COMPAX field bus interface

Interbus-S

from COMPAX software version >V5.0
from Interbus-S – Software version >V2.1

January 2000

HAUSER
We automate motion

CERTIFIED DIN EN ISO 9001 QUALITY SYSTEM

Main 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

Subject to technical modification. Data based on the technical prior art at the time of printing.

Contents

Compatibility

Using the setting P196="0" makes this software version V2.1 compatible with software version V1.15 ... V2.0.
With P196="0", the process data channel is automatically assigned as follows:
Process input data channel PE with INPUT_WORD
Process output data channel PA with OUTPUT_WORD

Device allocation

This documentation applies for these devices:

- ♦ COMPAX 10XXSL with the Option F2 (available 04/2000)
- ♦ COMPAX 25XXS with the Option F2
- ♦ COMPAX 45XXS with the Option F2
- ♦ COMPAX 85XXS with the Option F2
- ♦ COMPAX P1XXM with the Option F2
- ♦ COMPAX 02XXM with the Option F2
- ♦ COMPAX 05XXM with the Option F2
- ♦ COMPAX 15XXM with the Option F2
- ♦ OMPAX 35XXM with the Option F2

XX: any characters
F2: Interbus-S - Option

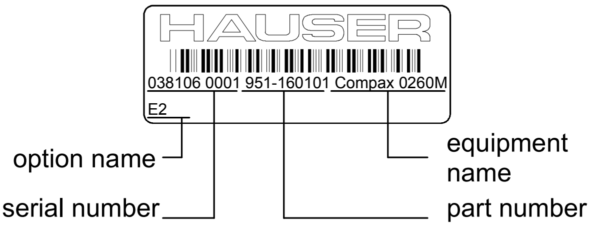
Key to unit designation

e.g.: COMPAX 0260M:

COMPAX:	name		
02:	performance class		
60:	Variant	e.g. "00":	Standard model
M:	Type of device	M:	multi-axis unit
		S:	single-axis unit

HAUSER rating plate

The rating plate is found on the top of the unit and contains the following information:



1. Introduction

The communication module Option F2 (IFM9) allows COMPAX-M/S to contact the Interbus-S by the company Phoenix. The Interbus-S protocol is based on a summation-frame message and is optimised for the cyclical and time equi-distant transmission of process data. The summation-frame message contains the information for or from all participants and is simultaneously sent to or received from all devices. The integration of parameter information in the cyclical protocol occurs sequentially, i.e. longer data blocks are broken down into individual, short information units and inserted in order into the cyclical protocol.

1.1 Addressing

Addressing individual participants occurs via their physical position in the ring and via central address lists held in the Master. This removes the need for manual participant addressing.

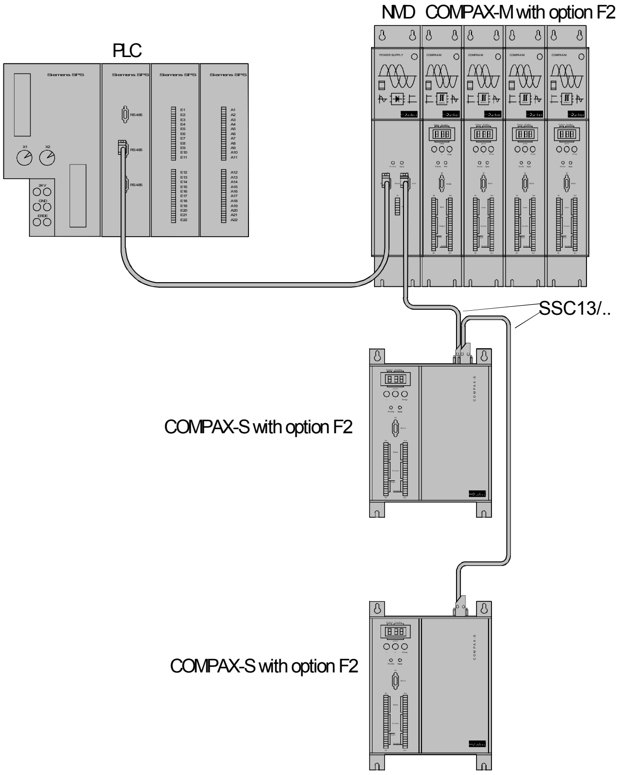
1.2 Bus connection

Each COMPAX-M/S with IFM-9 is a 2-cable remote bus participant.
Connection to the Interbus-S occurs via the 2-cable remote bus connection on the network module or on the COMPAX-S.

Mains power module

The X6 connection (Interbus-S IN = incoming remote bus) is connected via the pre-assembled ribbon cable with Option F2 of the 1. COMPAX-M/S which is connected to the network module. If additional COMPAX-M/S are connected to the network module, the Interbus-S is set up via ribbon cables from one device to the next.
The X7 connection (Interbus-S OUT = outgoing remote bus) is connected with the IFM-9 of the last COMPAX-M which is connected to the network module, if the ribbon cable short circuit plug is inserted in this device.

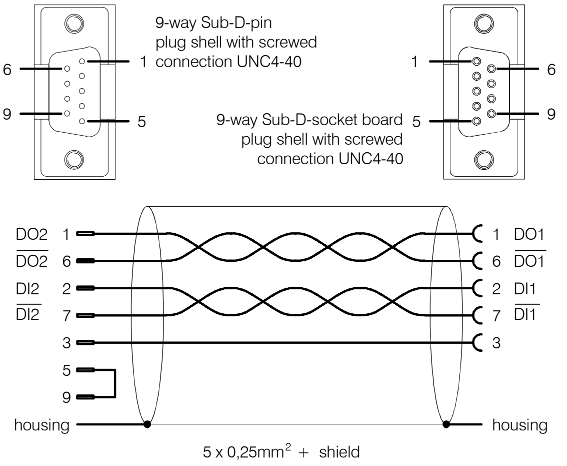
1.2.1 The bus wiring



The Interbus-S- signals are transferred along the existing ribbon cable within a COMPAX-M system network and a network module.

Cable plan SSK13/ :

SSC 13/..



Connections SSK13/ :

The assignment on the network module or on the COMPAX-S corresponds to the Interbus-S - standard.

Possible connections with SSK13/..:

- Network module X7 (OUT) → Network module X6 (IN)
- Network module X7 (OUT) → COMPAX-S X5 (IN)
- COMPAX-S X7 (OUT) → Network module X6 (IN)
- COMPAX-S X7 (OUT) → COMPAX-S X5 (IN)
- IIPC / SPS → COMPAX-S X5 (IN)
- IIPC / SPS → Network module X6 (IN)

If there is a direct power supply by COMPAX-M, the connection is made to the component block EAM5/01 instead of to the network module.

1.5 PDU Length

64 Byte
Maximum message length through the PCP channel.

1.6 Communication

Option F2 includes the Interbus-S communications software PCP 1.5 (Peripherals Communication Protocol), which allows it to transmit, in addition to cyclical process data, acyclic parameters (2 Byte data channel).
The following services are available:

Context Management

- Initiate Start communication connection
- Abort Quit communication connection
- Reject Reject a service

VFD Support

- Status Reading the device and user status
- Identify Reading the manufacturer, type and version

Variables Access

- Read Reading a variable (parameter)
- Write Writing a variable

1.3 Identification code (ID-Code)

When P196 Bit 0...2 < "4"

227	(0xE3)

When P196 Bit 0...2 = "4"

(Operation without acyclic channel = without PCP-communication)

3	(0x03)

1.4 Selectable Data Channels

The length of the process data depends on the Interbus-S software version and the COMPAX Parameter P196 (see also page 81).
The described features are not only valid for the process output data (PAD: data for COMPAX) but also for the process input data (PED: data from COMPAX

Process output data PAD: Master ⇒ COMPAX
Process input data PED: Master ⇐ COMPAX

Software-Version	P196	acyclic data channel	Process data length cyclic data channel	Data channel structure (split into bytes))							
less than 2.00	x	2 Byte PCP communication	2 bytes (1 word)	PCP	PCP	PD1	PD2				
From 2.00	0/1	2 Byte PCP communication	2 bytes (1 word)	PCP	PCP	PD1	PD2				
From 2.00	2	2 Byte PCP communication	4 byte (2 words)	PCP	PCP	PD1	PD2	PD3	PD4		
From 2.00	3	2 Byte PCP communication	6 byte (3 words)	PCP	PCP	PD1	PD2	PD3	PD4	PD5	PD6
From 2.10	Bit 0...2 ="1"	no acyclic data channel (no PCP communication)	8 byte (4 words)	PD1	PD2	PD3	PD4	PD5	PD6	PD7	PD8

PCP: **acyclic data channel:** is used for e.g. configuration.
PD: **cyclic process data:** is used e.g. for transferring actual values.

Acyclic data channel (PCP) for parameter data

This data channel will be described via your Interbus-S software. All objects can be transferred through this channel. The Interbus-S software splits longer objects which are then transferred in several cycles.

Cyclic process data

Only approved objects can be used on the cyclic process data channel. Whether an object has been approved for a PAD or PED channel is outlined in the respective objective descriptions under the heading PD “depictions” or in the object overview (page 9 onwards) in the column headed PD.

Assignment of PED and PAD:

There are various ways of loading objects onto the process data channels.

1. Via the COMPAX parameter P135 - P142
   These parameters are only accepted by COMPAX after Power off/on.
   Setting the process data channels with the COMPAX parameters corresponds to the settings with PED_INI and PAD_INI.
2. Via the objects PE_SELECT and PA_SELECT
   These objects allow the PD assignments to be set and/or changed during operation.

Setting: 8 Byte Process data channel (P196 Bit 0...2 ="4")

With P196 Bit 0...2 = "4", the acyclic data channel (PCP) for parameter data is turned off, making a 8 byte-wide process data channel available.
The process data channel is split into two parts, one of which has two possible assignments, PD1 and PD2, and another part which can be freely configured for assigning PD3 to PD8.

Assigning PD1 and PD2 with P196 Bit 7

P196 Bit 7="0"
PAD1-2 = STEUERWORT
PED1-2 = STATUSWORT
P196 bit 7 ="1"
PAD1-2 = CPX_STW
PED1-2 = CPX_ZSW

Assigning PD3 to PD8

The assignment is conducted as described above under the heading " Assignment of PED and PAD", but taking into consideration that the number of bytes to be assigned have all risen by 2.
In addition to this mode of operation (P196 Bit 0...2 ="4"), 2 new objects (for more details see page 64 onwards) are available, which may be assigned to the process data:

- ♦OBJECT_REQ: For PAD for transferral of any object (max. length of data 4 bytes) to COMPAX.
- ♦OBJECT_RSP: For PED as an acknowledgement for write access through OBJECT_REQ or as an answer to read access through OBJECT_REQ.

If you assign OBJECT_REQ to PAD and OBJECT_RSP to PED you will have access to all objects with a data length ≤ 4 bytes.
In addition to this fixed PD assignment, a new function which allows temporary assignment of the PD was created as a further possibility:

P196 Bit 5="0" OBJECT_REQ and OBJECT_RSP **cannot** be temporarily assigned to the PD (this allows a fixed assignment of the PD; as outlined above).

P196 Bit 5="1" OBJECT_REQ and OBJECT_RSP can be temporarily assigned to PD.
In addition to a permanent assignment of the process data (PD3 ... PD8: set for example with COMPAX parameters P135-P142) with cyclic values (e.g. target position value and actual position value), the OBJECT_REQ and OBJECT_RSP objects can be temporarily assigned to the PD (e.g. in order to change or read parameters) to be then re-switched to the set cyclic values (see page 64 onwards).

The answer/acknowledgement with OBJECT_RSP can also be turned off by P196 Bit 6. Meaning:

P196 Bit 6="0" OBJECT_RSP automatically assigns the PED as an answer/acknowledgement for OBJECT_REQ.

<table>
<tr><th colspan="3">Overview</th></tr>
<tr><td colspan="3"><b>P196 Bit 0...2 = "4": 8 byte process data channel</b></td></tr>
<tr><td colspan="3"><b>P196 Bit 7="0":</b> PD1/PD2: STEUERWORT/STATUSWORT
<b>P196 Bit 7="1":</b> PD1/PD2: CPX_STW/CPX_ZSW</td></tr>
<tr><td colspan="3">New objects: <b>OBJECT_REQ, OBJECT_RSP</b></td></tr>
<tr><td colspan="3"><b>P196 Bit 5="0":</b> temporary assignment of the PD with OBJECT_REQ and OBJECT_RSP <b>turned off</b></td></tr>
<tr><td colspan="3"><b>P196 Bit 5 ="1":</b> temporary assignment of the PD with OBJECT_REQ and OBJECT_RSP <b>turned on</b></td></tr>
<tr><td></td><td colspan="2"><b>P196 Bit 6 ="0":</b> OBJECT_RSP automatically <b>assigns</b> the PED as answer/acknowledgement for OBJECT_REQ</td></tr>
<tr><td></td><td colspan="2"><b>P196 Bit 6="1":</b> OBJECT_RSP does <b>not</b> automatically <b>assigns</b> the PED (as answer/acknowledgement for OBJECT_REQ).</td></tr>
<tr><td></td><td></td><td>OBJECT_RSP through <b>CONTROL="20" enabled</b> to PED or <b>disabled</b> through <b>CONTROL="21"</b></td></tr>
</table>

P196 Bit 1="1" OBJECT_RSP is **not** automatically assigned to the PED (as an answer/acknowledgement for OBJECT_REQ). There is however the possibility of enabling or disabling OBJECT_RSP for the PED temporarily if required, via the CONTROL object(command number 20 and 21).

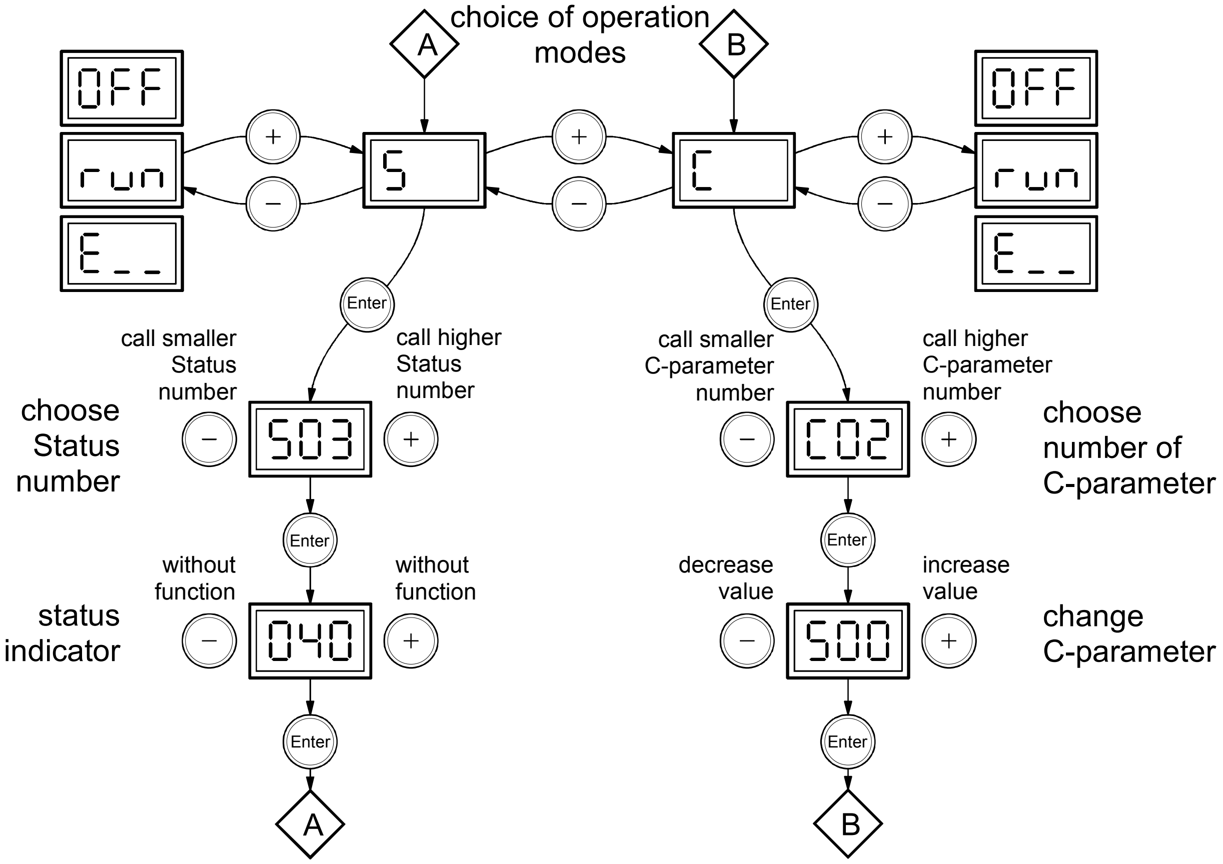
Controlling the temporary PD assignment

2 control bits (in STEUERWORT and CPX_STW: "ObjectReqEnable" and "FreezePAD") and a state bit (in STATUSTWORT and CPX_ZSW: "AckToggle") have been introduced to control the temporary PD assignment.

⇨ For more detailed explanations on the objects OBJECT_REQ and OBJECT_RSP, please see page 64.

1.4.1 Bus – Setting using the front cover

The bus protocol (P196) can be set using the COMPAX front cover. Procedure:


choice of operation modes
A
B
OFF
run
E_ _
S
C
+
−
Enter
call smaller Status number
call higher Status number
choose Status number
S03
without function
status indicator
040
call smaller C-parameter number
call higher C-parameter number
choose number of C-parameter
C02
decrease value
increase value
change C-parameter
500

Meaning:

C parameters	Meaning	Range	COMPAX parameters	Active on
C01	Address of unit	Automatically set!		
C02	Baud rate:	Automatically set!		
C03	Bus protocol	0...255	P196	**Power on!**
C04 - C11	reserved			

1.7 Data types

Physically, the types consist of one or more octets (Bytes). One byte consists of 8 bits (Bit 0 to 7). Bit 0 is the LSB (Least Significant Bit). A byte can also be depicted hexadecimally (0x00 ... 0xff).
If a data type consists of n byte, the following applies:

Data byte 1 (Byte in address x)	=	highest value byte
Data byte n (Byte in address x+n-1)	=	lowest value byte

The data coding in this chapter and the value ranges for the respective data types apply, unless otherwise explicitly stated in the data description of a COMPAX communication object.

1.7.1 Boolean

Depiction of the values TRUE and FALSE in a Byte (Octet).

Value range	TRUE or FALSE	Length	1 Byte
Coding	FALSE is depicted by the value 0x00, TRUE by the value 0xff		

1.7.2 Integer

Integer values are signed quantities.

Type	Value range	Length
Integer8	-128 ... +127	1 Byte
Integer16	-32 768 ... +32 767	2 Byte
Integer32	-2 147 483 647 ... +2 147 483 647 -2 147 483 648 (0x80 00 00 00) ⇨ Overflow	4 Byte

Coding	Two’s complement

In objects with the data type Integer32 it is possible that its value lies outside the value range.
If this is the case, the corresponding object has the value -2 147 483 648 (0x80 00 00 00) ⇨ Overflow.
E.G. this is possible with the object LAGE_IST, as the actual travel area of COMPAX M/S lies between +/- 4 000 000.

1.7.3 Unsigned

Unsigned values are unsigned quantities.

