Profile : Sensor/Actuator

Profile No. : 12

Date : March 7, 1995

Published by : INTERBUS-S Club e.V.
Geschäftsstelle
Postfach 1108, D-32817 Blomberg
Telephone: *49-5235- 34 21 00
Fax: *49-5235- 34 12 34

Order No. : 12

Contents Page

Preface

Within the framework of factory automation, increasingly powerful and flexible systems are needed in the field of industrial sensors and actuators. Digitized sensors and actuators can meet these requirements. However, open and standardized communication capabilities are needed to enable their complete integration into complex production sequences.

The basic concept of open systems is to enable an exchange of information between application functions implemented on hardware from a diversity of manufacturers.

These functions include defined application functions, a standard user interface for communications and a standard transmission medium.

The InterBus-S Club aims at standardizing the most important sensor/actuator device functions and to summarize them in this profile for sensors and actuators.

To be able to define the device functions of the sensor/actuator functions independent of the communication medium, an internationally recognized and standardized user interface, DIN 19 245, Part 2, was used for communications. This created compatibility with MMS.

The InterBus-S system, which meets the requirements of sensors and actuators with regard to real-time response and a standardized user interface, was chosen as the communication medium.

The Sensor/Actuator Profile is oriented to the user and manufacturer of sensors and actuators to be operated on the sensor/actuator bus.

For the user, this profile definition is a useful addition to standardized communication and represents a generally valid convention concerning the contents of data and the response of devices. These function definitions standardize a few essential device parameters of sensors and actuators. Consequently, hardware from different manufacturers exhibits the same response in the communication medium when these standard parameters are used.

An independent expert committee will be established to test products, which have been developed in accordance with this profile, for their conformance and to certify these products.

As standardization work is continuing, additions are to be expected.

Author:

Mr. Krumsiek Phoenix Contact, Blomberg, Germany

Introduction

This profile defines the basic functions which every sensor and actuator device must make available to the user. Basically, these functions are information on the use of the process data channel, communication functions, and device information. The Sensor/Actuator Profile constitutes the basis for all devices having server functionality. The Motion Control Profile and the Encoder Profile are already based on the Sensor/Actuator Profile. The indexes of the device parameters of a sensor/actuator device lie in the range of 6000 - 603F hex, i.e. this range cannot be used for the device parameter indexes of other profiles which are based on this Sensor/Actuator Profile. Figure 1 shows the relationship.

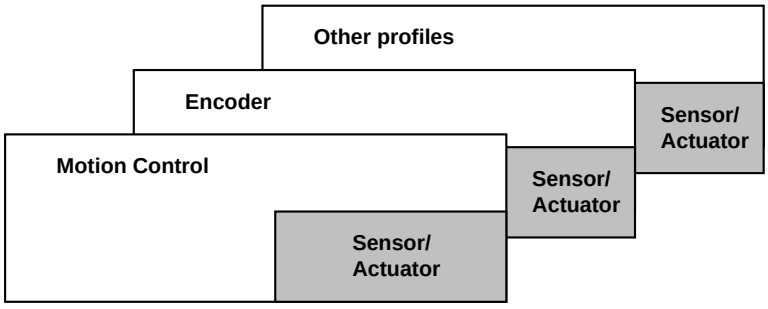

Figure 1: The Sensor/Actuator Profile is the basis profile

Sensor/Actuator Functions

This profile defines the application functions of sensors and actuators. As shown in Figure 2, the application functions are subdivided into device functions, communication functions, and device information. In addition, free areas for the manufacturer-specific functions are defined.

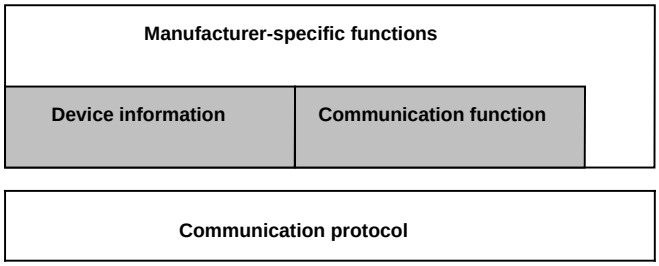

Figure 2: Application functions of sensors and actuators

Each application function is described with the aid of a function block.

Communication Functions

The communication functions handle the service execution and define the error messages.

Device Information

The device information is a function which defines how the generally valid device information is stored in a non-volatile memory.

Manufacturer-Specific Functions

The freely definable, manufacturer-specific functions may utilize standardized functions of all other function areas.

The device user can parameterize this application functions through device parameters for his application task.

With that, these generally valid parameters are defined for all devices which comply with the sensor/actuator profile and all other profiles that are based on this profile.

The parameter number (index) which is used to address these device parameters lies in the range 6000 - 603f hex, i.e. this range is reserved for the device parameters of all profiles which are based on this Sensor/Actuator Profile.

1. Scope of Application

This profile defines the basic functions which can be used in every profile for InterBus-S devices.

2. References

The application protocol and the data structures conform to the InterBus-S Club Guidelines.

The application interface for communication via the InterBus-S parameter channel also conforms to the InterBus-S Club Guidelines.

The definitions for data transfer through the process data channel are based on the InterBus-S Club Guidelines and the draft standard DIN 19258.

3. Terms

Device Profile

The device profile defines the application functions that are visible through communication. The application functions are mapped to the communication by the following definitions:

- by the communication profile,
- by interaction between the application functions, insofar as they are executed through the communication system, and
- by the communication services used and the communication objects that can be manipulated with them.

The result of this mapping is the visible response of the application. The definitions contained in an application profile enable interoperability in a field of application if permitted by the device characteristics used.

Characteristics of devices significant to the user are also defined.

A distinction is made between mandatory functions, optional and manufacturer-specific device functions, and parameters.

If the users restrict themselves to the mandatory functions or parameters, interchangeability of devices is possible if this is permitted by the device characteristics and settings used. With respect to communications, and regardless of the function, devices are always interchangeable if use is made of the same parameters.

Communication Profile

In relation to the specific application or hardware group, the communication profile limits or classifies the degrees of freedom contained in the specification of the data transfer medium. The communication profile defines communication services and parameters that are identified in the specification as being optional.

All optional functions and parameters that are not stated in the communication profile remain optional. Mandatory services and parameters are binding, even if not stated in the profile.

The profile also limits or defines value ranges of attributes and parameters.

The communication medium is InterBus-S.

Communication Interface

The communication interface is composed of a process data channel and a parameter channel.

Process Data Channel

The process data channel serves the purpose of swift transfer of process data. Through the process data channel, data is transferred in unacknowledged and equidistant form. Process data may be read and written.

The direction specified for the process data is viewed from the bus, i.e.,

- process output data is data which is transmitted from the control system to the terminal. The terminal reads this data from the process data channel and outputs it to the process.
- process input data is data that is transmitted from the terminal to the control system. The terminal writes this data to the process data channel and transmits it thus to the control system.

Process Data Channel Description

If the device transmits or receives several process data in parallel, the manufacturer must determine the structure and the meaning of this process data.

Parameter Channel

All communication objects can be accessed trough the parameter channel. The parameter channel services allow a confirmed access to device parameters, i. e., the access to the a device parameter is acknowledged by the device.

VFD Object

The Virtual Field Device (VFD) is an abstract model for describing the data and the behavior of a programmable controller from the point of view of its communication partner. The basis of the VFD model is the VFD object. The VFD object contains all objects and object descriptions that can be used by a communication partner through services. The object descriptions are contained in an object dictionary. There is precisely one object dictionary for each VFD.

Communication Reference

Each communication relationship between two devices is configured, independent of when it will be used. The configuration information is stored in each bus device in a communication relationship list (CRL). An application process identifies the communication relationship via a local communication reference. Thus, the communication reference is used for addressing the communication partner.

Error Message

The error message is returned when a service could not be executed.

Index, Subindex

The index is used to address a parameter (communication object). The subindex addresses a subparameter (element of a communication object) within a parameter created as a structure.

Device Parameter

This profile contains default values for all device parameters.

Substitute Values

When the optional communication objects are not implemented, the device responds according to the substitute value defined for this parameter.

Mandatory Range

The mandatory area is the range of values where a parameter, if implemented, can be parameterized in any case.

State Machine

Some functions are described in this profile with the aid of a state machine. A state represents a specific internal and external response. It can only be terminated by means of defined events. Corresponding state transitions are assigned to events. Actions can be executed at a transition. The response of the state is changed at the transition. When the transition is ended, the current state is followed by the new state.

Definition of the Functional Units

The device function is described with a functional unit (see Figure 3). The function is controlled and parameterized through the inputs. In addition, internal signals or parameters can influence the function. The output of the function can be connected to the inputs of other functions or made accessible via the bus.

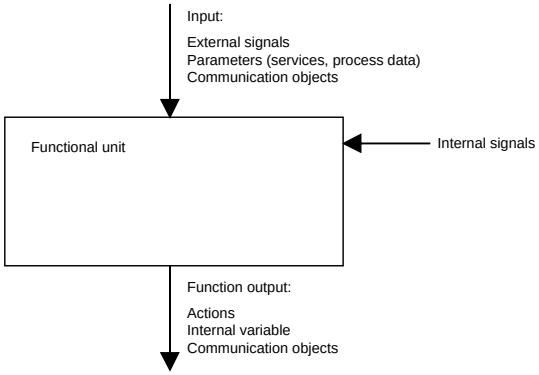

Figure 3: Functional unit

4. Abbreviations

Network-specific abbreviations

ALI	Application Layer Interface
IB-S	InterBus-S
CR	Communication reference
CRL	Communication relationship list
CRL header	Header of the communication relationship list
ID code	Identification code
m	Mandatory
MAP	Manufacturing Automation Protocol
o	Optional
OD	Object dictionary
PCP	Peripherals Communication Protocol
PA channel	Parameter channel
PD channel	Process data channel
PD-OD	Process data object dictionary
PMS	Peripherals Message Specification
S-OD	Static object dictionary
ST-OD	Static type dictionary
VFD	Virtual field device
.con	Confirmation primitive
.ind	Indication primitive
.req	Request primitive
.res	Response primitive

5. Device Characterization

A sensor or actuator constitutes the link between the automation equipment (PLC, host computer) and the process. In simple terms, the sensor converts physical variables from the process into electrical signals for the programmable controller. The actuator converts the electrical signals of the programmable controller into physical variables for the process.

Sensors and actuators may be active or passive nodes on the bus.

The market of general-purpose sensors and actuators requires a wide range of different devices with regard to functions and prices. Owing to the open structure of the Sensor/Actuator Profile, a whole variety of functions is covered and the devices do not have to be divided into device classes.

The device functions and parameters are classified into mandatory, optional and manufacturer-specific ones. If users restrict themselves to the mandatory functions or mandatory parameters, interchangeability of the sensors and actuators is possible.

With respect to communications, and regardless of the function, the devices are always interchangeable if use is made of the same parameters.

6. Application and Device Characteristics

6.1. General

This section describes the entire application from the viewpoint of communications. The application is divided into function blocks as shown in Figure 4:

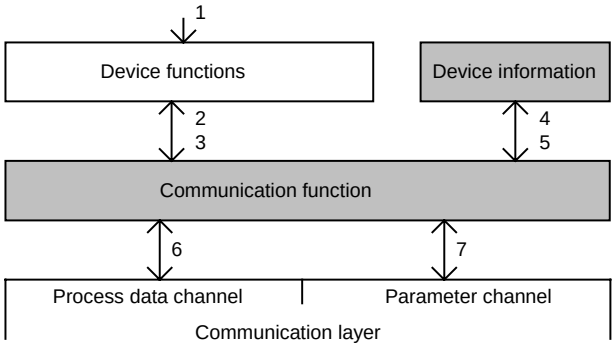

Figure 4: Function blocks of an application

Communication Function

The communication function executes all communication-specific functions.

Device Function

The device function executes all device-specific functions.

Device Information

The device information keeps information on the device in a non-volatile memory.

Communication Layer

The communication layer contains a layer 7 conforming to DIN 19245, Part 2, and a layer 2 conforming to the InterBus-S specification.

Interactions Between the Function Blocks

1 Process variables
2 Process data from the control system to the device functions
3 Process data from the device functions to the control system
4 Storage of device information
5 Reading out device information
6 Mapping onto the process data channel
7 Mapping onto the parameter channel

Function Groups

The following function groups are defined for this profile.

Function group		**Meaning**
Profile group	**Function IDs**	
1	1	communication function
1	2	device information

6.2. Diagnostic Indicators

The diagnostic indicators (LEDs) signal the network status at all remote bus devices. These LEDs must be fitted visibly at the device.

Designation at the device	**Name**	**Color**	**Function**
RC	**R**emote bus **C**heck	green	Monitoring the function of the incoming remote bus
RD	**R**emote bus **D**isable	red	Disabling the outgoing remote bus
BA	**B**us **A**ctive	green	Displays bus activity (ID cycles or data cycles)
TR	**T**ransmit / **R**eceive	green	Displays whether data is transmitted via the parameter channel

Note:
The "TR" LED is only needed if the parameter channel is implemented.

6.3. Communication Functions

6.3.1. Process Data Channel

Several process data can be transmitted in parallel via a process data channel (see example). In such a case, the user must know the structure of this process data channel. The device manufacturer must provide this information about the structure in the form of a process data channel description.

Process data channel of a drive:

	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
IN	Control word		Setpoint		Setpoint factor	
OUT	Status word		Actual value		....	

A process data channel description is implemented as a process data object dictionary. This process data object dictionary contains a process data object description for every process data item. A process data object description contains the following attributes:

Index

The index is the logical address of the process data item within the device. Index 0000hex is reserved for the header of the process data object dictionary. The process data item with the index 0001hex describes the process input data channel and the process data item with the index 0002hex describes the process output data channel.

