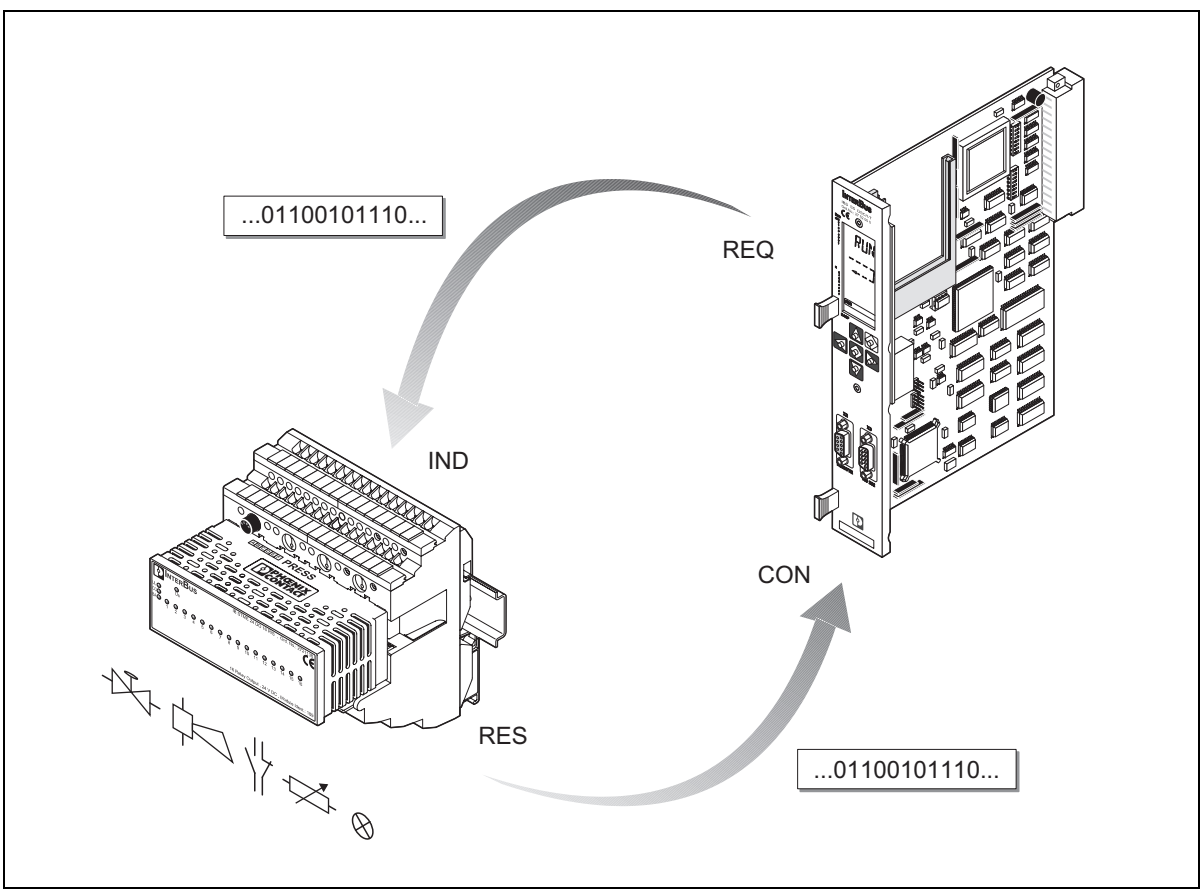
PHOENIX CONTACT
INNOVATION IN INTERFACE

User Manual

Firmware Services and Error Messages

Designation: IBS SYS FW G4 UM E

Order No.: 27 45 18 5



DOWNLOADED FROM WWW.SCATTS.CO.UK

Firmware Services and Error Messages

Designation: IBS SYS FW G4 UM E

Revision: D

Order No.: 27 45 18 5

This user manual is valid for:
Generation 4 Controller Board Firmware 4.60 or Later

5150D

DOWNLOADED FROM WWW.SCATTS.CO.UK

Please Observe the Following Notes:

In order to ensure the safe use of your device, we recommend that you read this manual carefully. The following notes provide information on how to use this manual.

Requirements of the User Group

The use of products described in this manual is oriented exclusively to qualified electricians or persons instructed by them, who are familiar with applicable national standards. Phoenix Contact accepts no liability for erroneous handling or damage to products from Phoenix Contact or third-party products resulting from disregard of information contained in this manual.

Explanation of Symbols Used

The *attention* symbol refers to an operating procedure which, if not carefully followed, could result in damage to hardware and software or personal injury.

The *note* symbol informs you of conditions that must strictly be observed to achieve error-free operation. It also gives you tips and advice on the efficient use of hardware and on software optimization to save you extra work.

The *text* symbol refers to detailed sources of information (manuals, data sheets, literature, etc.) on the subject matter, product, etc. This text also provides helpful information for the orientation in the manual.

We Are Interested in Your Opinion

We are constantly attempting to improve the quality of our manuals.

Should you have any suggestions or recommendations for improvement of the contents and layout of our manuals, we would appreciate it if you would send us your comments. Please use the universal fax form at the end of the manual for this.

Statement of Legal Authority

This manual, including all illustrations contained herein, is copyright protected. Use of this manual by any third party deviating from the copyright provision is forbidden. Reproduction, translation, or electronic

and photographic archiving or alteration requires the express written consent of Phoenix Contact. Violators are liable for damages.

Phoenix Contact reserves the right to make any technical changes that serve the purpose of technical progress.

Phoenix Contact reserves all rights in the case of patent award or listing of a registered design. Third-party products are always named without reference to patent rights. The existence of such rights shall not be excluded.

Internet

Up-to-date information on Phoenix Contact products is available on the Internet at **www.phoenixcontact.com**.

Table of Contents

Section 1

This section provides information about

- Performance characteristics of firmware 4.6x
- Differences between firmware generations G3 and G4
- Basics of communication between PCP devices and controller boards
- Functions for configuration frame, process data, and error/diagnostics management

1 Introduction and Overview

1.1 Introduction

This user manual describes how INTERBUS Generation 4 controller boards are directly accessed using firmware commands.

This version of the firmware manual describes the features of the current **firmware Version 4.6x**. The information can be found in Section **"Performance Characteristics of Firmware 4.6x and Later" on page 1-10**.

The Generation 4 firmware – referred to in the following as G4 – was developed to meet the requirements of a system bus for years to come. This includes restructuring services and modifying the state machine. Generation 4 controller boards can no longer operate with programs based on firmware 3.x.

For more detailed information on the differences between Generation 3 (G3) and Generation 4 (G4) firmware services, please refer to Section 1.2.

The firmware update involves the programming of the master. All INTERBUS devices can still be operated on the bus system. 8-wire remote bus devices are an exception to this rule.

The Generation 4 firmware can be used:

- For user-defined addressing of process data of individual devices using predefined process data descriptions (PDD)
- For the nesting of the bus topology on up to 16 levels
- To support Loop devices
- For various synchronization modes between the host system and the controller board including a defined bus cycle time
- To configure direct link data
- To configure process data preprocessing
- For event-controlled function activation
- To manage and parameterize alternative or changing bus devices
- For improved diagnostics for all devices

Generation 4 firmware thus provides a flexible interface that is almost identical for all controller boards.

As in the distributed shift register, the transmitted data passes through every individual device in the INTERBUS protocol. Each device can read its input data (e.g., input parameters, setpoint values, manipulated variables, etc.) at a position specially reserved for this device. Also, the output data (output parameters, actual values, etc.) can be directly copied to this position. A distinction is made between two mechanisms:

– The process data channel for cyclic I/O data exchange
– The parameter channel for transmitting longer data records

The Peripheral Communication Protocol (PCP) enables data to be transmitted using the parameter channel. The services of this protocol are designed to transmit long parameter records and programs, as well as to control programs using commands. For this, the firmware splits the services into small packets that are copied to the shift register one after the other upon every bus cycle. The destination device then combines this information to recreate the service.

For more detailed information, please refer to the Peripheral Communication Protocol (PCP) User Manual *IBS SYS PCP G4 UM E* (Order No. 27 45 16 9).

The process data channel and the parameter channel access different areas in the INTERBUS summation frame. Even if the parameter channel is subjected to heavy data traffic, the bus cycle time is not affected due to the hybrid structure of the summation frame.

This manual describes all services required to control safe data transmission using the process data channel.

Section 1 Introduces the firmware mechanisms with the aim of enabling the user to assign individual services to special application ranges. In addition, this section

– Provides an overview of the interfaces to the firmware
– Describes the state machine
– Explains the device numbers
– Explains addressing
– Includes the most important service sequences for the configuration
– Describes bus control (using examples)
– Describes error treatment (using examples)

Section 2 Serves as a reference work for the call parameters of all services, i.e., requests and corresponding confirmations.

Section 3 Lists all firmware error messages.

IBS CMD G4

The IBS CMD G4 program is used to configure the system.

IBS CMD G4 software enables interactive and control system-independent configuration, operation, and diagnostics of all devices connected within an INTERBUS network.

IBS CMD G4 runs on standard PCs under MS WINDOWS and can be used for numerous INTERBUS controller boards. The connection to a controller board can be established using three communication paths:

– Using a serial interface (RS-232 level) to the diagnostic interface of the controller board.
– Using the corresponding slot on PC controller boards, e.g., ISA bus.
– Using Ethernet for some Field Controllers.

1.2 Differences Between Generation 3.x and 4.x Firmware

1.2.1 Performance Characteristics

Firmware 3.x

– Unstructured command and message codes.
– A command can be followed by one or two messages or none at all.
– If bit 15 of the message code is set, parameters will follow.

Firmware 4.x

– Structured service codes (request and confirmation codes).

– Every request is confirmed (exception: reset and unconfirmed PCP services).
– When representing the request code in binary notation, the value "0" is always assigned to bit 15. The confirmation code following this request is identical, except for bit 15. Bit 15 of the confirmation always has the value "1". Thus, in hexadecimal notation, the confirmation code is always 8000_{hex} higher than the request code which it follows.

  Example:

Request:	Start_Data_Transfer_Request (0701_{hex})
Confirmation:	Start_Data_Transfer_Confirmation (8701_{hex}) $8701_{hex} = 0701_{hex} + 8000_{hex}$

– All services include a parameter count *(Parameter_Count)*, which indicates how many parameters will follow.
– Positive and negative confirmations have the same message code. The *Result* parameter indicates whether a service has been executed successfully.

1.2.2 Startup Behavior

Firmware 3.x

After the power-on selftest (POST), a Generation 3.x controller board generally reads the connected system configuration (bus configuration) and stores it as an "initial configuration". This configuration can be read from the G3 controller board using the "Send_Physical_Configuration" service. Thus, the user is able to compare the (physically) existing configuration with the planned configuration.

Most applications use the "Check_Physical_Configuration" service to transfer the planned configuration of the G3 controller board. Differences between the planned and present configuration are indicated by the service confirmation.

Firmware 4.x

After the successful power-on selftest (POST), the G4 controller board is in the "Ready" state (RDY). The configuration cannot be read automatically.

When configuring a system, it is advisable to start by creating one system configuration. The "Complete_Load_Configuration" service (see page 2-54) is used to load this configuration onto the controller board. When the loaded configuration is activated with the "Activate_Configuration" service (see page 2-64), the controller board checks it to ensure it corresponds to the configuration that is actually connected. Differences between the planned and present configuration are indicated by the confirmation.

It is also possible to read the connected system configuration into the G4 controller board using the "Create_Configuration" service (see page 2-62). The user can read this system configuration from the G4 controller board using the "Read_Configuration" service (see page 2-38) and compare it with the planned configuration.

1.2.3 Comparison Between Generation 3.x and 4.x Firmware Services

When using a Generation 4 controller board, use the "Alarm_Stop" service (G4) to stop INTERBUS data cycles. This service also resets the outputs. The "Stop_Data_Transfer" service (G4) only stops data cycles.

The "Stop_INTERBUS" service (G3) was used for this purpose on Generation 3 controller boards. The outputs were not reset.

The G3 and G4 services that are compared in Table 1-1 differ with regard to their functions. Please refer to Section 2 "Firmware Services" for the functions of G4 services.

Table 1-1 Comparison between G3 and G4 services

Generation 3 Firmware		Generation 4 Firmware	
Service (G3)		**Service (G4)**	
Request	Confirmation (positive/ negative)	Request	Confirmation
Start_INTERBUS		**Start_Data_Transfer**	
0001_{hex}	0088_{hex}/$00E3_{hex}$	0701_{hex}	8701_{hex}
Alarm_Stop		**Alarm_Stop**	
$004A_{hex}$	$00D8_{hex}$/ —	1303_{hex}	9303_{hex}
Stop_INTERBUS		**Stop_Data_Transfer**	
0002_{hex}	$00C6_{hex}$/ —	0702_{hex}	8702_{hex}
Configure_INTERBUS		**Create_Configuration**	
0023_{hex}	$00CA_{hex}$/ —	0710_{hex}	8710_{hex}
Warm_Start		**Reset_Controller_Board**	
$004C_{hex}$	— / —	0956_{hex}	—
Clear_Display		**Confirm_Diagnostics**	
$004E_{hex}$	$00E2_{hex}$/ —	0760_{hex}	8760_{hex}
Receive_CRL		**Load_CRL_Attributes_Loc**	
0053_{hex}	$00E8_{hex}$/$80F1_{hex}$	0264_{hex}	8264_{hex}
Init_Communication		Not required	
0054_{hex}	$00E9_{hex}$/$80F0_{hex}$		

Table 1-1 Comparison between G3 and G4 services

Generation 3 Firmware		Generation 4 Firmware	
Service (G3)		**Service (G4)**	
Request	Confirmation (positive/ negative)	Request	Confirmation
Check_Physical_Configuration		**Initiate_Load_Configuration**	
0058_{hex}	$00AB_{hex}/0068_{hex}$	0306_{hex}	8306_{hex}
		Complete_Load_Configuration	
		$030A_{hex}$	$830A_{hex}$
		Terminate_Load_Configuration	
		0308_{hex}	8308_{hex}
		Activate_Configuration	
		0711_{hex}	8711_{hex}
Send_Localbus_Module_Error		**Read_Device_State**	
$005B_{hex}$	$80EE_{hex}/00ED_{hex}$	0315_{hex}	8315_{hex}
Send_Physical_Configuration		**Complete_Read_Configuration**	
$005E_{hex}$	$80F4_{hex}/004A_{hex}$	$030B_{hex}$	$830B_{hex}$
Quit_Module_Error_All		**Control_Device_Function**	
0065_{hex}	$00FE_{hex}/80FF_{hex}$	0714_{hex}	8714_{hex}
Read		**Read**	
0081_{hex}	8181_{hex}	0081_{hex}	8081_{hex}
Write		**Write**	
0082_{hex}	8182_{hex}	0082_{hex}	8082_{hex}
Start		**Start**	
0083_{hex}	8183_{hex}	0083_{hex}	8083_{hex}
Stop		**Stop**	
0084_{hex}	8184_{hex}	0084_{hex}	8084_{hex}
Get_OD		**Get_OD**	
0088_{hex}	8188_{hex}	0088_{hex}	8088_{hex}
Initiate		**Initiate**	
$008B_{hex}$	$818B_{hex}/818C_{hex}$	$008B_{hex}$	$808B_{hex}$
Abort		**Abort**	
$008D_{hex}$	—	$088D_{hex}$	—

1.3 Performance Characteristics of Firmware 4.6x and Later

The following firmware functions are available in Version 4.6x and later.

These functions are dependent on the controller board used.

- Devices/bus segments can be switched in isolation in the event of a bus error (see “Isolated Switching in the Event of a Bus Error” on page 1-10).
- The firmware supports up to 8192 I/O points (with the same number of devices as before), depending on the controller board used.
- Up to 126 PCP devices (127 CR entries) can be addressed.
- The functions of the Rugged Line and Inline product ranges, in particular single-channel error messages and optical diagnostics, are also available for the 2 Mbaud transmission rate with this version.
- Optimized connection and disconnection of interfaces
  Generation 4 INTERBUS controllers provide a function for disconnecting the branching local bus and the outgoing remote bus interface as well as the remote bus branch interface, if one is present.
- Handling the parameterization memory
  The services for working with the parameterization memory have been extended. For more detailed information, please refer to Section “Services for Handling the Parameterization Memory” on page 1-11.

1.3.1 Isolated Switching in the Event of a Bus Error

In addition to the existing option of using the application and/or in the event of an error the control program (by sending a user-defined command) to switch devices/segments, branching INTERBUS remote bus branch and local bus interfaces can also be switched in isolation in firmware Version 4.6. This means that data transmission in the rest of the INTERBUS system is not interrupted.

Isolated disconnection is triggered when a bus error occurs. Isolated connection as a response to isolated disconnection depends on the application. The control program reconnects the bus segment once the error has been removed and acknowledged.

Preconditions for isolated switching:

– The system used is a dedicated SUPI 3 system.
– The bus terminal module, which is to be used for isolated switching, contains a SUPI 3 OPC.
– Only branching remote bus branch and local bus interfaces can be disconnected in isolation.

Error localization is not possible within a bus segment that has been disconnected in isolation.

Use the *Error characteristic* parameter in the configuration frame to configure isolated switching.

For additional information about isolated switching, please refer to data sheet DB GB IBS SYS SWITCHING.

1.3.2 Services for Handling the Parameterization Memory

The increasing size of projects requires the use of increasingly larger parameterization memory cards. Essentially, these large parameterization memories (16/32/64 MB) are handled in the same way as smaller memories (1/2/4 MB). The difference lies in the increased time required to complete certain processes, e.g., formatting the card and deleting data from the parameterization memory. In order to optimize handling, firmware services have been expanded and reprogrammed.

More detailed information can be found in Sections “Working With the Parameterization Memory” on page 1-61 and “Services for the Parameterization Memory” on page 2-174.

1.4 General Information

1.4.1 Communication With the Controller Board

The host system and the controller board use a shared memory area for the exchange of information. This memory area is designed either as a Dual-Port Memory (DPM) or as a Multi-Port Memory (MPM) if more than two devices are to be coupled. For greater clarity, the following descriptions are based on a Multi-Port Memory.

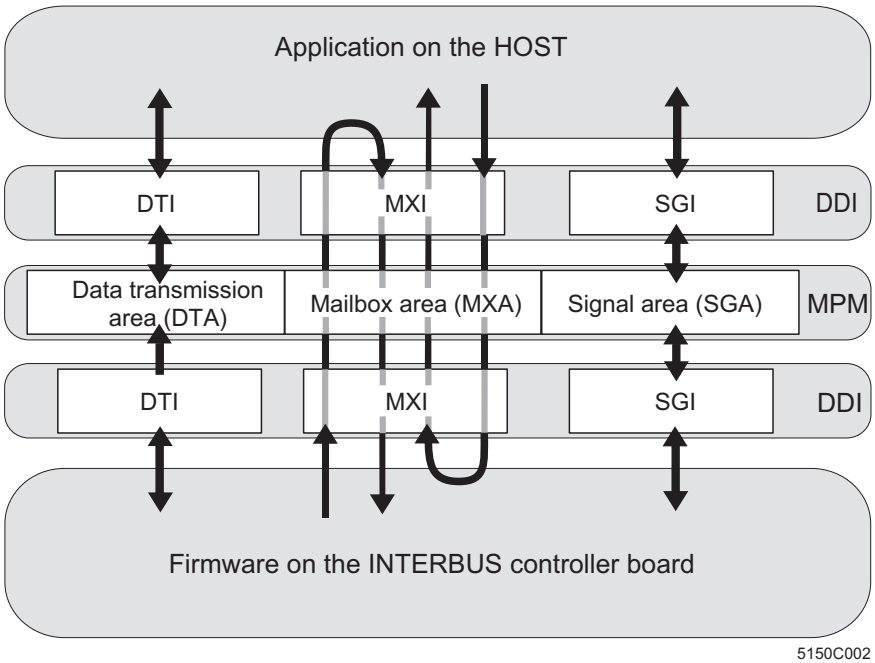


Figure 1-1 Communication mechanisms in the MPM

MPM description

The MPM enables the exchange of information between two or more devices (nodes). In general, the MPM is divided into the data transmission area (DTA), signal area (SGA) and mailbox area (MXA).

The size of the MPM and the location of the individual memory areas depend on the host system. Thus, exact addresses cannot be given. Please refer to the corresponding driver reference manual or the controller board description for this host-specific information.

It is not possible to store all information for all devices in the MPM in the MPM extended diagnostics. Only 251 entries can be stored instead of the possible 253.

Device driver interface

The functions of the device driver interface (DDI) facilitate access to the MPM. Thus, the firmware and the driver do not require hardware or host-specific information. The DDI with its basic functions is identical for all controller boards.

- The devices must open or close their MPM area for communication.
- Two functions are available for reading and writing process data in the DTA, two functions for transmitting and receiving services using the MXA, and two functions for operating the signal interface (SGI).

Process data interface

All input and output data of the connected I/O devices is provided in one memory area of the process data interface (DTI).

The DTI provides read or write access to the data area (DTA) for all MPM devices. From the point of view of the host system, all input data in the MPM is updated with information about the devices and output data is transmitted to the INTERBUS devices on every INTERBUS cycle.

By default, the host and controller board access the MPM in asynchronous mode, although synchronous mode access is available as an option. For this purpose, either the bus cycles are controlled with a signal from the host or the controller board generates a signal for controlling the host.

Almost all controller boards have a standard register set for diagnostics and for basic control of the bus system. The registers are mapped to the input and output area of the control system where they can be freely located (see Section 1.4.2, page 1-16).

Mailbox interface

The mailbox interface (MXI) transfers services between MPM accessors. For this, a protocol is implemented in the MPM enabling safe transmission and reading of messages. Most of the services consist of the request and the corresponding response. The status of the service execution and any additional parameters are transferred with every response.

A message always consists of a service code and the number of parameters, even if no parameters follow (parameter length = 0). For a detailed description of the service structure, refer to the introductory section of the individual services.

Signal interface

The signal interface is divided into the standard signal interface (SSGI) and extended signal interface (XSGI). The standard signal interface (SSGI) has an interface for bit-controlled configuration via predefined address areas in the signal area (SGA). The extended signal interface (XSGI) has an interface for the bit-controlled activation of services that can be parameterized by the user.

1.4.2 Standard Register

The standard register set consists of three diagnostic registers:

– Diagnostic status register (input word)
– Diagnostic parameter register (input word)
– Extended diagnostic parameter register (input word)

and three standard function registers:

– Standard function start register (output word)
– Standard function parameter register (output word)
– Standard function status register (input word)

The position of the registers in the MPM can be changed using the "Set_Value" firmware service (see page 2-18).

Diagnostic register

The controller board diagnostic registers (diagnostic status register, diagnostic parameter register, and extended diagnostic parameter register) represent the current state of the INTERBUS system for the user. Thus, it is possible to indicate the state of the bus system, error causes, as well as additional information to the application program.

Every bit in the diagnostic status register is assigned a state of the controller board. The states in the error bits (USER, PF, BUS, CTRL) are described in greater detail using the diagnostic parameter register and the extended diagnostic parameter register. These registers are always rewritten if one of the previously mentioned error bits is set. Otherwise, the register has the value 0000_{hex}.

By default, the extended diagnostic parameter register is located in the MPM at address $37E6_{hex}$ and can be moved to the I/O area using the "Set_Value" service (variable_ID $010C_{hex}$) (see “"Set_Value" Service” on page 2-18).

Please note that in the extended diagnostic parameter register, the data can only be updated after the "Confirm_Diagnostics_Request" service (0760_{hex}) has been sent.

Diagnostic status register

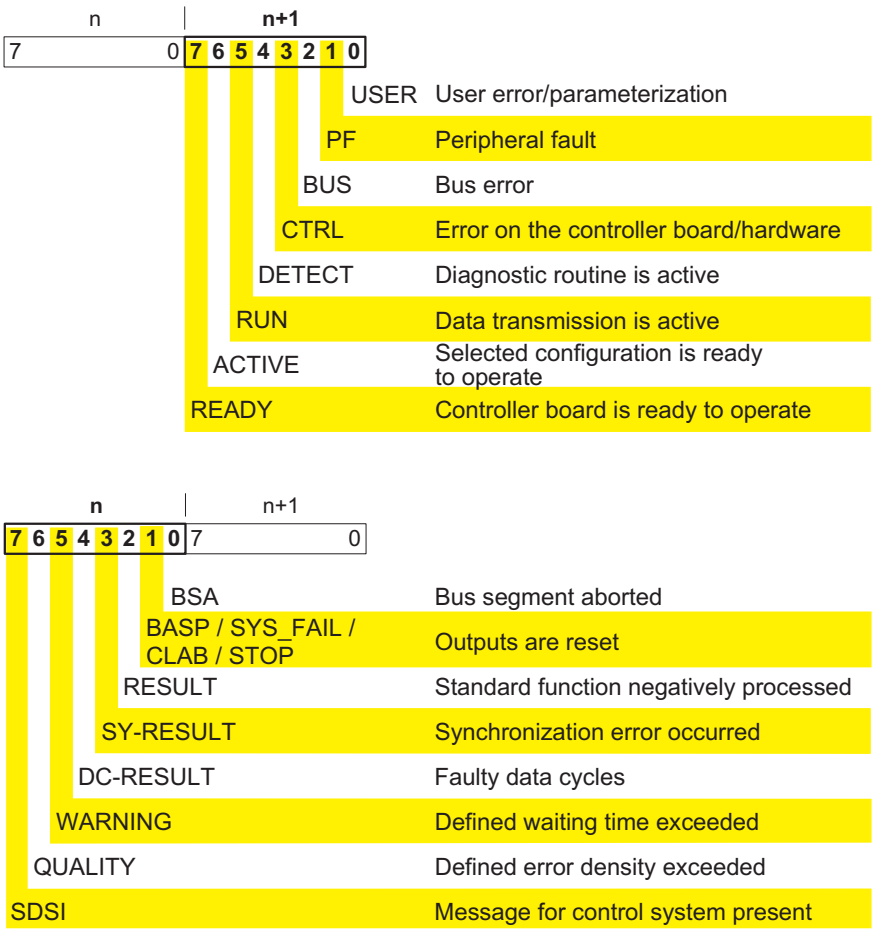


Figure 1-2 Diagnostic status register

Operating indicators

The READY, ACTIVE and RUN operating indicators show the current state of the INTERBUS system. The diagnostic parameter register is not used.

After the selftest the controller board is ready for operation. The READY indicator bit is set (READY = 1).

If the controller board has been configured and the configuration frame activated without errors, the system indicates it is active. The READY and ACTIVE indicator bits are set (READY = 1, ACTIVE = 1).

In addition, the RUN indicator bit is set when data exchange is started (READY = 1, ACTIVE = 1 and RUN = 1).

Error indicators

The DETECT error bit shows that an error is preventing further operation of the bus (DETECT = 1). The outputs fall back to the value ZERO. The diagnostic routine searches for the error cause.

Once the error cause has been detected, the DETECT error bit is reset (DETECT = 0) and the error is indicated in the USER, PF, BUS and CTRL bits. The diagnostic parameter register describes the error cause in more detail.

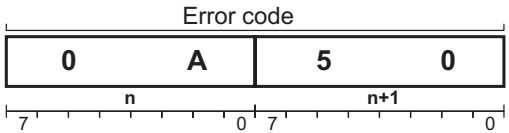
Table 1-2 Errors with bus disconnection

Error Bit/Location	Content of the Diagnostic Parameter Register
CTRL = 1 Probably controller board/hardware error.	Error code, Section 3
BUS = 1 Error on the remote bus or local bus segment.	Error location, Section 3.3

Table 1-3 Errors without bus disconnection

Error Bit/Location	Content of the Diagnostic Parameter Register
PF = 1 Fault on the application side of an INTERBUS device: – Short circuit at the output – Sensor/actuator supply not present	Error location, Section 3.3
USER = 1 User error, e. g. due to incorrect parameters	Error code, Section 3

Error code,
e.g., address overlap
(code $0A50_{hex}$)



Error location,
e.g., device number 3.1

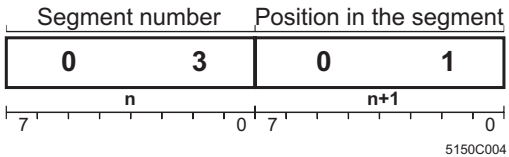


Figure 1-3 Content of the diagnostic parameter register (example)

Error location

For located remote or local bus errors, the diagnostic parameter register contains the error location:

– Error on local bus: device number of device, e.g., "1.3" for bus segment 1; device 3
– Error on remote bus: device number of bus terminal module, e.g., "1.0" for bus segment 1; device 0

The precise error location is only specified if there is no interface error (bit 7 equals 0). If an interface error has occurred (bit 7 equals 1), for example, the connected bus cannot be operated, only the faulty bus segment is specified. Bit 0 indicates whether the error location is on the outgoing remote bus interface (bit 0 equals 0) or on the branching remote bus interface (bit 0 equals 1).

Remember that in some cases, the INTERBUS diagnostics can only provide a restricted error localization function:

– Error localization is limited on an Inline station connected directly to the controller board.
– Under some circumstances, error locations in the Loop branch are mapped to the first available Inline module.

Standard function registers

The standard function registers enable predefined and frequently used functions to be executed by setting an output bit, and enable monitoring of the execution and result.

The standard function start register is used to execute the required function. To call the function, some additional information may be required in the corresponding standard function parameter register.

The following functions are assigned to the bits of the standard function start register:

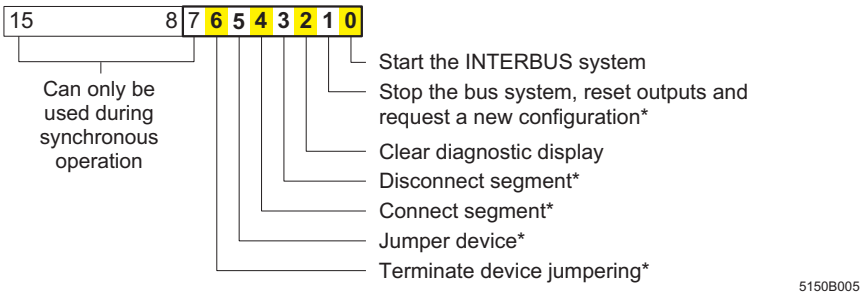


Figure 1-4 Assignment of frequently used functions in the standard function start register

The functions marked with an asterisk (*) require further parameters (e.g., a device number) for their execution. These parameters must be transferred in the separate standard function parameter register.

As only one standard function can ever be activated at a time, a new function can only be executed once the function activated previously has been completed.

The standard function status register controls the processing sequence. Each bit in the standard function start register is assigned a bit of the same name in the standard function status register. When processing a function, the corresponding status bit is active regardless of whether the function could be executed successfully or not. After processing the function, the status bit is set to "0".

The standard function RESULT bit in the diagnostic status register indicates whether the function was executed successfully.

RESULT = 1 Function not completed successfully

RESULT = 0 Function completed successfully

Function execution The following two diagrams illustrate the sequence of a function execution with and without parameter transfer.

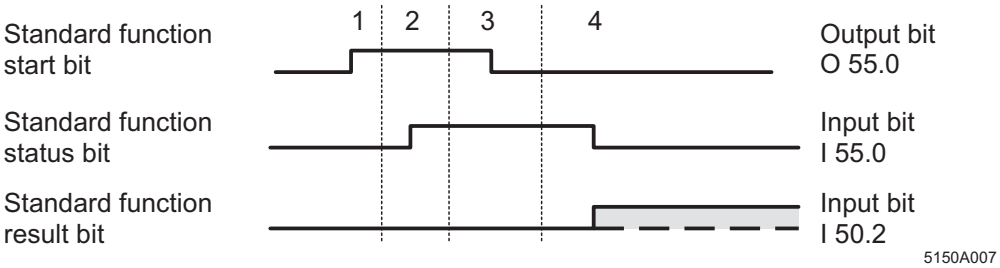


Figure 1-5 Sequence of a function execution without parameter transfer

1. The control program activates the function using the corresponding function bit in the standard function start register.
2. The function execution is indicated using the corresponding status bit in the standard function status register.
3. Resetting the function bit using the host prevents a new function activation.
4. The RESULT bit indicates the result of the function execution in the diagnostic status register.

   RESULT = 0 Function completed successfully
   RESULT = 1 Function not completed successfully

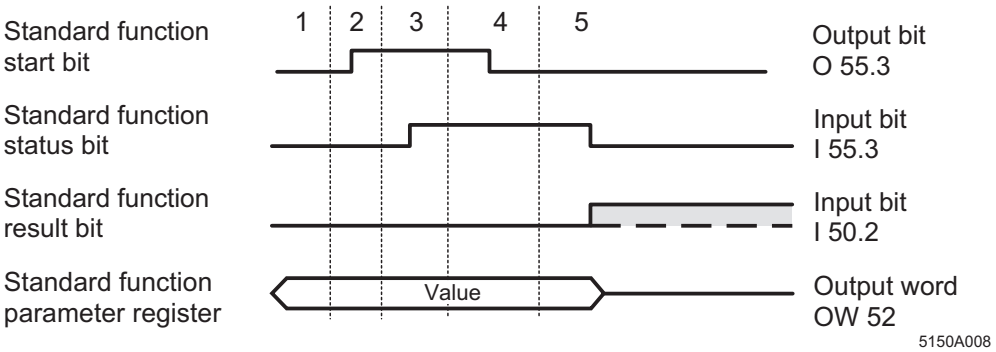


Figure 1-6 Sequence of a function execution with parameter transfer

1. Enter the required parameter in the standard function parameter register.
2. The control program activates the function using the corresponding function bit in the standard function start register.
3. The function execution is indicated using the corresponding status bit in the standard function status register.
4. Resetting the function bit using the host prevents a new function activation.
5. The RESULT bit indicates the result of the function execution in the diagnostic status register.

   RESULT = 0 Function completed successfully
   RESULT = 1 Function not completed successfully

1.4.3 State Machine

The state machine describes the defined sequence of states during INTERBUS startup and operation. In order to respond to errors at an early stage, the reliability of each service is checked using the state machine.

The following figure shows the state machine of the Generation 4 firmware:

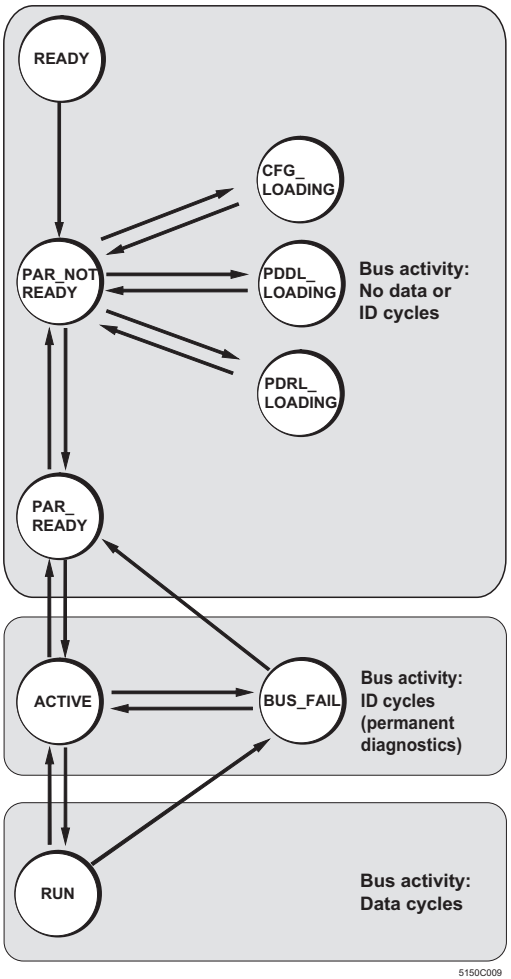


Figure 1-7 The state machine

Meaning of states

READY	The READY state is the initial state of the firmware after every startup (cold or warm start). In this state, no valid configuration is available. The bus cannot be activated.
PAR_NOT_READY	After initiating the parameterization phase, the system management is in the PAR_NOT_READY state. From this state it is possible to initiate the three parameterization sequences for loading the – Configuration frame – Process data description lists (PDDL) – Process data reference lists (PDRL) The validity of the configuration parameters read during the parameterization phase cannot be guaranteed.
CFG_LOADING	To load a configuration frame, the system must be in the CFG_LOADING state.
PDDL_LOADING	To define process data, the system must be in the PDDL_LOADING state.
PDRL_LOADING	To link process data, the system must be in the PDRL_LOADING state.
PAR_READY	After successful completion of the parameterization phase, the firmware is in the PAR_READY state. At least one configuration frame has been defined. This frame was checked for completeness and consistency during parameterization.

In the PAR_NOT_READY, PAR_READY, CFG_LOADING, PDDL_LOADING, and PDRL_LOADING states, the bus is inactive. No data or ID cycles are run.

ACTIVE	The bus is active. Only ID cycles are run. Data is not transmitted by the bus. The bus is operated with the configuration defined as the active configuration in the configuration frame.
RUN	Data cycles are run in this state. The actual transfer of data only takes place in this state.

BUS_FAIL State of the system management if bus errors occur in the ACTIVE or RUN state. ID cycles are still run with the greatest possible configuration.

The ACTIVE, RUN, and BUS_FAIL states can be read directly from the diagnostic status register.

1.4.4 Firmware Command Groups

The firmware services in the system management are closely connected. For clarification, most of the services can be divided into the following groups according to their functions: configuration, bus control, diagnostics, and general functions.

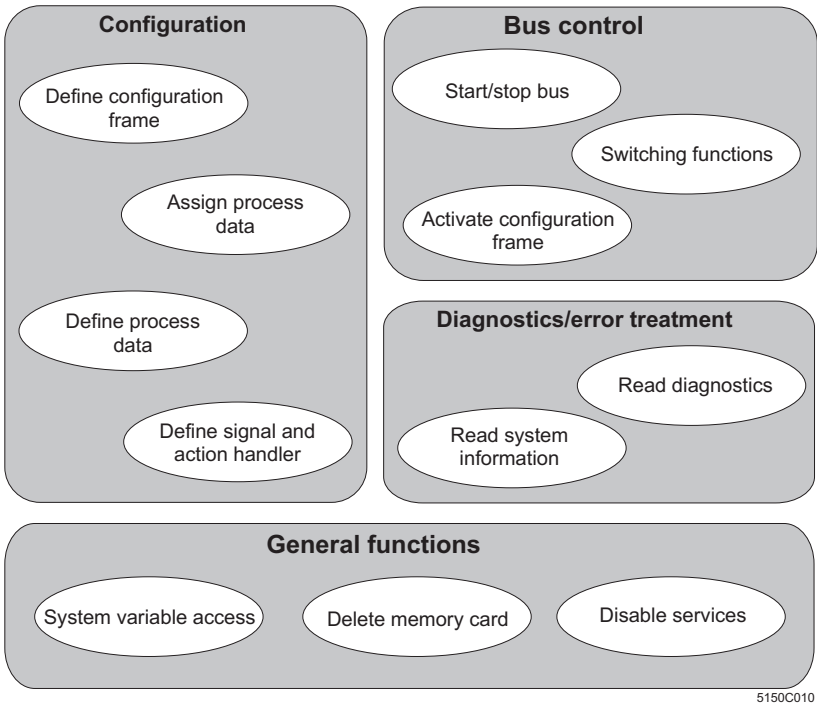


Figure 1-8 Overview of service groups

Configuration

Most services are responsible for the configuration of the controller board. These services are divided into the following areas: configuration frame management, process data management, and action and signal object management.

Configuration frame management

The configuration frame management can be used to process the information required for each device in the configuration frame. The configuration frame describes each of the specified INTERBUS devices in a separate numbered entry. It is possible to configure, delete or compare

several configuration frames. The sequence of the devices within the configuration frame corresponds to the physical bus configuration including all alternatives.

Process data management:

Process data management can be used to set up process data descriptions for each device via the process data description list in addition to assigning addresses in the MPM via the process data reference list. These process data descriptions can be used to address the information in the data word of a device directly down to an individual bit. This means that you can use the assignment of the process data to freely define a logic process data image. Addresses do not have to be changed when the bus configuration is modified - the internal assignment is simply adapted.

Action and signal object management:

Action and signal object management is a flexible mechanism for activating preconfigured functions using control bytes. These preconfigured functions are stored on the controller board. Application-specific responses can be initiated with just a single command. The required transfer parameters should be stored in previously defined addresses within the data area of the MPM.

Bus control

The bus control is responsible for the direct control of the bus system using firmware services. These services can be divided into two groups. First, there are the services for changing the system state on the bus. Second, there are the direct switching functions of individual devices, groups or alternative groups that can be used during operation without affecting the state machine.

Diagnostics/ error treatment

The diagnostic functions provide comprehensive information about the state of the bus system. This includes, for example, statistical information about errors in the bus system for each individual device and information about the hardware and software version.

General functions

The general functions cannot be assigned to one of the groups mentioned above. For example, this group includes services for accessing a memory card, if present, as well as services for accessing system variables.

1.4.5 Configuration Frame

The configuration frame illustrates the configuration of the bus system in table form. The bus system is described fully in terms of its topology and summation frame by the "device identification code" (ID code), "length code", and "bus level" entries, as well as by the entry position within the table.

Device ID:	The device ID indicates the INTERBUS device type and its function.
	Information about the ID code of INTERBUS devices can be found in the INTERBUS device list. Alternatively, contact the INTERBUS Club.
Device length code:	The length code describes the address area of an INTERBUS device and thus the length of the input and output process data in encoded form.
	“Structure of the Device Length Code” on page 1-37 in this section describes the composition of the length code for an INTERBUS device.
Bus level:	The bus level specifies on which level of the INTERBUS tree structure each device is located.

Starting from the controller board, the main remote bus line is at level zero. Bus terminal modules open a new branch (bus level) with their local bus or installation remote bus connection. The structure is limited to 16 levels.

The following entries are required for every device: "ID code", "length code", and "bus level". The sequence of the devices in the frame corresponds to their position in the bus system. Exceptions to this rule are alternative devices, which do not operate on the bus at the same time.

The "logical device number" and "logical group number" fields are the basis for mechanisms used to switch distributed device groups, bus segments, and alternative groups.

Table 1-4 Basic structure of a configuration frame

No.	Logical Device Number		Brief Device Description		Bus Level	Logical Group Number	
	Segment	Position	Length Code	ID Code		Group	Alternative
1							
2							
...							
n							
Bit	15 8	7 0	15 8	7 0	15 0	15 8	7 0

The defined configuration frame must not be exceeded by the connected bus configuration at any time. Also, the position of the devices must always correspond to the entries made in the configuration frame. The only exception is that the active configuration represents a subset of the configuration frame after switching off bus segments or groups.

To facilitate the explanations concerning the functions of the configuration frame, the terms *connected* and *active configuration*, as well as their interconnections will be explained first.

Connected configuration

The connected configuration comprises all devices actually connected to INTERBUS that can be physically reached if all bus segments are enabled. This does not include the alternatives disconnected in the current state. Thus, the connected configuration is a subset of the complete configuration frame.

Active configuration

The active configuration comprises all devices actually connected to INTERBUS that are entered with their data in the summation frame. This means that the active configuration corresponds to the connected configuration minus those bus segments that are disconnected at that time. The active configuration is always a subset of the connected configuration.

The following example illustrates the differences between the connected and active configuration:

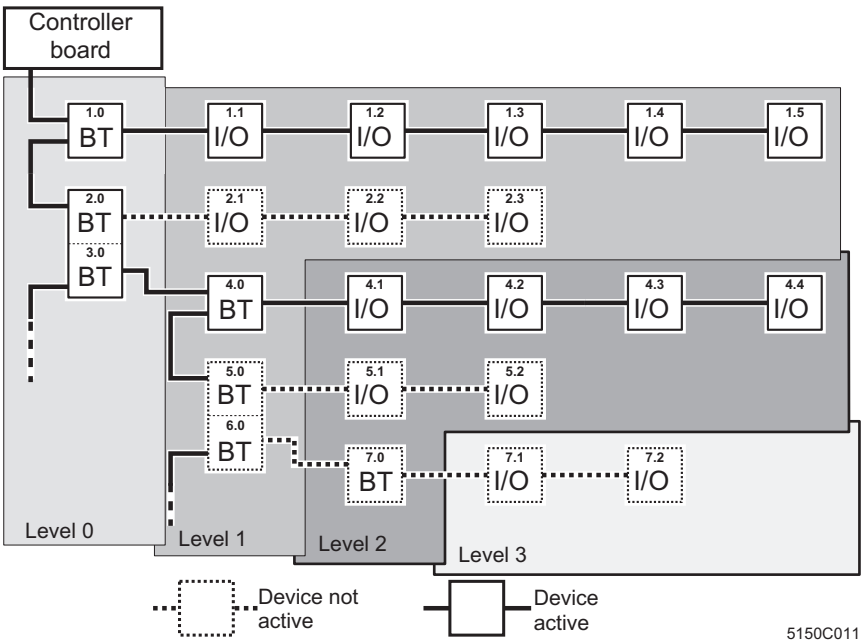


Figure 1-9 Bus configuration (connected and active configuration)

Table 1-5 Configuration frame

Log. Device Number	Brief Device Description	Bus Level	Logic Group No.	Active
1.0	...	0	–	Yes
1.1	...	1	–	Yes
1.2	...	1	–	Yes
1.3	...	1	–	Yes
1.4	...	1	–	Yes
1.5	...	1	–	Yes

Table 1-5 Configuration frame

Log. Device Number	Brief Device Description	Bus Level	Logic Group No.	Active
2.0	...	0	1	No
2.1	...	1	1	No
2.2	...	1	1	No
2.3	...	1	1	No
3.0	...	1	2	Yes
4.0	...	1	2	Yes
4.1	...	2	2	Yes
4.2	...	2	2	Yes
4.3	...	2	2	Yes
4.4	...	2	2	Yes
5.0	...	1	2	No
5.1	...	2	2	No
5.2	...	2	2	No
6.0	...	1	2	No
7.0	...	2	2	No
7.1	...	3	2	No
7.2	...	3	2	No

As well as the actual configuration data of the bus configuration (ID code, length code, device number, bus level, logical group number, etc.) each configuration frame also includes the following lists:

– Process data description list (PDDL)
– Process data reference list (PDRL)
– Communication reference list (CRL)
– Specific operating parameters (e.g., update time, timeout)
– Statistical diagnostic data, etc.

Logical Device Numbers

The logical device number consists of the bus segment (high byte) and the position within this bus segment (low byte).

Bus segment

A bus segment consists of a remote bus device and the incoming remote bus cable. If this remote bus device is a bus terminal module, the devices of the branching local bus (if present) also belong to this bus segment.

The bus terminal module of bus segment 1 is connected to the controller board by the remote bus. Bus segment 2 is connected to the bus terminal module of bus segment 1 by the remote bus.

Bus terminal modules for the installation remote bus and installation remote bus devices have their own bus segment number and are coupled with a lower device level.

The position information in the logical device number corresponds to the identification number of this device within the bus segment. Thus, every device is clearly identified by the bus segment and position number.

Physical device numbering

Physical numbering is the simplest method of numbering devices within the configuration frame. All devices are numbered according to the sequence in which they are physically located in the network.

When opening a new bus segment, all remote bus devices are assigned the position number 0 (x.0). The individual segments are numbered in ascending order. If a remote bus device opens a new level (branch) as well as another remote bus segment of that level, numbering will continue on the level which has just been opened, then on the next higher level.

The segment number remains constant within a bus segment and the other devices are identified by position in ascending numerical order.

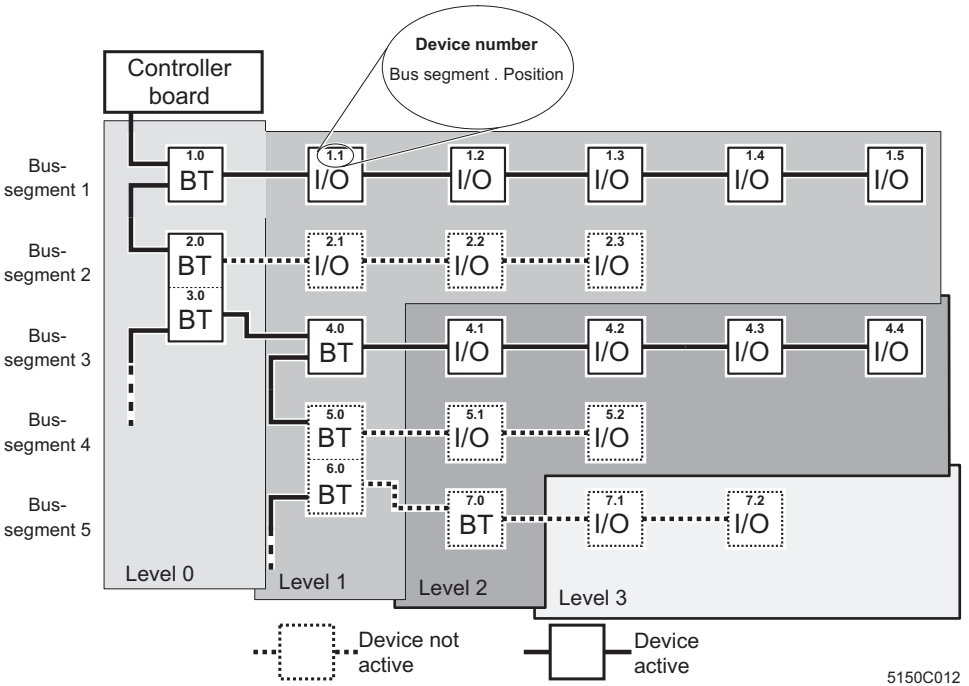


Figure 1-10 Structure of device numbers

Physical device numbering is only used in test mode and after the "Create_Configuration" service (see page 2-62). When devices are added, the device numbers of all lower-level devices are shifted.

Logical device numbering

Logical device numbering can be used to freely set the bus segment number and position number of a device. When a new branch or a new device is added, the numbering of existing devices does not change. Here, the following rules are to be observed:

– Bus terminal modules are always assigned position number "0".
– Local bus devices are always assigned the bus segment number of the higher-level bus terminal module.

Logical device numbering is carried out by downloading a predefined configuration frame using the services

– "Load_Configuration" (see page 2-31)
– "Complete_Load_Configuration" (see page 2-54)

To enable the loading of individual parts of the configuration frame, the firmware offers two different modes (device-oriented and list-oriented transmission). Using parameters each subset of the configuration frame can be accessed.

Device-oriented transmission

With device-oriented transmission, complete device descriptions are loaded to the configuration frame by indicating the physical device number (line number of the configuration frame) at the same time. In addition, the *Used_Attributes* parameter can be used to select individual columns for transmission.

List-oriented transmission

With list-oriented transmission, information from the configuration frame is loaded in lists (e.g., ID list, list of group numbers, etc.). It is sorted according to physical bus position. It is also possible to transmit partial lists by setting the *Start_Entry_No* and *Entry_Count* parameters.

