MOVIDYN®
Servo Controller

Manual

Communications Interfaces

Edition 01/97

16/042/95

0922 8764 / 0197

TÜV CERT DIN EN ISO 9001

SEW EURODRIVE

Table of contents

1 Introduction

The MOVIDYN® servo controllers are provided with several interfaces which permit the exchange of data with other units. The RS-232 is primarily used for the connection of a PC, whereas the RS-485 allows signals to be received directly. The option pcbs offer further possibilities for connecting other units. The functional independence of the interfaces and the resultant combinability of the option pcbs ensure a great degree of flexibility and a wide scope of application and allow a multitude of communications concepts to be implemented.
The following figure shows the interfaces and option pcbs which are currently available.

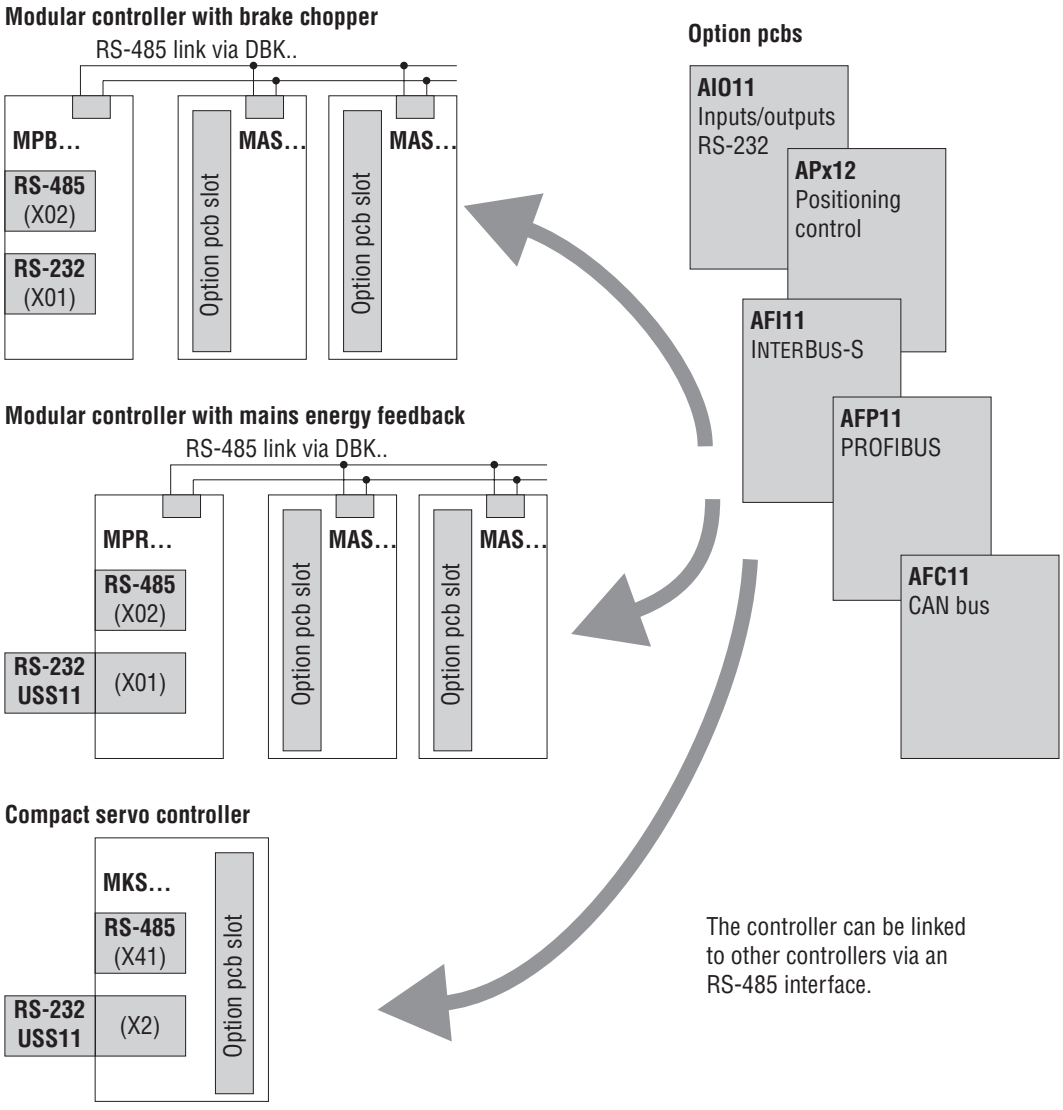


Figure 1: Interfaces for the MOVIDYN® servo controller family

Each power supply module and each compact servo controller comes with an **RS-485** interface as a basic feature.

The **RS-232** interface comes standard on **MPB... power supply modules** with brake chopper. The **MKS...** compact servo controller and the **MPR...** power supply module with mains energy feedback can be fitted with the **USS11** option pcb which offers an **RS-232**.

Options

AIO11
The AIO11 option provides additional analog and digital inputs/outputs as well as an RS-232 serial interface.

APx12
The APx12 options provides single-axis positioning control with an interface for incremental encoders (API12) or an SSI interface for absolute encoders (APA12).
The data protocols for setting parameters and programming the APx differ from those used for setting parameters in the MOVIDYN® servo controller. They are discussed in Section 5.

AFI11
The AFI11 option provides an INTERBUS-S interface to DIN 19258. It allows not only fast process data exchange but also complete adjustment of the servo controller parameters.

AFP11
The AFP11 option provides a PROFIBUS interface (DP, FMS slave) to DIN 19245. It allows control and complete parameter adjustment of the servo controller.

AFC11
The AFC11 option provides a CAN bus interface to CAN specification 2.0 Parts A and B. It allows control and complete parameter adjustment of the servo controller.

USS11
The USS11 option converts the signal levels of the X01 interface of the MPR... servo controller and the X2 interface of the MKS... controller to RS-232 signal levels. An automation unit or a PC can be connected to the 9-pin type D socket using a standard interface cable. This enables the user to set parameters and control the servo controller (e.g. with the MD_SHELL user interface).

This documentation only discusses the function and mode of operation of the serial interfaces. The communications protocol is dealt with in detail and examples are given to illustrate its use. For a detailed description of the AIO11 option pcb please refer to the "MOVIDYN® Servo Controller" Catalogue. All other option pcbs are dealt with in a separate user manual each.

2 Function of the interfaces

The unit interfaces (RS-232, RS-485, interfaces on option pcbs) allow all the servo controller parameters to be adjusted and enable all internal and external unit conditions (actual values, terminal signals) to be read.

Note that it is not possible to use both interfaces on the basic unit (RS-485/RS-232 or RS-485/USS11) at the same time, as they actually use the same transmission line.

Communication takes place on the master-slave principle, where the higher level controller (PC, PLC, IPC) acts as the master with the servo controller taking the function of the slave. This means that the drive does not initiate any send activities itself, but merely responds to enquiries from the master. The master always has control of the communications link.

The standard interface and the RS-232 interface on the AIO11 are independent of each other and have equal status. This means that if an axis module parameters set through both interfaces at the same time, the value that was sent last will be the one that is effective.

2.1 Execution principles of the communications protocol

The protocol which has been implemented was designed with regard to the following conditions:

- Shortest possible message lengths to achieve short response times
- Low implementation requirements and ease of portability to other systems
- Transmission of unit-independent data formats
- Limitation of data integrity in favour of fast protocol execution ability to increase amount of data to be transmitted to accommodate the expected functional enhancement of the unit
- Acyclic, acknowledged data traffic to minimize time-related demands on the drive.

Figure 2 illustrates the execution principles of the communications protocol of the serial interfaces. Seven different message types (frames) are used:

– ENQUIRY	Request a parameter value
– DATA	Acknowledgement with parameter value
– LONG_DATA	Acknowledgement with “long” parameter value (8 bytes)
– SELECT	Write a parameter value
– LONG_SELECT	Write a “long” parameter value (8 bytes)
– ACK (ACKNOWLEDGE)	Acknowledgement “understood”
– NACK (NOT ACKNOWLEDGE)	Acknowledgement “not understood”

An individual parameter is addressed using the index assigned to it. This assignment is dealt with in the document entitled “MOVIDYN® Parameter List”.

The messages are described in detail in the following. The message format is identical for both serial interfaces.

