Basic profile

10

V3.0

2018-04-19

09

Profile:	Basic
Profile number:	10
Version:	V3.0
Version counter:	09
Date:	2018-04-19
Publisher:	INTERBUS Club Deutschland e.V. Head office Postfach 1108, 32817 Blomberg, Germany Telephone : +49 5235 34 21 00 Fax : +49 5235 34 12 34

Copyright by INTERBUS Club Deutschland e.V.

All illustrations and descriptions have been created and checked to the best of the author's knowledge. However, this does not release the user from the responsibility to undertake their own examinations and tests. We reserve the right to make modifications of any type, particularly those that serve the purpose of technical progress.

INTERBUS Club Deutschland e.V. assumes no liability for erroneous handling or damage resulting from a failure to observe the information contained in this profile.

This profile, including all illustrations contained herein, is copyright protected.
Use of this profile by any third party deviating from the copyright provisions is forbidden.
Subject to modification

List of revisions

Index	Date	Modifications, additions or explanations
1.00	2002-10-14	Initial version
1.01	2004-08-12	• Supplement of the channel number in diagnostic object 0x0018 and the following • Supplement of priorities 81, 82, 83 “Error is removed”, diagnostic object 0x0018 and the following • Revision/supplement of the fault codes • Expansion of the handling description for diagnostic messages • Object 0x000F, DeviceProfile mandatory • Preface and appendix added • Editorial revision • Introduction of objects -0x002E – CheckSum -0x002F - PDOUT_Subst -0x0030 - PF_Code -0x0031 - PDIN_Subst -0x0032 - IBS_ID • Definition of values for objects -0x0024 – IBSResetCode -0x0020 – PDTimeoutCode
1.02	2005-01-28	• Error correction (Re/Gr/formats/etc.) • Object 0x0019, “ResetDiag”, name and meaning adapted • Object 0x000F, “DeviceProfile”, type changed • Fault codes expanded: 2344; 2345; 341X; 342X; 5230
1.10	06.06.20	• Object 0x0018, “DiagState.Channel” changed to “DiagState.Channel/Group”, meaning adapted • Object 0x0018, “DiagState. MoreFollows”, meaning adapted • Object 0x001A, “GetErrorRepMethod”, name and description expanded • Object 0x001D, “Password” adapted • Object 0x0025, “PDIN”, additional explanation • Object 0x0026, “PDOUT”, additional explanation • Object 0x002E, “CheckSum”, type changed UINT16 => UNIT32 • Object 0x0033, “DiagStateChannelNumber”, newly added • Object 0x0034, “DiagStateAddValue”, newly added • Object 0x0035, “NoOfModules”, newly added • Object 0x0036, “SubBusStructure”, newly added • Object 0x0037, “DeviceType”, newly added • Object 0x0038, “ObjDescrReq”, newly added • Object 0x0039, “ObjDescr”, newly added • Object 0x003A, “VersionCount”, newly added • Object 0xE805, “ObjDescrLong”, newly added • Object 0xE800, “DiagStateLong”, partially described as a reference to avoid double descriptions • Modular devices have been described. The Invoke ID, now “model number”, is used for module addressing • List of approved communication error types revised and supplemented • Data“Bit-String” data type added. • “Communication objects” section revised, restructured, and supplemented • “Object description” section added • Addition of error class 8, error code 1, profile-specific

1.11	2009-10-06	• Additional fault code revised • Handling of object 0x0038 “ObjDescrReq” and object 0x0039 “ObjDescr” expanded • Error class “8”, other, error code “1”, profile-specific, corrected consistently • Visible String data type (always terminated “0x00”) • Object 0x0037, “DeviceType”, revised slightly • Object 0x0008, “ProductID” renamed “SerialNo” to avoid misunderstandings • Fault codes 6800 and 6810 added, A000 and the following expanded and explained in more detail • Explanation of the object area for modular devices • Object 0x003B, “PDIN_Descr”, newly added • Object 0x003C, “PDOUT_Descr”, newly added • Object 0x0019, “ResetDiag”, expanded and substantiated
1.12	2010-11-19	• Device family supplemented by digital/analog IN/OUT • List of the data types completed • LanguageCode corrected • Length for visible string corrected • Object 0x003B, “PDIN_Descr”, revised, new types added • Object 0x003C, “PDOUT_Descr”, revised, new types added • Object 0x000C, “FirmwareVersion” and 0x000B “PChVersion”, expanded by entries if no FW/PChVersion is available • Handling of priorities in diagnostic object 0x0018 “DiagState” explained • Object 0x0019, “ResetDiag”, expanded and substantiated • Object 0x0018.5, “ResetDiag.MoreFollows”, substantiated • Additional fault codes added • Object 0x0037, “DeviceType”, error corrected R/W => R, can never have been R/W • Subindex “Access Rights”, “DisplayFormat”, “Resolution” and “Offset” added to object 0x0039 “ObjDescr”, description supplemented. • Size index for “power” and “el. resistance” added • Object 0x0020, 0x0024, 0x0030 UINT16 => Array of UINT16 changed and annotated •
2.0	2011-12-07	• Basic profile “neutralized”, i.e., reference to INTERBUS and PCP has been removed. • Comments on the version counters added • Object 0x003D, “WakeUpTime”, added • ResetCode explained in more detail • Object 0x002D, ResetParam code “02”, added • **Object 0x0039, “ObjDescr”, incompatible, revised (minimum and maximum moved)** • Translation table for the object names added to appendix • Description of download services (download write, upload read), supplemented and made more specific. • Object 0x0011 error in the string length (10+1) corrected • Object 0x0005, “Capabilities”, added • Object 0x0038, “ObjDescrReq”, length corrected

3.0	2018-04-19	• New additional fault code “8.2 Hardware is temporarily faulty” added • Description “Unique, consecutive fault number” substantiated • Fault codes revised and explanations improved • Various new fault codes determined, such as A014, A025, 6340, 8F00 • Warning/fault (alarm) meaning substantiated • 0x0005 “Capabilities” “Safety0” “ChPDWh_0” “SBM_0” added • Remedy of various grammatical and linguistic errors • Representation “N+1” used consistently • Hex notations; 0x unified • 3150 Polarity of supply voltage reversed • Object 0x0004, “DeviceFamily”, supplemented by “Safety” • Object 0x0005, “Capabilities”, data type changed: visible string => octet string • Object 0x0013, “DeviceDescFile” renamed to “OnBoardDeviceDescFileName”, and description substantiated. • **Object 0x0018 completely revised** • Object 0x0019, “ResetDiag”, rights changed: W = R/W • Object 0x0019, “ResetDiag”, meaning substantiated and supplemented • Object 0x001A, “GetErrorRepMethod”, description of bit 3 deleted • Object 0x0020, “PDTimeoutCode”, values added • Object 0x0024, “ResetCode”, values added • **Object 0x002A, ConflictDictionary, incompatible, modified** • Object 0x002D, “ResetParam”, rights now R/W and becomes mandatory, value 0x03 added • “Replacement value” definition renewed • Object 0x0030, “PF_Code”, values added • Object 0x0033 Withdrawn • Object 0x0034 Withdrawn • **Object 0x0035 NoOfModules => Sub-BusInfo including meaning changed** • Object 0x0036 ActSubBusStructure revised • Object 0x0039 set to R/W_D. Function explained. • Object 0x0039.10 ObjectDescr.RNR (R 2 bytes) changed in UINT16(Dec) • Object 0x0039.12 ObjDescr. access rights bit 0 - bit 3 changed due to corresponding implementation of active = 1 to “Active = 0” • Object 0x0039.12 ObjDescr. access rights bit 4 “Avoid Presentation” added • Object 0x0039.13, ObjDescr.DiplayFormat: 0x08, “date” added • Object 0x0039.16, ObjDescr.Symbol: description of bit strings added • Object 0x003A VersionCount description added and substantiated • Objects 0x003B, PDIN_Descr and 0x003C PDOUT_Descr type “SFCH”, “SafeI”, “SafeO_F” - Channel for safe data newly added, mandatory and description added • Object 0x003D, WakeUpTime, resolution added • Object 0x003D, “WakeUpTime”, description substantiated, legacy information added • Object 0x003E, “EnergyMgmt”, newly added • Object 0x0040, “ListOfObjToRestore”, newly added • Object 0x0041, “RefSubBusStructure”, newly added • Object 0x0042, “DevicesStatus”, newly added • Object 0x0043, “Sub-BusBehaviour”, newly added • Object 0x0044, “ControlSub-Bus”, newly added • Object 0x0045, “InitFWDownload”, newly added • Object 0x0047, “AddInfo”, newly added • Object 0x0047.01, “AddInfo.SafetyProtType”, newly added • Object 0x0047.02, “AddInfo.SafetyProtVers”, newly added • Object 0x0047.03, “AddInfo.LegacyInfo”, newly added

		• Object 0xE800, “DiagStateLong”: adapted to changed object 0x0018. • Object 0xE809, “BackUpDataCompr”, added • Object 0xE807, “DeviceFW”, added • Object 0xE808, “OnBoardDeviceDescFile”, added • Object 0xE806, “ComplDiagState”, newly added • Editorial revision of “Notification via ... the DiagState object” section • Editorial revision of “ConflictDictionary” section • “Service-Parameter Error Type” definition added • Overview of the data objects used added • “Evaluation in the case of complex data objects” section added • Presentation form amended for all objects • Messages, information, notifications (only to Prio 83) • Device family list 0004 supplemented by temperature modules • Section 8.2.11. “Explanation on handling diagnostic object 0x0018” added. • “Busy” function introduced • Error code (8) “B for Busy” introduced • “Object Record” and “Object Array” descriptions changed and error event described • Additional codes added (0x005x), error class 0x08, error code 0x01 • Additional codes added (0x001D, 0x001E) • Generator polynomials defined for upload read and download write • “Notification via reading the DiagState object” section revised (message via bit in PD omitted) • Boolean definition added: “True = 0xFF, False = 0x00” • R/WD introduced • Reserved bits/bytes = “0” specified • “Safety” introduced to precede device ranges • “Information on handling individual parameters by the master” section added • “Block parameterization” section substantiated • “Addressing” section added • “Substitute value behavior/switch-on behavior” section added • “Volatile and non-volatile parameters” section added • “Firmware download” section added • “Functional safety - safety” section added • The term “Terminal” was introduced • Description of the domain variable added • Macro services and domain variables/var lists set in relation to one another • Lengths and object types standardized within the tables • Communication profiles “636” and “637” added • Error types instead of communication error codes • “Undefined” added for data types and data codes 0x00 • “PDU length” section added • Representation of additional codes revised • Dependency information: explanation on dependencies added • “Parameter record identification” section revised
		• 0x001E expanded by submodule and subindex • Double description in object 0x002B ParamSet removed • 0x002D ResetParam reset to factory defaults, legacy reset revised • 0x003E.01 EnergyMgmt.ActualMode areas reserved • 0x0039.0C “ObjDescr.Access Rights” revised • 0x0039.07 “ObjDescr.UnitCode” supplemented and revised • 0x0040 “ListOfObjToRestore” supplemented with submodule number • 0x0048 => 0xE806 => 0xE809 numbers replaced • 0xE809 BackUpDataCompr description substantiated • 0xC000 – 0xC07F “ProjBasProf” introduced • Capabilities changed back to visible string and some IDs shortened • Alignment of 0x002A, 0x0035 objects adapted to 16 bits • Invoke ID deleted • Re/Gr word forms corrected • CurrDiagState renamed => ComplDiagState • intDeviceDescFile => OnBoardDeviceDescFile • intDeviceDescFileName => OnBoardDeviceDescFileName • Projection of basic profile changed to subbus module addressing • 0x0035.06 “SubBusInfo.StartAddressPBP” => “SubBusInfo.SelectSubBusModule” deleted • 0x002A.01 ConfGrNo conflict group number added • Additional Code 0x0024: “Index not available” added

Contents

1 Foreword

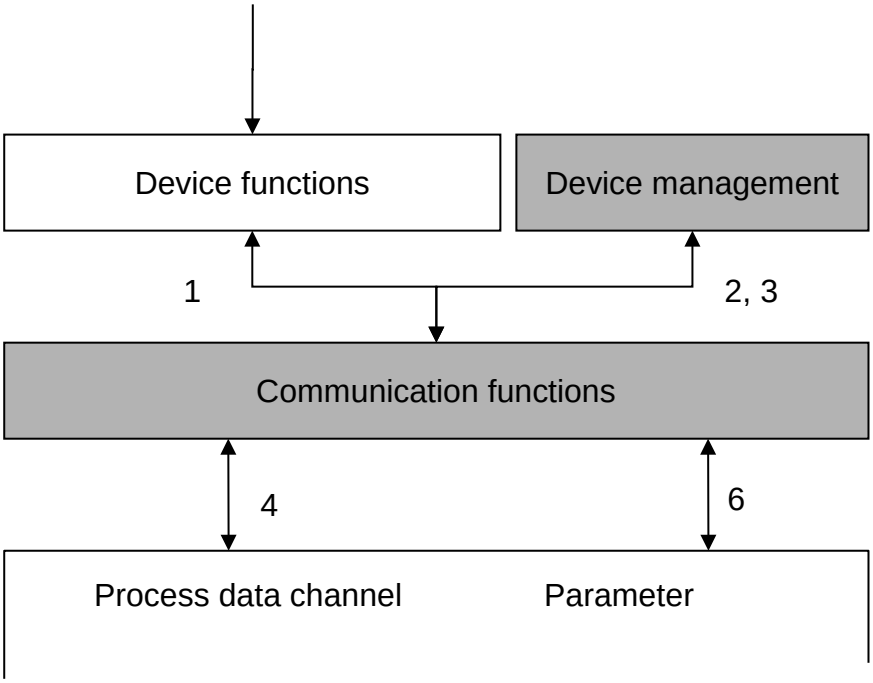
For factory-automated production of industrial sensors and actuators there is an ever-increasing need for high-performing and flexible systems. Intelligent field devices can fulfill these requirements. However, they must support open and standardized communication capabilities in order to be fully integrated in complex production procedures.
The basic thinking behind open systems is to enable the exchange of information between application functions implemented on devices from different manufacturers. This includes specified application functions, a uniform user interface for communication and a uniform transmission medium.

INTERBUS Club Deutschland e.V. has set itself the task of standardizing the most important field device functions and bringing them together in this profile. To enable the definition of field device functions regardless of the communication medium, an internationally recognized and standardized IEC 61158 user interface was used for communication. This provided continuity to the MMS (Manufacturing Message Specification ISO/IEC 9506).
No specific fieldbus system has been selected as a transmission medium. It must only meet the requirements for field communication in terms of realtime behavior and a standardized user interface.
The basic profile is aimed at all users and manufacturers of field devices that are to be operated on a fieldbus. This profile definition is a useful addition to standardized communication for the user and provides universal agreement on data content and device behavior. These function definitions standardize the important field device parameters. When using these standard parameters, devices from different manufacturers display uniform behavior on the communication medium.
An independent specialist body will be set up for conformance testing and certification of products with the basic profile. As standardization work continues, further additions can be expected.

2 Application and device properties

2.1 General

This section describes the entire application in terms of communication. The application is divided into the following function blocks:



Device functions
The device functions execute all the device-specific functions.

Device management
The device management manages the information, features and capabilities of a device. A non-volatile memory can be used for this.

Communication functions
The communication functions execute all the communication-specific functions.

Interaction between the function blocks
1 Process variables
2 Process data from the higher-level control system to the device functions
3 Process data from the device functions to the higher-level control system
4 Storing device information for management, diagnostics, identification
5 Reading device information for management, diagnostics, identification
6 Mapping to the process data channel
7 Mapping to the parameter channel

3 General

In addition to information about the current status of the device, modern field device management and diagnostics also include information about the device itself, historical status information, error images, etc.
Previously, very little of the information was provided as standard.
The aim of this document is to define a basic profile that will enable standardized management of devices that are equipped with a parameter channel.
This basic profile enables device manufacturers to provide end users (as well as tools) with considerably more information about their devices and associated features.
This “extra” information simplifies device startup, in particular, due to this standardization. As for service work, there is also now the option to access certain diagnostic information in a uniform and thus faster manner.

Understanding the profile:

> “Information and sequences that are similar in terms of content are also displayed in a standardized form, which aids manufacturers and users instead of restricting them.”

In order to meet this requirement, the relevant information is stored consistently in standard objects. Mechanisms are defined that control access to these objects and specify standard error types in the event errors.

4 Overview

4.1 Index areas of the communication objects

The communication objects in the basic profile are divided into the following index areas:

Index (hex)	**Name**	**Object type**
0	Object dictionary	-
0001 – 0047	General standard objects These objects are further described in this document.	Variable
0048 – 007F	Reserved	
0080 – 5FFF	Manufacturer-specific application objects Device manufacturers define their device-specific variable objects in this area.	Variable
6000 – BFFF	Profile-specific application objects Further specific application objects can be found in the relevant profiles (e.g., Drivecom, sensor/actuator, etc.)	Variable
C000 – C07F	Projection of basic profile for subbuses	Variable
C080 – DFFF	Reserved	
E000 – E3FF	Static variable lists - access via macro services	Variable list
E400 – E7FE	Dynamic variable lists - access via macro services	Variable list
E7FF	Creating and reading variable list definitions - access via macro services	Variable list structure
E800 – E809	General domain variables - access via macro services These objects are further described in this document.	Domain variable
E80A – E83F	Reserved domain variables - access via macro services	Domain variable
E840 – E8FF	Device-specific domain variables - access via macro services Device manufacturers define their device-specific domain variables in this area.	Domain variable
E900 – FFFF	Reserved	

4.2 Overview of the standard objects

Index (hex)	Object name	R/W	Length (in bytes)	Data type	M/O/D
0001	VendorName	R	57+1, max.	Visible string	**M**
0002	VendorID	R	6+1	Visible string	O
0003	VendorText	R	57+1, max.	Visible string	O
0004	DeviceFamily	R	57+1, max.	Visible string	**M**
0005	Capabilities	R	N x 8	Array of octet strings[8]	**M**
0006	ProductFamily	R	57+1, max.	Visible string	O
0007	ProductName	R	57+1, max.	Visible string	**M**
0008	SerialNo	R	57+1, max.	Visible string	O
0009	ProductText	R	57+1, max.	Visible string	O
000A	OrderNumber	R	57+1, max.	Visible string	**M**
000B	HardwareVersion	R	51, max.	Record (2 elements)	**M**
000C	FirmwareVersion	R	51, max.	Record (2 elements)	**M**
000D	PChVersion	R	51, max.	Record (2 elements)	**M**
000E	CommProfile	R	4+1, max.	Visible string	**M**
000F	DeviceProfile	R	4+1, max.	Visible string	**M**
0010	Reserved				
0011	ProfileVersion	R	51, max.	Record (2 elements)	**M**
0012	VendorURL	R	57+1, max.	Visible string	O
0013	OnBoardDeviceDescFileName	R	57+1, max.	Visible string	O
0014	Location	R/W	57+1, max.	Visible string	O
0015	EquipmentIdent	R/W	57+1, max.	Visible string	O
0016	ApplDeviceAddr	R/W	2	UINT16	O
0017	Language	R/W	56, max.	Record (2 elements)	**M**
0018	DiagState	R	207, max.	Record (6 elements)	**M**
0019	ResetDiag	R/W	1	UINT8	O
001A	GetErrorRepMethod	R/W	1	UINT8	O
001B	TestMode	R/W	2	UINT16	O
001C	ControlTrace	R/W	1	UINT8	O
001D	Password	R/W	40, max.	Octet string[40]	O
001E	SetPassword	R/W	45, max.	Record (5 elements)	O
001F	PDTimeout	R/W	2	UINT16	O
0020	PDTimeoutCode	R/W	N x 2	Array of UINT16 (N elements)	O
0021	PChTimeout	R/W	2	UINT16	**M**
0022	PChTimeoutCode	R/W	2	UINT16	O
0023	AbortCode	R/W	2	UINT16	O
0024	ResetCode	R/W	N x 2	Array of UINT16 (N elements)	O
0025	PDIN	R	PD length	Octet string[N]	**M**
0026	PDOUT	R/W	PD length	Octet string[N]	**M**
0027	GetExRight	R/W	1	UINT8	O
0028	ChangePDSet	R/W	2	UINT16	O
0029	ParamSetWriteControl	R/W	1	UINT8	O
002A	ConflictDictionary	R	N x 8	Array of records (N x 6 elements)	D
002B	ParamSet	R/W	2	UINT16	O
002C	ParameterMoment	R/W	20	Record (2 elements)	O
002D	ResetParam	R/W	1	UINT8	**M**
002E	ParamHash	R	4	UINT32	O
002F	PDOUT_Subst	R/W	PD length	Octet string[N]	D
0030	PF_Code	R/W	N x 2	Array of UINT16 (N elements)	O
0031	PDIN_Subst	R/W	PD length	Octet string[N]	D
0032	FieldBus_ID	R	3	Record (2 elements)	O
0033	*DiagStateChannelNo*	*R*	*3*	*Record (2 elements)*	*X*
0034	*DiagStateAddValue*	*R*	*6*	*Record (2 elements)*	*X*
0035	SubBusInfo	R	16	Record (6 elements)	O

0036	ActSubBusStructure	R/W	N x 8	Array of records (3 elements)	D
0037	DeviceType	R	8	Octet string[8]	**M**
0038	ObjDescrReq	R/W	3	Record (2 elements)	D
0039	ObjDescr	R/W_D	58, max.	Record (16 elements)	D
003A	VersionCount	R	8	Array of UINT16 (4 elements)	**M**
003B	PDIN_Descr	R	N x 12	Array of records (N x 3 elements)	**M**
003C	PDOUT_Descr	R	N x 12	Array of records (N x 3 elements)	**M**
003D	WakeUpTime	R	2	UINT16	**M**
003E	EnergyMgmt	R/W	5	Record (2 elements)	O
003F	Reserved				
0040	ListOfObjToRestore	R	N x 4	Array of records (3 elements)	**M**
0041	RefSubBusStructure	R/W	N x 8	Array of records (3 elements)	D
0042	ModuleStatus	R	N x 1	Array of bit strings[8]	D
0043	SubBusBehaviour	R/W	3	Record (3 elements)	D
0044	SubBusControl	R/W	1	UINT8 (hex)	D
0045	InitFWDownload	R/W	58, max.	Record (5 elements)	O
0047.01	AddInfo.SafetyProtType	R	8	Octet string[8]	D
0047.02	AddInfo.SafetyProtVers	R	8	Octet string[8]	D
0047.03	AddInfo.LegacyInfo	R	8	Octet string[8]	D
C000 – C07F	Projection of basic profile for subbuses	Def.	N	Complies in all respects with the corresponding object in the basic profile	D
E800	DiagStateLong	UR	N	Domain variable record	O
E801	DiagHistory	UR	N	Domain variable record	O
E802	DiagHistoryLong	UR	N	Domain variable record	O
E803	TraceBuffer	UR	N	Domain variable octet string	D
E804	LanguageAvailable	UR	N	Domain variable record	O
E805	ObjDescrLong	UR	N	Domain variable record	O
E806	ComplDiagState	UR	N	Domain variable record	O
E807	Device FW	UR/DW	N	Domain variable octet string	D
E808	OnBoardDeviceDescFile	UR	N	Domain variable octet string	O
E809	BackUpDataCompr	UR/DW	N	Domain variable octet string	O

R = Read only
W = Write only
R/W = Read/write
R/W_D = Read/write dependent
UR = Upload Read
DW = Download Write

M/D/O
M = Mandatory
O = Optional
D = Dependent

For details on the abbreviations, see Appendix.

5 Communication objects

The communication objects are not described in an object dictionary, instead they must usually be recognized implicitly by the user, e.g., from a description of the objects in the user manual for the relevant device.
This also applies to possible options/values. For information in permissible options/values of an object, please refert to the user manual.
The descriptions/permissible values are only defined if they might be required during runtime.

5.1 Data types

Index of type (dec)	Symbol description	Number of bytes
0	Undefined	N
1	Boolean*	1
2	INT8	1
3	INT16	2
4	INT32	4
5	UINT8	1
6	UINT16	2
7	UINT32	4
8	Floating point	4
9	Visible string (always terminated “0x00”)**	1,2,3 …
10	Octet string	1,2,3 …
11	Date	7
12	Time of day	6
13	Time difference	6
14	Bit string ***	1,2,3 …

* True = 0xFF, False = 0x00
** Visible strings are terminated 0x00 for simpler processing. When representing the length of an object, this is taken into account by using notation “N+1”. “N” is the number of bytes of the actually visible characters, and “+1” is the required 0x00 termination. “57+1, maximum” means that the object can have a maximum of 57 bytes of visible characters and the final 0x00, resulting in a total length of 58 bytes including the final 0x00.
*** Bit strings always have a length of n x 8 bits, where n∈N (n element of the natural numbers).

5.2 Data objects

Overview of the data objects used.