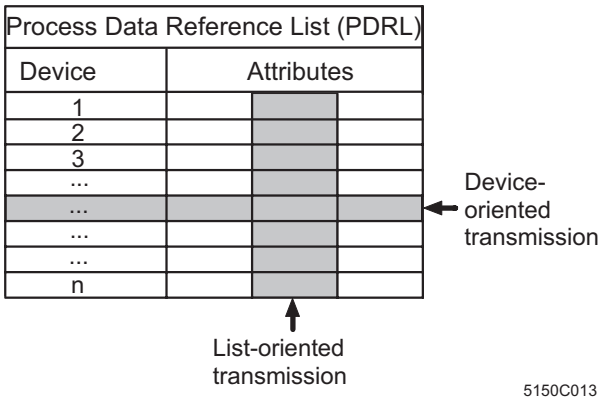


Figure 1-11 Device-oriented and list-oriented transmission

The *Used_Attributes*, *Device_Number* and *Entry_Count* parameters determine the type of loading. The *Used_Attributes* parameter selects the columns from the configuration matrix enabling combinations of any type. The *Device_Number* and *Entry_Count* parameters define how many list entries are loaded in the list and from which list position. It is only possible to load connected parts of the list.

The above parameters enable all loading types to be implemented. Thus, for example, the entire matrix will be loaded if all bits of the *Used_Attributes* parameter are set, the *Device_Number* is zero, and *Entry_Count* corresponds to the number of devices within the configuration frame.

Logical Group Numbers

The logical group number is divided into the group number and alternative information. The default setting does not include a logical group number ($FFFF_{hex}$ value).

Group numbers must only be assigned if bus segments that are physically separated are to be switched with one command or if alternative groups are provided within the bus configuration. This group number is only required for the called devices. All other devices can keep their standard number.

Assignment to groups

The high-order byte contains the group number. It can also be used to assign devices distributed in a system to a group. The low-order byte of the group number indicates if this group can be switched alternatively.

Alternative groups

Groups that can be switched alternatively enable the user to switch bus segments of a different structure to the same remote bus output of a bus terminal module. Alternative devices have the same group number but a different alternative number in the same configuration frame. The devices of different alternatives must always have different logical device numbers, as corresponding process data descriptions (PDDs) only refer to the device number. A logical device number may not be assigned twice.

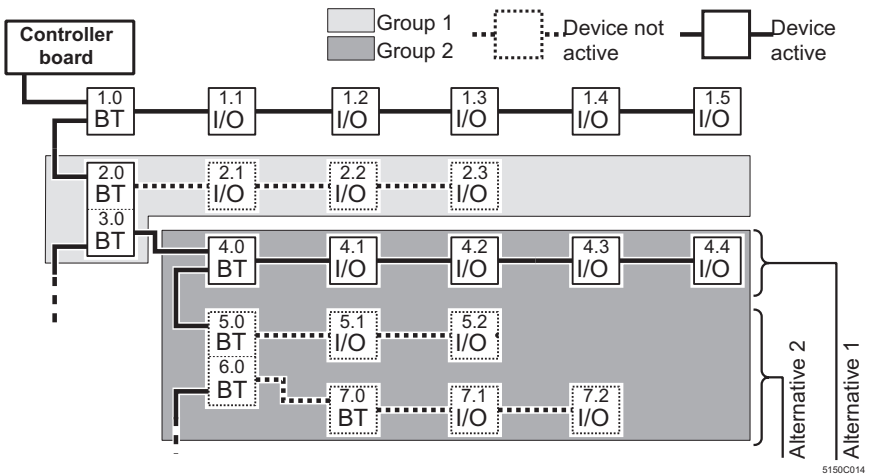


Figure 1-12 Bus configuration (groups and alternative groups)

Table 1-6 Configuration frame

Log. Device Number	Brief Device Description	Bus Level	Logic Group No.	Active
1.0	...	0	–	Yes
1.1	...	1	–	Yes
1.2	...	1	–	Yes
1.3	...	1	–	Yes
1.4	...	1	–	Yes
1.5	...	1	–	Yes
2.0	...	0	1.1	No
2.1	...	1	1.1	No
2.2	...	1	1.1	No
2.3	...	1	1.1	No
3.0	...	0	2.1	No
4.0	...	1	2.1	No
4.1	...	2	2.1	Yes
4.2	...	2	2.1	Yes
4.3	...	2	2.1	Yes
4.4	...	2	2.1	Yes
5.0	...	1	2.2	No
5.1	...	2	2.2	No
5.2	...	2	2.2	No
6.0	...	2	2.2	No
7.0	...	2	2.2	No
7.1	...	3	2.2	No
7.2	...	3	2.2	No

When starting up the system, alternative groups are not automatically linked, but must be added explicitly with the "Control_Active_Configuration" service (see page 2-104).

Structure of the Device Length Code

This section explains how the device length code (length code in the following) is formed and how it can be used to determine the width of the process data channel.

Length codes can be represented in two ways.

1 The length code, which is defined in INTERBUS standard DIN EN 50254 : 1999-07.

This code is only used in Layers 1 and 2 of the OSI reference model. Each code defines a certain length, which can be determined from a table.

2 The length code, which is made available to the user by the firmware.

In the firmware, the length code is converted according to the standard to a form which is clear to the user even without a reference table. This form is documented in the product documentation (e.g., package slips, data sheets) as the length code.

The length code generated by the firmware is represented in one byte, as shown in the following diagram.

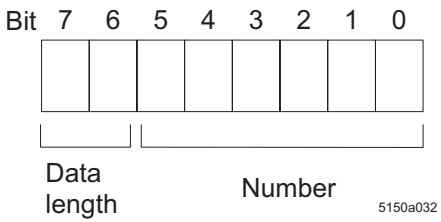


Figure 1-13 Structure of the length code

Bits 7 and 6 specify the data lengths. Where:

Bit 7	Bit 6	Data Length	
0	0	Word	(16 bits)
1	0	Byte	(8 bits)
0	1	Nibble	(4 bits)
1	1	Bit	(1 bit)

Bits 5 to 0 specify the number of data units of this length.

The data length specified is always the maximum possible length (e.g., not 2 bytes, but 1 word).

Examples:

Length code for a process data channel with a length of 2 words:	**Bit**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
		0	**0**	0	0	0	0	1	0
		Word		2					**Length code**
		0				**2**			**02_{hex}**

Length code for a process data channel with a length of 1 byte:	**Bit**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
		1	**0**	0	0	0	0	0	1
		Byte		1					**Length code**
		8				**1**			**81_{hex}**

Length code for a process data channel with a length of 1 nibble:	**Bit**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
		0	**1**	0	0	0	0	0	1
		Nibble		1					**Length code**
		4				**1**			**41_{hex}**

Length code for a process data channel with a length of 2 bits:	**Bit**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
		1	**1**	0	0	0	0	1	0
		Bit		2					**Length code**
		C				**2**			**$C2_{hex}$**

Overview of Length Codes

Assignment of length codes according to DIN EN 50254 to the length codes for the user:

DIN EN 50254		User	
Bit 12 to Bit 8	**Data Width**	**Length Code [hex]**	**Data Width**
0 0 0 0 0	0	00	0
0 1 1 0 0	1 bit	C1	1 bit
0 1 1 0 1	2 bits	C2	2 bits
0 1 0 0 0	4 bits	41	1 nibble
0 1 0 0 1	1 byte	81	1 byte
0 0 0 0 1	2 bytes	01	1 word
0 1 0 1 1	3 bytes	83	3 bytes
0 0 0 1 0	4 bytes	02	2 words
0 0 0 1 1	6 bytes	03	3 words
0 0 1 0 0	8 bytes	04	4 words
0 0 1 0 1	10 bytes	05	5 words
0 1 1 1 0	12 bytes	06	6 words
0 1 1 1 1	14 bytes	07	7 words
0 0 1 1 0	16 bytes	08	8 words
0 0 1 1 1	18 bytes	09	9 words
1 0 1 0 1	20 bytes	0A	10 words
1 0 1 1 0	24 bytes	0C	12 words
1 0 1 1 1	28 bytes	0E	14 words
1 0 0 1 0	32 bytes	10	16 words
1 0 0 1 1	48 bytes	18	24 words
1 0 0 0 1	52 bytes	1A	26 words
1 0 1 0 0	64 bytes	20	32 words
1 0 0 0 0	Reserved		
1 1 x x x			

1.4.6 Process Data

Process data management

In Generation 4 firmware, the process data channel is extended by process data management that can be flexibly adapted to all applications. The following functions can be executed using process data management services:

- Division of the process data channel of a device into several small parts, known as process data descriptions (PDDs), down to the size of one bit.
- Independent addressing of these PDDs in the host address area.
- Broadcast addressing of the PDDs.
- Definition of direct link data.
- Superimposition of PDDs with low priority by PDDs with higher priority.

Process data management includes services for defining and assigning PDDs.

Process data descriptions

The PDDs in the process data description list (PDDL) enable the optimum management of module inputs and outputs tailored to the application. Thus, the process image can be divided down to bit level and addressed separately.

Process data assignment

The process data reference list (PDRL) defines the assignment of the input and output data to the corresponding memory positions in the MPM. It can be used to distribute all inputs and outputs as required in the host address area and to access them there. It is a comprehensive mechanism for grouping data for all devices as is most appropriate on the basis of function or memory.

Process data linkage also offers basic mechanisms required for process data preprocessing on the controller board.

Advantages of User-Defined Process Data

The definition and assignment of process data descriptions (PDDs) offer decisive advantages:

1. Adding or removing a device does not cause existing addresses to be shifted in the MPM and thus in the application program. There will only be a modification in the process data reference list (PDRL).
2. Some devices have an I/O channel, which contains a number of logic functions. For example, the 32-bit output word of an INTERBUS-compatible frequency inverter is divided into control word and setpoint value. It is therefore possible to subdivide this double word into corresponding functional units.

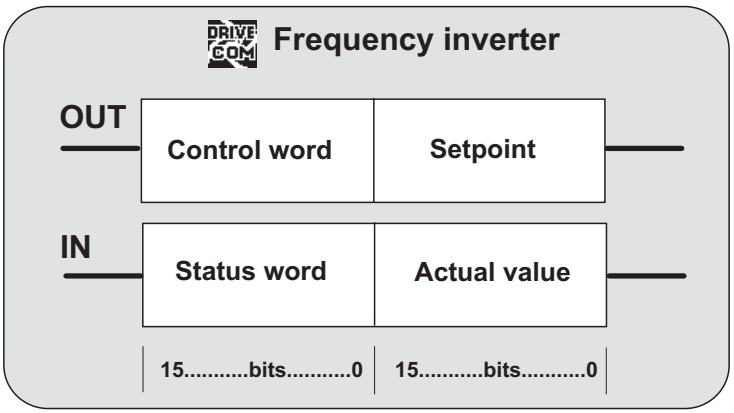


Figure 1-14 Process data channel of a frequency inverter

3. The process data can then be freely assigned in the address area of the control system and thus, for example, be combined in functional units. In addition, it is possible to implement a broadcast function through the definition of the process data reference list. When doing so, a setpoint definition, for example, will occupy only one address in the host system. The INTERBUS controller board writes the setpoint value from the host system to other inverters located in the system. In addition to process data descriptions via a double word, word or byte,

process data descriptions are also possible via a bit. In this case, status bits of different devices can be summarized in one word in the host system. Thus, several device states can be represented quickly and comprehensively by means of a simple word access.

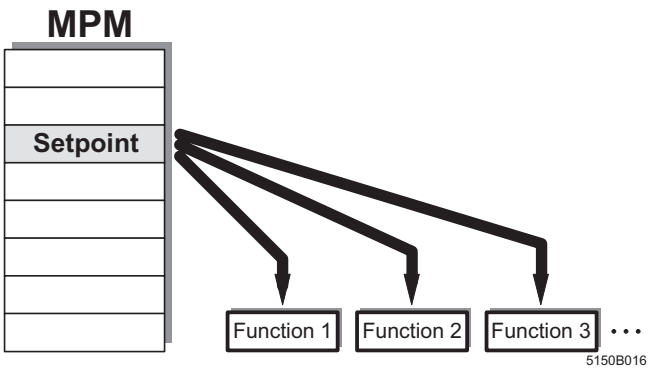


Figure 1-15 Broadcast

4. Process data descriptions of one bit are also required for direct links. A direct link means that output bits are directly set by the controller board. The link depends on the assigned input bits, which means it is independent of the control program. The program processing time of the application program is excluded. The reaction time is reduced to the INTERBUS response time.

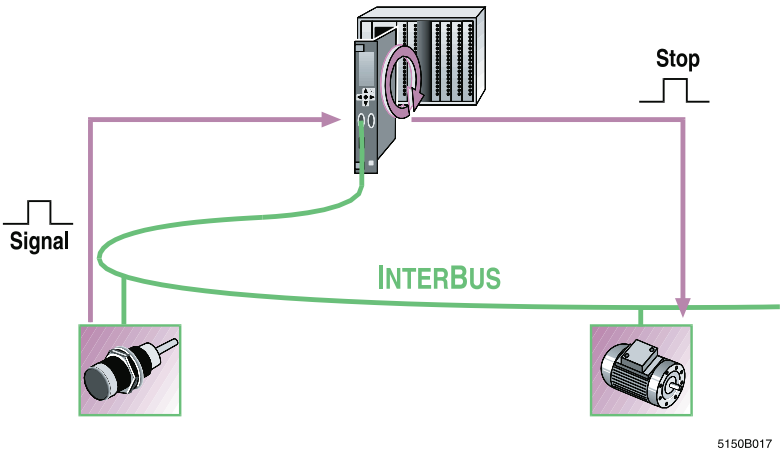


Figure 1-16 Direct link

Automatic Address Assignment of Process Data

In three cases, process data descriptions (PDDs) are automatically generated and process data is assigned to the MPM:

1. In test mode
2. When executing the "Create_Configuration" service
3. After loading the configuration frame with the "Terminate_Load_Configuration" service. The *Default_Parameter* parameter must be set to 0001_{hex} or 0003_{hex}.

Default PDD

By default, the process data channel of every device is written by the firmware using default process data descriptions (PDDs). The I/O data of the devices then constitutes one (IN or OUT) or two (IN and OUT) process data descriptions. These PDDs include the size, data direction, and names of the devices as well as the default index for the IN or OUT data direction.

Default PDRL

The firmware assigns process data for the IN or OUT area within the MPM. According to the physical sequence of the devices within the ring, process data is assigned starting from the base address and without gaps, if possible. When assigning the process data addresses in the MPM, the following rules are to be observed:

– Devices with one or more data words can only occupy even MPM addresses.
– Byte modules can begin on both even and odd byte addresses.
– Nibble modules can begin on the upper or lower nibble of every byte.

1.4.7 State Description During Configuration

Configuration is divided into three phases:

– Phase 1: Loading the configuration frame
– Phase 2: Defining process data descriptions (PDD)
– Phase 3: Assigning process data

The PDDL_LOADING (loading process data description list) and PDRL_LOADING (loading process data reference list) states do not have to be processed. After terminating the CFG_LOADING phase with the "Terminate_Load_Configuration" service, these lists are automatically generated from the information in the loaded configuration frame provided that the corresponding parameters have been set.

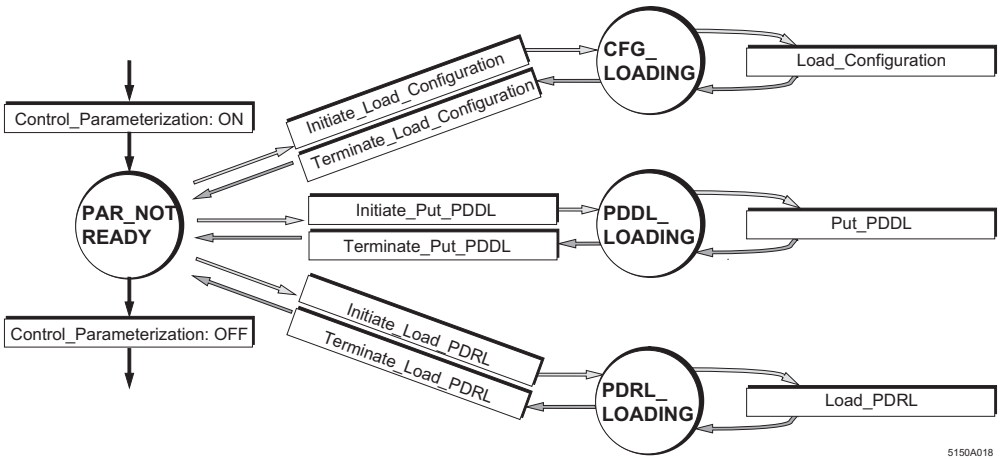


Figure 1-17 Services for controller board configuration

Phase 1: Loading the Configuration Frame

Bus device properties are stored in the configuration frame. These include, for example:

– Device type: digital/analog, number of channels
– ID code
– Position of the device within INTERBUS

When loading the configuration frame, the user determines the selection of modules and the bus topology as well as the definition of functionally related bus parts/bus segments as a group. All loading processes (including those of following lists) are started by an initialization service and stopped by a termination service.

Phase 2: Loading the Process Data Description List

A process data description list is created to manage module inputs and outputs tailored for the application as well as memory-optimized addressing. It is also possible to define descriptions for particularly time-critical bit operations, which are carried out independently by the controller board and without control through the application program, e.g., for positioning tasks.

Process data descriptions (PDDs) must not necessarily be created, as the controller board automatically defines default process data descriptions. For example, a process data description of one word is automatically generated for a digital input module with 16 inputs (1 word). When using a self-defined process data description list, the user can generate two process data descriptions of one byte each from this word. These process data descriptions can freely be assigned in the address area of the host system.

Phase 3: Creating a Process Data Reference List

The process data reference list defines the relationships between the process data. The user can use the list to address all inputs and outputs in any combination as required in the host system address area and respond flexibly to changes and extensions. An extension will not modify existing addresses in the application program. It will simply generate a modification to the process data reference list. This type of addressing, which is known as logical addressing, is not absolutely necessary, although it is recommended due to the increased flexibility it provides.

1.5 Functions for Configuration Frame Management

The configuration frame management supplies services for managing and modifying the configuration of the bus system. e.g.,:

– Loading, reading, and deleting the configuration frame
– Creating a configuration frame according to the current configuration
– Switching between different configuration frames

Two processes are available for storing a configuration frame on the controller board:

– Automatic generation of a configuration frame by the controller board according to the physical bus configuration
– Manual generation of an optimum configuration frame by the user

Configuration frames that are automatically generated or loaded by the user can be modified by overwriting the corresponding entries in the parameterization phase of the controller board. Modifying certain entries (e.g., ID code) of the configuration frame may affect process data and parameter channels. The controller board checks if these effects are so decisive that a previously configured process data description list (PDDL), the communication reference list (CRL) or the process data reference list (PDRL) must be declared invalid.

1.5.1 Automatic Loading of a Configuration

This process should only be carried out for test purposes as it does not offer the same diagnostics as when starting up a pre-defined configuration with a certain object in mind.

After calling the "Create_Configuration" service (0710_{hex}), the controller board automatically generates a configuration frame according to the bus configuration connected at that time. This configuration frame is stored under the frame reference indicated when calling the service in the configuration directory. If there is already a configuration frame under this frame reference, it will be overwritten.

In this process, all descriptions (PDR, CRL, etc.) and operating parameters assigned to the configuration frame are automatically generated or set. The configuration generated can be read with the "Read_Configuration" (0309_{hex}) service and compared with the planned configuration.

1.5.2 Manual Generation of an Optimum Configuration

The user may also load a new configuration frame. A frame reference of any kind can be assigned to the configuration frame. If there is already a configuration frame under this frame reference, it will be overwritten.

A configuration must be loaded step by step, as the number of parameters for a service is limited:

– Initiate the load process with the "Initiate_Load_Configuration" service (0306_{hex}).
– Transmit the configuration data to the controller board by calling the "Load_Configuration" service (0307_{hex}) several times.
– Terminate the load process with the "Terminate_Load_Configuration" service (0308_{hex}). In addition, this service checks the loaded data in its totality. If there are no errors, the data will be accepted. If the data is incorrect, the service will be followed by a negative confirmation. Also, when terminating the load process with the *Default_Parameter* parameter, it is possible to indicate whether the process data and/or parameter channel should be automatically generated according to the loaded configuration frame.

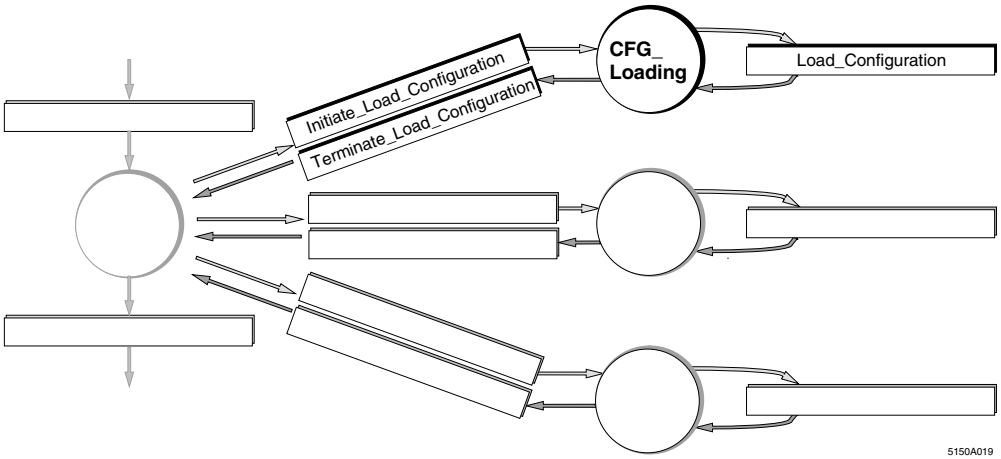


Figure 1-18 Loading a configuration frame

There are different ways of transmitting configuration data to the controller board: device-oriented, list-oriented, or a combination of both.

To support the operation of PLC function blocks and to minimize the number of parameters in the load service, the firmware also provides the "Complete_Load_Configuration" service ($030A_{hex}$). This service always causes one column (always complete) of the configuration frame to be loaded, which can be selected with the *Used_Attributes* parameter. An equivalent service is "Complete_Read_Configuration" ($030B_{hex}$).

1.6 Functions for Process Data Management

The functions for process data management are divided into two areas: process data definition (process data description list, PDDL) and process data assignment (process data reference list, PDRL).

1.6.1 Defining Process Data Objects of a Device

To define the process data objects of a device, proceed as follows:

– Initiate the load process with the "Initiate_Put_Process_Data_Description_List" service (0320_{hex}, see page 2-68).
– Transmit the data to the controller board by calling the "Put_Process_Data_Description_List" service (0321_{hex}, see page 2-70) several times.
– Terminate the load process with the "Terminate_Put_Process_Data_Description_List" service (0322_{hex}, see page 2-74).

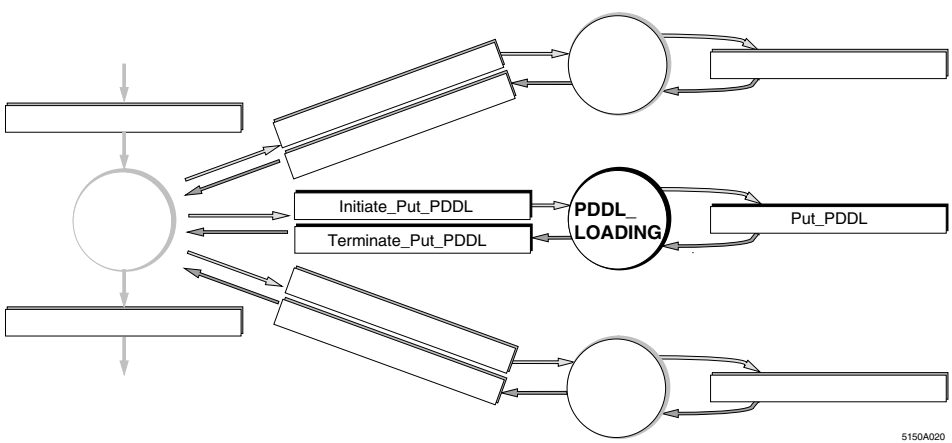


Figure 1-19 Loading the process data description list

When defining process data descriptions (PDDs) for a process data device (PD device), the following rules are to be observed:

– The object type (IN/OUT) of PDDs must correspond to the data direction of the PD device.

- PDDs can overlap within the internal address area of a PD device.
- IN and OUT PDDs must not exceed the internal address area of the respective PD device.
- The length of the PDD must correspond to the data type unless it is a data type of variable length.
- Bit strings must not exceed byte limits with their bit address and length: (bit address + length ≥ 8).

1.6.2 Defining Process Data Description Lists

Process data is device-oriented. Several process data items can be defined for every INTERBUS device. These are entered in the process data description list of the device. The "Put_Process_Data_Description_List" service (0321_{hex}) is used for this purpose. This service defines all the process data descriptions for a device. The service structure is shown in the following figure.

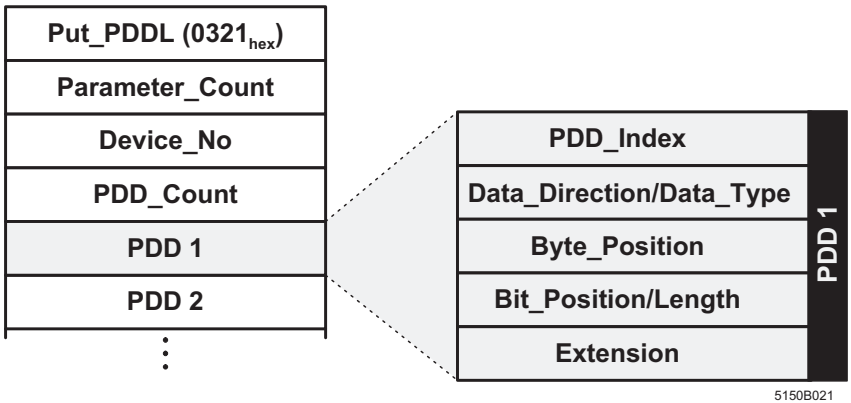


Figure 1-20 Service structure of the process data description list

The meanings of the individual parameters are as follows:

PDD_Index:	Indicates the process data description (PDD). Values in the range from 1000_{hex} to $BFFF_{hex}$ are permitted except for values 6010_{hex} (default PDD index for inputs) and 6011_{hex} (default PDD index for outputs).
Data_Direction:	Data direction, enter $0C_{hex}$ for inputs and $0D_{hex}$ for outputs
Data_Type:	Data type; enter $0A_{hex}$ for byte string process data and $0F_{hex}$ for bit string process data.
Byte_Position:	Byte offset
Bit_Position:	Bit offset
Length:	String length (a bit string must not exceed a byte limit)
Extension:	Extension, 0000_{hex}

The PDD index indicates whether it is a user-defined process data item or a default process data item. A default PDD for input data always has the index 6010_{hex} and a default PDD for output data the index 6011_{hex}. The process data type is defined within the type definition. Process data can be of the byte string or bit string type. For example, a word process data item consists of two bytes of process data, the parameter length is 02_{hex}. The *Byte_Position* and *Bit_Position* parameters indicate the byte and bit offset respectively. Within the extension a comment can be assigned as ASCII text.

The following figure shows an example of byte IN and OUT process data, as well as a bit process output data item with a length of 2 bits.

Figure 1-21 Byte IN and OUT process data

1.6.3 Configuring the Process Data Assignment

When configuring the process data assignment, proceed as follows:

– Initiate the load process with the "Initiate_Load_Process_Data_Reference_List" service (0324_{hex}, see page 2-80).
– Transmit the list to the controller board by calling the "Load_Process_Data_Reference_List" service (0325_{hex}, see page 2-82) several times.
– Terminate the load process with the "Terminate_Load_Process_Data_Reference_List" service (0326_{hex}, see page 2-86).

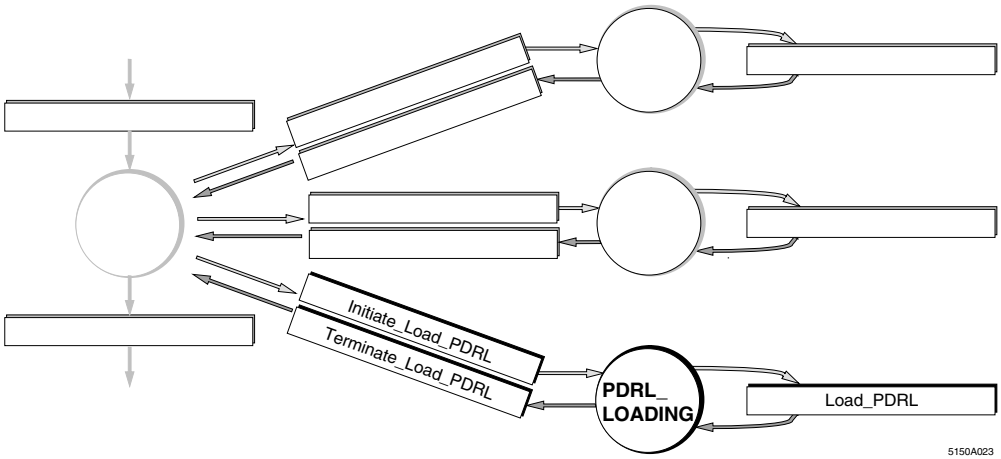


Figure 1-22 Loading the process data reference list

There are 3 options for linking process data:

1. Mapping the input process data item to an input address of the host system.
2. Mapping the output process data item to an output address of the host system.
3. Direct mapping of an input process data item to an output process data item (direct link).

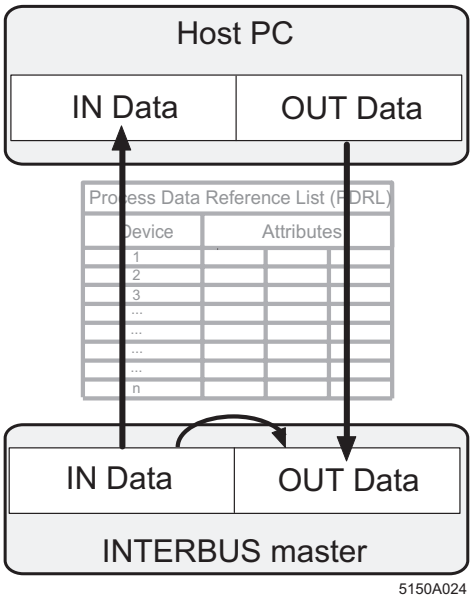


Figure 1-23 Process data assignment

Defining PDRL entries

When defining PDRL entries, the following rules are to be observed:

– In general, only entries that correspond to one of the combinations mentioned above are accepted.

– IN process data descriptions or addresses can be entered repeatedly as a source in the PDRL.

– Identical or overlapping OUT process data descriptions or addresses cannot be repeatedly indicated as a destination in the PDRL, unless one of the process data descriptions is a process data bit string. A process data bit string will overwrite the other object. This special case makes it possible to map individual bits or bit strings to subordinated objects.

Each assignment occupies six words within the "Load_Process_Data_Reference_List" service (0325_{hex}). The structure of this entry within the service is shown below:

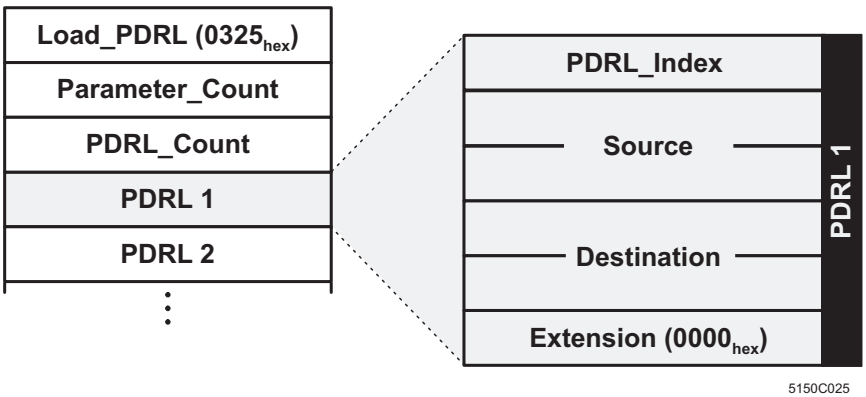


Figure 1-24 Structure of the "Load_Process_Data_Reference_List" service

Table 1-7 Parameters of the process data reference list

Parameters	Input Data	Output Data	Direct Link Data	—
1 **PDRL_Index**	0001 ... 3FFF	4000 ... 7FFF	8000 ... BFFF	C000 ... FFFF
2 **Source**	Device no.	Address	Device no.	—
3 **Source**	Index	Bit position; consistency	Index	
4 **Destination**	Address	Device no.	Device no.	—
5 **Destination**	Bit position; consistency	Index	Index	
6 **Extension**	Reserved (0000_{hex})			

1.7 Functions for Error and Diagnostic Management

Error management

Error management is responsible for error detection, indication and treatment, e.g.,:

- Indication of configuration errors occurring during operation
- Indication of system errors (hardware and firmware)
- Possibility of indicating individual bus errors
- Supply of services for
  - Reading errors
  - Confirming device status messages
  - Reading device states

Errors can be classified as follows:

User errors

User errors are caused by the user when sending a service (e.g., illegal parameters, state conflict, etc.). These errors cause a negative confirmation of the service but do not modify the bus state (e.g., data cycle stop).

System errors

System errors are hardware or firmware errors that can modify the bus state. System errors also include those errors indicated by the operating system. If a serious error has occurred, it generates - in addition to a bus reset - an error message and a state of the controller board, which does not enable service processing except for the "Reset_Controller_Board" service (0956_{hex}, see page 2-110).

Device status messages

Device status messages are events, which occur on INTERBUS devices and are indicated using the "Device_Fail_Indication" message (5340_{hex}, see page 2-206). These status messages do not change the bus state and can be confirmed by the user with the "Control_Device_Function" service (0714_{hex}, see page 2-107).

The "Read_Device_State" service (0315_{hex}, see page 2-116) can be used to read selectable status information of all active devices for diagnostic purposes.

Individual bus errors

Individual bus errors are detected by the data link layer and processed in bus error statistics. These individual bus errors cause an error analysis (ID cycle). However, they do not change the bus state as long as they are individual errors.

Bus error

Bus errors are caused, for example, by a configuration change, broken cables or strong interference. These errors cause the data cycles to be stopped (provided that data cycles were running previously) and an error message to be indicated by the "Bus_Error_Indication" message (6342_{hex}). The detailed cause of a bus error will not be indicated automatically but must be requested by the user with the "Get_Error_Info" service (0316_{hex}). This service reads the error cause from an error buffer. An error buffer can occupy one of the following three states:

Empty	No errors have occurred
Not yet analyzed	The error analysis is not yet complete
Analyzed	Up to ten error codes can be read including the error location

Error removal or appropriate switching services (e.g., switching off the defective segment) can be used to eliminate the error cause in the "Analyzed" state. New data cycles can then be initiated with the "Start_Data_Transfer" service (0701_{hex}).

Diagnostics

Diagnostics is a subcomponent of error management providing diagnostic and statistical services.

The statistic diagnostics for monitoring the bus quality runs automatically in the background. The diagnostic data always refers to the configuration frame activated at that time. The user can read this data using the "Get_Diag_Info" service ($032B_{hex}$).

In addition, the user can reset the statistical diagnostics using the "Control_Statistics" service ($030F_{hex}$).

1.8 Bus Control Functions

Bus control comprises services, which directly influence the system status during operation. They include the activation or deactivation of different configuration frames and the starting or stopping of data transport as well as the reset/alarm stop for the controller board. Configuration services with state changes are excluded from this definition.

Another area belonging to bus control includes switch functions for groups and alternatives.

1.8.1 Services for System State Control

Different services can be used to switch between the PAR_READY, ACTIVE, and RUN states during bus operation.

In the PAR_READY state, there is at least one complete configuration frame. A configuration frame is activated with the "Activate_Configuration" service (0711_{hex}). This service checks if the connected configuration corresponds to the configuration defined in the configuration frame. If there are no deviations, INTERBUS runs ID cycles. This configuration frame can be disabled with "Deactivate_Configuration" (0712_{hex}).

To start data transfer on the bus system, the "Start_Data_Transfer" service (0701_{hex}) must be executed from the ACTIVE state. Cyclic data traffic is disabled with the "Stop_Data_Transfer" service (0702_{hex}).

If errors occur when activating or starting the bus system, the state machine changes to the BUS_FAIL state. In this state, the required services can be called again after the error cause has been eliminated.

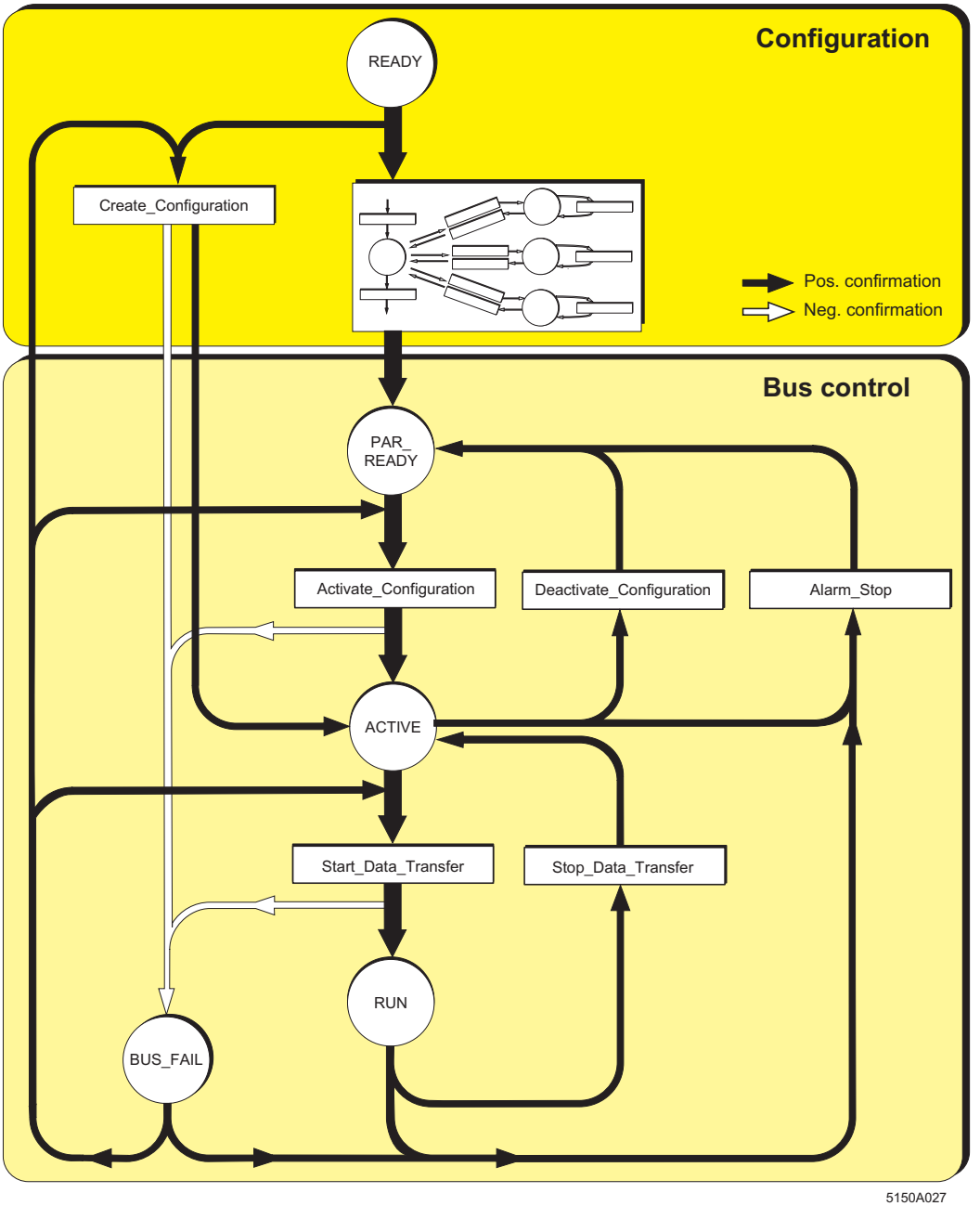


Figure 1-25 Services for system state control

1.8.2 Services for Switching Groups and Alternative Groups

The "Control_Active_Configuration" service (0713_{hex}) can be used to switch individual devices, bus segments, logical groups, and/or alternative device groups on or off during bus operation. A list of the affected devices is transferred as a parameter. The switch process affects the devices indicated in the list as well as all devices depending on them, i.e., all INTERBUS devices, which

– Belong to the same bus segment
– Belong to the same logical group
– Follow one of the INTERBUS devices indicated in the list within the physical ring

1.9 General Firmware Functions

The general firmware functions include services for working with the controller board parameterization memory.

1.9.1 Working With the Parameterization Memory

You can use this service to format the parameterization memory, transmit service sequences and parameter records stored in the controller board main memory to the parameterization memory or to open, close, and delete individual files on the parameterization memory. You can search for a specific file and write data to this file or read data from it.

Please note the following firmware properties on controller boards with plug-in parameterization memories:

New Parameterization Memory
If a new parameterization memory is used when booting a controller board, the parameterization memory will be formatted. If it is available, the message 3030_{hex} appears on the controller board display. Remember that depending on the size of the parameterization memory, the formatting process may take up to 12 minutes.

Large Parameterization Memory
It may take up to 10 minutes for a controller to boot if a full size parameterization memory (e.g., 64 MB) is used during this process. If it is available, the display will count through a value range between 3100 and 3200 during the boot process. This counting process will be repeated until the controller boot process has been completed.

Storing the IP Address
- On 1 MB memory cards:
1 MB parameterization memories cannot store an IP address. The controller saves the IP address internally when this card is used.

- On > 1 MB memory cards:
Memory cards > 1 MB can store an IP address.

Please refer to the module user manuals for more information.

1.9.2 Firmware Response To Specific Controller Board Commands

1.9.2.1 Debug or Auto Debug on the Controller Board

If a remote bus device has a SUPI 1 chip, the controller board debug and auto debug functions cannot be used.

Section 2

This section provides information about

– Firmware service tasks and calls
– Parameters for these services

2 Firmware Services

2.1 Overview

Table 2-1 Overview of services (according to command codes)

Code	Service	Page
0140_{hex}	Initiate_Load_Action_Object	2-133
0141_{hex}	Load_Action_Object	2-135
0142_{hex}	Terminate_Load_Action_Object	2-137
0143_{hex}	Read_Action_Object	2-139
0144_{hex}	Delete_Action_Object	2-142
0145_{hex}	Initiate_Load_Signal_Object	2-144
0146_{hex}	Load_Signal_Object	2-146
0147_{hex}	Terminate_Load_Signal_Object	2-152
0148_{hex}	Read_Signal_Object	2-154
0149_{hex}	Delete_Signal_Object	2-158
$014A_{hex}$	Initiate_Load_Event_Object	2-160
$014B_{hex}$	Load_Event_Object	2-162
$014C_{hex}$	Terminate_Load_Event_Object	2-167
$014D_{hex}$	Read_Event_Object	2-169
$014E_{hex}$	Delete_Event_Object	2-172
$014F_{hex}$	Change_Exclusive_Rights	2-13
0152_{hex}	Set_Indication	2-15
0158_{hex}	Program_Resident_Actions	2-174
0159_{hex}	Clear_Parameterization_Memory	2-176
$015B_{hex}$	File_Open	2-179
$015C_{hex}$	File_Close	2-182

Table 2-1 Overview of services (according to command codes)

Code	Service	Page
$015D_{hex}$	File_Remove	2-186
$015E_{hex}$	File_Write	2-189
$015F_{hex}$	File_Seek	2-192
0160_{hex}	File_Read	2-195
0165_{hex}	File_Remove_II	2-198
0166_{hex}	Get_Card_Information	2-200
0306_{hex}	Initiate_Load_Configuration	2-28
0307_{hex}	Load_Configuration	2-31
0308_{hex}	Terminate_Load_Configuration	2-36
0309_{hex}	Read_Configuration	2-38
$030A_{hex}$	Complete_Load_Configuration	2-54
$030B_{hex}$	Complete_Read_Configuration	2-57
$030C_{hex}$	Delete_Configuration	2-60
$030E_{hex}$	Control_Parameterization	2-11
$030F_{hex}$	Control_Statistics	2-131
0314_{hex}	Control_Device_Function_Not_Exclusive	2-107
0315_{hex}	Read_Device_State	2-116
0316_{hex}	Get_Error_Info	2-113
0317_{hex}	Compare_Configuration	2-51
0320_{hex}	Initiate_Put_Process_Data_Description_List	2-68
0321_{hex}	Put_Process_Data_Description_List	2-70
0322_{hex}	Terminate_Put_Process_Data_Description_List	2-74
0323_{hex}	Get_Process_Data_Description_List	2-76
0324_{hex}	Initiate_Load_Process_Data_Reference_List	2-80
0325_{hex}	Load_Process_Data_Reference_List	2-82

Table 2-1 Overview of services (according to command codes)

Code	Service	Page
0326_{hex}	Terminate_Load_Process_Data_Reference_List	2-86
0327_{hex}	Read_Process_Data_Reference_List	2-88
0328_{hex}	Compact_Load_Process_Data_Reference_List	2-91
0329_{hex}	Compact_Read_Process_Data_Reference_List	2-95
$032A_{hex}$	Get_Version_Info	2-120
$032B_{hex}$	Get_Diag_Info	2-124
0351_{hex}	Read_Value	2-25
0701_{hex}	Start_Data_Transfer	2-98
0702_{hex}	Stop_Data_Transfer	2-102
0710_{hex}	Create_Configuration	2-62
0711_{hex}	Activate_Configuration	2-64
0712_{hex}	Deactivate_Configuration	2-66
0713_{hex}	Control_Active_Configuration	2-104
0714_{hex}	Control_Device_Function	2-107
0750_{hex}	Set_Value	2-18
0760_{hex}	Confirm_Diagnostics	2-111
0956_{hex}	Reset_Controller_Board	2-110
1303_{hex}	Alarm_Stop	2-100

Table 2-2 Automatic indications

Code	Service	Page
4341_{hex}	Fault	2-204
5340_{hex}	Device_State	2-206
6342_{hex}	Bus_Error	2-207
$4B58_{hex}$	Lower_API_Fault	2-205

2.2 Notes on Service Descriptions

Use of services

The use of a service involves sending a service request and evaluating the service confirmation.

The codes of a service request and the subsequent service confirmation only differ in binary notation in bit 15. Bit 15 of a service confirmation is always set.
Thus, in hexadecimal notation, the code of a service confirmation is always 8000_{hex} higher than the code of the service request which it follows.

Example "Start_Data_Transfer"

Request:

"Start_Data_Transfer_Request" 0701_{hex}

Confirmation:

"Start_Data_Transfer_Confirmation" $8701_{hex} = 0701_{hex} + 8000_{hex}$

– *Result* parameter = 0000_{hex} ⇒ Service executed successfully

–*Result* parameter ≠ 0000_{hex} ⇒ Error during service execution

The service confirmation indicates the successful execution of a service via a positive message and provides data, if requested. The service confirmation indicates an error that occurred during service execution via a negative message.

The *Result* parameter of the service confirmation shows if the service was executed successfully *Result* parameter = 0000_{hex}, or if an error occurred (*Result* parameter ≠ 0000_{hex} describes the error cause).

Structure of a service description

A service request/confirmation consists of a block of data words. The parameters that are contained in this block are given in hexadecimal ($_{hex}$) or binary ($_{bin}$) notation.

The structure of all service descriptions is as follows:

2.x.x "Name_of_the_Service" Service

Task: Describes the functions of the service.

Prerequisite: All conditions, which must be met before a service is called to enable successful processing.

Syntax: **Name_of_the_Service_Request** **$Code_{hex}$**

Word 1	Code
Word 2	Parameter_Count
Word 3	Parameters
Word 4	Parameters
Word 5	Parameters
...	...
	Parameters

Bit	15 0

Key:

Code: $0xxx_{hex}$ Command code of the service request (hexadecimal notation)

Parameter_Count: Number of subsequent words

0000_{hex} If the service request does not have parameters.

$xxxx_{hex}$ Otherwise, length of the parameter data record (number of parameter words).

Parameter: Parameters are described individually. Parameters that are organized byte by byte are separated by a vertical line. If a parameter extends over several data words, this is indicated by a line with three dots.

Parameter blocks: Parameter blocks are marked in bold outline. The individual parameters are described in the following section.

Syntax: **Name_of_the_Service_Confirmation** **$Code_{hex}$**

Positive message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code: $8xxx_{hex}$ Message code of the service confirmation

Parameter_Count: Number of subsequent words

with a positive message:

$xxxx_{hex}$ Number of parameter words that are transferred with a positive message

with a negative message:

$xxxx_{hex}$ Number of parameter words that are transferred with a negative message

Result: Result of the service processing

0000_{hex} Indicates a positive message. The controller board executed the service successfully.

$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.

Add_Error_Info: Additional information on the error cause

2.3 Services for Parameterizing the Controller Board

2.3.1 "Control_Parameterization" Service

Task:

This service initiates or terminates the parameterization phase. This is necessary in order to ensure a defined startup behavior for INTERBUS. During the parameterization phase, for example, the validity of read objects is not ensured. Once the parameterization phase has been terminated, the *MPM_Node_Parameterization_Ready* bit is set in the MPM. When this bit is set, the PLC can recognize during startup when the parameterization sequence that is stored on the parameterization memory has been successfully processed.

Syntax:

Control_Parameterization_Request **$030E_{hex}$**

Word 1	Code
Word 2	Parameter_Count
Word 3	Control_Code

Bit	15 0

Key:

Code: $030E_{hex}$ Command code of the service request

Parameter_Count: Number of subsequent words

0001_{hex} 1 parameter word

Control_Code: Function of the service

0001_{hex} Initiate the parameterization phase

0000_{hex} Terminate the parameterization phase

Syntax: **Control_Parameterization_Confirmation** **$830E_{hex}$**

Positive message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code: $830E_{hex}$ Message code of the service confirmation

Parameter_Count: Number of subsequent words
with a positive message:
0001_{hex} 1 parameter word
with a negative message:
0002_{hex} 2 parameter words

Result: Result of the service processing
0000_{hex} Indicates a positive message. The controller board executed the service successfully.
$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.

Add_Error_Info: Additional information on the error cause

2.3.2 "Change_Exclusive_Rights" Service

Task:

An application can use the "Change_Exclusive_Rights" service to execute, request or pass on the right to execute exclusive services.

Prerequisite:

The requesting application can receive the exclusive right only if no other application has this right. Only one application may be the master at a time.

Exclusive services can only be called by one authorized application. In binary notation of the service code (command code or message code) of an exclusive service, bit 10 is set.