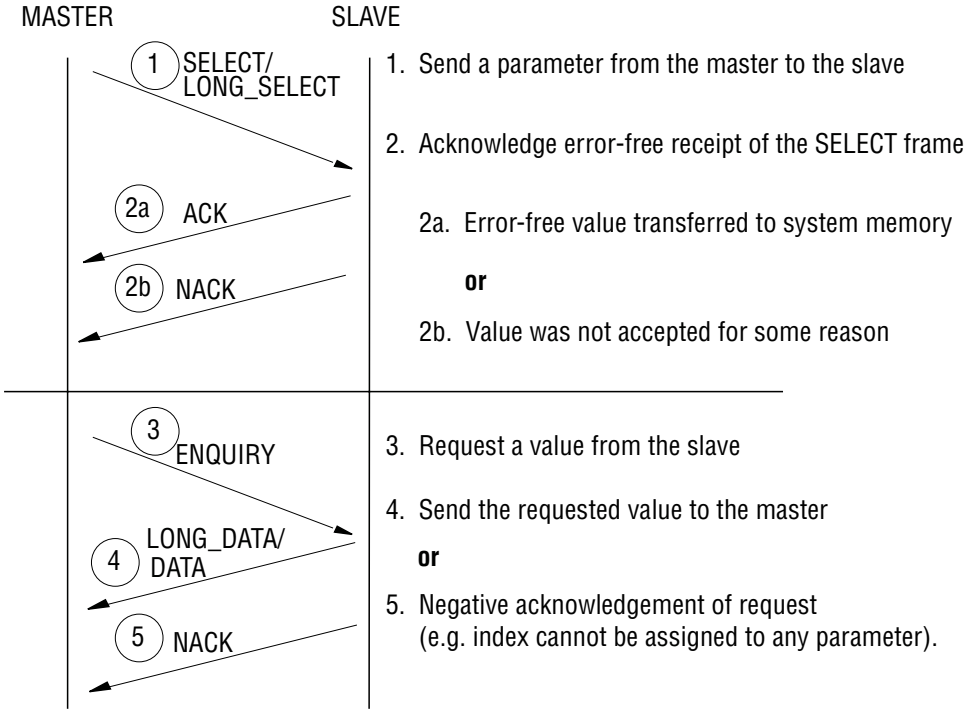


Figure 2: The various message types in the protocol

00212AEN

2.1.1 Description of MOVIDYN® message types

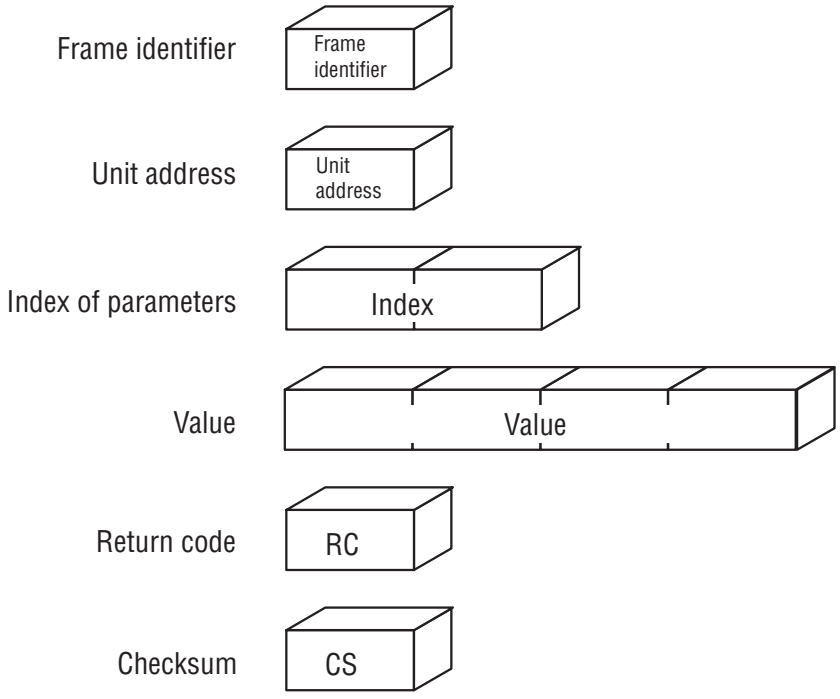


Figure 3: MOVIDYN® message components

0183AEN

Frame identifier:
Identifies whether the data sent is intended for the open- and closed-loop control system of the MOVIDYN® or applies to the API/APA. For data to/from MOVIDYN® the value is:
85 H (hexadecimal) for ENQUIRY frame
A9 H (hexadecimal) for SELECT frame
C8 H (hexadecimal) for DATA frame
D2 H (hexadecimal) for ACK frame
F3 H (hexadecimal) for NACK frame
CA H (hexadecimal) for LONG_DATA frame
AD H (hexadecimal) for LONG_SELECT frame
BB H (hexadecimal) for API/APA frame (see Section 2.1.3)

Unit address:
This identifies the servo module for which the sent message is intended within an axis system. The unit address can be set at each axis module via the S1 pushbutton on the front (see MOVIDYN® Installation and Operating Instructions).
Max. number of components that can be connected:
31 MKS... (compact servo controllers)
8 MPx... (power supply modules) with 23 MAS... (axis modules)
Value range: 0..5910

Index:
Contains a 16-bit number that specifies the requested parameter. The number is in hexadecimal format.

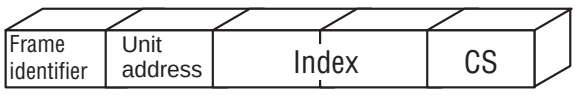
Value:
Contains the value of the parameter in the format specified in the parameter list.

Return code (RC):
The return code indicates the cause of an error in coded form. See the Appendix for table of possible return codes.

Checksum (CS):
This byte is used to check the data integrity of all the data in a frame. The checksum is the sum of all the transmitted bytes. The result then just shows the low byte.

2.1.1.1 ENQUIRY frame:

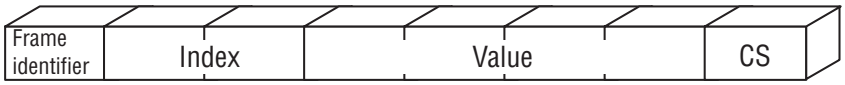
The higher-level control system sends this frame to the servo controller to read the value of the parameter encoded in the index. Following error-free receipt, the servo controller responds with a DATA frame. In the case of an error, it returns a NACK frame with the appropriate return code. The unique relationship between index and parameter is described in the "MOVIDYN® Parameter List".



00184AEN

2.1.1.2 DATA frame:

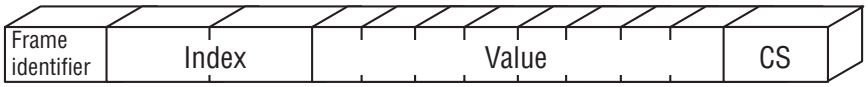
The MOVIDYN® uses this frame to send the requested data in response to a request (ENQUIRY frame) from the master.



00186AEN

2.1.1.3 LONG_DATA frame:

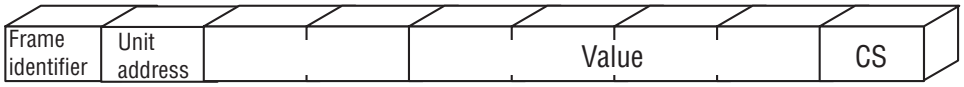
The MOVIDYN® uses this frame to send the requested data in 8-byte format in response to a request (ENQUIRY frame) from the master. Parameters returned in this format are specially marked in the "MOVIDYN® Parameter List".



00213AEN

2.1.1.4 SELECT frame:

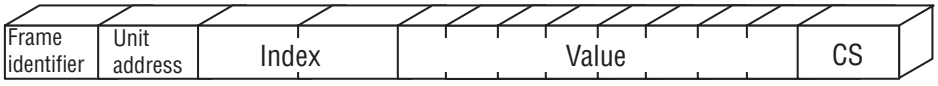
The higher-level control system sends this frame to the servo controller to overwrite a parameter in the unit. After successful receipt, the MOVIDYN® responds with an ACK frame or, in the case of an error, with a NACK frame.



00185AEN

2.1.1.5 LONG_SELECT frame:

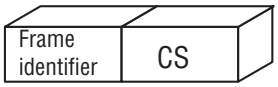
The higher-level control system sends this frame to the servo controller to overwrite an 8-byte parameter in the unit. After successful receipt, the MOVIDYN® replies with an ACK frame or, in the case of an error, with a NACK frame. The 8-byte parameters are specially marked in the "MOVIDYN® Parameter List".



00214AEN

2.1.1.6 ACK frame:

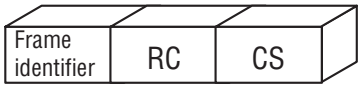
The servo controller uses this frame to acknowledge error-free receipt of the SELECT or LONG_SELECT frame.



00187EN

2.1.1.7 NACK frame:

This frame is used by the servo controller following receipt of a SELECT, LONG_SELECT or ENQUIRY frame to inform the higher-level control system that the requested service could not be carried out.