Object code	Object
0x00	Undefined
0x02	Domain variable
0x07	Simple variable
0x08	Array variable
0x09	Record variable
0x0a	Variable list Static and dynamic
0x0B	String variable

5.2.1 “Domain variable” object

Domain variables are objects with data lengths that can be larger than the maximum PDU size of the simple standard services. The “domain variable” object is used to transmit user-specific structured data of moderately large to very large size (bigger than (maximum PDU size) bytes). The object description must be recognized implicitly by the application program.

☞ The domain variable object can usually only be addressed fully using the Download Write and Upload Read macro services. In the 0x000E “CommProfile” object, the “upload/download protocol” feature is characterized by the values (634 or 635).

The total size of the domain variable must not exceed the maximum user data volume of [PDU size – 8) x 0xFFF0] (3.5 Mbytes, approximately).

In principle, any object index can be a domain variable. In order to make a simple distinction, the domain variables are preferably assigned a specific are, see “Index areas of the communication objects”. In this way, the master and slave implicitly know that they can usually be accessed using the macro services.

Depending on the system, there may also be separate standard services for the implementation of macro services. In the 0x000E “CommProfile” object, the “up/download protocol” feature is characterized by the values (636 or 637).
In this case, the domain variables can optionally be accessed using the simple standard services then, where required, partially via subindices, as long as the maximum PDU size of this service is not exceeded.

5.2.1.1 Domain variable - formal description

Object:	Domain variable
Object code:	0x02
Key attribute (M):	Index
Attribute (M):	Length (maximum PDU size – 8) x 0xFFF0 bytes)
Attribute (O):	Password

Services:

- (O) Read
- (O) Write
- (O) Download Write
- (O) Upload Read

5.2.2 “Simple variable” object

The “simple variable” object represents a single, simple variable. The object description must be recognized implicitly by the application program. The size of a simple variable must not exceed the maximum amount of user data of (PDU size - 6) bytes.

5.2.2.1 Simple variable - formal description

Object:	Simple variable
Object code:	0x07
Key attribute (M):	Index
Attribute (M):	Length (maximum (PDU size - 6) bytes)
Attribute (O):	Password

Services:
- (O) Read
- (O) Write
- (O) Information Report

5.2.3 “Array” object

The “array” object consists of a string of simple variables of the same data type. The object description must be recognized implicitly by the application program.
If an element of the object is to be addressed, a subindex must also be specified in the service next to the index. Subindex 1 accesses the first element of the object. Subindex 0 addresses the object as a whole. If errors due to an individual subindex occur while accessing the entire object, the corresponding error type is created and all the data is considered invalid.

There is no obligation to be able to address each element of an array individually.

The sum of user data of the individual array elements must not exceed the maximum permissible amount of user data of (PDU size - 6) bytes.
In this case, it is a logical obligation to be able to address each element of an array individually.

5.2.3.1 Array variable - formal description

Object:	Array variable
Object code:	0x08
Key attribute (M):	Index
Attribute (M):	Length per element (maximum (PDU size - 6) bytes)
Attribute (M):	Number of elements
Attribute (O):	Password

Services:
- (O) Read
- (O) Write
- (O) Download Write
- (O) Upload Read
- (O) Information Report

5.2.4 “Record” object

The “record” object consists of various elements. These are a string of simple variables of different data types and string variables. The object description must be recognized implicitly by the application program. The “record” object can be addressed as a whole or element by element. If the object is to be addressed as a whole, the associated index is specified in the service as subindex “0”.
If errors due to an individual subindex occur while accessing the entire object, the corresponding error type is created and all the data is considered invalid.
If an entire object is addressed via subindex “0”, consistency is automatically ensured, which would otherwise require considerable effort. It therefore does not make sense to transmit the 0x0018 “DiagState” object in its individual elements, for example. The effort required to ensure consistency of the individual elements to one another would be too high.

If an element of the object is to be addressed, a subindex must also be specified in the service next to the index. Subindex 1 accesses the first element of the object.

There is no obligation to be able to address each element of a record individually.

The sum of the user data of the individual record elements must not exceed the maximum permissible amount of user data of (PDU size - 6) bytes.
In this case, it is a logical obligation to be able to address each element of a record individually unless it is a domain variable.

5.2.4.1 Record variable - formal description

Object:	Record variable
Object code:	0x09
Key attribute (M):	Index
Attribute (M):	Length per element (maximum (PDU size - 6) bytes)
Attribute (M):	List of elements
Attribute (O):	Password

Services:

- (O) Read
- (O) Write
- (O) Download Write
- (O) Upload Read
- (O) Information Report

5.2.5 "Variable list" object

A variable list is a special form of a ⇨domain variable that has been assigned an own index area. It contains a summary of individual variables. The structure must be recognized implicitly by the application program here too.

Compiling several individual variables into a variable list is only useful for variables with small data types (e.g., UINT8, Interger32) In theory, the length of an individual variable in a variable list can be a maximum of (PDU size – 8) bytes. However, a variable of this type can be accessed more effectively using a normal read service.

5.2.5.1 Variable list - formal description

Object:	Variable list
Object code:	0x0A
Key attribute (M):	Index
Attribute (M):	Number of elements
Attribute (M):	List of element index
→ Attribute (M):	→ Index
Attribute (O):	Password

Services:

- (O) Read
- (O) Write
- (O) Download Write
- (O) Upload Read

5.2.5.2 Static variable list

Static variable lists can be created by the device manufacturer. These variable lists are found in the "static variable lists" index area (see index areas of the communication objects).

The structure of the static variable list can be read using the "VariableListRecord" variable (index E7FF). To do this, the index of the variable list to be read is written to object E7FF. Download Write is to be used for this, as object E7FF is a domain variable. The "VariableListRecord" can then be read using Upload Read access to object E7FF.

The "VariableListRecord" is transmitted according to the following structure:

Index (2 bytes)	Meaning
First value	Index of the first variable in the list
Second value	Index of the second variable in the list
...	
...	
Nth value	Index of the Nth variable in the list

5.2.5.3 Dynamic variable list

Dynamic variable lists are created using the “VariableListRecord” variable (index E7FF). These variable lists are found in the “dynamic variable lists” index area (see index areas for communication objects). The “VariableListRecord” is created using Download Write to object E7FF.

The “VariableListRecord” is transmitted according to the following structure:

Index (2 bytes)	Meaning
First value	Index under which the VariableListRecord should be created
Second value	Index of the first variable in the list
Third value	Index of the second variable in the list
...	
...	
N+1 value	Index of the Nth variable in the list

5.2.5.4 Variable lists transmission format

To write/read a variable list, all the values of the individual variables are entered in the data field of the Download Write/Upload Read one after the other.

	Meaning
First value	Contents of the first variable in the list
Second value	Contents of the second variable in the list
...	
...	
Nth value	Contents of the Nth variable in the list

5.2.6 “String variable” object

The “string variable” object represents a single, basic variable that is characterized by a specific data type (octet string, visible string or bit string).
The object description of the string variable object is defined static. A string variable is mapped to a real string variable that actually exists in the user system by means of the object description for the “string variable” object.
In the description of the “string variable” object, only one data type, either octet string, visible string or bit string, is permitted. A string variable has the same structure as a simple variable. They only differ in terms of variable length of the data types. The maximum length is configured in the memory. However, only the current length is transmitted.

5.2.6.1 String variable - formal description

Object:	String variable
Object code:	0x0B
Key attribute (M):	Index
Attribute (M):	Maximum length (maximum (PDU size - 6) bytes)
Attribute (O):	Password

Services:

- (O) Read
- (O) Write
- (O) Download Write
- (O) Upload Read
- (O) Information Report

6 Services

All basic variables, arrays, and records (amount of data < PDU size) are accessed using simple/standard services. These are:

- Read
- Write
- Information Report/Fetch

As these services are functions of parameter channel implementation, they are not described here. For additional information, please refer to the relevant parameter channel documentation*).

*) E.g., IBS PCP RE HB E (5052b.pdf) and IBS PCP PR HB E (5054b.pdf)

All domain variables, lists, arrays, and records (amount of data > PDU size) are accessed using the macro services: These are:

- Upload Read
- Download Write

If an object is accessed using the “Read” standard service and contains more data than allowed by the PDU size, access is created using error type: error class: 05, error code: 02, AddCode 0x0018 (service PDU size - object length is not compatible for this object).

If an object is accessed using the “Read” standard service and might contain more data than allowed by the PDU size, but currently does not, error-free access is possible.

If an object is accessed using the “Write” standard service and can take more data than allowed by the PDU size, error-free access is possible.

If an object is accessed using the “Download Write” macro service and can take less data than written with a PDU, access created using error type: error class: 05, error code: 02, AddCode 0x0018 (service PDU size - object length is not compatible for this object).

If an object is accessed using the “Upload Read” macro service and can supply less data than allowed by the PDU size, error-free access is possible.

Domain objects and variable lists that are explicitly defined by an object index in the assigned area can always be accessed using macro services.

6.1 Addressing

The objects of a device that are accessed using the services are addressed via the index and subindex.
The following specifications must be observed:

6.1.1 Addressing via subindex equal to “0”

- Access via subindex “0” is **mandatory** and is therefore always available.
- Subindex “0” addresses the entire index object, including all subobjects.
  In principle, it therefore is possible to access all the objects of all subindices using subindex “0”. The structure (number, empty subindices, data lengths, etc.) of the subobjects must therefore be recognized implicitly. For some implementations, this is the only possible way to access subobjects.
- As the sum of data of all the subindices of an index can be greater than the maximum PDU size, this can also mean that it might not possible to access all subobjects together using subindex “0”.
- In the case of reading, the error is reported with error type “PDU size problem” (error class = 0x05, error code = 0x02).
  For this reason, objects should be avoided where the sum of the data lengths of the subobjects is greater than the PDU size.
- In the case of writing, errors can also be reported with the error type “PDU size problem” (error class = 0x05, error code = 0x02).
  The slave can, however, also accept any smaller data length if this is appropriate in the application.
- Read/write access may differ for the subindices of an index. As long as “read only” objects are written with the same (already existing) values during common access, this is not considered an error and is consequently not answered with an error type.

6.1.2 Addressing via subindex unequal to “0”

- Access via the subindex unequal to “0” is optional even if the subobject itself is mandatory.
- Availability of this feature is therefore depending on the functions provided by the slave.
- Access via a subindex is always mandatory if the sum of data of all subindices of an index is greater than the PDU size.
- In principle each subobject addressed using a subindex is an independent object with its own properties and meanings.
- Each subobject addressed using a subindex can use the entire user data length available (maximum PDU size).

6.2 Data length

The amount of data to be transmitted varies – from “0” to “maximum PDU size”. It is specified with each service. The master and slave inform the relevant partner about the amount using the service.

The slave may accept any smaller data length if this is appropriate in the application.
This even is essential to ensure compatibility with previous devices on which objects have been extended.

Explanation example:
Following a revision, an object now has a length of 11 bytes divided into 5 subobjects. Before, it had a length of only 2 bytes and did not have any subindices. This means the old content can now be addressed via subindex “1” (2 bytes).
Writing 2 bytes to subindex “0” now is permissible, and subindex “1” will be written correctly. When reading via subindex “0”, the master (the application) has to ignore the third byte and all subsequent bytes for reasons of compatibility.

In order not to limit transmission efficiency, particularly in the case of strings, only the number of bytes that are actually required/defined will be transmitted.

6.3 Busy

There may be an exception in the event of a slave processing services. For example, the user data may not yet be available in the slave application due to pending operations.

If a service from the slave application cannot be terminated in the time normally expected (service timeout), this is reported with the “busy” error type (error class = 0x08, error code = 0x0B).
The master is notified that the slave application is active and the service is still being processed but is not yet complete.

The estimated time still required for the results to become available is given in ms in the additional code.
“0xFFFF” means that the estimated time for completion is unknown.

This error type is to be dispatched by the slave before the standard timeout period.

If the service is repeated with the same content within the estimated time given in the additional code + the standard timeout period, the following takes place before the standard timeout elapses:

- Either a positive response (this terminates the service)
- A different negative response (this also terminates the service)
- A negative response with this error type again In this case, the procedure described above is started again.

The valid result is actively reported by the slave to the master via the report/fetch mechanism following the time specified in the additional code + the standard timeout period at the latest.

The standard timeout period is defined in object 0x0021 “PChTimeout”.

6.4 Macro services

Objects with a data length greater than the maximum PDU size of the basic standard service and domain variable objects and variable list objects are accessed using the Download Write and Upload Read macro services.
If the Download Write and Upload Read macro services are implemented, the “Up/Download protocol” feature (634 or 635) must be entered in the 0x000E “CommProfile” object.

Depending on the system, there may also be separate standard services for the implementation of macro services.

In principle, any object can be addressed using the standard services (as long as the data length is smaller than the maximum PDU size) or using the macro services. To differentiate the accesses, there must be separate standard and macro services. This is specified in object 0x000E “CommProfile” object by means of the “up/download protocol” feature (636 or 637).

Handling of the macro services is described in the following:

6.4.1 Download function using write service (Download Write)

Unlike standard parameter channel implementation (which contains download services), large volumes of data (e.g., application program, firmware update, variable lists, etc.) are transmitted using the write service. To do this, a download transmission protocol is specified in the user data of the write service that allows for segmented transmission of bigger data volumes. Downloading can only be executed on domain variables and variable lists where there is no other way to clearly identify the service.

Structure of the Download Write PDU

Parameter	Req/Ind	Rsp/Cnf
Communication reference	M	M
Module number	M	M
Index	M	
Subindex	O	
Data field	M	
Segment number	M	
Data	M	
Result(+)		S
Result(-)		S
Error type		M

Communication reference/slot or device number

The communication reference addresses the desired communication partner.
(“Compact” implemented devices can only communicate with the master (not peer-to-peer), which means that the communication reference is always 2.)

Module number

The “module number” parameter is used to address submodules of a modular device.
For more detailed information, please refer to the “Modular devices” section.

Index

The index parameter addresses the object to be written.

Subindex

The value of the subindex parameter for the download function is “0”.

Data field

The data field parameter contains the download protocol.

Data field	
Segment number 2-byte download control	User data 1 … (PDU size – 8) byte download data blocks E.g., 1 ... 56 bytes for “Compact” implementation

Segment number

This parameter contains the number of the data block to be written, for which the following definition applies:

Segment number = 1	Initiation of a download sequence with the first data block.
Segment number = 2 ... 0xFFF0	Data block number N
Segment number = 0xFFFD	The last data block only contains • The number of transmitted data blocks (without this end data block) (2 bytes) and • The CRC16 residual polynomial[1)] (2 bytes)
Segment number = 0xFFFE	The last data block only contains • The number of transmitted data blocks (without this end data block) (2 bytes) and • The CRC32 residual polynomial[1)] (4 bytes)
Segment number = 0xFFFF	The last data block only contains • The number of transmitted data blocks (without this end data block) (2 bytes) and no CRC residual

[1)] The CRC is always generated using all the pure user data. The segment numbers are not part of the user data.
The following generator polynomials are used:
16-bit CRC: CRC CCITT:
$x^{16} + x^{12} + x^5 + 1$
Initial value: 0xFFFF
Start with least significant bit.
The result of the CRC calculation (residual polynomial) is accepted immediately.

32-bit CRC: CRC-32 (IEEE 802.3):
$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$
Initial value: 0xFFFF.FFFF
Start with least significant bit.
The result of the CRC calculation (residual polynomial) is inverted at the end.

The download server must check that the segment number sequence is in ascending order without any gaps and, in the event of an error, send a negative response to the client (usually the fieldbus master). In the case of a negative response, the client can transmit the correct segment or terminate transmission with an end segment.
Only an end data block can terminate the download service.

Data

This parameter contains the data blocks. The length of these data blocks depends on the maximum PDU size. For a 64-byte PDU, the maximum data block length is 56 bytes. The maximum data block length does not have to be used.

Result(+)

The “Result(+)” parameter indicates a positive result. The segment number may only be incremented if the result is positive.

Result(-)

The “Result(-)” parameter indicates a negative result.

Error type

The “Error type” parameter contains the error cause.

The content of the “Error type” parameter corresponds to the write service, with the following definition for the specific download sequence error:

Error class “8” - Other
Error code “1” – Profile-specific

Additional code (hex)	Meaning
0x00A0 Invalid segment number - segment missing	Invalid segment number (segment missing)
0x00A1 Resource unavailable	No more resources (memory) are available for downloading
0x00A2 Invalid CRC	Incorrect CRC
0x00A3 File open error	Error opening the file (if file system is available)
0x00A4 File write error	Error writing the file (if file system is available)
0x00A5 File close error	Error closing the file (if file system is available)
0x00A6 Segment missing	Fewer data blocks were received than specified in the last segment
0x00A7 Segment overrun	More data blocks were received than specified in the last segment
0x00A9 Invalid segment number – double segment	Segment number invalid (segment duplicated, segment ignored)

6.4.2 Upload function using read service (Upload Read)

Unlike standard parameter channel implementation (which contains upload services), large volumes of data (e.g., application program, backups, variable lists, etc.) are read using the read service. An upload protocol is defined in the subindex of the read service, which enables segmented transmission of larger volumes of data. If the service cannot otherwise be clearly marked, upload can be carried out to domain variables and variable lists.

Structure of the Upload Read PDU

Parameter	Req/Ind	Rsp/Cnf
Communication reference	M	M
Module number	M	M
Index	M	
Subindex	M	
Result(+)		S
Data field		M
Segment number		M
Data		M
Result(-)		S
Error type		M

Communication reference
Like download function using write service (Download Write)

Module number
Like download function using write service (Download Write)

Index
Like download function using write service (Download Write)

Subindex

An upload sequence is initiated using a special subindex, whereby the following definition applies:

Subindex = 0xFF Initiation of an upload sequence with request for the first data block
Subindex = 0x00 Request for the next data block
Subindex = 0x01 Request for the relevant previous data block (e.g., after an error)

Result(+)

The parameter indicates a positive result. The segment number may only be incremented if the result is positive.

Data field

The data field parameter contains the upload protocol.

Data field	
Segment number 2-byte upload control	User data 1 ... (PDU size – 8) byte download data blocks 1 ... 56 bytes for “Compact” implementation

Segment number

This parameter contains the number of the data block to be read, for which the following definition applies:

Segment number = 1	Initiation of an upload sequence with the first data block
Segment number = 2 ... 0xFFF0	Data block number N
Segment number = 0xFFFD	The last data block only contains • The number of transmitted data blocks (without this end data block) (2 bytes) and • The CRC16 residual polynomial[1)] (2 bytes)
Segment number = 0xFFFE	The last data block only contains • The number of transmitted data blocks (without this end data block) (2 bytes) and • The CRC32 residual polynomial[1)] (4 bytes)
Segment number = 0xFFFF	The last data block only contains • The number of transmitted data blocks (without this end data block) (2 bytes) and no CRC residual

[1)] The CRC is always generated using all the pure user data. The segment numbers are not part of the user data.
The following generator polynomials are used:
CRC-16: CRC-CCITT: $x^{16} + x^{12} + x^5 + 1$
CRC-32: IEEE 802.3: $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

The upload client must check the segment number sequence is in ascending order without gaps. In the event of an error, the client can request the last segment again with subindex 0xFE or restart the upload with subindex 0xFF.

Data

This parameter contains the data blocks. The length of these data blocks depends on the maximum PDU size. For a 64-byte PDU, the maximum data block length is 56 bytes. The maximum data block length does not have to be used.

Result(-)

This parameter indicates a negative result.

Error type

The “Error type” parameter contains the error cause.

The content of the “Error type” parameter corresponds to the read service, with the following definition for the specific upload sequence errors:

Error class “8” - Other
Error code “1” - Profile-specific

Additional code (hex)		Meaning
0x00A0	Invalid segment number - segment missing	Upload without initiation with subindex = 0xFF
0x00A3	File open error	Error opening the file (if file system is available)
0x00A5	File close error	Error closing the file (if file system is available)
0x00A8	File read error	Error reading the file (if file system is available).

6.4.3 Variable list download example

In this example, a “Compact” device uses the following objects, which are grouped into a variable list.

Index (hex)	Subindex	Data type	Length	Example value (hex)
0080	0	UINT8	1	FF
0081	7	UINT16	2	12 34
0090	3	UINT8	1	00
0091	0	Octet string	16	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
0100	0	Octet string	16	10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
0101	0	Octet string	16	20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F
0105	0	Octet string	16	30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F

The grouped variable list in this example is stored under index 0xE000.

Download Write

First Download Write request/indication

Parameter	Content (hex)
Com. ref.	02
Module number	00
Index	E0 00
Subindex	00
Segment no.	00 01
Data	FF 12 34 00 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F

Second Download Write request/indication

Parameter	Content (hex)
Com. ref.	02
Module number	00
Index	E0 00
Subindex	00
Segment no.	00 02
Data	30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F

Third Download Write request/indication

Parameter	Content (hex)
Com. ref.	02
Module number	00
Index	E0 00
Subindex	00
Segment no.	FF FF
Data	00 02

6.5 List of permissible communication error types

If an error occurs when writing/reading the object, e.g., transmission of invalid values, access is to be rejected using the appropriate error types (see below).

Communication error types only exist as a negative confirmation of a parameter channel service (standard services).

Error messages that occur during communication via the parameter channel:
The error type service parameter consists of the following parameters:

• Error class	1 byte, UINT8, hexadecimal notation
• Error code	1 byte, UINT8, hexadecimal notation
• Additional code	2 bytes, UINT16, hexadecimal notation

6.5.1.1 Error class and error code

Error class	Error code	Meaning
2 Application reference		This error class refers to the communication relationship which is used to process the service.
	1 Application unreachable	`0x0201` Access to the object is not possible because the application is not implemented/present. This error type is typical where an attempt is made to access a non-existing module of a modular device using the module number. Module not present or module number incorrect.
	0 Other	`0x0200` Errors of this class cannot be assigned to any of the previously given error types.
5 Service		This error class is reported in the event of a faulty service. This affects all parameters with the exception of the actual object.
	1 Object state conflict	`0x0501` The current object state prevents the service from being executed.
	2 Service PDU size	`0x0502` PDU size problem. If more data is to be transmitted than the agreed maximum PDU size allows, for example.
	3 Object constrain conflict	`0x0503` It is temporarily not possible to execute the service.
	4 Parameter inconsistent	`0x0504` The service contains inconsistent parameters.
	5 Illegal parameter	`0x0505` A parameter has an invalid value.
	0 Other	`0x0500` Errors of this class cannot be assigned to any of the previously given error types.
6 Access		This error class is reported in the case of faulty access. This only affects the actual object.
	1 Object invalidated	`0x0601` Access relates to a defined object that has an undefined reference attribute. This represents a permanent error for access to this object.
	2 Hardware fault	`0x0602` Access to the object failed due to a hardware fault. More detailed information about the cause can be found in the additional code (glob. = 8 and 9)
	3 Object access denied	`0x0603` The client's access rights are insufficient.

	4 Invalid address	0x0604 Access to an invalid internal address.
	5 Object attribute inconsistent	0x0605 The attribute has an invalid value. More detailed information about the cause can be found in the additional code (glob. = 1) If more or less user data bytes are to be transmitted than is permissible for a specific object, for example.
	6 Object access unsupported	0x0606 The object is not a variable access object.
	7 Object non-existent	0x0607 There is no existing object under this index/subindex.
	8 Type conflict	0x0608 The data does not correspond to the data type of the object.
	A Data not yet available	0x060A The object data can currently not be accessed (e.g., during re-parameterization).
	0 Other	0x0600 Errors of this class cannot be assigned to any of the previously given error types.
8 Other (application)	0 Other	0x0800 The service has not been executed. The reason is specific to the application/manufacturer and only affects the actual data item. More detailed information about the cause can be found in the additional code. A specific object value cannot be permissible in this specific application, for example.
	1 Device profile-specific	0x0801 The service has not been executed. The reason is specific to the device profile (0x000F). More detailed information about the cause can be found in the additional code. All the additional codes listed in this profile are profile-specific.
	B Busy	0x080B The service cannot be terminated in the time normally expected by the slave application. The estimated time still required for the data to become available is given in ms in the additional code. “0xFFFF” means that the estimated time for completion is unknown.

6.5.2 Additional code

The additional code is made up of global, specific and device-specific components. Provision of global and specific codes is optional. This means: additional code = 0x0000 is output if it is not possible to provide information on the cause of the error by the application.
The specific code contains a more detailed description of the error cause defined in the global code. If the error cause does not correspond logically to a specific code, specific code = 0 must be specified.

2 octets, normally with the following structure:

Bit 15 8	7 4	3 0
Device-specific	Global code	Specific code

The values for bits 8 to 15 (device-specific code) are reserved and currently filled with the value 0.

All the additional codes listed in this profile are profile-specific.