Syntax: **Change_Exclusive_Rights_Request** **$014F_{hex}$**

Word 1	Code
Word 2	Parameter_Count
Word 3	Function

Bit	15 0

Key:

Code: $014F_{hex}$ Command code of the service request

Parameter_Count: Number of subsequent words

0001_{hex} 1 parameter word

Function: Service function

0000_{hex} Pass on the exclusive right

0001_{hex} Request the exclusive right

Syntax: **Change_Exclusive_Rights_Confirmation** **$814F_{hex}$**

Positive message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code: $814F_{hex}$ Message code of the service confirmation

Parameter_Count: Number of subsequent words

with a positive message:

0001_{hex} 1 parameter word

with a negative message:

0002_{hex} 2 parameter words

Result: Result of the service processing

0000_{hex} Indicates a positive message. The controller board executed the service successfully.

$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.

Add_Error_Info: Additional information on the error cause

2.3.3 "Set_Indication" Service

Task: This service is used to enable or disable selected indications for particular interfaces.

Syntax: **Set_Indication_Request** **0152_{hex}**

Word 1	Code
Word 2	Parameter_Count
Word 3	Indication_Ability
Word 4	Distribution
Word 5	Indication_Code
Bit	15 0

Key:

Code: 0152_{hex} Command code of the service request

Parameter_Count: Number of subsequent words
0003_{hex} 3 parameter words

Indication_Ability: Enables or disables the indication for the specified interfaces (*Distribution* parameter):
0001_{hex} Enable interfaces.
0000_{hex} Disable interfaces.

Distribution: Interfaces for which the setting should be changed.
The parameter is a 16-bit field in which every bit corresponds to an interface. Set the corresponding bit to 1 on the interfaces whose settings you would like to change. Settings for the *Distribution* parameter:

- Bit 0 Diagnostic display on the front plate (if present). The diagnostic display shows all indications by default.
- Bit 1 Standard signal interface (SSGI)
- Bit 2 Mailbox interface (MXI) to the application program
- Bit 3 Diagnostic interface on the front plate of the controller board
- Bit 4 Mailbox interface (MXI) to MPM node 0

Bit 5	Mailbox interface (MXI) to MPM node 2
Bit 6	Mailbox interface (MXI) to MPM node 3
Bit 7…15	Reserved (always 0_{bin})

Remember that bit 2 must only be set if bits 4, 5, and 6 have not been set.

Indication_Code: The code of the indication you want to enable or disable,
e.g., 4341_{hex} for "Fault_Indication".

Syntax: **Set_Indication_Confirmation** **8152_{hex}**

Positive message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code: 8152_{hex} Message code of the service confirmation

Parameter_Count: Number of subsequent words
with a positive message:
0001_{hex} 1 parameter word
with a negative message:
0002_{hex} 2 parameter words

Result: Result of the service processing
0000_{hex} Indicates a positive message. The controller board executed the service successfully.
$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.

Add_Error_Info: Additional information on the error cause

2.3.4 "Set_Value" Service

Task: This service assigns new values to INTERBUS system parameters (variables). A new value is only accepted if no error was detected when the value range was checked.

The following system parameters are defined:

Table 2-3 System parameters

Variable ID	System Parameter	Value/Comment
0101_{hex}	State of the system management	Read only
0102_{hex}	Address of the diagnostic status register	The permissible address area depends on the host system used.
0103_{hex}	Address of the diagnostic parameter register	
0104_{hex}	Content/value of the diagnostic status register	Read only
0105_{hex}	Content/value of the diagnostic parameter register	Read only
0108_{hex}	Control of the preprocessing task	0 = Deactivate 1 = Enable 2 = Immediate reset of preprocessing outputs 3 = Reset of preprocessing outputs on PLC SYSFAIL
0109_{hex}	Address of the communication register (for Siemens S5 only)	Read only
$010C_{hex}$	Address of the extended diagnostic parameter register	The permissible address area depends on the host system used.
$010D_{hex}$	Content/value of the extended diagnostic parameter register	Read only
0110_{hex}	Slave_ID_Code (for slave boards and system couplers only)	Device code and process data length of the slave

Table 2-3 System parameters

Variable ID	System Parameter	Value/Comment
0117_{hex}	Address of the slave diagnostic status register	The permissible address area depends on the host system used.
0118_{hex}	Address of the slave diagnostic parameter register	
0119_{hex}	Content/value of the slave diagnostic status register	Read only
$011A_{hex}$	Content/value of the slave diagnostic parameter register	Read only
$011D_{hex}$	Slave configuration data (for slave boards and system couplers only)	Slave data (baud rate, etc.)
$020A_{hex}$	Enabling of optical diagnostics once the interfaces have been switched	Read only Value 1: Device at the start of the optical path Value 2: Device at the end of the optical path
0306_{hex}	Parameterization of the host interface (especially for S5/S7)	Parameterize the interrupt cable for the PLC here.
0407_{hex}	Base addresses for default addressing: 1st word digital IN base address 2nd word digital OUT base address 3rd word analog IN base address 4th word analog OUT base address	If the digital IN (or OUT) base address is equal to the analog IN (or OUT) base address, the digital and analog devices in the address area will be mixed.

Table 2-3 System parameters

Variable ID	System Parameter	Value/Comment
1101_{hex}	Address of the standard function start register	The permissible address area depends on the host system used.
1102_{hex}	Address of the standard function status register	
1103_{hex} 1200_{hex}	Address of the standard function parameter register (long word) 1103_{hex} Enter only one word address. The controller board automatically assigns the next higher word address for the second word. 1200_{hex} Enter the required address for each of the two words.	
1210_{hex}	Confirmation delay in ms (32-bit value) After setting the "Set_Value" service, a confirmation from the control system is sent back to the controller board delayed by the time span set in this parameter.	During PLC startup, a startup delay can also be implemented using this parameter. Value specified in ms
1711_{hex}	System parameter for reading the file system of the parameterization memory	
1712_{hex}		
1713_{hex}		
2200_{hex}	Operating mode	0000 0000_{hex}: Asynchronous 0000 0001_{hex}: Bussynchronous 0000 0002_{hex}: Programsynchronous 0600 0000_{hex}: Asynchronous with synchronization pulse
2204_{hex}	The response of the input data in the event of a bus error can be controlled with this parameter.	0 = Input data in the MPM is frozen (default setting) 1 = Input data in the MPM is set to zero

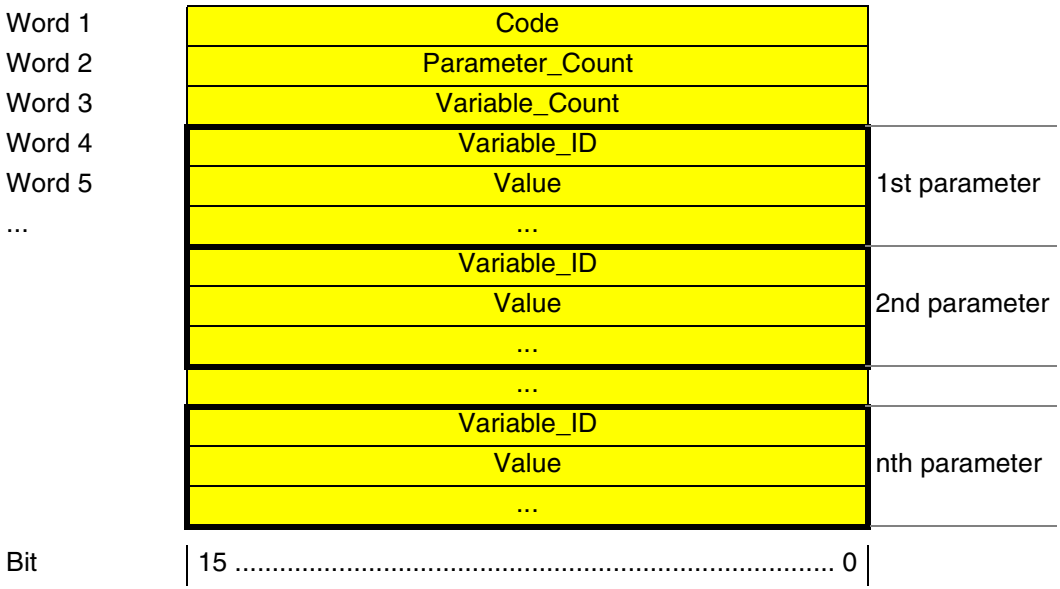
Table 2-3 System parameters

Variable ID	System Parameter	Value/Comment
2210_{hex}	Presetting of the PD cycle time in µs (32-bit value) The PD cycle time is the real time between the start of two successive process data transmission cycles (see 2216_{hex}).	Permissible value range: $0000\ 0000_{hex}$ … $0001\ FBD0_{hex}$ (corresponding to approximately 130 ms, maximum)
2211_{hex}	Bus timeout in µs (32-bit value) Default setting: 200 ms If no error-free data cycle is possible within this time, the controller board terminates the process data transmission and resets the bus.	Permissible value range: $0000\ 0000_{hex}$: $FFFF\ FFFF_{hex}$ The maximum bus timeout/bus warning time is approximately 71 minutes
2212_{hex}	Bus warning time in µs (32-bit value) Default setting: 0 (not activated) If no error-free process data transmission is possible within this time, the controller board sets the *warning bit* in the diagnostic status register.	
2215_{hex}	Activates or deactivates the error code $0BD2_{hex}$, which indicates that no data cycle could be transmitted within the bus warning time defined with the variable 2212_{hex}.	$0000\ 0001_{hex}$: Error code activated $0000\ 0000_{hex}$: Error code deactivated
2216_{hex}	Actual PD cycle time in µs (32-bit value) If the specified PD cycle time (2210_{hex}) cannot be maintained, the actual time between the start of two successive process data transmission cycles is shown here.	
$221B_{hex}$	The response of INTERBUS outputs when the PLC is stopped (SYSFAIL) can be controlled with this parameter.	0 = INTERBUS data cycles are run with outputs set to zero (default setting). Outputs of analog modules are also set to zero. 1 = If the controller board generates an alarm stop, INTERBUS is reset.

Table 2-3 System parameters

Variable ID	System Parameter	Value/Comment
2252_{hex}	Displays which SUPI types are contained in the active configuration (32-bit value).	0000 0000_{hex}: Only devices with SUPI older than SUPI3 0000 0100_{hex}: SUPI3 devices only 0000 0200_{hex}: SUPI3 devices and older devices
2254_{hex}	This value indicates the number of bus errors that must occur within 1,000,000 INTERBUS cycles in order for the quality bit to be set in the diagnostic status register.	Default value: n = 20 Please note: The quality bit is recalculated after 100,000 cycles.
$A255_{hex}$	Enable single channel diagnostics	0000 0404_{hex}: Disable all error messages 0000 $047C_{hex}$: Enable all error messages 0000 $046C_{hex}$: Only disable error messages for the voltage supply U_{S2}

Syntax: **Set_Value_Request** **0750_{hex}**



Key:

Code:	0750_{hex} Command code of the service request
Parameter_Count:	Number of subsequent words The value depends on the number and length of the system parameters. The length of the system parameter (16 words, maximum) is specified in the third hexadecimal position (bits 8 ... 11) of the *Variable_ID* parameter.
Variable_Count:	The number of system parameters to which new values are to be assigned
Variable_ID:	ID of the system parameter to which new values are to be assigned (see Table 2-3)
Value:	The new value of the system parameter

Syntax: **Set_Value_Confirmation** **8750_{hex}**

Positive message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code: 8750_{hex} Message code of the service confirmation

Parameter_Count: Number of subsequent words
with a positive message:
0001_{hex} 1 parameter word
with a negative message:
0002_{hex} 2 parameter words

Result: Result of the service processing
0000_{hex} Indicates a positive message. The controller board executed the service successfully.
$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.

Add_Error_Info: Additional information on the error cause

2.3.5 "Read_Value" Service

Task:

This service can be used to read INTERBUS system parameters (variables).

For a list of defined system parameters (variables), please refer to the description of the "Set_Value" service (Table 2-3 on page 2-18).

Syntax: **Read_Value_Request** **0351_{hex}**

Word 1	Code	
Word 2	Parameter_Count	
Word 3	Variable_Count	
Word 4	Variable_ID	1st parameter
Word 5	Variable_ID	2nd parameter
...	...	
	Variable_ID	nth parameter

Bit	15 0

Key:

Code:	0351_{hex} Command code of the service request
Parameter_Count:	Number of subsequent words $xxxx_{hex}$ = 1 + *Variable_Count*
Variable_Count:	Number of system parameters to be read
Variable_ID:	ID of the system parameter to be read

Syntax: **Read_Value_Confirmation** **8351_{hex}**

Positive message

Word 1	Code	
Word 2	Parameter_Count	
Word 3	Result	
Word 4	Variable_Count	
Word 5	Variable_ID	1st system parameter
...	Value	
	...	
	Variable_ID	2nd system parameter
	Value	
	...	
	...	
	Variable_ID	nth system parameter
	Value	
	...	

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code: 8351_{hex} Message code of the service confirmation

Parameter_Count: Number of subsequent words

with a positive message:

$xxxx_{hex}$ The value depends on the number and length of the system parameters. The length of the system parameter (16 words, maximum) is specified in the third hexadecimal position (bits 8... 11) of the *Variable_ID* parameter.

with a negative message:

0002_{hex} 2 parameter words

Result: Result of the service processing

	0000_{hex} Indicates a positive message. The controller board executed the service successfully.
	$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.
Variable_Count:	Number of read system parameters
Variable_ID:	ID of the read system parameter
Value:	Value of the system parameter
Add_Error_Info:	Additional information on the error cause

2.3.6 "Initiate_Load_Configuration" Service

Task:

The "Initiate_Load_Configuration" service prepares the controller board to transmit a configuration to the master using the following services:

– "Load_Configuration" (0307_{hex})
– "Complete_Load_Configuration" ($030A_{hex}$)

To transmit a new configuration frame (*New_Config* parameter = 0001_{hex}), specify the *Frame_Reference* and *Device_Count* parameters (total number of devices).

Prerequisite:

The parameterization phase must have been initiated with the "Control_Parameterization" ($030E_{hex}$) service before.

Syntax: **Initiate_Load_Configuration_Request** **0306_{hex}**

Word 1	Code	
Word 2	Parameter_Count	
Word 3	New_Config	
Word 4	Frame_Reference	
Word 5	Device_Count	
Word 6	Extension_Length	Extension
...	...	Extension

Bit	15 8	7 0

Key:

Code: 0306_{hex} Command code of the service request

Parameter_Count: Number of subsequent words
$xxxx_{hex}$ = 3 + (*Extension_Length* + 1)/2

New_Config:
0001_{hex} The configuration frame is created again. An existing configuration frame is overwritten.
0000_{hex} Updates an existing configuration frame.

Frame_Reference: Logical number under which the configuration frame is to be stored.
If you want to overwrite an existing configuration frame (*New_Config* parameter = 0000_{hex}), only individual entries of the existing configuration are overwritten.

Device_Count:	Number of INTERBUS devices in the existing configuration frame or the new one to be loaded.
Extension_Length:	Length (in bytes) of the *Extension parameter* Value range: 00_{hex} ... $7F_{hex}$ (0 ... 127 characters)
Extension:	Any information in ASCII code, for example, you can specify the name of the configuration frame in ASCII code.

Syntax: **Initiate_Load_Configuration_Confirmation** **8306_{hex}**

Positive message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code: 8306_{hex} Message code of the service confirmation

Parameter_Count: Number of subsequent words

with a positive message:

0001_{hex} 1 parameter word

with a negative message:

0002_{hex} 2 parameter words

Result: Result of the service processing

0000_{hex} Indicates a positive message. The controller board executed the service successfully.

$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.

Add_Error_Info: Additional information on the error cause

2.3.7 "Load_Configuration" Service

Task:

The configuration frame describes each of the specified INTERBUS devices in a separate numbered entry. The sequence and the numbering of the entries corresponds to the physical bus configuration including all alternatives.
This service transfers the configuration data to the controller board in the form of a list. Use the *Used_ Attributes* parameter to determine which attributes the list should contain.

The "Load_Configuration" service does not check the consistency among the attributes but only whether this data is permitted in principle, e.g., whether it is within the value range.

Prerequisite:

Ensure that the controller board has been prepared for transmission using the following services:

– "Control_Parameterization" ($030E_{hex}$)
– "Initiate_Load_Configuration" (0306_{hex})

Syntax:

Load_Configuration_Request **0307_{hex}**

Word 1	Code	
Word 2	Parameter_Count	
Word 3	Used_Attributes	
Word 4	Start_Entry_No	
Word 5	Entry_Count	
Word 6	Configuration_Entry	1st device
...	...	
	Configuration_Entry	nth device

Bit	15 0

Key:

Code: 0307_{hex} Command code of the service request

Parameter_Count: Number of subsequent parameter words

$xxxx_{hex}$ The value depends on the *Entry_Count* parameter and the *Used_Attributes* parameter.

Used_Attributes:	Choice of add-on attributes. The parameter is a 16-bit field in which every bit corresponds to an attribute. Set the corresponding bit to 1 on the attribute that you want to transmit (see the "Configuration_Entry" syntax on page 2-33). Settings for the *Used_Attributes* parameter: Bit 0 Device number Bit 1 Device code Bit 2 Reserved (always 0_{bin}) Bit 3 Device level Bit 4 Logical group number Bit 5 Error characteristic Bit 6...13 Reserved (always 0_{bin}) Bit 14 Additional device information Bit 15 Reserved (always 0_{bin}) Example: If the entries only consist of the device code and the device level, enter the value $000A_{hex}$ for the *Used_Attributes* parameter (bits 1 and bit 3 are set).
Start_Entry_No:	Number of the first device for which attributes are to be transmitted
Entry_Count:	Number of devices for which attributes are to be transmitted
Configuration_Entry:	Attribute values of the individual devices to be transmitted according to their order in the physical bus configuration (see syntax on page 2-33)

According to the following syntax, enter attributes in the "Configuration_Entry" parameter block that have been enabled with the *Used_ Attributes* parameter (disabled attributes are not entered).

When several entries with several attributes are loaded at the same time, first all the attributes of one entry are loaded, then those of the next entry.

Syntax **"Configuration_Entry"** **Attribute**

Word x	Bus_Segment_No	Position		Device number
Word x+1	Length_Code	ID_Code		Device code
Word x+2	Device_Level			Device level
Word x+3	Group	Alternative		Log. group number
Word x+4	Reserved		Bit 0	Error characteristic
Word x+5...(259)				Additional device information
Bit	15 8	7 0		

Attributes:

Bus_Segment_No: Number of the bus segment where the device is located.
Value range: 01_{hex} .. FF_{hex} (1 .. 255_{dec})

Position: Physical location in the bus segment
Value range: 00_{hex} .. $3F_{hex}$ (63_{dec}) for local bus
The *Bus_Segment_No* and *Position* parameters together form the device number.

Length_Code: Length code
The length code refers to the address space required by the device in the host.

ID_Code: ID code
The ID code indicates the device type. It is printed as *Module Ident* in decimal notation on the modules (in hexadecimal notation on RT modules).
The *Length_Code* and *ID_Code* parameters together form the device number.

Device_Level: Device level
Value range: 00_{hex} ... $0F_{hex}$, (0 ... 15_{dec})

Group: Group to which the device belongs

Alternative:	Alternative bus segment to which the device belongs The *Group* and *Alternative* parameters together form the local group number.
Error characteristic:	
Bit 0	= 1: Isolated switching active = 0: Isolated switching not active
Bits 1 to 15	Reserved
Extended device information:	For a description see "Attribute: Additional Device Information" on page 2-49

As shown in the example on page 2-49, the initial value for the Configuration_Entry after calling the Load_Configuration service is (Word x+5].

Syntax: **Load_Configuration_Confirmation** **8307_{hex}**

Positive message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code:	8307_{hex}	Message code of the service confirmation
Parameter_Count:	Number of subsequent words	
	with a positive message:	
	0001_{hex}	Always 1 parameter word
	with a negative message:	

	0002_{hex}	Always 2 parameter words
	Result:	Result of the service processing
	0000_{hex}	Indicates a positive message. The controller board executed the service successfully.
	$xxxx_{hex}$	Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.
Add_Error_Info:	Additional information on the error cause	

2.3.8 "Terminate_Load_Configuration" Service

Task:

This service terminates the loading of the configuration data in segments. The service also checks the loaded configuration data for permissibility and consistency. If no error is detected, the controller board stores the data in the configuration directory under the *Frame_Reference* given in the "Initiate_Load_Configuration" (0306_{hex}) service. If an error is detected, the service is acknowledged with a negative confirmation.

Remark:

The *Default_Parameter* parameter can also be used to indicate whether the process data channel (PD channel) and/or the PCP channel are to be parameterized according to the loaded configuration frame. In this case the firmware automatically creates the process data reference list ("physical addressing") and/or a communication relationship list (CRL).

The "Terminate_Load_Configuration" service does not activate the newly loaded configuration immediately. It is only activated with the "Activate_Configuration" service (0711_{hex}). This service can be used to load and configure configurations that are not connected to the bus.

Syntax: **Terminate_Load_Configuration_Request** **0308_{hex}**

Word 1	Code
Word 2	Parameter_Count
Word 3	Default_Parameter

Bit	15 0

Key:

Code: 0308_{hex} Command code of the service request

Parameter_Count: Number of subsequent words

0001_{hex} 1 parameter word

Default_Parameter: Indicates whether a default parameterization of the PCP and/or PD channel is to be carried out for the loaded configuration:

0000_{hex} No automatic parameterization

0001_{hex} Automatic parameterization of the process data channel through the creation of the process data reference list

0002_{hex} Automatic parameterization of the PCP channel through the creation of the communication relationship list

0003_{hex} Automatic parameterization of the process data and PCP channel

Syntax: **Terminate_Load_Configuration_Confirmation** **8308_{hex}**

Positive message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code: 8308_{hex} Message code of the service confirmation

Parameter_Count: Number of subsequent words

with a positive message:

0001_{hex} 1 parameter word

with a negative message:

0002_{hex} 2 parameter words

Result: Result of the service processing

0000_{hex} Indicates a positive message. The controller board executed the service successfully.

$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.

Add_Error_Info: Additional information on the error cause

2.3.9 "Read_Configuration" Service

Task:

This service reads various entries of the configuration directory depending on the *Frame_Reference* and *Start_Entry_No* parameters.

Frame_ Reference	Start_ Entry_No	Entries Read by the Service
0000_{hex}	Not relevant (0000_{hex})	Header information for the configuration directory (CFG_OD_Header), i.e., the number of all configured configuration frames in ascending order.
$>0000_{hex}$	0000_{hex}	Header information for the configuration frame (CFG_Header) selected with the *Frame_Reference* parameter.
$>0000_{hex}$	$>0000_{hex}$	Entries of the configuration frame (CFG_Entry) selected with the *Frame_Reference* parameter. Either the entire configuration frame or only one part, e.g., a single INTERBUS device description can be read.

Syntax:

Read_Configuration_Request **0309_{hex}**

Word 1	Code
Word 2	Parameter_Count
Word 3	Frame_Reference
Word 4	Used_Attributes
Word 5	Start_Entry_No
Word 6	Entry_Count

Bit: 15 0

Key:

Code:
- 0309_{hex} Command code of the service request

Parameter_Count: Number of subsequent words
- 0004_{hex} 4 parameter words

Frame_Reference: Number of the configuration frame
- 0000_{hex} Reads the configuration frame directory.
- $xxxx_{hex}$ Number of the configuration frame to be read.

Only relevant if *Frame_Reference* > 0000_{hex}

Used_Attributes: Attributes to be read

The parameter is a 16-bit field in which every bit corresponds to an attribute. Set the corresponding bit to 1 on the attributes to be read.

Settings for the *Used_Attributes* parameter:

Bit 0	Device number
Bit 1	Device code
Bit 2	Reserved (always 0_{bin})
Bit 3	Device level
Bit 4	Logical group number
Bit 5	Error characteristic
Bit 6...7	Reserved (always 0_{bin})
Bit 8	global_bus_error
Bit 9	separate_bus_error
Bit 10	Device status
Bit 11	Additional module status information (e.g., single channel diagnostics)
Bit 12	Visual diagnostic information
Bit 13	Reserved (always 0_{bin})
Bit 14	Additional device information
Bit 15	Reserved (always 0_{bin})

Start_Entry_No: Position of the first entry

0000_{hex}	Only reads the header information for the configuration frame.
$xxxx_{hex}$	Reads the entries from the configuration directory from this number onwards.

Entry_Count: Number of entries to be read

Bit 15 This bit determines the entries in the *Start_Entry_No* parameter:
- = 0: Physical device number
- = 1: Logical device number

The positive message transmits the requested entries from the configuration directory. Depending on the *Frame_Reference* and *Start_Entry_No* parameters in the service request, it has one of the following structures.

Syntax **Read_Configuration_Confirmation** **8309_{hex}**

1st structure

Positive message during service request with:

- Frame_Reference = 0000_{hex}
- Start_Entry_No Not relevant (= 0000_{hex})

Word 1	Code	
Word 2	Parameter_Count	
Word 3	Result	
Word 4	More_Follows	
Word 5	Frame_Reference	= 0000_{hex}
Word 6	Current_Configuration	
Word 7	Configuration_Count	
Word 8	Frame_Reference 1	
...	...	
	Frame_Reference n	

2nd structure

Positive message during service request with:

- Frame_Reference > 0000_{hex}
- Start_Entry_No = 0000_{hex}

Word 1	Code	
Word 2	Parameter_Count	
Word 3	Result	
Word 4	More_Follows	
Word 5	Frame_Reference	> 0000_{hex}
Word 6	Used_Attributes	Not relevant
Word 7	Start_Entry_No	= 0000_{hex}
Word 8	Frame_Device_Count	
Word 9	Active_Device_Count	
Word 10	Frame_IO_Bit_Count	
Word 11	Active_IO_Bit_Count	
Word 12	Frame_PCP_Device_Count	
Word 13	Active_PCP_Device_Count	

Word 14	Frame_PCP_Word_Count	
Word 15	Active_PCP_Word_Count	
Word 16	Extension	Extension
Bit	15 0	

3rd structure

Positive message during service request with:

– Frame_Reference > 0000_{hex}

– Start_Entry_No > 0000_{hex}

Word 1	Code	
Word 2	Parameter_Count	
Word 3	Result	
Word 4	More_Follows	
Word 5	Frame_Reference	
Word 6	Used_Attributes	
Word 7	Start_Entry_No	
Word 8	Entry_Count	
Word 9	Configuration_Entry	1st device
...	...	
	Configuration_Entry	nth device

Negative message

Word 1	Code	
Word 2	Parameter_Count	
Word 3	Result	
Word 4	Add_Error_Info	
Bit	15 0	

Key:

Code: 8309_{hex} Message code of the service confirmation

Parameter_Count: Number of subsequent words

with a positive message and if *Frame_Reference* = 0000_{hex}:

$xxxx_{hex}$ = 5 + *Configuration_Count*

	with a positive message and if *Frame_Reference* > 0000_{hex} and *Start_Entry_No* = 0000_{hex}:
	$000D_{hex}$ 13 parameter words
	with a positive message and if *Frame_Reference* > 0000_{hex} and *Start_Entry_No* > 0000_{hex}:
	$xxxx_{hex}$ The value depends on the number of devices in the configuration frame and the number of enabled attributes.
	with a negative message:
	0002_{hex} 2 parameter words
Result:	Result of the service processing
	0000_{hex} Indicates a positive message. The service request has been executed successfully. The data is available in the following parameters.
	$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed (see also *Add_Error_Info*).
Add_Error_Info:	Additional information on the error cause
More_Follows:	0000_{hex} Indicates that all requested entries are contained in the service confirmation.
	0001_{hex} Indicates that the service confirmation does not contain all requested entries, as the amount of data is larger than the mailbox (MXI) that is available for the services. Call the service again to read the remaining data.
Frame_Reference:	Number of the read configuration frame. The parameter contains the value that was transferred with the service request.
Current_Configuration:	Number of the currently activated configuration frame.

Configuration_Count:	Number of configured configuration frames.
Frame_Reference x:	Number of all configured configuration frames in ascending order.
Frame_Device_Count:	Number of configured INTERBUS devices in the selected configuration frame.
Active_Device_Count:	Number of active INTERBUS devices in the selected configuration frame.
Frame_IO_Bit_Count:	Number of configured I/O bits in the selected configuration frame.
Active_IO_Bit_Count:	Number of active I/O bits in the selected configuration frame.
Frame_PCP_Device_Count:	Number of configured PCP devices in the selected configuration frame.
Active_PCP_Device_Count:	Number of active PCP devices in the selected configuration frame.
Frame_PCP_Word_Count:	Number of configured PCP words in the selected configuration frame.
Active_PCP_Word_Count:	Number of active PCP words in the selected configuration frame.
Extension:	Information transmitted with the "Initiate_Load_Configuration" service, e.g., a name for the configuration frame.
Used_Attributes:	Read attributes The parameter contains the value that was transferred with the service request.
Start_Entry_No:	Position of the first entry or 0000_{hex} if only the header information was read.
Entry_Count:	Number of entries that are transmitted by the service confirmation. The *More_Follows* parameter indicates if there are further entries.

Configuration_Entry: Selected entries in the order of the physical bus configuration.
The attributes contained in every entry are enabled in the service request by the *Used_Attributes* parameter (see the "Configuration_Entry" syntax on page 2-44).

A configuration entry for a device does not have to contain all attributes. If an attribute is not enabled in the service request by the *Used_Attributes* parameter, the configuration entry is reduced by the relevant data words.

In the following, the structure of a configuration entry is shown where **all** attributes are enabled.

Syntax **"Configuration_Entry"** **Attribute:**

Word x	Bus_Segment_No		Position	Device Number
Word x+1	Length_Code		ID_Code	Device Code
Word x+2	Device_Level			Device level
Word x+3	Group		Alternative	Log. group number
Word x+4	Reserved	Bit 0		Error characteristic
Word x+5	SUPI_Type		Add_Info	"global_bus_ error"
Word x+6	Transmission_Error			
Word x+7	Device_Error			
Word x+8	Transmission_Fail_DO			
Word x+9	Transmission_Fail_DI			
Word x+10	MAU_FAIL_DO			
Word x+11	MAU_FAIL_DI			
Word x+12	MAU_Warning_DO		MAU_Warning_DI	"separate_ bus_error"
Word x+13	Out1_FAIL			
Word x+14	Out2_FAIL			
Word x+15	Reconfigure_Request			
Word x+16	Peripheral_State			
Word x+17	Microprocessor_Watchdog			
Word x+18	Device_State			Device status

Word x+19	Channel 16 ... 31		Additional module status information
Word x+20	Channel 0 ... 15		
Word x+21	Initiator	Power	
Word x+22	Length	Optical Info Forward Line	Visual diagnostic information
Word x+23	Optical Info Return Line	- -	
Word x+24...(278)			Additional device information

Bit	15 8	7 0

Key:

Attribute: Device Number

Bus_Segment_No: Number of the bus segment where the INTERBUS device is located

Value range: 00_{hex} ... FF_{hex} (0_{dec} ... 255_{dec})

Position: Physical location in the bus segment

Value range:
00_{hex} ... $3F_{hex}$ (0_{dec} ... 63_{dec}) for installation local bus)

Attribute: Device Code

Length_Code: Length code

The length code refers to the address space required by the INTERBUS device in the host.

ID_Code: ID code

The ID code describes the INTERBUS device function. It is printed as *Module Ident* in decimal notation on the modules (in hexadecimal notation on RT modules).

Attribute: Device Level

Device_Level:	Device level The device level describes the branch level at which the device is located. Value range: 00_{hex} to $0F_{hex}$ (0 to 15_{dec})

Attribute: Logical Group Number

Group:	Group to which the device belongs
Alternative:	Alternative bus segment to which the device belongs

Attribute: Error Characteristic

Bit 0	= 1: Isolated switching active = 0: Isolated switching not active
Bits 1 to 15	Reserved

Attribute: "global_bus_error"

SUPI_Type:SUPI type

xx00	Old SUPIs and SUPI2
xx01	SUPI1
xx03	SUPI3 and later

Add_Info: SUPI chip identification

00_{hex}	SUPI older than SUPI3
$A1_{hex}$	IB8052
$A2_{hex}$	LPC2
$A3_{hex}$	LPC1
$A4_{hex}$	SUPI3-DPC
$A5_{hex}$	SUPI3
$D0_{hex}$	SUPI3-OPC
FF_{hex}	Reserved

Transmission_Error: Transmission error

Device_Error: Peripheral fault

Attribute: "separate_bus_error"

Transmission_Fail_DO:	Counter for transmission errors in the data forward path.
Transmission_Fail_DI:	Counter for transmission errors in the data return path.
MAU_FAIL_DO:	Counter for cable interrupts in the data forward path.
MAU_FAIL_DI:	Counter for cable interrupts in the data return path.
MAU_Warning_DO:	Counter for deterioration of the transmission quality in the forward path (for fiber optic transmission); for chip LPC2: overloaded internal power source.
MAU_Warning_DI:	Counter for deterioration of the transmission quality in the return path (for fiber optic transmission); for chip LPC2: IBS protocol chip temperature too high.
Out1_FAIL:	Counter for errors on the RBST signal (jumper in the connector) on the outgoing interface or error on the unused OUT1 interface.
Out2_FAIL:	Counter for errors on the LBST signal (jumper in the connector) on the outgoing interface or error on the unused OUT2 interface.
Reconfigure_Request:	Counter for reconfiguration requests; for chip LPC2: voltage too low for initiators.
Peripheral_Fault:	Counter for peripheral faults.
Microprocessor_Watchdog:	Counter for resets of the connected microprocessor, for chip LPC2: exceeding the permitted output current of the power drive.

Attribute: Device Status

Device_State	Status of the device		
	Bit 0, 1	Not relevant	
	Bit 2	1_{bin}	Device active
		0_{bin}	Device not active
	Bit 3, 4	Not relevant	
	Bit 5	1_{bin}	Device jumpered
		0_{bin}	Device not jumpered
	Bits 6...11	Reserved	
	Bit 12	1_{bin}	Device indicates PF (peripheral fault)
		0_{bin}	Device does not indicate a PF
	Bits 13...15	Reserved	

Attribute: Additional Module Status Information

Channel 16...31	Bit 0...15	A set bit indicates a peripheral fault on one of the module channels 16...31.
Channel 0...15	Bit 0...15	A set bit indicates a peripheral fault on one of the module channels 0...15.
Initiator	Bit 8...15	A set bit indicates the failure of the initiator supply on one of the module groups 1 to 8.
Power	Bit 0...3	A set bit indicates the failure of one of the supply voltages U1 to U4.

Attribute: Visual Diagnostic Information

Length	Bit 8...13	Length of the incoming path
	Bit 14	Optical medium
	Bit 15	Enable status
Optical Info Forward Line	Bit 0...3	Power level
	Bit 4:	Reserved
	Bit 5	Distortion
	Bits 6, 7	Reserved
Optical Info Return Line	Bit 8...11	Power level
	Bit 12:	Reserved
	Bit 13	Distortion
	Bits 14,15	Reserved

Attribute: Additional Device Information

The additional device information is transmitted as a byte string with a maximum length of 255 bytes.

As the data items in this byte string are transmitted directly one after the other, configuration entries following the additional device information may start at uneven addresses if the length of the additional device information is uneven.

Structure of the additional device information:

x_0	x_{11}	x_{12}	x_{13}	x_{21}	x_{22}	x_{23}	...

Key:

		Example	Description
x_0 =	Length of the additional device information in bytes	$0F_{hex}$	
x_{11} =	Extension 1 ID	01_{hex}	Service info (IBS CMD G4), ASCII-coded; 12 characters, maximum 01_{hex} = Station name
x_{12} =	Length of extension 1 in bytes	07_{hex}	
x_{13} =	Extension 1	4D 6F 74 6F 72 30 31_{hex}	"Motor01"
x_{21} =	Extension 2 ID	02_{hex}	
x_{22} =	Length of extension 2 in bytes	04_{hex}	
x_{23} =	Extension 2	19 04 20 02_{hex}	Information can be freely defined
... =	Other extensions up to a maximum total byte string length of 255 bytes.		

In the following example, the "Device name" service info is transmitted as an additional device information extension. The start address begins at "Word x + 24", based on the Configuration_Entry on page 2-44. This is where the additional device information with the same address starts.

Example:
Transmission of service info device name (IBS CMD)

Word x+24	Length of the entire extension in bytes [$0E_{hex}$]	*Service info* extension ID	Extension:
Word x+25	Length of the *service info* extension [$0C_{hex}$]	Byte 2	
Word x+26	Byte 3	Byte 4	
Word x+27	Byte 5	Byte 6	*Service info*:
Word x+28	Byte 7	Byte 8	device name
Word x+29	Byte 9	Byte 10	
Word x+30	Byte 11	Byte 12	
Word x+31	Byte 13	- -	
Bit	15 8	7 0	

2.3.10 "Compare_Configuration" Service

Task:

This service transfers the configuration data to the controller board in the form of a list. Use the *Used_ Attributes* parameter to determine which attributes the list should contain.

The controller board compares this list with the configuration frame specified with the *Frame_Reference* parameter. If the list and the configuration frame are not identical, a negative confirmation is generated and the controller board transmits information about the error in the form of error codes.

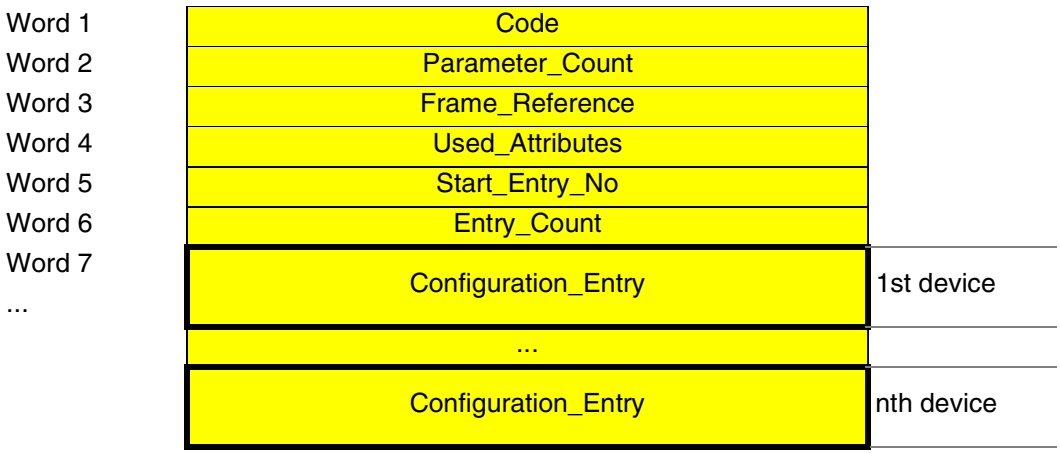
Prerequisite:

Ensure that the configuration frame to be specified with the *Frame_Reference* parameter is available on the controller board.

The configuration frame describes **each** of the specified INTERBUS devices in a separate numbered entry. The sequence and the numbering of the entries corresponds to the physical bus configuration including **all** alternatives.

Syntax:

Compare_Configuration_Request **0317_{hex}**



Key: Code: 0317_{hex} Command code of the service request

Parameter_Count:	Number of subsequent words $xxxx_{hex}$ The value depends on the number of entries and the number and type of attributes that you want to compare.
Frame_Reference:	Number of the configuration frame with which the controller board is to compare the transmitted list.
Used_Attributes:	Attributes of the configuration frame to be read. The parameter is a 16-bit field in which every bit corresponds to an attribute. Set the corresponding bits to 1 on the attribute that you want to read. Settings for the *Used_Attributes* parameter: Bit 0 Device number Bit 1 Device code Bit 2 Reserved (always 0_{bin}) Bit 3 Device level Bit 4 Logical group number Bit 5 Error characteristic Bit 6...15 Reserved (always 0_{bin})
Start_Entry_No:	Number of the first device in the configuration frame whose attributes are to be compared with the transmitted list.
Entry_Count:	Number of entries to be transmitted to the controller board.
Configuration_Entry:	Attribute values for an individual device. Only enter the attribute values that you selected with the *Used_Attributes* parameter and enter the entries in the order of the physical bus configuration. The *Configuration_Entry* parameters are described in the "Read_Configuration" service on page 2-38.

Syntax: **Compare_Configuration_Confirmation** **8317_{hex}**

Positive message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code: 8317_{hex} Message code of the service confirmation

Parameter_Count: Number of subsequent words

with a positive message:

0001_{hex} 1 parameter word

with a negative message:

0002_{hex} 2 parameter words

Result: Result of the service processing

0000_{hex} Indicates a positive message. The controller board executed the service successfully.

$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.

Add_Error_Info: Additional information on the error cause

2.3.11 "Complete_Load_Configuration" Service

Task:

This service transmits configuration data to the INTERBUS master in the form of attributes selected with *Used_Attributes*. It is specially adapted to the PLC programming requirements.

Prerequisite:

Ensure that the controller board has been prepared for transmission using the following services:

– "Control_Parameterization" ($030E_{hex}$, see page 2-11)

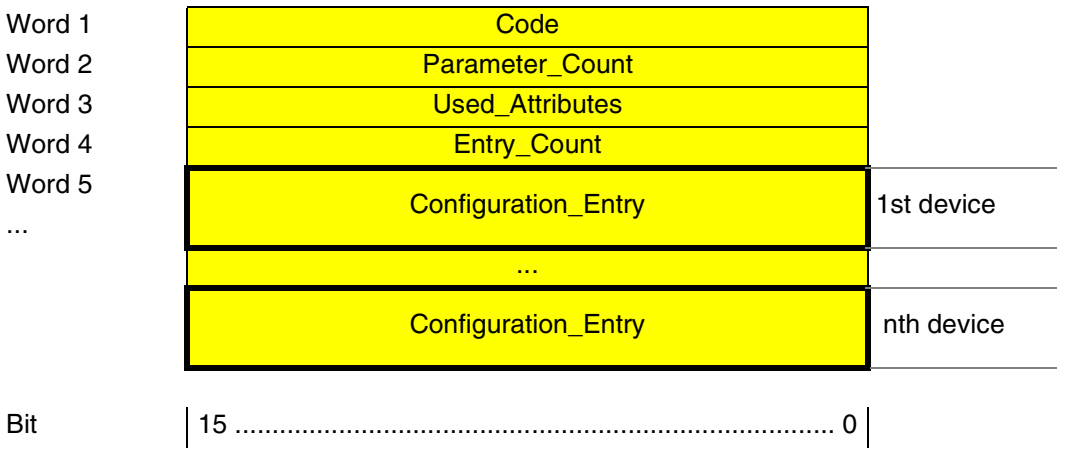
– "Initiate_Load_Configuration" (0306_{hex} see page 2-28)

Remark:

This service can be understood as a meta service for the "Load_Configuration" service (0307_{hex}, see page 2-31). This service transmits all entries of the configuration frame *(Start_Entry_No* = 1).

Syntax:

Complete_Load_Configuration_Request **$030A_{hex}$**



Key:

Code: $030A_{hex}$ Command code of the service request

Parameter_Count: Number of subsequent words

$xxxx_{hex}$ The value depends on the number of entries and the number and type of attributes that you want to transmit.

Used_Attributes: Attributes to be transmitted

The parameter is a 16-bit field in which every bit corresponds to an attribute. Set the corresponding bits to 1 on the attribute that you want to transmit.

Settings for the *Used_Attributes* parameter:

Bit 0	Device number
Bit 1	Device code
Bit 2	Reserved (always 0_{bin})
Bit 3	Device level
Bit 4	Logical group number
Bit 5	Error characteristic
Bit 6...13	Reserved (always 0_{bin})
Bit 14	Additional device information
Bit 15	Reserved (always 0_{bin})

Entry_Count: Number of entries to be transmitted

Configuration_Entry: Attribute values for an individual device

Only enter the attribute values that you selected with the *Used_Attributes* parameter and enter the entries in the order of the physical bus configuration. For the description of the *Configuration_Entry* parameters see the "Read_Configuration" service (0309_{hex}) on page 2-38.

Syntax: **Complete_Load_Configuration_Confirmation** **$830A_{hex}$**

Positive message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code: $830A_{hex}$ Message code of the service confirmation

Parameter_Count: Number of subsequent words

with a positive message:

0001_{hex} 1 parameter word

with a negative message:

0002_{hex} 2 parameter words

Result: Result of the service processing

0000_{hex} Indicates a positive message. The controller board executed the service successfully.

$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.

Add_Error_Info: Additional information on the error cause

2.3.12 "Complete_Read_Configuration" Service

Task: This service reads entries in the configuration directory in the form of one or more columns, which have been selected with the *Used_Attributes* parameter. It is specially adapted to the PLC programming requirements.

Remark: The service can be understood as a meta service for the "Read_Configuration" service (0309_{hex}). The *Start_Entry_No* parameter does not need to be specified, since this service reads all entries of the configuration frame *(Start_Entry_No* = 1).

Syntax: **Complete_Read_Configuration_Request** **$030B_{hex}$**

Word 1	Code
Word 2	Parameter_Count
Word 3	Used_Attributes

Bit	15 0

Key:

Code: $030B_{hex}$ Command code of the service request

Parameter_Count: Number of subsequent words

0001_{hex} Always 1 parameter word

Used_Attributes: The parameter is a 16-bit field in which every bit corresponds to an attribute. Set the corresponding bits to 1 on the attribute that you want to read.

Settings for the *Used_Attributes* parameter:

Bit 0	Device number
Bit 1	Device code
Bit 2	Reserved (always 0_{bin})
Bit 3	Device level
Bit 4	Logical group number
Bit 5	Error characteristic
Bit 6...7	Reserved (always 0_{bin})
Bit 8	global_bus_error
Bit 9	separate_bus_error
Bit 10	Device status
Bit 11	Additional module status information
Bit 12	Visual diagnostic information
Bit 13	Reserved (always 0_{bin})
Bit 14	Additional device information
Bit 15	Reserved (always 0_{bin})

Syntax: **Complete_Read_Configuration_Confirmation** **$830B_{hex}$**

Positive message

Word 1	Code	
Word 2	Parameter_Count	
Word 3	Result	
Word 4	More_Follows	
Word 5	Frame_Reference	
Word 6	Used_Attributes	
Word 7	Start_Entry_No	0001_{hex}
Word 8	Entry_Count	
Word 9	Configuration_Entry	1st device
...	...	
	Configuration_Entry	nth device

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code: $830B_{hex}$ Message code of the service confirmation

Parameter_Count: Number of subsequent words

with a positive message:

$xxxx_{hex}$ The value depends on the number of entries and the number and type of attributes that you want to read.

with a negative message:

0002_{hex} 2 parameter words

Result:	Result of the service processing 0000_{hex} Indicates a positive message. The controller board executed the service successfully. $xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.
Add_Error_Info:	Additional information on the error cause
More_Follows:	0000_{hex} Indicates that all requested entries are contained in the service confirmation. 0001_{hex} Indicates that the service confirmation does not contain all requested entries as the amount of data is larger than the mailbox (MXI) that is available for the services. Call the "Read_Configuration" service (0309_{hex}) to read the remaining data.
Frame_Reference:	Number of the active configuration frame
Used_Attributes:	Read attributes The parameter contains the value that was transferred with the service request.
Start_Entry_No:	Number of the first entry. 0001_{hex} With this service all entries are read, starting with the first entry.
Entry_Count:	Number of entries that are transferred by the service confirmation.
Configuration_Entry:	Entries in the order of the physical bus configuration. The attributes contained in every entry are enabled in the service request by the *Used_Attributes* parameter. For the description of the *Configuration_Entry* parameters see "Read_Configuration" service (0309_{hex}) on page 2-38.

2.3.13 "Delete_Configuration" Service

Task:

This service deletes an inactive configuration frame from the configuration directory.

Syntax:

Delete_Configuration_Request **$030C_{hex}$**

Word 1	Code
Word 2	Parameter_Count
Word 3	Frame_Reference

Bit	15 0

Key:

Code:	$030C_{hex}$	Command code of the service request
Parameter_Count:	Number of subsequent words	
	0001_{hex}	1 parameter word
Frame_Reference:	Number of the configuration frame to be deleted	

Syntax: **Delete_Configuration_Confirmation** **$830C_{hex}$**

Positive message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code: $830C_{hex}$ Message code of the service confirmation

Parameter_Count: Number of subsequent words
with a positive message:
0001_{hex} 1 parameter word
with a negative message:
0002_{hex} 2 parameter words

Result: Result of the service processing
0000_{hex} Indicates a positive message. The controller board executed the service successfully.
$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.

Add_Error_Info: Additional information on the error cause

2.3.14 "Create_Configuration" Service

Task:

This service causes the controller board to automatically generate a configuration frame from the currently connected configuration and to activate it in order to start the bus. After the execution of the service the controller board is in the *Active* state (display: *ACTV*).

The new configuration frame and the active configuration are stored in the configuration directory under the number specified in the *Frame_Reference* parameter. If there is already a configuration frame under this number, this frame is overwritten. In addition, the controller board generates default process data description lists, a default process data reference list, and a default communication relationship list (CRL) according to the currently connected bus configuration. In the device descriptions the attributes are initialized as follows:

Device_Number:	According to the active configuration
Length_Code:	According to the active configuration
ID_Code:	According to the active configuration
Device_Level:	According to the active configuration
Group_Number:	For all INTERBUS devices $FFFF_{hex}$ (i.e., no group number)
Device_State:	All INTERBUS devices are active

Syntax: **Create_Configuration_Request** **0710_{hex}**

Word 1	Code
Word 2	Parameter_Count
Word 3	Frame_Reference

Bit	15 0

Key:

Code:	0710_{hex} Command code of the service request
Parameter_Count:	Number of subsequent words 0001_{hex} 1 parameter word
Frame_Reference:	Number of the configuration frame to be generated automatically

Syntax: **Create_Configuration_Confirmation** **8710_{hex}**

Positive message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code: 8710_{hex} Message code of the service confirmation

Parameter_Count: Number of subsequent words
with a positive message:
0001_{hex} 1 parameter word
with a negative message:
0002_{hex} 2 parameter words

Result: Result of the service processing
0000_{hex} Indicates a positive message. The controller board executed the service successfully.
$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.

Add_Error_Info: Additional information on the error cause

2.3.15 "Activate_Configuration" Service

Task:

This service enables the controller board to check the configuration data of the configuration frame for

– conformance with the currently connected configuration

– address overlaps

If no errors are detected, the controller board activates this configuration frame (display: *ACTV*) and runs ID cycles at regular intervals. The number of the configuration frame is indicated to the controller board by the *Frame_Reference* parameter.

Prerequisite:

If you want to activate a configuration frame, another configuration frame cannot be active at the same time. Use the "Deactivate_Configuration" service (0712_{hex}) to deactivate a configuration frame.