00188EN

List of possible return codes in the MOVIDYN® NACK frame

Description	Return code value (hexadecimal)
Illegal index	10
Function, parameter not implemented	11
Read access only	12
Parameter lock activated	13
Factory setting running	14
Parameter value too large	15
Parameter value too small	16
Necessary option pcb for function or parameter not installed	17
Fault in system software	18
Parameter protected from access *)	1B
End stage is not blocked	1C
Invalid parameter value	1D

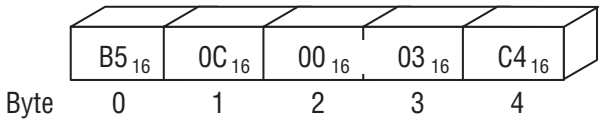
*) An index used internally by SEW has been requested, same effect as code 10.

2.1.2 Application examples

The following examples illustrate the execution sequence of the protocol and the use of the associated frames.

2.1.2.1 Reading the heat sink temperature of the axis module

The application program installed on a PLC is required to evaluate the heat sink temperature of the axis module with the address 12 for safety purposes. The address has previously been set using the pushbutton on the front of the module.

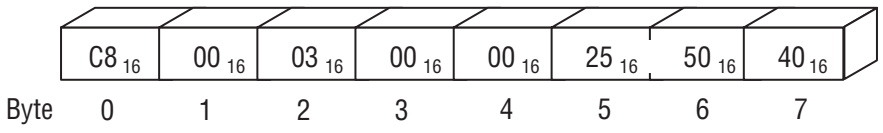


00189AXX

The PLC (master) sends an ENQUIRY frame with the following format:

Frame identifier:	B5 (identifier for ENQUIRY frame)
Address of axis module:	0C (0C hex = 12 decimal)
Index:	00 03 (0003 = heat sink temperature)
Checksum:	C4 (B5 + 0C + 00 + 03 = C4)

After error-free receipt of the ENQUIRY frame, the MOVIDYN® replies with a DATA frame containing the value for the heat sink temperature (25.5 °C)

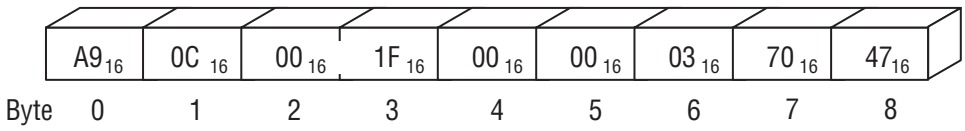


00190AXX

Frame identifier:	C8 (identifier for DATA frame)
Index:	00 03 (0003 = heat sink temperature)
Value:	00 00 25 50 = 25.50 °C
Checksum:	40 (C8 + 00 + 03 + 00 + 00 + 25 + 50 = 0140)

2.1.2.2 Writing ramp 1 to CW

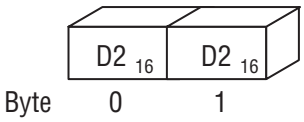
The PLC is required to set the time for the 1st acceleration ramp (CW) to a value of 3.7 seconds. The following SELECT frame must be sent to the servo controller for this purpose:



00191AXX

Frame identifier: A9 (identifier for SELECT frame)
Address of axis module: 0C (0C hex = 12 decimal)
Index: 00 1F (001F = 31 decimal = ramp 1 to CW)
Value: 00 00 03 70 = 3.7 sec
Checksum: 47 (A9 + 0C + 00 + 1F + 00 + 00 + 03 + 70 = 0147)

The servo controller acknowledges error-free transfer of the value to the system memory with an ACK frame.



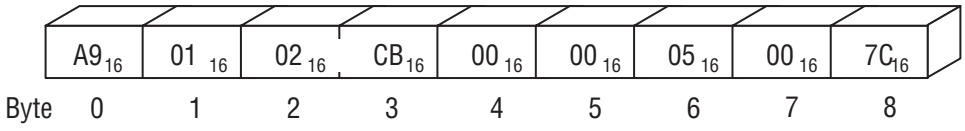
00192AXX

Frame identifier: D2 (identifier for ACK frame)
Checksum: D2

2.1.2.3 Writing an IPOS variable (only with IPOS option)

The PLC is required to write IPOS variable 5 with a value of 123000 decimal = 1E078 hexadecimal to the unit with address 1.
This requires the following sequence:

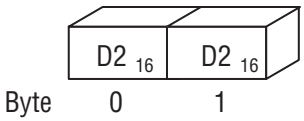
1. Write number of variable (5) to index 715 (data pointer). This selects variable 5 for the next write or read operations. Index 715 (= 02CB hex) must be written by a SELECT frame:



00215AXX

Frame identifier: A9 (identifier for SELECT frame)
Address of axis module: 01
Index: 02CB (02CB hex = 715 decimal / index 715: data pointer)
Value: 00 00 05 00 (variable 5)
Checksum: 7C (A9 + 01 + 02 + CB + 00 + 00 + 05 + 00 = 017C)

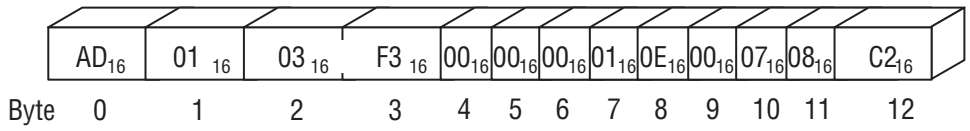
2. Axis 1 acknowledges error-free transfer with an ACK frame.



00192AXX

Frame identifier: D2 (identifier for ACK frame)
Checksum: D2

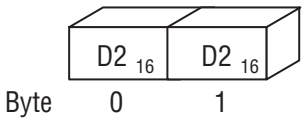
3. Write value of variable (123000 decimal = 1E078 hexadecimal) to index 1011 (data value). Index 1011 (= 03F3 hex) must be written by a LONG_SELECT frame:



00216AXX

Frame identifier: AC (identifier for LONG_SELECT frame)
Index: 03 F3 (03 F3 hex = 1011 decimal / index 1011: data value)
Value: 00 00 00 01 0E 00 07 08 (1E078 hex = 123000)
Checksum: C2 (00 + 00 + 00 + 01 + 0E + 00 + 07 + 08 = C2)

4. Axis 1 acknowledges error-free transfer with an ACK frame.



00192AXX

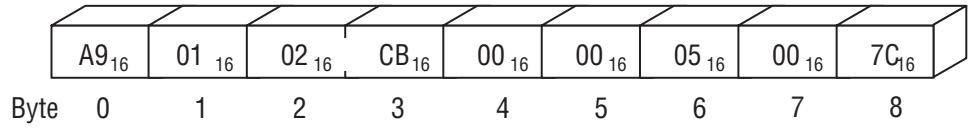
Frame identifier: D2 (identifier for ACK frame)
Checksum: D2

Variable 5 has now been assigned the value 123000.

2.1.2.4 Reading an IPOS variable (only with IPOS option)

The PLC is required to read the value of IPOS variable 5 (axis address 1)
This requires the following sequence:

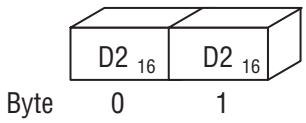
1. Write number of variable (5) to index 715 (data pointer). This selects variable 5 for the next write or read operations. Index 715 (= 02CB hex) must be written by a SELECT frame:
Frame identifier: A9 (identifier for SELECT frame)



00215AXX

Address of axis module: 01
Index: 02CB (02 CB hex = 715 decimal / index 715: data pointer)
Value: 00 00 05 00 (variable 5)
Checksum: 7C (A9 + 01 + 02 + CB + 00 + 00 + 05 + 00 = 017C)

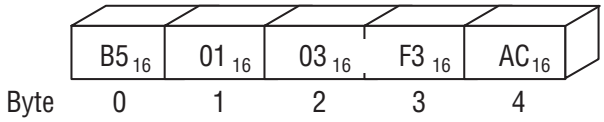
2. Axis 1 acknowledges error-free transfer with an ACK frame.



00192AXX

Frame identifier: D2 (identifier for ACK frame)
Checksum: D2

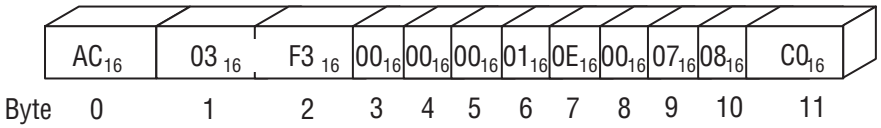
3. Request value of previously determined variable 5 from index 1011 (data value). An ENQUIRY frame is used for this purpose (1011 dec = 03F3 hex):



00217AXX

Frame identifier: B5 (identifier for ENQUIRY frame)
Address of axis module: 01
Index: 03 F3 (03 F3 hex = 1011 decimal / index 1011: data value)
Checksum: AC (B5 + 01 + 03 + F3 = 01AC)

4. Axis 1 replies with a LONG_DATA frame containing the value of variable 5 as 1E078 hexadecimal = 123000 decimal:

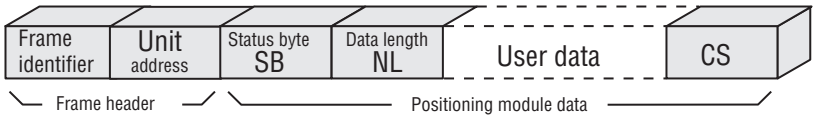


00218AXX

Frame identifier: AC (identifier for LONG_DATA frame)
Index: 03 F3 (03 F3 hex = 1011 decimal / index 1011: data value)
Value: 00 00 00 01 0E 00 07 08 (1E078 hex = 123000 decimal)
Checksum: C0 (00 + 00 + 00 + 01 + 0E + 00 + 07 + 08 = 01C0)

2.1.3 Description of message types for API / APA

The protocol for API/APA parameter adjustment is an ASCII protocol as far as the presentation of the user data is concerned. A message sent to API/APA consists of a frame header and the subsequent data for the positioning module.