Type	Value range	Length
Unsigned8	0...255	1 Byte
Unsigned16	0...65 535	2 Byte
Unsigned32	0...4 294 967 295	4 Byte

Coding	binary

2. Object index

2.1 Communication objects: Overview sorted by Symbol

Command	Symbol	Service	Index	Index pointer	Sub-index	Byte	PD	See page
Deceleration time	ACCEL_NEG	wr	0x5feb	37	0	2	-	48
Acceleration time	ACCEL_POS	wr	0x5fea	36	0	2	-	47
Operation mode display	BETR_ART_AZ	rd	0x6061	66	0	2	-	16
Operation mode selection code	BETRIEBSART	rd/wr	0x6060	65	0	2	-	16
Special commands for COMPAX XX70	CAM_CMD	wr	0x5fcc	6	1...6	4	-	39
Read and write curve memory	CAM_MEM	rd/wr	0x5fcb	5	0	3	-	62
Set and read curve memory pointer	CAM_MEM_P	rd/wr	0x5fca	4	0	2	-	62
Command input in ASCII format	COMMAND	rd/wr	0x5fe6	32	0	20	-	23
Control commands	CONTROL	wr	0x5fe9	35	0	1	A	22
CPX control word	CPX_STW	rd/wr	0x5fd2	12	0	2	A	18
CPX status word	CPX_ZSW	rd	0x5fd3	13	0	2	I	19
Max motor speed value	DREHZAHLMAX	rd/wr	0x6080	81	0	2	-	44
Function group description	FKT_GRUPPE	rd	0x600f	61	1...7	4	-	14
Traverse speed actual value	GESCHW_IST	rd	0x606c	70	0	4	-	46
Max speed value	GESCHW_MAX	rd/wr	0x607f	80	0	4	-	44
Reference run speed	GESCHW_REF	rd/wr	0x6099	88	0	4	-	46
Set and read pointer	GOTO	rd/wr	0x5ff2	44	0	1	-	60
Position limit value min-max	GRENZEN	rd/wr	0x607d	78	1...2	4	-	36
Process input data description	IN_SELECT	rd/wr	0x5ffd	55	0...2	3	-	77
Logic state of the 16 digit. inputs	INPUT_WORD	rd	0x5ff8	50	0	2	I	51
Position actual value	LAGE_IST	rd	0x6064	67	0	4	I	35
Target position default	POSITION_TARGET	rd/wr	0x607a	76	0	4	A	33
Torque actual value	MOMENT_IST	rd	0x6077	74	0	2	-	30
Torque max value	MOMENT_MAX	rd/wr	0x6072	71	0	2	-	29
Read/write program memory (ASCII)	N	rd/wr	0x5ff1	43	1 ... 250	32	-	54
Rated torque motor	NENNMOMENT	rd/wr	0x6076	73	0	2	-	29
Motor nominal current	NENNSTROM	rd/wr	0x6075	72	0	2	-	30
Read/write program memory (binary)	Nx	rd/wr	0x5fd5	15	1 ... 250	20	-	54
Write/read objects via process data channel	OBJECT_REQ	wr	0x5fc5	-	0	6	A	66
write/read acknowledgement/ answer obj. via PDK	OBJECT_RSP	rd	0x5fc6	-	0	6	I	66
Enable process output data	OUT_ENABLE	rd/wr	0x5fff	57	0	1	-	78
Process input data description	OUT_SELECT	rd/wr	0x5ffe	56	0...2	3	-	77
Set or reset a digital output	OUTPUT	wr	0x5ff6	48	1...16	1	-	51
Mask outputs	OUTPUT_MASK	rd/wr	0x5ff5	47	0	2	-	52
Logic state of the 16 digit. Outputs	OUTPUT_WORD	rd/wr	0x5ff7	49	0	2	I/O	51
Reduce traverse speed	OVERRIDE	rd/wr	0x5fee	40	0	1	A	41
Read/write parameters in ASCII format	P	rd/wr	0x5fe7	33	1 ... 250	32	-	23
Enable PA data	PA_ENABLE	rd/wr	0x6002	60	0	1	-	74
PA data description	PA_SELECT	rd/wr	0x6001	59	0...13	19	-	73
Initialise the PA data description	PAD_INI	rd/wr	0x5fd1	11	1...4	3	-	76
PE data description	PE_SELECT	rd/wr	0x6000	58	0...13	19	-	72
Initialise the PE data description	PED_INI	rd/wr	0x5fd0	10	1...4	3	-	75
Polarities	POLARITAET	rd/wr	0x607e	79	0	1	-	38
Positioning window	POS_FENSTER	rd/wr	0x6067	69	0	4	-	37
Absolute positioning	POSA	wr	0x5ff0	42	0	6	-	32
Relative positioning	POSR	wr	0x5fef	41	0	6	-	33
Change traverse speed	POSR0SPEED	wr	0x5fed	39	0	3	-	42
Comparator function (active low)	POSROUTPUTN	wr	0x5fe3	29	1...16	4	-	53
Comparator function (active high)	POSROUTPUTP	wr	0x5fe2	28	1...16	4	-	53
Speed step profile	POSRXSPEEDY	wr	0x5fe4	30	0	8	-	42
Speed step profile with ACCEL	PRXSDYALZ	wr	0x5fc7	1	0	10	-	43
Read/write parameters in binary format	Px	rd/wr	0x5fd4	14	1 ... 250	4	-	24

Command	Symbol	Service	Index	Index pointer	Sub-index	Byte	PD	See page
Lag	RAMPE_NEG	rd/wr	0x6084	84	0	4	-	49
Rapid stop	RAMPE_NOTS	rd/wr	0x6085	85	0	4	-	49
Acceleration	RAMPE_POS	rd/wr	0x6083	83	0	4	-	48
Accelerate reference run	RAMPE_REF	rd/wr	0x609a	89	0	4	-	50
Ramp form speed	RAMPENFORM	rd/wr	0x6086	86	0	2	-	47
Reference measurement offset	REALNULL	rd/wr	0x607c	77	0	4	-	36
Reference run selection code	REF_MODE	rd/wr	0x6098	87	0	2	-	38
Status query in ASCII format	S	rd	0x5fe8	34	1...39	32	-	28
Actual position	S1	rd	0x5ff9	51	0	6	-	26
COMPAX operating hours	S10	rd	0x5fdb	21	0	4	-	26
Loop counter for a running REPEAT loop	S11	rd	0x5fda	20	0	2	-	27
Position of the absolute value sensor	S12	rd	0x5fd9	19	0	4	-	35
Target position	S2	rd	0x5fdf	25	0	4	-	27
Diagnosis values	S23_S26	rd	0x5fd8	18	1...4	2	-	27
Contour error	S3	rd	0x5ffa	52	0	2	I	15
Error message	S30	rd	0x5fd7	17	1...2	1	-	26
Device identification	S31_S39	rd	0x5fd6	16	1...9	3	-	15
Current axis process speed	S4	rd	0x5ffc	54	0	2	I	45
Current motor torque	S5	rd	0x5ffb	53	0	2	I	30
Temperature of the power output stage	S6	rd	0x5fde	24	0	2	-	26
Control voltage and intermediate circuit voltage	S7_S8	rd	0x5fdd	23	1...2	2	-	31
Number of axis motion cycles	S9	rd	0x5fdc	22	0	4	-	27
Contour error window	SCHLEPP_FEN	rd/wr	0x6065	68	0	4	-	37
Traverse speed	SPEED	rd/wr	0x5fec	38	0	2	A	41
Run program record N	START_N	wr	0x5ff4	46	0	1	A	60
Start program from record N	START_N_GO	wr	0x5fe5	31	0	1	A	61
Status byte	STATUSBYTE	rd	0x5fcf	9	0	1	I	17
Status word	STATUSWORD	rd	0x6041	64	0	2	I	20
Control byte	STEUERBYTE	rd/wr	0x5fce	8	0	1	A	17
Control word	CONTROLWORD	rd/wr	0x6040	63	0	2	A	19
Fault code	STOERUNG	rd	0x603f	62	0	2	-	25
Transfer current position in record N	TEACH_N	wr	0x5ff3	45	0	1	-	61
Traverse speed	VERF_GESCHW	rd/wr	0x6081	82	0	4	-	45
COMPAX – enter or read variables	VX	rd/wr	0x5fcd	7	40	4	-	62
Synchronisation with automatic reverse travel	WAITPOSA	wr	0x5fe0	26	0	4	-	39
Synchronisation without automatic reverse travel	WAITPOSR	wr	0x5fe1	27	0	4	-	40
Intermediate circuit voltage	ZWK_SPG	rd	0x6079	75	0	2	-	31

Column PD shows whether the respective object can be depicted on the process data channel. The following are explanations of the abbreviations in column PD:

- E Object can be depicted in the process input data.
- A Object can be depicted in the process output data.
- E/A Object can be depicted in the process input and output data.
- - not possible to depict on the process data channel.

2.2 Communication objects: Overview sorted by Index

Command	Symbol	Service	Index	Index Index	Sub-index	Byte	PD	See page
Write/read objects via process data channel	OBJECT_REQ	wr	0x5fc5	-	0	6	A	66
Write/read acknowledgement/ answer obj. via PDK	OBJECT_RSP	rd	0x5fc6	-	0	6	I	66
Speed step profile with ACCEL	PRXSDYALZ	wr	0x5fc7	1	0	10	-	43
Set and read curve memory pointer	CAM_MEM_P	rd/wr	0x5fca	4	0	2	-	62
Read and write curve memory	CAM_MEM	rd/wr	0x5fcb	5	0	3	-	62
Special commands for COMPAX XX70	CAM_CMD	wr	0x5fcc	6	1...6	4	-	39
COMPAX – Input or read variables	VX	rd/wr	0x5fcd	7	40	4	-	62
Control byte	STEUERBYTE	rd/wr	0x5fce	8	0	1	A	17
Status byte	STATUSBYTE	rd	0x5fcf	9	0	1	I	17
Initialise the PE data description	PED_INI	rd/wr	0x5fd0	10	1...4	3	-	75
Initialise the PA data description	PAD_INI	rd/wr	0x5fd1	11	1...4	3	-	76
CPX control word	CPX_STW	rd/wr	0x5fd2	12	0	2	A	18
CPX status word	CPX_ZSW	rd	0x5fd3	13	0	2	I	19
Read/write parameter in binary format	Px	rd/wr	0x5fd4	14	1 ... 250	4	-	24
Read/write program memory (binary)	Nx	rd/wr	0x5fd5	15	1 ... 250	20	-	54
Device identification	S31_S39	rd	0x5fd6	16	1...9	3	-	15
Error message	S30	rd	0x5fd7	17	1...2	1	-	26
Diagnosis values	S23_S26	rd	0x5fd8	18	1...4	2	-	27
Position of the absolute value sensor	S12	rd	0x5fd9	19	0	4	-	35
Loop counter for a running REPEAT loop	S11	rd	0x5fda	20	0	2	-	27
COMPAX operating hours	S10	rd	0x5fdb	21	0	4	-	26
Number of axis motion cycles	S9	rd	0x5fdc	22	0	4	-	27
Control voltage and intermediate circuit voltage	S7_S8	rd	0x5fdd	23	1...2	2	-	31
Temperature of power output stage	S6	rd	0x5fde	24	0	2	-	26
Target position	S2	rd	0x5fdf	25	0	4	-	27
Synchronisation with automatic reverse travel	WAITPOSA	wr	0x5fe0	26	0	4	-	39
Synchronisation without automatic reverse travel	WAITPOSR	wr	0x5fe1	27	0	4	-	40
Comparator function (active high)	POSROUTPUTP	wr	0x5fe2	28	1...16	4	-	53
Comparator function (active low)	POSROUTPUTN	wr	0x5fe3	29	1...16	4	-	53
Speed step profile	POSRXSPEEDY	wr	0x5fe4	30	0	8	-	42
Start program from record N	START_N_GO	wr	0x5fe5	31	0	1	A	61
Command input in ASCII format	COMMAND	rd/wr	0x5fe6	32	0	20	-	23
Read/write parameters in ASCII format	P	rd/wr	0x5fe7	33	1 ... 250	32	-	23
Status query in ASCII format	S	rd	0x5fe8	34	1...39	32	-	28
Control commands	CONTROL	wr	0x5fe9	35	0	1	A	22
Acceleration time	ACCEL_POS	wr	0x5fea	36	0	2	-	47
Deceleration time	ACCEL_NEG	wr	0x5feb	37	0	2	-	48
Traverse speed	SPEED	rd/wr	0x5fec	38	0	2	A	41
Change traverse speed	POSR0SPEED	wr	0x5fed	39	0	3	-	42
Reduce traverse speed	OVERRIDE	rd/wr	0x5fee	40	0	1	A	41
Relative positioning	POSR	wr	0x5fef	41	0	6	-	33
Absolute positioning	POSA	wr	0x5ff0	42	0	6	-	32
Read/write program memory (ASCII)	N	rd/wr	0x5ff1	43	1 ... 250	32	-	54
Set and read record pointer	GOTO	rd/wr	0x5ff2	44	0	1	-	60
Transfer current position in record N	TEACH_N	wr	0x5ff3	45	0	1	-	61
Run program record N	START_N	wr	0x5ff4	46	0	1	A	60
Mask outputs	OUTPUT_MASK	rd/wr	0x5ff5	47	0	2	-	52
Set or reset a digital output	OUTPUT	wr	0x5ff6	48	1...16	1	-	51
Logic state of the 16 digit. outputs	OUTPUT_WORD	rd/wr	0x5ff7	49	0	2	I/O	51
Logic state of the 16 digit. inputs	INPUT_WORD	rd	0x5ff8	50	0	2	I	51
Actual position	S1	rd	0x5ff9	51	0	6	-	26
Contour error	S3	rd	0x5ffa	52	0	2	I	15
Current motor torque	S5	rd	0x5ffb	53	0	2	I	30
Current axis process speed	S4	rd	0x5ffc	54	0	2	I	45
Process input data description	IN_SELECT	rd/wr	0x5ffd	55	0...2	3	-	77
Process input data description	OUT_SELECT	rd/wr	0x5ffe	56	0...2	3	-	77

Command	Symbol	Service	Index	Index Index	Sub-index	Byte	PD	See page
Enable process output data	OUT_ENABLE	rd/wr	0x5fff	57	0	1	-	78
PE data description	PE_SELECT	rd/wr	0x6000	58	0...13	19	-	72
PA data description	PA_SELECT	rd/wr	0x6001	59	0...13	19	-	73
Enable PA data	PA_ENABLE	rd/wr	0x6002	60	0	1	-	74
Function group description	FKT_GRUPPE	rd	0x600f	61	1...7	4	-	14
Fault code	STOERUNG	rd	0x603f	62	0	2	-	25
Control word	CONTROLWORD	rd/wr	0x6040	63	0	2	A	19
Status word	STATUSWORD	rd	0x6041	64	0	2	I	20
Operating mode selection code	BETRIEBSART	rd/wr	0x6060	65	0	2	-	16
Operating mode display	BETR_ART_AZ	rd	0x6061	66	0	2	-	16
Position actual value	LAGE_IST	rd	0x6064	67	0	4	I	35
Contour error window	SCHLEPP_FEN	rd/wr	0x6065	68	0	4	-	37
Positioning window	POS_FENSTER	rd/wr	0x6067	69	0	4	-	37
Traverse speed actual value	GESCHW_IST	rd	0x606c	70	0	4	-	46
Torque max value	MOMENT_MAX	rd/wr	0x6072	71	0	2	-	29
Motor nominal current	NENNSTROM	rd/wr	0x6075	72	0	2	-	30
Rated torque motor	NENNMOMENT	rd/wr	0x6076	73	0	2	-	29
Torque actual value	MOMENT_IST	rd	0x6077	74	0	2	-	30
Intermediate circuit voltage	ZWK_SPG	rd	0x6079	75	0	2	-	31
Target position default	POSITION_TARGET	rd/wr	0x607a	76	0	4	A	33
Reference measurement offset	REALNULL	rd/wr	0x607c	77	0	4	-	36
Position limit value min-max	GRENZEN	rd/wr	0x607d	78	1...2	4	-	36
Polarities	POLARITAET	rd/wr	0x607e	79	0	1	-	38
Max speed	GESCHW_MAX	rd/wr	0x607f	80	0	4	-	44
Max motor speed value	DREHZAHLMAX	rd/wr	0x6080	81	0	2	-	44
Process speed	VERF_GESCHW	rd/wr	0x6081	82	0	4	-	45
Acceleration	RAMPE_POS	rd/wr	0x6083	83	0	4	-	48
Lag	RAMPE_NEG	rd/wr	0x6084	84	0	4	-	49
Rapid stop	RAMPE_NOTS	rd/wr	0x6085	85	0	4	-	49
Ramp form speed	RAMPENFORM	rd/wr	0x6086	86	0	2	-	47
Reference run selection code	REF_MODE	rd/wr	0x6098	87	0	2	-	38
Reference run speed	GESCHW_REF	rd/wr	0x6099	88	0	4	-	46
Reference run acceleration	RAMPE_REF	rd/wr	0x609a	89	0	4	-	50

Column PD shows whether the respective object can be depicted on the process data channel. The following are explanations of the abbreviations in column PD:

- E Object can be depicted in the process input data.
- A Object can be depicted in the process output data.
- E/A Object can be depicted in the process input and output data.
- - not possible to depict on the process data channel.

Objects with an index >0x6000 are COMPAX–specific objects!

2.3 Device identification

2.3.1 FKT_GRUPPE

Function group description
This parameter displays the function groups from which functions are integrated in the device. The parameter is a field with 4*n entries. Each entry consists of the profile group, the profile version, the function group identifier and the function version.

Object Description

Index	0x600f				
Symbol	FKT_GRUPPE	**Length**	4	**Access groups**	0
Object code	Array	**Elements**	7	**Password**	0
Data type	Octet String	**Access rights**	Read all	**PD Map**	Not possible

Data Description

Data byte	**Meaning**	**Data byte**	**Meaning**
1...2	Profile number	3...4	Function groups number

Profile number		
Data byte 1		**Data byte 2**
b15 b12	**b11 b4**	**b3 b0**
	Profile group	Profile version

Profile group	**Meaning**	**Profile group**	**Meaning**
0x00	No profile (manufacturer specific)	0x07	Encoder
0x01	Sensor/actuator	0x08	Process controller
0x02	DRIVECOM	0x09	Robot control
0x03	reserved	0x0A	Screw control
0x04	Interface components	0x0B	ISO valve terminal
0x05	reserved	0x0C	Weld control
0x06	reserved	0x0D	Operating/display device

Function groups No.	
Data byte 3	Data byte 4
b15 b8	b7 b0
Function group identification	Function version

Fn. group	**Meaning**	**Funct. group**	**Meaning**
0x01	Status equipment device control	0x06	Position functions
0x02	General functions	0x07	Torque functions
0x03	Operation mode functions	0x08	Actual value generator
0x04	Speed functions 1	0x09	Factor functions
0x05	Speed functions 2	0x0a	Program functions

Example

Read 3. function group description.

Service	Read request	**Param. counter**	3	**Sub-index**	3
Command Code	0x8081	**Index**	0x600f		

2.3.2 S31_S39

Device identification.

Object Description

Index	0x5fd6				
Symbol	S31-S39	**Length**	6	**Access groups**	0
Object code	Array	**Elements**	9	**Password**	0
Data type	Octet string	**Access rights**	Read all	**PD Map**	Not possible

Data Description

Coding	BCD	**Value range**	000000000000 ... 999999999999

Data byte [Bit]	Assignment	Data byte [Bit]	Assignment
1 [7,6,5,4]	10^{11}	4 [7,6,5,4]	10^{5}
1 [3,2,1,0]	10^{10}	4 [3,2,1,0]	10^{4}
2 [7,6,5,4]	10^{9}	5 [7,6,5,4]	10^{3}
2 [3,2,1,0]	10^{8}	5 [3,2,1,0]	10^{2}
3 [7,6,5,4]	10^{7}	6 [7,6,5,4]	10^{1}
3 [3,2,1,0]	10^{6}	6 [3,2,1,0]	10^{0}

Sub-index	Assignment	Sub-index	Assignment
1	Software version	6	Date, version of bus option
2	Software date	7	Device identification
3	Job number	8	Device family
4	Part number	9	Device
5	Version		

In Sub-index 2 and 6, the data bytes are assigned as follows:

Data byte	1	2	3	4	5	6
Assignment	Day	Month	Year	Version		Identifier

2.4 Control

2.4.1 Operating mode

Operation mode selection code.
Selection function that determines the operation mode.

Object Description

Index	0x6060				
Symbol	BETRIEBSART	**Length**	2	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Integer16	**Access rights**	Read/write all	**PD Map**	Not possible

Data Description

Selection code	Operation mode	Selection code	Operation mode
-2	Reset mode	1	Target position input
-1	Continuous mode	3	Speed input 2

Example

The operation mode "Position target input" must be set.

Service	Write request	**Index**	0x6060	**1. data byte**	0x00
Command Code	0x8082	**Sub-index**	0	**2. data byte**	0x01
Param. counter	4	**Length**	2		

2.4.2 BETR_ART_AZ

Operation mode display.
This parameter indicates the current operation mode.
The meaning of the displayed value corresponds to the operation mode selection code.

Object Description

Index	0x6061				
Symbol	BETR_ART_AZ	**Length**	2	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Integer16	**Access rights**	Read all	**PD Map**	Not possible

2.4.3 STEUERBYTE

Allows program start from record 1 - 15.
The record pointer is set to the corresponding program record.
The program can be started, stopped, and continued.

⇨ The machine zero is approached with the record selection = "0000" and "Start".
An error acknowledgement is possible.