Syntax:

Activate_Configuration_Request **0711_{hex}**

Word 1	Code
Word 2	Parameter_Count
Word 3	Frame_Reference

Bit	15 0

Key:

Code: 0711_{hex} Command code of the service request

Parameter_Count: Number of subsequent words
0001_{hex} 1 parameter word

Frame_Reference: Number of the configuration frame to be activated

Syntax: **Activate_Configuration_Confirmation** **8711_{hex}**

Positive message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code: 8711_{hex} Message code of the service confirmation

Parameter_Count: Number of subsequent words

with a positive message:

0001_{hex} 1 parameter word

with a negative message:

0002_{hex} 2 parameter words

Result: Result of the service processing

0000_{hex} Indicates a positive message. The controller board executed the service successfully.

$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.

Add_Error_Info: Additional information on the error cause

2.3.16 "Deactivate_Configuration" Service

Task:

This service deactivates the specified configuration frame. No ID cycles are run any more. After the execution of the service the controller board is in the *Ready* state (display: *RDY*).

If you previously executed the "Stop_Data_Transfer" service (0702_{hex}), this service has left behind "frozen" output states (set outputs). These outputs are reset when the configuration frame is deactivated.

Prerequisite:

The specified configuration frame must not only exist, it must also be active when the service is called.

Syntax: **Deactivate_Configuration_Request** **0712_{hex}**

Word 1	Code
Word 2	Parameter_Count
Word 3	Frame_Reference

Bit	15 0

Key:

Code: 0712_{hex} Command code of the service request

Parameter_Count: Number of subsequent words
0001_{hex} 1 parameter word

Frame_Reference: Number of the configuration frame to be deactivated

Syntax: **Deactivate_Configuration_Confirmation** **8712_{hex}**

Positive message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code: 8712_{hex} Message code of the service confirmation

Parameter_Count: Number of subsequent words

with a positive message:

0001_{hex} 1 parameter word

with a negative message:

0002_{hex} 2 parameter words

Result: Result of the service processing

0000_{hex} Indicates a positive message. The controller board executed the service successfully.

$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.

Add_Error_Info: Additional information on the error cause

2.4 Services for Defining Process Data Descriptions

2.4.1 "Initiate_Put_Process_Data_Description_List" Service

Task:

This service initiates the definition of process data descriptions. It prepares the controller board for the "Put_Process_Data_ Description_List" service (0321_{hex}, see page 2-70).

Prerequisite:

Ensure that the controller board has been prepared for parameterization using the following services:

– "Control_Parameterization" ($030E_{hex}$)
– "Initiate_Load_Configuration" (0306_{hex})
– "Complete_Load_Configuration" ($030A_{hex}$)

The configuration frame must be loaded since it contains the device numbers to which the process data descriptions are to be assigned.

Syntax: **Initiate_Put_Process_Data_Description_List_Request** **0320_{hex}**

Word 1	Code
Word 2	Parameter_Count

Bit	15 0

Key:

Code: 0320_{hex} Command code of the service request

Parameter_Count: Number of subsequent words
0000_{hex} No parameter word

Syntax: **Initiate_Put_Process_Data_Description_List_Confirmation** **8320_{hex}**

Positive message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code: 8320_{hex} Message code of the service confirmation

Parameter_Count: Number of subsequent words

with a positive message:

0001_{hex} 1 parameter word

with a negative message:

0002_{hex} 2 parameter words

Result: Result of the service processing

0000_{hex} Indicates a positive message. The controller board executed the service successfully.

$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.

Add_Error_Info: Additional information on the error cause

2.4.2 "Put_Process_Data_Description_List" Service

The firmware automatically defines a process data description (PDD) for every INTERBUS device. This description comprises its entire process data.

For example, for an INTERBUS device with 16 bits of input and 16 bits of output data, the firmware automatically defines:

– A process data description of 16 input bits
– A process data description of 16 output bits

Task:

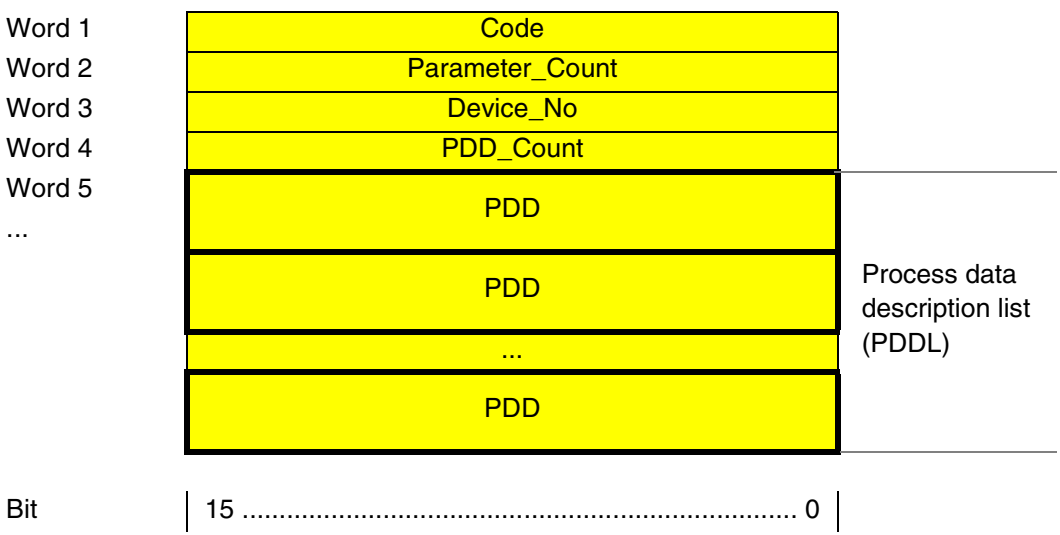
The "Put_Process_Data_Description_List" service can be used to define several process data descriptions for one INTERBUS device, thereby splitting up the device process data into smaller elements. The service writes these definitions to the process data description list.

Prerequisite:

Ensure that the controller board has been prepared using the following services:

– "Control_Parameterization" ($030E_{hex}$)
– "Initiate_Put_Process_Data_Description_List" (0320_{hex})

Syntax: **Put_Process_Data_Description_List_Request** **0321_{hex}**



Key: Code: 0321_{hex} Command code of the service request

Parameter_Count:	Number of subsequent words $xxxx_{hex}$ The number of parameter words depends on the number and size of the process data descriptions.
Device_No:	Device number of the INTERBUS device in the process data description list (PDDL) of which the process data descriptions (PDD) are to be written.
PDD_Count:	Number of process data descriptions.
PDD:	Process data description (PDD) (see the following syntax).

Syntax

Process data description (PDD):

PDD_Index	
Data_Direction	Data_Type
Byte_Position	
Bit_Position	Length
Extension_Length	Extension
...	Extension

Bit	15 8	7 0

Key:

PDD_Index:	Index of the process data description Assign a different *PDD_Index* for every process data description in order to identify it clearly. Every *PDD_Index* may be assigned only once per device. Permissible value range for the *PDD_Index*: 0000_{hex} ... $7FFF_{hex}$ Already assigned: 6010_{hex} Default IN process data descriptions 6011_{hex} Default OUT process data descriptions
Data_Direction:	Data direction of the process data description: $0C_{hex}$ For IN process data $0D_{hex}$ For OUT process data

Data_Type:	Type of the process data description $0A_{hex}$ Byte string process data item $0F_{hex}$ Bit string process data item
Byte_Position:	Byte offset (byte in the process data area of the device): 0000_{hex} 1st byte (bits 0 ... 7), i.e., the low-order byte 0001_{hex} 2nd byte (bits 8 ... 15), i.e., the next higher byte 0002_{hex} 3rd byte (bits 16 ... 23) ... nth byte depending on the size of the device
Bit_Position:	Bit offset Beginning of the bit string in the byte selected with the *Byte_Position* parameter. Permissible value range: 00_{hex} ... 07_{hex} for bit 0 to bit 7
Length:	Length of the process data item Value range: 1 ... 64 bytes for byte string process data item 1 ... 8 bits for bit string process data item A bit string must not exceed a byte boundary.
Extension_Length:	Length of the *Extension* parameter Value range: 00_{hex} ... $7F_{hex}$, i.e., 0 ... 127 bytes
Extension:	Information in ASCII code for this process data description, e.g., a name

Syntax: **Put_Process_Data_Description_List_Confirmation** **8321_{hex}**

Positive message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code: 8321_{hex} Message code of the service confirmation

Parameter_Count: Number of subsequent words

with a positive message:

0001_{hex} 1 parameter word

with a negative message:

0002_{hex} 2 parameter words

Result: Result of the service processing

0000_{hex} Indicates a positive message. The controller board executed the service successfully.

$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.

Add_Error_Info: Additional information on the error cause

2.4.3 "Terminate_Put_Process_Data_Description_List" Service

Task: This service terminates a write access to a process data description list (PDDL).

Syntax: **Terminate_Put_Process_Data_Description_List_Request** **0322_{hex}**

Word 1	Code
Word 2	Parameter_Count

Bit	15 0

Key:

Code: 0322_{hex} Command code of the service request

Parameter_Count: Number of subsequent words

0000_{hex} No parameter word

Syntax: **Terminate_Put_Process_Data_Description_List_Confirmation**

8322_{hex}

Positive message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code: 8322_{hex} Message code of the service confirmation

Parameter_Count: Number of subsequent words

with a positive message:

0001_{hex} 1 parameter word

with a negative message:

0002_{hex} 2 parameter words

Result: Result of the service processing

0000_{hex} Indicates a positive message. The controller board executed the service successfully.

$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.

Add_Error_Info: Additional information on the error cause

2.4.4 "Get_Process_Data_Description_List" Service

Task:

This service reads one or more process data descriptions (PDDs) from the process data description list (PDDL) of an INTERBUS device.

Use the *Start_PDD_Index* parameter to specify whether you want to read:

– Only the header of the process data description list
– One or more process data descriptions

Syntax:

Get_Process_Data_Description_List_Request **0323_{hex}**

Word 1	Code
Word 2	Parameter_Count
Word 3	Device_No
Word 4	Start_PDD_Index
Word 5	PDD_Count

Bit	15 0

Key:

Code: 0323_{hex} Command code of the service request

Parameter_Count: Number of subsequent words
0003_{hex} Always 3 parameter words

Device_No: Device number of the INTERBUS device from the process data description list (PDDL) of which you want to read process data descriptions (PDDs).

Start_PDD_Index: 0000_{hex} Read only the header of the process data description list.
$xxxx_{hex}$ Index of the first process data description (PDD) to be read.

PDD_Count: Number of process data descriptions (only relevant if *Start_PDD_Index* > 0000_{hex})

Syntax: **Get_Process_Data_Description_List_Confirmation** **8323_{hex}**

Positive message and *Start_PDD_Index* > 0000_{hex}

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	More_Follows
Word 5	Device_No
Word 6	Start_PDD_Index
Word 7	PDD_Count
...	PDD
	...
	PDD

Process data description list (PDDL)

Positive message and *Start_PDD_Index* = 0000_{hex}

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	More_Follows
Word 5	Device_No
Word 6	Start_PDD_Index
Word 7	PDD_Count

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code: 8323_{hex} Message code of the service confirmation

Parameter_Count: Number of subsequent words with positive message and *Start_PDD_Index* > 0000_{hex}:

	$xxxx_{hex}$ The number of parameter words depends on the number and size of the process data description.
	with positive message and *Start_PDD_Index* parameter = 0000_{hex}:
	0005_{hex} 5 parameter words
	with a negative message:
	0002_{hex} 2 parameter words
Result:	Result of the service processing
	0000_{hex} Indicates a positive message, The controller board executed the service successfully.
	$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.
Add_Error_Info:	Additional information on the error cause
More_Follows:	0000_{hex} Indicates that all requested process data descriptions are contained in the service confirmation.
	0001_{hex} Indicates that the service confirmation does not contain all requested entries as the amount of data is larger than the mailbox (MXI) that is available for the services. Call the service again to read the remaining data. Enter the index of the last transmitted process data description, incremented by 0001_{hex} for the *Start_PDD_Index* parameter.
Device_No:	Device number of the INTERBUS device from the process data description list (PDDL) for which the process data descriptions (PDD) were read.
Start_PDD_Index:	Index of the first process data description that was read.
PDD_Count:	Number of process data descriptions that were read or exist in the process data description list (PDDL).

PDD:	Process data descriptions (see the "Put_Process_Data_Description_List" syntax on page 2-70 and onwards).

2.5 Services for Assigning Process Data

2.5.1 "Initiate_Load_Process_Data_Reference_List" Service

Task:

This service initiates the definition of process data references (PDRs). It prepares the controller board for the "Load_Process_Data_Reference_List" service (0325_{hex}).

Use the *New_PDRL* parameter to indicate whether the service is to overwrite an existing process data reference list (PDRL). This is useful if you have created a process data reference list with the "Create_Configuration" service (0710_{hex}) before ("physical addressing"), but now want to carry out your own configuration.

Prerequisite:

Ensure that the controller board has been prepared for parameterization ("Control_Parameterization"). The configuration frame must be loaded ("Initiate_Load_Configuration", then "Complete_Load_ Configuration"). All process data descriptions used in the PDRL must be loaded ("Initiate_Put_Process_Data_Description_List", then "Put_Process_Data_Description_List").

Syntax: **Initiate_Load_Process_Data_Reference_List_Request** **0324_{hex}**

Word 1	Code
Word 2	Parameter_Count
Word 3	New_PDRL

Bit	15 0

Key:

Code:	0324_{hex}	Command code of the service request
Parameter_Count:	Number of subsequent words	
	0001_{hex}	1 parameter word
New_PDRL:	Method of operation	
	0000_{hex}	Does not overwrite an existing PDRL.
	0001_{hex}	Overwrites an existing PDRL.

Syntax: **Initiate_Load_Process_Data_Reference_List_Confirmation** **8324_{hex}**

Positive message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code: 8324_{hex} Message code of the service confirmation

Parameter_Count: Number of subsequent words

with a positive message:

0001_{hex} 1 parameter word

with a negative message:

0002_{hex} 2 parameter words

Result: Result of the service processing

0000_{hex} Indicates a positive message. The controller board executed the service successfully.

$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.

Add_Error_Info: Additional information on the error cause

2.5.2 "Load_Process_Data_Reference_List" Service

Task:

This service is used to define the process data references (PDRs) in a process data reference list. You may define process data references for input data, output data and direct link data.

The order in which process data references (PDRs) are transmitted for input data, output data and direct link data is of no importance. The controller board stores the data in the order of the ascending *PDR indices* (see "process data reference" syntax).

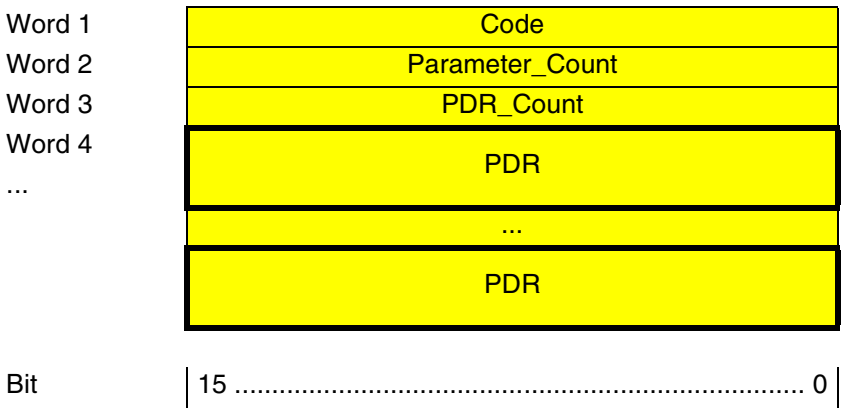
Prerequisite:

Ensure that the controller board has been prepared using the following services:

– "Control_Parameterization" ($030E_{hex}$)
– "Initiate_Load_Process_Data_Reference_List" (0324_{hex})

Syntax:

Load_Process_Data_Reference_List_Request **0325_{hex}**



Key:

Code:	0325_{hex}	Command code of the service request
Parameter_Count:	Number of subsequent words	
	$xxxx_{hex}$	The value depends on the number and size of the process data references.
PDR_Count:	Number of process data references to be transmitted	
PDR:	Process data references	

Syntax

Process data reference for **input data**:

Word x+1	PDR_Index		
Word x+2	Device_No		Source
Word x+3	PDD_Index		
Word x+4	Destination_Address		Destination
Word x+5	Bit_Position	Data_Cons	
Word x+6	Extension_Length	Extension	
Word x+7	...	Extension	

Process data reference for **output data**:

Word x+1	PDR_Index		
Word x+2	Source_Address		Source
Word x+3	Bit_Position	Data_Cons	
Word x+4	Device_No		Destination
Word x+5	PDD_Index		
Word x+6	Extension_Length	Extension	
Word x+7	...	Extension	

Process data reference for **direct link data**:

Word x+1	PDR_Index		
Word x+2	Source_Device_No		Source
Word x+3	Source_PDD_Index		
Word x+4	Dest_Device_No		Destination
Word x+5	Dest_PDD_Index		
Word x+6	Extension_Length	Extension	
Word x+7	...	Extension	

Bit	15 8	7 0

Key:

PDR_Index: Index of the process data reference (PDR)
Permissible value range:
0001_{hex} ... $3FFF_{hex}$ for input data
4000_{hex} ... $7FFF_{hex}$ for output data
8000_{hex} ... $BFFF_{hex}$ for direct link data

Assign a different *PDR_Index* for every process data reference in order to identify it clearly. Every *PDR_Index* may be assigned only once.

Device_No:	INTERBUS device (Segment . Position)
PDD_Index:	Index of the process data item
Source_Address:	Source address in the MPM (output data), recommended value range: 0000_{hex} ... 0400_{hex}
Destination_Address:	Destination address in the MPM (input data), recommended value range: 1000_{hex} ... 1400_{hex}
Bit_Position:	Beginning of a bit string at the destination/source address in the MPM Permissible value range: 00_{hex} ... 07_{hex} for bit 0 ... bit 7
Data_Cons:	Data consistency for process data access Permissible values: 00_{hex} 16 bits (standard) 01_{hex} 32 bits (e.g., encoders, operator interfaces) 02_{hex} 8 bits for modules or process data that are less than/equal to 8 bits 03_{hex} 64 bits (e.g., encoders, operator interfaces)

The data consistency ensures that the specified data width comes from one INTERBUS cycle.
The standard value for I/O modules is 16 bits. For INTERBUS devices which demand the consistent transmission of larger data widths, you must increase the data consistency (e.g., encoders, operator interfaces, or analog modules with a resolution higher than 16 bits). The data consistency must be identical within one word.
You may

- Divide one word into two bytes which are consistent in themselves (8 bits each)
- Define one word as being consistent in itself (16 bits)
- Assign one word to a larger consistency area (32 or 64 bits)

It is not permitted to assign a data consistency of 8 bits to one byte of a word and another larger consistency area to the other byte.

Extension_Length:	Length of the *Extension* parameter Value range: 00_{hex} ... $7F_{hex}$, i.e., 0 ... 127 characters
Extension:	Information in ASCII code for this process data reference, e.g., a comment
Source_Device_No:	INTERBUS device the input data for which is to be assigned
Source_PDD_Index:	Index of the IN process data item

Dest_Device_No:	INTERBUS device to which you want to assign the data
Dest_PDD_Index:	Index of the OUT process data item

Syntax: **Load_Process_Data_Reference_List_Confirmation** **8325_{hex}**

Positive message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code:	8325_{hex}	Message code of the service confirmation
Parameter_Count:		Number of subsequent words
		with a positive message:
	0001_{hex}	1 parameter word
		with a negative message:
	0002_{hex}	2 parameter words
Result:		Result of the service processing
	0000_{hex}	Indicates a positive message. The controller board executed the service successfully.
	$xxxx_{hex}$	Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.
Add_Error_Info:		Additional information on the error cause

2.5.3 "Terminate_Load_Process_Data_Reference_List" Service

Task: This service terminates the definition of process data references (PDRs).

Syntax: **Terminate_Load_Process_Data_Reference_List_Request** **0326_{hex}**

Word 1	Code
Word 2	Parameter_Count

Bit	15 0

Key:

Code: 0326_{hex} Command code of the service request

Parameter_Count: Number of subsequent words

0000_{hex} No parameter word

Syntax: **Terminate_Load_Process_Data_Reference_List_Confirmation**

8326_{hex}

Positive message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code: 8326_{hex} Message code of the service confirmation

Parameter_Count: Number of subsequent words

with a positive message:

0001_{hex} 1 parameter word

with a negative message:

0002_{hex} 2 parameter words

Result: Result of the service processing

0000_{hex} Indicates a positive message. The controller board executed the service successfully.

$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.

Add_Error_Info: Additional information on the error cause

2.5.4 "Read_Process_Data_Reference_List" Service

Task:

This service can be used to read the current process data reference list (PDRL).

Use the *Start_PDR_Index* parameter to indicate whether you want to read:

– Only the header of the process data reference list
– One or more process data references

Syntax: **Read_Process_Data_Reference_List_Request** **0327_{hex}**

Word 1	Code
Word 2	Parameter_Count
Word 3	Start_PDR_Index
Word 4	Entry_Count

Bit	15 0

Key:

Code: 0327_{hex} Command code of the service request

Parameter_Count: Number of subsequent words
0002_{hex} 2 parameter words

Start_PDR_Index:
0000_{hex} Read only the header of the process data reference list (PDRL)
$xxxx_{hex}$ Index of the first process data reference (PDR) to be read

Entry_Count: Number of process data references (only relevant if *Start_PDR_Index* > 0000_{hex})

Syntax: **Read_Process_Data_Reference_List_Confirmation** **8327_{hex}**

Positive message and *Start_PDR_Index* > 0000_{hex}

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	More_Follows
Word 5	Start_PDR_Index
Word 6	PDR_Count
Word 7	PDR
...	...
	PDR

Process data reference list

Positive message and *Start_PDR_Index* = 0000_{hex}

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	More_Follows
Word 5	Start_PDR_Index
Word 6	PDR_Count

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code: 8327_{hex} Message code of the service confirmation

Parameter_Count: Number of subsequent words

with positive message and *Start_PDR_Index* > 0000_{hex}:

$xxxx_{hex}$ The value depends on the number and size of the process data references.

with positive message and
Start_PDR_Index = 0000_{hex}

0004_{hex} 4 parameter words

with a negative message:

0002_{hex} 2 parameter words

Result: Result of the service processing

0000_{hex} Indicates a positive message. The controller board executed the service successfully.

$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.

Add_Error_Info: Additional information on the error cause

More_Follows:

0000_{hex} Indicates that all requested process data references are contained in the service confirmation.

0001_{hex} Indicates that the service confirmation does not contain all requested process data references since the amount of data is larger than the mailbox (MXI) that is available for the services. Call the service again to read the remaining data. Enter the index of the last transmitted process data reference, incremented by 0001_{hex} for the *Start_PDR_Index* parameter.

Start_PDR_Index: Index of the first process data reference (PDR) read.

Permissible value ranges for the *PDR_Indices*:

For input data	0000_{hex} ... $3FFF_{hex}$
For output data	4000_{hex} ... $7FFF_{hex}$
For direct link data	8000_{hex} ... $BFFF_{hex}$

PDR_Count: Number of process data references (PDR) read or included in the process data reference list (PDRL).

PDR: Process data references in the order of the ascending indices (see page 2-83).

2.5.5 "Compact_Load_Process_Data_Reference_List" Service

Task:

PLC users can use this service to configure the controller board without support from software such as IBS CMD SWT. A host address list for the IN and OUT process data is created for this purpose. The firmware then automatically generates corresponding entries for the process data reference list using these lists.
The "Compact_Load_Process_Data_Reference_List" service replaces the following service sequence:

– "Initiate_Load_Process_Data_Reference_List" (0324_{hex})
– "Load_Process_Data_Reference_List" (0325_{hex})
– "Terminate_Load_Process_Data_Reference_List" (0326_{hex})

It permits an address assignment as used for controller boards with firmware Version 3.x.

Transmit the addresses in the host-specific address format. The firmware converts the host-specific address format into the physical 32-bit addresses of the MPM.

Syntax: **Compact_Load_Process_Data_Reference_List_Request** **0328_{hex}**

Word 1	Code		
Word 2	Parameter_Count		
Word 3	Address_Direction		
Word 4	Entry_Count		
Word 5	Data_Cons	PD_Address	1st device
Word 6	Data_Cons	PD_Address	2nd device
	...		...
	Data_Cons	PD_Address	nth device

Bit	15 | 14	13 0

Key:

Code: 0328_{hex} Command code of the service request

Parameter_Count: Number of subsequent words
$xxxx_{hex}$ = 2 + *Entry_Count*

Address_Direction: Data direction for addresses:
1000_{hex} Input addresses
2000_{hex} Output addresses

Entry_Count: Number of entries in the host address list

Data_Cons: Data consistency for process data access

Permissible values:

00_{bin}	16 bits (standard)
01_{bin}	32 bits (e.g., encoders, operator interfaces)
10_{bin}	8 bits for modules or process data that are less than/equal to 8 bits
11_{bin}	64 bits (e.g., encoders, operator interfaces)

The data consistency ensures that the specified data width comes from one INTERBUS cycle.
The standard value for I/O modules is 16 bits. For INTERBUS devices which demand the consistent transmission of larger data widths, you must increase the data consistency (e.g., encoders, operator interfaces, or analog modules with a resolution higher than 16 bits). The data consistency must be identical within one word.
You may

– Divide one word into two bytes which are consistent in themselves (8 bits each)
– Define one word as being consistent in itself (16 bits)
– Assign one word to a larger consistency area (32 or 64 bits)

It is not permitted to assign a data consistency of 8 bits to one byte of a word and another larger consistency area to the other byte.

PD_Address: Input address list

(*Address_Direction* = 1000_{hex}):

For all devices with IN process data, enter the host addresses here, in the order of their physical locations. For devices with more than one IN process data word (e.g., with 32 bits), the next higher addresses are automatically assigned as well.

Enter the value $FFFF_{hex}$ here for all modules without IN process data (e.g., dedicated bus terminal modules or output modules) in the *Data_Cons*/*PD_Address* word.

Output address list

(*Address_Direction* = 2000_{hex}):

For all devices with OUT process data, enter the host addresses here, in the order of their physical locations. For devices with more than one OUT process data word (e.g., with 32 bits), the next higher addresses are automatically assigned as well.

Enter the value $FFFF_{hex}$ here for all modules without OUT process data (e.g., dedicated bus terminal modules or input modules) in the *Data_Cons*/*PD_Address* word.

Assign the start addresses of INTERBUS devices with address areas of 16 bits (or more) to even addresses only.

Syntax: **Compact_Load_Process_Data_Reference_List_Confirmation**

8328_{hex}

Positive message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code: 8328_{hex} Message code of the service confirmation

Parameter_Count: Number of subsequent words

with a positive message:

0001_{hex} 1 parameter word

with a negative message:

0002_{hex} 2 parameter words

Result: Result of the service processing

0000_{hex} Indicates a positive message. The controller board executed the service successfully.

$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.

Add_Error_Info: Additional information on the error cause

2.5.6 "Compact_Read_Process_Data_Reference_List" Service

Task:

This service can be used to read the address lists of your control or computer system that have been configured with the "Compact_Load_Process_Data_Reference_List" (0328_{hex}) service. The firmware converts the physical 32-bit addresses of the MPM into the specified address format of your control or computer system.

Prerequisite:

This service can only be executed successfully if the process data reference lists were configured with the "Compact_Load_Process_Data_Reference_List" (0328_{hex}) service.

Syntax: **Compact_Read_Process_Data_Reference_List_Request** **0329_{hex}**

Word 1	Code
Word 2	Parameter_Count
Word 3	Address_Direction

Bit	15 0

Key:

Code: 0329_{hex} Command code of the service request

Parameter_Count: Number of subsequent words
0001_{hex} 1 parameter word

Address_Direction: Data direction for addresses:
1000_{hex} Input address list
2000_{hex} Output address list

Syntax: **Compact_Read_Process_Data_Reference_List_Confirmation**

8329_{hex}

Positive message

Word 1	Code			
Word 2	Parameter_Count			
Word 3	Result			
Word 4	Address_Direction			
Word 5	Entry_Count			
Word 6	Data_Cons	PD_Address		1st device
Word 7	Data_Cons	PD_Address		2nd device
...	...			...
	Data_Cons	PD_Address		nth device

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 | 14	13 0

Key:

Code: 8329_{hex} Message code of the service confirmation

Parameter_Count: Number of subsequent words

with a positive message:

$xxxx_{hex}$ = 3 + *Entry_Count*

with a negative message:

0002_{hex} 2 parameter words

Result: Result of the service processing

$xxxx_{hex}$ Indicates a positive message. The controller board executed the service successfully.

0002_{hex} Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.

Add_Error_Info:	Additional information on the error cause
Address_Direction:	Data direction for addresses: 1000_{hex} Input address list 2000_{hex} Output address list
Entry_Count:	Number of entries in the address list
PD_Address:	Input address list (*Address_Direction* = 1000_{hex}): The input address list contains the host addresses for all devices with IN process data in the order of their physical locations. For devices with more than one IN process data word (e.g., with 32 bits), the next higher addresses are automatically assigned as well. Output address list (*Address_Direction* = 2000_{hex}): For all devices with OUT process data, enter the host addresses here, in the order of their physical locations. For devices with more than one OUT process data word (e.g., with 32 bits), the next higher addresses are automatically assigned as well. The firmware enters the value $FFFF_{hex}$ as the address for INTERBUS devices without process data of the corresponding list type.

2.6 Services for Direct INTERBUS Access

2.6.1 "Start_Data_Transfer" Service

Task:

This service activates the cyclic data traffic on the bus. After the execution of the service, the controller board is in the *Run* state (display: *RUN*).

Prerequisite:

Before the service is called, the controller board must be in the *Active* state, i.e., a configuration frame has been activated and ID cycles are already being run at regular intervals.

Syntax:

Start_Data_Transfer_Request **0701_{hex}**

Word 1	Code
Word 2	Parameter_Count

Bit	15 0

Key:

Code: 0701_{hex} Command code of the service request

Parameter_Count: Number of subsequent words

0000_{hex} No parameter word

Syntax: **Start_Data_Transfer_Confirmation** **8701_{hex}**

Positive message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code: 8701_{hex} Message code of the service confirmation

Parameter_Count: Number of subsequent words

with a positive message:

0001_{hex} 1 parameter word

with a negative message:

0002_{hex} 2 parameter words

Result: Result of the service processing

0000_{hex} Indicates a positive message. The controller board executed the service successfully.

$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.

Add_Error_Info: Additional information on the error cause

2.6.2 "Alarm_Stop" Service

Task:

This service triggers a long reset on the bus. Data traffic is stopped. Modules with process data set their outputs to the value 0. The command is executed directly after the current data cycles have been completed. After the execution of the service, the controller board is in the *Ready* state (display: *RDY*).

Syntax:

Alarm_Stop_Request **1303_{hex}**

Word 1	Code
Word 2	Parameter_Count

Bit	15 0

Key:

Code: 1303_{hex} Command code of the service request

Parameter_Count: Number of subsequent words

0000_{hex} No parameter word

Syntax: **Alarm_Stop_Confirmation** **9303_{hex}**

Positive message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code: 9303_{hex} Message code of the service confirmation

Parameter_Count: Number of subsequent words
with a positive message:
0001_{hex} 1 parameter word
with a negative message:
0002_{hex} 2 parameter words

Result: Result of the service processing
0000_{hex} Indicates a positive message. The controller board executed the service successfully.
$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.

Add_Error_Info: Additional information on the error cause

2.6.3 "Stop_Data_Transfer" Service

Task:

This service stops the cyclic data traffic on the bus. After the execution of the service, the controller board is in the *Active* state (display: *ACTV*) and runs ID cycles at regular intervals.

The controller board does not switch the output data to the safe state. Any set outputs are **not reset** but remain static.

Prerequisite:

The controller board must be in the *Run* state before the service is called.

Syntax: **Stop_Data_Transfer_Request** **0702_{hex}**

Word 1	Code
Word 2	Parameter_Count
Word 3	Stop_Type

Bit	15 0

Key:

Code: 0702_{hex} Command code of the service request

Parameter_Count: Number of subsequent words
0001_{hex} 1 parameter word

Stop_Type: Stop type:
0000_{hex} Data transfer stop

Syntax: **Stop_Data_Transfer_Confirmation** **8702_{hex}**

Positive message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code: 8702_{hex} Message code of the service confirmation

Parameter_Count: Number of subsequent words

with a positive message:

0001_{hex} 1 parameter word

with a negative message:

0002_{hex} 2 parameter words

Result: Result of the service processing

0000_{hex} Indicates a positive message. The controller board executed the service successfully.

$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.

Add_Error_Info: Additional information on the error cause

2.6.4 "Control_Active_Configuration" Service

Task:

This service can be used to selectively switch INTERBUS devices on or off. Depending on whether this service is used in the active or inactive state of the bus system, the changes in the configuration frame become effective immediately or when the "Activate_Configuration" (0711_{hex}) service is called.

Syntax: **Control_Active_Configuration_Request** **0713_{hex}**

Word 1	Code	
Word 2	Parameter_Count	
Word 3	Switch_Code	
Word 4	Entry_Count	List of INTERBUS device numbers
Word 5	Device_No	
Word 6	Device_No	
...	...	
	Device_No	

Bit	15 0

Key:

Code: 0713_{hex} Command code of the service request

Parameter_Count: Number of subsequent words
$xxxx_{hex}$ = 2 + *Entry_Count*

Switch_Code: This parameter specifies the states into which the listed INTERBUS devices are to be switched.

Possible values are:

0000_{hex} Segment_Off
The specified device and all devices that depend on it are switched off. These are:
- All devices belonging to the same bus segment
- All devices belonging to the same logic group
- Devices that are located after device *Device_No x* in the data ring

0001_{hex} Segment_On
The specified device and all devices that depend on it are switched on. Please observe the special treatment for groups that can be switched alternatively.

0002_{hex} Device_Off
Only the specified device is switched off in the configuration frame. For this, all devices that depend on this device must be switched off individually and manually.

0003_{hex} Device_On
Only the specified device is switched on in the configuration frame. For this, all devices that depend on this device must be switched on individually and manually.

0004_{hex} Device_Disable
Only the specified device is switched off in the configuration frame. It must not remain physically in the data ring and must be jumpered manually.

0005_{hex} Device_Enable
Only the specified device is switched on in the configuration frame. Insert it manually into the data ring.

Entry_Count: Number of devices in the list of INTERBUS device numbers.

Device_No: INTERBUS device number of the device to be switched.

Syntax: **Control_Active_Configuration_Confirmation** **8713_{hex}**

Positive message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code: 8713_{hex} Message code of the service confirmation

Parameter_Count: Number of subsequent words

with a positive message:

0001_{hex} 1 parameter word

with a negative message:

0002_{hex} 2 parameter words

Result: Result of the service processing

0000_{hex} Indicates a positive message. The controller board executed the service successfully.

$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.

Add_Error_Info: Additional information on the error cause

2.6.5 "Control_Device_Function" Service

Task:

This service can be used to send control commands to one or more INTERBUS devices; for example, to confirm device status errors or to set an alarm output.

Code 0314_{hex} can be used to call an identical but non-exclusive service, "Control_Device_Function_Not_Exclusive".

Syntax:

Control_Device_Function_Request **0714_{hex}**

Word 1	Code	
Word 2	Parameter_Count	
Word 3	Device_Function	
Word 4	Entry_Count	
Word 5	Device_No	List of INTERBUS devices
Word 6	Device_No	
...	...	
	Device_No	
Bit	15 0	

Key:

Code: 0714_{hex} Command code of the service request

Parameter_Count: Number of subsequent words

$xxxx_{hex}$ = 2 + *Entry_Count*

Device_Function: Control command to be sent to the devices.

The following control commands are available:

0001_{hex} Set_Alarm:
Set bus terminal module alarm outputs.

0002_{hex} Reset_Alarm:
Reset bus terminal module alarm outputs.

0003_{hex} Conf_Dev_Err
Confirm a device peripheral fault (PF).

0004_{hex} Conf_Dev_Err_All
Confirm all device peripheral faults (PF).

	As devices do not have to be specified, the list of device numbers is not required for this control command. Set the *Entry_Count* parameter = 0000_{hex}.
	0005_{hex} Reserved
	0006_{hex} Reserved
	0007_{hex} Refresh optical diagnostics
Entry_Count:	Number of devices to which the control command is to be sent. Set this parameter to 0000_{hex}, if the parameter *Device_Function* = 0004_{hex}.
Device_No:	Device numbers of the devices to which the control command is to be sent (not required if *Device_Function* = 0004_{hex}).

Syntax: **Control_Device_Function_Confirmation** **8714_{hex}**

Positive message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code: 8714_{hex} Message code of the service confirmation

Parameter_Count: Number of subsequent words

with a positive message:

0001_{hex} 1 parameter word

with a negative message:

0002_{hex} 2 parameter words

Result: Result of the service processing

0000_{hex} Indicates a positive message. The controller board executed the service successfully.

$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.

Add_Error_Info: Additional information on the error cause

2.6.6 "Reset_Controller_Board" Service

Task:

The "Reset_Controller_Board" service initiates a controller board reset.

Prerequisite:

Before calling this service, ensure that the state of your system permits a controller board reset.

Syntax:

Reset_Controller_Board_Request **0956_{hex}**

Word 1	Code
Word 2	Parameter_Count
Word 3	Reset_Type

Bit	15 0

Key:

Code: 0956_{hex} Command code of the service request

Parameter_Count: Number of subsequent words

0001_{hex} 1 parameter word

Reset_Type: Reset type:

0000_{hex} Cold start

0001_{hex} Warm start

This service is not acknowledged. Ensure that the controller board is only accessed again after a waiting time of **7** seconds. During this time, lock access to the controller board. Do not call up any mailbox functions during this time.

If the driver is called up during this time, errors may arise in the INTERBUS system, which prevent the startup of the controller board.

2.7 Diagnostic Services

2.7.1 "Confirm_Diagnostics" Service

Task:

This service updates the content of the diagnostic registers, the error areas in the MPM, and the LED diagnostic indicators on the front panel of the controller board.

Syntax: **Confirm_Diagnostics_Request** **0760_{hex}**

Word 1	Code
Word 2	Parameter_Count

Bit	15 0

Key:

Code: 0760_{hex} Command code of the service request

Parameter_Count: Number of subsequent words

0000_{hex} No parameter word

Syntax: **Confirm_Diagnostics_Confirmation** **8760_{hex}**

Positive message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code: 8760_{hex} Message code of the service confirmation

Parameter_Count: Number of subsequent words

with a positive message:

0001_{hex} 1 parameter word

with a negative message:

0002_{hex} 2 parameter words

Result: Result of the service processing

0000_{hex} Indicates a positive message. The controller board executed the service successfully.

$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.

Add_Error_Info: Additional information on the error cause

2.7.2 "Get_Error_Info" Service

Task: This service can be used to read the exact error cause and location after a bus error has been indicated. A maximum of ten errors are analyzed.

Syntax: **Get_Error_Info_Request** **0316_{hex}**

Word 1	Code	
Word 2	Parameter_Count	

Bit	15 0

Key:

Code: 0316_{hex} Command code of the service request

Parameter_Count: Number of subsequent words

0000_{hex} No parameter word

Syntax: **Get_Error_Info_Confirmation** **8316_{hex}**

Positive message, as long as error localization is still in progress

Word 1	Code	
Word 2	Parameter_Count	
Word 3	Result	
Word 4	Entry_Count	= 0001_{hex}
Word 5	Error_Code	= $0BDF_{hex}$
Word 6	Add_Error_Info	= $FFFF_{hex}$

Positive message, if error localization has been completed

Word 1	Code	
Word 2	Parameter_Count	
Word 3	Result	
Word 4	Entry_Count	
Word 5	Error_Code	1st error
Word 6	Add_Error_Info	
Word 7	Error_Code	2nd error
Word 8	Add_Error_Info	
...	...	...
	Error_Code	nth error
	Add_Error_Info	

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:	Code:	8316_{hex} Message code of the service confirmation
	Parameter_Count:	Number of subsequent words with positive message (during error localization): 0004_{hex} 4 parameter words with positive message (after error localization): $00xx_{hex}$ = 2 + 2 × *Entry_Count* (20 words, maximum) with a negative message: 0002_{hex} Always 2 parameter words
	Result:	Result of the service processing 0000_{hex} Indicates a positive message. The controller board executed the service successfully. $xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.
	Entry_Count:	Number of indicated errors (10, maximum). Every error description consists of two words: *Error_Code* and *Add_Error_Info.*
	Error_Code:	Information on the error type
	Add_Error_Info:	with positive message: Error location (*Bus segment* **.** *Position*), if it could be located. with negative message: Additional information on the error cause

2.7.3 "Read_Device_State" Service

Task:

This service reads selectable status information of all INTERBUS devices of the active configuration for test and diagnostic purposes.

Syntax:

Read_Device_State_Request **0315_{hex}**

Word 1	Code
Word 2	Parameter_Count
Word 3	Device_State_Mask

Bit	15 0

Key:

Code: 0315_{hex} Command code of the service request

Parameter_Count: Number of subsequent words

0001_{hex} 1 parameter word

Device_State_Mask: Status information to be read

The parameter is a 16-bit field in which every bit corresponds to an item of status information. Set the corresponding bits to 1 on the items of status information that you want to read.

Bit assignments:

Bit 0: Alarm output
The alarm output of the specified INTERBUS device is set.

Bit 1: Error output
The specified INTERBUS device is indicating an error.

Bit 2: IVC detection
The voltage is too low for the initiators.

Bit 3: TEMP detection
The ambient temperature is too high.

Bit 4: CSD detection
The internal power source is overloaded.

Bit 5: PSD detection
The permitted output current of the power drive was exceeded.

Bits 6 - 8 Reserved (always 0_{bin})

Bit 9: MAU detection of the incoming remote bus interface (data ring forward path)
MAU = **M**edium **A**ttachment **U**nit
The set bit indicates that the transmission path is still functioning at the specified interface and that the attenuation is too high, e.g., for fiber optic paths.

Bit 10: MAU detection of the incoming remote bus interface (data ring return path)

Bit 11: Peripheral fault
Depending on the value of bit 12 (peripheral fault mode), the INTERBUS device indicates a peripheral fault or a microprocessor reset.

Bit 12: Peripheral fault mode
- 0_{bin} Bit 11 indicates a peripheral fault.
- 1_{bin} Bit 11 indicates a microprocessor reset.

Bit 13: Peripheral fault extension
If bit 13 and bit 11 (peripheral fault) are set, the single channel diagnostics provides additional details about the peripheral fault. This data can be read with the "Read_Configuration_Request" service (0309_{hex}) (see page 2-38).

Bit 14, 15 Reserved (always 0_{bin})

Syntax: **Read_Device_State_Confirmation** **8315_{hex}**

Positive message

Word 1	Code	
Word 2	Parameter_Count	
Word 3	Result	
Word 4	More_Follows	
Word 5	States_Count	
Word 6	Device_No	1st message
Word 7	Device_States	
...	...	
	Device_No	nth message
	Device_States	

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code: 8315_{hex} Message code of the service confirmation

Parameter_Count: Number of subsequent words

with a positive message:

$xxxx_{hex}$ $= 3 + 2 \times$ *States_Count*

with a negative message:

0002_{hex} 2 parameter words

Result: Result of the service processing

0000_{hex} Indicates a positive message. The controller board executed the service successfully.

$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.

Add_Error_Info: Additional information on the error cause

More_Follows:	0000_{hex}	Indicates that all requested status messages are contained in the service confirmation.
	0001_{hex}	Indicates that the service confirmation does not contain all requested status messages since the amount of data is larger than the mailbox (MXI) that is available for the services. Confirm the transmitted status messages with the "Control_Device_Function" (0714_{hex}) service and then call the "Read_Device_State" (0315_{hex}) service again.
States_Count:	Number of status messages Every status message consists of two parameters: *Device_No* and *Device_States*.	
Device_No:	Number of the INTERBUS device that provided the status information.	
Device_States:	Status information of the device The meaning of the bits correspond to that of the *Device_State_Mask* parameter in the service request.	

2.7.4 "Get_Version_Info" Service

Task: This service can be used to read the type, version, manufacturing date, etc. of the hardware and firmware of your controller board.

Syntax: **Get_Version_Info_Request** **$032A_{hex}$**

Word 1	Code
Word 2	Parameter_Count

Bit	15 0

Key:
- Code: $032A_{hex}$ Command code of the service request
- Parameter_Count: Number of subsequent words
  - 0000_{hex} No parameter word

Syntax: **Get_Version_Info_Confirmation** **$832A_{hex}$**

Positive message

Word 1	Code			
Word 2	Parameter_Count			
Word 3	Result			
Words 4 + 5	FW_Version	(byte 1)	FW_Version	(byte 2)
	FW_Version	(byte 3)	FW_Version	(byte 4)
Words 6 ... 8	FW_State	(byte 1)	...	
	...		FW_State	(byte 6)
Words 9 ... 11	FW_Date	(byte 1)	...	
	...		FW_Date	(byte 6)
Words 12 ... 14	FW_Time	(byte 1)	...	
	...		FW_Time	(byte 6)
Words 15 ... 24	Host_Type	(byte 1)	...	
	...		Host_Type	(byte 20)
Words 25 +26	Host_Version	(byte 1)	Host_Version	(byte 2)
	Host_Version	(byte 3)	Host_Version	(byte 4)

Words 27 ... 29	Host_State (byte 1)	...
	...	Host_State (byte 6)
Words 30 ... 32	Host_Date (byte 1)	...
	...	Host_Date (byte 6)
Words 33 ... 35	Host_Time (byte 1)	...
	...	Host_Time (byte 6)
Words 36 + 37	Start_FW_Version (byte 1)	Start_FW_Version (byte 2)
	Start_FW_Version (byte 3)	Start_FW_Version (byte 4)
Words 38 ... 40	Start_FW_State (byte 1)	...
	...	Start_FW_State (byte 6)
Words 41 ... 43	Start_FW_Date (byte 1)	...
	...	Start_FW_Date (byte 6)
Words 44 ... 46	Start_FW_Time (byte 1)	...
	...	Start_FW_Time (byte 6)
Words 47 ... 50	HW_Art_No (byte 1)	...
	...	HW_Art_No (byte 8)
Words 51 ... 65	HW_Art_Name (byte 1)	...
	...	HW_Art_Name (byte 30)
Words 66 + 67	HW_Motherboard_ID (byte 1)	HW_Motherboard_ID (byte 2)
	HW_Motherboard_ID (byte 2)	HW_Motherboard_ID (byte 4)
Word 68	HW_Version (byte 1)	HW_Version (byte 2)
Words 69 ... 78	HW_Vendor_Name (byte 1)	...
	...	HW_Vendor_Name (byte 20)
Words 79 ... 84	HW_Serial_No (byte 1)	...
	...	HW_Serial_No (byte 12)
Words 85 ... 87	HW_Date (byte 1)	...
	...	HW_Date (byte 6)

Bit	15 0

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit | 15 ..... 0 |

Key:

Code: $832A_{hex}$ Message code of the service confirmation

Parameter_Count: Number of subsequent words
with a positive message:
0055_{hex} 55 parameter words
with a negative message:
0002_{hex} 2 parameter words

Result: Result of the service processing
0000_{hex} Indicates a positive message. The controller board executed the service successfully.
$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.

Add_Error_Info: Additional information on the error cause

Version information for the hardware and firmware. Every byte indicates the ASCII code for a character:

FW_Version:	Version of the firmware core (e.g., 33 2E 39 37_{hex} for "Version 3.97")	(4 bytes)
FW_State:	Status of the firmware (e.g., 62 65 64 61 00 00_{hex} for "beta" with preliminary versions)	(6 bytes)
FW_Date:	Creation date of the firmware (e.g., 33 31 31 30 39 35_{hex} for 10/31/95)	(6 bytes)

FW_Time:	Creation time of firmware (e.g., 31 34 31 30 32 30_{hex} for 14:10:20)	(6 bytes)
Host_Type:	Type of the host-specific firmware interface (e.g., IBS MA S5 DSC)	(20 bytes)
Host_Version:	Version of the host-specific firmware interface	(4 bytes)
Host_State:	State of the host-specific firmware interface	(6 bytes)
Host_Date:	Creation date of the host-specific firmware interface	(6 bytes)
Host_Time:	Creation time of the host-specific firmware interface	(6 bytes)
Start_FW_Version:	Version of start firmware	(4 bytes)
Start_FW_State:	Status of start firmware	(6 bytes)
Start_FW_Date:	Creation date of start firmware	(6 bytes)
Start_FW_Time:	Creation time of start firmware	(6 bytes)
HW_Art_No:	Controller board order number	(8 bytes)
HW_Art_Name:	Controller board order designation	(30 bytes)
HW_Motherboard_ID:	Motherboard identification (e.g., 32 43_{hex} for "2C" on IBS S5 DSC)	(4 bytes)
HW_Version:	Hardware version	(2 bytes)
HW_Vendor_Name:	Controller board manufacturer	(20 bytes)
HW_Serial_No:	Controller board serial number	(12 bytes)
HW_Date:	Controller board manufacture date	(6 bytes)

2.7.5 "Get_Diag_Info" Service

Task:

This service can be used to read the statistical and global diagnostic information of a configuration frame. The positive confirmation message contains the diagnostic information in several blocks, which belong together logically. Each block contains a counter (*Changed_Info_Count*, or *CIC*), which is incremented when the diagnostic information of this block changes.

Syntax: **Get_Diag_Info_Request** **$032B_{hex}$**

Word 1	Code
Word 2	Parameter_Count
Word 3	Frame_Reference
Word 4	Diag_Info_Attr

Bit	15 0

Key:

Code: $032B_{hex}$ Command code of the service request

Parameter_Count: Number of subsequent words
0002_{hex} 2 parameter words

Frame_Reference: Number of the configuration frame for which the diagnostic information is to be read

Diag_Info_Attr: Selection of the diagnostic information to be read.

The parameter is a 16-bit field in which every bit corresponds to a diagnostic block. Set the corresponding bits to 1 on the diagnostic block that you want to read.

Bit assignments (the most important are printed in bold):

Bit 0 Changed_Info_Count (CIC)
Bit 1 Statistics_State
Bit 2 Global_Count
Bit 3 **Top_Ten_CRC**
Bit 4 **Last_Ten_PF**
Bit 5 IPMS_Error
Bit 6 **Bus_Error**
Bit 7 PF_Error (peripheral fault)
Bit 8 Time_Out_Error
Bit 9...15 Reserved (always 0_{bin})

The following service description shows the positive confirmation message displayed if you have enabled all permissible attributes with the *Diag_Info_Attr* parameter.

In practice, please enable only one attribute at a time to ensure that the diagnostic information is read selectively and in a clearly structured manner. You are advised to read the ***Changed_Info_Count*** diagnostic block first, in order to ascertain what information has changed.