00193AEN

The frame header is identical for all message types.

Frame identifier:
Shows whether the data sent is meant for the open- and closed-loop control system of the MOVIDYN® or applies to the API/APA. For data to/from API/APA, this value is: BB (hexadecimal).

Unit address:
This identifies which servo module within an axis system the message is destined for, or from which module the message originates. The reference address is set on each axis module using the S1 pushbutton on the front (see MOVIDYN® Installation and Operating Instructions).
Value range: $0..59_{10}$

Status byte (SB):
The status byte controls the data transmission between the higher-level control system and the API/APA. Figure 4 describes the format of the status byte. The status byte is bit-mapped.

Description of the bits in the status byte:

Bit 0:
This indicates whether the message is a request message (ENQUIRY frame) or a data transmission message (SELECT frame). Request messages require acknowledgement in the form of the requested data or a NACK frame. Data transmission messages are only acknowledged with an ACK frame for successful transmission or a NACK frame if the transmission contained an error. The response on receipt of a NACK frame is determined by the user.

Bit 1:
Not used.

Bit 2:
Reserved.

Bit 3:
This bit indicates whether or not the data transmission is to be acknowledged with an ACK or a NACK frame resp. This only applies to data transmission messages (SELECT frames).

Bit 4:
From firmware version V2.0, the checksum can be calculated including the axis address by setting bit 4.

Bit 5:
With bit 5 = 1, the calculation of the checksum is shortened by the most significant bit. The range of values then is 0 to 7F. This measure is necessary because certain values used for the checksum were evaluated as the start byte of a new data transmission and this led to collisions on the bus. This bit is no longer relevant from IPOS V.71 and system V.15.

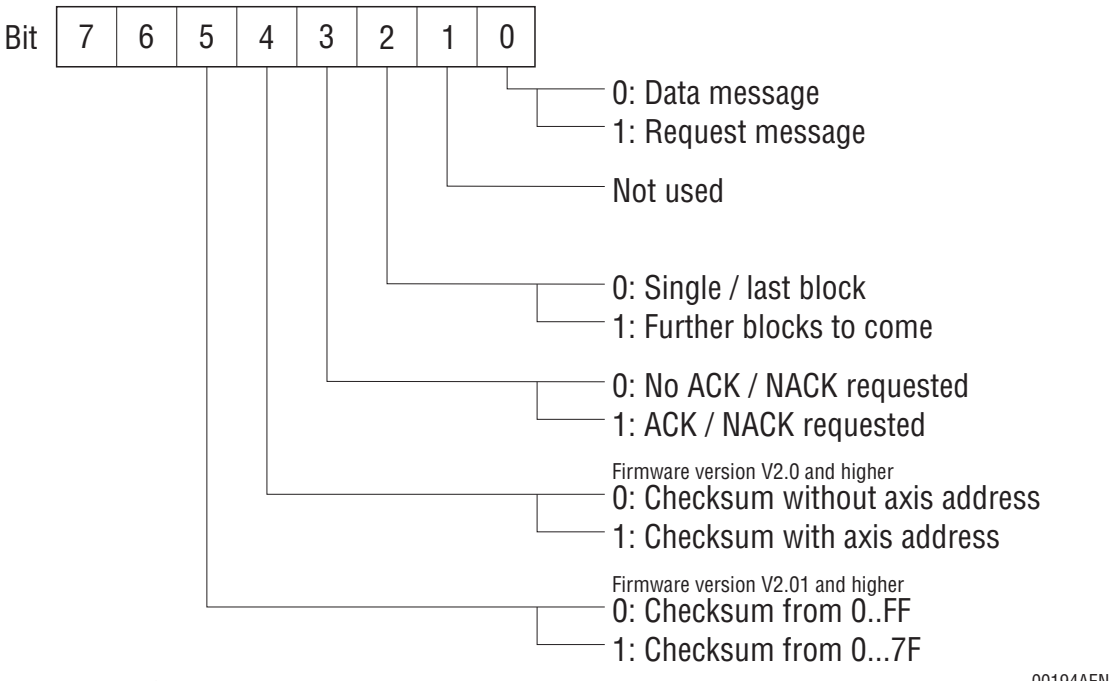


Figure 4: Format of the status byte

Data Length (NL):
This byte contains the number of user data bytes including the checksum byte CS. The status byte and the data length byte are not included. The data length is in hexadecimal format.

User data:
The commands in ASCII format contained in the manual constitute the user data.

Checksum (CS):
This byte is used to check the integrity of all the data in a frame.
The checksum byte is obtained from the following:
CS = status byte (SB) + data length (NL) + sum of user data bytes

If bit 4 is set in the status byte:
CS = axis address + status byte (SB) + data length (NL) + sum of user data bytes

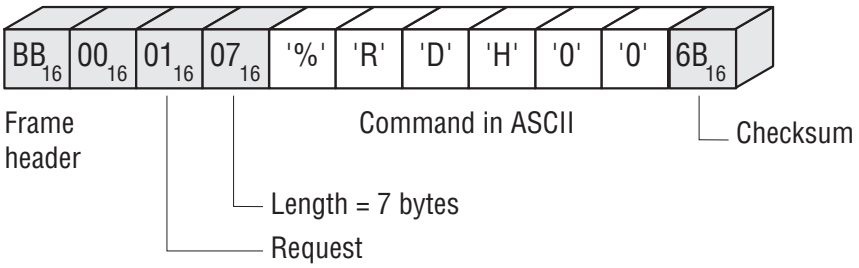
The bytes are added byte by byte, ignoring the carry digits.

2.1.3.1 ENQUIRY frame:

This type of message is used to request data from the API/APA. For example, the value of a variable is to be requested from the API. The required command is contained in the user manual and is as follows:

%**RD H00**: read the contents of variable H00

A request message with the following format is to be sent:



00195AEN

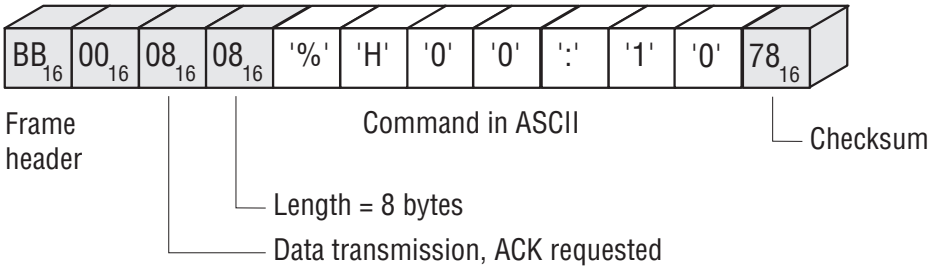
Note:
It is not necessary to insert a space between 'D' and 'H'.

2.1.3.2 SELECT frame:

Data can be transmitted to the API/APA with this message type. For example, a variable in the API/APA is to be assigned the value 10. The command for this is:

%**H00:10** ; set variable H00 = 10

A data transmission message in the following format is required:

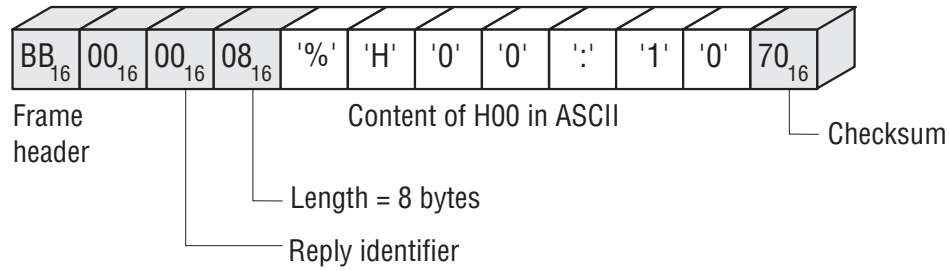


00196AEN

The SELECT frame is either acknowledged with an ACK frame for successful data transmission, or a NACK frame for an abortive transfer, depending on the status byte condition.

