Serial communication using intelligent field devices

- using COMPAX servo-control as an example-

Communications using:

- **RS232**
- **RS485 ASCII / binary**
- **RS485 field bus**
- **Interbus-S**
- **Profibus**
- **CAN-bus**

november 96

HAUSER
We automate motion

CERTIFIED DIN EN ISO 9001 QUALITY SYSTEM
Reg. Nr. 36 38

Head Office
Parker Hannifin GmbH
EMD-HAUSER
P. O. Box: 77607-1720
Robert-Bosch-Str. 22
D-77656 Offenburg, Germany
Phone: +49 (0)781 509-0
Fax: +49 (0)781 509-176
http://www.Parker-EMD.com

Great Britain:
Parker Hannifin plc
EMD-Digiplan
21 Balena Close
Poole, Dorset
BH17 7DX UK
Phone: +44 (0)1202 69 9000
Fax: +44 (0)1202 69 5750
http://www.Parker-EMD.com

29.09.99 192-020018 N1

Parker

Contents

Note

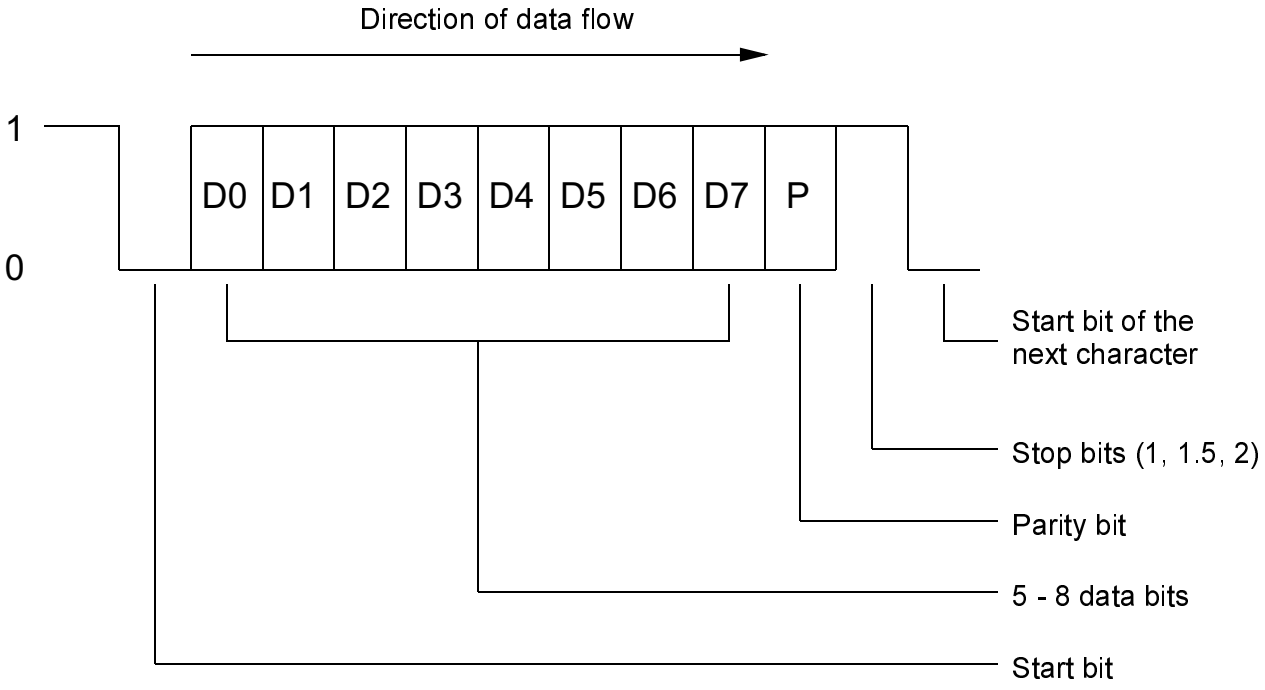
The COMPAX parameters are described here as an example only. The current configurations which apply can be found in the operating instructions of the individual bus systems.

Operating instructions available for individual bus systems:

- RS485: article no.: 190-040026
- Interbus-S: article no.: 190-040020
- Profibus: article no.: 190-040036
- CAN-Bus: article no.: 190-040037
- The RS485 using a field bus is described in the COMTAC instructions.
- RS232 is in the COMPAX-M/S product handbook

1 Asynchronous data transmission

Data to be sent are transmitted serially, i.e. bit by bit, during which a particular procedure must be maintained so that the sender and the receiver can understand one another.
The sender and the receiver both work at the same clock frequency.
Small data packets are transmitted, consisting of a start bit, the data bits, a parity bit if required, and at least one stop bit.



If no transmission takes place, then the data transfer line is high. On sending the **start bit**, the status is changed to low; the receiver knows that data will now be transmitted and the receiver is synchronised.
After the start bit, the actual character to be transmitted then follows consisting of between 5 to 8 **data bits**, as agreed. The lowest value bit of the character to be sent is always transmitted first and the highest value bit transmitted last.
After the data bits, there is the option of sending a **parity bit** for reasons of data security.
Finally, the **stop bits** are sent, which signal the end of the data transmission of a character. The asynchronous transmission protocol permits 1, 1.5 or 2 stop bits.
This closes the transmission of a character. The line now remains on high until the start bit of the next character is transmitted.

There are interfaces which work using a negative logic. In this case, the statuses 0 and 1 or high and low must be exchanged for one another in the above diagram and description which relate to a positive logic. The basic principle of serial transmission does not change at all.

1.1 Parity

The parity bit is used as a protection against transmission errors. However, only simple bit errors can be recognised. The following parity settings are possible:

Parity	Meaning
No	No parity bit is added
Even	The parity bit is placed so that there is an even number of single bits (data bits and parity bit)
Odd	The parity bit is placed so that there is an odd number of single-bits (data bits and parity bit)
Mark	The parity bit is always set to 1
Space	The parity bit is always set to 0

1.2 Transmission speed

How fast data is sent depends on the set transmission speed, the so-called **baud rate**. The transmission speed is given in baud units (= bits per second), which define the bit width.
At a baud rate of 9600 baud, the bit width corresponds to 1/9600 = 104 μs and thus the transmission of a data packet (1 start bit, 8 data bits, 1 stop bit) takes approx. 1 ms.

1.3 Transmission errors

On receiving a data packet, the following errors can arise:

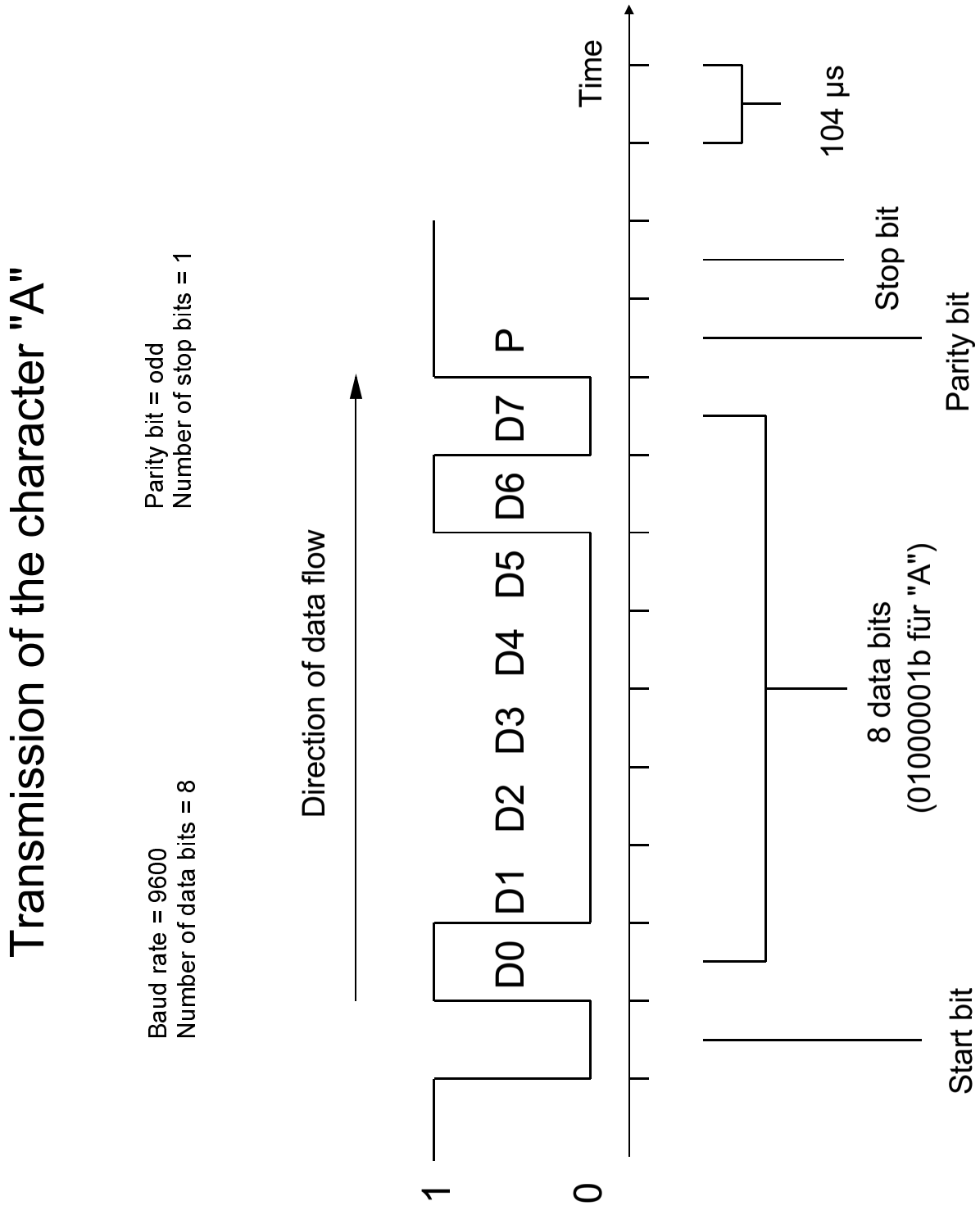
Error type	Meaning
Framing	The data format received does not correspond to the format set (number of start, stop, and data bits and baud rate).
Break	The receiver line remained on low for longer than the duration of a data packet (idle state is high).
Overrun	The receiver has more data supplied than it can process
Parity	The intended parity does not agree with that received.

1.4 Parameters

For correct serial data transmission, the following parameters must be set to the same values in the sender and in the receiver:

- **Baud rate**
- **Number of data bits**
- **Parity**
- **Number of stop bits**

1.5 Transmission of the character "A"



2 The ASCII code

In serial data transmissions, letters and numbers are generally transmitted in an ASCII character code. The assignment between alphanumeric characters and the associated binary representation is standardised through the ASCII code (American Standard Code for Information Interchange).
A distinction is made between the 94 characters which can be represented and the 34 special or control characters which cannot be represented (0 - 32, 127).

2.1 The ASCII code table

Hex	Dec	ASCII	Hex	Dec	ASCII	Hex	Dec	ASCII	Hex	Dec	ASCII
00	0	NUL	20	32	SP	40	64	@	60	96	`
01	1	SOH	21	33	!	41	65	A	61	97	a
02	2	STX	22	34	"	42	66	B	62	98	b
03	3	ETX	23	35	#	43	67	C	63	99	c
04	4	EOT	24	36	$	44	68	D	64	100	d
05	5	ENQ	25	37	%	45	69	E	65	101	e
06	6	ACK	26	38	&	46	70	F	66	102	f
07	7	BEL	27	39	'	47	71	G	67	103	g
08	8	BS	28	40	(	48	72	H	68	104	h
09	9	HT	29	41	)	49	73	I	69	105	i
0A	10	LF	2A	42	*	4A	74	J	6A	106	j
0B	11	VT	2B	43	+	4B	75	K	6B	107	k
0C	12	FF	2C	44	,	4C	76	L	6C	108	l
0D	13	CR	2D	45	-	4D	77	M	6D	109	m
0E	14	SO	2E	46	.	4E	78	N	6E	110	n
0F	15	SI	2F	47	/	4F	79	O	6F	111	o
10	16	DLE	30	48	0	50	80	P	70	112	p
11	17	DC1	31	49	1	51	81	Q	71	113	q
12	18	DC2	32	50	2	52	82	R	72	114	r
13	19	DC3	33	51	3	53	83	S	73	115	s
14	20	DC4	34	52	4	54	84	T	74	116	t
15	21	NAK	35	53	5	55	85	U	75	117	u
16	22	SYN	36	54	6	56	86	V	76	118	v
17	23	ETB	37	55	7	57	87	W	77	119	w
18	24	CAN	38	56	8	58	88	X	78	120	x
19	25	EM	39	57	9	59	89	Y	79	121	y
1A	26	SUB	3A	58	:	5A	90	Z	7A	122	z
1B	27	ESC	3B	59	;	5B	91	[	7B	123	{
1C	28	FS	3C	60	<	5C	92	\	7C	124	|
1D	29	GS	3D	61	=	5D	93	]	7D	125	}
1E	30	RS	3E	62	>	5E	94	^	7E	126	~
1F	31	US	3F	63	?	5F	95	_	7F	127	DEL

2.2 Meaning of special / control characters in the ASCII code

Hex	Dec	ASCII	Meaning	Meaning
00	0	NUL	Zero	Filler
01	1	SOH	Start of Heading	Start of heading
02	2	STX	Start of Text	Start of text
03	3	ETX	End of Text	End of text
04	4	EOT	End of Transmission	End of transmission
05	5	ENQ	Enquiry	Interrogation
06	6	ACK	Acknowledge	Positive response
07	7	BEL	Bell	Bell
08	**8**	**BS**	**Backspace**	**Backspace**
09	9	HT	Horizontal Tabulation	Horizontal tabulation
0A	**10**	**LF**	**Line Feed**	**Line feed**
0B	11	VT	Vertical Tabulation	Vertical tabulation
0C	12	FF	Form Feed	Form feed
0D	**13**	**CR**	**Carriage Return**	**Carriage return**
0E	14	SO	Shift Out	Shift out
0F	15	SI	Shift In	Shift in
10	16	DLE	Data Link Escape	Data transmission conversion
11	17	DC1 (X-ON)	Device Control 1	Device control 1 (Software handshake: Sender on)
12	18	DC2	Device Control 2	Device control 2
13	19	DC3 (X-OFF)	Device Control 3	Device control 3 (Software handshake: Sender off)
14	20	DC4	Device Control 4	Device control 4
15	21	NAK	Negative Acknowledge	Negative response
16	22	SYN	Synchronous Idle	Synchronisation
17	23	ETB	End of Transmission Block	End of transmission block
18	24	CAN	Cancel	Invalid
19	25	EM	End of Medium	End of medium
1A	26	SUB	Substitute	Substitution
1B	27	ESC	Escape	Conversion
1C	28	FS	File Separator	File separation
1D	29	GS	Group Separator	Group separation
1E	30	RS	Record Separator	Record separation
1F	31	US	Unit Separator	Unit separation
20	32	SP	Space	Space
7F	127	DEL	Delete	Delete

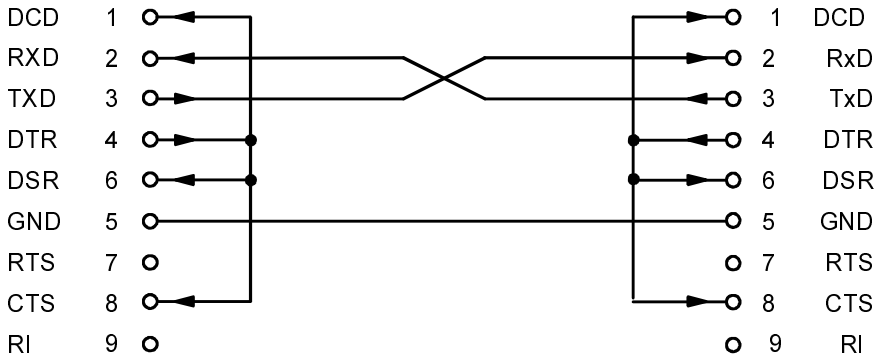
3 The RS232 C interface

In the RS232 standard, a high level is defined as a voltage of between +3V and +15V, and a low level as a voltage between -3V and -15V. In this, data is transmitted in negative logic, and control signals in positive logic. In addition to both data transfer lines, there are six control lines defined which can be used to control the exchange of data.
The max. length of line is 15m, and the max. rate of data is 20kbits/s.
The RS232 interface is a point to point connection.

Data and control lines of a PC standard RS232 interface:

Pin configuration		Abbreviation	Signal name	Meaning	I/O
9 pin	25 pin				
1	8	DCD	Data Carrier Detected	Received signal level	Input
2	**3**	**RXD**	**Receive Data**	**Receive data**	**Input**
3	**2**	**TXD**	**Transmit Data**	**Transmit data**	**Output**
4	20	DTR	Data Terminal Ready	End device ready	Output
5	7	GND	Ground	Ground	-
6	6	DSR	Data Set Ready	Operational readiness	Input
7	4	RTS	Request to Send	Switch on transmission section	Output
8	5	CTS	Clear to Send	Readiness to transmit	Input
9	22	RI	Ring Indicator	Call arriving	Input

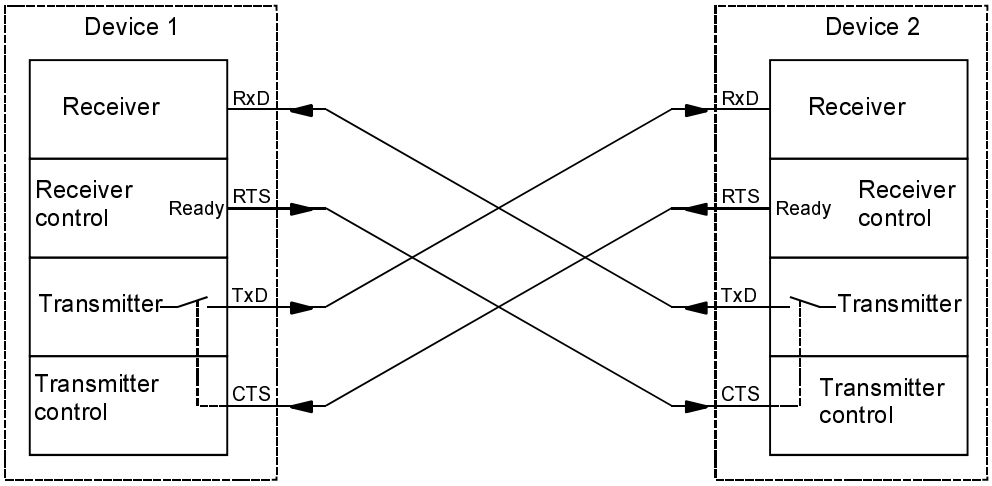
In principle, for mutual exchange of data over the RS232 interface, only the TXD and RXD data transfer lines are required along with the GND ground line.
However, the majority of devices use some of the control lines. For this reason it is necessary to assign the corresponding outputs and never to leave them open.



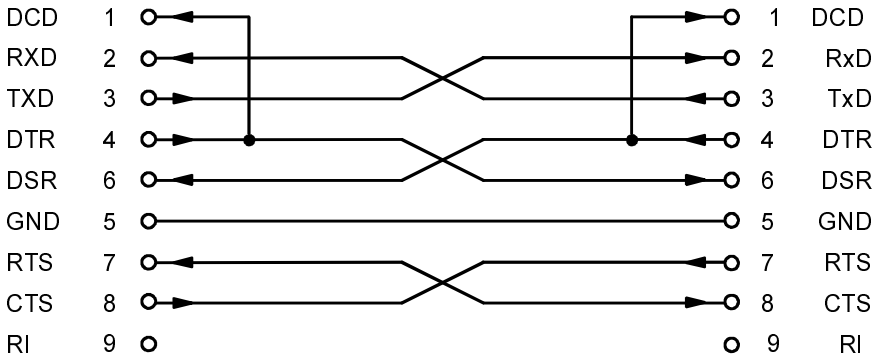
There exist no standards for dealing with control lines and which are therefore heavily dependent on the operating software of the devices which are to communicate with one another.

3.1 Hardware handshake

Very often, both control lines RTS and CTS are used for this.
The receiving device sets the RTS line to low if it requires a pause in transmission. The transmitting device recognises this on its CTS line and stops the transmission for as long as this is required by the receiving device.