Syntax: **Get_Diag_Info_Confirmation** **$832B_{hex}$**

Positive message

Word 1	Code	
Word 2	Parameter_Count	
Word 3	Result	
Word 4	Frame_Reference	
Word 5	Diag_Info_Attr	
Word 6	Diag_Block	Diagnostic blocks, as defined in the *Diag_Info_Attr* attribute
...		

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code: $832A_{hex}$ Message code of the service confirmation

Parameter_Count: Number of subsequent words

with a positive message:

$xxxx_{hex}$ The value depends on the number and size of the selected diagnostic blocks.

with a negative message:

0002_{hex} 2 parameter words

Result:	Result of the service processing 0000_{hex} Indicates a positive message. The controller board executed the service successfully. $xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.
Add_Error_Info:	Additional information on the error cause
Frame_Reference:	Indicates the configuration frame from which the diagnostic information was read. The parameter contains the value that was transferred with the service request.
Diag_Info_Attr:	Selection of the read diagnostic information. The parameter contains the value that was transferred with the service request.
Diag_Block:	All requested diagnostic information in the form of diagnostic blocks, which are described below. The individual diagnostic blocks are listed in the order of the corresponding bit numbers of the *Diag_Info_Attributes* parameter.

Syntax

Changed_Info_Count diagnostic block

Words 1 + 2	Top_Ten_CRC_CIC	(bits 31 ... 16)	Changed_ Info_Count
	Top_Ten_CRC_CIC	(bits 15 ... 0)	
Words 3 + 4	Last_Ten_PF_CIC	(bits 31 ... 16)	
	Last_Ten_PF_CIC	(bits 15 ... 0)	
Words 5 + 6	IPMS_Error_CIC	(bits 31 ... 16)	
	IPMS_Error_CIC	(bits 15 ... 0)	
Words 7 + 8	Bus_Error_CIC	(bits 31 ... 16)	
	Bus_Error_CIC	(bits 15 ... 0)	
Words 9 + 10	PF_CIC	(bits 31 ... 16)	
	PF_CIC	(bits 15 ... 0)	
Words 11 +12	Time_Out_Error_CIC	(bits 31 ... 16)	
	Time_Out_Error_CIC	(bits 15 ... 0)	

The diagnostic block contains the contents of the 32-bit counters for the diagnostic blocks. The counter is incremented by "1" each time the information for the respective diagnostic block changes.

Top_Ten_CRC_CIC:	Counter for *Top_Ten_CRC* diagnostic block
Last_Ten_PF_CIC:	Counter for *Last_Ten_PF* diagnostic block
IPMS_Error_CIC:	Counter for *IPMS_Error* diagnostic block
Bus_Error_CIC:	Counter for *Bus_Error* diagnostic block
PF_CIC:	Counter for *PF* diagnostic block
Time_Out_Error_CIC:	Counter for *Time_Out_Error* diagnostic block

Syntax

Statistics_State diagnostic block

Word 1	Statistics_State

Statistics_State: Status of the statistical diagnostics:

0000_{hex} Statistical diagnostics is activated.

0001_{hex} Statistical diagnostics is deactivated.

Syntax

Global_Count diagnostic block

Words 1 + 2	Cycle_Count	(bits 31 ... 16)
	Cycle_Count	(bits 15 ... 0)
Words 3 + 4	Cycle_Error_Count	(bits 31 ... 16)
	Cycle_Error_Count	(bits 15 ... 0)
Words 5 + 6	ID_Cycle_Count	(bits 31 ... 16)
	ID_Cycle_Count	(bits 15 ... 0)
Words 7 + 8	ID_Cycle_Error_Count	(bits 31 ... 16)
	ID_Cycle_Error_Count	(bits 15 ... 0)
Words 9 + 10	Data_Cycle_Count	(bits 31 ... 16)
	Data_Cycle_Count	(bits 15 ... 0)
Words 11 + 12	Data_Cycle_Error_Count	(bits 31 ... 16)
	Data_Cycle_Error_Count	(bits 15 ... 0)

Global_Count

The *Global_Count* diagnostic block contains the number of INTERBUS cycles (data and ID) as 32-bit counter contents.

Cycle_Count:	Count of INTERBUS cycles
Cycle_Error_Count:	Count of faulty INTERBUS cycles
ID_Cycle_Count:	Count of ID cycles
ID_Cycle_Error_Count:	Count of faulty ID cycles
Data_Cycle_Count:	Count of data cycles
Data_Cycle_Error_Count:	Count of faulty data cycles

Syntax

Top_Ten_CRC diagnostic block

Words 1 + 2	Top_Ten_CRC_CIC	(bits 31 ... 16)	Top_Ten_CRC
	Top_Ten_CRC_CIC	(bits 15 ... 0)	
Word 3	1st Device_No_Bus_Fault		
Words 4 + 5	Error_Count	(bits 31 ... 16)	
	Error_Count	(bits 15 ... 0)	
Word 6	Error_Code		
...	...		
Word 39	10th Device_No_Bus_Fault		
Words 40 + 41	Error_Count	(bits 31 ... 16)	
	Error_Count	(bits 15 ... 0)	
Word 42	Error_Code		

The *Top_Ten_CRC* diagnostic block contains the list of all INTERBUS devices with the most frequently occurring bus errors. According to the number of bus errors (*Error_Count*), the ten devices are listed in descending order.

Top_Ten_CRC_CIC:	The 32-bit counter is incremented by "1" each time the information for the *Top_Ten_CRC* diagnostic block changes. The *Top_Ten_CRC* diagnostic block comprises 10 items of diagnostic information, each of which is 4 words in length:
Device_No_Bus_Fault:	INTERBUS device number (*Device_No*)
Error_Count:	Error number of this error (32-bit counter)
Error_Code:	Information on the error type

Syntax

Last_Ten_PF diagnostic block

Words 1 + 2	Last_Ten_PF_CIC	(bits 31 ... 16)	Last_Ten_PF
	Last_Ten_PF_CIC	(bits 15 ... 0)	
Word 3	1st Device_No_PF		
Word 4	Error_Code		
...	...		
Word 21	10th Device_No_PF		
Word 22	Error_Code		

The *Last_Ten_PF* diagnostic block contains a list of the last ten INTERBUS devices on which a peripheral fault (PF) occurred. The device on which the most recent peripheral fault occurred is indicated first:

Last_Ten_PF_CIC:	The 32-bit counter is incremented by "1" each time the information for the *Last_Ten_PF* diagnostic block changes. The *Last_Ten_PF* diagnostic block comprises 10 items of diagnostic information, each of which is 24 words in length:
Device_No_PF:	INTERBUS device number (*Device_No*)
Error_Code:	Information on the error type

Syntax

IPMS_Error diagnostic block

Words 1 + 2	IPMS_Error_CIC	(bits 31 ... 16)	IPMS_Error
	IPMS_Error_CIC	(bits 15 ... 0)	
Word 3	Type		

IPMS_Error_CIC:	The 32-bit counter is incremented by "1" each time the information for the *IPMS_Error* diagnostic block changes.
Type:	Content of the IPMS error register (see the IBS IPMS3 UM E User Manual for the master protocol chip).

Syntax

Bus_Error diagnostic block

Words 1 + 2	Bus_Error_CIC	(bits 31 ... 16)
	Bus_Error_CIC	(bits 15 ... 0)
Word 3	Type	
Word 4	Device_No	

Bus_Error

Bus_Error_CIC: The 32-bit counter is incremented by "1" each time the information for the *Bus_Error* diagnostic block changes.

Type: Information on the error type

Device_No: Number of the INTERBUS device

Syntax

PF diagnostic block

Words 1 + 2	PF_CIC	(bits 31 ... 16)
	PF_CIC	(bits 15 ... 0)
Word 3	Type	
Word 4	Device_No	

PF

PF_CIC: The 32-bit counter is incremented by "1" each time the information for the *PF* diagnostic block changes.

Type: Information on the error type

Device_No: Number of the INTERBUS device

Syntax

Time_Out_Error diagnostic block

Words 1 + 2	Time_Out_Error_CIC	(bits 31 ... 16)
	Time_Out_Error_CIC	(bits 15 ... 0)
Word 3	Type	
Word 4	Device_No	

Time_Out_ Error

Time_Out_Error_CIC: The 32-bit counter is incremented by "1" each time the information for the *Time_Out_Error* diagnostic block changes.

Type: Information on the error type

Device_No: Number of the INTERBUS device

Words 3 and 4 of the *Time_Out_Error* diagnostic block are transmitted a total of ten times, i.e., for the last 10 devices.

2.7.6 "Control_Statistics" Service

Task: This service can be used to control statistical diagnostics.

Syntax: **Control_Statistics_Request** **$030F_{hex}$**

Word 1	Code
Word 2	Parameter_Count
Word 3	Control_Code

Bit	15 0

Key:

Code: $030F_{hex}$ Command code of the service request

Parameter_Count: Number of subsequent words

0001_{hex} 1 parameter word

Control_Code: Functions of the service

0000_{hex} Reset
Resetting the statistical diagnostic information (see "Get_Diag_Info" service ($032B_{hex}$) on page 2-124).

Syntax: **Control_Statistics_Confirmation** **$830F_{hex}$**

Positive message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code: $830F_{hex}$ Message code of the service confirmation

Parameter_Count: Number of subsequent words

with a positive message:

0001_{hex} 1 parameter word

with a negative message:

0002_{hex} 2 parameter words

Result: Result of the service processing

0000_{hex} Indicates a positive message. The controller board executed the service successfully.

$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.

Add_Error_Info: Additional information on the error cause

2.8 Services for Defining Functions

Functions can be defined for the controller board. A function consists of a service sequence in any order (*Action_Object*) plus the associated data records (*Signal_Object*). These data records can be constants or may be loaded when the function is executed. Thus you may vary the function, depending on your system conditions, for example.

2.8.1 "Initiate_Load_Action_Object" Service

Task:

This service initiates the creation of service sequences. It prepares the controller board for the "Load_Action_Object" (0141_{hex}) service.

Syntax:

Initiate_Load_Action_Object_Request **0140_{hex}**

Word 1	Code
Word 2	Parameter_Count

Bit	15 0

Key:

Code: 0140_{hex} Command code of the service request

Parameter_Count: Number of subsequent words

0000_{hex} No parameter word

Syntax: **Initiate_Load_Action_Object_Confirmation** **8140_{hex}**

Positive message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code: 8140_{hex} Message code of the service confirmation

Parameter_Count: Number of subsequent words

with a positive message:

0001_{hex} 1 parameter word

with a negative message:

0002_{hex} 2 parameter words

Result: Result of the service processing

0000_{hex} Indicates a positive message. The controller board executed the service successfully.

$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.

Add_Error_Info: Additional information on the error cause

2.8.2 "Load_Action_Object" Service

Task:

This service can be used to create a service sequence (*Action_Object*). A service sequence is stored under an index (*Action_Index*). A service sequence consists of several services (*Actions*) that are indicated via their command code. It is possible to indicate a reference (*Action_Index*) to another service sequence instead of a command code.

The index of the service sequence (Action_Index) must be unique, i.e., it may only be assigned once.

Prerequisite:

The controller board must have been prepared for creating service sequences using the "Initiate_Load_Action_Object" (0140_{hex}) service.

Syntax: **Load_Action_Object_Request** **0141_{hex}**

Word 1	Code	
Word 2	Parameter_Count	
Word 3	Action_Index	
Word 4	Action_Count	Sequence (*Action_Object*)
...	1st action	
	2nd action	
	...	

Bit	15 0

Key:

Code:	0141_{hex} Command code of the service request
Parameter_Count:	Number of subsequent words $xxxx_{hex}$ = 2 + *Action_Count*
Action_Index:	Unique index of the service sequence. Every *Action_Index* may be assigned only once. The action indices 0000_{hex} ... 0063_{hex} are reserved for firmware-internal service sequences.
Action_Count:	Number of services (*Actions*) of this service sequence.

Action: Command code or reference (*Action_Index*) to a service sequence.

Set bit 15 to 1 if a reference to a service sequence is given. For example, enter the value 8567_{hex} for the *Action_Index* 0567_{hex}.

Syntax: **Load_Action_Object_Confirmation** **8141_{hex}**

Positive message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code: 8141_{hex} Message code of the service confirmation

Parameter_Count: Number of subsequent words

with a positive message:

0001_{hex} 1 parameter word

with a negative message:

0002_{hex} 2 parameter words

Result: Result of the service processing

0000_{hex} Indicates a positive message. The controller board executed the service successfully.

$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.

Add_Error_Info: Additional information on the error cause

2.8.3 "Terminate_Load_Action_Object" Service

Task: This service terminates the creation of service sequences.

Syntax: **Terminate_Load_Action_Object_Request** **0142_{hex}**

Word 1	Code
Word 2	Parameter_Count

Bit	15 0

Key:

Code: 0142_{hex} Command code of the service request

Parameter_Count: Number of subsequent words

0000_{hex} No parameter word

Syntax: **Terminate_Load_Action_Object_Confirmation** **8142_{hex}**

Positive message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code: 8142_{hex} Message code of the service confirmation

Parameter_Count: Number of subsequent words

with a positive message:

0001_{hex} 1 parameter word

with a negative message:

0002_{hex} 2 parameter words

Result: Result of the service processing

0000_{hex} Indicates a positive message. The controller board executed the service successfully.

$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.

Add_Error_Info: Additional information on the error cause

2.8.4 "Read_Action_Object" Service

Task: This service reads out the command codes of a service sequence (Action Object).

Syntax: **Read_Action_Object_Request** **0143_{hex}**

Word 1	Code
Word 2	Parameter_Count
Word 3	Action_Index

Bit	15 0

Key:

Code:	0143_{hex} Command code of the service request
Parameter_Count:	Number of subsequent words 0001_{hex} 1 parameter word
Action_Index:	Index of the service sequence the command codes of which you want to read

Syntax: **Read_Action_Object_Confirmation** **8143_{hex}**

Positive message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Action_Index
Word 5	Action_Count
Word 6	1st action
...	2nd action

Service sequence

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code: 8143_{hex} Message code of the service confirmation

Parameter_Count: Number of subsequent words

with a positive message:

$xxxx_{hex}$ = 3 + *Action_Count*

with a negative message:

0002_{hex} 2 parameter words

Result: Result of the service processing

0000_{hex} Indicates a positive message. The controller board executed the service successfully. The message contains the requested command code.

$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.

Add_Error_Info:	Additional information on the error cause
Action_Index:	Index of the service sequence (*Action_Object*) the command codes of which were read.
Action_Count:	Number of read command codes or references, i.e., the length of the service sequence.
Action:	Command code or reference (*Action_Index*) to a service sequence. If bit 15 is set to 1, it is a reference to a service sequence. For example, the value 8567_{hex} means that it is a reference to the service sequence with the *Action_Index* 0567_{hex}.

2.8.5 "Delete_Action_Object" Service

Task: This service deletes a service sequence (*Action_Object*).

Syntax: **Delete_Action_Object_Request** **0144_{hex}**

Word 1	Code
Word 2	Parameter_Count
Word 3	Action_Index

Bit	15 0

Key:

Code: 0144_{hex} Command code of the service request

Parameter_Count: Number of subsequent words

0001_{hex} 1 parameter word

Action_Index: Index of the service sequence to be deleted

$FFFF_{hex}$ Deletes all configured service sequences.

$xxxx_{hex}$ Deletes the service sequence with this index.

Syntax: **Delete_Action_Object_Confirmation** **8144_{hex}**

Positive message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code: 8144_{hex} Message code of the service confirmation

Parameter_Count: Number of subsequent words

with a positive message:

0001_{hex} 1 parameter word

with a negative message:

0002_{hex} 2 parameter words

Result: Result of the service processing

0000_{hex} Indicates a positive message. The controller board executed the service successfully.

$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.

Add_Error_Info: Additional information on the error cause

2.8.6 "Initiate_Load_Signal_Object" Service

Task:

This service initiates the configuration of parameter records (*Signal Objects*). It prepares the controller board for the "Load_Signal_Object" (0146_{hex}) service.

Syntax: **Initiate_Load_Signal_Object_Request** **0145_{hex}**

Word 1	Code
Word 2	Parameter_Count

Bit	15 0

Key:

Code: 0145_{hex} Command code of the service request

Parameter_Count: Number of subsequent words

0000_{hex} No parameter word

Syntax: **Initiate_Load_Signal_Object_Confirmation** **8145_{hex}**

Positive message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code: 8145_{hex} Message code of the service confirmation

Parameter_Count: Number of subsequent words

with a positive message:

0001_{hex} 1 parameter word

with a negative message:

0002_{hex} 2 parameter words

Result: Result of the service processing

0000_{hex} Indicates a positive message. The controller board executed the service successfully.

$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.

Add_Error_Info: Additional information on the error cause

2.8.7 "Load_Signal_Object" Service

Task:

This service can be used to configure parameter records (*Signal Objects*) for service sequences.

Prerequisite:

The controller board must have been prepared for configuration using the "Initiate_Load_Signal_Object" (0145_{hex}) service.

Syntax: **Load_Signal_Object_Request** **0146hex**

Word 1	Code				
Word 2	Parameter_Count				
Word 3	Signal_Index				
Word 4	Action_Index				
Word 5	Signal_Type				
Word 6	Bit_Number				
Word 7	Start_Register_Address				
Word 8	State_Register_Address				
Word 9	Result_Register_Address				
Word 10	Write_PB_Count (n_{Write})				Number
Word 11	W_Mode	Res.	Write_Parameter_Count		Write parameter block (action 1)
Word 12	Write_Parameter				
...	...				
...	...				...
	W_Mode	Res.	Write_Parameter_Count		Write parameter block (action *x*)
	Write_Parameter				
	...				
	Read_PB_Count (n_{Read})				Number
	R_Mode	Res.	Read_Parameter_Count		Read parameter block (action 1)
	Read_Parameter				
	...				
	...				
	R_Mode	Res.	x. Read_Parameter_Count		Read parameter block (action *x*)
	Read_Parameter				
	...				
Bit	15/14	13/12	11..... 0		

Key:

Code: 0146_{hex} Command code of the service request

Parameter_Count: Number of subsequent words

$xxxx_{hex}$ The value depends on the number and size of the write and read parameter blocks.

Signal_Index: Index of the parameter record.

The index of the parameter record must be unique, i.e., it may be assigned only once.

Value ranges:

0000_{hex} ... 0063_{hex}
Reserved for firmware-internal parameter records.

0064_{hex} ... 1000_{hex}
Parameter records with these indices are activated on controller board startup.

1001_{hex} ... $FFFF_{hex}$
Parameter records with these indices are activated by the application program.

Action_Index: Index of the service sequence to which the parameters of this parameter record are to be available.

Signal_Type: Indication for the behavior of the service sequences and parameter records.

The parameter is a 16-bit field in which every bit corresponds to a behavior. Set the corresponding bits to 1 for the behavior that you want to occur.

Bit assignments:

Bit 0 - 6 Reserved (always 0_{bin})

Bit 7 Resident flag.
Specifies whether the service sequence and parameter record are to be stored in the parameterization memory when calling the "Program_Resident_Actions" (0158_{hex}) service.

1_{bin} The service sequence and parameter record are stored.

0_{bin} The service sequence and parameter record are not stored.

Bit 8 - 15 Reserved (always 0_{bin})

Bit_Number: Location of the signal bit within the byte address
Value range: Bit 0 ... 7

Start_Register_Address: Location of the start bit (byte address in the MPM)

State_Register_Address: Location of the status bit (byte address in the MPM)

Result_Register_Address: Location of the result bit (byte address in the MPM)

Write_PB_Count: Number of subsequent write parameter blocks. This number must be identical to the number of services of the referenced service sequence.

W_Mode: Structure of the following write parameter block and the unit (words or bytes) of the *Write_Parameter_Count parameter*:

- 00_{bin} Compact data parameter mode
  Write_Parameter_Count determines in words the number of subsequent fixed parameters. The *k* th *Write_Parameter_Count* is identical to the *Parameter_Count* of the *k*th service of the referenced service sequence.
- 10_{bin} Compact address parameter mode
  Write_Parameter_Count determines in bytes the number of subsequent parameters, so that the firmware can read them from the MPM address in the following word onwards.
- 11_{bin} Split parameter mode
  The split parameter mode permits MPM addresses and fixed write parameters within the write parameter block. *Write_Parameter_Count* shows the sum of the number of subsequent MPM addresses and the write parameters (in bytes). Fixed write parameters are identified with the value $FFFF_{hex}$ and added to the end of the MPM address block.

res. Reserved (always 00_{bin})

Write_Parameter_Count: Number of subsequent write parameters in words or bytes, depending on the *W_Mode*

Write_Parameter: Data to be written (see the following examples).

Read_PB_Count: Number of subsequent read parameter blocks. This number must be identical to the number of services of the referenced service sequence.

R_Mode: Structure of the read parameter block and the unit (words/bytes) of the *Read_Parameter_Count* parameter; see *W_Mode*:

- 10_{bin} Compact address parameter mode
- 11_{bin} Split parameter mode

Read_Parameter_Count: Number of subsequent read parameters in words or bytes, depending on the *R_Mode*

Read_Parameter: Data to be read.

Example **Write parameter block in the compact data parameter mode:**

Mode	Res.	Write_Parameter_Count
1st fixed parameter		
2nd fixed parameter		
3rd fixed parameter		

Bit	15/14	13/12	11..... 0

Key:

Mode: 00_{bin} Compact data parameter mode

Res.: 00_{bin} Reserved for later extensions

Write_Parameter_Count: 003_{hex} Number of subsequent **words** (Value range: 001_{hex} ... $1F4_{hex}$).

Write_Parameter: Parameter record

Example **Write parameter block in the compact address parameter mode:**

Mode	Res.	Write_Parameter_Count
MPM address		

Bit	15/14	13/12	11..... 0

Key:

Mode: 10_{bin} Compact address parameter mode

Res.: 00_{bin} Reserved

Write_Parameter_Count: Number of **bytes** in the buffer specified by the following parameter; e.g., 006_{hex} for a buffer 6 bytes in length
(Value range: 001_{hex} ... $3E8_{hex}$).

MPM address: MPM start address of the byte buffer

Example **Write parameter blocks in the split parameter mode:**

Mode	Res.	Write_Parameter_Count	
1st MPM address			
2nd MPM address			
1st Dummy_Parameter			1st fixed byte
3rd MPM address			
2nd Dummy_Parameter			2nd fixed byte
3rd Dummy_Parameter			3rd fixed byte
00_{hex}		1st fixed byte	
00_{hex}		2nd fixed byte	
00_{hex}		3rd fixed byte	

Bit	15/14	13/12	11..... 0

Key:

Mode: 11_{bin} Split parameter mode

Res.: 00_{bin} Reserved for later extensions

Write_Parameter_Count: Sum of the number of fixed **bytes** and the number of variable **bytes** in the buffer specified by the following parameter.
(Value range: 001_{hex} ... $3E8_{hex}$)

MPM address: MPM addresses of variable bytes

Dummy_Parameter: $FFFF_{hex}$ Dummy parameter for the fixed bytes, which appear at the end of the MPM address block.

Syntax: **Load_Signal_Object_Confirmation** **8146_{hex}**

Positive message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code: 8146_{hex} Message code of the service confirmation

Parameter_Count: Number of subsequent words

with a positive message:

0001_{hex} 1 parameter word

with a negative message:

0002_{hex} 2 parameter words

Result: Result of the service processing

0000_{hex} Indicates a positive message. The controller board executed the service successfully.

$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.

Add_Error_Info: Additional information on the error cause

2.8.8 "Terminate_Load_Signal_Object" Service

Task:

The service completes the configuration of parameter records (*Signal Objects*).

Syntax:

Terminate_Load_Signal_Object_Request **0147_{hex}**

Word 1	Code
Word 2	Parameter_Count

Bit	15 0

Key:

Code: 0147_{hex} Command code of the service request

Parameter_Count: Number of subsequent words

0000_{hex} No parameter word

Syntax: **Terminate_Load_Signal_Object_Confirmation** **8147_{hex}**

Positive message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code: 8147_{hex} Message code of the service confirmation

Parameter_Count: Number of subsequent words

with a positive message:

0001_{hex} 1 parameter word

with a negative message:

0002_{hex} 2 parameter words

Result: Result of the service processing

0000_{hex} Indicates a positive message. The controller board executed the service successfully.

$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.

Add_Error_Info: Additional information on the error cause

2.8.9 "Read_Signal_Object" Service

Task:

The "Read_Signal_Object" service reads out the parameter record (*Signal Object*) of a service sequence.

Syntax: **Read_Signal_Object_Request** **0148_{hex}**

Word 1	Code
Word 2	Parameter_Count
Word 3	Signal_Index

Bit	15 0

Key:

Code: 0148_{hex} Command code of the service request

Parameter_Count: Number of subsequent words
0001_{hex} 1 parameter word

Signal_Index: Index of the parameter record (*Signal Object*) to be read

Syntax: **Read_Signal_Object_Confirmation** **8148_{hex}**

Positive message

Word 1	Code	
Word 2	Parameter_Count	
Word 3	Result	
Word 4	Signal_Index	
Word 5	Action_Index	
Word 6	Signal_Type	
Word 7	Bit_Number	
Word 8	Start_Register_Address	
Word 9	State_Register_Address	
Word 10	Result_Register_Address	
Word 11	Write_PB_Count (n_{Write})	Number
...	W_Mode / Res. / Write_Parameter_Count	Write parameter block (action 1)
	Write_Parameter	
	...	
	...	...

W_Mode	Res.	Write_Parameter_Count	Write parameter block (action *x*)
Write_Parameter			
...			
Read_PB_Count (n_{Read})			Number
R_Mode	Res.	Read_Parameter_Count	Read parameter block (action 1)
Read_Parameter			
...			
...			
R_Mode	Res.	x. Read_Parameter_Count	Read parameter block (action *x*)
Read_Parameter			
...			

Negative message

Word 1	Code		
Word 2	Parameter_Count		
Word 3	Result		
Word 4	Add_Error_Info		
Bit	15/14	13/12	11..... 0

Key:

Code: 8148_{hex} Message code of the service confirmation

Parameter_Count: Number of subsequent words

with a positive message:

$xxxx_{hex}$ The value depends on the number and size of the write and read parameter blocks.

with a negative message:

0002_{hex} 2 parameter words

Result: Result of the service processing

0000_{hex} Indicates a positive message. The controller board executed the service successfully.

$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.

Add_Error_Info: Additional information on the error cause

Signal_Index:	Index of the parameter record read
Action_Index:	Index of the service sequence for which the parameters of this parameter record are available.
Signal_Type:	Interrupt behavior The parameter is a 16-bit field in which every bit corresponds to an interrupt behavior. Set the corresponding bits to 1 for the interrupt behavior that you want to occur. Bit assignments: Bit 0 Interrupt on positive edge of the status bit Bit 1 Interrupt on negative edge of the status bit Bit 2 Interrupt on negative edge of the status bit and while the result bit is set (= 1_{bin}) Bit 3 Interrupt on negative edge of the status bit and while the result bit is not set (= 0_{bin}) Bit 4 Interrupt on positive edge of the start bit Bit 5 Interrupt on negative edge of the start bit Bits 6 - 15 Reserved
Bit_Number:	Location of the signal bit within the byte address Value range: 0_{bin} ... 7_{bin}.
Start_Register_Address:	Byte address in the MPM (offset) for the location of the start bit
State_Register_Address:	Byte address in the MPM (offset) for the location of the status bit
Result_Register_Address:	Byte address in the MPM (offset) for the location of the result bit
Write_PB_Count:	Number of subsequent write parameter blocks. This number must be identical to the number of services of the referenced service sequence.
W_Mode:	Structure of the subsequent write parameter block. The *W_Mode* parameter determines the unit (words or bytes) of the *Write_Parameter_Count* parameter:

	00_{bin}	Compact data parameter mode *Write_Parameter_Count* determines in words the number of subsequent fixed parameters. The *k*th *Write_Parameter_Count* is identical to the *Parameter_Count* of the *k*th service of the referenced service sequence.
	10_{bin}	Compact address parameter mode *Write_Parameter_Count* determines in bytes the number of subsequent parameters, so that the firmware can read them from the MPM address in the following word onwards.
	11_{bin}	Split parameter mode The split parameter mode permits MPM addresses and fixed write parameters within the write parameter block. *Write_Parameter_Count* shows the sum of the number of subsequent MPM addresses and the write parameters (in bytes). Fixed write parameters are identified with the value $FFFF_{hex}$ and added to the end of the MPM address block.
Res.:	Reserved (always 00_{bin})	
Write_Parameter_Count:	Number of subsequent write parameters in words or bytes, depending on the *mode*.	
Write_Parameter:	Data to be written	
Read_PB_Count:	Number of subsequent read parameter blocks. This number must be identical to the number of services of the referenced service sequence.	
R_Mode:	Structure of the read parameter block and the unit (words/bytes) of the *Read_Parameter_Count* parameter; see *W_Mode*:	
	10_{bin}	Compact address parameter mode
	11_{bin}	Split parameter mode
Read_Parameter_Count:	Number of subsequent read parameters in words or bytes, depending on the *mode*.	
Read_Parameter:	Data to be read.	

2.8.10 "Delete_Signal_Object" Service

Task: This service deletes a parameter record (*Signal Object*).

Syntax: **Delete_Signal_Object_Request** **0149_{hex}**

Word 1	Code
Word 2	Parameter_Count
Word 3	Signal_Index

Bit	15.....0

Key:

Code: 0149_{hex} Command code of the service request

Parameter_Count: Number of subsequent words

0001_{hex} 1 parameter word

Signal_Index: Index of the parameter record to be deleted.

$FFFF_{hex}$ Deletes all configured parameter records.

$xxxx_{hex}$ Deletes the parameter record with this index.

Syntax: **Delete_Signal_Object_Confirmation** **8149_{hex}**

Positive message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code: 8149_{hex} Message code of the service confirmation

Parameter_Count: Number of subsequent words

with a positive message:

0001_{hex} 1 parameter word

with a negative message:

0002_{hex} 2 parameter words

Result: Result of the service processing

0000_{hex} Indicates a positive message. The controller board executed the service successfully.

$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.

Add_Error_Info: Additional information on the error cause

2.8.11 "Initiate_Load_Event_Object" Service

Task:

This service initiates the configuration of event descriptions (*Event Object*). It prepares the controller board for the "Load_Event_Object" ($014B_{hex}$) service.

Syntax: **Initiate_Load_Event_Object_Request** **$014A_{hex}$**

Word 1	Code
Word 2	Parameter_Count

Bit	15 0

Key:

Code: $014A_{hex}$ Command code of the service request

Parameter_Count: Number of subsequent words

0000_{hex} No parameter word

Syntax: **Initiate_Load_Event_Object_Confirmation** **$814A_{hex}$**

Positive message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code: $814A_{hex}$ Message code of the service confirmation

Parameter_Count: Number of subsequent words

with a positive message:

0001_{hex} 1 parameter word

with a negative message:

0002_{hex} 2 parameter words

Result: Result of the service processing

0000_{hex} Indicates a positive message. The controller board executed the service successfully.

$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.

Add_Error_Info: Additional information on the error cause

2.8.12 "Load_Event_Object" Service

Task:

This service is used to configure event descriptions (*Event Objects*). Event descriptions provide the parameter records for service sequences.

Syntax:

Load_Event_Object_Request **$014B_{hex}$**

Word 1	Code
Word 2	Parameter_Count
Word 3	Event_Index
Word 4	Message_Code
Word 5	Notification_Byte_Address
Word 6	Acknowledge_Byte_Address
Word 7	Event_Signal_Bit
Word 8	Event_Signal_Type
Word 9	Event_Interface_Selector
Word 10	Interface_Parameter
...	

Bit	15 0

Key:

Code: $014B_{hex}$ Command code of the service request

Parameter_Count: Number of subsequent words

$xxxx_{hex}$ The number of parameter words depends on the selected interface (*Event_Interface_Selector* parameter) and the associated number of parameters (*Interface_Parameter* parameter block).

Event_Index: Index of the configured event

Message_Code: Message code of the event that occurred (access via *Event_Code*).

Notification_Byte_Address:

Byte address in the MPM (offset) for the location of the notification bit.

Acknowledge_Byte_Address:

Byte address in the MPM (offset) for the location of the acknowledge bit.

Event_Signal_Bit: Location of the signal bit within the byte address.

Value range: 0 ... 7.

Event_Signal_Type: Behavior of the event descriptions.

The parameter is a 8-bit field in which every bit corresponds to a behavior of the event descriptions (*Event Objects*). Set corresponding bits to 1 for the behavior that you want to occur.

Bit assignments:

Bits 0 ... 6 Reserved (always 0_{bin})

Bit 7 Resident flag; use this bit to indicate whether the event description is to be stored on the parameterization memory when the "Program_Resident_Actions" (0158_{hex}) service is called:

1_{bin} Event description is stored.

0_{bin} Event description is not stored.

Event_Interface_Selector:

Interface to which the parameters are sent:

- The event is to be indicated via the mailbox.
- The event is to be indicated via the SGI.
- The event is to trigger an action.
- The event is to trigger an action and be indicated via the XSGI at the same time.
- The event (configuration) is to be indicated via the SSGI.

Interface_Parameter: The *Event_Interface_Selector* parameter defines the structure of the *Interface_Parameter* parameter block:

1. *Event_Interface_Selector* = E_MXI_INTERFACE
   Indicating the event via the mailbox interface. In this case, no further parameters follow in the *Interface_Parameter* parameter block.
2. *Event_Interface_Selector* = E_XSGI_INTERFACE
   Indicating the events via the extended signal interface. A read parameter address list is defined in the *Interface_Parameter* parameter block. Please refer to the "Load_Signal_Object" (0146_{hex}) service for the structure and meaning.

Syntax:

Word 10	Number of Read_Copy_Address blocks
Word 11	Number of Read_Copy_Addresses (n)
Word 12	Read_Copy_Address 1
...	Read_Copy_Address 2
	...
	Read_Copy_Address n

Bit	15 0

3. *Event_Interface_Selector* = E_ACTION_INTERFACE
   Indicating the event by triggering an action in the action handler. The service sequence (*Action Object*) referenced by the *Action_Index* must be provided by the action handler. The *Action_Index*, several Write_Parameter blocks and Read_Parameter blocks are defined in the *Interface_Parameter* parameter block. For the structure and meaning, please refer to the "Load_Signal_Object" (0146_{hex}) service on page 2-135.

Syntax:

Word 10	Action_Index
Word 11	Number of Write_Parameter blocks (m)
Word 12	Number of Write_Parameters of the 1st action (n)
Word 13	Write_Parameter 1
...	Write_Parameter 2
	...
	Write_Parameter n
	...
	...

Number of Write_Parameters of the mth action (n)
Write_Parameter 1
Write_Parameter 2
...
Write_Parameter n
Number of Read_Parameter blocks (m action)
Number of Read_Parameters of the 1st action (n)
Read_Parameter 1
Read_Parameter 2
...
Read_Parameter n
...
...
Number of Read_Parameters of the mth action (n)
Read_Parameter 1
Read_Parameter 2
...
Read_Parameter n

Bit	15 0

4. *Event_Interface_Selector* =
   E_ACTION_INTERFACE_AND_E_XSGI_INTERFACE
   Indicating the event by triggering an action in the action handler. In addition, the corresponding read parameters of the event are indicated (copied) via the XSGI in the *Interface_Parameter* parameter block.

Syntax:

Word 10	Action_Index
Word 11	Number of Write_Parameter blocks (m action)
Word 12	Number of Write_Parameters of the 1st action (n)
Word 13	Write_Parameter 1
...	Write_Parameter 2
	...
	Write_Parameter n
	...

...
Number of Write_Parameters of the mth action (n)
Write_Parameter 1
Write_Parameter 2
...
Write_Parameter n
Number of Read_Parameter blocks (m action)
Number of Read_Parameters of the 1st action (n)
Read_Parameter 1
Read_Parameter 2
...
Read_Parameter n
...
...
Number of Read_Parameters blocks of the mth action (n)
Read_Parameter 1
Read_Parameter 2
...
Read_Parameter n
Number of Read_Copy blocks
Number of Read_Copy_Addresses (n)
Read_Copy_Address 1
Read_Copy_Address 2
...
Read_Copy_Address n

Bit	15 0

5. *Event_Interface_Selector* = E_SSGI_INTERFACE (8)
   Indicating the event via the standard signal interface is currently not supported.

2.8.13 "Terminate_Load_Event_Object" Service

Task: This service terminates the configuration of event descriptions *(Event Objects)*.

Syntax: **Terminate_Load_Event_Object_Request** **$014C_{hex}$**

Word 1	Code
Word 2	Parameter_Count

Bit	15 0

Key:

Code: $014C_{hex}$ Command code of the service request

Parameter_Count: Number of subsequent words

0000_{hex} No parameter word

Syntax: **Terminate_Load_Event_Object_Confirmation** **$814C_{hex}$**

Positive message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code: $814C_{hex}$ Message code of the service confirmation

Parameter_Count: Number of subsequent words

with a positive message:

0001_{hex} 1 parameter word

with a negative message:

0002_{hex} 2 parameter words

Result: Result of the service processing

0000_{hex} Indicates a positive message. The controller board executed the service successfully.

$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.

Add_Error_Info: Additional information on the error cause

2.8.14 "Read_Event_Object" Service

Task: This service reads the parameter record of an event description *(Event Object)*.

Syntax: **Read_Event_Object_Request** **$014D_{hex}$**

Word 1	Code
Word 2	Parameter_Count
Word 3	Event_Index

Bit	15 0

Key:

Code: $014D_{hex}$ Command code of the service request

Parameter_Count: Number of subsequent words
0001_{hex} 1 parameter word

Event_Index: Index of the event to be read

Syntax: **Read_Event_Object_Confirmation** **$814D_{hex}$**

Positive message

Word 1	Code
Word 2	Parameter_Count
Word 3	Event_Index
Word 4	Message (event or indication)_Code
Word 5	Notification_Byte_Address
Word 6	Acknowledge_Byte_Address
Word 7	Event_Signal_Bit
Word 8	Event_Signal_Type
Word 9	Event_Interface_Selector
Word 10	Interface_Parameter
...	

Negative message

Word 1	Result
Word 2	Add_Error_Info

Bit	15 0

Key:

Code: $814D_{hex}$ Message code of the service confirmation

Parameter_Count: Number of subsequent words

with a positive message:

$xxxx_{hex}$ The number of parameter words depends on the selected interface (*Event_Interface_Selector* parameter) and the associated number of parameters (*Interface_Parameter* parameter block).

with a negative message:

0002_{hex} 2 parameter words

Result: Result of the service processing

0000_{hex} Indicates a positive message. The controller board executed the service successfully.

$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the called service successfully. The *Result* parameter indicates why the service could not be executed.

Add_Error_Info: Additional information on the error cause

Event_Index: Index of the configured event

Message_Code: Message code of the event that occurred

Notification_Byte_Address:

Byte address in the MPM (offset) for the location of the notification bit

Acknowledge_Byte_Address:

Byte address in the MPM (offset) for the location of the acknowledge bit

Event_Signal_Bit: Location of the signal bit within the byte address

Event_Signal_Type: Behavior of the event descriptions

The parameter is a 8-bit field in which every bit corresponds to a behavior of the event descriptions (*Event Objects*). Set corresponding bits to 1 for the behavior that you want to occur. Bit assignments:

Bits 0 ... 6 Reserved (always 0_{bin})

Bit 7 Resident flag.
Use this bit to indicate whether the event description is to be stored on the parameterization memory when the "Program_Resident_Actions" (0158_{hex}) service is called:
1_{bin} Event description is stored.
0_{bin} Event description is not stored.

Event_Interface_Selector:

Interface to which the parameters are sent

Interface_Parameter: The *Event_Interface_Selector* parameter defines the structure of the *Interface_Parameter* parameter block. (Please refer to the "Load_Event_Object" service on page 2-162 for a description).

2.8.15 "Delete_Event_Object" Service

Task: This service deletes the parameters of an event description *(Event Object)*.

Syntax: **Delete_Event_Object_Request** **$014E_{hex}$**

Word 1	Code
Word 2	Parameter_Count
Word 3	Event_Index

Bit	15 0

Key:

Code: $014E_{hex}$ Command code of the service request

Parameter_Count: Number of subsequent words

0001_{hex} 1 parameter word

Event_Index: Index of the event description to be deleted.

$FFFF_{hex}$ Deletes all configured event descriptions.

$xxxx_{hex}$ Deletes the event description with this index.

Syntax: **Delete_Event_Object_Confirmation** **$814E_{hex}$**

Positive message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code: $814E_{hex}$ Message code of the service confirmation

Parameter_Count: Number of subsequent words

with a positive message:

0001_{hex} 1 parameter word

with a negative message:

0002_{hex} 2 parameter words

Result: Result of the service processing

0000_{hex} Indicates a positive message. The controller board executed the service successfully.

$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.

Add_Error_Info: Additional information on the error cause

2.9 Services for the Parameterization Memory

2.9.1 "Program_Resident_Actions" Service

Task:

This service transmits the service sequences (Action Object) and parameter records *(Signal Object)* stored in the controller board main memory to the parameterization memory. Service sequences and parameter records that are identified with bit 7 of the *Signal_Type* parameter as resident when the "Load_Signal_Object" (0146_{hex}) service is called are stored permanently in the memory.

Syntax:

Program_Resident_Actions_Request **0158_{hex}**

Word 1	Code
Word 2	Parameter_Count

Bit	15 0

Key:

Code: 0158_{hex} Command code of the service request

Parameter_Count: Number of subsequent words

0000_{hex} No parameter word

Syntax: **Program_Resident_Actions_Confirmation** **8158_{hex}**

Positive message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code: 8158_{hex} Message code of the service confirmation

Parameter_Count: Number of subsequent words

with a positive message:

0001_{hex} 1 parameter word

with a negative message:

0002_{hex} 2 parameter words

Result: Result of the service processing

0000_{hex} Indicates a positive message. The controller board executed the service successfully.

$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.

Add_Error_Info: Additional information on the error cause

2.9.2 "Clear_Parameterization_Memory" Service

Task: This service formats the parameterization memory.

Syntax: **Clear_Parameterization_Memory_Request** **0159_{hex}**

Word 1	Code
Word 2	Parameter_Count
Word 3	Action_Flag
Word 4	Area_Flag

Bit	15 0

Key:

Code: 0159_{hex} Command code of the service request

Parameter_Count: Number of subsequent words
0002_{hex} 2 parameter words

Action_Flag: Action to be executed.
0002_{hex} Format parameterization memory
$000B_{hex}$ Service started

Area_Flag: Area limitations
$000A_{hex}$ No limitations

Syntax: **Clear_Parameterization_Memory_Confirmation** **8159_{hex}**

Positive message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result

Negative message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Add_Error_Info

Bit	15 0

Key:

Code: 8159_{hex} Message code of the service confirmation

Parameter_Count: Number of subsequent words

with a positive message:

0001_{hex} 1 parameter word

with a negative message:

0002_{hex} 2 parameter words

Result: Result of the service processing

0000_{hex} Indicates a positive message. The controller board executed the service successfully.

$xxxx_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The *Result* parameter indicates why the service could not be executed.

Add_Error_Info: Additional information on the error cause

The state of the "Clear_Parameterization_Memory" service can be requested using the "Read_Value" 0351_{hex} service (see page 2-25) with the following settings.

Parameter_Count: 0002_{hex}

Variable_Count: 0001_{hex}

Variable_ID: 1711_{hex}

Please note that the service must be sent cyclically. The fact that read access interrupts the clear process increases the time taken to complete the clear process.

Syntax: **Read_Value_Confirmation** **8351_{hex}**

Positive message

Word 1	Code			
Word 2	Parameter_Count			$000A_{hex}$
Word 3	Result			0000_{hex}
Word 4	Variable_Count			0001_{hex}
Word 5	Variable_ID			1711_{hex}
Words 6 + 7	File_Format_Result			
	File_Format_Add_Err			
Words 8 + 9	Max_Duration	(bits 31 ... 16)		
	Max_Duration	(bits 15 ... 0)		
Words 10 +11	Current_Duration	(bits 31 ... 16)		
	Current_Duration	(bits 15 ... 0)		
Word 12	Ready flag	Reserved		
Bit	15 8	7 0		

Key:

Code:	8351_{hex}	Message code of the service confirmation
Parameter_Count:	Number of subsequent words	
Result:	Result of the service processing	
Variable_Count:	Number of read system parameters	
Variable_ID:	ID of the read system parameter	
File_Format_Result:	Error code, see Section "Error Codes" on page 3-3	
File_Format_Add_Err:	Additional error information	
Max_Duration:	Indicates the number of steps (ticks) required for this service.	
Current_Duration:	Indicates the number of steps (ticks) since the start of the process.	
Ready flag:	= 0:	Service still being processed
	<> 0:	Service processing complete

2.9.3 "File_Open" Service

Task:

This service opens a file. The permitted access right is determined with the *Access* and *Mode* parameters.

Syntax: **File_Open_Request** **$015B_{hex}$**

Word 1	Code		
Word 2	Parameter_Count		
Words 3 + 4	Access	(bits 31 ... 16)	
	Access	(bits 15 ... 0)	
Words 5 + 6	Mode	(bits 31 ... 16)	0200_{hex}
	Mode	(bits 15 ... 0)	0400_{hex}
Words 7 ... (13)	Name_Length	Name	File name
	...	...	
	...	Name	

Bit	15 8	7 0

Key:

Code: $015B_{hex}$ Command code of the service request

Parameter_Count: Number of subsequent words
$000x_{hex}$ = 4 + (*Name_Length* + 1)/2
Value range: 0005_{hex} ... $000B_{hex}$

Access: 32-bit flag used to set access rights:
$0000\ 0001_{hex}$ Open file (to read only)
$0000\ 0002_{hex}$ Open file (to write only)
$0000\ 0008_{hex}$ Open file (to append data)
$0000\ 0010_{hex}$ Create file (on demand)
$0000\ 0020_{hex}$ Open file exclusively
$0000\ 0040_{hex}$ Open file (to delete data)
$0000\ 0100_{hex}$ Binary file

Mode: 32-bit flag ($0200\ 0400_{hex}$)

Name_Length: Length of the file name (in bytes)

Name: Name of the file to be opened
(12 characters, maximum; MS DOS format)

Syntax: **File_Open_Confirmation** **$815B_{hex}$**

Positive message

Word 1	Code		
Word 2	Parameter_Count		
Word 3	Result		
Words 4 + 5	File_Handle	(bits 31 ... 16)	Access ID
	File_Handle	(bits 15 ... 0)	
Words 6 ... (12)	Name_Length	Name	File name
	...	...	
	...	Name	

Negative message

Word 1	Code		
Word 2	Parameter_Count		
Word 3	Result		
Words 4 + 5	File_Error	(bits 31 ... 16)	Add. error information
	File_Error	(bits 15 ... 0)	
Words 6 ... (12)	Name_Length	Name	File name
	...	...	
	...	Name	

Bit	15 8	7 0

Key:

Code: $815B_{hex}$ Message code of the service confirmation

Parameter_Count: Number of subsequent words

with positive and negative message:

$000x_{hex}$ = 3 + (*Name_Length* +1)/2

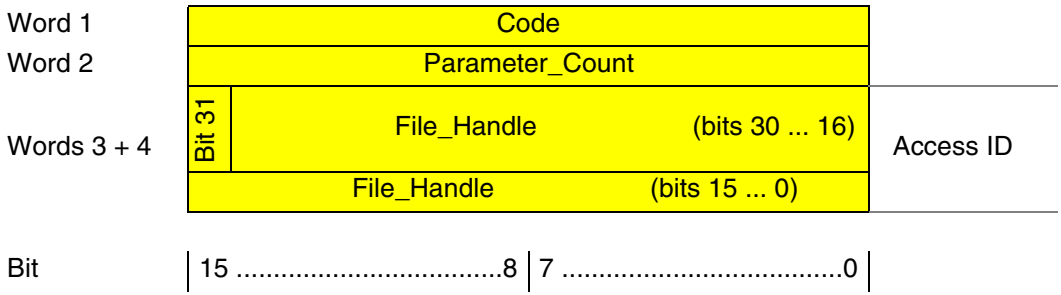
Value range: 0004_{hex} ... $000A_{hex}$

Result:	Result of the service processing 0000_{hex} Indicates a positive message. The controller board executed the service successfully and opened the file on the memory card. $09DE_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The file on the memory card could not be opened (see error codes, page 3-20).
File_Handle:	Access ID 32-bit access ID associated with the file name. This ID must be used for the remaining file operations.
Name_Length:	Length of the file name (in bytes)
Name:	Name of the file to be opened (12 characters, maximum; MS DOS format)
File_Error:	Additional information on the error cause (32-bit value)

2.9.4 "File_Close" Service

Task: This service closes an open file.

Syntax: **File_Close_Request** **$015C_{hex}$**



Key:

Code: $015C_{hex}$ Command code of the service request

Parameter_Count: Number of subsequent words
0002_{hex} 2 parameter words

File_Handle: Access ID
The access ID (32-bit value) is associated with the file name. This is received from the "File_Open_Confirmation" ($815B_{hex}$) service when a file is opened successfully.

Bit 31 (No-Wait-Bit)
= 1: The controller immediately generates a confirmation, the "File_Close" service is processed in the background.
= 0: The confirmation is only generated by the controller once the "File_Close" service has been completed.

Syntax: **File_Close_Confirmation** **815C$_{hex}$**

Positive message

Word 1	Code			
Word 2	Parameter_Count			
Word 3	Result			0000$_{hex}$
Words 4 + 5	Bit 31	File_Handle	(bits 30 ... 16)	Access ID
	File_Handle		(bits 15 ... 0)	

Negative message

Word 1	Code			
Word 2	Parameter_Count			
Word 3	Result			09DD$_{hex}$
Words 4 + 5	Bit 31	File_Handle	(bits 30 ... 16)	Access ID
	File_Handle		(bits 15 ... 0)	
Words 6 + 7	File_Error		(bits 31 ... 16)	Add. error information
	File_Error		(bits 15 ... 0)	

Bit	15 8	7 0

Key:

Code: 815C$_{hex}$ Message code of the service confirmation

Parameter_Count: Number of subsequent words
with a positive message:
0003$_{hex}$ 3 parameter words
with a negative message:
0005$_{hex}$ 5 parameter words

Result: Result of the service processing
0000$_{hex}$ Indicates a positive message. The file on the memory card was closed.
09DD$_{hex}$ Indicates a negative message. The file on the memory card could not be closed. (See error codes, page 3-20.)

File_Handle:	Access ID The parameter contains the value that was transferred with the service request.
File_Error:	Additional information on the error cause (32-bit value)

The state of the "File_Close" service can be requested using the "Read_Value" 0351_{hex} service (see page 2-25) with the following settings.

Parameter_Count:	0002_{hex}
Variable_Count:	0001_{hex}
Variable_ID:	1712_{hex}

Please note that the service must be sent cyclically. The fact that read access interrupts the close process increases the time taken to complete the close process.