Additional code [hex]	Meaning
0000	No detailed information on the cause of error
0010	Service parameter with invalid value
0011	Subindex not available
0012	Object access is not a request
0013	Reserved service code
0014	Subslot not supported
0015	Object access type not supported on this object
0016	Object access request index for this AccessType must equal 0x0000
0017	Object access request length for this AccessType must equal 0
0018	Object length is not suitable for this object (see also: additional code 001D and 001E
0019	Object is ReadOnly and may not be overwritten.Error class = 0x06; error code = 0x03
001 A	Object is WriteOnly and cannot be read Error class = 0x06; error code = 0x03
001B	Write/read service for this object is not supported Error class = 0x05; error code = 0x05
001C	Upload Read or Download Write is required for access to the object due to the object length. Error class = 0x05; error code = 0x02
001D	Object length is not suitable for this object (0018) – too much data is transmitted Error class = 0x06; error code = 0x05
001E	Object length is not suitable for this object (0018) – Not enough data is transmitted Error class = 0x06; error code = 0x05
0020	Service cannot be executed at present
0021	Service cannot be executed at present as the device is currently being controlled locally.
0022	Service cannot be executed in current device state (device control).
0023	Service cannot be executed at present as no object dictionary is available.
0024	Index not available
0030	Value range of a parameter out of range (the server cannot provide the value) Error class = 0x08; error code = 0x01
0031	Parameter value too large Error class = 0x08; error code = 0x01
0032	Parameter value too small Error class = 0x08; error code = 0x01
0040	Dependency ignored Collision with other values, dependency ignored
0041	Communication object cannot be mapped to the process data
0042	Process data length exceeded
0050	FW update: error class = 0x08; error code = 0x01 Firmware incorrect for the device – Firmware could not be processed by the device (general, no detailed information). See also “Firmware update” section

0051	FW update: Upload Read or Download Write required for access to the object due to the object length. FW header or update version incorrect
0052	FW update: error class = 0x08; error code = 0x01 FW below minimum FW version – Firmware incorrect for device (e.g., hardware too old)
0053	FW update: error class = 0x08; error code = 0x01 Indicates the option to bypass the download of a FW update block to the device
0080	Hardware fault
0081	Application failed
0082	Hardware is temporarily faulty
00A0	Invalid segment number e.g., upload without initiation with subindex == 0xFF
00A1	Resource not available No more resources (memory) are available for downloading
00A2	Wrong CRC
00A3	Error opening the file (if file system is available)
00A4	Error writing the file (if file system is available)
00A5	Error closing the file (if file system is available)
00A6	Segment missing Fewer data blocks were received than specified in the last segment
00A7	One segment too much More data blocks were received than specified in the last segment
00A8	Error reading the file (if file system is available).
00A9	Invalid segment number (segment duplicated, segment ignored)
00B1	The password cannot be replaced (deleted).
00B2	The password cannot be added (too many passwords).
00B3	The password cannot be assigned for the desired type of access.

Codes that are not listed are reserved.

If a manufacturer cannot assign an error message to any of the above codes, the INTERBUS Club should be contacted.

7 Standard objects

The following objects are described in the form:

Index (hex)	Unique object identifier The service uses this index to access the object.
.Subindex (hex)	Additional unique identifier for the subobject The services uses this index to access the subobject.*)
Object name	Name of the object
R/W	Restricted access, see below
Length	Length of the object in bytes
Data type (representation)	Data type of the object, see above Form in which the value is represented.
Meaning	Explanation of the object contents
M/D/O	Implementation instruction M = Mandatory This object must be implemented. O = Optional This object must be implemented exactly in this way if the device supports the corresponding (optional) function. D = Dependent Whether the object must be implemented depends on whether another associated object was implemented or another specified condition is fulfilled. Subobjects must generally be implemented together with the object. The sufficient condition in each case is defined under "**D**ependency info".

*) Separate access via a subindex larger than "0" is optional, even if the subobject itself is mandatory.

Access restrictions

R	Read only, this object can only be read
W	Write only, this object can only be written
R/W	Read/write, this object can be read and written
R/W_D	Read/write, this object can be read and can also be written under certain conditions
UR	Upload Read, this object can be read using the "Upload Read" service The "Read" standard service can only be successfully executed if the amount of data is < PDU size. Otherwise, error type: error class: 05, error code: 02, AddCode 0x0018 (service PDU size - Object length for this object incorrect) is generated*)
DW	Download Write, this object can be written using the "Download Write" service. The "Write" standard service can always be executed because the amount of data is < PDU size.*)

*) See also Section "Domain-variable object"

To ensure standard access using tools, the objects, if they exist, must generally be implemented exactly in the form defined in the following. However, the device manufacturer

can adapt the contents to meet requirements. For example, a hardware status that cannot be determined by the device firmware for technical reasons could be represented as follows:

000B	HardwareVersion	R	51 bytes, maximum	Record (2 elements)	Hardware version (device or communication module)	**M**
.1	• BuildDate	R	10+1 bytes	Visible string	0000-00-00	**M**
.2	• Version	R	39+1 bytes, maximum	Visible string	Not available	**M**

If only some of the bits in a bit string or only some of the bits or bytes in an octet string have a function, the reserved bits and bytes are defined “0”.
If values not equal to “0” are written, an error type must be generated in order to recognize incompatibility.

7.1 Identification

The information specified in the following objects describes the device itself, the manufacturer and the field of application of the device. On dispatch, the data entered must match up with what is printed on the device.

Index (hex)	**Object name**	**R/ W**	**Length**	**Data type** (representation)	**Meaning**	**M/O**
0001	VendorName	R	57+1 bytes, maximum	Visible string (text)	Manufacturer name e.g., Phoenix Contact GmbH & Co KG	**M**
0002	VendorID	R	6+1 bytes	Visible string (text)	Manufacturer ID Unique manufacturer identifier according to Organizationally Unique Identifiers (OUI): http://standards.ieee.org/regauth/oui/index.shtml e.g., “00A045”	O
0003	VendorText	R	57+1, maximum bytes	Visible string (text)	Manufacturer text Comments on the manufacturer e.g., address, industry, etc.	O
0012	VendorURL	R	57+1, maximum bytes	Visible string (text)	Manufacturer URL URL of the manufacturer's homepage, product page, promotion page, etc. e.g., www.phoenixcontact.com	O
0007	ProductName	R	57+1 bytes, maximum	Visible string (text)	Product name Manufacturer-specific unique product designation e.g., IB IL 24 DO 8	**M**
0008	SerialNo (ProductID)	R	57+1 bytes, maximum	Visible string (text)	Serial number Manufacturer-specific unique device code, e.g., 123456789	O
0009	ProductText	R	57+1 bytes, maximum	Visible string (text)	Product text Manufacturer-specific product text e.g., digital output module	O
000A	OrderNumber	R	57+1 bytes, maximum	Visible string (text)	Order No. Manufacturer-specific unique product type ID, e g., order number e.g., 2726269	**M**

0037	DeviceType	R	8 byte	Octet string[8] (hex)	Device type Manufacturer-specific device/module identification not equal to “0” The manufacturer-specific device/module identification, together with the VendorID, enables exchange and operation of similar devices within a configuration. For example, a 16-channel output module with screw connection technology can be replaced by a module with spring-cage connection technology even though it does not have the same order number. Different functions also require a different DeviceType. If this object is used in a modular device, it can also identify a dummy. “0” if the module is not (yet) installed. Nevertheless, data can be mapped to the process data in this case.	**M**
0032	FieldBus_ID	R	3 bytes	Record (2 elements)	Fieldbus identification	O
.01	• ID code	R	1 byte	UINT8 (dec)	ID code Fieldbus specific ID code (8 bits, usually represented as a decimal number)	O
.02	• PDLength	R	2 bytes	UINT16 (dec)	Process data length Number of process data bits (unit: bit)	O
000B	HardwareVersion	R	51 bytes, max.	Record (2 elements)	Hardware version (device or communication module) *On dispatch, the data entered must match up with what is printed on the device.*	**M**
.01	• BuildDate	R	10+1 bytes	Visible string (text)	Manufacturing date Version date Format YYYY-MM-DD according to ISO 8601 e.g., 2002-11-29	**M**
.02	• Version	R	39+1 bytes, max.	Visible string (text)	Version ID e.g., 4.01 Beta customer	**M**
000C	FirmwareVersion	R	51 bytes, maximum	Record (2 elements)	Firmware version Components of a device or communication module that can be modified from externally (also updated, if required), e.g., firmware controller A, firmware controller B, bootloader, FPGA image, etc. There is only one user-relevant entry that is determined by the manufacturer for all these components. Manufacturer-specific objects that specify the individual components more precisely are permissible. *On dispatch, the data entered must match up with what is printed on the device.* *See also “Definition of terms”*	**M**
.01	• BuildDate	R	10+1 bytes	Visible string (text)	Manufacturing date Version date Format YYYY-MM-DD according to ISO 8601 e.g., 2002-05-03 If no FW is available, “0000-00-00” is entered	**M**
.02	• Version	R	39+1 bytes, max.	Visible string (text)	Version ID e.g., 1.03 customer If no FW is available, “-” is entered	**M**

000D	PChVersion	R	51 bytes, maximum	Record (2 elements)	Parameter channel version Parameter channel version Parameter channel implementation version	**M**
.01	• BuildDate	R	10+1 bytes	Visible string (text)	Manufacturing date Version date Format YYYY-MM-DD according to ISO 8601 e.g., 2016-12-01 If no parameter channel is available, “0000-00-00” is entered	**M**
.02	• Version	R	39+1 bytes, maximum	Visible string (text)	Version ID e.g., “PCP Compact V1.10” If no parameter channel is available, “--” is entered	**M**
0005	Capabilities	R	N x 8 bytes	Array of visible strings[8] (text)	Device properties Properties/functions of the device in addition to the basic functions. All the properties/functions listed in the following are defined in this basic profile.	**M**
		• “Nothing” No additional functions. The entry can also occur several times as a dummy. • “Safety0” The slave supports secure data communication. This takes place in both directions. • “Energy0” The slave supports energy management. • “ChPDWh0” The slave supports change of the process data width. • “SubMa_0” The slave is a subbus master. There is at least one additional subsystem below this slave. • “FwUpdt0” The slave supports the firmware update. The type is exactly 8 characters long. Unused characters are to be filled with 0x00. If the slave supports the “Capabilities” object but no other additional function, the “Nothing” entry is included at least once.				
0006	ProductFamily	R	57+1 bytes, maximum	Visible string (text)	Product range Manufacturer-specific product range e.g., Inline	O
0004	DeviceFamily	R	57+1 bytes, maximum	Visible string (text)	Device range (e.g., I/O, drive) according to specifications, see below, e.g., motion control/frequency converters This object can contain several entries, which are separated by a comma followed by a space.	**M**
000E	CommProfile	R	4+1 bytes, maximum	Visible string (text)	Communication profile System-specific profile ID, see below e.g., 634	**M**

000F	DeviceProfile	R	4+1 bytes, maximum	Visible string (text)	Device profile Identification of the application profile according to which the device is specified, see below e.g., “10” for the basic profile or “22” for the DRIVECOM profile	**M**
0011	ProfileVersion	R	51 bytes, maximum	Record (2 elements)	Profile version Version designation of this profile	**M**
.01	• BuildDate	R	10+1 bytes	Visible string (text)	Manufacturing date “2018-04-19”	**M**
.02	• Version	R	39+1 bytes, maximum	Visible string (text)	Version ID Basic profile V3.0	**M**
0013	OnBoardDeviceDescFileName	R	57+1 bytes, maximum	Visible string (text)	Name of the onboard device description file File name of the internal (onboard) device description file, e.g., XYZ.xml If no onboard device description file exists, this object will also not exist.	O
003A	VersionCount	R	8 bytes	Array of UINT16 (hex) (4 elements)	Version counter Unique, ascending numbering as an integer for the version of the corresponding objects (components) with the following indices: • 0x0011 “ProfileVersion” • 0x000D “PChVersion” • 0x000B “HardwareVersion” • 0x000C “FirmwareVersion” If a component is changed, the value in the associated object (see above) must be adapted. In this case, the associated VersionCount subindex must be increased by at least 1. Gaps between two version counter readings are permissible.	**M**
.01	• ProfileVersion	R	2 bytes	UINT16 (hex)	Profile version 0x0009 for this profile.	**M**
.02	• PChVersion	R	2 bytes	UINT16 (hex)	Parameter channel version e.g., 0x0002 for “Compact” version implementation If a device does not have a parameter channel, 0x0000 is entered.	**M**
.03	• HardwareVersion	R	2 bytes	UINT16 (hex)	Hardware version e.g., 0x0002 for the hardware version	**M**
.04	• FirmwareVersion	R	2 bytes	UINT16 (hex)	Firmware version e.g., 0x0105 for the firmware version If a device does not have firmware, 0x0000 is entered.	**M**
0047	AddInfo	R	16 bytes	Record (2 elements)	Additional information Miscellaneous additional information is contained in the following subindices.	D

.01	• SafetyProtType	R	8 bytes	Octet string[8] (text)	Safety protocol type • “IBSSafe0” INTERBUS-Safety • “SBT_0” SafetyBridge protocol • “PROFI_0” PROFISafe protocol • “CIP_0” CIP safety protocol • “FSoE_0” FSoE safety protocol • “OpenS_0” Open safety protocol The type is exactly 8 characters long. Unused characters are to be filled with 0x00. **D**ependency info: This object must be implemented if the device supports a safety protocol.	D
.02	• SafetyProtVers	R	8 bytes	Octet string[8] (text)	Safety protocol version Exactly 8 characters are available for this version. Unused characters are to be filled with 0x00.	D
.03	• LegacyInfo	R / W	8 bytes	Octet string[8] (text)	Information about previous version The user can enter a manufacturer-specific code in order for the device to behave like a specific (previous) version of the device, if possible. Example: Value “1.50” can be entered for a device with firmware version 1.62. The device must then behave like a device with firmware version 1.50. The parameters are not automatically reset to the default values. If the specified identifier is not supported, a negative confirmation with error type 0x08, 0x01 and AddCode, e.g., 0x0032, is created. **D**ependency info: This object must be implemented if an exchange-compatible previous version exists for the device that does not have at least one object/subobject that is implemented in the current version.	D
0014	Location	R / W	Maximum 57+1 bytes	Visible string (text)	Installation location Text that was stored by the device user in this parameter. This text indicates the installation location of the device and is stored in a non-volatile memory. E.g., machine 1, back-left The content is normally written to the device only once, during startup (naming).	O

0015	EquipmentIdent	R / W	Maximum 57+1 bytes	Visible string (text)	Equipment ID Text that was stored in this parameter by the device user. The equipment ID is stored in a non-volatile memory. The device user can, for example, store a description about device operation here. E.g., M1H747h.l. The content is normally written to the device only once, during startup (naming).	O
0016	ApplDeviceAddr	R / W	2 bytes	UINT16 (dec)	Application-specific device address (user-defined device number) Any identification for this device in this specific application. This number does not have to be unique. Management is left entirely to the end user. E.g., 123 The content is normally written to the device only once, during startup (naming).	O

7.1.1 Device range (DeviceFamily index 0x0004)

A unique designation for the device range.

En	De
Actuator	Aktor
Bus Coupler	Buskoppler
Closed Loop Controller	Regler
Dosing Device	Dosiergerät
Drive	Antrieb
Drive - Frequency Inverter	Antrieb - Frequenzumrichter
Drive - Motor Starter	Antrieb - Motorschalter
Drive - Servo Amplifier	Antrieb - Servoverstärker
Drive - Stepper Motor Controller	Antrieb - Schrittmotor-Steuerungcontroller
Encoder	Encoder
Gateway	Gateway
General	Allgemeines
HMI	HMI
HMI Display	HMI-Anzeige
HMI Operator Panel	HMI-Bediengerät
Hydraulic Device	Hydraulik-Gerät
I/O	E/A
I/O analog IN / OUT	E/A analog IN / OUT
I/O analog IN	E/A analog IN
I/O analog OUT	E/A analog OUT
I/O digital IN / OUT	E/A digital IN / OUT
I/O digital IN	E/A digital IN
I/O digital OUT	E/A digital OUT
I/O Temperature Module	E/A- Temperaturmodul
I/O Function Module	E/A-Funktionsmodul
Identification System	Identifikationssystem
Media Converter active	Medienkonverter aktiv
Media Converter passive	Medienkonverter passiv
NC	NC
NC/RC	NC/RC
PC	PC
PC Board	PC-Karte
PLC	SPS
PLC board	SPS-Karte
Pneumatic Device	Pneumatik-Gerät
Positioning Controller	Positionier-Steuerung
Power Supply	Stromversorgung
Robot Control	Roboter
Safety	Funktionale Sicherheit
Sensor	Sensor
Switching Device	Schaltgerät
Technology Controller	Technologie-Steuerung
Valve	Ventil
Weighing or Batching System	Wiege- oder Dosiersystem
Welding Controller	Schweißsteuerung
Wrenching Controller	Schraubersteuerung
Safety-*)	Safety-*)

*) “Safety” can be used to precede every device range.

If a manufacturer cannot assign a device to the above scheme, the INTERBUS Club should be contacted.

7.1.2 Communication profile (CommProfile, index 0x000E)

This parameter contains a system-specific communication profile code.
The following communication profiles apply for “Compact” implementation:

63	Implementation	Channel type	Suitable for
632	Compact	Fieldbus management, compact parameter channel	No cyclic process data, downloading/uploading of variable lists, programs, etc.
633	Compact Process data	Fieldbus management, cyclical process data, compact parameter channel	Parameterizable devices, only parameters, no downloading/uploading of variable lists, programs, etc.
634	Compact Process data Up/download protocol	Fieldbus management, cyclical process data, compact parameter channel, download, upload	Complex devices, downloading/uploading of variable lists
635	Compact Up/download protocol	Fieldbus management, compact parameter channel, download/upload	Complex devices, downloading/uploading of variable lists, no cyclic process data
636	Compact Process data Explicit up/download standard services	Fieldbus management, cyclical process data, compact parameter channel, downloading/uploading as explicit standard services	Complex devices, downloading/uploading of variable lists
637	Compact Explicit up/download standard services	Fieldbus management, compact parameter channel, downloading/uploading as explicit standard services	Complex devices, downloading/uploading of variable lists, no cyclic process data

7.1.3 Device profile (DeviceProfile, index 0x000F)

This parameter contains information about the implemented device type profile of the device.
Structure of the parameter:

B15 .....b12	b11 .....b8	b7 .....b4	b3 .....b0
Profile group			Version

DeviceProfile	Meaning
0000	No profile
0010	Basic profile (basis for all other profiles)
0012	Sensor/actuator
0020	DRIVECOM process data only
0021	DRIVECOM frequency converter
0022	DRIVECOM servo
0030	Reserved
0040	Controller boards
0050	Reserved
0060	Reserved
0070	Encoder
0080	Process controller
0090	Robot controllers
00A0	Wrenching controllers
00B0	ISO valves
00C0	Welding controllers
00D0	Operating/display units
00E0	Hydraulics devices
FFFF	More than one device profile
All others	Reserved

If a device supports more than one device profile, 0xFFFF is entered into the DeviceProfile parameter.
Although shown here as a hex value, the parameter is transmitted as a visible string with a length of 4+1. The individual digits of the hex codes are transmitted as ASCII characters.

7.2 Device diagnostics

These objects are used to send diagnostic information about the status of the device and any connected I/O devices to the application. The current diagnostic information for the device is stored in the “DiagState” object. In addition, this information can be stored in long form in the domain variables (index 0xE800).
A history of the diagnostic information can be found in the “DiagHistory” domain variable or the “DiagHistoryLong” domain variable for the long form.

A negative service response reports an error while accessing an object (in particular due to the transmitted parameter content).
An additional report about object “DiagState” 0x0018 must not be provided.

7.2.1 Objects

Index (hex)	Object name	R/W	Length	Data type (representation)	Meaning	M/O
0018	DiagState	R	207 bytes, maximum	Record 6 elements	Diagnostic state Current diagnostic state of the device in short form The diagnostic change speed should not fall below one second.	**M**
.01	• Lfd.Nr.	R	2 bytes	UINT16 (dec)	Consecutive no. Unique, consecutive fault number since the last power up reset or history reset. The number is counted on for each fault, not separately for incoming and outgoing. The fault number is also counted up for faults that might not have been reported because of their priority.	**M**
.02	• Priority	R	1 byte	UINT8 (hex)	Priority Fault priority See “Classification of messages”	**M**
.03	• Channel	R	1 byte	UINT8 (dec)	Channel Channel on which the fault occurred. “0xFF” refers to the entire device. For additional information see “MoreFollows”	**M**
.04	• Code	R	2 bytes	UINT16 (hex)	Fault code, see below	**M**
.05	• MoreFollows	R	1 byte	Bit string 8 (bin)	Additional information Information for interpreting the following data	**M**
		= “0x00” No further information Bit0 = 1: There is additional information on this fault that can be read via the “DiagStateLong” object. Bit1 : Reserved (=0) Bit2 : Reserved (=0) Bit3 : Reserved (=0) Bit4 = 1: There are additional simultaneously occurring diagnostic events that can be read via the 0xE806 “ComplDiagState” object. Bit 5-6: Reserved (=0) Bit7 = 1: Identifier for this extended version of object 0x0018				
.06	• Reserved	R	2 bytes	Octet string[2] (hex)	Reserved This entry is reserved for 0x00.00	**M**

.07	• SubModNo	R	1 byte	UINT8 (dec)	Submodule number If the device is a modular device, the corresponding submodule is specified here. See “Modular devices - subsystems” section If the device is not a modular device, “0” is entered here.	**M**
.08	• FunctionGroup	R	8 bytes	Octet string[8] (text)	Function group Function groups that report diagnostics. The type is exactly 8 characters long. Unused characters are to be filled with 0x00. The permissible function groups are listed in objects 0x003B.1 and 0x003C.1. If there are several groups of one type (e.g., 4 channels each of a 16-channel DO module), the corresponding G number is appended. For example, a DI 32 is made up of four groups: DI-G1, DI-G4 The manufacturer-specific designation (“Relay OUT”) is specified in the diagnostic text (0x0018.11). Remark: Usually, “DIAG”, “CMD”, “STATUS”, etc., are no useful function groups.	**M**
.09	• AddValue	R	4 bytes	Octet string[4] (hex)	Additional information “Additional value” to the current diagnostic state of the device. Within certain limitations, the user has free use: In the case of subsystems, the subsystem-specific diagnostic codes are entered here. Where applicable, manufacturers are also able to enter any legacy codes for their diagnostics here. Alternatively, a value associated with diagnostics can be entered, e.g., “current temperature when exceeding the limit temperature”. Usage described in the PROFINET I&M profile is often followed. Legacy information: former object 0x0034	**M**
.0A	• TextLength	R	1 byte	UINT8 (dec)	Text length Length of the following diagnostic text in bytes	**M**

.0B	• Text	R	99+1 bytes, maximum	Visible string (text)	Diagnostic text The occurred fault should be explained in more device-specific detail here for the system operator. In addition to the fault type, the function groups affected and the channels should also be specified. Furthermore, the terminal point can be specified if there is a reference to this point. Finally, the user should always be given an option for action. *Example:* *“Overrange at AI channel 3 on terminal point 03, 13. Check sensor signal.“* The text is just an example and is not intended for generic interpretation. The string is terminated 0x00. Default: “Status OK”	**M**
E806	ComplDiagState	R		Array of Records	Complete current diagnostic state Device diagnostic information in short form for all messages still pending. If the object is addressed via subindices, these are to be assigned in ascending order without a break.	O
.01	• DiagState1				Oldest diagnostic state in short form - Object 0x0018	D
	…				…	D
.N	• DiagStateN				Newest diagnostic state in short form - Object 0x0018	D
E800	DiagStateLong	R		Domain variable Record	Diagnostic state (long form) Current diagnostic information of the device in long form	O
Seg.1 -a	• Lfd.Nr.				- Object 0x0018.1	D
-b	• Priority				- Object 0x0018.2	D
-c	• Channel				- Object 0x0018.3	D
-d	• Code				- Object 0x0018.4	D
-e	• MoreFollows				- Object 0x0018.5	D
-f	• Reserved				- Object 0x0018.6	D
-g	• SubModNo				- Object 0x0018.7	D
-h	• FunctionGroup				- Object 0x0018.8	D
-i	• AddValue				- Object 0x0018.9	D
-j	• Date	R	10+1 bytes	Visible string (text)	Date on which the fault occurred, in the format YYYY-MM-DD. “0000-00-00” = No date available	D
-k	• Time	R	8+1 bytes	Visible string (text)	Time at which the fault occurred, in the format hh:mm:ss. “00:00:00” = No time available	D

