual operation

4.4.1 Using RS485 interfaces for master/slave operation

The master/slave function shown in the figure below allows for implementing automatic functions such as speed synchronization, shared load and torque control (slave). The RS485 interface (X13:10/X13:11) or the system bus interface (CAN 1) can be used as communication link. *P100 Setpoint source = Master SBus* or *P100 Setpoint source = Master RS485* must be set on the slave. The process output data PO1 - PO3 (P870, P871, P872) are automatically set by the firmware. A programmable terminal function "Slave free run. "P60x binary inputs basic unit /P61x binary inputs option, it is possible to separate the slave from the master setpoint and switch to local control mode.

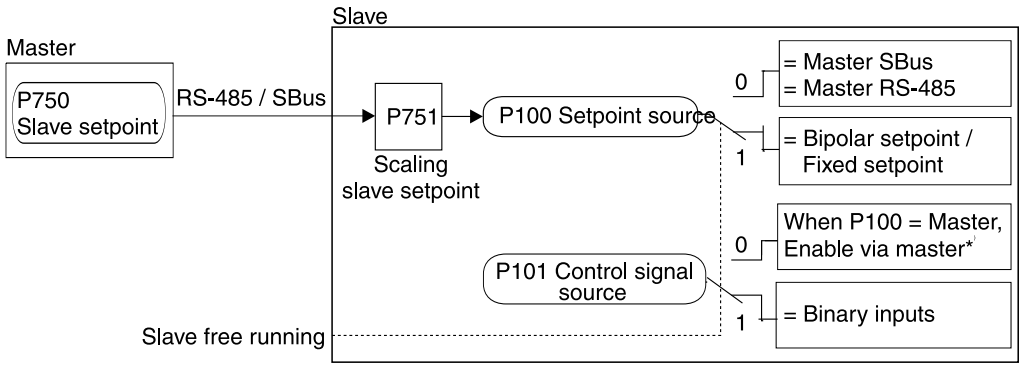
TIP

For the slave, the process data P87x are automatically assigned as follows:

– PO1 = Control word 1
– PO2 = Speed or current in M-control
– PO3 = IPOS PO data
– PI1 = Status word 1
– PI2 = Speed
– PI3 = IPOS PI data

PI3 and PO3 are not used. They are available in IPOS$^{plus®}$ as required.

If a fieldbus card is plugged in the slave, only the parameter channel is available for the output data. The master can read the automatically assigned process input data via fieldbus.



*) DIØØ "/Controller inhibit" and the programmed binary inputs Enable, CW and CCW must also receive a "1" signal.

	TIP
	P811 RS485 group address must be set to the same value for master and slave. For master/slave operation via RS485 interface, set *P811 RS485 Group address* to a value greater than 100. If you have made the setting in parameter *P750 slave setpoint* that slave setpoints are used via RS485, then MOVIDRIVE® can no longer respond to requests (process and parameter telegrams) from another RS485 master (P100/101 ≠ RS485) as slave via this RS485 interface.

Connection check

A connection check is always active for communication link via RS485 interface. *P812 RS485 timeout interval* is without function. The slave inverters must receive a valid RS485 telegram within the fixed time interval of t = 500 ms. If the time is exceeded, the slave drives will stop at the emergency stop ramp and error message F43 "RS485 timeout" will be issued.

	NOTICE
	If a timeout is not recognized, the drive will continue to move despite disconnected controller. Possible consequences: Damage to the system. Only one of the two RS485 interfaces must be used for timeout monitoring. As the RS485 timeout is active for both RS485 interfaces, the second interface is not monitored for timeout when the DBG60B keypad is installed. The DBG60B keypad permanently sends request telegrams to the inverter and in this way triggers the timeout mechanism.

Overview of functions of master/slave operation

Function	Master		Slave	
	P750 Slave setpoint	P700 operating mode 1	P100 Setpoint source	P700 Operating mode 1
Speed synchronization: • **Master controlled** • **Slave controlled**	SPEED (RS485+SBus1) SPEED (RS485) SPEED (SBus1)	VFC VFC & GROUP VFC & HOIST V/f CHARACTERIS-TICS V/f & DC BRAKING	MASTER SBus1 MASTER RS485:	VFC VFC & GROUP VFC & HOIST V/f CHARACTERIS-TICS V/f & DC BRAKING
Speed synchronization: • **Master speed controlled** • **Slave controlled**	SPEED (485+SBus1) SPEED (RS485) SPEED (SBus1)	VFC n-CONTROL VFC n-REG & ... CFC CFC/SERVO & IPOS CFC/SERVO & SYNC	MASTER SBus1 MASTER RS485:	VFC VFC & GROUP VFC & HOIST
Speed synchronization: • **Master speed controlled** • **Slave controlled** • **Drives do not have a rigid mechanical connection.**	SPEED (485+SBus1) SPEED (RS485) SPEED (SBus1)	VFC n-CONTROL VFC n-REG & ... CFC/SERVO CFC/SERVO & IPOS CFC/SERVO & SYNC	MASTER SBus1 MASTER RS485:	VFC n-CONTROL VFC n-CTRL & GROUP VFC n-CTRL & HOIST CFC SERVO
Speed synchronization: • **Master controlled** • **Slave controlled** • **Drives do not have a rigid mechanical connection.**	SPEED (485+SBus1) SPEED (RS485) SPEED (SBus1)	VFC VFC & GROUP VFC & HOIST	MASTER SBus1 MASTER RS485:	VFC n-CONTROL VFC n-CTRL & GROUP VFC n-CTRL & HOIST CFC SERVO
Load distribution: • **Master controlled** • **Slave controlled**	LOAD SHAR. (RS485+SBus1) LOAD SHAR. (RS485) LOAD SHAR. (SBus1)	VFC VFC & GROUP VFC & HOIST	MASTER SBus1 MASTER RS485:	VFC VFC & GROUP VFC & HOIST
Load distribution: • **Master speed controlled** • **Slave controlled**	LOAD SHAR. (RS485+SBus1) LOAD SHAR. (RS485) LOAD SHAR. (SBus1)	VFC n-CONTROL VFC n-REG & ... CFC/SERVO CFC/SERVO & IPOS CFC/SERVO & SYNC	MASTER SBus1 MASTER RS485:	VFC VFC & GROUP VFC & HOIST VFC & FLYING START
Load distribution: • **Master speed controlled** • **Slave controlled**	Not possible			
Load distribution: • **Master controlled** • **Slave controlled**	Not possible			
Torque control of the slave: • **Master speed controlled** • **Slave torque controlled**	TORQUE (RS485+SBus1) TORQUE (RS485) TORQUE (SBus1)	CFC/SERVO CFC/SERVO & IPOS CFC/SERVO & SYNC	MASTER SBus1 MASTER RS485:	CFC/SERVO & M-CTRL.

Speed synchronization

The actual speed of the master is transferred to the slave. Set the torque ratio for the slave inverter using *P751 Scaling slave setpoint*. Leave *P324 Slip compensation 1/P334 Slip compensation 2* of the slave at the value as the startup setting.

Example:

Parameter	Setting on the master	Setting on the slave
P100 Setpoint source	e.g. UNIPOL./FIXED SETP	MASTER SBus
P101 Control signal source	e.g. TERMINALS	Not in effect
P700 Operating mode 1	VFC n-CONTROL	VFC 1
P750 Slave setpoint	SPEED (SBus)	MASTER-SLAVE OFF
P751 Scaling slave setpoint	Not in effect	1 (then 1 : 1)
P810 RS485 Address	Set different values	
P811 RS485 group address	Not in effect	
P881 Address SBus 1	Set different values	
P882 SBus group address	Set the same value (0 - 63)	
P884 SBus baud rate	Set the same value (125, 250, 500 or 1000 kBaud)	

Load sharing

This function lets two inverters control the same load. The rotating field frequency of the master is transferred to the slave. It is assumed in this case that the shafts of the motors corresponding to the master and the slave are rigidly connected together. You are recommended to use the same motors with the same gear ratios, otherwise different delays may result during starting/stopping due to the pre-magnetizing time and the brake release/application time. *P751 Scaling slave setpoint* must be set to the value "1".

TIP

P324 Slip compensation 1 /P334 Slip compensation 2 of the slave must be set to 0.

Better behavior can be accomplished by setting the slave as follows:

- *P138 Ramp limitation VFC*: OFF
- *P115 Filter setpoint*: 0 s
- Ramps P130 / P131 /P132 / P133: 0 s
- *P301 Minimum speed 1 / P311 Minimum speed 2*: 0 min^{-1}

Example:

Parameter	Setting on the master	Setting on the slave
P100 Setpoint source	e.g. BIPOL./FIXED SETP	MASTER RS485
P101 Control signal source	e.g. TERMINALS	Not in effect
P324 Slip compensation 1	Do not change	0
P700 Operating mode 1	VFC 1	VFC 1
P750 Slave setpoint	LOAD SHAR. (RS485)	MASTER-SLAVE OFF
P751 Scaling slave setpoint	Not in effect	1 (then 1 : 1)
P810 RS485 Address	Set different values	
P811 RS485 group address	Set the same value (101 - 199)	
P881 Address SBus 1	Set different values	
P882 SBus group address	Not in effect	
P884 SBus baud rate	Not in effect	

Torque control

The slave inverter receives the torque setpoint of the master directly (the correcting variable of the speed controller). This enables high quality load sharing, for example. This load sharing setting should be preferred if the drive configuration permits it. Set the torque ratio using *P751 Scaling slave setpoint*.

Example:

Parameter	Setting on the master	Setting on the slave
P100 Setpoint source	e.g. UNIPOL./FIXED SETP	MASTER RS485
P101 Control signal source	e.g. TERMINALS	Not in effect
P700 Operating mode 1	CFC	CFC & M-CONTROL
P750 Slave setpoint	TORQUE (RS485)	MASTER-SLAVE OFF
P751 Scaling slave setpoint	Not in effect	1 (then 1 : 1)
P810 RS485 Address	Set different values	
P811 RS485 group address	Set the same value (101 - 199)	
P881 Address SBus 1	Set different values	
P882 SBus group address	Not in effect	
P884 SBus baud rate	Not in effect	

TIP

For more detailed information on master/slave operation, refer to the MOVIDRIVE® MDX60B/61B system manual.

4.4.2 Using the RS485 interfaces in IPOS$^{plus®}$

In the IPOS$^{plus®}$ positioning and sequence control system integrated in MOVIDRIVE® B, you can

- directly access the process data transmitted via RS485 using the commands *GETSYS PO data* and *SETSYS PI data*.
- access process and parameter data of other SEW drives connected via RS485 using the MOVILINK® command.

For this purpose, set the bus type to the value "1" for accessing the XT socket (master functionality), and to value "2" for access via X13.

TIP

For more information on IPOS$^{plus®}$ commands, refer to the "IPOS$^{plus®}$ Positioning and Sequence Control" manual.

4.4.3 Using RS485 interfaces for manual operation

Manual operation of MOVIDRIVE® B serves as startup assistance. It is used to move the drive for teach mode without active PLC (see chapter 8.9).

5 CAN Interfaces of MOVIDRIVE® B

As standard, MOVIDRIVE® B is equipped with two CAN interfaces (SBus):

- CAN 1 (SBus 1), connection via X12 on the MOVIDRIVE® B basic unit
- CAN 2 (SBus 2), connection via X30 and X31 on the DFC11B option card

Both CAN interfaces correspond to CAN specification 2.0 A and B but use only 11-bit identifiers (COB-ID 0 - 2047). The CAN interfaces can be configured and used independent of one another.

- For direct connection of control systems for process and parameter data transfer according to the CANopen or MOVILINK® profile
- For direct connection to SEW fieldbus gateways (e.g. DF..B/UOH11B) to allow operation on various bus and control systems.
- For interconnecting several MOVIDRIVE® B units for master/slave operation.
- For exactly synchronizing several MOVIDRIVE® B units (ISYNC).
- For connection to an engineering PC via PC-CAN interface or SEW fieldbus gateway.
- For sending and receiving user-defined data (CAN layer 2 communication) in IPOS$^{plus®}$.

Telegrams received from MOVIDRIVE® B via CAN interface are **not** passed on via the other CAN interface.

5.1 Connecting and installing CAN

5.1.1 Connecting the two CAN interfaces CAN 1 and CAN 2

Both CAN interfaces can be connected in a pin-compatible manner using a separable 3-pole terminal block. For the CAN 2 interface, the DFC11B option card provides an additional Sub-D9 plug (X30) connected in series according to CiA standard (CAN in automation).

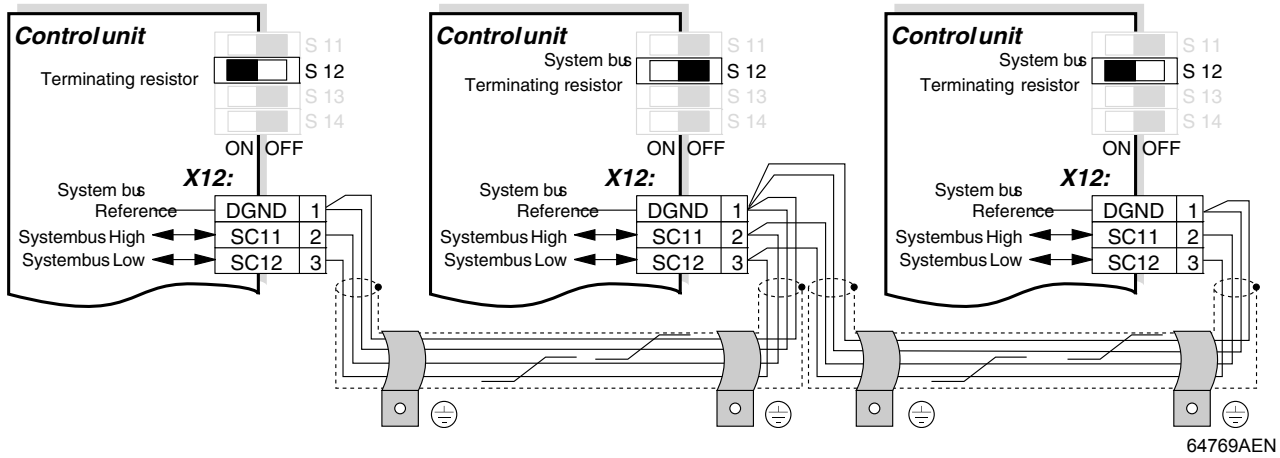
Terminal description of the CAN interfaces

Signal	Terminal	
	CAN 1: X12 (MDX B) CAN 2: X31 (DFC11B)	CAN 2: X30 (DFC11B)
DGND[1)]	1	3, 6
CAN High	2	7
CAN Low	3	2
N. C.	-	1, 4, 5, 8, 9

1) DGND of the CAN 2 interface is independent from DGND of the basic unit

- CAN 1 (SBus 1) at terminal X12 in the MOVIDRIVE® B basic unit is not electrically isolated.
- CAN 2 (SBus 2) is available electrically isolated via the DFC11B option (terminals X30, X31).

Wiring diagram



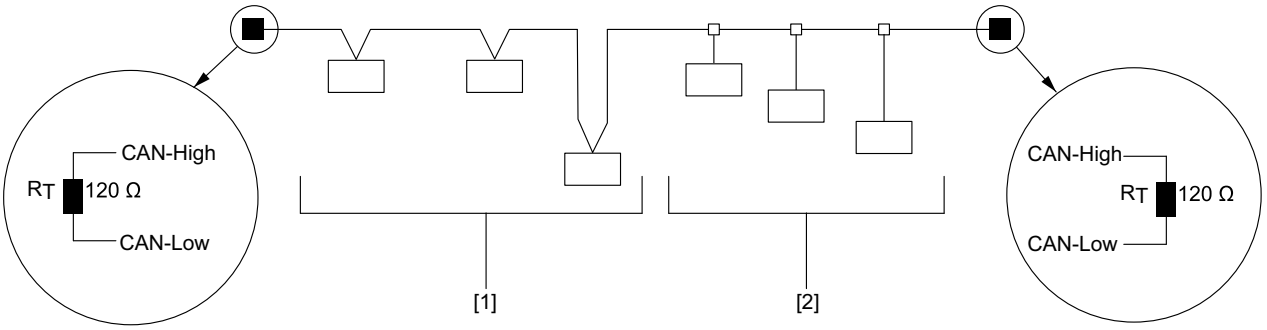
64769AEN

An 120 ohm terminating resistor can be connected for both CAN interfaces via DIP switch.

- CAN 1: DIP switch S12 on the MOVIDRIVE® B basic unit
- CAN 2: DIP switch "R" on the DFC11B option

CAN network

The CAN network (see figure below) should always have a linear bus structure without [1] (or only with very short) stub lines [2]. The network must have exactly one terminating resistor R_T = 120 ohm installed on both ends of the bus.



64357AXX

MOVIDRIVE® B units come equipped with CAN transceivers with a fan-out of more than 100:1. This means you can connect 100 units via CAN network without having to take special measures.

Cable length

The permitted total cable length depends on the baud rate setting (P884/P894):

- 125 kBaud → 500 m (1640 ft)
- 250 kBaud → 250 m (820 ft)
- 500 kBaud → 100 m (328 ft)
- 1000 kBaud → 25 m (82 ft)

	TIPS
	• Do not operate both CAN interfaces with 1000 kBaud to limit the interrupt load for MOVIDRIVE® B. If a CAN interface is operated wtih 1000, the other CAN interface must be set to 125 kBaud. • The option cards DCS..B use the CAN 2 interface. The baud rate of the CAN 2 interface is set to 500 kBaud by default. • MOVIDRIVE® *compact* MCH4_A units must not be combined with other MOVIDRIVE® B units in the CAN network if the baud rate is set to 1000 kBaud.

	NOTICE
	Swapping the CAN connections. CAN transceiver in MOVIDRIVE® B or on DFC11B option might be destroyed. Make sure the CAN interfaces are connected correctly.

5.1.2 Shielding and routing cables

Correct shielding of the bus cable attenuates electrical interference that can occur in industrial environments. The following measures ensure the best possible shielding:

- Manually tighten the mounting screws on the connectors, modules, and equipotential bonding conductors.
- Apply the shielding of the bus cable on both ends over a large area.
- Route signal and bus cables in separate cable ducts. Do not route them parallel to power cables (motor leads).
- Use metallic, grounded cable racks in industrial environments.
- Route the signal cable and the corresponding equipotential bonding close to each other using the shortest possible route.
- Avoid using plug connectors to extend bus cables.
- Route the bus cables closely along existing grounding surfaces.

	CAUTION
	In case of fluctuations in the ground potential, a compensating current may flow via the bilaterally connected shield that is also connected to the protective earth (PE). Make sure you supply adequate equipotential bonding in accordance with relevant VDE regulations in such a case.

CAN cable specification

- Use a 2 x 2-core twisted and shielded copper cable (data transmission cable with braided copper shield). The cable must meet the following specifications:
  - Cable cross section 0.25 - 0.75 mm^2 (AWG 23 - AWG 19)
  - Cable resistance 120 Ω at 1 MHz
  - Capacitance per unit length ≤ 40 pF/m at 1 kHz

  Suitable cables are e.g. CAN bus or DeviceNet cables.

Shield contact

- Connect the shield on both sides to the electronics shield clamp of MOVIDRIVE® B over a large area. With a 2-core cable, additionally connect the shield ends to GND.

NOTICE

There must not be any difference of potential between the units which are connected using CAN 1 (Sbus 1). This may affect the functionality of the units.

Take suitable measures to avoid potential displacement, such as connecting the unit ground connectors using a separate cable.

5.2 Configuration parameters of the CAN interfaces

Set the following parameters to enable communication via CAN interfaces:

Parameter			
No.	**Name**	**Setting**	**Meaning**
100	Setpoint source	SBus 1/2	The inverter obtains its setpoint from the SBus.
101	Control signal source	SBus 1/2	The inverter receives its control commands from the SBus.
836 **837**	Response to SBus timeout 1/2	Factory set to: EMERG.STOP/FAULT	The error response is programmed that is triggered by system bus timeout monitoring.
870 **871** **872**	Setpoint description PO1 Setpoint description PO2 Setpoint description PO3	Factory set to: CONTROL WORD 1 SPEED NO FUNCTION	The content of process input data words PO1/PO2/PO3 is defined. This is necessary so MOVIDRIVE® can allocate the appropriate setpoints.
873 **874** **875** **876**	Actual value description PI1 Actual value description PI2 Actual value description PI3 Enable PO data	Factory set to: STATUS WORD 1 SPEED NO FUNCTION ON	The content of process input data words PI1/PI2/PI3 is defined. This is necessary so MOVIDRIVE® 07 can allocate the appropriate actual values. The process data must be enabled to have the unit apply the setpoints.
881 **882**	SBus address 1/2	0 - 63	Setting of the SBus address used for exchanging parameter and process data.
882 **892**	SBus group address 1/2	0 - 63	Setting for the SBus group address used for receiving group parameter and group process data.
883 **893**	SBus timeout interval 1/2	0 - 650 s	Monitoring time for data transmission via SBus. MOVIDRIVE® performs the fault response set in P836 if there is no data traffic on the SBus within this time. Data transmitted via SBus is not monitored when P815 is set to 0 or 650 s.
884 **894**	SBus baud rate 1/2	125/250/500/1000 kBaud	The transmission speed of the SBus is set.
885 **895**	SBus synchronization ID 1/2	0 - 2047	P817 is used for setting the identifier (address) of the synchronization message in the inverter for the SBus. Make sure there is no overlap between the identifiers for the process data or parameter data telegrams.
887	External controller synchronization	On/off	By default, the time base of MOVIDRIVE® units is slightly smaller than 1 ms. For synchronization with an external controller, the time base can be set to exactly 1 ms.
888	Synchronization time	1 - 5 - 10 ms	This parameter is used to specify the time interval for synchronous data transfer (see P885/P895).
889/ **899**	Parameter channel 2	Yes / No	Parameter channel 2 is only needed with the MOVI-PLC® option. In this case, master/slave operation via SBus is not possible.

TIP

Refer to the MOVIDRIVE® MDX60B/61B system manual for a detailed description of the parameters.

5.3 MOVILINK® profile via CAN

The MOVILINK® profile via CAN (SBus) is an application profile from SEW-EURODRIVE specifically adjusted to SEW inverters. This chapter serves to support you with diagnostics and creating own applications.

5.3.1 Telegrams

Different telegram types were defined for communication with a master (e.g. controller). These types of telegrams can be divided into three categories:

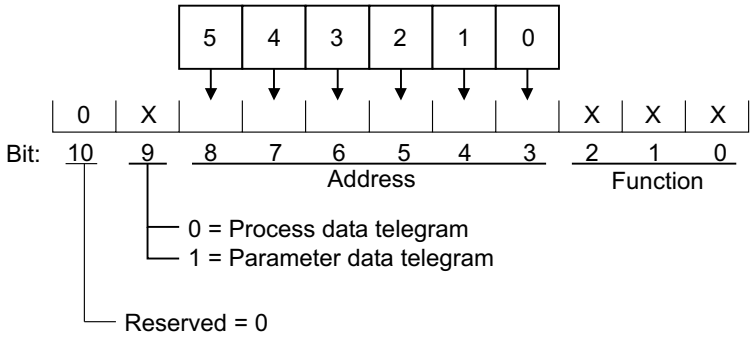
- Synchronization telegrams
- Process data telegrams
- Parameter telegrams

CAN bus identifier

On the SBus, it is necessary to differentiate between these various types of telegrams by means of identifiers (ID or COB-ID (Communication Object Identifier)). Therefore, the identifier of a CAN telegram is generated from the telegram type and the address set using parameters P881/P891 (SBus address) or P882/P892 (SBus group address).

The CAN bus identifier consists of 11 bits because only standard identifiers are used. The 11 bits of the identifier are divided into 3 groups:

- Function (bit 0 - 2)
- Address (bits 3 - 8)
- Process data/parameter data changeover (bit 9)



02250BEN

Bit 9 is used to distinguish between process and parameter data telegrams. The address of parameter and process data telegrams includes the SBus address (P881/P891) of the unit that is addressed by a request; the address of group parameter and group process data telegrams includes the SBus group address (P882/P892).

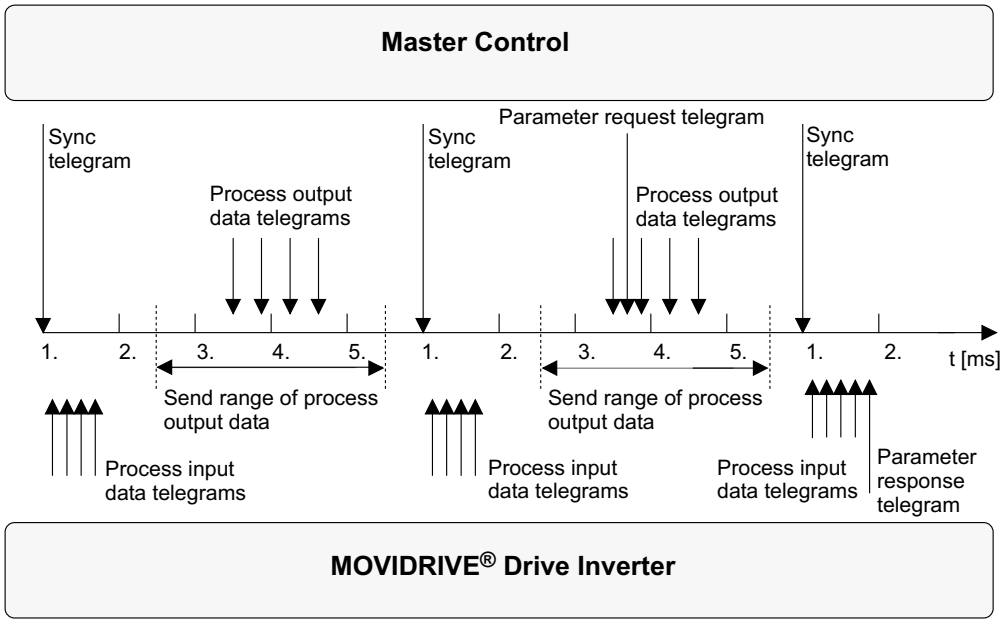
Creating the identifiers

The following table shows the relationship between the type of telegram and the address when creating the identifiers for SBus MOVILINK® telegrams:

Identifier	Telegram type
8 × SBus address + 3	Process output data telegram (PO)
8 × SBus address + 4	Process input data telegram (PI)
8 × SBus address + 5	Synchronizable process output data telegram (PO sync)
8 × SBus group address + 6	Group process output data telegram (GPO)
8 × SBus address + 512 + 3	Parameter request telegram (channel 1)
8 × SBus address + 512 + 4	Parameter response telegram (channel 1)
8 × SBus address + 512 + 5	Parameter response telegram (channel 2)
8 × SBus group address + 512 + 6	Group parameter request telegram
8 × SBus address + 512 + 7	Parameter response telegram (channel 2)
See P885/P895	Sync telegrams

Synchronization telegram

A fixed time base of 5 milliseconds can be defined for the transmission of process data and parameter data. In the first millisecond of a cycle, the master controller must send a synchronization telegram to the connected drive inverters.



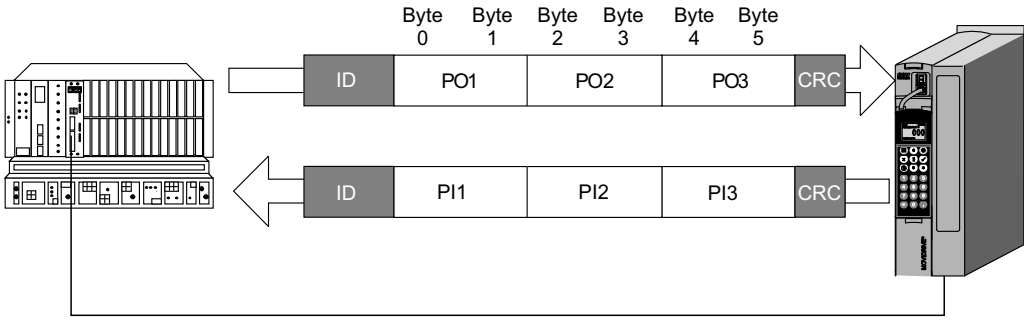
01020BEN

The synchronization message is a broadcast message. This means all drive inverters receive this message. The identifier of this message is set to zero at the factory. You can choose any value between 0 and 2047 but make sure there is no overlapping with the identifiers of the process or parameter data telegrams.

Process data telegrams for 3 process data words

The process data telegrams comprise a process output data telegram and a process input data telegram. The process output data telegram is sent from the master to a slave and contains the setpoints for the slave. The process input data telegram is sent from the slave to the master and contains actual values of the slave.

A telegram with 6 bytes of user data is required for transmitting 3 process data words. The next page describes the transmission of up to 10 process data words.



64299AXX

The process data at sent at certain points of time during the fixed time base of 5 milliseconds. A distinction is made between synchronous and asynchronous process data.

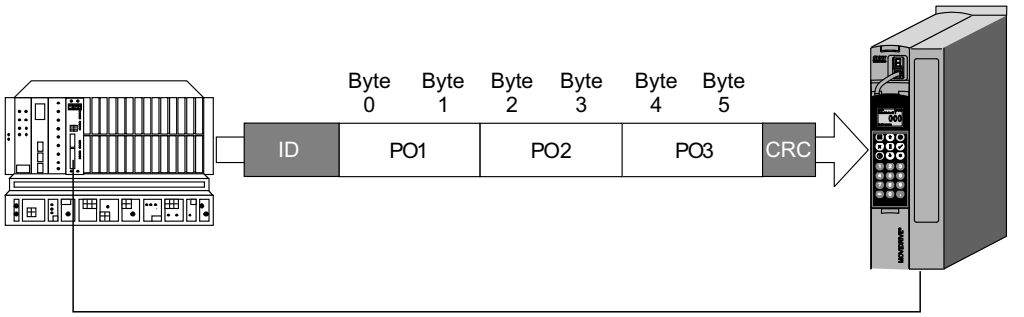
The synchronous process data are sent within the time interval at certain points of time. The master controller must send the process output data at the earliest 500 ms after the second millisecond and at the latest 500 ms before the first millisecond. MOVIDRIVE® sends the process input data as response in the first millisecond.

Asynchronous process data are not sent during the time interval. The master controller sends process output data at any time. MOVIDRIVE® responds with a process input data telegram within a maximum of 1 millisecond.

The coding of process input and output data words is described in chapter "SEW unit profile".