For the hardware handshake, both devices must be connected with a null modem.



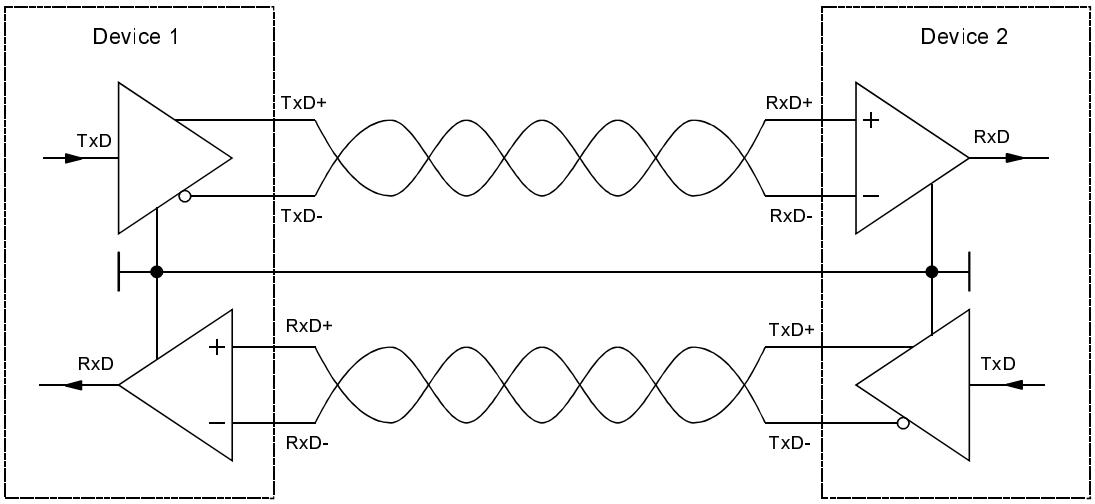
The DTR and DSR signals are required by some devices but not by COMPAX.

3.2 Software handshake

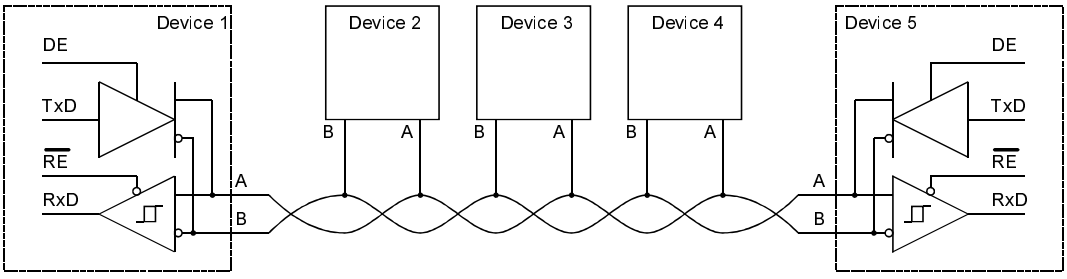
For this, both control characters DC1 and DC3 of the ASCII code are used.
If the device receives the character DC3 (X_OFF), then it will automatically switch off its transmitter; if it receives the character DC1 (X_ON), then it will switch its transmitter on again.

4 The RS422/RS485 interface

This push-pull connection which is balanced to earth offers a high rate of data transmission, is relatively insensitive to faults and only requires a supply voltage of + 5V.
The max. line length is 1200m; the max. data rate is 10Mbits/s.
A bundle-assembled two-wire aerial cable is recommended as a transmission medium.



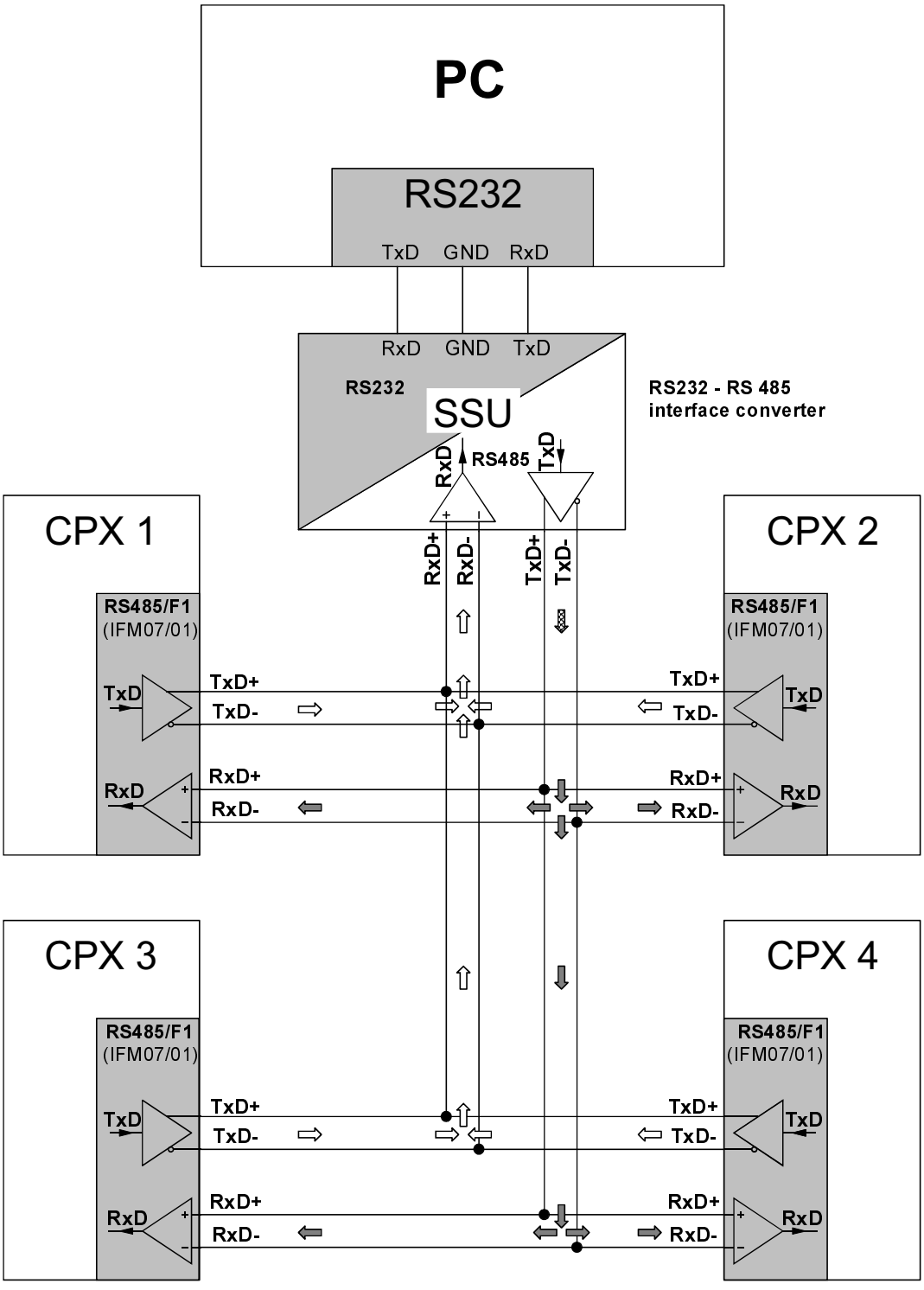
The **RS422** interface is a point to point connection.
(Four-wire technology)



The **RS485** interface is a multi-point connection.
(two-wire technology)

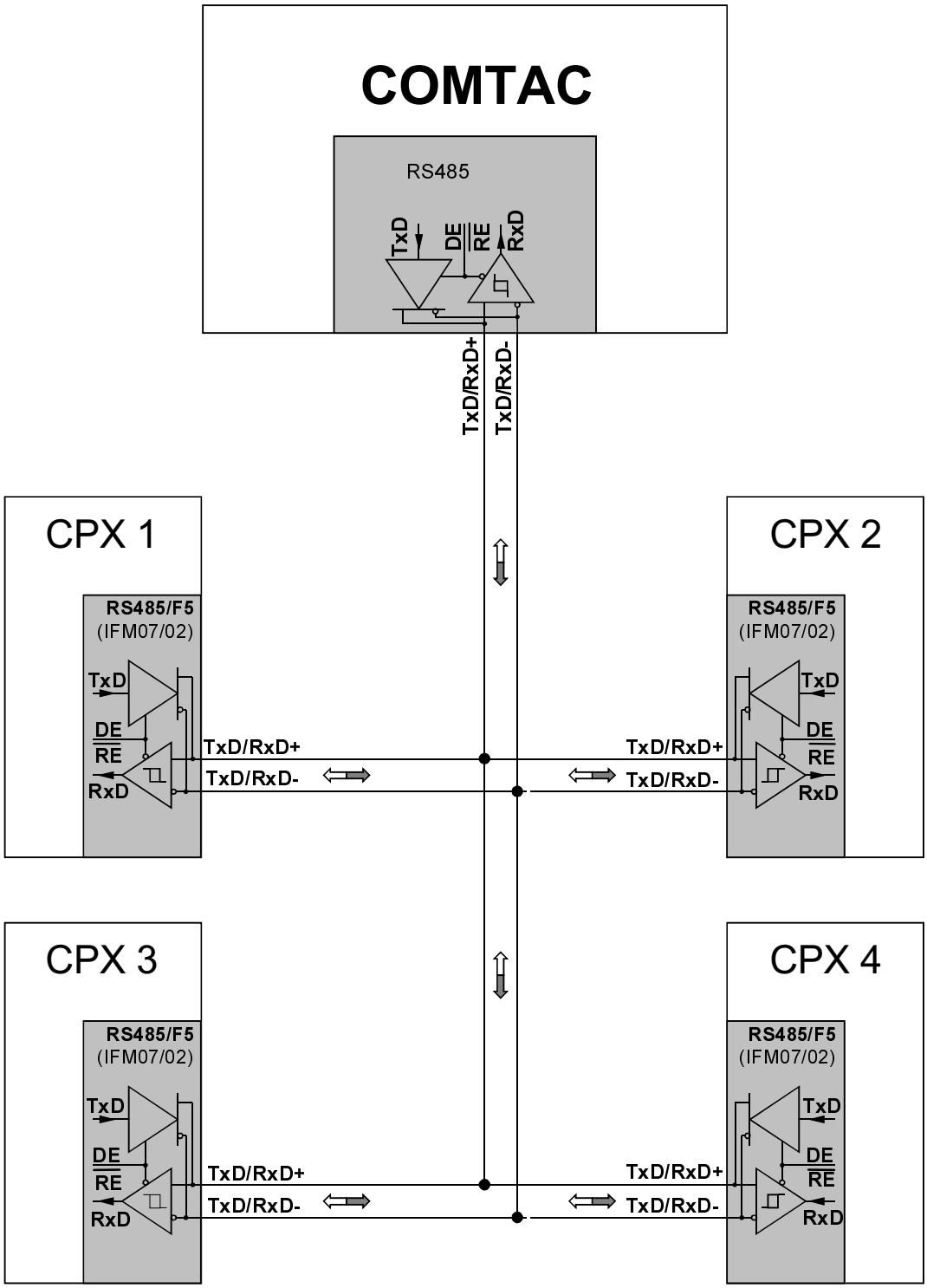
4.1 RS485 four wire connection PC/SSU⇔COMPAX (with F1 option)

for instructions in ASCII or in binary format



4.2 RS485 two-wire connection COMTAC⇔COMPAX

for instructions in ASCII and binary format or for the field bus protocol



5 General message construction for transmission protocols

Data to be transmitted is packed into a message frame (telegram).

Start	Address	Control info	Data	Data security	Stop

Principal telegram construction

Using the frame information, each station can distinguish:

- **when a telegram starts and stops (start/stop identifier)**
- **whether the telegram is intended for itself (address)**
- **which meaning the received data has (control information)**
- **whether the telegram was correctly transmitted (data security)**

6 Access procedures

With bus systems in which all stations are connected in parallel on the transmission medium, only one bus station can ever send data on the transmission medium at any one time.
There are three distinct access procedures:

- **Polling (master-slave procedure)**
- **Token Passing**
- **CSMA/CD** (**C**arrier **S**ense **M**ultible **A**ccess with **C**ollision **D**etection)

6.1 Polling (master-slave procedure)

This procedure is used in single master systems (1 master and several slaves).

- Bus access is administered by the master.
- Only the master may send messages of its own accord.
- The slaves must wait for a request from the master.
- If the master sends a request telegram to a slave, then generally the slave will confirm the receipt of this message using a response telegram.
- One exception to this is messages which all slaves receive simultaneously (broadcast); these are not acknowledged by the slaves.

6.2 Token Passing

- Token passing is used in multi-master systems (several masters and slaves).
- The right to access (token) is passed from master to master.
- The master which has the token can communicate with all the other stations.

6.3 CSMA/CA

(**C**arrier **S**ense **M**ultible **A**ccess with **C**ollision Avoidance)

- This procedure is used in bus systems in which all stations have equal rights (e.g. CAN-Bus)
- Each station can access the bus at any time.

- For this reason, access contention must be clearly recognised and resolved.

  - Before sending its telegram, each station checks whether the transmission medium is free (**Carrier Sense**).

  - If there is no activity on the transmission medium, then the station will start to send its message. In this, another station can possibly simultaneously start to transmit a message (**Multiple Access**).

  - Protocols based on prioritised messages are applied bit by bit to the bus via a message identifier. As soon as a message with a higher priority is recognised, the stations with lower priorities end their access. Only the station with the highest priority retains access to the bus.
    The message identifier of the station with access is not destroyed in this (**Collision Avoidance**), as the dominant bits (low level) in the pull up resistance carry through the highest priority message.

  - After a certain delay, the station re-attempts to transmit the message.

7 Overview: Technical data on COMPAX interfaces

Characteristic	RS232	RS485 standard	RS485 field bus	Interbus-S	Profibus	CAN-Bus
Option	Standard	F1 / F5	F1 / F5	F2	F3	F4
Baud rate [Baud]	4800 9600	150 - 115.2k	28.8k - 345.6k	500k	9600 - 1.5M	20K - 1M
Max. line length	15m	1.2km	1.2km	13km	4.8km	1.3km
max. number of stations	2	31	31	64	126	31
Access procedures		Polling	Polling	Distributed ring shift register	Token passing / polling	CSMA/CA
Data format	ASCII/ binary	ASCII/ binary		Common frame / fixed telegram length	special frame formats / variable telegram length (max. 246 bytes)	special fixed frame formats / variable telegram length (max. 8 byte)
Cyclical data	no	no	yes	yes	yes	no
Application:	simple point to point connection	open system for almost any station	Optimised COMTAC - COMPAX interface	Sensor/ actuator bus Fast time-constant access to all stations	Large amounts of data Very flexible	Fast access to individual stations for closed systems (series applications)

8 COMPAX RS485 Standard protocol

This protocol is for single-master operations (polling procedure).
The master transmits a request telegram to the COMPAX; COMPAX acknowledges this request using a corresponding response telegram.
Both ASCII and binary data transmissions are possible.

8.1 ASCII data transmission

Address	Data	Option character	End-of-text character	BCC

request telegram

Address	Data	End-of-text character	BCC

Response telegram

address
Definines which COMPAX should be addressed.
1 to 32 (ASCII format; one or two digits)permitted.
Using the address 32 (broadcast address), all connected COMPAXes can be addressed simltaneously. In this case, they will not send a response.

data (request)
COMPAX command in plain text (ASCII format)
or the *execute character* ("**!**" = 0x21).
On transmitting the *execute character*, the last command received is repeated and/or carried out (buffer option).

Data (response)
this differs according to request and device status:
- ⇨ no data (request acknowledgement for a command which requires **no** data)
- ⇨ Data requested in ASCII format
- ⇨ Error message in ASCII format (in cases of errors)

Option character
Can be added by choice to activate special functions.

Option character	Function
Buffer ("," = 0x2C)	Command filed in buffer but not carried out
No-Ack ("**/**" = 0x2F)	Command not acknowledged by CPX
Echo ("**?**" = 0x3F)	Command immediately sent back to sender

End bof text character
The control character **CR** (0x0D) is used to identify the end of the telegram.

BCC
The block-check character is optional and is used for data security.
BCC is constructed from the byte by byte XOR operation of the telegrams, exclusively from the end-of-text character.

8.2 Binary data transmission

For time critical applications, it is possible to transmit a number of commands in COMPAX-internal binary format (adjustable using parameters).
(Time savings through not using the format conversion ASCII⇨binary).
Request and response telegrams have different constructions.

Address	Control info	Data	BCC

Request telegram

Address	End of text character	BCC

Response telegram

Address
Defines which COMPAX should be addressed.
1 to 32 allowed (ASCII format; one or two figure).
Using address 32 (broadcast address), all connected COMPAXes can be addressed simultaneously. In this case, they will not send a response.

Control info

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Identifier for binary data	Length of data field (number of bytes) + 1						

Data
COMPAX commands in binary format

BCC
The block-check character is optional and is used for data security.
The BCC is created in bytes from the XOR operation of the telegram, exclusively from the end of text character.

8.2.1 Command list

The following commands are possible in binary transmission:

Command	Control info	Data (hexadecimal)
POSA x (x ⇨ Position in DSP format	0x88 1 0001000b	41 xx xx xx xx xx xx LSB MSB
POSR x (x ⇨ Position in DSP format	0x88 1 0001000b	52 xx xx xx xx xx xx LSB MSB
SPEED x (x ⇨ speed in DSP format	0x88 1 0001000b	53 xx xx xx xx xx xx LSB MSB
ACCEL y (y ⇨ acceleration time in binary format	0x84 1 0000100b	4C yy yy MSB LSB
ACCEL- y (y ⇨ braking time in binary format	0x84 1 0000100b	44 yy yy MSB LSB
OUTPUT y = 1 (y ⇨ Output No. in binary format	0x85 1 0000101b	4F 00 yy 31 MSB LSB
OUTPUT y = 0 (y ⇨ Output No. in binary format	0x85 1 0000101b	4F 00 yy 30 MSB LSB
POSR x OUTPUT y=1 (x ⇨ Position in DSP format (y ⇨ Output no. in binary format	0x8C 1 0001100b	52 xx xx xx xx xx xx 4F 00 yy 31 LSB MSB MSB LSB
POSR x OUTPUT y=0 (x ⇨ Position in DSP format (y ⇨ Output no. in binary format	0x8C 1 0001100b	52 xx xx xx xx xx xx 4F 00 yy 30 LSB MSB MSB LSB
POSR x SPEED y (x ⇨ Position in DSP format (y ⇨ speed in DSP format	0x8C 1 0001100b	52 xx xx xx xx xx xx 53 yy yy yy yy yy yy LSB MSB LSB MSB

8.2.2 DSP numeric format

In this format, a number is represented using 24 bits (3 bytes) in front of the decimal point (whole number section) and 24 bits (3 bytes) after the decimal point (fraction section).

2^{23} ..... 2^{2} 2^{1} 2^{0}	2^{-1} 2^{-2} 2^{-3} ..... 2^{-24}
3 bytes before the point	3 bytes after the point

☞ *Negative numbers are represented as a two's complement.*

8.2.3 Format conversion

*[2] **Format conversion**
This format can be created as follows from any number with figures after the point:
Example: number = 450.5

1. Multiply number by 2^{24} .
   $450.5*2^{24} = 7558135808$.
2. Convert 7558135808 into a hexadecimal number (possibly into an integer first)
   =>.0x00 01 C2 80 00 00 ≡ before the point, after the point ≡ MSB,.... LSB, MSB,.... LSB.
3. These bytes must now be entered in the given sequence in the commands. The sequence of the bytes is round the other way. The sequence of the bits must not be reversed.

This conversion also applies to negative numbers.

☞ *In the case of binary commands, values are given in DSP format in the reverse byte sequence; first 3 bytes before the point (LSB ... MSB) then 3 bytes after the point (LSB... MSB).*

8.3 Example

Example 1:
The command "POSA7500"is sent to the COMPAX with the address 12
in ASCII format (without BCC).

