hilscher

COMPETENCE IN COMMUNICATION

Protocol interface manual

InterBus Master

Hilscher Gesellschaft für Systemautomation mbH
Rheinstraße 15
D-65795 Hattersheim
Germany

Tel. +49 (0) 6190 / 9907 - 0
Fax. +49 (0) 6190 / 9907 - 50

Sales: +49 (0) 6190 / 9907 - 0
Hotline and Support: +49 (0) 6190 / 9907 - 99

Sales e-mail: sales@hilscher.com
Hotline and Support e-mail: hotline@hilscher.com

Homepage: http://www.hilscher.com

Index	Date	Version	Chapter	Revision
1	27.05.96	V1.200	all	created
2	27.11.97	V1.302	all	adaption of new header definitions and new protocol interface manual structure
3	11.02.98	V2.000	all	upgrading to the new generation 4 master protocol software with extended and changed diagnostic
4	29.06.98	V2.008	all	description of the bus parameter and slave parameter download function PCP message interface
5	03.09.98	V2.00A	all	extention of the Error Codes in PCP Protocol
6	14.03.99	V2.018	all	correction of command in IBM_Slave_Diag message to 66dec. documentation of Get_Configuration and Set_Configuration function documentation of Delete_Database function new Protocol-parameter 'usWatchdogtime' implemented explanation of the PCP-Server services new chapter: get DEVICE operative without SyCon Online configuration change functionality by using IBM_Download extended explanation of Err_Event PCP capable devices now downloadable via dual-port IBM_Download procedure
6	26.04.99	V2.022	all	new data length for the lengthcodes 12 and 13 , 1 and 2 Bits now
7	08.06.99	V2.026		extention of the slave parameter header functionality, alternative grouping
8	28.09.99	V2.037		- extention of explanation of error numbers in chapter 3.3 Error number 33,34. - Distinction of 2Kilobyte and 8Kilobyte DEVICEs and the addesses of protocol tates and protocol parameter - new statistic protocol states implemented - change maximum PCP object length from 237 to 240
9	14.08.00			- new chapter Get_ObjectDictionary and Identify
10	13.07.01	V2.080		- new return errors after downloading a DEVICE parameter data set and master parameter data set. - new parameter bMinimizeSyncJitter in protocol parameter area - new command IBM_Control_Active_Configuration - new variable Ext_Global_Bits in the global state information field - further error codes in the chapter error code in the PCP protocol - remove error 37 in global state field description. This error can not occur at this position - addtional error code header in a negative PCP response message with msg.f = 0x81

Although this protocol implementation has been developed with great care and intensively tested, Hilscher Gesellschaft für Systemautomation mbH cannot guarantee the suitability of this protocol implementation for any purpose not confirmed by us in writing.

Guarantee claims shall be limited to the right to require rectification. Liability for any damages which may have arisen from the use of this protocol implementation or its documentation shall be limited to cases of intent.

We reserve the right to modify our products and their specifications at any time in as far as this contributes to technical progress. The version of the manual supplied with the protocol implementation applies.

1 Introduction

This manual describes the user interface of InterBus master for the communication interfaces and the communication module. The aim of this manual is to support the integration of these devices into own applications based on device driver functions or direct access into the dual-port memory.

The general mechnism for the data transfer, for example how to send and receive a message or how to do a warmstart, is protocol independent and for each hardware the same procedure and is described therefore in the 'general definitions' toolkit manual.

All parameters and data have basically the description LSB/MSB. This corresponds to the convention of the Microsoft-C-compiler. The storage format of word oriented send and receive process data of the handled I/O devices is configureable.

1.1 Protocol Signification

To manage the InterBus protocol 2 tasks are involved in the system. Therefore following entries for the protocol signification in the variables `TaskiName` are done:

```
Task2Name: 'PLC-TASK'
Task3Name: 'IBM     '
```

1.2 The Process Data Interface

The DEVICE handles 512 bytes send and 512 bytes receive process data in the lower kbyte of the dual-port memory for the InterBus. To exchange the data between the DEVICE and the HOST use the device driver function `DevExchangeIO()`or read and write directly into these locations.
After calling the function `DevExchangeIO()`the function decides on its own which handhake mechanism has to be used to read and write the process data in the rigth manner from and to the DEVICE. If these locations are accessed directly without using the device driver functionality, than you have to use the right handshake mechanism to ensure that the data is overgiven safety and valid. See chapter 'IO Communication with a Process Image' in the 'toolkit general definitions' manual.

Parameter	Address	Description
SndPd	000h	Send Process Data (HOST → DEVICE → Network)
RecvPd	200h	Receive Process Data (Network → DEVICE → HOST)

2 Protocol Parameters

Some important parameters can be handed over to the DEVICE online from the HOST application. They have a higher priority than the static parameters in the internal FLASH memory usually configured by SyCon configuration tool. This ensures for example on each startup of the DEVICE the same behavior for the process data handshake, don't caring what is configured by SyCon.

2.1 Using Device Driver Function to Write

To hand over these parameters use the device driver function `DevPutTaskParameter()`. For parameter `usNumber` use value 2, because the parameters must handed over to the task 2. For parameter `usSize` use value 6, to fix the length of the structure. Point the parameter `pvData` to the following structure below.

```
typedef struct IBM_PLC_PARAMETERStag {
  unsigned char   bMode;
  unsigned char   bReserved;
  unsigned char   bFormat;
  unsigned short  usWatchDogTime;
  unsigned char   bMinimizeSyncJitter;
  unsigned char   abReservedA[2];
  unsigned char   abReservedB[8];
} IBM_PLC_PARAMETER;

/* values for bMode */
#define IBM_SET_MODE_BUSSYNC_DEVICE_CONTROLLED  0
#define IBM_SET_MODE_BUFFERED_DEVICE_CONTROLLED 1
#define IBM_SET_MODE_UNCONTROLLED               2
#define IBM_SET_MODE_BUFFERED_HOST_CONTROLLED   3
#define IBM_SET_MODE_BUSSYNC_HOST_CONTROLLED    4
/* values for bFormat */
#define IBM_FORMAT_MOTROLA 0x01
#define IBM_FORMAT_INTEL   0x00
```

After setting up your values in the structure and copy it with `DevPutTaskParameter()` into the assigned dual-port memory area, the warmstart command must be performed with the `DevReset()`function. The most important parameter in this function `usMode` must be set up to 3 = WARMSTART. After the warmstart is finished without error the new parameters are active.

2.2 Direct Write Access in Dual-port Memory

First the parameters must be written down into the corresponding area of the dual-port memory. Then a warmstart command must be activated by setting the `Init` bit in the variable `DevFlags`. Then the DEVICE will set them valid (see the chapter 'initialization of the DEVICE' in the toolkit manual 'general definitions' for handle of the init procedure).

structure element	type	address 2K DPM	address 8K DPM	parameter
bMode	byte	6C0H	1EC0H	process data delivery (0,1,2,3,4)
bFormat	byte	6C2H	1EC2H	storage format of word oriented process data (0,1)
usWatchDogTime	word	6C3H	1EC3H	HOST-supervision time in multiples of a msec.
bMinimizeSyncJitter	byte	6C5H	1EC5H	minimize the jitter in bus sychronous process data handshake mode

2.3 Explanation of the Protocol Parameters

The first parameter `bMode` fixes the handshake mode for the process data. The explanation of the different modes and their behavior can be read in the toolkit manual 'general definitions'.
The second parameter `bFormat` changes the storage format of word oriented process data from MSB/LSB to LSB/MSB convention and vice versa. In case of analog InterBus devices for example which have normally word defined process data, the values are swapped in their layout format to be compatible to different data interpreting HOST systems.

The parameter `usWatchDogTime` fixes the time in multiples of 1msec. the DEVICE has to supervise the HOST program if it has started the HOST-watchdog functionality once.

If set to value 1 the parameter `bMinimizeSyncJitter` reduces the jitter of the Bus Sychronous Hot Controlled process data handshake in general. The jitter-time the outputs are set valid to the InterBus Slave devices after initiating the handshake is then less or equal 35µsec for each cycle. So the parameter has only influence on the DEVICEs behavior, if the handshake mode Bus Sychronous Host Controlled is configured at the same time. Using this mode needs carful HOST programming else the card could perform a resest if misused.
To reduce the jitter in the bus sychronous mode the DEVICE itself deactivates its internal cyclic timer routine which normally triggers the hardware-watchdog, updates further cells in the dual-port memory and counts task timers and can cause if interrupting the standard sychronous handshake a jitter of 300µsec. Disabling the cyclic timer routine within the DEVICE begins with the first HOST initiated handshake. Instead of calling the timer routine by a hardware timer now, in minimized jitter mode calling the timer routine is coupled directly to the HOST controlled handshake. After triggering the handshake from HOST side, the latest given outputs are driven to the slaves and their inputs are collected and copied back into the process data input area and the handshake is confirmed.

But then the DEVICE now calls the timer routine afterwards. Because this takes up to 300µsec and keeps the DEVICE busy, a next process data handshake shouldn't be initiated by the HOST faster then 500µsec after getting the confirmation of the last one.
Following problems can now occur :

- if the HOST application stops to trigger, the DEVICE will perform an automatic reset after 1.5seconds, because the hardware watchdog isn't triggered any more within the DEVICE. To prevent this the HOST application can set the NotReady bit in the cell DevFlags to stop communication or calls the function DevSetHostState() together with the parameter HOST_NOT_READY. Then the DEVICE will activate the cyclic timer routine again.
- if the HOST application starts the next process data handshake faster than the DEVICE needs to execute the timer routine which is 300µsec, the handshake jitter will be extended to at least 300µsec.

3 Protocol States

The protocol states built by the DEVICE form the diagnostic interface between the HOST and the InterBus.

The first structure is a statistic information field. This field informs for example about the number of driven process data cycles as well as the number of defective cycles. So this field can be used then on HOST side to calculate the transmission quality for example.
The second established structure informs about global bus states as well as individual states of the managed device stations. The structure will be actualized event driven on every change value in it. To hold the information preferably compact, the devicespecific informations are held in state bit fields. The first 4 state variables in the structure inform about global master and network state informations. After this field an unused reserved area of 26 byte is following. The following first 16 bytes characterize each device as configured and handled. The next 16 bytes characterizes each device as active or inactive in the network, followed by 16 bytes which serve to refer the diagnostic bit of each device.

3.1 Using Device Driver Functions

Use the device driver function `DevGetTaskState()` to read the states. For parameter `usNumber` use value 1 for structure 1 and value 2 for structure 2. For parameter `usSize` use value 64, which is the length of each structure below. Point `pvData` to the corresponding defined structure in your HOST application:

Structure 1:

```
typedef struct IBM_STATISTICStag {
  unsigned char abReservedI[16];     /* reserved area 1 *
  unsigned long ulCycleCnt;          /* number of driven data cycles */
  unsigned long ulDefectiveCycleCnt; /* number of defective data cycles */
  unsigned long ulDiagCycleCnt;      /* number of driven diagno. cycles */
  unsigned long ulDeviceErrorCnt;    /* number of reported device errors */
  unsigned char abReservedII[16];    /* reserved area 2 */
  unsigned char abReservedIII[16];   /* reserved area 3 */
} IBM_STATISTICS;
```

Structure 2:

```
typedef struct IBM_DIAGNOSTICStag {
  struct
  {
    unsigned char bCtrl     : 1;
    unsigned char bAClr     : 1;
    unsigned char bNonExch  : 1;
    unsigned char bPrhlErr  : 1;
    unsigned char bEvent    : 1;
    unsigned char bNRdy     : 1;
    unisgned char bI1Err    : 1;
    unisgned char bI2Err    : 1;
  } bGlobalBits;

  unsigned char   bIBM_State;

  struct
  {
    unsigned char bErr_Dev_Adr;
    unsigned char bErr_Event;
  } tError;


  unsigend short  usNumOfDefectiveDataCycles;
  unsigned short  usNumOfNetworkReinits;
  unsigned char   bExtGlobalBits;
  unsigned char   abReserved[7];


  unsigned char   abSl_cfg[16];
  unsigned char   abSl_state[16];
  unsigned char   abSl_diag[16];
} IBM_DIAGNOSTICS;
```

3.2 Direct Read Access in Dual-port Memory

Read the structure directly from the following dual-port memory location:

variable	type	address 2K DPM	address 8K DPM	variable
Reserved-I	16 bytes	700H	1F00H	reserved byte area for further use
Cycle_Cnt	long	710H	1F10H	number of driven process data cycles
Defective_Cycle_Cnt	long	714H	1F14H	number of defective process data cycles
Diag_Cycle_Cnt	long	718H	1F18H	number of driven diagnostic cycles
Device_Diag_Cnt	long	71CH	1F1CH	number of device related and reported diagnostics
Reserved-II	16 bytes	720H	1F20H	reserved byte area for further use
Reserved-III	16 bytes	730H	1F40H	reserved byte area for further use

variable	type	address 2K DPM	address 8K DPM	short signification
Global_Bits	1 byte	740H	1F40H	collective global error and status bits
IBM_State	1 byte	741H	1F41H	main state of the master system
Err_Dev_Adr	1 byte	742H	1F42H	error source and location
Err_Event	1 byte	743H	1F43H	corresponding error number
Defective_Datacycles	1 word	744H-745H	1F44H-1F45H	number of defective data cycles
Network_Rescans	1 word	746H-747H	1F46H-1F47H	number of necessary network rescans and network reinitializations
Ext_Global_Bits	1 byte	748H	1F48H	extended collective error and status bits
reserved	7 bytes	748H-74FH	1F48H-1F4FH	reserved for further use
Sl_cfg	16 bytes	750H-75FH	1F50H-1F4FH	see the table below
Sl_state	16 bytes	760H-76FH	1F60H-1F6FH	see the table below
Sl_diag	16 bytes	770H-77FH	1F70H-1F7FH	see the table below

3.3 Explanation of the Protocol States

- Global_Bits

D7	D6	D5	D4	D3	D2	D1	D0
I2ERR	I1ERR	NRDY	EVE	PRHL	NEXC	ACLR	CTRL

CONTROL-ERROR:
configuration or
heavy runtime error

AUTO-CLEAR-ERROR:
DEVICE stopped the communication to all devices and reached the auto-clear end state

NON-EXCHANGE-ERROR
the communication to at least one device is faulty and no process data is exchange with it.

PERIPHERAL-ERROR:
at least one device reports peripheral fault. Source of this error can be a short circuit in one of the device outputs or the not connected peripheral voltage.