2.1.3.3 DATA frame

The API/APA acknowledges a correct request with a reply message, which basically has the following format, but may differ in the user data according to the requested value:

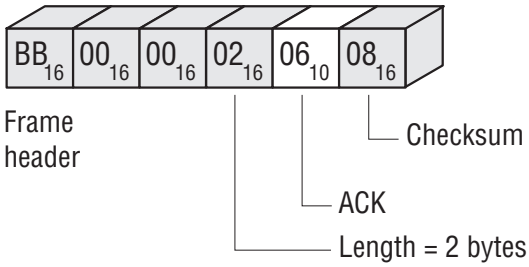


00197AEN

An incorrect request will be acknowledged by a NACK frame.
The user must extract the frame header, status byte, length byte and checksum from the data to obtain the user data in ASCII format.

2.3.1.4 ACK frame:

The API/APA acknowledges successful data transmission on receipt of a SELECT frame with an ACK frame with the following format:

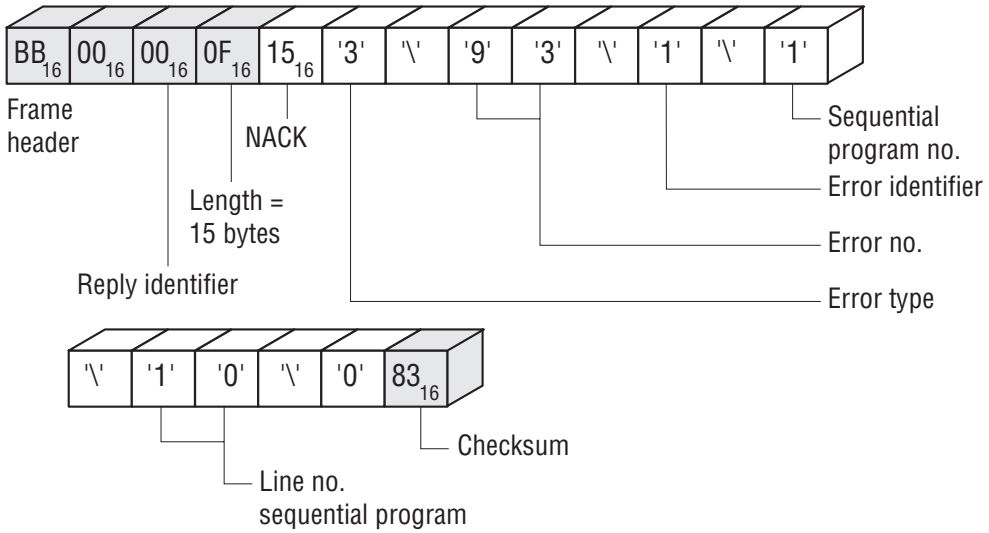


00198AEN

The value for ACK = 06_{10}.

2.3.1.5 NACK frame:

If the data transmission contains errors (noise on the data line, syntax errors etc.) the API/APA acknowledges a request message or an incorrect SELECT frame with a NACK frame as shown below:



00199AEN

The value for NACK = 15_{16}.

In addition to the number of the data transmission error, the NACK frame also contains the program number currently selected and the program line number of the application program in use in the API/APA, which was processed at the time the error was reported. If the API/APA is not in automatic mode, both the program and line numbers will be transmitted as “- -” or “- - -”.

The **error type** parameter indicates whether the error was a syntax error or an asynchronous error:

Error type = 3: syntax error
Error type = 4: asynchronous error

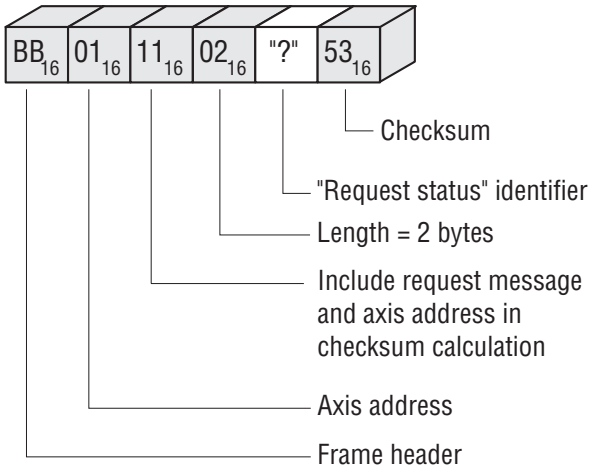
The **error identifier** parameter indicates whether the error acknowledged by the NACK frame was a minor error that does not require resetting, or whether a reset must be initiated with the %RES command:

Error identifier = 0: no error present
Error identifier = 1: error, but no reset necessary
Error identifier ≥ 2: error; reset via % RES necessary

2.1.3.6 Application example:

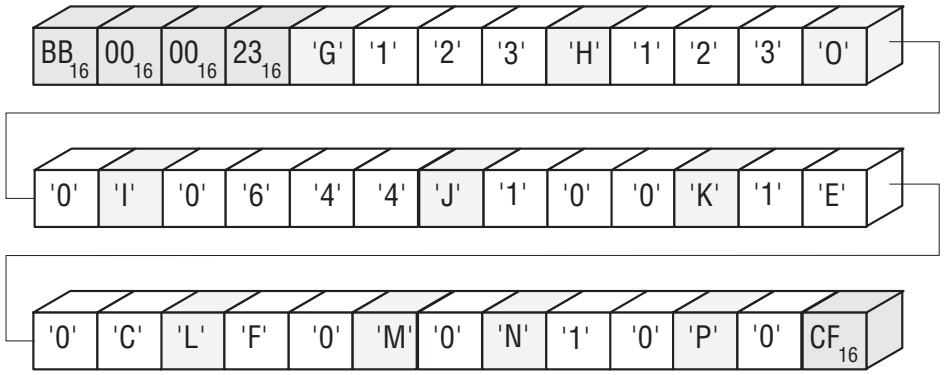
Reading the API/APA status

Error messages that occur asynchronously can be displayed by requesting the API/APA status. To request the status, a message containing the command "?" is sent to the API/APA.



00227AEN

Format of the response string in response to an "?" API status request:

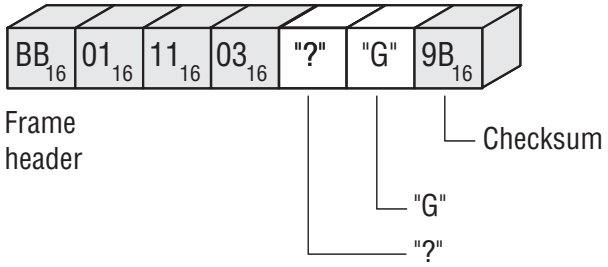


00200AEN

The response message contains the following status messages in ASCII format:

- G... Actual position (decimal)
- H... Target position (decimal)
- 0... Lag error (decimal)
- I... Status of positioning processor in the API/APA (hexadecimal, 4-digit)
- J... Override (decimal)
- K... Status of digital inputs (hexadecimal, 4-digit)
- L... Status of digital outputs (hexadecimal, 2-digit)
- M... Currently active program no. (decimal)
- N... Current line number (decimal)
- P... Status of hardware limit switches (decimal, 1-digit)
  where
  - 0 = no limit switch closed
  - 1 = positive limit switch closed
  - 2 = negative limit switch closed
  - 3 = both limit switches closed

Firmware version 2.0 and higher, the process data can also be called up separately.
Example: **?G** → request actual position only:



00228AEN

Note:
If the API/APA is not automatic mode, hyphens are entered for the program and program line numbers as shown below:

Program number: 'M' '-' '-'
Line number: 'N' '-' '-' '-'

If there is an error in the API/APA, the sequence shown below is included in the response message. An error sequence is inserted after the limit switch status data. The last character always represents the checksum. Error sequence: 'Z' a '\' b '\' c '\' d '\' e '\' f

Meaning:

'Z' = Identifier for error string
a = Error identifier
b = Error number
c = Error type
d = 1st parameter (current program number in automatic mode, otherwise set to '0')
e = 2nd parameter (current line number in automatic mode, otherwise set to '0')
f = 3rd parameter (always 0)

The individual parameters are separated by the backslash '\' character. Appendix 2 contains a list of all the error messages that may occur.

3 RS-232 interfaces

3.1 Technical data of the RS-232 interfaces

- Standard: DIN 66020 (V.24)
- Baud rate 9600 baud
- Start bits: 1 start bit
- Stop bits: 1 stop bit
- Data bits: 8 data bits
- Parity: None
- Data direction: Bidirectional
- Operating mode: Simplex, asynchronous

3.2 Connection to the RS-232 interface

The connection to the RS-232 serial interface on the units is made via a 4-core screened cable, the screen being connected to ground at one end only. The unit interfaces X01 and X11 are provided as 9-pin type D connectors, only 4 connections (pins 2-5) being used. The remaining connections are blank. Figure 5 shows a schematic of the connector pinout. Connection 2 and pin 3 are the data lines. The PC controls the send direction of the data on the subsequent RS-485 cable via pin 4. Pin 5 is the ground connection for the units.