Group process data telegram for 3 process data words

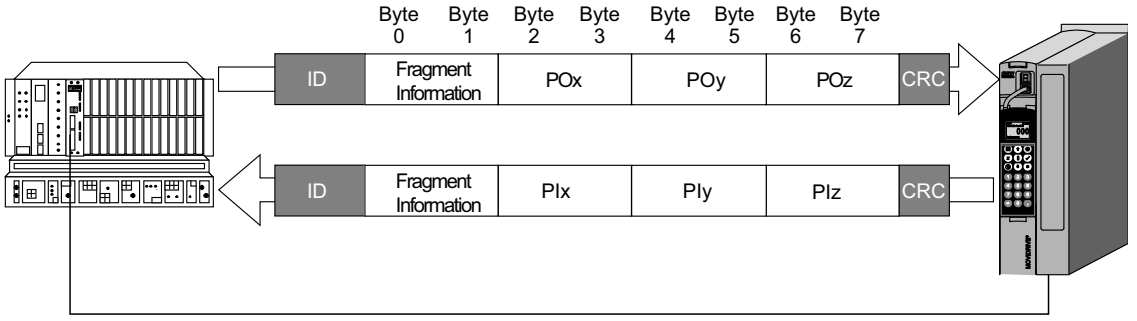
The master sends the group process data telegram to one or more slaves with the same SBus group address. It has the same structure as the process output data telegram. This telegram can be used for sending the same setpoint values to several slaves which share the same SBus group address. The slaves do not respond to the telegram.



64300AXX

Process data telegrams for up to 10 process data words

The transmission of more than 3 process data words is carried out across an "unacknowlegded fragmentation channel". The telegrams always have a length of 8 bytes even if fewer data are required for data transmission. The reason is to prevent incorrect interpretation of process data. Telegrams with a length of 8 bytes are always fragmented.



64302AXX

The CAN telegrams are structured as follows:

Byte no.	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Fragmentation type		Fragmentation count					
1	Len (Byte)							
2	IO Data							
3	IO Data							
4	IO Data							
5	IO Data							
6	IO Data							
7	IO Data							

"Len (Byte)" is the number of data bytes to be transmitted in total (e.g. 10 PD = 20 bytes).

Fragmentation type:

Fragmentation type	Meaning
0	First fragment
1	Middle fragment
2	Last fragment

"Fragmentation count" begins with "0" and is increased by one with each fragment.

Sequence

First, the master transmits all fragments to the slave. Once the slave has accepted transmission, it returns as many fragmented process data as it has received.

The following 4 telegrams are sent when transmitting 10 process data words:

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
	Fragment	Length	IO data		IO data		IO data	
Message 1	0x00	20dec	PD word 1		PD word 2		PD word 3	
Message 2	0x41		PD word 4		PD word 5		PD word 6	
Message 3	0x42		PD word 7		PD word 8		PD word 10	
Message 4	0x83		PD word 10		Reserved		Reserved	

Parameter telegrams

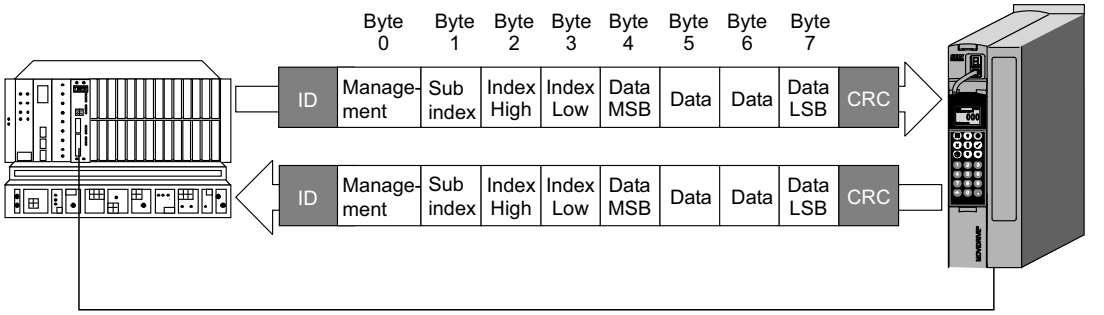
Parameter telegrams (see figure below) consist of a parameter request telegram and a parameter response telegram.

The master sends the parameter request telegram to read or write a parameter value. Its structure is as follows:

- Management byte
- Subindex
- Index high byte
- Index low byte
- 4 data bytes

The management byte specifies the service to be performed. The index specifies the parameter for which the service is to be performed. The 4 data bytes contain the number value to be read or written (see "Fieldbus Unit Profile" manual).

The slave sends the parameter response telegram in response to the parameter request telegram from the master. Request and response telegrams have the same structure.

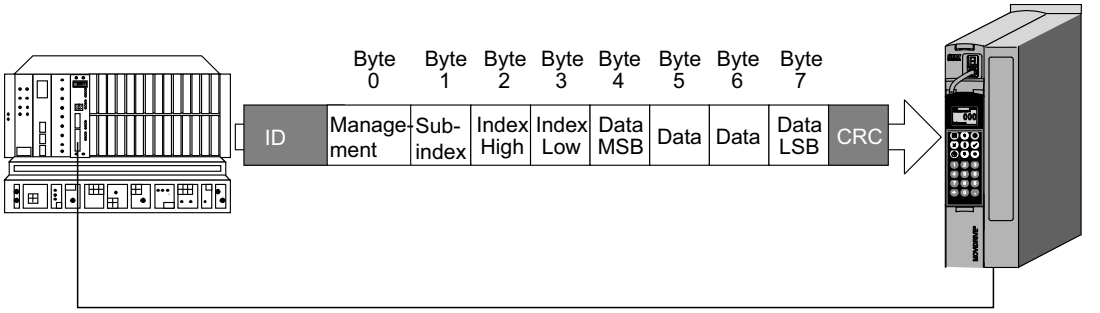


64303AEN

Parameter telegrams can also be synchronous or asynchronous telegrams. Synchronous parameter telegrams are responded to within 5 milliseconds. The response telegram is sent in the first millisecond. Asynchronous parameter telegrams are responded to independent of any time interval.

Group parameter telegram

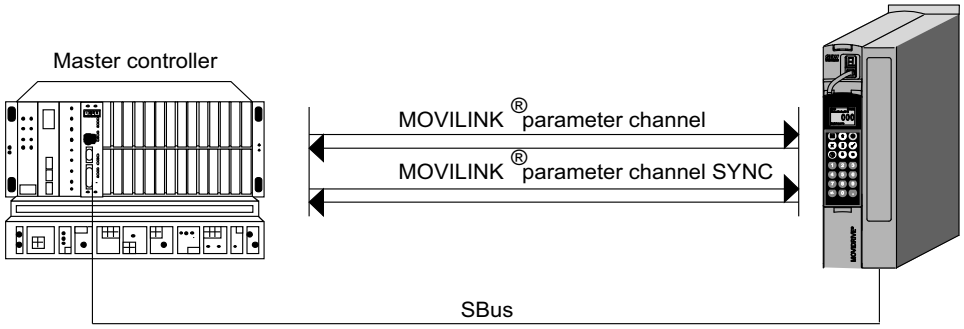
The master sends the group parameter telegram to one or more slaves with the same SBus group address. It has the same structure as the process request telegram. You can only write parameters to the slave units with this telegram. The slaves do not respond to the telegram.



64304AEN

5.3.2 Parameter setting via CAN (SBus MOVILINK®)

With the SBus, MOVIDRIVE® inverters support the "MOVILINK® parameter channel" and "MOVILINK® parameter channel SYNC".



63801AEN

	TIP
	For a detailed description of the parameter channel structure, refer to the "SEW unit profile" chapter.

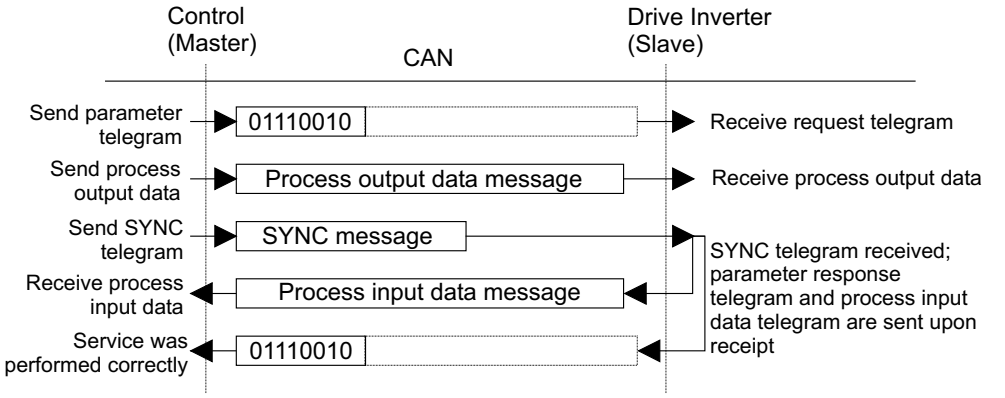
In conjunction with the CAN bus, it is important to distinguish between synchronized and non-synchronized parameters:

- With **non-synchronized** parameter telegrams (handshake bit = 0), the service acknowledgement is independent of the sync telegram.
- With **synchronized** parameter telegrams (handshake bit = 0), the service acknowledgement is sent in the first millisecond upon reception of the sync telegram.

Parameter setting procedure with CAN

Taking the example of the WRITE SYNC service is intended to represent a process of setting parameters between the controller and drive inverter via SBus-MOVILINK® (see figure below). To simplify the process, the figure 16 only shows the management byte of the parameter telegram.

While the controller prepares the parameter telegram for the WRITE SYNC service, the drive inverter receives SYNC telegrams and also receives and returns process data telegrams. The service is activated once the parameter request telegram has been received. The drive inverter then interprets the parameter telegram and processes the WRITE SYNC service. At the same time, it responds to all the process data telegrams. The parameter response telegram is not sent until the SYNC telegram has been received.



5.4 CANopen profile via CAN

CANopen communication is implemented according to the specification DS301 version 4.02 of CAN in automation (siehe www.can-cia.de). A specific unit profile, such as DS 402, is not implemented.

An EDS file for configuring CANopen master systems is available for download at www.sew-eurodrive.de under "Documentation/Software".

The CANopen profile provides the following COB-ID (Communication Object Identifier) and functions:

Type	COB ID	Function and properties in MOVIDRIVE®
NMT	000hex	Network management
Sync	080hex	Synchronous message with dynamically configurable COB ID
Emcy	0890hex + node no.	Emergency message with dynamically configurable COB ID
Tx-PDO1 Rx-PDO1	180hex + node no. 200hex + node no.	For 10 process input data words (PI) that are mapped as required in the TX-PDOs. For 10 process output data words (PO) that are mapped as required in the RX-PDOs. Various transmission modes (synchronous, asynchronous, event) Dynamically configurable length of PDOs
Tx-PDO2 Rx-PDO2	280hex + node no. 300hex + node no.	
Tx-PDO3 Rx-PDO3	380hex + node no. 400hex + node no.	
SDO	580hex + node no. 600hex + node no.	SDO channel for parameter data exchange with the CANopen master
Guard-ing/heart-beat	700hex + node no.	Guarding and Heartbeat are supported: • Heartbeat producer • Heartbeat consumer (single) • Lifetime protocol (guarding)

MOVIDRIVE® B saves all parameters to the non-volatile memory card; that is, the settings of the CANopen communication range (CANopen index 0x1000 - 0x1FFF) are also reset to the last available value once the system is restarted.

The CANopen communication range can be reset to its original settings using the NMT service "Reset_Communication". If you want to reset all parameters of MOVIDRIVE® B, you can use parameter P802 or write value "1" to index 8594.

	TIP
	You can check the status of the CANopen slave (NMT status, guarding, identifier, and mapping of PDOs) of MOVIDRIVE® B in the MOVITOOLS® MotionStudio context menu of MOVIDRIVE® under [Diagnostics] / [CANopen configuration].

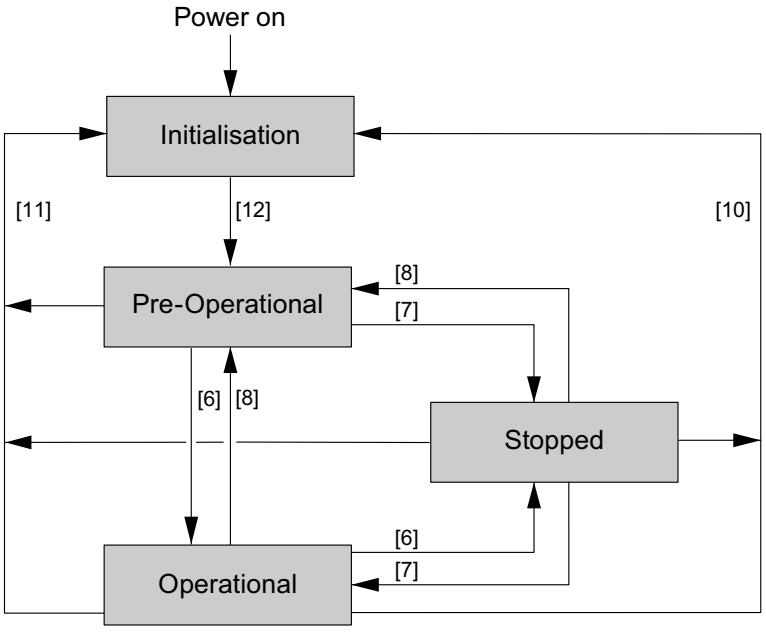
5.4.1 Configuring the CANopen interface of MDX B and network management (NMT)

Baud rate and CANopen slave address

Parameters P884/P894 are used to set the CAN baud rate. The baud rate can only be set to 1 Mbaud when the baud rate of the other Sbus is 125 kBaud. The CANopen slave address can be set using parameters P886/P896. If the CAN baud rate or CANopen slave address is changed, CAN communication will restart.

Unit states and NMT services

MOVIDRIVE® B untersupports the so-called "Minimum Capability Device". This means the following states are supported: "pre-operational", "operational" and "stopped" (see figure below).



64336AXX

The unit states can be changed at any time using NMT services. Possible commands:

- Node_Start indication [6]
- Node_Stop indication [7]
- Enter_Pre-Operational_State indication [8]
- Reset_Node indication [10]

  This command is used to reset the entire inverter. This means an inverter fault is acknowledged and the default settings in the object directory and the CANopen communication range (0x1000 to 0x1FFF) become active.
- Reset_Communication [11]

  This command causes the communication parameters in the object directory to be reset (index 0x1000 to 0x1FFF).
- Initialization finished [12]

  The node automatically changes to "pre-operational" and sends a guarding telegram (700hex + slave ID) as "boot-up" message.

The CAN telegrams are structured as follows:

NMT service	COB ID	Byte 1	Byte 2
Node_Start	0x0000	0x01	Node ID
Node_Stop		0x02	
Enter_Pre-Operational_State		0x80	
Reset_Node		0x81	
Reset_Communication		0x82	

The node ID corresponds with the address set in parameter P886/896. The value "0" is permitted for the node ID. In this case, all CANopen devices will be addressed.

NMT services are not acknowledged by the slave.

MOVIDRIVE® B supports the so-called "Minimum Capability Device", i.e. the following states are supported:

- Pre-operational

  In "pre-operational" state, the device can only communicate via SDOs.
- Operational

  In "operational" state, PDOs and SDOs can be exchanged.
- Stopped

  In "stopped" state, neither PDOs nor SDOs can be exchanged.

When switching MOVIDRIVE® B back on, the CANopen interfaces are always automatically in "pre-operational" state. After switching back on, they go to "operational" state if index 0x27FA subindex 1 (SBus 1) or subindex 2 (SBus 2) is 1.

If MOVIDRIVE® B remains in "booting" state after switching it on again, no boot-up message could be sent. This means no other CAN unit can be accessed via SBus. Check the connection and parameter setting of all unit on CAN.

5.4.2 Process data exchange

MOVIDRIVE® B can send up to 10 process input data words (PI data) and receive up to 10 process output data words via CANopen. 10 process data words can be assigned different PDOs using CANopen mapping. The PDO transmission modes can be used to optimize the bus load or define which process data words are to be sent and how often the transmission is to take place. Process data words 1 to 3 can be directly processed by the inverter. Process data words 4 to 10 can only be processed by IPOS$^{plus®}$ programs. Parameters P870 - P875 are used to define the function of process data words 1 - 3.

After communication has been reset, the following default settings apply:

- RX-PDO1 with PO1, PO2, PO3
- RX-PDO2 with PO4, PO5, PO6
- RX-PDO3 with PO7, PO8, PO9, PO10
- TX-PDO1 with PI1, PI2, PI3
- TX-PDO2 with PI4, PI5, PI6
- TX-PDO3 with PI7, PI8, PI9, PI10
- All RX-PDOs and TX-PDOs operate in the synchronous transmission mode.
- TX-PDO2 and TX-PDO3 are disabled and can be activated if required.

Configuring the COB IDs

After the communication/application has been reset, all COB IDs are reinitialized in accordance with the DS301 CANopen standard depending on the CANopen slave address (P886/P896).

	Index of the COB ID (hex)	Subindex of the COB ID (hex)	COB ID after communication reset
RX-PDO1	1400	1	200hex + slave address
RX-PDO2	1401		300hex + slave address
RX-PDO3	1402		400hex + slave address
TX-PDO1	1800		40000180hex + slave address
TX-PDO2	1801		C0000280hex + slave address
TX-PDO3	1802		C0000380hex + slave address

The COB ID (32 bit value) is set up according to the following schema:

Bit	Value	Meaning
31	0	PDO is active
	1	PDO is inactive
30	0	RTR may be used for this PDO
	1	RTR may not be used for this PDO
29	0	11-bit CAN ID
	1	29-bit CAN ID
28 - 11	0	Must always be "0" if bit 29 is "0"
	X	Bits 28 - 11 of the 29-bit COB ID if bit 29 is "1"
10 - 0	X	Bits 10 - 0 of the COB ID

	TIPS
	Observe the following notes: • With MOVIDRIVE® B, 29-bit COB IDs and RTR telegrams are not permitted for requesting TxPDOs. Therefore, bits 29 and 28 - 11 of the PDO COB IDs must always be set to "0". MOVIDRIVE® B automatically sets bit 30 to "1". • COB IDs can only be changed in "pre-operational" unit state. • New values are only accepted for the COB ID when it is an 11-bit ID (that is, bit 29 must not be set) and if the ID has not already been assigned to another PDO or the emergency object (see chapter "Emergency object).

Example

The COB ID of RX-PDO3 is to be changed from 401_{hex} to 403_{hex}. The COB ID of RX-PDO3 is stored in index 1402_{hex}, subindex 1.

- Send NMT service for "enter pre-operational".
- The value 403_{hex} must be written to index 1402, subindex 1.

Transmission mode

Various transmission modes can be selected for each **TX-PDO**:

- Event-controlled and synchronous (value 0):

  Every time the process data of a TX-PDO changes, this TX-PDO is sent after the next SYNC pulse.
- Cyclic and synchronous (value 1 - 240):

  After every SYNC pulse (1st to 240th, depending on the value), the TX-PDO is sent regardless of whether the content of the TX-PDO has changed or not.
- Manufacturer-specific (value 254):

  This TX-PDO is sent when the corresponding RX-PDO is received.

  Example: TX-PDO2 has the transmission mode 254. A TX-PDO2 is sent directly after a valid RX-PDO2 has been received (valid means that the length is not too long or too short).
- Event-controlled and asynchronous (value 255):

  This TX-PDO is set by MOVIDRIVE® B every time a value of the TX-PDO changes.

	TIPS
	If the speed, current, position or similar values that change quickly are set via the TX-PDO, this causes a very high load on the bus. The inhibit time can be used to limit the bus load to predictable values (see section "Inhibit time").

Default setting for all PDOs: Cyclic and synchronous (transmission mode = 1).

Various transmission modes can be selected for each **TX-PDO**:

- Synchronous (value 0 - 240):

  When the next SYNC pulse is received (it does not matter whether the value is 0 or 240), the data of the RX-PDO is transferred to the PO data buffer of MOVIDRIVE® B. This transmission process can be used to first send several PDOs from the master to the MOVIDRIVE® B and then transfer them consistently to the PO data buffer of MOVIDRIVE® B at the same time with a SYNC pulse.

- Manufacturer-specific (value 254):

  The corresponding TX-PDO is sent when an RX-PDO is received.

  Example: TX-PDO2 also has the transmission mode 254. A TX-PDO2 is sent directly after a valid RX-PDO2 has been received (valid means that the length is not too long or too short).

- Event-controlled and asynchronous (value 255):

  When an RX-PDO is received, its data is immediately transferred to the PO data buffer.

Default setting for all PDOs: Cyclic and synchronous (transmission mode = 1).

	TIPS
	Observe the following notes: • The transmission modes for the TX-PDOs 1 - 3 can be changed using object 1800_{hex} to 1802_{hex}, subindex 2. They are all 8-bit values. • The transmission modes for the RX-PDOs 1 - 3 can be changed using object 1400_{hex} to 1402_{hex}, subindex 2. They are all 8-bit values. • Transmission modes 241 - 253 are reserved and cannot be selected. • The transmission mode can only be changed when the unit is in the pre-operational state. • For information on the SYNC pulse, refer to chapter "SYNC object".

Inhibit time

This inhibit time is a lock-out time for TX-PDOs. The inhibit time for a TX-PDO begins once this object has been sent. This object cannot be sent to the CANopen bus again until the inhibit time has elapsed. The inhibit time is given in multiples of 100 µs; that is, 15670 corresponds to 1.567 seconds. The maximum inhibit time is 6.5535 s. MOVIDRIVE® B processes inhibit times with a resolution of 1.0 ms, i.e. the value 15 (corresponding to an inhibit time of 1.5 ms) is treated as 2 ms. The valid value range is "0" and "10 - 65535".

	TIPS
	• The inhibit time may only be changed in "pre-operational" unit state. The default value for the inhibit time of all PDOs is 0 (disabled). • The inhibit time for the TX-PDOs 1 - 3 can be changed using object 1800_{hex} to 1802_{hex}, subindex 3.

Mapping process data in the CANopen PDOs

Only the 10 objects from index 2020_{hex} (PO1 - 10) can be mapped to RX-PDO 1 - 3. A PDO can contain a maximum of 4 of these 16-bit values.

Name	Index (hex)	Subindex (hex)
PO1	2020	1
PO2		2
PO3		3
PO4		4
PO5		5
PO6		6
PO7		7
PO8		8
PO9		9
PO10		A

The TX-PDOs can be formed from the following objects:

Name	Index (hex)	Subindex (hex)
PI1	2021	1
PI2		2
PI3		3
PI4		4
PI5		5
PI6		6
PI7		7
PI8		8
PI9		9
PI10		A

The mapping can only be changed when the unit is in the pre-operational state.

If you want to change the mapping for a PDO, do the following:

- Set the mapping length of the PDO to "0". The mapping length for RX-PDO1 - 3 is stored in index 1600_{hex} - 1602_{hex}, subindex 0.
- Now you can change the mapping for this PDO. The value to be written is calculated from 65535 × index + 256 × subindex + 16. The result for RX-PDOs mapping entries is $20210x10_{hex}$, with x being a value between 1 and A_{hex} and designating the number of the PO or PI data word.
- Finally set the mapping length of the PDO to the number of IDs mapped in this PDO.

X = 1 - 3	Index (hex)	Subindex (hex)	Comment
Number of PDs mapped in RX-PDO X	15FF + X	0	Can be set to a value between 0 and 4
1st PO mapped in RX-PDO X		1	PO1 - PO10 can be mapped
2nd PO mapped in RX-PDO X		2	
3rd PO mapped in RX-PDO X		3	
4th PO mapped in RX-PDO X		4	

Mapping of TX-PDOs is stored in subindices 0 - 4 analog to mapping of RX-PDOs via indices $1A00_{hex}$ - $1A02_{hex}$.

5.4.3 SYNC object

The SYNC object is used to transfer the process data of several PDOs consistently and at a specific time to the process data buffer of MOVIDRIVE® B. All the PDOs which are to be synchronized with the SYNC object have to be operated in transmission mode 0 - 240. If the transmission mode of TX-PDO is set to 4, MOVIDRIVE® B will send this TX-PDO after every fourth SYNC message. With RX-PDOs, its data will be transferred to the PO data buffer with every SYNC message.

Changing the COB ID of the SYNC object

In "initializing status", the MOVIDRIVE® B defines the COB ID of the SYNC object as 080_{hex}. The COB ID should be changed in the "pre-operational" unit state. Although it can also be changed in the operational status, the CAN controller is temporarily separated from the bus. This means process data losses may occur in the operational status. As the MOVIDRIVE® B only is a SYNC consumer and only works with 11-bit COB IDs, bits 30 and 29 must always be set to "0". The structure of the COB ID and the meaning of the individual bits are described in the table below. The COB ID of the SYNC message is addressed as 32-bit value via index 1005_{hex}, subindex 0.

Bit				
31 (MSB)	**30**	**29**	**28 - 11**	**10 - 0 (LSB)**
X	0/1	0	0	11-bit identifier

Bit	Value	Meaning
31 (MSB)	X	Do not care
30	0	Device does not generate SYNC message
	1	Device generates SYNC message
29	0	11-bit ID (CAN 2.0A)
	1	29-bit ID (CAN 2.0B)
28 - 11	0	If bit 29 = 0
	X	If bit 29 =1: bits 28 - 11 of 29-bit-SYNC-COB-ID
10 - 0 (LSB)	X	Bits 10 - 0 of SYNC-COB-ID

5.4.4 The emergency object

The emergency object is always sent once by MOVIDRIVE® B when an error is detected and when this error no longer exists. MOVIDRIVE® B sends an emergency object in the event of the following errors.

- MOVIDRIVE® B sets the error bit in its status word.

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
0	FF_{hex}	Error register (object 1001_{hex})	0	Status word 1 of MOVIDRIVE® B, low	Status word 1 of MOVIDRIVE® B, high	0	0

- MOVIDRIVE® B operates in DC 24 V backup mode only; rotating field voltage is missing.

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
0	31_{hex}	Error register (object 1001_{hex})	0	0	0	0	0

- The lifeguarding protocol was disabled but was not executed cyclically within the timeout time.

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
30_{hex}	81_{hex}	Error register (object 1001_{hex})	0	0	0	0	0

COB ID of the emergency object

The COB ID should be changed in the "pre-operational" unit state. Although it can also be changed in the operational status, the CAN controller is temporarily separated from the bus. This means process data losses may occur in the operational status. Bit 29 must always be 0 because MOVIDRIVE® B only works with 11-bit COB IDs. The structure of the COB ID and the meaning of the individual bits are described in the following tables. If MOVIDRIVE® B is not to send an EMCY object , you can deactivate it by setting bit 31. The COB ID is addressed as unsigned long via index 1014_{hex}, subindex 0.

Bit				
31 (MSB)	30	29	28 - 11	10 - 0 (LSB)
0/1	0	0	0	11-bit identifier

Bit	Value	Meaning
31 (MSB)	0	EMCY exists / is valid
	1	EMCY does not exist / is not valid
30	0	reserved (always 0)
29	0	11-bit ID (CAN 2.0A)
	1	29-bit ID (CAN 2.0B)
28 - 11	0	If bit 29 = 0
	X	If bit 29 =1: bits 28 - 11 of 29-bit SYNC COB ID
10 - 0 (LSB)	X	Bits 10 - 0 of SYNC COB ID

In "initializing status", the MOVIDRIVE® B defines the COB ID of the emergency object as 080_{hex} + slave address.

Inhibit time of the emergency object

The inhibit time of the emergency object on the CANopen bus is given as unsigned 16 (2 bytes) via index 1015_{hex}, subindex 0. The inhibit time is defined as a multiple of 100 µs, i.e. the value 3000 corresponds to an inhibit time of 300 ms.

5.4.5 Heartbeat and lifetime

Heartbeat and lifetime

The following applies: A MOVIDRIVE® B node either uses the heartbeat protocol or the lifetime protocol. Combined operation is not possible under any circumstances.

Lifetime (guarding)

The CANopen controller sends a node-guarding object with set RTR bit to the MOVIDRIVE® B CANopen slave. The slave responds by sending a node-guarding object with a data length of 1 byte. The node-guarding object always has the fixed COB ID 700_{hex} + CANopen slave address. The slave expects these node-guarding objects cyclically within a timeout time. If the controller exceeds this timeout time, the error response for CAN timeout active is triggered (P836 and P837) in MOVIDRIVE® B.

The timeout time can be set in milliseconds with the indices 0x100C ("guard time") and 0x100D ("life time factor"). This timeout time is calculated as the product of lifetime factor and guard time. Timeout times shorter than 10 ms are rejected.

	TIP
	Parameters P883 and P893 are used to read the timeout time set by the controller. The timeout time must not be modified. It results from the CAN open objects 0x100C and 0x100D set by the controller.

Node-guarding is only active once the first node-guarding object sent by the master is received. If the product of life time factor × guard time is 0, the node-guarding function is deactivated and the heartbeat mechanism can be used (see section "Hearbeat").

Heartbeat

MOVIDRIVE® B is a heartbeat producer. The time interval in which heartbeats are produced can be set using index 1017_{hex}, subindex 0 by means of an unsigned 16 value. This value corresponds to the heartbeat in ms, i.e. 3000 means that a heartbeat is sent every 3 s. The heartbeat object always has the fixed COB ID 700_{hex} + CANopen slave address.

The default value for index 1017_{hex}, subindex 0 is "0", which means hearbeat is disabled. MOVIDRIVE® B can be configured as a heartbeat consumer at the same time. MOVIDRIVE® B can monitor exactly one other CANopen node to check whether it is producing the heartbeats within a timeout time. The timeout time and node number of the node to be monitored is defined with the object 1016_{hex}, subindex 1.

	Bit		
	31 - 24	**23 - 16**	**15 - 0**
Value	Reserved, always 0	CANopen address to be monitored	Heartbeat consume timeout in ms
Data type		UNSIGNED8	UNSIGNED16

If index 1017_{hex}, subindex 0 and index 1016_{hex}, subindex 1 are set to 0, the heartbeat function is deactivated and the guarding protocol can be used.

5.4.6 Parameter access via SDO

MOVIDRIVE® B supports an SDO channel. The COB IDs are fixed for this SDO channel. For RX-SDO, the COB ID is "600_{hex} + CANopen slave address", and for TX-SDO the COB ID is "580_{hex} + CANopen slave address". The SDO channel supports expedited and non-expedited transfers. SDO mechanisms are described in detail in the CANopen specification DS301.

Example:

- The transfer mode of TXPDO1 (index 0x1800 subindex 2) is to be read.
- The request telegram with the COB ID 600hex + CANopen slave address contains the 8 data bytes "40 00 18 02 xx xx xx xx" (hexadecimal notation).
  - 40 = read command
  - 00 18 = index (low byte first)
  - 02 = subindex
  - xx xx xx xx = without meaning
- The response telegram with the COB ID 580hex + CANopen slave address contains the 8 data bytes "4F 00 18 02 01 xx xx xx xx" (hexadecimal notation).
  - 4F = 1 byte read
  - 00 18 = index
  - 02 = subindex
  - 01 = value (= synchronous)
  - xx xx xx = without meaning
- The following SDO commands and responses are important:
  - 2F = write 1 byte (command)
  - 2B = write 2 bytes (command)
  - 23 = write 4 bytes (command)
  - 60 = successfully written (response)
  - 4F = 1 byte read (response)
  - 4B = 2 bytes read (response)
  - 43 = 4 bytes read (response)
  - 80 = error while executing service (response)

In the 4-byte data range of an SDO telegram, the valid data bytes are entered in Intel format aligned to the left (low byte first). All communication specific indices of MOVIDRIVE® B are listed in the EDS file "mdxb.eds".