Object Description

Index	0x5FCE				
Symbol	STEUERBYTE	**Length**	1	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Octet String	**Access rights**	Read/write all	**PD Map**	PAD

Data Description

Data byte [Bit]	Assignment	Meaning
1 [7]	Quit (not when P190=22)	Error acknowledge with a positive edge
1 [6]	Stop with ramp P10 without clearing	with a positive edge
1 [5]	Continue ("1") / new start ("0") with Start	Continue: Continue program New start: Prog. start at selected record
1 [4]	Start / Stop	↑ Start after the response defined in Bit 5 ↓ Stop
1 [3]	Record select (2^3)	**Note!** With the record selection = "0000" and "New start", the machine zero point is approached
1 [2]	Record select (2^2)	
1 [1]	Record select (2^3)	
1 [0]	Record select (2^0)	

⇨ The "Quit" command is not accepted for P190=22 (DRIVECOM profile 22) (Error message aknowledgement as outlined in the status chart on page 77).

2.4.4 STATUSBYTE

The status byte shows information about the status of the device as well as messages.

Object Description

Index	0x5FCF				
Symbol	STATUSBYTE	**Length**	1	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Octet String	**Access rights**	Read all	**PD Map**	PED

Data Description

Data byte [Bit]	Assignment
1 [7]	Machine zero point approached
1 [6]	Idle after stop
1 [5]	Programmed nominal position reached
1 [4]	Motor stalled
1 [3]	Lag error
1 [2]	Ready for start (see below)
1 [1]	Warning
1 [0]	Fault

Meaning of "Ready for Start"

"Ready for Start" is used to control the program and is set

- when the program is halted by a WAIT START instruction and is waiting for the START signal.
- after an interruption with STOP or BREAK and these signals no longer occur.
- after a rectified error.
- after power on.
- at the program end with the command END.

"Ready for Start" has no meaning when commands are input directly.

2.4.5 CPX_STW

Activation of device control commands and setting/resetting of virtual inputs E17...E32
Control of the COMPAX M/S via the CPX control word is only possible when the relevant bits in P221 are set.
The virtual inputs E17...E32 can be queried in the data memory.

Object Description

Index	0x5FD2				
Symbol	CPX_STW	**Length**	2	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Octet string	**Access rights**	Read/write all	**PD Map**	PAD

Data Description

Data byte [Bit]	Assignment	Data byte [Bit]	Assignment
1 [15...8]	E32...E25	2 [7..0]	E24...E17

Data byte [Bit]	Function without shift	Function with shift	Enable
1 [7..2]	none	None	P222/Bit 7 ... 0
1 [1]	**ObjectReqEnable**	**ObjectReqEnable**	see below
1 [0]	**FreezePAD**	**FreezePAD**	see below
2 [7..6]	none	None	P221/Bit 7 ... 6
2 [5]	STOP	BREAK	P221/Bit 5
2 [4]	START	None	P221/Bit 4
2 [3]	QUIT (not when P190=22)	Teach real zero	P221/Bit 3
2 [2]	Hand-	Approach real zero (RZ)	P221/Bit 2
2 [1]	Hand+	Approach machine zero (MZ)	P221/Bit 1
2 [0]	SHIFT		P221/Bit 0

The functions ***ObjectReqEnable*** and ***FreezePAD*** are only active in CPX_STW when
PD-Length = 4 and P196 Bit 7 = 1 and P196 Bit 5 = 1 !
For a detailed description, please see page 64

The "Quit" command is not accepted for P190=22 (DRIVECOM profile 22) (Error message aknowledgement as outlined in the status chart on page 77).

By partially reassigning the input functions to STEUERWORT, the function is limited by the multiple function E1. :
Example: If a function with E1 occupies the control word (e.g. Teach real null), then additional E1 functions (such as the "QUIT" function) are ignored by the inputs.

Therefore: If you need all the input functions, the function must be completely reassigned, either to the inputs (P221 = 0) or to the control word (P221 = 63).

Command recognition

The control word is cyclically transmitted from the Interbus Master via the bus.
Note! When a PLC is the Master, the control word may not be present for too short a time.
The PLC and Interbus-S cycles are asynchronous. If the control word is output for just one PLC cycle (scan), data may be lost.
Rectify the problem with:

- A control word which is available for a sufficiently long time

or

- by re-reading the object "CPX_STW". The command has been recognised if the change is in this object.

COMPAX - E/A - Functions over the control word (Data bits 1[1], 1[0], 2[7], ... 2[2])

COMPAX does not recognise a direct reassignment of the E/A functions by removing a function and setting another function simultaneously; **Exception:** STOP and BREAK (these are always recognised immediately).
Therefore proceed as follows:

- Remove the previous functions by sending a "null telegram" (allow status to remain until it has been recognised by the COMPAX).
- When the COMPAX is ready, set a new function.

Example: Switch from Hand+ to Hand-

- Reset Hand+: data bit 1[0] = "0"
- Wait until COMPAX has recognised through the Interbus-S (till "Ready to start" has been set) or Handshake through CPX_STW.
- Set Hand-: data bit 1[1] = "1"

2.4.6 CPX_ZSW

The CPX status word shows information about the status of the device as well as messages.
COMPAX Software Version 3.64 or above allows the status information S16 and S17 to be assigned to CPX_ZSW via parameter 203 Bit 0 = 1.

Object description

Index	0x5FD3				
Symbol	CPX_ZSW	**Length**	2	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Octet String	**Access rights**	Read all	**PD Map**	PED

Data description (P203 Bit 0 = 0)

Data byte [Bit]	Assignment	Data byte [Bit]	Assignment
1 [7]	Status A16	2 [7]	Motor stalled
1 [6]	Status A15	2 [6]	Lag error
1 [5]	Status A14	2 [5]	Idle after stop
1 [4]	Status A13	2 [4]	Target position reached
1 [3]	Status A12	2 [3]	Ready for start
1 [2]	Status A11	2 [2]	MN was reached
1 [1]	Status A10	2 [1]	No warning
1 [0]	Status A9/***AckToggle***	2 [0]	No fault

Data description (P203 Bit 0 = 1)

Data byte [Bit]	Assignment	Data byte [Bit]	Assignment
1 [7]	---	**2 [7] 2 [6]** 0 0	**OUTPUT A0 = x** after OUTPUT A0 = 0
1 [6]	RUN ("0"= off or turned off in the event of an error)	0 1 1 0	after OUTPUT A0 = 1 after OUTPUT A0 = 2
1 [5]	Reserved	2 [5]	Idle after stop
1 [4]	Stop over input E6	2 [4]	Target position reached
1 [3]	Program memory running	2 [3]	Ready for start
1 [2]	Command active	2 [2]	MN was reached
1 [1]	Service password active	2 [1]	No warning
1 [0]	Password 302 active/***AckToggle***	2 [0]	No fault

The status information ***AckToggle*** only appears in CPX_ZSW when **PD-Length = 4 and P196 Bit 7 = 1 and OBJECT_REQ** is on the PAD channel or can be temporarily assigned to the PAD channel. **(P196 Bit 5 = 1)**!
For a description please see page 64onwards.

2.4.7 STEUERWORT

Some bits in the control word influence the status equipment of the device control, enabling functions and determining the operating status of the devices.
The manufacturer-specific bits (Data byte 1 bits 2...6) serve to activate devices control commands, which are only effective if P221 is used to provide the appropriate enabling (see CPX_STW).
The SPM bit (Data byte 1 bit 7) is used to acknowledge an existing pop-up message.

Object Description

Index	0x6040				
Symbol	STEUERWORT	**Length**	2	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Octet String	**Access rights**	Read/write all	**PD Map**	PAD

Data Description

Data byte [Bit]	Assignment	Data byte [Bit]	Assignment
1 [7]	SPM	2 [7]	Reset fault
1 [6]	STOP	2 [6]	Target position relative or absolute
1 [5]	START	2 [5]	
1 [4]	Hand-	2 [4]	New set point
1 [3]	Hand+	2 [3]	Enable operation
1 [2]	Approach machine zero (MZ)	2 [2]	Rapid stop
1 [1]	***ObjectReqEnable***	2 [1]	Disable voltage
1 [0]	***FreezePAD***	2 [0]	Power on

💣 The functions ***ObjectReqEnable*** and ***FreezePAD*** are only active in STEUERWORT when
PD-Length = 4 and P196 Bit 7 = 0 and P196 Bit 5 = 1 !
For a description, please see page 64.onwards

2.4.8 STATUSWORT

The status word displays information regarding the status of the devices and messages,
when SPM (status word) is not equal to SPM (control word).
The status word displays pop-up messages, when SPM (status word) is not equal to SPM (control word).

Object Description

Index	0x6041				
Symbol	STATUSWORT	**Length**	2	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Octet String	**Access rights**	read all	**PD Map**	PED

Data Description

Data byte [Bit]	Assignment	Data byte [Bit]	Assignment
1 [7]	SPM	2 [7]	Warning
1 [6]	Idle after stop	2 [6]	Switch on disabled
1 [5]	MN was reached	2 [5]	Rapid stop
1 [4]	Actual value acknowledgement	2 [4]	Voltage disabled
1 [3]	Limit value	2 [3]	Fault
1 [2]	Set point reached	2 [2]	Operation enabled
1 [1]	Remote	2 [1]	Switched on
1 [0]	Message/***AckToggle***	2 [0]	Ready to start

💣 The status information ***AckToggle*** only appears in STATUSWORT when
PD-Length = 4 and P196 Bit 7 = 0 and OBJECT_REQ is on the PAD channel or can be assigned to the PAD channel temporarily. **(P196 Bit 5 = 1)**!
For a description, please see page 64onwards.

2.4.9 Pop-up message processing

COMPAX can display the following pop-up messages via the status word:
☞ an error has occurred.
☞ the programmed set point has been reached.
☞ the programmed comparator point has been reached.
The pop-up messages can be enabled individually via P193 (activated).

Pop-up message	Valency
automatic error message	1
automatic "position reached" message	2
automatic comparator switch points report	4

⇨ The required settings can be obtained by inputting the sum of the significants in P193

If the pop-up message processing is active and there is a pop-up message, COMPAX interrupts the normal status display in the status word, toggles the pop-up message flag "SPM" and displays the actual pop-up message in the status word.
The Master accepts the pop-up message and acknowledges it by toggling the "SPM" in the control word.

⇨ The status word displays a pop-up message, when SPM (status word) is not equal to SPM (control word).

Status word with pop-up messages

Data byte [Bit]	Assignment	Data byte [Bit]	Assignment
1 [7]	SPM	2 [7]	Error No. or Comparator No.
1 [6]	Pop-up message identifier 1 = error 2 = set point reached 3 = Comparator point reached	2 [6]	
1 [5]		2 [5]	
1 [4]		2 [4]	
1 [3]		2 [3]	
1 [2]		2 [2]	
1 [1]		2 [1]	
1 [0]		2 [0]	

2.4.10 CONTROL

Control commands. Compax commands which do not require additional values.
The required commands are activated by reading in the relevant command number.

Object Description

Index	0x5fe9				
Symbol	CONTROL	**Length**	1	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Unsigned8	**Access rights**	write all	**PD Map**	PAD

Data Description

Command no.	Function	Command No.	Function
1	Go to machine home	12	Drive dead with opened brake
2	Program start	13	Program jump via external inputs
3	Stop program/positioning	14	Deactivate password protection (GOTO 302)
4	Break off program/positioning	15	Activate password protection (GOTO 270)
5	Acknowledge error	16	Deactivate password protection (GOTO 620)
6	Read current position as real null	17	Declare curve valid
7	Declare valid	18	Not-Stop with clear
8	Declare configuration valid	19	Not-Stop without clear
9	Traverse speed from external encoder	20	OBJECT_RSP temporarily enabled on PED
10	Drive under torque with opened brake	21	OBJECT_RSP temporarily disabled on PED
11	Drive dead with closed brake	22-24	Not assigned

Example

The drive must approach the machine zero point.

Service	Write request	**Index**	0x5fe9	**Data byte**	0x01
Command Code	0x8082	**Sub-index**	0		
Param. counter	4	**Length**	1		

2.4.11 COMMAND

All COMPAX commands which exist for the RS232 interface can be transferred with this object in plain text (ASCII-String in upper case letters).

Object Description

Index	0x5fe6				
Symbol	COMMAND	**Length**	20	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Visible string	**Access rights**	Read/write all	**PD Map**	Not possible

Data Description

Coding	ASCII	**Value range**	0x20 ... 0x7f

Data byte	Assignment	Data byte	Assignment
1	1. character of the command string	20	20. character of the command string

Example

The drive must travel 15.8 mm relative (RS232 command: "POSR15.8").

Service	Write request	**Length**	20	**5. data byte**	0x31 ("1")
Command Code	0x8082	**1. data byte**	0x50 ("P")	**6. data byte**	0x35 ("5")
Param. counter	13	**2. data byte**	0x4f ("O")	**7. data byte**	0x2e (".")
Index	0x5fe6	**3. data byte**	0x53 ("S")	**8. data byte**	0x38 ("8")
Sub-index	0	**4. data byte**	0x52 ("R")	**9...20. data byte**	0x20 (" ")

2.4.12 P

Write/read COMPAX parameters in ASCII format.
The corresponding parameter is selected using the Sub-index (Sub-index = parameter No.).

Object Description

Index	0x5fe7				
Symbol	P	**Length**	32	**Access groups**	0
Object code	Array	**Elements**	250	**Password**	0
Data type	Visible-String	**Access rights**	read/write all	**PD Map**	not possible

Data Description

Data format	ASCII	**Value range**	0x20 ... 0x7f

Data byte	Assignment	Data byte	Assignment
1	1. character of the ASCII response	32	32. character of the ASCII response

Example

The COMPAX parameter 3 must have the value -1055.88. Command: P003=-1055.88

Service	Write request	**2. data byte**	0x30 ("0")	**9. data byte**	0x35 ("5")
Command Code	0x8082	**3. data byte**	0x30 ("0")	**10. data byte**	0x35 ("5")
Param. counter	19	**4. data byte**	0x33 ("3")	**11. data byte**	0x2e (".")
Index	0x5fe7	**5. data byte**	0x3D ("=")	**12. data byte**	0x38 ("8")
Sub-index	3	**6. data byte**	0x2d ("-")	**13. data byte**	0x38 ("8")
Length	32	**7. data byte**	0x31 ("1")	**14...32. data byte**	0x20 (" ")
1. data byte	0x50 ("P")	**8. data byte**	0x30 ("0")		

2.4.13 Px

Read/write COMPAX parameters in binary format.
The corresponding parameter is selected using the Sub-index (Sub-index = parameter No.).

Object Description

Index	0x5fd4				
Symbol	Px	**Length**	4	**Access groups**	0
Object code	Array	**Elements**	250	**Password**	0
Data type	Integer32	**Access rights**	Read/write all	**PD Map**	Not possible

Data Description

Parameter	Resolution	Parameter	Resolution
001 .. 005	1 ⇔ 0.001	035 .. 036	1 ⇔ 0.000001
006 .. 010	1	037 .. 049	1 ⇔ 0.001
011 .. 016	1 ⇔ 0.001	050 .. 072	1
017 .. 020	1	073 .. 099	1 ⇔ 0.001
021 .. 022	1 ⇔ 0.000001	100 .. 186	1
023 .. 029	1	187 .. 196	1 ⇔ 0.001
030 .. 034	1 ⇔ 0.001	197 .. 250	1

Example

Parameter 11 must be given the value 5000.

Service	Write request	**Sub-index**	11	**3. data byte**	0x4b
Command Code	0x8082	**Length**	4	**4. data byte**	0x40
Param. counter	5	**1. data byte**	0x00		
Index	0x5fd4	**2. data byte**	0x4c		

2.5 Diagnosis

2.5.1 STOERUNG

Fault code. If COMPAX is in the "Fault" status, the fault code is given a value not equal to 0.
If COMPAX is not in the "Fault" status, this object then gives the value 0.

Object Description

Index	0x603f				
Symbol	STOERUNG	**Length**	2	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Octet String	**Access rights**	Read all	**PD Map**	Not possible

Data Description

Data byte [Bit]	Coding	Data byte [Bit]	Coding
1 [7 ... 4]	Main groups	2 [7 ... 0]	Details
1 [3 ... 0]	Subgroups		

Fault codes:

Code	COMPAX error
0000h	E00 Stop/no error
6320h	E01 Not configured
6320h	E02 Limits
1000h	E03
1000h	E04
7300h	E05 MN-Ini not found
1000h	E06
8200h	E07 Processor error
8200h	E08 Synchronous stop is enabled
8200h	E09 Drive not working
8611h	E10 Contour error too large
8600h	E11 Prog. position not reached
7320h	E12 Slip error
7300h	E13 Ini 1 not damped
7300h	E14 Ini 2 not bed.
7320h	E15 Error in the 2. position measuring system
6300h	E16 Record number does not exist
6300h	E17 Record number too large
6300h	E18 Record 250 is assigned
6300h	E19 No room in memory
8612h	E20 Target position behind pos. limit stop
8612h	E21 Target position behind neg. limit stop
8500h	E22 MN is not approached
6200h	E23 Command is not allowed
8400h	E24 Speed is invalid
8600h	E25 Position is invalid
6300h	E26 END command missing for REPEAT
6320h	E27 Parameter cannot be written to
1000h	E28
6320h	E29 Motor values missing
5500h	E30 Hardware fault
6320h	E31 Parameter error
6320h	E32 Parameter error
6300h	E33 Data memory error
6300h	E34 Data memory error
5500h	E35 Hardware fault

Code	COMPAX error
5500h	E36 Hardware fault
5111h	E37 Auxiliary voltage +15V missing
5120h	E38 Voltage in the intermediate circuit too high
4210h	E39 Temperature too high
1000h	E40
5410h	E41 Output stage signals error
7303h	E42 Resolver error
2300h	E43 Output loaded
5111h	E44 Pos. auxiliary voltage outside tolerance
5111h	E45 neg. auxiliary voltage outside tolerance
5112h	E46 24V too large
5112h	E47 24V too small
4310h	E48 Thermostatic switch motor indicates error
7121h	E49 Motor/drive indicates disabling
8612h	E50 Limit switch 1 activated
8612h	E51 Limit switch 2 activated
7200h	E52 Error in emergency stop controlling
7120h	E53 Motor loaded
8400h	E54 Speed too high
8000h	E55 External emergency stop
8000h	E56 Emergency stop directly in COMPAX M
5120h	E57 Voltage in the intermediate circuit too low
4200h	E58 Temperature getting too high
1000h	E59
7200h	E60 Slip warning
1000h	E61
1000h	E62
1000h	E63
1000h	E64
7305h	E65 Encoder module not enabled
1000h	E66
1000h	E67
1000h	E68
1000h	E69
1000h	E70
1000h	E71...E255

2.5.2 S30

Error message.
This object contains the error number of the current error and the last occurring error.
If the error number of the current error = 0, there is no error.

Object Description

Index	0x5fd7				
Symbol	S30	**Length**	1	**Access groups**	0
Object code	Array	**Elements**	2	**Password**	0
Data type	Unsigned8	**Access rights**	Read all	**PD Map**	Not possible

Data Description

Sub-index	Assignment	Sub-index	Assignment
1	Error number of current error	2	Error number of last error

2.5.3 S6

Temperature of the power output stage.

Object Description

Index	0x5fde				
Symbol	S6	**Length**	2	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Unsigned16	**Access rights**	Read all	**PD Map**	Not possible

Data Description

Unit	degrees Celsius	**Resolution**	1 ⇔ 0.1°C

2.5.4 S10

COMPAX operating hours

Object Description

Index	0x5fdb				
Symbol	S10	**Length**	4	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Unsigned32	**Access rights**	read all	**PD Map**	Not possible

Data Description

Unit	Hours	**Resolution**	1 ⇔ 0.1 h

2.5.5 S9

Number of axis motion cycles.

Object Description

Index	0x5fdc				
Symbol	S9	**Length**	4	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Unsigned32	**Access rights**	read all	**PD Map**	not possible

2.5.6 S11

Loop counter for a running REPEAT loop.

Object Description

Index	0x5fda				
Symbol	S11	**Length**	2	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Unsigned16	**Access rights**	read all	**PD Map**	not possible

2.5.7 S23_S26

Status of the drive, the switch, the limits and the output stage.

Object Description

Index	0x5fd8				
Symbol	S23-S26	**Length**	2	**Access groups**	0
Object code	Array	**Elements**	4	**Password**	0
Data type	Octet String	**Access rights**	read all	**PD Map**	not possible

Data Description

Data byte [Bit]	Assignment	Data byte [Bit]	Assignment
1 [7 ... 0]	Status bits	2 [7 ... 0]	Status bits

Sub-index	Assignment	Sub-index	Assignment
1	Status drive	3	Status limits
2	Status switch	4	Status final step

⇨ For explanations of status bits, please see product manual COMPAX-M/S!