Object Code

This attribute characterizes the process data item as input or output data.

Data Type

This attribute defines how the process data item is to be interpreted.

Length

This attribute describes the length of the process data item in bytes.

Internal Address

This attribute specifies the position of the process data item within the process data channel of the device.

Extension

Contains additional information (e.g. name of the process data item).

Example for a drive:

Process data descriptions

Process data name	Index	Object code	Data type	Length	Internal addr.	Extension
Input data	0001	PD-IN	Octet String	6	0.0	
Output data	0002	PD-OUT	Octet String	6	0.0	
Status word	6041	PD-IN	Octet String	2	0.0	
Control word	6040	PD-OUT	Octet String	2	0.0	
Actual value	6044	PD-IN	Integer16	2	2.0	
Setpoint value	6042	PD-OUT	Integer16	2	2.0	
Malfunction code	603F	PD-IN	Octet String	2	4.0	
Setpoint factor Z	604B	PD-OUT	Integer16	2	4.0	

6.3.2. Parameter Channel

The communication functions (see Figure 24) comprise of functions and include the following communication specific functions:

- Execution of the services;
- Control of communications;
- Mapping of the process data onto the communication objects;
- Process data monitoring;
- Communication monitoring.

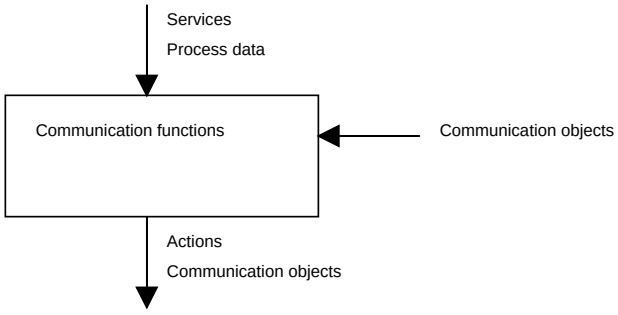

Figure 5: Communication functions

Services

The communication function executes the following functions:

- Initiate (establish connection);
- Abort (abort connection);
- Status (read device status);
- Identify (read manufacturer name, type and version);
- Get-OD (read object descriptions);
- Read (read communication objects);
- Write (write to communication objects).

Process Data

Process data is data which is transmitted via the process data channel and mapped onto communication objects. The process data control function maps the process data onto the communication objects.

Actions

Actions which can be triggered by the communication-specific functions can be parameterized. They basically define the behavior of the device when communications fails.

Communication Objects

Communication objects can be manipulated over the bus with the help of services. The communication objects are mapped onto the real parameters in the device. The data of the communication objects can be transmitted via services or process data. All communication objects, which are accessible with the *read* or *write* service are described in the object dictionary.

An access to non-implemented optional communication objects is always negatively confirmed. (Error message: error class = access (6) error code = object-access-unsupported (6))

6.4.1.1. Connection Establishment

The connection establishment function shown in Figure 6 is executed when a request for a connection establishment has been received from the bus (Initiate service indication). The connection is only established if there is an object dictionary and a VFD object.

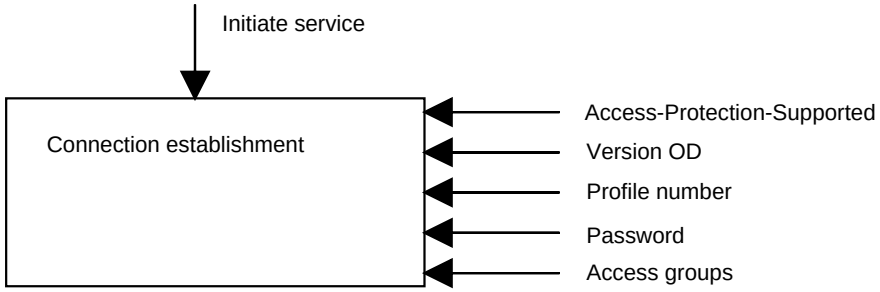

Figure 6: Connection establishment

Initiate Service

The Initiate service requests a connection establishment. The initiator of the connection establishment must set the service parameters to the following values:

Version-OD
This service parameter is of no importance in this profile.

Profile Number
This service parameter is of no importance in this profile.

Access-Protection-Supported
This service parameter is of no importance in this profile.

Password
This parameter contains the password which applies to any access that is carried out to the communication objects in this communication relationship. No access protection by means of a password is provided for the access to communication objects of this profile. Thus, this value is not taken into account in this device. If manufacturer-specific communication objects are defined with an access protection, the communication link must be established with the appropriate password (see manufacturer's declaration).

Access Groups
This parameter contains an allocation of the device user to certain access groups. On this communication relationship, this allocation applies to any access to the communication objects of this device. No access protection via access groups is provided for the access to communication objects that are defined in this profile. This, the value is not taken into account in this device. If manufacturer-specific communication objects are defined with an access protection, the communication link must be established with the corresponding value for the Access-Groups service parameters (see manufacturer's declaration).

The device returns the following service parameters when the service is responded:

Access-Protection-Supported

This service parameter indicates that the access is protected, it is to be set to the value TRUE.

Version OD

The value for the version of the object dictionary has to be assigned in a manufacturer-specific way. The value should be changed whenever the object dictionary is changed.

Profile Number

The value of the *profile-number* service parameter is 0012 hex for this profile. If more detailed profiles that are based on this profile are supported, the corresponding profile number will be returned. The profile number comprises of the components *profile group* and *profile version*. The profile number shows to which profile the device corresponds. A device may only report the corresponding profile number, if all functions that are specified in the profile are available.

b15 b12	b11 b4	b3 b0
	Profile group	Profile version

Profile Group

This parameter declares the profile group in which this function group is described. For a DRIVECOM device, at least a function of profile groups 2 and 1 must be available.

Value range: 0-FF hex

Profile-Group	Meaning
0	No profile
1	Sensor/Actuator
2	DRIVECOM
3	Reserved
4	Controller boards
5	Reserved
6	Reserved
7	Encoder
8	Process controllers
9	Robot controller
A	Wrenching controller
B	ISO valve terminal
C	Welding controller
D	MMI

Profile Version:

This parameter contains the profile version identifier of the corresponding profile group.

Value range: 0-F hex

Password

The value of the *password* service parameter is 0 hex for this profile.

Access Groups

The value of the *access-groups* service parameter is 0 hex for this profile.

Error Message

The error-type service parameter comprises of the following parameters:

- error class;
- error code;
- additional code.

Error class and error code

Error class	Error code	Meaning
0 initiate	0	Not permitted
0 initiate	1 max-pdu-size-insufficient	The PDU length is insufficient.
0 initiate	2 feature-not-supported	The requested service is not supported.
0 initiate	3 version-od-incompatible	The versions of the object dictionary are incompatible.
0 initiate	4 user-initiate-denied	The PMS user rejects the Initiate service
0 initiate	5 password-error	A connection has already been established with this password.
0 initiate	6 profile-number-incompatible	The profile of the client is not supported.
0 initiate	7 other	The error cannot be assigned to any code

Additional code:

2 octets with the following structure:

Bit 15 8 7 4 3 0

Reserved	Global code	Specific code

The bits 8 to 15 have to be assigned the value 0.

Global code	Specific code	Meaning
0	0	No detailed information on the reason of the error
1	0	Object dictionary does not yet exist
2	0	VFD object does not yet exist

All codes which are not listed are reserved.

Mapping the Device Function onto Communications

The Initiate response is returned with the following service parameters.

Initiate.res parameters	Value
Invoke ID	Invoke ID from Initiate.ind
Communication reference	Communication reference from Initiate.ind
Version OD	Manufacturer-specific
Profile number	0012 hex (if only the Sensor/Actuator Profile is supported)
Access-protection-supported	TRUE
Password	Manufacturer-specific
Access groups	Manufacturer-specific

6.4.1.2. Connection Abort

The connection abort function shown in Figure 7 is called when an *abort* service was received from the bus.

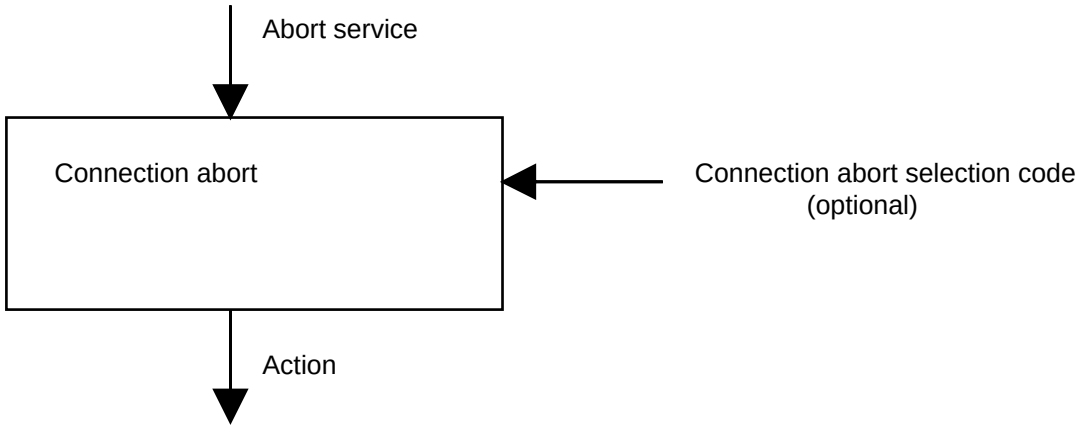

Figure 7: Connection abort

'Connection Abort Selection Code'

The 'connection abort selection code' defines which function is executed if the function is aborted. Should the selection code object not be available, no action will be carried out.

Selection code	Meaning of the selection function
-32768 ... -1	Manufacturer-specific
0	No action
1	Malfunction
2 ... 32767	Reserved for profiles

Object class	Optional
Access	Read only
Process data mapping	Not possible
Unit	-
Value range	Integer16
Mandatory range	0 (no action)
Substitute value	0 (no action)

Action

The function is carried out which is defined in the 'connection abort selection code'.

Error Message

Yes, see *read* or *write* function.

Mapping the Device Function onto Communication

Object description: 'connection abort selection code' (see Table 1).

Table 1: Object description: 'connection abort selection code'

Object attribute	Value hex	Meaning
Index	xxxx	Connection abort selection code
Variable name	-	Not available
Object code	07	Simple variable
Data-type index	03	Integer16
Length	02	2 bytes
Password	00	No password
Access groups	00	No access groups
Access rights	0300	Read all, write all
Local address	xxxx	Manufacturer-specific
Extension	-	Non-existent

6.4.1.3. Aborting the Connection

The *aborting the connection* function shown in Figure 8 releases the connection. The function is used, for example, if the object dictionary or the VFD object of the device is to be changed locally.

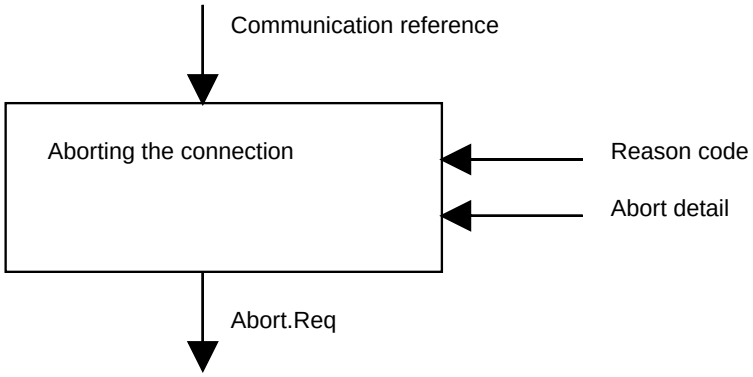

Figure 8: Aborting the connection

Communication Reference

This service parameter contains the communication reference of the communication relationship on which the communication is to be released.

Reason Code

The reason code gives the reason for the aborting of the connection. The following reasons are possible:

- Disconnect:

  The client disconnects the connection

- Version OD incompatible:

  If the device wants to change the object dictionary locally. In this case the versions of the object dictionaries (Source-OD and Remote-OD) of the two communication partners are incompatible.

- Password error:

  A communication relationship has already been established with the same password.

- Profile number incompatible:

  The profile of the server is not supported.

- Limited services permitted:

  The device assumed the logical status "limited number of services".

Reason code	Meaning
1	Disconnect
2	Version OD incompatible
3	Password error
4	Profile number incompatible
5	Limited services permitted

Abort Detail

This parameter contains additional information on the reason of the aborting.

Abort detail	Meaning
-127 ... -1	Manufacturer-specific
0	No abort detail
1 ... 128	Reserved for future profile versions

Abort.Req

The abort service request is used to communicate the aborting of the connection to the communication partner.

Error Recovery

None

Mapping the Device Functions onto Communication

The following service parameters are specified when the connection is aborted:

Abort identifier = 0 (user)

Reason code:

Reason code	Meaning
1	Disconnect
2	Version OD incompatible

Abort detail	Meaning
-127 ... -1	Manufacturer-specific
0	No abort detail
1 ... 128	Reserved for future profile versions

6.4.1.4. Identification

The device is automatically identified by the *identification* function block (see Figure 9).

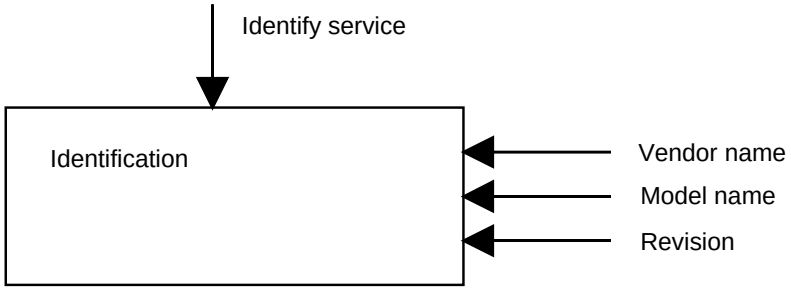

Figure 9: Identification

Vendor Name

This parameter contains the name of the device manufacturers, represented by max. 16 ASCII characters.