Parameter access via SDOs to the SEW-specific parameters of MOVIDRIVE® B

All SEW-specific parameters of the MOVIDRIVE® B (0x2000-0xFFFF) are stored in the corresponding index with subindex 0.

Example: You have to access index 8300_{dec}, subindex 0 to read the software version of MOVIDRIVE® B.

CANopen only allows the "read" and "write" services for manufacturer-specific object via SDO. If you want to use the SEW-specific services of the MOVILINK® fieldbus unit profile (e.g. "read minimum", "read maximum", "read default", "write volatile", ...), you can do so using objects 0x2066 and 0x2067. Object 0x2067 (SIGNED32) contains the data that should be performed in the next MOVILINK® service or the result of the last MOVILINK® service, if it was successful. Writing the object 0x2066 triggers the MOVILINK® service. The object 0x2066 (UNSIGNED32) is structured as follows:

Bit 31 - bit 24	Bit 23 - bit 16	Bit 15- bit 8	Bit 7 - bit 0
Administration	Reserved	Index high	Index low

For a detailed description of the management byte, refer to the chapter "SEW unit profile". In the event of an error, "abort SDO" signals a general error. You can read the exact MOVILINK® error code from parameter 0x2067.

Example 1:The maximum possible value (service "Read maximum", 0x35) is to be determined by the inverter for the index 0x2116 (ramp up CW).

To do so, the management byte must assume the value 0x35, reserved is set to 0. The object 0x2066 is written using the value 0x35002116 via an SDO. The maximum possible value can then be read from index 0x2067 using a reading SDO access.

Example 2:The index 0x2116 (ramp up CW) is to be written to the volatile memory with the value 0x1234 (service "write volatile", 0x33).

Before the service is executed, the MOVILINK® service data (Index 0x2067) must be set to the value 0x1234. This is done using a writing SDO in index 0x2067. The MOVILINK® service is then performed by writing the value 0x33002116 to index 0x2066.

Example 3:The index 0x2116 (ramp up CW) is to be written to the volatile memory with the value 0x4000000 ("write volatile" service, 0x33).

Before the service is executed, the MOVILINK® service data (Index 0x2067) must be set to the value 0x1234. This is done using a writing SDO in index 0x2067. The MOVILINK® service is then performed by writing the value 0x33002116 to index 0x2066. The unit signal "general error" via CANopen. Read to 0x2067 then returns the MOVILINK® error code 0x08000015 (value too great).

5.4.7 Hard synchronization for synchronous operation or positioning several MDX-B units

When the controller sends the CANopen SYNC object, all synchronous PDOs sent are transferred consistently to MOVIDRIVE® B using the SYNC object, if the factory settings are used. The individual controller systems (processor time slices) of several axes on one CANopen line are not synchronized with one another (soft synchronization). However, the period of the SYNC object can be set as required in the controller.

If the controllers (processor time slices) are to synchronize several units with the SYNC signal (hard synchronization), parameter P885 / P895 must be set to the COB ID of the CANopen SYNC object (usually the value "128" = "80hex"). In this case, parameters P887 and P888 also have to be set matching to the synchronization period of the controller.

5.4.8 Other unit properties in the CANopen profile

- The master/slave functionality is not available with the CANopen profile but only with the MOVILINK® profile via SBUS.
- The CANopen via SBus communication profile also affects the following IPOS$^{plus®}$ functions:

  GETSYS PO data, SETSYS PI data and MOVILINK® (see chapter "Using the CAN interfaces in IPOS$^{plus®}$ (depending on the profile)")

5.4.9 CANopen-specific objects of MOVIDRIVE® B

Index	Subindex	Function	Access	Type	Mappable	Default value
1000	0	device type	ro	UNSIGNED32	NO	0
1001	0	error register	ro	UNSIGNED32	NO	
1002	0	manuf. Status register	ro	UNSIGNED32	NO	
1005	0	COB-ID SYNC message	rw	UNSIGNED32	NO	80hex
1008	0	manufacturer device name	ro	STRING	NO	MDXB
100B	0	node id of server	ro	UNSIGNED32	NO	
100C	0	guard time [ms]	rw	UNSIGNED16	NO	0
100D	0	life time factor	rw	UNSIGNED8	NO	0
100E	0	COB ID lifetime	rw	UNSIGNED32	NO	0
100F	0	no. of SDO	ro	UNSIGNED32	NO	1
1014	0	COB-ID EMCY message	rw	UNSIGNED32	NO	80hex+slave ID
1015	0	inhibit time EMCY [ms]	rw	UNSIGNED16	NO	0
1016	0	consumer heartbeat time / no. of entries	ro	UNSIGNED32	NO	1
1016	1	producer 1	rw	UNSIGNED32	NO	0
1017	0	producer heartbeat time [ms]	rw	UNSIGNED16	NO	0
1018	0	identity object / no. of entries	ro	UNSIGNED32	NO	1
1018	1	vendor id	ro	UNSIGNED32	NO	89
1200	0	server SDO parameter / no. of entries	ro	UNSIGNED32	NO	2
1200	1	COB-ID Client → Server	ro	UNSIGNED32	NO	580hex+slave ID
1200	2	COB-ID Server → Client	ro	UNSIGNED32	NO	600hex+Slave-ID
RX-PDO communication parameter						
1400	0	RX-PDO1 parameter / no. of entries	ro	UNSIGNED32	NO	2
1400	1	COB-ID	rw	UNSIGNED32	NO	200hex+slave ID
1400	2	transmission type	rw	UNSIGNED8	NO	1
1401	0	RX-PDO2 parameter / no. of entries	ro	UNSIGNED32	NO	2
1401	1	COB-ID	rw	UNSIGNED32	NO	300hex+slave ID
1401	2	transmission type	rw	UNSIGNED8	NO	1
1402	0	RX-PDO3 parameter / no. of entries	ro	UNSIGNED32	NO	2
1402	1	COB-ID	rw	UNSIGNED32	NO	400hex+slave ID
1402	2	transmission type	rw	UNSIGNED8	NO	1

Index	Subindex	Function	Access	Type	Mappable	Default value
RX-PDO1 mapping parameter						
1600	0	no. of entries	rw	UNSIGNED8	NO	3
1600	1	1. Mapping	rw	UNSIGNED32	NO	20200110hex
1600	2	2. Mapping	rw	UNSIGNED32	NO	20200210hex
1600	3	3. Mapping	rw	UNSIGNED32	NO	20200310hex
1600	4	4. Mapping	rw	UNSIGNED32	NO	0
RX-PDO2 mapping parameter						
1601	0	no. of entries	rw	UNSIGNED8	NO	3
1601	1	1. Mapping	rw	UNSIGNED32	NO	20200410hex
1601	2	2. Mapping	rw	UNSIGNED32	NO	20200510hex
1601	3	3. Mapping	rw	UNSIGNED32	NO	202610hex
1601	4	4. Mapping	rw	UNSIGNED32	NO	0hex
RX-PDO3 mapping parameter						
1602	0	no. of entries	rw	UNSIGNED8	NO	4
1602	1	1. Mapping	rw	UNSIGNED32	NO	20200710hex
1602	2	2. Mapping	rw	UNSIGNED32	NO	20200810hex
1602	3	3. Mapping	rw	UNSIGNED32	NO	20200910hex
1602	4	4. Mapping	rw	UNSIGNED32	NO	20200A10hex
TX-PDO1 communication parameter						
1800	0	TX-PDO1 parameter / no. of entries	ro	UNSIGNED32	NO	3
1800	1	COB-ID	rw	UNSIGNED32	NO	40000180hex + slave ID
1800	2	transmission type	rw	UNSIGNED8	NO	1
1800	3	inhibit time [100 µs]	rw	UNSIGNED16	NO	0
1801	0	TX-PDO2 parameter / no. of entries	ro	UNSIGNED32	NO	3
1801	1	COB-ID	rw	UNSIGNED32	NO	C0000280hex + slave ID
1801	2	transmission type	rw	UNSIGNED8	NO	1
1801	3	inhibit time [100 µs]	rw	UNSIGNED16	NO	0
1802	0	TX-PDO3 parameter / no. of entries	ro	UNSIGNED32	NO	3
1802	1	COB-ID	rw	UNSIGNED32	NO	C0000380hex + slave ID
1802	2	transmission type	rw	UNSIGNED8	NO	1
1802	3	inhibit time [100 µs]	rw	UNSIGNED16	NO	0
TX-PDO1 mapping parameter						
1A00	0	no. of entries	rw	UNSIGNED8	NO	3
1A00	1	1. Mapping	rw	UNSIGNED32	NO	20210110hex
1A00	2	2. Mapping	rw	UNSIGNED32	NO	20210210hex
1A00	3	3. Mapping	rw	UNSIGNED32	NO	20210310hex
1A00	4	4. Mapping	rw	UNSIGNED32	NO	0
TX-PDO2 mapping parameter						
1A01	0	no. of entries	rw	UNSIGNED8	NO	3
1A01	1	1. Mapping	rw	UNSIGNED32	NO	20210410hex
1A01	2	2. Mapping	rw	UNSIGNED32	NO^	20210510hex
1A01	3	3. Mapping	rw	UNSIGNED32	NO	20210610hex
1A01	4	4. Mapping	rw	UNSIGNED32	NO	0hex

Index	Subindex	Function	Access	Type	Mappable	Default value
TX-PDO3 mapping parameter						
1A02	0	no. of entries	rw	UNSIGNED8	NO	4
1A02	1	1. Mapping	rw	UNSIGNED32	NO	20210710hex
1A02	2	2. Mapping	rw	UNSIGNED32	NO	20210810hex
1A02	3	3. Mapping	rw	UNSIGNED32	NO	20210910hex
1A02	4	4. Mapping	rw	UNSIGNED32	NO	20210A10hex
Process data for mapping						
2020	0	PO Data / no. of entries	ro	UNSIGNED32	NO	10
2020	1	PO 1	rrw	SIGNED16	YES	0
2020	2	PO 2	rrw	SIGNED16	YES	0
2020	3	PO 3	rrw	SIGNED16	YES	0
2020	4	PO 4	rrw	SIGNED16	YES	0
2020	5	PO 5	rrw	SIGNED16	YES	0
2020	6	PO 6	rrw	SIGNED16	YES	0
2020	7	PO 7	rrw	SIGNED16	YES	0
2020	8	PO 8	rrw	SIGNED16	YES	0
2020	9	PO 9	rrw	SIGNED16	YES	0
2020	10	PO 10	rrw	SIGNED16	YES	0
2021	0	PI Data / no. of entries	ro	UNSIGNED32	NO	10
2021	1	PI 1	ro	SIGNED16	YES	0
2021	2	PI 2	ro	SIGNED16	YES	0
2021	3	PI 3	ro	SIGNED16	YES	0
2021	4	PI 4	ro	SIGNED16	YES	0
2021	5	PI 5	ro	SIGNED16	YES	0
2021	6	PI 6	ro	SIGNED16	YES	0
2021	7	PI 7	ro	SIGNED16	YES	0
2021	8	PI 8	ro	SIGNED16	YES	0
2021	9	PI 9	ro	SIGNED16	YES	0
2021	10	PI 10	ro	SIGNED16	YES	0
Access to all MOVILINK® services						
2066	0	Movilink Service	rw	UNSIGNED32	NO	0
2067	0	Movilink Data	rw	UNSIGNED32	NO	0

5.5 Other unit functions via CAN interfaces

In addition to the exchange of process and parameter data between controller and MOVIDRIVE® B, the CAN interfaces can be used for other additional functions, such as IPOS$^{plus®}$, master/slave operation, or internal synchronous operation (ISYNC).

5.5.1 Using CAN interfaces for master/slave operation

Master/slave operation is only possible via CAN 1 (SBus 1). In this case, the MOVILINK® profile must be activated in parameter P880.

	TIP
	P882 SBus group address must be set to the same value for master and slave. For operation via system bus (e.g. master/slave operation), the bus terminating resistors at the start and end of the system bus must be activated (S12 = ON). If the sending of slave setpoints via SBus is set using parameter P750, MOVIDRIVE® can respond to requests (process data telegrams) of another SBus master (P100/101 ≠ SBus (CAN)) via this SBus interface.

Connection check

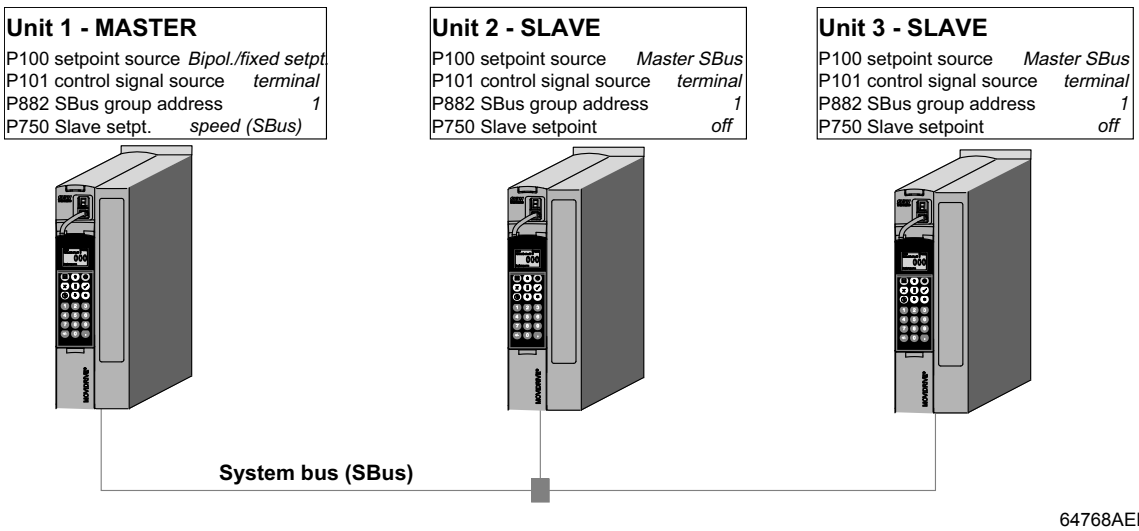
If communication takes place via SBus, *P883 SBus timeout interval* applies. If *P883 SBus timeout interval* is set to 0, data transmission via SBus will not be monitored. For a detailed description of the setting and operation of the master/slave function, refer to chapter 4.4.1.

Example

Master/slave operation can be executed via SBus 1. Every millisecond, the master sends setpoint and control word to the slaves via SBus group telegram.

The same SBus group address (P882) must be set for master and slaves. The valid address range is 0 - 63. Permitted baud rates for master/slave operation are 500 kBaud and 1000 kBaud (P884). Make sure that the baud rate is the same for master and slaves.

The master uses SBus group telegrams for master/slave communication. Therefore, the master cannot be controlled by other stations via SBus group telegrams as this would disturb arbitration on the CAN bus.



5.5.2 Using CAN interfaces in IPOS$^{plus®}$ (depending on the profile)

In the IPOS$^{plus®}$ positioning and sequence control system integrated in MOVIDRIVE® B, you can

- directly access the process data transmitted via SBus using the commands GETSYS PO data and SETSYS PI data.

  Depending on the chosen profile (P880/890), the data are read or written in CANopen format (low byte first) or in MOVILINK® format (high byte first).

- access process and parameter data of other SEW drives connected via CAN using the MOVILINK® command.

  Depending on whether MOVILINK® or CANopen was set as profile (P880/890), the MOVILINK® structure differs for parameter access.

Set the bus type as follows:

- "5" for access via CAN1 (SBus 1)
- "8" for access via CAN 2 (SBus 2)

	TIP
	For more information on IPOS$^{plus®}$ commands, refer to the "IPOS$^{plus®}$ Positioning and Sequence Control " manual.

5.5.3 Using CAN interfaces in IPOS$^{plus®}$ (independent of the profile)

Variable telegrams were introduced to create an open CAN bus interface. You have the free choice of the identifier with which to send the telegrams with the variable telegrams; the 8 bytes of data on the CAN bus are used for the content of two variables.

In this way, an interface is provided that can be used to directly access layer 2 of the CAN bus. Consequently, the maximum processing speed is achieved for variable transmission via the CAN bus.

The CAN bus is multimaster-capable which means every station can send a message. All bus stations always listen actively to see which messages are being sent on the bus. Each station filters the relevant telegrams and makes the data available to the application.

These features allow for working with an object-oriented approach. The stations send objects and those stations which want to process these objects receive them.

The variable telegrams are created independent of the profile set with the SCOM services (TRCYCL, TRACYCL und REC).

An offset of 1,000,000hex is necessary for the identifier in the SCOMDEF command to access SBus 2. Use the SCOMST command instead of the SCOMON command. This command lets you start or stop SBus 1 and SBus 2 together or separately.

	TIP
	Every station can send and receive objects. However, the following rules have to be adhered to taking account of the identifiers reserved for MOVILINK® and CANopen telegrams: 1. A particular identifier is only allowed to be sent by one station. This means those identifiers used for sending messages in the MOVILINK® / CANopen profile are no longer available for the exchange of variables. 2. An SBus identifier must be used only once in a unit. This means those identifiers used for the SBus MOVILINK® / CANopen profile in a unit are no longer available for the transmission of variables.

	TIP
	For more information on IPOS$^{plus®}$ commands, refer to the "IPOS$^{plus®}$ Positioning and Sequence Control" manual.

5.5.4 Using CAN interfaces for integrated synchronous operation (ISYNC via SBus)

	TIP
	Refer to the "MOVIDRIVE® MDX61B Internal Synchronous Operation" manual for detailed information. If you do not only want to synchronize several axes but also want to implement connection via fieldbus/SBus to a higher-level controller, we recommend that you use the MOVI-PLC® DH..B option.

Basic operating principle

A cyclical SBus telegram transfers a master position (e.g., the actual position of the master or the value of the virtual encoder) from the master to the slaves via the SBus (CAN). Sending and transferring the master position at equal intervals avoids aliasing. Therefore, the time slices of the inverter are synchronized with a synchronization message.

Settings for operation using SBus 1

1. **Master**

   A An SBus telegram cyclically (e.g. every 1 ms) sends the master position to the slaves. It is created in IPOS$^{plus®}$ using the *_SBusCommDef* command:

Structure	Element	Setting
SCTRCYCL	ObjectNo	As small as possible (see 4 D)
	CycleTime	1 - 5 (see M filter description)
	Offset	0
	Format	4 (4-byte, Motorola format)
	DPointer	e.g. 511 (= ActPosMot), 376 (= VEncoder)

   B The synchronization message is an SBus telegram with high priority (small object number), which is sent cyclically every 5 ms. It is created in IPOS$^{plus®}$ using the *_SBusCommDef* command:

Structure	Element	Setting
SCTRCYCL	ObjectNo	0, 1 or 2
	CycleTime	5
	Offset	0
	Format	0 (0 byte)
	DPointer	e.g. 511 (= ActPosMot), 376 (= VEncoder)

   C The synchronization ID (P817 or P885/895 with MOVIDRIVE® B) must be **different** from the object number of the synchronization message (see 1 B) and all other object numbers sent on the SBus.

2. **Slaves**

   A The synchronization ID (P817 or P885/895 with MOVIDRIVE® B) must be **the same** as the object number of the master's synchronization message (see 1 B).

   B The master position of the master is received and stored in an H variable.

It is created in IPOS$^{plus®}$ using the *_SBusCommDef* command:

Structure	Element	Setting
SCREC	ObjectNo	See 1A
	Format	4 (4-byte, Motorola format)
	DPointer	xxx

C The H variable (Hxxx) defined under 2 B is selected as setpoint source for the position using *H430 MasterSource*.

D The filter time for the position setpoint (H446) must be set to a value greater than or equal to the cycle time of the setpoint telegram (see 1 A "CycleTime").

3. **Slaves, which also act as master for other slaves**

A The synchronization ID (P817 or P885/895 with MOVIDRIVE® B) must be **the same** as the object number of the synchronization message of the master (see 1 B).

B The master position of the master is received and stored in an H variable. It is created in IPOS$^{plus®}$ using the *_SBusCommDef* command:

Structure	Element	Setting
SCREC	ObjectNo	See 1A
	Format	4 (4-byte, Motorola format)
	DPointer	xxx

C The H variable (Hxxx) defined under 2 B is selected as setpoint source for the position using *H430 MasterSource*.

D The filter time for the position setpoint (H446) must be set to a value greater than or equal to the cycle time of the setpoint telegram (see 1 A "CycleTime").

E Analogous to 1 A, a cyclic telegram is created for sending the master position of this master to its slaves. The object number should be as small as possible (see 4 D) but greater than the object number chosen for 1 A.

4. **The following requirements must be fulfilled for successful internal synchronization via SBus:**

A The object number of the synchronization message must be the smallest object number on the SBus **(e.g. "0")**.

B The time slice of the master must not be synchronized by other participants. Therefore, P817 or P885/P895 (with MOVIDRIVE® B) for the master must be different from all object numbers existing on the SBus **(e.g. "2047")**.

C Observe the sequence in which the sending of master position and synchronization message is created in the master using the *_SBusCommDef* command. Always the master position first and then the synchronization message.

D Following the synchronization message, the object number of the master position should be the smallest object number on the SBus, e.g. "1". The smallest object number on the SBus "8 × SBus address + 3" is defined via the MOVILINK® profile. Sufficiently high prioritized telegrams for internal synchronization are achieved with SBus addresses greater than or equal to 1.

5. **Other important points**

   A If internal synchronization is operated via SBus, all units connected to this SBus (except for 1 C) have to be synchronized to the same synchronization telegram (see 1 B).
   The following units cannot be synchronized and must therefore not be connected:
   – UFx option as fieldbus or diagnostics interface
   – Diagnostics PC with MOVITOOLS® MotionStudio via PC CAN interface
   – MOVITRAC®

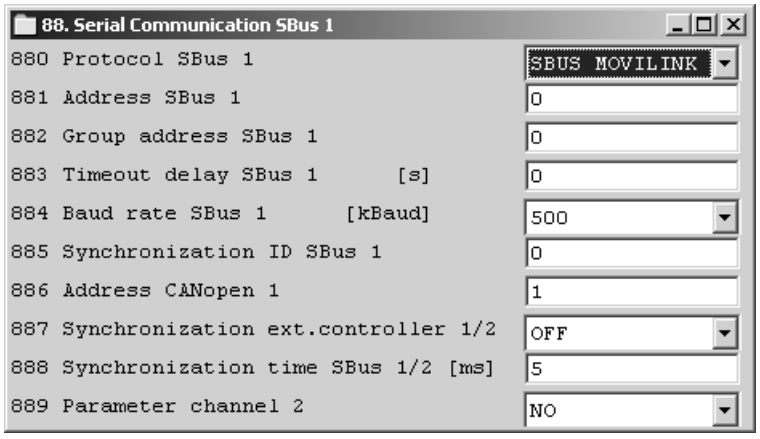
   B The number of telegrams is limited depending on the baud rate of the SBus:
   – Baud rate = 1 MBaud: Synchronization + 4 telegrams
   – Baud rate = 500 kbaud: Synchronization + 2 telegrams
   – No synchronization is possible if the baud rate is set to 250 or 125 kBaud.

   Individual cases where these limits are exceeded have to be checked thoroughly.

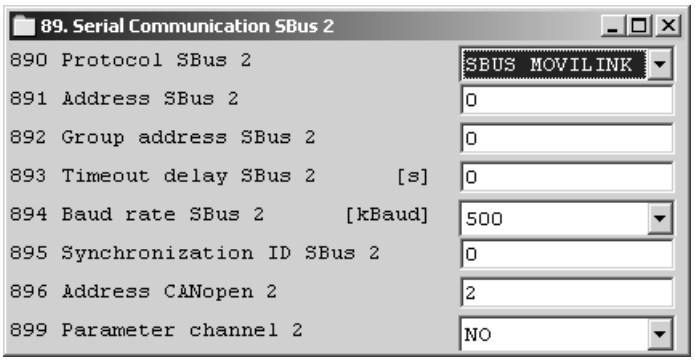
   C Ensure that the entire calculated bus utilization does not exceed 70 % for additional data exchange between slaves. The bus utilization is calculated in bits per second using the formula:
   Number of telegrams × bits per telegram × 1/cycle time
   Example: 2 telegrams à 100 bits in 1 ms cycle
   = 200000 bits/s = 200 kBaud
   This results in the following bus load percentage in reference to the selected baud rate.
   Example: 200 kBaud / 500 kBaud = 40 % < 70 %

   D The object number of the master position should be the smallest object number on the SBus following the synchronization message, for example 1. The smallest object number on the SBus "8 × SBus address + 3" is defined via the MOVILINK® profile. Sufficiently high prioritized telegrams for internal synchronization are achieved with SBus addresses greater than or equal to 1.

6. Settings in **MOVITOOLS® MotionStudio**



12111AEN



12112AEN

The following parameters are important in parameter groups P88x and P89x for MOVIDRIVE® B:

- P880/P890 Protocol SBus 1/2
  Only the MOVILINK® protocol allows for this synchronization

- P884/P894 SBus baud rate 1/2
  All stations on the SBus must be set to the same baud rate. Baud rate, maximum cable length, and telegrams/time depend on one another.

- P885/P895
  See points 1B, 2A, and 4A. The time slice can be synchronized in the MOVILINK® protocol via both SBus interfaces. It is important that synchronization messages are received via only one SBus.

- P887 Synchronization ext. controller 1/2
  The time base for all units to be synchronized must be either about 1 ms (standard for MOVIDRIVE® A; P887 = OFF for MOVIDRIVE® B) or exactly 1 ms (P887 = ON, only for MOVIDRIVE® B).

6 Fieldbus Interfaces via Option Card for MOVIDRIVE® B

MOVIDRIVE® MDX61B (not MDX60B) can be equipped with a fieldbus option card for easy connection to the following bus systems:

- DFP21B for PROFIBUS DP
- DFD11B for DeviceNet
- DFI11B for INTERBUS copper
- DFI21B for INTERBUS fiber optic cable
- DFE24B for EtherCAT and SBUS^{plus}
- DFE32B for PROFINET IO
- DFE33B for EtherNet/IP and Modbus/TCP
- DFC11B for CAN 2 (SBus 2)

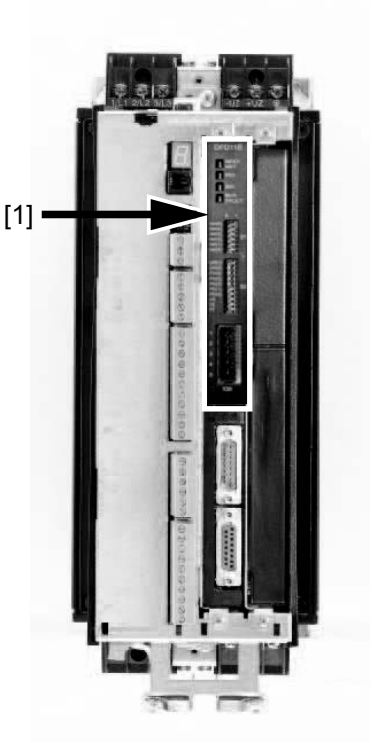
A manual is available for each fieldbus option mentioned aboved (except for DFC11B). This manual provides detailed information on connection, startup, scope of functions, and diagnostics options.

The DFC11B is not actually a fieldbus option. It offers a connection option for CAN 2 (SBus 2). Refer to the manual for detailed information on connection, startup, scope of functions, and diagnostics options (see chapter "CAN interfaces of MOVIDRIVE® MDX60B/61B").

Other options cards, such as DCS..B safety monitor, use the CAN 2 connection internally. Simultaneous operation of the DCS..B option with the DFC11B option is not permitted.

6.1 Installing a fieldbus option card in MOVIDRIVE® MDX61B

	TIPS
	Only SEW-EURODRIVE personnel may install or remove option cards for MOVIDRIVE® MDX61B size 0. • Users may only install or remove option cards for MOVIDRIVE® MDX61B sizes 1 to 6. • The fieldbus option card must be plugged into the fieldbus socket [1]. • The fieldbus option is powered by MOVIDRIVE® B. A separate voltage supply is not required.



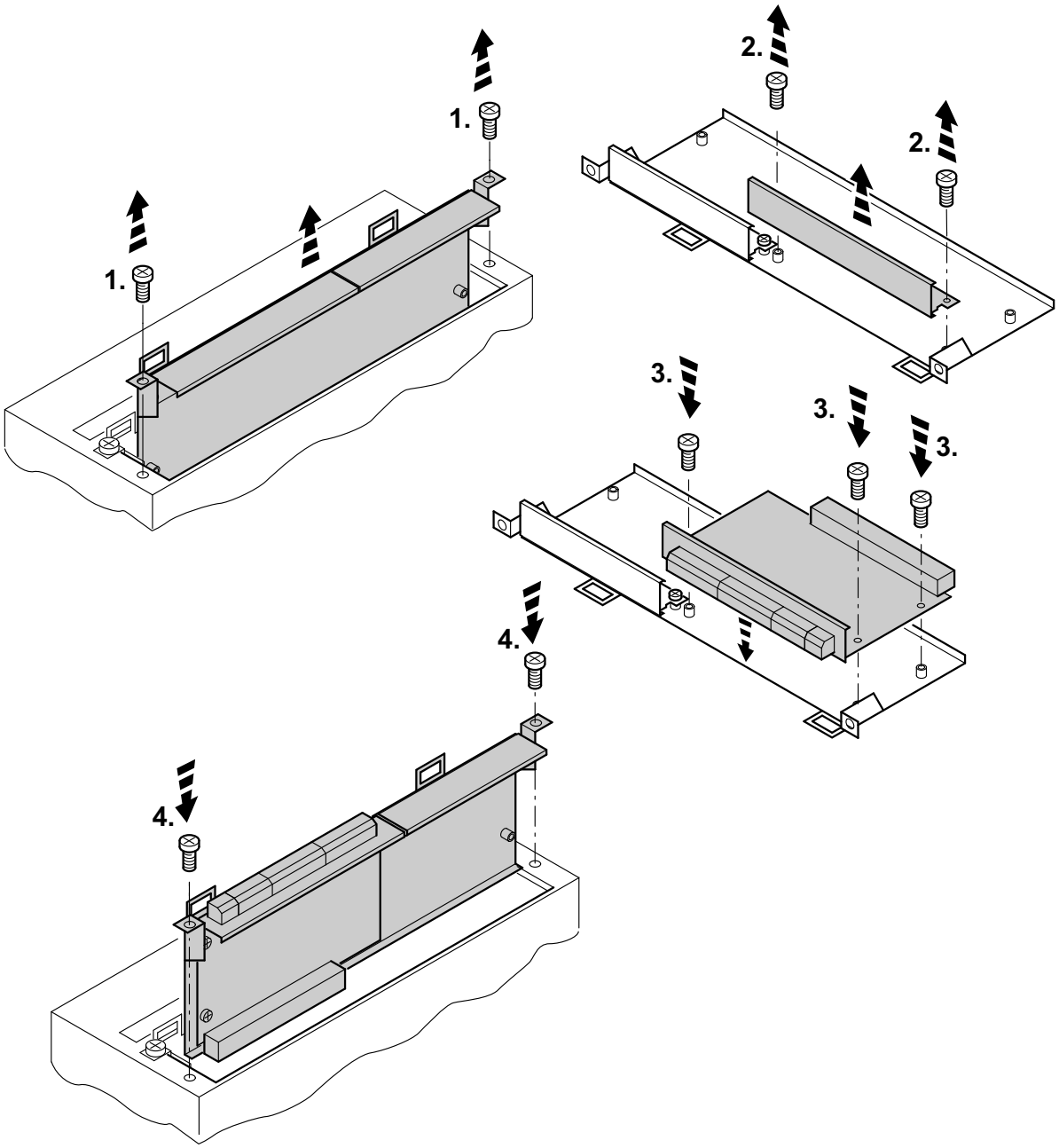
62594AXX

6.1.1 Before you start

Observe the following notes before installing or removing an option card:

- Disconnect the inverter from the power. Switch off the DC 24 V and the line voltage.
- Take appropriate measures to protect the option card from electrostatic charge (use discharge strap, conductive shoes, etc.) before touching it.
- **Before installing** the option card, remove the keypad and the front cover (→ MOVIDRIVE® MDX60B/61B operating instructions, section "Installation").
- **After having installed** the option card, replace the keypad and the front cover (→ MOVIDRIVE® MDX60B/61B operating instructions, section "Installation").
- Keep the option card in its original packaging until immediately before you are ready to install it.
- Hold the option card by its edges only. Do not touch any of the components.