	Address		Data								End-of-text character
ASCII	1	2	P	O	S	A	7	5	0	0	CR
hex	0x31	0x32	0x50	0x4F	0x53	0x41	0x37	0x35	0x30	0x30	0x0D
dec	49	50	80	79	83	65	55	53	48	48	13

Request telegram

	Address		End-of-text character
ASCII	1	2	CR
hex	0x31	0x32	0x0D
dec	49	50	13

Response telegram

Example 2:
The command "S1" is sent in ASCII format (without BCC) to the COMPAX with address 7,
which is currently at the actual position 1230.50 mm .

	Address	Data		End-of-text character
ASCII	7	S	1	CR
hex	0x37	0x53	0x31	0x0D
dec	55	83	49	13

Request telegram

	Address	Data										
ASCII	7	S	0	0	1	:				+	1	2
hex	0x37	0x53	0x30	0x30	0x31	0x3A	0x20	0x20	0x20	0x2B	0x31	0x32
dec	55	83	48	48	49	58	32	32	32	43	49	50

Data								End-of-text character
3	0	.	5	0		m	m	CR
0x33	0x30	0x2E	0x35	0x30	0x20	0x6d	0x6d	0x0D
51	48	46	53	48	32	109	109	13

Response telegram

Example 3:
The command "POSA7500" is sent to the COMPAX with address 25
in binary format without BCC.

	Address		Control info	Data							End-of-text character
hex	0x32	0x35	0x88	0x41	0x00	0x00	0x00	0x4C	0x1D	0x00	0x0D
dec	50	53	136	65	0	0	0	76	29	0	13

Request telegram

Response telegram as in example 1.

8.4 Interface parameters

The following CPX parameters are relevant to the RS485 standard protocol:

No.	Meaning	Min	Standard	Max	valid
P194	**Device address**	**0**	**99**	**31**	**Power on**
P195	**Baud rate**	**150**	**9600**	**115200**	**Power on**
P196	**Operating mode**	**0**	**0**	**255**	**Power on**

8.5 Setting operating mode using P196

The RS485 standard protocol is fixed at 8 data bits and one stop bit.
The parity bit can be switched on if required.
Further protocol functions can be switched on/off using P196

Bit	Function	Activation	
0	Software handshake	0 = Off	1= On
1	Transmission with parity	0 = Off	1= On
2	ODD/EVEN	0 = Odd	1= Even
3	Transmission using BCC	0 = without BCC	1 = with Bcc
4	Timeout monitoring	0 = Off	1 = On
5	End of text character of COMPAX response	0 = CR LF >	1 = CR
6	COMPAX response without/with address	0 = without	1 = with
7			

8.6 Timeout monitoring (optional)

If COMPAX has recognised its address, then time monitoring is activated.
This checks whether within the defined period of time (transmission time of 5 characters) a further character is received. After each received character, the time monitoring is re-started until such time as the end-of-text character or the BCC is received. If no character is received within the monitoring period, then the timeout error (E73) is generated.

8.7 Special error messages

NR	Cause
E70	Parity error; received parity bit <> intended parity bit
E71	Overflow error; received telegram > 40 characters
E72	BCC error; received BCC <> intended BCC
E73	Timeout error; pause in transmission between two telegram characters was too long

These errors must not be acknowledged. The drive is *not* dead.
The error message is deleted on the next error-free transmission.

9 COMPAX RS485 field bus protocol using COMTAC as master

This protocol works on the master/slave principle (polling procedure).
The master (COMTAC) sends a request telegram to the COMPAX;
COMPAX acknowledges this request with a corresponding response telegram.

Address	Control info	Data	BCC

Request telegram

Status info	Data	BCC

Response telegram

Address
Defines which COMPAX should be addressed. 1 to 99 (binary format) is permitted. Using the address 255 (broadcast address), all connected COMPAXes can be addressed simltaneously. In this case, they will not send a response.

Control info
The control information consists of one byte and describes the following message types:

Message type	Meaning
0	Initialisation message
1	Request device information (plain text)
2	Reset interface
16	Cyclical data exchange
32	Acyclic data exchange - write data word(e)
33	Acyclic data exchange - read data word(e)
51	Acyclic data exchange - write data
52	Acyclic data exchange - read data

Data (request)
Differs according to message type

Data (response)
Differs according to message type of request telegram

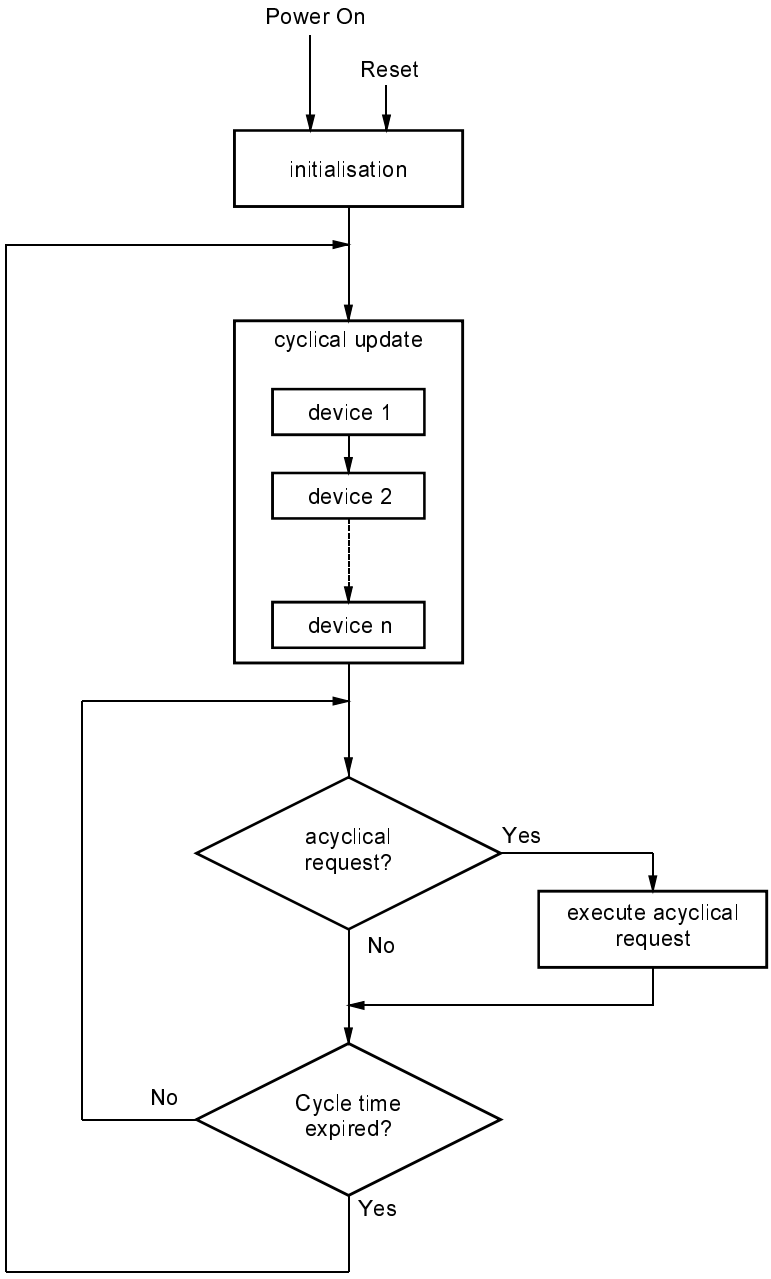
Status info
Status information consists of one byte and has the following meaning:

Bit	Function	Activation	
0	Device identification	0 = without cycl. data	1= with cycl. data
1 - 4	no		
5	Alarm	0 = no alarm	1= alarm occurred
6	Error	0 = no error	1 = error occurred
7	data	0 = no data	1 = data ready

BCC
XOR operation of telegrams and 0xFF in bytes.

9.1 Protocol sequence

COMTAC first sends the initialisation message to all device addresses allowed on the bus. In doing this, COMTAC discovers which type of devices are connected to the bus, which addresses these have and how many I/O bytes must be exchanged cyclically with these devices. These data are entered in a poll-list.
After the initialisation phase, COMTAC starts the cyclical update.
Using the poll-list, COMTAC sends the new values for the output bytes to the devices using cyclical data and receives as a response the current status of the input bytes.
If the poll list is used up, then if necessary an acyclic data exchange with a device is carried out. If after this the cycle time has expired, then a new cyclical update is carried out. If the cycle time has not yet expired, then if necessary a further acyclic data exchange is carried out with a device, etc.



9.2 Cycle time

The minimum achievable update time for I/O information on the field bus modules depends on the transmission speed settings:

Baud rate [bits/s]	Cycle time [msec]	Character time [µsec]
345 600	7	32
172 800	10	64
57 600	15	192
28 800	30	384

It can never be lower than this time, in order to allow the slave devices time for a minimum of pre-processing. The cycle time increases if many slave devices are connected to COMTAC. The actual cycle time(update time) is calculated using the formula:

$$T_{cyc} = ((5 * nST + nIO) * T_{char}) + (nST * T_{admin})$$

T_{cyc} = actual cycle time
nST = number of stations
nIO = number of input/output bytes
T_{char} = character time (e.g. 32 µs at 345 600 baud)
T_{admin} = administration time(= 100 µs)

A COMPAX has 10 bytes of input data and 6 bytes of output data.
COMTAC is designed so that a max. of 31 COMPAX (= 496 I/O data bytes) can be administered over the field bus.

Number COMPAX	Actual cycle time [msec]			
	345 600	172 800	57 600	28 800
1	0.772	1.444	4.132	8.164
2	1.544	2.888	8.264	16.328
3	2.316	4.332	12.396	24.492
4	3.088	5.776	16.528	32.656
5	3.860	7.220	20.660	40.820
6	4.632	8.664	24.792	48.984
31	23.932	44.764	128.092	253.084

Cycle time for n COMPAX devices = (21 * Tchar) + 100 µs) * n)

9.3 Cyclical data of COMPAX

Output data from the master	Byte	COMTAC command
Digital outputs Bit 15 ⇨ A16 ..... Bit 0 ⇨ A1	2	CPXOUT(x) = y
Speed reduction (override) 0 to 255 (255 ⇨ 100% ... 128 ⇨ 50% ... 0 ⇨ 0% (of speed set)	2	CPXOVR(x) = y
Control word (emulation of input functions)	2	CPXCTR(x) = y

Input data of masters	Byte	COMTAC command
Status of digital inputs Bit 15 ⇨ E16 ..... Bit 0 ⇨ E1	2	y = CPXINP(x)
Status of digital outputs Bit 15 ⇨ A16 ..... Bit 0 ⇨ A1	2	y = CPXOUT(x)
Status word (display of output status information)	2	y = CPXSTS(x)
Actual position	4	y = CPXPOS(x)

9.3.1 Control word

Bit	Function without shift	Function with shift	Release
15 ... 8	none	none	P222/Bit 7 ... 0
7...6	none	none	P221/Bit 7 ... 6
5	STOP	BREAK	P221/Bit 5
4	START	none	P221/Bit 4
3	QUIT	Teach real null	P221/Bit 3
2	Hand-	approach real zero (RZ)	P221/Bit 2
1	Hand+	approach machine zero (MZ)	P221/Bit 1
0	Shift		

9.3.2 Status word

Bit	Configuration
15 ... 8	Status of outputs A16 ... A8
7	Contouring error
6	Motor blocked
5	No movement after stop
4	Programmed target position reached
3	Ready for start
2	Machine zero (MZ) approached
1	no warning
0	no fault

9.4 Interface parameters

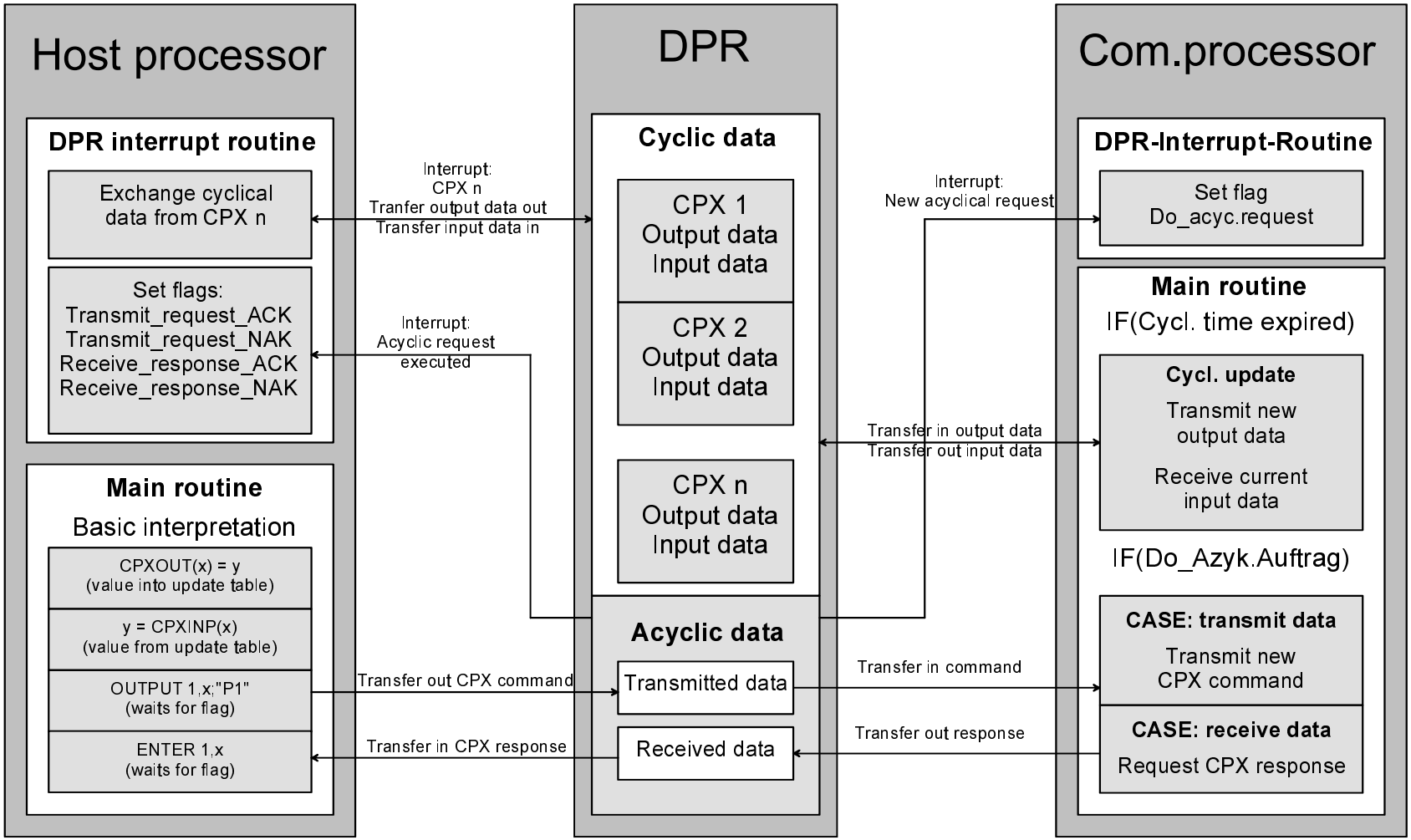
The following CPX parameters are relevant for the RS485 field bus protocol:

No.	Meaning	Min	Standard	Max	Valid
P194	Device address	1	99	255	Power on
P195	Baud rate	28800 / 57600 / 172800 / 345600			Power on
P196	Operating mode	164			Power on

9.5 Operating mode setting with P196

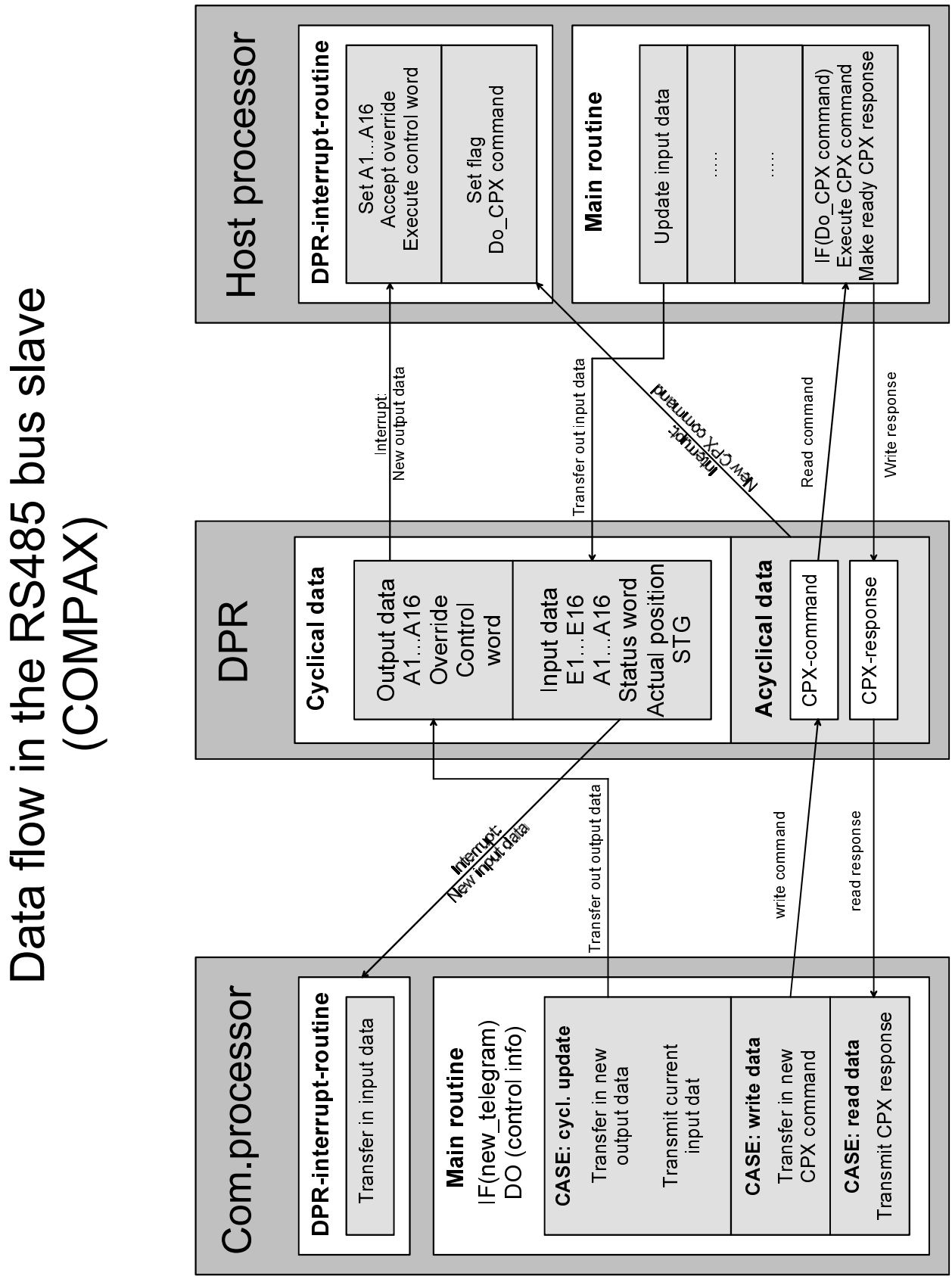
The RS485 field bus protocol works with 8 data bits, 1 address bit and one stop bit.
No additional settings are possible.
The actual parity bit is used as an address bit.
The master (COMTAC) sends the address with the address bit = 1 for synchronisation purposes (start of a new request telegram).
All other characters are always sent with the address bit = 0 .

Data flow in the RS485 field bus master (COMTAC)



DPR: Dual Port RAM

9.7 Data flow in the field bus slave



10 Interbus-S

For data transmission, Interbus-S uses the master-slave procedure.
All stations - master and slaves - are connected in a ring structure. The ring is constructed using several lines within a VSI cable.

All devices on the bus are seen as one logical station.
All information on process data is transmitted in a common frame simultaneously to all stations in each cycle. Using the interval positions of individual information in the common frame, each station can pick up the data meant for it.

The VSI protocol differentiates between ID cycles and data cycles; however, both are the same from the point of view of protocol procedures.

Because of the ring structure, full-duplex operations are possible; this means that simultaneous transmission and receiving of data is possible.