Model Name

The parameter contains the model name of the device, represented by max. 16 ASCII characters.

Revision

This parameter contains the revision of the device, represented by max. 16 ASCII characters.

Error Message

None

Mapping the Device Function onto Communication

Communication Reference

The Service.res is called from the Service.ind using the communication reference.

Invoke ID

The Service.res is called with the invoke ID from the Service.ind.

VFD object:

Attribute	Meaning	Length
Vendor name	Manufacturer name	16 bytes
Model name	Device name	16 bytes
Revision	Device version	16 bytes

6.4.1.5. Read Communication Object List

With the function shown in Figure 10, all descriptions of the device parameters can be read out. The device makes the data available to the object dictionary.

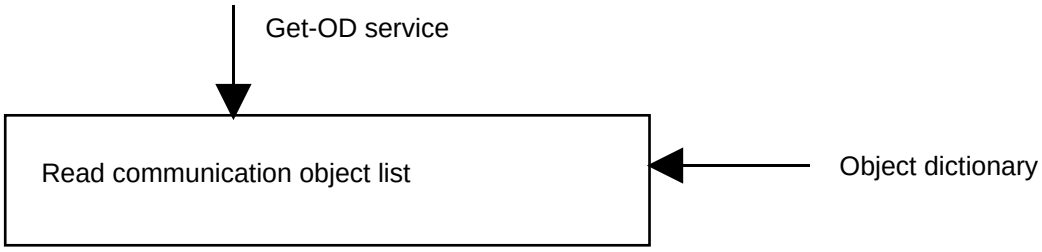

Figure 10: Read communication object list

Get-OD Service

This service reads out the object dictionary.

Object Dictionary

The object dictionary contains the description of all communication objects.

Error Message

The *error-type* service parameter comprises of the following parameters:

- error class;
- error code;
- additional code.

The additional-code parameter has the value zero.

Error class and error code:

Error class	Error code	Meaning
5 Service	4 Parameter-Inconsistent	Access to name with the length zero
5 Service	5 Illegal-Parameter	The 'Access-Specification' service parameter has an invalid value
6 Access	7 Object-Non-Existent	There is no object under this index.
6 Access	8 Type-Conflict	The data of the 'Name' parameter is not of the visible-string data type.
6 Access	9 Name-Access-Unsupported	Access by means of name although a symbol length of zero is specified in the OD.

Mapping the Device Function onto Communication

Communication Reference

The Service.res is called with the communication reference from the Service.ind.

Invoke ID

The Service.res is called with the invoke ID from the Service.ind.

Index

The index parameter defines the object the object description of which is to be read out.

List of Object Descriptions

This service parameter contains the object description of the object that was defined with the index.

Error Type

The error-type parameter contains the error reason.

6.4.1.6. Status Function

The *status* function block (see Figure 11) carries out a status check. The status indicates the device status.

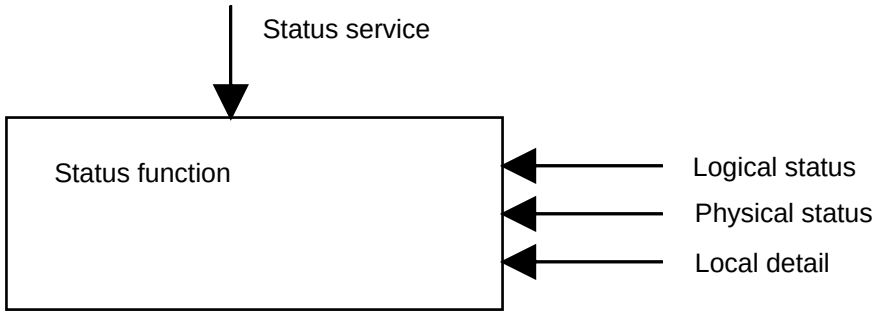

Figure 11: Status function

Logical Status

This parameter defines the communication capability of the device.

	Logical status
0	Ready for communications
2	Limited number of services

Physical Status

This parameter provides a rough overview of the operating state of the device.

	Physical status
0	Ready for operation
1	Partly ready for operation
2	Not ready for operation
3	Service required

Local Detail

This parameter provides the local status of the application and the device. The meaning of the individual bits is defined in profiles. The data type is a bit string with the length 24 bits.

Error Message

None

Mapping the Device Function onto Communication

Communication Reference

The Service.res is called with the communication reference from the Service.ind.

Invoke ID

The Service.res is called with the invoke ID from the Service.ind.

VFD object:

Attribute	Meaning
Logical status	Status of communications
Physical status	Operating state of the device
Local detail	Local status of the application, profile specific

Logical status

This attribute contains information about the communication status of the device:

0 <=> Ready for communication

2 <=> Limited number of services

4 <=> OD-LOADING-NON-INTERACTING

5 <=> OD-LOADING-INTERACTING

6.4.1.7. Read Function

With the function shown in Figure 12, the read access to communication objects is carried out. The data which is transmitted with the *read* service is read from the corresponding device parameters. The allocation of device parameter to communication object can be taken from the object dictionary.

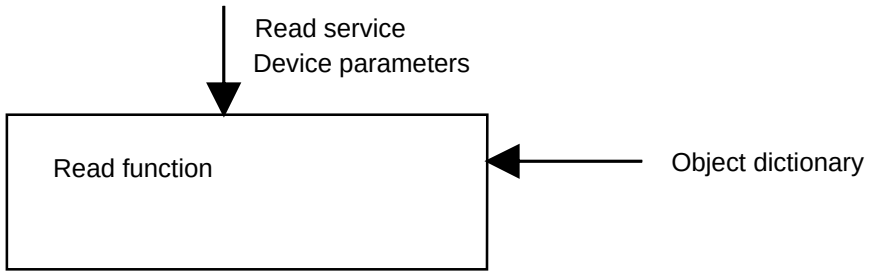

Figure 12: Read function

Read Service

The read service transmits the data which is to be read from a communication object.

Device Parameters

The data which is to be communicated with the read service is read from the device parameters.

Error Message

The error-type service parameter comprises of the following parameters:

- error class;
- error code;
- additional code.

Error class and error code

Error class	Error code	Meaning
6 Access	2 Hardware-Fault	The access to the object failed due to a hardware error. Details on the reason are given in the additional code (glob.= 8)
6 Access	3 Object-Access-Denied	The access rights of the clients are insufficient.
6 Access	5 Object-Attribute-Inconsistent	A service parameter assumed an invalid value. Details on the reason are given in the additional code (glob.=1)
6 Access	6 Object-Access-Unsupported	The object is not a variable-access object
6 Access	7 Object-Non-Existent	There is no object under this index.
8 Other	0 Other	The service was not executed. Details on the reason are given in the additional code.

Additional code:

The additional code consists of a global, specific and a device- or profile-specific part. The specification of a global or specific code is optional. That means: if the error reason cannot be exactly specified by the application, the additional code = 0000 hex must be output. The specific code contains a detailed description of the error reason which is specified in the global code. If the error reason has not the meaning of a specific code, the specific code = 0 must be specified. The device- or profile-specific code (bit 8 to 15) is defined by the device manufacturer or other profiles.

2 octets with the following structure:

Bit 15 8 7 4 3 0

Device- or profile-specific	Global code	Specific code

The values for the bits 8 to 15 are reserved and have currently the value 0.

Global code [hex]	Specific code [hex]	Meaning
0	0	No details of the reason of the error
1	0	Service parameter with an invalid value
1	1	Subindex non-existent
2	0	The service can currently not be executed.
2	1	The service can currently not be executed owing to local control.
2	2	The service cannot be executed in the current device status (device control).
2	3	The service can currently not be executed, because there is no object dictionary.
3	0	The parameter is out of the value range (the server does not want to provide the value)
8	0	Hardware fault
8	1	The application failed.

All codes which are not listed are reserved.

Mapping the Device Function onto Communication

Communication Reference

The Service.res is called with the communication reference from the Service.ind.

Invoke ID

The Service.res is called with the invoke ID from the Service.ind.

Index

This parameter defines the object of the server to be read.

Data

The data is sent to the client in the ’data’ parameter of the Read.res.

Error Type

The error-type parameter contains the reason of the error.

6.4.1.8. Write Function

This function which is shown in Figure 13 is used to write the data sent with the write service to a device parameter. The allocation of device parameter to communication object can be taken from the object dictionary. A write access to a device parameter with a write service can be confirmed negatively with the corresponding error message.

There are two types of this function: single parameterization or block parameterization (see the following sections). The block parameterization is carried out to parameterize parameters which are independent of one another.

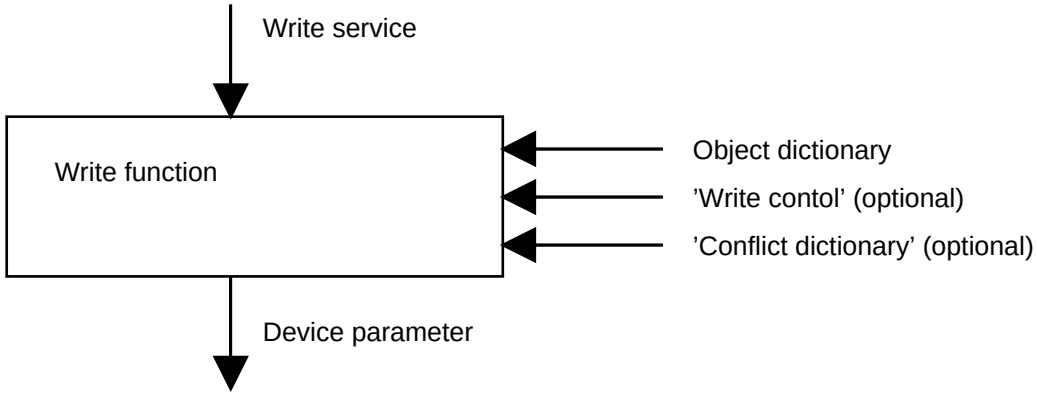

Figure 13: Write function

Write Service

The write service sends the data that is to be written to a communication object. The data length must be compared with the entry in the object dictionary. If the two data lengths are not identical, the service is confirmed negatively.

Device Parameters

The data sent with the write service is written to the device parameters.

Write Control

This parameter controls the transitions between single parameterization and block parameterization. It is described in the section "Block Parameterization".

Conflict Dictionary

This parameter is used in the "Block Parameterization" mode only. Thus, it is described in the section "Block Parameterization".

Error Message

The error-type service parameter comprises of the following parameters:

- error class;
- error code;
- additional code.

Error class and error code

Error class	Error code	Meaning
6 Access	2 Hardware-Fault	The access to the object failed due to a hardware error. Details on the reason are given in the additional code (glob.= 8).
6 Access	3 Object-Access-Denied	The access rights of the clients are insufficient.
6 Access	5 Object-Attribute-Inconsistent	A service parameter assumed an invalid value. Details on the reason are given in the additional code (glob.=1).
6 Access	6 Object-Access-Unsupported	The object is not a variable-access object.
6 Access	7 Object-Non-Existent	There is no object under this index.
8 Other	0 Other	The service was not executed. Details on the reason are given in the additional code.

Additional code:

The additional code consists of a global, specific and a device- or profile-specific part. The specification of a global or specific code is optional. That means: if the error reason cannot be exactly specified by the application, the additional code = 0000hex must be output. The specific code contains a detailed description of the reason of the error which is specified in the global code. If the reason of the error does not correspond to a specific code, the specific code = 0 must be specified. The device- or profile-specific code (bit 8 to 15) is defined by the device manufacturer or other profiles.

2 octets with the following structure:

Bit 15 8 7 4 3 0

Device- or profile-specific	Global code	Specific code

The values for the bits 8 to 15 are reserved and have currently the value 0.

Global code [hex]	Specific code [hex]	Meaning
0	0	No details of the reason of the error
1	0	Service parameter with an invalid value
1	1	Subindex non-existent
2	0	The service can currently not be executed.
2	1	The service can currently not be executed owing to local control.
2	2	The service cannot be executed in the current device status (device control).
2	3	The service can currently not be executed, because there is no object dictionary.
3	0	The parameter is out of the value range.
3	1	The value of the parameter is too great.
3	2	The value of the parameter is too small.
3	3	Reserved (because of DRIVECOM Profile 21)
3	4	Reserved (because of DRIVECOM Profile 21)
3	5	Reserved (because of DRIVECOM Profile 21)
4	0	Collision with other values
4	1	The communication object cannot be mapped onto the process data.
4	2	Process data length exceeded
8	0	Hardware fault
8	1	The application failed.

All codes which are not listed are reserved.

6.4.1.8.1. Single Parameterization

In the single parameterization status, the written parameter content is checked for boundaries and conflicts with other parameter contents. The parameter content is valid only when there is a positive acknowledgment.

6.4.1.8.2. Block Parameterization

In the event of changes, dependencies between parameter contents may cause temporary incompatibilities which would require a certain order or even mutual step by step change (e.g. error message 'collision with other values'). In this case, the 'block parameterization' can be activated and thus the meaning of the positive confirmation can be changed. The positive confirmation means that the parameter content was accepted but does not become effective.

Block parameterization refers to the dependent parameters defined by the manufacturer. The checks can be carried out immediately or must be executed at least when a change to the single parameterization is made. For this purpose, at least the dependent parameters are buffered.

Errors which cannot be excluded by the use of block parameterization can also cause an immediate negative confirmation.