6.1.2 Basic procedure for installing/removing an option card (MDX61B, sizes 1 - 6)



60039AXX

1. Remove the two retaining screws holding the card retaining bracket. Pull the card retaining bracket out evenly from the slot (do not twist!).
2. Remove the 2 retaining screws from the black cover plate on the card retaining bracket. Remove the black cover plate.
3. Position the option card onto the retaining bracket so that the three retaining screws fit into the corresponding bores on the card retaining bracket.
4. Insert the retaining bracket with the installed option card into the slot, pressing slightly so it is seated properly. Secure the card retaining bracket with the two retaining screws.
5. To remove the option card, follow the instructions in reverse order.

6.2 Parameters for configuring communication via fieldbus option

The following parameters are used for configuring the communication in addition to the DIP switches on the fieldbus option cards. The factory setting of the individual parameters is underlined.

	TIP
	Using parameters P090 - P099 you can monitor process data and settings that are chosen by means of the setpoint source set in P100 and P101.

Parameter			
No.	**Name**	**Setting**	**Meaning**
100	Setpoint source	TERMINALS RS485 FIELDBUS SBus	This parameter is used to set the setpoint source for the inverter.
101	Control signal source	TERMINALS RS485 FIELDBUS SBus	This parameter is used to set the source of the control signals for the inverter (CONTROLLER INHIBIT, ENABLE, CW, CCW, ...). Control via IPOS$^{plus®}$ and terminal is taken into account disregarding of P101.
780	IP address	000.000.000.000 - 192.168.10.x - 223.255.255.255	Use P780 to set the IP address for linking MOVIDRIVE® B via Ethernet. If the DHCP is activated using P785, the value specified by the DHCP server will be displayed.
781	Subnetwork mask	000.000.000.000 - 255.255.255.0 - 255.255.255.255	Factory setting at delivery is a class C network. The subnetwork mask divides the network into subnetworks. The set bits determine which part of the IP address represents the address of the subnetwork. If DHCP is activated, the value specified by the DHCP server will be displayed.
782	Standard gateway	000.000.000.000 - 255.255.255.255	The standard gateway is addressed if the desired communication partner is not within the actual network. The standard gateway will have to be part of the actual network. If DHCP is activated, the value specified by the DHCP server will be displayed.
783	Baud rate	-	Display value, cannot be altered. Shows the current baud rate of the Ethernet connection. The value "0" is displayed during the initialization phase.
784	MAC address	-	Display value, cannot be altered. Indicates the MAC address, i.e. the layer 2 Ethernet address of the interface, that is clearly assigned worldwide.
785	DHCP / Startup configuration	DHCP / Saved parameters	• DHCP: The option is assigned its IP parameters (P780 - P782) by a DHCP server when the supply voltage is switched on. • Saved IP parameters: The option is started with the saved IP parameters when the supply voltage is switched on.

Parameter			
No.	**Name**	**Setting**	**Meaning**
819	Fieldbus timeout interval	0 - 0.5 - 650 s	P819 sets the monitoring time for data transmission via the implemented fieldbus (DFx). MOVIDRIVE® B performs the error response set in P831 Response FIELDBUS TIMEOUT if there is no data traffic via the fieldbus for the period set in P819. When P819 is set to the value 0 or 650, data transmission via the fieldbus is not monitored. The timeout interval is automatically specified by the master except for Modbus/TCP. Changing this parameter does not have any effect.
831	Fieldbus timeout response	RAPID STOP/WARN.	The error is only triggered in the ENABLED inverter status. P831 programs the error response which is triggered by the fieldbus timeout monitoring.
870 **871** **872**	Setpoint description PO1 Setpoint description PO2 Setpoint description PO3	Factory set to: CONTROL WORD 1 SPEED NO FUNCTION	P870/P871/P872 define the content of the process output data words PO1/PO2/PO3.
873 **874** **875** **876**	Actual value description PI1 Actual value description PI2 Actual value description PI3 Enable PO data	Factory set to: STATUS WORD 1 SPEED NO FUNCTION ON	The content of process input data words PI1/PI2/PI3 is defined.
887	Synchronization ext. controller	On/off	By default, the time base of MOVIDRIVE® units is slightly smaller than 1 ms. For synchronization with an external controller, the time base can be set to exactly 1 ms.
888	Synchronization time	1 - 5 - 10 s	Cycle time for new setpoints of a master control. See also P885 Synchronization ID SBus 1 / P895 Synchronization ID SBus 2 / P887 Synchronization ext. controller and P970 DPRAM synchronization.
970	DPRAM synchronization	On/off	MOVIDRIVE® B allows for synchronized operation with option cards (e.g. DHP11B, DFE24B). ON: Synchronized operation with option card is activated. **Important:** The inverters may either be synchronized by SBus1, SBus2 or by DPRAM. The inverters must **not be synchronized from interfaces at the same time**. SEW-EURODRIVE recommends to set P885/895 to an identifier that is not used in the entire CAN network. You need parameters P888 and P916 to implement synchronization with interpolating setpoint processing. OFF: Synchronized operation with the option card is not activated.
971	Synchronization phase	(-2) - 0 - 2 s	Time interval between clock signal and data transfer

6.3 Process and parameter access via fieldbus

The access to process and parameter data differs strongly depending on the bus systems and controller used.

What applies to most bus systems is that up to 10 process input data words (PI) with 16 bits each are sent to the controller and up to 10 process output data words (PO) with 16 bits each are sent from the controller to MOVIDRIVE® via a dual-port RAM. In addition, certain options (e.g. DFE24B) allow for cyclically reading and writing 8 IPOS$^{plus®}$ double words.

6.4 Other unit functions via fieldbus option card

The following additional functions can be used via fieldbus options in addition to the exchange of process and parameter data between controller and MOVIDRIVE®.

6.4.1 Using the fieldbus options in IPOS$^{plus®}$

Process data transmitted via fieldbus can be directly accessed in the IPOS$^{plus®}$ position and sequence control system integrated in MOVIDRIVE® using the commands GETSYS PO data and SETSYS PI data. Set the bus type to "3" for this purpose.

	**TIP**
	You find a detailed description of IPOS$^{plus®}$ in the "IPOS$^{plus®}$ Positioning and Sequence Control System" manual.

6.4.2 Engineering via fieldbus

In addition to accessing process and parameter data by the controller, some fieldbus options (e.g. DFP21B, DFE32B or DFE33B) also allow for parallel, **controller-independent** engineering access to MOVIDRIVE® via fieldbus. In this case, the engineering PC is directly connected to the bus system using an appropriate interface (e.g. PROFIBUS or Ethernet). The controller need not necessarily be in RUN mode for this purpose. The engineering access is described in the relevant manuals for the fieldbus options.

6.4.3 Engineering via fieldbus and controller

With some fieldbus options (e.g. DFP21B, DFE24B or DFI), access to MOVIDRIVE® is possible using an engineering PC through the controller via fieldbus in addition to the access to process and parameter data via controller.

For this purpose, the engineering PC is connected to an engineering interface of the controller (e.g. Ethernet interface of Siemens S7, of the EtherCAT or Interbus master) and then passes the telegrams of the MOVITOOLS® MotionStudio engineering software to the MOVIDRIVE® via fieldbus. The controller usually has to be in RUN mode to do so.

The engineering access via controller and fieldbus is described in the relevant manuals for the fieldbus options.

6.4.4 Diagnostics via WEB server

The fieldbus options for industrial Ethernet (DFE32B and DFE33B) have a web server that offers easy access to status and diagnostic information of the MOVIDRIVE®. On a diagnostics PC in the Ethernet network you can have a read access to diagnostic parameters, for example in the Internet Explorer by entering the IP address of MOVIDRIVE®.

Diagnostics via web server is described in manuals of the fieldbus options.

6.4.5 Motion control

With some fieldbus options (e.g. DFE24B EtherCAT) you can synchronize unit-internal time slices to an external communication cycle specified by the fieldbus. In this way, you can implement clock-synchronous transmission of process setpoints and actual values for motion control applications without aliasing effects.

Synchronization via fieldbus places high demands on controllers and MOVIDRIVE®. Refer to the fieldbus option manuals for the necessary settings.

7 SEW Unit Profile

MOVIDRIVE® offers digital access to all drive parameters and functions via the communication interfaces. The drive inverter is controlled via the fast, cyclical process data. Via this process data channel, you can enter setpoints, such as setpoint speed, ramp generator time for acceleration/deceleration, etc. as well as trigger various drive functions such as enable, controller inhibit, normal stop, rapid stop, etc. At the same time you can use this channel to read back actual values from the drive inverter, such as actual speed, current, unit status, error number or reference signals.

In combination with the IPOS$^{plus®}$ positioning and sequency control integrated in the drive inverter, you can also use the process data channel as direct connection between PLC and IPOS$^{plus®}$. In this case, the process data are not evalued by the drive inverter but directly by the IPOS$^{plus®}$ program.

While the process data exchange generally occurs cyclically, the drive parameters can be read or written acyclically using READ and WRITE services. This parameter data exchange enables you to implement applications in which all the important drive parameters are stored in the master programmable controller, so that there is no need to make parameter settings manually on the drive inverter itself.

The use of a fieldbus system requires additional drive system monitoring such as time monitoring of the fieldbus (fieldbus timeout) or even special emergency stop concepts. The MOVIDRIVE® monitoring functions can be customized to your application. You can determine, for instance, which of the drive inverter's error responses should be triggered in the event of a bus error. A rapid stop is a good idea for many applications, although this can also be achieved by "freezing" the last setpoints so the drive continues operating with the most recently valid setpoints (such as with a conveyor belt). As the functions of the control terminals are still active in fieldbus operation, you can still implement fieldbus-independent emergency stop concepts via the terminals of the drive inverter.

The MOVIDRIVE® inverter offers numerous diagnostic options for startup and service. For example, you can use the DBG60B keypad to control both setpoint values sent from the higher-level controller as well as the actual values. You are also supplied with a variety of additional information about the status of the communication interfaces. An even more convenient diagnostic option provides the MOVITOOLS® MotionStudio engineering software. It lets you set all drive and communication parameters and displays detailed information of interfaces and unit status.

7.1 Process data

Process data (PD) are all time-critical (realtime) data of a process that have to be processed or transmitted quickly. They are characterized by high dynamic properties and actuality. Process data are, for example, setpoints and actual values of the drive inverter, but also peripheral states of limit switches. Process data are exchanged cyclically between programmable controller and drive inverter.

The actual control of the MOVIDRIVE® inverter is carried out using process data.

Process input data (PI) and process output data (PO) are basically handled separately. This means you can specify for your application the kind of output data (setpoints) to be sent from the controller to the drive inverter and the process input data (actual values) to be sent by MOVIDRIVE® to the higher-level controller.

However, to control the drive inverter via communication interface, the inverter must first be switched to the relevant control signal source and setpoint source. Distinguishing between control signal and setpoint source allows for the most various combinations. For example, the drive can be controlled via fieldbus and uses the analog setpoint as setpoint. Next, the parameters for describing the process output data are used for informing the drive inverter how to interpret the received process data.

Parameter *P100 setpoint source* is used to specify the communication interface which the drive inverter uses for processing the setpoint.

Parameter	Communication interface
P100 Setpoint source	RS485
	Fieldbus
	SBus
	...

Parameter *P101 control signal source* is used to specify how the drive inverter is controlled. The inverter expects the control world of the source set in this parameter.

Parameter	Inverter control via
P101 Control signal source	Terminals
	RS485
	Fieldbus
	SBus

Setting: TERMINALS

With this setting, the drive inverter is controlled using only binary inputs and, if required, using the IPOS$^{plus®}$ control program.

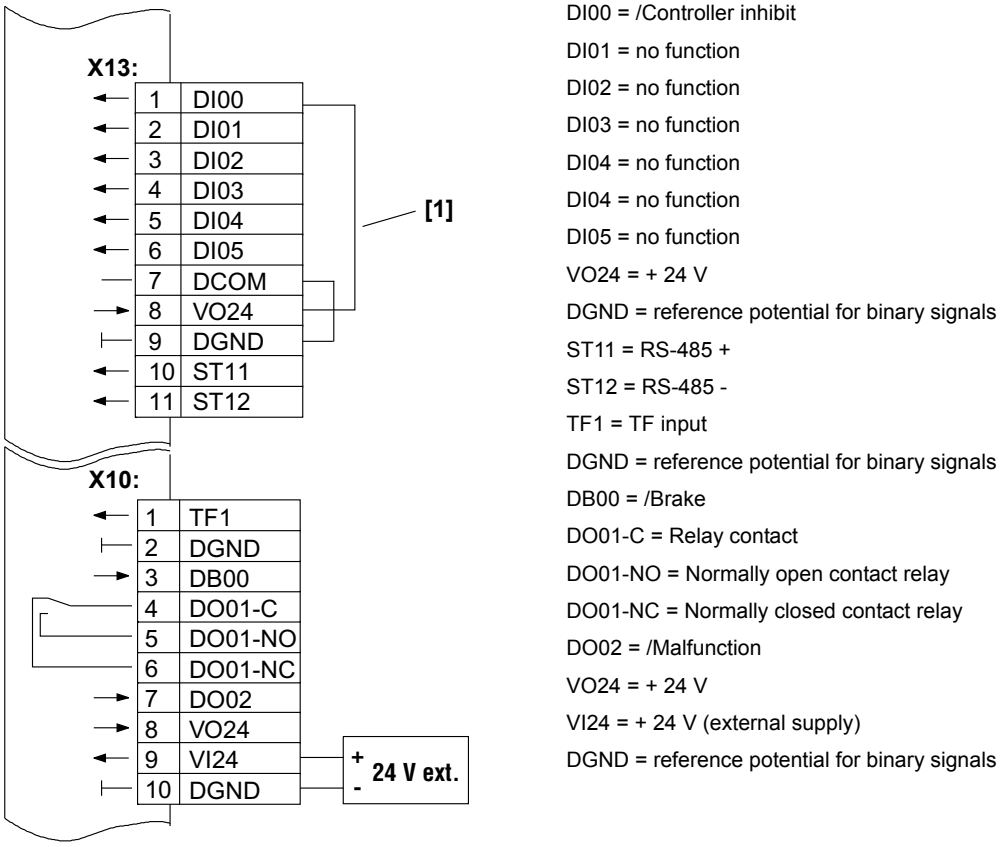
Setting: RS485, FIELDBUS, SBus

With this setting, the control word defined in the process output data channel is updated by the set control signal source (RS485 / FIELDBUS / system bus).

The binary inputs and the IPOS$^{plus®}$ control program continues to be involved in the control.

	NOTICE
	For safety reasons, you must also **always** enable the drive inverter at the terminals for control via process data. Consequently, you must wire or program the terminals in such a way that the inverter is enabled via the binary inputs.

For example, the simplest way of enabling the drive inverter at the terminals is to connect the DI00 binary input (function / *CONTROL INHIBIT*) to a +24 V signal and to program binary inputs DI01 through DI07 to *NO FUNCTION*. The following figure gives an example of terminal wiring and parameter setting for controlling the drive inverter solely using process data.

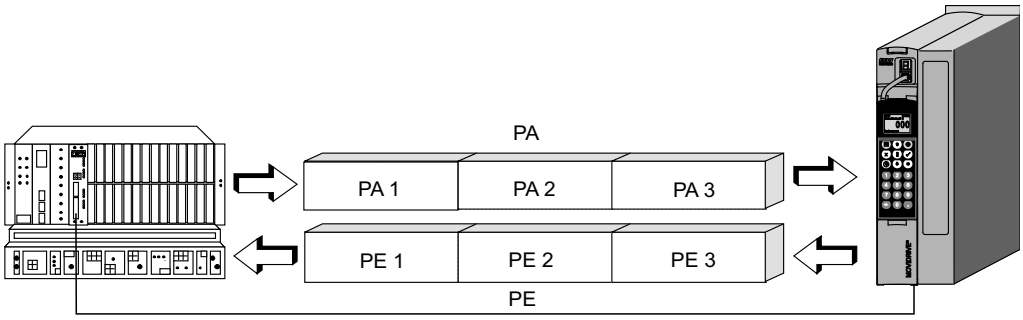


DI00 = /Controller inhibit
DI01 = no function
DI02 = no function
DI03 = no function
DI04 = no function
DI04 = no function
DI05 = no function
VO24 = + 24 V
DGND = reference potential for binary signals
ST11 = RS-485 +
ST12 = RS-485 -
TF1 = TF input
DGND = reference potential for binary signals
DB00 = /Brake
DO01-C = Relay contact
DO01-NO = Normally open contact relay
DO01-NC = Normally closed contact relay
DO02 = /Malfunction
VO24 = + 24 V
VI24 = + 24 V (external supply)
DGND = reference potential for binary signals

Enabling the power output stage using a device jumper [1]
01234BXX

7.2 Process data configuration

The MOVIDRIVE® inverter can be controlled with 1 to 10 (with RS485 with 1 to 3) process data words via the communication interfaces. The number of process input data (PI) and process output data (PO) is identical.



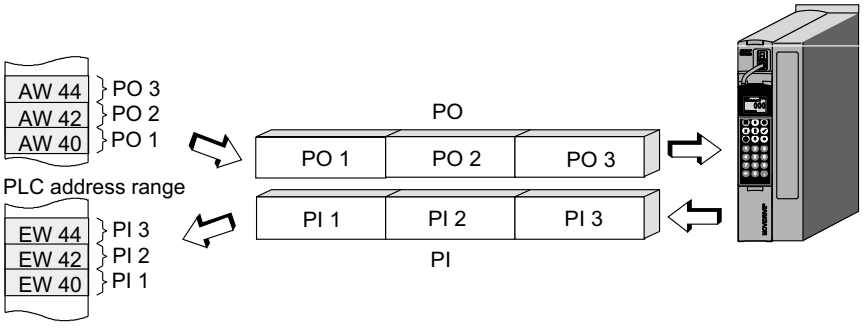
54943AXX

The process data configuration is set using DIP switches on the option card (e.g. DFI11B) or via the SBus master when starting up the bus system (e.g. PROFIBUS-DP or RS485). In this way, the inverter is automatically set properly. You can use the keypad or the MOVITOOLS® MotionStudio fieldbus monitor to check the current process data configuration under the menu item *P090 Fieldbus PD configuration*.

Depending on the installed fieldbus option card, the following process data configurations might take effect.

P090 PD configuration	
1 process data word + parameter channel	1PD+PARAM
1 process data word	1PD
2 process data words + parameter channel	2PD+PARAM
2 process data words	2PD
....	....
10 process data words + parameter channel	10PD+PARAM
10 process data words	10PD

Only the number of process data (that is 1 PD - 10 PD) is interesting for process data control of the drive inverter. These process data are usually mapped in the I/O or peripheral area when programmable logic controllers are used as fieldbus master. This means the I/O or peripheral area of the PLC must provide sufficient memory space for the drive inverter's process data (see following figure). The address between process data of the drive inverter and the PLC address area is usually assigned on the fieldbus master module.



55022AEN

7.3 Process data description

The process data description defines the content of the process data to be transmitted. The user can individually assign all the process data words.

The following 6 fieldbus parameters are available for defining the first three process data words:

- P870 Setpoint description PO1
- P871 Setpoint description PO2
- P872 Setpoint description PO3
- P873 Actual value description PI1
- P874 Actual value description PI2
- P875 actual value description PI3

If one of the above mentioned parameters is changed, acceptance of process output data for setpoint processing via fieldbus is automatically disabled. Only when the fieldbus parameter is activated again

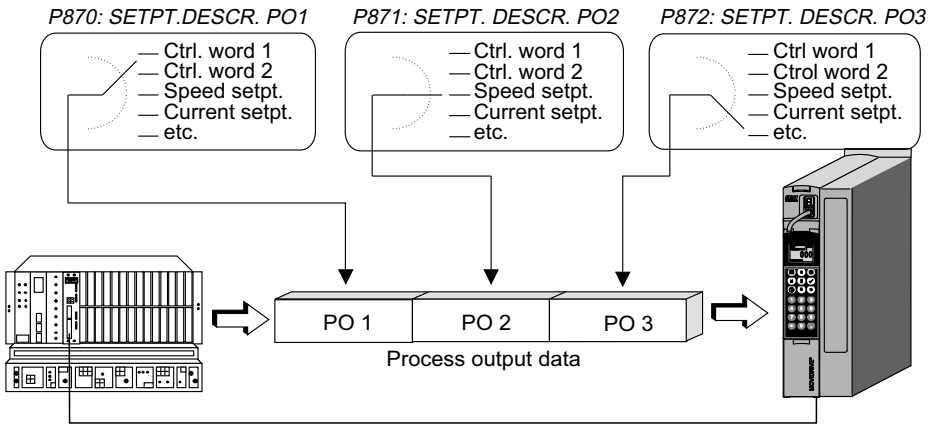
- P876 PO data enable = ON

will the received process output data be processed according to the new actual and setpoint value descriptions.

Process data words 4 to 10 can only be read and written using IPOS$^{plus®}$.

Setpoint description PO data

The parameters *setpoint description PAx* define the content of those process output data words which the higher-level programmable controller sends via the fieldbus system (see following figure).



55025AEN

You can use the process output data words PO1, PO2 and PO3 to transmit the mentioned setpoints across the process output data channel. You can decide yourself in which process data word the more significant part (high) or the less significant part (low) is transmitted.

Assignment	Meaning	Scaling
NO FUNCTION	The setting *NO FUNCTION* has the effect that the drive inverter does not use this process output data word for processing setpoints. The content of the process output data word programmed to *NO FUNCTION* is ignored although the controller might specifiy a real setpoint via the fieldbus system. The NO FUNCTION setting just disables the processing of the process output data word in the inverter system. However, you can access the process output data at any time using IPOS$^{plus®}$.	
SPEED	Set to *SPEED*, the *MOVIDRIVE®* inverter interprets the setpoint value transmitted by this process data word to be the speed setpoint if the selected operating mode (*P700 operating mode 1*, P701 operating mode 2) allows a speed setpoint. If no speed setpoint has been programmed although a communication interface (FIELDBUS, RS485, system bus) has been set as setpoint source, the inverter will use speed setpoint = 0.	1 digit = 0.2/min
CURRENT	Set to *CURRENT*, the drive inverter will interpret the setpoint specified in this process data word as current setpoint if a variant with torque control is set as operating mode (*P700 Operating mode 1*). Else, the drive inverter ignores the current setpoint.	1 digit = 0.1 % I_N
POSITION LO / HI	When set to *POSITION HI / POSITION LO*, the drive inverter passes the setpoint received via these process output data (usually a position setpoint) as 32-bit value directly to the IPOS$^{plus®}$ program in IPOS$^{plus®}$ variable *499 SP.PO.BUS (setpoint position bus)*. The position setpoints must be split into two process data words because the position is usually specified as signed 32-bit value. This means you have to specify the higher-value position value *(POSITION HI)* as well as the lower-value position setpoint *(POSITION LO)*. Else, the drive inverter will not accept these process output data in the IPOS$^{plus®}$ program.	
MAX. SPEED	Set to *MAX. SPEED* means the MOVIDRIVE® inverter interprets the transmitted setpoint as speed limit. The speed limit is specified in rpm and is interpreted as value for both directions of rotation. The supported value range of the speed limit via fieldbus corresponds to the value range of parameter *P302 Maximum speed 1*. Specifying the speed limit via fieldbus automatically disables parameters *P302 Maximum speed 1, P312 maximum speed 2*.	1 digit = 0.2/min
MAX. CURRENT	Set to *MAX. CURRENT* means the MOVIDRIVE® inverter interprets the transmitted process output data as current limit. The current limit is specified in percent with reference to the rated inverter current, in the unit % I_N and is interpreted as value for both directions of rotation. The supported value range of the current limit via fieldbus corresponds to the value range of parameter *P303 Current limit 1*. The current limits that can be set using parameters *P303 Current limit 1* and *P313 Current limit 2* are still valid when the current limit is specific using process data. This means these parameters are to be regarded as maximum effective current limit.	1 digit = 0.1 % I_N

Assignment	Meaning	Scaling
SLIP	Set to *SLIP* means the MOVIDRIVE® inverter interprets the transmitted process output data word as slip compensation value. Specifying the slip compensation via fieldbus automatically disables parameters *P324 Slip compensation 1* and *P334 Slip compensation 2*. Specifying the slip compensation via process data channel is only technically meaningful in the *VFC N-CONTROL* operating mode because the torque can be influenced directly by changing the slip compensation. The value range of this slip compensation value is identical with the value range of parameter *P324 Slip compensation 1* and corresponds to a speed range of 0 - 500 rpm. If the slip specified using process data is outside this value range, the maximum will take effect when the minimum and maximum values are exceeded.	1 digit = 0.2 / min
RAMP	Set to *RAMP*, the MOVIDRIVE® inverter considers the transmitted setpoint value to be an acceleration or deceleration ramp. The specified value corresponds to a time in ms and refers to a speed change of 3000 rpm. The rapid stop and emergency stop function is not affected by this process ramp. When transmitting the process ramp via the fieldbus system, ramps t11, t12, t21 and t22 become ineffective.	1 digit = 1 ms
CONTROL WORD 1 / CONTROL WORD 2	The assignment of process output data with control word 1 or 2 allows for activating nearly all the drive functions via fieldbus system. For a description of control words 1 and 2, please refer to the chapter "Control word definition".	
SPEED [%]	Set to *SPEED [%]* means the MOVIDRIVE® inverter interprets the setpoint transmitted in this process data word as speed setpoint in percent. The relative speed setpoint always refers to the currently applicable maximum speed limit, which means either P302/312 or MAX. SPEED or PO speed limit.	4000_{hex} = 100 % n_{max}
IPOS PO-DATA	The setting *IPOS PO-DATA* has the effect that the drive inverter does not use this process output data word for processing setpoints. The inverter system ignores the content of the process output data word programmed to *IPOS-PO-DATA* and is available for sole processing in the IPOS$^{plus®}$ control program. Within IPOS$^{plus®}$, you can use the command *GetSys PO-Data* to directly access the process output data of the communciation interfaces. For more detailed information, refer to the IPOS$^{plus®}$ positioning and sequence control system manual.	Three words with individually coded 16 bits each can be exchanged between the higher-level controller and IPOS$^{plus®}$.

Special cases of PO data processing

Setting the process output data description separately allows for a great variety of combinations. Not all of them are technically meaningful, however.

In addition to the process output data, the digital input terminals are generally also used. In special cases, also the analog setpoint of the MOVIDRIVE® inverter is used.

Setpoint specification via fieldbus missing	If a communication interface is entered as setpoint source and if no setpoint is programmed for the process output data description, then the setpoint = 0 is generated in the inverter.
No control word specification via fieldbus	If a communication interface is entered as control signal source and if no control word is programmed for the process output data description, then the ENABLE control command is specified in the inverter.
Double assignment of the process output data channel	If several process output data words have the same setpoint description, only the process output data word that is read first will apply. The order in which the process output data words are processed in the inverter is PO1 - PO2 - PO3. This means if PO2 and PO2 have the same setpoint description, only PO2 will take effect. The content of PO3 is ignored.

32-bit process output data

Process data that are longer than 16 bits and therefore occupy more than one process data word, will not be processed by the drive inverter until they are fully mapped on the process data channel.

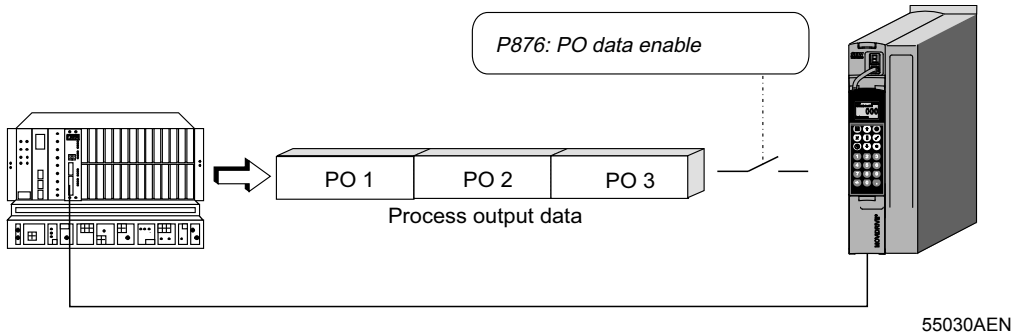
NOTICE

Position setpoints have to be transferred consistently.

Possible consequences: The drive inverter might move to undefined positions because, for example, an old position setpoint low and an already new position setpoint high would be effective at the same time.

For more information on how to ensure data consistency and the resulting programming techniques, refer to the project planning manual of the master interface module of your programmable controller.