IBS common frame

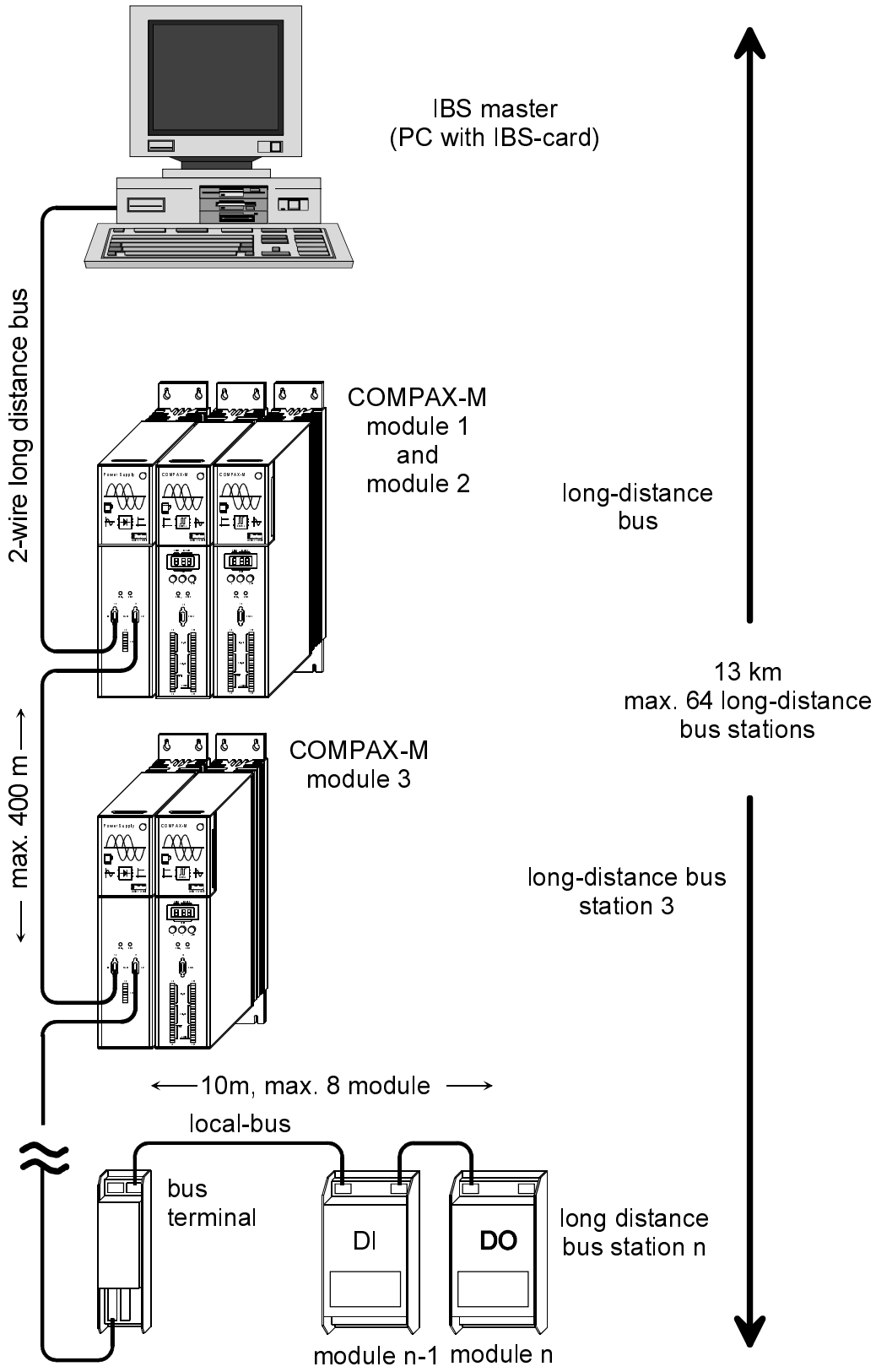
Loop check
Using this, the master can distinguish whether a cycle was processed without problems with regard to data length. If the loop-back word received (16 bits) is not the same as that transmitted, then something has changed in the interconnection of the system or the loop-back word was corrupted.

Data
User information on individual devices (G1 to Gn) of the VSI ring.
The master transmits a control word in an identity cycle to each device and receives the identification code from each device (ID-code).
In a data cycle, the master sends each slave its process output data (POD) and receives its process input data from each slave (PID).

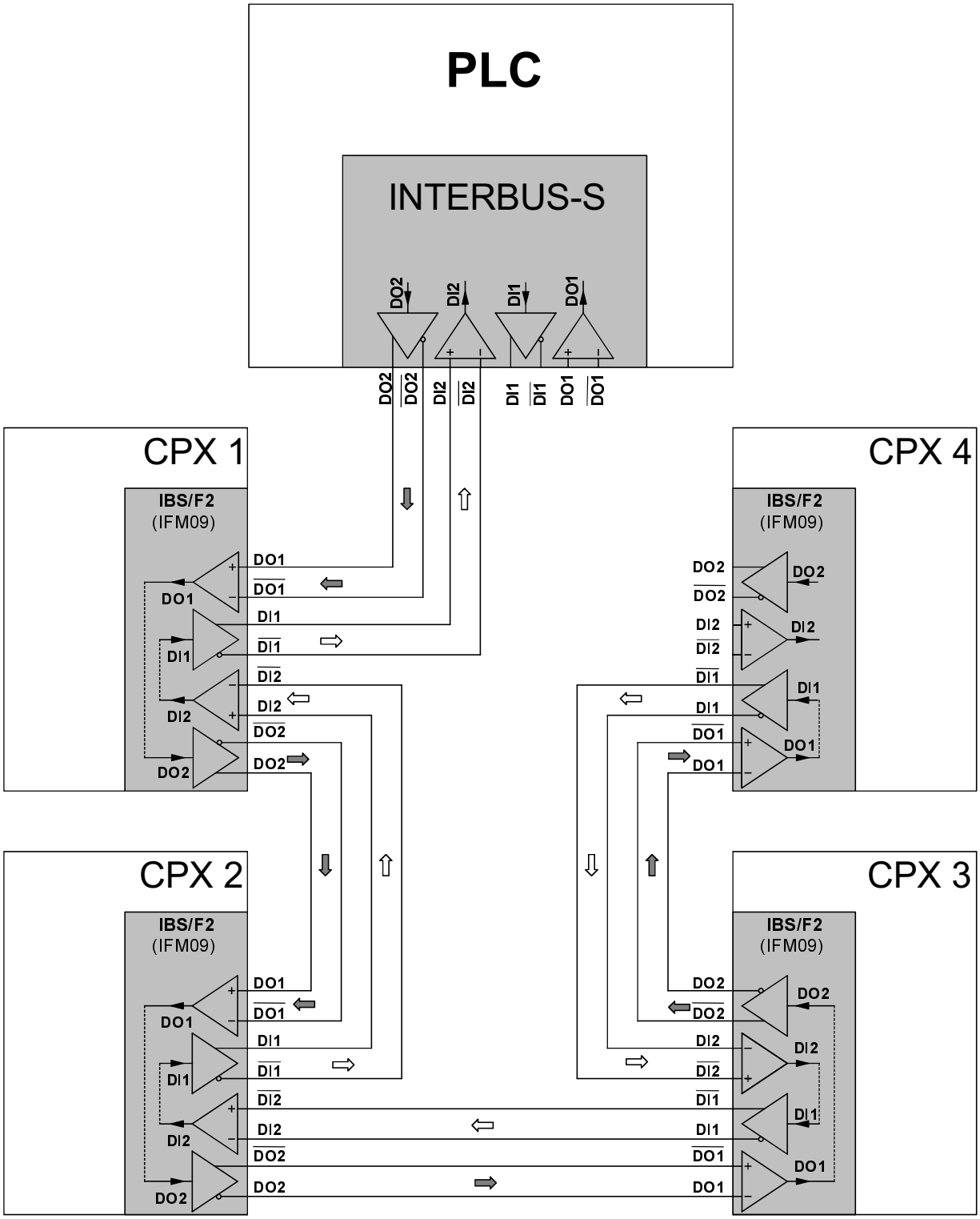
FCS
The **F**rame **C**heck **S**equence (16 bit) is used for data security.
A **C**yclic **R**edundancy **C**heck (CRC) is used.

Control
Using these closing 16 bits, the result of the CRC is passed on and the transfer of data is handled.

10.1 IBS topology

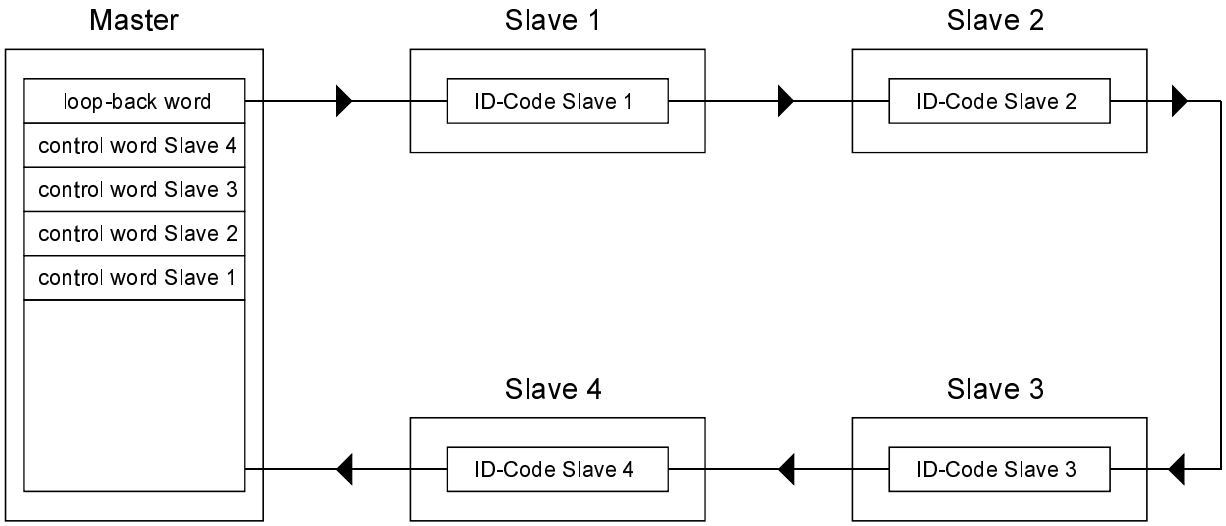


10.2 RS485 two-wire connection SPS⇔COMPAX for the IBS protocol

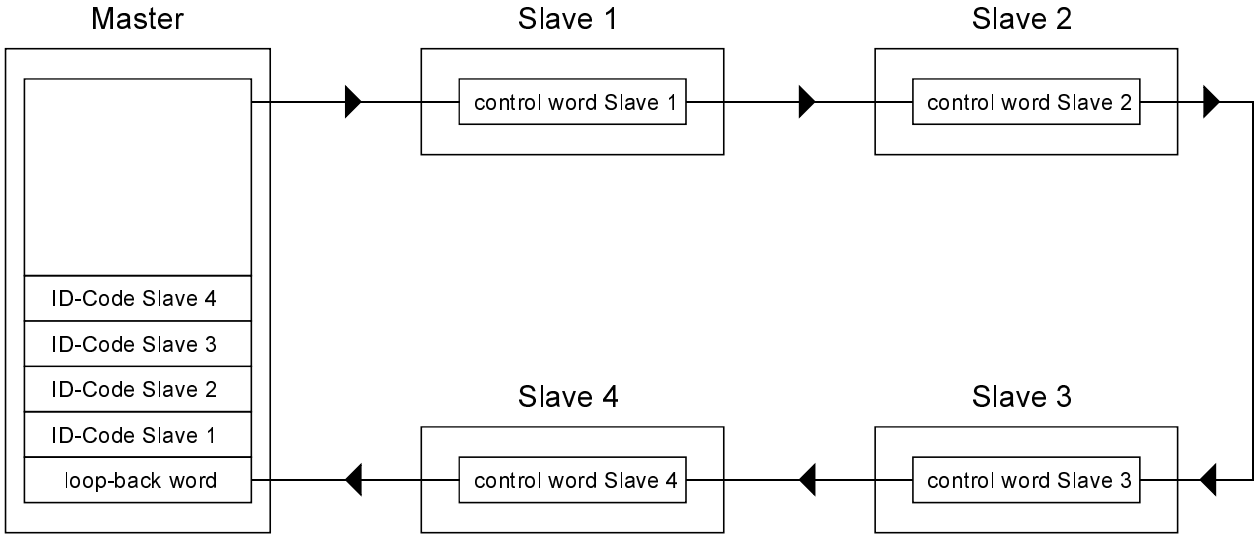


10.3 Identification cycle

The Interbus-S ID cycle is used by the master to recognise the real system configuration, to localise errors and to transmit control information.
At the start of the system, the master sends control information to the stations and simultaneously receives back their ID-codes.



Distribution of information before the ID-cycle



Distribution of information after the ID-cycle

Through the evaluation of the ID-code list, the master can determine how many stations are connected, what type of stations they are and the physical locations of the stations in the ring.

10.3.1 Identification code

The ID-code is 16 bits wide and is defined as follows:

Management			Data width					Device groups							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Management
Using these bits, each station can transmit a message to the master in an ID-cycle.

ID 15	ID 14	ID 13	Meaning
x	x	1	Reconfiguration request
x	1	x	CRC error
1	x	x	Module error

Data width
Using these bits, each station informs the master how large its data shift register is.

ID 12	ID11	ID10	ID9	ID8	Meaning	ID 12	ID11	ID10	ID9	ID8	Meaning
0	0	0	0	0	no data	0	1	1	0	1	5 bytes
0	0	0	0	1	1 word	0	1	1	1	0	6 words
0	0	0	1	0	2 words	0	1	1	1	1	7 words
0	0	0	1	1	3 words	1	0	0	0	0	reserved
0	0	1	0	0	4 words	1	0	0	0	1	26 words
0	0	1	0	1	5 words	1	0	0	1	0	16 words
0	0	1	1	0	8 words	1	0	0	1	1	24 words
0	0	1	1	1	9 words	1	0	1	0	0	32 words
0	1	0	0	0	1 nibble	1	0	1	0	1	10 words
0	1	0	0	1	1 byte	1	0	1	1	0	12 words
0	1	0	1	0	3 nibbles	1	0	1	1	1	14 words
0	1	0	1	1	3 bytes	1	1	x	x	x	reserved
0	1	1	0	0	5 nibbles						

Device group
Using this, the master can sort the slaves according to their functionality.

ID 7	ID 6	ID 5	ID 4	ID 3	ID 2	ID 1	ID 0	Device group
0	0	0	0	x	x	0	0	digital FB-Tn without data
0	0	0	0	x	x	0	1	digital FB-Tn with OUT data
0	0	0	0	x	x	1	0	digital FB-Tn with IN data
0	0	0	0	x	x	1	1	dig. FB-Tn with IN/OUT data
0	0	0	1	x	x	x	x	reserved for bus master
0	0	1	0	x	x	0	0	digital FB-Tn without data
0	0	1	0	x	x	0	1	digital FB-Tn with OUT data
0	0	1	0	x	x	1	0	digital FB-Tn with IN data
0	0	1	0	x	x	1	1	dig. FB-Tn with IN/OUT data
0	0	1	1	x	x	0	0	analog FB-Tn without data
0	0	1	1	x	x	0	1	analog FB-Tn with OUT data
0	0	1	1	x	x	1	0	analog FB-Tn with IN data
0	0	1	1	x	x	1	1	ana. FB-Tn with IN/OUT data

ID 7	ID 6	ID 5	ID 4	ID 3	ID 2	ID 1	ID 0	Device group
0	1	x	x	x	x	0	0	analog LB-Tn without data
0	1	x	x	x	x	0	1	analog LB-Tn with OUT data
0	1	x	x	x	x	1	0	analog LB-Tn with IN data
0	1	x	x	x	x	1	1	ana. LB-Tn with IN/OUT data
1	0	x	x	x	x	0	0	digital LB-Tn without data
1	0	x	x	x	x	0	1	digital LB-Tn with OUT data
1	0	x	x	x	x	1	0	digital LB-Tn with IN data
1	0	x	x	x	x	1	1	dig. LB-Tn with IN/OUT data
1	1	0	x	x	x	0	0	LB-Tn with 2 PCP words
1	1	0	x	x	x	0	1	LB-Tn with 4 PCP words
1	1	0	x	x	x	1	0	LB-Tn with PCP words reserved
1	1	0	x	x	x	1	1	LB-Tn with 1 PCP word
1	1	1	x	x	x	0	0	FB-Tn with 2 PCP words
1	1	1	x	x	x	0	1	FB-Tn with 4 PCP words
1	1	1	x	x	x	1	0	FB-Tn with PCP words reserved
1	1	1	x	x	x	1	1	FB-Tn with 1 PCP word

10.3.2 COMPAX ID-code

COMPAX has the ID-code 227 (0xE3) which stands for a long-distance bus station with 1 PCP word.

10.3.3 Control data

The master can influence the system at any time using the 16 bit wide control word.
Using bits 8 and 9, the local and/or long-distance bus modules linked to the bus coupler can be reset.
Using bits 10 and 11, the continuing local and/or long distance bus interfaces can be switched on or off.
If the "masking" bit (bit 15) is set, then all other bits of the control word are not evaluated by the module. This is used, for example, in reading the system configuration.

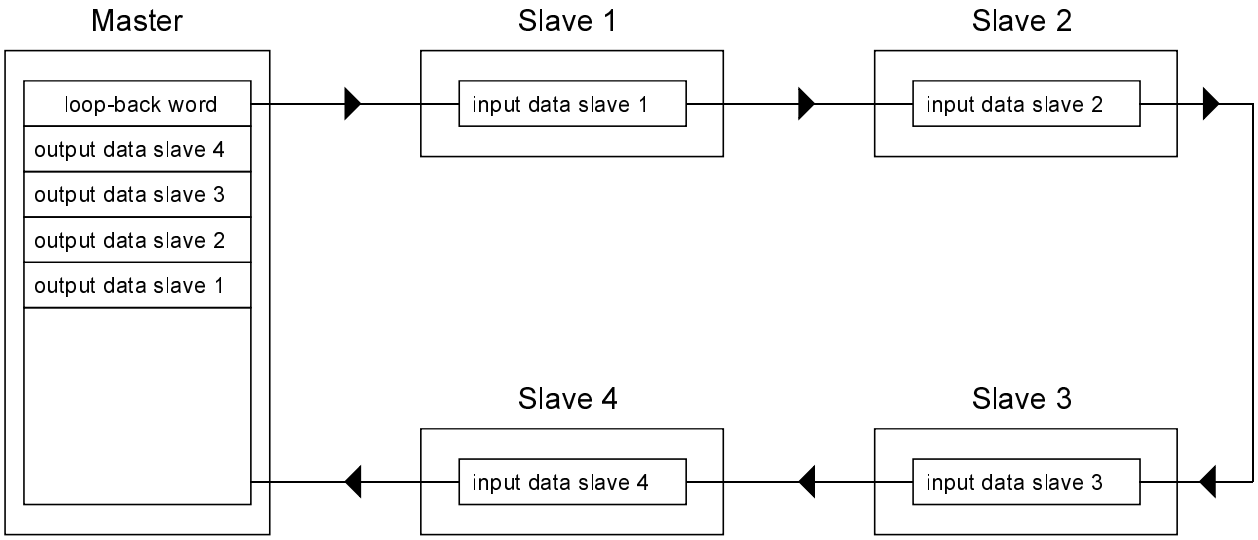
Bit	Meaning		
0 ... 7	reserved		
8	Local bus reset (bus coupler only)	0 ⇨ no reset;	1 ⇨ reset
9	Long-distance bus reset	0 ⇨ no reset;	1 ⇨ reset
10	Local bus interface switch (bus coupler only)	0 ⇨ on;	1 ⇨ off
11	continuing long-distance bus interfaces switch	0 ⇨ on;	1 ⇨ off
12 ... 14	reserved		
15	Masking 0 ⇨ no masking;	1 ⇨ masking	

10.3.4 Loop-back word

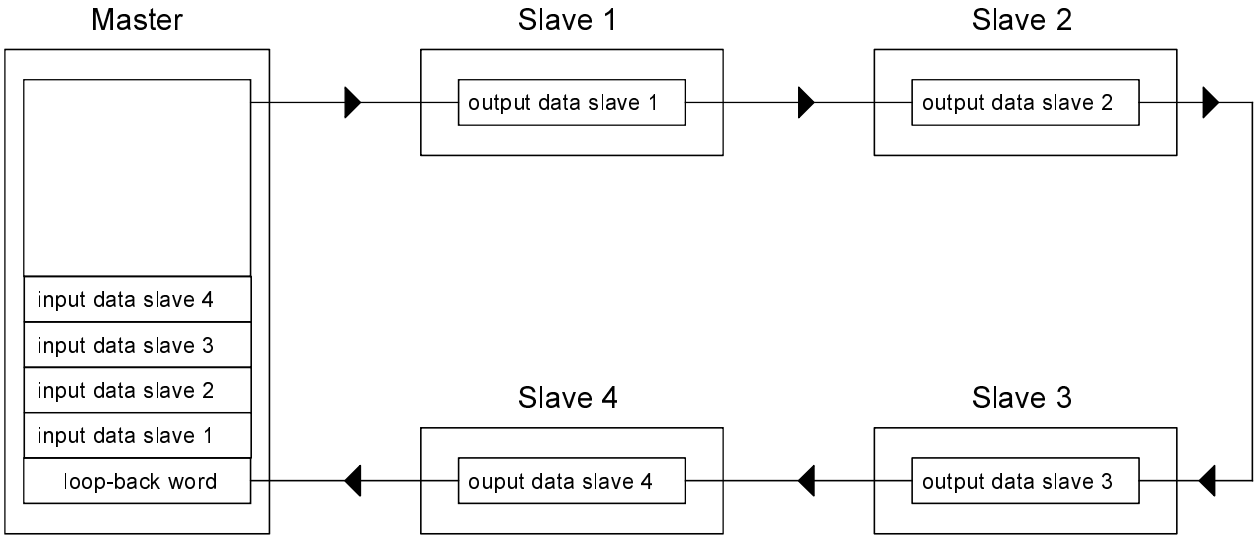
The loop-back word is a 16 bit data pattern, which clearly differs from all valid ID-codes. In one transmission cycle, the bus master first of all sends the loop-back word to the first station. The input information from individual stations is pushed ahead into the master before the loop back word. After the loop back word, the output information is transmitted from the master to the linked stations.