-l	• TimeofOperation	R	4 bytes	UINT32 (dec)	Absolute operating hours counter reading, at which the fault occurred. 0 ... 4294967295 seconds About 136 years = “0” no operating hours counter available	D
-m	• ParamSet	R	2 bytes	UINT16 (dec)	Current valid parameter record number at which the fault occurred. “0000” = No specific parameter record number available	D
-n	• TraceData	R	2 bytes	UINT16 (dec)	Start index for reporting associated trace data, if applicable. = “0000” no trace data available	D
Seg.2 -a	• RecipientLength	R	2 bytes	UINT16 (dec)	Length of the subsequent recipient of the message text in characters (bytes) = “0x00” Not specified	D
-b	• Recipient	R	Any	Visible string (text)	Message recipient	D
Seg.N -a	• Senderlength	R	2 bytes	UINT16 (dec)	Length of the subsequent sender of the message text in characters (bytes) = “0x00” Not specified	D
-b	• Sender	R	Any	Visible string (text)	Message sender	D
Seg. M -a	• TextLength				- Object 0x0018.10	D
-b	• Text				- Object 0x0018.11	D
E801	DiagHistory	R		Domain variable Record	Diagnostic archive Diagnostic information of the device in short form with history (the oldest information/lowest fault number is transmitted first) The structure is the same as that of 0x0018 DiagState	O
	• DiagState1				Oldest diagnostic state in short form	D
	…				…	D
	• DiagStateN				Newest diagnostic state in short form	D
E802	DiagHistoryLong	R		Domain variable Record	Diagnostic archive (long form) Diagnostic information of the device in long form with history (the oldest information/lowest fault number is transmitted first) The structure is the same as that of DiagStateLong	O
	• DiagStateLong1				Oldest diagnostic state in long form	D
	…					D
	• DiagStateLongN				Newest diagnostic state in long form	D

0019	ResetDiag	R/W	1 byte	UINT8 (hex)	Acknowledge diagnostic message Deletes the corresponding diagnostic memory of the device and acknowledges the diagnostic message(s).	O
		= “0x00” No restrictions for diagnostic messages or cancels existing (0x06, 0x07) restrictions = “0x01” Deletes the diagnostic history. This value is only supported if the 0xE801 “DiagHistory” or 0xE802 “DiagHistoryLong” object exists. = “0x02” Deletes (and acknowledges), where possible, all reported, and pending, not yet reported DiagStates (errors), if they have been remedied and are no longer active. This affects objects: 0x0018, 0x0033, 0x0034, 0xE800, 0xE806. = “0x03” Deletes (and acknowledges) the entire diagnostics. Combination of parameters “0x01” and “0x02” = “0x04” Deletes (and acknowledges) all DiagStates already reported using the information report, if several states have been reported. = “0x05” Deletes (and acknowledges) only the most recent (pending) reported DiagState. This affects objects: 0x0018, 0x0033, 0x0034, 0xE800 and indirectly object 0xE806. = “0x06” Deletes (and acknowledges) all errors (similar to “0x02”), also if the cause still exists; does not allow any new diagnostic messages. = “0x07” Deletes (and acknowledges) everything and does not allow any new diagnostic messages, not even in the history. Combination of parameters “0x01” and “0x06”. As a result, diagnostics are switched off. Else: Reserved. Following completion of the action, the contents are automatically reset to 0x00. Exception: 0x06 and 0x07. They remain set until another value is written. In this way, it is possible to recognize that the diagnostic function is deactivated. If the fault cause is not removed, and if the fault also cannot be removed using 0x0019 “ResetDiag”, it cannot be deleted (acknowledged) either (exception: data items 0x06 and 0x07). In this case, a negative response with error type 0x08, 0x01, 0x0022 (error class 0x08, error code 0x01, additional code 0x0022 “Service cannot be executed in the current device status. The error has not been not removed.”) is to be sent. If none or some of the objects affected by the corresponding parameters are implemented, the service will, however, be acknowledged positively. If acknowledgement is performed using object 0x0019 although there is no pending diagnostic message, a positive response is to be sent.				
001A	GetErrorRepMethod	R/W	1 byte	Bit string 8 (hex)	Fault reporting method Specifies the method used to report an fault to the higher-level system. Setting of the corresponding bit activates the method. .	O

		Bit 0 Switches generation of an information report using the contents of the “DiagState” object in the event of an fault on (= “1”)/off (= “0”). Bit 1 Switches generation of an explicit peripheral fault status message using the contents of the “DiagState” object in the event of an fault on (= “1”)/off (= “0”). Bit 2 Switches generation of a process data status message (any process data bit to be selected by the manufacturer) using the contents of the “DiagState” object in the event of an fault on (= “1”)/off (= “0”). If several report methods are selected simultaneously, the diagnostic message (and the subsequent diagnostic messages) must be stored in object 0x0018 until they have been read, even if one or more other diagnostic messages have already been sent via the report mechanism.				
0033	*DiagStateChannelNo*	*R*	*3 bytes*	*Record (2 elements)*	This object must no longer be used. It has been fully integrated in object 0x0018 “DiagState”. *Additional “channel number” information on the current diagnostic status of the device*	*X*
.01	• *Lfd.Nr.*	*R*	*2 byte*	*UINT16 (dec)*	*- Object 0x0018.1*	*X*
.02	• *ChannelNo*	*R*	*1 byte*	*UINT8 (dec)*	*If the device has channels that are grouped together, the affected channel of the corresponding group can be entered here.* *If “DiagState” is read, this object must be maintained consistently at “DiagState” at least for the next read access.* *If “DiagState” is reported as an information report, and this object is relevant for this purpose, it is also reported via an information report. Chronologically after DiagState.*	*X*
0034	*DiagStateAddValue*	*R*	*6 bytes*	*Record (2 elements)*	*This object must no longer be used. It has been fully integrated into object 0x0018 “DiagState”.* *Additional information on the “additional value” of the current diagnostic status of the device*	*X*
.01	• *Lfd.Nr.*	*R*	*2 bytes*	*UINT16 (dec)*	*- Object 0x0018.1*	*X*
.02	• *AddValue*	*R*	*4 bytes*	*UINT32 (hex)*	*A value associated with channel diagnostics can be entered here, e.g., “current temperature when exceeding the limit temperature”.* *Details: see PROFINET I&M profile* *If “DiagState” is polled, this object must be maintained consistently at “DiagState” at least for the next read access.* *If “DiagState” is reported as an information report, and this object is relevant for this purpose, it is also reported via an information report. Chronologically after DiagState.*	*X*

001B	TestMode	R/W	2 bytes	UINT16 (hex)	Test mode Switching to test mode using a manufacturer-specific code. This also activates other test parameters. Using value “0” returns to normal operating mode. = “0x0000” Normal operating mode (default)	O

The depth of the diagnostic history, i.e., the number of entries in the diagnostic archives, can be freely selected by the device developer based on the requirements and resources.

7.2.1.1 Timing of diagnostic messages

In most systems (higher-level networks, controllers, etc.), diagnostic messages are treated with a lower priority than process data. To ensure that the diagnostic messages are able to reach their addressees and the channels available for this are not overloaded, a diagnostic message must be pending for a sufficiently long time before it is reported as outgoing.
For this reason, a time of one second has been determined as the time that must be observed between an incoming and a outgoing message.

7.2.1.2 Notification by reading the DiagState object

Available diagnostic information is reported (for example, via a group module fault input (/StatErr) of the fieldbus slave protocol block) according to its priority. Several items of diagnostic information may also be available at the same time. The master or a software tool can retrieve the relevant information via the parameter channel.

Messages sent via the status bit (e.g., bit 7 in byte 0 of the process data channel) cannot be processed generically and should therefore be avoided.

The current diagnostic information in DiagState(Long) remains until

- This information has been read at least once
  and
- The cause for this diagnostic information does no longer exist

Or

- It is overwritten by a higher-priority message

Only then is next information of equal or lower priority made available (e.g., “Fault no. 2”).

The status bit(s) in the process data channel or module fault input remain(s) until

- Each item of diagnostic information has been read at least once
  and
- No more diagnostic information is available.

Only then is the actual information (“Status OK”) made available.

7.2.1.3 Notification via message (info report) from the “DiagState” object

Alternatively, errors can be reported to the master using the “Information Report” service (only once using the “DiagState” object) This function is enabled/disabled using the “GetErrorRepMethod” object. When the “GetErrorRepMethod” function is activated, each new item of diagnostic information (contents: object 0x0018) must be reported once automatically to the master (without a request from the master).
In this case, the evaluation of the information report must be present on the fieldbus master side, e.g., in the system, as a PLC function block, or HLL program.

his does not affect normal diagnostic handling of object 0x0018 “DiagState” (see above). This means:

The current diagnostic information in DiagState(Long) remains until

- The cause for this diagnostic information does no longer exist

Or

- It is overwritten by a higher-priority message

Only then is next information of equal or lower priority made available (e.g., “Fault no. 2”).

The status bit(s) in the process data channel or module fault input remain(s) until

- No more fault information is available

Only then is the actual information (“Status OK”) made available.

7.2.1.4 Classification of faults (priority of messages)

The term “fault” refers to all device application states that deviate from the normal state of the device. The importance of this state for the application is indicated by the priority.

Prio 1 (red) **Fault** (alarm)
Meaning: A fault is present that must be responded to.
It is necessary for the user to take action. A fault will not disappear without the user being involved.
Example: An fault leads to an activity in the drive, but does not necessarily require the system to be stopped with immediate effect.

Prio 2 (yellow) **Warning**
Meaning: Risk of an fault
A warning does not require action to be taken in the device.
It is not necessary for the user to take action. A warning will disappear without the user being involved.
Example: Limit value not reached or exceeded

Prio 83 (green) **Information**, message, notification (only exists as an outgoing message)
Example: General operating message, 10,000 operating hours have elapsed

Prio 81, 82 **Removed**
Meaning: The fault reported with the same number has been removed.

Messages and information are defined by the user. Warnings and faults can be predefined or defined by the user.

An outgoing message (diagnostic state) overwrites the corresponding incoming message (diagnostic state), if this has not yet been retrieved by the master. This reduces the volume of communication traffic without losing any information.
This results in the following sequence of priorities:

Coding	Priority	Meaning
0x81 – highest priority	Prio 1 (red) outgoing	Fault (alarm) removed
0x01	Prio 1 (red) incoming	Fault (alarm)
0x82	Prio 2 (yellow) outgoing	Warning removed
0x02	Prio 2 (yellow) incoming	Warning
0x83 – lowest priority	Prio 3 (green)	Information, message
0x00	-	No fault/warning/information present

If the device is in the “fault” state, parameter 0x0018.2 “DiagState.Priority” contains a value that is not equal to “0”.
If the device is not in the “fault” state, this parameter contains the “fault gone” (0x8X) value or “0” (if all gone errors have been reported and there are no more errors present).

7.2.1.5 Fault codes

The aim of standardizing the fault codes is to provide the user with quick and easy guide for remedying problems without requiring detailed knowledge of the device. A device-specific instruction for the user should be included in the diagnostic text of the fault code. There may also be several diagnostic texts (0x0018.6) for an fault code.

The fault code is represented as an octet string with the length of 2 bytes. It is coded hierarchically, beginning with a rough distinction that is gradually refined.

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

An fault code must also be interpreted accordingly (three parts).
E.g.,: 0x5112:
Faults in the device hardware (only inside the device housing), faults in the power inside and through the device, power supply: here +24 V

If the device is in the “fault” state, parameter 0x0018.4 “DiagState.Code” contains a value that is not equal to “0”.
If the device is not in the “fault” state, the parameter contains an fault code with value “outgoing” (0x8X) or “0” (if all gone errors have been reported and there are no more errors present).

7.2.1.5.1 Main groups and subgroups

Code (hex)	Meaning
0000	**No fault**
1000	**General fault**
2000	**Current (flow too high)**
2100	Fault affecting the supply of the device or “device input side” signals
2200	Fault affecting internal power supply units or “device-internal” signals
2300	Fault in the power supply or signals for the I/O devices connected to the device (device output side)
3000	**Voltage**
3100	Fault affecting the supply voltage of the device or “device input side” signals
3200	Fault affecting internal power supply units or “internal device” signals
3300	Fault on signals for the I/O devices connected to the device (device output side)
3400	Fault in the supply voltage for the I/O devices
4000	**Temperature - not within the permissible range**
4100	Ambient temperature not OK
4200	Temperature in the device not OK
4300	Temperature of an external peripheral device (e.g., drive) not OK
4400	Temperature of supply unit not OK
5000	**Device hardware fault (only inside device housing)**
5100	Fault in the supply inside and through the device
5200	Fault in the device application
5300	Fault of the operator interface and display unit on the device
5400	Fault in the power section of the device
5500	Fault in communication with the integrated additional assembly
5600	Fault in the internal data memory
6000	**Device software fault**
6100	Internal software fault (firmware)
6200	Fault in the user software on the device
6300	User parameter (data record) not OK
6800	Device configuration not OK
7000	**Fault on additional assembly/assemblies**
7100	Fault in the power section
7200	Fault in measuring circuit
7300	Fault in the sensor (as a component connected permanently to the device)
7400	Fault in the calculation circuit
7500	Fault in communication with additional assembly
7600	Fault in external data memory
7700	Open circuit/cable fault
8000	**Monitoring of device function**
8100	Fault in field communication
8200	Fault in closed-loop control
8300	Torque controller fault
8400	Speed controller fault
8500	Position controller fault
8600	Positioning controller fault
8700	Synchro controller fault
8800	Winding controller fault
8900	Fault on external sensor (separate device)
8A00	Fault in external actuator
8B00	Preventive maintenance required (condition monitoring)
8F00	Safety fault
9000	**Faults on external devices**
A000	**Fault on a modular device - subsystem**
BXXX	**Reserved**
F000	**Fault in additional functions**

7.2.1.5.2 Main groups with subgroups and details

Code (hex)	Meaning
0000	**No fault**
1000	**General fault**
	Please refer to the manual or contact the manufacturer's support team.
2000	**Current (flow too high)**
2100	**Fault affecting the supply of the device or “device input side” signals**
2110	Short circuit/ground fault on device supply voltage - general
2120	Ground fault on device supply voltage - general
2121	Ground fault, phase L1 for 3-phase connection
2122	Ground fault, phase L2 for 3-phase connection
2123	Ground fault, phase L3 for 3-phase connection
2126	VCC ground fault (supply voltage, 24 V, typical)
2127	GND ground fault
2130	Short circuit on device supply voltage – general (e.g., 24 V and GND)
2131	Short circuit, phases L1-L2 for 3-phase connection
2132	Short circuit, phases L2-L3 for 3-phase connection
2133	Short circuit, phases L3-L1 for 3-phase connection
2140	Short circuit/cross-circuit on the I/O devices
2141	Cross-circuit between signal outputs and/or other voltages
2142	Cross-circuit between a signal output and a clock signal
2146	Short circuit (cross-circuit) of a signal input to VCC
2147	Short circuit (cross-circuit) of a signal input to GND
2150	Open signal lines (input signal not connected)
2160	Signal line break/cable break (input signal)
2180	Input is not activated (not switching)
2181	Input circuit (e.g,. load) faulty
2200	**Fault affecting internal power supply units or “device-internal” signals**
2213	Overcurrent during startup
2214	Overcurrent during operation
2220	Continuous overcurrent from an internal voltage source
2221	Continuous overcurrent no. 1
2222	Continuous overcurrent no. 2
2230	Short circuit/ground fault of an internal voltage source
2240	Ground fault
2250	Short circuit
2300	**Fault in the power supply or signals for the I/O devices connected to the device (device output side)**
2310	Continuous overcurrent on peripheral supply - general
2311	Continuous current of source no. 1
2312	Continuous overcurrent of source no. 2
2320	Short circuit/ground fault in peripheral supply - general
2330	Ground fault in peripheral supply voltage
2331	Ground fault, phase U for 3-phase connection
2332	Ground fault, phase V for 3-phase connection
2333	Ground fault, phase W for 3-phase connection
2326	Ground fault, VCC (supply voltage, 24 V, typical)
2327	Ground fault, GND
2340	Short circuit/overload on the I/O devices
2341	Short circuit, phases U-V for 3-phase connection
2342	Short circuit, phases V-W for 3-phase connection
2343	Short circuit, phases W-U for 3-phase connection
2344	Overload of a signal output
2345	Overload of initiator supply
2346	Short circuit (cross-circuit) of a signal output to VCC
2347	Short circuit (cross-circuit) of a signal output to GND
2350	Continuously open signal lines (output)
2360	Signal line break/cable break (output signal)
2370	Cross-circuit
2371	Cross-circuit between signal outputs and/or other voltages
2372	Cross-circuit between a signal output and a clock signal
2380	Output is not activated (not switching)
2381	Signal output circuit (e.g., load) faulty

3000	**Voltage**
3100	**Fault affecting the supply voltage of the device or “device input side” signals**
3110	Supply overvoltage
3111	Mains overvoltage, phase L1
3112	Mains overvoltage, phase L2
3113	Mains overvoltage, phase L3
3120	Supply undervoltage
3121	Mains undervoltage, phase L1
3122	Mains undervoltage, phase L2
3123	Mains undervoltage, phase L3
3130	Supply voltage failure
3131	Phase failure L1
3132	Phase failure L2
3133	Phase failure L3
3134	Phase sequence not OK
3140	Supply voltage frequency not OK
3141	Frequency too high
3142	Frequency too low
3150	Polarity of supply voltage reversed
3180	Fault affecting the signal input – general
3181	Open signal line (input)
3182	Signal line break/cable break (input signal)
3183	Unexpected signal change at signal input
3184	Surge voltage at signal input (voltage greater than the permissible range)
3185	Signal voltage in non-defined area
3186	Undervoltage at signal input (voltage lower than the permissible range)
3200	**Fault affecting internal power supply units or “internal device” signals**
3210	Surge voltage, device-internal
3211	Surge voltage no. 1
3212	Surge voltage no. 2
3220	Undervoltage, device-internal
3221	Undervoltage no. 1
3222	Undervoltage no. 2
3230	Charging fault
3300	**Fault on signals for the I/O devices connected to the device (device output side)**
3310	Output overvoltage
3311	Output overvoltage, phase U
3312	Output overvoltage, phase V
3313	Output overvoltage, phase W
3320	Armature circuit
3321	Armature circuit interrupted
3330	Field circuit
3331	Field circuit interrupted
3340	Output undervoltage
3400	**Fault in the supply voltage for the I/O devices**
3401	Supply voltage for the I/O devices - surge voltage
3402	Supply voltage for the I/O devices – missing
3403	Supply voltage for the I/O devices - undervoltage
3410	Fault in the initiator supply - general
3411	Initiator supply - undervoltage
3412	Initiator supply missing
3413	Initiator supply - surge voltage
3420	Fault in the actuator supply - general
3421	Actuator supply - undervoltage
3422	Actuator supply missing
3423	Actuator supply - surge voltage
4000	**Temperature - not within the permissible range**
4100	**Ambient temperature not OK**
4110	Ambient overtemperature
4120	Ambient undertemperature
4130	Intake air temperature
4140	Exhaust air temperature
4200	**Temperature in the device not OK**
4210	Overtemperature in the device
4220	Undertemperature in the device
4300	**Temperature of an external peripheral device (e.g., drive) not OK**
4310	Overtemperature in the external peripheral device (e.g., drive)
4320	Undertemperature in external peripheral device (e.g., drive)
4400	**Temperature of supply unit not OK**

4410	Overtemperature of supply unit
4420	Undertemperature of supply unit
5000	**Device hardware fault (only inside device housing)**
5010	Component fault
5100	**Fault in the supply inside and through the device**
5110	Fault in low-voltage power supply units - general
5111	Supply +/-15 V
5112	Supply +24 V
5113	Supply +5 V
5114	Supply +3.3 V
5115	Supply +2.5 V
5116	Supply +1.2 V
5118	U8 = Manufacturer-specific
....	..... manufacturer-specific for U8 to U15
511F	U15 = Manufacturer-specific
5120	Fault in the air supply
5130	Fault in the paint supply
5140	Fault in the supply of the intermediate circuit
5150	Fault in the supply (power supply unit) of the initiator
5151	Internal short circuit
5160	Fault in the supply (power supply unit) of the I/O devices supplied by the device.
5200	**Fault in the device application**
5210	Measuring circuit faulty
5220	Calculation circuits faulty
5230	Communication (device-internal only) faulty
5300	**Fault of the operator interface and display unit on the device**
5400	**Fault in the power section of the device**
5410	Output stage faulty
5420	Chopper faulty
5430	Input stages faulty
5440	Contactors/relays faulty
5441	Channel 1
…	…
5448	Channel 8
5450	Fuses defective
5451	S1 = L1
5452	S2 = L2
5453	S3 = L3
5454	S4 = Manufacturer-specific
....	.... for S5, S6, S7, S8
5459	S9 = Manufacturer-specific
5500	**Fault in communication with the integrated additional assembly**
5510	Interface no. 1
5520	Interface no. 2
5600	**Fault in the internal data memory**
5610	Permanently installed RAM faulty
5620	Permanently installed EPROM faulty
5630	Permanently installed EEPROM faulty
5640	Permanently installed flash faulty
6000	**Device software fault**
6010	Software reset (watchdog) occurred
6100	**Internal software fault (firmware)**
6110	Firmware missing (e.g., SD card missing)
6120	Firmware inconsistent (e.g., SD card contains corrupt data)
6130	Application on the device not yet ready (part of application still booting, self-tests being performed, etc.)
6200	**Fault in the user software on the device**
6210	Process data index not available
6211	Variable number not available
6300	**User parameter (data record) not OK**
6301	Data record no. 1
..	from 2 to 14 accordingly
630F	Data record no. 15
6310	Loss of internally stored user parameters
6320	Inconsistency between internally stored user parameters (e.g., identified by CRC)
6330	User parameters/startup parameters not yet initialized
6340	Interdependent user parameters/startup parameters are not consistent with one another
6800	**Device configuration not OK**
6810	Data in the process data configuration has been mapped twice

7000	**Fault on additional assembly/assemblies**
	Permanently connected to the device, can be applied by the user, if necessary, part of complete device, usually delivered with device
7100	**Fault in the power section**
7110	Brake chopper defective
7111	Brake chopper failure
7112	Brake chopper overcurrent
7113	Brake chopper circuit
7120	Motor defective
7121	Motor blocked
7122	Motor missing or commutation faulty
7123	Motor tilted
7140	Replaceable relay/contactor (by the user) faulty
7150	Replaceable fuse (by the user) faulty
7200	**Fault in measuring circuit**
7300	**Fault in the sensor (as a component connected permanently to the device)**
7301	Tachometer faulty
7302	Tachometer polarity reversal
7303	Resolver 1 faulty
7304	Resolver 2 faulty
7305	Incremental encoder 1 faulty
7306	Incremental encoder 2 faulty
7307	Incremental encoder 3 faulty
7308	Sensor 8 faulty
…	Sensor 9 to 14 accordingly
730F	Sensor 15 faulty
7310	Speed sensor faulty
7320	Position sensor faulty
7400	**Fault in the calculation circuit**
7500	**Fault in communication with additional assembly**
7501	Short circuit in the data signal line
7502	Line break in data signal line
7503	Fault in data signal transmission (e.g., due to EMC)
7510	Serial interface no. 1 faulty
7520	Serial interface no. 2 faulty
7600	**Fault in external data memory**
7610	Replaceable RAM faulty
7620	Replaceable EPROM faulty
7630	Replaceable EEPROM faulty
7640	Replaceable flash faulty
7700	**Open circuit/cable fault**
7701	Cable 1 faulty
…	Cable 2 to 14 accordingly
770F	Cable 15 faulty
7710	Open circuit, sensor cable
8000	**Monitoring of device function**
8100	**Fault in field communication**
8110	Process data update timeout elapsed
8120	Host data update timeout elapsed
8121	PD channel handshake timeout elapsed
8150	Buffer handling faulty
8151	Send buffer, general
8152	Send buffer full
8153	Send buffer overflow
8159	Receive buffer, general
815A	Receive buffer full
815B	Receive buffer overflow
8200	**Fault in closed-loop control**
8210	System deviation, desired < actual. Deviation present longer than a specified period of time (manufacturer-specific)
8211	Maximum manipulated variable reached/exceeded
8220	System deviation, desired < actual. Deviation present longer than a specified period of time (manufacturer-specific)
8221	Minimum manipulated variable reached
...	Reserved for profile-specific system errors
827f	Reserved for profile-specific system errors
8300	**Torque controller fault**
8311	Excess torque
8312	High-inertia starting

8313	Static torque
8321	Insufficient torque
8331	Torque break
8400	**Speed controller fault**
8500	**Position controller fault**
8600	**Positioning controller fault**
8611	Following fault
8612	Reference limit exceeded
8700	**Synchro controller fault**
8800	**Winding controller fault**
8900	**Fault on external sensor (separate device)**
8910	Measuring value overrange
8920	Measuring value underrange
8930	Additional circuit for external sensor faulty
8A00	**Fault in external actuator**
8B00	**Preventive maintenance required (condition monitoring)**
8F00	**Safety fault**
8F01	Symmetry violation (switching sequence incorrect, discrepancy time monitoring, antivalent, equivalent)
8F08	Light test fault (output cannot be switched on when inactive)
8F09	Dark test fault (output cannot be switched off when active)
8F0F	Failure state (critical fault, safety not guaranteed, the module is shut down)
9000	**Faults on external devices**
A000	**Fault on a modular device - subsystem**
A001	No module present/module missing
A002	Incorrect module present
A003	Replaced (the expected module was replaced with a compatible module)
A004	More modules in the subbus than expected
A010	General fault in the module
A012	Application on the module not ready
A013	Module has executed a reset
A014	Module parametrization missing or faulty
A020	Subbus communication fault
A021	Subbus fault - timeout
A022	Multiple transmission errors on the subbus
A023	Subbus I/O data communication data
A024	Subbus management data communication fault
A025	Ring fault – the “open ring” bus structure is interrupted
A030	Subbus configuration fault
A031	I/O data configuration mapped twice
A033	The real process data length does not match the configured process data length. (is longer)
A040	Common errors
A041	Hardware fault – module must be replaced
A042	Firmware fault – module firmware or the entire module must be replaced
A043	Subbus asynchronous to higher-level system
BXXX	**Reserved**
F000	**Fault in additional functions**
F001	Delay
F002	Subsynchronous operation
F003	Lifting system
F004	Control

Codes that are not listed are reserved.
NOTE: additional fault codes are defined in the profiles.