Enable PO data



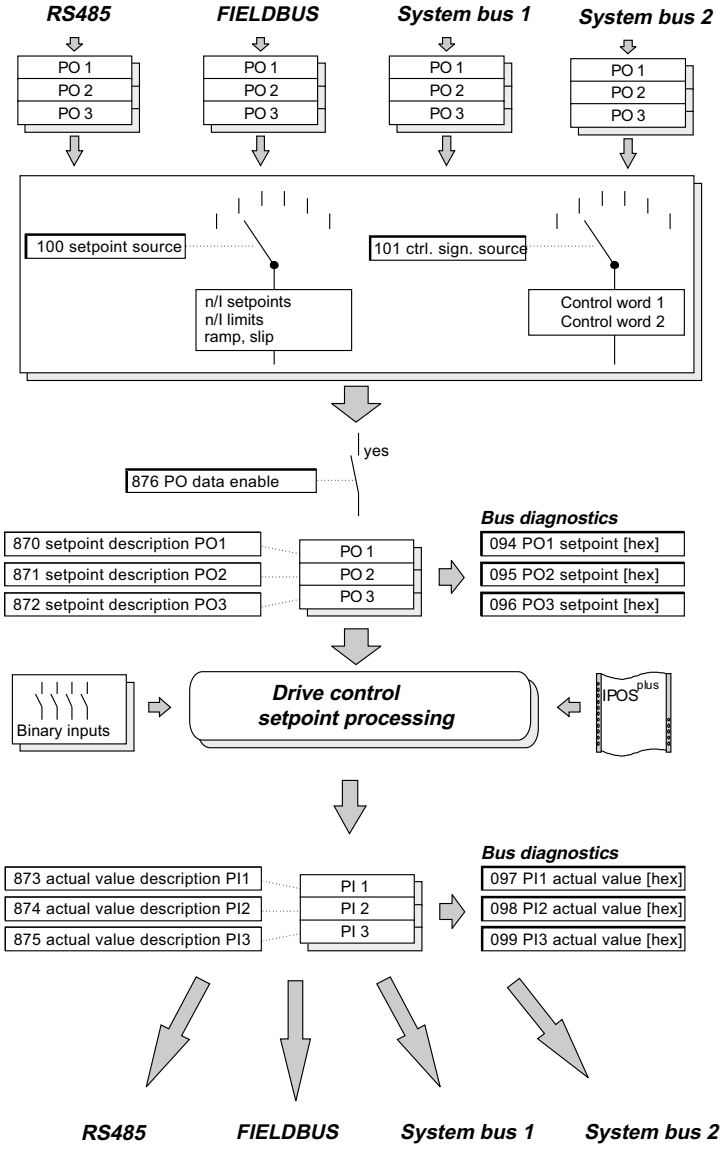
55030AEN

Changing the parameter *Setpoint description PO1 - PO3* causes the automatic disabling of process output data with *PO data enable = No*. The process output data channel is not available for processing until the parameter *PP data enable is set to YES* (e.g. by the higher-level controller).

NO	Process output data deactivated. Setpoint processing of the drive inverter continues until the fieldbus setpoints are activated again with the last valid (frozen) process output data.
YES	Process output data enabled. The drive inverter uses the process output data specified by the master.

PO/PI data processing

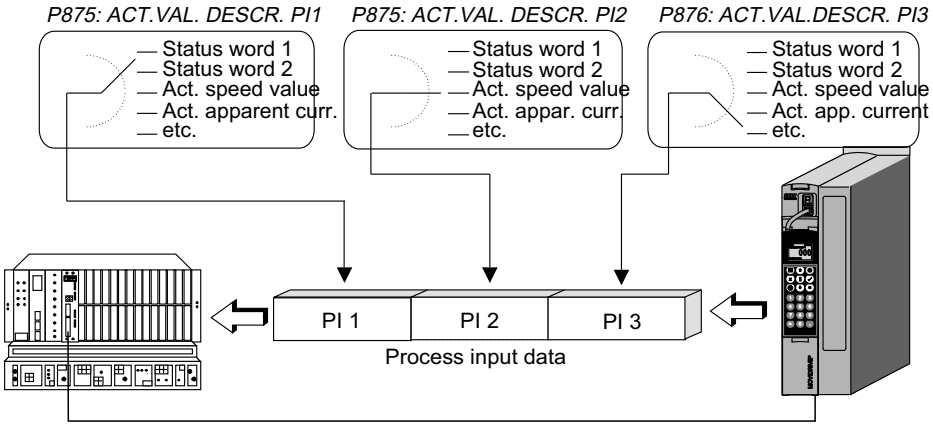
The process input data of the inverter (actual values, condition information, etc.) can be read by all communication interfaces of the inverter and is therefore not connected with the control signal and setpoint source.



63787AEN

Actual value description of PI data

The parameters *Actual value description PI1 - PI3* define the content of the process input data words transmitted by the drive inverter to the higher-level controller (see figure below). Each process data word is defined with an individual parameter. Therefore, three parameters are necessary for describing the process input data.



55029AEN

The following parameters can be transmitted across the process data channel using process input data words PI1 to PI3. 32-bit values, such as the actual position, are transmitted in two process data words. You can decide yourself in which process data word the more significant part (high) and the less significant part (low) is transmitted.

Assignment	Meaning	Scaling
NO FUNCTION	Assigning a process input data word with *NO FUNCTION* means that the inverter system does not update this process input data word. In this case, MOVIDRIVE® returns the value 0000hex to the higher-level controller.	
SPEED	Set to *SPEED*, the drive inverter returns the current actual speed in rpm to the higher-level automation system. The actual speed can only be sent back properly if the inverter can determine the actual motor speed via speed feedback. For applications with slip compensation, the deviation from the real motor speed solely depends on the accuracy of the slip compensation set by the user.	1 digit = 0.2 / min
OUTP.CURRENT	With the setting *OUTPUT CURRENT*, the drive inverter returns the current actual value of the output current in [% I_n] to the higher-level automation system (in percent, with reference to the rated current of the drive inverter).	1 digit = 0.1 % I_N
ACTIVE CURRENT	By assigning a process input word *ACTIVE CURRENT*, the inverter provides the actual active current value in % I_N to the higher-level automation system.	1 digit = 0.1 % I_N
POSITION LO / HI	The actual position values must be divided into two process data words because the position is transmitted as integer32. This means you have to specify both the *actual position value high* and the *actual position value low*. The drive inverter only provides valid actual position values in operating modes with speed feedback.	
STATUS WORD 1/ STATUS WORD 2	Assigning status word 1 or status word 2 to the process input data allows for accessing status information as well as fault and reference messages.	
SPEED [%]	Set to *SPEED [%]*, the drive inverter returns the current actual speed in % n_{max} / P302 to the higher-level automation system.	4000_{hex} = 100 % n_{max}

Assignment	Meaning	Scaling
IPOS PI-DATA	Set to *IPOS PI* (IPOS Process Input Data), an individual actual value can be transmitted from the IPOS$^{plus®}$ program to the higher-level controller via process input data. This setting allows for exchanging up to 48 individually coded bits between the IPOS$^{plus®}$ program and the higher-level controller using the process data channel. You can directly write process input data in IPOS$^{plus®}$ using the command *SetSys PI-Data*. For more detailed information, refer to the IPOS$^{plus®}$ positioning and sequency control system manual.	Three words with individually coded 16 bits each can be exchanged between the higher-level controller and IPOS$^{plus®}$.

Scaling of process data

The process data are always transmitted as fixed-point values to make for simple calculation in the ongoing system process. Parameters with identical units of measurement receive the same scaling so that the higher-level automation device can directly compare the set and actual values in the application program. There are the four different process data types:

- Speed in rpm
- Current in % I_N (rated current)
- Ramp in ms
- Position in increments.

The different versions of the control or status word are coded as bit field and will be described in a separate chapter.

Process data	Type	Resolution	Reference	Range
Speed setpoint / Actual speed value / Speed limiting slip compensation	Integer 16	1 digit = 0.2 rpm		-6553.6 ... 0 ... +6553.4 rpm 8000_{hex} ... 0 ... $7FFF_{hex}$
Relative speed setpoint [%] / Relative actual speed value [%]	Integer 16	1 digit = 0.0061 % (4000_{hex} = 100 %)	Maximum speed of the inverter	- 200 % ... 0 ... + 200 % - 0.0061 % 8000_{hex} ...0 ... $7FFF_{hex}$
Apparent current actual value / Actual active current value / Current setpoint Current limitation	Integer 16	1 digit = 0.1 % I_N	Rated current of the drive inverter	-3276.8 % .... 0 ..... +3276.7 % 8000_{hex} .... 0 ..... $7FFF_{hex}$
Process ramp up / Process ramp down	Unsigned 16	1 digit = 1 ms	delta-f = 100 Hz	0 ms ... 65535 ms 0000hex ... $FFFF_{hex}$
Actual position value / Position setpoint	Integer 32	1 motor revolution = 4096 increments, i.e. 1 digit = 360°/4096		-188.743.680° .... 0 ..... +188.743.679° -524 288 .... 0 .... +524287 motor revolutions 8000 0000_{hex} ... 0 ... 7FFF $FFFF_{hex}$ high low high low

Positive speed values correspond to a CW rotation with proper connection of the motor, or to CW = UPWARDS for hoist applications.

NOTICE

Consistent handling of the two process output data words for positions.

Possible consequences: The servo inverter might move to undefined positions because, for example, an old position setpoint low and an already new position setpoint high would be effective at the same time.

When handling position setpoints in the application program of the higher-level automation controller make sure that the two process output data used for transmitting the position are handled consistently. This means that the position setpoint high is always transmitted together with the position setpoint low.

Examples

Process data	Value	Scaling	Transmitted process date
Speed	CW 400 rpm	400/0.2 = 2000_{dec} = $07D0_{hex}$	2000_{dec} or $07D0_{hex}$
	CCW 750 rpm	- (750/0.2) = 3750_{dec} = $F15A_{hex}$	-3750_{dec} or $F15A_{hex}$
Relative speed	CW 25 % f_{max}	25 x (16384/100) = 4096_{dec} = 1000_{hex}	4096_{dec} or 1000_{hex}
	CCW 75 % f_{max}	-75 x (16384/100) = -12288_{dec} = $D000_{hex}$	-12288_{dec} or $D000_{hex}$
Current	45 % I_N	(45/0.1) = 450_{dec} = $01C2_{hex}$	450_{dec} or $01C2_{hex}$
	115.5 % I_N	(115.5/0.1) = 1155_{dec} = 0483_{hex}	1155_{dec} or 0483_{hex}
Ramp	300 ms	300 ms → 300_{dec} = $012C_{hex}$	300_{dec} or $012C_{hex}$
	1.4 s	1.4 s = 1400 ms → 400_{dec} = 0578_{hex}	1400_{dec} or 0578_{hex}
Position	35 rev. CCW	-35 x 4096 = - 143360_{dec} = FFFD $D000_{hex}$	FFFD $D000_{hex}$ high low
	19 rev. CW	19 x 4096 = 77824_{dec} = 0001 3000_{hex}	0001 3000_{hex} high low

7.4 Sequence control

7.4.1 Definition of the control word

The control word is 16 bits wide. Each bit has been assigned a drive inverter function. The low byte consists of eight fixed function bits that are always valid. The assignment of the more significant control bits varies for the different control words.

Functions that are generally not supported by the drive inverter cannot be activated by the control word. The individual control word bits are considered as reserved and must be set to logical 0 by the user.

Basic control block

The less-significant part of the control word (bits 0 to 7) contains 8 fixed function bits for the most important drive functions. The following overview shows the assignment of the basic control block.

Bit	Function
0	Controller inhibit = "1" / Enable = "0"
1	Enable = "1" / Rapid stop = "0"
2	Enable = "1" / Stop = "0"
3	Hold control: Not active = "1" / Active = "0"
4	Ramp generator selection: Integrator 1 = "1" / Integrator 2 = "0"
5	Parameter set switchover: Parameter set 2 = "1" / Parameter set 1 = "0"
6	Reset: Reset pending fault = "1" / Not active = "0"
7	Reserved
8	Depends on control word
9	
10	
11	
12	
13	
14	
15	

7.4.2 Linking safety-relevant control commands

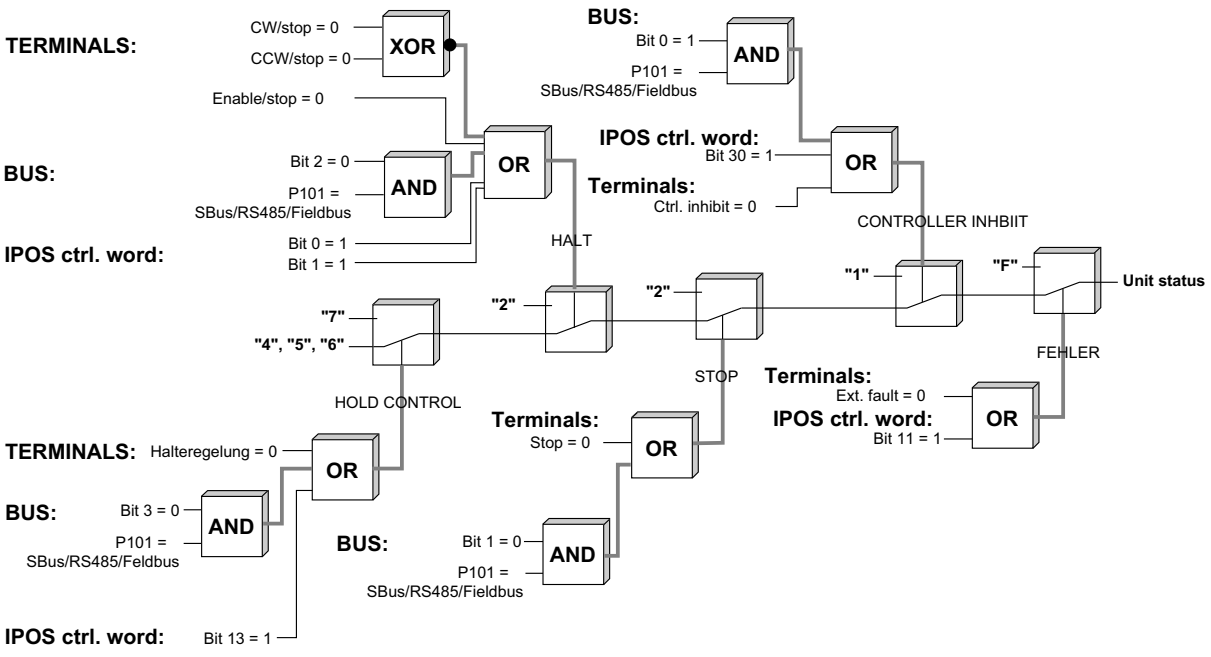
Basically, the control commands

- CONTROLLER INHIBIT
- RAPID STOP / STOP
- STOP
- HOLD CONTROL
- ENABLE

can be activated simultaneously using the set control signal source, the binary inputs, and the IPOS$^{plus®}$ control program. To link these control functions under safety-relevant aspects, the individual control commands are prioritized. For enabling the drive inverter, for example, the following figure shows that all three processing blocks (terminal processing, control word processing, and IPOS$^{plus®}$ program) have to generate the eneable. As soon as one of the three processing blocks triggers a control command of higher priority (such as *STOP* or *CONTROLLER INHIBIT*), the higher prioritized command will become effective.

After switching on the drive inverter, IPOS$^{plus®}$ generally issues the control command *ENABLE* so that the drive can be controlled immediately even without the IPOS$^{plus®}$ program.

The binary inputs will generally remain active even if the inverter is controlled via the process data *(P101 control signal source = RS485/FIELDBUS/SBus)*. Safety-relevant functions, such as controller inhibit and enable are equally processed by the terminal strip and the fieldbus. This means the drive inverter must have first been enabled at the terminals to control via fieldbus. All other functions that can be activated via terminals and the control word, will be processed with an OR function. The following figure shows the unit status (7-segment display) depending on the various control signal sources (terminals, bus, or IPOS control word).



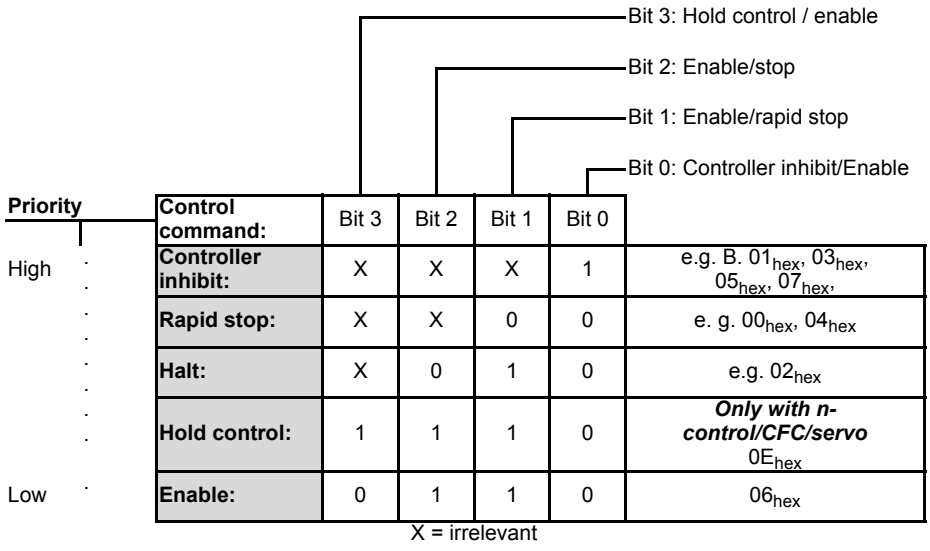
65058AEN

For safety reasons, the basic control block is defined in such a way that the inverter with the control word setting 0000_{hex} will assume the state *No enable* because all commercially available fieldbus master systems will reset the ouputs to 0000_{hex} in the event of an error. In this case, the inverter will execute a rapid stop and then activate the mechanical brake.

7.4.3 Control commands

Controlling the inverter with bits 0 - 3

If the inverter was enabled at the terminals, it can be controlled with bits 0 - 2, or bits 0 - 3 for applications with speed feedback of the basic control block.



Priority	Control command:	Bit 3	Bit 2	Bit 1	Bit 0	
High	**Controller inhibit:**	X	X	X	1	e.g. B. 01_{hex}, 03_{hex}, 05_{hex}, 07_{hex},
	Rapid stop:	X	X	0	0	e. g. 00_{hex}, 04_{hex}
	Halt:	X	0	1	0	e.g. 02_{hex}
	Hold control:	1	1	1	0	***Only with n-control/CFC/servo*** $0E_{hex}$
Low	**Enable:**	0	1	1	0	06_{hex}

X = irrelevant

Control command "Controller inhibit"

You can disable the power output stages of the inverter and set them to high impedance using the control command *Controller inhibit*. At the same time, the inverter activates the mechanical motor brake so that the drive will immediately come to a standstill through mechanical braking. Motors without mechanical brake will coast to standstill when using this control command.

You activate the control command *Controller inhibit* by setting *bit 0: Controller inhibit/enable* in the control word because all other bits are not important. This setting will assign the highest priority to this control bit in the control word.

Control command "Rapid stop"

Using the *rapid stop* control command lets you have the inverter brake the motor at the currently applicable rapid stop ramp. The following rapid stop ramps set via parameters basically take effect:

- P136 T13 stop ramp (with active parameter set 1)
- P146 T23 stop ramp (with active parameter set 2)

The process ramp that may have been set via fieldbus does not affect the rapid stop.

The control command is activated by resetting *bit 1: Enable/rapid stop*.

Control command "Stop"

The control command **Stop** will have the inverter brake the motor at the stop ramp. If the process ramp is transmitted via the fieldbus system, this control command will use the currently indicated ramp value as brake ramp. Else, the drive inverter uses the typical integrator ramp for this control command depending on the set parameters and integrator set.

The control command *stop* is triggered with *bit 2: Enable/stop*.

Control command "Enable"

You enable the inverter via the fieldbus system with the control command **Enable**. If the process ramp is transmitted via the fieldbus system, this control command will use the currently indicated ramp value as brake ramp. Else, the drive inverter uses the typical integrators *ramp up* for this control command depending on the set parameters and integrator set.

All three bits must be set to **Enable** (110_{bin}) for the control command *Enable*.

Control command "Hold control"

Setting bit 3 to the value "1" lets you activate the *hold control* function is speed-controlled operation. The function triggers a stop at the applicable integrator ramp with subsequent hold control. In operating modes without speed feedback, this bit is not relevant and the function is not activated.

Selecting the valid parameter set

The applicable parameter set is selected using bit 5 in the control word. A parameter set can only be changed in *controller inhibit* condition.

This bit is ORed with the input terminal function *parameter set changeover*. This means the logical state "1" of the input terminal OR the control word bit activates parameter set 2.

Reset after an error

In case of an error, bit 6 of the control word will execute a reset via the process data channel. A reset can only be triggered with a a 0/1 edge in the control word.

7.4.4 Control word 1

Control word 1 includes the most important drive functions of the basic control block as well as the function bits for setpoint functions that are generated in the MOVIDRIVE® inverter in the higher-order byte.

Bit	Functionality	Assignment
0	Fixed definition	Controller inhibit "1" / Enable "0"
1		Enable "1" / Rapid stop "0"
2		Enable "1" / Stop "0"
3		Hold control
4		Integrator switchover
5		Parameter set switch-over
6		Reset
7		Reserved
8	Direction of rotation for motor potentiometer	0 = CW direction of rotation 1 = CCW direction of rotation
9 10	Motor potentiometer acceleration Motor potentiometer deceleration	10 9 0 0 = no change 1 0 = down 0 1 = up 1 1 = no change
11 12	Selection of the internal fixed setpoints n11 - n13 or n21 - n23	12 11 0 0 = Speed setpoint via process output data word 2 0 1 = Internal setpoint n11 (n21) 1 0 = Internal setpoint n12 (n22) 1 1 = Internal setpoint n13 (n23)
13	Fixed setpoint switchover	0 = Fixed setpoints of the active parameter set selectable via bit 11/12 1 = Fixed setpoints of the other parameter set selectable via bit 11/12
14	Reserved	Set reserved bits to zero.
15	Reserved	Set reserved bits to zero.

These internal setpoint functions are activated by setting parameter P100 to fixed setpoint or motor potentiometer and setting the matching bits in control word 1. Any speed setpoint entered via an SBus process output data word will no longer be effective!

Motor potentiometer via fieldbus

The setpoint function motor potentiometer is controlled via the fieldbus interface in the same way as with the standard input terminals. The process ramp that may be entered via an additional process output data word has no effect on the motor potentiometer function. Only the following motor potentiometer integrators will be used.

- P150 T3 Ramp up
- P151 T4 Ramp down

7.4.5 Control word 2

Control word 2 contains the function bits for the most important drive functions in the basis control block; the virtual input terminals in the higher-order part. These are freely-programmable input terminals that are not physically available due to missing hardware (option cards). In this way, the input terminals are represented on the virtual input terminals of the fieldbus. Each virtual terminal is assigned to an optional and **physically unavailable** input terminal. Its functionality can be programmed as required.

Bit	Function	Definition
0	Controller inhibit "1" / Enable "0"	Fixed definition
1	Enable "1" / Rapid stop "0"	
2	Enable "1" / Stop "0"	
3	Hold control	
4	Integrator switchover	
5	Parameter set switch-over	
6	Reset	
7	Reserved	
8	Virtual terminal 1 = P610 / Binary input DI10	Virtual input terminals
9	Virtual terminal 2 = P611 / Binary input DI11	
10	Virtual terminal 3 = P612 / Binary input DI12	
11	Virtual terminal 4 = P613 / Binary input DI13	
12	Virtual terminal 5 = P614 / Binary input DI14	
13	Virtual terminal 6 = P615 / Binary input DI15	
14	Virtual terminal 7 = P616 / Binary input DI16	
15	Virtual terminal 8 = P617 / Binary input DI17	

NOTICE

If both the fieldbus option card and DIO11 are inserted in the drive inverter, the inputs of the DIO11 option have priority. In this case, the virtual inputs are not evaluated.

7.4.6 Status word definition

The status word is 16 bits wide. The less significant byte, the basic status block, consists of eight status bits with fixed definition that reflect the most important drive states. The assignment of the more significant status bits varies for different status words.

Basic status block

The basic status block of the status word contains the condition information required for nearly any drive application.

Bit	Function / assignment	Definition
0	Output stage enabled "1" / Output stage inhibited "0"	Fixed definition
1	Inverter ready "1" / Inverter not ready "0"	
2	PO data enabled "1" / PO data disabled "0"	
3	Current ramp generator set: Integrator 2 "1" / Integrator 1 "0"	
4	Current parameter set: Parameter set 2 "1" / Parameter set 1 "0"	
5	Fault/warning: Fault/warning pending "1" / No fault "0"	
6	CW limit switch active "1" / CW limit switch inactive "0"	
7	CCW limit switch active "1" / CCW limit switch inactive "0"	

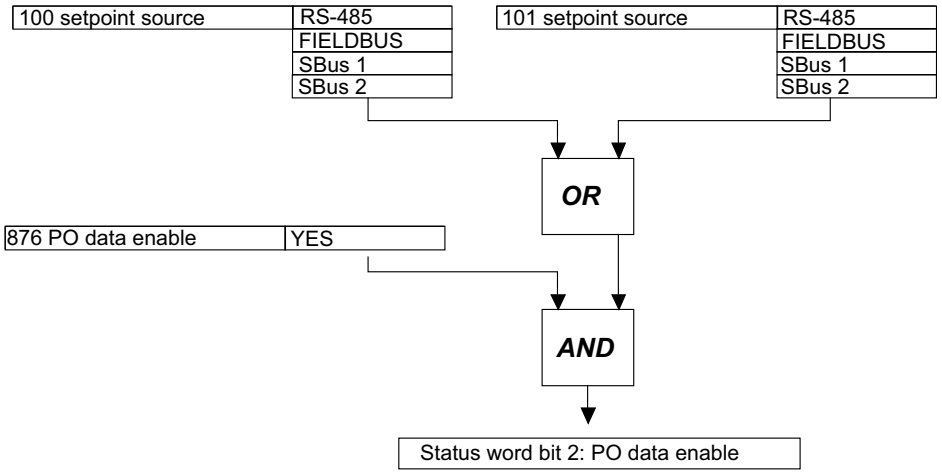
Signal "inverter ready"

The value *inverter ready = 1* in status bit 1 of the status word indicates that the inverter is ready to respond to control commands from an external control. The inverter is not ready, if

- MOVIDRIVE® signals a fault
- the factory setting is active (setup)
- no supply voltage is present

Signal "PO data enabled"

With *PO data enabled = 1*, bit 2 signals that the inverter responds to control values and setpoints from the communication interfaces. The following figure shows the conditions that have to be met for the PO data to be enabled:



54681BEN

Fault/warning

With bit 5 in the status word, the inverter signals a possible fault or warning. The result of a **fault** is always that the inverter is no longer ready for operation. A **warning**, however, can occur temporarily without affecting the operating behavior of the inverter. Therefore, you should also evaluate status bit 1 **inverter ready** (requirement: mains voltage ON).

Bit 1: Ready	Bit 5: Fault/warning	Meaning
0	0	Inverter not ready for operation
0	1	Malfunction
1	0	Inverter is ready for operation
1	1	Warning

Limit switch processing

Limit switch processing is active if two input terminals of the inverter are programmed to *CW limit switch* or *CCW limit switch*. In this way, the higher-level controller is informed about the current status of the limit switches so it is able to specify the travel process in opposite direction. The terminal signals of the limit switches are 0-active while the condition of the limit switches is indicated in the drive inverter as 1-active.

7.4.7 Status word 1

Status word 1 contains the status information in the basic status block and the **unit status** or the **fault number** in the higher-level status byte. Depending on the fault bit, the unit status is displayed for fault bit = 0 or for problems (fault bit = 1), the error number. The fault bit is reset by resetting the fault and the current unit status is displayed. The meaning of the fault numbers and the unit status is described in the system manual or in the MOVIDRIVE® MDX60B/61B operating instructions.

Bit	Function	Definition
0	Output stage enabled	Fixed definition
1	Inverter ready	
2	PO data enabled	
3	Current ramp generator set	
4	Current parameter set	
5	Fault/warning	
6	Limit switch CW active	
7	Limit switch CCW active	
8	**Fault/warning?** Bit 5 = 1 → fault number: 01 Overcurrent 02 ... Bit 5 = 0 → Unit status: 0x1 Controller inhibit 0x2 ...	Unit status/Fault number
9		
10		
11		
12		
13		
14		
15		

7.4.8 Status word 2

Status word 2 contains both the status information in the basis status block and the virtual output terminals DO10 - DO17 in the higher-level byte. By programming the terminal functions for the output terminals, all the conventional signals can be processed via the fieldbus system.

Bit	Function	Definition
0	Output stage enabled	Fixed definition
1	Inverter ready	
2	PO data enabled	
3	Current ramp generator set	
4	Current parameter set	
5	Fault/warning	
6	Limit switch CW active	
7	Limit switch CCW active	
8	Virtual terminal 1 = P630 / Binary output DO10	Virtual output terminals
9	Virtual terminal 2 = P631 / Binary output DO11	
10	Virtual terminal 3 = P632 / Binary output DO12	
11	Virtual terminal 4 = P633 / Binary output DO13	
12	Virtual terminal 5 = P634 / Binary output DO14	
13	Virtual terminal 6 = P635 / Binary output DO15	
14	Virtual terminal 7 = P636 / Binary output DO16	
15	Virtual terminal 8 = P637 / Binary output DO17	

NOTICE

If both the fieldbus option card and DIO11 are inserted in the drive inverter, the inputs of the DIO11 option have priority. In this case, the virtual inputs are not evaluated.

7.4.9 Status word 3

Status word 3 contains the IPOS$^{plus®}$ status messages for positioning tasks in addition to the condition information in the basic status block. The *unit status* or the *fault number* is indicated in the more significant status byte. Depending on the fault bit, the unit status is displayed for fault bit = 0 or for problems (fault bit = 1), the error number. The fault bit is reset by resetting the fault and the current unit status is displayed.

Bit	Function	Definition
0	Motor is turning	Fixed definition
1	Inverter ready	
2	IPOS reference	
3	IPOS in position	
4	Brake released	
5	Fault/warning	
6	Limit switch CW active	
7	Limit switch CCW active	
8	**Fault/warning?** Bit 3 = 1 → fault number: 01 Overcurrent 02 ... Bit 3 = 0 → Unit status: 0x1 Controller inhibit 0x2 ...	Unit status/Fault number
9		
10		
11		
12		
13		
14		
15		

7.4.10 Fault number and unit status

TIP

You find a current list with fault numbers and unit states in the parameter directory matching the firmware of your units. For more detailed information, refer to the operating instructions and the MOVIDRIVE® MDX60B/61B system manual.

Unit status

The 7-segment display shows the operating condition of MOVIDRIVE® and, in the event of an error, an error or warning code.