2.5.8 S

All Compax status values that exist for the RS232 interface can be read with this object. The response is made available in plain text (as ASCII string).
The corresponding status is selected using the Sub-index (Sub-index = status No.).

Object Description

Index	0x5fe8				
Symbol	S	**Length**	32	**Access groups**	0
Object code	Array	**Elements**	250	**Password**	0
Data type	Visible-String	**Access rights**	read all	**PD Map**	not possible

Data Description

Coding	ASCII	**Value range**	0x20 ... 0x7f

Data byte	Assignment	Data byte	Assignment
1	1. character of the response string	32	32. character of the response string

Example

Read S23.

Service	Read Request	**Param. counter**	3	**Sub-index**	23
Command Code	0x8081	**Index**	0x5fe8		

2.6 Torque, current, voltage

2.6.1 MOMENT_MAX

Max. torque value.
This value is the maximum permissible torque for the motor.

Object Description

Index	0x6072				
Symbol	MOMENT_MAX	**Length**	2	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Unsigned16	**Access rights**	read/write all	**PD Map**	not possible

Data Description

Unit	Per-thousands of motor rated torque	**Resolution**	1 ⇔ 1 per-thousand

Example

The max. torque of the motor should be 400 per-thousands of the motor rated torque.

Service	Write request	**Index**	0x6072	**1. data byte**	0x01
Command Code	0x8082	**Sub-index**	0	**2. data byte**	0x90
Param. counter	4	**Length**	2		

2.6.2 NENNMOMENT

Rated torque motor.
This value can be found on the rating plate of the motor.

Object Description

Index	0x6076				
Symbol	NENNMOMENT	**Length**	2	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Unsigned16	**Access rights**	Read/write all	**PD Map**	Not possible

Data Description

Unit	Nm	**Resolution**	1 ⇔ 0.1 Nm

Example

The rated torque of the motor HDX115C6-88S is 5.2 Nm.

Service	Write request	**Index**	0x6076	**1. data byte**	0x00
Command Code	0x8082	**Sub-index**	0	**2. data byte**	0x34
Param. counter	4	**Length**	2		

2.6.3 MOMENT_IST

Torque actual value.
The torque actual value corresponds to the current torque in the drive motor.

Object Description

Index	0x6077				
Symbol	MOMENT_IST	**Length**	2	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Integer16	**Access rights**	Read all	**PD Map**	Not possible

Data Description

Unit	Per-thousands of motor rated torque	**Resolution**	1 ⇔ 1 per-thousand

2.6.4 S5

Current motor torque.
Value in % of the rated torque.

Object Description

Index	0x5ffb				
Symbol	S5	**Length**	2	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Integer16	**Access rights**	Read all	**PD Map**	PED

Data Description

Unit	%	**Resolution**	1 ⇔ $^{1}/_{64}$%; (6400 ⇔ 100%)

2.6.5 NENNSTROM

Motor nominal current.
This value can be found on the rating plate of the motor.

Object Description

Index	0x6075				
Symbol	NENNSTROM	**Length**	2	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Unsigned16	**Access rights**	Read/write all	**PD Map**	not possible

Data Description

Unit	Ampere	**Resolution**	1 ⇔ 0.1 A

Example

The nominal current of the motor HDX115C6-88S is 5.1 Ampere.

Service	Write request	**Index**	0x6075	**1. data byte**	0x00
Command Code	0x8082	**Sub-index**	0	**2. data byte**	0x33
Param. counter	4	**Length**	2		

2.6.6 ZWK_SPG

Intermediate circuit voltage.
This parameter describes the current intermediate circuit voltage in the drive controller.

Object Description

Index	0x6079				
Symbol	ZWK_SPG	**Length**	2	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Unsigned16	**Access rights**	read all	**PD Map**	not possible

Data Description

Unit	Volt	**Resolution**	1 ⇔ 1 V

2.6.7 S7_S8

Control voltage and power or intermediate circuit voltage.

Object Description

Index	0x5fdd				
Symbol	S7-S8	**Length**	2	**Access groups**	0
Object code	Array	**Elements**	2	**Password**	0
Data type	Unsigned16	**Access rights**	read all	**PD Map**	not possible

Data Description

Unit	Volt	**Resolution**	1 ⇔ 0.1 V

Sub-index	Assignment	Sub-index	Assignment
1	Control voltage	2	intermediate circuit voltage

Example

Read current intermediate circuit voltage.

Service	Read Request	**Param. counter**	3	**Sub-index**	2
Command Code	0x8081	**Index**	0x5fdd	**Length**	0

2.7 Positioning

2.7.1 POSA

Absolute positioning. Reference point is real zero (RN).
Positioning is done with the acceleration time (brake time) set by ACCELL-POS (ACCEL-NEG) and the velocity set by SPEED.
If these values were not set, then valid are **substitute values:** SPEED: Parameter P002; ACCEL: Parameter P006

Object Description

Index	0x5ff0				
Symbol	POSA	**Length**	6	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Octet String	**Access rights**	Write all	**PD Map**	Not possible

Data Description

Data format	BCD	**Unit**	mm (or inch)
Value range	-4 000 000 000 ... +4 000 000 000	**Resolution**	1 ⇔ 0.001 mm (or inch)

Data byte [Bit]	Assignment	Data byte [Bit]	Assignment
1 [7 ..... 0]	Value sign (0x00 ⇔ **+**; 0xff ⇔ **-**)		
2 [7,6,5,4]	10^9	4 [3,2,1,0]	10^4
2 [3,2,1,0]	10^8	5 [7,6,5,4]	10^3
3 [7,6,5,4]	10^7	5 [3,2,1,0]	10^2
3 [3,2,1,0]	10^6	6 [7,6,5,4]	10^1
4 [7,6,5,4]	10^5	6 [3,2,1,0]	10^0

Example

The drive must travel to the absolute position 7350.150 mm.

Service	Write request	**Sub-index**	0	**3. data byte**	0x07
Command Code	0x8082	**Length**	6	**4. data byte**	0x35
Param. counter	6	**1. data byte**	0x00	**5. data byte**	0x01
Index	0x5ff0	**2. data byte**	0x00	**6. data byte**	0x50

2.7.2 POSR

Relative positioning. Reference point is the current position.

Object Description

Index	0x5fef				
Symbol	POSR	**Length**	6	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Octet String	**Access rights**	Write all	**PD Map**	Not possible

Data Description

Data format	BCD	**Unit**	mm (or inch)
Value range	-4 000 000 000 ... +4 000 000 000	**Resolution**	1 ⇔ 0.001 mm (or inch)

Data byte [Bit]	Assignment	Data byte [Bit]	Assignment
1 [7 ..... 0]	Value sign (0x00 ⇔ **+**; 0xff ⇔ **-**)		
2 [7,6,5,4]	10^9	4 [3,2,1,0]	10^4
2 [3,2,1,0]	10^8	5 [7,6,5,4]	10^3
3 [7,6,5,4]	10^7	5 [3,2,1,0]	10^2
3 [3,2,1,0]	10^6	6 [7,6,5,4]	10^1
4 [7,6,5,4]	10^5	6 [3,2,1,0]	10^0

Example

The drive should travel relative to -37891.210 mm.

Service	Write request	**Sub-index**	0	**3. data byte**	0x37
Command Code	0x8082	**Length**	6	**4. data byte**	0x89
Param. counter	6	**1. data byte**	0xff	**5. data byte**	0x12
Index	0x5fef	**2. data byte**	0x00	**6. data byte**	0x10

2.7.3 LAGE_ZIEL

Target position default.
Absolute positioning. Reference point is the real null (RN) or relative positioning: set by the data byte 2 Bit 6 (0 = absolute; 1 = relative).
Positioning is done with the acceleration time (brake time) set by ACCELL-POS (ACCEL-NEG) and the velocity set by SPEED.
If these values were not set, then valid are **substitute values:** SPEED: Parameter P002; ACCEL: Parameter P006

Object Description

Index	0x607a				
Symbol	LAGE_ZIEL	**Length**	4	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Integer32	**Access rights**	Read/write all	**PD Map**	PAD

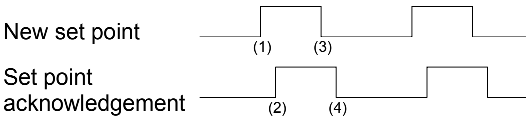
Data Description

Unit	mm (or inch)	**Resolution**	1 ⇔ 0.001 mm (or inch)

The object "target position" can be assigned to the cyclical process output data channel. Then you can cyclically specify new set points. The acceptance of a new set point (see also PA_ENABLE Bit 7) requires a handshake. This is done using the following bits:

- Control word byte 2 bit 4 "new set point" and
- Status word byte 1 bit 4 "acknowledge record point"

Function:

	Transition	Meaning	Conditions
 New set point Set point acknowledgement (1) (3) (2) (4)	1	New target value	Target val.- acknowledgement. ="0" actual value can be transferred
	2	Set point acknow-ledgement	Set point acknowledgement ="1" Set point recognised
	3	New set point	New set point ="0"
	4	Set point acknow-ledgement	Target val.- acknowledgement ="0" New target value can be transferred

Automatic acceptance of the target position from the PAD channel for value changes can be turned off with the Bit 7 from PA_ENABLE (see page 74).

Example

The drive must travel to the absolute position -1000.000 mm.

Service	Write request	**Sub-index**	0	**3. data byte**	0xbd
Command Code	0x8082	**Length**	4	**4. data byte**	0xc0
Param. counter	5	**1. data byte**	0xff		
Index	0x607a	**2. data byte**	0xf0		

2.7.4 S1

Actual position
Current position in relation to real zero.

Object Description

Index	0x5ff9				
Symbol	S1	**Length**	6	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Octet String	**Access rights**	Read all	**PD Map**	Not possible

Data Description

Data format	BCD	**Unit**	mm (or inch)
Value range	-4 000 000 000 ... +4 000 000 000	**Resolution**	1 ⇔ 0.001 mm (or inch)

Data byte [Bit]	Assignment	Data byte [Bit]	Assignment
1 [7 ..... 0]	Value sign (0x00 ⇔ **+**; 0xff ⇔ **-**)		
2 [7,6,5,4]	10^9	4 [3,2,1,0]	10^4
2 [3,2,1,0]	10^8	5 [7,6,5,4]	10^3
3 [7,6,5,4]	10^7	5 [3,2,1,0]	10^2
3 [3,2,1,0]	10^6	6 [7,6,5,4]	10^1
4 [7,6,5,4]	10^5	6 [3,2,1,0]	10^0

2.7.5 LAGE_IST

Position actual value.
Current drive position.

Object Description

Index	0x6064				
Symbol	LAGE_IST	**Length**	4	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Integer32	**Access rights**	Read all	**PD Map**	PED

Data Description

Unit	mm (or inch)	**Resolution**	1 ⇔ 0.001 mm (or inch)

2.7.6 S2

Target position.
End position of the current or last positioning cycle implemented.

Object Description

Index	0x5fdf				
Symbol	S2	**Length**	4	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Integer32	**Access rights**	Read all	**PD Map**	Not possible

Data Description

Unit	mm (or inch)	**Resolution**	1 ⇔ 0.001 mm (or inch)

2.7.7 S12

Position of the absolute value sensor (Option A1).

Object Description

Index	0x5fd9				
Symbol	S12	**Length**	4	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Integer32	**Access rights**	Read all	**PD Map**	Not possible

Data Description

Unit	mm (or inch)	**Resolution**	1 ⇔ 0.001 mm (or inch)

2.7.8 REALNULL

Reference measurement offset.
Difference between real null point and machine zero point.
After the reference run, all positioning processes refer to the real null point.

Object Description

Index	0x607c				
Symbol	REALNULL	**Length**	4	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Integer32	**Access rights**	Read/write all	**PD Map**	Not possible

Data Description

Unit	mm (or inch)	**Resolution**	1 ⇔ 0.001 mm (or inch)

Example

The real null point should be -500.000 mm.

Service	Write request	**Sub-index**	0	**3. data byte**	0x5e
Command Code	0x8082	**Length**	4	**4. data byte**	0xe0
Param. counter	5	**1. data byte**	0xff		
Index	0x607c	**2. data byte**	0xf8		

2.7.9 GRENZEN

Position limit value min-max.
The position limit values are software end limits and correspond to the absolute position limits, within which the set values and actual values (in absolute form) must be moved. Each new target position is checked with these limits. They always refer to the machine zero point; therefore they must be corrected using the reference measurement offset.

Object Description

Index	0x607d				
Symbol	GRENZEN	**Length**	4	**Access groups**	0
Object code	Array	**Elements**	2	**Password**	0
Data type	Integer32	**Access rights**	Read/write all	**PD Map**	Not possible

Data Description

Unit	mm (or inch)	**Resolution**	1 ⇔ 0.001 mm (or inch)

Sub-index	Assignment	Sub-index	Assignment
1	Negative limit stop	2	Positive limit stop

⇨ Note the number range for the Integer32 formats (see page 9 under "1.7.2 Integer").
Limits outside -2 147 483 - +2 147 483 units cannot be depicted.

Example

The position limit value min. (neg. limit stop) is set at -1650.000 mm.

Service	Write request	**Sub-index**	1	**3. data byte**	0xd2
Command Code	0x8082	**Length**	4	**4. data byte**	0x0b
Param. counter	5	**1. data byte**	0xff		
Index	0x607d	**2. data byte**	0xe6		

2.7.10 POS_FENSTER

Positioning window.
The positioning window lies symmetrically around the target position.
Once the position actual value lies within this window, the Bit "Position reached" is set in the status word.

Object Description

Index	6067				
Symbol	POS_FENSTER	**Length**	4	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Unsigned32	**Access rights**	Read/write all	**PD Map**	Not possible

Data Description

Unit	mm (or inch)	**Resolution**	1 ⇔ 0.001 mm (or inch)

Example

The positioning window is set to 15.000 mm.

Service	Write request	**Sub-index**	0	**3. data byte**	0x3a
Command Code	0x8082	**Length**	4	**4. data byte**	0x98
Param. counter	5	**1. data byte**	0x00		
Index	0x6067	**2. data byte**	0x00		

2.7.11 SCHLEPP_FEN

Contour error window.
The contour error window lies symmetrically around the currently set position set point.
If the current position indicator actual value lies outside this window, a contour error occurs.

Object Description

Index	0x6065				
Symbol	SCHLEPP_FEN	**Length**	4	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Unsigned32	**Access rights**	Read/write all	**PD Map**	Not possible

Data Description

Unit	mm (or inch)	**Resolution**	1 ⇔ 0.001 mm (or inch)

Example

The contour error window is set to 10.000 mm.

Service	Write request	**Sub-index**	0	**3. data byte**	0x27
Command Code	0x8082	**Length**	4	**4. data byte**	0x10
Param. counter	5	**1. data byte**	0x00		
Index	0x6065	**2. data byte**	0x00		

2.7.12 S3

Contour error.
Difference between set and actual position in a positioning cycle.

Object Description

Index	0x5ffa				
Symbol	S3	**Length**	2	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Integer16	**Access rights**	read all	**PD Map**	PED

Data Description

Unit	mm (or inch)	**Resolution**	1 ⇔ $^{1}/_{256}$ mm (or inch)

2.7.13 POLARITAET

Polarities.
The set point and actual values are multiplied, depending on the polarity, with 1 or -1. This allows the user to reverse the direction of orientation

Object Description

Index	0x607e				
Symbol	POLARITAET	**Length**	1	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Octet String	**Access rights**	Read/write all	**PD Map**	Not possible

Data Description

Data bit 7	Assignment	Data bit 7	Assignment
= 1	Reversal of direction (motor left)	= 0	default (motor right)

➪ The other bits are irrelevant for COMPAX.

Example

The motor should turn clockwise.

Service	Write request	**Index**	0x607e	**1. data byte**	0x80
Command Code	0x8082	**Sub-index**	0		
Param. counter	4	**Length**	1		

2.7.14 REF_MODE

Reference run selection code.
Selection function with which the reference run method is written.

Object Description

Index	0x6098				
Symbol	REF_MODE	**Length**	2	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Integer16	**Access rights**	Read/write all	**PD Map**	Not possible

Data Description

Selection code	Method	Selection code	Method
8	MN-Ini. approach in direction 1	12	MN-Ini. approach in direction 2

The COMPAX parameter P213 is influenced by this object.
The direction in which the machine zero is approached is also influenced by the COMPAX parameters P212 (machine zero mode) and P3 (speed of machine zero travel).

Example

The machine zero point initiator is approached with the positive direction in the reference run.

Service	Write request	**Index**	0x6098	**1. data byte**	0x00
Command Code	0x8082	**Sub-index**	0	**2. data byte**	0x08
Param. counter	4	**Length**	2		

2.7.15 CAM_CMD

Special commands for COMPAX XX70.
The Sub-index is used to select the corresponding command (Sub-index = CAM command).

Object Description

Index	0x5fcc				
Symbol	CAM_CMD	**Length**	4	**Access groups**	0
Object code	Array	**Elements**	6	**Password**	0
Data type	Integer32	**Access rights**	Write all	**PD Map**	Not possible

Data Description

Sub-index	Command	Resolution
1	SETC	1
2	SETM	1 ⇔ 0.001
3	SETS	1 ⇔ 0.001
4	POSR CAM	-
5	LOOP	1
6	VF	-

2.7.16 WAITPOSA

Synchronisation with automatic reverse travel (clocked command; COMPAX XX50).
Starting from the rest position of the drive, a complete synchronisation move is carried out.
The value for this object is the processing interval (length of material when cutting).

Object Description

Index	0x5fe0				
Symbol	WAITPOSA	**Length**	4	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Integer32	**Access rights**	Write all	**PD Map**	Not possible

Data Description

Unit	mm (or inch)	**Resolution**	1 ⇔ 0.001 mm (or inch)

Example

A cutting length of 205.000 mm is required.

Service	Write request	**Sub-index**	0	**3. data byte**	0x20
Command Code	0x8082	**Length**	4	**4. data byte**	0xc8
Param. counter	5	**1. data byte**	0x00		
Index	0x5fe0	**2. data byte**	0x03		

2.7.17 WAITPOSR

Synchronisation without automatic reverse travel (clocked command; COMPAX XX50).
Starting from the rest position of the drive, a complete synchronisation move is carried out.
The value for this object is the processing interval (length of material when cutting).

Object Description

Index	0x5fe1				
Symbol	WAITPOSR	**Length**	4	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Integer32	**Access rights**	Write all	**PD Map**	Not possible

Data Description

Unit	mm (or inch)	**Resolution**	1 ⇔ 0.001 mm (or inch)

Example

A cutting length of 720.000 mm is required.

Service	Write request	**Sub-index**	0	**3. data byte**	0xfc
Command Code	0x8082	**Length**	6	**4. data byte**	0x80
Param. counter	5	**1. data byte**	0x00		
Index	0x5fe1	**2. data byte**	0x0a		

2.8 SPEED

2.8.1 SPEED

Traverse speed in % of the nominal speed (nominal rpm * travel per motor revolution).
The value is valid until a new value is programmed.
The set speed can be reduced by using the OVERRIDE object.
A speed change during the positioning cycle is possible by using the POSR0SPEED object.

Object Description

Index	0x5fec				
Symbol	SPEED	**Length**	2	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Integer16	**Access rights**	Read/write all	**PD Map**	PAD

Data Description

Unit	%	**Resolution**	1 ⇔ $^1/_{64}$%; (6400 ⇔ 100%)

Example

The drive should travel at 75% of the nominal speed.

Service	Write request	**Index**	0x5fec	**1. data byte**	0x12
Command Code	0x8082	**Sub-index**	0	**2. data byte**	0xc0
Param. counter	4	**Length**	2		

2.8.2 OVERRIDE

Reduce traverse speed.
Software emulation of an external potentiometer on the override input (X11.6).

Object Description

Index	0x5fee				
Symbol	OVERRIDE	**Length**	1	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Unsigned8	**Access rights**	Read/write all	**PD Map**	PAD

Data Description

Unit	%	**Resolution**	1 ⇔ $^1/_{255}$%; (255 ⇔ 100%)

Example

The traverse speed must be reduced by 50%.

Service	Write request	**Index**	0x5fee	**Data byte**	0x80
Command Code	0x8082	**Sub-index**	0		
Param. counter	4	**Length**	1		

2.8.3 POSR0SPEED

Changing traverse speed during a positioning cycle.

Object Description

Index	0x5fed				
Symbol	POSR0SPEED	**Length**	3	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Octet String	**Access rights**	Write all	**PD Map**	Not possible

Data Description

Data format	BCD	**Unit**	%
Value range	1 ... 600 000	**Resolution**	1 ⇔ 0.001 %

Data byte [Bit]	Assignment	Data byte [Bit]	Assignment
1 [7,6,5,4]	10^5	2 [3,2,1,0]	10^2
1 [3,2,1,0]	10^4	3 [7,6,5,4]	10^1
2 [7,6,5,4]	10^3	3 [3,2,1,0]	10^0

Example

The drive should continue travel at 35% of the nominal speed.