Syntax: **Read_Value_Confirmation** **8351_{hex}**

Positive message

Word 1	Code		
Word 2	Parameter_Count		$000A_{hex}$
Word 3	Result		0000_{hex}
Word 4	Variable_Count		0001_{hex}
Word 5	Variable_ID		1712_{hex}
Words 6 + 7	File_Close_Result		
	File_Close_Add_Err		
Words 8 + 9	Max_Duration	(bits 31 ... 16)	
	Max_Duration	(bits 15 ... 0)	
Words 10 +11	Current_Duration	(bits 31 ... 16)	
	Current_Duration	(bits 15 ... 0)	
Word 12	Ready flag	Reserved	
Bit	15 8	7 0	

Key:

Code:	8351_{hex}	Message code of the service confirmation
Parameter_Count:	Number of subsequent words	
Result:	Result of the service processing	
Variable_Count:	Number of read system parameters	
Variable_ID:	ID of the read system parameter	

File_Close_Result:	Error code, see Section "Error Codes" on page 3-3
File_Close_Add_Err:	Additional error information
Max_Duration:	Indicates the number of steps (ticks) required for this service
Current_Duration:	Indicates the number of steps (ticks) since the start of the process
Ready flag:	= 0: Service still being processed <> 0: Service processing complete

2.9.5 "File_Remove" Service

Task: This service deletes the indicated file.

Syntax: **File_Remove_Request** **$015D_{hex}$**

Word 1	Code		
Word 2	Parameter_Count		
Words 3 ... (9)	Name_Length	Name	File name
	...	...	
	...	Name	

Bit	15 8	7 0

Key:

Code: $015D_{hex}$ Command code of the service request

Parameter_Count: Number of subsequent words
$000x_{hex}$ = (*Name_Length* + 1)/2
Value range: 0001_{hex} ... 0007_{hex}

Name_Length: Length of the file name (in bytes)

Name: Name of the file to be deleted
(12 characters, maximum; MS DOS format)

Syntax: **File_Remove_Confirmation** **$815D_{hex}$**

Positive message

Word 1	Code		
Word 2	Parameter_Count		
Word 3	Result		0000_{hex}
Words 4 ... (10)	Name_Length	Name	File name
	...	...	
	...	Name	

Negative message

Word 1	Code		
Word 2	Parameter_Count		
Word 3	Result		$09DF_{hex}$
Words 4 ... (10)	Name_Length	Name	File name
	...	...	
	...	Name	
Words (11) + (12)	File_Error	(bits 31 ... 16)	Add. error information
	File_Error	(bits 15 ... 0)	

Bit	15 8	7 0

Key:

Code: $815D_{hex}$ Message code of the service confirmation

Parameter_Count: Number of subsequent words

with a positive message:

$000x_{hex}$ = 1 + (*Name_Length* + 1)/2
Value range: 0001_{hex} ... 0008_{hex}

with a negative message:

$000x_{hex}$ = 3 + (*Name_Length* + 1)/2
Value range: 0003_{hex} ... $000A_{hex}$

Result: Result of the service processing

0000_{hex} Indicates a positive message.
The file on the memory card was deleted.

$09DF_{hex}$ Indicates a negative message.
The file on the memory card could not be deleted (see error codes, page 3-21).

Name_Length:	Length of the file name (in bytes)
Name:	Name of the file that was deleted (12 characters, maximum; MS DOS format)
File_Error:	Additional information on the error cause (32-bit value)

2.9.6 "File_Write" Service

Task: This service writes data into the file via the *File_Handle* access ID that was opened previously with the "File_Open" service for writing.

Syntax: **File_Write_Request** **$015E_{hex}$**

Word 1	Code		
Word 2	Parameter_Count		
Words 3 + 4	File_Handle	(bits 31 ... 16)	Access ID
	File_Handle	(bits 15 ... 0)	
Word 5	Data_Length		Data to be written
Words 6 ...	Data	Data	
	...	...	

Bit	15 8	7 0

Key:

Code:	$015E_{hex}$ Command code of the service request
Parameter_Count:	Number of subsequent words $xxxx_{hex}$ = 3 + *Data_Length*/2
File_Handle:	Access ID The access ID (32-bit value) is associated with the file name. This is received from the "File_Open_Confirmation" ($815B_{hex}$) service when a file is opened successfully.
Data_Length:	Length of the *Data* parameter (in bytes), number of bytes to be written
Data:	Data to be written

Syntax: **File_Write_Confirmation** **$815E_{hex}$**

Positive message

Word 1	Code		
Word 2	Parameter_Count		
Word 3	Result		
Words 4 + 5	File_Handle	(bits 31 ... 16)	Access ID
	File_Handle	(bits 15 ... 0)	
Word 6	Data_Length		

Negative message

Word 1	Code		
Word 2	Parameter_Count		
Word 3	Result		
Words 4 + 5	File_Handle	(bits 31 ... 16)	Access ID
	File_Handle	(bits 15 ... 0)	
Word 6	Data_Length		
Words 7 + 8	File_Error	(bits 31 ... 16)	Add. error information
	File_Error	(bits 15 ... 0)	

Bit	15 0

Key:

Code: $815E_{hex}$ Message code of the service confirmation

Parameter_Count: Number of subsequent words

with a positive message:

0004_{hex} 4 parameter words

with a negative message:

0006_{hex} 6 parameter words

Result: Result of the service processing

0000_{hex} Indicates a positive message. The controller board executed the service successfully and wrote data to the file.

$09DB_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. It was not possible to write to the file (see error codes, page 3-20).

File_Handle:	Access ID The parameter contains the value that was transferred with the service request.
Data_Length:	Length of the data The parameter contains the value that was transferred with the service request.
File_Error:	Additional information on the error cause (32-bit value)

2.9.7 "File_Seek" Service

Task:

This service sets the file pointer to the address specified with the parameters *Whence* and *Offset*.

Syntax: **File_Seek_Request** **$015F_{hex}$**

Word 1	Code		
Word 2	Parameter_Count		
Words 3 + 4	File_Handle	(bits 31 ... 16)	Access ID
	File_Handle	(bits 15 ... 0)	
Words 5 + 6	Offset	(bits 31 ... 16)	Distance from reference address
	Offset	(bits 15 ... 0)	
Words 7 + 8	Whence	(bits 31 ... 16)	Reference address
	Whence	(bits 15 ... 0)	
Bit	15 0		

Key:

Code: $015F_{hex}$ Command code of the service request

Parameter_Count: Number of subsequent words
0006_{hex} 6 parameter words

File_Handle: Access ID
The access ID (32-bit value) is associated with the file name. This is received from the "File_Open_Confirmation" ($815B_{hex}$) service when a file is opened successfully.

Offset: Offset value
Offset value (32-bit value) is the distance of the new address of the file pointer from the output address specified by the *Whence* parameter.

Whence: Offset reference
Offset reference is the 32-bit output address from which the new address (*Offset parameter*) of the file pointer is calculated.

SEEK_END	File end	(2_{dec})
SEEK_CUR	Current address	(1_{dec})
SEEK_SET	File beginning	(0_{dec})

Syntax: **File_Seek_Confirmation** **$815F_{hex}$**

Positive message

Word 1	Code		
Word 2	Parameter_Count		
Word 3	Result		
Words 4 + 5	File_Handle	(bits 31 ... 16)	Access ID
	File_Handle	(bits 15 ... 0)	
Word 6	Count_Of_Possible_Seek	(bits 31 ... 16)	Like *Offset*
Word 7	Count_Of_Possible_Seek	(bits 15 ... 0)	

Negative message

Word 1	Code		
Word 2	Parameter_Count		
Word 3	Result		
Words 4 + 5	File_Handle	(bits 31 ... 16)	Access ID
	File_Handle	(bits 15 ... 0)	
Words 6 + 7	Count_Of_Possible_Seek	(bits 31 ... 16)	Offset seek steps carried out
	Count_Of_Possible_Seek	(bits 15 ... 0)	
Words 8 + 9	File_Error	(bits 31 ... 16)	Add. Error information
	File_Error	(bits 15 ... 0)	

Bit	15 0

Key:

Code: $815F_{hex}$ Message code of the service confirmation

Parameter_Count: Number of subsequent words

with a positive message:
0005_{hex} 5 parameter words

with a negative message:
0007_{hex} 7 parameter words

Result:	Result of the service processing 0000_{hex} Indicates a positive message. The controller board executed the service successfully and repositioned the pointer. $09DC_{hex}$ Indicates a negative message. The controller board could not execute the service successfully. The pointer could not be repositioned (see error codes, page 3-20).
File_Handle:	Access ID The parameter contains the value that was transferred with the service request.
Count_Of_Possible_Seek:	Number of offset steps that were carried out. The value of the *Count_Of_Possible_Seek* parameter corresponds to the value of the *Offset* parameter in the service request.
File_Error:	Additional information on the error cause (32-bit value)

2.9.8 "File_Read" Service

Task: This service reads data from a file. The file is called via the *File_Handle* access ID.

Syntax: **File_Read_Request** **0160_{hex}**

Word 1	Code		
Word 2	Parameter_Count		
Words 3 + 4	File_Handle	(bits 31 ... 16)	Access ID
	File_Handle	(bits 15 ... 0)	
Word 5	Data_Length		

Bit	15 0

Key:

Code: 0160_{hex} Command code of the service request

Parameter_Count: Number of subsequent words
0003_{hex} 3 parameter words

File_Handle: Access ID
The access ID (32-bit value) is associated with the file name. This is received from the "File_Open_Confirmation" ($815B_{hex}$) service when a file is opened successfully.

Data_Length: Number of data bytes to be read from the file

Syntax: **File_Read_Confirmation** **8160_{hex}**

Positive message

Word 1	Code		
Word 2	Parameter_Count		
Word 3	Result		
Words 4 + 5	File_Handle	(bits 31 ... 16)	Access ID
	File_Handle	(bits 15 ... 0)	
Word 6	Data_Length		Data read
Words 7 ...	Data	Data	
	...	...	

Negative message

Word 1	Code		
Word 2	Parameter_Count		
Word 3	Result		
Words 4 + 5	File_Handle	(bits 31 ... 16)	Access ID
	File_Handle	(bits 15 ... 0)	
Words 6 + 7	File_Error	(bits 31 ... 16)	Add. error information
	File_Error	(bits 15 ... 0)	

Bit	15 0

Key:

Code: 8160_{hex} Message code of the service confirmation

Parameter_Count: Number of subsequent words

with a positive message:

$xxxx_{hex}$ = 4 + *Data_Length*/2

with a negative message:

0005_{hex} 5 parameter words

Result: Result of the service processing

0000_{hex} Indicates a positive message. The controller board executed the service successfully and read the requested data.

	$09DA_{hex}$ Indicates a negative message. The controller board could not execute the service successfully and the data was not read (see error codes, page 3-20).
File_Handle:	Access ID The parameter contains the value that was transferred with the service request.
Data_Length:	Number of bytes that were read
Data:	Read data (byte by byte)
File_Error:	Additional information on the error cause (32-bit value)

2.9.9 "File_Remove_II" Service

Task:

This service deletes the indicated file.

This service differs from the "File_Remove" $015D_{hex}$ service because confirmation is received immediately and not after the service has been processed.

Syntax: **File_Remove_II_Request** **0165_{hex}**

Word 1	Code		
Word 2	Parameter_Count		
	Name_Length	Name	
Words 3 ... (9)	...	...	File name
	...	Name	

Bit	15 8	7 0

Key:

Code: 0165_{hex} Command code of the service request

Parameter_Count: Number of subsequent words
$000x_{hex}$ = (*Name_Length* + 1)/2
Value range: 0001_{hex} ... 0007_{hex}

Name_Length: Length of the file name (in bytes)

Name: Name of the file to be deleted
(12 characters, maximum; MS DOS format)

The state of the "File_Remove_II" service can be requested using the "Read_Value" 0351_{hex} service (see page 2-25) with the following settings.

Parameter_Count: 0002_{hex}

Variable_Count: 0001_{hex}

Variable_ID: 1713_{hex}

Please note that the service must be sent cyclically. The fact that read access interrupts the remove process increases the time taken to complete the remove process.

Syntax: **Read_Value_Confirmation** **8351_{hex}**

Positive message

Word 1	Code		
Word 2	Parameter_Count		$000A_{hex}$
Word 3	Result		0000_{hex}
Word 4	Variable_Count		0001_{hex}
Word 5	Variable_ID		1713_{hex}
Words 6 + 7	File_Remove_Result		
	File_Remove_Add_Err		
Words 8 + 9	Max_Duration	(bits 31 ... 16)	
	Max_Duration	(bits 15 ... 0)	
Words 10 +11	Current_Duration	(bits 31 ... 16)	
	Current_Duration	(bits 15 ... 0)	
Word 12	Ready flag	Reserved	
Bit	15 8	7 0	

Key:

Code:	8351_{hex}	Message code of the service confirmation
Parameter_Count:	Number of subsequent words	
Result:	Result of the service processing	
Variable_Count:	Number of read system parameters	
Variable_ID:	ID of the read system parameter	
File_Remove_Result:	Error code, see Section "Error Codes" on page 3-3	
File_Remove_Add_Err:	Additional error information	
Max_Duration:	Indicates the number of steps (ticks) required for this service	
Current_Duration:	Indicates the number of steps (ticks) since the start of the process	
Ready flag:	= 0:	Service still being processed
	<> 0:	Service processing complete

2.9.10 Get_Card_Information Service

Task:

This service reads information from the parameterization memory.

You can use the content of the *Directory_Flag* to indicate whether the directory or the technical data of the parameterization memory should be read.

The structure of the confirmation is determined by the setting of the *Directory_Flag* parameter.

2.9.10.1 Reading the Directory

Syntax: **Get_Card_Information_Request** **0166_{hex}**

Word 1	Code
Word 2	Parameter_Count
Word 3	Directory_Flag
Word 4	Start_Flag

Bit	15 0

Key:

Code:	0166_{hex}	Command code of the service request
Parameter_Count:	Number of subsequent words	
	0002_{hex}	2 parameter words
Directory_Flag:	0001_{hex}	Reads the parameterization memory directory
Start_Flag:	= 0:	Reads the directory starting with the first entry
	= 1:	Reads the directory starting with the last entry read plus 1

If the remainder of the directory is to be read with the next request, the Start_Flag must be set to 1. Requirement: The More_Follows_Flag must have been set in the confirmation.

Syntax: **Get_Card_Information_Confirmation** **8166_{hex}**

Positive message

Word 1	Code
Word 2	Parameter_Count
Word 3	Result
Word 4	Directory_Flag
Word 5	More_Follows
Word 6	File_Counts
Word 7	Directory entry 1
Word 8	Directory entry 2
Word 9	...
Word 10	Directory entry n - 1 (n = File_Counts)
Word 11	Directory entry n (n = File_Counts)

Bit	15 0

Key:

Code:	8166_{hex}	Message code of the service confirmation
Parameter_Count:	Number of subsequent words	
Result:	Result of the service processing	
Directory_Flag:	0001_{hex}	Parameterization memory directory
More_Follows:	0001_{hex}	The directory is still being read.
File_Counts:	$xxxx_{hex}$	Number of directory entries
Directory entry x:	Entries in the directory according to the following structure:	

Entries in the Directory

Filename	Data [00]	Data [01]
Filename	Data [02]	Data [03]
Filename	Data [04]	Data [05]
Filename	Data [06]	Data [07]
Filename	Data [08]	Data [09]
Filename	Data [10]	Data [11]

Filename	Data [12]	Data [13]
Filesize	File_Size	(bits 31 ... 16)
Filesize	File_Size	(bits 15 ... 0)

Bit	15 0

Key:

Filename:	File name
Filesize:	File size
Data [x]:	ASCII-coded data

2.9.10.2 Reading the Parameterization Memory Description

This is where the technical data of the parameterization memory, which is available once the parameterization memory has been formatted, is read. You can, for example, read the maximum available memory from the card once it has been formatted.

Syntax: **Get_Card_Information_Request** **0166_{hex}**

Word 1	Code
Word 2	Parameter_Count
Word 3	Directory_Flag
Word 4	Reserved

Bit	15 0

Key:

Code:	0166_{hex}	Command code of the service request
Parameter_Count:	Number of subsequent words	
	0002_{hex}	2 parameter words
Directory_Flag:	0002_{hex}	Read the maximum data of the parameterization memory
Start_Flag:	= 0:	Not relevant (always = 0)

Syntax: **Get_Card_Information_Confirmation** **8166_{hex}**

Positive message

Word 1	Code	
Word 2	Parameter_Count	
Word 3	Result	
Word 4	Directory_Flag	
Word 5	Max_Number_Of_Files	
Word 6	Max_File_Size	(bits 31 ... 16)
Word 7	Max_File_Size	(bits 15 ... 0)
Word 8	Volume_Size	(bits 31 ... 16)
Word 9	Volume_Size	(bits 15 ... 0)

Bit	15 0

Key:

Code:	8166_{hex}	Message code of the service confirmation
Parameter_Count:	Number of subsequent words	
Result:	Result of the service processing	
Directory_Flag:	0002_{hex}	The data read indicates the maximum data of the parameterization memory used.
Max_Number_Of_Files:	Maximum number of files that can be saved	
Max_File_Size:	Maximum file size permitted by the type of parameterization memory used	
Volume_Size:	Maximum available memory	

The number of files that the parameterization memory can save once it has been formatted is determined by the file system. Once this number has been reached, the parameterization memory must be reformatted in order to save additional files.

Example:
Using the Max_Number_Of_Files parameter, you read the maximum number of files that can be saved on the parameterization memory, e.g., 60. Once 45 files have been saved, 15 more can be saved before you will need to reformat the memory in order to save more files. Even if, for example, 8 files are deleted using a "File_Remove" service, only 15 files can be saved. Only once the memory has been reformatted can another 60 files be saved.

2.10 Automatic Indications of the Controller Board

With the following indications the controller board automatically provides information to the application program.

First, the "Set_Indication" (0152_{hex}) service (see page 2-15) must be used to enable error code transmission to the application program using the mailbox interface of the controller board .

2.10.1 "Fault" Indication

Meaning:

This indication transmits the controller board error codes to the application program.

Syntax: **Fault_Indication** **4341_{hex}**

Word 1	Code
Word 2	Parameter_Count
Word 3	Entry_Count
Word 4	Error_Code
Word 5	Add_Error_Info

Bit	15 0

Key:

Code: 4341_{hex} Code of the indication

Parameter_Count: Number of subsequent words
0003_{hex} 3 parameter words

Entry_Count: Number of entries consisting of 2 words:
0001_{hex} The "fault" indication always transmits only one error code at a time.

Error_Code: Error code that explains the error in more detail, e.g.,:
$0B8D_{hex}$ A synchronous interrupt occurred in the *Asynchronous* operating mode.
$0B8F_{hex}$ Overflow of the IPMS input FIFO.
$0BD2_{hex}$ The bus warning time elapsed.

Add_Error_Info: Additional information on the error cause

2.10.2 "Lower_API_Fault" Indication

Meaning:

This indication transmits the controller board error codes to the application program.

The indication appeared along with the error message 0914_{hex}.

Syntax:

Lower_API_Fault_Indication **$4B58_{hex}$**

Word 1	Code
Word 2	Parameter_Count
Word 3	Entry_Count
Word 4	...
Word 5	Error_Code

Bit	15 0

Key:

Code: $4B58_{hex}$ Code of the indication

Parameter_Count: Number of subsequent words
0003_{hex} 3 parameter words

Entry_Count: Number of entries consisting of 2 words:
0001_{hex} The "Lower_API_Fault" indication always transmits only one error code at a time.

Error_Code: Error code that explains the error in more detail.

2.10.3 "Device_Fail" Indication

Meaning:

This indication transmits error messages for a maximum of ten INTERBUS devices from the controller board to the application program.

Syntax:

Device_Fail_Indication **5340_{hex}**

Word 1	Code	
Word 2	Parameter_Count	
Word 3	Entry_Count	
Word 4	Error_Code	1st error
	Device_No	
...	...	
Word (21)	Error_Code	(10)th Error
	Device_No	

Bit	15 0

Key:

Code: 5340_{hex} Code of the indication

Parameter_Count: Number of subsequent words
$00xx_{hex}$ = 1 + 2 × *Entry_Count*
Value range: 0001_{hex} ... 0017_{hex}

Entry_Count: Number of transmitted error messages, each consisting of two words.
A maximum of 10 error codes and the corresponding device numbers are transmitted.

Error_Code: Error code.
Section 3 describes the meaning of the error codes.

Device_No: INTERBUS device on which the error occurred. The device is identified by its device number.

2.10.4 "Bus_Error" Indication

Meaning: Data traffic on the bus was stopped.

Cause: Errors or disturbances during data transmission.

Remedy: Read the error causes using the "Get_Error_Info" service (0316_{hex}, see page 2-113).

Syntax: **Bus_Error_Indication** **6342_{hex}**

Word 1	Code	
Word 2	Parameter_Count	
Bit	15 0	

Key:

Code: 6342_{hex} Code of the indication

Parameter_Count: Number of subsequent words

0000_{hex} No parameter word

Section 3

This section provides information about

– The meaning of error codes
– Error causes
– Notes for error remedies and error removal

3 Error Codes

3.1 Error Codes for User Errors (USER FAIL)

0902_{hex} (CTRL FAIL)

Meaning: The controller board could not process the service called last.

Cause: Hardware or firmware error on the controller board.

Remedy: Replace the controller board.

0903_{hex} (CTRL FAIL)

Meaning: Insufficient memory on the controller board.

Cause: The main memory may, for example, be too fragmented.

Remedy: Use the "Reset_Controller_Board" service (0956_{hex}) to execute a warm start of the controller board, and try again.

0904_{hex} (USER FAIL)

Meaning: You specified inconsistent parameters.

Remedy: Check the parameters of the service called last for inconsistencies.

0905_{hex} (USER FAIL)

Meaning: You specified invalid parameters.

Remedy: Check the parameters of the service called last.

0906_{hex} (USER FAIL)

Meaning: Access to this object is not supported.

Remedy: Check the last access.

0907_{hex} (USER FAIL)

Meaning: You tried to access an object that does not exist.

Remedy: Check the last access.

0908_{hex} (USER FAIL)

Meaning: The firmware cannot process two services at the same time.

Cause: You called a service, which causes a processing conflict with another service called before.

Remedy: Wait for the service called previously to be completed, and then try again.

Add_Error_Info: Code of the service called previously.

0909_{hex} (USER FAIL)

Meaning: The confirmation, which follows a service exceeds the maximum permissible mailbox size.

Remedy: Check the command. If this does not solve the problem, please contact Phoenix Contact.

Add_Error_Info: Size of this confirmation in bytes.

$090A_{hex}$ (USER FAIL)

Meaning: The *Parameter_Count* parameter does not agree with the number of subsequent words.

Add_Error_Info: Incorrect *Parameter_Count* parameter value.

$090B_{hex}$ (USER FAIL)

Meaning: The controller board cannot enable the specified service.

Cause: You entered an incorrect password when enabling this service.

Remedy: Use the correct password or a service, which is not disabled.

0910_{hex} (USER FAIL)

Meaning: The controller board cannot process the called service.

Cause: The called service is disabled.

Remedy:
- Enable the service before calling it.
- Use a service, which has not been disabled.

Add_Error_Info: Code of the disabled service.

0911_{hex} (USER FAIL)

Meaning: The controller board cannot enable the specified service.

Cause: The service has already been enabled.

Remedy: Call the service.

Add_Error_Info: Code of the already enabled service.

0912_{hex} (USER FAIL)

Meaning: The controller board cannot process the called service.

Cause: The service is protected. It can only be enabled with a password.

Remedy:
- Enable the service before calling it.
- Use a service, which is not protected.

0913_{hex} (USER FAIL)

Meaning: The controller board cannot process the called service.

Cause: The service is not supported for this controller board.

Remedy: Use a service supported by this controller board.

0914_{hex}

Meaning: The transmit buffer is full. Thus, messages can no longer be transmitted to the host.

Cause: There are too many messages in the transmit buffer, as the host does not accept the pending messages or a lot of messages have been generated in a very short time. For example, this may be caused by a loose contact in the I/O voltage for the INTERBUS devices.

Remedy: Ensure that the host accepts the pending messages.

0915_{hex}

Meaning: The receive buffer is full. The host can no longer transmit messages.

Cause: There are too many messages in the receive buffer, as the controller board does not accept the pending messages or a lot of messages have been generated by the host in a very short time.

Remedy: Ensure that the controller board accepts the pending messages.

0918_{hex} (USER FAIL)

Meaning: You called an unknown service code.

Remedy: Check the last call.

Add_Error_Info: Unknown service code.

0919_{hex}

Meaning: The service exceeds the maximum permissible mailbox size.

Cause: You called a service, which is longer than 1024 bytes.

Remedy: Use only services, which are not longer than 1024 bytes including all parameters. To transmit a larger data record, you can call some services more than once to transmit the data record in multiple steps.

091A$_{hex}$ (USER FAIL)

Meaning: You specified an unknown value for the *Action_Index* parameter.

Add_Error_Info: *Action_Index* parameter.

091B$_{hex}$ (USER FAIL)

Meaning: You specified a value for the *Action_Index* parameter, which is already used elsewhere.

Add_Error_Info: *Action_Index* parameter.

091C$_{hex}$

Meaning: An incorrect request occurred.

Cause: Unexpected overlap in the action handler.

Remedy: Please contact Phoenix Contact.

091D$_{hex}$

Cause: Unexpected overlap in the action handler.

Remedy: Please contact Phoenix Contact.

091E$_{hex}$

Cause: Unexpected overlap in the action handler

Remedy: Please contact Phoenix Contact.

091F$_{hex}$

Meaning: An internal conflict occurred due to an unknown action index.

Cause: During the configuration of the signal handler, the selected action index or desired action object could not be found.

Remedy: Create the desired action index or the desired action object before configuring the signal handler.

0920_{hex} (USER FAIL)

Cause: You specified a value for the *Signal_Index* parameter, which is already used elsewhere.

Add_Error_Info: *Signal_Index* parameter.

0921_{hex}

Meaning: State conflict in the signal handler

Cause:
1. Insufficient memory available on the hardware.
   Too many signals initiated by the user.
2. A program is blocking the PCP connection. The PCP connection can no longer be used.

Remedy:
1. Ensure that there are not too many signals initiated.
2. Execute a warm start of the controller board using the "Reset_Controller_Board" service (0956_{hex}).

0922_{hex}

Meaning: State conflict in the action handler.

Cause:
- Insufficient memory available on the hardware.
- Too many actions initiated by the user.

Remedy: Ensure that there are not too many actions initiated.

0924_{hex}

Meaning: State conflict in the event handler

Cause:
- Insufficient memory available on the hardware.
- Too many events initiated by the user.

Remedy: Ensure that there are not too many events initiated.

0925_{hex} (USER FAIL)

Cause: You specified an undefined value for the *Event_Index* parameter.

Add_Error_Info: *Event_Index* parameter.

0926_{hex} (USER FAIL)

Cause: You specified a value for the *Event_Index* parameter, which is already used elsewhere.

Add_Error_Info: *Event_Index* parameter.

0928_{hex} (USER FAIL)

Meaning: The controller board cannot process the called service.

Cause: You called an exclusive service without being authorized to do so.

Remedy:
- Using the "Change_Exclusive_Rights" service ($014F_{hex}$), change the assignment of the right to call exclusive services (if permitted).
- Use a non-exclusive service

0929_{hex}

Meaning: A message cannot be transmitted.

Cause: The specified bit number of the target address is unknown.

Remedy: Select a permitted bit number and send the message again to the respective target:
Bit number 0: Diagnostic display on the front plate, if present
Bit number 1: Standard signal interface (SSGI)

Add_Error_Info: Incorrect bit number

$092A_{hex}$

Meaning: Indication code that is not permitted or is unknown.

Cause: An indication code that is not permitted was used as a parameter.

Remedy: Indicate the code of the indication to be enabled or disabled, e.g., 4341_{hex} for the indication "Fault_Indication".

Add_Error_Info: Indication code that is incorrect or is not permitted.

$092B_{hex}$ (USER FAIL)

Cause:
- A device used a communication reference, which had not been assigned to it.
- A channel was opened via the V.24 interface that cannot be accessed.
- The interface was modified while a PCP channel was open.

$092D_{hex}$ (USER FAIL)

Meaning: The controller board rejected a PCP service.

Cause: Another PCP service is still being processed on this communication reference with the same Invoke ID.

Remedy: Wait for the PCP confirmation of the active service, and then send the new service.

$092E_{hex}$

Cause: Insufficient memory available on the controller board for initializing the standard signal interface (SSGI).

Remedy: Use the "Reset_Controller_Board" service (0956_{hex}) to execute a warm start of the controller board, and try again.

0930_{hex} (USER FAIL)

Meaning: The controller board cannot process the "Set_Value" service (0750_{hex}).

Cause: You specified wrong parameter values when calling the service.

Remedy: Check the call.

0931_{hex} (USER FAIL)

Meaning: The controller board could not process the service called last.

Cause: You used a non-defined value for the *Variable_ID* parameter.

Remedy: Check the *Variable_ID* parameter.

0932_{hex} (USER FAIL)

Meaning: The specified MPM accessor could not enable the exclusive rights.

Cause: The specified MPM accessor did not have the exclusive rights.

Remedy: Use the "Change_Exclusive_Rights" service ($014F_{hex}$) to enable the exclusive rights only for the MPM accessor, which has these rights.

0933_{hex} (USER FAIL)

Meaning: The request for exclusive rights was denied.

Cause: A different MPM accessor still has the exclusive rights.

Remedy: Use the "Change_Exclusive_Rights_Request" service ($014F_{hex}$) to enable the exclusive rights only when no other MPM accessor already has these rights.

0934_{hex} (USER FAIL)

Meaning: The request for exclusive rights was denied.

Cause: The MPM accessor already has the exclusive rights.

0935_{hex} (USER FAIL)

Meaning: You exceeded the permissible value range when specifying a communication reference (CR).

Remedy: Assign only communication references within the range of $2 \leq CR \leq 127$.

0936_{hex}

Meaning: Incorrect Variable_ID code.

Cause: Either the "Set_Value" or "Read_Value" service was sent with an unknown Variable_ID code.

Remedy: Transmit the service again with the correct ID code.

Add_Error_Info: Incorrect Variable_ID code.

0937_{hex}

Meaning: Incorrect Variable_ID component.

Cause: Either the "Set_Value" or "Read_Value" service was sent with an unknown Variable_ID component.

Remedy: Transmit the service again with the correct Variable_ID component.

Add_Error_Info: Incorrect Variable_ID component.

0938_{hex} (USER FAIL)

Meaning: You used a value for the *Variable_ID* parameter that is reserved for the firmware.

Add_Error_Info: Reserved *Variable_ID* parameter.

0939_{hex} (USER FAIL)

Meaning: You used a value for the *Variable_ID* parameter that is not enabled.

Add_Error_Info: *Variable_ID* parameter not enabled.

$093A_{hex}$ (USER FAIL)

Meaning: The controller board could not process the "Set_Value" service (0750_{hex}).

Cause: You specified an incorrect value for the *Variable_ID* parameter. In bits 8 to 11 of the *Variable_ID* parameter, the length of the variable in words was entered incorrectly.

Remedy: Check the call of this service.

Add_Error_Info: Incorrect *Variable_ID*.

$093B_{hex}$ (USER FAIL)

Meaning: The controller board could not process the service called last.

Cause: The variable length encoded in the *Variable_ID* parameter is not identical to the length of the specified variable.

Remedy: Check the call of the service called last.

$093C_{hex}$ (USER FAIL)

Cause: You defined a signal handle object incorrectly.

Remedy: Check the definition of the signal object.

$093D_{hex}$ (USER FAIL)

Cause: You specified an invalid bit number in the signal object.

Remedy: Define the signal object correctly.

Add_Error_Info: Invalid bit number.

$093E_{hex}$ (USER FAIL)

Cause: You exceeded the maximum permissible number of write parameters.

Remedy: Check the service call.

$093F_{hex}$ (USER FAIL)

Meaning: You exceeded the maximum permissible number of read parameters.

Cause: You specified too many parameters or incorrect parameters.

Remedy: Check the service call.

0940_{hex} (USER FAIL)

Meaning:	Do not enter write parameter addresses for a resident action.
Cause:	You specified write parameter addresses for a resident action.
Remedy:	Check the service call.

0941_{hex}

Meaning:	The controller board could not process the boot sequence without errors.
Cause:	Error in a service call.
Remedy:	Check the call of the service specified in the *Add_Error_Info* parameter.
Add_Error_Info:	Code of the service where the error occurred.

0942_{hex} (USER FAIL)

Meaning:	The controller board could not process the "Clear_Parameterization_Memory" service (0159_{hex}).
Cause:	You specified an incorrect value for the *Action_Flag* parameter.
Remedy:	Check the call of this service.
Add_Error_Info:	Incorrect parameter value.

0943_{hex} (USER FAIL)

Meaning:	The controller board could not process the "Clear_Parameterization_Memory" service (0159_{hex}) without errors.
Cause:	You specified an incorrect address field with the *Area_Flag* parameter.
Remedy:	Check the call of this service.
Add_Error_Info:	Incorrect parameter value.

0944_{hex} (USER FAIL)

Meaning: The controller board could not process the "Clear_Parameterization_Memory" service (0159_{hex}).

Cause: You specified a non-defined *Action_Flag*.

Remedy: Check the call of this service.

0945_{hex} (USER FAIL)

Meaning: The controller board could not process the "Clear_Parameterization_Memory" service (0159_{hex}). Access to the parameterization memory was denied.

Cause: Parameterization memory not available or not properly inserted.

Remedy: Make sure that the parameterization memory has been properly inserted.

0946_{hex} (USER FAIL)

Meaning: Access to the parameterization memory was denied.

Cause: The parameterization memory is write-protected.

Remedy: Remove the write protection of the parameterization memory and try again.

0947_{hex} (USER FAIL)

Cause: You defined an action object incorrectly.

Remedy: Check the definitions of the action objects.

Add_Error_Info: Action index of the incorrect action object.

0948_{hex} (USER FAIL)

Cause: You defined a signal object incorrectly.

Remedy: Check the definitions of the signal objects.

Add_Error_Info: Signal index of the incorrect signal object.

0949_{hex} (USER FAIL)

Cause: You defined an event object incorrectly.

Remedy: Check the definitions of the event objects.

Add_Error_Info: Event index of the incorrect event object.

0950_{hex}

Meaning: The controller board cannot print the information.

Cause: The diagnostic interface of the controller board is not in print mode.

Remedy: Switch the diagnostic interface to print mode using the "Set_Value" service (0750_{hex}).

0951_{hex}

Meaning: The controller board cannot print the information.

Cause: The buffer of the diagnostic interface is full.

Remedy: Wait a moment, then try again.

0952_{hex} (USER FAIL)

Meaning: You used a non-defined value for the *Variable_ID* parameter.

Remedy: Check the definitions of the *Variable_ID* and try again.

0953_{hex} (USER FAIL)

Meaning: The controller board could not process the service called last.

Cause: You specified an invalid parameter.

Remedy: Check the call of this service.

0954_{hex} (USER FAIL)

Meaning: The controller board could not process the service called last for the diagnostic interface.

Cause: You specified an incorrect value for the *Parameter_Count* parameter.

Remedy: Check the call of this service. The *Parameter_Count* parameter must correspond to the number of subsequent words.

0955_{hex}

Meaning: The message was not read or sent.

Cause: The message exceeds the maximum permissible mailbox size of 1024 bytes of the diagnostic interface.

Remedy: If the error code occurs after a request, you can read the message in several steps. Some services can be called several times to read large data records. Enter an address offset for the second call.

0956_{hex}

Meaning: The message was not read or sent.

Cause: When sending PCP services via the SSGI, a PCP service was written to an SSGI box with a communication reference, which does not correspond to the SSGI box.

Remedy: Only write a PCP service in the SSGI box, if the box number corresponds to the communication reference.

0957_{hex}

Meaning: All remote PCP services of the established connection were aborted.

Cause: The connection was interrupted due to a PMS or PNM7 abort on a communication reference.

Remedy: Check the application program and start it again.

0960_{hex}

Meaning: The *Variable_ID* parameter used for the diagnostic display is invalid.

Remedy: Use the correct value for the *Variable_ID* parameter.

Add_Error_Info: Invalid *Variable_ID* parameter.

0970_{hex}

Meaning: When communicating with the controller board (e.g., via IB Loader), an error occurred in the IEC 61131 runtime system of the controller board.

Remedy: Send the services more slowly, i.e., build in a waiting time between the services.

$09D0_{hex}$

Meaning: You are using a parameterization memory that is not supported by your firmware.

Remedy: Replace the existing parameterization memory with a suitable one.

$09D1_{hex}$

Meaning: The parameterization memory that you are using is faulty.

Remedy: Replace the existing parameterization memory with an intact memory.

$09D2_{hex}$

Meaning: You are using a parameterization memory that is not supported by your firmware.

Remedy: Replace the existing parameterization memory with a suitable one.

$09D3_{hex}$

Meaning: The number of open files on the parameterization memory can no longer be managed.

$09D4_{hex}$

Meaning: You tried to save a file on the parameterization memory that has a name with too many characters.

Remedy: Rename the file.

$09D5_{hex}$

Meaning: You violated the file access rights to an open file on the parameterization memory of the controller board.

Cause: You tried, for example, to write to a file that has read access only.

Remedy: Note the access rights when using files.

$09D6_{hex}$

Meaning: Insufficient memory available on the parameterization memory of the controller board.

Remedy:
- Delete some files.
- Save the files and format the parameterization memory again.

$09D8_{hex}$

Meaning: Insufficient memory available on the parameterization memory.

Cause: The parameterization memory may, for example, be too fragmented.

Remedy: Save the files and format the parameterization memory again.

$09D9_{hex}$

Meaning: The desired file does not exist or cannot be found on the parameterization memory of the controller board.

$09DA_{hex}$

Meaning: The desired file could not be read by the parameterization memory of the controller board.

Cause: An error occurred while reading the file.

$09DB_{hex}$

Meaning: A file could not be written on the parameterization memory of the controller board.

Cause: An error occurred while writing the file.

$09DC_{hex}$

Meaning: The desired file on the parameterization memory of the controller board could not be accessed.

Cause: An error occurred when the file was accessed using the "File_Seek" service ($015F_{hex}$). The file pointer could not be positioned correctly.

$09DD_{hex}$

Meaning: A file on the parameterization memory of the controller board could not be closed.

Cause: An error occurred while closing the file.

$09DE_{hex}$

Meaning: A file on the parameterization memory of the controller board could not be opened.

Cause: An error occurred while opening the file because:

- No parameterization memory is inserted.
- The file is faulty.
- The file is not available.
- The parameterization memory is write-protected or faulty.

$09DF_{hex}$

Meaning: The desired file could not be deleted from the parameterization memory of the controller board.

Cause: An error occurred while deleting the file.

$09F0_{hex}$

Meaning: There is a maximum number of files that the parameterization memory of the controller board can handle. This number was exceeded.

Remedy:
- Delete some files.
- Save the files and format the parameterization memory again.

$09F1_{hex}$

Meaning: The file structure on the parameterization memory of the controller board cannot be read.

Cause: The parameterization memory was written with an older firmware version.

Remedy: Format the parameterization memory again.

$09F2_{hex}$

Meaning: A file on the parameterization memory of the controller board cannot be processed. The file handle is incorrect or invalid.

Remedy: Enter the right file handle for the file to be processed.

$09FC_{hex}$

Meaning: It is not possible to write to the parameterization memory. A timeout is triggered.

Cause: The parameterization memory cannot be programmed.

Remedy: Replace the parameterization memory.

$0A02_{hex}$ (USER FAIL)

Meaning: The controller board could not process the service called last.

Cause: You sent a service, which is not permitted in the current state of the controller board. For example, you cannot send the "Start_Data_Transfer" service (0701_{hex}) when the controller board is in the *Ready* state. To start data transfer, the controller board must be in the *Active* state.

Remedy: Set the controller board to the required state before calling the desired service.

Add_Error_Info: Current state of the controller board:

0001_{hex}	Ready (display: *RDY*).
0002_{hex}	Parameterization not Ready.
0004_{hex}	Loading the configuration (Loading CFG).
0008_{hex}	Loading the process data description list (Loading PDDL).
0010_{hex}	Loading the process data reference list (Loading PDRL).
0020_{hex}	Parameterization Ready.
0040_{hex}	Controller board running sporadic ID cycles (display: *ACTV*).
0080_{hex}	Controller board running data cycles (display: *RUN*).
0100_{hex}	Bus fail.
0200_{hex}	Controller board looking for error (display: *Look For Fail*)

$0A03_{hex}$

Meaning: Insufficient memory available on the controller board.

Cause: The main memory may, for example, be too fragmented.

Remedy: Use the "Reset_Controller_Board" service (0956_{hex}) to execute a warm start of the controller board, and try again.

$0A04_{hex}$ (USER FAIL)

Meaning: The controller board could not process the service called last.

Cause: You specified inconsistent parameters.

Remedy: Check the call of this service.

Add_Error_Info: Number of the inconsistent parameter.

$0A05_{hex}$ (USER FAIL)

Meaning: The controller board could not process the service called last.

Cause: You specified an invalid parameter.

Remedy: Check the call of this service.

Add_Error_Info: Number of the invalid parameter.

$0A06_{hex}$ (USER FAIL)

Meaning: The controller board could not access the object.

Cause: You attempted to access an object. This access is not supported by the system management for the object, e.g., a write access to an object to which only read access is permitted.

$0A07_{hex}$ (USER FAIL)

Meaning: The controller board could not access the object.

Cause: You tried to access an object that does not exist.

$0A08_{hex}$ (USER FAIL)

Meaning:	You called a service, which causes a processing conflict with another service called before.
Cause:	The firmware cannot process two services at the same time.
Remedy:	Wait for the service called previously to be completed, and then try again.
Add_Error_Info:	Code of the service called previously.

$0A10_{hex}$ (USER FAIL)

Meaning:	The controller board cannot process the "Set_Value" service (0750_{hex}).
Cause:	You specified an incorrect value for the *Variable_ID* parameter.
Add_Error_Info:	Number of the incorrect Variable_ID.

$0A11_{hex}$ (USER FAIL)

Meaning:	The controller board cannot process the "Set_Value" service (0750_{hex}).
Cause:	You exceeded the permitted value range when specifying a parameter.
Remedy:	Check the parameters of this service.

$0A12_{hex}$ (USER FAIL)

Cause:	The value range of the *Device_Level* parameter was exceeded.
Remedy:	Check the *Device_Level* parameter. The values from 00_{hex} to $0F_{hex}$ (corresponding to 0 to 15_{dec}) are permitted.
Add_Error_Info:	Position in the configuration frame.

$0A14_{hex}$ (USER FAIL)

Cause: You specified an invalid value for the *Error_Char_Flag* parameter.

Remedy: Check the *Error_Char_Flag* parameter. Only values 0000_{hex} and 0001_{hex} are permitted.

Add_Error_Info: Position in the configuration frame.

$0A15_{hex}$ (USER FAIL)

Cause: You specified an invalid value for the *Position* parameter.

Remedy: Check the *Position* parameter. Values from 00_{hex} to $3F_{hex}$ (0_{dec} to 63_{dec}) are permitted.

Add_Error_Info: Position in the configuration frame.

$0A16_{hex}$ (USER FAIL)

Cause: You specified an invalid value for the INTERBUS device number (Segment . Position).

Remedy: Check the INTERBUS device numbers:
Permitted values for *Segment* range from 01_{hex} to FF_{hex} (1_{dec} to 255_{dec}).
Permitted values for *Position* range from 00_{hex} to $3F_{hex}$ (0_{dec} to 63_{dec}).

Add_Error_Info: Invalid INTERBUS device number.

$0A17_{hex}$ (USER FAIL)

Cause: You assigned an unknown ID code.

Remedy: Check the parameters of the configuration frame.

Add_Error_Info: Position in the configuration frame.

$0A18_{hex}$ (USER FAIL)

Cause: You activated an invalid bit in the *Used_Attributes* parameter.

Remedy: Check the *Used_Attributes* parameter of the corresponding service.

Add_Error_Info: Invalid *Used_Attributes* parameter.

$0A19_{hex}$ (USER FAIL)

Meaning: You exceeded the maximum bus configuration when accessing a configuration frame.

Remedy: Check the last access to the configuration frame.

Add_Error_Info: Number of INTERBUS devices.

$0A1A_{hex}$ (USER FAIL)

Meaning: The specified *Frame_Reference* does not exist or no *Frame_Reference* could be assigned.

Remedy: Check the *Frame_Reference* parameter.

Add_Error_Info: Invalid *Frame_Reference* (if specified).

$0A1B_{hex}$ (USER FAIL)

Meaning: The specified configuration frame could not be deleted.

Cause: The specified configuration frame is currently activated. It is not possible to delete the activated configuration frame.

Remedy: If you really want to delete the activated configuration frame, first deactivate it using the "Deactivate_Configuration" service (0712_{hex}).

Add_Error_Info: Number of the configuration frame.

$0A1C_{hex}$ (USER FAIL)

Cause: You exceeded the permitted number of specified or connected INTERBUS devices. The maximum permissible number of INTERBUS devices is 255.

Add_Error_Info: Number of specified or connected INTERBUS devices.

$0A1D_{hex}$ (USER FAIL)

Meaning: The specified configuration frame could not be overwritten.

Cause: You tried to overwrite an existing configuration frame with new configuration data. The size (number of INTERBUS devices) was not identical to the existing frame. If you want to create a configuration frame of a different size, create it under a new, unused *Frame_Reference*.

Remedy: Compare the existing configuration frame with the new configuration data.

$0A1E_{hex}$ (USER FAIL)

Meaning: The new extension could not be assigned to the configuration frame.

Cause: You tried to assign a new extension to an existing configuration frame, and this extension is too long for the space provided for the extension in the configuration frame.

Remedy: Adapt the size of the new extension to the space provided.

$0A1F_{hex}$ (USER FAIL)

Meaning: The specified configuration frame could not be deactivated.

Cause: The specified configuration was already inactive.

Add_Error_Info: *Frame_Reference* parameter

$0A20_{hex}$ (USER FAIL)

Meaning: The first physical INTERBUS device in the INTERBUS network was not switched off.

Cause: The first physical INTERBUS device cannot be switched off.

$0A21_{hex}$ (USER FAIL)

Cause: You assigned an INTERBUS device number more than once.

Remedy: Check the INTERBUS device numbers in the configuration frame.

Add_Error_Info: Position in the configuration frame.

$0A22_{hex}$ (USER FAIL)

Cause: You assigned inconsistent INTERBUS device numbers.

Remedy: Check the INTERBUS device numbers in the configuration frame.

Add_Error_Info: Position in the configuration frame.

$0A23_{hex}$ (USER FAIL)

Cause: You assigned inconsistent INTERBUS device levels.

Remedy: Check the configuration frame.

Add_Error_Info: Position in the configuration frame.

$0A24_{hex}$ (USER FAIL)

Meaning: Within one local bus segment the connected devices have different operating states.

Cause: Not all devices were switched on or off using the "Control_Active_Configuration" service (0713_{hex}).

Remedy: Switch all devices in a local bus segment on or off.

Add_Error_Info: Position in the configuration frame.

$0A25_{hex}$ (USER FAIL)

Meaning: The controller board did not accept your group definition.

Cause: You assigned different group numbers for INTERBUS devices in one bus segment.

Remedy: Always assign the same group number to all INTERBUS devices in one bus segment.

Add_Error_Info: Position in the configuration frame.

$0A26_{hex}$ (USER FAIL)

Meaning: The controller board did not accept the definitions of an alternative group.

Cause: You assigned invalid alternative group numbers.

Add_Error_Info: Position in the configuration frame.

$0A27_{hex}$

Meaning: The controller board did not accept the definitions of an alternative group.

Cause: The first device in the alternative group is not a bus terminal module.

Add_Error_Info: Position in the configuration frame.

$0A28_{hex}$ (USER FAIL)

Meaning: The controller board could not activate the specified groups.

Cause: You tried to activate alternative groups at the same time.

Add_Error_Info: Position in the configuration frame.

$0A29_{hex}$

Meaning: There was a conflict of mutual group dependencies when INTERBUS devices were switched on or off.

Cause: Using the "Control_Active_Configuration" service (0713_{hex}) you caused inconsistencies when switching mutually dependent groups.

Add_Error_Info: Position in the configuration frame.

$0A2A_{hex}$

Meaning: Conflict of mutual INTERBUS device dependencies (active/inactive).

Cause: Using the "Control_Active_Configuration" service (0713_{hex}) you caused inconsistencies when switching mutually dependent INTERBUS devices.

Add_Error_Info: Position in the configuration frame.

0A2B$_{hex}$ (USER FAIL)

Meaning: Status conflict within a group.

Cause: You assigned different statuses to INTERBUS devices belonging to one group.

Remedy: Always assign the same status to the INTERBUS devices of one group.

Add_Error_Info: Position in the configuration frame.

0A2C$_{hex}$ (USER FAIL)

Cause: You specified an INTERBUS device number, which does not exist.

Add_Error_Info: Non-existent INTERBUS device number.

0A2D$_{hex}$ (USER FAIL)

Meaning: Too many PCP devices.

Cause:
- You connected more then the permitted number of PCP devices.
- You configured more then the permitted number of PCP devices.

Remedy: Reduce the number of connected or configured PCP devices. A maximum of 126 devices is permitted, depending on the controller board used and the firmware version.

0A2E$_{hex}$ (USER FAIL)

Meaning: The permitted number of internal indirect address list entries was exceeded. You have reached the firmware memory limit.

Cause: You have too many modules that occupy only one byte or one nibble of address space in the data ring.

Remedy:
- Reduce the number of modules occupying only one byte or one nibble of address space. The maximum permissible number of internal indirect address list entries is 384.
- Arrange the modules so that the devices that require less than 1 word of address space are next to each other.

$0A2F_{hex}$ (USER FAIL)

Meaning: The controller board could not execute the "Initiate_Load_Configuration" service (0306_{hex}).

Cause: You entered a 0 for the number of INTERBUS devices.

Remedy: Specify the correct number of INTERBUS devices.

$0A30_{hex}$ (USER FAIL)

Meaning: Incorrect entry in the process data description list (PDDL).

Remedy: Check the process data description list.

$0A31_{hex}$ (USER FAIL)

Meaning: The controller board could not process the "Put_Process_Data_Description_List" service (0321_{hex}) or the "Get_Process_Data_Description_List" service (0323_{hex}).

Cause: You used an INTERBUS device number, which does not exist.

Add_Error_Info: Non-existent INTERBUS device number in the format [RRLL].
RR = Remote bus segment number
LL = Local bus segment number

$0A32_{hex}$ (USER FAIL)

Meaning: The controller board could not process the "Put_Process_Data_Description_List" service (0321_{hex}).

Cause: You specified an invalid value in a process data description for the *PDD_Index* parameter.