EVENT-NOTIFICATION:
at least one defective process data cycle was detected or network has been rescanned and reinitialized

HOST-NOT-READY-NOTIFICATION:
indicates if the HOST program has set its state to operative or not. If the bit is set the HOST program ist not ready to communicate

OUTGOING-INTERFACE-1-ERROR:
at least one physical defective outgoing interface 1(local bus branch or installation branch) of one device was detected during the InterBus ID-scan. Because the defective interface generates a timeout after scanning it, it was deactivated.

OUTGOING-INTERFACE-2-ERROR
at least one physical defective outgoing interface 2(remote bus branch) of one device was detected during the InterBus ID-scan. Because the defective interface generates a timeout after scanning it, it was deactivated.

The bit field serves as the collective display of global error notifications comming from the network or the master at runtime. Because the errors and the location can either occur at the DEVICE itself or at the handled devices, they are distinguished within two following bytes. One byte fixes the exact error location (bus address 0-127 for devices, 255 for the master globally) and an exact error event (error number). If more than one errror is determined, the location value shows always the faulty participant with the closest position to the master in the InterBus.

- The `CTRL` bit indicates heavy runtime errors. Some of them can occur during startup procedure of the master. For example if the IBS controller chip IX1 of the card do not respond or the configuration of SyCon has inconsistencies. Other errors can occur during runtime, for example if the HOST program do not trigger its watchdog cells in time. Detailed information about the error can be read out from the cells `Err_Dev_Adr` and `Err_Event.`

- The ACLR bit will be set, when the master stops the communication to all its handled device because of missed devices. Before doing this, is sets all output values of the left device to the save zero condition. The behavior, if the master shall shut down or not, when it lost the contact to at least one device, is configureable in SyCon configuration tool or in the bus parameter download procedure. After the master has shut down only a warm- or coldstart of it can reactivate the communication again.
- An activated NEXC bit incdicates that one of the configured device is not operational because of an configuration fault or simply because it's not present in the network. Detailed information about the error can be read out from the cells Err_Dev_Adr and Err_Event.
- Some IB-devices have the capability to indicate on InterBus side that they have detected low power or a short ciruit in the in the external periphery. If at least one device reports this error it is shown in the bit PRHL globally. The cell Err_Dev_Adr indicates which of the handled device reports this error.(if more peripheral faults were detected at the same time, the value shows the physically closest device to the master in the InterBus ring indicating this fault).
- The bit EVE will be set during the InterBus process data cycle runtime only. If it is set once it will not be reseted any more until the DEVICE is rested globally. It indicates that at least one defective process data cycle was detected or in case of network configuration changes a network reset and rescan had to be executed. The variables Defective_Datacycles and Network_Rescans represent the number of defective data cycle which were detected respectively how many network rescans were done.
- The HOST program can set its state to 'operative' or 'non operative' by accessing directly to the dual-port memory with the NotRdy bit in the cell DevFlags or when device driver functions are used with the DevSetHostState() function. The NRDY bit indicates now, if the HOST has set its state to 'operative' = 0 or 'non operative' = 1. If SyCon configuration tool is used for example in debugging session via serial diagnostic port, this bit is also read out and shown and indicates now if the HOST program has set its state to ready or not.
- If the I1ERR bit is set by the DEVICE, at least one local bus interface or remote bus branch interface (called outgoing interface 1) of a device was detected during the ID-scan, which has produced a timeout after it was opened in this session. This error can only occur at InterBus branch interfaces, because these are the only components which have interface 1 to make branching in InterBus possible. The cell Err_Dev_Adr indicates at which of the handled device this error was detected.(if more defective interfaces were detected at the same time, the value shows the physically closest device to the master in the InterBus ring first).
- If the I2ERR bit is set by the DEVICE, at least one remote bus interface (called outgoing interface 2) of a device was detected during the ID-scan, which has produced a timeout after it was opened in this session. This error can only occur either at InterBus branch interfaces or at remote bus devices, because both are having the outgoing interface 2 to connect it to the next remote bus device.(if more defective interfaces were detected at the same time, the value shows the physically closest device to the master in the InterBus ring first).

- Octet 2: IBM_State

This variable represents the main state of the master system. Following values are possible:

00H:	state	OFFLINE
40H:	state	STOP
80H:	state	CLEAR
C0H:	state	OPERATE

- Octet 3: Err_Dev_Adr

Some bits in the `Global_Bit` field indicating errors in the network or in the DEVICE itself have always a closer error desciption. In these cases the variable `Err_Dev_Adr` represents the source of the error. The source where the error was detected can be either the DEVICE itself., then the variable contains the value 255, or the error was detected at or reported by a network device. Then the variable is filled up with this station address directly and has a range from 0 to 127.

- Octet 4: Err_Event

To complete the error description the variable `Err_Event` delivers next to the error source the corresponding error number. All possible numbers are listed below.

- Octet 5-6: Defective_Datacycles

The DEVICE counts in this variable the number of defective process data cycles. An increasing number here is an indication for an electronical influenced network surrounding or for not well wired InterBus device connections. Normally this counter should not be incremented by the DEVICE, but if so please check all your device connections and bus wiring. Note: A defective data cycle is also counted, if one device reports a peripheral error or a reconfiguration.

- Octet 7-8: Network_Rescans

On heavy network errors, for example a disconnected module during runtime, the DEVICE will automatically execute a network reset and rescan to look for the error location. Depending on how the card is configured, the master stops then or rerun the network with all refound modules. For each of such network resets and rescans the variable `Network_Rescans` will be incremented by a value of one.

- Ext_Global_Bits

D7	D6	D5	D4	D3	D2	D1	D0
				PUE	MFAIL	MWARN	WARN

WARNING:
DEVICE detected an increased number of defective data cycle within a defined time period. Please check diagnostic informaion for datailed error location.

MAU-WARNING:
at least one slave has reached the maximum possible power of its optical transmitter to guarantee an errorfree interbus transmission. The optical interface must be checked of this slave.

MAU-FAIL:
at least one slave device has detected a high signal input level for a minimum time of 64bits at one of its phyical interbus intefraces. This error is an indiaction for a loose connection at this slave, please check wiring.

POWER-UP-EVENT:
DEVICE has detected a slave device which has performed a power up reset during runtime. This is not allowed normally during runtime and the slave power must be checked if it is stable.

reserved for further use

- Variable Sl_cfg

This variable is a field of 16 bytes and contains the paremeterization state of each device station. The following table shows, which bit is related to which slave station address:

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Offset								
750H	7	6	5	4	3	2	1	0
751H	15	14	13	12	11	10	9	8
752H	23	22	21	20	19	18	17	16
...								
75FH	127	126	125	124	123	122	121	120

Table of the relation between node address and the Sl_Cfg bit

If the Sl_cfg bit of the corresponding slave is logical
'1', the slave is configured in the master, and serviced in its states.
'0', the slave is not configured in the master.

- Variable Sl_state

This variable is a field of 16 bytes and contains the state of each slave station. The following table shows, which bit is related to which slave station address:

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Offset								
760H	7	6	5	4	3	2	1	0
761H	15	14	13	12	11	10	9	8
762H	23	22	21	20	19	18	17	16
...								
76FH	127	126	125	124	123	122	121	120

Table of the relation between slave station address and the Sl_state bit

If the Sl_state bit of the corresponding slave station is logical
'1', the slave and the master are exchanging their I/O data.
'0', the slave and the master are not exchanging their I/O data.

- Variable Sl_diag

This variable is a field of 16 bytes containg the diagnostic bit of each slave. The following table shows the relationship between the slave station address and the corresponding bit in the variable Sl_diag.

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Offset								
770H	7	6	5	4	3	2	1	0
771H	15	14	13	12	11	10	9	8
772H	23	22	21	20	19	18	17	16
...								
77FH	127	126	125	124	123	122	121	120

Table of the relationship between slave station address and the Sl_diag variable

If the Sl_diag bit of the corresponding slave station is logical

'1', latest received slave diagnostic data are available in the internal diagnostic buffer. This data can be read by the user with a message which is described in the chapter 'The message interface' in this manual.

'0', since the last diagnostic buffer read access of the HOST, no values were change in this buffer.

Following error numbers are valid for `Err_event`, if `Err_dev_adr` is 255:

err_event	description	error source	help
0	no actual error		
52	unknown process data handshake	warmstart	check warmstart parameters in the HOST program.
56	no device table found	DEVICE	DEVICE is not configured via SyCon
57	IBS controller chip is defective and do not respond	DEVICE	replace DEVICE
101	configured ident-code or length code different to connected network constellation	network	check configured length codes or ID-codes of all configured slave devices
102	too many devices are connected to the DEVICE	network	reduce connected device number
103	configuration has changed during the ID-Scan cause by interruption of the ID-scan cycle because of short non diagnosticable network errors.	network	wait until DEVICE does next ID-scan automatically
104	set up of the actual network configuration after the main InterBus ID-scan failed	network	contact technical support
105	interruption of the ID-scan cycle because of short non diagnosticable network errors, caused by installation errors or a defective slave module	network	wait until DEVICE does next ID-scan automatically
106	expected, already scanned slave module is missing during next ID-scan cycle	network	wait until DEVICE does next ID-scan automatically
107	configuration has changed during runtime, a running device is not responding any more	network	check your network and wait for the next automatic ID-scan
108	no connection to the InterBus. Interruption of the connection between DEVICE and first remote bus module in the network	network	check the connection between DEVICE and first network device
120	local bus inconsistent	configuration	a local bus segment contains remote bus slaves in the same level.
121	inconsistent group or alternative	configuration	a local bus segment defined as group does contain different group or alternative numbers. Or a leading branch IB device of an alternative, has non or a wrong alternative number
122	group zero	configuration	a slave does have an alternative number, but not a group number. A group number must be defined for that device too.
220	HOST watchdog failed, timeout occured	HOST	check the HOST program if it is running and retriggering the software watchdog

221	HOST program does not acknowledge the process data indication in time when process data handshake mode 0 is used	HOST	check if the HOST program is fast enough to acknowlege fast bus cycles in sychronous mode
224	error in IBS controller communication	DEVICE	contact technical support

following error numbers are valid for `Err_event`, if `Err_dev_adr` is < > 255:

err_event	description	error source	help
0	no error event		
	InterBus network specific error codes		
30	device was missing in the last activated network scan cycle	device / configuration	check configuration or wiring
	A configured slave module could not be detected within the InterBus network. A physical problem of wiring could not basically be detected, so the problem is either a network configuration error or the module is not really connected at its physical position to the previous InterBus module in the InterBus ring. So please compare the actual activated module list with the real physical connected one. If no difference can be detected, check the wire between the module and its physical previous module. If it seems to be ok watch the LED 'RC' or 'CC' on the module. If it is not statically on the wiring is not ok. Note that the standard InterBus cable must contain a special pin bridge in the outgoing plug connector so that the module to which the module shall be connected to can recognize a further connected device. Please check this pin bridge within the outgoing plug connecter in its functionality.		
31	device reports other identification code than the configured value	device / configuration	compare configured identification code of the module with the real present one
	The module itself could be scanned in the last activated network scan, but the InterBus specific identification code that was reported by it within this scan differs from the configured value. The identification code fixes the slave modules functionality, defines its class and defines the support of I/O process data. The master denies the process data access if the values are different. So please compare the identification code that is configured with the modules real identification code. If no slave device manual is available the identification code can usually be found printed on the front panel of the module.		
32	device reports other length code than the configured value	device / configuration	compare configured length code of the module with the real present one
	The module itself could be scanned in the last activated network scan, but the InterBus specific length code that was reported by it within this scan differs from the configured value. The length code fixes the slave modules process data length within the InterBus ring. The master denies the process data access if the values are different. So please compare the length code that is configured with the modules real length code. Please refer to the length code list in a chapter below of this manual to get the relation between length code and real process data width, if no device description manual is available.		
33	further device at outgoing interface 1detected which are not configured	device / configuration	check the real configuration for these non configured devices
	This problem is actually caused by a configuration error. The next configured device has a different bus segment level than the device which is configured at this position. This can have two reasons. 1. The bus segment level of the following device is configured wrong 2. The master has detected really at least one further device at the outgoing interface 1 of this station which is not configured. An outgoing interface 1 is only available at InterBus branch interfaces so this problem could be actual located within the connected branch of this module. Please compare the actual configuration with the real physical present one ,especially in missing devices and configured bus segment levels and reconfigure the InterBus configuration in the missing station.		

34	further device at outgoing interface 2 detected which are not configured	device / configuration	check the real configuration for these non configured devices
	This problem is actually caused by a configuration error. The next configured device has a different bus segment level than the device which is configured at this position. This can have two reasons. 1. The bus segment level of the following device is configured wrong 2. The master has detected really at least one further device at the outgoing interface 2 of this station. An outgoing interface 2 is available in local and remote bus installations. If this problem occurs within a local bus, the whole local bus is deactivated. If this problem occurs within the remote bus, the previous faultless InterBus ring remains operative. Please compare the actual configuration with the real physical present one ,especially in missing devices and configured bus segment levels and reconfigure the InterBus configuration in the missing station.		
36	device reports peripheral error	device	check if the power of the external periphery of this module is connected or if outputs producing short circuits
	If a slave module signals a module error, the module has either detected a failure of its peripheral power supply or a short circuit at at least one of its peripheral inputs or outputs. The module is basically operative and remains active in the InterBus network. The exact error source cannot be defined here, because it is module manufacturer specific to report such an error. But basically the LEDs on the module can be checked first. The LED 'Us' should be checked to get the indication if the power of the periperal system is present. Then, depending on the modules functionality, the LEDs of the inputs and outputs or special error I/O LEDs should be watched, which can indicate an I/O error source.		
40	defective outgoing interface 1(local bus branch or installation branch)	device	check the wiring of the corresponding IB interface
	The outgoing remote or local bus branch interface of the module produces an InterBus timeout error, when the interface was 'opened' logical and scanned for further connected devices during the scan-cycle. A timeout error is normally produced when a network wire is plugged into the outgoing interface, but no further module is really connected to it. Please check the connection between the module and the next following branch module. If it seems to be ok then watch the LEDs on the branch modules. In case of a local bus branch all branch modules must have a lightning 'RC' or 'CC' LED which indicates principle master connection. If one is not statically on then the wiring between this and the previous module is interrupted. Please check this wire connection or simply replace the cable before modules are replaced. If all LEDs are on, at least the send transmission direction seems to be ok. So then either one of the following modules is defective and do not send back any InterBus information or the ingoing line of this branch modules interface is physically defect. In case of a remote bus branch the LEDs 'RC' or 'CC' must be statically on at the directly following remote bus module of the branch. If not so, the wire connection between both modules could be interrupted. Because an InterBus cable contain send and receive line within one wire, the error can come either from the outgoing branch coupler interface or from the ingoing interface of the following module. Before modules are replaced, the replacement of the wire should be tried first. If the LED is indicated then the error can come from the backgoing line of the following module or from the ingoing line of the branch modules interface. Please try replacement of the modules each by each.		
41	defective outgoing interface 2(remote bus)	device	check the wiring of the corresponding IB interface