10.4 Data cycle

In the data cycles, the output data (OUT data) is transported from the master to the stations, and in the same cycle the input data (IN data) from the stations are transported to the master. As in the IDcycle, in the data cycle too the master first pushes the loop back word through the ring.



Distribution of data before the data cycle



Distribution of data after the data cycle

Using the ID cycle, the master knows the exact physical location of the individual stations in the ring. Thus it is able to address each station through this by writing the output information to the corresponding position in the process output image, and reading the input information from the memory locations of the process input image belonging to the station.

10.5 Data classes

10.5.1 Process data

Process data are information on status (target and actual values) such as motor rotational speed, switch settings, start/stop commands, etc. These have a direct effect on processing. Process data are identified by their being exchanged quickly and cyclically between the control and the sensors and actuators. One further identifier is its information content. This only has a few bits, for example, only the information 1 = open or 0 = closed is transmitted from or to a valve. An analog value is transmitted according to the accuracy required as a value with 8, 12, 16 or 32 bits.

10.5.2 Parameter data

Parameter data deal with data blocks such as initialisation data for operator and display panels, frequency converters or servo controllers. In contrast to the process data, parameter data only seldom change; this means that they have a low dynamic response and only seldom need to be updated. Parameter data are therefore acyclic data which are only transmitted when required, for example in the start-up phase of a machine, on conversion of processing equipment, or if faults arise.
Transmission speed requirements for parameter data is lower than for process data, as these do not have direct effects on inputs/outputs.
The amount of information in parameter data lies between a few bytes and a few hundred bytes per data set.

10.6 Data transmission channels

With the Interbus-S, process data and parameter data are transmitted over two data transmission channels which are independent of one another, the process data channel and the parameter data channel. In this way, data transmission is additionally optimised. Dependent on their function, not all stations need to support both channels. However, intelligent devices such as servo controllers, which transmit process data and parameter data, require both channels.

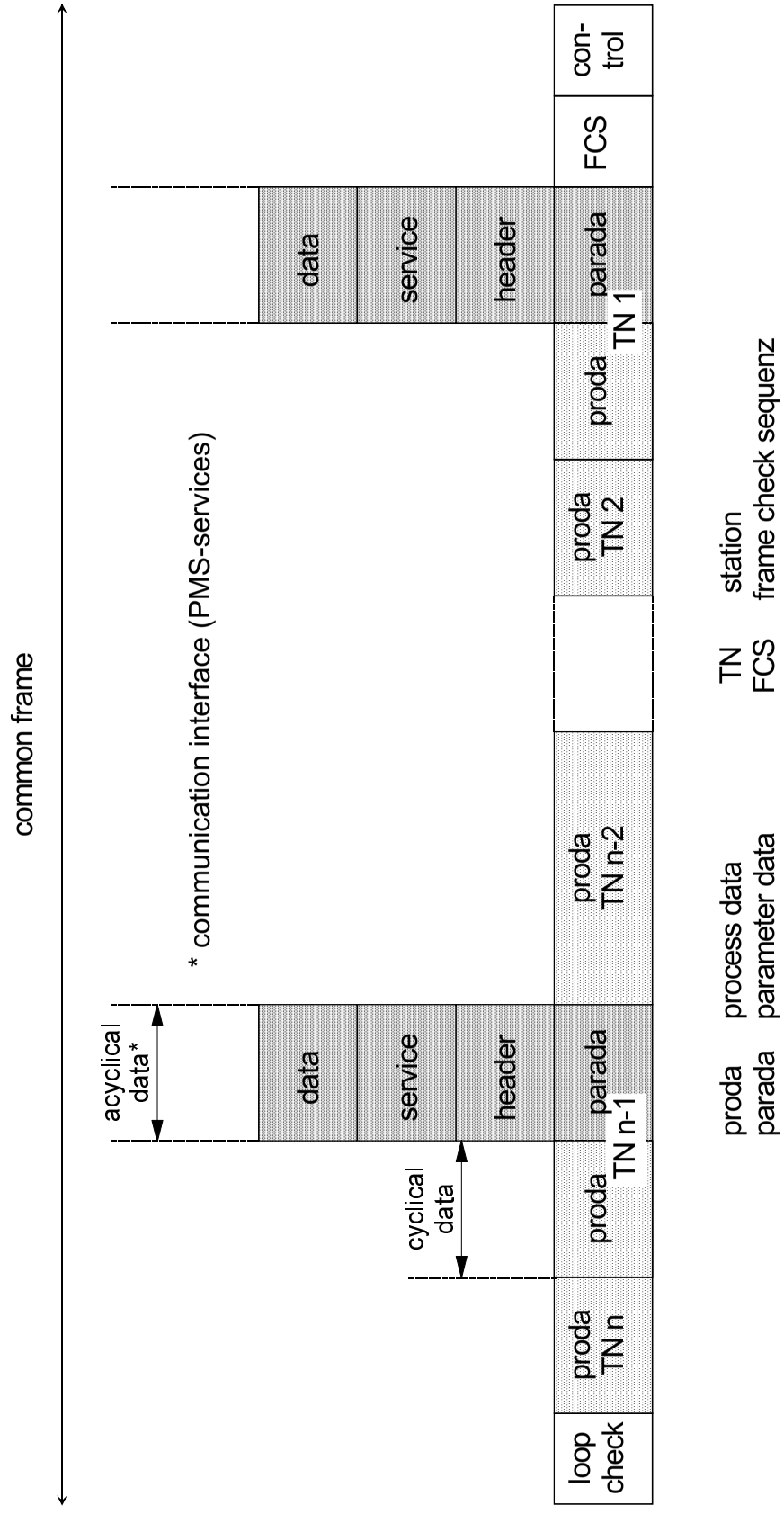
10.6.1 Process data channel

Cyclically transmitted process data is accessed over the process data channel. A current map of the input status is made available to the application program. The application program on the other hand stores the output data which were transmitted on the process data channel to the outputs. This direct memory access avoids laborious service access procedures.

10.6.2 Parameter data channel

Using this, acyclic parameter data (complex data structures) are transmitted if required. The parameter data channel is integrated in the transmission protocol. In the common frame, gaps are left in those spaces where stations are addressed which exchange parameter data. If a transmission of parameter data is necessary, the parameter block is disassembled into individual segments, which are as large as the gaps. One of these segments is transmitted in each cycle until the whole parameter block has been sent. The segments are re-assembled at the receiver. The simultaneous transmission of process data and parameter data is called a hybrid transmission procedure.

10.6.3 Hybrid transmission procedures



10.7 Communication interface

In order that the intelligent stations in an Interbus-S system can communicate with one another, the type and method of parameter data transmission is determined by the **P**eripherals **M**essage **S**pecification (PMS). PMS is a user interface according to the international MMS model (**M**anufacturing **M**essage **S**pecification) and lies on layer 7 of the OSI layer model. Access to parameter data is carried out over the PMS communication service.

For transmission, the parameter data are disassembled and re-assembled again after transmission. In Interbus-S, this accepts the **P**eripherals **C**ommunication **P**rotocol (PCP). This protocol software makes available services necessary for the assembly and disassembly of connections as well as the services for data transfer.
The formal description of this service is laid down in the PMS.

10.7.1 Communication entities

For simple data exchange between devices from different manufacturers, a unified representation of the data to be transmitted, covering all manufacturers, is required. This is achieved by the mapping of device parameters and also device functions on so-called communication entities, which are described using several attributes such as index, data type, entity type, name etc..

10.7.2 Entity description

Entity description contains all the characteristics of an entity

Index	Entity code	Data type	Length	Password	Access groups	Access rights	Symbol
0x5FF0	Simple variable	Octet string	6	0	0	write-all	POSA

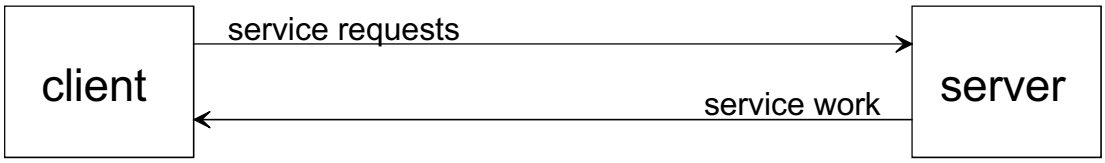
Example for an entity description

10.7.3 Entity directory

The index is listed together with the description of the characteristics (attributes) of the entity in a standardised list, the entity directory (ED). This is comparable to the entry in a telephone directory (the index corresponds to the telephone number; the sub-index to the extension). Each station which exchanges information over the parameter channel has its own entity directory in which all its communication entities are described .
Once entered in the ED, the entities (the device functions/device parameters) are accessible to other PCP stations.

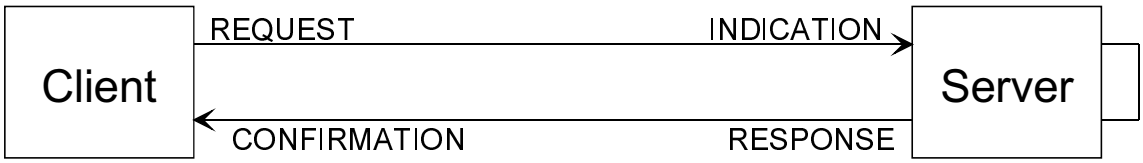
10.7.4 Access to communication entities

Access to communication entities is carried out using a request-response procedure. One station sends a request to another station. The service requester is the client; the station which is to carry out the request is the service supplier, the server.

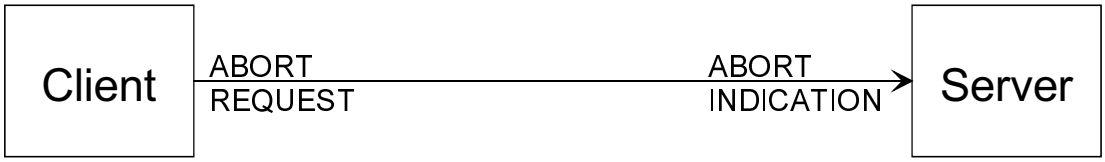


Basically, using Interbus-S, master and slaves can accept both client and server functions. For service requests and service work, the client and the server use communication services.
This type of acknowledged service comprises four basic operations:

Operation	Function
Service request	the client informs the server of a service request
Service indication	informs the server of the service input
Service response	the server presents the service acknowledgement to the client
Service confirmation	confirms to the client that the service has been carried out (service acknowledgement)



There are also unconfirmed services which differ from confirmed services in that a service request is carried out but no acknowledgement is returned.
One example of such as service is the connection clearance (abort).



10.7.5 Access security

For data which should only be accessed by a particular station, access security can be set up by the device manufacturer, i. e. the right of access to communication entities is limited to certain stations.
Access security can be set up by

- ⇨ defining access groups, all of which can access certain communication entities.
- ⇨ defining passwords. A station can only access a communication entity if it knows the password defined for that communication entity.

10.7.6 Communication relationships

Before data can be exchanged between two PCP stations on the Interbus-S, a logical connection must exist between the two stations.
This type of logical connection is called a communication relationship.

10.7.7 Communication relationship list

The IBS master applies a list for each PCP station, in which all permitted communication relationships are specified, independent of the time of use.
In these communication relationship lists (CRLs) are stored the connection type and the connection parameters, under which the communication relationship can be constructed.
The logical connections configured in the communication relationship list ensure the trouble-free exchange of data between two communication stations.
The connection parameters of both communication partners are checked during the connection to see that they agree before information is exchanged.

CR	Size of the transmitter buffer Low-Prio	Size of the input buffer Low-Prio	Supported PMS services

Construction of a CRL input

Some of the attributes of the CRL input are not supported by Interbus-S. They are only available on the Profibus for reasons of compatibility and are equipped with standard values.

Communication reference
The CRL is constructed line by line. Each permitted communication relationship receives a number, the communication reference (CR). This clearly identifies the communication relationship. This is necessary so that individual devices can be differentiated. The numbering of the devices is carried out automatically by the IBS master on initialising the IBS system. It assigns numbers starting from 2 following the sequence of the physical bus construction.

Size of the transmitter buffer Low-Prio
This attribute contains the maximum possible length of the **P**rotocol **D**ata **U**nit (PDU) in the lowest priority transmission direction which can be processed in this communication relationship.
For COMPAX, this value is = 64 bytes.

Size of the input buffer Low-Prio
This attribute contains the maximum possible length of the message (PDU) in the lowest priority reception direction which can be processed in this communication relationship.
For COMPAX, this value is = 64 bytes.

Supported PMS services
This attribute gives information on which services are supported in this communication relationship.

10.7.8 Communication services

The Peripherals Communication Protocol offers the user a number of standardised PMS services which can be divided into three groups:

User services
Access to communication entities is carried out using this service. User services also include the reading and writing of device parameters and starting and stopping programs to PCP stations.

Administration services
These services are required for the exchange of information using the communication entities. For example, using the service "Get OV", the description of a communication entity is read out.

Management services
The management services facilitate and ensure the operation of the communication system. The assembly and disassembly of a connection are included in these services.

10.7.9 Example of the operation of services

A COMPAX needs to be parameterized. Before a parameter can be transmitted, first of all the logical connection between the master and the COMPAX must be constructed. This step is initiated as a client by the master.

For the construction of a connection, the basic service operation Initiate-Request is used. For this, the CR of the communication partner (the server to be addressed) must be given, amongst other things.
After checking whether the construction of a connection is permitted, the server conveys the success or failure of the connection using the Initiate-Response. In the user program, therefore, plans must be made for two possible results:

1. The Initiate Confirmation with a positive result.
   The connection is made.

2. The Initiate Confirmation with a negative result.
   Using the automatic test mechanisms, it has been determined in the system that a connection is not permitted, because the inputs in the CRLs do not correspond.
   An evaluation of the error message is required.

10.8 Overview of the PMS services

10.8.1 User services

Service	Meaning
Start	Starts a program (after reset)
Stop	Stops a program
Read	Reads a variable
Write	Writes a variable
Information report	Unacknowledged transmission of a parameter

10.8.2 Administration services

Service	Meaning
Status	Reads the devices/user status
Identify	Reads the manufacturer's name, type and version
Get-OV	Reads an entity description

10.8.3 Management services

Service	Meaning
Initiate	Make a connection
Abort	Break a connection
Reject	Reject an unpermitted service

10.9 Process data control

This function determines which communication entities are mapped on the process data channel so that they can be accessed cyclically.
Dependent on the direction of the data flow, differentiations are made between process output data (POD) and process input data (PID). POD are the data which the master sends to the slave devices; PID are the data which the master receives from the slave devices.
COMPAX has 2 bytes (1 word) of process data, which means 2 bytes of POD and 2 bytes of PID.

10.9.1 POD control

Using the process output data it is possible to describe cyclically one of the following COMPAX communication entities:

Entity name	Meaning	Index
CONTROL	Control commands	0x5FE9
SPEED	Traversing speed	0x5FEC
OVERRIDE	Reduce traversing speed	0x5FEE
OUTPUT_WORD	Log. status of the 16 digital outputs	0x5FF7
START_N	Carry out program set N	0x5FF4
START_N_GO	Program start from set N	0x5FE5

The entity which is mapped on the POD channel is determined by the value of the entity OUT_SELECT. After Power_ON, the entity OUTPUT_WORD is automatically mapped on the process output data.

Using the entity OUT_ENABLE, the allocation (display) of the POD can be blocked/released on the corresponding communication entity.
If the entity OUT_SELECT is described, then automatically the POD channel is blocked through OUT_ENABLE and therefore must be explicitly released again.

10.9.2 PID control

Using the process input data, it is possible to read cyclically one of the following COMPAX communication entities:

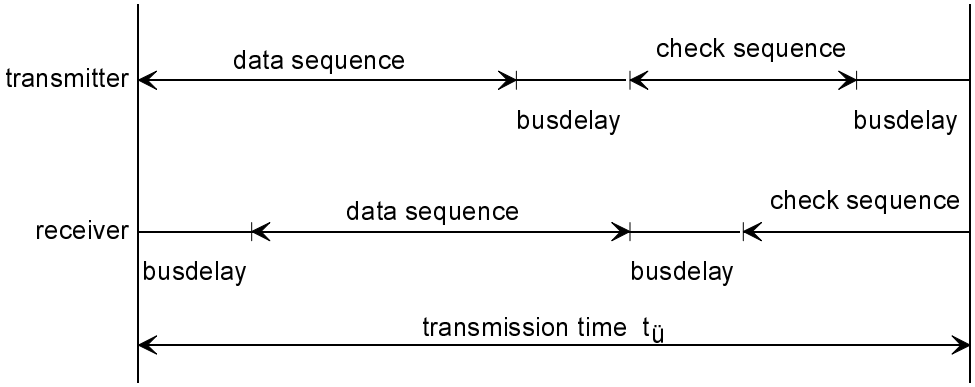
Entity name	Meaning	Index
S3	Contouring error	0x5FFA
S4	Current traversing speed	0x5FFC
S5	Current torque of the motor	0x5FFB
INPUT_WORD	Log. status of the 16 digital inputs	0x5FF8
OUTPUT_WORD	Log. status of the 16 digital outputs	0x5FF7

The entity which is mapped on the PID channel is determined by the value of the entity IN_SELECT. After Power_ON, the entity INPUT_WORD is automatically mapped on the process input data.

10.10 Data cycle time

The transmission time in an Interbus-S system depends basically on the amount of information to be transmitted. A certain transmission time is required for each individual bit at a fixed transmission rate. As the common frame telegram in an Interbus-S system is always the same length, the data cycle time in an existing system is always constant.