Remedy: Assign values in the range from 0000_{hex} to $7FFF_{hex}$ for the *PDD_Index* parameter, with the exception of the values 6010_{hex} and 6011_{hex}, which are reserved for default process data descriptions. Every *PDD_Index* may be assigned only once.

Add_Error_Info: Index of the process data description.

$0A33_{hex}$ (USER FAIL)

Meaning: You specified an incorrect data direction for an INTERBUS device.

Cause:
- You defined a process data description for OUT process data for an INTERBUS device, which has only IN process data.
- You defined a process data description for IN process data for an INTERBUS device, which has only OUT process data.

Remedy: Use the *Data_Direction* parameter to define the data direction for the process data descriptions as follows:
$0C_{hex}$ for IN process data (IN-PDD)
$0D_{hex}$ for OUT process data (OUT-PDD)

Add_Error_Info: Index of the process data description.

$0A34_{hex}$ (USER FAIL)

Cause: You exceeded the internal address area of an INTERBUS device when describing a process data item.

Add_Error_Info: Index of the process data object.

$0A35_{hex}$ (USER FAIL)

Cause: When writing to a process data item, you specified a data type and a data length that do not correspond.

Remedy: Check the process data description.

Add_Error_Info: Index of the process data description.

$0A36_{hex}$ (USER FAIL)

Meaning:	Error when defining a process data item.
Cause:	Using the "Put_Process_Data_Description_List" service (0321_{hex}) you defined a bit string process data item that exceeds a byte boundary.
Remedy:	Only define bit string process data that does not exceed byte boundaries and, therefore, has a maximum length of 8 bits.
Add_Error_Info:	Index of the process data description.

$0A40_{hex}$ (USER FAIL)

Meaning:	Incorrect entry in the process data reference list (PDRL).
Remedy:	Check the process data reference list.
Add_Error_Info:	*PDRL_Index* of the incorrect PDRL entry.

$0A41_{hex}$ (USER FAIL)

Meaning:	Incorrect entry in the process data reference list.
Cause:	You did not specify an OUT process data item as the target.
Remedy:	Specify an OUT process data item as the target.
Add_Error_Info:	*PDRL_Index* of the incorrect PDRL entry.

$0A42_{hex}$ (USER FAIL)

Meaning:	Incorrect entry in the process data reference list.
Cause:	You did not specify an IN process data item as the source.
Remedy:	Specify an IN process data item as the source.
Add_Error_Info:	*PDRL_Index* of the incorrect PDRL entry.

$0A43_{hex}$ (USER FAIL)

Meaning:	Incorrect entry in the process data reference list.
Cause:	You specified an INTERBUS device number, which does not exist.
Remedy:	Check the process data reference list.
Add_Error_Info:	*PDRL_Index* of the incorrect PDRL entry.

$0A44_{hex}$ (USER FAIL)

Meaning:	Incorrect entry in the process data reference list.
Cause:	You specified a data consistency for a process data item, which does not correspond to its length.
Remedy:	Check the process data reference list.
Add_Error_Info:	*PDRL_Index* of the incorrect PDRL entry.

$0A45_{hex}$ (USER FAIL)

Meaning:	Incorrect entry in the process data reference list.
Cause:	You specified a value for the *PDD_Index* parameter, which does not exist.
Remedy:	Check the process data reference list.
Add_Error_Info:	*PDRL_Index* of the incorrect PDRL entry.

$0A46_{hex}$ (USER FAIL)

Meaning:	Addressing error.
Cause:	For the "Compact_Load_Process_Data_Reference_List" service (0328_{hex}) or "Load_Process_Data_Reference_List" service (0325_{hex}), you specified an incorrect value for the *Address_Direction* parameter.
Remedy:	Enter: 1000_{hex} for an input address list 2000_{hex} for an output address list
Add_Error_Info:	*PDRL_Index* of the incorrect PDRL entry.

$0A47_{hex}$ (USER FAIL)

Meaning: Incorrect entry in the process data reference list.

Cause: You specified an incorrect or unknown value for the *PDRL_Index* parameter.

Remedy: Check the process data reference list.

Add_Error_Info: *PDRL_Index* of the incorrect PDRL entry.

$0A48_{hex}$ (USER FAIL)

Meaning: By defining a direct link you can map an IN process data item directly to an OUT process data item of the same process data length. One of these entries in the PDRL is not correct.

Cause: You tried to map:

- An IN process data item to another IN process data item.
- An OUT process data item to another OUT process data item.
- An IN process data item to an OUT process data item with a different process data length.

Remedy: Check the direct link definitions.

Add_Error_Info: *PDRL_Index* of the incorrect PDRL entry.

$0A49_{hex}$ (USER FAIL)

Meaning: The controller board could not start up the connected bus configuration.

Cause: For the "Compact_Load__Process_Data_Reference_List" service (0328_{hex}) you specified a number of INTERBUS devices, which differs from the actual number of devices in the connected bus configuration using the *Entry_Count* parameter.

Remedy: Using the "Compact_Read__Process_Data_Reference_List" service (0329_{hex}), check the assignment of the process data and compare it with the number of INTERBUS devices of your connected bus configuration.

Add_Error_Info: Identification whether IN or OUT list.

$0A4A_{hex}$ (USER FAIL)

Meaning:	The controller board could not read the process data reference list (PDRL).
Cause:	Using the "Compact_Read_Process_Data_Reference_List" service (0329_{hex}), you tried to read a PDRL loaded with the "Load_Process_Data_Reference_List" service (0325_{hex}).
Remedy:	PDRLs loaded with the "Load_Process_Data_Reference_List" service (0325_{hex}) are to be read with the "Read_Process_Data_Reference_List" service (0327_{hex}).

$0A4B_{hex}$ (USER FAIL)

Meaning:	Error when assigning a process data item.
Cause:	Using the "Load_Process_Data_Reference_List" service (0325_{hex}) you assigned a bit string process data item so that it exceeds a byte boundary.
Remedy:	Shift the bit string process data item until it does not exceed the byte boundary.
Add_Error_Info:	Bit position of the incorrectly assigned bit string process data item in the MPM.

$0A4C_{hex}$ (USER FAIL)

Meaning:	Error when assigning a process data item.
Cause:	You assigned an odd MPM address to a process data item with a data consistency of 16, 32, or 64 bits. However, this is only permitted for process data with a data consistency of 8 bits.
Remedy:	Always assign even MPM addresses to process data with a data consistency of 16, 32, or 64 bits.
Add_Error_Info:	*PDRL_Index* of the incorrect PDRL entry.

$0A50_{hex}$ (USER FAIL)

Meaning: Error when assigning a process data item.

Cause: You assigned an output process data description or an input host address several times.

Remedy: Each output process data description should only be assigned one host address and each input host address should only be assigned one process data description.

Add_Error_Info: *PDRL_Index* of the incorrect PDRL entry.

Additional information for system coupler cards:

If a larger bus configuration (system coupler card and number of devices > 225, all with 32 bits of DIO) is read with the Create_Configuration service (0710_{hex}), the default I/O data will be stored in address area 0000_{hex} to $03FF_{hex}$. This causes a conflict with system coupler address 0380_{hex}.

Remedy: Assign the system coupler to an address located after the address area of the default I/O data.

$0A51_{hex}$ (USER FAIL)

Meaning: The controller board did not accept the *Frame_Reference* parameter.

Cause: The specified value is not permitted for the *Frame_Reference* parameter.

Remedy: Change the value of the *Frame_Reference* parameter.
Permissible value range: 1≤ *Frame_Reference* ≤ 254

$0A52_{hex}$ (USER FAIL)

Cause: You tried to jumper an active device.

Remedy: First switch off the device using "Control_Active_Configuration" (0713_{hex}).

Add_Error_Info: Position in the configuration frame.

$0A53_{hex}$ (USER FAIL)

Meaning: The maximum number of remote bus devices was exceeded.

Cause: The maximum permissible number of remote bus devices was exceeded by the user when using the "Load_Configuration" service (0307_{hex}) or when reading the connected devices automatically using the "Create_Configuration" service (0710_{hex}).

Remedy: Reduce the number of remote bus device to less than 254.

$0A54_{hex}$ (USER FAIL)

Meaning: The maximum number of I/O points was exceeded.

Cause: The maximum permissible number of I/O points was exceeded by the user when using the "Load_Configuration" service (0307_{hex}) or when automatically reading the connected devices using the "Create_Configuration" service (0710_{hex}).

Remedy: Reduce the number of I/O points to the maximum number of digital inputs and outputs for the input and output data area. Depending on the type of controller board, you may use up to 8192 I/O points. To obtain the exact number, refer to the documentation for your controller board.

$0A55_{hex}$

Meaning: The configuration data transmitted to the controller board using the "Compare_Configuration" service (0317_{hex}) does not correspond to the configuration frame data specified with the *Frame_Reference* parameter.

Cause: The device number *(Segment/Position)* specified by the *Add_Error_Info* parameter has been entered differently.

Remedy: Check the list of transmitted configuration data against the specified configuration frame.

Add_Error_Info: Line number of the configuration frame, which does not correspond to the transmitted configuration data.

$0A56_{hex}$

Meaning: The configuration data transmitted to the controller board using the "Compare_Configuration" service (0317_{hex}) does not correspond to the configuration frame data specified with the *Frame_Reference* parameter.

Cause: The device code (length and ID code) specified by *Add_Error_Info* parameter has been entered differently.

Remedy: Check the list of transmitted configuration data against the specified configuration frame.

Add_Error_Info: Line number of the configuration frame, which does not correspond to the transmitted configuration data.

$0A57_{hex}$

Meaning: The configuration data transmitted to the controller board using the "Compare_Configuration" service (0317_{hex}) does not correspond to the configuration frame data specified with the *Frame_Reference* parameter.

Cause: The device level specified by the *Add_Error_Info* parameter has been entered differently.

Remedy: Check the list of transmitted configuration data against the specified configuration frame.

Add_Error_Info: Line number of the configuration frame, which does not correspond to the transmitted configuration data.

$0A58_{hex}$

Meaning: The configuration data transmitted to the controller board using the "Compare_Configuration" service (0317_{hex}) does not correspond to the configuration frame data specified with the *Frame_Reference* parameter.

Cause: The group number (group and alternative) specified by the *Add_Error_Info* parameter has been entered differently.

Remedy: Check the lists of transmitted configuration data against the specified configuration frame.

Add_Error_Info: Line number of the configuration frame, which does not correspond to the transmitted configuration data.

$0A59_{hex}$

Meaning: The local system coupler must not be switched.

Cause: You tried to switch the slave part of the local system coupler.

$0A5A_{hex}$

Meaning: The "Control_Active_Configuration" service (0713_{hex}) was used in an attempt to switch on the active devices.

Cause: The active device cannot be switched on with this service.

Add_Error_Info: Line number of the configuration frame.

$0A5B_{hex}$

Meaning: The configuration data transmitted to the controller board using the "Compare_Configuration" service (0317_{hex}) does not correspond to the configuration frame data specified with the *Frame_Reference* parameter.

Cause: The device specified by the *Add_Error_Info* parameter, which is to be switched in isolation, has been entered differently.

Remedy: Compare the list of transmitted configuration data with the specified configuration frame.

Add_Error_Info: Line number of the configuration frame, which does not correspond to the transmitted configuration data.

$0A5C_{hex}$

Meaning: The "Create_Configuration" service (0710_{hex}) was used in an attempt to read a local bus configuration.

Cause: The local bus read contains too many bus devices (>=63).

Remedy: Check your local bus configuration.

$0A60_{hex}$ (USER FAIL)

Meaning:	The controller board could not assign a configuration frame.
Cause:	You specified a value for the *Frame_Reference* parameter, under which no configuration frame has been created.
Remedy:	Observe the required order of services. Create the configuration frame before accessing it.

$0A63_{hex}$ (USER FAIL)

Meaning:	A diagnostic register is in the wrong address area.
Cause:	You placed the diagnostic status register or the diagnostic parameter register in the output address area.
Remedy:	Always place these registers in the input address area.
Add_Error_Info:	*Variable_ID* of the incorrectly assigned register.

$0A64_{hex}$ (USER FAIL)

Meaning:	Address conflict between a diagnostic register and an IN process data item.
Cause:	The diagnostic status register or the diagnostic parameter register is located in an address area, which the controller board uses for IN process data. This may occur after calling the "Create_Configuration" service (0710_{hex}).
Remedy:	Using the "Set_Value" service (0750_{hex}), place the diagnostic status register and the diagnostic parameter register in address areas, which are not occupied by IN process data.
Add_Error_Info:	*Variable_ID* of the register that caused the address conflict.

$0A65_{hex}$ (USER FAIL)

Meaning: Error when assigning a register.

Cause: You assigned an odd MPM address to a register with a data consistency of 16, 32, or 64 bits. However, this is only permitted for registers with a data consistency of 8 bits.

Remedy: Always assign even MPM addresses to registers with a data consistency of 16, 32, or 64 bits.

Add_Error_Info: *Variable_ID* of the register to which the wrong byte address has been assigned.

$0A70_{hex}$

Meaning: The controller board could not process the "Get_Diag_Info" service ($032B_{hex}$).

Cause: You tried to enable a reserved attribute with the *Diag_Info_Attr* parameter.

$0A80_{hex}$

Meaning: The configuration was rejected, the bus cannot be started up.

Cause: You parameterized a bus, which consists of SUPI 3 devices parameterized for isolated disconnection. The bus terminal module, whose branching branch interface was parameterized for isolated disconnected, does not have an OPC chip.

Remedy: Only use devices with SUPI 3 OPC as bus terminal modules.

Add_Error_Info: Line number in the configuration frame.

$0AFB_{hex}$

Meaning: The controller board did not accept the specified data consistency.

Cause: You assigned two different data consistencies when defining the process data references of a memory cell in the MPM.

Remedy: Check the assignments of the data consistencies.

$0AFC_{hex}$

Meaning: A hardware error occurred on the controller board.

Remedy: Replace the controller board.

$0AFD_{hex}$

Meaning: The contents of the address decoder EEPROMs has changed.

Remedy: Execute a power-up/reset of your controller board and restart your control system.

$0AFE_{hex}$

Meaning: An address overlap was detected when reading the connected bus structure.

Cause: The addresses read for the bus devices overlap the address of the communication register.

Remedy:
- Set the address for the communication register in CMD the same as it is set on the controller board.
- Set a higher address for the communication register on the controller board that is not assigned during addressing, or change the addresses of the bus device.
- To read the configuration frame using the CMD software tool, place the communication register in a higher address area on the controller board.

Special remark: Only relevant for Siemens S5 or Bosch PLCs.

$0B02_{hex}$ (USER FAIL)

Cause:

1. In bus-synchronous operating mode:
   - No bus cycle time is set
   - The set bus cycle time is too short
2. For firmware Version 4.15 or earlier this may be because:
   - There is an empty configuration frame
   - The first device after the controller board is switched off

Remedy:

1. Set an appropriate bus cycle time.
2. – Activate a correct configuration frame
   – Switch on the first device

$0B00_{hex}$, $0B01_{hex}$, $0B03_{hex}$

Meaning: A firmware error occurred on the controller board.

Remedy: Replace the controller board.

$0B80_{hex}$ (USER FAIL)

Meaning: The controller board cannot process the "Set_Value" service (0750_{hex}) or "Read_Value" service (0351_{hex}).

Cause: You specified an invalid value for the *Variable_ID* parameter.

Remedy: Use only the values specified in the description of the "Set_Value" service (0750_{hex}) or "Read_Value" service (0351_{hex}).

$0B81_{hex}$ (USER FAIL)

Meaning: The controller board cannot process the "Set_Value" service (0750_{hex}).

Cause: You exceeded the permitted value range when specifying a parameter.

Remedy: Use only the values specified in the description of the "Set_Value" service (0750_{hex}).

$0B83_{hex}$ (USER FAIL)

Meaning: The controller board cannot process the "Control_Device_Function" service (0714_{hex}).

Cause: You specified an invalid value for the *Device_Function* parameter.

Remedy: Use only the values specified in the description of the "Control_Device_Function" service (0714_{hex}).

$0B84_{hex}$ (USER FAIL)

Meaning: The controller board cannot process the "Control_Device_Function" service (0714_{hex}).

Cause: An error was detected in the list of physical INTERBUS device positions.

Remedy: Check the list of physical INTERBUS device positions.

$0B85_{hex}$ (USER FAIL)

Meaning: The configuration cannot be created.

Cause: An error was detected in the active configuration.

Remedy: Check the list of the active configuration.

$0B86_{hex}$ (USER FAIL)

Meaning: The controller board cannot process the "Control_Device_Function" service (0714_{hex}).

Cause: You tried to use this service for an INTERBUS device without a command register.

$0B87_{hex}$ (USER FAIL)

Meaning: The controller board cannot process the "Control_Device_Function" service (0714_{hex}).

Cause: You specified a number for the *Entry_Count* parameter, which is greater than the actual number of INTERBUS devices.

Remedy: Check the service call.

$0B88_{hex}$

Meaning: The controller board cannot process the "Control_Device_Function" service (0714_{hex}).

Cause: Firmware error on the controller board.

Remedy: Replace the controller board.

$0B8A_{hex}$ (USER FAIL)

Meaning: After activating a new or modified configuration, the controller board changed to the stop state.

Cause: You sent the "Alarm_Stop" service (1303_{hex}) too early.

Remedy: Send the "Alarm_Stop" service (1303_{hex}) later.

$0B8C_{hex}$ (USER FAIL)

Meaning: The controller board could not activate the configuration frame.

Cause: You exceeded the maximum permissible number of I/O bits.

Remedy: Reduce the number of INTERBUS devices. Depending on the type of controller board, you may use up to 8192 I/O points. To obtain the exact number, refer to the documentation for your controller board.

$0B8D_{hex}$ (USER FAIL)

Meaning: A synchronous interrupt was illegally initiated.

Cause: You initiated a synchronous interrupt in the MPM, although the controller board was in asynchronous mode or the bus was not started.

Remedy: Initiate synchronous interrupts only when the controller board is operating in a synchronous mode and the bus is running.

$0B8E_{hex}$ (USER FAIL)

Meaning: Unexpected MPM synchronous interrupt (diagonal handshake interrupt) initiated.

Cause: You initiated a diagonal handshake interrupt in the MPM, which is required for synchronous operating modes, although the controller board was working in asynchronous mode or the bus was not started.

Remedy: Initiate synchronous interrupts only when the controller board is operating in a synchronous mode and the bus is running.

$0B8F_{hex}$ (USER FAIL)

Meaning: Overflow of the input FIFO for the INTERBUS protocol master chip (IPMS).

Cause:
1. You accessed a data area with a data width, which is smaller than the data consistency defined for this area.
2. The transmission quality is low because the bus was installed incorrectly.

Remedy:
1. Access this data area only with the data width that is equal to the full data consistency defined for this data area.
2. Check the transmission path.

$0B90_{hex}$

Meaning: The last INTERBUS cycle was aborted.

Cause: A hardware fault has occurred on the controller board.

Remedy: Replace the controller board.

$0B91_{hex}$ (USER FAIL)

Meaning: In a bus state that is not ACTIVE or RUN, an MPM accessor caused a timeout.

Cause: A data consistency error occurred. This is due to the fact that not all of the bits have been accepted from the MPM.

$0B92_{hex}$ (USER FAIL)

Meaning:	A READ_PD service could not be processed completely.
Cause:	A data cycle could not be executed within a timeout as, for example, the bus is not in the RUN state or a bus error occurred during service execution.
Remedy:	Change the bus state to RUN and then call the READ_PD service once again.

$0B93_{hex}$ (USER FAIL)

Meaning:	After an I/O timeout, the outputs on the device are reset.
Cause:	The INTERBUS protocol chip could not process I/O data for a specified time. The protocol chip may be faulty.
Remedy:	Restart the system.
Add_Error_Info:	Device number (Segment . Position) of the INTERBUS device.

$0B94_{hex}$

Meaning:	Incorrect diagnostic indications may have occurred.
Cause:	A local bus device, which does not support local bus diagnostics, is connected to a bus terminal module with local bus diagnostics.
Remedy:	Replace the local bus device.
Add_Error_Info:	Physical position of the first local bus device without local bus diagnostics.

$0B97_{hex}$

Meaning:	A scan time (mean PD cycle time) must be assigned for the program-synchronous or bus-synchronous mode.
Cause:	The scan time was not specified.
Remedy:	Enter a value for the scan time.

$0BB1_{hex}$ (PF)

Meaning: The specified INTERBUS device is indicating a peripheral fault.

Remedy: Check the specified INTERBUS device.

Add_Error_Info: Device number (Segment . Position) of the INTERBUS device.

$0BB2_{hex}$

Meaning: Reconfiguration request of the specified INTERBUS device.

Cause: The reconfiguration button was pressed on the specified INTERBUS device.

Add_Error_Info: Device number (Segment . Position) of the INTERBUS device.

$0BB4_{hex}$

Meaning: The microprocessor of the specified device performed a reset.

Remedy: Check this INTERBUS device.

Add_Error_Info: Error location (Segment . Position).

$0BB5_{hex}$

Meaning: The transmission quality on the data forward path of the incoming bus interface (IN) of the specified device has deteriorated.

Remedy: Check the data lines of the incoming bus interface.

Add_Error_Info: Device number (Segment . Position) of the INTERBUS device.

$0BB6_{hex}$

Meaning: The transmission quality on the data return path of the incoming bus interface (IN) of the specified device has deteriorated.

Remedy: Check the data lines of the incoming bus interface.

Add_Error_Info: Device number (Segment . Position) of the INTERBUS device.

$0BB7_{hex}$

Meaning: The specified device indicated an I/O timeout and reset all outputs, if available. The response is identical to the response given upon a bus reset.

Add_Error_Info: Device number (Segment . Position) of the INTERBUS device.

$0BC0_{hex}$

Meaning: Error indicated by system coupler.

Cause:
- The lower-level bus changed to the STOP state due to a bus error.
- Voltage reset of the system coupler master part.

Add_Error_Info: Device number (Segment . Position) of the INTERBUS device.

$0BC2_{hex}$

Meaning: The voltage is too low for the initiators.

Cause:
1. Too many devices are configured in the installation local bus ring.
2. The total extension of the installation local bus ring is too large.

Remedy:
1. Check the configuration and reduce the number of connected devices.
2. Reduce the extension of the installation local bus.

Add_Error_Info: Device number (Segment . Position) of the INTERBUS device.

$0BC3_{hex}$

Meaning: The temperature of the INTERBUS protocol chip is too high.

Cause: The ambient temperature is too high.

Remedy:
- Reduce the ambient temperature.
- Provide separate ventilation for the device.

Add_Error_Info: Device number (Segment . Position) of the INTERBUS device.

$0BC4_{hex}$

Meaning: The internal power source for a Loop device is overloaded.

Cause: The connected load of sensors and/or actuators is too high.

Remedy:
- Check the number of connected sensors/actuators.
- Check the sensors/actuators for errors.

Add_Error_Info: Device number (Segment . Position) of the INTERBUS device.

$0BC5_{hex}$

Meaning: The permitted output current of the power drive was exceeded.

Cause: The connected load of sensors and/or actuators is too high.

Remedy:
- Check the number of connected sensors/actuators.
- Check the sensors/actuators for errors.

Add_Error_Info: Device number (Segment . Position) of the INTERBUS device.

$0BD0_{hex}$

Meaning: The controller board could not start up the configuration.

Cause: No bus configuration connected.

Remedy: Connect a bus configuration.

$0BD1_{hex}$

Meaning: A data cycle exceeded the specified PD cycle time.

The controller board indicates this only if it has been activated with the "Set_Value" service (0750_{hex}) before.

Cause: The process data could not be updated within the specified PD cycle time.

Remedy: Check your system or increase the value set for the PD cycle time using the "Set_Value" service (0750_{hex}).

$0BD2_{hex}$

Meaning: The bus warning time has elapsed.

Cause: No data cycle could be transmitted within the bus warning time specified with the "Set_Value" service (0750_{hex}).

Remedy:
- Check your system.
- Increase the bus warning time with the "Set_Value" service (0750_{hex}).

$0BD4_{hex}$

Meaning: A single bus error occurred.
The controller board indicates this only if the function has been activated before.

$0BD5_{hex}$

Meaning: The branching branch interface of the device specified in the additional info has branched into the connected segment due to an error.

Cause: An error has occurred in the bus segment connected to the branching branch interface of the device.

Remedy: Check your bus configuration.

Add_Error_Info: Device number of the connected INTERBUS device.

$0BDB_{hex}$

Meaning: The process preprocessing task is no longer in the RUN state. The bus was stopped to avoid data transmission without preprocessing.

Cause: Error in the preprocessing program (e.g., division by zero) or intended stopping of the preprocessing by the user.

$0BDC_{hex}$

Meaning: An alarm stop was generated by the controller board in the ACTIVE or RUN state.

Cause: The alarm stop was generated due to a SYSFAIL in the host system.

$0BDD_{hex}$

Meaning: The bus was stopped due to an MPM timeout caused by the user. Otherwise, inconsistent data could have been transmitted.

Cause: For example, the timeout can be caused by a byte access to a 16 bit-locked MPM address.

Remedy: Access only those data areas, which are equal to the full data consistency defined for this data area.

$0BDE_{hex}$

Meaning: A synchronization error occurred in *bus-synchronous* operating mode.

Remedy:
- Check your system.
- Increase the PD cycle time with the "Set_Value" service (0750_{hex}).

$0BDF_{hex}$ (LOOK FOR FAIL)

Meaning: The controller board has stopped data transmission and is searching for the error location and error cause.

Cause: A bus error occurred.

Remedy: Wait until the search for the error has been completed. The controller board will inform you of the result.

$0BE0_{hex}$

Meaning: Error localization could not be terminated.

Cause: During error localization (Look for Fail), the user sent the "Alarm_Stop" service (1303_{hex}).

3.2 Error Codes for General Bus Errors (BUS FAIL)

The error codes **0BE1$_{hex}$** to **0BEA$_{hex}$** are assigned to general bus errors where the error location or area cannot be clearly defined. The display of the controller board only shows the last two characters, i.e., **E1$_{hex}$** to **EA$_{hex}$**.

Please note that once these errors have occurred and the diagnostic information has been read from the diagnostic parameter registers with the "Read_Value" service, the error code is mapped in the diagnostic parameter register and the value 0000_{hex} is mapped in the extended diagnostic parameter register.

E1$_{hex}$ or 0BE1$_{hex}$ (BUS FAIL)

Meaning:

A serious error occurred that caused the bus system to be switched off. However, no error could be found when checking the current configuration. This points to the cause being an intermittent error.

Cause:

The error occurs due to:

- Installation errors
- A faulty INTERBUS device

Remedy:

Check your system for:

- Missing or incorrect shielding of the bus cables (connectors)
- Missing or incorrect grounding/equipotential bonding
- Poor connections in the connector (loose contact, cold junction)
- Cable breaks in remote and local bus cabling
- Voltage dips on the communications power for remote bus devices

If entries are made in the Top Ten, they can provide information on the error location. Top Ten information can be viewed using the diagnostic display or the "Get_Diag_Info" service ($032B_{hex}$).

Please note that once the $E0_{hex}$ and $E1_{hex}$ errors have occurred and the diagnostic information has been read from the diagnostic parameter registers with the "Read_Value" service, the error code is mapped in the diagnostic parameter register and the error location is mapped in the extended diagnostic parameter register.

$E2_{hex}$ or $0BE2_{hex}$ (BUS FAIL)

Meaning:

The controller board detected changes in the configuration, which do not enable data traffic over the bus to be continued.

Cause:

- You exceeded the maximum permissible number of INTERBUS words.
- You exceeded the maximum permissible number of INTERBUS devices.

$E4_{hex}$ or $0BE4_{hex}$ (BUS FAIL)

Meaning:

A serious error occurred when acquiring the bus configuration via the "Create_Configuration" service (0710_{hex}). This error caused the bus system to be switched off, and the error location could not be detected. This points to the cause being an intermittent error. The error rate can be very high.

Cause:

The error occurs due to:

- Installation errors
- A faulty INTERBUS device

Remedy:

Check your system for:

- Missing or incorrect shielding of the bus cables (connectors)
- Missing or incorrect grounding/equipotential bonding
- Poor connections in the connector (loose contact, cold junction)
- Cable breaks in remote and local bus cabling
- Voltage dips on the communications power for remote bus devices

The diagnostic options available on the controller board can be improved if you know the configuration of the bus system. For this reason, it is better to load the desired configuration onto the controller board using the "Load_Configuration" service (0307_{hex}) or the "Complete_Load_Configuration" service ($030A_{hex}$) before activating it with the "Activate_Configuration" service (0711_{hex}).

$E6_{hex}$ or $0BE6_{hex}$ (BUS FAIL)

Meaning:

A serious error occurred that caused the bus system to be switched off. However, no error could be found when checking the current configuration. This points to the cause being an intermittent error. The error affected data cycles but not ID cycles.

Cause:

The error occurs due to:

- Installation errors
- A faulty INTERBUS device
- When jumpering devices with a data width of less than 8 bits (due to the transport of data of these devices in the controller board memory)

Remedy:

Check your system for:

- Missing or incorrect shielding of the bus cables (connectors)
- Missing or incorrect grounding/equipotential bonding
- Poor connections in the connector (loose contact, cold junction)
- Cable breaks in remote and local bus cabling
- Voltage dips on the communications power for remote bus devices

If entries are made in the Top Ten, they can provide information on the error location. Top Ten information can be viewed using the diagnostic display or the "Get_Diag_Info" service ($032B_{hex}$).

$E7_{hex}$ or $0BE7_{hex}$ (BUS FAIL)

Meaning:

The controller board could not activate the configuration when the following services were processed:

- "Activate_Configuration" (0711_{hex})
- "Control_Active_Configuration" (0713_{hex})

The error location could not be detected.

Cause: The error occurs due to:
- Installation errors
- A faulty INTERBUS device

Remedy: Check your system for:
- Missing or incorrect shielding of the bus cables (connectors)
- Missing or incorrect grounding/equipotential bonding
- Poor connections in the connector (loose contact, cold junction)
- Cable breaks in remote and local bus cabling
- Voltage dips on the communications power for remote bus devices

$E8_{hex}$ or $0BE8_{hex}$ (BUS FAIL)

Meaning: A serious error occurred that caused the bus system to be switched off. When checking the current configuration, the diagnostic algorithm detected errors but could not locate the precise error location. This points to the cause being an intermittent error. The error rate can be very high.

Cause: The error occurs due to:
- Installation errors
- A faulty INTERBUS device

Remedy: Check your system for:
- Missing or incorrect shielding of the bus cables (connectors)
- Missing or incorrect grounding/equipotential bonding
- Poor connections in the connector (loose contact, cold junction)
- Cable breaks in remote and local bus cabling
- Voltage dips on the communications power for remote bus devices

If entries are made in the Top Ten, they can provide information on the error location. Top Ten information can be viewed using the diagnostic display or the "Get_Diag_Info" service ($032B_{hex}$).

$E9_{hex}$ or $0BE9_{hex}$ (BUS FAIL)

Meaning: A serious error occurred that caused the bus system to be switched off. When checking the current configuration, the diagnostic algorithm detected errors but could not locate the precise error location. This points to the cause being an intermittent error. The error rate can be very high.

Cause: The error occurs due to:

- Installation errors
- A faulty INTERBUS device

Remedy: Check your system for:

- Missing or incorrect shielding of the bus cables (connectors)
- Missing or incorrect grounding/equipotential bonding
- Poor connections in the connector (loose contact, cold junction)
- Cable breaks in remote and local bus cabling
- Voltage dips on the communications power for remote bus devices

If entries are made in the Top Ten, they can provide information on the error location. Top Ten information can be viewed using the diagnostic display or the "Get_Diag_Info" service ($032B_{hex}$).

EA_{hex} or $0BEA_{hex}$ (BUS FAIL)

Meaning: The "Control_Device_Function" service (0714_{hex}) could not be executed.

Cause: A fatal error occurred.

Remedy: Repeat the service if the controller board is still in the RUN or ACTIVE state. If diagnostics is active, you must wait for the result. The bus error indicated shows the error location.

$0BF0_{hex}$ (BUS FAIL)

Meaning: Data transmission was temporarily interrupted. The controller board reset all outputs and stopped data transmission. The diagnostic display indicates an INTERBUS device number. The error can be found:

- In the preceding bus segment of a local bus
- In the preceding bus segment of a ST compact station
- In the bus segments of a preceding remote bus branch (e.g., installation remote bus)
- In the bus segment of the indicated INTERBUS device

Cause:

- Voltage reset of an INTERBUS device in the specified area
- Cable break in the specified bus segment
- The jumper (RBST or LBST) in the connector for the outgoing bus is faulty for an INTERBUS device in the specified area

Add_Error_Info: Device number (Segment . Position) of the INTERBUS device.

$0BF1_{hex}$ (BUS FAIL)

Meaning: Data transmission is interrupted at a bus terminal module.

Cause:

- The connector for the outgoing remote bus branch has not been plugged in.
- The jumper (LBST) in the connector for the outgoing remote bus branch is faulty.

Remedy: Check the connector for the outgoing remote bus branch.

Add_Error_Info: Device number (Segment . Position) of the INTERBUS device.

$0BF2_{hex}$ (BUS FAIL)

Meaning: Data transmission is interrupted at a bus terminal module.

Cause:

- The connector for the outgoing remote bus has not been plugged in.
- The jumper (RBST) in the connector for the outgoing remote bus is faulty.

Remedy: Check the connector for the outgoing remote bus.

Add_Error_Info: Device number (Segment . Position) of the INTERBUS device.

0BF3$_{hex}$ (BUS FAIL)

Meaning:

Data transmission is interrupted:

- At a bus terminal module
- At a local bus device
- Within an IB ST compact station

Cause:

Local bus

- The connector for the outgoing local bus has not been plugged in.
- The jumper (RBST or LBST) in the connector for the outgoing local bus is faulty.

ST compact station

- The ST cable is not plugged in.
- The RBST connection led via the next module of the IB ST compact station is interrupted.

Add_Error_Info:

Device number (Segment . Position) of the INTERBUS device.

0BF4$_{hex}$ (BUS FAIL)

Meaning:

Transmission error (CRC error) on the data forward path of the incoming bus interface (IN) of the indicated INTERBUS device.

Cause:

Transmission errors

Remedy:

Check the segment of the specified INTERBUS device for:

- Missing or incorrect shielding of the bus cables (connectors)
- Missing or incorrect grounding/equipotential bonding
- Poor connections in the connector (loose contact, cold junction)
- Voltage dips on the communications power for remote bus devices
- Faulty fiber optic assembly

Add_Error_Info:

Device number (Segment . Position) of the INTERBUS device.

$0BF5_{hex}$ (BUS FAIL)

Meaning: Transmission error (CRC error) on the data return path of the incoming bus interface (IN) of the indicated INTERBUS device.

Remedy: Check the segment of the specified INTERBUS device for:

- missing or incorrect shielding of the bus cables (connectors)
- Missing or incorrect grounding/equipotential bonding
- Poor connections in the connector (loose contact, cold junction)
- Voltage dips on the communications power for remote bus devices
- Faulty fiber optic assembly

Add_Error_Info: Device number (Segment . Position) of the INTERBUS device.

$0BF6_{hex}$ (BUS FAIL)

Meaning: Data transmission was temporarily interrupted. The controller board reset all outputs and stopped data transmission. The diagnostic display indicates an INTERBUS device number. The error can be found:

- In the preceding bus segment of a local bus
- In the preceding bus segment of a ST compact station
- In the bus segments of a preceding remote bus branch (e.g., installation remote bus)
- In the bus segment of the indicated INTERBUS device

Cause: Possible causes:

- Voltage reset of an INTERBUS device in the specified area
- Cable break in the specified bus segment
- The jumper (RBST or LBST) in the connector for the outgoing bus is faulty for an INTERBUS device in the specified area

Add_Error_Info: Device number (Segment . Position) of the INTERBUS device.

$0BF8_{hex}$ (BUS FAIL)

Meaning: Multiple error when acquiring I/O data at the specified device. It was not possible to find the exact location of the error.

Cause: The error occurs due to:

- Installation errors
- A faulty INTERBUS device

Possible error locations:

– Indicated devices
– The complete bus located prior to the device
– All devices connected to the OUT2 interface of the indicated device

Remedy:

Check your system for:

– missing or incorrect shielding of the bus cables (connectors)
– Missing or incorrect grounding/equipotential bonding
– Poor connections in the connector (loose contact, cold junction)
– Cable breaks in remote and local bus cabling
– Voltage dips on the communications power for remote bus devices

If entries are made in the Top Ten, they can provide information on the error location. Top Ten information can be viewed using the diagnostic display or the "Get_Diag_Info" service ($032B_{hex}$).

Add_Error_Info:

Device number (Segment . Position) of the INTERBUS device.

$0BF9_{hex}$ (BUS FAIL)

Meaning:

Multiple error at the specified device during quick diagnostics. It was not possible to find the exact location of the error.

Cause:

The error occurs due to:

– Installation errors
– A faulty INTERBUS device

Possible error locations:

– Indicated devices
– The complete bus located prior to the device
– All devices connected to OUT2 of the indicated device

Remedy: Check your system for:

- Missing or incorrect shielding of the bus cables (connectors)
- Missing or incorrect grounding/equipotential bonding
- Poor connections in the connector (loose contact, cold junction)
- Cable breaks in remote and local bus cabling
- Voltage dips on the communications power for remote bus devices

If entries are made in the Top Ten, they can provide information on the error location. Top Ten information can be viewed using the diagnostic display or the "Get_Diag_Info" service ($032B_{hex}$).

Add_Error_Info: Device number (Segment . Position) of the INTERBUS device.

$0BFA_{hex}$ (BUS FAIL)

Meaning: Multiple error at the specified device during startup or permanent diagnostics.

Cause: The error occurs due to:

- Installation errors
- A faulty INTERBUS device

Possible error locations:

- Indicated devices
- The complete bus located prior to the device
- All devices connected to OUT2 of the indicated device

Remedy: Check your system for:

- Missing or incorrect shielding of the bus cables (connectors)
- Missing or incorrect grounding/equipotential bonding
- Poor connections in the connector (loose contact, cold junction)
- Cable breaks in remote and local bus cabling
- Voltage dips on the communications power for remote bus devices

If entries are made in the Top Ten, they can provide information on the error location. Top Ten information can be viewed using the diagnostic display or the "Get_Diag_Info" service ($032B_{hex}$).

Add_Error_Info: Device number (Segment . Position) of the INTERBUS device.

$0BFB_{hex}$ (BUS FAIL)

Meaning: Error detected using quick diagnostics.

Possible error locations:

- Indicated devices
- The complete bus located prior to the device
- All devices connected to OUT2 of the indicated device

Remedy: Check your system for:

- Missing or incorrect shielding of the bus cables (connectors)
- Missing or incorrect grounding/equipotential bonding
- Poor connections in the connector (loose contact, cold junction)
- Cable breaks in remote and local bus cabling
- Voltage dips on the communications power for remote bus devices

If entries are made in the Top Ten, they can provide information on the error location. Top Ten information can be viewed using the diagnostic display or the "Get_Diag_Info" service ($032B_{hex}$).

Add_Error_Info: Device number (Segment . Position) of the INTERBUS device.

0BFC$_{hex}$ (BUS FAIL)

Meaning:

Data transmission was temporarily interrupted. The controller board reset all outputs and stopped data transmission. The diagnostic display indicates an INTERBUS device number. The error can be found:

- In the preceding bus segment of a local bus
- In the preceding bus segment of a ST compact station
- In the bus segments of a preceding remote bus branch (e.g., installation remote bus)
- In the bus segment of the indicated INTERBUS device.
- On the transmission paths or devices in the branches, which precede the indicated INTERBUS device

Cause:

- Voltage reset of an INTERBUS device in the specified area
- Cable break in the specified bus segment
- The jumper (RBST or LBST) in the connector for the outgoing bus is faulty for an INTERBUS device in the specified area.

Add_Error_Info:

Device number (Segment . Position) of the INTERBUS device.

3.3 Error Codes for Remote Bus and Local Bus Errors

The *Add_Error_Info* provides the coded error location for remote or local bus errors. The exact error position is only indicated if no interface error occurred. In the case of an interface error, only the faulty bus segment will be indicated. Bit 7 indicates whether an interface error occurred. The meanings of bits 0 to 6 will also change. This results in three different states, which have the following bit combinations in the *Add_Error_Info*.

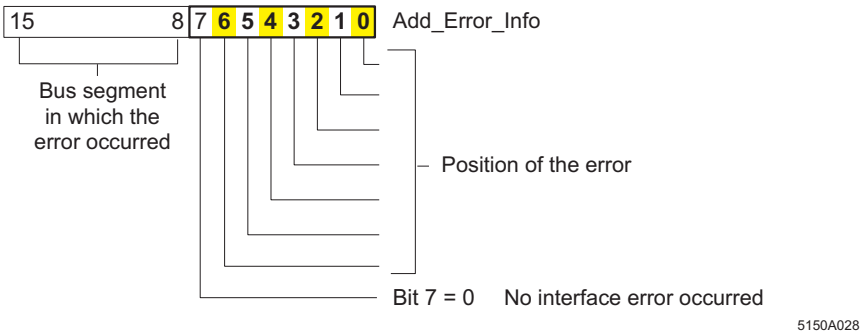


Figure 3-1 No interface error occurred

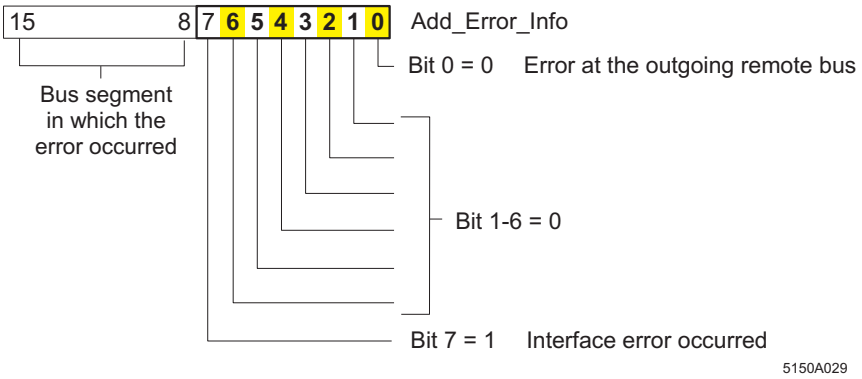


Figure 3-2 Error on the outgoing remote bus interface

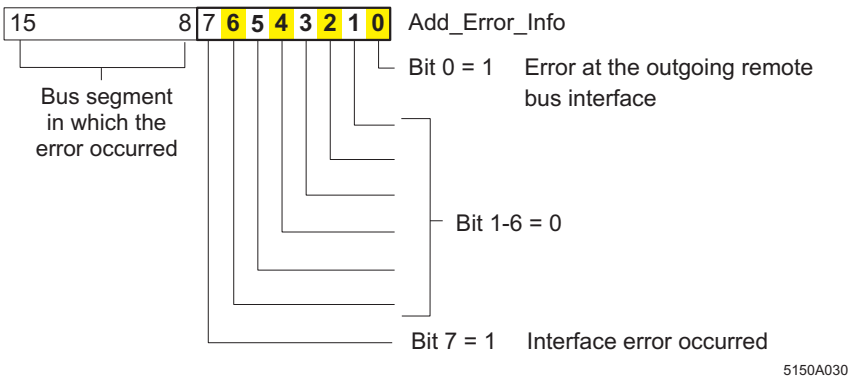


Figure 3-3 Error on the outgoing local bus interface

$0C10_{hex}$ to $0C13_{hex}$ (RB FAIL) and
$0D10_{hex}$ to $0D13_{hex}$ (LB FAIL)

Meaning: An INTERBUS device is missing.

Cause: A device entered in the active bus configuration and not marked as switched off is missing in the connected bus configuration.
The active configuration is the quantity of INTERBUS devices connected to the INTERBUS system whose data is within the summation frame during bus cycles. The active configuration may differ from the connected bus configuration only when physically connected bus segments have been switched off.

Remedy: Compare the active configuration with the connected bus configuration, taking any disabled bus segments into account.

Add_Error_Info: Error location (Segment . Position).

$0C14_{hex}$ to $0C17_{hex}$ (RB FAIL) and
$0D14_{hex}$ to $0D17_{hex}$ (LB FAIL)

Meaning: Multiple error in the segment of the indicated INTERBUS device.

Cause: Transmission errors

Remedy: Check the segment of the specified INTERBUS device for:
- Missing or incorrect shielding of the bus cables (connectors)
- Missing or incorrect grounding/equipotential bonding
- Poor connections in the connector (loose contact, cold junction)
- Voltage dips on the communications power for remote bus devices

Add_Error_Info: Error location (Segment . Position).

$0C18_{hex}$ to $0C1B_{hex}$ (RB FAIL) and
$0D18_{hex}$ to $0D1B_{hex}$ (LB FAIL)

Meaning: Multiple timeout in the segment of the indicated INTERBUS device.

Cause: Transmission errors

Remedy: Check the segment of the specified INTERBUS device for:
- Missing or incorrect shielding of the bus cables (connectors)
- Missing or incorrect grounding/equipotential bonding
- Poor connections in the connector (loose contact, cold junction)
- Voltage dips on the communications power for remote bus devices

Add_Error_Info: Error location (Segment . Position).

$0C1C_{hex}$ to $0C1F_{hex}$ (RB FAIL) and
$0D1C_{hex}$ to $0D1F_{hex}$ (LB FAIL)

Meaning: Transmission error (CRC error) on the data forward path of the incoming bus interface (IN) of the indicated INTERBUS device.

Cause: Transmission errors

Remedy: Check the segment of the specified INTERBUS device for:
- Missing or incorrect shielding of the bus cables (connectors)
- Missing or incorrect grounding/equipotential bonding
- Poor connections in the connector (loose contact, cold junction)
- Voltage dips on the communications power for remote bus devices

Add_Error_Info: Error location (Segment . Position).

$0C20_{hex}$ to $0C23_{hex}$ (RB FAIL) and
$0D20_{hex}$ to $0D23_{hex}$ (LB FAIL)

Meaning:	The Medium Attachment Unit (MAU) firmware component diagnosed an interruption in data transmission.
Cause:	Cable break on the data forward path of the incoming bus interface (IN) of the indicated INTERBUS device.
Remedy:	Check the cables, connectors, and INTERBUS connections for interruptions and repair them, if required.
Add_Error_Info:	Error location (Segment . Position).

$0C24_{hex}$ to $0C27_{hex}$ (RB FAIL) and
$0D24_{hex}$ to $0D27_{hex}$ (LB FAIL)

Meaning:	Transmission error (CRC error) on the data return path of the incoming bus interface (IN) of the indicated INTERBUS device.
Cause:	Transmission errors
Remedy:	Check the segment of the specified INTERBUS device for: – Missing or incorrect shielding of the bus cables (connectors) – Missing or incorrect grounding/equipotential bonding – Poor connections in the connector (loose contact, cold junction) – Voltage dips on the communications power for remote bus devices
Add_Error_Info:	Error location (Segment . Position).

$0C28_{hex}$ to $0C2B_{hex}$ (RB FAIL) and
$0D28_{hex}$ to $0D2B_{hex}$ (LB FAIL)

Meaning: The Medium Attachment Unit (MAU) diagnosed an interruption in data transmission.

Cause: Cable break on the data return path of the incoming bus interface (IN) of the indicated INTERBUS device.

Remedy: Check the cables, connectors, and INTERBUS connections for interruptions and repair them, if required.

Add_Error_Info: Error location (Segment . Position).

$0C2C_{hex}$ to $0C2F_{hex}$ (RB FAIL) and
$0D2C_{hex}$ to $0D2F_{hex}$ (LB FAIL)

Meaning: Unexpected change of the RBST or LBST signal.

Cause: Missing or faulty jumper (loose contact, cold junction) in the outgoing interface of the preceding bus device.

Remedy: Check the segment of the specified INTERBUS device for poor connections in the connector (loose contact, cold junction). Solder a jumper or ensure the proper connection of the already existing jumper to generate an error-free RBST or LBST signal.

Add_Error_Info: Error location (Segment . Position).

$0C30_{hex}$ to $0C33_{hex}$ (RB FAIL)

Meaning: Multiple error in the segment of the indicated INTERBUS device.

Cause:
1. Transmission errors
2. The specified INTERBUS device has a SUPI 1 slave chip, which is operated in µP mode. This mode is not supported by the firmware of your controller board.

Remedy:
1. Check the segment of the specified INTERBUS device for:
   - Missing or incorrect shielding of the bus cables (connectors)
   - Missing or incorrect grounding/equipotential bonding
   - Poor connections in the connector (loose contact, cold junction)
   - Voltage dips on the communications power for remote bus devices
2. Replace the indicated device with a device, which has a SUPI 3 chip.

Add_Error_Info: Error location (Segment . Position).

$0C40_{hex}$ to $0C43_{hex}$ (RB FAIL) and
$0D40_{hex}$ to $0D43_{hex}$ (LB FAIL)

Meaning: The length code of the specified INTERBUS device is not identical to the entry in the configuration frame.

Remedy: Adapt the active configuration frame to the bus configuration.

Add_Error_Info: Error location (Segment . Position).

$0C44_{hex}$ to $0C47_{hex}$ (RB FAIL) and
$0D44_{hex}$ to $0D47_{hex}$ (LB FAIL)

Meaning: The ID code of the specified INTERBUS device is not identical to the entry in the configuration frame.

Remedy: Adapt the active configuration frame to the bus configuration.

Add_Error_Info: Error location (Segment . Position).

$0C48_{hex}$ to $0C4B_{hex}$ (RB FAIL) and
$0D48_{hex}$ to $0D4B_{hex}$ (LB FAIL)

Meaning: The bus can be read but not started up. Only ID cycles but not data cycles can be run.

Cause:
- The data register of the specified INTERBUS device is interrupted.
- The number of data registers of the specified INTERBUS device is not identical to the length code entered in the configuration frame.

Add_Error_Info: Error location (Segment . Position).

$0C4C_{hex}$ to $0C4F_{hex}$ (RB FAIL) and
$0D4C_{hex}$ to $0D4F_{hex}$ (LB FAIL)

Meaning: The specified INTERBUS device has an invalid ID code.

Add_Error_Info: Error location (Segment . Position).

$0D50_{hex}$ to $0D53_{hex}$ (LB FAIL)

Meaning: The specified INTERBUS device has the ID code of a remote bus device, but is located in a local bus.

Add_Error_Info: Error location (Segment . Position).

$0C54_{hex}$ to $0C57_{hex}$ (RB FAIL) and
$0D54_{hex}$ to $0D57_{hex}$ (LB FAIL)

Meaning: The specified INTERBUS device has a SUPI 1 slave chip, which is operated in µP mode. This mode is not supported by the firmware of your controller board.

Remedy: Replace the device with a device, which has a SUPI 3 chip.