The manufacturer defines the dependency of parameters and documents it for the user. There may be a dependency due to physical relations or the dependency is dynamically defined by the activation or deactivation of the block of the block parameterization.

The transitions between single parameterization and block parameterization are triggered by the 'write control' parameter. When the block parameterization is quit the parameters must be consistent so that they can be completely accepted and the access is confirmed positively. In contrast to this, the write access to the 'write control' parameter is negatively confirmed when the status changes from block parameterization to single parameterization and existing incompatibility. In this case, the incompatible parameters retain the value which was valid before block parameterization took place.

During the block parameterization, a read access returns the value before the block parameterization took place for dependent parameters.

'Write Control'

This parameter controls the transitions between single parameterization and block parameterization.

False : single parameterization

True : block parameterization

The following actions are carried out when the parameter content is changed:

False -> True:

Change from single parameterization to block parameterization and update of the 'conflict dictionary' parameter

True -> False:

Change of block parameterization to single parameterization.

Check for consistency:

- If consistency is given, the parameter contents are accepted and the write access to the 'write control' parameter is confirmed positively.
- If inconsistency is given the old contents of parameters involved in the collision remain valid. The indexes and the error message of the parameters involved in the conflict is stored in the 'conflict dictionary'. The write access to the 'write control' parameter is confirmed negatively.

Object class:	Optional
Access:	Write and read
Process data mapping:	Not possible
Unit:	-
Value range:	Boolean
Mandatory range:	True, false
Substitute value:	False

'Conflict Dictionary'

This parameter contains the indexes and the error messages (add. code) of the parameters involved in the conflict. If 'block write control' is negatively confirmed, the 'conflict dictionary' must be updated, however, it can also be updated after every write access in and/or out of the 'block parameterization' state.

Conflict dictionary

Index
Add.-Code
Index
Add.-Code
...
...

Object class	Optional
Access:	Read
Process data mapping:	Not possible
Unit:	-
Value range:	Unsigned 16
Mandatory range:	-
Substitute value:	-

Mapping the Device Function onto Communication

Communication Reference

The Service.res is called with the communication reference from the Service.ind.

Invoke ID

The Service.res is called with the invoke ID from the Service.ind.

Index

This parameter defines the object of the server to be written.

Data

The data is sent in the parameter 'data' from the client to the Service.ind.

Error Type

This parameter contains the reason for the error.

Object description: 'write control' (see Table 2).

Table 2: Object description: 'write control'

Object attribute	Value hex	Meaning
Index	600A	Write control
Variable name	-	Non-existent
Object code	07	Simple variable
Data-type index	01	Boolean
Length	01	1 byte
Password	00	No password
Access groups	00	No access groups
Access rights	0300	Read all, write all
Local address	xxxx	Manufacturer-specific
Extension	-	Non-existent

Object description: 'conflict dictionary' (see Table 3).

Table 3: Object description: 'conflict dictionary'

Object attribute	Value hex	Meaning
Index	604C	Conflict dictionary
Variable name	-	Non-existent
Object code	08	Array
Number of elements	xx	n elements
Data-type index	05	Octet string
Length	02	2 bytes
Password	00	No password
Access groups	00	No access groups
Access rights	0001	Read all
Local address	xxxx	Manufacturer-specific
Extension	-	Non-existent

Value description

Subindex (hex)	Meaning
01	Index
02	Add.-Code
03	Index
04	Add.-Code
x	...

6.4.1.9. Process Data Control

This function maps the process data to be transmitted over the process data channel to the communication objects.

The process data channel can be implemented with a width from 0 to 26 words (see InterBus-S Management Documentation). The server device can read and write each byte.

The data which the server device reads from the process data channel is called process output data. The data which the device writes into the process data channel is called process input data.

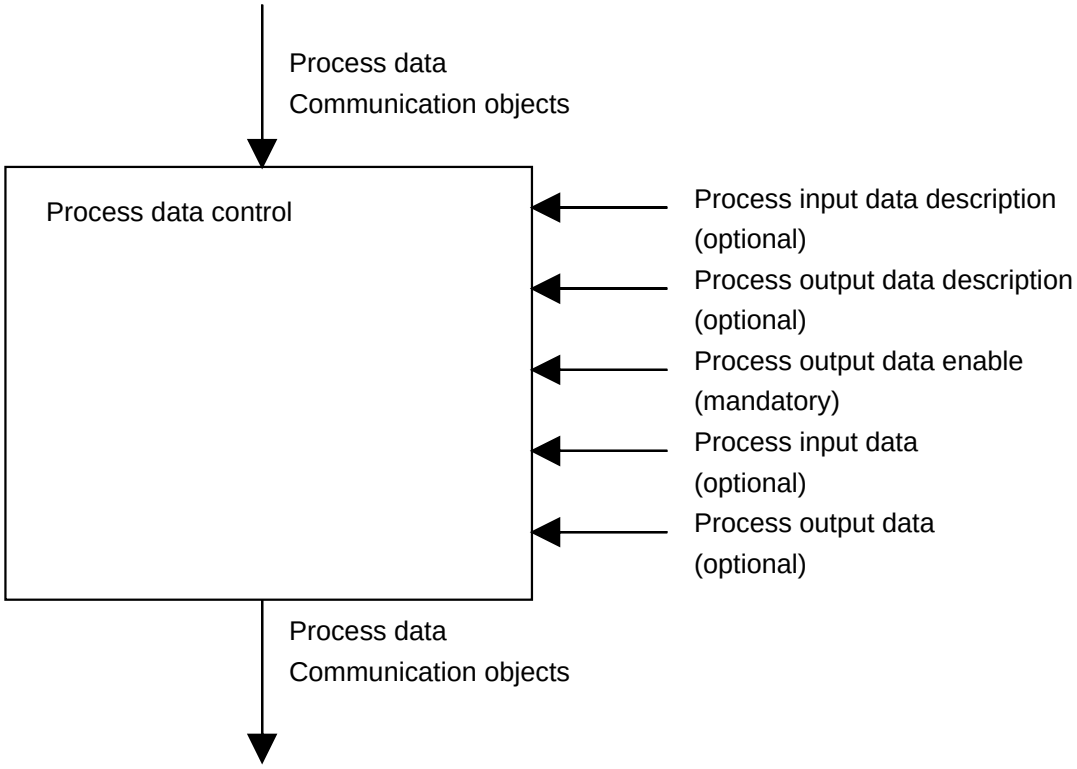

Figure 14: Process data control

By mapping the communication objects onto the process input data the corresponding server device parameters are cyclically read via the process data channel.

The process output data which is mapped onto a communication object is cyclically transmitted to the device parameters.

The allocation of process data to certain communication objects can be parameterized. This can be done with the communication objects ’process input data description’ and ’process output data description’. The process output data can be enabled or disabled with the ’process output data enable’ parameter.

Method of functioning:

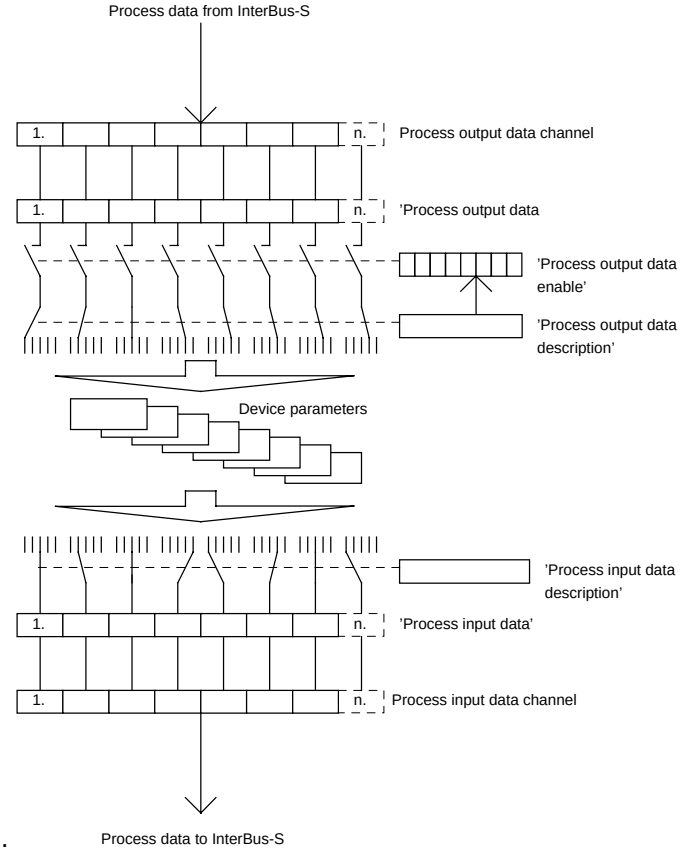

Figure 15: Method of functioning

The functionality of the process data control can be made clearer with the help of switches. The 'process input data description' parameter (input data for the master) determines which device parameter is mapped onto which byte in the InterBus-S process data channel. The 'process output data description' parameter (output data for the master) determines which bytes of the InterBus-S process data are mapped onto which device parameters.

The allocation of the device parameters to the bytes of the process data channel is done with the parameter identifier (index, subindex). The size of the parameters 'process output data description' and 'process input data description' depends on the length of the process data channel (defined by hardware). The first element of the two parameters specifies the length of the process data channel. Various data types (and thus lengths) can be mapped onto the process data channel.

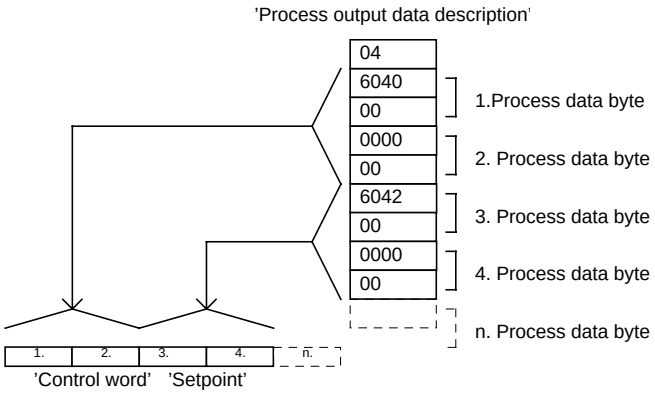

Figure 16: Allocation of the parameter content to the process data channel

NOTE:
The parameter identifier 0000,xx is permitted and is interpreted as 'not assigned'.

In order to change the configuration of the process data it is necessary that the connection between InterBus process data and device parameters can be interrupted in a defined way. This can be done with the 'process output data enable' parameter.

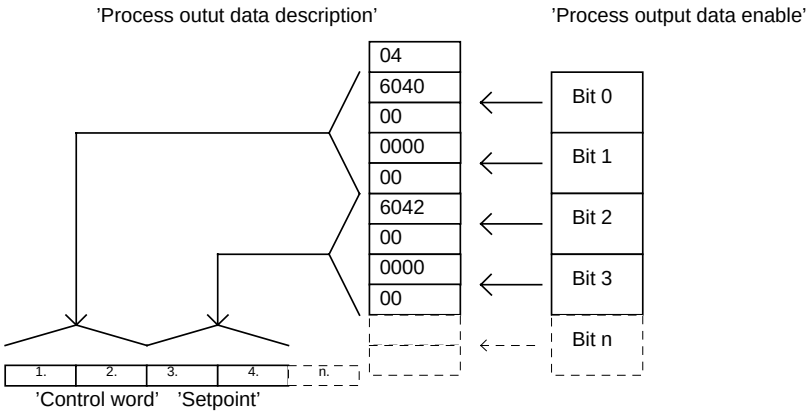

Figure 17: Method of functioning of the 'process output data enable' parameter

A bit of the 'process output data enable' parameter is assigned to a byte of the process output data channel. Here, the following assignment applies:

Bit = 0 The corresponding parameter mapping is disabled

Bit = 1 The corresponding parameter mapping is enabled.

If a parameter is assigned to several bytes, the bit is valid which is assigned to the first byte of the parameter. This corresponds to the position of the index entry in the associated process output description or process input data description. All other possibly associated bits are irrelevant.

The writing of the 'process output data description' parameter automatically interrupts the connection between process output data and the corresponding device parameter. This is done by resetting the appropriate control bits in the 'process output data enable' parameter.

Example 1:

If the mapping of the first parameter in the 'process output data description' parameter is changed (with subindex<>0 of the 'process output data description'), bit 0 is reset. The other bits are not changed.

Example 2:

If the mapping of all parameters in the 'process output data description' parameter is changed (with subindex<>0 of the 'process output data description'), all bits are reset.

Exemplary assignment of the process data channel with the following parameters:

1. Parameter A Data type : Unsigned8
2. Parameter B Data type : Unsigned8
3. Not assigned
4. Parameter C Data type : Unsigned8

Subindex of the parameter 'process input data descr.'	**Meaning of the parameter elements**	**Byte number in the process data channel**
1	Length of the process data channel	-
2	Index of a device parameter A (8-bit parameter)	1
3	Subindex of a device parameter A	1
4	Index of a device parameter B (8-bit parameter)	2
5	Subindex of a device parameter B	2
6	Not assigned. Index = 0000	3
7	Not assigned. Subindex =00	3
8	Index of a device parameter C (8-bit parameter)	4
9	Subindex of a device parameter C	4
...		...

Exemplary assignment of the process data channel with the following parameters:

1. Parameter A Data type : Unsigned16
2. Parameter B Data type : Unsigned8
3. Parameter C Data type : Unsigned8

Subindex of the parameter 'process input data descr.'	**Meaning of the parameter elements**	**Byte number in the process data channel**
1	Length of the process data channel	-
2	Index of a device parameter A (16-bit parameter)	1
3	Subindex of a device parameter A	1
4	0000 (assigned through parameter A)	2
5	00 (assigned through parameter A)	2
6	Index of a device parameter B (8-bit parameter)	3
7	Subindex of a device parameter B	3
8	Index of a device parameter C (8-bit parameter)	4
9	Subindex of a device parameter C	4
...		...