Service	Write request	**Index**	0x5fed	**1. data byte**	0x03
Command Code	0x8082	**Sub-index**	0	**2. data byte**	0x50
Param. counter	5	**Length**	3	**3. data byte**	0x00

2.8.4 POSRXSPEEDY

Speed step profile.
Every revolution step profile can have a maximum of 8 revolution steps. The position value is given as a relative measurement. It is referenced to the positioning start point.

Object Description

Index	0x5fe4				
Symbol	POSRXSPEEDY	**Length**	8	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Octet String	**Access rights**	Write all	**PD Map**	Not possible

Data Description

Data byte	Assignment	Data byte	Assignment
1	Highest value byte of the position	4	Lowest value byte of the position

Coding	Complement to two	**Unit**	mm (or inch)
Value range	-2 147 483 648 ... +2 147 483 647	**Resolution**	1 ⇔ 0.001 mm (or inch)

Data byte	Assignment	Data byte	Assignment
5	Highest value byte of the speed	8	Lowest value byte of the speed

Coding	binary	**Unit**	%
Value range	1 ... 600 000	**Resolution**	1 ⇔ 0.001 %

Example

The drive should continue travel at 33% of the nominal speed after 580 mm.

Service	Write request	**Length**	8	**5. data byte**	0x00
Command Code	0x8082	**1. data byte**	0x00	**6. data byte**	0x00
Param. counter	7	**2. data byte**	0x08	**7. data byte**	0x80
Index	0x5fe4	**3. data byte**	0xd9	**8. data byte**	0xe8
Sub-index	0	**4. data byte**	0xa0		

2.8.5 PRXSDYALZ

Speed step profile.
Every revolution step profile can have a maximum of 8 revolution steps. The position value is given as a relative measurement. It is referenced to the positioning start point.

Object Description

Index	0x5fc7				
Symbol	PRXSDYALZ	**Length**	10	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Octet String	**Access rights**	Write all	**PD Map**	Not possible

Data Description

Data byte	Assignment	Data byte	Assignment
1	Highest value byte of the position	4	Lowest value byte of the position

Coding	Two’s complement	**Unit**	mm (or inch)
Value range	-2 147 483 648 ... +2 147 483 647	**Resolution**	1 ⇔ 0.001 mm (or Inch)

Data byte	Assignment	Data byte	Assignment
5	Highest value byte of the speed	8	Lowest value byte of the speed

Coding	binary	**Unit**	%
Value range	1 ... 600 000	**Resolution**	1 ⇔ 0.001 %

Data byte	Assignment	Data byte	Assignment
9	MSB of the ramp time	10	LSB of the ramp time

Coding	binary	**Unit**	ms
Value range	0 ... 65 000	**Resolution**	1 ⇔ 1 ms

2.8.6 DREHZAHLMAX

Maximum motor speed value.
The maximum speed is given for both directions of rotation with a resolution of 1 rpm.
These are for the protection of the motor and can be found in the motor data sheet.

Object Description

Index	0x6080				
Symbol	DREHZAHLMAX	**Length**	2	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Unsigned16	**Access rights**	Read/write all	**PD Map**	Not possible

Data Description

Unit	Rpm	**Resolution**	1 ⇔ 1 rpm

Example

The max. speed of the motor is 6000 rpm.

Service	Write request	**Index**	0x6080	**1. data byte**	0x17
Command Code	0x8082	**Sub-index**	0	**2. data byte**	0x70
Param. counter	4	**Length**	2		

2.8.7 GESCHW_MAX

Maximum speed
The maximum speed applies to both rotation directions.

Object Description

Index	0x607f				
Symbol	GESCHW_MAX	**Length**	4	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Unsigned32	**Access rights**	Read/write all	**PD Map**	Not possible

Data Description

Unit	%	**Resolution**	1 ⇔ 0.001 %

Example

The maximum speed is set to 95% of the nominal speed.

Service	Write request	**Sub-index**	0	**3. data byte**	0x73
Command Code	0x8082	**Length**	4	**4. data byte**	0x18
Param. counter	5	**1. data byte**	0x00		
Index	0x607f	**2. data byte**	0x01		

2.8.8 VERF_GESCHW

Traverse speed.
Given in % of the nominal speed (nominal rpm * travel per motor revolution).
The value is valid until a new value is programmed.
The set speed can be reduced by using the OVERRIDE object.
A speed change during the positioning cycle is possible by using the POSR0SPEED object.

Object Description

Index	0x6081				
Symbol	**VERF_GESCHW**	**Length**	4	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Integer32	**Access rights**	Read/write all	**PD Map**	Not possible

Data Description

Unit	%	**Resolution**	1 ⇔ 0.001 %

Example

The drive should travel at 66% of the nominal speed.

Service	Write request	**Sub-index**	0	**3. data byte**	0x01
Command Code	0x8082	**Length**	4	**4. data byte**	0xd0
Param. counter	5	**1. data byte**	0x00		
Index	0x6081	**2. data byte**	0x01		

2.8.9 S4

Current axis process speed.
Value in % of the nominal speed (nominal rpm * travel per motor revolution).

Object Description

Index	0x5ffc				
Symbol	S4	**Length**	2	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Integer16	**Access rights**	Read all	**PD Map**	PED

Data Description

Unit	%	**Resolution**	1 ⇔ $^1/_{64}$%; (6400 ⇔ 100%)

2.8.10 GESCHW_IST

Traverse speed - actual value.
Value in % of the nominal speed (nominal rpm * travel per motor revolution).

Object Description

Index	0x606c				
Symbol	GESCHW_IST	**Length**	4	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Integer32	**Access rights**	Read all	**PD Map**	Not possible

Data Description

Unit	%	**Resolution**	1 ⇔ 0.001 %

2.8.11 GESCHW_REF

Reference run speed.
Speed set point for approaching the machine zero point.
Given in % of the nominal speed (nominal rpm * travel per motor revolution).

Object Description

Index	0x6099				
Symbol	GESCHW_REF	**Length**	4	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Unsigned32	**Access rights**	Read/write all	**PD Map**	Not possible

Data Description

Unit	%	**Resolution**	1 ⇔ 0.001 %

Example

The reference run speed is set to 20% of the nominal speed.

Service	Write request	**Sub-index**	0	**3. data byte**	0x4e
Command Code	0x8082	**Length**	4	**4. data byte**	0x20
Param. counter	5	**1. data byte**	0x00		
Index	0x6099	**2. data byte**	0x00		

2.9 Acceleration

2.9.1 RAMPENFORM

Ramp form speed.
Selection function with which the acceleration process is written.

Object Description

Index	0x6086				
Symbol	RAMPENFORM	**Length**	2	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Integer16	**Access rights**	Read/write all	**PD Map**	Not possible

Data Description

Selection code	Ramp shape	Selection code	Ramp shape
-1	square	2	without jerk
0	linear		

Example

The drive must travel with a squared ramp form.

Service	Write request	**Index**	0x6086	**1. data byte**	0xff
Command Code	0x8082	**Sub-index**	0	**2. data byte**	0xff
Param. counter	4	**Length**	2		

2.9.2 ACCEL_POS

Acceleration time.
Time setting for the acceleration process.
Also the time setting for the deceleration process as long as the object ACCEL-NEG or RAMPE-NEG has not been written to.

The time specification applies to nominal speed (100%).

$$t_a = \frac{\text{SPEED}}{100\%} \bullet \text{ACCEL\_POS}$$

Object Description

Index	0x5fea				
Symbol	ACCEL_POS	**Length**	2	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Unsigned16	**Access rights**	Write all	**PD Map**	Not possible

Data Description

Value range	0 ... 65 000		
Unit	ms	**Resolution**	1 ⇔ 1 ms

Example

The next positioning process must be implemented with an acceleration time of 1000 ms.

Service	Write request	**Index**	0x5fea	**1. data byte**	0x03
Command Code	0x8082	**Sub-index**	0	**2. data byte**	0xe8
Param. counter	4	**Length**	2		

2.9.3 ACCEL_NEG

Deceleration time.
Time setting for the deceleration process.

The time specification applies to nominal speed (100%).

$$t_a = \frac{SPEED}{100\%} \bullet ACCEL\_NEG$$

Object Description

Index	0x5feb				
Symbol	ACCEL_NEG	**Length**	2	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Unsigned16	**Access rights**	Write all	**PD Map**	Not possible

Data Description

Value range	0 ... 65 000		
Unit	ms	**Resolution**	1 ⇔ 1 ms

Example

The next positioning process must be implemented with an acceleration time of 2500 ms.

Service	Write request	**Index**	0x5feb	**1. data byte**	0x09
Command Code	0x8082	**Sub-index**	0	**2. data byte**	0xc4
Param. counter	4	**Length**	2		

2.9.4 RAMPE_POS

Acceleration.
Time setting for the acceleration process.
Also the time setting for the deceleration process as long as the object ACCEL-NEG or RAMPE-NEG has not been written to.

The time specification applies to nominal speed (100%).

$$t_a = \frac{SPEED}{100\%} \bullet RAMPE\_POS$$

Object Description

Index	0x6083				
Symbol	RAMPE_POS	**Length**	4	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Unsigned32	**Access rights**	Read/write all	**PD Map**	Not possible

Data Description

Value range	0 ... 65 000		
Unit	ms	**Resolution**	1 ⇔ 1 ms

Example

The acceleration time is set to 470 ms.

Service	Write request	**Sub-index**	0	**3. data byte**	0x01
Command Code	0x8082	**Length**	4	**4. data byte**	0xd6
Param. counter	5	**1. data byte**	0x00		
Index	0x6083	**2. data byte**	0x00		

2.9.5 RAMPE_NEG

Lag.
Time setting for the deceleration process.

The time specification applies to nominal speed (100%).

$$t_a = \frac{SPEED}{100\%} \bullet RAMPE\_NEG$$

Object Description

Index	0x6084				
Symbol	RAMPE_NEG	**Length**	4	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Unsigned32	**Access rights**	Read/write all	**PD Map**	Not possible

Data Description

Value range	0 ... 65 000		
Unit	ms	**Resolution**	1 ⇔ 1 ms

Example

The deceleration time is set to 1525 ms.

Service	Write request	**Sub-index**	0	**3. data byte**	0x05
Command Code	0x8082	**Length**	4	**4. data byte**	0xf5
Param. counter	5	**1. data byte**	0x00		
Index	0x6084	**2. data byte**	0x00		

2.9.6 RAMPE_NOTS

Rapid stop.
Time setting for the deceleration process if the command rapid stop (Bit 2 in the control word) is given, a limit switch is activated or after an emergency stop.

The time specification applies to nominal speed (100%).

$$t_a = \frac{SPEED}{100\%} \bullet RAMPE\_NOTS$$

Object Description

Index	0x6085				
Symbol	RAMPE_NOTS	**Length**	4	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Unsigned32	**Access rights**	Read/write all	**PD Map**	Not possible

Data Description

Value range	0 ... 65 000		
Unit	ms	**Resolution**	1 ⇔ 1 ms

Example

The braking time for rapid stop is set to 125 ms.

Service	Write request	**Sub-index**	0	**3. data byte**	0x00
Command Code	0x8082	**Length**	4	**4. data byte**	0x7d
Param. counter	5	**1. data byte**	0x00		
Index	0x6085	**2. data byte**	0x00		

2.9.7 RAMPE_REF

Reference run acceleration.
Acceleration time for approaching the machine zero point.

The time specification applies to nominal speed (100%).

$$t_a = \frac{\text{SPEED}}{100\%} \bullet \text{RAMPE\_REF}$$

Object Description

Index	0x609a				
Symbol	RAMPE_REF	**Length**	4	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Unsigned32	**Access rights**	Read/write all	**PD Map**	Not possible

Data Description

Value range	0 ... 65 000		
Unit	ms	**Resolution**	1 ⇔ 1 ms

Example

The acceleration time for the reference run is set to 733 ms.

Service	Write request	**Sub-index**	0	**3. data byte**	0x02
Command Code	0x8082	**Length**	4	**4. data byte**	0xdd
Param. counter	5	**1. data byte**	0x00		
Index	0x609a	**2. data byte**	0x00		

2.10 Inputs/outputs

2.10.1 INPUT_WORD

Logic state of the 16 digital inputs.
Some inputs are assigned fixed control functions.

Input	Assignment	Input	Assignment
1	SHIFT	1 & 3	Find real null (RN)
2	Hand+	1 & 4	Teach real zero
3	Hand-	1 & 5	Reserved
4	Quit	1 & 6	Break
5	START	9...13	Freely assignable in standard model
6	STOP	14	Activate label reference
7...8	Freely assignable in standard model	15	Faster start
1 & 2	Find machine zero (MN)	16	Label input

Object Description

Index	0x5ff8				
Symbol	INPUT_WORD	**Length**	2	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Octet String	**Access rights**	Read all	**PD Map**	PED

Data Description

Data byte [Bit]	Assignment	Data byte [Bit]	Assignment
1 [7]	Status input 16	2 [7]	Status input 8
1 [0]	Status input 9	2 [0]	Status input 1

2.10.2 OUTPUT_WORD

Logic state of the 16 digital outputs.
Some outputs are assigned a fixed status information.

Output	Assignment	Output	Assignment
1	no fault	5	Programmed nominal position reached
2	No warning	6	Idle after stop
3	Machine zero has been approached	7...15	Freely assignable in standard model
4	Ready for start	16	Label present after max. feed length

This object allows the outputs to be set or reset to default.
Each output which must be influenced via the Interbus-S, must be specifically enabled with the object OUTPUT-MASK. The output thereby loses any status information which was assigned to it.

Object Description

Index	0x5ff7				
Symbol	**OUTPUT_WORD**	**Length**	2	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Octet string	**Access rights**	Read/write all	**PD Map**	PED & PAD

Data Description

Data byte [Bit]	Assignment	Data byte [Bit]	Assignment
1 [7]	Status output 16	2 [7]	Status output 8
1 [0]	Status output 9	2 [0]	Status output 1

2.10.3 OUTPUT

Set or reset a digital output.
The corresponding output is selected using the Sub-index (Sub-index = output no.).
Some outputs have a fixed status information assigned (see OUTPUT-WORD).

Object Description

Index	0x5ff6				
Symbol	OUTPUT	**Length**	1	**Access groups**	0
Object code	Array	**Elements**	16	**Password**	0
Data type	Boolean	**Access rights**	Write all	**PD Map**	Not possible

Data Description

Data byte	Function	Data byte	Function
= 0xff (TRUE)	Output [Sub-index] = 1	= 0x00 (FALSE)	Output [Sub-index] = 0

Example

Output 13 must be set.

Service	Write request	**Index**	0x5ff6	**Data byte**	0xff
Command Code	0x8082	**Sub-index**	13		
Param. counter	4	**Length**	1		

2.10.4 OUTPUT_MASK

Mask outputs.
Each output which must be influenced via the Interbus-S, must be specifically enabled (masked).
The output thereby loses any status information which was assigned to it.
After Power On, the OUTPUT-MASK has the value 0, i.e. all outputs are disabled for the IBS (not masked).

Object Description

Index	0x5ff5				
Symbol	OUTPUT_MASK	**Length**	2	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Octet string	**Access rights**	Read/write all	**PD Map**	Not possible

Data Description

Data byte [Bit]	Assignment	Data byte [Bit]	Assignment
1 [7]	Mask output 16	2 [7]	Mask output 8
1 [0]	Mask output 9	2 [0]	Mask output 1

Example

Outputs 9 - 16 must be masked.

Service	Write request	**Index**	0x5ff5	**1. data byte**	0xff
Command Code	0x8082	**Sub-index**	0	**2. data byte**	0x00
Param. counter	4	**Length**	2		

2.10.5 POSROUTPUTP

Comparator function (active high).
Set an unassigned output within a positioning cycle.
The position value is given as a relative measurement. It is referenced to the positioning start point.
A maximum of 8 comparators can be set for a positioning process.
The corresponding output is selected using the Sub-index (Sub-index = output no.).

Object Description

Index	0x5fe2				
Symbol	POSROUTPUTP	**Length**	4	**Access groups**	0
Object code	Array	**Elements**	16	**Password**	0
Data type	Integer32	**Access rights**	Write all	**PD Map**	Not possible

Data Description

Unit	mm (or inch)	**Resolution**	1 ⇔ 0.001 mm (or inch)

Example

Output 11 must be set when the drive has travelled 888 mm.

Service	Write request	**Sub-index**	11	**3. data byte**	0x65
Command Code	0x8082	**Length**	4	**4. data byte**	0x00
Param. counter	5	**1. data byte**	0x00		
Index	0x5fe2	**2. data byte**	0x04		

2.10.6 POSROUTPUTN

Comparator function (active low).
Reset an unassigned output within a positioning cycle.
The position value is given as a relative measurement. It is referenced to the positioning start point.
A maximum of 4 comparators can be set for a positioning process.
The corresponding output is selected using the Sub-index (Sub-index = output no.).

Object Description

Index	0x5fe3				
Symbol	POSROUTPUTN	**Length**	4	**Access groups**	0
Object code	Array	**Elements**	16	**Password**	0
Data type	Integer32	**Access rights**	Write all	**PD Map**	Not possible

Data Description

Unit	mm (or inch)	**Resolution**	1 ⇔ 0.001 mm (or inch)

Example

Output 11 must be reset when the drive has travelled 1888 mm.

Service	Write request	**Sub-index**	11	**3. data byte**	0xa7
Command Code	0x8082	**Length**	4	**4. data byte**	0x40
Param. counter	5	**1. data byte**	0x00		
Index	0x5fe3	**2. data byte**	0x13		

2.11 Programming

2.11.1 N

Reading and writing the program memory with command records in plain text (ASCII format).
The record number is determined with the sub-index (Sub-index = record No.)

Object Description

Index	0x5ff1				
Symbol	N	**Length**	32	**Access groups**	0
Object code	Array	**Elements**	250	**Password**	0
Data type	Visible string	**Access rights**	Read/write all	**PD Map**	Not possible

Data Description

Data format	ASCII	**Value range**	0x20 ... 0x7f
Data byte 1	1st character of the command record	**Data byte 32**	32nd character of the command record

Example

Record 5 must be written with the command "POSA 250".

Service	Write request	**Length**	20	**5. data byte**	0x20 (" ")
Command Code	0x8082	**1. data byte**	0x50 ("P")	**6. data byte**	0x32 ("2")
Param. counter	19	**2. data byte**	0x4f ("O")	**7. data byte**	0x35 ("5")
Index	0x5ff1	**3. data byte**	0x53 ("S")	**8. data byte**	0x30 ("0")
Sub-index	5	**4. data byte**	0x41 ("A")	**9...32. data byte**	0x20 (" ")

⇨ Long, combined commands with 32 characters cannot be completely displayed.
These commands can however be written with the use of command abbreviations.

2.11.2 Nx

Reading and writing the program memory with command records in binary format.
The record number is determined with the Sub-index (Sub-index = record No.)

Object Description

Index	0x5fd5				
Symbol	Nx	**Length**	20	**Access groups**	0
Object code	Array	**Elements**	250	**Password**	0
Data type	Octet string	**Access rights**	Read/write all	**PD Map**	Not possible

Data Description

Data byte	Assignment
1	Record length (number of relevant record data)
2	1st character of the record content
...	
20	last character of the record contents

COMPAX – Command codes

Definition of the command code
A command code consists of 1 byte.