If a manufacturer cannot assign an fault to any of the above codes, the INTERBUS Club should be notified.

7.2.2 Trace data

Traces are consecutive records of specific device data. They can be used, for example, to recapitulate sequences in order to detect errors or problems, or to archive data.
Basically, there may be very different traces. Contents, structure, scope, and handling are determined by the device manufacturer.

Index (hex)	**Object name**	**R/W**	**Length**	**Data type** (representation)	**Meaning**	**M/O**
001C	ControlTrace	R/W	1 byte	UINT8 (hex)	Logging control Controls. a recording process “0x00” Resets the recording and the corresponding parameters “0x01” Starts a recording “0x02” Stops a recording Additional details are specified by the manufacturer	O
E803	TraceBuffer	R	N bytes	Domain variable octet string (hex)	Logging buffer Contains the recorded trace data. The structure is specified by the manufacturer	D

Trace data is generally transmitted using the upload system. When uploading trace data, the basic information should be contained in the header.

7.3 User data management

7.3.1 Process data management

This section defines the behavior of process data. This includes the behavior in the event of a timeout and fieldbus reset as well as reading and writing process data via the parameter channel.

Index (hex)	**Object name**	**R/W**	**Length**	**Data type** (representation)	**Meaning**	**M/O**
001F	PDTimeout	R/W	2 bytes	UINT16 (dec)	Process data monitoring time Maximum permissible time in ms (e.g., 100 ms) by which new data must have been transmitted via the process data channel. The action that is then triggered is specified in object 0x0020 “PDTimeoutCode”. 0xFFFF = Disabled	O
0020	PDTimeoutCode	R/W	N x 2 bytes	Array of UINT16 (hex)	Process data monitoring code If the process data monitoring time elapses, the function assigned to the “process data monitoring code” is carried out by the device. This type of function can be, for example, the defined shutdown of a drive or the continuous transmission of fault telegrams via a serial interface. 0x0000 “0” is output to all output bits 0x0001 “1” is output to all output bits 0x0002 Hold the last valid value 0x0003 Accept the substitute value from object 0x002F “PDOUT_Subst” 0x0004 Output the minimum permissible value 0x0005 Output the maximum permissible value 0x0010 Only in the case of subbuses: apply the subbus-specific behavior 0x0006 ... 00x0009 and 0x0011 ... 0x7FFF Reserved 0x8000 up to 0xFFFF Manufacturer-specific Since different behavior may be required for each channel/each group, this parameter is set up as an array The number of array elements is defined by the number of elements of the corresponding PDOUT (0x0026) / (PDIN 0x0025) object. N= Number of elements If only one entry is supported (written) although the device has several elements (channels/groups), the parameter is valid for the entire device.	O
0024	ResetCode	R/W	N x 2 bytes	Array of UINT16 (hex)	Fieldbus reset code Substitute value behavior during bus reset (PDOUT) Substitute value behavior when valid process data is missing	O

<table>
<tr><td></td><td></td><td colspan="4">In the event of a bus reset, the function assigned to the “ResetCode” is carried out by the device.

This type of function can be, for example, the defined shutdown of a drive or the continuous transmission of fault telegrams via a serial interface.

0x0000 “0” is output to all output bits
0x0001 “1” is output to all output bits
0x0002 Hold the last valid value
0x0003 Accept the substitute value from object 0x002F “PDOUT_Subst”
0x0004 Output the minimum permissible value
0x0005 Output the maximum permissible value
0x0010 Only in the case of subbuses: apply the subbus-specific behavior

0x0006 ... 00x0009 and 0x0011 ... 0x7FFF Reserved
0x8000 up to
0xFFFF Manufacturer-specific

Since different behavior may be required for each channel/each group, this parameter is set up as an array The number of array elements is defined by the number of elements of the corresponding PDOUT (0x0026) / (PDIN 0x0025) object.

N= Number of elements
If only one entry is supported (written) although the device has several elements (channels/groups), the parameter is valid for the entire device.

See also “Substitute value behavior/switch-on behavior” section</td><td></td></tr>
<tr><td>0030</td><td>PF_Code</td><td>R/W</td><td>N x 2 bytes</td><td>Array of UINT16 (hex)</td><td>Peripheral fail code
Substitute value behavior during peripheral faults (PDIN)
In the event of an fault on the I/O devices of the fieldbus protocol chip, which prevents the processing of input process data, the function specified here is carried out by the device.

This type of function can be displayed, for example, via the /StatErr input or µP WD on the protocol chip.

0x0000 “0” is output to all input bits
0x0001 “1” is output to all input bits
0x0002 Hold the last valid value
0x0003 Accept the substitute value from object 0x0031 “PDIN_Subst”
0x0004 Output the minimum permissible value
0x0005 Output the maximum permissible value
0x0010 Only in the case of subbuses: apply the subbus-specific behavior

0x0006 ... 00x0009 and 0x0011 ... 0x7FFF Reserved
0x8000 up to
0xFFFF Manufacturer-specific

Since different behavior may be required for each channel/each group, this parameter is set up as an array The number of array elements is defined by the number of elements of the corresponding PDOUT (0x0026) / (PDIN 0x0025) object.

N= Number of elements
If only one entry is supported (written) although the device has several elements (channels/groups), the parameter is valid for the entire device.</td><td>O</td></tr>
</table>

0025	PDIN	R	PD length	Octet string[N]	IN process data Input process data Mapping of the IN process data (from the device to the master) to an object The PD bytes, like the bytes in the octet string, are counted in ascending order starting at the (top) left with 0. If the process data is structured (e.g., several channels), this object should also be structured and individual structure elements accessed via the subindex.	**M**
.01	• Part 1	R		Dependent	First part of the process data item, e.g., value of the first channel or the first function group	**O**
.02	• Part 2	R		Dependent	Second part of the process data item, e.g., value of the second channel or the second function group	**O**
...	•					
.N	• Part N	R		Dependent	Nth part of the process data item, e.g., value of the Nth channel or the Nth function group	**O**
0026	PDOUT	R/W_D	PD length	Octet string[N]	OUT process data Output process data Mapping of the OUT process data (from the master to the device) to a parameter object.	**M**
		Data should be accessed that maps the peripheral states as accurately as possible. This also applies to the substitute values. The PD bytes, like the bytes in the octet string, are counted in ascending order starting at the (top) left with 0. If the process data is structured (e.g., several channels), this object should also be structured and individual structure elements accessed via the subindex. In any case, data consistency of a value (e.g., all bytes of a 32-bit analog value) must be ensured. If the “GetExRight” object is not implemented, the status of the PDOUT object is “Read-only”. If the “GetExRight” object is implemented, the status of the PDOUT object is “Read and write”. If the value for the “GetExRight” object = “0”, write access to the PDOUT object is rejected with Error class “8” - Other Error code “1” – Profile specific Additional code 0x0022 “Service cannot be executed in current device state.” **NOTE:** If access to OUT process data, for example, is possible using tools, it should be noted that potential process data watchdog functions may not work here.				
.01	• Part 1	R		Dependent	First part of the process data item, e.g., value of the first channel or the first function group	**O**
.02	• Part 2	R		Dependent	Second part of the process data item, e.g., value of the second channel or the second function group	**O**
...	•					
.N	• Part N	R		Dependent	Nth part of the process data item, e.g., value of the Nth channel or the Nth function group	**O**

002F	PDOUT_Subst	R/W	PD length	Dependent	OUT process data substitute Substitute value for OUT process data Substitute value for the OUT process data (from the master to the device) in the event of fault. The PD bytes, like the bytes in the octet string, are counted in ascending order starting at the (top) left with 0. **D**ependency info: Must be implemented if the value 0x0003 is permitted for object 0x0020 “PDTimeoutCode” or object 0x0024 “ResetCode”. See relevant section	D
0031	PDIN_Subst	R/W	PD length	Dependent	IN process data substitute Substitute value for IN process data Substitute value for the IN process data (from the device to the master) in the event of an fault in the connected I/O devices. The PD bytes, like the bytes in the octet string, are counted in ascending order starting at the (top) left with 0. **D**ependency info: Must be implemented if the value 0x0003 is permitted for object 0x0030 “PF_Code”. See relevant section	D
0027	GetExRight	R/W	1 byte	UINT8 (hex)	Get exclusive process data write rights Request exclusive write access This parameter can be used to request exclusive write access to the process outputs via the parameter channel. Following a positive confirmation, the data is no longer updated via the process data channel. A change to the process outputs are made via the “PDOUT” object, which can now be read from and written to. The exclusive rights are reset each time a connection is aborted or the bus is reset. **Note:** This action may have serious consequences for the connected process. This is why this object should be password-protected. = “0x00” Output data via the PD channel = “0x01” Output data via the parameter channel Else: Reserved **NOTE:** If access to OUT process data, for example, is possible using tools, it should be noted that potential process data watchdog functions may not work here.	O

0028	ChangePDSet	R/W	2 bytes	UINT16 (dec)	Change process data settings Set process data assignment Selects one of the possible process data assignments defined by the manufacturer and activates it. 0x0000 Default 0x0001 to 0x7FFF Reserved 0x8000 to 0xFFFF Manufacturer-specific The change to the process data assignments usually also the contextual significance and structure of the process data might require a new assignment of the process data. If, as a result, the data length, device type and/or length code change, these changes can only come into effect with the next bus reset or if explicitly instigated by the bus master.	O
003B	PDIN_Descr	R	N x 12 bytes	Array of records (N x 3 elements)	Process data description IN process data description Description of the process data structure. This enables tools and systems to automatically assign the process data the correct variables, to take into account the endianness or to assign it to a suitable connection. All types can occur any number of times. The description must be continuous and arranged in ascending order, starting on the left. The number of array elements is defined by the number of elements of the PDIN (0x0025) object. N = Number of elements PDIN (0x0025)	**M**

.01	• Type	R	8 bytes	Octet string[8] (text)	Type of I/O data item. The following are currently defined: Management data types: • "ACK" - Response to command • "DIAG" - Diagnostic information • "NC" - Not connected, not in use • "STATUS" - Status information User data types: • "AI" - Analog IN • "AO_F" - Analog OUT - feedback • "CmpP" - Complex protocol • "CNT" - Counter • "DI" - Digital IN • "DO_F" - Digital OUT - feedback • "DRV" - Drive • "MotSt" - Motor starter • "NO" - Number • "PLC" - PLC (data of an intelligent module) • "POS" - Positioning data • "PWM" - PWM data • "SFCH" - Channel for safe data • "SubMa" - Subbus master • "SubSl" - Subbus slaves container Preceding "S_" (e.g., "S_DO_F") indicates a safety data type. The type is exactly 8 characters long. Unused characters are to be filled with 0x00. Example: "AI" - `0x41 49 00 00 00 00 00 00`	**M**
.02	• ChNo	R	2 bytes	UINT16 (dec)	Channel number Number of channels of a type Note: A channel always refers to a number of bits (even when there is only one) which collectively have a meaning. Example: Four analog inputs represent four channels or 16 digital inputs represent 16 channels	**M**
.03	• ChLength	R	2 bytes	UINT16 (dec)	Channel length Length of a channel in bits	**M**
003C	PDOUT_Descr	R	N x 12 bytes	Array of records (N x 3 elements)	Process data description OUT process data description Description of the process data structure This enables tools and systems to automatically assign the process data the correct variables, to take into account the endianness or to assign it to a suitable connection. All types can occur any number of times. The description must be continuous and arranged in ascending order, starting on the left. The number of array elements is defined by the number of elements of the PDOUT (0x0026) object. N = Number of elements PDOUT (0x0026)	**M**

.01	• Type	R	8 bytes	Octet string[8] (text)	Type of I/O data item. The following are currently defined: Management data types: • "CMD" - Commands • "CTRL" - Control • "NC" - Not connected, not in use User data types: • "AO" - Analog OUT • "CmpP" - Complex protocol • "CNT" - Counter • "DO" - Digital OUT • "DRV" - Drive • "MotSt" - Motor starter • "NO" - Number • "PLC" - PLC (data of an intelligent module) • "POS" - Positioning data • "PWM" - PWM data • "SFCH" - Channel for safe data • "SubMa" - Subbus master • "SubSl" - Subbus slaves container The type is exactly 8 characters long. Unused characters are to be filled with 0x00. Example: "CNT"- `0x43 4E 54 00 00 00 00 00`	**M**
.02	• ChNo	R	2 bytes	UINT16 (dec)	Channel number Number of channels of a type Note: A channel always refers to a number of bits (even when there is only one) which collectively have a meaning. Example: Four analog inputs represent four channels or 16 digital inputs represent 16 channels	**M**
.03	• ChLength	R	2 bytes	UINT16 (dec)	Channel length Length of a channel in bits	**M**

Example of a four-channel analog OUT module:
Content of 0x003B "PDIN_Descr", 2 entries (hex):

```
53 54 41 54 55 53 00 00 00 10 00 01
4E 43 00 00 00 00 00 00 00 08 00 08
```

Content of 0x003C "PDOUT_Descr", 2 entries (hex):

```
4E 43 00 00 00 00 00 00 00 02 00 08
41 4F 00 00 00 00 00 00 00 04 00 10
```

7.3.2 Substitute value behavior/power-on behavior

A distinction is made between:

7.3.2.1 Power-on behavior:

Following power-on of the power supply, the inputs/outputs must adopt a safe (defined) state within the meaning of the Machinery Directive (inactive, “0”, typical).
Following power-on, this state must be maintained until

- The device has valid parametrization that uses the inputs/outputs (not deactivated)
- And
  - Valid output data has been received
    Or
  - Valid input data can be provided

Where possible, this involves extending the “PowerOff” state.
Unused (deactivated) inputs/outputs do not exit power-on behavior. If they have been active before, the inputs/outputs change to power-on behavior.

Note: *Following the start, it is not mandatory to write a parameter in order to change to this state if there is already a channel being actively used in the locally stored parametrization.*

7.3.2.2 Substitute value behavior (failsafe behavior):

In the event of the **absence of valid output data***) the device must change to the “Substitute value behavior” state with regard to its outputs, if

- A parametrization exists for which the outputs are used (are activated)

And

- Valid process data has been exchanged at least once before

For the OUT process data, the substitute value behavior is maintained until valid OUT process data is received.

If **no valid input data can be made available**, the device must change to the “Substitute value behavior” state, with regard to its inputs, if

- A parametrization exists for which the inputs are used (are active)

Only inputs/outputs used adopt the substitute value behavior. Inputs/outputs not used do not exit power-on behavior.

Note: *Following the start, it is not mandatory to write a parameter in order to achieve this state if there is already a channel being actively used within the locally stored parametrization.*

*) The system determines in which way “absence of valid output data” is defined. Usually, it is defined by expiry of a timeout (reset), receipt of a corresponding command, or expiry of the validity of a process data determiner.

7.3.4 Functional safety – safety

Entry in object 0x0005 “Capabilities”: “Safety0”

The basic profile does not describe methods or protocols that ensure secure data transmission in the sense of functional safety.
The support for safety devices is limited to ensure that they can be identified in the system. Objects and entries are defined at the appropriate positions to enable the master to recognize

- That it is a device from the “Safety” device range (DeviceFamily, index 0x0004)
- Where the safety-relevant data can be found in the process data channel (PDIN_Descr, index 0x003B, and PDOUT_Descr, index 0x003C)
- That safe I/O data can be read by the standard system too (PDIN, index 0x0025, and PDOUT, index 0x0026)
- Which errors are relevant to safety (fault code 0x8FXX)
- Which safety protocols are supported (SafetyProtType Index 0x0047.1)
- Etc.

7.3.5 Parameter channel management

This section specifies the performance of the parameter channel in the case of timeout and communication abort.

Index (hex)	**Object name**	**R/W**	**Length**	**Data type** (representation)	**Meaning**	**M/O**
0021	PChTimeout	R/W	2 bytes	UINT16 (dec)	Parameter channel monitoring time Maximum permissible time in ms (e.g., 500 ms) by which new data must have been transmitted via the parameter channel/the service must have been completed. This applies equally to both read and write operations. The action that is then triggered at the slave is specified in object 0x0022 “PChTimeoutCode”. The master can cancel the request, if it has not received confirmation of its request within this time, and report a PCh timeout to the higher-level system. Exception: “Busy” Default: 500 ms 0xFFFF = Disabled The slave must reject any times that it is not able to realize.	**M**
0022	PChTimeoutCode	R/W	2 bytes	UINT16 (hex)	Parameter channel monitoring code If the communication monitoring time elapses, the function assigned to the parameter channel monitoring code is carried out. This type of function can be, for example, the defined shutdown of a drive or the continuous transmission of fault telegrams via a serial interface. 0x0000 No action (default) 0x0001 Repetition of the service (expect) 0x0002 Expect abort of the service 0x0003 Perform active abort of the service 0x0004 to 0x7FFF Reserved 0x8000 to 0xFFFF Manufacturer-specific	O
0023	AbortCode	R/W	2 bytes	UINT16 (hex)	Connection abort code If a connection is aborted on the parameter channel, the function that is assigned to the connection abort code is carried out. This type of function can be, for example, the defined shutdown of a drive or the continuous transmission of fault telegrams via a serial interface. 0x0000 No action (default) 0x0001 to 0x7FFF Reserved 0x8000 to 0xFFFF Manufacturer-specific	O

7.4 Device management

Field devices use increasingly complex functions, which must be suitably parameterized by the application or software tools. The following establishes a basic structure that enables the usual principal handling procedures to be standardized.

Dependency info:
If object 0x0029 “ParamSetWriteControl” is implemented, object 0x002A “ConflictDictionary” must also be implemented.

Index (hex)	**Object name**	**R/W**	**Length**	**Data type** (representation)	**Meaning**	**M/O**
0029	ParamSetWriteControl	R/W	1 byte	UINT8 (hex)	Block parameterization control See “Interdependent parameters” section below	O
002A	ConflictDictionary	R	N x 8 bytes	Array of records N x 6 elements	Conflict dictionary Result of block parameterization See “Interdependent parameters” section below	D
First interdependent parameter in conflict with at least one other parameter.						
.01	• ConfGrNo_1	R	1 byte	UINT8 (dec)	Conflict group number of the first dependent parameter This number allows the following parameter to be assigned to one of several groups of parameters that are in conflict with one another.	D
.02	• Subslot_1	R	1 byte	UINT8 (dec)	Subslot of the first dependent parameter	D
.03	• Index_1	R	2 bytes	UINT16 (hex)	Index of the first dependent parameter	D
.04	• Subindex_1	R	1 byte	UINT8 (hex)	Subindex of the first dependent parameter	D
.05	• Element_1	R	1 byte	UINT8 (hex)	Element in the record of the first dependent parameter	D
.06	• Add. Code_1	R	2 bytes	UINT16 (hex)	Add. information on the first dependent parameter	D
	• …					
Nth interdependent parameter that is in conflict with at least one other parameter.						
.6xN-5	• ConfGrNo_N	R	1 byte	UINT8 (dec)	Conflict group number of the Nth dependent parameter	D
.6xN-4	• Subslot_N	R	1 byte	UINT8 (dec)	Subslot of the Nth dependent parameter	D
.6xN-3	• Index_N	R	2 bytes	UINT16 (hex)	Index of the Nth dependent parameter	D
.6xN-2	• Subindex_N	R	1 byte	UINT8 (hex)	Subindex of the Nth dependent parameter	D
.6xN-1	• Element_N	R	1 byte	UINT8 (hex)	Element in the record of the Nth dependent parameter	D
.6xN	• Add. Code_N	R	2 bytes	UINT16 (hex)	Add. information on the Nth dependent parameter	D

002B	ParamSet	R/W	2 bytes	UINT16 (hex)	Parameter record identification (index) if the device has parameter records, the active parameter record (number/index) is set or displayed here. If there is an existing description of the parameter record, the parameter record identification is equivalent to the index of the description. See “Parameter record identification” section	O
		• 0x0000 The parameter record of the device was not yet initialized via the bus. The default parameter record applies. • 0x0001 – 0xFEFF The parameter record of the device, which has been defined via mechanisms of the basic profile or freely by the manufacturer, was initialized via the bus. For identification, it recived the freely definable number/index 0x0001 - 0xFEFF. • 0xFF00 – 0xFFFD Reserved • 0xFFFE It is not guaranteed that the previously loaded parameter record has not changed. This value can only be read. • 0xFFFF The device was switched to “local mode” or manual mode and it is not guaranteed that the previously loaded parameter record has not changed.				
002C	ParameterMoment	R/W	20 bytes	Record (2 elements)	Time of last parameterization modification	O
.01	• Date	R/W	10+1 bytes	Visible string (text)	Date YYYY/MM/DD	D
.02	• Time	R/W	8+1 bytes	Visible string (text)	Time hh:mm:ss	D
002D	ResetParam	R/W	1 byte	UINT 8 (hex)	Reset parameterization Used as a command to undo all settings and replace them with default values.	**M**

		0x00 No action 0x01 Reset parameterization This command is used to undo all settings and replace them with factory default values with regard to the state defined in 0x47.3 “LegacyInfo”. This also applies to passwords and other user-defined settings. This value is **mandatory**. 0x02 Reset application parameterization This command is used to undo just the application parameters and replace them with default values with regard to the state defined in 0x47.3 “LegacyInfo”. The parameters listed in the “Identification” section are not application parameters. 0x03 Legacy reset As with 0x01, relating only to the delivery state. A so-called “out-of-the-box” state is created. 0x04 Reset legacy application parameterization. As with 0x02, relating only to the delivery state. Following completion of the action, the contents are automatically reset to 0x00. If object 0x47.3 “LegacyInfo” is not available, values 0x01 and 0x03 as well as 0x02 and 0x04 have the same meaning.				
002E	ParamHash	R	4 bytes	UINT32 (hex)	Hash value Device-specifically generated, unique value (e.g., CRC16) which ensures integrity of the parameter data. The security mechanism used for this involves all the device parameter data. The hash value is regenerated each time a parameter is modified and enables the user to check whether all parameter settings are still unchanged. To do this, the current hash value must be compared with the hash value read during parameterization. The way the hash value is generated is left up to the manufacturer. A “change counter” is also a possibility. The important thing is that the same hash value is supplied for identical parameter records.	O

0040	ListOfObjToRestore	R	N x 4 bytes	Array of records (3 elements)	List of objects to be restored List of objects to be restored List of all application objects [submodule no. (1 byte) + index (2 bytes) + subindex (1 byte)] with content that must be restored in the event of device replacement. Ssee “Device replacement application” section) “Factory default” is restored for all objects via object 0x002D “ResetParam”. Only the objects with content that currently deviates from the content prescribed in the “Factory default” needs to be considered. The list can but does not have to be dynamic. In the case of simple devices, this can include all application objects. It could, however, also only be object 0xE809 “BackUpDataCompr”.	**M**
.01	• SubModNo1 • Index1 • Subindex1	R	4 bytes	Record (3 elements)	Submodule number, index and subindex of the first object (Subindex = “0” is permissible if the entire object can be read/written as a result)	M
.02	• SubModNo2 • Index2 • Subindex2	R	4 bytes	Record (3 elements)	Submodule number, index and subindex of the second object (Subindex = “0” is permissible if the entire object can be read/written as a result)	M
	•					
.N	• SubModNoN • IndexN • SubindexN	R	4 bytes	Record (3 elements)	Submodule number, index and subindex of the Nth object (Subindex = “0” is permissible if the entire object can be read/written as a result)	M

Note on object 0x0040 “ListOfObjToRestore”:
The system (higher-level network/controller) decides whether parameters are generally to be stored and how modifications are to be identified (in particular when they are independent of the system and supplied directly into the module from outside).