If the parameter map onto the process data channel is to be replaced by a parameter with a larger data width, the maps of the subsequent parameter channel bytes have to be cleared first. The bytes can be cleared by writing 0000,00 to the corresponding parameter identifier. If the mapping cannot be carried out, the service is confirmed negatively.

'Process Input Data Description'

This parameter contains the data that defines which process input data is mapped onto which communication objects. Communication objects that can be mapped onto process input data are marked in the respective parameter descriptions. If the entered indexes do not agree with the subindexes, this parameter is not written to and an error message is sent.

Object class	Optional
Access	Read and write
Process data mapping	Not possible
Unit	-
Value range Subindex 1: Subindex 2: Subindex 3: Subindex 4: Subindex 5: .... Subindex n: Subindex n+1:	 Unsigned8 Unsigned16 Unsigned8 Unsigned16 Unsigned8 Unsigned16 Unsigned8
Mandatory range	-
Substitute value	-

'Process Output Data Description'

This parameter contains the data that defines onto which communication objects the process output data is mapped. When this parameter is written to the mapping of the process output data onto the communication object is interrupted. Communication objects which can be mapped onto process output data are marked in the respective parameter descriptions. If the entered indexes do not agree with the subindexes, this parameter is not written to and an error message is sent. When this parameter is written to, the corresponding bits in the 'process output data enable' parameter are cleared.

Object class	Optional
Access	Write and read
Process data mapping	Not possible
Unit	-
Value range Subindex 1: Subindex 2: Subindex 3: Subindex 4: Subindex 5: .... Subindex n: Subindex n+1:	 Unsigned8 Unsigned16 Unsigned8 Unsigned16 Unsigned8 Unsigned16 Unsigned8
Mandatory range:	-
Substitute value	-

'Process Output Data Enable'

A bit of the 'process output data enable' parameter is assigned to a byte of the process output data channel. Here, the following assignment applies:

Bit = 0 The corresponding process data value is disabled.

Bit = 1 The corresponding process data word is enabled.

If one parameter is assigned to several bytes, the logical state of all associated bits applies:

TRUE = enabled

FALSE = disabled

Object class:	Mandatory
Access:	Write and read
Process data mapping:	Not possible
Unit:	-
Value range:	-
Mandatory range:	0, FFhex
Substitute value:	See mandatory range

'Process Input Data'

This parameter contains the data which is transmitted over the process data channel to the host system. The parameter size depends on the process data channel length.

Object class:	Optional
Access:	Write and read
Process data mapping:	Not possible
Unit:	-
Value range:	-
Mandatory range:	-
Substitute value:	-

'Process Output Data'

This parameter contains the data which is sent to the device via the process data channel. The parameter size depends on the length of the process data channel.

Object class:	Optional
Access:	Read only
Process data mapping:	Not possible
Unit:	-
Value range:	-
Mandatory range:	-
Substitute value:	-

Mapping the Device Function onto Communication

Object description: 'process input data description'

Object attribute	Value	Meaning
Index	6000	Process input data description
Variable name	-	Non-existent
Object code	09	Record
Data-type index	20	PDB structure
Password	00	No password
Access groups	00	No access groups
Access rights	0300	Read all, write all
Local address	xxxx	Manufacturer-specific
Extension	-	Non-existent

Object description: 'process output data description'

Object attribute	Value	Meaning
Index	6001	Process output data description
Variable name	-	Non-existent
Object code	09	Record
Data-type index	20	PDB structure
Password	00	No password
Access groups	00	No access groups
Access rights	0300	Read all, write all
Local address	xxxx	Manufacturer-specific
Extension	-	Non-existent

Object description: 'process output data enable'

Object attribute	Value	Meaning
Index	6002	Process output data enable
Variable name	-	Non-existent
Object code	07	Simple variable
Data-type index	0A	Octet string
Length	0n	n byte (manufacturer-specific)
Password	00	No password
Access groups	00	No access groups
Access rights	0300	Read-all, write-all
Local address	xxxx	Manufacturer-specific
Extension	-	Non-existent

Object description: ’process input data’

Object attribute	Value	Meaning
Index	6010	Process input data
Variable name	-	Non-existent
Object code	07	Simple variable
Data-type index	0A	Octet string
Length	n	n byte
Password	00	No password
Access groups	00	No access groups
Access rights	0300	Read-all, write-all
Local address	xxxx	Manufacturer-specific
Extension	-	Non-existent

Object description: ’process output data’

Object attribute	Value	Meaning
Index	6011	Process output data
Variable name	-	Non-existent
Object code	07	Simple variable
Data-type index	0A	Octet string
Length	n	n byte
Password	00	No password
Access groups	00	No access groups
Access rights	0300	Read all, write all
Local address	xxxx	Manufacturer-specific
Extension	-	Non-existent

6.4.1.10. Process Data Monitoring

The process data monitoring (see Figure 18) monitors the data transmission over the process data channel. If the data transmission is longer inactive than the set monitoring time, a parameterizable action is triggered.

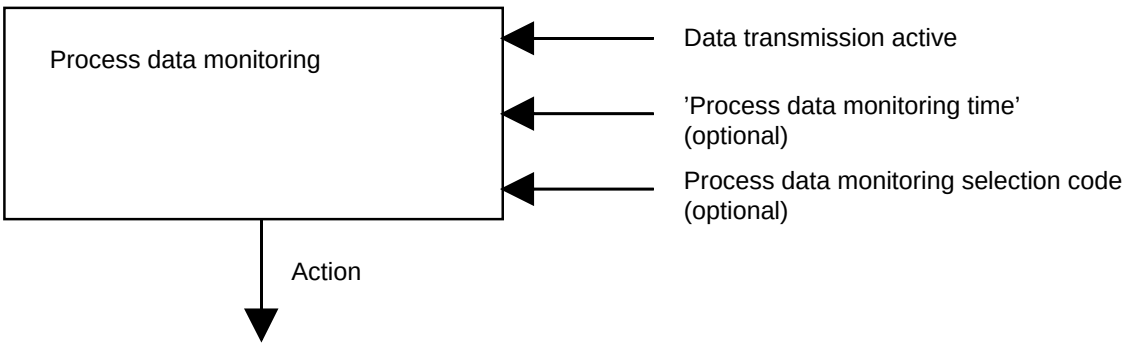

Figure 18: Process data monitoring

Data transmission active

This internal signal indicates that new process data was transmitted.

'Process data monitoring time'

The 'process data monitoring time' parameter determines the maximum time until new process data is transmitted over the process data channel. The values of the transmission time are given in ms. The value 65535 (FFFF hex) disables the process data monitoring. The manufacturer can limit the value range of each device.

Object class:	Optional
Access:	Write and read
Process data mapping:	Not possible
Unit:	ms
Value range:	0 to 65534 ms and 65535 = disabled
Mandatory range:	65535 (disabled)
Substitute value:	65535 (disabled)

'Process Data Monitoring Selection Code'

The 'process data monitoring selection code' defines the function which is to be triggered if no new process data is transmitted within the set monitoring time. If the communication object 'process data monitoring selection code' is not available, no action is triggered.

Selection code	Meaning of the selection function
-32768 ... -1	Manufacturer-specific
0	No action
1 ... 32767	Reserved for further profiles

Object class:	Optional
Access:	Write and read
Process data mapping:	Not possible
Unit:	ms
Value range:	Integer16
Mandatory range:	0 (no action)
Substitute value:	0 (no action)

Action

A function is triggered which is defined in the 'process data monitoring selection code'.

Error Message

Yes, see *read* or *write* function.

Mapping the Device Function onto Communication

Object description: 'process data monitoring time' (see Table 4).

Table 4: Object description: 'process data monitoring time'

Object attribute	Value hex	Meaning
Index	6003	Process data monitoring time
Variable name	-	Non-existent
Object code	07	Simple variable
Data-type index	06	Unsigned16
Length	02	2 bytes
Password	00	No password
Access groups	00	No access groups
Access rights	0300	Read all, write all
Local address	xxxx	Manufacturer-specific
Extension	-	Non-existent

Object description: 'process data monitoring selection code' (see Table 5).

Table 5: Object description: 'process data selection code'

Object attribute	Value hex	Meaning
Index	6004	Process data monitoring selection code
Variable name	-	Non-existent
Object code	07	Simple variable
Data type index	03	Integer16
Length	02	2 bytes
Password	00	No password
Access groups	00	No access groups
Access rights	0300	Read all, write all
Local address	xxxx	Manufacturer-specific
Extension	-	Non-existent

6.4.1.11. Communication Monitoring

The communication monitoring (see Figure 19) monitors the data transmission over the communication channel. If the data transmission is longer inactive than the set transmission time, a parameterizable action is triggered. The transmission time should be adapted to the bus transmission time.

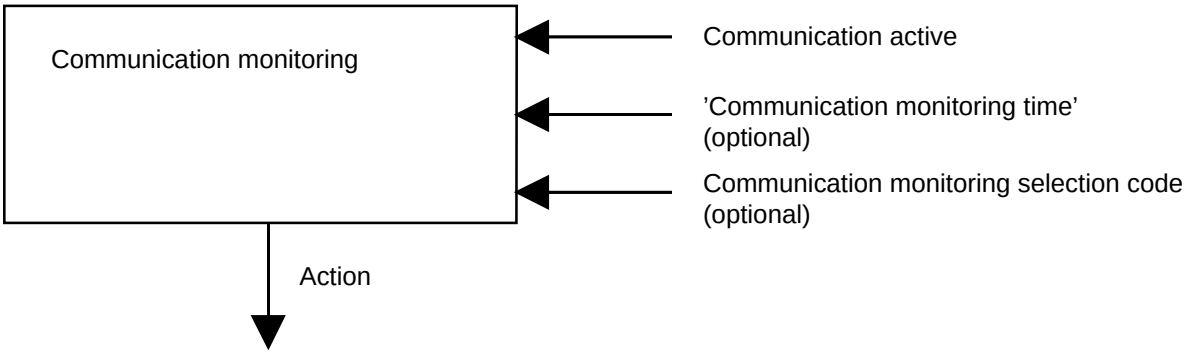

Figure 19: Communication monitoring

Communication Active

Communication actions over the communication channel.

'Communication Monitoring Time'

The 'communication monitoring time' parameter defines the maximum time until new data is transmitted over the communication channel. The values of the transmission time are given in ms. The value 65535 (FFFF hex) disables the communication monitoring. The manufacturer can limit the value range of each device.

Object class:	Optional
Access:	Write and read
Process data mapping:	Not possible
Unit:	ms
Value range:	0 to 65534 ms and 65535 = disabled
Mandatory range:	65535 (disabled)
Substitute value:	65535 (disabled)

'Communication Monitoring Selection Code'

The 'communication monitoring selection code' defines the function which is to be triggered if no new process data is transmitted over the communication channel within the set communication monitoring time.

Selection code	Meaning of the selection function
-32768 ... -1	Manufacturer-specific
0	No action
1 ... 32767	Reserved for further profiles

Object class:	Optional
Access:	Write and read
Process data mapping:	Not possible
Unit:	ms
Value range:	Integer16
Mandatory range:	0 (no action)
Substitute value:	0 (no action)

Action

A function is triggered which is defined in the 'communication monitoring selection code'.

Error Message

Yes, see *read* or *write* function.

Mapping the Device Function onto Communication

Object description: 'communication monitoring time' (see Table 6).

Table 6: Object description: 'communication monitoring time'

Object attribute	Value hex	Meaning
Index	6005	Communication monitoring time
Variable name	-	Non-existent
Object code	07	Simple variable
Data-type index	06	Unsigned16
Length	02	2 bytes
Password	00	No password
Access groups	00	No access groups
Access rights	0300	Read all, write all
Local address	xxxx	Manufacturer-specific
Extension	-	Non-existent

Object description: 'communication monitoring selection code' (see Table 7).

Table 7: Object description: 'communication monitoring selection code'

Object attribute	Value hex	Meaning
Index	6006	Communication monitoring selection code
Variable name	-	Non-existent
Object code	07	Simple variable
Data-type index	03	Integer16
Length	02	2 bytes
Password	00	No password
Access groups	00	No access groups
Access rights	0300	Read all, write all
Local address	xxxx	Manufacturer-specific
Extension	-	Non-existent

6.4. Sensor/Actuator Function

6.4.1. Function Group Description

The function group description (see Figure 20) contains information on the device function groups which the device currently supports.

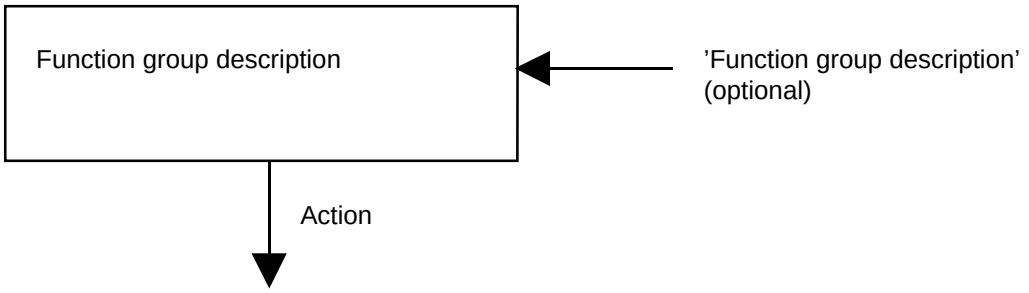

Figure 20: Function group description

'Function group description'

This parameter contains information about the function groups of the device. The parameter is a field with 4 * n entries. Value range of n (max. PDU length).