	The outgoing remote interface of the module produces an InterBus timeout error, when the interface was 'opened' logical and scanned for further connected devices during the scan-cycle. A timeout error is normally produced when a network wire is plugged into the outgoing interface, but no further module is really connected to it. Please check the connection between the module and the next following remote bus module. If it seems to be ok then watch the LED 'RC' or 'CC' on the following module. If it is not statically on the trasnmission line in the direction branch module to remote module is not ok. Replace wire before replacing the modules. If the LED is indicated then either the following module is defective and do not send back any InterBus information or the ingoing interface of the branch module is physically defect.		
42	device hasn't reported its ident and length code right in the last network scan cycle	network	check surrounding of the device if some other electrical disturbing devices can be found
	On each network scan cycle that is necessary during runtime because of process data cycle errors, the master checks the actual active identification list of devices against the internal configured list. If a module that was actually classified as active once reports back a different length or ID-code this error is reported. Such an error normally can only occur if a device was powered up again and is simultaniously capable to report in its ID-regioster the socalled 'µP not ready = 0x0038' identification code during its startup. So this error event is a direct indication for a powered down slave module. Check the shielding of the network wire and look for the right grounding of the module. Especially check the power of the modules logic for dropouts or spikes.		
46	device handler stopped	DEVICE	master has stopped the InterBus communication to that device

err_event	description	error source	help
	Download configuration errors in case of SyCon download		
70	double address configured	DEVICE configuration	contact technical support
71	device data set length faulty	DEVICE configuration	contact technical support
72	process data configuration length faulty	DEVICE configuration	contact technical support
73	additional table length faulty	DEVICE configuration	contact technical support
74	PCP data length faulty	DEVICE configuration	contact technical support
75	size of whole data set inconsistent	DEVICE configuration	contact technical support
76	additional table inconsistent	DEVICE configuration	contact technical support
77	maximum output process data offset overstepped	DEVICE configuration	contact technical support
78	maximum input process data offset overstepped	DEVICE configuration	contact technical support
79	maximum offset addresses overstepped > 255	DEVICE configuration	contact technical support
80	module count in comparison oto the offsets inconsistent	DEVICE configuration	contact technical support

81	output module number unequal output offset number	DEVICE configuration	contact technical support
82	input module number unequal input offset number	DEVICE configuration	contact technical support
83	real output length unequal to configured modules length	DEVICE configuration	contact technical support
84	real input length unequal to configured modules length	DEVICE configuration	contact technical support
85	overlapped output data configured	DEVICE configuration	contact technical support
86	overlapped input data configured	DEVICE configuration	contact technical support

err_event	description	error source	help
87	output device has also defined input modulss	DEVICE configuration	contact technical support
88	input device has also defined output modulss	DEVICE configuration	contact technical support
89	output device has defined input modules	DEVICE configuration	contact technical support
90	input device has defined output modules	DEVICE configuration	contact technical support
91	device is configured to impossible installation depth	DEVICE configuration	contact technical support
92	configured ident code not supported by the DEVICE	DEVICE configuration	contact technical support

4 The Message Interface

The following send and receive messages are exchanged with the DEVICE via its mailboxes in the structure like it is described in the chapter 'definition of the message interface' in the toolkit manual.

To put and get messages to respectively from the DEVICE through its mailboxes use the device driver functions `DevPutMessage()` or `DevGetMessage()`. With direct access to the dual-port memory you must write the message in the `DevMailbox` or read the message out of the `HostMailbox` with the mechanism described in the toolkit manual.

4.1 The PLC-Task

The PLC task manages the process input and output data and handle the steering of the IBS cycles corresponding the parameterization. Therefore the task communicates with the IBM task and starts the process data cycles according to the parametrized operation mode. The task has implemented the following functions:

- activate data cycles
- mapping of the physical addresses of the data to the logical addresses of the data in the dual-port memory.

The task manages following message commands:

IBM_Shared_Memory	write consistent data block into the send process data `SndPd` during one InterBus cycle or read consistent data block from `RecvPd` during one InterBus cycle

4.1.1 IBM_Shared_Memory

command message			
variable	type	value	description
msg.rx	byte	2	receiver = PLC-Task
msg.tx	byte	16	transmitter = HOST
msg.ln	byte	 8 8+m	length of the message read access write access
msg.nr	byte	j	number of message (optional)
msg.a	byte	0	no answer number
msg.f	byte	0	no error
msg.b	byte	17	command : IBM_Shared_Memory
msg.e	byte	0	unused
msg. DeviceAdr	byte	0	device address unused
msg. DataArea	byte	 0 1 2	data area: data fnc decides the data area receive process data area send process data area
msg. DataAdr	word	 0-255 0-511 0-255	address offset refer to the data type word-offset address Byte-offset address word-offset address if bit access
msg. DataIdx	byte	0-15	bit position within the word offset address if bit access
msg. DataCnt	byte	m	count of read or write data referring to the datatype
msg. DataType	byte	 6 5,10 14	datatype : TASK_TDT_UINT16: word TASK_TDT_UINT8: octet-string TASK_TDT_BIT: bit

msg. DataFnc	byte	
		1
		2
			function :
			TASK_TFC_READ = read access
			TASK_TFC_WRITE = write access

msg.d[0]	byte	x	write access: first data to be written read access: unused
...	...	...	...
msg.d[m-1]	byte	z	write access: last data to be written read access: unused

The command serves the user program to write data of a definite length into the send process data buffer or to read from the receive process data. The command can be used in all process data handshake modes.
With the command the data types word, bytes or bits can be selected. The firmware of the DEVICE guarantees that the read or write access of the data will be done safely during two active DP cycles.
This command is an other possibility to get access to the process data in the dual-port memory. Its disadvantage is the slower access then the direct reading or writing the dual-port memory. But the main advantage is that you have the access to the process data also via a message, when an old driver for example has already message functionality in it. We use this message in all diagnostic tools too.

The data type is fixed in the byte `msg.DataType`. Only the values decimal 6 for words, 5 or 10 for byte strings and 14 for bits are allowed.

The read access is distinguished from the write access in the byte `msg.Function`. A 1 is valid for read access and 2 for write access.

The data area to be read from, is fixed in `msg.DataArea`. 1 is valid for the receive process data buffer and 2 for the send process data buffer. In case of the value 0, the value placed in `msg.Function` decides the data area automatically.

The count of the data to be read or to be written is fixed by the value of `msg.DataCnt`. The count refers to the chosen data type. Maximum permitted values are 119 for words, 239 for byte and 255 for bits.

The offset address is fixed in the word `msg.DataAdr`. The specified address must be refered on the chosen data type and is interpreted from the DEVICE as the relative address to the start address in the send process data or the receive process data. The maximum values are decimal 255 for word, 511 or byte and 255 for bit access. In case of bit access the value in `msg.DataIdx` additionally fixes the relative offset in the word to be read or to be written. For the other accesses the value doesn't have any meaning.

The data at `msg.d[]` are unused, if read access is chosen, while in write access in this area the send data must be written in. Words must be written in Intel format - LSB before MSB - and bits must be put in there in packed form. For example to write 5 bits, the first data byte `msg.d[0]` must be placed in the bits 0-4 to be valid.

answer message to the user			
variable	type	value	description
msg.rx	byte	16	receiver = HOST
msg.tx	byte	2	transmitter = PLC-Task
msg.ln	byte	 8+m 8 0	length of the message read access write access error
msg.nr	byte	j	number of message
msg.a	byte	17	answer: IBM_Shared_Memory
msg.f	byte	0 f	no error error code see following table
msg.b	byte	0	no command
msg.e	byte	0	unused
msg. DeviceAdr	byte	0	device address unused
msg. DataArea	byte	 0 1 2	data area: data fnc decides the data area receive process data area send process data area
msg. DataAdr	word	 0-255 0-511 0-255	address offset refer to the data type word-offset address Byte-offset address word-offset address if bit access
msg. DataIdx	byte	0-15	bit position within the word offset address if bit access
msg. DataCnt	byte	m	count of read or write data referring to the datatype
msg. DataType	byte	 6 5,10 14	data type : TASK_TDT_UINT16: word TASK_TDT_UINT8: octet-string TASK_TDT_BIT: bit

msg. DataFnc	Byte	
		1
		2
			function :
			TASK_TFC_READ = read access
			TASK_TFC_WRITE = write access

msg.d[0]	Byte	x	read access: first data be read write access: unused
...	...	...	...
msg.d[m-1]		z	read access: last data be read write access: unused

In the answer message the `msg.f` byte gives the information, if the desired command could be executed. If the byte is 0, an positive job result is send back.

4.2 The IBM-Task

The IBM task handles the communication with the fieldbus controller and does the mapping of the process data from their physical position in the InterBus ring to the configured logical addresses in the dual-port memory.

Because the task holds the direct contact to the IB controller it recognizes every error that occurs in the network, so that the second function of this task is the InterBus configuration and diagnostic. Therefore the IB-device specific and the global bus status information are also managed by this task.

The task manages following message commands:

IBM_Start_Seq	start download of multiplexed IB-device parameters
IBM_End_Seq	end of a multiplexed download sequence
IBM_Download	non static download of bus and IB-device parameters
IBM_Device_Diag	read out the status structure of an IB-device
IBM_Get_Physical_Configuration	executes an automatic network scan of the connected IB-devices and returns the constellation.
IBM_Set_Configuration	Enables and disables InterBus slaves during runtime
IBM_Control_Active_Configuration	Enables and disables slaves, groups or alternatives.

4.2.1 Starting and Stopping Communication during Runtime

4.2.1.1 Using Device Driver Function to Write

Use the function `DevSetHostState()` together with the parameter `HOST_NOT_READY` to stop the network communication. Use the parameter `HOST_READY` to start or restart the communication.

4.2.1.2 Direct Write Access in Dual-Port

To start and stop the InterBus communication of the DEVICE you have to clear and set the bit `NotRdy` in the cell `bDevFlags`. Clearing the bit will start the network communication while setting the bit stops the communication.

ATTENTION: Stopping the communication will always cause a reset of the network modules output data.

4.2.2 Deleting Existing Data Base in the DEVICE

Normally the configuration will be downloaded by the SyCon configuration tool statically into the FLASH memory. The DEVICE reads out this data block during its startup. If all parameters are valid the DEVICE starts its slave handlers and goes into the mode OPERATE. Then the message download procedure like it is described in the chapters below can not be used any more.

If no static download of the configuration data is wished, all these data can be handed over online to the DEVICE by a message download from the HOST program using the functions Start,End and Download. But before doing this, you have to prevent the DEVICE to start up with possible downloaded static parameters. This can be done by deleting the data base by message service before. Then the DEVICE starts up without finding any configuration data base and then the online message download can be proceeded by the HOST program like it is described in the following chapters.
IMPORTANT NOTE! If no data base exists within the DEVICE, the DEVICE must be initialized with protocol parameters (see chapter: protocol parameters) before the message download is done, to fix the process data handshake mode and the storage format etc.

	command message			
	variable	type	value	signification
Message header	msg.rx	byte	0	receiver = RCS-Task
	msg.tx	byte	16	transmitter = user at HOST
	msg.ln	byte	2	length of the message
	msg.nr	byte	j	number of the message
	msg.a	byte	0	no answer
	msg.f	byte	0	no error
	msg.b	byte	6	command = data base access
	msg.e	byte	0	extention, not used
Service header	msg.d[0]	Byte	4	mode = delete data base
	msg.d[1]	Byte	8	startsegment of the data base

	answer message			
	variable	type	value	signification
Message header	msg.rx	byte	16	receiver = user at HOST
	msg.tx	byte	0	transmitter = RCS-Task
	msg.ln	byte	1	length of message
	msg.nr	byte	j	number of the message
	msg.a	byte	6	answer = data base access
	msg.f	byte	f	error, state
	msg.b	byte	0	no command
	msg.e	byte	0	extension

The time for deleting the data base depents on the used FLASH memory, so sending back the answer message can take up to 3 seconds

4.2.3 IBM_Start_Seq

	command message			
	variable	type	value	signification
Message header	msg.rx	byte	3	receiver = IBM-Task
	msg.tx	byte	16	transmitter = user at HOST
	msg.ln	byte	4	length of the message
	msg.nr	byte	j	number of the message
	msg.a	byte	0	no answer
	msg.f	byte	0	no error
	msg.b	byte	67	command = IBM_Start_Seq
	msg.e	byte	0	extention, not used
IBM_START_SEQ_REQUEST	msg.d[0]	Byte	0	Req_Adr, unused
	msg.d[1]	Byte	0-126	Area_Code, InterBus-slave address
	msg.d[2]	Word	0-65535	Timeout, not supported

The command starts a blocked download in the stated `Area_Code`. To complete the download the command `IBM_End_Seq` must be called after finishing the download sequence.

	answer message			
	variable	type	value	signification
Message header	msg.rx	byte	16	receiver = user at HOST
	msg.tx	byte	3	transmitter = IBM-Task
	msg.ln	byte	1	length of message
	msg.nr	byte	j	number of the message
	msg.a	byte	67	answer = IBM_Start_Seq
	msg.f	byte	f	error, state
	msg.b	byte	0	no command
	msg.e	byte	0	extention, not used
IBM_START_SEQ_CONFIRM	msg.d[0]	byte	240	Max_Len_Data_Unit

The value `Max_Len_Data_Unit` fixes the maximum length of the parameter `Data` per `IBM_Download` message.

Possible values for `msg.f` are the following:

error number msg.f	signification
0	no error
52	CON_NI, Area_Code unknown

See below the corresponding structures in the header file:

```
IBM_START_SEQ_REQUEST
IBM_START_SEQ_CONFIRM
```

4.2.4 IBM_End_seq

	command message			
	variable	type	value	signification
Message header	msg.rx	byte	3	receiver = IBM-Task
	msg.tx	byte	16	transmitter = user at HOST
	msg.ln	byte	1	length of the message
	msg.nr	byte	j	number of the message
	msg.a	byte	0	no answer
	msg.f	byte	0	no error
	msg.b	byte	69	command = IBM_End_Seq
	msg.e	byte	0	extention, not used
IBM_END_SEQ_REQUEST	msg.d[0]	byte	0 = NEW_ENTRY 1=CHANGE_ENTRY 2 =REMOVE_ENTRY 3=INSERT_ENTRY	Req_Add, defines function

The command ends the blocked download and activates the previously sequentially downloaded data.