7.4.1 Device replacement application

With the help of this parameter list, device replacement can be optimally carried out using the following procedure:
Following/during startup of the “system”, the contents of the parameters in this list (this parameter record) are to be “saved”.
If device replacement is required, a factory reset must always be carried out first on the compatible replacement device, and the previously saved parameter record then be written to the replacement device.

7.4.2 Interdependent parameters

Different parameters may be dependent on one another. This means that the validity of a parameter value may be dependent on other parameters and can thus only be evaluated if all the interdependent parameters use their final values.
The “block parameterization” mechanism is used to ensure this.

7.4.2.1 Block parameterization

Block parameterization serves to enable the joint transfer of interdependent parameters. In terms of communication, writing of several interdependent parameters using block parameterization is considered and treated like writing a single parameter.
Block parameterization is introduced via parameter “*ParamSetWriteControl = 01”* and terminated using “*ParamSetWriteControl = 00”*.
The dependencies are only checked upon completion of block parameterization. Correspondingly, any resulting conflicts are reported at this time and not during writing of the relevant parameters.
Any conflicts that occur are stored in the “Conflict dictionary” (object ConflictDictionary, index 0x002A).

7.4.2.1.1 Block parameterization control (ParamSetWriteControl Index 0x0029)

Transition between individual parameterization and block parameterization is triggered using this parameter.

ParamSetWriteControl = “0x00”: terminates block parameterization, individual parameterization from now on
ParamSetWriteControl = “0x01”: initiates block parameterization

The following actions are carried out when the parameter contents are modified:

Initiation of block parameterization (ParamSetWriteControl = “0x00” -> “0x01”)
- Initiation of block parameterization
- No individual parameterization
- Reset of the “Conflict dictionary” list parameter

Completion of block parameterization (ParamSetWriteControl = "0x01" -> "0x00")

- Termination of block parameterization, individual parameterization from now on
- Assessment of parameter compatibility (parameter conflicts)
  It is not mandatory for all the parameters used for assessment to have been written as part of block parameterization. The module itself determines the objects that are to be used for the plausibility check.
  - o If compatible,
    - The parameter contents are accepted
    - Write access to the "ParamSetWriteControl" parameters is acknowledged positively
  - o If incompatible,
    - The old contents of all the parameters required for block parameterization remain in effect and the newly written parameters have no effect
    - The parameter is entered in the "Conflict dictionary"
    - Write access to the "ParamSetWriteControl" parameter is acknowledged negatively, as follows.

Error class "8", other, error code "1", profile-specific

Additional code	Meaning
0x0040 DependencyIgnored	Collision with other values, dependency ignored As a rule, dependent values were not taken into consideration.

7.4.2.1.2 Conflict dictionary (ConflictDictionary - index 0x002A)

The parameter contains the indices and error types (additional code) for the parameter involved in the conflict.

- If the values of two or more interdependent parameters were to cause a conflict, these parameters would be entered in the conflict dictionary.
- If the values of two or more interdependent parameters no longer cause a conflict, these parameters are deleted from the conflict dictionary.

This applies whether block parameterization is active or not.

The conflict dictionary must be updated in the event of:

- Negative acknowledgement in the case of completion of block parameterization (negative confirmation to writing of ParamSetWriteControl = "0x01" -> "0x00")
- Individual parameterization of interdependent parameters

The conflict dictionary can be updated:

- After each write access to dependent parameters in "Block parameterization" state

The conflict dictionary is deleted:

- On initiation of block parameterization
- When writing any other interdependent parameter outside block parameterization (individual parameterization)

The conflict dictionary is empty if no consistency violations are identified during parameterization.

The conflict dictionary can only be filled up to the maximum PDU size. If additional entries exist, these will be omitted.

7.4.3 Parameter record identification

This parameter is used to identify the currently effective device parameter record. These can, for example, refer to a welding program or formula.
Parameter records can be predefined by the manufacturer or loaded by the user depending on the device using individual or block parameterization or "Download Write" services.
If the device cannot save the received device parameter values so that they are not lost in the event of a power failure, the device automatically sets the parameter record identification to “0000” when the mains voltage is switched on. The device user can evaluate this information and reinitialize accordingly.

A parameter record is always a combination of parameters. There are various handling options available:

1. The combination can be specified by the manufacturer but cannot be read via objects. It is simply assigned a number. It is therefore also not mandatory for the objects that are comprised in the parameter record to be described in object 0x002B.02 - 0x002B.N “ParamSet. ObjectN”. The meaning can be described in the user manual.

2. The combination can be specified by the manufacturer, and the content and the structure are made available via object XYZ. It is assigned a number. The XYZ object must then be defined as described below.
   In this case, the parameter record can only contain objects that can be addressed individually via their individual index/subindex.

3. Same as 2., but the manufacturer allows users to define parameter records themselves. In this case, object XYZ can be written.

Definition of the object that contains a parameter record:

Index (hex)	**Object name**	**R/W**	**Length**	**Data type** (representation)	**Meaning**	**M/O**
XYZ	ParamSet1	R/W_D	N bytes	Record (3 elements)	Parameter record for an example	O
.01	• NoOfObj	R/W_D	2 bytes	UINT16 (dec)	Number of objects Number of objects combined in this parameter record	O
.02	• Structure	R/W_D	N x 5 bytes	Array of records (3 elements)	Here, the combined objects are listed in the form • Index (2 bytes) • Subindex (1 byte) • Length of the user data in bytes (2 bytes)	O
.03	• UserData	R/W_D	N bytes	Record (N elements)	The user data of the objects listed above is given consecutively without separator here.	O

7.4.4 Volatile and non-volatile parameters

A number of objects contain non-volatile parameters. This means that they are rarely written, maybe during startup, and are maintained even after power-down.
There are, however, also a number of objects with content that is constantly changing and with data that is lost in the case of power-down.

In both cases, the behavior is defined by the name.

There also are mixed forms, although far less frequently.

(1) Objects with parameters that need to be defined in the event of power-on but which might be constantly changing during the runtime.

Such parameters must be spread to the basic forms. This means that there is one object that contains the volatile data and one associated object that contains the non-volatile start value of the volatile parameter following power-on. Therefore, handling of the flash memory commonly used is taken into account.

(2) Objects with parameters that need to maintain the last of the constantly changing values recorded during runtime require appropriate storage technology, no definition of handling.

7.4.5 Data backup

Index (hex)	**Object name**	**R/W**	**Length**	**Data type** (representation)	**Meaning**	**M/O**
E809	BackUpDataCompr	UR/ DW	N bytes	Domain variable octet string (hex)	Compressed data for device backup The object contains data that is necessary to back up/restore the device. This can but must not be all the device data. The structure of this object is not important. It is device-/manufacturer-specific. It must only be ensured that the device behaves identically when the content of this object is read from a device, and when this content is successfully loaded to object 0xE809 “BackUpDataCompr” of a device of the same device type code. It is permissible that less data is transmitted to the slave (e.g., newer FW) than the slave might be expecting. (The content of the BackUpDataCompr comes from a device with an older FW, for example) For reasons of compatibility, additional data should only be attached at the end. The parameters that have not been transmitted must then be assigned default values.	O

7.4.6 Firmware update

Entry in object 0x0005 “Capabilities”: “FwUpdt0”

The following describes a method for FW update of an I/O module. From the point of view of the I/O device, the process is independent of the bus system that is on a higher level than the station (e.g., Sercos, PROFINET, PROFIBUS, EtherNet/IP™) as communication takes place via the parameter channel.

Marginal conditions

- The I/O module must support the Download Write service (see basic profile, Section 6.1).
- The “upload/download protocol” property (634, 635, 636 or 637) is entered in the CommunicationProfile (object 0x000E).
- The “FwUpdt0” property is entered in Capabilities (object 0x0005).
- Object “InitFwDownload” (0x0045) and Download Write object “DeviceFw” (0xE807) are implemented.
- One instance (bootloader) must carry out the corresponding checks and receive the FW update file.
- The FW update is not permissible in the “RUN” state. The master ensures that the process data is set to invalid/set to substitute values before and during the FW update.
- The marginal conditions apply to devices with microcontroller.

Dependency info:
If object 0x0045 “InitFWDownload” is implemented, object 0xE807 “DeviceFW” must also be implemented.

Index (hex)	**Object name**	**R/W**	**Length**	**Data type** (representation)	**Meaning**	**M/O**
0045	InitFWDownload	R/W	58 bytes, maximum	Record (4 elements)	Initializing firmware download Contains information for the following FW download and sets the module to the FW update mode.	O
.01	Status	R	1 byte	UINT8 (hex)	Status information with regard to the FW download	D
		• Bit 0: ready for download This bit is set by the device as soon as it is ready to receive a new FW via object 0xE807. This bit is typically set when the device switched to the FW update task following initialization of the update using command 0x01. The bit is reset with the appearance of bit 3 “FW-Update successful” (or power-up). • Bit 1: device FW invalid This bit shows that the device contains a faulty FW/no FW. It is set when the bootloader starts up and uses the checksum to determine that the FW is missing or defective. The bit is reset with the appearance of bit 15 “FW-Update successful” (or power-up) • Bit 2: reserved • Bit 3: FW update successful This bit is set by the slave following successful FW update and is only reset by a read operation and a restart. This ensures that successful firmware update can also be controlled by various instances. Independently, the bit is always reset using command 0x01 “Init FW-Update”.				
.02	Control	R/W	1 byte	UINT8 (hex)	Commands relating to the FW download	D
		• 0x00 No action • 0x01 CMD: init FW update Using the subsequent fields, the I/O device checks whether the FW is appropriate and then returns to the FW update task, if applicable. Alternatively, a corresponding error type is returned. • 0x02 CMD: prepare restart The master completes the update using this command, no further blocks are executed. The I/O device waits for the reset/power-down signal/command, or the microcontroller is automatically reset (according to the wiring) Following completion of the action, the contents are automatically reset to 0x00.				
.03	HeaderVersion	R/W	2 bytes	UINT16 (hex)	Version of the header of the FW update file It is incremented in the case of changes to the header High byte: Incompatible change Low byte: Compatible extension	D
.04	UpdateVersion	R/W	2 bytes	UINT16 (hex)	Version of the update process Is incremented in the event of changes to the update process High byte: Incompatible change Low byte: Compatible extension	D

.05	BaseData	R/W	N bytes	Octet string (hex)	Basic data to check the preconditions for a FW update. The information is manufacturer-specific but should include: • Firmware version/versions • HW version/versions • Device type • Vendor ID • Checksum • Reserved for future expansions	D
E807	DeviceFW	UR/ DW	N bytes	Domain variable octet string (hex)	Firmware file Contains the firmware file	D

Error types

The device can reject access to the FW update objects (0x0045 and 0xE807). The following error types are defined for this purpose:

Device rejects the FW update request using object 0x0045

Error class	Error code	AddInfo	Comment
0x08	0x01	0x0022	FW update not possible in the current device status
0x08	0x01	0x0050	General
0x08	0x01	0x0051	Incorrect major header/update version
0x08	0x01	0x0052	Firmware not suitable for hardware
0x08	0x0B	0 x 0000	Device is not yet ready for a restart (busy), in the event of a control restart The master might also report a timeout (0x0F12).

Device rejects the FW update using object 0xE807

Error class	Error code	AddInfo	Comment
0x08	0x01	0x0022	Device declines write access to object 0xE807 as the FW update has not yet been started via object 0x0045.
0x08	0x01	0x00A4	Device cannot write the segment, e.g., due to problems with the flash access.
0x06	0x02	0x0080	Device cannot write the segment, e.g., due to problems with the flash access: hardware defective
0x08	0x01	0x0050	A block type written via object 0xE807 cannot be processed by the device, e.g., the length is incorrect
0x08	0x01	0x0053	FW block identical -> Skip block

In addition to the error types defined here, the remaining appropriate error types (e.g., for the Download Write protocol) can be used.

Procedure

The FW update is initiated via object 0x0045. Using the content of object 0x0045, the device checks whether the new FW is suitable and acknowledges this positively, or otherwise negatively (error type, see above).
The device then changes to FW update mode and should report the “Ready for update” status change as information to the master via object 0x0018 “DiagState”, so that the master can enter this information in its diagnostic log book.
The device then receives the firmware block via Download Write to object 0xE807. Following the complete download, the checksum (base data) calculated during download is compared.
The master then initiates the restart of the device using object 0x0045. The device acknowledges this positively if everything is OK, otherwise negatively. If everything is OK, it waits until the restart for an explicit signal for restart, or the microcontroller is automatically restarted (according to the wiring). Alternatively, a power-off – power-on is carried out.

7.4.7 Password protection

Index (hex)	Object name	R/W	Length	Data type (representation)	Meaning	M/O
001D	Password	R/W	Maximum 40 bytes	Octet string[40] (text)	Password This command is used to transmit a password (e.g., “Superuser”), which is valid for subsequent actions. Where access is attempted without a password or with an incorrect password, access to password-protected objects must be rejected with reference to the absent access rights. The password is deactivated in the event of: • Communication breakdown • Bus stop (bus reset) • No password is written (string length is “0”) • Writing an incorrect password When the object is read, 8x“*” and 0x00 “0x2A2A2A2A 2A2A2A2A0x00” are returned. There can be no access protection for this object.	O
001E	SetPassword	W	45 bytes, maximum	Record (5 elements)	Set password One or more password(s) are set for a specific index using this object. The password is only deleted using the “ResetParam” command. Then the default password is valid again. The manufacturer may have defined a default password for any object To prevent modifications to the password, this object itself can be protected. The one who protects the “SetPassword” object actually is the “Superuser”. Passwords can be preset by the manufacturer. These are not necessarily deletable.	O
.01	• SubModNo	W	1 byte	UINT8 (hex)	Submodule number “FF” as a submodule sets the password for all submodules. 0x00 is returned when the object is read.	D
.02	• Index	W	2 bytes	UINT16 (hex)	Index “FFFF” as an index sets the password for all objects. 0x0000 is returned when the object is read.	D
.03	• Subindex	W	1 byte	UINT8 (hex)	Subindex “FF” as the subindex sets the password for all subobjects. 0x00 is returned when the subobject is read.	D

.04	• Add/Replace	W	1 byte	Bit string 8 (bin)	Add/replace 0x0X = All existing passwords are replaced 0x1X = Add to existing passwords 0xX1 = Applies for read access 0xX2 = Applies for write access 0xX3 = Applies for read and write access 0x00 is returned when the object is read.	D
.05	• Password	W	40 bytes, maximum	Octet string[40] (text)	Password for object “Index” If “Replace” 0x0X is selected but no password is entered, password protection is canceled. When the object is read, 8x“*” and 0x00 “0x2A2A2A2A 2A2A2A2A0x00” are returned.	D

If an error occurs when writing/reading the objects, e.g., invalid values are transmitted, access must be rejected using the appropriate error types (see “List of permissible error types” section).

Error class: 0x08, error code: 0x01

Additional code (hex)	**Meaning**
00B1	The password cannot be replaced (deleted).
00B2	The password cannot be added (too many passwords).
00B3	The password cannot be assigned for the desired type of access.

7.4.8 Energy management (pre-implementation)

Entry in object 0x0005 “Capabilities”: “Energy0”

Entry “Energy0” in object 0x0005 Capabilities indicates that the device supports energy management.

Index (hex)	**Object name**	**R/W**	**Length**	**Data type** (representation)	**Meaning**	**M/O**
003D	WakeUpTime	R	2 bytes	UINT16 (dec)	Startup time Time until ready to operate in ms Time from switch-on of the supply voltage (if there are several supply voltages, the time when the last relevant one is switched on) until the device is ready to operate (in ms). Here, “ready to operate” means that the device responds to all communication requests (e.g., writing of parameters) of the master in the defined manner. Basic settings, booting of the communication processor, etc., are complete. The WakeUpTime does not normally include applicative times, such as selftests, calibration of the front end, or times resulting from dependencies on the control program. Legacy information: If this parameter is not available, a default value of <500 ms is assumed.	**M**
003E	EnergyMgmt	R/W	5 bytes	Record (2 elements)	Energy management The energy-saving modes that are decisive for energy management are to be specified by the manufacturer and can be freely defined in the range 0x01 ... 0x7F. Note: Energy management always refers to the direct and indirect I/O devices of the device. If it cannot be separated from the communication part, this can mean, for example, that the entire device is switched off. This object can then no longer be read back.	O

.01	• ActualMode	R/W	1 byte	UINT8 (hex)	Current energy-saving mode 0x00 - Power off 0x01 Mode01 (standby) 0x02 – 0x0F Reserved 0x10 – 0x7F Manufacturer-specific 0x80 – 0x87 Reserved 0x88 - Undefined It is left to the device to select the most effective energy-saving mode (might be the only one). In doing so, the device must observe the pause time potentially specified. 0x89 – 0xFE Reserved 0xFF - Fully operational If this object has not been created, it is assumed that the module only supports modes “0x00” and “0xFF”. The desired energy-saving mode is written. The energy-saving mode in which the device is currently operating is read.	D
.02	• Break	R/W	4 bytes	UINT32 (dec)	Pause time Time in seconds (sec) that the device should stay in energy-saving mode. Thereafter, the device automatically switches back in to mode “0xFF - Fully operational” “0x0000.0000” - Reserved “0xFFFF.FFFF” - Infinite (device does not change independently to another energy-saving mode) The desired pause time is written. The time that the device will remain in energy-saving mode is read.	D

7.5 Multilingual capacity

The “Language” object can be used to read the current language selected and select the language.
The “LanguageAvailable” domain variable indicates which languages are available. The first entry contains the default language.

Index (hex)	**Object name**	**R/W**	**Length**	**Data type** (representation)	**Meaning**	**M/O**	**Example**
0017	Language	R/W	65 bytes, maximum	Record (2 elements)	Language Object for selecting the device language The currently valid language may be accessed or changed here.	**M**	
.01	• LanguageCode	R/W	5+1 bytes	Visible string (text)	Language code Language code according to ISO 639-1 and country code according to ISO 3166-1-Alpha-2 code, separated by a “-”, if desired, e.g., “en” or “en-us” If a country code is not being used, 0x00 will be entered. The language code of the language to be selected is entered here. Following a positive response, all text is to be output in the selected language.	**M**	en
.02	• NameLanguage	R	50 bytes, maximum	Visible string (text)	Language name The text string for the currently valid language may be read here. The name changes as soon as a new “LanguageCode” is entered.	**M**	English
E804	LanguageAvailable	R		Domain variable Record	Available languages Object for device language selection - displays all available languages Language name, already in the corresponding language	O	
Seg.1 -a	• LanguageCode1 (default)	R	5+1	Visible string (text)	Language code according to ISO 639-1 and country code according to ISO 3166-1-Alpha-2 code (default), separated by a “-” if desired	D	en-gb
-b	• NameLanguage1 (default)	R	50, maximum	Visible string (text)	Text string for the 1st language available (default)	D	English

Seg.2 -a	• LanguageCode2	R	5+1	Visible string (text)	Language code according to ISO 639-1 and country code according to ISO 3166-1-Alpha-2 code of the second language available, separated by a “-” if desired	O	en
-b	• NameLanguage2	R	50, maximum	Visible string (text)	Text string for the second language available	O	Deutsch
Seg.3 -a	• LanguageCode3	R	5+1	Visible string (text)	Language code according to ISO 639-1 and country code according to ISO 3166-1-Alpha-2 code of the third language available, separated by a “-” if desired	O	fr
-b	• NameLanguage3	R	50, maximum	Visible string (text)	Text string for the third language available	O	Français
Seg.4 -a	• LanguageCode4	R	5+1	Visible string (text)	Language code according to ISO 639-1 and country code according to ISO 3166-1-Alpha-2 code of the fourth language available, separated by a “-” if desired	O	es
-b	• NameLanguage4	R	50, maximum	Visible string (text)	Text string for the fourth language available	O	Español
	...						
	...						
Seg.N -a	• LanguageCodeN	R	5+1	Visible string (text)	Language code according to ISO 639-1 and country code according to ISO 3166-1-Alpha-2 code of the Nth language available, separated by a “-” if desired	O	it
-b	• NameLanguageN	R	50, maximum	Visible string (text)	Text string for the Nth language available	O	Italiano

The “LanguageCode” subobject (subindex 1) contains the language codes according to ISO 639-1 and the country code according to ISO 3166-1-Alpha-2 code, separated by a “-” if desired. If a country code is not being used, 0x00 will be entered.
By writing this subobject, the language will be selected. This language then appears in the “NameLanguage” subobject.
In the case of write access to the entire “Language” object, the entry for the “NameLanguage” subobject is ignored because the “NameLanguage” subobject has status “read only”.

Each segment in the “LanguageAvailable” domain variable contains both the “LanguageCode” and the “NameLanguage” subobjects of the available language. Therefore, the individual read services of the “Upload Read” macro service (almost) always have different lengths, but each language entry is always transmitted in its entirety in one segment. This results in easy handling.

7.6 Modular devices - subsystems

Entry in object 0x0005 “Capabilities”: “SubMa_0”

7.6.1 Basics

In addition to compact devices, there are also an increasing number of modular devices. These are on a lower level than the fieldbus and are, therefore, referred to as subsystems with submodules from the point of view of a higher-level network. Their handling should also be taken into account in this profile. A modular device is defined as follows:

- It has exactly one communication access module (header).
- It has N modules, whereby 0 <= N < 253.
  Module numbers N = 253, N = 254 and N =255 are reserved.
- A module does not necessarily have to be active/present.
- Each module has its own independent set of objects.
  This object area does not need to be the object area described in this basic profile, but is determined by the subsystem.

This profile describes devices on a lower level than the fieldbus system. Of course, the fieldbus itself can be implemented as a modular station or even as a gateway below another fieldbus/network that may have its own rules and profiles. In this case, this is referred to as a subsystem from the point of view of the higher-level fieldbus/network.

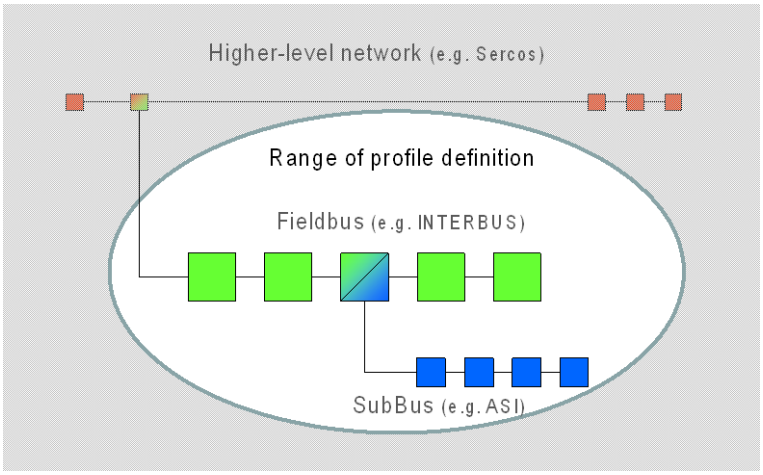


Figure: Range of definitions in the basic profile

The mechanisms and objects described in this section are to be implemented if the device is a subbus master.

7.6.2 Parameters

Usually, subbus modules also have parameters. These can be directly addressed.
The “ModuleNumber” parameter is used to address objects in the submodules of a modular device.
That actually is the module for which access was performed. In this way, it is possible to address the objects in the submodules via the

- Module number
- Index in the submodule
- Subindex in the submodule

Directly, without having to use a tunnel protocol, for example.

The behavior for compact devices does not change either. Any module number may be used here as usual. However, for reasons of consistency we recommend selecting module number “0x00”.
The header of a modular device can always be addressed using module number “0”, and the individual modules can be addressed via the corresponding module number.
Addressing of the modules begins with “1”.

As described above, a compact device must always be addressed using module number “0”, the head of a modular device always using module number “0”, and the individual submodules (N) uisng 0 < N < 253.
0xFF as the module number is not permissible on modular devices but may appear in a diagnostic message as the ID for the entire subsystem.

7.6.3 Diagnostics

Fault codes 0xA0XX are defined to be able to report system errors in the modular system (e.g., module missing) or in communication.
The standard fault codes (see “Fault codes” section) are to be used to report application errors of modules in the modular system (e.g., I/O voltage missing). In this case, detailed information (if available) can be read from object 0x0018 “DiagState” of the module, which is addressed via the module number. An fault code proprietary to the system or submodule can be stored in subobject 0x0018.09 “DiagState.AddValue”.

Dependency info:
If object 0x0035 “SubBusInfo” is implemented, all other objects in this section must also be implemented.