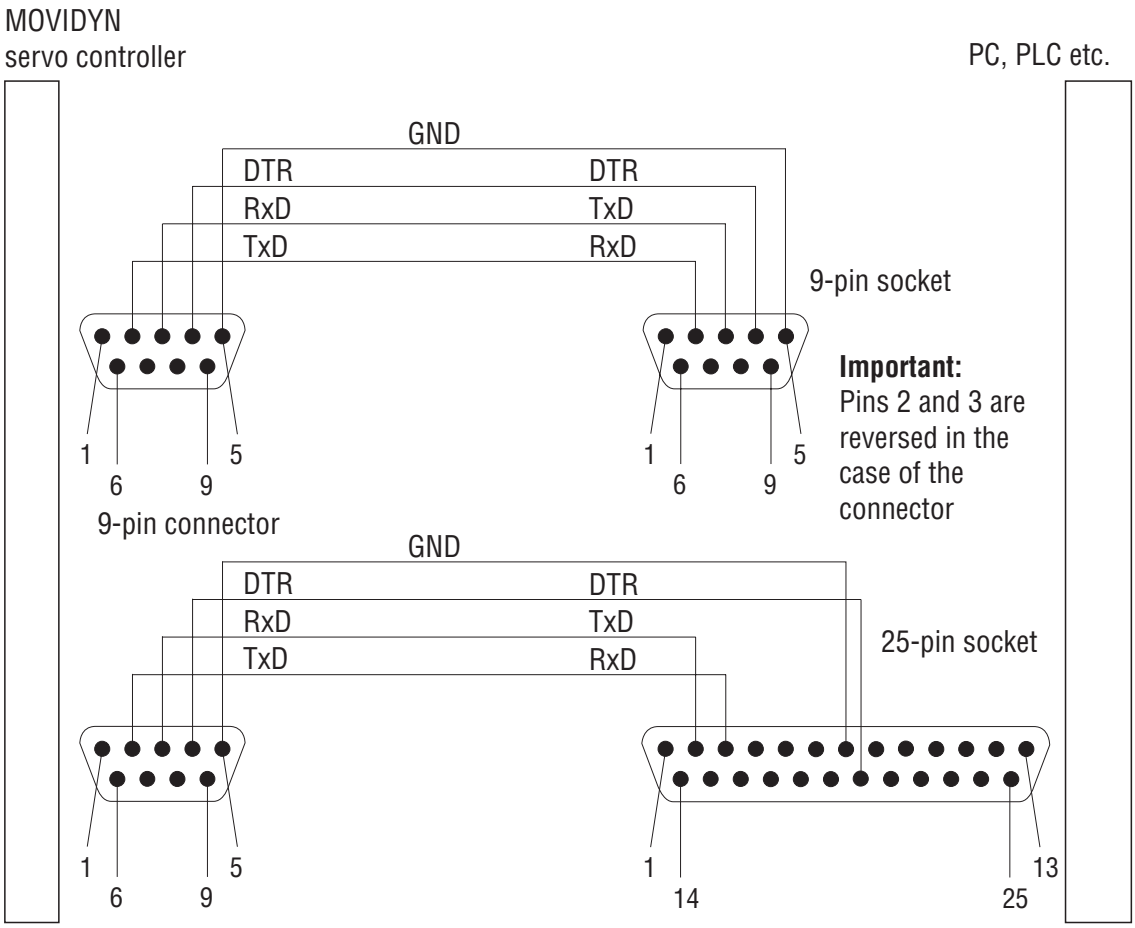


Figure 5: RS-232 connector pinout

00178BEN

4 RS-485 interfaces

4.1 Technical data of the RS-485 interfaces

- Standard: RS-485
- Baud rate: 9600 baud
- Start bits: 1 start bit
- Stop bits: 1 stop bit
- Data bits: 8 data bits
- Parity: None
- Data direction: Bidirectional
- Operating mode: Half duplex, asynchronous

4.2 Connection to the RS-485 interface

The RS-485 interface is connected to the power supply module / compact servo controller using pins 1, 2, 3 on the appropriate terminal block. This interface should be used if the distance between the higher-level control system and the drive unit is greater than 5 m, as the susceptibility to interference of the RS-485 interface is considerable less than that of an RS-232 interface due to transmission of differential clock signals. However, in this case the higher-level control system must have an RS-485 output (e.g. plug-in card, RS-485 adapter).
A twisted, shielded two-wire cable is used as transmission medium. The cable is connected to pins 1 and 2 of each terminal block and the shield is connected to pin 3. The polarity of the connections to pins 1 and 2 must be observed with respect to the connections to the higher-level control system. Figure 6 shows the connection to the RS-485 interface of the various servo controllers.

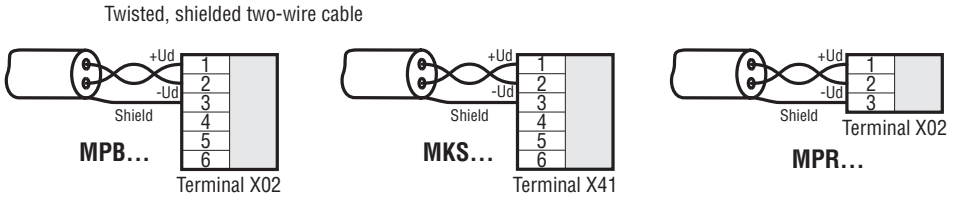


Figure 6: RS-485 connection to MPB..., MKS... and MPR..

00219AEN

Important:
If the RS-485 interface is used, the RS-232 interface on the MPB... or via USS11 on the MPR... or MKS... may not be used at the same time.

For control of the send direction of the data, please refer to the technical specification of the particular RS-485 connection to the higher-level control system.

5 Control of the direction of communication on the serial interfaces

In controlling the data direction on the serial interfaces, certain peripheral conditions must be observed concerning the timing sequence. As can be seen from Figure 1, the axis modules are connected to each other via an RS-485 link. The link between the RS-485 and RS-232 level is via two servo converters. As the RS-485 interface operates in simplex mode, a bus station must always reverse the data direction for sending and receiving.
Control of the send direction is handled by a signal of the sender's serial interface; this signal is given the name DTR in accordance with the RS-232 standard. Figure 7 shows the timing sequence for switching the DTR signal when sending and receiving data via an RS-232 interface in a timing diagram.

Important:
The signals in Figure 7 are shown with the signal levels of the RS-232 interface (± 10 V). The valid levels for control signals are:

TTL	RS-232
0	-10 V
1	+10 V

For design reasons, the switching of the DTR signal differs for the various servo controllers:

Servo controller	DTR Handling	Remarks
MPB	DTR switched	
MPR, MKS with FIS31 (Part no. FIS31: 821 595 2.10)	DTR switched	
MPR, MKS with FIS31 (Part no. FIS31: 821 595 2.11/2.12)	DTR not switched, but DTR permanently at "1" level (+10V)	
MPR, MKS with USS11	DTR not required	USS11 reverses data direction automatically