The parameter Req_Add defines the download function in case of download a slave data set.
To defined a new entry use the command NEW_ENTRY.
Is is possible to delete an existing entry by setting the parameter to REMOVE_ENTRY. All other entries that are set up at execution time of this command with higher Area_Code number will be corrected downwards. That means if entry 3 is removed for example, entry 4 will be 3 and entry 5 will be 4 and so on.
It is possible to insert a new entry between to existing data sets by setting the parameter to INSERT_ENTRY. All other entries that are set up at execution time of this command with higher and equal Area_Code number will be corrected upwards. That means if entry 3 is inserted for example, old entry 3 will be 4 and old entry 4 will be 5 and so on.
It is possible to simply change and existing entry if the parameter is set up to value CHANGE_ENTRY. Other entries are not influenced by this command.

answer message			
variable	type	value	signification
msg.rx	byte	16	receiver = user at HOST
msg.tx	byte	3	transmitter = IBM-Task
msg.ln	byte	0	length of message
msg.nr	byte	j	number of the message
msg.a	byte	69	answer = IBM_End_Seq
msg.f	byte	f	error, state
msg.b	byte	0	no command
msg.e	byte	0	extention, not used

Possible values for `msg.f` are the following :

error	signification
0	no error
52	CON_NI, Area_Code unknown
57	CON_SE, sequence error
70	slave address already configured
71	data set field length faulty
72	configuration field data length faulty
73	additonal table field length faulty
74	pcp field length faulty
75	whole data set size inconsistent
76	additional table inconsistent
77	output process data offset address oversteps maximum range
78	input process data offset address oversteps maximum range
79	too much input and output offset addresses configured
80	number of configured modules does not correspond to the number of configured offset addresses
81	number of configured output modules does not correspond to the number of output offset addresses
82	number of configured input modules does not correspond to the number of input offset addresses
83	the real output process data length of the slave resulting from its length code is smaller than the resulting value from the configured output modules
84	the real input process data length of the slave resulting from its length code is smaller than the resulting value from the configured input modules
85	address conflict of output process data
86	address conflict of input process data
87	ID-code of module indicates process output data only , but inputs modules are also defined
88	ID-code of module indicates process input data only, but output modules are also defined
89	ID-code of module indicates process output data, but no output modules defined
90	ID-code of module indicates process input data only, but no input modules defined
91	slave configured to wrong level or level out of range 0 - 12
92	configured length code is unknown and can not be handled
93	a slave data shall be removes that doesn't exits
94	a slave data set of an active slave shall be changed. That is not possible in online configuration mode. Use IBM_Set_Configuartion to switch off the module
95	an entry shall be removed that configuration data differs from the message data

See below the corresponding structure in the header file:

`IBM_END_SEQ_REQUEST`

4.2.5 IBM_Download

	command message			
	variable	type	value	signification
Message header	msg.rx	byte	3	receiver = IBM-Task
	msg.tx	byte	16	transmitter = user at HOST
	msg.ln	byte	m + 4	length of the message
	msg.nr	byte	j	number of the message
	msg.a	byte	0	no answer
	msg.f	byte	0	no error
	msg.b	byte	68	command = IBM_Download
	msg.e	byte	0	extention, not used
IBM_DOWNLOAD_REQUEST	msg.d[0]	byte	0 = NEW_ENTRY 1=CHANGE_ENTRY 2 =REMOVE_ENTRY 3=INSERT_ENTRY	Req_Add, defines function
	msg.d[1]	byte	 0-126 127	Area_Code, IB-slave number master bus parameters
	msg.d[2]	word	0-760	Add_Offset
	msg.d[4-240]	m bytes	0-255	Data[240]

This command allows to hand over the master bus parameters or the slave parameter data files. This is commendable, if no static data base exists on the DEVICE and the parameterization should happen from the HOST program without using SyCon-IBM tool.

Two ways to download data files have been implemented. A data file can be downloaded either in one call (single download) or if it is to large in block calls (sequenced download) into an internal download area (length 1000 bytes). After the download cycle is finished completly the specified data is checked and copied afterwards into the task access area. Then the next download can be started into the freed download area.

The parameter `Req_Add` defines the download function in case of download a slave data set.
To defined a new entry use the command `NEW_ENTRY`.
Is is possible to delete an existing entry by setting the parameter to `REMOVE_ENTRY`. All other entries that are set up at execution time of this command with higher `Area_Code` number will be corrected downwards. That means if entry 3 is removed for example, entry 4 will be 3 and entry 5 will be 4 and so on.
It is possible to insert a new entry between to existing data sets by setting the parameter to `INSERT_ENTRY`. All other entries that are set up at execution time of this command with higher and equal `Area_Code` number will be corrected upwards. That means if entry 3 is inserted for example, old entry 3 will be 4 and old entry 4 will be 5 and so on.
It is possible to simply change and existing entry if the parameter is set up to value `CHANGE_ENTRY`. Other entries are not influenced by this command.

The parameter Area_Code fixes the destination area (master parameter or slave parameter file). The offset in the download area where the data will be copied from the message is fixed in the variable Add_Offset.
If a node data file shall be transferred sequenced, the command IBM_Start_Seq must be activated before to initialize the download sequence. The sequence will be finished after the command IBM_End_Seq is called. Even then the parameters will be checked and be set valid if no error is recognized. The download of the bus parameters needs no sequenced download.

See below the corresponding structure in the header file:

IBM_DOWNLOAD_REQUEST

answer message			
variable	type	value	signification
msg.rx	byte	16	receiver = user at HOST
msg.tx	byte	3	transmitter = IBM-Task
msg.ln	byte	0	length of message
msg.nr	byte	j	number of the message
msg.a	byte	68	answer = IBM_Download
msg.f	byte	f	error, state
msg.b	byte	0	no command
msg.e	byte	0	extention, not used

Possible values for `msg.f` are the following:

error number msg.f	signification
0	no error
52	CON_NI, Area_Code unknown
57	CON_SE, sequence error
70	slave address already configured
71	data set field length faulty
72	configuration field data length faulty
73	additonal table field length faulty
74	pcp field length faulty
75	whole data set size inconsistent
76	additional table inconsistent
77	output process data offset address oversteps maximum range
78	input process data offset address oversteps maximum range
79	too much input and output offset addresses configured
80	number of configured modules does not correspond to the number of configured offset addresses
81	number of configured output modules does not correspond to the number of output offset addresses
82	number of configured input modules does not correspond to the number of input offset addresses
83	the real output process data length of the slave resulting from its length code is smaller than the resulting value from the configured output modules
84	the real input process data length of the slave resulting from its length code is smaller than the resulting value from the configured input modules
85	address conflict of output process data
86	address conflict of input process data
87	ID-code of module indicates process output data only , but inputs modules are also defined
88	ID-code of module indicates process input data only, but output modules are also defined
89	ID-code of module indicates process output data, but no output modules defined
90	ID-code of module indicates process input data only, but no input modules defined
91	slave configured to wrong level or level out of range 0 - 12
92	configured length code is unknown and can not be handled
93	a slave data shall be removes that doesn't exits
94	a slave data set of an active slave shall be changed. That is not possible in online configuration mode. Use IBM_Set_Configuartion to switch off the module
95	an entry shall be removed that configuration data differs from the message data
96	the length indicator for the PCP relevant structure IBM_DEV_PCP_DATA is invalid. Please check that if PCP is confivgured the length is set to value 47dec.
97	the check of the IBM_KBL_ENTRY_STAT failed. The structure contains invalid data. please check the contents of the structure in accordance to the chapter 'coding of the device parameter data set'

98	after loading the bus parameter the termination of the KBL-list failed. internal error, consult Hilscher
99	allocation of resources fo all PCP-KBL-entries of all slaves failed. internal error, consult Hilscher.
120	a local bus segment contains remote bus slaves in the same level.
121	a local bus segment defined as group does contain different group or alternative numbers. Or a leading branch IB device of an alternative, has non or a wrong alternative number
122	a slave does have an alternative number, but a group number. A group number must be defined for that device too.

4.2.5.1 The Download of the Master Parameters

4.2.5.1.1 Coding of the Master Parameter Data Set

variable name	type	explanation
bBaudRate	byte	selects the baudrate; at the moment 500kBits only supported
bFormat	byte	global storage format for word oriented process data
bAutoClear	byte	behavior if a slave component is defective
bReserved	byte	reserved byte
bScanTimeInterval	byte	scan time for missing devices in multiples of 800msec
bTimeoutDataCycle	byte	timeout to execute a valid data cycle in multiples of 8 msec, before network is rescanned and reseted
bMaxNumOfBundledError	byte	maximum number of bundle data cycle error before network is rescanned and reseted
usNumOfIDScanAfterError	word	maximum number of ID scans directly following a defective data cycle to get the network reoperative before network is reseted
bNumOfStopBits	byte	Number of stopbits the master sends with each IB data telegram.

The `bBaudRate` value isn't processed by the DEVICE at the moment. So the variable can be seen as an reserved value for further use.

The `bFormat` value changes the interpretation of as word oriented declared process data from LSB/MSB to MSB/LSB and vice versa.

```
#define IBM_INTEL        0
#define IBM_MOTOROLA     1
```

The `bAutoClear` parameter fixes the behavior of the DEVICE if one or many devices are defective in the network or reporting an error:

- `#define  IBM_AUTO_CLEAR_OFF                   0x00`
  The DEVICE don't cares the status of the connected devices and the automatic network reset in case of an error is disabled. Depending on the configured `ScanTimeInterval` value the master tries to get all modules operative every configured interval time.
- `#define  IBM_AUTO_CLEAR_MISSING               0x01`
  The DEVICE will stop the communication to the whole network and will reset it, if it detects a missing device after the first network scan or during process data transfer runtime.
- `#define  IBM_AUTO_CLEAR_ON_MOD_ERR            0x02`
  The DEVICE will stop the communication to the whole network and will reset it, if at least one device reports the InterBus-S specific module error. Module error capable devices report such an error normally if they have detected a short circuit or low power in their external peripherals.

- `#define IBM_AUTO_CLEAR_ON_MOD_ERR_MISSING 0x03`
  The DEVICE will stop the communication to the whole network and will reset it, if it detects a missing device after the first network scan or during process data transfer runtime or if at least one device reports the InterBus-S specific module error.

The parameter `bScanTimeInterval` enables or disables the automatic network scan for missing devices. If the value is set to 0 than this function is disabled. The first network scan which is normally done by the DEVICE once after its initialization is not influence by this value and is done anyway. Values unequal 0 configure a scan time in multiples of 800msec. One thing is important to know: the process data transfer is interrupted during the scan so that a hold input process data could be measured. The outputs of the modules during this scan aren't influences and hold the old value. We recommend here to choose the value 7 = 5600msec.

The variable `bTimeoutDataCycle` is the time in multiples of 8 millisecond, the DEVICE tries to execute and finish a started data cycle when a data cycle tranfers error was detected. After a data cycle error the DEVICE starts always an ID scan of the actual connected network before the data cycle is started again. If the timer value is overstepped because of multiple data cycle errors the network is resetted automatically. Depending on the choosen values of `bAutoClear` and `bScanTimeInterval` the DEVICE stops the whole communication or tries to reinitialize the network. We recommend here a value of 100 = 800msec.

Sometimes it can happen that a bundled count of directly following data cycles are defective. The maximum number of permissible bundled data cycle error is fixed in the variable `bMaxNumOfBundledError`. We recommend here to choose the value 20. If the value is overstepped the DEVICE will react depending on the choosen values of `bAutoClear` and `bScanTimeInterval` and stops the whole communication or tries to reinitialize the network

If a data cycle error happens the DEVICE will automatically start an ID-Scan to locate the error sources within the network. If also the following scan could not be finished without an error, the DEVICE retries the ID scan `usNumOfIDScanAfterError` times before the DEVICE behave like in the variables `bAutoClear` and `bScanTimeInterval` configured.

Some revisions of old InterBus slave chips generations like SUPI-I or II, have the characteristic to need an enlarged stopbit in each received telegram to synchronize them right in any case on each incoming telegram. Our used master chip can enlarge the stopbit only by one whole bit so the variable `bNumOfStopBits` changes between one or two send stopbits in each telegram. We recommend to use the two stopbits = value 1, because you don't know even if you have some older chips running in your system or if you have a SUPI-III generation only network.

```
#define  IBM_1STOPBITS  0
#define  IBM_2STOPBITS  1
```

See the below the corresponding structure in the header file:

```
BUS_IBM
```

4.2.5.1.2 Download Message of the Bus Parameters

	command message			
	variable	type	value	signification
Message header	msg.rx	byte	3	receiver = user at HOST
	msg.tx	byte	16	transmitter = IBM-Task
	msg.ln	byte	14	length of message
	msg.nr	byte	j	number of the message
	msg.a	byte	0	no answer
	msg.f	byte	0	error, status
	msg.b	byte	68	command = IBM_Download
	msg.e	byte	k	extension
BUS_IBM	msg.d[0]	byte	0	Req_Adr, not used
	msg.d[1]	byte	127	Area_Code
	msg.d[2]	word	0	Add_Offset
	msg.d[4]	byte	0	Baud_Rate
	msg.d[5]	byte	0,1	Format
	msg.d[6]	byte	0,1,2,3	Auto_Clear
	msg.d[7]	byte	0	Reserved
	msg.d[8]	byte	0-255	Scan_Time_Interval
	msg.d[9]	byte	0-255	Timeout_DataCycle
	msg.d[10]	byte	0-255	Max_Num_Of_BundledError
	msg.d[11]	word	0-65535	Num_Of_IDScan_AfterError
	msg.d[13]	byte	0,1	Num_Of_StopBits

After this message is sent to the DEVICE, it will always perform a network reset (outputs will be cleared in the slave modules) and start to compare the configured slave configuation with the connected configuration. If differences are detected, then the DEVICE will perform depending on the parameter `Auto_Clear`, directly a shut down afterwards, or will continue and try to establish the connection to all right configured slave devices.

REMARK-I: the bus parameter can be downloaded multiple times. Every call will cause a reset and a rescan of the connected network.
REMARK-II: all alternative declared interfaces in the slave parameter data set will be always be disabled automatically by the DEVICE after the bus parameter are downloaded.
REMARK-III: if the command `IBM_Set_Configuration` was used previously before the download of the bus parameter is used again, the DEVICE will set up the bus constellation in accordance to this last received configuration command. This command has always priority against the restriction of REMARK-II.