Index (hex)	**Object name**	**R/ W**	**Length**	**Data type** (representation)	**Meaning**	M/ O
0035	SubBusInfo	R	16 bytes	Record (5 elements)	Subbus information Basic information on the connected subbus.	O
.01	• SubBusType	R	8 bytes	Octet string[8]	Subbus designation The following are currently defined: • 0x4153495F30330000 ASI_03 • 0x43414E5F30310000 CAN_01 • 0x44414C495F303100 DALI_01 • 0x4942535F30310000 IBS_01 • 0x494F4C5F30310000 IOL_01 • 0x50425F3031000000 PB_01 The type is exactly 8 characters long. Unused characters are to be filled with 0x00.	D
.02	• MTStructFormat • MT length • PD IN length • PD OUT length	R	3 bytes	Record 1 entry	Field lengths Field lengths in objects 0x0036 “ActSubBusStructure” and 0x0041 “RefSubBusStructure” Field length of “ModuleType” in bytes (typically: 4) Field length of “PD IN length” in bytes (typically: 2) Field length of “PD OUT length” in bytes (typically: 2)	D

.03	• ActNoOfModules	R	1 byte	UINT8 (dec)	Number of connected subbus modules Number of currently connected subbus modules = Number of entries in “ActSubBusStructure”	D
.04	• LastMappedModule	R	1 byte	UINT8 (dec)	Last subbus module directly mapped Last subbus module that can be reached via the cyclic process data channel (with regard to “RefSubBusStructure”)	D
.05	• RemainingSystem	R	1 byte	UINT8 (dec)	Residual system Status information on whether a subbus residual system is currently being operated or not. • 0x00: No residual system is currently being operated. The actual configuration (“ActSubBusStructure”) matches the reference configuration (“RefSubBusStructure”). • 0x01: A residual system is currently being operated. Requirement: the option in object 0x0043.3 “Sub-BusBehaviour.Remaining” = ON is selected and one or more modules of the desired configuration (“RefSubBusStructure”) are not present in the actual configuration (“ActSubBusStructure”).	D
0036	ActSubBusStructure	R	N x 8 bytes, typically	Array of records (N x 3 elements)	Actual subsub configuration The current structure of the modular device and mapping of the module process data to the device process data is specified here. The device type and length of the process data mapping can be called here for each module under the subindex with its module number. This object is also suitable for describing compact devices. Mandatory object for modular devices.	D
.Na	• ModuleType	R	See object 0x0035.2 4 bytes, typically	UINT32 (hex)	Device type identification of the Nth module specified on the subbus. This can be an address number in the simplest case. “0”: there is no module at this position “0xFF ... FF”: reserved.	D
.Nb	• PDINLength	R	2 bytes	UINT16 (dec)	Length in bits on the IN process data channel of the Nth module	D
.Nc	• PDOUTLength	R	2 bytes	UINT16 (dec)	Length in bits on the OUT process data channel of the Nth module	D

0041	RefSubBusStructure	R/ W	N x 8 bytes, typically	Array of records (N x 3 elements)	Desired subbus configuration The prescribed structure of the modular device and mapping of the module process data to the device process data are specified here. The device type and length of the process data mapping can be called here for each module under the subindex with its module number. This object is also suitable for describing compact devices. Mandatory object for modular devices.	D
.Na	• ModuleType	R/ W	See object 0x0035. 2 4 bytes, typically	UINT32 (hex)	Device type identification of the Nth module specified on the subbus. This can be an address number in the simplest case. “0”: there is no module at this position “0xFF..FF”: reserved.	D
.Nb	• PDINLength	R/ W	2 bytes	UINT16 (dec)	Length in bits on the IN process data channel of the Nth module	D
.Nc	• PDOUTLength	R/ W	2 bytes	UINT16 (dec)	Length in bits on the OUT process data channel of the Nth module	D
0042	ModuleStatus	R	N x 1 byte	Array of bit strings[8]	Module status on the subbus The current module status is determined here, with regard to the desired configuration “RefSubBusStructure”. Bit0 = 1: Module reports an fault Bit1 = 1: Module reports a warning Bit2 = 1: Module reports information Bit7 = 0: Module cannot be reached = 1: Module can be reached	D
.01	• Module1	R	1 byte	UINT8 (hex)		
.02	• Module2	R	1 byte	UINT8 (hex)		
	• ….	R	1 byte	UINT8 (hex)		
.N	• ModuleN	R	1 byte	UINT8 (hex)		

0043	SubBusBehaviour	R	3 bytes	Record (3 elements)	Subbus behavior Definition of the behavior of the subbus system	D
.01	• AutoSetup	R/ W	1 byte	UINT8 (hex)	Automatic startup Following startup of the subbus master, the connected subbus can be started in a number of different ways: • 0x00 - OFF No configuration comparison, no parameterization of the subbus. The connected subbus is started up as far as possible. • 0x01 – ON (default) Following successful configuration comparison and parameterization, if applicable, of the subbus, the subbus is automatically started up. The outputs remain in switch-on/substitute value mode. The input data should be valid. The subbus master waits for the corresponding activities in the higher-level system to activate the outputs.	
.02	• AutoReStart	R/ W	1 byte	UINT8 (hex)	Automatic restart • 0x00 - OFF Following a bus fault, the program will wait for command 0x03 “Restart” of object SubBusControlCMD in order to start up again. • 0x01 - ON (default) As soon as a permissible configuration is recognized (see also “Remaining”), the subbus is reactivated and also parameterized, if applicable.	
.03	• Remaining	R/ W	1 byte	UINT8 (hex)	Residual system • 0x00 - OFF All submodules must be reachable in order to be started during startup or following an fault. • 0x01 - ON (default) The subbus is started up with the reachable submodules (residual system) during startup or following an fault.	

0044	SubBusControl	R/ W	1 byte	Record 1 element	Subbus control	D
.01	• SubBusControlCMD	R/ W	1 byte	UINT8 (hex)	Subbus control • 0x00 – “NoAction”: No activity • 0x01 – “Reconfigure”: Delete stored desired configuration (RefSubBusStructure), redefine actual configuration (ActSubBusStructure), and fix, if applicable • 0x02 – “SaveConfig”: Accept actual configuration as the desired configuration. • 0x03 – “Restart”: Subbus restart, restart of the subbus following a bus error if no “AutoReStart” is configured in object 0x0043. • 0x04 – “Stop”: Stop or reset of the subbus Restart using command “Restart” or “Reconfigure”	D
C000 - C07F	ProjBasProf	See object in the basic profile			Projection of basic profile to the subbus modules	D
		Generic mapping of the properties described by the objects of the basic profile should also be possible for the submodules. As the necessary data in the integrated subsystems can be located in any position, the content of the objects of the basic profile is projected (as far as possible and useful) into this index range into the address range of the subbus master. As the subbus number for these objects cannot be used, the number is mapped to the subindex. *Example:* *The serial number (if it existed) (0x0008 in the basic profile) of the third submodule could generically be reached under* *index: 0xC008.03,* *even if it can be found a second time at a submodule-specific position.* The limitation caused by the fact that the subobjects of the basic profile can no longer be addressed, is accepted here, as they can still be fully mapped due to using subindex = 0x00. If the object to be projected does not exist in the subbus module, • error type 0x0607 (No object exists under this index/subindex) • additional code 0x00A1 (Resource unavailable) is generated.				
.01 .. .FF	• Proj<Objectname>	See object in the basic profile (Subindex = 0x00)			Projected objects of the basic profile	D

In the structures above, modules that are not created may cause “gaps” when using slot-oriented systems. As access to the objects is possible both entirely and via the subindex, the data record that is empty as a result must nevertheless be implemented. The contents are then filled with “0” or set to “inactive”.

7.7 Object description

During startup and servicing, it is not only important to know the desired parameterization, but also to know the actual parameterization of the device. This requires knowledge of the existing user objects.
These objects and their meaning can be read using the two following profile objects, as they are mutually dependent.

Dependency info:
All of the mandatory objects not defined in the basic profile must be described via objects 0x0038 “ObjDescrReq” and 0x0039 “ObjDescr”.

Index (hex)	**Object name**	**R/W**	**Length**	**Data type**	**Meaning**	**M/O**
0038	ObjDescrReq	R/W	3 bytes	Record (2 elements)	Object description request Object whose description is requested	D
.01	• Index	R/W	2 bytes	UINT16 (hex)	Index	D
.02	• Subindex	R/W	1 byte	UINT8 (hex)	Subindex	D
0039	ObjDescr	R/W_D	58 bytes, maximum	Record (16 elements)	Object description Description of the object whose index was requested. **Self-incrementing** to the next valid index/subindex at the time of the next access. Details, see below	D
.01	• Index	R	2 bytes	UINT16 (hex)	Index Object number e.g., 0x6051	D
.02	• Subindex	R	1 byte	UINT8 (hex)	Subindex Structure number of an object e.g., 0x00	D
.03	• ObjectCode	R	1 byte	UINT8 (hex)	Object code As specified under “data objects” e.g., 0x07 - simple variable Evaluation in the case of complex data objects, see below	D
.04	• IndexOfType	R	1 byte	UINT8 (hex)	Data type code As specified under “data types” e.g., 0x0A – octet string	D
.05	• Length	R	1 byte	UINT8 (dec)	Length of the (sub)object in bytes e.g., 0x04	D
.06	• UnitText	R	5+1 bytes	Visible string (text)	Unit of value - 0x00 terminates e.g., U/min	D
.07	• UnitCode	R	1 byte	UINT8 (hex)	Unit variable code of the value according to table “Indices of physical variables” e.g., 0x11	D
.08	• UnitCodeExp	R	1 byte	INT8 (hex)	Unit exponent index according to table “Indices of physical variables” As a function of the “UnitCode”, the “UnitCodeExp” can either contain an exponent code or a special meaning.	D

		The special meaning must be determined from table “Indices of physical variables”. All codes with special meaning are greater than +64. These units are, for example, the day, hour, or minute, or non-SI-compatible units such as degrees Fahrenheit, e.g., 95. For SI units, the unit index specifies the unit exponent code, and thus the decimal power of the unit value. The value exponent code has a value range from -127 to 128. For example, • Unit index 0 for 10^0 • Unit index 3 for 10^3 • Unit index -3 for 10^{-3} etc. The listed unit exponent codes for SI-compatible values (unit exponent code <64) are only used as examples. The value indices for other SI-compatible prefixes (Pico, etc.) are created accordingly.				
.09	• Offset	R	2 bytes	INT16 (dec)	Offset Offset of the corresponding value. This must be added to the process data to calculate the measured value, taking the resolution into account. Example: The process data value is 12345, the offset is -15000, this results, taking the above resolution into account, in a measured value of -8.84 V	D
.0A	• RDR	R	2 bytes	UINT16 (dec)	**R**esolution**D**imension**R**ange Dimension range of the resolution $\text{Resolution} = \frac{RDR}{RNR}$ Dimension range of the resolution related to the unit in which the variable is displayed and in which the process dat item fluctuate. Is used for calculating the resolution. Example: 10 (volts) Resolution = 10 V / 3000 = 3.333 mV The default is “1” here.	D
.0B	• RNR	R	2 bytes	UINT16 (dec)	**R**esolution**N**umber**R**ange Number range of the resolution $\text{Resolution} = \frac{RDR}{RNR}$ Number range of the resolution related to the dimension range in which the process data fluctuates. Is used for calculating the resolution. Example: 3000 Resolution = 10 V / 3000 = 3.333 mV The default is “1” here.	D

.0C	• Access rights	R	1 byte	Bit string 8 (bin)	Access rights Bit 0 = ReadAccess (read rights) *1 = Reading allowed* *0 = Reading not allowed* Bit 1 = WriteAccess (write rights) *1= Writing allowed* *0 = Writing not allowed* Bit 2 = Read_WithPassword (Bit 0 = “0”) 1= *Reading allowed after setting the password* *0= Read rights are independent of the password* Bit 3 = Write_WithPassword (Bit 0 = “0”) *1= Writing allowed after setting the password* *0= Write rights are independent of the password* Possible combinations Bit [3 ... 0]: *Bit [3210]* *[0001] = Read only* *[0010] = Write only (only rarely useful)* *[0011] = (Read & write) allowed* *[0110] = Write allowed & read allowed after setting the password* *[1001] = Read allowed & write allowed after setting the password* *[1100] = (Read & write) allowed after setting the password* Bit 4 = Avoid presentation *In tools/engineering, the content of the object should* *1= Not be shown* *0 = Be shown* Bits 5-7 Reserved	D
.0D	• DisplayFormat	R	1 byte	UINT8 (hex)	Display format 0x00 - Undefined 0x01 - Binary 0x02 - Unsigned decimal 0x03 - Signed decimal 0x04 - Hexadeximal 0x05 - Text 0x06 - Float 0x07 - Time 0x08 - Date 0x09 - 0xFF Reserved	D
.0E	• Min	R	“Length” byte	According to data type	Lower limit Lower permissible limit of a value. The length is based on 0x0039.5 “ObjDescr.Length”. e.g., -500 For all string variables, there is no minimum value. In this case, the length is 0.	D
.0F	• max	R	“Length” byte	According to data type	Upper limit Upper permissible limit of a value. The length is based on 0x0039.5 “ObjDescr.Length”. e.g., 500 For all string variables, there is no maximum value. In this case, the length is 0.	D

.10	• Symbol	R	M+1 byte	Visible string (text)	Name/designation of the object The length of the name can be filled up to the maximum PDU size. For example, set rotation speed The object is terminated with 0x00. Note: On this basis, the following symbol structure is specified to enable the naming of the individual bits in a bit string. <Designation of the bit string>; <Name of Bit0>; <Name of Bit1>; .. <Name of BitN> A semicolon ";" is used as the separator. 0x00 follows the name of the last bit. It is possible to skip the designation of an unused bit with ";;" (two semicolons). Additional bits accordingly.	D
E805	ObjDescrLong	R	N segments	Domain variable record	Object description (long form) Description of all available objects (index/subindex) One description is transmitted per segment. Strings are filled with 0x00.	O
Seg.1	ObjDescr1	R	57+1 bytes	Record	See above	D
…	…	…	…	...	...	
Seg.N	ObjDescr1	R	57+1 bytes	Record	See above	D

Example of the content of object 0x0039:
Index:0x0039; subindex: 0; data: [1]6051 [2]00 [3]07 [4]0A [5]04 [6]552F6D696E00 [7]11 [8]00 [9]C568 [10]000A [11]0BB8 [12]0E [13]05 [14]FFFFFF0C [15]000001F4 [16]536F6C6C647265687A61686C00

Object 0x0039 is auto-incrementing. This means that, after a read access, the description of the next available index/subindex is provided for the next read access. When the last subindex of an object has been read, the first subindex of the next object is returned during the following read access. The index is thus incremented and the subindex starts again from 0.

The process starts again from the beginning when the last subindex of the last index has been reached.

Example:
0x0200.2, 0x0200.3, 0x0221.0, 0x0240.0, 0x0240.1, 0x0240.2, 0x0240.3, 0x0240.4, …
0x0200.0, 0x0200.1, 0x0200.2, 0x0200.3, …

The auto-increment function of object 0x0039 can easily be used by an application to read all available indices/subindices and thereby obtain a complete map of all available objects in a device. The application reads object 0x0039 for this purpose until the index/subindex is repeated.

Empty objects or unavailable indices and subindices are skipped.

If an unavailable index/subindex is requested by object 0x0038 "ObjDescrReq", a negative confirmation with error type, error class 0x08 (application), error code 0x01 (data item uses

an invalid value), additional code 0x0030 (parameter out of range) will be issued. The description of the next available index/subindex will then be provided in object 0x0039 “ObjDescr”.

For basic profile objects, a description does not have to be stored as it (including subindices) is known across the system as a result of the basic profile. Only object 0x0039 contains the index. The remaining entries are missing (length of user data item: 2 bytes).

Index (hex)	Object name	R/W	Length	Data type	Meaning	M/O
0039	ObjDescr	R/W_D	2 bytes	Record (1 element)	Object description Description of the basic profile object	O
.01	• Index	R	2 bytes	UINT16 (hex)	Object index e.g., 0x0007	D

In this way, availability of optional objects (including all their subindices) is displayed. A basic profile object must, therefore, not be skipped.

Example:
A device has objects: 0x6051, 0x6055, 0x6056, 0x6059. Except for object 0x6056, they are all simple objects without subobjects. Subindices 01, 02, 03, 07 and 08 exist for object 0x6056.
If the description is read via object 0x0039, the auto-increment function generates the following:

Write.req: index:0x0038; subindex: 0; data: 0x6000 00 (object 0x6000.0 is requested)
Read.cnf no.1: index:0x0039; subindex: 0; data: [1]6051 [2]00 [3]07 [4]0A [5]04 … [16]536F6C6C00
Read.cnf no.2: index:0x0039; subindex: 0; data: [1]6055 [2]00 [3]07 [4]0A [5]04 … [16]536F6C00
Read.cnf no.3: index:0x0039; subindex: 0; data: [1]6056 [2]00 [3]09
Read.cnf no.4: index:0x0039; subindex: 0; data: [1]6056 [2]01 [3]07 [4]03 [5]02 … [16]506F7765723100
Read.cnf no.5: index:0x0039; subindex: 0; data: [1]6056 [2]02 [3]07 [4]03 [5]02 … [16]506F7765723200
Read.cnf no.6: index:0x0039; subindex: 0; data: [1]6056 [2]03 [3]07 [4]03 [5]02 … [16]506F7765723300
Read.cnf no.7: index:0x0039; subindex: 0; data: [1]6056 [2]07 [3]07 [4]07 [5]04 … [16]4170706C657300
Read.cnf no.8: index:0x0039; subindex: 0; data: [1]6056 [2]08 [3]07 [4]06 [5]02 … [16]506561727300
Read.cnf no.9: index:0x0039; subindex: 0; data: [1]6059 [2]00 [3]07 [4]0F [5]01 … [16]4d794269747300
Read.cnf no.10: index:0x0039; subindex: 0; data: [1]0001
Read.cnf no.11: index:0x0039; subindex: 0; data: [1]0004
…
Read.cnf no.N: index:0x0039; subindex: 0; data: [1]6051 [2]00 [3]07 [4]0A [5]04 … [16]536F6C6C00

Exception
Process data objects 0x0025 and 0x0026 are an exception since their availability and function are process-specific, however, their structures are manufacturer-specific and may only be determined in this way. As such, they must be described completely. The process data ranges that are marked with “NC” in objects 0x003B “PDIN_Descr” or 0x003C “PDOUT_Descr” must be described under the corresponding subindex. “Unused” is to be used as the 0x0039.10 ObjDescr.Symbol.
If a process data direction is not being used but object 0x0025 or 0x0026 exists for technical reasons, this does not need to be further taken into account. The index is skipped.

If object 0x0039 is not available, process data objects 0x0025 and 0x0026 are specified by default as bit strings of corresponding length.

All other objects are to be described.

R/W_D
Under certain conditions, the manufacturer may allow for the description of an object to be specified via this mechanism, or for the description to be modified. If the manufacturer supports this function, object 0x0039 can also be enabled for writing. However, in such a case, the object must be password-protected.

Evaluation of complex data objects

For complex data objects, evaluation of object 0x0039 “ObjDescr” ends after subindex 0x0039.3 “ObjDescr.ObjectCode”. Any other 0x0039.4 indices and the following indices are not important for the data objects in the following table and, therefore, cannot be evaluated.

Object code	Object
0x02	Domain variable
0x08	Array variable
0x09	Record variable
0x0a	Variable list Static and dynamic

The description of the entire record/array (subindex 0) is not provided.
In the case of record and array objects, only the individual elements are described (via subindex).

Table: “Indices of physical variables”

Physical variable	Variable index (UnitCode)	Unit	Unit exponent index (UnitCodeExp) (incomplete)
	0	Without dimension	0
Length (l)	1	Kilometer (km)	3
Length (l)	1	Meter (m)	0
Length (l)	1	Millimeter (mm)	-3
Area (A)	2	Square meter (m²)	0
Volume (V)	3	Cubic meter (m³)	0
		Liter (l)	-3
Time (t)	4	Second (s)	0
		Minute (min)	70
		Hour (h)	74
		Day (d)	77
		Half waves	78
		Millisecond (ms)	-3
		Microsecond (µs)	-6
Force (F)	6	Newton (N)	0
Reactive power (Q)	8	Var (var)	0
(Real) power (P)	9	Watt (W)	0
Apparent power (S)	10	Voltampere (VA)	0
RPM (n)	11	Rounds/second (U/s)	0
		Rounds/minute (U/min)	73
		Rounds/hour (U/h)	74
Angle ($\alpha, \beta, \gamma, ..$)	12	Radian (rad)	0
		Second (“)	75
		Minute (')	76
		Degree (°)	77
		Gradian (gon)	78
Speed (v)	13	Meters/second (m/s)	0
		Millimeters/second (mm/s)	-3
		Millimeters/minute (mm/min)	79
		Meters/minute (m/min)	80
		Kilometers/minute (km/min)	81
		Millimeters/hour (mm/h)	82
		Meters/hour (m/h)	83
		Kilometers/hour (km/h)	84
Torque (M)	16	Newtonmeter (Nm)	0
Temperature (T)	17	Kelvin (K)	0
		Degrees Celsius (°C)	94
		Degrees Fahrenheit (°F)	95
Electrical voltage (U)	21	Volt (V)	0
Electric current (I)	22	Ampere (A)	0
Electrical resistance (R)	23	Ohm (Ω)	0
Ratio	24	Percent (%)	0
Frequency (f)	28	Hertz (Hz)	0
Steps	32	Steps	0
Encoder resolution	33	Steps/revolution	0
Flow (Q)	34	Cubic meters/second (m³/s)	0
		Liters/second (l/s)	-3
		Milliliters/second (ml/s)	-6
		Cubic meters/minute (m³/min)	85
		Liters/minute (l/min)	86
		Milliliters/minute (ml/min)	87
Acceleration (a)	35	Meters/second² (m/s²)	0
Pressure (P)	40	Pascal (Pa)	0
		Bar (bar)	5
		Technical atmosphere (at)	85
Efficiency (η)	41	Without dimension	0
(Real) power factor ($\cos\varphi$)	42	Without dimension	0
Work/energy (W/E)	44	Joule (J)	0

The unit exponent index corresponds to the exponent (to the basis of 10) as standard:

E.g.,
UnitCodeExp 3 for 10^3 ⇨ k (kilo)
UnitCodeExp 0 for 10^0 ⇨ without
UnitCodeExp -3 for 10^{-3} ⇨ m (milli)
UnitCodeExp -6 for 10^{-6} ⇨ µ (micro)
UnitCodeExp -9 for 10^{-9} ⇨ n (nano)

7.8 Onboard device description

Each device is represented in the engineering system of the target system by its digital equivalent - the device description file, There is, however, not only one device description file, as each system has its own defined format.
To enable online parameterization using manufacturer tools without a system-specific engineering system, a reduced and compressed onboard device description file is stored on the device based on its resources. A web browser, for example, can perform an adequate representation of the parameters using this file.

Index (hex)	**Object name**	**R/W**	**Length**	**Data type** (representation)	**Meaning**	**M/O**
E808	OnBoardDeviceDescFile	UR	N bytes	Domain variable octet string (hex)	Onboard device description file (compressed) It contains all the information needed for a tool, such as a web browser, to parameterize the device.	O

8 Handling of individual parameters by the master

8.1 0x002D - ResetParam

The master may not regularly write to object 0x002D “ResetParam” during startup. Similarly, this object must only be included in the device description file.
Triggering the “Reset Parameter” action writes the default values to the module flash. Writing to the flash too frequently destroys the flash.

The master can only systematically trigger the “Reset Parameter” action, if it recognizes a new module that might potentially have been located in a different structure.
However, the action can by suppressed by the application, if a specific parameterization that should remain unchanged was written to a device outside the system.

Simple older devices may potentially not support this object. It must then be assumed that all parameters for this device are written.

8.2 Handling the error types

In terms of availability, the master must generally handle the error types as tolerantly as possible.
This means that, under certain circumstances, the following error types can be ignored.

- 0x06, 0x05 “Attributes do not match”
- 0x06, 0x07 “Object not available”
- 0x08, 0x01 “Value not supported”

This is required to increase availability of the system, and is implemented in the majority of cases for which an action should generally be triggered.
The default value is accepted as the value in this case, if it is useful for the desired action, or a default behavior of the device is assumed.

A finite number of repetitions (3, minimum) is useful for the following error types.

- 0x05, 0x01 “Object State Conflict”
- 0x05, 0x03 “Object Constrain Conflict”
- 0x06, 0x0A “Data not yet available”

This should prevent the applications from being unavailable for “technical reasons”. The corresponding information or an entry should nevertheless be entered in the diagnostic log book.

8.3 Extensions – access via subindex

The master must assume that individual objects will be extended in future. These extensions can/will, however, only be achieved using “attachments”.

Addressing via additional subindices is possible without any problems, as they are independent (sub)objects.

When **reading** objects via subindix “0” (the entire object is read), more data can be received than expected. All the elements known to date still have the same meaning. New, additional elements cannot be evaluated because they are not known. This should, however, not affect the function of the known elements and of the device.

When **writing** objects via subindex “0” (the entire object is written), a slave will reject too much data. The master must then process the parameters for the older version. The slave can accept less data, if possible. This allows a newer device to be compatible and operated in an old system.