7-segment display	Unit status (high byte in status word 1)	Meaning
0	**0**	24 V operation (inverter not ready)
1	**1**	Controller inhibit active
2	**2**	No enable
3	**3**	Standstill current
4	**4**	Enable
5	**5**	n-control (speed control)
6	**6**	M-control (torque control)
7	**7**	Hold control
8	**8**	Factory setting
9	**9**	Limit switch contacted
A	**10**	Technology option
c	**12**	IPOS$^{plus®}$ reference travel
d	**13**	Flying start
E	**14**	Calibrate encoder
F	**Fault number is indicated in the status word**	Fault indicator (flashing)
H	**The actual unit status is indicated**	Manual operation
t	**16**	Inverter is waiting for data
U	**17**	"Safe Stop" active
• (blinking dot)	-	IPOS$^{plus®}$ program is running
Flashing display	-	STOP via DBG60B
⊐1 ...⊐9	-	RAM defective

WARNING

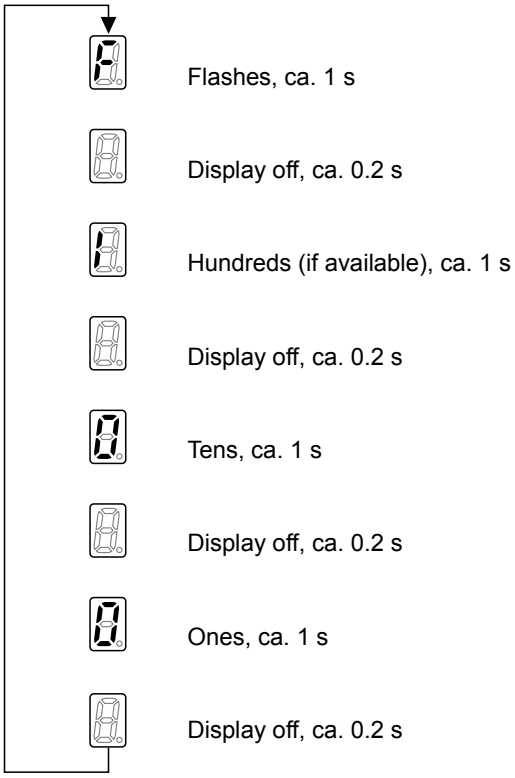
Incorrect interpretation of display U = "Safe stop" active.

Severe or fatal injuries.

The display U = "Safe stop" active is not safety-related and may not be used as a safety function!

Fault number (fault code)

The fault code is shown in a 7-segment display. The following display sequence is used (e.g. fault code 100):



Following a reset or if the fault code resumes the value "0", the display switches to the operating display.

The suberror code is displayed in MOVITOOLS® (as of version 4.50) or in the DBG60B keypad.

7.5 Monitoring functions

For safe operation of the MOVIDRIVE® inverter via the communication interfaces, additional monitoring functions have been implemented that may trigger an operator-defined drive function in case of a bus error. Two individual parameters are available for each communication interface.

- Timeout interval
- Timeout response

This parameter defines an application-specific drive behavior in case of a communication error.

Timeout error message / timeout interval / timeout response

The inverter generates a timeout if no new data are received via the bus system within a preset time frame (timeout interval). The timeout response can be set and defines the malfunction (fault/warning) and the fault response of the drive.

Timeout error message

MOVIDRIVE® generates a separate **timeout error message for every communication interface**:

Communication interface	Fault number	Timeout error message
Fieldbus	F 28	F-BUS TIMEOUT
RS485	F 43	RS485 TIMEOUT
SBus 1	F 47	SBUS 1 TIMEOUT
SBus 2	F 46	SBUS 2 TIMEOUT

NOTICE

Both RS485 interfaces are monitored together. This means if a DBG60B keypad is connected to the XT port, monitoring for whether cyclical telegrams continue to be received via the second RS485 interface is no longer possible.

Timeout interval

The **timeout interval** can be set individually for each communication interface.

Communication interface	Parameter number	Parameter name	Timeout interval
Fieldbus	819	Fieldbus timeout interval	0.50 seconds
RS485	812	RS485 Timeout interval	0.00 seconds
SBus 1	883	SBus 1 Timeout interval	0.10 seconds
SBus 2	893	SBus 2 Timeout interval	0.10 seconds

Timeout response

The **timeout response** can be set individually for each communication interface.

Parameter number	Parameter name	Timeout response
831	Response FIELDBUS TIMEOUT	RAPID STOP/WARN.
833	Response RS485 TIMEOUT	RAPID STOP/WARN.
836	Response to SBus1 TIMEOUT	RAPID STOP/WARN.
837	Response to SBus2 TIMEOUT	RAPID STOP/WARN.

Timeout monitoring is useful for all communciation interfaces. However, it may vary considerably between the individual bus systems.

Parameters for fieldbus timeout	Value range
Unit	Seconds
Range	0.01 s to 650.00 s in 10-ms steps
Special case	0 or 650.00 = Fieldbus timeout disabled
Factory setting	0.5 s

TIP

With nearly all fieldbus systems (exception: Modbus/TCP and MOVILINK® via RS485 and SBus), the timeout interval (P819 or P883/893) is set automatically by the controller.

Parameters P819, P883 and P893 only serve as indicators.

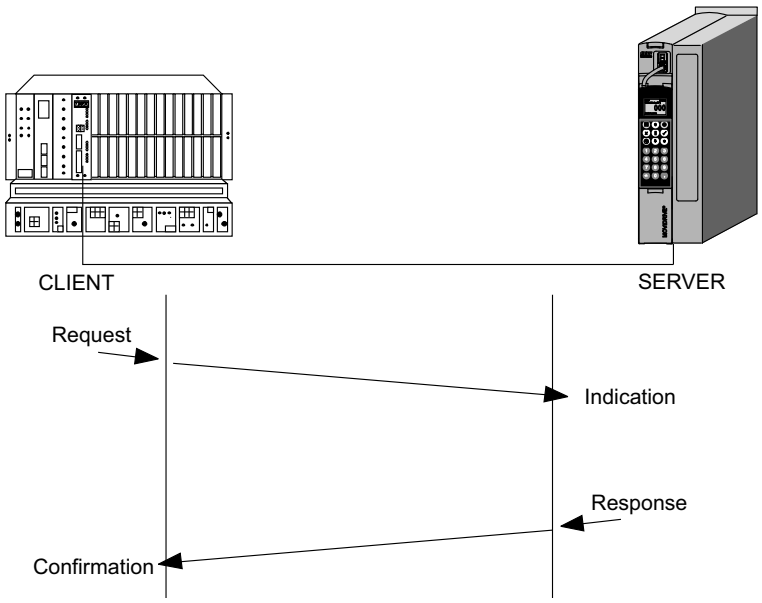
7.6 Setting the inverter parameters

The drive parameters of the inverter are usually accessed using the bus-specific READ and WRITE services. Additional services can be executed for all bus system using the MOVILINK® parameter channel. This parameter channel is available in all bus systems and is explained in detail below.

Also refer to the documentation for the fieldbus option card to obtain additional programming information on using the MOVILINK® parameter channel with the various bus systems.

Parameter setting procedure

The parameters of the MOVIDRIVE® inverter are usually set based on a client/server model. This means the inverter provides the requested information only when prompted by the higher-level programmable controller. This means that MOVIDRIVE® usually only has server functionality (see following figure).



54673AXX

7.6.1 Structure of the MOVILINK® parameter channel

The MOVILINK® parameter channel enables access to all drive parameters of the inverter, regardless of the bus in use. Special services are available in this parameter channel to being able to read different parameter information. It is made up of a management byte, a reserved byte, an index word and four data bytes.

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Manage-ment	Subindex	Index high	Index low	MSB data	Data	Data	LSB data
		Parameter index			4-byte data		

Management of the parameter channel (byte 0)

The entire procedure for setting parameters is coordinated using management byte 0. This byte provides important service parameters such as service identifier, data length, version and status of the service performed.

Index addressing (bytes 1 - 3)

Byte 2 "index high", byte 3 "index low", and byte 1 "subindex" determine the parameter to be read or written via the fieldbus system. All parameters of the MOVIDRIVE® inverter are listed in the MOVIDRIVE® MDX60B/61B system manual. A special number (index) is assigned to each parameter. This number is used to read or write the parameter.

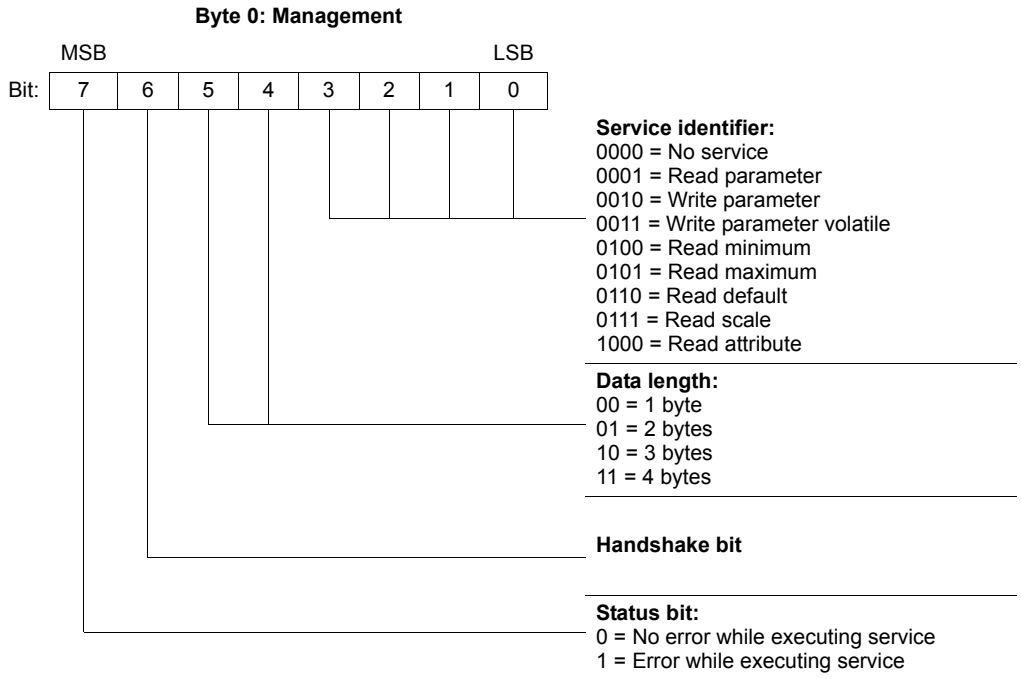
Data range (bytes 4 - 7)

The data is located in byte 4 to byte 7 of the parameter channel. This means 4 bytes of data can be transmitted per service. The data is always entered with right-justification; that is, byte 7 contains the least significant data byte (Data LSB) whereas byte 4 is the most significant data byte (Data MSB).

Management byte

Bits 0 - 3 contain the service identifier and define the service to be executed.

Bits **4 and bit 5** specify the data length in bytes; it should be set to 4 bytes for all SEW drive inverters.



Bit 6 is the handshake bit. It has a different meaning depending on the bus system:

- With SBus 1 (CAN) and set handshake bit (= 1), the response telegram is sent after the synchronization message (see chapter 5.3.2)
- With RS485 and fieldbus, the handshake bit serves as acknowledgement bit between client and server when using the cyclic transmission method. The parameter channel is transmitted cyclically, maybe with the process data. For this reason, the implementation of the service in the inverter must be triggered by edge control using handshake bit 6. For this purpose, the value of this bit is toggled for each new service to be executed. The inverter uses the handshake bit to signal whether the service has been executed or not. The service was executed if the handshake bit received in the control is identical with the transmitted handshake bit.

Status bit 7 indicates whether it was possible to execute the service properly or if errors occurred.

Response

The response to a parameter setting request is structured as follows:

- The management byte of the response telegram is structured like that in the request telegram.
- The status bit indicates whether the service was executed successfully:
  - If the status bit is set to "0", bytes 4 to 7 of the response telegram will contain the requested data.
  - If the status bit is set to "1", an error code is indicated in the data area (bytes 4 to 7) (see chapter "Incorrect service execution").

Description of the parameter services

Bits 0 - 3 of the management byte are used to define the individual parameter services. MOVIDRIVE® supports the following parameter services:

No service

This coding indicates that there is no parameter service.

Read parameter

This parameter service is used to read a drive parameter.

Write parameter

This parameter service is used for non-volatile writing of a drive parameter. The written parameter value is stored non-volatile (e.g. in EEPROM). This service should not be used for cyclic write accesses because the memory modules allow for only a limited number of write cycles.

Write parameter volatile

This parameter service is used to write a drive parameter volatile, if the parameter permits this. The written parameter value is only saved volatile in the RAM of the inverter, which means it is lost when switching off the inverter. When switching the inverter back on again, the value that was last written with write parameter will be available again.

Read minimum

This service can be used to determine the smallest drive parameter value (minimum) that can be set. Coding takes place in the same manner as the parameter value.

Read maximum

This service can be used to determine the largest drive parameter value (maximum) that can be set. Coding takes place in the same manner as the parameter value.

Read default

This service can be used to determine the factory setting (default) of a drive parameter. Coding takes place in the same manner as the parameter value.

Read Scale

This service can be used to determine the scaling of a parameter. The inverter provides a so-called measurement index and a conversion index.

Byte 4	Byte 5	Byte 6	Byte 7
MSB data	Data	Data	LSB data
Reserved		Measurement index	Conversion index

Measurement index

The measurement index is used for coding physical values. This index is used to inform a communication partner about the physical value of the associated parameter value. Coding takes place according to the sensor/actuator profile of the PROFIBUS user organization (PUO). The entry FF_{hex} means that no measurement index is specified. You can also obtain the measurement index from the parameter directory of the inverter.

Conversion index:

The conversion index serves for converting the transmitted parameter value into an SI basic unit. Coding takes place according to the sensor/actuator profile of the PROFIBUS user organization (PUO).

Example:

Drive parameter: P131 ramp t11 down CW

Measurement index: 4 (= time with second as measurement unit)

Conversion index: - 3 (10^{-3} = milli)

Transmitted numerical value:3000 dec

The drive inverter interprets the numerical value received via bus as follows:
3000 s $\times$ 10^{-3} = 3 s

Read attribute

This service can be used for reading the access attributes and the index of the next parameter. The following table shows the coding of the data for this parameter service.

Byte 4	Byte 5	Byte 6	Byte 7
MSB data	Data	Data	LSB data
Next available index		Access attributes	

The coding of the access attributes is unit-specific. For MOVIDRIVE® inverters, the attribute definition results from the following table.

Byte 6 Bit	Byte 7 Bit	Meaning
	0	1 = Parameter allows write access
	1	1 = Parameter is permanently saved on EEPROM
	2	1 = Factory setting overwrites RAM value
	3	1 = Factory setting overwrites EEPROM value
	4	1 = EEPROM value is valid after initialization
	5	1 = Controller inhibit condition not necessary for write access
	6	1 = Password requried
8	7	00 = Parameter is generally valid 01 = Parameter is assigned to parameter set 1 10 = Parameter is assigned to parameter set 2 11 = Parameter is assigned to both parameter sets
9 - 15		Reserved

Parameter list

For detailed information on coding and access attributes of all parameters, refer to the parameter list.

Incorrect service execution

If the received handshake bit is identical to the transmitted handshake bit, the inverter has executed the service.

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Manage-ment	Subindex	Index high	Index low	**Error class**	**Error code**	**Add. code high**	**Add. code low**

↓

Status bit = 1: Service executed incorrectly

7.6.2 Return codes of parameterization

In the event of an incorrect parameter setting, the drive inverter sends back various return codes to the master that set the parameters. These codes provide detailed information about what caused the error. All of these return codes are structured in accordance with EN 50170. The inverter distinguishes between the following elements:

- Error class
- Error code
- Additional code

These return codes apply to all MOVIDRIVE® communication interfaces.

Error class

The error class element provides a more exact classification of the error type. The following error classes are distinguished in accordance with EN 50170.

Class (hex)	Designation	Meaning
1	vfd state	Status error of the virtual field device
2	application reference	Error in application program
3	definition	Definition error
4	resource	Resource error
5	service	Error during execution of service
6	access	Access error
7	ov	Error in the object list
8	other	Other error (see additional code)

The error class is generated by the communication software of the fieldbus interface if there is an error in communication. This statement does not apply to *Error class 8 = Other error*. Return codes sent from the drive inverter system are all included in *Error class 8 = Other error*. The error can be identified more precisely using the additional code element. The Ethernet error code will then be "0".

Error code

The error code element allows for a more detailed identification of the error cause within the error class and is generated by the communications software of the fieldbus interface in the event of faulty communication.

Additional code

The additional code contains SEW-specific return codes for faulty parameterization of the drive inverter. These codes are returned to the master under *Error class 8 = Other error*. The following table shows all possible codings for the additional code.

MOVILINK®			
	Additional code		
Error class	**High**	**Low**	**Description**
		0x00	Unknown error
		0x01	Illegal Service
		0x02	No Response
		0x03	Different Address
		0x04	Different Type
		0x05	Different Index
		0x06	Different Service
		0x07	Different Channel
		0x08	Different Block
		0x09	No Scope Data
		0x0A	Illegal Length
		0x0B	Illegal Address
0x05	00	0x0C	Illegal Pointer
		0x0D	Not enough memory
		0x0E	System Error
		0x0F	Communication does not exist
		0x10	Communication not initialized
		0x11	Mouse conflict
		0x12	Illegal Bus
		0x13	FCS Error
		0x14	PB Init
		0x15	SBUS - Illegal Fragment Count
		0x16	SBUS - Illegal Fragment Type
		0x17	Access denied
			Not used

MOVILINK®			
	Additional code		
Error class	High	Low	Description
0x08	00	0x00	No Error
		0x10	Illegal Index
		0x11	Not yet implemented
		0x12	Read only
		0x13	Parameter Blocking
		0x14	Setup runs
		0x15	Value too large
		0x16	Value too small
		0x17	Required Hardware does not exist
		0x18	Internal Error
		0x19	Access only via RS485 (via X13)
		0x1A	Access only via RS485 (via XT)
		0x1B	Parameter protected
		0x1C	"Controller inhibit" required
		0x1D	Value invalid
		0x1E	Setup started
		0x1F	Buffer overflow
		0x20	"No Enable" required
		0x21	End of File
		0x22	Communication Order
		0x23	"IPOS Stop" required
		0x24	Autosetup
		0x25	Encoder Nameplate Error
		0x29	PLC State Error

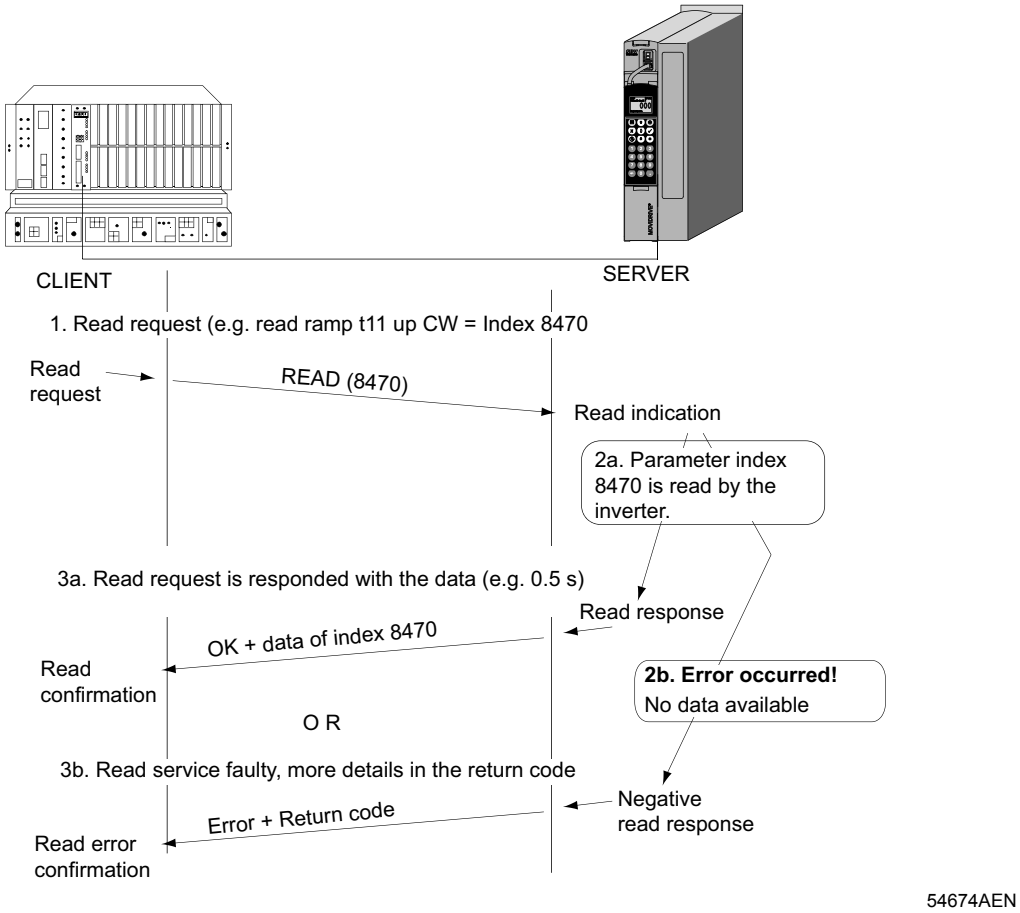
Example: Invalid parameter setting

An incorrect index was entered when executing a read or write service.

	Code (hex)	Meaning
Error class	0x08	Other
Error code	0x00	-
Add. code high	0x00	-
Add. code low	0x10	Illegal Index

7.6.3 Example: Reading a parameter (READ)

A parameter is read via communication interfaces with a *read request* from the programmable controller of the MOVIDRIVE® inverter (see figure below).



54674AEN

If the read service cannot be executed in the drive inverter, the programmable controller will receive a *negative read response*. In this way, the programmable controller receives a negative acknowledgement *(read error confirmation)* with exact identification of the error.

Reading a parameter cyclically

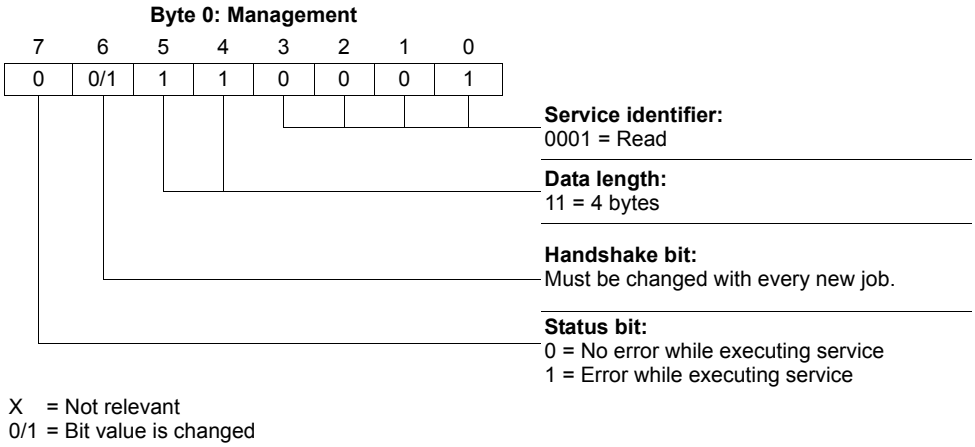
For the cyclic transmission method, the handshake bit has to be changed to activate service processing (execution of READ service). When using acyclic PDU types, every inverter processes every request telegram and in this way always executes the parameter channel.

The parameters are set as follows:

1. Enter the index of the parameter to be read in byte 2 (index high) and byte 3 (index low).
2. Enter the service identifier for the read service in the management byte (byte 0).
3. With cyclic PDU types, the read service is not passed to the inverter until the handshake bit is changed. With acyclic PDU types, the parameter channel is always executed.

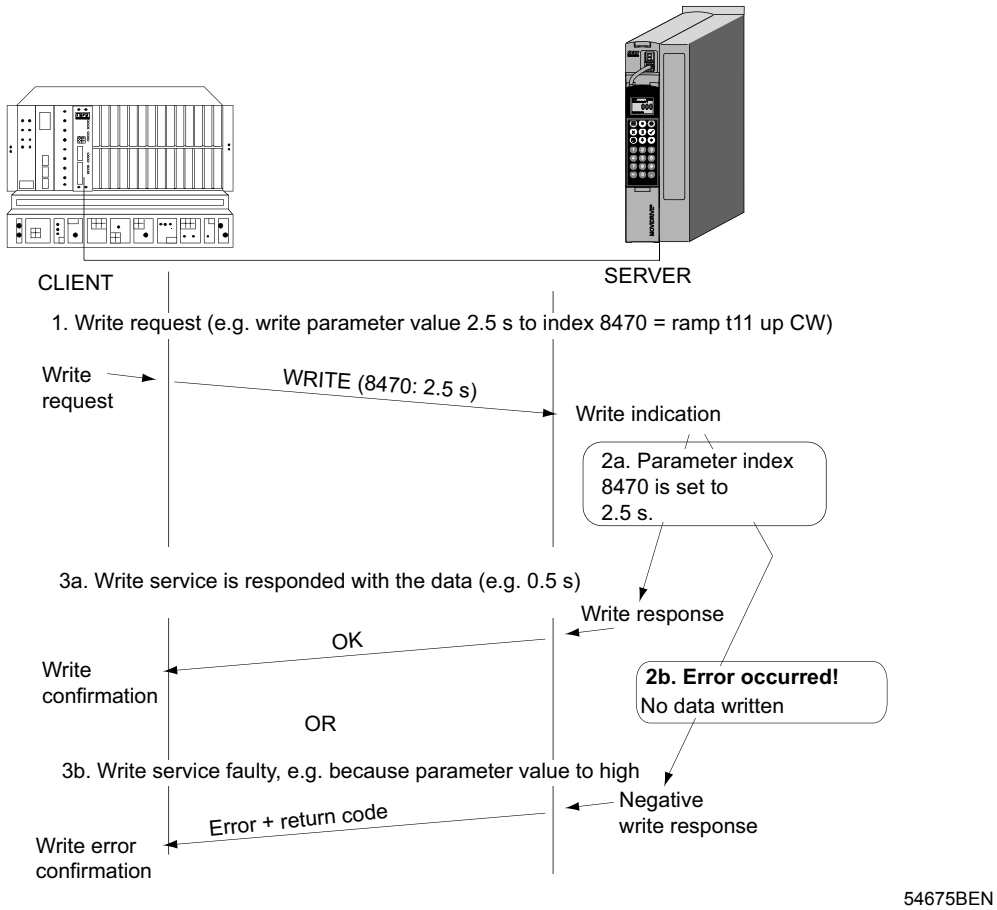
Since this is a read service, the sent data bytes (bytes 4 - 7) and the data length (in the management byte) are ignored and do not need to be set.

The inverter now processes the read service and sends the service confirmation back by changing the handshake bit.



7.6.4 Example: Writing a parameter (WRITE)

A parameter is written as it is read via the fieldbus interface (see figure below).



If the write service cannot be executed in the drive inverter, for example because incorrect parameter data were transmitted, the programmable controller will receive a *negative read response*. In this way, the programmable controller receives a negative acknowledgement *(write error confirmation)* with exact identification of the error.

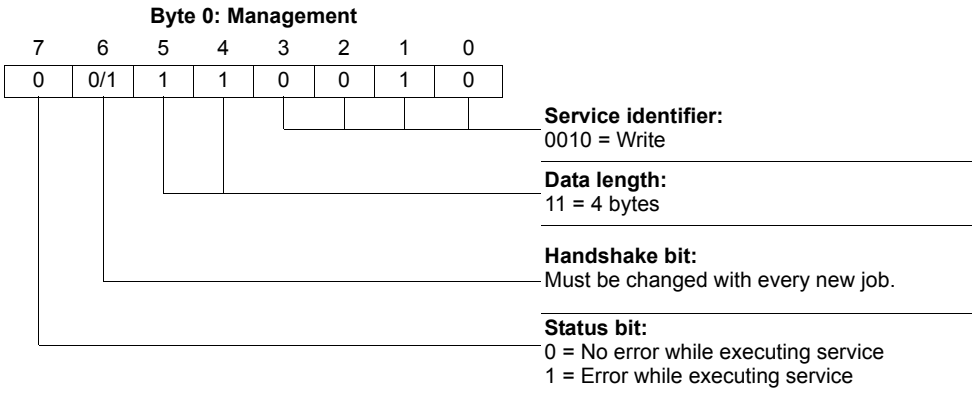
Writing a parameter cyclically

For the cyclic transmission method, the handshake bit has to be changed to activate service processing (execution of WRITE service). When using acyclic PDU types, every inverter processes every request telegram and in this way always executes the parameter channel.

The parameters are set as follows:

1. Enter the index of the parameter to be written in byte 2 (index high) and byte 3 (index low).
2. Enter the data to be written in bytes 4 - 7.
3. Enter the service identifier and the data length for the write service in the management byte (byte 0).
4. With cyclic PDU types, the WRITE service is not passed to the inverter until the handshake bit is changed. With acyclic PDU types, the parameter channel is always executed.

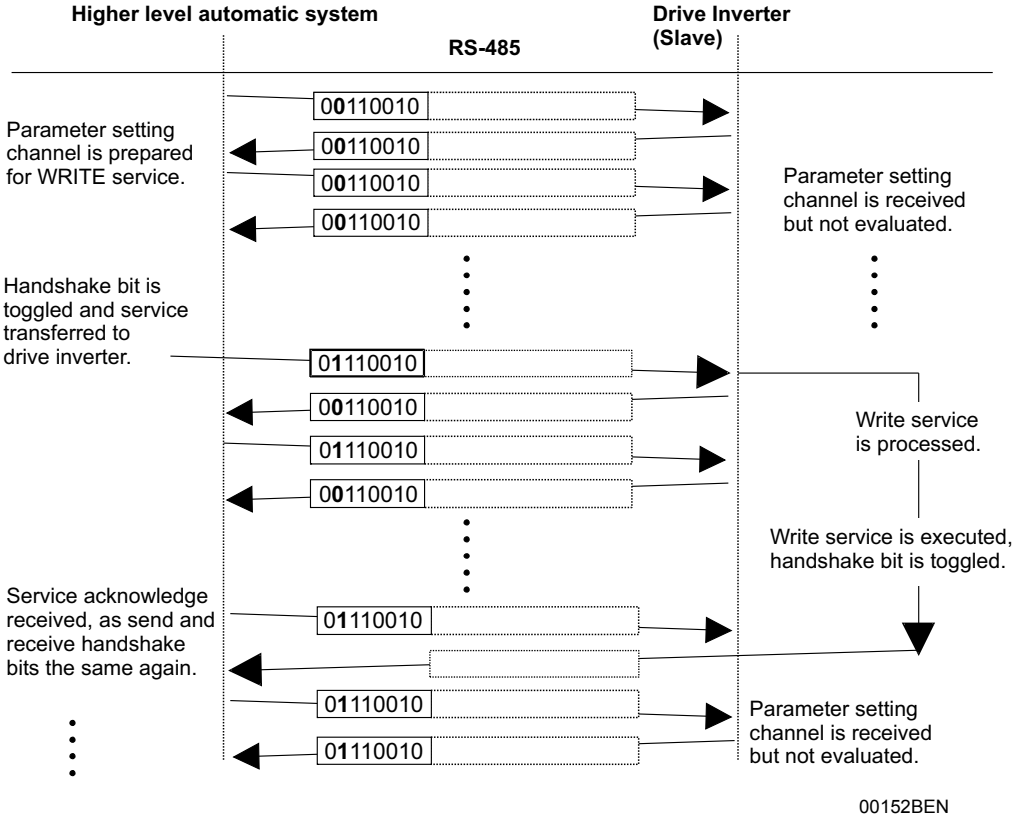
The inverter now processes the write service and sends the service confirmation back by changing the handshake bit.