Attribute	Value
Index, name	600F, function group description
Object class	Optional
Access	Read only
Process data mapping	Manufacturer-specific
Unit	-
Value range	-
Mandatory range	-
Default value	-
Substitute value	-

Example of a DRIVECOM device:

	Profile number		Function group number	
	Profile group	Version	Function group identifier	Version
00120100	Sensor/Actuator	2	Communication function	0
002201xx	DRIVECOM	1	Device control state machine	x
002zyyxx	DRIVECOM	z	An application function yy	x

Profile Number

This parameter shows in which profile the function is described which is implemented in this device. The meaning of the profile numbers is defined in Section "Connection Establishment".

Function Group Identifier

This parameter identifies a function group within a profile group. Value range: 0-FF hex

Function group identifier for the Sensor/Actuator Profile:

Function group identifier	Meaning
0	Reserved
1	Communication functions
2	Sensor/Actuator functions

The following function groups are defined, for example, for the DRIVECOM profile.

Function group identifier	Meaning
0	Reserved
1	Device control state machine
2	General-purpose functions
3	Operating mode functions
4	Speed functions 1(old)
5	Speed functions 2(new)
6	Position functions

Function Version

This parameter identifies the version of a device function group.

Value range: 0-FF hex

EXAMPLE 1:

	Profile number		Function group number	
	Profile group	Version	Function group identifier	Version
00 12 01 00	Sensor/Actuator	2	Communication function	0
00 12 01 00	Sensor/Actuator	2	Communication function	0
00 12 02 00	Sensor/Actuator	2	Device information	0
00 21 00 00	DRIVECOM 21	1	-	-
00 22 01 00	DRIVECOM	2	Device control	0
00 22 02 00	DRIVECOM	2	Speed function old	0
00 22 03 00	DRIVECOM	2	Speed function new	0
00 22 04 00	DRIVECOM	2	Position functions	0
00 22 05 00	DRIVECOM	2	Torque functions	0

Error Message

Yes, see *read* or *write* function.

Mapping the Device Function onto Communication

Object description: 'function group description' (see Table 8).

Table 8: Object description: 'function group description'

Object attribute	Value	Meaning
Index	600F	Function group description
Variable name	-	Non-existent
Object code	08	Array
Number of elements	n	n elements
Data-type index	A	Octet string
Length	4	4 bytes
Password	00	No password
Access groups	00	No access groups
Access rights	0001	Read all
Local address	xxxx	Manufacturer-specific
Extension	-	Non-existent

6.4.2. Device Information

This function which is shown in Figure 21 manages the following device functions:

- Identifying number;
- Parameter set identifier;
- Series number;
- Date of calibration;
- Device description;
- Model description;
- Date of parameterization;
- Function calls description.

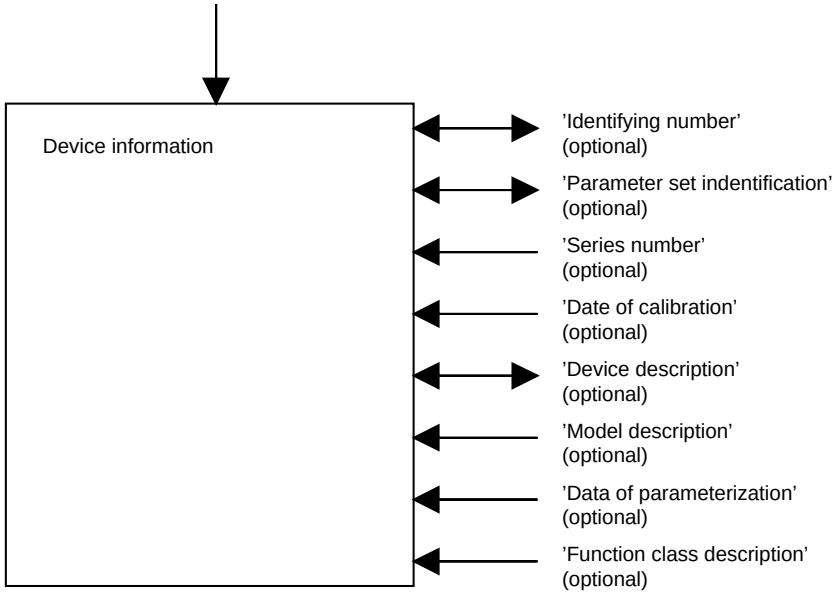

Figure 21: Device information

'Identifying number'

The user can use this function to store an Identifying number in a non-volatile way in the device. The value can be freely selected. Thus it is possible that every number on the bus is assigned several times. By assigning the Identifying numbers on the bus in an appropriate way, for example, the replacing of a device or the mixing up of bus connection can be detected.

Object class:	Optional
Access:	Write and read
Process data mapping:	Not possible
Unit:	None
Value range:	Unsigned16
Mandatory range:	-
Substitute value:	-

'Parameter Set Identifier'

This parameter is used to identify the currently effective device parameter set. If the device cannot store the received device parameter values so that values are kept when the power fails, the device automatically sets this parameter set identifier to 0 after power on. The user can evaluate this information and initialize the set again.

0	The parameter set of the device has not yet been initialized over the bus.
1-254	The parameter set of the device was initialized over the bus. For identification, it received the freely selectable identifying numbers 1-254.
255	The device was or is set to local mode and it is ensured that the previously loaded parameter set is still unchanged.

The user of the device can enter all values between 0 and 255.

Object class:	Optional
Access:	Write and read
Process data mapping:	Not possible
Unit:	None
Value range:	Unsigned8
Mandatory range:	-
Substitute value:	-

'Series Number'

This parameter contains the series number. This parameter can only be read.

Object class:	Optional
Access:	Read only
Process data mapping:	Not possible
Unit:	None
Value range:	Unsigned32
Mandatory range:	-
Substitute value:	-

'Date of Calibration'

This parameter contains the date of calibration. This parameter can only be read.

Object class:	Optional
Access:	Read only
Process data mapping:	Not possible
Unit:	None
Value range:	See 'date' data type
Mandatory range:	-
Substitute value:	-

'Device Description'

This parameter contains a text which the device user stored in this parameter. The device description is stored in a non-volatile memory. The device user can store in this parameter, for example, a description on how the device is stored in the system. The text has a fixed length of 64 characters.

Object class:	Optional
Access:	Write and read
Process data mapping:	Not possible
Unit:	None
Value range:	See 'visible string' data type
Mandatory range:	-
Substitute value:	-

'Model Description'

This parameter contains a text which the device manufacturer stored in this parameter. It may contain, for example, a short device description. The text has a fixed length of 64 characters and can only be read.

Object class:	Optional
Access:	Read only
Process data mapping:	Not possible
Unit:	None
Value range:	See 'visible string' data type
Mandatory range:	-
Substitute value:	-

'Date of Parameterization'

This parameter contains the date of the last change of any device parameter.

Object class:	Optional
Access:	Read only
Process data mapping:	Not possible
Unit:	None
Value range:	See 'Date' data type
Mandatory range:	-
Substitute value:	-

Mapping the Device Function onto Communication

Object description: 'identifying number' (see Table 9).

Table 9: Object description: 'identifying number'

Object attribute	Value hex	Meaning
Index	6008	Identifying number
Variable name	-	Non-existent
Object code	07	Simple variable
Data-type index	06	Unsigned16
Length	02	2 bytes
Password	00	No password
Access groups	00	No access groups
Access rights	0300	Read all, write all
Local address	xxxx	Manufacturer-specific
Extension	-	Non-existent

Object description: 'parameter set identifier' (see Table 10).

Table 10: Object description: 'parameter set identifier'

Object attribute	Value hex	Meaning
Index	6009	Parameter set identifier
Variable name	-	Non-existent
Object code	07	Simple variable
Data-type index	05	Unsigned8
Length	01	1 byte
Password	00	No password
Access groups	00	No access groups
Access rights	0300	Read all, write all
Local address	xxxx	Manufacturer-specific
Extension	-	Non-existent

Object description: 'series number' (see Table 11).

Table 11: Object description: 'series number'

Object attribute	Value hex	Meaning
Index	600A	Series number
Variable name	-	Non-existent
Object code	07	Simple variable
Data-type index	07	Unsigned32
Length	04	4 bytes
Password	00	No pass word
Access groups	00	No access groups
Access rights	0001	Read all
Local address	xxxx	Manufacturer-specific
Extension	-	Non-existent

Object description: 'date of calibration' (see Table 12).

Table 12: Object description: 'date of calibration'

Object attribute	Value hex	Meaning
Index	600B	Date of calibration
Variable name	-	Non-existent
Object code	07	Simple variable
Data type index	0B	Date
Length	07	7 bytes
Password	00	No password
Access groups	00	No access groups
Access rights	0001	Read all
Local address	xxxx	Manufacturer-specific
Extension	-	Non-existent

Object description: 'device description' (see Table 13).

Table 13: Object description: 'device description'

Object attribute	Value hex	Meaning
Index	600C	Device description
Variable name	-	Non-existent
Object code	07	Simple variable
Data-type index	09	Visible string
Length	40	64 bytes
Password	00	No password
Access groups	00	No access groups
Access rights	0300	Read all, write all
Local address	xxxx	Manufacturer-specific
Extension	-	Non-existent

Object description: 'model description' (see Table 14).

Table 14: Object description: 'model description'

Object attribute	Value hex	Meaning
Index	600D	Model description
Variable name	-	Non-existent
Object code	07	Simple variable
Data-type index	09	Visible string
Length	40	64 bytes
Password	00	No password
Access groups	00	No access groups
Access rights	0001	Read all
Local address	xxxx	Manufacturer-specific
Extension	-	Non-existent

Object description: ’date of parameterization’ (see Table 15).

Table 15: Object description: ’date of parameterization’

Object attribute	Value hex	Meaning
Index	600E	Date of parameterization
Variable name	-	Non-existent
Object code	07	Simple variable
Data-type index	0B	Date
Length	07	7 bytes
Password	00	No password
Access groups	00	No access groups
Access rights	0001	Read all
Local address	xxxx	Manufacturer-specific
Extension	-	Non-existent

6.4.3. Malfunction Function

The malfunction function (see Figure 22) manages the 'malfunction code' parameter. As a result of a device malfunction, the 'malfunction code' parameter is set to the corresponding value (see malfunction list). The parameter is reset to the value 0 by the malfunction reset action of the device control.

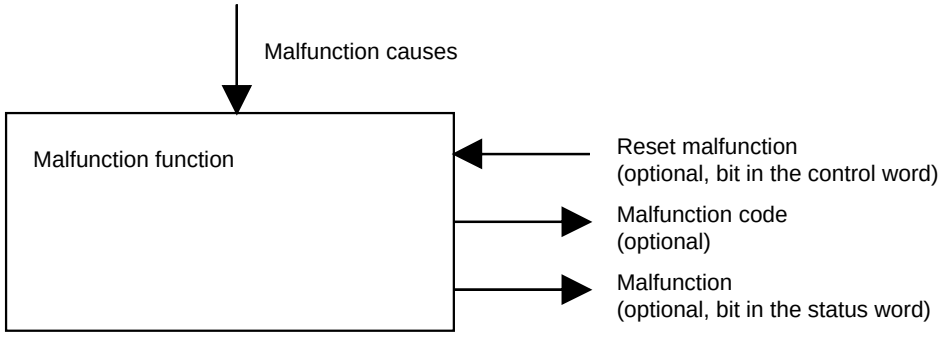

Figure 22: Malfunction function

'Malfunction Code'

The 'malfunction code' is represented as an octet string with a length of 2 bytes. It is coded hierarchically, ranging from a coarse distinction to one that becomes increasingly finer (see Table 16).

Bit	Grouping
15 . . . 12	Main groups
11 . . . 8	Subgroups
7 . . . 0	Details

The parameter is assigned a value unequal to zero when the controller is in the malfunction state. The parameter is assigned the value 0 when the controller is not in the malfunction state.

When there is precisely one cause of a malfunction, the value assigned to this cause in the 'malfunction code' parameter can be read out unchanged until the malfunction state no longer applies. This is the case whenever the cause of the malfunction has been remedied and the malfunction reset command has been issued.

When there are several simultaneous causes of a malfunction, one of them is indicated in the 'malfunction code' parameter. When only the indicated malfunction cause is remedied and the reset malfunction command is issued, the malfunction state is not terminated because the other malfunction causes still apply. One of these malfunction causes is then indicated in the 'malfunction code' object.

Object class:	Optional
Access:	Read only
Process data mapping:	Not possible
Unit:	None
Value range:	0 to 65535
Mandatory range:	-
Substitute value:	-

Table 16: Malfunction codes and malfunction causes

Code hex	Meaning
0000	**No malfunction**
1000	**General malfunction**
2000	**Current**
2100	Current, device input side
2200	Current, device-internal
2300	Current, device output side
3000	**Voltage**
3100	Line voltage
3200	Voltage device internal
3300	Output voltage
4000	**Temperature**
4100	Ambient temperature
4200	Device temperature
4300	Drive temperature
4400	Supply temperature
5000	**Device hardware (only inside the device housing)**
5100	Supply
5200	Control
5300	Operating and display unit
6000	**Device software**
6100	Internal software
6200	User software
6300	Data set
7000	**Supplementary modules**
7100	Power
7200	Measurement circuit
7300	Sensor
7400	Computing circuit
7500	Communication
7600	Data memory
8000	**Monitoring**
8100	Communication
8110	Process data monitoring
8120	Host monitoring
8200	Closed-loop control
9000	**External malfunction**
F000	**Additional functions**

All codes that are not listed are reserved or defined in other profiles.

Error Message

Yes, see *read* or *write* function.

Mapping the Device Function onto Communication

Object description: 'malfunction code' (see Table 17).