Sorted by command code

Code	Command
0x20	Empty instruction (No Operation)
0x01	**VALIDP / C / F**
0x41	**POSA** value / **POSA HOME**
0x42	**GOSUB**
0x43	**SETC** n
0x45	**END**
0x47	**GOTO**
0x49	**IF Ex=y** ... / **IF ERROR ... / IF STOP ...**
0x4A	**IF <Operand1> < Comparison operator > <Operand2> ...**
0x4B	**LOOP** n
0x4C	**ACCEL** value
0x4D	**SETM** Value
0x4F	**OUTPUT Ax=y**
0x50	**Pn=. . .**
0x51	**SETS**
0x52	**POSR** value / **POSR CAM**
0x53	**SPEED** value / **SPEED SYNC**
0x54	**REPEAT** Value
0x55	**RETURN**
0x56	**Vn=. . .**
0x57	**WAIT** value / **WAIT START**
0x61	**POSA** Parameter
0x6B	**LOOP** Parameter
0x6C	**ACCEL** Parameter
0x6D	**SETM** Parameter
0x72	**POSR** Parameter
0x73	**SPEED** Parameter
0x74	**REPEAT** Parameter
0x77	**WAIT** Parameter
0xC1	**POSA** Variable
0xCB	**LOOP** Variable
0xCC	**ACCEL** Variable
0xCD	**SETM** Variable
0xD2	**POSR** Variable
0xD3	**SPEED** Variable
0xD4	**REPEAT** Variable
0xD7	**WAIT** Variable

Sorted by command code

Code	Command
0x6C	**ACCEL** Parameter
0xCC	**ACCEL** Variable
0x4C	**ACCEL** value
0x45	**END**
0x42 0x47	**GOSUB** **GOTO**
0x4A	IF <Operand1> <Comparison operator> <Operand2> **...**
0x49	**IF Ex=y** ... / **IF ERROR ... / IF STOP ...**
0x20	Empty instruction (No Operation)
0x4B 0x6B	**LOOP n** **LOOP** Parameter
0xCB	**LOOP** Variable
0x4F	**OUTPUT** Ax=y
0x50	**Pn=. . .**
0x61	**POSA** Parameter
0xC1	**POSA** Variable
0x41	**POSA** value / **POSA HOME**
0x72	**POSR** Parameter
0xD2	**POSR** Variable
0x52	**POSR** value / **POSR CAM**
0x74	**REPEAT** Parameter
0xD4	**REPEAT** Variable
0x54	**REPEAT** Value
0x55	**RETURN**
0x43	**SETC** n
0x6D	**SETM** Parameter
0xCD	**SETM** Variable
0x4D	**SETM** Value
0x51	**SETS**
0x73	**SPEED Parameter**
0x3D	**SPEED** Variable
0x53	**SPEED** value / **SPEED SYNC**
0x01	**VALIDP / C / F**
0x56	**Vn=. . .**
0x77	**WAIT Parameter**
0x7D	**WAIT Variable**
0x57	**WAIT VALUE/ WAIT START**

Operand codes

An operand consists of 7 bytes; 1 byte for the type indicator and 6 data bytes.

Operand	Type	D1	D2	D3	D4	D5	D6
Parameter	**0x50**	**No.H**	Nr.L	0x00	0x00	0x00	0x00
Status	0x53	No.H	Nr.L	0x00	0x00	0x00	0x00
Variable	0x56	No.H	Nr.L	0x00	0x00	0x00	0x00
Constants	0x20	NL	NM	NH	VL	VM	VH

Comparison operator codes

A comparison operator consists of 1 byte.

Comparison operator	Symbols	Code
Equal	=	0x3D
Less than	<	0x3C
More than	**>**	**0x3E**
Equal to/less than	<=	0xBC
Equal to/greater than	>=	0xBE
Does not equal	<>	0xBB

Arithmetic operator codes

An arithmetic operator consists of 1 byte.

Arithmetic Operator	Symbols	Code
Addition	+	0x2B
Subtraction	-	0x2D
Multiplication	*	0x2A
Division	/	0x2F
Whole number division	\	0x5C
Modulo calculation	%	0x25

The following set memory – command code table is a result of the application of these codes. All of the commands are listed individually here!

COMPAX Set memory-command code table

Command	Code							
ACCEL Parameter	0x6C	No.H	Nr.L					
ACCEL Variable	0xCC	No.H	Nr.L					
ACCEL Value	0x4C	MSB	LSB					
END	0x45	0x00						
GOSUB EXT	0x42	0x00	0x00					
GOSUB Value	0x42	MSB	LSB					
GOTO EXT	0x47	0x00	0x00					
GOTO Value	0x47	MSB	LSB					
IF ERROR GOSUB n	0x49	0x00	0xFF	0x31	0x42	n MSB	n LSB	
IF Ex**=**y **GOSUB** n	0x49	x MSB	x LSB	y	0x42	n MSB	n LSB	
IF Ex**=**yy **GOSUB** n	0x49	x MSB	x LSB	y1	y2	0x42	n MSB	n LSB
IF Ex**= . . .**	...	...	...	...	...	...	...	...
IF Ex**=**yyyyyyyy **GOSUB** n	0x49	x MSB	x LSB	y1	y2	y3	y4	y5
	y6	y7	y8	0x42	n MSB	n LSB		
IF Ex**=**y **GOTO** n	0x49	x MSB	x LSB	y	0x47	n MSB	n LSB	
IF Ex**=**yy **GOTO** n	0x49	x MSB	x LSB	y1	y2	0x47	n MSB	n LSB
IF Ex**= . . .**	...	...	...	...	...	...	...	...
IF Ex**=**yyyyyyyy **GOTO** n	0x49	x MSB	x LSB	y1	y2	y3	y4	y5
	y6	y7	y8	0x47	n MSB	n LSB		
IF <Operand1**> <**Comparison operator**> <**Operand2**> GOTO** n	0x4A	O1Type	O1D1	O1D2	O1D3	O1D4	O1D5	O1D6
	Vglop	O2Type	O2D1	O2D2	O2D3	O2D4	O2D5	O2D6
	0x47	n MSB	n LSB					
IF <Operand1**> <** Comparison operator **> <**Operand2**> GOSUB** n	0x4A	O1Type	O1D1	O1D2	O1D3	O1D4	O1D5	O1D6
	Vglop	O2Type	O2D1	O2D2	O2D3	O2D4	O2D5	O2D6
	0x42	n MSB	n LSB					
IF STOP GOSUB n	0x49	0x00	0xFE	0x31	0x42	n MSB	n LSB	
LOOP n	0x4B	n MSB	n LSB					
LOOP Parameter	0x6B	No.H	Nr.L					
LOOP Variable	0xCB	No.H	No .L					
OUTPUT Ax**=**y	0x4F	x MSB	x LSB	y				
OUTPUT Ax**=**yy	0x4F	x MSB	x LSB	y1	y2			
OUTPUT Ax**=. . .**	...	...	...	...	...	...		
OUTPUT Ax**=**yyyyyyyy	0x4F	x MSB	x LSB	y1	y2	y3	y4	y5
	y6	y7	y8					
OUTPUT A0=y	0x4F	0x00	0x00	y				
POSA HOME	0x41	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	
POSA Parameter	0x61	No.H	Nr.L	0x00	0x00	0x00	0x00	
POSA Variable	0x1C	No.H	Nr.L	0x00	0x00	0x00	0x00	
POSA Value t	0x41	NL	NM	NH	VL	VM	VH	
POSR CAM	0x52	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	
POSR Parameter	0x72	No.H	Nr.L	0x00	0x00	0x00	0x00	
POSR Variable	0x2D	No.H	Nr.L	0x00	0x00	0x00	0x00	
POSR Value	0x52	NL	NM	NH	VL	VM	VH	
REPEAT Parameter	0x74	No.H	Nr.L					
REPEAT Variable	0x4D	No.H	Nr.L					
REPEAT Value t	0x54	MSB	LSB					
RETURN	0x55	0x00						
SETC n	0x43	n MSB	n LSB					
SETM Value	0x4D	NL	NM	NH	VL	VM	VH	
SETM Parameter	0x6D	No.H	Nr.L	0x00	0x00	0x00	0x00	
SETM Variable	0xCD	No.H	Nr.L	0x00	0x00	0x00	0x00	
SETS	0x51							
SPEED Parameter	0x73	No.H	Nr.L	0x00	0x00	0x00	0x00	
SPEED Variable	0x3D	No.H	Nr.L	0x00	0x00	0x00	0x00	

Command	Code							
SPEED Value	0x53	NL	NM	NH	VL	VM	VH	
SPEED SYNC	0x53	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	
VALIDP	0x01	0x56	0x50					
VALIDC	0x01	0x56	0x43					
VALIDF	0x01	0x56	0x46					
WAIT Parameter	0x77	No.H	Nr.L					
WAIT Variable	0x7D	No.H	Nr.L					
WAIT Value	0x57	MSB	LSB					
WAIT START	0x57	0x00	0x00					
POSA Value **WAIT** Value	0x41	NL	NM	NH	VL	VM	VH	0x57
	MSB	LSB						
POSA . . . **WAIT** . . .	...	...	...	...	...	...	...	
(Combinations examples.: ... Value ... .V12, ...P40 ... .V10)	...	...	...	...	...	...	...	
POSA Variable **WAIT** Variable	0x1C	No.H	Nr.L	0x00	0x00	0x00	0x00	0x7D
	No.H	Nr.L						
POSR Value **OUTPUT A**x**=**y	0x52	NL	NM	NH	VL	VM	VH	0x4F
	x MSB	x LSB	y					
POSR Parameter **OUTPUT A**x**=**y	0x72	No.H	Nr.L	0x00	0x00	0x00	0x00	0x4F
	x MSB	x LSB	y					
POSR Variable **OUTPUT A**x**=**y	0x2D	No.H	Nr.L	0x00	0x00	0x00	0x00	0x4F
	x MSB	x LSB	y					
POSR Value t **SPEED** Value	0x52	NL	NM	NH	VL	VM	VH	0x53
	NL	NM	NH	VL	VM	VH		
POSR . . . **SPEED** . . .	...	...	...	...	...	...	...	
(Combinations examples.: ... Value ... .V12, ...P40 ... .V10)	...	...	...	...	...	...		
POSR Variable **SPEED** Variable	0x2D	No.H	Nr.L	0x00	0x00	0x00	0x00	0x3D
	No.H	Nr.L	0x00	0x00	0x00	0x00		
POSR Value **SPEED** Value **ACCEL** Value	0x52	NL	NM	NH	VL	VM	VH	0x53
	NL	NM	NH	VL	VM	VH	0x4C	MSB
	LSB							
POSR ... **SPEED** ... **ACCEL** ...	...	...	...	...	...	...	...	
(Combinations examples.: ... Value ... .V12 ... .V13, ...P40 ... .V10 ... .P41)	...	...	...	...	...	...	...	
	...							
POSR Variable **SPEED** Variable **ACCEL** Variable	0x2D	No.H	Nr.L	0x00	0x00	0x00	0x00	0x3D
	No.H	Nr.L	0x00	0x00	0x00	0x00	0xCC	No.H
	Nr.L							
POSR Value **WAIT** Value	0x52	NL	NM	NH	VL	VM	VH	0x57
	MSB	LSB						
POSR . . . **WAIT** . . .	...	...	...	...	...	...	...	...
(Combinations examples.: ... Value ... .V12, ...P40 ... .V10)	...	...						
POSR Variable **WAIT** Variable	0x2D	No.H	Nr.L	0x00	0x00	0x00	0x00	0x7D
	No.H	Nr.L						
SPEED Value **WAIT** Value	0x53	NL	NM	NH	VL	VM	VH	0x57
	MSB	LSB						
SPEED . . . **WAIT** . . .	...	...	...	...	...	...	...	...
(Combinations examples.: ... Value ... .V12, ...P40 ... .V10)	...	...						
SPEED Variable **WAIT** Variable	0x3D	No.H	Nr.L	0x00	0x00	0x00	0x00	0x7D
	No.H	Nr.L						
WAIT POSA Value	0x57	0x00	0x00	0x41	NL	NM	NH	VL
	VM	VH						
WAIT POSR Value	0x57	0x00	0x00	0x52	NL	NM	NH	VL
	VM	VH						
Pn**=<**Operand1**> [<**Arithmetic Operator**> <**Operand2**>]**	0x50	n MSB	n LSB	O1Type	O1D1	O1D2	O1D3	O1D4
	O1D5	O1D6	AriOp	O2Type	O2D1	O2D2	O2D3	O2D4
	O2D5	O2D6						

Command	Code							
Vn**=<**Operand1**> [<**Arithmetic Operator**> <**Operand2**>]**	0x56	n MSB	n LSB	O1Type	O1D1	O1D2	O1D3	O1D4
	O1D5	O1D6	AriOp	O2Type	O2D1	O2D2	O2D3	O2D4
	O2D5	O2D6						

Key:

No.H:	High byte of the parameter/variable number
Nr.L:	Low byte of the parameter/variable number
MSB:	High byte of an integer value
LSB:	Low byte of an integer value
NL:	Low byte of the fractional digit of a value in DSP number format
NM:	Mid byte of the fractional digit of a value in DSP number format
NH:	High byte of the fractional digit of a value in DSP number format
VL:	Low Byte of the integral digit of a value in DSP number format
VM:	Mid Byte of the integral digit of a value in DSP number format
VH:	High byte of the integral digit of a value in DSP number format
O1Type:	Type indicator of the 1. operand
O1D1...O1D6:	Data for the 1st operand
O2Type:	Type indicator of the 2. operand
O2D1...O2D6:	Data for the 2nd operand
Vglop:	Comparison operator
AriOp:	Arithmetic Operator
y (y1, y2, ...)	y=0x30 for "1" and y=0x31 for "0"

2.11.3 GOTO

Set and read record pointer.

Object Description

Index	0x5ff2				
Symbol	GOTO	**Length**	1	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Unsigned8	**Access rights**	Read/write all	**PD Map**	Not possible

Data Description

Data format	Binary	**Unit**	Record number
Value range	1 ... 250	**Resolution**	1

Example

The record pointer is set to command record 18.

Service	Write request	**Index**	0x5ff2	**Data byte**	0x12
Command Code	0x8082	**Sub-index**	0		
Param. counter	4	**Length**	1		

2.11.4 START_N

Run program record N.
Only this record is processed. The record pointer remains at this program record.

Object Description

Index	0x5ff4				
Symbol	START_N	**Length**	1	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Unsigned8	**Access rights**	Write all	**PD Map**	PAD

Data Description

Data format	Binary	**Unit**	Record number
Value range	1 ... 250	**Resolution**	1

Example

Program record 26 must be processed.

Service	Write request	**Index**	0x5ff4	**Data byte**	0x1a
Command Code	0x8082	**Sub-index**	0		
Param. counter	4	**Length**	1		

2.11.5 START_N_GO

Start program at record N.
The record pointer is set to the corresponding program record and then the program is started.

Object Description

Index	0x5fe5				
Symbol	START_N_GO	**Length**	1	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Unsigned8	**Access rights**	Write all	**PD Map**	PAD

Data Description

Data format	Binary	**Unit**	Record number
Value range	1 ... 250	**Resolution**	1

Example

The program must be processed from record 50.

Service	Write request	**Index**	0x5fe5	**Data byte**	0x32
Command Code	0x8082	**Sub-index**	0		
Param. counter	4	**Length**	1		

2.11.6 TEACH_N

Take over current position in record N.
The command "POSA *current position*" is stored in record N.

Object Description

Index	0x5ff3				
Symbol	TEACH_N	**Length**	1	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Unsigned8	**Access rights**	Write all	**PD Map**	Not possible

Data Description

Data format	Binary	**Unit**	Record number
Value range	1 ... 250	**Resolution**	1

Example

The current position must be stored in record 70.

Service	Write request	**Index**	0x5ff3	**Data byte**	0x46
Command Code	0x8082	**Sub-index**	0		
Param. counter	4	**Length**	1		

2.11.7 CAM_MEM_P

Special command for COMPAX XX70.
Set and read curve memory pointer.

Object Description

Index	0x5fca				
Symbol	CAM_MEM_P	**Length**	2	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Unsigned16	**Access rights**	Read/write all	**PD Map**	Not possible

Data Description

Data format	Binary	**Unit**	Curve memory number
Value range	1 ... 5460	**Resolution**	1

2.11.8 CAM_MEM

Special command for COMPAX XX70.
Read and write the curve memory.
The curve memory number is defined by the current value of the curve memory pointer (CAM_MEM_P).
The curve memory pointer is automatically incremented after this object is accessed.

Object Description

Index	0x5fcb				
Symbol	CAM_MEM	**Length**	3	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Octet string	**Access rights**	Read/write all	**PD Map**	Not possible

Data Description

Data byte	1	2	3
Meaning	Record memory contents		
Assignment	MSB	...	LSB

2.11.9 VX

Enter or read COMPAX variable.
The corresponding variable is selected using the Sub-index (Sub-index = variable No.).
Sub-index = 40 addresses variable 0 of the COMPAX, with which all variables can be set to the same value.

Object Description

Index	0x5fcd				
Symbol	VX	**Length**	4	**Access groups**	0
Object code	Array	**Elements**	40	**Password**	0
Data type	Integer32	**Access rights**	Read/write all	**PD Map**	Not possible

Data description (P196 Bit 4 = "0")

Variable	Resolution	Variable	Resolution
001 .. 005	1 ⇔ 0.001	023 .. 029	1
006 .. 010	1	030 .. 034	1 ⇔ 0.001
011 .. 016	1 ⇔ 0.001	035 .. 036	1 ⇔ 0.000001
017 .. 020	1	037 .. 040	1 ⇔ 0.001
021 .. 022	1 ⇔ 0.000001		

Data description (P196 Bit 4 = "1") (New from Interbus-S - Software V2.12 onwards)

Variable	Resolution
001 .. 040	1 ⇔ 0.001

OBJECT_REQ and OBJECT_RSP for 8 byte process data channel

The two new objects OBJECT_REQ and OBJECT_RSP make it possible to access all communication objects (or certain elements) with a data width of ≤ 4 byte via the process data channel.
The object OBJECT_REQ can be assigned to the PAD channel (PAD3...PAD8):

- ♦permanently with P139 = 6276352 (=0x5fc500)
- ♦temporarily (*if P196 Bit 5 = 1*) as follows:

1. Save values of PAD3...PAD8 if necessary (save target values).
2. ***FreezePAD*** (STEUERWORT/CPX_STW Data byte1 Bit 1) **= 1**
   This freezes the previous PAD, no more value changes on the PAD are made.
3. Wait until ***AckToggle***(STATUSWORT/CPX_ZSW Data byte1 Bit0) has **changed its value**.
4. PAD3...PAD8 = 0
5. ***ObjectReqEnable*** (STEUERWORT/CPX_STW Data byte1 Bit0) **= 1**
6. Wait until ***AckToggle***(STATUSWORT/CPX_ZSW Data byte1 Bit0) has **changed its value**.
   PAD3...PAD8 will now be transferred to OBJECT_REQ
7. write and/or read all required objects
8. ***ObjectReqEnable*** (STEUERWORT/CPX_STW Data byte1 Bit0) **= 0**
9. Wait until ***AckToggle***(STATUSWORT/CPX_ZSW Data byte1 Bit0) has **changed its value**.
10. Reset PAD3...PAD8 to their original target values
11. ***FreezePAD*** (STEUERWORT/CPX_STW Data byte1 Bit 1) **= 0**
12. Wait until ***AckToggle***(STATUSWORT/CPX_ZSW Data byte1 Bit0) has **changed its value**.
    PAD3...PAD8 will now be transferred to the original objects

Object OBJECT_RSP can be assigned to the PED channel (PED3...PED8):

- ♦permanently with P135= 6276608 (0x5fc6000)
- ♦temporarily parallel with ***ObjectReqEnable*** (STEUERWORT/CPX_STW Data byte1 Bit0) **= 1** (*if P196 Bit 6=0* and *P196 Bit 5 = 1*))
  or separately via the object CONTROL

Example: P196 Bit 0..2 = "4" (8 Byte process data),
P196 Bit 7 ="0" (PD1/2 = STEUERWORT / STATUSWORT),
P196 Bit 5 = "1" (OBJECT_REQ and OBJECT_RSP can be temporarily assigned to PD),
P196 Bit 6 = "0" (OBJECT_RSP automatically assigns PED as answer to OBJECT_REQ)

Basic setting for PAD with P139="6322688" (LAGE_ZIEL=PAD3-6) and P142="6289152" (OUTPUT_WORD=PAD7-8) and for PED with P135="6317056" (LAGE_IST=PED3-6) and P138="6289408" (INPUT_WORD=PED7-8)
In this example you should make a backup copy of the contents of LAGE_ZIEL and OUTPUT_WORD before you switch to OBJECT_REQ.