The following times occur in an IBS system:



If data is sent from the master to the stations, then station data to the master will meet with a certain delay - the bus delay. After transmitting data in the data sequence, the master waits until the last input data of the data sequence, the loop-back word, has been re-received. Then it starts the check sequence. Here too, the data to the master meet with a delay. Therefore to calculate the transmission time, the following times must be taken into consideration:

Data cycle time $t_{cyc} = t_{data\ sequence} + t_{check\ sequence} + 2*t_{bus\ delay}$

If individual values are placed in this formula, then the following formula is given to calculate the transmission time:

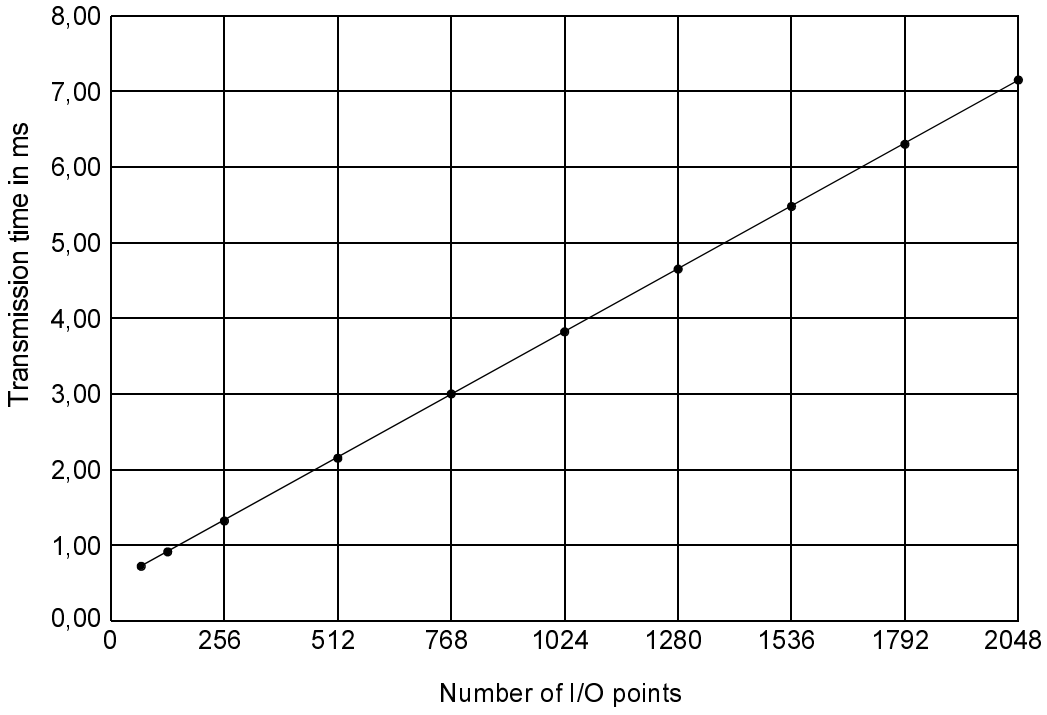
$$t_{cyc} = [13*(6+n) + 4*m] * t_{Bit} + t_{SW} + 2*t_{Ph}$$

t_{cyc} = data cycle time in milliseconds
n = number of user data bytes
m = number of stations installed on long-distance bus
t_{bit} = bit duration = 2 µs at 500 kbits/s
t_{SW} = Software running time = 200 µs
t_{Ph} = running time on the transmission medium
for copper: t_{Ph} = 0,016 ms *l/km
l = length of the long-distance bus cable in kilometers

The formula to calculate the IBS data cycle time is made up of the following

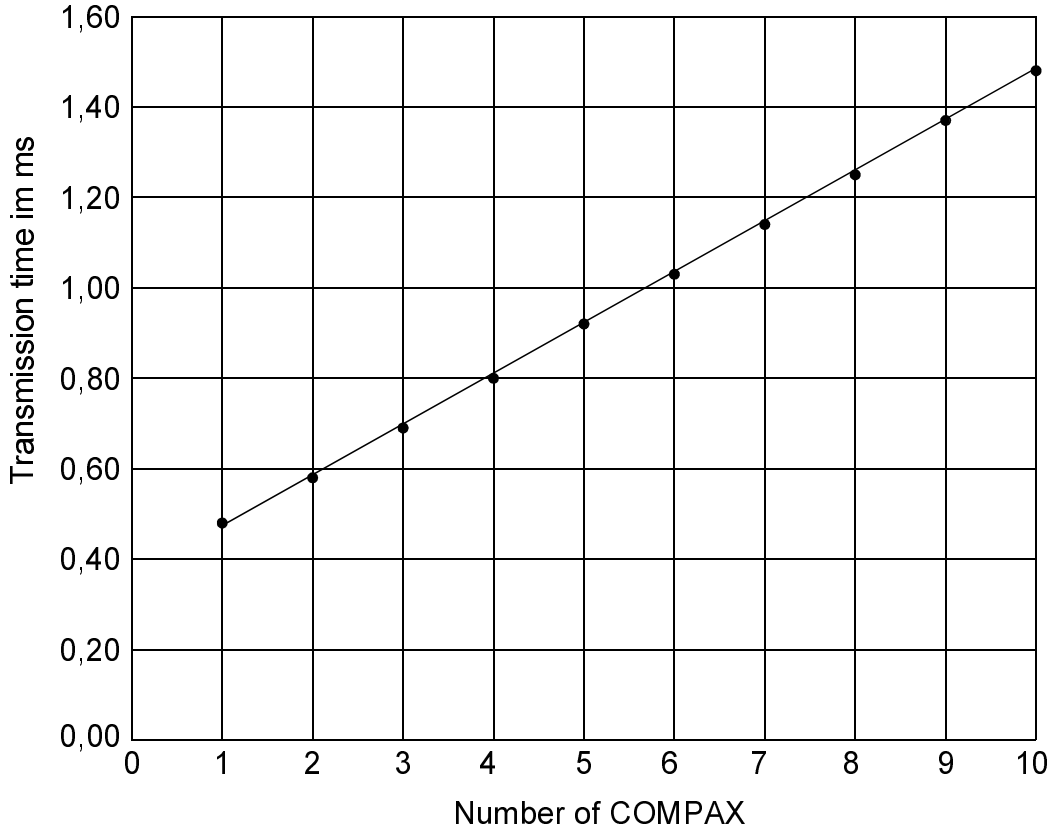
- ☞ Bit transmission time
- ☞ Software run time in master
- ☞ Run time on the transmission medium

The software run time is determined by the firmware of the IBS master and is thus dependent on implementation. The run time on the transmission medium depends on the length of the long-distance bus cable. Both contributions to the formula are relatively small compared to the pure bit transmission time.
The main contribution to the data cycle time is determined by the bit transmission time. This is calculated from the product of all the bits to be transmitted and the run time per bit. The number of bytes in the common frame telegram is given by the number of user data bytes plus 6 bytes of frame information. Frame information is made up from 2 bytes of loop-back word, 2 bytes of data security information (CRC) and 2 bytes of end information. To calculate the number of all the bits to be transmitted, the number of bytes in the common frame telegram must be multiplied by 13, as in transmitting data from one station to the next, 8 bits are packed into 5 bits of frame information.
The formula given above results in an approximately linear course for the data cycle time graph over the amount of I/O information.
For the following diagram, the following assumptions are made:
- Number of stations on long-distance bus: 10
- Length of long-distance bus cable: 1.2 km



10.10.1 Data cycle times using COMPAX

Compax has a total of 4 bytes (2 words) of user data. Of this, 2 bytes (1 word) of process data and 2 bytes (1 word) are required for the PCP communication.
The following diagram is based on an assumed line length of 0.1 km.
Each COMPAX is a station on a long-distance bus.



10.11 Transmission time of a PCP service

The calculation of transmission times for a PCP service is made using the following formula:

$$T_D = T_L + ((N+O_D-1)/m+1)*T_{cyc} + T_{L7}$$

T_D = PCP service transmission time [ms]
T_L = latency = 2*data cycle time [ms]
O_D = service dependent overhead [byte]
N = number of user data [byte]
m = width of param. data channel [byte] - 1 byte (code octet)
T_{cyc} = data cycle time [ms]
T_{L7} = layer 7 run time [ms]

10.11.1 Write request time

This is the time from transmitting a write service by the client until the arrival at the server.
O_D = 7 bytes for a Write-Request service
T_{L7} = 4 ms (average value for implementation today)

$$T_{Wreq} = T_L + ((N + 7\text{-}1)/m+1) * T_{cyc} + 4ms$$
$$= 2*T_{cyc} + (N+7)* T_{cyc} + 4ms$$
$$= (N+9)* T_{cyc} + 4\ ms$$

10.11.2 Write Response time

This is the time from transmitting the acknowledgement of a write service by the server until the arrival at the client.
N = 0 byte
O_D = 4 bytes for a Write-Response service
T_{L7} = 4 ms (average value for implementation today)

$$T_{Wrsp} = T_L + ((0 + 4\text{-}1)/m+1) * T_{cyc} + 4ms$$
$$= 2*T_{cyc} + 4* T_{cyc} + 4ms$$
$$= 6* T_{cyc} + 4\ ms$$

10.11.3 An example using COMPAX

The IBS system consists of a master and 6 COMPAX axials. In this there is a data cycle time of 1.03 ms at a system cable length of 100m.
The acceleration time (2 bytes), the speed (2 bytes) and the position (6 bytes) are transmitted in succession to one of the COMPAX.

T_{Wrsp} = 6* 1.03ms + 4 ms = 10.18 ms
T_{Accel} = 11* 1.03 + 4 ms + 10.18 ms = 25.51 ms
T_{Speed} = 11* 1.03 + 4 ms + 10.18 ms = 25.51 ms
T_{Pos} = 15* 1.03 + 4 ms + 10.18 ms = 29.63 ms
T_{Bsp} = 25.51 ms + 25.51 ms + 29.63 ms = 80.65 ms

As a comparison:
With the RS232 interface of the COMPAX, the transmission time for one character at 9600 baud is approx. 1 ms.
For the example stated, the following is to be transmitted

- AL1000$^{C}_{R}$ = 7 characters + 3 characters response
- SD80$^{C}_{R}$ = 5 characters + 3 characters response
- PA10000$^{C}_{R}$ = 8 characters + 3 characters response

This results in a transmission time of 29 ms.

11 Profibus

Profibus implementation can consist of several logical masters and slaves, where slaves can also be addressed by different masters. For Profibus data transmission, two different procedures are defined. For the exchange of message data between an active (master) and a passive (slave) station, the master-slave procedure is used, and to transmit the right of transmission from master to master the Token Passing procedure is used.

Physically, the Profibus has the shape of a line, where short spur lines are possible. As a transmission medium, bundle-assembled shielded 2-wire aerial cable with matching resistors conforming to the standard (DIN19245 part 1, or modified in part 3) is used between supply voltage, ground and signal lines. Data transmission is carried out according to the RS485 standard. One segment may have a maximum length of 1.2 km and consist of 32 stations. Several segments can be joined using repeaters, so that in total 127 stations are possible (repeaters require their own address). Addresses lie in the range between 0 to 126, the address 127 is reserved for multi and broadcast.

The transmission speed can be adjusted in stages dependent on the length of the line from 9.6 to 500 kbits/s, on the other hand DIN 19245 / part 3 additionally suggests higher transmission speeds starting from 1,5 Mbit/s. In the most recent implementations, there are already plans for up to 12 Mbit/s.

Limits are imposed by the bus topology, where only selected transmission and reception of data is possible, this being half-duplex operation. The maximum message length at the lowest level (layer 1) is 255 bytes per telegram (gross baud rate).

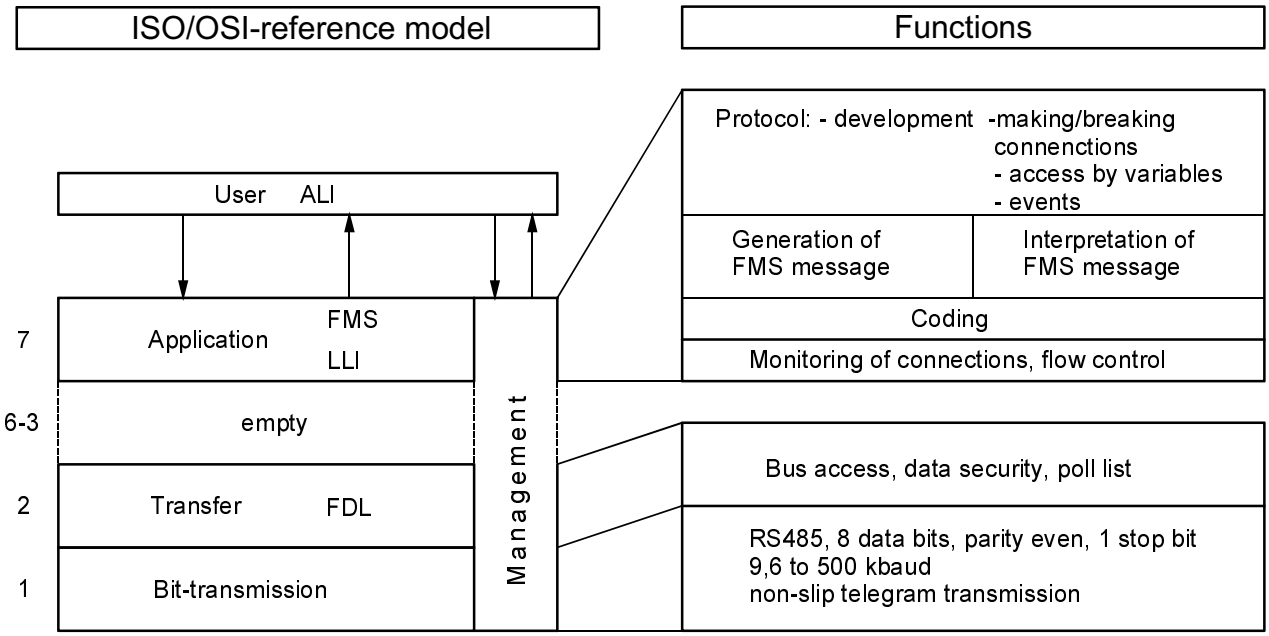
11.1 The Profibus communication model

The Profibus architecture follows the ISO/OSI 7-layer reference model and is divided into three hierarchical layers: application layer (layer 7 of the reference model), data security layer (layer 2) and physical layer (layer 1).

The specification for both lower layers is in DIN 19245 part 1. In the highest layer of the Profibus specification, the application layer (DIN 19245 part 2), which is itself divided into two sublayers (Fieldbus Message Specification, FMS and Lower Layer Interface, LLI), subfunctions (e.g. information display for layer 6) of layers 3 to 7 of the reference model are implemented, without maintaining their strict organisation. Original protocols for these layers are greatly simplified for reasons of efficiency, e.g. combining short messages into blocks, segmenting long messages and routing through a network have been removed. The protocols of the Profibus are thus reduced to the function scope necessary, in order that they fulfil the time requirements.

So that real application processes can communicate with one another (programs running which are not themselves part of the communication layer, such as application programs with operating systems), they exchange information on process entities. For this, the process entities to be exchanged must be made known as a communication entity to the communication system. A public entity directory made available by the stations (ED), standardised device characteristics, identical services and unified interfaces are the basis for open communications between devices from different manufacturers. This unified view of a device is known as a virtual field device (VFD).

In addition to the functions with regard to communications, the operation of a communication system also requires a multitude of organisational functions. In order to implement these, each layer of the original OSI/ISO reference model is extended to incorporate network management.



11.2 The Profibus application layer

11.2.1 Entities, entity directories

The exchange of information is object oriented. In the communication of applications, the client (master) accesses local server entities (slaves), in order to exchange structured information. Each entity has a data type (boolean, integer, floating point, octet string, etc.), and by means of attributes such as index (short addresses), the name of the entity, password, access groups, access types (reading, writing), entity status (e.g. no reading can be carried out during the update), and a few other specific descriptions (e.g. reference to the memory address of the entity) an access key is defined. Individual entities must be addressed on access, as generally more than one entity exists per station. Using the Profibus, addressing can be carried out via the index, optionally via entity names (max. 32 characters) or even physically for service purposes (e.g. memory address).

All entity descriptions are entered in an entity directory (ED). The definition of the entity description is carried out at the station which also has the entity. This directory is called the Source-ED. Communication partners (clients) have either a complete or possibly only a partial copy of this directory (Remote-ED). Each active station has a number of Remote-ED's in addition to its own Source-ED. The ED of each active station can be called up during run time using a special service (Get-ED). The ED itself can consist of a maximum of 6 subdirectories. Structure information on the ED is stored in the ED header. The ED header can always be read out over index 0 of the ED. The type definitions for simple and aggregate data types are in the static data type and data structure directory. The actual entity descriptions (see above) are recorded in the static entity directory in which, for example, the data types are used from the static data type and data structure directory. In addition, there are other optional subdirectories such as the dynamic variable list directory.

11.2.2 Communication relationship lists

An entry in the communication relationship list (CRL) describes a logical channel operated by the user to communicate with another station. During the planning phase, the descriptions are picked up by the CRL and during run time the accuracy of the communication is monitored using this entry. Although the CRL basically fulfils the same function as in the Interbus-S, it is split into two and is of more complex construction. It consists of the FMS-CRL and the LLI-CRL. The CRL is built line by line and can be addressed via an index (communication reference, CR).

Each entry in the FMS-CRL consists of a static and a dynamic part. It contains the FMS-specific part of the description of a communication relationship. As with the ED, under the index 0, structure information (e.g. number of entries in the FMS-CRL) is stored in the list itself.
An entry (CR > 0) in the FMS-CRL consists of one static and one dynamic part.

Static part of FMS-CRL

CR	max. PDU transmission length (high/low priority)	max. PDU receive length (high/low priority)	supported FMS services

Max. outstanding client services (acknowledged)	Max. outstanding server services /acknowledged)	CR-type	Symbol	VFD-Pointer

Dynamic part of FMS-CRL

Outstanding client services (acknowledged)	Outstanding server services (acknowledged)	CR state

CR: Communication reference
The CR is the (local) unique address (index) for a line in the CRL. In this way, each communication relationship can be uniquely referenced.

Max.PDU transmission length
This states the maximum length of the FMS-Protocol-Data-Unit (FMS-PDU) in the direction of transmission.

Max.PDU input length
This states the maximum length of the FMS-Protocol-Data-Unit (FMS-PDU) in the input direction.
supported FMS services
using this octet string (8 bytes), all possible FMS services supported by the device are named. Typical services are Read, Write, Start, Stop, Get-OV, Initiate, Abort and also Phys-Read and Phys-Write.

Max. outstanding client services (acknowledged)
This states how many acknowledged services may be due as clients to this communication relationship.

Max. outstanding server services (acknowledged)

This states how many acknowledged services may be due as servers to this communication relationship.

CR type
This attribute states whether this communication relationship works connection-oriented or connection-free. As in connection-free communication relationships the reverse LAN channel is actually“ missing, this can only be used for carrying out unacknowledged services.

Symbol
Each communication reference can be given a symbolic name, and the symbol length is stored in the CRL header.

VFD-Pointer
This is the pointer to the allocated virtual field device (VFD), which contains all entities and entity descriptions of communication entities which can be used by services, for example, the ED is a component of the VFD. If the pointer is set to 0xFFFFFFFF, then the entry is not supported.

Outstanding client services (acknowledged)
This states how many acknowledged services are due at that moment as clients to this communication relationship.

Max. outstanding server services (acknowledged)
This states how many acknowledged services are due at that moment as servers to this communication relationship.

CR-State
This attribute states whether the communication relationship is built or not or whether it is being built.

Each entry in the LLI-CRL consists of a static and a dynamic part. It contains the LLI-specific part of the description of a communication relationship. Under index 0, the header belonging to the CRL with structure information can be read out.

Static part of the LLI-CRL

CR	Address (local SAP, remote addr., remote SAP)	LLI-context (Type, max. SCC, RCC, SAC, RAC,ACI)	Conditional (CCI, multi-plier, connection attribute)

Dynamic part of the LLI-CRL

Status	Remote address	Remote SAP	Special counter (request counter)	Image memory	Poll entry	New	Old

Addresses
Local LSAP states the service access point of the actual FDL-layer, which is to be used for this CR. Remote Address states the address of the Remote FDL (device address) to which the CR refers. The Remote LSAP states the service access point of the Partner-FDL.

LLI-context

The context contains all attributes of a CR, which must be compatible with the corresponding attributes of the communication partner, for example whether it is an acyclical master-master connection or an acyclical connection between a master and a slave. The input time interval for an acyclic connection is fixed.