Add_Error_Info: Error location (Segment . Position).

$0C58_{hex}$ to $0C5B_{hex}$ (RB FAIL) and
$0D58_{hex}$ to $0D5B_{hex}$ (LB FAIL)

Meaning: Data transmission is interrupted at the outgoing remote bus interface (OUT1) of the specified INTERBUS device.

Cause:
1. The connector is not plugged in.
2. The jumper for connector identification (RBST or LBST) is faulty.

Remedy:
1. Check that the connector is plugged in.
2. Check whether the jumper for connector identification (RBST or LBST) is faulty.

Add_Error_Info: Error location (Segment . Position).

$0C5C_{hex}$ to $0C5F_{hex}$ (RB FAIL) and
$0D5C_{hex}$ to $0D5F_{hex}$ (LB FAIL)

Meaning: Data transmission is interrupted at the branching bus interface (OUT2) of the specified INTERBUS device.

Cause:
1. The connector is not plugged in.
2. The bridge for connector identification (RBST or LBST) is faulty.

Remedy:
1. Check that the connector is plugged in.
2. Check whether the jumper for connector identification (RBST or LBST) is faulty.

Add_Error_Info: Error location (Segment . Position).

$0C60_{hex}$ to $0C63_{hex}$ (RB FAIL) and
$0D60_{hex}$ to $0D63_{hex}$ (LB FAIL)

Meaning: Data transmission was temporarily interrupted at the outgoing remote bus interface (OUT1) of the specified INTERBUS device (with SUPI 3), although this interface is not in use.

Cause: The SUPI 3 protocol chip detected a CRC or MAU error.

Remedy: Replace the INTERBUS device.

Add_Error_Info: Error location (Segment . Position).

$0C64_{hex}$ to $0C67_{hex}$ (RB FAIL) and
$0D64_{hex}$ to $0D67_{hex}$ (LB FAIL)

Meaning: Data transmission was temporarily interrupted at the branching bus interface (OUT2) of the specified INTERBUS device (with SUPI 3), although this interface is not in use.

Cause: The SUPI 3 protocol chip detected a CRC or MAU error.

Remedy: Replace the INTERBUS device.

Add_Error_Info: Error location (Segment . Position).

$0C68_{hex}$ to $0C6B_{hex}$ (RB FAIL) and
$0D68_{hex}$ to $0D6B_{hex}$ (LB FAIL)

Meaning: The SUPI 3 protocol chip of the specified INTERBUS device detected an I/O timeout.

Add_Error_Info: Error location (Segment . Position).

$0C6C_{hex}$ to $0C6F_{hex}$ (RB FAIL) and
$0D6C_{hex}$ to $0D6F_{hex}$ (LB FAIL)

Meaning: The specified INTERBUS device executed a reset.

Cause: The specified INTERBUS device is not being supplied with enough power or is faulty.

Remedy:
- Check this INTERBUS device.
- Check the supply voltage of this INTERBUS device to determine whether it conforms to the nominal value and whether the permissible AC voltage component is exceeded. Refer to the relevant data sheet for the values.
- Check whether the power supply unit for the bus terminal module is overloaded. Refer to the relevant data sheets for the maximum permissible output current of the bus terminal module and for the typical current consumption of the connected local bus devices.

Add_Error_Info: Error location (Segment . Position).

$0C70_{hex}$ to $0C73_{hex}$ (RB FAIL) and
$0D70_{hex}$ to $0D73_{hex}$ (LB FAIL)

Meaning: Data transmission was aborted. In an INTERBUS device whose protocol chip is run in microprocessor mode, the microprocessor failed to initialize the protocol chip.

Cause:
1. The controller board tried to switch the bus to the *ACTIVE* state faster than the microprocessor of the INTERBUS device could initialize the protocol chip.
2. The INTERBUS device is faulty.

Remedy:
1. Delay the call of the "Activate_Configuration" service (0711_{hex}) until the microprocessor has initialized the protocol chip.
2. Replace the INTERBUS device.

Add_Error_Info: Error location (Segment . Position).

$0C74_{hex}$ to $0C77_{hex}$ (RB FAIL) and
$0D74_{hex}$ to $0D77_{hex}$ (LB FAIL)

Meaning: Data transmission was aborted.

Cause: An invalid mode has been set on the protocol chip of an INTERBUS device.

Remedy:
– Set a valid mode
– Replace the device

Add_Error_Info: Error location (Segment . Position).

$0C78_{hex}$ (RB FAIL) and $0D78_{hex}$ (LB FAIL)

Meaning: If the dynamic PCP channel is switched on, the data length of the specified device is not identical to the configuration frame.

Add_Error_Info: Error location (Segment . Position).

$0C79_{hex}$ (RB FAIL) and $0D79_{hex}$ (LB FAIL)

Meaning: If the dynamic PCP channel is switched on, the ID code of the specified device is not identical to the configuration frame.

Add_Error_Info: Error location (Segment . Position).

$0C7A_{hex}$ (RB FAIL) and $0D7A_{hex}$ (LB FAIL)

Meaning: The width of the dynamic PCP channel of the specified device is not permitted.

Add_Error_Info: Error location (Segment . Position).

$0C7B_{hex}$ (RB FAIL) and $0D7B_{hex}$ (LB FAIL)

Meaning: The specified device has an ID code for devices with a dynamic PCP channel, but cannot use this channel.

Cause:
- The specified device is not provided with a corresponding protocol chip.
- The firmware of the controller board does not support devices with dynamic PCP channels yet.

Add_Error_Info: Error location (Segment . Position).

$0C7C_{hex}$ (RB FAIL) and $0D7C_{hex}$ (LB FAIL)

Meaning: The dynamic PCP channel of the specified device is switched on, even though it should be switched off.

Add_Error_Info: Error location (Segment . Position).

$0C7D_{hex}$ (RB FAIL) and $0D7D_{hex}$ (LB FAIL)

Meaning: The dynamic PCP channel of the specified device is switched off, even though it should be switched on.

Add_Error_Info: Error location (Segment . Position).

$0C7E_{hex}$ (RB FAIL) and $0D7E_{hex}$ (LB FAIL)

Meaning: The INTERBUS protocol chip in the specified device cannot be operated in the specified configuration.

Cause: The INTERBUS protocol chip does not support the necessary functions.

Remedy: Replace the device.

Add_Error_Info: Error location (Segment . Position).

$0C80_{hex}$ to $0C83_{hex}$ (RB FAIL) and $0D80_{hex}$ to $0D83_{hex}$ (LB FAIL)

Meaning: Multiple error at the outgoing bus interface (OUT1) of the specified INTERBUS device

Cause: Fault on the bus cable connected to this bus interface, of the following INTERBUS device, or of a device of any subsequent local bus.

Remedy: Check this part of the system for:

- Missing or incorrect shielding of the bus cables (connectors)
- Missing or incorrect grounding/equipotential bonding
- Poor connections in the connector (loose contact, cold junction)
- Voltage dips on the communications power for remote bus devices
- Faulty fiber optic assembly

Add_Error_Info: Error location (Segment . Position).

$0C84_{hex}$ to $0C87_{hex}$ (RB FAIL) and $0D84_{hex}$ to $0D87_{hex}$ (LB FAIL)

Meaning: Multiple timeout at the outgoing bus interface (OUT1) of the specified INTERBUS device.

Cause: Fault
- On the bus cable connected to this bus interface
- On the following INTERBUS device
- On a device in any following local bus

Remedy: Check this part of the system for:
- Missing or incorrect shielding of the bus cables (connectors)
- Missing or incorrect grounding/equipotential bonding
- Poor connections in the connector (loose contact, cold junction)
- Voltage dips on the communications power for remote bus devices
- Faulty fiber optic assembly

Add_Error_Info: Error location (Segment . Position).

$0C88_{hex}$ to $0C8B_{hex}$ (RB FAIL) and
$0D88_{hex}$ to $0D8B_{hex}$ (LB FAIL)

Meaning: An unexpected device was detected at the outgoing bus interface (OUT1) of the specified INTERBUS device.

Cause:
- An INTERBUS device is connected that has not been entered in the active configuration.
- An INTERBUS cable is connected without any further INTERBUS devices.

Remedy: Check the bus configuration.

Add_Error_Info: Error location (Segment . Position).

$0C8C_{hex}$ to $0C8F_{hex}$ (RB FAIL) or
$0D8C_{hex}$ to $0D8F_{hex}$ (LB FAIL)

Meaning: Only ID cycles but not data cycles can be run.

Cause:
- Interrupted data register of the INTERBUS device connected to the outgoing remote bus interface (OUT1) of the specified INTERBUS device.
- The number of data registers for the INTERBUS device connected to the outgoing remote bus interface (OUT1) of the specified INTERBUS device is not identical to the length code.

Add_Error_Info: Error location (Segment . Position).

$0C90_{hex}$ to $0C93_{hex}$ (RB FAIL)

Meaning: The specified INTERBUS device could not activate the following bus segment.

Cause: The INTERBUS device connected to the outgoing interface (OUT1) of the specified INTERBUS device executed a voltage reset or is faulty.

Remedy:
- Check this INTERBUS device.
- Check the supply voltage of this INTERBUS device to determine whether it conforms to the nominal value and whether the permissible AC voltage component is exceeded. Refer to the relevant data sheet for the values.
- Check whether the power supply unit for the bus terminal module is overloaded. Refer to the relevant data sheets for the maximum permissible output current of the bus terminal module and for the typical current consumption of the connected local bus devices.

Add_Error_Info: Error location (Segment . Position).

$0C94_{hex}$ to $0C97_{hex}$ (RB FAIL)

Meaning: An INTERBUS device with the ID code of a local bus device was found at the outgoing remote bus interface (OUT1) of the specified INTERBUS device.

Remedy: Connect a remote bus device.

Add_Error_Info: Error location (Segment . Position).

$0C98_{hex}$ to $0C9B_{hex}$ (RB FAIL) or
$0D98_{hex}$ to $0D9B_{hex}$ (LB FAIL)

Meaning: The INTERBUS device connected to the outgoing remote bus interface (OUT1) of the specified INTERBUS device has an invalid ID code.

Remedy: Replace this device.

Add_Error_Info: Error location (Segment . Position).

$0CA0_{hex}$

Meaning: The controller board cannot start up the bus configuration.

Cause: An attempt was made to parameterize a bus for isolated disconnection, which does not contain only SUPI 3 OPC devices.

Remedy: Only use devices with SUPI 3.

Only set isolated switching in a dedicated SUPI 3 system.

$0D9C_{hex}$ to $0D9F_{hex}$ (LB FAIL)

Meaning: The local bus connected directly to the controller board consists of more INTERBUS devices than have been entered in the active configuration.

Remedy: Check this local bus.

Add_Error_Info: Error location (Segment . Position).

$0CC0_{hex}$ to $0CC3_{hex}$ (RB FAIL) and
$0DC0_{hex}$ to $0DC3_{hex}$ (LB FAIL)

Meaning: Multiple error at the branching bus interface (OUT2) of the specified INTERBUS device.

Cause:

- An INTERBUS cable is connected to the outgoing bus interface (OUT2) without any further INTERBUS devices.
- A local/remote bus cable is faulty that belongs to the local/remote bus of the specified INTERBUS device.
- A faulty INTERBUS device is connected to the local/remote bus of the specified INTERBUS device.
- Failure of the voltage supply (communications power U_L) for the module electronics made available by the bus terminal module.
- Failure of the voltage supply (communications power U_L) for the bus terminal module.

Remedy: Check this local/remote bus.

Add_Error_Info: Error location (Segment . Position).

$0CC4_{hex}$ to $0CC7_{hex}$ (RB FAIL) and
$0DC4_{hex}$ to $0DC7_{hex}$ (LB FAIL)

Meaning: Multiple timeout at the branching bus interface (OUT2) of the specified INTERBUS device.

Cause:
- A local/remote bus cable is faulty that belongs to the local/remote bus of the specified INTERBUS device.
- A faulty INTERBUS device is connected to the local/remote bus of the specified INTERBUS device.
- Failure of the voltage supply (communications power U_L) for the module electronics made available by the bus terminal module.
- Failure of the voltage supply (communications power U_L) for the bus terminal module.

Remedy: Check this local/remote bus.

Add_Error_Info: Error location (Segment . Position).

$0CC8_{hex}$ to $0CCB_{hex}$ (RB FAIL) and
$0DC8_{hex}$ to $0DCB_{hex}$ (LB FAIL)

Meaning: An unexpected device was detected at the branching bus interface (OUT2) of the specified INTERBUS device.

Cause:
- An INTERBUS device is connected that has not been entered in the active configuration.
- An INTERBUS cable is connected without any further INTERBUS devices.

Remedy: Check your INTERBUS network at the specified error location.

Add_Error_Info: Error location (Segment . Position).

$0CCC_{hex}$ to $0CCF_{hex}$ (RB FAIL) and
$0DCC_{hex}$ to $0DCF_{hex}$ (LB FAIL)

Meaning: The bus can be read but not started up. Only ID cycles but not data cycles can be run.

Cause:
1. The data register of the INTERBUS device connected to OUT2 is interrupted.
2. The number of data registers for the INTERBUS device connected to the branching bus interface (OUT2) of the specified INTERBUS device is not identical to the length code entered in the configuration frame.

Remedy:
1. Replace the INTERBUS device, which is connected to the branching bus interface (OUT2) of the specified INTERBUS device.
2. Adapt the entry for the length code in the configuration frame.

Add_Error_Info: Error location (Segment . Position).

$0CD0_{hex}$ to $0CD3_{hex}$ (RB FAIL) and
$0DD0_{hex}$ to $0DD3_{hex}$ (LB FAIL)

Meaning: After opening the branching bus interface (OUT2) of the specified INTERBUS device, further INTERBUS devices, in addition to a bus terminal module, were added to the data ring.

Cause: The INTERBUS device connected to the branching bus interface (OUT2) of the specified INTERBUS device executed a reset or is faulty.

Remedy:
- Check this INTERBUS device.
- Check the supply voltage of this INTERBUS device to determine whether it conforms to the nominal value and whether the permissible AC voltage component is exceeded. Refer to the relevant data sheet for the values.
- Check whether the power supply unit for the bus terminal module is overloaded. Refer to the relevant data sheets for the maximum permissible current output of the bus terminal module and for the typical current consumption of the connected local bus devices.

Add_Error_Info: Error location (Segment . Position).

$0CD4_{hex}$ to $0CD7_{hex}$ (RB FAIL) and
$0DD4_{hex}$ to $0DD7_{hex}$ (LB FAIL)

Meaning: Local bus error in the 8-wire local bus connected to the specified INTERBUS device.

Cause:
- A local bus cable is faulty that belongs to the local bus of the specified INTERBUS device.
- A faulty INTERBUS device is connected to the local bus of the specified INTERBUS device.
- Failure of the voltage supply (communications power U_L) for the module electronics made available by the bus terminal module.

Remedy: Check this local bus.

Add_Error_Info: Error location (Segment . Position).

$0CD8_{hex}$ to $0CDB_{hex}$ (RB FAIL) and
$0DD8_{hex}$ to $0DDB_{hex}$ (LB FAIL)

Meaning: The local bus connected to the specified bus terminal module consists of more local bus devices than were entered in the active configuration.

Add_Error_Info: Error location (Segment . Position).

$0CDC_{hex}$ to $0CDF_{hex}$ (RB FAIL) and
$0DDC_{hex}$ to $0DDF_{hex}$ (LB FAIL)

Meaning: The INTERBUS device connected to the branching bus interface (OUT2) of the specified INTERBUS device has an invalid ID code.

Add_Error_Info: Error location (Segment . Position).

8040_{hex}

Meaning: The specified INTERBUS device is indicating a single channel error.

Remedy: Check the channel of the specified INTERBUS device.

Add_Error_Info: INTERBUS device number (Segment . Position: Channel).

8060_{hex}

Meaning: The specified INTERBUS device is indicating a short circuit at the output.

Remedy: Check the output protective circuit of the specified INTERBUS device.

Add_Error_Info: INTERBUS device number (Segment . Position: Channel).

8080_{hex}

Meaning: The specified INTERBUS device indicates an initiator supply error in one or more groups.

Remedy: Check the group(s) of the specified INTERBUS device.

Add_Error_Info: INTERBUS device number (Segment . Position).

$80A0_{hex}$

Meaning: The specified INTERBUS device indicates a voltage supply error in one or more groups.

Remedy: Check the group(s) of the specified INTERBUS device.

Add_Error_Info: INTERBUS device number (Segment . Position).

$80B1_{hex}$

Meaning: The specified INTERBUS device indicates a configuration error.

Remedy: Check the parameters of the specified INTERBUS device.

Add_Error_Info: INTERBUS device number (Segment . Position).

$80B2_{hex}$

Meaning: The specified INTERBUS device indicates an error in the I/O electronics.

Remedy: Check the sensors and actuators connected to the specified INTERBUS device.

Add_Error_Info: INTERBUS device number (Segment . Position).

$80B4_{hex}$

Meaning: The specified INTERBUS device indicates that the temperature has been exceeded.

Remedy: Check the sensors and actuators connected to the specified INTERBUS device.

Add_Error_Info: INTERBUS device number (Segment . Position).

8400_{hex}

Meaning: The specified INTERBUS device indicates an error at the outgoing interface (OUT2).

Cause: There is an error in the lower-level installation local bus.

Remedy: Check the modules connected to the specified INTERBUS device. You can find the installation local bus device, which is causing the error, by counting the specified number of devices beginning with the last module.

Add_Error_Info: INTERBUS device number (Segment . Position).

3.4 Error Codes for System Errors

$0Fxx_{hex}$ (xx = any digit or letter)

Meaning: A system error (e.g., hardware or firmware error) occurred on the controller board.

Remedy: Replace the controller board.

$0FA4_{hex}$ (CTRL)

Meaning: Writing the parameterization memory was aborted.

Cause: A checksum error was detected on the parameterization memory.

Remedy: Format the parameterization memory with a firmware Version ≥ 472.

$0FC3_{hex}$ (CTRL)

Meaning: Incompatibility with the FB 14 function block when acknowledging peripheral faults (PF) automatically with a firmware Version ≤ 4.15

1010_{hex}

Cause:
1. CPU is not plugged in.
2. A firmware error occurred on the controller board.

Remedy:
1. Operate your controller board with CPU.
2. Replace the controller board.

1011_{hex}

Meaning: A hardware error occurred on the controller board.

Remedy: Replace the controller board.

1012_{hex}

Meaning: An error occurred in the boot firmware.

Cause:
1. There is an error in the firmware.
2. The download of the new firmware failed during the firmware update.
3. A hardware error has occurred.

Remedy:
1. Download the firmware again. If the error message is still generated, a hardware error has occurred. Please contact Phoenix Contact.
2. Repeat the download.
3. Replace the controller board.

1013_{hex}

Meaning: A firmware error occurred.

Cause:
1. There is an error in the firmware.
2. The download of the new firmware failed during the firmware update.
3. A hardware error has occurred.

Remedy:
1. Download the firmware again. If the error message is still generated, a hardware error has occurred. Please contact Phoenix Contact.
2. Repeat the download.
3. Replace the controller board.

1019_{hex} to $101E_{hex}$ (MPM Manager Error)

Meaning: An error occurred on the controller board when the MPM was accessed.

Remedy: Replace the controller board.

1020_{hex} to 1025_{hex} (Flash EEPROM Error)

Meaning: An error occurred on the controller board when the parameterization memory was accessed.

Remedy: Replace the controller board.

1030_{hex} to 1036_{hex} (Power on Selftest Error)

Cause:
1. The PLC is running when the controller board is reset.
2. A hardware error occurred on the controller board during the selftest.

Remedy:
1. Stop the PLC before resetting the controller board.
2. Replace the controller board.

1051_{hex} to 1055_{hex} (RS-232 Error)

Meaning: An error occurred on the controller board when the diagnostic interface was accessed.

Remedy: Replace the controller board.

1056_{hex}

Meaning: The firmware download was aborted.

Cause: Timeout in the transmission protocol or at the diagnostic interface.

Remedy: Check the diagnostic interface and RS-232 cable connection, and then start transmission again.

1057_{hex}

Meaning: The firmware download was aborted.

Cause: The program initiated a restart.

Remedy: Repeat the firmware download.

1101_{hex} (Host Adaptation Error)

Meaning: An error occurred in the host-specific part of the firmware on the controller board.

Remedy: Replace the controller board.

6342_{hex} (Bus Error Indication)

Meaning:	The controller board has detected an error in the connected bus system during INTERBUS operation.
Cause:	The bus cannot be operated any longer. The controller board is searching for the error location.
Remedy:	Read the error causes with the "Get_Error_Info" service (0316_{hex}).

3.5 Error Codes for Controller Boards with Slave Part

1210_{hex}

Meaning: A firmware error occurred.

Cause: There is a faulty device in the INTERBUS ring.

Remedy: Replace the device.

Add_Error_Info: Number of faulty devices.

1211_{hex}

Meaning: Slave type is set incorrectly.

Cause:
1. Slave number set is not permitted.
2. A hardware error has occurred.

Remedy:
1. Set the correct slave number (0 or 1).
2. Replace the device.

Add_Error_Info: Incorrect slave number.

1212_{hex}

Meaning: A hardware error occurred.

Cause: There is a device with an incorrect protocol chip in the INTERBUS ring. Only devices with protocol chip SUPI 3 and higher are permitted.

Remedy: Remove the device with the incorrect protocol chip.

1213_{hex}

Cause: The slave was initialized with an illegal number of words.

Remedy: Reduce the number of words and initialize the slave again.

1215_{hex}

Meaning: Error in microprocessor mode

Cause:
1. Data lengths have been used that are not permitted in microprocessor mode (e.g., zero words).
2. The slave part of the system coupler does not have the correct ID code.

Remedy:
1. Remove the external supply voltage of the system coupler briefly to initialize the slave part of the system coupler again.
2. Enter a correct ID code for the slave part.

1217_{hex}

Meaning: Initialization error

Cause: The INTERBUS protocol chip has been initialized more than once. The INTERBUS protocol chip SUPI 3 can only be initialized once if the *NOT-READY* ID code is present in the hardware. The SUPI 3 can be initialized more than once if a different ID code is present.

Remedy:
– Ensure that the protocol chip is initialized only once.
– Disconnect the INTERBUS protocol chip from the supply, and then try again.

3.6 Error Codes for Controller Boards with COP Board

Error Codes for User Errors

1402_{hex}

Meaning: The coprocessor board could not process the service called last.

Cause: A status conflict occurred in the boot loader.
For example, the "PC104_Download_Open_File_Request" service (0291_{hex}) or the "PC104_Download_Terminate_Request" service (0294_{hex}) was sent when a file was open or the "PC104_Download_Send_File_Request" service (0292_{hex}) or the "PC104_Download_Close_File_Request" service (0293_{hex}) was sent when a file was not open.

Remedy: Check the last and previous service calls.

Add_Error_Info: Service for which the status conflict occurred:
0001_{hex} "PC104_Download_Initiate_Request" (0290_{hex})
0002_{hex} "PC104_Download_Open_File_Request" (0291_{hex})
0003_{hex} "PC104_Download_Send_File_Request" (0292_{hex})
0004_{hex} "PC104_Download_Close_File_Request" (0293_{hex})
0005_{hex} "PC104_Download_Terminate_Request" (0294_{hex})

1410_{hex}

Meaning: When a file was opened using the "PC104_Download_Open_File_Request" service (0291_{hex}), an error occurred.

Add_Error_Info: Indication of the file in which the error occurred:
0001_{hex} "bootld.ini" file
0002_{hex} Other file

1411_{hex}

Meaning: When a file was written using the "PC104_Download_Send_File_Request" service (0292_{hex}), an error occurred.

Add_Error_Info: Always 0001_{hex}

1412_{hex}

Meaning: When a file was closed using the "PC104_Download_Close_File_Request" service (0293_{hex}), an error occurred.

Add_Error_Info: Indication of the file in which the error occurred:
0001_{hex} "bootld.ini" file
0002_{hex} Other file

1413_{hex}

Meaning: An error occurred when opening the "bootld.ini" file.

Add_Error_Info: Always 0001_{hex}

1414_{hex}

Meaning: You tried to open a file using the "PC104_Download_Open_File_Request" service (0291_{hex}), which is invalid. During a firmware download, the "bootld.ini" file must always be opened first. This file must contain the names of all of the other files to be opened.

Remedy: Check the sequence of the files for the download and the entries in the "bootld.ini" file.

Add_Error_Info: 0001_{hex} First file is not the "bootld.ini" file
0002_{hex} File is a system file
0003_{hex} File name not entered in the "bootld.ini" file

1420_{hex}

Meaning: The firmware download was aborted.

Cause: The download was aborted using the button on the PC keyboard.

Add_Error_Info: Abort position in the boot loader:
0001_{hex} or 0002_{hex}

1421_{hex}

Meaning: The firmware download was aborted.

Cause: An error occurred during the firmware download.

Add_Error_Info: Always 0000_{hex}

1422_{hex}

Meaning: The boot process was aborted.

Cause: The board does not contain the main firmware.

Add_Error_Info: Always 0000_{hex}

1430_{hex}

Meaning: The checksum check was aborted.

Cause: A file could not be opened during the checksum check.

Add_Error_Info: Position in the firmware
0001_{hex} or 0002_{hex}

1431_{hex}

Meaning: The checksum check was aborted.

Cause: An error occurred when reading a file during the checksum check.

Add_Error_Info: Always 0001_{hex}

1433_{hex}

Meaning: The checksum for the boot firmware (DOS + boot loader) is incorrect.

Add_Error_Info: Correct checksum

1434_{hex}

Meaning: The checksum for the main firmware is incorrect.

Add_Error_Info: Correct checksum

1435_{hex}

Meaning: The boot process was aborted.

Cause: No main firmware was found during the checksum check.

Add_Error_Info: Always 0001_{hex}

2002_{hex}

Meaning: The coprocessor board could not process the service called last.

Cause: A state conflict occurred in the firmware.
For example, the "PC104_Download_Open_File_Request" service (0291_{hex}) or the "PC104_Download_Terminate_Request" service (0294_{hex}) was sent when a file was open or the "PC104_Download_Send_File_Request" service (0292_{hex}) or the "PC104_Download_Close_File_Request" service (0293_{hex}) was sent when a file was not open.

Remedy: Check the last and previous service calls.

Add_Error_Info: Service for which the status conflict occurred:
0001_{hex} "PC104_Download_Initiate_Request" (0290_{hex})
0002_{hex} "PC104_Download_Open_File_Request" (0291_{hex})
0003_{hex} "PC104_Download_Send_File_Request" (0292_{hex})
0004_{hex} "PC104_Download_Close_File_Request" (0293_{hex})
0005_{hex} "PC104_Download_Terminate_Request" (0294_{hex})

2010_{hex}

Meaning: When a file was opened using the "PC104_Download_Open_File_Request" service (0291_{hex}), an error occurred.

Add_Error_Info: Indication of the file in which the error occurred:
0001_{hex} "bootld.ini" file
0002_{hex} Other file

2011_{hex}

Meaning: When a file was written using the "PC104_Download_Send_File_Request" service (0292_{hex}) or the "PC104_File_Transfer_Write_Request" service ($02B6_{hex}$), an error occurred.

Add_Error_Info: For an error in the "PC104_Download_Send_File_Request" service (0292_{hex}), always 0001_{hex}.
For an error in the "PC104_File_Transfer_Write_Request" service ($02B6_{hex}$), the error code of the relevant file system driver.

2012_{hex}

Meaning: When a file was closed using the "PC104_Download_Close_File_Request" service (0293_{hex}), an error occurred.

Add_Error_Info: Indication of the file in which the error occurred:
0001_{hex} "bootld.ini" file
0002_{hex} Other file

2013_{hex}

Meaning: An error occurred when opening the "bootld.ini" file.

Add_Error_Info: Always 0001_{hex}

2014_{hex}

Meaning: You tried to open a file using the "PC104_Download_Open_File_Request" (0291_{hex}) service, which is invalid. During a firmware download, the "bootld.ini" file must always be opened first. This file must contain the names of all of the other files to be opened.

Remedy: Check the sequence of the files for the download and the entries in the "bootld.ini" file.

Add_Error_Info: 0001_{hex} First file is not the "bootld.ini" file.
0002_{hex} File is a system file
0003_{hex} File name not entered in the "bootld.ini" file

2015_{hex}

Meaning: Error when deleting a file, e.g., when deleting the boot project.

2020_{hex}

Meaning: You used an unknown message code.

Add_Error_Info: Unknown message code

2021_{hex}

Meaning: Unauthorized access.

Cause: For example, accessing a protected variable with an incorrect password.

Remedy: Use the correct password.

2022_{hex}

Meaning: An error occurred when writing to an INI file.

Cause: You tried to delete a variable, which is not available.

2023_{hex}

Meaning: An error occurred when reading from an INI file.

Cause: You tried to read a variable, which is not available.

2024_{hex}

Meaning: An error occurred when opening a file.

Add_Error_Info: Error code of the relevant file system driver.

2025_{hex}

Meaning: An error occurred when reading a file.

Add_Error_Info: Error code of the relevant file system driver.

2026_{hex}

Meaning: An error occurred when closing a file.

Add_Error_Info: Error code of the relevant file system driver.

2027_{hex}

Meaning: An invalid file handle was detected for the file transfer for a service.

2028_{hex}

Meaning: An invalid communication reference was used for the "PCP_Read_With_Name_Request" service (0098_{hex}) or the "PCP_Write_With_Name_Request" service (0097_{hex}).

Add_Error_Info: Incorrect communication reference.

2029_{hex}

Meaning: An error occurred during the "PC104_File_Transfer_Ioctl_Request" service ($02B8_{hex}$).

Add_Error_Info: Error code of the ioctl() VxWorks function

$202A_{hex}$

Meaning: For the "PC104_File_Transfer_Ioctl_Request" service ($02B8_{hex}$), the number of subsequent bytes (*No_of_Bytes* parameter) is too great.

Add_Error_Info: *No_of_Bytes* parameter.

$202B_{hex}$

Meaning: The "PC104_File_Transfer_Ioctl_Request" service ($02B8_{hex}$) contains a function that is not permitted in the *Function* parameter.

Add_Error_Info: *Function* parameter.

$202C_{hex}$

Meaning: A file could not be opened with the "PC104_File_Transfer_Open_Request" service ($02B4_{hex}$).

Cause: The maximum number of open files has been reached.

Remedy: Close at least one of the open files.

$202D_{hex}$

Meaning: Incorrect entry in the SVC file.

Cause: Neither the keyword "CMD" nor a hexadecimal number (0xXXXX) appears between two '#' signs .

Remedy: Check the SVC file.

Add_Error_Info: Line number with incorrect entry.

$202E_{hex}$

Meaning: Incorrect entry in the SVC file.

Cause: The SVC has not been logically created.

Remedy: Check the SVC file.

Add_Error_Info: Line number with incorrect entry.

$202F_{hex}$

Meaning: The maximum number of parameters was exceeded for a service in an SVC file.

Remedy: Check the SVC file and reduce the number of parameters.

Add_Error_Info: Line number with incorrect entry.

2030_{hex}

Meaning: A negative service confirmation was received.

Error Codes for Warnings

2111_{hex}

Meaning: The battery supply has failed. In the event of a power failure, the date and time of the realtime clock as well as the retain data may be lost.

Remedy: Replace the battery.

2112_{hex}

Meaning: The IP address has not yet been parameterized or equals "0.0.0.0".

Remedy: Enter an IP address.

2113_{hex}

Meaning: The "vxwusr.ini" file is faulty.

Cause: Examples are:

– Error in the checksum
– No checksum available

Error Codes for Runtime Errors

Errors, which occur during the runtime of the IEC -61131 runtime system are described in error category $22xx_{hex}$.

2211_{hex}

Meaning: Stack overflow in a task for the IEC 61131 runtime system.

Remedy: Increase the stack.

2212_{hex}

Meaning: In a user task for the IEC 61131 runtime system, the memory area was exceeded in an array.

Remedy: Check the access to the array and increase the dimensions of the array if necessary.

2214_{hex}

Meaning: A division by zero was made in a user task for the IEC 61131 runtime system.

Remedy: Check the user task.

2216_{hex}

Meaning: During a floating point calculation in a user task for the IEC 61131 runtime system, the value range was exceeded.

Remedy: Check the user task.

2217_{hex}

Meaning: The task watchdog of a task for the IEC 61131 runtime system is launched because the execution time is too long.

Remedy: Check the task for the IEC 61131 runtime system.

2218_{hex}

Meaning: The runtime of a task for the IEC 61131 runtime system is too long. Tasks with lower priority (e.g., communication tasks) no longer receive any processor time.

Remedy: Check the task.

2219_{hex}

Meaning: You called a function block, which does not exist in the firmware or in the IEC 61131 runtime system.

$221A_{hex}$

Meaning: One program in the IEC 61131 runtime system was stopped from the program.

$221B_{hex}$

Meaning: An unexpected breakpoint occurred.

$221C_{hex}$

Meaning: An internal exception occurred.

$221D_{hex}$

Meaning: A string error occurred in the program.

$221E_{hex}$

Meaning:
- A division by zero was made in a user task for the IEC 61131 runtime system.
- The task watchdog of a task for the IEC 61131 runtime system is launched because the execution time is too long.

Remedy: Check the task.

Error Codes for Fatal Errors

Fatal errors are described in error categories $23xx_{hex}$, $24xx_{hex}$, and $26xx_{hex}$. If fatal errors occur frequently, please contact Phoenix Contact.

Error category $23xx_{hex}$ contains errors, which occur in the DDI (Device Driver Interface). The low-order byte of this error code is the error code of the DDI. For information on the error codes of the DDI, please refer to the Driver Reference Manual for PC Controller Boards (Order No. 27 45 17 2).

Error category $24xx_{hex}$ contains fatal errors, which occur in the firmware.

2410_{hex}

Meaning: The software watchdog for the COP has been triggered.

2411_{hex}

Meaning: An error occurred when initializing the IEC 61131 runtime system.

2412_{hex}

Meaning: An error occurred when initializing the serial interface COM1 for the IEC 61131 runtime system.

2413_{hex}

Meaning: An error occurred when initializing the serial interface COM2 for the IEC 61131 runtime system.

2414_{hex}

Meaning: An error occurred when initializing the Ethernet connection for the IEC 61131 runtime system.

2415_{hex}

Meaning: An error occurred when initializing communication between the IEC 61131 runtime system and the message handler.

2416_{hex}

Meaning: An error occurred when initializing the IEC 61131 runtime system driver "IBS_IO".

2417_{hex}

Meaning: An error occurred when initializing the IEC 61131 runtime system driver "PSTD_IO".

2418_{hex}

Meaning: An error occurred when initializing the IEC 61131 runtime system exception handler.

2419_{hex}

Meaning: No memory is available for the system flags for the IEC 61131 runtime system.

$241A_{hex}$

Meaning: An error occurred when registering the memory for the system flags on the IEC 61131 runtime system.

2420_{hex}

Meaning: For devices on which the flash memory must be partitioned into two drives, either the drives are of the wrong size or the second drive is not available.

2421_{hex}

Meaning: The "vxworks.ini" file is faulty.

2422_{hex}

Meaning: Error in the IB loader.

Cause: The specified SVC file cannot be read.

2423_{hex}

Meaning: Error in the IB loader.

Cause: The timeout time has elapsed and no service confirmation was received.

2425_{hex}

Meaning: Error in the IB loader.

Cause: An service indication was received when a service confirmation was expected.

2426_{hex}

Meaning: Error in the IB loader.

Cause: While waiting for a service confirmation, an invalid message was received (neither a service message nor a service confirmation).

2427_{hex}

Meaning: Error in the IB loader.

Cause: The timeout elapsed before the busy flag was set by the IEC 61131 runtime system.

2428_{hex}

Meaning:	Error in the IB loader.
Cause:	An error occurred when opening a data channel to a node. For more information, please refer to the section on the DDI_DevOpenNode() DDI routine in the Driver Reference Manual for PC Controller Boards (Order No. 27 45 17 2).
Add_Error_Info	Error code for the DDI_DevOpenNode() DDI routine.

2429_{hex}

Meaning:	Error in the IB loader.
Cause:	An error occurred while sending a message or command to a mailbox. For more information, please refer to the section on the DDI_MXI_SndMessage() DDI routine in the Driver Reference Manual for PC Controller Boards (Order No. 27 45 17 2).
Add_Error_Info	Error code for the DDI_MXI_SndMessage() DDI routine.

$242A_{hex}$

Meaning:	Error in the IB loader.
Cause:	An error occurred while receiving a message or command from a mailbox. For more information, please refer to the section on the DDI_MXI_RcvMessage() DDI routine in the Driver Reference Manual for PC Controller Boards (Order No. 27 45 17 2).
Add_Error_Info	Error code for the DDI_MXI_RcvMessage() DDI routine.

$242B_{hex}$

Meaning:	Error in the IB loader.
Cause:	An error occurred while activating the notification mode for a data channel. For more information, please refer to the section on the DDI_SetMsgNotification() DDI routine in the Driver Reference Manual for PC Controller Boards (Order No. 27 45 17 2).
Add_Error_Info	Error code for the DDI_SetMsgNotification() DDI routine.

$26xx_{hex}$

Meaning: An exception occurred at the processor. The low-order byte contains the vector number of the exception.

A 1 List of Figures

Section 1

Section 3

A 2 List of Tables

Section 1

Section 2

A 3 Definition of Abbreviations

CR	Communication Reference
CRC	Cyclic Redundancy Check
CRL	Communication Relationship List
DDI	Device-Driver-Interface
DPM	Dual-Port-Memory
DTA	Data-Transmission-Area
DTI	Data-Transmission-Interface
G4	Generation 4
IB, IBS	INTERBUS
IN-PD	IN Process Data
MAU	Medium Attachment Unit
MMS	Manufacturing Message Specification
MPM	Multi-Port-Memory
MXA	Mailbox Area
MXI	Mailbox-Interface
LB	Local Bus
OD	Object Dictionary
OUT-PD	OUT Process Data
PCP	Peripheral Communication Protocol
PD	Process Data

PDD	Process Data Description
PDDL	Process Data Description List
PDR	Process Data Reference
PDRL	Process Data Reference List
RB	Remote Bus
SGA	Signal Area
SGI	Signal Interface
SSGI	Standard-Signal-Interface
XSGI	Extended Signal Interface

A 4 Glossary

Action object
An action object is either a service configured using parameters or a predefined service sequence.

Alternative group
An alternative group is a part of the configuration that can be connected as an alternative to a individual "BT module". It is entered in the low byte of the "group number". (See also "group")

Bus segment
A bus segment consists of a "remote bus" device and the "I/O module"s connected to it. The preceding cable is also part of the segment.

Client
A client is a communication device that requests a service from a "server".

Client-server model
This model defines the communication mechanisms between a "service requester" ("client") and a "service provider" ("server"). With these communication mechanisms the client can use the functions of the server. There are "communication service"s to access the functions of the server.

Communication reference
The communication reference is a number that is assigned to each "PCP device". It designates the address of the logical connection. The INTERBUS "controller board" always has the communication reference 1. The user can assign the communication references in an ascending order beginning with 2.

Communication relationship
With PCP communication, the communication relationship establishes the logical connection between two devices. The requirement for this is the physical possibility of communication, i.e. both devices must be connected to each other via the network and be "capable of PCP". The "communication relationship list" contains information for each communication relationship.

Communication relationship list
A communication relationship list is a list for PCP communication in which the connection parameters of the "communication relationship" between two devices are stored. During connection establishment, a compatibility check of the connection parameters in the CRLs of both devices is made. The relevant connection parameters are the transmit and receive buffer sizes as well as the supported "PCP service"s. Instead of "connection parameters", one also speaks of suitable ""context conditions"".

The communication relationship list of a device contains the description of all communication relationships of this device regardless of when they are used.

Configuration frame The configuration frame contains the entire configuration of the "controller board" including all "alternative group"s. The configuration frame includes all devices of the "complete bus configuration".

Confirmation A confirmation is a reply of the "server" to a "request" of the "client". The server sends the confirmation as a "response".

Controller board The host controller board connects programmable logic controllers ("PLC"s) or computer systems (PCs, VMEbus systems etc.) to the INTERBUS system. The controller board takes over the master function and controls the "data traffic" in the INTERBUS system, independent of the "control or computer system" in which it is installed.

Current bus configuration The current "bus configuration" is the physical bus configuration that is currently operated by the "controller board".

Device code The device code is a data word to identify the characteristics of an INTERBUS device. It consists of the "length code" (high byte) and "ID code" (low byte).

Device number With "INTERBUS", there are "logical device number"s and "physical device number"s.

Device number, logical Each "INTERBUS device" of a configuration frame is assigned a unique device list. This device number is specified in the form "Segment.Position" (Seg.Pos). The logical device number 0.0 is reserved for the "controller board". The numbers "1.0" to "254.254" can be assigned. Each remote bus device receives the "position number" 0. Each local bus device receives the "segment number" of the associated remote bus device.

Device number, physical The physical device number identifies the order of the devices determined by the bus system structure. It is assigned from 1 to 512 in an ascending order without gaps.

Event object An event object is a service sequence configured using parameters that is started using the "indication" service primitive. The indication triggering the start of the event must be determined by the user.

Group

The user can combine INTERBUS-"device"s into groups. Each device may only be assigned to one group. These groups are switched on and off. This switching can be controlled by the application program. When a group is switched off, the "local bus" in which this group appears is always shut down completely. As of firmware Generation 4 even "remote bus" devices can be switched on or off. This no longer requires a group definition since only the logical device number (see "device number, logical") needs to be entered when devices are switched.

Group number

Devices can be combined to a group with the group number. The group number consists of the "group" (high byte) and the "alternative group" (low byte).

Host

Host is the denomination for the "control or computer system" into which the "controller board" is integrated.

ID code

Each INTERBUS device has an ID code (identification code) that is used by the "controller board" to identify the device. The ID code specifies the type of device in the data telegram. It indicates whether it is an analog or digital module or a bus terminal module; whether it is an input or output module; and whether it is a PCP device or a master. It uses the low byte of the "device code".

Indication

The "server" receives an indication for a "request" of the "client". The server responds to an indication with a "response".

IN process data

The IN process data is the part of the input data which is cyclically transmitted from the INTERBUS devices to the INTERBUS controller board.

INTERBUS

INTERBUS is a fieldbus standardized according to IEC 61158 for the serial transmission of data on the sensor/actuator level.

INTERBUS device

An INTERBUS device is part of an "INTERBUS module" and participates in the data exchange via the INTERBUS system. Each INTERBUS device has only one protocol chip. The devices are identified through the "device code". There are also INTERBUS modules that include several devices.

INTERBUS module

All complex technical components that are used for the data transmission over INTERBUS. An INTERBUS module can contain numerous "INTERBUS device"s.

Known configuration

The known configuration is the "INTERBUS" configuration present in the main memory of the "controller board".

Length code

The length code provides the number and type of representation of the "process data" ("bit", "nibble", "byte", "word") by using the high byte of the "device code".

Mailbox area

In the multi-port memory the mailbox area serves as communication platform between a host and a controller board.

OUT process data

The OUT process data is the part of the "output data" which is cyclically transmitted from the "INTERBUS controller board" to the "INTERBUS device"s.

PCP

The Peripheral Communication Protocol (PCP) belongs to the INTERBUS protocol and controls the transmission of "parameter data". Special PCP services are available for this purpose.

PCP object

Structured memory area for data that is exchanged between two devices e.g., measured values, program parts, device parameters, etc. The data is described in the "Object Dictionary" of a device. It can be accessed by other communication devices.

PCP service

A PCP service is a service used to establish and abort a connection as well as in data exchange between two "INTERBUS device"s.

Peripheral Message Specification

PMS is a subset of MMS communication services, especially adapted to the field of sensors/actuators.

Physical device position

The physical device position is the position of the "bus device" in the "summation frame". The first bus device is assigned position "0". The physical device position corresponds to the physical device number if the entire "configuration frame" is active.

Process data

Process data is input and output information sent to and from INTERBUS devices. Process data changes continually and must be continuously updated. This information is transmitted with every bus cycle via the "process data channel" (see also "parameter data").

Protocol

A protocol is a set of conventions for "communication" between devices and processes.

Request

A request is a service call of the "client" at the "server". The client receives a "confirmation" as a reply. The server receives the request as an "indication".

Response

A response is a reply of the "server" to an "indication" of the "client". The client receives the reply as a "confirmation".

Server

A server is a communications device that responds to a service from a "client". The server makes its objects through a service available to other devices.

Slave

A slave is a device in the network which can only participate in the data exchange after it has been addressed by the "master".

Summation frame

The summation frame is a transmission protocol in which all physical "device"s are treated as if they were one logical device. All "process data" is transmitted simultaneously to all devices during a cycle. On the basis of the location of the information in the summation frame, each device can accept the data that is determined for it.

A 5 Index

E

F

G

I

L

M

R

S

U

V

W

We Are Interested in Your Opinion!

We would like to hear your comments and suggestions concerning this document.

We review and consider all comments for inclusion in future documentation.

Please fill out the form on the following page and fax it to us or send your comments, suggestions for improvement, etc. to the following address:

Phoenix Contact GmbH & Co. KG
Marketing Services
Dokumentation INTERBUS
32823 Blomberg
GERMANY

Phone +49 - (0) 52 35 - 30 0
Telefax +49 - (0) 52 35 - 34 18 08
E-Mail tecdoc@phoenixcontact.com

FAX Reply

Phoenix Contact GmbH & Co. KG
Marketing Services
Dokumentation INTERBUS

Date: _____

Fax No: +49 - (0) 52 35 - 3-4 18 08

From:

Company:		Name:	
		Department:	
Address:		Job function:	
City, ZIP code:		Phone:	
Country:		Fax:	

Document:

Designation: IBS SYS FW G4 UM E Revision: D Order No.: 27 45 18 5

My Opinion on the Document

Form	**Yes**	**In part**	**No**
Is the table of contents clearly arranged?	☐	☐	☐
Are the figures/diagrams easy to understand/helpful?	☐	☐	☐
Are the written explanations of the figures adequate?	☐	☐	☐
Does the quality of the figures meet your expectations/needs?	☐	☐	☐
Does the layout of the document allow you to find information easily?	☐	☐	☐
Contents	**Yes**	**In part**	**No**
Is the phraseology/terminology easy to understand?	☐	☐	☐
Are the index entries easy to understand/helpful?	☐	☐	☐
Are the examples practice-oriented?	☐	☐	☐
Is the document easy to handle?	☐	☐	☐
Is any important information missing? If yes, what?	☐	☐	☐

Other Comments:

SCATTERGOOD & JOHNSON LTD
ELECTRICAL ENGINEERING & FLUID CONTROL DISTRIBUTORS
Est.1899

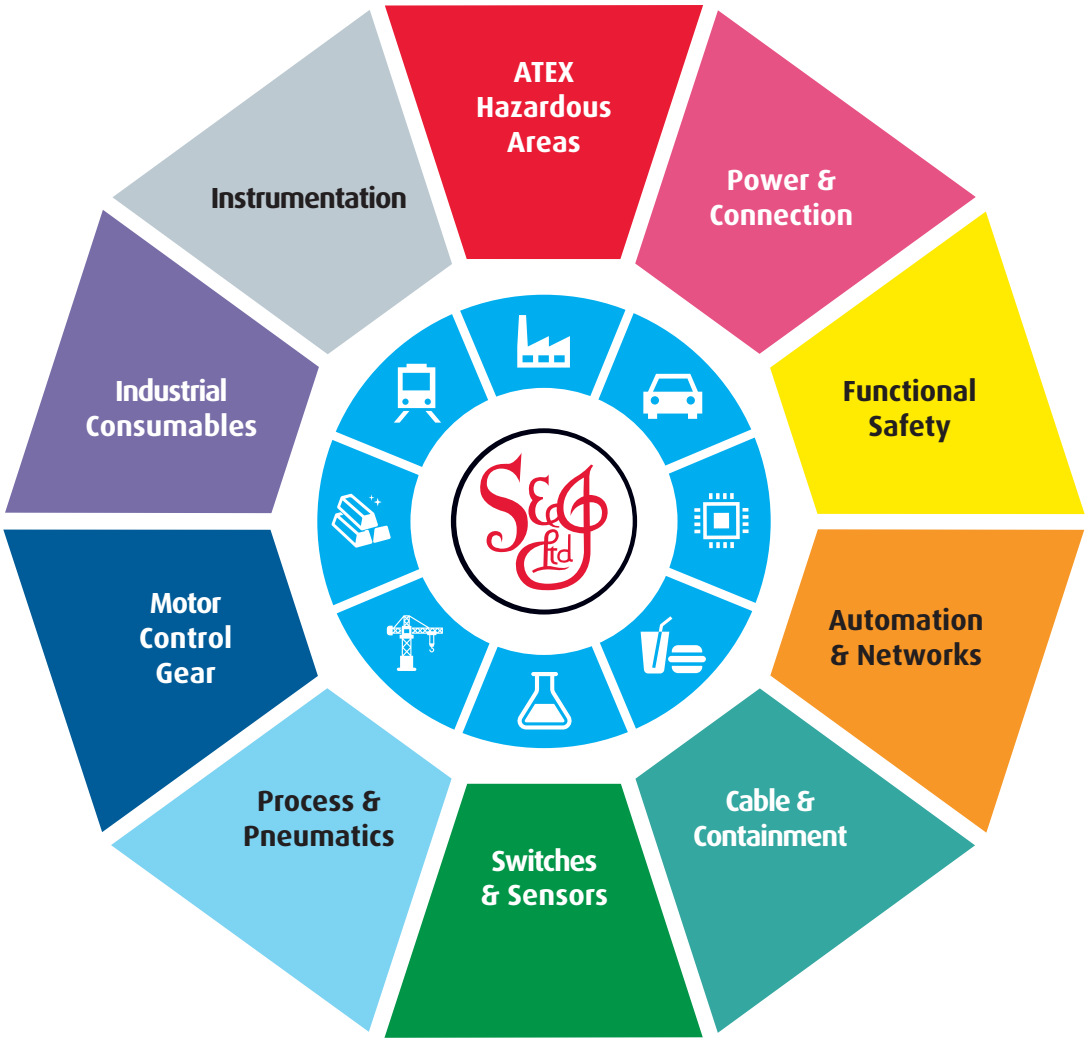
At Scattergood & Johnson Ltd, we pride ourselves on being a technical distributor to specialist industries.

Working with a range of quality product suppliers across a number of specialist markets, we are not your average 'box shifter' - we are your technical and supply chain partner.

We fully support every product we sell - for free! Our internal team and external sales engineers can answer any product or application question, no matter the complexity.

Backing up this technical ability is a range of 50,000+ products available from stock for nationwide next day delivery (same day if required!), or you can collect what you need from any of our trade counters around the UK.

Select your specialist interest below to learn more about how we can help.



Online, In Branch and On the Road - Scattergood & Johnson Ltd, there when you need us.

www.scatts.co.uk