4.2.5.2 Download of the Device Parameter Data Sets

4.2.5.2.1 Coding of the Device Parameter Data Set

	variable name	type	explanation
IBM_DEV_PRM_HEADER	usDevParalen	word	length of whole data set inclusive the length parameter
	bDvFlag	byte	enables or disables this device parameter data set in the DEVICE
	bLengthCode	byte	InterBus specific length code of the device
	bIdentCode	byte	InterBus specific identification code of the device
	bInstallDepth	byte	installation level of the device within the network
	bGroupNumber	byte	assigned group number, for simultanous group en-or disabling.
	bAlternativeNumber	byte	assigned alternative number for alternative grouping
	bOctetString[8]	octet string	8 bytes reserved for further use, set to 0
IBM_DEV_CFG_DATA	usCfgDataLen	word	Length of the following I/O module data configuration inclusive the length of the size indicator
	ausTypeLength[...]	word array	I/O configuration data, see corresponding HEADER file for structure
IBM_DEV_PRM_ADD_TAB	usAddTabLen	word	Length of the following add. tab data inclusive the length of the size indicator
	bInputCount	byte	number of input offsets following
	bOutputCount	byte	number of output offsets following
	ausIO_Offsets[...]	word array	I/O offset addresses in the dual-port memory see corresponding HEADER file for structure
IBM_DEV_PCP_DATA	usPcpDataLen	word	Length of the following PCP channel specific data inclusive the length of the size indicator
	IBM_KBL_ENTRY_ STAT	structure	communication reference table for PCP capable devices

The main length indicator `usDevParaLen` fixes the length of the whole data block inclusive the length of the size indicator itself. The length can be calculated with the formula:

```
usDevParaLen =  sizeof( IBM_DEV_PRM_HEADER)
                  usCfgDataLen +
                  usAddTabLen +
                  usPcpDataLen+
```

This variable is followed by a special bit field called `bDvFlag`, declaring the parameter data set as active or inactive. Only if the `ACTIVE` bit is set the DEVICE will activate the network access for this device. That means if the bit is not set, the slave parameter data set is not relevant for the DEVICE, but checked for data consistencies within the set.

D7	D6	D5	D4	D3	D2	D1	D0
ACTIVE	ALTI1	ALTI2	Reserved for further use				

0 = interface 2 is not alternative grouping capable
1 = interface 2 is alternative grouping capable

0 = interface 1 is not alternative grouping capable
1 = interface 1 is alternative grouping capable

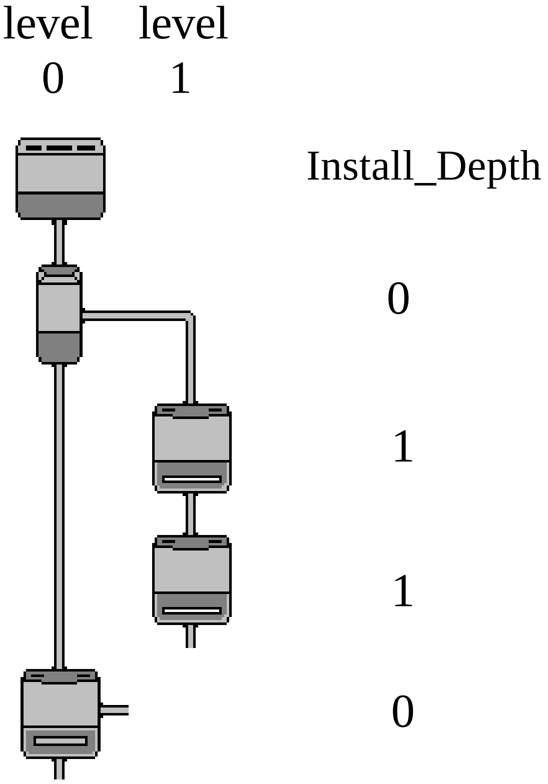
0 = device inactive in the actual configuration
1 = device active in the actual configuration

The next two bits are only relevant for InterBus branch modules or remote bus only modules. The bits `ALTI1` and `ALTI2` matches to the particular real physical interface of the module. In case of modules which have only one outgoing remote interface the `ALTI1` interface bit is not relevant. If the corresponding bits are set for the two types of interfaces, the DEVICE will disable the real physical interfaces, if no modules are configured and connected to them. This guarantees, that no timeout during the InterBus process data exchange occurs, when a cable is connected to these interfaces. This is necessary for example if new alternative slave groups wanted to be appended to the branch.
A disabled interface can be visible detected, if either the LD-LED for local bus branche, or the RD -LED for remote bus branches is light on at the corresponding module.

The `bLengthCode` is a device specific parameter value that normally can be read from the device label or manual and indicates how many process data the device will reserve within the InterBus process data shifting ring. The DEVICE will compare this value with the real present one and denies the access to this device if both values are unequal. Here is the list of all supported length codes

real process data length width counted in bytes	length code of module
0	0
2	1
4	2
6	3
8	4
10	5
16	6
18	7
4 Bit	8
1	9
12 Bit	10
3	11
1 Bit	12
2 Bit	13
12	14
14	15
reserved, not supported	16
52	17
32	18
48	19
64	20
20	21
24	22
28	23
reserved, not supported	24-32
reserved, not supported	33-255

The bIDCode is a device specific parameter value that normally can be read from the device label or manual and classifies the type of module, if it is for example a remote bus or local bus device, or if it is a PCP capable device. The DEVICE will compare this value with the real present one and denies the access to this device if both values are unequal.
Because the InterBus allowes the segmentation of the network by socalled branch interfaces which are capabable to switch on and off they outgoing interfaces, next to the physically position in the InterBus ring, the bInstallDepth locates the module within the network exactly. The DEVICE compares the configured value with the real installation level and denies the access to this device if both values are unequal. The installation depth of the first module in the ring begins always with level value 0 and is increased by the value 1 for each passed level. Here is an example:



It is possible to combine devices to groups, so that they can be switches on or off together simultanously, indepentant of being direclty connected together or not. The variable `bGroupNumber` assigns the device to a specific group. It can have a value range of 0 up to 255. The value 0 means that the device isn't assigned to a group.

A device can be also assigned to an alternative segment. An alternative is a connected and related part of the network, that can be alternativly activated next to the rest of the network during runtime. If the master finds now alternatives, it switches off the physical interface of the previous connected branch device, so that all alternatives are deactivated directly after the startup. The HOST program itself decides during runtime by using the command `IBM_Control_Active_Config- uration` which of the alternative branches shall be activated and which not.

The `ausTypeLength[...]` array informs the DEVICE how the physical process data within the slave's InterBus shifting register, fixed by its length code and ID-code, shall be composed together. One entry in the `ausTypeLength` table must result a corresponding entry in the `ausIO_Offsets[...]` table which contains the dual-port memory offset address where to lay down the module data in case of input and where the read out the module data in case of output logically. The table itself has the following structure.

variable name	type	explanation
ausTypeLength[...]	word array	type of process data and its length

see structure `IBM_DEV_CFG_DATA` in the header file

Here is the bitwise definition of one aus`TypeLength` entry that must be used for every configured socalled logical module:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
Dir	WordSwap	reserved	process data length												

process data length in mutilples of bits 0 - 8191 dec

1 = word swap active for modules with even byte length
0 = word swap deactive

1 = module type output process data
0 = module type input process data

To distinguish if the module is either an output or an input one in the view of the DEVICE, the upper bit `Dir` in the `ausTypeLength` decides the data direction. If the bit is set then the module is defined as an output module. The order of the module type don't care and module can be mixed configured.
The bit `WordSwap` is only relevant if the `process data length` of the configured module has a multiple size of 16 bits. That means 16,32,48 etc. . If it is set then the DEVICE will swap automatically the resulting byte order word wise if the global address format for the process data in the master parameter is set to mode `IBM_MOTOROLA`.

If a device for example has length code of 1 and an ID-code of 3 this constellation indicates a device with 16 bits input and output process data. Then the CfgData table could look like.

variable name	contents
usCfgDataSize	0006 hex = 6 bytes in length, inclusive the length indicator
ausTypeLength[0]	8010 hex = 16 bits output process data
ausTypeLength[1]	4010 hex = 16 bits input process data, word swap active

One entry in the CfgData table must result a corresponding entry in the AddTab table, if it is not configured there as 'don't care' module. In this table the offset address in the dual-port memory of each process data is held down, where the DEVICE has to start later the reading of the data as outputs and writing it to the device or starts to write it into as inputs during the process data cycle transfer. See the following structure:

variable name	type	explanation
bInputCount	byte	number of inputs following in the IO_Offset table
bOutputcount	byte	number of outputs following in the IO_Offset table
ausIO_Offsets[...]	word array	IO_Offsets in the order: first all input offsets then all output offsets

see structure IBM_DEV_PRM_ADD_TAB in the header file.

The ausIO_Offset's have to be placed in order to each configured I/O module in the table CfgData so that the DEVICE has a relationship between both tables and can associate them together later when doing the I/O exchange. For an output process data module it results a corresponding output offset, for an input process data module it results a corresponding input offset.
If inputs and outputs are configured at the same time, the offset table must contain first all input offsets and then all output offsets. All offsets must be configured as byte offsets, except an offset for a single bit with a process data length of 1 in the CfgData table.There an offset must be set up like the following figure illustates:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
Bit offset			Byte offset												

byte offset in the dualport memory 0 - 3583 dec

corresponding bit offset in the byte 0 -7dec

To complete the last example above, here is a related IBM_DEV_PRM_ADD_TAB example:

The word input module data shall be located at byte 4-5dec. The two byte outputs shall be located at byte 0-1dec:

variable name	contents
usAddTabLen	0008 hex = 8 bytes in length, inclusive the length indicator
bInputCount	01 hex = 1 input offset following
bOutputcount	01 hex = 1 output offset following
ausIO_Offsets[0]	0004 hex = input module to addressbyte 4-5
ausIO_Offsets[1]	0000 hex = output module to addres byte 0-1

The `usPcpDataLen` must be set fix to the value 47dec in case of a PCP-capable device. It's the length of this size indicator = 2 plus the fixed size of the communication reference structure `IBM_KBL_ENTRY_STAT`. A device is PCP capable, if the two upper bits 7 and 6 in its `bIdentCode` are set to logical 1. If a device is PCP incapable then the size must be set to 2 only. Naturally the following PCP structure `IBM_KBL_ENTRY_STAT` must not be appended in this case.

```
typedef struct {
    unsigned char   bComRef;
    unsigned char   bLocalLsap;
    unsigned char   bRemoteLsap;
    unsigned char   bRemAddr;
    unsigned char   bReserved;
    unsigned char   bLliSap;
    unsigned char   bConnType;
    unsigned char   bMaxScc;
    unsigned char   bMaxRcc;
    unsigned char   bMaxSac;
    unsigned char   bMaxRac;
    unsigned long   ulAci;
    unsigned short  usReserved;
    unsigned char   bMultiplier;
    unsigned char   bConnAttr;
    unsigned char   bReqLen_h;
    unsigned char   bReqLen_l;
    unsigned char   bIndLen_h;
    unsigned char   bIndLen_l;
    unsigned char   abServSup[SUP_SERV_LEN];
    unsigned char   abSymbol[SYMBOL_LEN];
    unsigned char*  ptVfdPointer;
} IBM_KBL_ENTRY_STAT;
```

bComRef: the communication reference is a clear specification for a communication relationship . Each slave can be configured by one communication reference. The value range goes from 2 to 63. This communication reference is used later during runtime in the services IBM_READ or IBM_WRITE to address the slave PCP channel. The communciation reference has nothing to do with the real physical position of the slave in the InterBus network, it's only a logic address. If more slaves are PCP capable and are configured, than the different bComRef variables of the slaves must be set without any gap so that all values together results a straight order from 2 up to 63 in maximum.
bLocalLsap: Local service access point. Without meaning, not handled and must be set fix to the value 128dec.
bRemoteLsap: Remote service acces point:. Without meaning, not handled and must be set fix to the value 128dec.
bRemAddr: remote address. Configures the remote partner within this communication reference for which the communication will be used to. This remote address is calculated by counting the number of PCP slaves that are configured physically up to the current slave's position in the ring. So if it's the 3'rd PCP slave in the ring for example, the remote address is 3. The value 0 is reserved for the master. The value ranges from 1 to 63.
bLliSap: this atttribute configures the LLI-User. For the standard PMS connection this attribute must be set fix to the value 0.
bConnTyp: configures the type of connection. In the InterBus protocol only the master-master-acyclic method is allowed, so that the value must be set fix to 0.

bMaxScc: maximum send confirmed Request Counter. This attribute sets the maximum allowed number of parallel confirmed services. For the standard PMS connection the value is fix 1.
bMaxRcc: maximum receive confirmed Request Counter. This attribute sets the maximum allowed number of parallel confirmed services. For the standard PMS connection the value is fix 1.
bMaxSac: maximum send acknowledge Request Counter. This attribute sets the maximum allowed number of parallel confirmed services. For the standard PMS connection the value is fix 1.
bMaxRac: maximum receive acknowledge Request Counter. This attribute sets the maximum allowed number of parallel confirmed services. For the standard PMS connection the value is fix 1.
ulAci: this attribute sets if a connection supervision shall be activated or not. Most of the remote PCP slaves do not support this feature, so that the value should be set to 0.
bMultiplier: this value has no meaning for the current PCP protocol stack and must be set to value 128.
bConnAttr: this attribute contains further information about the connection that shall be established. This value must be set fix to 0.
bReqLen_h: this value contains the maximum number of high prior PMS-PDUs data bytes in send direction in the view of the master, that could be send. High prior messages are not allowed so the value must be set to 0.
bReqLen_l: this value contains the maximum number of low prior PMS-PDUs data bytes in send direction in the view of the master, that could be send. This value range from minimum 51 up to 246 bytes in maximum. Be sure that the corresponding slave the connection shall be established to will support this size in its receive buffer, else it would deny the connection initalization.
bIndLen_h: this value contains the maximum number of high prior PMS-PDUs data bytes in receive direction in the view of the master. High prior messages are

not allowed so the value must be set to 0.
bIndLen_l: this value contains the maximum number of low prior PMS-PDUs data bytes in receive direction in the view of the master. This value can range from minimum 51 up to 246 bytes in maximum. The choice which value should be taken here depends on the slave's maximum possible send length. So the receive value must be greater or at least equal to the maximum transmit length of the slave.
The abServSup[...] is an array of 6 bytes. The first three bytes define in a bit list,. which services the master could request through this connection as a client from the slave. The bytes 4 to 6 defines which services the master must be able to receive in case of requests from the slave. The field is checked during the establishment of the connection. If the request bit list then indicates more supported services then the slave on its side will support, the connection establishment will be denied. Each pair of these three bytes have the same definition in their bit constellation, but they can be configured different. The most connections support the services READ and WRITE, so the corresponding bits in the field are usually set. The service GET_OD = get object dictionary is not necessary to be configured at all, but is often helpful to get out the objects that are supported by the slave.
Here is the definition of these bits:

Byte 1 and byte 4:

D7	D6	D5	D4	D3	D2	D1	D0
GET_OD	n.support	n.support	n.support	n.support	n.support	n.support	n.support

Byte 2 and byte 5:

D7	D6	D5	D4	D3	D2	D1	D0
n.support	n.support	READ	WRITE	n.support	n.support	n.support	n.support

Byte 3 and byte 6:

D7	D6	D5	D4	D3	D2	D1	D0
INF-REP.	n.support	n.support	n.support	n.support	n.support	n.support	n.support

abSymbol[..]: This variable can contain the symbolic name of the communication reference. The first byte contains the length and the following next 11 bytes contain the name.
ptVfdPointer: this pointer is unused and must be configured to the value 0

Download example of a device parameter data set with the address 4, without using the sequenced download procedure.

command message			
variable	type	value	signification
msg.rx	byte	3	receiver = IBM-Task
msg.tx	byte	16	transmitter = user at HOST
msg.ln	byte	0-240	length of message
msg.nr	byte	j	number of the message
msg.a	byte	0	no answer
msg.f	byte	0	error, status
msg.b	byte	68	command = IBM_Download
msg.e	byte	k	extension
msg.d[0]	byte	0	Req_Adr, not used
msg.d[1]	byte	4	Area_Code = device address
msg.d[2]	word	0	Add_Offset, 0 = beginning of the internal buffer
msg.d[4]	word	???	Device_Para_Len = length of the following data set + 2
msg.d[6]	byte	0x80	Dv_Flag = ACTIVE
msg.d[7]	byte	0x03	ID_Code = 0x03, input and output module
msg.d[8]	byte	0x01	Length_Code = 16 bit data width
msg.d[9]	byte	0x00	Install_Depth = level 0
msg.d[10-19	10 bytes	0	Octet1 - Octet10 (reserved)
msg.d[20]	word	???	Dev_Cfg_Data_Len = size of the following Dev_Cfg_Data_Table +2
msg.d[22]	structure		Dev_Cfg_Data_Table
msg.d[...]	word	???	Dev_Add_Tab_Len, length of following Dev_Add_Tab + 2
msg.d[...]	byte	???	Input_count, number of following input offsets
msg.d[...]	byte	???	Output_count, number of following output offsets
msg.d[...]	word array		IO_Offsets[...], byte offsets in the dual-port memory where to locate the data
msg.d[...]	word	2	Dev_Pcp_Data_Len fixed to value 2

4.2.6 Example of Message Device Parameter Data Sets hexdecimal

No Input and output: IBS 24 BK-T, branch interface

03 10 1C 08 00 00 44 00, message header
00 00 00 00, rem adr etc
18 00 80 00 34 00 00 00 00 00 00 00 00 00 00 00, ID=34 length = 0
02 00 , no input and output modules
04 00 00 00 , no input and output offset
02 00 , for PCP extention

Input only: IBS 24 DI/LC-Local bus device/Level 1

03 10 20 08 00 00 44 00, message header
00 01 00 00, rem adr etc
1C 00 80 01 96 01 00 00 00 00 00 00 00 00 00 00, ID=96 length = 1
04 00 10 00, 16 bit input module
06 00 01 00 00 00, 1 input offset = 0
02 00, for PCP extention

Output only: IBS 24 DO/LC-Local bus device/Level 1

03 10 20 08 00 00 44 00, message header
00 02 00 00, rem adr etc
1C 00 80 01 95 01 00 00 00 00 00 00 00 00 00 00, ID=95 length = 1
04 00 10 80, 16 bit output module
06 00 00 01 00 00, 1 output offset = 0
02 00, for PCP extention

Input / Output: IBS 24 BK-I/O-T branch interface/Level 0

03 10 24 08 00 00 44 00, message header
00 03 00 00, rem adr etc
20 00 80 01 0B 00 00 00 00 00 00 00 00 00 00 00, ID=0B length = 1
06 00 10 80 10 00, 16 bit output module/16 bit input module
08 00 01 01 02 00 02 00, 1 input offset = 2 / 1 output offset = 2
02 00, for PCP extention

Input / Output: CIF 30-IBS, PCP-capable, services READ,WRITE, max PDU size = 64

03 10 51 08 00 00 44 00, message header
00 03 00 00, rem adr etc
4D 00 80 15 F0 00 00 00 00 00 00 00 00 00 00 00, ID = F0, length = 15
06 00 80 00 80 80,   80(128dec) bit input module / 80 bit output module
08 00 01 01 07 00 02 00, 1 input offset = 7 / 1 output offset = 2
2F 00 02 80 80 01 00 00 00 01 01 01 01, CR = 2, L/RLsap = 80, maxXcc = 1
00 00 00 00 00 00 80 00, Aci = 0 , multiplier = 80
00 40 00 40 00 30 00 00 30 00, req.length = 40, ind length = 40, READ,WRITE
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

4.2.7 IBM_Device_Diag

command message			
variable	type	value	description
msg.rx	Byte	3	receiver = IBM-Task
msg.tx	Byte	16	transmitter = HOST
msg.ln	Byte	8	length of message header = 8
msg.nr	Byte	j	number of message (optional)
msg.a	Byte	0	no answer number
msg.f	Byte	0	no error
msg.b	Byte	66	command : IBM_Device_Diag
msg.e	Byte	0	unused
msg. DeviceAdr	Byte	0...127	Dev_Adr
msg. DataArea	Byte	0	unused
msg. DataAdr	Word	0	unused
msg. DataIdx	Byte	0	unused
msg. DataCnt	Byte	0	unused
msg. DataType	Byte	0	unused
msg. DataFnc	Byte	0	unused

The command serves to read out the internally stored device specific diagnostic structures. The device number must be fixed in `msg.DeviceAdr` and corresponds to the physical position of the device module in the network. The value range goes from 0 to 127.

The diagnostic structure can be requested anytime from the DEVICE. The corresponding status bit (`Sl_diag`) of the device in the global bus status field indicates, if the status structure of it has changed since the HOST last read access. So the HOST has to request the status diagnostic information of a device only then, when its relevant bit in `Sl_diag` area in the global status field is set.

On each read access of the HOST the error buffer will be deleted and refilled with zero values. The counter for the actual stored error values will also be reseted, next to the corresponding `Sl_diag` bit of the device.

answer message			
variable	type	value	signification
msg.rx	byte	16	receiver = user at HOST
msg.tx	byte	3	transmitter = IBM-Task
msg.ln	byte	8+107max	length of message
msg.nr	byte	j	number of the message
msg.a	byte	66	answer = IBM_Device_Diag
msg.f	byte	0	error, state
msg.b	byte	0	no command
msg.e	byte	0	extension
msg. DeviceAdr	byte	0...127	Dev_Adr
msg. DataArea	byte	0	data area, unused
msg. DataAdr	word	0	data address unused
msg. DataIdx	byte	0	data index unused
msg. DataCnt	byte	107	data count = length of diagnosis structure
msg. DataType	byte	0	data type unused
msg. Function	byte	0	function read unused
msg.d[0]	byte		Devicestatus_1
msg.d[1]	byte		Real_length_code
msg.d[2]	byte		Real_ident_code
msg.d[3-4]	word		Num_of_CRC_errors
msg.d[5]	byte		Online_error
msg.d[6]	byte		Num_of_entries
msg.d[7... 106max]	union		Error_Data[...]

The reading of the diagnostic information of a device causes the reset of the corresponding diagnostic bit in the 'global bus status field' of the dual-port memory. Should the remote address in `msg.DeviceAdr` be out of range, the answer message delivers the error code 161. Otherwise no error is recognized and the message contains valid diagnostic data.
Every time the diagnostic field is read out, the internal `Num_Of_Entries` counter will be reseted and the `Event_Data` field is cleared with 0.

`Devicestatus_1`:

D7	D6	D5	D4	D3	D2	D1	D0
Deactivated	Interface_1_Error	Interface_2_Error	Reconfiguration	Cfg_ Fault	Peripheral_Fault	Error_Buff_Ovfl.	No_Response
							device not responding and missing
						Error buffer overflow	
					device has detected perihperal power failure or short circuit.		
				differences between device ident or length code and the configured values. So check both configured value in SyCon or in the online downloaded configuration.			
			device reports reconfiguration request				
		outgoing interface 2 of the device is defective and causes a timeout. The interface was switched off by the DEVICE.					
	outgoing interface 1 of the device is defective.and causes a timeout. The interface was switched off by the DEVICE						
device is deactivated in actual configuration and not handled. To enable the handling of this device run SyCon tool and activate it in the actual configuration or change the download configuration value 'active' in variable Dv_Flag when using the online download method							

`Real_length_code`:

This value is read out from the device directly and inserted here transparent. In case of a configuration error this value can be compared with the configured value.

`Real_ident_code`:

This value is read out from the device directly and inserted here transparent. In case of a configuration error this value can be compared with the configured value.

`Num_of_CRC_errors`:

For each reported checksum error this counter will be incremented globally. An increasing counter is an indication for electrical disturbance in the field bus in the surrounding of the device.

`Online_error`:

In this byte the actual online error of the device station is held down. See the table `Err_event` of the global bus status field for possible entries.

Num_of_entries:

This value indicates how many entries the following Error_data buffer contains. For each reported and detected error its number is stored into this buffer while the Num_of_Entries value is incremented by one.

Error_Data[...]:

variable name	type	explanation
Error[0]	byte	detected error. Value range is decribed below in the next table
Reserved[0]	byte	no used, but reserved
... if more errors are detected		
Error[x]		detected error. Value range is decribed below in the next table
Reserved[x]		no used, but reserved

Error[...]:

Error	description	error source	help
0	no error event		
30	device was missing in the last activated network scan cycle	device / configuration	check if the configured module is present in the network or check wiring
31	device reports other identification code than the configured value	device / configuration	compare configured identification code of the module with the real present one
32	device reports other length code than the configured value	device / configuration	compare configured length code of the module with the real present one
33	further device at outgoing interface 1detected which are not configured	device / configuration	check the real configuration for these non configured devices
34	further device at outgoing interface 2 detected which are not configured	device / configuration	check the real configuration for these non configured devices
35	device was missing in the last activated network scan cycle	device / configuration	serach the whole branch where the device is located for other configuration faults
36	device reports peripheral error	device	check if the power of the external periphery of this module is connected or if outputs producing short circuits

Error	description	error source	help
37	device reports configuration request	device	reset the master DEVICE and the InterBus will be reconfigurated
38	device has detected a checksum error while data transmission	device	check surrounding of the device if some other electrical disturbing devices can be found
40	defective outgoing interface 1(local bus branch or installation branch)	device	check the wiring of the corresponding IB interface
41	defective outgoing interface 2(remote bus)	device	check the wiring of the corresponding IB interface
42	device has not reported its ident and length code right in the last made network scan cycle	network	check surrounding of the device if some other electrical disturbing devices can be found
43	device missed during runtime, because of interrupted InterBus connection	network	check network wiring between this device and the physically present device before
44	the contact to this module was lost, because of an interrupted network connection in a local bus branch	local bus branch	check network wiring between this device and the physically present devices before
45	in the last made network scan cycle during runtime, this device was the physically last one to which the DEVICE could establish the InterBus scan	network	check network wiring between this device and the physically present device behind
46	the connection to this module was forced stopped	HOST program	the HOST forced the DEVICE to shut down the communication to all devices

See below the corresponding structure in the header file:

```
IBM_SINGLE_DEVICE_DIAGNOSTIC
```

4.2.8 IBM_Get_Physical_Configuration

	command message			
	variable	type	value	description
Message header	msg.rx	Byte	3	receiver = IBM-Task
	msg.tx	Byte	16	transmitter = HOST
	msg.ln	Byte	0	unused
	msg.nr	Byte	j	number of message (optional)
	msg.a	Byte	0	no answer number
	msg.f	Byte	0	no error
	msg.b	Byte	75	command : IBM_Get_Physical_Configuration
	msg.e	Byte	0 4 8	get the length code of all connected devices get the ID-Code of all connected devices get the physical installation level of all connected devices

This command serves to read in the actual connected InterBus slave devices in their length and ID-code and their installation level. The `msg.e` byte distinguishes the different functions. When this command is performed the whole connected and running InterBus network will be reseted and the outputs are brought into save zero condition. The PCP communciation to existing connections are also aborted and can not be initialized again. To bring back the card into normal operation you have to perform a cold or warmstart to it.

`Msg.e` must be set to 0 first to start the real physical scan of the network. The values 4 and 8 then will cause no network access any more and the values are reported back from the internal buffer that was filled once at the scan before. If the sequence is not kept in this way, the procedure is denied by the DEVICE with sequence error.

answer message			
variable	type	value	signification
msg.rx	byte	16	receiver = user at HOST
msg.tx	byte	3	transmitter = IBM-Task
msg.ln	byte	x	length of message
msg.nr	byte	j	number of the message
msg.a	byte	75	answer : IBM_Get_Physical_Configuration
msg.f	byte	see table below	error, state
msg.b	byte	0	no command
msg.e	byte	4 8 12	msg.d[...] contains length codes msg.d[...] contains ID-Codes msg.d[...] contains installation levels
msg.d[0]	byte	a	length code or ID-code or installation level of physical first InterBus slave
msg.d[1]	byte	b	length code or ID-code or installation level of physical second InterBus slave
msg.d[x-1]	byte	c	length code or ID-code or installation level of physical x InterBus slave

Possible values for `msg.f` are the following :

error number msg.f		signification
0	`msg.d[0]` contains valid data	no error
150	sequence error, please check msg.e of HOST command message	HOST program
101	expected ID or length code can not be found within a device during set up of the network	network
102	too many devices are connected to the DEVICE	network
103	configuration has changed during the ID-Scan	network
104	set up the actual network configuration after the ID-scan failed	network
105	device which was just scanned produce timeout now	network
106	expected device is missing, while setting up the configuration	network
107	configuration has changed during runtime, a running device is not responding any more	network
108	no connection to the InterBus	network

In case of returning the installation levels, the byte values per slave device contain some further information about the device in its upper nibble.

To get the real installation level use the following mask to filter the lower four bits:

```
#define INSTALLATIONDEPTH_MSK 0x0f
```

The following mask indicates if this device is an remote or local bus device:

```
#define LOCALBUSDEVICE_MSK    0x10
```

The following mask indicates if the outgoing remote interface was detected as phyiscally defective during the ID-scan:

```
#define I2_ERROR_MSK          0x20
```

The following mask indicates if the outgoing branch interface was detected as phyiscally defective during the ID-scan:

```
#define I1_ERROR_MSK          0x40
```

In case of returning the length levels, the byte values per slave device contain some further information about the device in its upper three bits.