Conditional
Amongst other things, the monitoring period for a cyclical connection is given here. The multiplier states how the CR should be entered in the poll list of the FDL.

The dynamic part contains, above all, internal information which must not be planned. This includes, for example, an internal status, the current selected remote address.

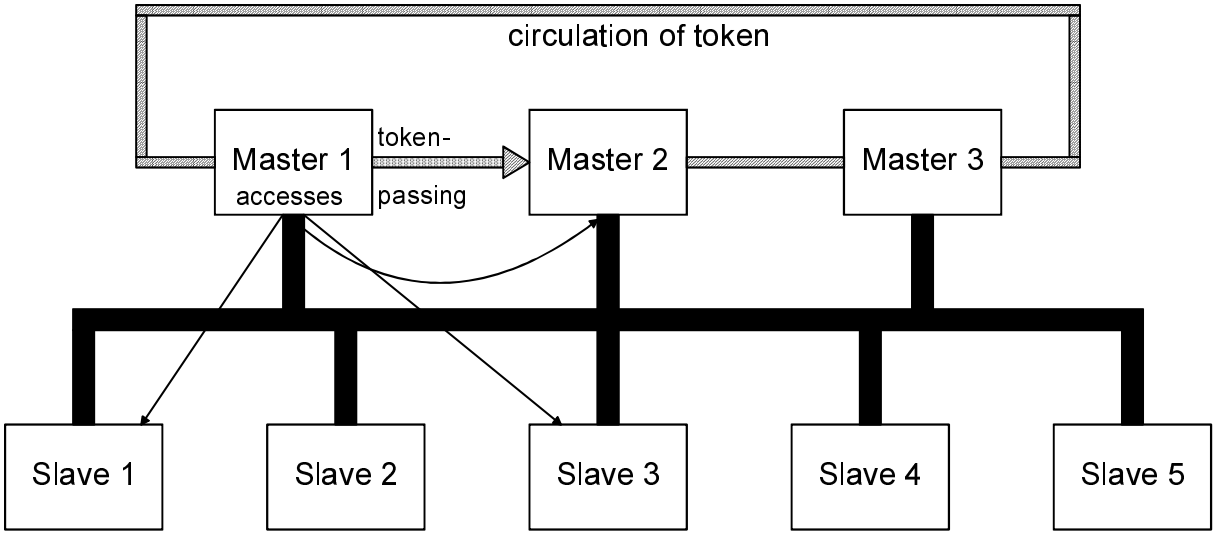
Poll entry
The poll entry is important as it determines whether the remote stations entered in the poll list via the CR should also be unpolled or whether the poll list entry should be blocked.

Map memory
The map memory of the master/slave-LLI-CRL contains the cyclical data to be implemented (reading/writing).

11.3 Transmission oriented layers

11.3.1 Bus access

Layer 2 (Data Link Layer, FDL) is responsible for controlled bus access. For this, Profibus uses a hybrid process: distributed according to the token-passing procedure between active stations (masters) for passing on the token (right of transmission) and centralised according to the master-slave procedure for communication between active and passive stations. Passive stations are passive“ with regard to bus access; such stations only respond to demands from a master station. Only the active station in possession of the token can control message cycles on the bus.



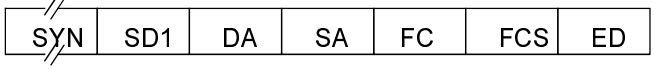
The token rotates in a logical ring formed by all active stations. If there is only one active station, then on expiration of the token holding-time, this will pass the token back to itself. Each active station passes the token to the active station with the next highest address, and that with the highest to that with the lowest. The circulation time of the token determines the interval until the the token has passed once around the logical ring and which is thus minimum time to transmit a high priority message.

If a station has possession of the token, it can then handle acyclic and cyclical message cycles. In acyclical exchanges, individual message cycles are sporadically handled if there is a request from the application layer. With cyclical message exchanges, stations which are entered in the poll list (active and passive stations) are polled using Send and Request Data“. However, when using cyclical requests, this is not started until all high-priority acyclical cycles have been processed. Even during cyclical operations, low priority acyclical message cycles are subordinate.

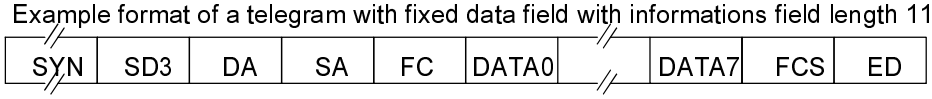
11.3.2 Telegram construction

A telegram on the bus consists of a sequence of UART characters. A UART character is a start-stop character for asynchronous transmissions and has the following bit distribution: 1 start bit (binary 0“-signal), 8 data bits, 1 even parity bit and one stop bit (binary 1“-signal). Before each call telegram, a pause time of at least 33 bits (binary 1“-signal) must be maintained. Between individual UART characters, however, no pause is permitted. The receiver checks each call to see whether the idle time (33 bits) has been maintained, otherwise the start, the DA/SA, FCS and stop bytes on each telegram will be checked. For each UART character, the start, stop and parity bits are checked. If the check has a negative result, then the complete telegram is rejected and must be repeated. For additional data security, the telegram uses a Hamming distance of 4, which means that two successive UART characters differ in at least 2 binary places.

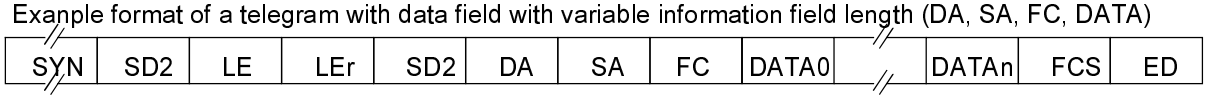
Example format of a telegram without data field with information field length 3 (DA. SA, FC)



Example format of a telegram with fixed data field with informations field length 11 (DA, SA, FC, DATA)



Example format of a telegram with data field with variable information field length (DA, SA, FC, DATA)



SYN synchronisation bits (minimum 33 idle bits = '1'), only in request frames
SD1..4 different start bytes
DA destination address
SA source adress
FC control byte
FCS check byte
ED Endebyte
LE length byte (4 to 249)
LEr length byte repetition

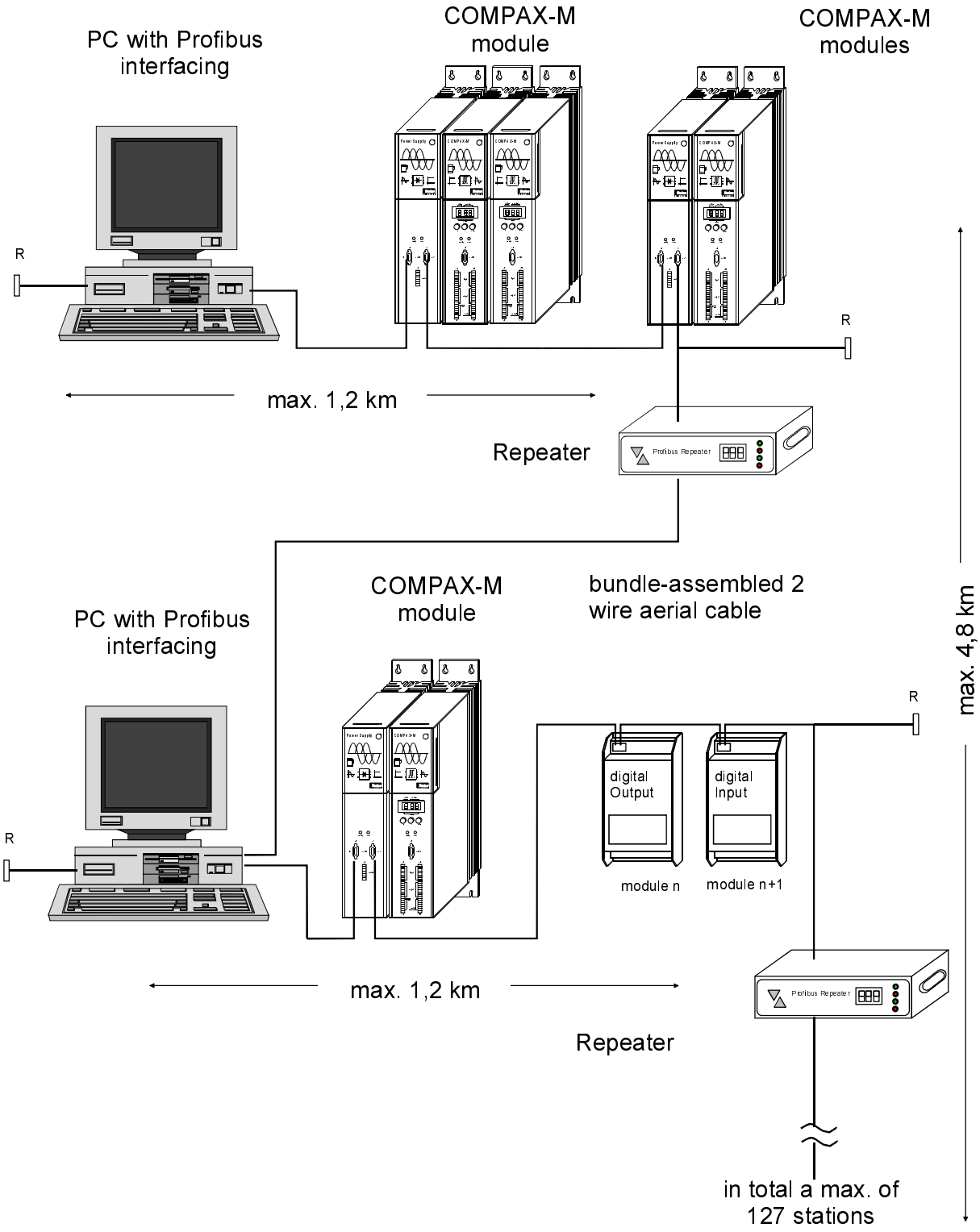
11.4 Profibus DP

For time critical applications and applications which do not require the high functionality of the FMS services (DIN 19245 part 2) and the continuity of MMS, an alternative protocol has been developed using Profibus-DP (DIN 19245 part 3). DP stands for distributed peripherals. The service developments of layer 7 (part 2 of the Profibus standard) are removed and data traffic is mapped directly by the application (via the Direct-Data-Link-Mapper, DDLM) onto the FDL. The DDLM offers the user easy access to FDL over standard communication functions. The functionality of Profibus-DP corresponds in this way to a fast I/O or sensor/actuator bus.

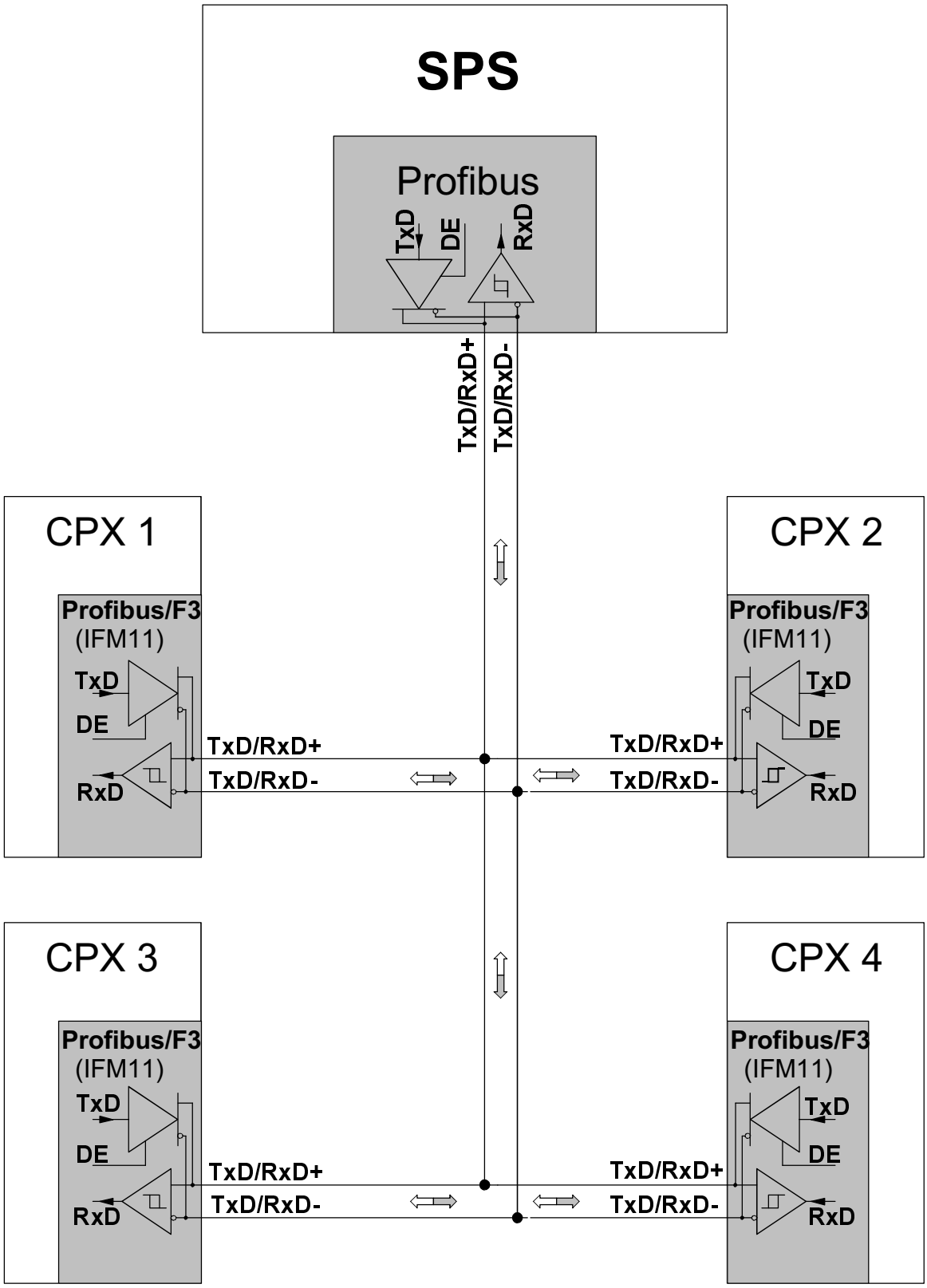
The bus access and transmission protocol described in DIN 19245 part 1 is completely maintained, only the maximum baud rate is increased. In order to fulfil the increased transmission rates, the line parameters are simultaneously re-adjusted (surge impedance, core cross-sections, line terminals, line capacity).

A Profibus-DP system can also be designed as a hybrid network; this means that in addition to DP-masters and slaves, the integration of FMS masters and slaves is also possible. Inevitably, the maximum transmission speed must remain limited to 500 kbits/s (part 1 of the Profibus standard). It is possible to have devices which can be operated in parallel to the Profibus-DP and the Profibus protocol. With Profibus-DP, in contrast to the FMS protocol, there is a fixed assignment between the DP-master and DP-slave. In planning, each DP-slave is allocated to a fixed DP-master. If there are also FMS masters/slaves in the network, then communication between DP and FMS masters is required for the purpose of token passing. At the user level, however, no information can be exchanged between FMS masters and DP slaves or between DP masters and FMS slaves or between DP and FMS masters. Profibus-DP networks are frequently implemented as mono-master systems; for reasons of efficiency, a maximum of 3 DP masters should be used in a Profibus-DP interconnected operation.

11.5 An example of Profibus topology



11.6 RS485 two-wire connection SPS and COMPAX



11.7 COMPAX as Profibus and Profibus-DP slave

COMPAX can be operated as a pure FMS or DP slave. In addition, COMPAX can operate both protocols simultaneously (hybrid operation), i.e. COMPAX can serve a DP master and several FMS masters simultaneously.

11.8 Compax as Profibus slave

For communications, entities are made available which are to be used for the control of COMPAX and for the reading out of status information. All descriptions of the communication entities are recorded in the ED. The entities can be summarised in the following groups:

Control
By means of the entities Control byte“, Status byte“, Control word“ and Status word“, COMPAX can be started and stopped, reference runs initiated, manual runs carried out and basic statuses interrogated. This also includes messages such as Machine zero approached“ or Programmed target position reached“.
Using the entity Command“, all commands which are defined for the COMPAX RS232 interface are given as ASCII strings.
In addition, entities are made available for manipulating the parameter sets of the servo-controller.

Diagnosis
Here different (only readable) entities are made available for interrogation of status values of the servo-controller
This includes variables such as the current torque of the motor, specific voltage level and also, for example, the temperature of the output transformer.

Positioning
Using this entity group, target position values (relative or absolute) are given for COMPAX and position values such as actual and target positions made ready. In addition, contouring errors at any moment can be retrieved.

Speed / acceleration
Here speed values or traversing profiles can be stated and the current traversing speed can be read out.

Inputs and outputs
The COMPAX digital inputs are partly free inputs or have a fixed allocated meaning; these include the initiation of a reference run via inputs 1 and 2, for example. Configurations for digital inputs can also be stated directly by the Profibus master via the entity Control word“ . In this, permission to assign inputs remotely must obtained in advance via the entity INPUT_MASK“ by COMPAX. Inputs can be read via the entity INPUT_WORD“.
Digital outputs can be set or reset via the entities OUTPUT_WORD“ or OUTPUT“ . In this, COMPAX must be informed using OUTPUT_MASK“ that selected outputs are being remotely set. As the entity OUTPUT_WORD“ is also readble, the status of the digital outputs can also be interrogated.

Programming

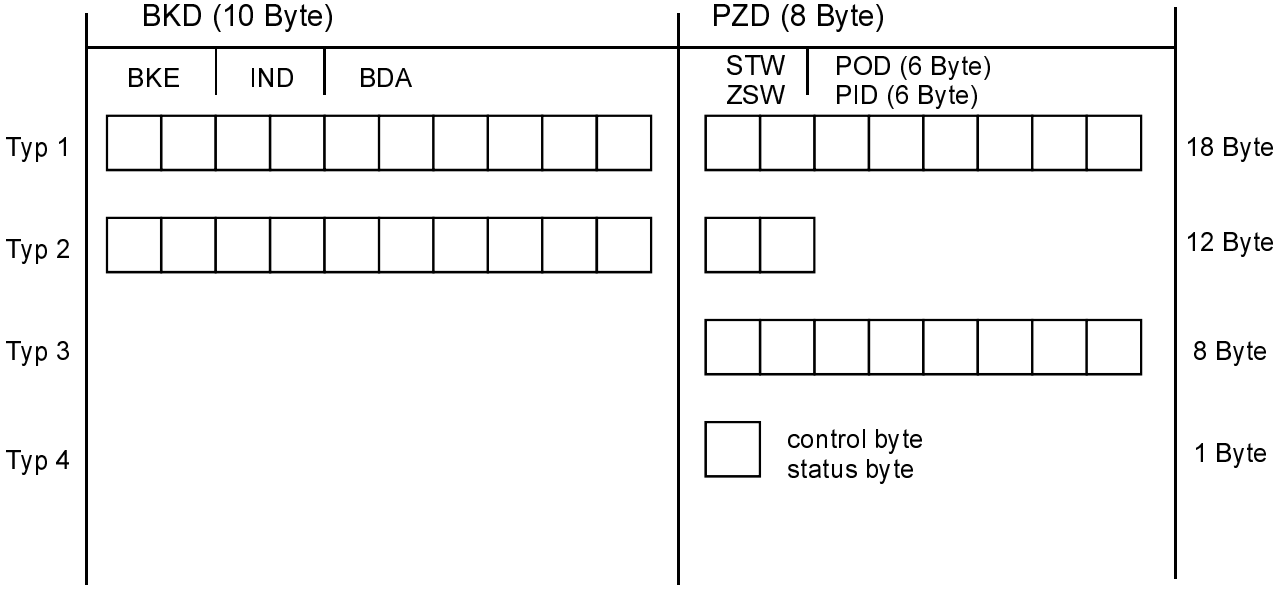
Using this entity group, the block memory can be read or written line by line by the servo-controller.

Process data control

Using these functions, certain communication entities can be placed on the process data channel (available in Profibus-DP and CPDE types 1 and 3 modes). These entities are then continually updated; this means that process output data (POD) are written cyclically by the DP master and process input data (PID) are cyclically read (for process data control, see below).