Table 1: Method of handling the DTR signal for the various servo controllers

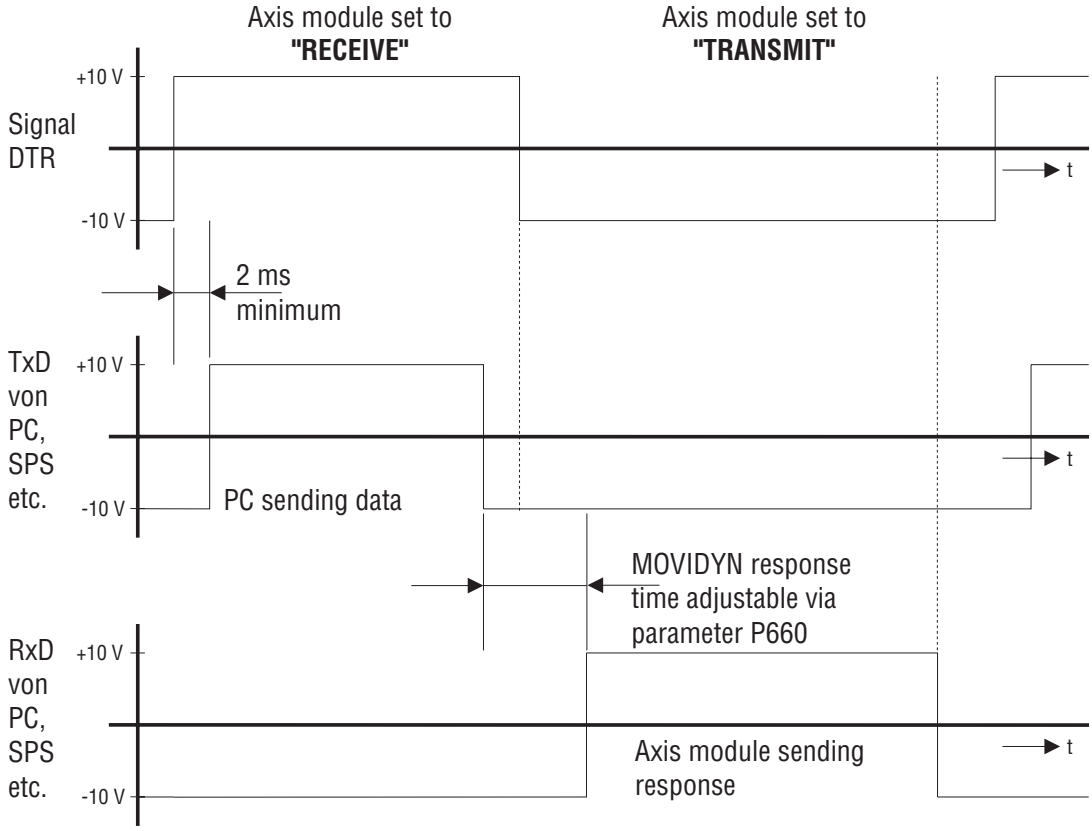


Figure 7: Signal level on the interface lines

00182BEN

Operational sequence description:

a) The DTR signal must be switched to send at least 2ms before starting to transmit the send data.

b) After transmission of the data (send buffer empty), the DTR signal is switched back to receive after a delay of at least 1ms. This time is the time taken to send one byte and ensures that the last bits of the final byte of an ongoing transmission are actually transmitted. To simplify the link-up to a PLC, it is possible to program a minimum delay time in the MOVIDYN® that must elapse before a servo controller responds. To do this, the 07SO <time parameter> (P660) can be set in the range 0 to 200 ms using MD_SHELL.
The DTR signal must be reset to receive no later than t = < time > -1ms after the last byte from the host has been sent, to enable the response from the MOVIDYN® to be received without any problems.

c) A new transmission may only start at the earliest 2ms after receipt of the final byte of the response from the MOVIDYN® by resetting the DTR signal as before.

Tip:
Usually, with most communications processors, the RTS signal can be used as DTR.

6 Interface monitoring functions for APx operation

6.1 Timeout monitoring during data transmission

In addition to using the checksum byte for checking the integrity of the data transmission, other data transmission monitoring features are used. Once data transmission has begun, the complete message must be transmitted in full after a total of t = 500 ms at the latest. If this is not the case, all data received prior to this timeout is deleted and the API/APA will wait for a new valid frame header. This ensures that a message cut-off in mid-transmission is recognized and a successful retransmission can subsequently be initiated at the correct place.

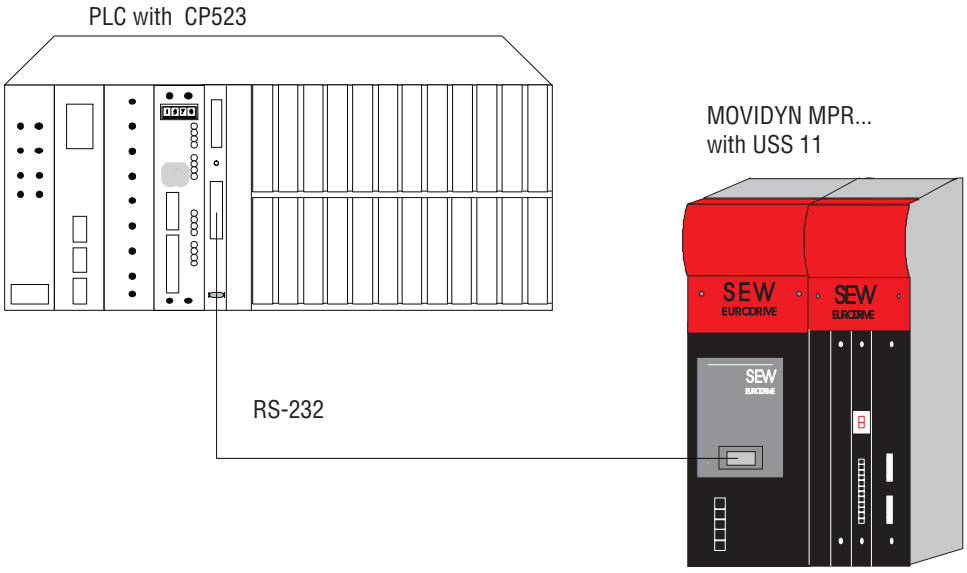
6.2 Timeout monitoring in remote mode

The API/APA can be switched to "remote mode" mode using the "%+R" command. This allows direct monitoring of the positioning control by the higher-level control system (corresponds to "Manual operation" menu in MD_POS).
This then takes place by issuing the standard commands as described in the API/APA user manual. In this mode, timeout monitoring in the API/APA starts immediately when an activation command is issued. If no further communication follows between the higher-level control system and the API/APA during this period, an error is assumed and the drive is stopped. The time interval is t_{max} = 500 ms.

This function enables a breakdown in communications between the API/APA and the higher-level control system to be detected to prevent the drive from operating without any form of control. To prevent this timeout error, continuous communication must be maintained with the API/APA in this operating mode. The simplest way is to request the status continuously with the "?" command. The timeout counter in the positioning control will then be reset with every request. "Remote mode" can be disabled with the "%-R" command, causing timeout monitoring to be deactivated again.

7 Communicating with a PLC

For a PLC and a servo controller to communicate, the PLC must be fitted with a communications module (communications processor, CP) which allows the transmission protocol to be freely programmed. In the case of the Simatic S5, for example, you may use the CP523 module to connect the MOVIDYN® servo controller to the PLC via the RS-232 interface. This communications module supports the communications mode “transparent”, i.e. the transmitted protocol can be freely programmed by the user.



00229AEN

Figure 8: PLC and MOVIDYN® communicating via RS-232

7.1 System requirements

To establish a communications link between a Simatic-S5 and the MOVIDYN® servo controller using the RS-232 serial interface, you will need the following hardware components:

1 MPR... or MKS... with USS 11 or 1 MPB...
1 CP523 communications module for S5 115-U
1 interface cable (see Fig. 9)

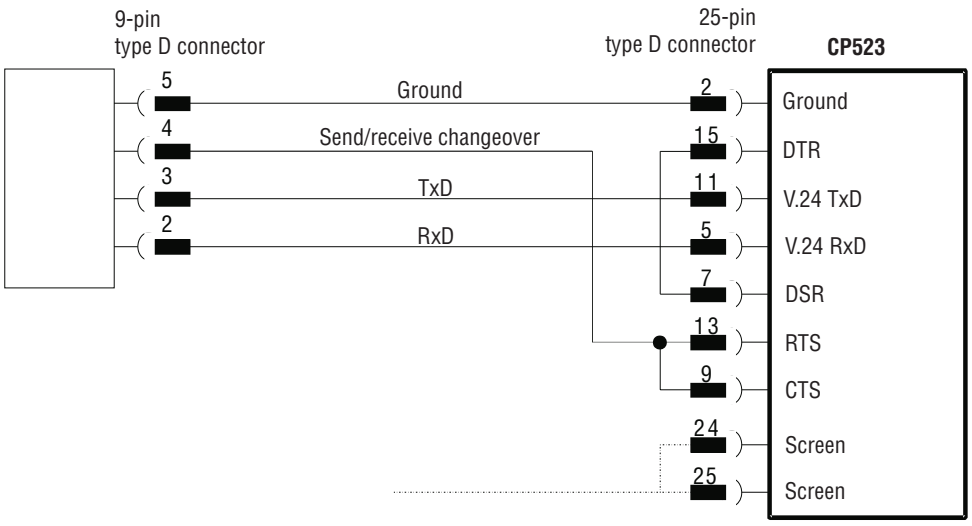


Figure 9: Pin assignment of the interface cable for the CP523 - MOVIDYN®

00230AEN

7.2 Initializing the PLC module CP523 in the application program

In the communications mode the CP523 communications processor allows up to 256 bytes to be transmitted with a CPU request. In the communications mode “transparent” the CP523 does not interpret any characters. All data transmitted with a send request is therefore directly output on the interface. At the same time the CP523 stores all data received directly in the receive buffer allowing the received data to be picked up with the appropriate CPU request.

As the length of the frames used for communication with the MOVIDYN® servo controller varies, the maximum frame length must be set to accommodate the longest possible frame. API/APA frames may be longer than other frames.