0/1 = Bit value is changed

The data length is 4 bytes for all parameters of SEW drive inverters.

Using the WRITE service as an example, the following figure represents a process of setting parameters between the controller and inverter using a cyclic PDU type. To simplify the sequence, only the management byte of the parameter channel is shown here.



The drive inverter only receives and returns the parameter channel while the controller is preparing the parameter channel for the write service. The service is not activated until the moment when the handshake bit is changed (in this example, when it changes from 0 to 1). The drive inverter now interprets the parameter channel and processes the write service, but continues to respond to all telegrams with handshake bit = 0. The executed service is acknowledged with a change of the handshake bit in the response message of the drive inverter. The master now detects that the received handshake bit is once again the same as the one which was sent. It can now prepare another parameter setting procedure.

7.7 Notes on parameterization

By parameterizing the MOVIDRIVE® inverter via the fieldbus system, you can generally reach all drive parameters. As some drive parameters are directly connected with the communciation via fieldbus system, you should observe the following notes on parameterization.

Parameter setting in CONTROLLER INHIBIT condition

Some parameters can only be changed (written) in *CONTROLLER INHIBIT* drive status. The inverter signals this by a negative acknowledgement of the write service. Refer to the parameter list to see what parameters are subject to this limitation. In general, these parameters can be changed during an error or *24 V operation*.

Factory setting

Activating the factory setting means nearly all parameters are reset to their default values. The consequence for bus operation is that the control signal source and setpoint source are reset to their default values.

	TIP
	The drive inverter must be enabled at the terminals for control via process data. This means that the drive is enabled under certain conditions after a factory setting. Therefore, make sure before activating the factory setting that the signals of the digital binary inputs will not trigger a drive inverter enable once the factory setting has been restored. As a precaution, do not switch on the mains voltage until you have completed inverter parameterization.

Parameter lock

The parameter lock is activated by setting *P803 Parameter lock* = *Yes*. It protects all adjustable parameters from being changed. Activating parameter lock is useful when all drive inverter parameters have been set and need not be changed anymore. This parameter lets you, for example, use the keypad of the drive inverter to protect the drive inverter parameters from being changed.

	TIP
	The parameter lock generally prevents the writing of parameters. Consequently, write access via communication interfaces is also blocked when the parameter lock is active.

8 Operating MOVITOOLS® MotionStudio

8.1 About MOVITOOLS® MotionStudio

8.1.1 Tasks

The software package enables you to perform the following tasks:

- Establishing communication with units
- Executing functions with the units

8.1.2 Establishing communication with the units

The SEW Communication Server is integrated into the MOVITOOLS® MotionStudio software package for establishing communication with the units.

The SEW Communication Server allows you to create **communication channels**. Once the channels are established, the units communicate via these communication channels using their communication options. You can operate up to four communication channels at the same time.

MOVITOOLS® MotionStudio supports the following communication channels:

- Serial (RS485) via interface adapters
- System bus (SBus) via interface adapters
- Ethernet
- EtherCAT
- Fieldbus
- PROFIBUS DP/DP-V1
- S7-MPI

The available channels can vary depending on the units and its communication options.

8.1.3 Executing functions with the units

The software package offers uniformity in executing the following functions:

- Parameterization (for example in the parameter tree of the unit)
- Startup
- Visualization and diagnostics
- Programming

The following basic components are integrated into the MOVITOOLS® MotionStudio software package, allowing you to use the units to execute functions:

- MotionStudio
- MOVITOOLS®

All functions communicate using **tools**. MOVITOOLS® MotionStudio provides the right tools for every unit type.

8.2 First steps

8.2.1 Starting the software and creating a project

Proceed as follows to start MOVITOOLS® MotionStudio and create a project:

1. Start MOVITOOLS® MotionStudio in the WINDOWS® start menu via the following path:

   "Start\Program\SEW\MOVITOOLS MotionStudio\MOVITOOLS MotionStudio"

2. Create a project with name and storage location.

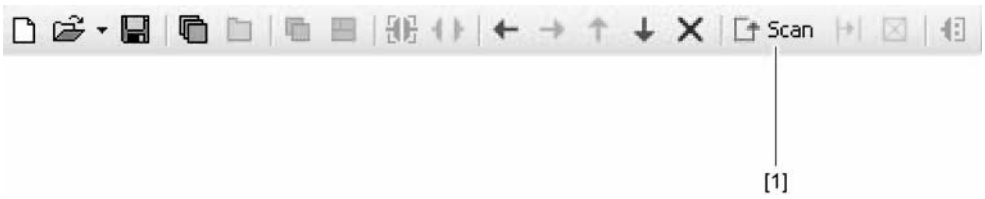
8.2.2 Establishing communication and scanning the network

Proceed as follows to establish a communication with MOVITOOLS® MotionStudio and scan your network:

1. Set up a communication channel to communicate with your units.

   Refer to the section dealing with the respective type of communication for detailed information.

2. Scan your network (unit scan). To do so, click the [Start network scan] button [1] in the toolbar.



64334AXX

3. Select the unit you want to configure.
4. Open the context menu with a right mouse click.

   As a result you will see a number of unit-specific tools to execute various functions with the units.

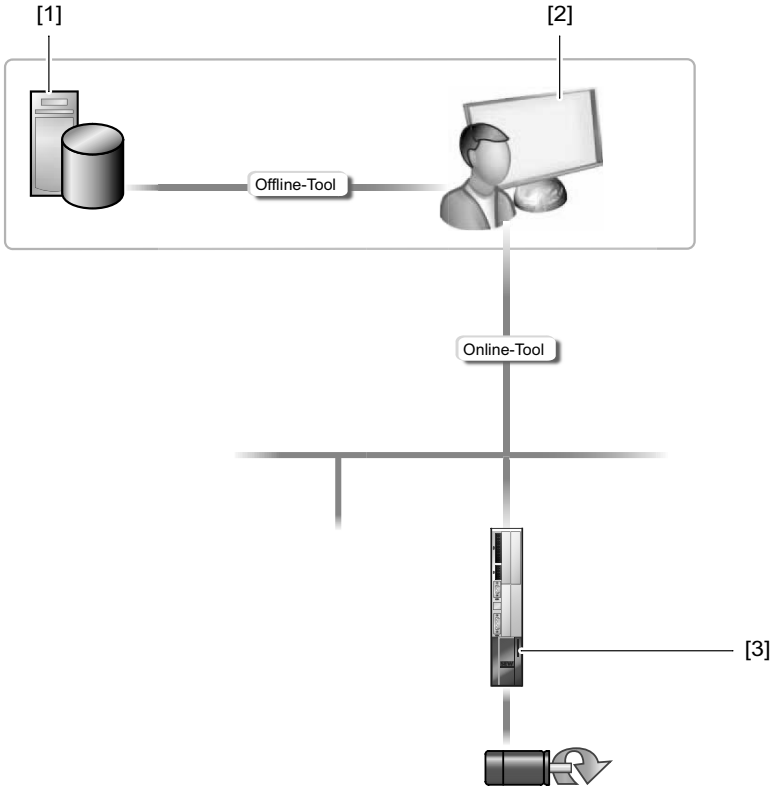
8.3 Communication mode

8.3.1 Overview

MOVITOOLS® MotionStudio differentiates between "online" and "offline" communication mode.

You can select the communication mode. Unit-specific offline or online tools are provided depending on the communication mode you have selected.

The following figure illustrates the two types of tools:



64335AXX

Tools	Description
Offline tools	Changes made using offline tools affect **"ONLY"** the RAM [2]. • Save your project so that the changes can be stored on the hard disk [1] of your PC. • To transfer the changes also to your unit [3], perform a download.
Online tools	Changes made using online tools affect **"ONLY"** the unit [3]. • To transfer the changes to the RAM [2], perform an upload. • Save your project so that the changes can be stored on the hard disk [1] of your PC.

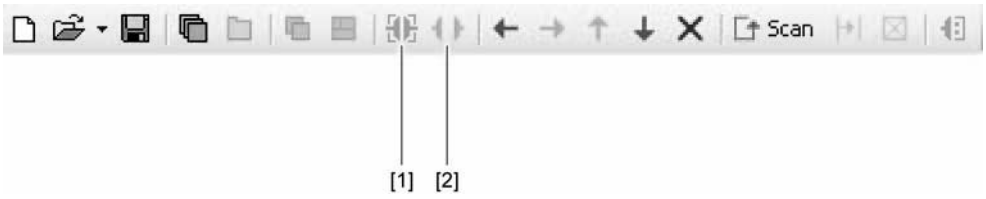
	TIP
	The "online" communication mode is "NOT" a response message which informs you that you are currently connected to the unit or that your unit is ready for communication. • Should you require this feedback, observe section "Setting the cyclical accessibility test" in the online help (or the manual) of MOVITOOLS® MotionStudio.

	TIP
	• Project management commands (such as "download" and "upload"), the online unit status, and the "unit scan" operate independently of the set communication mode. • MOVITOOLS® MotionStudio starts up in the communication mode that you set before you closed down.

8.3.2 Selecting communication mode (online or offline)

Proceed as follows to select a communication mode:

1. Select the communication mode:
   - "Online" [1] for functions (online tools) that should directly influence the unit.
   - "Offline" [2] for functions (offline tools) that should influence your project.



64337AXX

2. Select the unit node.
3. Right-click to open the context menu and display the tools for configuring the unit.

8.4 Serial communication (RS485) via interface adapters

8.4.1 Engineering via interface adapters (serial)

Since your unit supports the "serial" communication option, you can use a suitable interface adapter for engineering.

The interface adapter is additional hardware that you can obtain from SEW-EURODRIVE. You can use it to connect your engineering PC to the corresponding communication option of the unit.

The following table shows you the different types of interface adapters available, and for which units they are suitable.

Type of interface adapter (option)	Part number	Scope of delivery	Units
USB11A (USB to RS485)	0824 831 1	2 connection cables: • TAE connection cable with 2 RJ10 plugs • USB connection cable with USB-A plug and USB-B plug	• MOVIDRIVE® B • MOVITRAC® 07A • MOVITRAC® B • MOVIFIT® MC/FC/SC • MOVIGEAR® • UFx11A fieldbus gateways • DFx fieldbus gateways • DHx MOVI-PLC® controller • MFx/MQx fieldbus interfaces for MOVIMOT®
UWS21B (RS232 to RS485)	1820 456 2	2 connection cables: • TAE connection cable with 2 RJ10 plugs • Connection cable with 9-pin D-sub plug	
UWS11A (RS232 to RS485) for support rail	822 689 X	none	

Since most PCs are now equipped with USB interfaces instead of RS232 interfaces, the following chapters only deal with the USB11A interface adapter.

How to connect the interface adapter to MOVIDRIVE® B is described in chapter 4, "Serial Interfaces of MOVIDRIVE® B".

8.4.2 Taking the USB11A interface adapter into operation

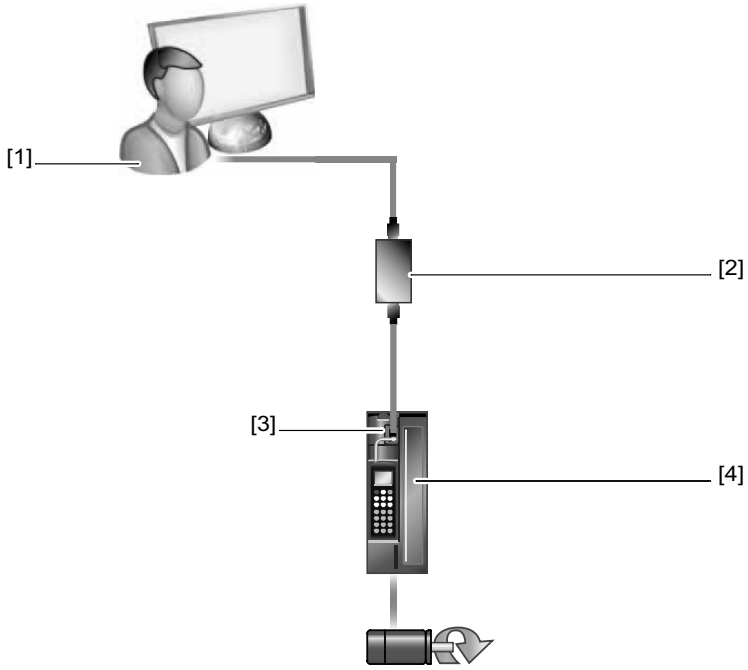
Overview

The USB11A interface adapter uses a COM redirector. This assigns the first free COM port to the interface adapter.

The following describes how to connect the USB11A interface adapter to your unit and, if necessary, how to install the driver.

Connecting USB11A to MOVIDRIVE® B

The following figure shows how the USB11A interface adapter [2] is connected with MOVIDRIVE® B [4] and with the PC [1] using the XT socket [3].



64340AXX

[1] PC

[2] USB11A with two connection cables (included in the scope of delivery)

[3] XT socket of MOVIDRIVE® B

[4] MOVIDRIVE® B

To connect the USB11A interface adapter with the PC and MOVIDRIVE® B, proceed as follows:

1. Connect the USB11A interface adapter [2] with the two connection cables provided.
2. Plug the RJ10 connector of the first connection cable into the XT socket [3] of MOVIDRIVE® B [4].
3. Plug the USB-**A** connector of the second connection cable into the free USB interface on your PC [1].
4. If you are operating the interface adapter with MOVITOOLS® MotionStudio for the first time, you will have to install the required driver.

Installing the drivers

The drivers for the USB11A interface adapter are installed together with MOVITOOLS® MotionStudio. This also applies to the driver for the COM redirector. As a prerequisite, the interface adapter must have been connected to your PC during the MOVITOOLS® MotionStudio installation process.

If you need to install the drivers manually, you can do so via the MOVITOOLS® MotionStudio installation path.

Proceed as follows to **manually** install the driver for the USB11A interface adapter:

1. Make sure that you have local administrator rights on your PC.
2. Connect the USB11A interface adapter to a free USB connection on your PC.

   Your PC will detect the new hardware and launch the hardware wizard.
3. Follow the instructions of the hardware wizard.
4. Click on [Browse] and go to the MOVITOOLS® MotionStudio installation folder.
5. Enter the following path:

   "`..\Program Files\SEW\MotionStudio\Driver\FTDI_V2`"
6. Click the [Next] button to install the driver and assign the first free COM port of the PC to the interface adapter.

Checking the COM port of the USB11A on the PC

Proceed as follows to check which virtual COM port has been assigned to the USB11A on the PC:

1. Select the following menu item on your PC:

   [Start] / [Setup] / [Control panel] / [System]
2. Open the "Hardware" tab.
3. Click the [Device manager] button.
4. Open the "Connections (COM and LPT)" folder.

   You will see which virtual COM port has been assigned to the interface adapter, e.g.: "USB serial port (COM3)".

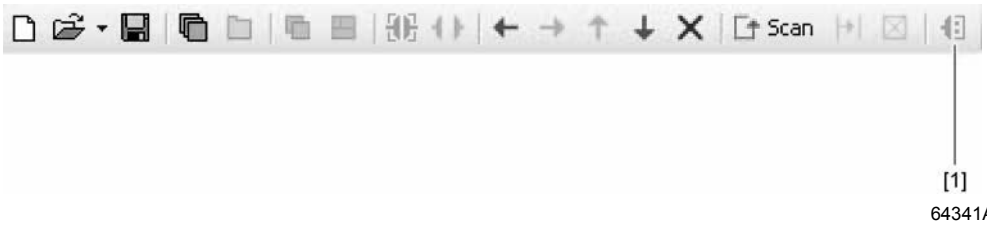
	TIP
	Change the COM port of the USB11A to prevent a conflict with another COM port. It is possible that a different hardware (such as an internal modem) is assigned the same COM port as the USB11A interface adapter. • Select the COM port of USB11A in the device manager. • In the context menu, click the [Properties] button and assign the USB11A another COM port. • Restart your system for the changes to become effective.

8.4.3 Configuring serial communication

You must have a serial connection between your PC and the units you want to configure. You can establish one, for example, using the USB11A interface adapter.

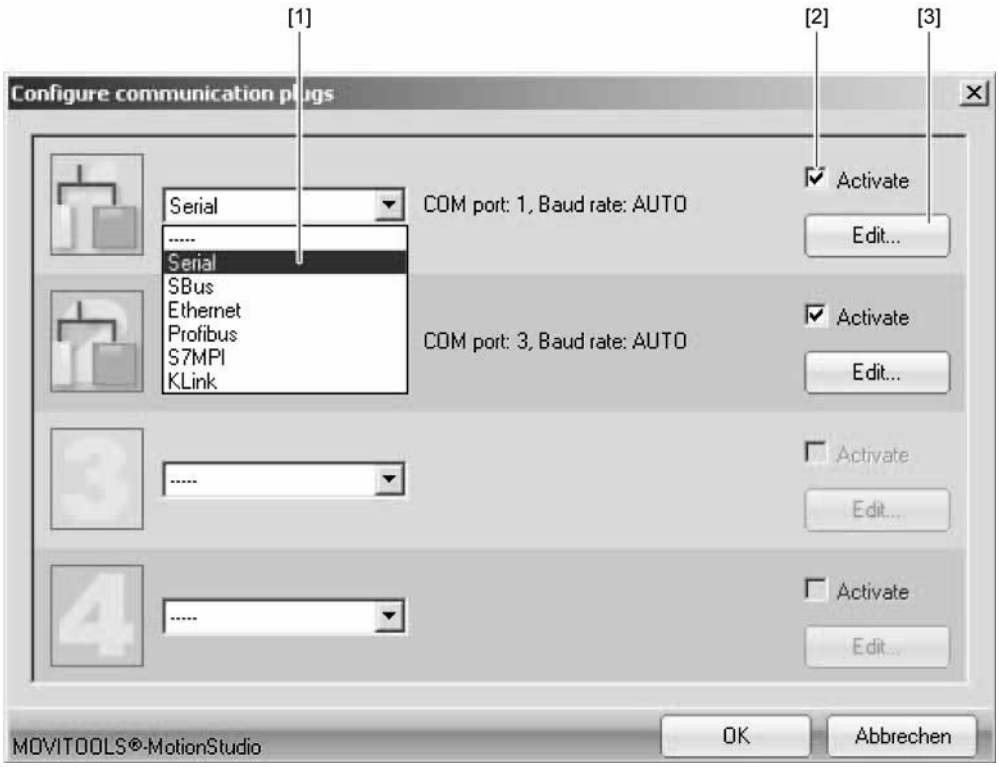
Proceed as follows to configure serial communication:

1. Click on [Configure communication plugs] [1] in the toolbar.



64341AXX

This opens the "Configure communication plugs" window.



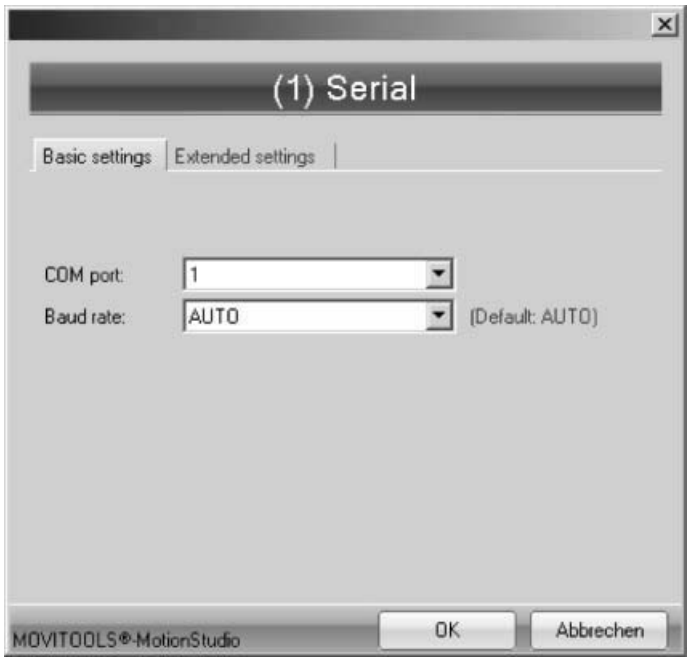
64342AEN

2. From the selection list [1], choose "Serial" as the communication type.

   In the example, "Serial" is activated as the communication type for the first communication channel [2].

3. Press the [Edit] button [3] on the right side of the "Configure communication plugs" window.

   This will display the settings for the "Serial" communication type.



12078AEN

4. If necessary, change the preset communication parameters on the tab pages [Basic settings] and [Advanced settings]. When doing so, refer to the detailed description of the communication parameters (page 123).

8.4.4 Serial communication parameter (RS485)

The following table describes the [Basic setting] for the serial (RS485) communication channel:

Communication parameters	Description	Note
COM port	Serial port connected to the interface adapter	• If there is no value entered here, the SEW Communication Server uses the first available port. • A USB interface adapter is indicated by the addition "(USB)".
Baud rate	Transmission speed with which the connected PC communicates with the unit in the network via the communication channel	• Possible values: – 9.6 kBit/s – 57.6 kBit/s – AUTO (default setting) • Find the correct value for the connected unit in the documentation. • If you set "AUTO", the units are scanned with both baud rates in succession. • Set the starting value for automatic baud rate detection under [Settings] / [Options] / [Communication].

The following table describes the [Advanced setting] for the serial (RS485) communication channel:

Communication parameters	Description	Note
Parameter telegrams	Telegram with a single parameter	Used for transmitting a **single parameter** of a unit.
Multibyte telegrams	Telegram with several parameters	Used for transmitting the **complete parameter set** of a unit.
Timeout	Waiting time in [ms] that the master waits for a response from a slave after it has sent a request.	• Default setting: – 100 ms (parameter telegram) – 350 ms (multibyte telegram) • Increase the value if not all units are detected during a network scan.
Retries	Number of request retries after the timeout is exceeded	Default setting: 3

8.5 Communication SBus (CAN) via interface adapter

8.5.1 Engineering via interface adapters (SBus)

Since your unit supports the "SBus" communication option, you can use a suitable interface adapter for engineering.

The interface adapter is additional hardware that you can obtain from SEW-EURODRIVE. You can use it to connect your engineering PC with the respective communication option of the unit.

The following table shows the different types of interface adapters available, and for which units they are suitable:

Interface adapter type (option)	Order no.	Scope of delivery	Units
PC-CAN interface from SEW (incl. prefabricated connection cable with integrated terminating resistor)	1821 059 7	• Prefabricated cable with 9-pole Sub-D-connector for connection to the unit, length 2 m • A 120 ohm terminating resistor is fitted to one end of the prefabricated cable (between CAN_H and CAN_L).	• MOVIAXIS® • MOVIDRIVE® B • MOVITRAC® B • MOVI-PLC® (*basic* and *advanced*)
PCAN-USB ISO from Peak	IPEH 002022	• Without connection cable • Without terminating resistor	

To connect the PC CAN interface to the unit, you need an additional connection cable with terminating resistor. The scope of delivery of the PC CAN interface from SEW includes a prefabricated connection cable on the unit with terminating resistor. Therefore, only this PC CAN interface is described in the following chapter.

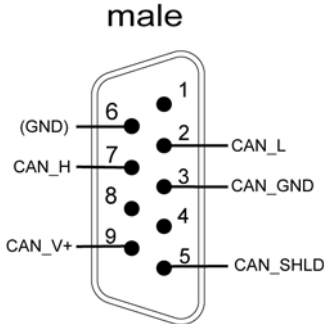
8.5.2 Taking the USB-CAN interface into operation

Overview

The following chapter describes how to connect the PC-CAN interface from SEW-EURODRIVE to the SBus interface of your unit and what you have to observe when doing so.

CAN pin assignment

The figure below shows the pin assignment of the 9-pin D-Sub plug of the PC-CAN interface from SEW (view from top):

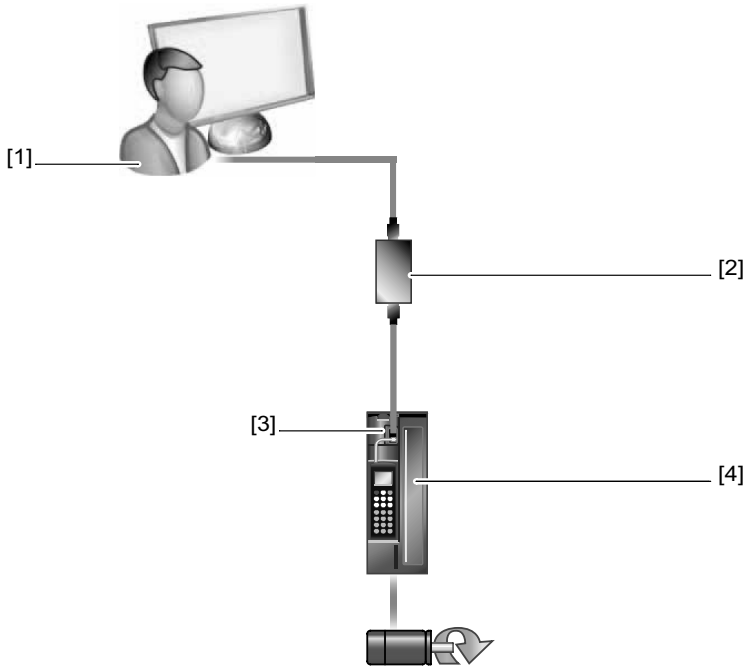


64773AXX

Connecting the USB-CAN interface to the unit

Chapter 4, "CAN Interfaces of MOVIDRIVE® B" provides a description of how to connect the CAN interfaces of MOVIDRIVE® B.

The figure shows how the USB-CAN interface adapter [2] from SEW-EURODRIVE is connected with MOVIDRIVE® B [4] and with the PC [1] using the SBus interface [3].



[1] PC

[2] USB-CAN interface with prefabricated connection cable with terminating resistor (included in the scope of delivery)

[3] SBus interface, e.g. X30 of the DFC11B option

[4] MOVIDRIVE® B

To connect the USB-CAN interface with the PC and MOVIDRIVE® B, proceed as follows:

1. Connect the 9-pin D-sub connector of the USB-CAN interface with the prefabricated connection cable. Make sure that the cable end with the terminating resistor leads to the USB-CAN interface.
2. Connect the second cable end (without terminating resistor) with the SBus interface X30 of the DFC11B option [3] in MOVIDRIVE® B [4].
3. If the USB-CAN interface is connected to the first or last device in a network, switch on the terminating resistor on the DFC11B option (DIP switch "R" to "ON").
4. Plug the USB-**A** connector of the USB cable into a free USB interface on your PC [1].

8.5.3 Configuring communication via SBus

You need an SBus connection between your PC and the units you want to configure. You can use a USB-CAN interface for this purpose.

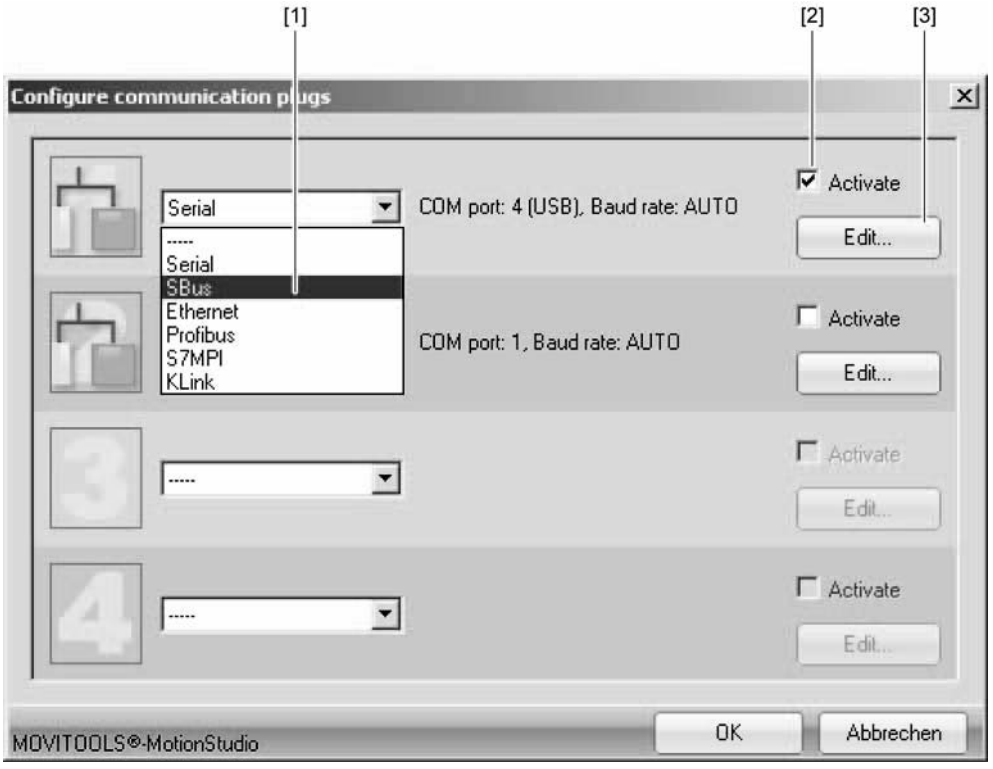
Proceed as follows to configure an SBus communication:

1. Click on "Configure communication plugs" [1] in the toolbar.

[1] "Configure communication plugs" icon

Doing so will open the "Configure communication plugs" window.

2. From the selection list [1], select "SBus" as the communication type.



[1] "Communication type" selection list

[2] "Activated" check box

[3] [Edit] button

In the example, "SBus" is activated as the communication type for the first communication channel [2].

3. Press the [Edit] button [3] on the right side of the "Configure communication plugs" window.

12113ADE

This will display the settings for the "SBus" communication type.

4. It might be necessary to change the preset communication parameters on the tab pages [Basic settings] and [Advanced settings]. When doing so, refer to the detailed description of the communication parameters.

8.5.4 Communication parameters for SBus

The following table describes the [Basic setting] for the SBus communication channel:

Communication parameters	Description	Note
Baud rate	Transmission speed with which the connected PC communicates with the unit in the network via the communication channel.	• Adjustable values (permitted total cable length): – 125 kBaud (500 m) – 250 kBaud (250 m) – 500 kBaud (100 m) (default) – 1 MBaud (25 m) • All connected units must support the same baud rate.

The following table describes the [Advanced setting] for the SBus communication channel:

Communication parameters	Description	Note
Parameter telegrams	Telegram with a single parameter	Used for transmitting a **single parameter** of a unit.
Multibyte telegrams	Telegram with several parameters	Used for transmitting the **complete parameter set** of a unit.
Timeout	Waiting time in [ms] that the master waits for a response from a slave after it has sent a request.	• Default setting: – 100 ms (parameter telegram) – 350 ms (multibyte telegram) • Increase the value if not all units are detected during a network scan.
Retries	Number of request retries after the timeout is exceeded	Default setting: 3

8.6 Communication via Ethernet, fieldbus or SBUS^{plus}

8.6.1 Connecting the unit with the PC via Ethernet

The engineering access via Ethernet, fieldbus options or SBUS^{plus} is described in detail in the relevant manuals of the fieldbus option cards.