11.9 COMPAX as Profibus-DP slave

With the Profibus-DP, cyclical data traffic continually takes place between the master and slaves. In this each has an octet area defined which can be read or written. In COMPAX, each of these octet strings can have a length of 1, 8, 12 or 18 bytes. According to the length used, this input or output area is differently structured. Such structured areas in COMPAX are described, following the Profibus profile for variable-speed drives, as command process data entities (CPDE). There are four types selectable through COMPAX parameters:



BKD	command recognition data
PZD	process data
BKE	command recogntion (1. and 2. octet)
IND	subindex (3rd octet), frame no. (4th octet)
BDA	command data (5th to 10th octet)
STW	control word
ZSW	status word
POD	process output data
PID	process input data

Each CPDE on being installed in the output or input area is only readable (CPDE-read) or writeable (CPDE-write). The larger the CPDE is, the more extensive the amount of information which can be transmitted. In the maximum configuration (CPDE type 1), using a CPDE-write, the DP-master gives the DP-slave requests to process commands (PRC) as well as process data (PRD), consisting of a control word (CTW) and target values (POD). By means of the CPDE-read, the DP-master retrieves responses on command processing (PRC), as well as the process data (PRD), which consists of a status word (STW) and actual values (PID). Through the cyclical transmission of CPDE's, a special

handshake is required; by means of CPDE-write, a request from the DP-master is repeated until a response is received by COMPAX. Conversely, COMPAX makes the response available until such time as a new request is received from the master.

From the point of view of contents, almost all communication entities of the ED available over the FMS-protocol (see above) are covered by the PRC of the CPDE types 1 and 2. In addition, it is possible to request or change COMPAX entities directly via the BDA. A condition for this is that the size of the data field of the communication entity does not exceed the number of BDA bytes (6 bytes).

11.9.1 Communication entities on the PID/POD

It is possible to read cyclic communication entities via the PID or to write them via the POD. For this, CPDE type 1 or 3 must be set.

The following entities can be read cyclically:

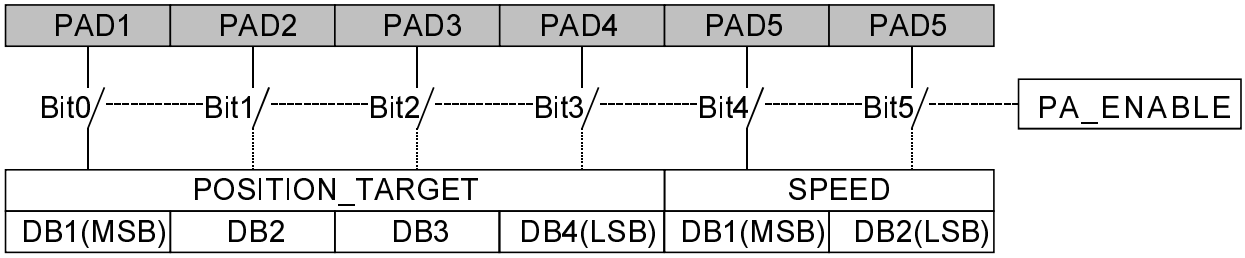
Entity name	Meaning	Index	Length
POSITION_ACTUAL	**Position actual value**	**0x12C1**	**4**
S3	**Contouring error**	**0x12C2**	**2**
S4	**Current traversing speed**	**0x12C3**	**2**
S5	**Current torque of the motors**	**0x12C5**	**2**
STATUS_BYTE	**Status byte**	**0x12CD**	**1**
INPUT_WORD	**log. status of the digital inputs**	**0x12DD**	**2**
OUTPUT_WORD	**log. status of the digital outputs**	**0x12E0**	**2**

The following entities can be written cyclically:

Entity name	Meaning	Index	Length
CONTROL BYTE	Control byte	0x12CC	1
CONTROL	Control commands	0x12D0	1
POSITION_TARGET	Target position entry	0x12D6	4
SPEED	Traversing speed	0x12D7	2
OVERRIDE	Reduce traversing speed	0x12DA	1
OUTPUT_WORD	Set/reset outputs	0x12E0	2
START_N	Carry out program set N	0x12EC	1
START_N_GO	Program start from set N	0x12ED	1

As both PID and also POD have a data width of 6 bytes, as many entities as required can be placed until the 6 bytes are filled up. A condition for this is, however, that the entities are assigned to offsets (starting points) which will not lead to an overlapping.

The assignment of communication entities to PID and POD can be achieved using communication entities PI_SELECT“ and PO_SELECT“. Using the entities PID_INI“ and POD_INI“, the assignment on switching on (i.e. on COMPAX Power On) of the PID or POD channel is defined using communication entities. Using the communication entity PA_ENABLE“, the writing using the POD channel must be explicitly released byte by byte.



12. The CAN-Bus

The CAN-Bus (Controller Area Network) originated in vehicle technology, where data has to be exchanged for synchronisation purposes between different control devices such as anti-theft alarms and ignition. As many of these technological requirements are of interest for general automation tasks, the CAN-Bus can today be found in industrial applications in many different areas.

CAN-Bus implementation consists of several masters which can exchange information with equal rights of access without a central entity being involved. The CAN-Bus has the physical form of a two-way closed line. A bundle-assembled shielded two-wire aerial cable is used as a transmission medium. The signals are transmitted differentially in the baseband; the coupling is carried out either conforming to ISO-DIS 11898 (CAN High Speed) or to ISO-DIS 11519-1 (CAN Low Speed) over monolithic or discrete tranceivers; special wiring for RS485 drivers is also possible. In order to be able to transmit at one Mbit/s, the maximum network expansion should be 40 m; at a lower data transmission speed, 1000 m can be achieved. The number of stations is unlimited in theory; however, in practice, the majority only have up to 64 stations.

Regarding bus access procedure, a modified variant of the CSMA/CD procedure is used, which is called CSMA/CA (Carrier Sense Multible Access with Collision Avoidance). This procedure ensures that the highest priority message is transmitted non-destructively without repetition in the case of simultaneous access.

The CAN-Bus system is highly reliable through there being suitable measures for error recognition, handling and limitation. Due to short latency for highest priority messages (e.g. approx. 150 µs at 1 Mbit/s), a fast reaction time for alarm messages is possible. In connection with the data rates which can be achieved, CAN is also suitable as a field bus system in the lower field area of automation installations.

With the specification of application layer protocols, e.g. CAL (CAN Application Layer), CAN fulfils the conditions of an open system. In addition to CAL, there are other layer-7 protocols: DeviceNet, SDS etc.. There are currently additional standards for device profiles for the drive engineering of variable speed drives and their sensors/actuators in preparation, which are adaptations of known profiles of established field buses (e.g. DRIVECOM). These device profiles are based on the layer-7 imlementation CAL, and are called CANOpen device profiles.
Even without the explicit design of a layer-7 protocol, CAN-Bus protocol stacks on the layer-2 level offer comfortable access to the communication functions.

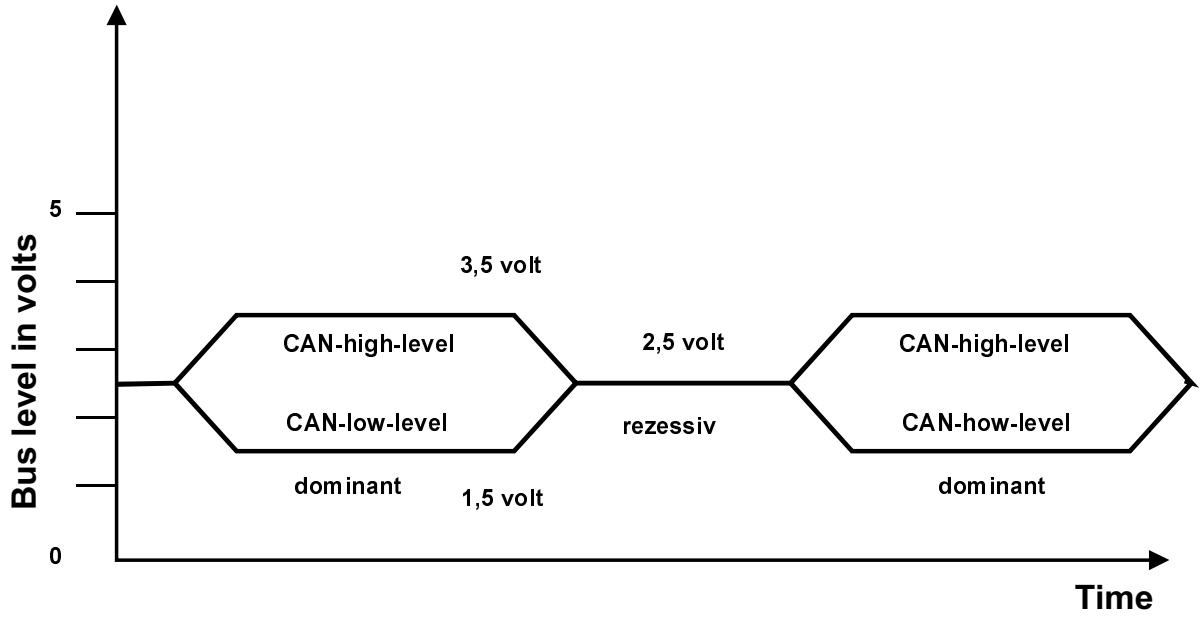
12.1 Transaction oriented layers

12.1.1 Physical layer

The representation of signals is carried out using the Non-Return-to-Zero procedure, which means that there are no forced signal exchanges (e.g. as in Manchester code) or explicit start bits as in transmission using UART characters (for example, see Profibus, RS485 ASCII transmission) in the synchronisation of bits between the transmitter and the

receivers. This results in increased demands for synchronous operations of oscillators (frequency precision), because the distance between possible signal edges is that much larger. However, as support, the maximum number of bits at the same level is limited to five and after this a filler bit with a complementary level is added by the transmitter to the bit flow (bit stuffing), which is removed by the receivers. Bit stuffing is not applied to all parts of the telegram frame.

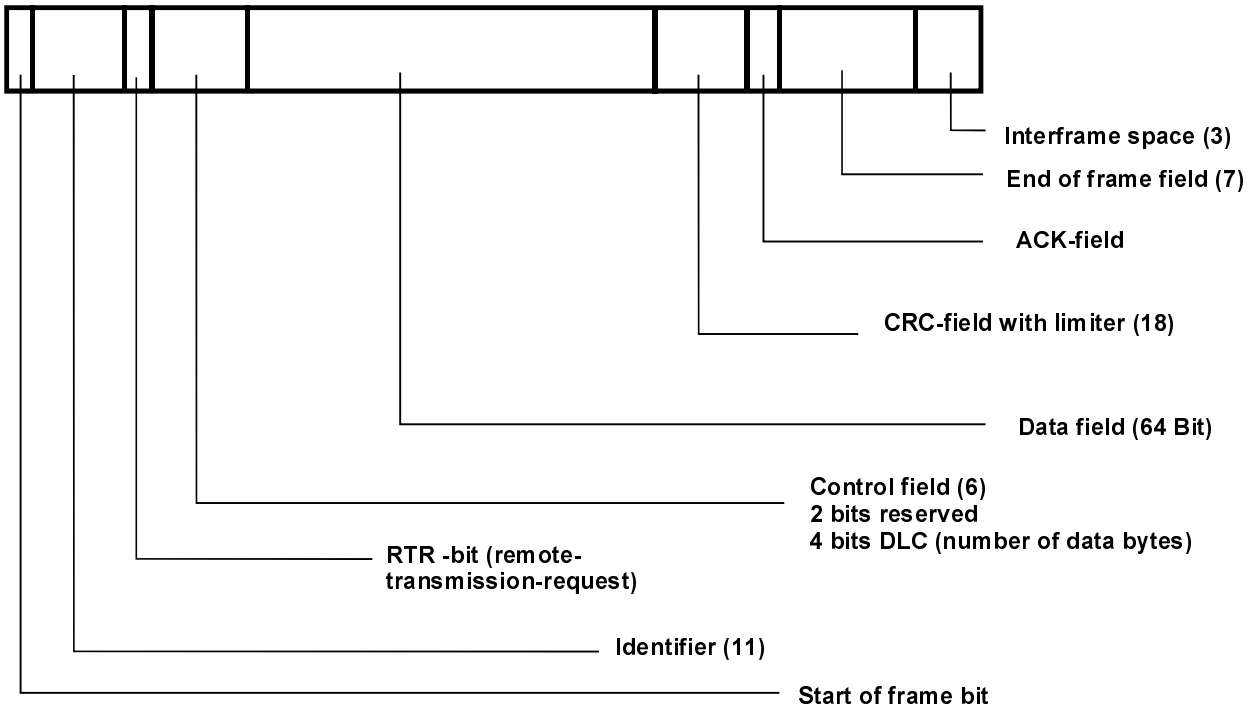
Under ISO-DIS 11898 (CAN high speed), each CAN station node must be able to deliver defined output level differences (between both transmission lines). For a dominant bit, 1.5 to 3.0 volts must be made available, and for a recessive bit -500 to 50 millivolts potential difference. Bus coupling can occur either through dicretely assembled or integrated transceivers. The latter, in addition to bus coupling, also offer further functions such as protection against thermal stress or a special standby mode.



12.1.2 Telegram construction

The actual message must be packed into a frame, so that the message can be evaluated by the correct receiver, bus access can be regulated and CRC-errors can be determined. In addition, the whole message is packed into start and stop identifiers. Differentiation is made between data, data requests, errors and overload telegrams.

The data telegram consists of different data fields which are explained in the following.



Start of Frame Bit
This bit carries the dominant character and is used as the first active signal after bus idle for synchronising all stations wanting access to the station which first started bus arbitration.

Identifier and RTR bit
This bit field is used to determine the priority of a message; in addition, the content of the message is determined (e.g. for the CAL layer) or the destination is indirectly determined (as receivers only receive messages with certain identifiers). The standard identifier format covers 11 bits, the extended format 29 bits.
The RTR bit (remote transmission request) included in the arbitration is used to show whether data is transmitted or requested.

Control field
This bit field states how many user data bytes (0 to 8) are used in the data field. Two additional bits are available for control purposes between standard and extended formats.

Data field
The user data are transmitted in the data field. It is possible to transmit a maximum of 8 bytes in one message frame. The user data themselves can be structured to be application-specific. COMPAX uses the first two bytes to identify message identifiers, message pointers and special command codes.

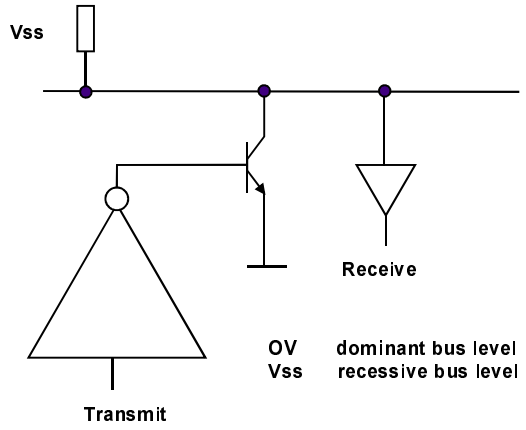
CRC-Field
A CRC check sum is set up by the transmitter over the message frame (up to and including the data field). Each passive station sets up an internal check sum on the data received and compares it to the CRC check sum received. If there are differences, then this will be reported dominantly in the acknowledge field.

Acknowledge Field
In this field, the receipt of CRC error-free messages is confirmed dominantly by all passive stations (the first bit is already reported on only one error free receipt). Not until the second bit is there the possibility of displaying the receipt of a wrong polynominal by individual stations, as this is transmitted recessively. Following this, an error telegram closes by requesting the message to be repeated.

Stop identifier field
The meaasge frame is closed using this sequence. Even before the end of the transmission of the final bit of the message frame, all passive stations can recognise that the transmission of the message was error free through the acknowledge field and if required accept it in the input buffer.

12.1.3 Bus access

Access to the transmission medium is carried out decentrally and uncoordinatedly, as there is neither a fixed bus coordinator nor a right of access in the form of a token. Technically real, parallel applications are independent with regard to their bus access, so that bus access conflicts can arise if several stations simultaneously wish to transmit data. To resolve this bus access a procedure is necessary which ensures that finally only one station has the right to write. Bus access is decided in this by means of a bit by bit comparison of the priority of the identifier (11 bits) (arbitration). A condition for this is that all stations interested in writing start to place their identifier on the bus at exactly the same time. Arbitration cannot be started until the bus is free (defined time of open-circuit potential).



Discrete circuit implementation for dominant and/or recessive bus level

12.1.4 Error recognition and handling

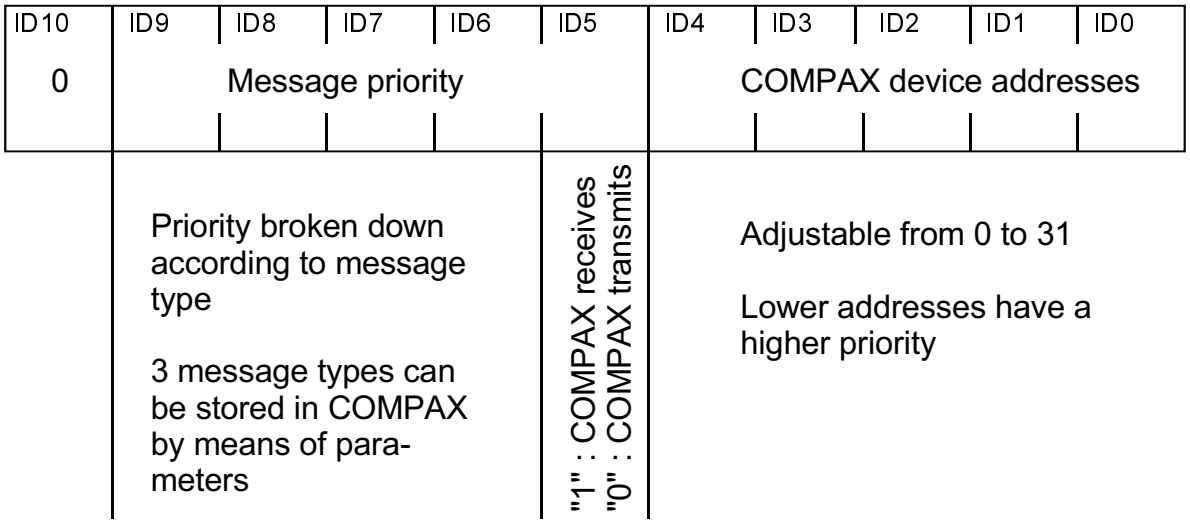
With the CAN-Bus, mechanisms for error recognition and handling are a part of the system concept.
Amongst other things, the rules on bit stuffing and the telegram format are monitored. The message frame is checked by a CRC sum and the transmitted level is monitored by its own receiver (no error messages during the arbitration phase). In addition, localisation procedures are made available by the CAN-Bus for independent isolation of defective

station modules, e.g. after a defined number of transmission errors in transmitting, the corresponding module can be switched off.

12.2 Identifier allocation using COMPAX

For communications to or from COMPAX, there is the following identifier construction.

Identifier distribution in COMPAX



The identifiers of the COMPAX controller can be configured using the parameters of the servo-controller. The parameterisable part covers the respective device addresses in the area 0 to 31 and three different message priorities. Identifier bit 5 determines whether the primary control (logical“ master) is sending data to COMPAX or is receiving it.

12.3 Message exchange using COMPAX

For each received message frame, COMPAX either sends the desired response or an Acknowledge“ or Not Acknowledge“ back with the error type. Conversely, on certain events such as fault“, position reached“ or comparator point reached“ COMPAX obviously transmits a message to the primary control. This type of message is called a spontaneous message.

The first two bytes of the data field are fixed in their use for determining the type of message (message recognition), for command coding and for further reduction.As with the Profibus or Interbus-S, the following is possible over the CAN-Bus: positioning (absolute, relative), the input of traversing speed and ramp values, the diagnosis of status values, the writing and reading of controller parameters and the setting, clearing and reading of digital outputs. The block memory of COMPAX can be both read and written line by line. In a special version of COMPAX, the cam disk“, reading and writing of the curve memory is possible. If commands cannot be transmitted in a message frame, the command is transmitted in up to three message frames. An example of this is the reading/writing of the block memory, as here in the first frame the line number in the block memory is addressed.