PAD Master ⇒ COMPAX							
PAD1	PAD2	PAD3	PAD4	PAD5	PAD6	PAD7	PAD8
STEUERWORT							
1[1] ObjectReqEnable = "0"							
1[0] FreezePAD = "0"							

PED Master ⇐ COMPAX							
PED1	PED2	PED3	PED4	PED5	PED6	PED7	PED8
STATUSWORT							
1[0] AckToggle = "0" (angenommener Anfangszustand)							

FreezePAD = "1" COMPAX freezes previous PAD; does not read from PAD anymore

PAD Master ⇒ COMPAX							
PAD1	PAD2	PAD3	PAD4	PAD5	PAD6	PAD7	PAD8
STEUERWORT							
1[1] ObjectReqEnable = "0"							
1[0] FreezePAD = "1"							

Wait until COMPAX AckToggle = "1" message:

PED Master ⇐ COMPAX							
PED1	PED2	PED3	PED4	PED5	PED6	PED7	PED8
STATUSWORT							
1[0] AckToggle = "1"							

Describe PAD with "0" and set ObjectReqEnable = "1"

PAD Master ⇒ COMPAX							
PAD1	PAD2	PAD3	PAD4	PAD5	PAD6	PAD7	PAD8
STEUERWORT							
1[1] ObjectReqEnable = "1"							
1[0] FreezePAD = "1"							

PED Master ⇐ COMPAX							
PED1	PED2	PED3	PED4	PED5	PED6	PED7	PED8
STATUSWORT							
1[0] AckToggle = "0"							

Transfer PAD with OBJECT_REQ (P23=200% write)

PAD Master ⇒ COMPAX							
PAD1	PAD2	PAD3	PAD4	PAD5	PAD6	PAD7	PAD8
STEUERWORT							
1[1] ObjectReqEnable = "1"							
1[0] FreezePAD = "1"							

PED Master ⇐ COMPAX							
PED1	PED2	PED3	PED4	PED5	PED6	PED7	PED8
STATUSWORT							
1[0] AckToggle = "1"		POSITION_TARGET					

This procedure allows you to write and read several objects.

Switch back to cyclic process data:

Set ObjectReqEnable = "0"

PAD Master ⇒ COMPAX							
PAD1	PAD2	PAD3	PAD4	PAD5	PAD6	PAD7	PAD8
STEUERWORT							
1[1] ObjectReqEnable = "0"							
1[0] FreezePAD = "1"		POSITION_TARGET					

Answer from COMPAX by changing AckToggle

PED Master ⇐ COMPAX							
PED1	PED2	PED3	PED4	PED5	PED6	PED7	PED8
STATUSWORT							
1[0] AckToggle = "0"		POSITION_TARGET					

Set PAD to the original value and set FreezePAD = "0"

PAD Master ⇒ COMPAX							
PAD1	PAD2	PAD3	PAD4	PAD5	PAD6	PAD7	PAD8
STEUERWORT							
1[1] ObjectReqEnable = "0"							
1[0] FreezePAD = "0"		POSITION_TARGET					

PED Master ⇐ COMPAX							
PED1	PED2	PED3	PED4	PED5	PED6	PED7	PED8
STATUSWORT							
1[0] AckToggle = "1"		POSITION_TARGET					

OBJECT_REQ

Write or read communication via process data channel.
The object is selected with the 7 Bit-wide index pointer (please see footnote); the respective element of an object is selected with the sub-index. The highest bit of the data byte 1 determines the object access (0 = write; 1 = read).
The acknowledgement for a write-access or the answer for a read-access is passed onto the object OBJECT_RSP.
Every value change in OBJECT_REQ is acknowledged by the complementation of ***AckToggle*** (STATUSWORT/CPX_ZSW Data byte1 Bit 0). If the value change results in the writing or reading of an object, AckToggle is only complemented when the write/read process is completely finished.
If the index pointer does not equal 0, every value change in OBJECT_REQ results in writing or reading the selected objects or one of its elements. If an object is described with the same value in succession (z. B. POSR), the index pointer must be set to 0 and then reset to its original value.

Object Description

Index	0x5FC5				
Symbol	OBJECT_REQ	**Length**	6	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Octet string	**Access rights**	Write all	**PD Map**	PAD

Data Description

Data byte [Bit]	Assignment	Data byte[Bit]	Assignment
1[7]	0 = Object write; 1 = Object read	3	Object-Data byte 1
1 [6 ... 0]	Index(Object) pointer[1]	4	Object-Data byte 2
		5	Object-Data byte 3
2	Sub-index	6	Object-Data byte 4

OBJECT_RSP

Acknowledgement for a write-access or answer to a read-access to an object via OBJECT_REQ.

Object Description

Index	0x5FC6				
Symbol	OBJECT_RSP	**Length**	6	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Octet string	**Access rights**	Read all	**PD Map**	PED

Data Description

Data byte	Assignment -Write-access	Assignment –Read-access	Assignment -error
1	like OBJECT_REQ	like OBJECT_REQ	0xFF
2	like OBJECT_REQ	like OBJECT_REQ	like OBJECT_REQ
3	like OBJECT_REQ	Object data byte 1	Error number
4	like OBJECT_REQ	Object data byte 2	0x00
5	like OBJECT_REQ	Object data byte 3	0x00
6	like OBJECT_REQ	Object data byte 4	0x00

No.:	Meaning	No.:	Meaning
82	Wrong index(object)-pointer.	84	Object cannot be read.
83	Incorrect Sub-index.	85	Object cannot be written.

⇨ The specific COMPAX error messages are outlined in the COMPAX product manual.

[1] Please ensure that the index pointer and not the index of the required object is entered. You will find the respective index pointer for the onject in the object overview from page 10onwards.

2.12 Process data control

This function emulates the process data, which are transmitted via the process data channel, on the communication objects.
The COMPAX process data channel has a width of 2, 4 or 6 bytes; depending on the value in parameter 196. Each byte can be read and written by COMPAX.
The data read by COMPAX from the process data channel are called process output data (PA data). The data written by COMPAX into the process data channel are called process input data (PE data).
Through the emulation of COMPAX communication objects to the PE data, the latter are cyclically read on the process data channel. The PA data which are emulated to a COMPAX communication object cyclically describe this object.
The allocation of process data to certain communication objects is determined by the objects "PE_SELECT" and "PA_SELECT" (the objects "IN_SELECT" and "OUT_SELECT" are previous names).
Both objects "PED_INI" and "PAD_INI" determine which allocation shall be valid after Power-On (this corresponds to the settings through the COMPAX - Parameter).
The PA data can be enabled or disabled with the objects "OUT_ENABLE" or "PA_ENABLE".
After Power On the PA data are enabled!

2.12.1 PAD_Steuerung

The process output data can be used to cyclically write to the following COMPAX communication objects.

Object name	Description	Index		COMPAX -[2] Parameter	Byte	see
		dec	hex	P139 ... P142	Number	page
STEUERBYTE	Control byte	24526	0x5fce		1	17
STEUERWORT	Control word	24640	0x6040		2	19
CPX_STW	COMPAX control word/virtual inputs	24530	0x5fd2		2	18
CONTROL	Control commands	24553	0x5fe9		1	22
LAGE_ZIEL	Target position default	24698	0x607a		4	33
SPEED	Traverse speed	24556	0x5fec		2	41
OVERRIDE	Reduce traverse speed	24558	0x5fee		1	41
OUTPUT_WORD	16 Dig. Set/reset outputs	24567	0x5ff7		2	51
START_N	Execute program record N	24564	0x5ff4		1	60
START_N_GO	Program start beginning at record N	24549	0x5fe5		1	61
OBJECT_REQ	Read and write objects through PAD	24517	0x5fc5	6276352	6	65

Because the PAD channel has a max. length of 8 bytes, it is not possible to have simultaneous access all the objects listed here. This means you need to make an appropriate selection.

[2] Index * 256 + Sub-index

Setting the PAD:

◆ using the object "Process Output Data Description" (PA_SELECT; see page 73),

or

◆ using the COMPAX parameters P139, P140, P141, P142 (corresponds to the object PAD_INI; see page 76).

You may place each of the named objects on the PAD channel according to its required bytes.

Set the corresponding COMPAX parameter to the value given for the respective object (see table above).

Object:	Length in byte	Possible assignment in the PAD channel					
		PAD1	PAD2	PAD3	PAD4	PAD5	PAD6
PD-Length=8Byte (P196 Bit 0...2="4")		PAD3	PAD4	PAD5	PAD6	PAD7	PAD8
		P139	P140	P141		P142	
STEUERBYTE	1	▬	▬	▬		▬	
CONTROL	1	▬	▬	▬		▬	
OVERRIDE	1	▬	▬	▬		▬	
START_N	1	▬	▬	▬		▬	
START_N_GO	1	▬	▬	▬		▬	
STEUERWORT	2	▬	▬				
			▬	▬			
				▬	▬		
						▬	▬
CPX_STW	2	▬	▬				
			▬	▬			
				▬	▬		
						▬	▬
SPEED	2	▬	▬				
			▬	▬			
				▬	▬		
						▬	▬
OUTPUT_WORD	2	▬	▬				
			▬	▬			
				▬	▬		
						▬	▬
LAGE_ZIEL	4	▬	▬	▬	▬		
			▬	▬	▬	▬	
				▬	▬	▬	▬
OBJECT_REQ	6	▬	▬	▬	▬	▬	▬

⮡ Please ensure that there is no double assignment on the PAD channel.
Double assigning occurs, for example, if the POSITION_TARGET is in PAD1 - PAD4, and P141 is used to assign PAD3 again. The proper action in this case would be: LAGE_ZIEL on PAD1 - PAD4 via P139 = and P140 = P141 = 0!

⮡ PA_SELECT allows the channels to be freely assigned. You will find depictions of the various possibilities of parameters P139-P142.

PADs disable / enable

The PAD's can be individually disabled and enabled using the object "Enable Process Output Data" (PA_ENABLE see page 74). This means an object is only written with the value from the PAD channel if the corresponding PAD's are also enabled.

After "Power on" the PAD's are enabled, i.e., normally no setting needs to be made here.
After power on, the COMPAX parameters P139, P140, P141 and P142 initialise the objects PA_INI and PA_SELECT and thereby the PAD channel.
The PAD_INI object can be used to read and write these parameters.

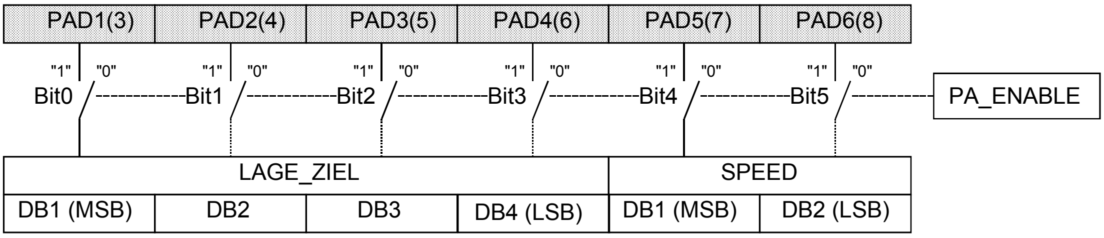
Notes

Note the following when configuring the PAD:

- After Power On, the PAD's are enabled if a valid configuration for the PAD is entered in the COMPAX parameters P139 ... P142.
- Note the length (number of bytes) of an object. An object can be represented on the PADs if the corresponding number of PAD bytes are free, i.e. not occupied by any other objects.
- Using the null object (Index and Sub-index = 0) or by setting the corresponding COMPAX parameter to "0", an object can again be removed (deleted) from the PAD channel.

Example assignments for the PAD's

The POSITION_TARGET and SPEED objects are represented on the PAD's.[3]



Object "PA_SELECT" (see page 73)

Using the object "PA_SELECT", the PAD assignment can be changed during operation:

Sub-index	Meaning	value	
		dec	hex
1	PAD length (not variable)	6	0x06
2	Object index assigned for PAD1 (3)	24698	0x607a
3	Object sub-index assigned for PAD1 (3)	0	0x00
4	Object index assigned for PAD2 (4)	0	0x0000
5	Object sub-index assigned for PAD2 (4)	0	0x00
6	Object index assigned for PAD3 (5)	0	0x0000
7	Object sub-index assigned for PAD3 (5)	0	0x00
8	Object index assigned for PAD4 (6)	0	0x0000
9	Object sub-index assigned for PAD4 (6)	0	0x00
10	Object index assigned for PAD5 (7)	24556	0x5fec
11	Object sub-index assigned for PAD5 (7)	0	0x00
12	Object index assigned for PAD6 (8)	0	0x0000
13	Object sub-index assigned for PAD6 (8)	0	0x00

After changing the PAD - assignment via object "PA_SELECT", PA_ENABLE is set to "0" to avoid an undefined condition. After a PAD change the PAD's must be manually enabled again using the PA_ENABLE object.

So that this setting of the PAD channel is already available upon power-up, the corresponding COMPAX parameters (P139 ... P142) resp. PAD_INI are to be assigned as follows:

Sub-index (Parameter)	Meaning	value	
		dec	hex
1 (P139)	Object index and sub-index assigned for PAD1 (3)	6322688	0x607a00
2 (P140)	Object index and sub-index assigned for PAD2 (4)	0	0x000000
3 (P141)	Object index and sub-index assigned for PAD3 (5)	0	0x000000
4 (P142)	Object index and sub-index assigned for PAD5 (7)	6286336	0x5fec00

[3] The PAD numbers in brackets are valid for a PD length of 8 bytes (P196 Bit 0...2 = "4").

2.12.2 PED_Steuerung

The process input data can be used to cyclically read from the following COMPAX communication objects.

Object name	Description	Index		COMPAX - Parameter[4]	Byte	see
		dec	hex	P135 ... P138	Number	page
STATUSBYTE	Status byte	24527	0x5fcf		1	17
STATUSWORT	Status word	24641	0x6041		2	20
CPX_ZSW	COMPAX status word	24531	0x5fd3		2	19
LAGE_IST	Actual position value	24676	0x6064		4	35
INPUT_WORD	Log. state of the 16 dig. inputs	24568	0x5ff8		2	51
OUTPUT_WORD	Log. state of the 16 dig. outputs	24567	0x5ff7		2	51
S3	Lag error	24570	0x5ffa		2	15
S4	Current traverse speed	24572	0x5ffc		2	45
S5	Current motor torque	24571	0x5ffb		2	30
OBJECT_RSP	COMPAX – answer to OBJECT_REQ	24518	0x5fc6	6276608	6	66

Because the PED channel has a max length of 8 byte, it is not possible to read all listed objects simultaneously. This means you need to make an appropriate selection.

Setting the PED

- via the object "Process Input Data Description",

or

- via the COMPAX parameters P135, P136, P137, P138 (corresponds to the object PED_INI).

You may place each of the named objects on the PED channel according to its required bytes.

Set the corresponding COMPAX parameter to the value given for the respective object (see table above).

	Length in byte	Possible assignment in the PED channel					
		PED1	PED2	PED3	PED4	PED5	PED6
PD-Length=8Byte (P196 Bit 0...2="4")		PED3	PED4	PED5	PED6	PED7	PED8
Object:		P135	P136	P137		P138	
STEUERBYTE	1	▬	▬	▬		▬	
STATUSWORT	2	▬	▬ ▬	▬ ▬	▬	▬	▬
CPX_ZSW	2	▬	▬ ▬	▬ ▬	▬	▬	▬
S3	2	▬	▬ ▬	▬ ▬	▬	▬	▬
S4	2	▬	▬ ▬	▬ ▬	▬	▬	▬
S5	2	▬	▬ ▬	▬ ▬	▬	▬	▬
INPUT_WORD	2	▬	▬ ▬	▬ ▬	▬	▬	▬
OUTPUT_WORD	2	▬	▬ ▬	▬ ▬	▬	▬	▬
LAGE_IST	4	▬	▬ ▬	▬ ▬ ▬	▬ ▬ ▬	▬ ▬	▬
OBJECT_RSP	6	▬	▬	▬	▬	▬	▬

Ensure that there is no double assignment in the PED channel.

Double assignment occur when e.g. LAGE_IST is on PED1 - PED4 and a further assignment of PED3 (5) is undertaken via P137.

The proper action in this case would be: POSITION_ACTUAL to PED1 - PED4 using P135 = 6317056 and P136 = P137 = 0!

[4] Index * 256 + Sub-index

The COMPAX parameters P135, P136, P137 and P138 initiate the object PI_SELECT and thereby the PED channel after the COMPAX is turned on.
These parameters can be read and written to using the PED_INI object.

PE_SELECT allows the channels to be freely assigned. You will find depictions of the various possibilities of parameters P135 - P138.

Notes

Note the following when configuring the PED:

- Note the length (number of bytes) of an object. An object can be represented on the PED's only if the corresponding number of PED bytes is available, i.e. are not occupied by any other objects.
- An object can be removed (deleted) again from the PED channel either by using the null object (Index and Sub-index = 0) or by setting the corresponding COMPAX parameter to "0".

Example for configuring the PED's

Represent the object INPUT_WORD, S3 and S4 on the PEDs.

PED1 (3)[5]	PED2 (4)	PED3 (5)	PED4 (6)	PED5 (7)	PED6 (8)
INPUT_WORD		S3		S4	
DB1 (MSB)	DB2 (LSB)	DB1 (MSB)	DB2 (LSB)	DB1 (MSB)	DB2 (LSB)

Configure PE_SELECT as follows:

Using the object "PE_SELECT", the PAD assignment can be changed during operation:

Sub-index	Meaning	value	
		dec	hex
1	PED length (not variable)	6	0x06
2	Object index assigned for PED1 (3)	24568	0x5ff8
3	Object sub-index assigned for PED1 (3)	0	0x00
4	Object index assigned for PED2 (4)	0	0x0000
5	Object sub-index assigned for PED2 (4)	0	0x00
6	Object index assigned for PED3 (5)	24570	0x5ffa
7	Object sub-index assigned for PED3 (5)	0	0x00
8	Object index assigned for PED4 (6)	0	0x0000
9	Object sub-index assigned for PED4 (6)	0	0x00
10	Object index assigned for PED5 (7)	24572	0x5ffc
11	Object sub-index assigned for PED5 (7)	0	0x00
12	Object index assigned for PED6 (8)	0	0x0000
13	Object sub-index assigned for PED6 (8)	0	0x00

To ensure that this setting of the PED channel is present upon Power On, the corresponding COMPAX parameters (P135 ... P138) or PED_INI must be assigned as follows:

Sub-index	Meaning	value	
(Parameter)		dec	hex
1 (P135)	Object index and sub-index assigned for PED1 (3)	6289408	0x5ff800
2 (P136)	Object index and sub-index assigned for PED2 (4)	0	0x000000
3 (P137)	Object index and sub-index assigned for PED3 (5)	6289920	0x5ffa00
4 (P138)	Object index and sub-index assigned for PED5 (7)	6290432	0x5ffc00

[5] The PED numbers in brackets are valid for a PD length of 8 bytes (P196 Bit 0...2 = "4").

2.12.6 PE_SELECT

Process Input Data - Description.
This parameter contains the data that define which process input data are emulated on which communication objects.
Communication objects which can be emulated on PED data are designated in the respective object descriptions.

Object Description

Index	0x6000				
Symbol	PE_SELECT			**Access groups**	0
Object code	Record			**Password**	0
Data type	PDB structure	**Access rights**	Read/write all	**PD Map**	not possible

Data Description

Sub-index	Assignment	Data type	Length
1	Length of the process data channel	Unsigned 8	1
2	Device parameter index, which assigns 1st (3rd)[6] PE data byte	Unsigned16	2
3	Device parameter sub-index, which assigns 1st (3rd) PE data byte	Unsigned 8	1
4	Device parameter index, which assigns 2nd (4th) PE data byte	Unsigned16	2
5	Device parameter sub-index, which assigns 2nd (4th) PE data byte	Unsigned 8	1
6	Device parameter index, which assigns 3rd (5th) PE data byte	Unsigned16	2
7	Device parameter sub-index, which assigns 3rd (5th) PE data byte	Unsigned 8	1
8	Device parameter index, which assigns 4th (6th) PE data byte	Unsigned16	2
9	Device parameter sub-index, which assigns 4th (6th) PE data byte	Unsigned 8	1
10	Device parameter index, which assigns 5th (7th) PE data byte	Unsigned16	2
11	Device parameter sub-index, which assigns 5th (7th) PE data byte	Unsigned 8	1
12	Device parameter index, which assigns 6th (8th) PE data byte	Unsigned16	2
13	Device parameter sub-index, which assigns 6th (8th) PE data byte	Unsigned 8	1

Example

The object "LAGE_IST" must be emulated on the PE data.
PD-Length ≤ 6: The 1st data byte from "LAGE_IST" assigns the 3rd byte of the PE data; the 4th ,5th , 6th data byte the others.
PD-Length = 8: The 1st data byte from "LAGE_IST" assigns the 5th byte of the PE data; the 6th ,7th , 8th data byte the others.

Service	Write request	**Index**	0x6000	**1. data byte**	0x60
Command Code	0x8082	**Sub-index**	0x06	**2. data byte**	0x64
Param. counter	5	**Length**	2		

[6] The PD numbers in brackets are valid for a PD length of 8 bytes (P196 Bit 0...2 = "4").

2.12.7 PA_SELECT

Process Output Data - Description.
This parameter contains the data that define which process output data are emulated on which communication objects. Communication objects which can be emulated on PA data are designated in the respective object descriptions.