To get the real length code use the following mask to filter the lower five bits:

```
#define LENGTH_MSK            0x1F

#define MODULE_ERROR          0x80
#define CRC_ERROR             0x40
#define RECONFIGURATION       0x20
```

4.2.9 IBM_Set_Configuration

command message			
variable	type	value	description
msg.rx	Byte	3	receiver = IBM-Task
msg.tx	Byte	16	transmitter = HOST
msg.ln	Byte	x	number of active slave devices 1..126max
msg.nr	Byte	j	number of message (optional)
msg.a	Byte	0	no answer number
msg.f	Byte	0	no error
msg.b	Byte	76	command : IBM_Set_Configuration
msg.e	Byte	0	unused
msg.d[0]	Byte	1 0	Slave Device 0 enable Slave Device 0 disabe
msg.d[1]	Byte	1 0	Slave Device 1 enable Slave Device 1 disabe
-	-	-	-
msg.d[x-1]	Byte	1 0	Slave Device x-1 enable Slave Device x-1disabe

This command serves to change the active constellation of the connected InterBus network. It is possible to switch on and off specific InterBus devices.
If the `msg.d[i]` of the corresponding slave module `i` it set to 1 = 'enable, the module will be enabled and is taken into the InterBus ring. The next process data cycle after execution of this command will include the modules process data.
If the `msg.d[i]` of the corresponding slave module `i` it set to 0 = 'disable', the module will be disabled. The next process data cycle will not include the modules process data any more.

Pay attention to the following points, which results from physical characteristics of the InterBus.

- Slave devices of a local bus branch are switchable only together. That means, if a device of a local bus branch shall be disabled, it is necessary not only to set `msg.d[i]` of the corresponding slave to 0, but all the other `msg.d[i]` of the other slave modules of this branch too. If the local bus branch shall be enabled again, all `msg.d[i]` of the local bus branch must be set to 1.
- If a device shall be disabled, that is located within the remote bus or installation remote bus, the following `msg.d[i]` of all slaves which actually have a higher or equal installation level must be set to 0 too.

The DEVICE checks both mentioned consistencies within the wished new InterBus constellation. Futhermore the DEVICE checks the `msg.ln` parameter. This parameter must always be equal to the actual number of configured slave devices.

REMARK: The last used `IBM_Set_Configuration` bus constellation which could be executed without error is saved within the DEVICE. If afterwards new bus parameters are configured with the `IBM_Download` command, the DEVICE will perform a network reset first, but set up the bus constellation in accordance to this last `IBM_Set_Configuration` command.

answer message			
variable	type	value	signification
msg.rx	byte	16	receiver = user at HOST
msg.tx	byte	3	transmitter = IBM-Task
msg.ln	byte	x	length of message
msg.nr	byte	j	number of the message
msg.a	byte	76	answer : IBM_Set_Configuration
msg.f	byte	see table below	error, state
msg.b	byte	0	no command
msg.e	byte	0	unused

Possible values for `msg.f` are the following :

error number msg.f	explanation	signification
0	command could be executed without an error	no error
101	expected ID or length code can not be found within a device during set up of the network	a device that should be enabled reports a different ID or length code or is missing in the configuration
103	configuration has changed during the ID-Scan	multiple network error
104	inconsistent InterBus branch	more slave devices were detected while switching on a branch than expected
105	InterBus timeout	opening an Interbus branch produces timeout
107	configuration has changed during runtime, a running device is not responding any more	the DEVICE has no access to the InterBus any more. Check wiring between DEVICE and first slave
154	requested mesage inconsistant	msg.ln and number of configured devices do not match or not all local bus module in local bus are disabled or further remote devices are not disable

4.2.10 IBM_Control_Active_Configuration

command message			
variable	type	value	description
msg.rx	Byte	3	receiver = IBM-Task
msg.tx	Byte	16	transmitter = HOST
msg.ln	Byte	x+1	number of slaves to be influenced
msg.nr	Byte	j	number of message (optional)
msg.a	Byte	0	no answer number
msg.f	Byte	0	no error
msg.b	Byte	82	command : IBM_Control_Active_Configuration
msg.e	Byte	0	unused
msg.d[0]	Byte	0,1	bSwitch_Code
msg.d[1]	Byte	0-126	first device_no
-	-	-	
msg.d[x]	Byte	0-126	x device_no

This service allows to selectively activate or deactivate IBS devices.

The parameter `bSwitch_Code` specifies into which states the IBS devices listed in the list of IBS device numbers beginning at `msg.d[1...]` are to be switched. If the value

`#define IBM_SEGMENT_OFF 0x00` is used, the specified IBS device and all devices that depend on this device are switched off. These are:
- all IBS devices belonging to the same bus segment (local bus).
- all IBS belonging to the same logical group.
- IBS devices which come physically after the specified IBS device.

`#define IBM_SEGMENT_ON 0x01` is used, the specified IBS device and all devices that depend on this IBS device are switched on. Please observe the special treatment for groups that can be switched alternatively.

The parameter list `bDevice_No` defines the device numbers of the IBS devices that are to be switched. The value range goes from 0 up to 126 and reflects the physical position of the IBS device in the ring, start counting at the first device and the value 0. The list can consist of an array of devices that shall be influenced. For every stated device the master will switch the state and all dependant other devices.

answer message			
variable	type	value	signification
msg.rx	byte	16	receiver = user at HOST
msg.tx	byte	3	transmitter = IBM-Task
msg.ln	byte	x	length of message
msg.nr	byte	j	number of the message
msg.a	byte	82	answer : IBM_Control_Active_Configuration
msg.f	byte	see table below	error, state
msg.b	byte	0	no command
msg.e	byte	0	unused

Possible values for `msg.f` are the following :

error number msg.f	explanation	detailed information
0	command could be executed without an error	no error
101	expected ID or length code can not be found within a device during set up of the network	a device that should be enabled reports a different ID or length code or is missing in the configuration
103	configuration has changed during the ID-Scan	multiple network error
104	inconsistent InterBus branch	more slave devices were detected while switching on a branch than expected
105	InterBus timeout	opening an Interbus branch produces timeout
107	configuration has changed during runtime, a running device is not responding any more	the DEVICE has no access to the InterBus any more. Check wiring between DEVICE and first slave
109	active,inactive conflict	an IBS device should be activated, but at least one dependant physically previous slave is still disable. This slave must be enabled before.
110	alternative conflict	an alternative group should be activated while a second alternative is already active in the same group. Switch off this alternative group first, before enabling the wished alternative

4.3 The ALPMLIPD-Task

The function of data transfer in PCP protocol is divided into individual layers within the InterBus. Each layer has a defined functionality in this kind of InterBus communication. In our InterBus PCP implementation all layers are combined together in one task called ALPMLIPD. So it builds up the layer ALI, PMS and LLI in one step. The ALPMLIPD supports the follwing services:

IBM_Identify	read out the identification information of a slave device
IBM_Get_Objectdictionary	read out the object dictionary and description of a slave device
Client-Services:	
IBM_Read_Request	read a defined object of a device
IBM_Write_Request	write a defined object of a device with a value
Server-Services:	
IBM_Read_Indication	read a defined object from HOST
IBM_Write_Indication	write a defined object to HOST
IBM_Abort	close an established communication of a device

4.3.1 IBM_Identify

command message			
variable	type	value	description
msg.rx	Byte	1	receiver = ALPMLIPD-Task
msg.tx	Byte	16	transmitter = HOST
msg.ln	Byte	8	length of extended header
msg.nr	Byte	j	number of message (optional)
msg.a	Byte	0	no answer
msg.f	Byte	0	no error
msg.b	Byte	17	command : IBM_Request
msg.e	Byte	0	unused
msg. DeviceAdr	Byte	2-63	communication reference
msg. DataArea	Byte	0	unused
msg. DataAdr	Word	0	unused
msg. DataIdx	Byte	0	unused
msg. DataCnt	Byte	0	unused
msg. DataType	Byte	0	unused
msg. DataFnc	Byte	129	IBM_Identify

The service Identify serves to read identification information from a slave device. The command message does not have any further specific parameters.

answer message			
variable	type	value	signification
msg.rx	Byte	16	receiver = HOST
msg.tx	Byte	1	transmitter = ALPMLIPD-Task
msg.ln	Byte	56	length of message
msg.nr	Byte	j	number of the message
msg.a	Byte	17	answer = IBM_Confirmation
msg.f	Byte	f	error, state see corresponding table
msg.b	Byte	0	no command
msg.e	Byte	0	extension
msg. DeviceAdr	Byte	2-63	communication reference
msg. DataArea	Byte	0	unused
msg. DataAdr	Word	0	unused
msg. DataIdx	Byte	0	unused
msg. DataCnt	Byte	x	number of read data bytes maximum 240 bytes
msg. DataType	Byte	0	unused
msg. Function	Byte	129	IBM_Identify
msg.d[0...15]	Byte Array	ASCII-string	vendor_name[16]
msg.d[16...31]	Byte Array	ASCII-string	model_name[16]
msg.d[32...47]	Byte Array	ASCII-string	revision[16]

The resulting response message contains the slave device specific identification information, Vendor_Name, Model-Name and Revision, that was delivered back from the device.

Each string describes in ASCII format the given information of the device. The first byte in each of the 3 16-bytes strings defines the length of the containment.

4.3.2 IBM_Get_ObjectDictionary

command message			
variable	type	value	description
msg.rx	Byte	1	receiver = ALPMLIPD-Task
msg.tx	Byte	16	transmitter = HOST
msg.ln	Byte	8 20	length in case of index access length in case of variable name access
msg.nr	Byte	j	number of message (optional)
msg.a	Byte	0	no answer
msg.f	Byte	0	no error
msg.b	Byte	17	command : IBM_Request
msg.e	Byte	0	unused
msg. DeviceAdr	Byte	2-63	communication reference
msg. DataArea	Byte	0,1,2	access specification
msg. DataAdr	Word	0-65535	object index
msg. DataIdx	Byte	0	unused
msg. DataCnt	Byte	0	unused
msg. DataType	Byte	0	unused
msg. DataFnc	Byte	132	IBM_Get_OD
msg.d[0]	Byte		Stringlength of variable name
msg.d[1...11]	Byte array		variable name, 0 terminated.

With the Get_ObjectDictionary service it is possible to read one or more object descriptions from the device. The service distinguishes the long from and the short form description which can be requested. The long form is optional and must be supported by the slave device. The number of transmitted Object descriptions depend on their length and the maximum PDU transmit size.

```
#define ACC_SPEC_GETOV_ALL_INDEX        0
#define ACC_SPEC_GETOV_ALL_NAME         1
#define ACC_SPEC_GETOV_STARTINDEX       2
#define ACC_SPEC_GETOV_INDEX_LONG       0x80
#define ACC_SPEC_GETOV_NAME_LONG        0x81
#define ACC_SPEC_GETOV_STARTINDEX_LONG  0x82
```

In case of the short form access the backcoming object description will not inlcude

- Description
- Password
- Access-Groups
- Access-Rights
- Local-Address
- Name
- Extention

- Local-Adress-OV-OB
- Local-Adress-ST-OV
- Local-Adress-S-OV
- Local-Adress-DV-OV
- Local-Adress-DP-OV

For reading the simple object description the object index must inserted in the `msg.DeviceAdr`. Or in case of name access the name of the variable must be inserted in the `msg.d[0...11]` field. For reading just some or all object descriptions the index of the first (startindex) must be inserted in `msg.DeviceAdr`.

answer message			
variable	type	value	signification
msg.rx	Byte	16	receiver = HOST
msg.tx	Byte	1	transmitter = ALPMLIPD-Task
msg.ln	Byte	56	length of message
msg.nr	Byte	j	number of the message
msg.a	Byte	17	answer = IBM_Confirmation
msg.f	Byte	f	error, state see corresponding table
msg.b	Byte	0	no command
msg.e	Byte	0	extension
msg. DeviceAdr	Byte	2-63	communication reference
msg. DataArea	Byte	0	unused
msg. DataAdr	Word	0-65535	object index
msg. DataIdx	Byte	0	unused
msg. DataCnt	Byte	x	number of read data bytes maximum 240 bytes
msg. DataType	Byte	0	unused
msg. Function	Byte	132	IBM_Get_OD
msg.d[0...240]	Byte Array	VAR_TYP	Object structure see below

```
typedef struct {
         USIGN16  index;
         USIGN8   obj_code;
         USIGN8   nof_elements;
         USIGN16  index_of_type;
         USIGN8   length;
         T_ACCESS access;
         USIGN8   *int_addr;
         STRINGV  symbol[SYMBOL_LEN];
         STRING8  extension[EXTEN_LEN];
} T_VAR_TYP_LONG;

typedef struct {
         USIGN16  index;
         USIGN8   obj_code;
         USIGN8   nof_elements;
         USIGN16  index_of_type;
         USIGN8   length;
} T_VAR_TYP_SHORT;
```

4.3.3 IBM_Read_Request

command message			
variable	type	value	description
msg.rx	Byte	1	receiver = ALPMLIPD-Task
msg.tx	Byte	16	transmitter = HOST
msg.ln	Byte	8	length of extended header
msg.nr	Byte	j	number of message (optional)
msg.a	Byte	0	no answer
msg.f	Byte	0	no error
msg.b	Byte	17	command : IBM_Request
msg.e	Byte	0	unused
msg. DeviceAdr	Byte	2-63	communication reference
msg. DataArea	Byte	0	unused
msg. DataAdr	Word	0-65535	object index
msg. DataIdx	Byte	0-255	object subindex
msg. DataCnt	Byte	0	unused
msg. DataType	Byte	0	unused
msg. DataFnc	Byte	1	IBM_Read

The HOST command serves to read out a specific object from a PCP capabable slave device. The communication reference defines the communication partner which shall be addressed with this service. This value must be fixed in `msg.DeviceAdr`. The value 0,1 are reserved and cannot be used. The master supports up to 64 communication references, so the upper limit is 63 for this value. The object index in `msg.DataAdr` and object subindex in `msg.DataIdx` are fixing the object to be read. A subindex of zero refers always to the entire object - thus permitting even a complete array to be read. A subindex not equal to zero refers to an element in an array and is not permissible for simple objects. Please note that the first element of an array has the subindex 1. The service IBM_Read = 1 must be inserted in `msg.DataFnc`.