8.7 Executing functions with the units

8.7.1 Parameterizing units in the parameter tree

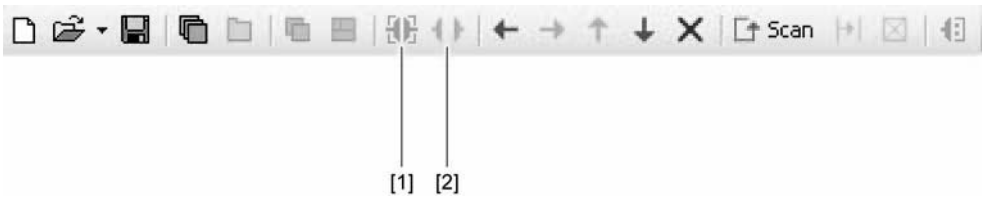
The parameter tree displays all unit parameters arranged in folders.

You can manage the unit parameters using the context menu or toolbar. The following chapter describes how to read or change unit parameters.

8.7.2 Reading/changing unit parameters

To read or change unit parameters, proceed as follows:

1. Switch to the required view (project view or network view).
2. Select the communication mode:
   - Click the [Switch to online mode] button [1] if you want to read or change parameters directly on the **unit**.
   - Click the [Switch to offline mode] button [2] if you want to read or change parameters in the **project**.

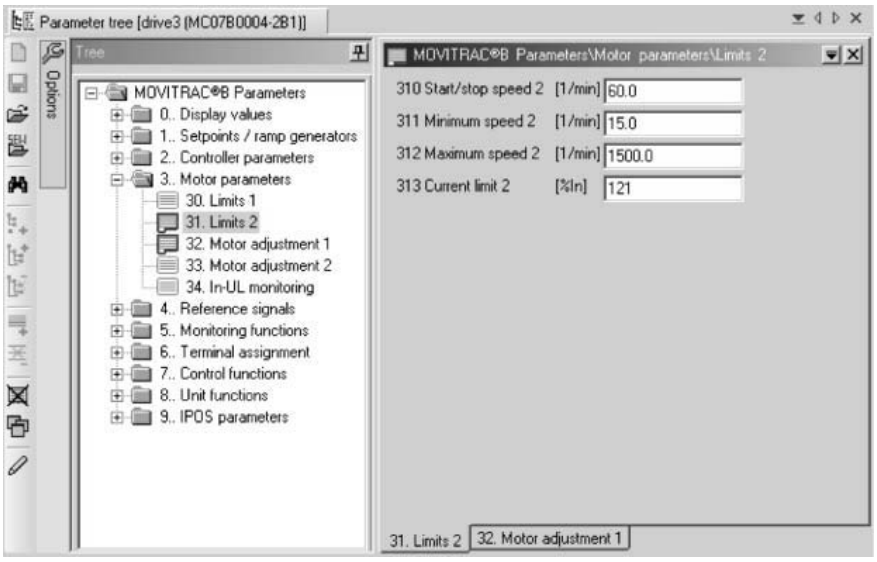


64337AXX

3. Select the unit you want to set parameters for.
4. Open the context menu and select the [Parameter tree] command.

   This opens the "Parameter tree" view on the right section of the screen.

5. Expand the "Parameter tree" up to the node you require.



12079AEN

6. Double-click to display a particular group of unit parameters.
7. Press the enter key to finalize any changes you make to numerical values in the input fields.

8.7.3 Starting up the units (online)

To startup units (online), proceed as follows:

1. Switch to the network view.
2. Click the [Switch to online mode] button [1].

64354AXX

3. Select the unit you want to startup.
4. Open the context menu and select the command [Startup] / [Startup wizard].

   This opens the startup wizard.
5. Follow the instructions of the startup wizard and then load the startup data into your unit.

	TIPS
	• For detailed information about the unit parameters, refer to parameter list for the unit. • For detailed information about how to use the startup wizard, refer to the MOVITOOLS® MotionStudio online help.

8.7.4 Unit-internal scope

The unit-internal scope memory of MOVIDRIVE® lets you record unit states or process data and have them displayed on your PC for diagnostic purposes. You will find additional information in the online help of MOVITOOLS® MotionStudio.

8.8 Bus monitor

The engineering user interface of MOVITOOLS® MotionStudio lets you use the process data monitor function under the menu item "Bus monitor". This function provides you with convenient startup and diagnostics options for using the inverter in a communication system. You can choose between the two operating modes *Monitor* and *Control*. Monitor mode is a mere diagnostic mode that lets you monitor process data channels, whereas control mode lets you make changes using your PC.

8.8.1 Diagnostic mode of the bus monitor

In *Monitor* mode, the bus monitor in MOVITOOLS® MotionStudio displays the setpoints and actual values, which are exchanged between master controller and MOVIDRIVE® inverter, so you can monitor and analyze them.

You obtain all the information of the three process data channels, such as description of process input data PI1 - PI3 (actual values) and process output data PO1 - PO3 (setpoints) as well as their values which are currently being transmitted via the bus system.

8.8.2 Control using bus monitor

In *Control* mode, you can use the bus monitor for controlling the inverter manually using the PC. In this mode, the inverter has the same drive behavior as if it was controlled using the communication interfaces. This operating mode allows for easy integration into the process data control concepts of the MOVIDRIVE® inverter, for example.

As MOVITOOLS® MotionStudio communicates with the inverter via serial interface, you can also get familiar with the functionality of the inverter's process data without bus master by specifying all setpoints manually using bus monitor (*Control* operating mode).

8.9 Manual operation

Manual operation in MOVITOOLS® MotionStudio allows for manual control (specification of speed setpoint and control commands) via the serial interfaces of MOVIDRIVE® B.

9 Bus Diagnostics

The MOVIDRIVE® inverter offers all kinds of diagnostic information for bus operation. These diagnostic options include the process data description parameters as well as the menu area P090 - P099 with the parameters for bus diagnostics. These parameters allow for easy communication diagnostics using even DBG60B.

This chapter provides an overview of the parameters for configuration, possibilities for process data diagnostics, and other bus-specific diagnostic options. The figure below shows all communication parameters of the MOVIDRIVE® inverter, which can also be used for detailed diagnostic purposes.

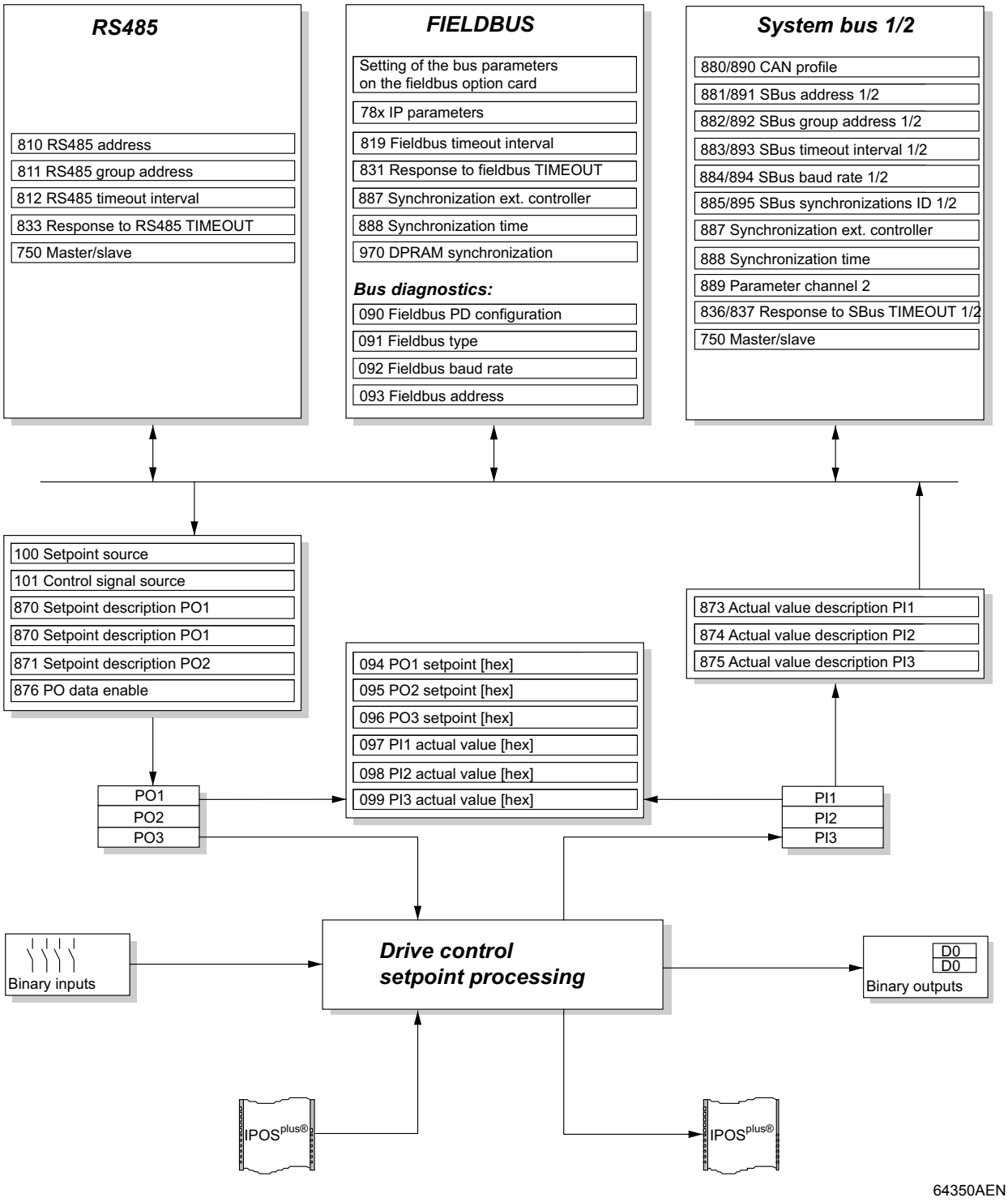
9.1 Checking the parameter setting

The parameters of the MOVIDRIVE® inverter can be read and written using any communication interface. We recommend that you use the keypad or the MOVITOOLS® MotionStudio engineering program to check the parameter setting.

For example, parameters written by a controller via fieldbus, can also be read and checked using other interfaces. For the assignment of menu number of keypad and parameter index, please refer to the MOVIDRIVE® MDX60B/61B system manual and the parameter list.

No check is required whether a parameter has been written successfully because the inverter will issue an error message if the parameter setting is incorrect.

The figure below gives an overwie of the communication parameters of MOVIDRIVE®.



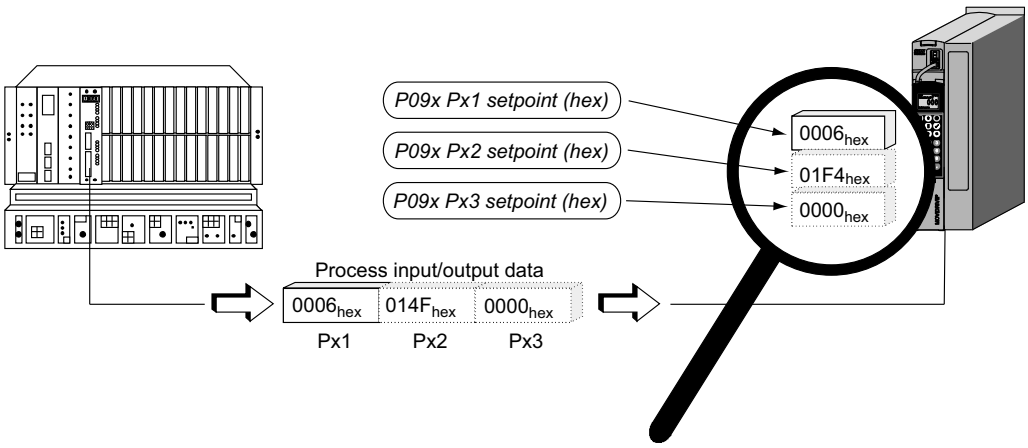
9.2 Diagnostics of process input and output data

After having set the communication parameters, you should check the process data which are sent from and to the controller. Ensure the correct sequence of bytes ("Motorola" or "Intel"), uniform number format (decimal or hexadecimal), and uniform method of counting the bits in a byte or word (bits 0 - 7 or bits 1 - 8). Convenient fieldbus master connections offer simple diagnostic options, such as LED panels on the front cover that let you diagnose the individual process data of the fieldbus.

Fieldbus monitor parameters

For simple access to these control and setpoints values, the MOVIDRIVE® inverter offers the fieldbus monitor parameters for direct access to the process data received and sent by the fieldbus system (see figure below).

- P094 PO1 setpoint (hex)
- P096 PO2 setpoint (hex)
- P098 PO3 setpoint (hex)
- P095 PI1 actual value (hex)
- P097 PI2 actual value (hex)
- P099 PI3 actual value (hex)



54676AEN

For this purpose, MOVITOOLS® MotionStudio reads the process data sent from and received by the drive inverter. Not all process data cycles are displayed due to different transmission speeds. Checking the displayed values will usually let you quickly find the cause of the fault.

Fieldbus monitor

The fieldbus monitor parameters let you control all process data in hexadecimal form using the keypad of the inverter. The fieldbus monitor (see chapter "MOVITOOLS® MotionStudio Engineering Diagnostic Software") of the MOVITOOLS® MotionStudio engineering programm even offers profile-compliant interpretation of the process data, such as the display of speed setpoints in rpm.

Unit-internal scope

For diagnosing cyclically transmitted process data, you can record input and output values over an extended period of time using the unit-internal scope and have these values displayed in MOVITOOLS® MotionStudio. In this way even faulty setpoints can be detected that only occur rarely. For more information on unit-internal scope, refer to the online help of MOVITOOLS® MotionStudio.

9.3 Diagnostic options for RS485 communication

In the event of problems in the communication of MOVIDRIVE® using the RS485 interfaces, first check all the parameters of the RS485 interfaces.

If parameters P100 and P101 are set to "RS485", you can monitor the transmitted actual values using parameters P094 - P099.

Possible causes of error

- Incorrect wiring:
  - Are the RS485 interfaces of all units properly connected?
  - Is the ground potential of the units connected with one another?
  - Is a proper cable used and has its shield been connected on both ends over a large surface area?
  - Is the bus designed as linear bus structure (without stub lines, not a tree or star structure)?
  - No additional terminating resistors must be connected to units from SEW-EURODRIVE.
- Have baud rate, timeout interval, and addresses/address ranges been set so they are suitable for the interaction of all units?
- Only one RS485 master is permitted on an RS485 bus. A master results from the
  - Master/slave function
  - IPOS$^{plus®}$ program (MOVILINK® commands)
  - Engineering PCs
  - Controllers and operator panels
- For RS485 masters that were not developed by SEW-EURODRIVE, ensure correct telegram structure, pause before the start delimiter, and character delay.
- Also make sure that RS485 telegrams are transmitted correctly in terms of structure, pause before the start delimiter, and character delay when using modems, COM servers and other gateways.
- If manual operation (of the control function in the fieldbus monitor) is still active, another process data assignment might be active than without manual operation.
- Has timeout monitoring been set correctly, and are cyclic data sent?
- If you receive an error message when accessing a parameter, check the operating status of the unit or whether the parameter lock is active. Other causes are indicated in the return codes.

Software tools

The following software tools are integrated in MOVITOOLS® MotionStudio for testing and diagnosing communication via RS485. Connect the diagnostic PC to the RS485 bus for this purpose using the UWS21B interface adapter (see chapter 4.1 "Connecting and installing RS485 interfaces").

- MoviScan (mtscan)

  MoviScan is a data and telegram monitor for the serial interface. Depending on the setting in MoviScan, you can record all bytes or only valid MOVILINK® telegrams. Specify the COM port by choosing [Setting] / [Communication ports] from the menu. The baud rate has a fixed setting of 9600 Baud.

- MoviTele (mttele)

  MoviTele is a test program you can use to send individual process or parameter data telegrams. The response of the unit is displayed in decimal or hexadecimal notation. Specify the COM port by choosing [Setting] / [Communication ports] from the menu. The baud rate has a fixed setting of 9600 Baud.

To start the programs, first set the authorization level to 100 in MOVITOOLS® MotionStudio in the menu [Settings] / [Authorization level] (password 4387). Next, you can start the "mtscan" or "mttele" program from the context menu [MOVITOOLS] / [Internal applications].

Example: Reading an index via RS485

Reading the parameter *P160 Fixed speed n11* from an inverter with RS485 address 2.

Request:

Byte	Value	Meaning	Interpretation	Help
0	0x02	Start delimiter	Request	Chapt. "Telegrams", section "Structure of response telegrams".
1	0x02	RS485 address	2	Chapter "Addressing and transmission method".
2	0x86	User data type	Acyclic 8 bytes	Chapter "Structure and length of user data".
3	0x31	Management byte	Read parameter, 4 bytes long	Chapter "Structure of the MOVILINK® parameter channel".
4	0x00	Subindex		
5	0x21	Index	8489	
6	0x29			
7	0x00	Data	150000	
8				
9				
10				
11	0xBF	Block check character		Chapter "Structure and length of user data", section "Creating block check characters"

Response:

Byte	Value	Meaning	Interpretation	Help
0	0x1D	Start delimiter	Response	Chapt. "Telegrams", section "Structure of response telegrams".
1	0x02	RS485 address	2	Chapter "Addressing and transmission method".
2	0x86	User data type	Acyclic 8 bytes	Chapter "Structure and length of user data".
3	0x31	Management byte	Read parameter, 4 bytes long	Chapter "Structure of the MOVILINK® parameter channel".
4	0x00	Parameter subindex		
5	0x21	Parameter index	8489	
6	0x29			
7	0x00	Data	150000	
8	0x02			
9	0x49			
10	0xF0			
11	0x1B	Block check character		Chapter "Structure and length of user data", section "Creating block check characters".

9.4 Diagnostic options for CAN communication

In the event of problems in the communication of MOVIDRIVE® using the CAN interfaces, first check all the parameters of this interface.

If parameters P100 and P101 are set to "SBus1/SBus2", you can monitor the transmitted setpoints and actual values using parameters P094 - P099.

Possible causes of error

- Incorrect wiring:
  - Are the CAN interfaces of all units properly connected (CAN high, CAN low, CAN Gnd)?
  - Is the ground potential of the units connected with one another?
  - Is a proper cable used and has its shield been connected on both ends over a large surface area?
  - Is the bus designed as linear bus structure (without stub lines, not a tree or star structure)? Does the total cable length match the set baud rate?
  - An 120 ohm terminating resistor must be connected each at the start and the end, or must be activated using DIP switches.
- Is the baud rate and profile setting (CANopen, MOVILINK®) the same for all units? In the CANopen profile, you can check the NMT status and the PDO parameterization with the diagnostic tool "CANopen Configuration" in MOVITOOLS® MotionStudio.
- A certain CAN identifier must only be sent by one station on the CAN. Transmit telegrams are defined by the following functions:
  - Master/slave function
  - IPOSplus® program (SCOM and MOVILINK® commands)
  - Engineering PCs
  - Controllers
- Depending on the CAN interface, an identifier may be defined only once in a unit as identifier for a transmit or receive telegram. The assignmentof identifiers results from:
  - the profile with its predefined identifiers
  - synchronization identifiers
  - an IPOS$^{plus®}$ program (SCOM and MOVILINK® commands)
- If SCOM objects that were defined in an IPOS$^{plus®}$ program are not sent, check whether transmission was activated with SCOMSTAT.
- If manual operation (or the control function in the fieldbus monitor) is still active, then setpoint transmission via CAN is blocked and another process data assignment might be active.
- Has timeout monitoring been set correctly, and are cyclic data sent?
- If you receive an error message when accessing a parameter, check the operating status of the unit or whether the parameter lock is active. Other causes are indicated in the return codes.
- Bus utilization

  A CAN telegram with 8 bytes of useful data has a maximum length of 130 bits and is transmitted at 500 kBaud in 260 µs. This means a maximum of 3 CAN telegrams can be transmitted per millisecond resulting in a maximum of 7 CAN telegrams at 1 MBaud.

  Calculate a reserve of 25% (at least 1 telegram per monitoring interval) for resending telegrams that might have been damaged by EMC.

Software tools

The following software tools are integrated in MOVITOOLS® MotionStudio for testing and diagnosing communication via CAN. The diagnostic PC is connected to a CAN network via PC-CAN interface for this purpose (see chapter "MOVITOOLS® via SBus").

- PCANview

  PCANview shows all the telegrams transmitted via CAN and provides an overview of cycle times and number of telegrams. PCANview also allows for sending individual telegrams.

- sCAN

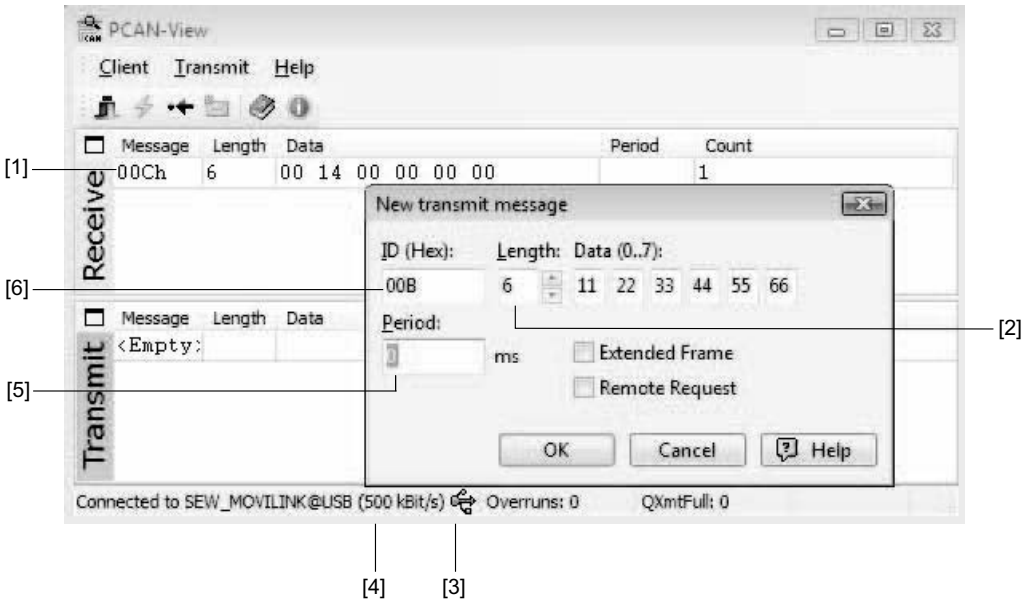
  sCAN is a data and telegram monitor with memory function for CAN networks. The display can be set according to the profile and matching the data transmitted via CAN. Filter settings, the definition of a trace file to which the read-in data is written, and other analysis options offer all the functions required for diagnosing the CAN.

To start the programs, shortcuts are created under [Start] / [Programs] / [SEW] / [sCAN and PCAN Tools] when installing MOVITOOLS® MotionStudio.

Example: Process data exchange using PCAN View

For exchanging process data between PC and MOVIDRIVE® B via SBus (CAN), connect the USB-CAN interface as described in chapter "Communication SBus (CAN) via interface adapter". In MOVIDRIVE®, you first have to choose the MOVILINK® profile using P880/890, read the SBus address P881/891, and read the SBus baud rate using P884/894 (e.g. with the DBG60B keypad).

Now start the PCAN View program. At the beginning, you have to choose a network with the baud rate matching MOVIDRIVE®. If no suitable CAN network is available for selection, you can create CAN networks using the PCAN Nets Configuration program. The created networks will be available next time you start PCAN View. The selected baud rate [4] (here: 500 kBit/s) and the USB symbol [3] are displayed in the status bar of PCAN View.



64781AXX

A new transmit message is created for sending process data:

- The identifier [6] is calculated as described in the "Telegrams" chapter as follows:

  8× SBus address + 3 (here: 8 ×1 + 3 = 11 = $00B_{hex}$)

- The length [2] of non-fragmented process data telegrams is 6 bytes = 3 words.
- Setting the period [5] to 0 ms means a telegram is sent when pressing the space bar. If the period is set to a time > 0 ms, PCAN View automatically carries out transmission at the specifed cycle. Not every Windows version can properly send if the period is set to a time under 10 ms.
- If the PC-CAN interface is properly connected and the address and baud rate are set so that they match, MOVIDRIVE® will respond to every process data telegram with the relevant message ID [1] (here: 8 ×1 + 4 = 11 = $00C_{hex}$).
- Refer to chapters 7.3 and 9.2 for a description of how to check and interpret actual values.
- If MOVIDRIVE® does not immediately respond to a process data telegram with its actual values, either the wrong identifier is used for the request or not all the parameters have been set correctly in MOVIDRIVE®.
- If the message BUSHEAVY appears in the status bar of PCAN View, an error has occurred due to incorrect connection or parameter setting.

9.5 Diagnostic options for communication via fieldbus option card

In the event of problems in the communication of MOVIDRIVE® using the fieldbus option card, first check all the parameters of this interface.

If parameters P100 and P101 are set to "Fieldbus", you can monitor the transmitted setpoints and actual values as well a some fieldbus settings using parameters P094 - P099. If one of the industrial Ethernet interfaces (DFE32B, DFE33B) is used as fieldbus option, also check the IP parameters (P78x).

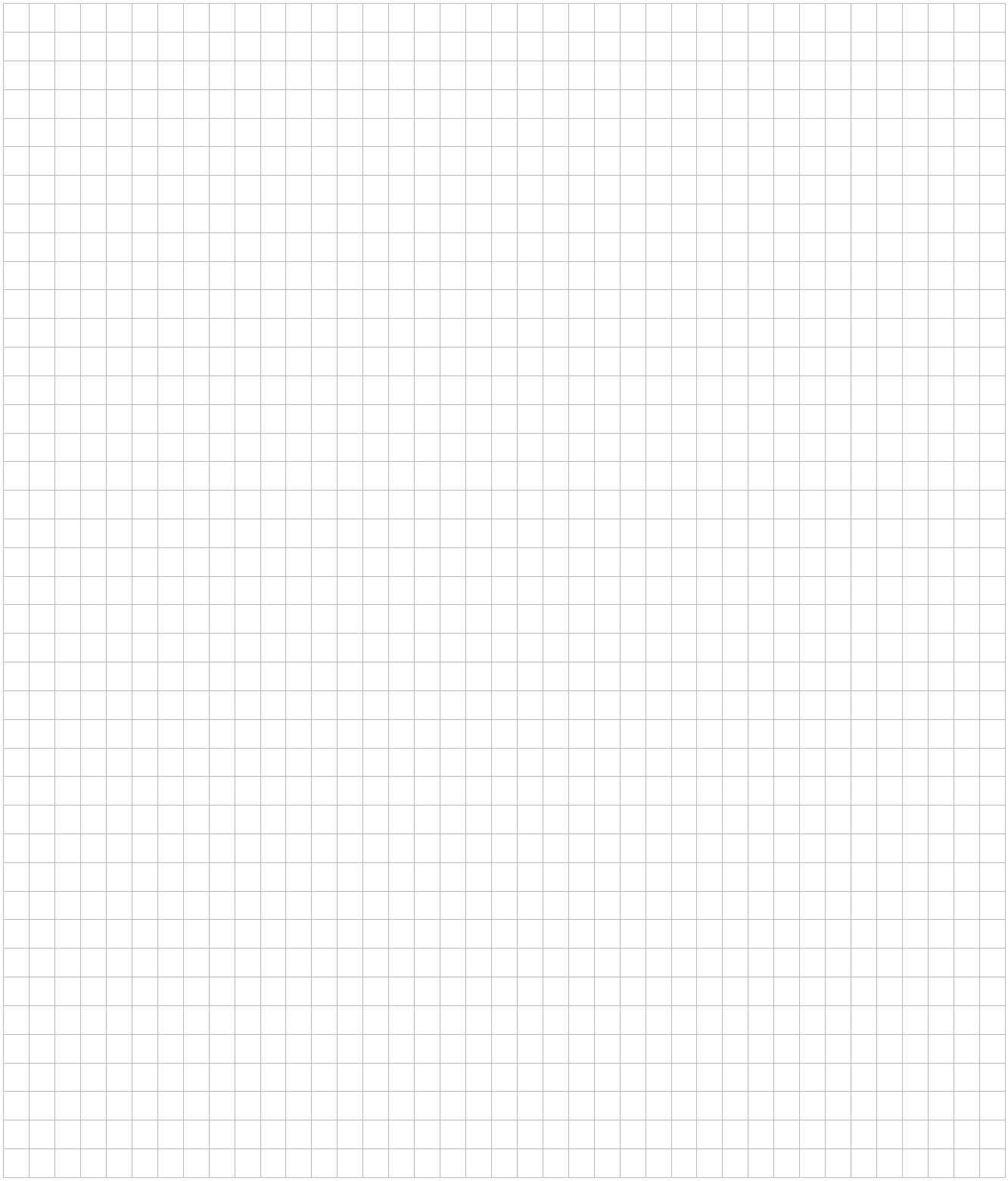
Possible causes of error

- Incorrect wiring:
  - Has the fieldbus been connected as described in the manual of the fieldbus option?
  - Is the ground potential of the units connected with one another?
  - Is a proper cable used and has its shield been connected on both ends over a large surface area?
- Have baud rate and addresses of all units on the fieldbus been set so they match one another? DIP switch settings take effect after a power-on reset.
- Which status is indicated by the LED on the fieldbus option card (see manual for the fieldbus option card)?
- What status information on the fieldbus and the individual slaves signals the fieldbus master (see documention for the fieldbus master)?
- If manual operation (or the control function in the fieldbus monitor) is still active, then setpoint transmission via CAN is blocked and another process data assignment might be active.
- Has timeout monitoring been set correctly, and are cyclic data sent?
- If you receive an error message when accessing a parameter, check the operating status of the unit or whether the parameter lock is active. Other causes are indicated in the return codes.

Software tools

You can use the fieldbus monitor in MOVITOOLS® MotionStudio for testing and diagnosing the communication. Refer to the manuals of the fieldbus option cards for more bus-specific test and diagnostic tools.

10 Index



How we're driving the world

With people who think fast and develop the future with you.

With a worldwide service network that is always close at hand.

With drives and controls that automatically improve your productivity.

With comprehensive knowledge in virtually every branch of industry today.

With uncompromising quality that reduces the cost and complexity of daily operations.

SEW-EURODRIVE
Driving the world

With a global presence that offers responsive and reliable solutions. Anywhere.

With innovative technology that solves tomorrow's problems today.

With online information and software updates, via the Internet, available around the clock.

SEW-EURODRIVE GmbH & Co KG
P.O. Box 3023 · D-76642 Bruchsal / Germany
Phone +49 7251 75-0 · Fax +49 7251 75-1970
sew@sew-eurodrive.com

→ **www.sew-eurodrive.com**