Table 17: Object description: 'malfunction code'

Object attribute	Value hex	Meaning
Index	603F	Malfunction code
Variable name	-	Non-existent
Object code	07	Simple variable
Data-type index	0A	Octet string
Length	02	2 bytes
Password	00	No password
Access groups	00	No access groups
Access rights	0001	Read all
Local address	xxxx	Manufacturer-specific
Extension	-	Non-existent

7. Data Structures

This chapter lists the data structures of all user data.

The parameters of a process controller are stored in an object dictionary. This object dictionary is used for describing the parameters. It contains data on the index, data type, object type, the access rights, etc. The index is used for addressing the parameter when writing or reading is to take place. This object dictionary can be read out with the 'read communication object list' function.

7.1. Structure of the Object Dictionary

Table 18: Structure of the object dictionary

Index	Object dictionary
0000	OD object description
0001 001F	(DIN 19245/Part 2) Static type dictionary
0020 003F	(Profiles) Static type dictionary
0040 005F	(Free for manufacturer) Static type dictionary
. . .	. . .
2000 5FFF	(Free for manufacturer) Static object dictionary
6000 603F	(Devices according to Sensor/Actuator Profile) Static object dictionary
6040 9FFF	(Profiles) Static object dictionary
A000 BFFF	(Free for manufacturer) Dynamic variable list dictionary
C000 DFFF	(Profiles) Dynamic variable list dictionary
E000 EFFF	(Free for manufacturer) Dynamic program invocation dictionary
F000 FFFF	(Profiles) Dynamic program invocation dictionary

If the device supports function which are defined in further profile versions or other profiles, the corresponding objects appear in the profile areas.

Object Description of the Zero Objects

This object description is for the indexes to which no object is assigned (e.g. optional objects that are not supported).

Mapping the Device Function onto Communication

Object description: 'zero object' (see Table 19).

Table 19: Object description: 'zero object'

Object attribute	Value hex	Meaning
Index	xxxx	Zero object
Object code	00	Zero object

Table 20 contains the list of all parameters that can be accessed via communication.

Table 20: List of all parameters that can be accessed via communication

Index	Type	Object	Parameter name	m/o
6000	PDB structure	Record	PA data description	o
6001	PDB structure	Record	PA data description	o
6002	Octet string	Var	PA data enable	m
6003	Unsigned16	Var	PD monitoring time	o
6004	Integer16	Var	PD monitoring selection code	o
6005	Unsigned16	Var	Communication monitoring time	o
6006	Integer16	Var	Communication monitoring selection code	o
6007	Integer16	Var	Connection abort selection code	o
6008	Unsigned16	Var	Identifying number	o
6009	Unsigned8	Var	Parameter set identifier	o
600A	Unsigned32	Var	Series number	o
600B	Date	Var	Date of calibration	o
600C	Visible string	Var	Device description	o
600D	Visible string	Var	Model description	o
600E	Date	Var	Date of parameterization	o
600F	Octet string	Array	Function group description	m
6010	Octet string	Var	Process input data	m
6011	Octet string	Var	Process output data	m
6012	Boolean	Var	Write control	o
6013	Unsigned 16	Array	Conflict dictionary	o
6014		Zero		
....				
603E		Zero		
603F	Octet string	Var	Malfunction code	o

m = mandatory o = optional

7.2. Data Types

Boolean

Representation of the values TRUE or FALSE in an octet.

Notation: Boolean
Value range: TRUE or FALSE
Coding: FALSE is represented by the value 0,
TRUE by the value FF hex.

Representation of the value TRUE at the communication interface

	MSB							LSB
Byte in address = n	1	1	1	1	1	1	1	1

Representation of the value FALSE at the communication interface

	MSB							LSB
Byte in address = n	0	0	0	0	0	0	0	0

Integer

Integer value are signed quantities.

Notation: Integer8, Integer16, Integer32

Value range:

Data type	Value range	Length
Integer8	-128 _ i _ 127	1 octet
Integer16	-32768 _ i _ 32767	2 octets
	31 31	
Integer32	-2 _ i _ 2 - 1	4 octets

Coding: Two's complement representation
SB = 0: positive number including zero
SB = 1: negative number

Representation of an Integer8 at the communication interface

	MSB							LSB
Byte in addr = n	S	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Representation of an Integer16 at the communication interface

	MSB							LSB
Byte in addr = n	SB	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8
Byte in addr = n+1	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Representation of an Integer32 at the communication interface

	MSB							LSB
Byte in addr = n	SB	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
Byte in addr = n+1	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
Byte in addr = n+2	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
Byte in addr = n+3	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

Unsigned

Unsigned values are signless quantities.

Notation: Unsigned8, Unsigned16, Unsigned32

Value range:

Data type	Value range	Length
Unsigned8	0 <= i <= 255	1 octet
Unsigned16	0 <= i <= 65535	2 octets
Unsigned32	0 <= i <= 4294967295	4 octets

Coding: Binary

Representation of an Unsigned8 at the communication interface

	MSB							LSB
Byte in addr = n	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

Representation of an Unsigned16 at the communication interface

	MSB							LSB
Byte in addr = n	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
Byte in addr = n+1	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

Representation of an Unsigned32 at the communication interface

	MSB							LSB
Byte in addr = n	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
Byte in addr = n+1	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
Byte in addr = n+2	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}
Byte in addr = n+3	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

Floating Point

Notation: Floating point (4 octets)
Value range: see IEEE Std 754 Short Real Number (32 Bits)
Coding: see IEEE Std 754 Short Real Number (32 Bits)

	MSB							LSB	
Byte in addr = n	S	2^7	2^6	2^5	2^4	2^3	2^2	2^1	Exponent (E)
Byte in addr = n+1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	
Byte in addr = n+2	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}	Fraction (F)
Byte in addr = n+3	2^{-16}	2^{-17}	2^{-18}	2^{-19}	2^{-20}	2^{-21}	21^{-22}	2^{-23}	

Visible String

Notation: Visible string
Value range: see ISO 646 and ISO 2375: Defining registration number 2 + SPACE
Coding: see ISO 646

Representation of a visible string at the communication interface

	MSB LSB
Byte in addr = n	First character
Byte in addr = n+1	Second character
Byte in addr = n+2	Third character
Byte in addr = n+3	Fourth character
etc.	etc.

Octet String

Notation: Octet string
Coding: Binary

Representation of an octet string at the communication interface

	MSB LSB
Byte in addr = n	First octet
Byte in addr = n+1	Second octet
Byte in addr = n+2	Third octet
Byte in addr = n+3	Fourth octet
etc.	etc.

Date

The *date* data type consists of a calendar date and a time.

Notation: Date/time
Value range: 0 ms to 99 years
Coding: In 7 octets

Parameter	Value range	Meaning of the parameter
ms	0...59 999	Milliseconds
min	0...59	Minutes
DST	0,1	0: Standard time, 1: daylight saving time
RSV		Reserved
h	0...23	Hours
Wk-day	1...7	Weekday: 1 =Monday, 7 =Sunday
Mo-day	1...31	Day of the month
Month	1...12	Month
Year	0...99	Year (without century)

Representation of a date at the communication interface

	MSB							LSB	
Byte in addr = n	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	
Byte in addr = n+1	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	ms (0..59999)
Byte in addr = n+2	Reserved		2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	min (0..59)
Byte in addr = n+3	DST	Reserved		2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	h (0...23)
Byte in addr = n+4	Wk-day			Mo-day					Wk-day(1..7)
	2^{2}	2^{1}	2^{0}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	Mo-day (1...31)
Byte in addr = n+5	Reserved		2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	Month (0...12)
Byte in addr = n+6	Reser-ved	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	Year (0...99)

Time-Of-Day

The Time-Of-Day data type consists of a time and an optional date. The time is specified in milliseconds passed since midnight. At midnight the counting starts with the value zero. The date is represented in days relative to January 1, 1984. On January 1, 1984, the date specification starts with the value zero.

Notation: Time-Of-Day
Value range: $0 \_ i \_ (2^{28} - 1)$ ms
$0 \_ i \_ (2^{16} - 1)$ days
Coding: The time is represented as a binary value with 32 bits (4 octets). The first four (MSB) bits always have the value zero. The (optional) date specification is coded as a binary value in 16 bits (2 octets). Thus, Time-Of-Day is an octet string consisting of 4 or 6 octets.

Representation of a Time-Of-Day at the communication interface

	MSB							LSB	
Byte in addr = n	0	0	0	0	2^{27}	2^{26}	2^{25}	2^{24}	Number of milliseconds since midnight
Byte in addr = n+1	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
Byte in addr = n+2	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	
Byte in addr = n+3	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	
Byte in addr = n+4	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	Number of days since 1.1.84 (optional)
Byte in addr = n+5	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	

Time Difference

The Time Difference data type consists of a time specification in milliseconds and an optional number of days. The structure corresponds to the data type Time-Of-Day, however, it specifies a time difference.

Notation: Time-Difference
Value range: $0 \_ i \_ (2^{32} -1)$ ms
$0 \_ i \_ (2^{16} -1)$ days
Coding: The time is represented as a binary value with 32 bits (4 octets). The first four (MSB) bits always have the value zero. The (optional) day specification is coded as a binary value in 16 bits (2 octets). Thus, Time Difference is an octet string which consists of 4 or 6 octets.

Representation of a Time Difference at the communication interface

	MSB							LSB	
Byte in addr = n	0	0	0	0	2^{27}	2^{26}	2^{25}	2^{24}	Number of milliseconds
Byte in addr = n+1	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
Byte in addr = n+2	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	
Byte in addr = n+3	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	
Byte in addr = n+4	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	Number of days (optional)
Byte in addr = n+5	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	

Bit String

The following figure shows the numbering scheme for the individual bits of the bit string data type.

Only integer positive multiples of number 8 are permitted as length of the bit string (in bits).

Notation: Bit string
Value range: Binary

Representation of a Bit String at the communication interface

	MSB							LSB
Byte in addr = n	0	1	2	3	4	5	6	7
Byte in addr = n+1	8	9	10	11	12	13	14	15
Byte in addr = n+2				etc.				
etc.				etc.				

Mapping the Device Function onto Communication

The data types are stored as objects in the static type dictionary.

Static Type Dictionary

Index	Object code	Description	
(hex)		Symbol length	Symbol
1	5	7	Boolean
2	5	8	Integer8
3	5	9	Integer16
4	5	9	Integer32
5	5	9	Unsigned8
6	5	10	Unsigned16
7	5	10	Unsigned32
8	5	14	Floating point
9	5	14	Visible string
A	5	12	Octet string
B	5	4	Date
C	5	11	Time-Of-Day
D	5	15	Time-Difference
E	5	10	Bit string

A zero object is entered for non-implemented data types.

'Process Data Description Structure'

Depending on how the process data channel is implemented, the 'process data description structure' data type structure description is created. The data type structure description always contains one element for the process data length and then for every byte of the process data channel 2 elements for index and subindex each for input or output data .

Subindex	Meaning	Data type
1	Process data length	Unsigned8
2	1-Index process I/O data	Unsigned16
3	1-Subindex process I/O data	Unsigned8
. . .		
52	1-Index process I/O data	Unsigned16
53	1-Subindex process I/O data	Unsigned8

Mapping the Device Function onto Communication

Object description: ’process data description structure’ (see Table 21).

Table 21: Object description: ’process data description structure’

Object attribute	Value	Meaning
Index	0020	Process data description structure
Object code	06	Data type structure description
Number of elements	xx	n bytes (depend on the impl. of the bus connection, manufacturer-specific)
Data-type index	0005	Unsigned8
Length	1	1 byte
Data-type index	0006	Unsigned16
Length	2	2 bytes
Data-type index	0005	Unsigned8
Length	1	1 byte
...	...	...
Data type index	0006	Unsigned16
Length	2	2 bytes
Data type index	0005	Unsigned8
Length	1	1 byte

7.3. Application Data

This section describes the structure of the user data of the communication devices:

- Parameter description data;
- Parameter description data of the device parameters;
- Limit values of the parameter description data;
- Default setting of the parameter description data.

7.3.1. Parameter Description Data

The parameter description data describe a device parameter which triggers a defined device action in a device function.

The parameter description data consists of the following data:

- Module type;
- Quantity index;
- Unit index;
- Offset;
- Resolution (measuring range/numeric range);
- Min./max values.

Module Type

This parameter, for example, provides information on how the parameter value is encoded.

Quantity Index

The quantity index provided the physical quantity of the measurement value, measurement range and the lower measurement range value in an encoded form. The value range is 0 to 256. The coding is given in the table.

Unit Index

The unit index specifies the unit of measured value, the measurement range and the lower measuring range value. The unit index has the value range - 127 to 128. The value of the unit index corresponds to the respective power of ten of the standard unit, thus

unit index 0 for 10^0

unit index 3 for 10^3

unit index -3 for 10^{-3}

etc.

The listed unit indexes for SI-compatible units (unit index < 64) should only be taken as an example. The unit indexes for further SI-compatible prefixes (*pico-* etc.) result analogously. Unit indexes which are larger than +64 have a special meaning which has to be taken from the table. These units are, for example, the units day, hour, minute or units which are not SI-compatible, such as Fahrenheit.

Offset

This subparameter specifies the offset from the zero point. The value range is Integer16.

Resolution

This subparameter specifies the resolution of the measurement value. The parameter is calculated according to the following formula:

$$R = \frac{MR}{N}$$

where:

R resolution;
MR measuring range;
N number range.

The physical measurement value is calculated in accordance with the following formula:

$$PM = M \times R$$

where:

PM physical measurement value;
M measurement value;
R resolution.