4.3.4 IBM_Read_Confirmation

	answer message			
	variable	type	value	signification
	msg.rx	byte	16	receiver = HOST
	msg.tx	byte	1	transmitter = ALPMLIPD-Task

	msg.ln	byte	8+x
			8
			12
				if msg.f = 0 length of message
				if msg.f != 0 and if msg != 0x81
				if msg.f = 0x81

	msg.nr	byte	j	number of the message
	msg.a	byte	17	answer = IBM_Confirmation
	msg.f	byte	f	error, state see corresponding table
	msg.b	byte	0	no command
	msg.e	byte	0	extension
	msg. DeviceAdr	byte	2-63	communication reference
	msg. DataArea	byte	0	unused
	msg. DataAdr	word	0-65535	object index
	msg. DataIdx	byte	0-255	object subindex
	msg. DataCnt	byte	x	number of read data bytes maximum 240 bytes
	msg. DataType	byte	0	unused
	msg. Function	byte	1	IBM_Read
read data	msg.d[0...(x-1)]	byte array		read data
or instead of				
optional additional error description in case of msg.f = 0x81	msg.d[0]	byte	0-255	Error Class
	msg.d[1]	byte	0-255	Error Code
	msg.d[2-3]	word	0-65535	Additional Code

The DEVICE answer message contains the read data as a transparent byte stream in the `msg.d[...]` location. The number of read data bytes is fixed in `msg.DataCnt`. In case of an error the variable `msg.f` contains an error code of the corresponding error table described in one of the next chapters called 'Error Codes in PCP Protocol'.

If the slave device denies the access of the wished service and the DEVICE responds with `msg.f` error `0x81`, then additional error information is included in the response message. This four byte structure information is directly taken from the slaves negative response message transparently. The meaning of the different Error Classes and Codes are defined by the slaves manufactuerer and are normally explained in the description manual of the slave product itself.

4.3.5 IBM_Write_Request

command message			
variable	type	value	description
msg.rx	Byte	1	receiver = ALPMLIPD-Task
msg.tx	Byte	16	transmitter = HOST
msg.ln	Byte	8+x	length of message
msg.nr	Byte	j	number of message (optional)
msg.a	Byte	0	no answer
msg.f	Byte	0	no error
msg.b	Byte	17	command : IBM_Request
msg.e	Byte	0	unused
msg. DeviceAdr	Byte	2-63	communication reference
msg. DataArea	Byte	0	unused
msg. DataAdr	Word	0-65535	object index
msg. DataIdx	Byte	0-255	object subindex
msg. DataCnt	Byte	x	number of data bytes to be written maximum 240bytes
msg. DataType	Byte	0	unused
msg. DataFnc	Byte	2	IBM_Write
msg.d[0...(x-1)]	byte array		write data

The HOST command serves to write a specific object of a PCP capable slave device. The communication reference defines the communication partner which shall be addressed with this service. This value must be fixed in `msg.DeviceAdr`. The value 0,1 are reserved and cannot be used. The master supports up to 64 communication references, so the upper limit is 63 for this value. The object index in `msg.DataAdr` and object subindex in `msg.DataIdx` are fixing the object to be read. A subindex of zero refers always to the entire object - thus permitting even a complete array to be written. A subindex not equal to zero refers to an element in an array and is not permissible for simple objects. Please note that the first element of an array has the subindex 1. The service IBM_Write = 2 must be inserted in `msg.DataFnc`.

4.3.6 IBM_Write_Confirmation

answer message			
variable	type	value	signification
msg.rx	byte	16	receiver = HOST
msg.tx	byte	1	transmitter = ALPMLIPD-Task

msg.ln	byte	8
		12
			if msg.f != 0x81
			if msg.f = 0x81

msg.nr	byte	j	number of the message
msg.a	byte	17	answer = IBM_Confirmation
msg.f	byte	f	error, state see corresponding table
msg.b	byte	0	no command
msg.e	byte	0	extension
msg. DeviceAdr	byte	2-63	communication reference
msg. DataArea	byte	0	unused
msg. DataAdr	word	0-65535	object index
msg. DataIdx	byte	0-255	object subindex
msg. DataCnt	byte	0	unused
msg. DataType	byte	0	unused
msg. Function	byte	2	IBM_Write
msg.d[0]	byte	0-255	Error Class
msg.d[1]	byte	0-255	Error Code
msg.d[2-3]	word	0-65535	Additional Code

optional additional error description in case of msg.f = 0x81

The DEVICE answer informs about success or failure of the requested service. In case of an error the variable msg.f contains an error code of the corresponding error table described in one of the next chapters called 'Error Codes in PCP Protocol'.

If the slave device denies the access of the wished service and the DEVICE responds with msg.f error 0x81, then additional error information is included in the response message. This four byte structure information is directly taken from the slaves negative response message transparently. The meaning of the different Error Classes and Codes are defined by the slaves manufactuerer and are normally explained in the description manual of the slave product itself.

4.3.7 IBM_Read_Indication

command message			
variable	type	value	description
msg.rx	Byte	16	receiver = HOST
msg.tx	Byte	1	transmitter = ALPMLIPD-Task
msg.ln	Byte	8	length of extended header
msg.nr	Byte	j	number of message (optional)
msg.a	Byte	0	no answer
msg.f	Byte	0	no error
msg.b	Byte	17	command : IBM_Indication
msg.e	Byte	0	unused
msg. DeviceAdr	Byte	2-63	communication reference
msg. DataArea	Byte	0	unused
msg. DataAdr	Word	0-65535	object index
msg. DataIdx	Byte	0-255	object subindex
msg. DataCnt	Byte	0	unused
msg. DataType	Byte	0	unused
msg. DataFnc	Byte	1	IBM_Read

The DEVICE indication serves to read a specific object from the HOST. When the HOST receives the service and has finished its execution, then the corresponding `IBM_Read_Response` service has to be send back to the DEVICE.

The object index in `msg.DataAdr` and object subindex in `msg.DataIdx` are fixing the wished HOST object to be read.

4.3.8 IBM_Read_Response

answer message			
variable	type	value	signification
msg.rx	byte	1	receiver = ALPMLIPD-Task
msg.tx	byte	16	transmitter = HOST
msg.ln	byte	8+238max	length of message
msg.nr	byte	j	number of the message
msg.a	byte	17	answer = IBM_Response
msg.f	byte	f	error, state see corresponding table
msg.b	byte	0	no command
msg.e	byte	0	extension
msg. DeviceAdr	byte	2-63	communication reference
msg. DataArea	byte	0	unused
msg. DataAdr	word	0-65535	object index
msg. DataIdx	byte	0-255	object subindex
msg. DataCnt	byte	1-240	number of read data bytes
msg. DataType	byte	0	unused
msg. Function	byte	1	IBM_Read
msg.d[0-237]	byte array		read data

This HOST command build the answer to a previously requested `IBM_Read_Indication` message. The HOST has to hand over the data of the corresponding object index and subindex in the `msg.d[...]` field.

4.3.9 IBM_Write_Indication

command message			
variable	type	value	description
msg.rx	Byte	16	receiver = HOST
msg.tx	Byte	1	transmitter = ALPMLIPD-Task
msg.ln	Byte	8+238max	length of message
msg.nr	Byte	j	number of message (optional)
msg.a	Byte	0	no answer
msg.f	Byte	0	no error
msg.b	Byte	17	command : IBM_Indication
msg.e	Byte	0	unused
msg. DeviceAdr	Byte	2-63	communication reference
msg. DataArea	Byte	0	unused
msg. DataAdr	Word	0-65535	object index
msg. DataIdx	Byte	0-255	object subindex
msg. DataCnt	Byte	0	number of data bytes to be written
msg. DataType	Byte	0	unused
msg. DataFnc	Byte	2	IBM_Write
msg.d[0-237]	byte array		write data

The DEVICE indication serves to write a specific object of the HOST. When the HOST receives the service and has finished its execution, then the corresponding `IBM_Write_Response` service has to be send back to the DEVICE.

The object index in `msg.DataAdr` and object subindex in `msg.DataIdx` are fixing the wished HOST object to be written. The data that should be written is located in `msg.d[...]` area.

4.3.10 IBM_Write_Response

answer message			
variable	type	value	signification
msg.rx	byte	1	receiver = ALPMLIPD-Task
msg.tx	byte	16	transmitter = HOST
msg.ln	byte	8	length of extended header
msg.nr	byte	j	number of the message
msg.a	byte	17	answer = IBM_Response
msg.f	byte	f	error, state see corresponding table
msg.b	byte	0	no command
msg.e	byte	0	extension
msg. DeviceAdr	byte	2-63	communication reference
msg. DataArea	byte	0	unused
msg. DataAdr	word	0-65535	object index
msg. DataIdx	byte	0-255	object subindex
msg. DataCnt	byte	0	unused
msg. DataType	byte	0	unused
msg. Function	byte	2	IBM_Write

This HOST command build the answer to a previously requested `IBM_Write_Indication` message.

4.3.11 IBM_Abort

command message			
variable	type	value	description
msg.rx	Byte	1	receiver = ALPMLIPD-Task
msg.tx	Byte	16	transmitter = HOST
msg.ln	Byte	8	length of extended header
msg.nr	Byte	j	number of message (optional)
msg.a	Byte	0	no answer
msg.f	Byte	0	no error
msg.b	Byte	17	command : IBM_Request
msg.e	Byte	0	unused
msg. DeviceAdr	Byte	2-63	communication reference
msg. DataArea	Byte	0	unused
msg. DataAdr	Word	0	unused
msg. DataIdx	Byte	0	unused
msg. DataCnt	Byte	0	unused
msg. DataType	Byte	0	unused
msg. DataFnc	Byte	165	IBM_Abort

The service Abort serves to close a PCP connection to a slave device. The command message does not have any further specific parameters.

answer message			
variable	type	value	signification
msg.rx	byte	16	receiver = HOST
msg.tx	byte	1	transmitter = ALPMLIPD-Task

msg.ln	byte	8
		12
			if msg.f = 0 length of message
			if msg.f = 0x81

msg.nr	byte	j	number of the message
msg.a	byte	17	answer = IBM_Confirmation
msg.f	byte	f	error, state see corresponding table
msg.b	byte	0	no command
msg.e	byte	0	extension
msg. DeviceAdr	byte	2-63	communication reference
msg. DataArea	Byte	0	unused
msg. DataAdr	Word	0	unused
msg. DataIdx	Byte	0	unused
msg. DataCnt	Byte	0	unused
msg. DataType	Byte	0	unused
msg. Function	byte	165	IBM_Abort

optional additional error description in case of msg.f = 0x81	msg.d[0]	byte	0-255	Error Class
	msg.d[1]	byte	0-255	Error Code
	msg.d[2-3]	word	0-65535	Additional Code

4.3.12 Error Codes in PCP Protocol

Definition	No. hex	No. dez	Description
	0x00	0	No error
ALI_INITIATE_ERR	0x41	65	Connection could not be opened By the first request the connection has to be opened by sending an initiate-telegram. If the remote-partner confirmed this initiate negative, the connection could not be opened and the request will be reject with this error. Please check the local and remote configuration of this CR in KBL
ALI_REJECT_PAR_SRV	0x43	67	Too many parallel services on one CR ALI received Reject service with Reject Code for Max-Service-Overflow.
ALI_REJECT_PDU_LENGTH	0x45	69	requested PDU length exceeds the configured maximum PDU length
ALI_REJECT_SRV_NOT_SUPP	0x46	70	requested service is not supported by the client master DEVICE
ALI_REMOTE_ERR	0x81	129	Error in application of remote-partner The communication partner (server) has reject the request with an error. Possible reasons can be for example: a) Access on an non existing object b) Data-length of sending data is not consistent to data-length of object c) buffer overflow
ALI_UNKNOWN_SERVICE	0x82	130	Unknown function in requested message Check the function code in requested message
ALI_LOCAL_ERR	0x83	131	PCP communication basically not or wrong initalized for this slave no InterBus connection to this slave during runtime, connection aborted or the communication reference is basically wrong initialized.
ALI_F_VFD_WRONG_STATE	0x87	135	Local state does not allow to send The master device don't have an actual configuration active, please make a download of the configuration
ALI_F_TIMEOUT	0x8F	143	Timout of remote partner. The service could be sent ot the remote station sucessfully, but the remote station does not answer the request in time
ALI_CR_INVALID	0x97	151	Invalid Communication Reference Please check requested CR parameter in message
ALI_UNKNOWN_SERVICE	0x9B	155	Invalid INTERBUS-PCP service Check service of your request message

5 General Procedure how to get the DEVICE operative without SyCon

Like in the chapters above described, the DEVICE supports the online configuration without using the SyCon configuration tool. That means the DEVICE must be initalized in its protocol parameters first (see. chapter protocol parameters), then a warmstart must be proceeded. After that the network specific parameter must be download via message functionality `Start_Seq, Download, End_Seq`. By using these functions, the network device specific parameters must be downloaded first and then the bus parameter must follow. The download of the bus parameter is the trigger point for the DEVICE to start its network activity the first time. Remember that these download parameter aren't stored in the DEVICE FLASH memory and are lost if the DEVICE is reseted or powered down.
The just described procedure does only work, if the DEVICE isn't configured by SyCon configuration tool, else the found FLASH configuration will always have higher priority then the HOST defined configuration download parameter. To ensure that the DEVICE itself is not preconfigured by SyCon with a static FLASH configuration, for example if you have receive a new delivered one, you have to proceed the following initial sequence to get every DEVICE working:

5.1 Using Device Driver Functions

1. `DevOpenDriver()`: Enable the link of the application to the device driver
2. `DevInitBoard()`: Link application to the specific DEVICE
3. `DevPutTaskParameter()`: Set up the protocol parameter
4. `DevReset(WARMSTART)`: Execute a warm start command to DEVICE
5. `DevGetBoardInfo(GET_DRIVER_INFO)`: Read driver state
6. Examine the variable `bHostFlags` in the backcoming driver state structure
7. If `bHostFlags` indicates the bits `RDY` and `RUN` then the DEVICE is configured by SyCon. Then execute Delete database message to DEVICE by using `DevPutMessage()` and `DevGetMessage()` procedure. Goto step 3 again
8. Use now `DevPutMessage()` and `DevGetMessage()` procedure to download the network specific configuration. After the download of the bus parameter data set the DEVICE automatically starts up the network.

5.2 Using direct access to the dual-port memory

1. Examine the cell `bHostFlags` directly. If cell indicates the bits `RDY` and `RUN` then the DEVICE is configured by SyCon. Then execute `delete database` message (see chapter in this manual) to DEVICE by using corresponding message algorithm described in the `toolkit general definitions` manual. Goto step 1. If cell indicates `RDY` only then goto step 2.
2. Write protocol specific parameter into corresponding `Task2Parameter` area.
3. Write WARMSTART = 0x40 into cell `bDevFlags` to execute the DEVICE's warmstart.
4. Now download the network specific configuration via message procedure. After the download of the bus parameter data set the DEVICE automatically starts up the network.