Object Description

Index	0x6001				
Symbol	PA_SELECT			**Access groups**	0
Object code	Record			**Password**	0
Data type	PDB structure	**Access rights**	read/write all	**PD Map**	not possible

Data Description

Sub-index	Assignment	Data type	Length
1	Length of the process data channel	Unsigned 8	1
2	Device parameter index, which assigns 1st (3rd)[7] PA data byte	Unsigned16	2
3	Device parameter sub-index, which assigns 1st (3rd) PA data byte	Unsigned 8	1
4	Device parameter index, which assigns 2nd (4th) PA data byte	Unsigned16	2
5	Device parameter sub-index, which assigns 2nd (4th) PA data byte	Unsigned 8	1
6	Device parameter index, which assigns 3rd (5th) PA data byte	Unsigned16	2
7	Device parameter sub-index, which assigns 3rd (5th) PA data byte	Unsigned 8	1
8	Device parameter index, which assigns 4th (6th) PA data byte	Unsigned16	2
9	Device parameter sub-index, which assigns 4th (6th) PA data byte	Unsigned 8	1
10	Device parameter index, which assigns 5th (7th) PA data byte	Unsigned16	2
11	Device parameter sub-index, which assigns 5th (7th) PA data byte	Unsigned 8	1
12	Device parameter index, which assigns 6th (8th) PA data byte	Unsigned16	2
13	Device parameter sub-index, which assigns 6th (8th) PA data byte	Unsigned 8	1

Example

The object "LAGE_ZIEL" must be emulated on the PA data.
PD-Length ≤ 6: The 1st data byte from "LAGE_ZIEL" assigns the 2nd byte of the PA data; the 3rd ,4th , 5th data byte the others.
PD-Length = 6: The 1st data byte from "LAGE_ZIEL" assigns the 4th byte of the PA data; the 5th ,6th , 7th data byte the others.

Service	Write request	**Index**	0x6001	**1. data byte**	0x60
Command Code	0x8082	**Sub-index**	0x04	**2. data byte**	0x7a
Param. counter	5	**Length**	2		

[7] The PD numbers in brackets are valid for a PD length of 8 byte (P196 Bit 0...2 = "4").

2.12.8 PA_ENABLE

Enable process output data.
Each bit of this parameter is associated with a byte of the process output data channel.
Meaning:

Bit = 0 the corresponding process data value is disabled
Bit = 1 the corresponding process data value is enabled

If an object takes up several bytes on the PA data channel, the logic state of the bit which is associated with the first byte of this object is the one used, and the other associated bits are not relevant.

Object Description

Index	0x6002				
Symbol	PA_ENABLE	**Length**	1	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Octet string	**Access rights**	Read/write all	**PD Map**	Not possible

Data Description

Bit	Assignment	Bit	Assignment
7	="1": autom. Acceptance of a modified LAGE_ZIEL of the PA data is disabled	3	4th (6th) byte of the PA data
6	None	2	3rd (5th) byte of the PA data
5	6th (8th)[8] byte of the PA data	1	2nd (4th) byte of the PA data
4	5th (7th) byte of the PA data	0	1 (3rd). byte of the PA data

Bit	Function	Data byte	Function
= 0 (FALSE)	Process data value disabled	= 1 (TRUE)	Process data value enabled

Example

The process data value which assigns the 3rd (5th) byte of the PA data (and other if applicable) should be enabled.
The other process data values are disabled.

Service	Write request	**Index**	0x6002	**Data byte**	0x04
Command Code	0x8082	**Sub-index**	0		
Param. counter	4	**Length**	1		

[8] The PD numbers in brackets are valid for a PD length of 8 bytes (P196 Bit 0...2 = "4").

2.12.9 PED_INI

Initialising the process input data description.
This object contains the data that define which process input data are emulated on which communication objects after COMPAX is switched on.
Communication objects which can be emulated on PED data are designated in the respective object descriptions.

Object Description

Index	0x5fd0				
Symbol	PED_INI	**Length**	3	**Access groups**	0
Object code	Array	**Elements**	4	**Password**	0
Data type	Octet String	**Access rights**	read/write all	**PD Map**	not possible

Data Description

Sub-index	PE assignment	Data byte 1	Data byte 2	Data byte 3
1	Stipulate object which will assign the 1st (3rd)[9] PE data byte.	Object Index (High Byte)	Object Index (Low Byte)	Object Sub-index
2	Stipulate object which will assign 2nd (4th) PE data byte.	Object Index (High Byte)	Object Index (Low Byte)	Object Sub-index
3	Stipulate object which will assign 3rd (5th) PE data byte.	Object Index (High Byte)	Object Index (Low Byte)	Object Sub-index
4	Stipulate object which will assign 5th (7th) PE data byte.	Object Index (High Byte)	Object Index (Low Byte)	Object Sub-index

Example

The object "INPUT-WORD" must be emulated on the PE data after 'Power On'.
The 1st data byte from "INPUT-WORD" assigns the 5th (7th) byte of the PE data; the 2nd data byte the others.

Service	Write request	**Index**	0x5fd0	**1. data byte**	0x5f
Command Code	0x8082	**Sub-index**	0x04	**2. data byte**	0xf8
Param. counter	5	**Length**	3	**3. data byte**	0

[9] Die PD - Nummern in der Klammer gelten bei einer PD-Length von 8 Byte (P196 Bit 0...2 = "4").

2.12.10 PAD_INI

Initialising the Process Output Data - Description.
This object contains the data that define which process output data are emulated on which communication objects after COMPAX is switched on.
Communication objects which can be emulated on PA data are designated in the respective object descriptions.

Object Description

Index	0x5fd1				
Symbol	PAD_INI	**Length**	3	**Access groups**	0
Object code	Array	**Elements**	4	**Password**	0
Data type	Octet String	**Access rights**	read/write all	**PD Map**	not possible

Data Description

Sub-index	PA assignment	Data byte 1	Data byte 2	Data byte 3
1	Stipulate object which will assign 1st (3rd)[10] PA data byte.	Object Index (High Byte)	Object Index (Low Byte)	Object Sub-index
2	Stipulate object which will assign 2nd (4th) PA data byte.	Object Index (High Byte)	Object Index (Low Byte)	Object Sub-index
3	Stipulate object which will assign 3rd (5th) PA data byte.	Object Index (High Byte)	Object Index (Low Byte)	Object Sub-index
4	Stipulate object which will assign 5th (7th) PA data byte.	Object Index (High Byte)	Object Index (Low Byte)	Object Sub-index

Example

The object "OUTPUT-WORD" must be emulated on the PE data after 'Power On'.
The 1st data byte from "OUTPUT_WORD" assigns the 3rd (5th) byte of the PA data; the 2nd data byte the others.

Service	Write request	**Index**	0x5fd1	**1. data byte**	0x5f
Command Code	0x8082	**Sub-index**	0x03	**2. data byte**	0xf7
Param. counter	5	**Length**	3	**3. data byte**	0

[10] Die PD - Nummern in der Klammer gelten bei einer PD-Length von 8 Byte (P196 Bit 0...2 = "4").

2.12.3 IN_SELECT

Process Input Data - Description.
This object defines, which communication object or communication object element will be depicted on the 1st (3rd)[11] byte (and if applicable, the subsequent bytes) process input data of the process data channel..

Object Description

Index	0x5ffd				
Symbol	IN_SELECT			**Access groups**	0
Object code	Record			**Password**	0
Data type	15	**Access rights**	read/write all	**PD Map**	not possible

Data Description

Sub-index	Assignment	Data type	Length
1	Device parameter index, which assigns 1st (3rd) PE data byte	Unsigned16	2
2	Device parameter sub-index, which assigns 1st (3rd) PE data byte	Unsigned 8	1

Example

The object "INPUT-WORD" must be emulated on the PE data.
The 1st data byte from "INPUT-WORD" assigns the 1st (3rd) byte of the PE data; the 2nd data byte the others.

Service	Write request	**Index**	0x5ffd	**1. data byte**	0x5f
Command Code	0x8082	**Sub-index**	0x00	**2. data byte**	0xf8
Param. counter	5	**Length**	3	**3. data byte**	0

2.12.4 OUT_SELECT

Process Output Data - Description.
This object defines which communication object or element of a communications object is emulated on the 1. byte (and if necessary on the following bytes) of the process output data of the process data channel.

Object Description

Index	0x5ffe				
Symbol	OUT_SELECT			**Access groups**	0
Object code	Record			**Password**	0
Data type	15	**Access rights**	read/write all	**PD Map**	not possible

Data Description

Sub-index	Assignment	Data type	Length
1	Device parameter index, which assigns 1st (3rd) PA data byte	Unsigned16	2
2	Device parameter sub-index, which assigns 1st (3rd) PA data byte	Unsigned 8	1

Example

The object "OUTPUT-WORD" must be emulated on the PA data.
The 1st data byte from "OUTPUT-WORD" assigns the 1st (3rd) byte of the PA data; the 2nd data byte the others.

Service	Write request	**Index**	0x5ffe	**1. data byte**	0x5f
Command Code	0x8082	**Sub-index**	0x00	**2. data byte**	0xf7
Param. counter	5	**Length**	3	**3. data byte**	0

[11] Die PD - Nummern in der Klammer gelten bei einer PD-Length von 8 Byte (P196 Bit 0...2 = "4").

2.12.5 OUT_ENABLE

Enable process output data.
This parameter is allocated to the 1st (3rd)[12] byte of the process output data channel.
Meaning:

"OUT_ENABLE" = FALSE	the corresponding process data value is disabled
"OUT_ENABLE" = TRUE	the corresponding process data value is enabled

Object Description

Index	0x5fff				
Symbol	OUT_ENABLE	**Length**	1	**Access groups**	0
Object code	Simple var.			**Password**	0
Data type	Boolean	**Access rights**	read/write all	**PD Map**	not possible

Data Description

Data byte	Function	Data byte	Function
= 0x00 (FALSE)	Process data value disabled	= 0xff (TRUE)	Process data value enabled

Example

The process data value which assigns 1st (3rd) byte of the PA data (and if necessary the others) should be enabled.

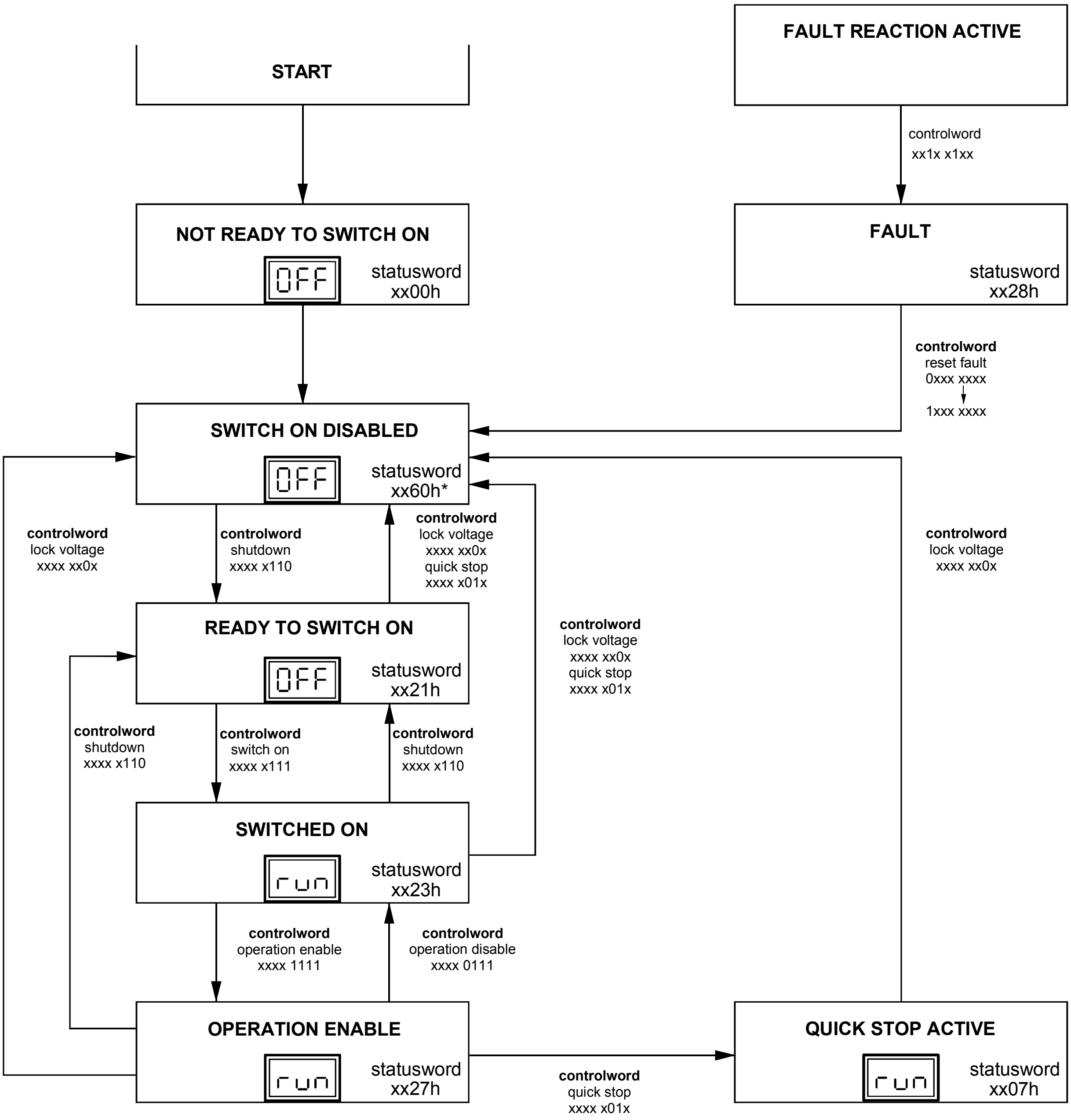
Service	Write request	**Index**	0x5fff	**Data byte**	0xff
Command Code	0x8082	**Sub-index**	0		
Param. counter	4	**Length**	1		

[12] Die PD - Nummern in der Klammer gelten bei einer PD-Length von 8 Byte (P196 Bit 0...2 = "4").

3. DRIVECOM-PROFILE 22, from COMPAX Software-Version 3.01

From program version V3.01onwards, the operating type 'DRIVECOM-PROFIL' is available in all COMPAX variants. It can be activated **via P190=22**. Please note that P190 must be at this value by 'Power-On', in order that the turn on procedure defined in the DRIVECOM profile can run completely (this varies from the normal COMPAX behaviour after 'Power-on').
According to DRIVECOM specifications there are a series of statuses that are accepted by the device in sequence. The corresponding status equipment is integrated in the COMPAX. The conditions are outlined in diagrams and tables in the documentation 'DRIVECOM-PROFILE technology'. The following description of device statuses is therefore only given as a supplement to the DRIVECOM documentation:

Conditions diagram



➪ The COMPAX command OUTPUT A0=... may not be used for active operating type 'DRIVECOM-PROFIL' .

Status	Description
NICHT-EINSCHALTBEREIT:	◆ Self-test is running ◆ Initialisation is running ◆ Drive function is disabled ◆ All functions are disabled
EINSCHALTSPERRE:	◆ Software/hardware initialisation is terminated ◆ Communication via all interfaces is enabled ◆ Modifications to parameters, variables, records is possible ◆ Drive function is disabled (motor dead) ◆ Display shows 'OFF'
EINSCHALTBEREIT:	◆ Communication through all interfaces is enabled ◆ Modifications to parameters, variables, records is possible ◆ Drive function is disabled (motor dead) ◆ Display shows 'OFF'
EINGESCHALTET:	◆ Communication through all interfaces is enabled ◆ Modifications to parameters, variables, records is possible ◆ Drive function is disabled (but motor has current) ◆ Display shows 'run'
BETRIEB-FREIGEGEBEN:	◆ Communication through all interfaces is enabled ◆ Modifications to parameters, variables, records is possible ◆ Drive function is enabled (motor has current) ◆ Display shows 'run'
SCHNELLHALT-AKTIV:	◆ Communication through all interfaces is enabled ◆ Modifications to parameters, variables, records is possible ◆ Drive implements the STOP function (motor has current) ◆ Display shows 'run'
STÖRUNG:	◆ Communication through all interfaces is enabled ◆ Modifications to parameters, variables, records is possible ◆ Drive function is disabled ◆ Display shows 'Exx'
STÖRUNGSREAKTION:	◆ Error handling only, no static device status

4. COMPAX Parameters for the Interbus-S

No.	Meaning	Minimum value	Default value	Maximum value	When valid
P190	Set operating type "DRIVECOM profile 22" ="0": DRIVECOM profile defined turn on procedure **inactive** ="22": DRIVECOM profile defined turn on procedure **active**				Power on
P191	Bus time-out ="0": no response, except error message E73, during a time-out ="1": stop with E73 and shut down during activation of holding brake				VP
P193	Pop-up messages ="1": autom. Error message ="2": autom. "position reached" - message ="4": autom. comparator switch points report				immediately
P196	Process data length & protocol **Bit 0..2 = 0** PD-Length 1 word, PED/PAD=INPUT/OUTPUT_WORD **Bit 0..2 = 1** PD-Length 1 word **Bit 0..2 = 2** PD-Length 2 words **Bit 0..2 = 3** PD-Length 3 words **Bit 0..2 = 4** PD-Length 4 words; no PCP communication **Bit 4 = 0** Resolution from V1...V40 like P1...P40 **Bit 4 = 1** Resolution from V1...V40: 1 ⇔ 0.001 **Bit 5 = 0** OBJECT_REQ/RSP can **not** be assigned to PD temp. **Bit 5 = 1** OBJECT_REQ/RSP can be assigned to PD temp. **Bit 6 = 0** OBJECT_RSP autom. assigns PED3..8 if ObjectReqEnable = 1 **Bit 6 = 1** OBJECT_RSP must be assigned to PED3..8 explicitly **Bit 7 = 0** PED/PAD1..2=STATUS/STEUERWORT if PD-Length = 4 **Bit 7 = 1** PED/PAD1..2=CPX_ZSW/CPX_STW if PD-Length = 4				Power on
P135	Object index and sub-index which assigns the 1. (3.)[13] PE data byte after turning on. **Value:** Index • 256 + Sub-index	0	0	16777215	Power on
P136	Object index and sub-index which assigns the 2. (4.) PE data byte after turning on. **Value:** Index • 256 + Sub-index	0	0	16777215	Power on
P137	Object index and sub-index which assigns the 3. (5.) PE data byte after turning on. **Value:** Index • 256 + Sub-index	0	0	16777215	Power on
P138	Object index and sub-index which assigns the 5. (7.) PE data byte after turning on. **Value:** Index • 256 + Sub-index	0	0	16777215	Power on
P139	Object index and sub-index which assigns the 1. (3.) PA data byte after turning on. **Value:** Index • 256 + Sub-index	0	0	16777215	Power on
P140	Object index and sub-index which assigns the 2. (4.) PA data byte after turning on. **Value:** Index • 256 + Sub-index	0	0	16777215	Power on
P141	Object index and sub-index which assigns the 3. (5.) PA data byte after turning on. **Value:** Index • 256 + Sub-index	0	0	16777215	Power on
P142	Object index and sub-index which assigns the 5. (7.) PA data byte after turning on. **Value:** Index • 256 + Sub-index	0	0	16777215	Power on
P203	Assigning status S16 and S17 to CPX_ZSW Bit 0=0 CPX_ZSW has a standard assignment Bit 0=1 S16, S17 assigns CPX_ZSW Bit 1=0 LAGE_IST has a standard assignment Bit 1=1 S13 assigns LAGE_IST				immediately
P221	Standard functions of the digital inputs accessible from the CONTROLWORD object. Physical inputs freely available. Is written by the object INPUT_MASK data byte 2.	0	0	255	immediately
P222	Standard functions of the digital inputs E9...E16 accessible from the CONTROLWORD object. Physical inputs freely available. Is written by the object INPUT_MASK data byte 1.	0	0	255	immediately
P223	Outputs A1...A8 are accessible from object OUTPUT_WORD. Written by the object OUTPUT_MASK data byte 2.	0	0	255	immediately
P224	Outputs A9...A16 are accessible from object OUTPUT_WORD. Written by the object OUTPUT_MASK data byte 1.	0	0	255	immediately

[13] Die PD - Nummern in der Klammer gelten bei einer PD-Length von 8 Byte (P196 Bit 0...2 = "4").

5. COMPAX Interbus-S Error messages

No.	Cause	Remedy / Causes	Acknowledge with	No power to drive
E73	Time-out error The error response is influenced with P191.	Re-send the characters	[1]	nein[2]

[1] No acknowledgement necessary; the error message is cancelled after the next errorless transmission.

[2] Depends on P191.

6. Index