Example 1:

$$R = \frac{16\ \mathrm{V}}{65\ 565}$$

$$PM = 32\ 782 \times \frac{16\ \mathrm{V}}{65\ 565} = 8\ \mathrm{V}$$

Example 2:

$$R = \frac{100\ \mathrm{V}}{10\ 000}$$

$$PM = 1\ 000 \times \frac{100\ \mathrm{V}}{10\ 000} = 10\ \mathrm{V}$$

Min./ Max. Values

These two parameters specify the minimum and the maximum value.

Example:

Subparameter	Initialization data	Meaning
Module type	12 bits	12 bit converter
Quantity index	22	Ampere
Unit index	-3	10-3
Offset	4	4 mA
Resolution	16 / 65565	4-20 mA (16 mA)
Min./Max. values		

Table 22: Quantity index and unit index

Physical quantity	Quantity index	Unit	Unit index
	0	without dimension	0
Length	1	Meter	0
		Millimeter	-3
		Kilometer	3
		Micrometer	-6
Area	2	Square meter	0
		Square millimeter	-6
		Square kilometer	6
Volume	3	Cubic meter	0
		Liter	-3
		Milliliter	-6
Time	4	Second	0
		Minute	70
		Hour	74
		Day	77
		Millisecond	-3
		Microsecond	-6
Active power	9	Watt	0
		Kilowatt	3
		Megawatt	6
		Milliwatt	-3
Apparent power	10	Voltampere	0
		Kilovoltampere	3
		Megavoltampere	6
Revolutions per unit time	11	1/second	0
		1/minute	73
		1/hour	74
Angle	12	Radiant	0
		Second	75
		Minute	76
		(Old)Degree	77
		New degree	78
Speed	13	Meter/second	0
		Millimeter/second	-3
		Millimeter/minute	79
		Meter/minute	80
		Kilometer/minute	81
		Millimeter/hour	82
		Meter/hour	83
		Kilometer/hour	84
Torque	16	Newtonmeter	0
		Kilonewtonmeter	3
		Meganewtonmeter	6
Temperature	17	Kelvin	0
		Degree Celsius	94
		Degree Fahrenheit	95
Electrical voltage	21	Volt	0
		Kilovolt	3
		Millivolt	-3
		Microvolt	-6
Electrical current	22	Ampere	0
		Milliampere	-3
		Kiloampere	3
		Microampere	-6
Ratio	24	Percent	0
Frequency	28	Hertz	0
		Kilohertz	3
		Megahertz	6
		Gigahertz	9
Steps	32	Steps	0
Encoder resolution	33	Steps/revolution	0
Throughput	34	Cubic meter/second	0
		Liter/second	-3
		Millimeter/second	-6
		Cubic meter/minute	85
		Liter/minute	86
		Millimeter/minute	87
Acceleration	35	$1/Second^2$	0

7.3.1.1. Parameter Description Data of the Device Parameters

The parameter description data corresponds to the predefined structure in the InterBus-S Sensor/Actuator Profile. The parameters of a device function are described with a parameter description. The parameters and the parameter description are stored in the device as device parameters.

Example:

Index	Type	Object	Parameter name	m/o
6050	Integer16	Var	Parameter value	
6051	Unsigned8	Var	Module type	
6052	Unsigned8	Var	Quantity index	
6053	Integer8	Var	Unit index	
6054	Integer16	Var	Offset	
6055	Integer16	Array	Resolution	
6057	Integer16	Array	Min/Max values	

7.3.2. Limit Values of the Parameter Description Data

The limit values of the parameter description data specify to which values a setting parameter can be set. This data must be specified in the data sheet.

7.3.3. Default Setting of the Parameter Description Data

The default setting of the parameter description data to which the sensor is set upon delivery. This data must be specified in the data sheet

- Module type;
- Quantity index;
- Unit index;
- Offset;
- Resolution (measurement range/numeric range);
- Min./max. values.

7.3.4. Selection Code

The selection code allows to select one function from several profile-specific or manufacturer-specific functions. The code for the profile-specific function is a value in the range from 0 to 32767. The code for the manufacturer-specific functions is a value in the range from -32768 to -1. The selection of non-defined functions is confirmed negatively.

Selection code	Meaning of the selection function
-32768 ... -1	Manufacturer-specific
0 ... 32767	Profile-specific

8. Operating Phases of the Application

This section describes the possible operating phases of the device. The section is broken down as follows:

- Initialization/abort;
- Operation;
- Startup phase and configuration phase

8.1. Initialization/Abort

Initialization

After power on or a reset of the device, the initialization is started. The following actions can be carried out:

- initialization of the communication interface
- initialization of the process data
- initialization of the parameters

Initialization of the communication interface

The communication relationship is parameterized with the following values:

PMS-CRL-Parameter	Value
PMS-Service-Supported	Read.ind/res,Write.ind/res (all other services are optional)
Max-Outstanding-Services	1
Max-PDU-Sending-High-Prio	0
Max-PDU-Sending-Low-Prio	Manufacturer-specific
Max-PDU-Receiving-High-Prio	0
Max-PDU-Receiving-Low-Prio	Manufacturer-specific

Initialization of the process data

The process input and output registers are preset to the value zero.

Initialization of the parameters

During initialization, the following communication objects must be parameterized with the corresponding stored values - or if not available - with substitute values.

Communication object	Substitute value	
Process data monitoring time	65535	disabled
Process data monitoring selection code	0	no response
Communication monitoring time	65535	disabled
Communication monitoring selection code	0	no response
Connection abort selection code	0	no response

Abort

The following is performed:

- a reset of the process data

When the communication and application unit are independent of each other, the process input data is set to zero, if the application unit fails.

9. Communication Profile

9.1. Layer 1

This section describes the interface to the transmission medium. The possible interfaces are selected in further profiles.

9.1.1. Installation Remote Bus Interface

Connector

CONINVERS connector (IP 65)

or

9-pos. subminiature D connector (IP 20), up to 1A

9.1.2. Remote Bus Interface

Transmission Protocol

2-wire remote bus

Connector

subminiature D 9-pos. (male) for the "incoming interface" to the previous device

subminiature D 9-pos. (female) for the "outgoing interface" to the next device

9.1.2. Local Bus Interface

Transmission Protocol

8-wire local bus

Connector

subminiature D 15-pos. (male) for the "incoming interface" to the previous device

subminiature D 15-pos. (female) for the "outgoing interface" to the next device

9.2. Layer 2

This section describes all definitions concerning Layer 2.

9.2.1. Configuration of the InterBus-S Registers

The arrangement of the data registers of an InterBus-S device, and thus the addressing on the I/O level is defined in the following. Example of an InterBus-S device with a 2-byte parameter channel:

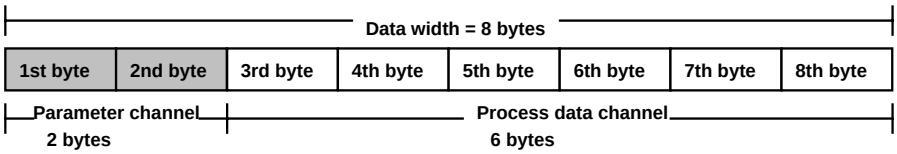

Figure 23: Relations between data width, process data channel and parameter channel

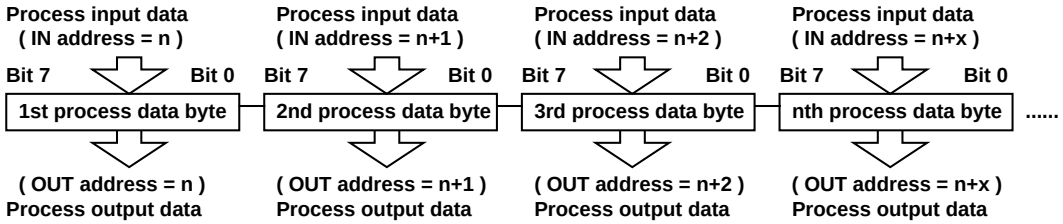

Figure 24: Addressing the process data

Process data direction:

- Process input data is transmitted from the device to the bus system.
- Process output data is transmitted from the bus system to the device.

9.2.2. Identification of the InterBus-S Devices

The ID code consists of the following:

b15 b13	b12 b8	b7 b0
Message	**Data width**	**ID code**

Message

With this bit in the ID code messages are transmitted to the controller board.

Table 23: Messages

b 15	b 14	b13	Meaning
1	x	x	Device message
x	1	x	CRC error
x	x	1	Reserved

Device message
This message is generated when the device detected a malfunction of the periphery. A malfunction in the periphery can be defined in detail in further profiles.

CRC error
This message is generated if transmission errors were detected (from the protocol chip).

Data Width

The data width indicates how many bits the device used on the bus. If a device has, for example, 16-bit inputs and 32-bit outputs, it occupies 32 bits (4 bytes) in the ring (the higher value is decisive). The length of the parameter channel is defined in the ID code.

Table 24: Data width

Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Data with
0	0	0	0	0	0
0	1	1	0	0	1 Bit
0	1	1	0	1	2 Bits
0	1	0	0	0	4 Bits
0	1	0	0	1	1 Byte
0	1	0	1	0	12 Bits
0	0	0	0	1	2 Bytes
0	1	0	1	1	3 Bytes
0	0	0	1	0	4 Bytes
0	0	0	1	1	6 Bytes
0	0	1	0	0	8 Bytes
0	0	1	0	1	10 Bytes
0	1	1	1	0	12 Bytes
0	1	1	1	1	14 Bytes
0	0	1	1	0	16 Bytes
0	0	1	1	1	18 Bytes
1	0	1	0	1	20 Bytes
1	0	1	1	0	24 Bytes
1	0	1	1	1	28 Bytes
1	0	0	1	0	32 Bytes
1	0	0	1	1	48 Bytes
1	0	0	0	1	52 Bytes
1	0	1	0	0	64 Bytes
1	0	0	0	0	Reserved
1	1	x	x	x	Reserved

x = "don't care"

ID Code

Description of the device function		ID Code (dec)	ID Code (hex)
Bus coupler			
BK with 8-wire local bus branch	BK-8L-LB	52	34
BK with 2-wire local bus branch	BK-2L-LB	8	08
BK with I/O addresses and 8-wire local bus branch or 2-wire remote bus branch	BK-I/O	11	0B
BK with 2-wire remote bus branch	BK-2L-RB	12	0C
Remote bus device, digital			
Digital device with output addresses	DO	1	01
Digital device with input addresses	DI	2	02
Digital device with input and output addresses	DIO	3	03
Profile-compatible digital device with output addresses	PROFILE DO	13	0D
Profile-compatible digital devices with input addresses	PROFILE DI	14	0E
Profile-compatible digital devices with input and output addresses	PROFILE DIO	47	2F
ISO valve terminals	ISO valve terminals	5	05
Remote bus device, analog			
Analog devices with output addresses	AO	49	31
Analog devices with input addresses	AI	50	32
Analog devices with input and output addresses	AIO	51	33
Profile-compatible analog devices with output addresses	PROFILE AO	53	35
Profile-compatible analog devices with input addresses	PROFILE AI	58	3A
Profile-compatible analog devices with input and output addresses	PROFILE AIO	59	3B
ENCOM with input addresses	ENCOM	54	36
ENCOM with input and output addresses	ENCOM	55	37
Remote bus device with parameter channel			
Devices with parameter channel (2 PCP words)	PA channel	240	F0
Devices with parameter channel (4 PCP words)	PA channel	241	F1
Devices with parameter channel (1 PCP word)	PA channel	243	F3
DRIVECOM (2 PCP words)	DRIVECOM	224	E0
DRIVECOM (4 PCP words)	DRIVECOM	225	E1
DRIVECOM (1 PCP word)	DRIVECOM	227	E3
ENCOM (2 PCP words)	ENCOM	244	F4
ENCOM (4 PCP words)	ENCOM	245	F5
ENCOM (1 PCP word)	ENCOM	247	F7
Profile-compatible devices (2 PCP words)	PROFILE PA channel	228	E4
Profile-compatible devices (4 PCP words)	PROFILE PA channel	229	E5
Profile-compatible devices (1 PCP word)	PROFILE PA channel	231	E7
Local bus device, digital			
Digital devices with output addresses	DO	189	BD
Digital devices with input addresses	DI	190	BE
Digital devices with input and output addresses	DIO	191	BF
Profile-compatible digital devices with output addresses	PROFILE DO	181	B5
Profile-compatible digital devices with input addresses	PROFILE DI	182	B6
Profile-compatible digital devices with input and output addresses	PROFILE DIO	183	B7
Wrenching controllers	Wrenching controllers	187	BB
Local bus devices, analog			
Analog devices with output addresses	AO	125	7D
Analog devices with input addresses	AI	126	7E
Analog devices with input and output addresses	AIO	127	7F
Profile-compatible analog devices with output addresses	PROFILE AO	121	79
Profile-compatible analog devices with input addresses	PROFILE AI	122	7A
Profile-compatible analog devices with input and output addresses	PROFILE AIO	123	7B
ENCOM with input addresses	ENCOM	102	66
ENCOM with input and output addresses	ENCOM	103	67
Local bus device with parameter channel			
Devices with parameter channel (2 PCP words)	PA channel	220	DC
Devices with parameter channel (4 PCP words)	PA channel	221	DD
Devices with parameter channel (1 PCP word)	PA channel	223	DF
DRIVECOM (2 PCP words)	DRIVECOM	192	C0
DRIVECOM (4 PCP words)	DRIVECOM	193	C1
DRIVECOM (1 PCP word)	DRIVECOM	195	C3
ENCOM (2 PCP words)	ENCOM	212	D4
ENCOM (4 PCP words)	ENCOM	213	D5
ENCOM (1 PCP word)	ENCOM	215	D7
Profile-compatible devices (2 PCP words)	PROFILE PA channel	216	D8
Profile-compatible devices (4 PCP words)	PROFILE PA channel	217	D9
Profile-compatible devices (1 PCP word)	PROFILE PA channel	219	DB