To initialize the communications interface a parameter block is transmitted to the CP523. You may use the CPU request “Transmit parameter data” to do this. The following tables show the settings for these parameter blocks which are used to initialize communication with the MOVIDYN® servo controller.

Byte	Meaning		Value for MOVIDYN®
0	Request no. “Transmit parameter data”		90 hex
1	Parameter block no.		00 hex
2	Baud rate:	9600 baud	08 hex
3	Parity check:	none	04 hex
4	Busy signal:	NO	00 hex
5	Interface:	V.24	01 hex
6	Data format: 10 bits, 1 start bit + 8 data bits + 1 stop bit		05 hex
7	Hardware handshake:	ON	01 hex (important for DTR/RTS switching)

Table 2: Transfer buffer assignment for parameter block 0 when using the “Transmit parameterizing data” request.

Byte	Meaning		Value for MOVIDYN®
0	Request no. “Transmit parameter data”		90 hex
1	Parameter block 7, Communications mode “transparent”		71 hex
2 + 3	Character delay time:	10ms	0001 hex
4 + 5	Maximum frame length:	256 bytes	0100 hex
6 + 7	Irrelevant		0000 hex

Table 3: Transfer buffer assignment for parameter block 7 when using the “Transfer parameterizing data” request.

For details of the parameter data transfer please refer to the user manual of the CP523 communications processor.

7.3 Example: Reading the heat sink temperature

To read out the heat sink temperature of a servo controller with the address 0, the CP523 is to send an ENQUIRY frame to the servo controller which should have the format shown in Section 2.1.1.1. The transfer buffer and the transfer procedure are as follows:

1) Execute the CPU request “Send frame” (A001 hex) with a data length = 5 bytes (the ENQUIRY frame has a length of five bytes) in the application program.
2) In the transfer buffer, transfer the data to the CP523.

Byte	Meaning		Value for MOVIDYN®
0	Start delimiter SD		B5 hex
1	Address:	0	00 hex
2 + 3	Index of the parameter “heat sink temperature”		0003 hex
4	Checksum: B5+00+00+03 = B8		00B8 hex
5 - 7	Irrelevant		00 hex

Table 4: Transferring the ENQUIRY frame to the CP523

3) The CP523 automatically sends the ENQUIRY frame to the servo controller.
4) The CP523 automatically receives the DATA frame from the servo controller and stores it in the receive buffer.
5) The contents of the receive buffer are transferred to the application program with the CPU request “Receive frame” (A080 hex): The transfer buffer now contains the DATA frame as shown in table 5.

Byte	Meaning	Value for MOVIDYN®
0	Start delimiter SD	C8 hex
1 + 2	Index of the parameter “heat sink temperature”	0003 hex
3 + 4	More significant part of the parameter value	e.g. 0000 hex
5 + 6	Less significant part of the parameter value	e.g. 2500 hex
7	Checksum: C8+00+03+00+00+25+00=	F0 hex

Table 5: Contents of the transfer buffer after receiving the DATA frame

6) The checksum in byte 7 of the receive buffer is then evaluated. If the recalculated checksum corresponds to the checksum in byte 7, the frame was received correctly. If not, the read process must be repeated.

In the above example a heat sink temperature of 25.0 °C was read out.

7.4 Example: Writing the parameter “T11 ramp up”

To set the parameter “T11 ramp up” of a servo controller with the address 0 to 3.7s, the CP523 is to send a SELECT frame to the servo controller which should have the format shown in section 2.1.1.4. The transfer buffer and the transfer procedure are as follows:

1) Execute the CPU request “Send frame” (A001 hex) with a data length = 9 bytes (the SELECT frame has a length of 9 bytes) in the application program.
2) As the SELECT frame is 9 bytes long, two transfer buffers must be transferred, one after the other, to the CP523. The first transfer buffer contains byte 0 to byte 7 of the SELECT frame, while the second transfer buffer contains only the checksum.

Byte	Meaning	Value for MOVIDYN®
0	Start delimiter SD	A9 hex
1	Address (of no significance for RS-232)	00 hex
2 + 3	Index of the parameter “T11 ramp up”	001F hex
4 + 5	More significant part of the parameter value	0000 hex
6 + 7	Less significant part of the parameter value	0370 hex

Table 6: First transfer buffer of the SELECT frame

Byte	Meaning	Value for MOVIDYN®
0	Checksum: A9+00+00+1F+00+00+03+70=3B	3B hex
1 - 7	Irrelevant	00 hex

Table 7: 2nd transfer buffer of the SELECT frame

3) After transfer of all 9 bytes the CP523 automatically sends the SELECT frame to the servo controller.
4) The CP523 automatically receives the ACK frame (or in the event of an error the NACK frame) from the servo controller and stores it in the receive buffer.
5) The contents of the receive buffer are transferred to the application program with the CPU request “Receive frame” (A080 hex). In this example the transfer buffer contains the ACK frame shown in Table 8.

Byte	Meaning	Value for MOVIDYN®
0	Start delimiter SD	D2 hex
1	Checksum: D2	D2 hex

Table 8: Contents of the transfer buffer after receiving the DATA frame

6) The checksum in byte 1 of the receive buffer is then evaluated. If the recalculated checksum corresponds to the checksum in byte 1, ACK is (SD = FCS = D2 hex), the frame was received correctly. If not, the read process must be repeated.

Appendix 1

List of all API/APA error messages which may be accessed via the interface

a) Error messages during data transmission

Meaning	Error number
Program memory full	50
Positioning module off-line	51
Error flag index	52
Error K ID	53
K index missing	54
Error assignment	55
Wrong K index	56
Error T ID	57
Error assignment value	58
Error delimiter ':'	59
Wrong T index	60
Wrong H index	61
Wrong C index	62
Wrong assignment operator	63
Program or line number missing	64
Wrong program number	65
Program does not exist	66
Wrong stop string	67
Wrong output no.	68
Wrong output mask	69
CAN not supported yet	70
Wrong output port	71
Wrong R or R ID	72
Wrong position value	73
Wrong speed value	74
Wrong V ID	75
Remote activation/deactivation not permitted	76
Error program name	77
Program already exists	78
Wrong line number	79
Program does not exist	80
Illegal command	81
Wrong program or line number	82
Error ID ')'	83
Error compare operation	84
Wrong input no.	85
Wrong input port	86
Error ID '('	87
Wrong compare value	88
Only comparison with 0 permitted	89
Wrong Z index	90
Wrong time	92
Command not recognized	93
Wrong M index	94
Not in manual mode	95
Error programming flash EPROM	96
Illegal data transmission in automatic mode	97
Checksum error	98

b) Asynchronous error messages which may occur during API/APA operation

Meaning	Error number
Emergency stop	27
Hardware limit switches reversed	29
Error boot synchronization	40
Watchdog tripped	41
Lag error	42
Positive hardware limit switch approached	50
Negative hardware limit switch approached	51
Positive software limit switch approached	52
Negative software limit switch approached	53
Reference position not defined	54
Addressed hardware not installed	56
Selected program does not exist	57
Jump on non-existent line number	58
Called subroutine does not exist	59
Target position outside travelling range	60
Division by zero	63
Max. subroutine nesting depth reached	64
Command error positioning controller...	65
Program memory full	66
Timeout in remote mode	67
Target position not reached	68
No feed enable	69
Error code SSI interface	70
Code 1:	Error SSI interface
Code 2:	Communications error SSI interface
Code 3:	Parity powerfail error SSI encoder
Code 4:	Lag error SSI module
Error code CAN	71
Code 1:	CAN timeout
Code 2:	CAN receive buffer full
Code 3:	CAN controller overflow
Code 4	CAN controller error
Code 100: error CAN module	
Index overflow (index. address)	72
Illegal command during axis movement	73
Position outside calculation limit	74

Asynchronous error messages with error nos. < 50 cannot be reset by addressing the API/APA with the %RES command. This can only be done by resetting the input terminals of the axis module by pressing the s1 key on the axis module or by effecting a reset via the interface, not however by addressing the MOVIDYN® system software by sending a MOVIDYN® message.

Appendix 2

Additional API/APA interface commands

%RES:
Error reset with error nos. ≥ 50

%SAV:
Save machine parameters, travelling programs, table positions and variables (variable no. ≥ 50) in non-volatile memory of the API/APA. Saving onto non-volatile memory can take up to 20 s max. The command is only acknowledged after the storing operation onto non-volatile memory has been completed.

We are available, wherever you need us.
Worldwide.

SEW-EURODRIVE right around the globe is your competent partner in matters of power transmission with manufacturing and assembly plants in most major industrial countries.

SEW-EURODRIVE GmbH & Co · P.O.Box 30 23 · D-76642 Bruchsal/Germany
Tel. +49-7251-75-0 · Fax +49-7251-75-19 70 · Telex 7 822 391
http://www.SEW-EURODRIVE.com · sew@sew-eurodrive.com