The corresponding information or an entry should nevertheless be entered in the diagnostic log book.

9 Appendix A

9.1 Definition of terms

Substitute values
There can be substitute values for inputs as well as for outputs.
For outputs, they are values that are sent to the I/O devices instead of the process data in case the process data is no longer valid. This is the case following a bus reset or a process data timeout, for example.
For inputs, these are values that are transmitted to the process data instead of the application data if the application is no longer able to deliver valid data. This is the case following a peripheral voltage failure or a microcontroller failure, for example.

Error type
The error type is the error message that is returned when a service could not be executed (negative confirmation).
The error type service parameter consists of the following parameters:

Error class	1 byte, UINT8, hexadecimal notation
Error code	1 byte, UINT8, hexadecimal notation
Additional code	2 bytes, UINT16, hexadecimal notation

Firmware
Firmware is the generic term for all components of a device that can be changed externally (without opening the device). These changes are mostly known as updates.
Components that can usually be changed include:

- Firmware communication processor
- Firmware application processor
- Bootloader
- FPGA image
- etc.

All components are regarded to belong together. One version is allocated to them all together.

Function group
A function group describes the function of the device. The function is defined by means of a unit that is controlled and parameterized by inputs. In addition, internal signals and parameters can influence the function. The output of a function group can be connected to the inputs of other functions or made accessible via the bus.

A function group can include:

- Inputs and outputs
- Internal signals and variables
- Actions
- External signals
- Parameters (services, process data)
- Communication objects

Device parameters
Device parameters are parameters that represent identification, diagnostics and the function of a device externally via communication. The default values for all basic device parameters - those that should be the same on all devices - are defined in this profile.

Device profile
The device profile specifies the application functions, which are visible via communication. The application functions are mapped to the communication by defining the following:

- The communication profile
- The interactions between application functions that are executed via the communication system
- The communication services used and the communication objects that are manipulated using them

Mapping makes the behavior of the application visible. The application profile definitions ensure interoperability in an area of application, provided that the devices properties used support this.
In addition, the profile defines device properties, which are important for the user. A distinction is made between mandatory, optional and manufacturer-specific functions and parameters. If the user uses only mandatory functions or parameters, devices can be replaced, provided this is supported by the device properties and settings used. In terms of communication, the devices can always be replaced by others with the same parameters, regardless of their function.

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

Communication interface
The communication interface consists of a process data channel and a parameter channel.

Communication profile
The communication profile restricts or classifies the degree of freedom included in the specification of the transmission medium, according to the application or device group. The communication profile defines communication services and parameters, which are marked as optional in the specification. All optional functions and parameters that are not specified in the communication profile remain optional. Mandatory services and parameters are binding even if they are not specified in the profile. Value ranges for attributes and parameters are also specified in the profile.

Communication reference
The communication reference is used to address the communication partner. Every communication relationship between two devices is configured regardless of when it is used. The configuration is stored in every bus device in a communication relationship list (CRL). An application process identifies the communication relationship via a local communication reference.

Parameter channel
The parameter channel is a communication channel through which all communication objects can be accessed. Parameter channel services provide acknowledged access to device parameters, i.e., access to a device parameter is confirmed by the device.

PDU
Protocol Data Unit. Data unit to which all information of a protocol on usage and administration is transmitted.

PD PDU
Process data PDU. Data unit to which the process data of a protocol is transmitted.

Process data description
The manufacturer describes the structure and meaning of the process data in the process data description.

Process data channel
The process data channel is used for the fast transmission of process data. Data is transmitted in an unacknowledged way at regular intervals (equidistant) via the process data channel. Process data can be read and written.
The direction of the process data is regarded as from the bus, i.e.,

- OUT process data is data that is transmitted from the control system to the device. The device reads this data from the process data channel and transmits it to the process, depending on the function.
- IN process data is data that is transmitted from the device to the control system. The device writes this data to the process data channel and thereby transmits it to the control system.

Terminal
A terminal is the physical end point of a signal to a device. It is usually an electrical terminal connection to which a signal wire is connected.

State machine
A state machine is a defined procedure with branches and repetitions that is schematically displayed. In this profile, some functions are described using a state machine. A state represents a specific internal and external behavior. It can only be exited by defined events. These events are assigned corresponding state transitions. Actions can be executed in a transition. This modifies the state behavior. When transition is complete, the current state changes to the next state.

9.2 Symbols and abbreviations

Abbreviations

ALI	Application layer interface
con	Confirmation primitive
D	Dependent (Depending on other objects/specific properties. Must be implemented if dependency exists.)
ind	Indication primitive
CRL	Communication relationship list
CRL header	Header of the communication relationship list
CR	Communication reference
M	Mandatory (obligatory, must always be implemented)
MAP	Manufacturing automation protocol
O	Optional (can but most not be implemented)
OD	Object dictionary
PCP	Peripherals communication protocol
PCh	Parameter channel
PD	Process data
PMS	Peripherals message specification
R	Read only, this object can only be read
R/W	Read/write, this object can be read and written
req	Request primitive
res	Response primitive
S	Selection of one or the other parameter
W	Write only, this object can only be written
X	This object should no longer be used in future

10 Appendix B

10.1 Translation table for the object names

Index	Subindex	Objektname (Basisprofil)	DE	EN-US	ES	FR	IT	ZH-CN
0001		VendorName	Herstellername	Manufacturer name	Nombre del fabricante	Nom fabricant	Nome produttore	制造商名称
0002		VendorID	Herstellerkennung	Manufacturer ID	Identificación del fabricante	Identification fabricant	Identificazione produttore	制造商ID
0003		VendorText	Herstellertext	Manufacturer text	Texto del fabricante	Texte fabricant	Testo produttore	制造商文本
0004		DeviceFamily	Gerätefamilie	Device range	Familia de dispositivos	Gamme d'appareils	Gruppo dispositivi	设备范围
0005		Capabilities	Geräteeigenschaften	Device properties	Propiedades de los dispositivos	Propriétés de l'appareil	Caratteristiche del dispositivo	设备属性
0006		ProductFamily	Produktfamilie	Product range	Línea de productos	Gamme de produits	Famiglia di prodotti	产品类别
0007		ProductName	Produktname	Product name	Nombre del producto	Nom de produit	Nome del prodotto	产品名称
0008		SerialNo	Seriennummer	Serial number	Número de serie	Numéro de série	Numero serie	序列号
0009		ProductText	Produkttext	Product text	Texto del producto	Texte produit	Testo del prodotto	产品文本
000A		OrderNumber	Artikel-Nr.	Order no.	Código de artículo	Référence	Cod. art.	订货号
000B		HardwareVersion	Hardware-Version	Hardware version	Versión de hardware	Version matériel	Versione hardware	硬件版本
	.01	BuildDate	Herstelldatum	Manufacturing date	Fecha de fabricación	Date de fabrication	Data di produzione	制造日期
	.02	Version	Versionskennung	Version ID	Identificación de la versión	Identifiant de version	Riconoscimento versione	版本ID
000C		FirmwareVersion	Firmware-Version	Firmware version	Versión de firmware	Version firmware	Versione firmware	固件版本
	.01	BuildDate	Herstelldatum	Manufacturing date	Fecha de fabricación	Date de fabrication	Data di produzione	制造日期
	.02	Version	Versionskennung	Version ID	Identificación de la versión	Identifiant de version	Riconoscimento versione	版本ID
000D		PChVersion	Parameterkanalversion	Parameter channel version	Versión de canal de parámetros	Version canal de paramètres	Versione canale parametro	参数通道版本
	.01	BuildDate	Herstelldatum	Manufacturing date	Fecha de fabricación	Date de fabrication	Data di produzione	制造日期
	.02	Version	Versionskennung	Version ID	Identificación de la versión	Identifiant de version	Riconoscimento versione	版本ID
000E		CommProfile	Kommunikationsprofil	Communication profile	Perfil de comunicación	Profil de communication	Protocollo di comunicazione	通讯配置文件

Index	Subindex	Objektname (Basisprofil)	DE	EN-US	ES	FR	IT	ZH-CN
000F		DeviceProfile	Geräteprofil	Device profile	Perfil del dispositivo	Profil d'appareil	Profilo del dispositivo	设备配置文件
0010		Reserved	Reserviert	Reserved	Reservado	Réservé	Riservato	保留
0011		ProfileVersion	Profilversion	Profile version	Versión de perfil	Version profil	Versione profilo	配置文件版本
	.01	BuildDate	Herstelldatum	Manufacturing date	Fecha de fabricación	Date de fabrication	Data di produzione	制造日期
	.02	Version	Versionskennung	Version ID	Identificación de la versión	Identifiant de version	Riconoscimento versione	版本ID
0012		VendorURL	Hersteller-URL	Manufacturer URL	URL del fabricante	URL fabricant	URL produttore	制造商URL
0013		OnBoardDeviceDescFileName	Name der OnBoard Gerätebeschreibungsdatei	Name of the onboard device description file	Nombre del archivo de descripción del dispositivo Onboard	Nom du fichier de description de l'appareil embarqué	Nome del file di descrizione del dispositivo OnBoard	机载设备描述文件名称
0014		Location	Einbauort	Installation location	Lugar de montaje	Emplacement de montage	Luogo di installazione	安装位置
0015		EquipmentIdent	Betriebsmittelkennzeichen	Equipment ID	Identificador de equipo	Identification équipement	Caratteristiche apparecchiatura	设备ID
0016		ApplDeviceAddr	Applikationsspezifische Geräteadresse	Application-specific device address	Dirección de dispositivo específica de la aplicación	Adresse appareil spécifique à l'application	Indirizzo dispositivo specifico per applicazione	应用特定的设备地址
0017		Language	Sprache	Language	Idioma	Langue	Lingua	语言
	.01	LanguageCode	Sprachencode	Language code	Código de idioma	Code de langue	Codice lingua	语言代码
	.02	NameLanguage	Sprachenname	Language name	Nombre de idioma	Nom de la langue	Nome lingua	语言名称
0018		DiagState	Diagnosezustand	Diagnostic state	Estado de diagnóstico	Etat de diagnostic	Stato diagnosi	诊断状态
	.01	Lfd.Nr.	Laufende Nummer	Consecutive number	Número de orden	Numéro d'ordre	Numero progressivo	连续号
	.02	Priority	Priorität	Priority	Prioridad	Priorité	Priorità	优先级
	.03	Channel	Kanal	Channel	Canal	Canal	Canale	通道
	.04	Code	Störungscode	Fault code	Código de fallo	Code de défaut	Codice di guasto	错误代码
	.05	MoreFollows	Zusatzinformationen	Additional information	Información adicional	Informations complémentaires	Informazioni supplementari	更多信息
	.06	Reserved	Reserviert	Reserved	Reservado	Réservé	Riservato	保留
	.07	SubModNo	Sub-Modul-Nummer	Submodule number	Número de submódulo	Numéro sous-module	Numero di sottomodulo	子模块号
	.08	FunctionGroup	Funktionsgruppe	Function group	Grupos funcionales	Groupe fonctionnel	Gruppo funzionale	功能组
	.09	AddValue	Zusatzinformation	Additional information	Información adicional	Information complémentaire	Informazione supplementare	更多信息
	.0A	TextLength	Textlänge	Text length	Longitud del texto	Longueur de texte	Lunghezza del testo	文本长度
	.0B	Text	Diagnosetext	Diagnostic text	Texto de diagnóstico	Texte de diagnostic	Testo di diagnosi	诊断文本

Index	Subindex	Objektname (Basisprofil)	DE	EN-US	ES	FR	IT	ZH-CN
0019		ResetDiag	Diagnosemeldungen quittieren	Acknowledge diagnostic message	Acusar recibo de mensajes de diagnóstico	Acquitter messages de diagnostic	Conferma messaggi di diagnosi	确认诊断信息
001A		GetError ClassRepMethod	Meldemethode für Störung	Fault reporting method	Método de mensaje para el fallo	Méthode de signalisation des défauts	Metodo di segnalazione in caso di guasto	错误报告方法
001B		TestMode	Testbetrieb	Test mode	Modo de prueba	Mode d'essai	Funzionamento di prova	测试模式
001C		ControlTrace	Steuerung der Protokollierung	Logging control	Sistema de control del registro	Commande de l'établissement de protocoles	Controllo della registrazione	记录控制
001D		Password	Passwort	Password	Contraseña	Mot de passe	Password	密码
001E		SetPassword	Passwort setzen	Set password	Establecer contraseña	Définir mot de passe	Imposta password	设置密码
	.01	SubModNo	Sub-Modul-Nummer	Submodule number	Número de submódulo	Numéro sous-module	Numero di sottomodulo	子模块号
	.02	Index	Index	Index	Índice	Index	Indice	索引
	.03	Subindex	Subindex	Subindex	Subíndice	Sous-index	Indice subordinato	子索引
	.04	Add/Replace	Ergänzen/Ersetzen	Add/replace	Completar/reemplazar	Compléter/remplacer	Completa/Sostituisci	添加/替换
	.05	Passwort	Passwort	Password	Contraseña	Mot de passe	Password	密码
001F		PDTimeout	Prozessdaten-Überwachungszeit	Process data monitoring time	Tiempo de monitorización de datos de proceso	Temps de surveillance données de process	Tempo di monitoraggio dei dati di processo	过程数据监控时间
0020		PDTimeoutCode	Prozessdaten-Überwachungscode	Process data monitoring code	Código de monitorización de datos de proceso	Temps de surveillance données de process	Codice di monitoraggio dei dati di processo	过程数据监控代码
0021		PChTimeout	Parameterkanal-Überwachungszeit	Parameter channel monitoring time	Tiempo de monitorización de canal de parámetros	Temps de surveillance canal de paramètres	Tempo di monitoraggio del canale parametri	参数通道监控时间
0022		PChTimeoutCode	Parameterkanal-Überwachungscode	Parameter channel monitoring code	Código de monitorización de canal de parámetros	Code de surveillance canal de paramètres	Codice di monitoraggio del canale parametri	参数通道监控代码
0023		AbortCode	Verbindungsabbruchcode	Connection abort code	Código de interrupción de conexión	Code de rupture de connexion	Codice di interruzione collegamento	连接中断代码
0024		ResetCode	Ersatzwertverhalten bei Bus-Reset (PDOUT)	Substitute value behavior during bus reset (PDOUT)	Comportamiento de valor sustitutivo en reset de bus (PDOUT)	Comportement des valeurs de substitution avec RAZ de bus (PDOUT)	Andamento del valore sostitutivo in caso di reset del bus (PDOUT)	总线复位期间的替换值特性 (PDOUT)
0025		PDIN	Eingangsprozessdaten	IN process data	Datos de proceso de entrada	Données d'entrée de process	Dati di processo in ingresso	IN过程数据

Index	Subindex	Objektname (Basisprofil)	DE	EN-US	ES	FR	IT	ZH-CN
	.01 … .0N	Part N	N-ter Teil des Prozessdatums	Nth part of process data item	Parte nº n de datos de proceso	Nième partie de la donnée de process	Parte Nesima della data di processo	过程数据项的第N部分
0026		PDOUT	Ausgangsprozessdaten	OUT process data	Datos de proceso de salida	Données de sortie de process	Dati di processo in uscita	OUT过程数据
	.01 … .0N	Part N	N-ter Teil des Prozessdatums	Nth part of process data item	Parte nº n de datos de proceso	Nième partie de la donnée de process	Parte Nesima della data di processo	过程数据项的第N部分
0027		GetExRight	Exklusive Schreibrechte anfordern	Request exclusive write access	Solicitar permisos de escritura exclusivos	Demander droits exclusifs en écriture	Richiedi diritti di scrittura esclusivi	请求排他写入权限
0028		ChangePDSet	Prozessdatenzuordnungen einstellen	Set process data assignment	Configurar asignaciones de datos de proceso	Régler affectations de données de process	Imposta assegnazioni dei dati di processo	设置过程数据分配
0029		ParamSetWriteControl	Steuerung der Blockparametrierung	Block parameterization control	Sistema de control de parametrización de bloque	Commande du paramétrage par blocs	Controllo della parametrizzazione di blocco	块参数设置控制
002A		ConflictDictionary	Konfliktverzeichnis	Conflict dictionary	Registro de conflictos	Répertoire de conflits	Cartella conflitti	冲突对象
	.01	ConfGrNo	Konfliktguppennummer	Number of conflict group	Número de grupo de conflictos	Numéro de groupe de conflits	Numero di gruppo conflitti	冲突组数目
	.02	Subslot_1	Subslot des ersten abhängigen Parameters	Subslot of the first dependent parameter	Subslot del primer parámetro dependiente	Subslot du premier paramètre dépendant	Slot subordinato del primo parametro dipendente	第一个相关参数的子槽位
	.03	Index_1	Index des ersten abhängigen Parameters	Index of the first dependent parameter	Índice del primer parámetro dependiente	Index du premier paramètre dépendant	Indice del primo parametro dipendente	第一个相关参数的索引
	.04	Subindex_1	Subindex des ersten abhängigen Parameters	Subindex of the first dependent parameter	Subíndice del primer parámetro dependiente	Sous-index du premier paramètre dépendant	Sottoindice del primo parametro dipendente	第一个相关参数的子索引
	.05	Element_1	Element im Record des ersten abhängigen Parameters	Element in the record of the first dependent parameter	Elemento en registro del primer parámetro dependiente	Elément de l'enregistrement du premier paramètre dépendant	Elemento nel record del primo parametro dipendente	第一个相关参数记录中的元素
	.06	Add.Code_1	Zusatzinformation zum ersten abhängigen Parameters	Additional information about the first dependent parameter	Información adicional sobre el primer parámetro dependiente	Informations complémentaires relatives au premier paramètre dépendant	Informazione supplementare sul primo parametro dipendente	有关第一个相关参数的附加信息
	…							
002B		ParamSet	Parametersatzkennung	Parameter record identification	Identificador de juego de parámetros	Identifiant du jeu de paramètres	Codice record parametri	参数记录标识
002C		ParameterMoment	Zeitpunkt der letzten Änderung der Parametrierung	Time of last parameterization modification	Momento de la última modificación de la parametrización	Moment de la dernière modification du paramétrage	Ora dell'ultima modifica della parametrizzazione	上次参数化修改的时间

Index	Subindex	Objektname (Basisprofil)	DE	EN-US	ES	FR	IT	ZH-CN
	.01	Date	Datum	Date	Fecha	Date	Data	日期
	.02	Time	Uhrzeit	Time	Hora	Heure	Ora	时间
002D		ResetParam	Parametrierung zurücksetzen	Reset parameterization	Restablecer la parametrización	Réinitialiser le paramétrage	Resetta parametrizzazione	复位参数设置
002E		ParamHash	Hash-Wert	Hash value	Valor hash	Valeur Hash	Valore Hash	哈希值
002F		PDOUT_Subst	Ersatzwert Ausgangsprozessdaten	Substitute value for OUT process data	Valor sustitutivo datos de proceso de salida	Valeur de substitution données de sortie de process	Valore sostitutivo dei dati di processo in uscita	OUT过程数据的替代值
0030		PF_Code	Ersatzwertverhalten bei Peripheriefehler (PDIN)	Substitute value behavior during peripheral fault (PDIN)	Comportamiento de valor sustitutivo en caso de error en periferia (PDIN)	Comportement de valeur de substitution en cas d'erreur périphérique (PDIN)	Andamento sostitutivo in caso di errori periferici (PDIN)	外围设备故障期间的替换值特性 (PDIN)
0031		PDIN_Subst	Ersatzwert Eingangsprozessdaten	Substitute value for IN process data	Valor sustitutivo datos de proceso de entrada	Valeur de substitution données d'entrée de process	Valore sostitutivo dati di processo in ingresso	IN过程数据的替代值
0032		FieldBus_ID	Feldbusidentifikation / reserviert	Fieldbus identification/reserved	Identificación de bus de campo / reservada	Identification bus de terrain / réservé	Identificazione bus di campo / riservato	现场总线标识/保留
	.01	ID-Code	ID-Code	ID code	Código de ID	Code ID	Codice ID	识别码
	.02	PDLength	Prozessdatenlänge	Process data length	Longitud de datos de proceso	Longueur données de process	Lunghezza dei dati di processo	过程数据长度
0035		SubBusInfo	Sub-Bus-Information	Sub-bus information	Información de subbus	Informations sous-bus	Informazione bus subordinato	子总线信息
	.01	SubBusType	Bezeichnung des Sub-Busses	Sub-bus designation	Denominación del subbus	Désignation du sous-bus	Denominazione del bus subordinato	子总线名称
	.02	MTStructFormat	Feldlängen	Field lengths	Longitudes de campo	Longueurs de champs	Lunghezze di campo	字段长度
	.03	ActNoOfModules	Anzahl angeschlossener Sub-Bus-Module	Number of connected sub-bus modules	Número de módulos subbus conectados	Nombre de modules sous-bus raccordés	Numero di moduli bus subordinato collegati	连接的子总线模块的数量
	.04	LastMappedModule	Letztes direkt abgebildetes Sub-Bus-Modul	Last sub-bus module directly mapped	Último módulo subbus representado directamente	Dernier module sous-bus directement représenté	Ultimo modulo bus subordinato raffigurato direttamente	最后一个子总线模块已直接映射
	.05	RemainingSystem	Restsystem	Residual system	Sistema residual	Système résiduel	Sistema residuo	剩余系统
0036		ActSubBusStructure	Sub-Bus-Istkonfiguration	Actual sub-bus configuration	Configuración actual del subbus	Configuration réelle sous-bus	Configurazione effettiva bus subordinato	实际的子总线组态
0037		DeviceType	Gerätetyp	Device type	Tipo de dispositivo	Type d'appareil	Tipo di dispositivo	设备类型
0038		ObjDescrReq	Anfrage Objektbeschreibung	Request object description	Solicitud de descripción de	Demande description d'objet	Richiesta descrizione oggetto	请求对象描述

Index	Subindex	Objektname (Basisprofil)	DE	EN-US	ES	FR	IT	ZH-CN
					objeto			
	.01	Index	Index	Index	Índice	Index	Indice	索引
	.02	Subindex	Subindex	Subindex	Subíndice	Sous-index	Indice subordinato	子索引
0039		ObjDescr	Objektbeschreibung	Object description	Descripción de objeto	Description d'objet	Descrizione oggetto	对象描述
	.01	Index	Index	Index	Índice	Index	Indice	索引
	.02	Subindex	Subindex	Subindex	Subíndice	Sous-index	Indice subordinato	子索引
	.03	ObjectCode	Objektcode	Object code	Código de objeto	Code d'objet	Codice oggetto	对象代码
	.04	IndexOfType	Datentypindex	Data type index	Índice de tipo de datos	Index type de données	Indice tipo dati	**数据**类型索引
	.05	Length	Länge des (Sub-)Objekts	Length of (sub) object	Longitud del (sub)objeto	Longueur du (sous) objet	Lunghezza del (sotto)oggetto	（**子**）对象的长度
	.06	UnitText	Einheit des Werts	Unit of value	Unidad del valor	Unité de la valeur	Unità del valore	值的单位
	.07	UnitCode	Einheiten-Größencode	Unit size code	Código de tamaño de unidades	Unités code de taille	Codice di grandezza unità	单位尺寸代码
	.08	UnitCodeExp	Einheitenindex	Unit index	Índice de unidades	Index d'unités	Indice unità	单位索引
	.09	Offset	Offset	Offset	Offset	Décalage	Offset	**偏移量**
	.0A	RDR	Dimensionsbereich der Auflösung	Dimension range of the resolution	Gama de dimensiones de la resolución	Plage dimensionnelle de la résolution	Campo dimensionale della risoluzione	**分辨率的尺寸范**围
	.0B	RNR	Zahlenbereich der Auflösung	Number range of the resolution	Rango numérico de la resolución	Plage numérique de la résolution	Campo numerico della risoluzione	**分辨率的数字范**围
	.0C	AccessRights	Zugriffsrechte	Access rights	Derechos de acceso	Droits d'accès	Diritti di accesso	访问权限
	.0D	DisplayFormat	Anzeigeformat	Display format	Formato de visualización	Format d'affichage	Formato di visualizzazione	显示格式
	.0E	Min	Untere Grenze	Lower limit	Límite inferior	Limite inférieure	Limite inferiore	**下限**
	.0F	Max	Obere Grenze	Upper limit	Límite superior	Limite supérieure	Limite superiore	**上限**
	.10	Symbol	Name/Bezeichnung des Objekts	Name/designation of the object	Nombre / denominación del objeto	Nom/désignation de l'objet	Nome/Denominazione dell'oggetto	对象的名称**/称号**
003A		VersionCount	Versionszähler	Version counter	Contador de versiones	Compteur de versions	Contatore versione	**版本**计数器
	.01	ProfileVersion	Profilversion	Profile version	Versión de perfil	Version profil	Versione profilo	**配置文件版本**
	.02	PChVersion	Parameterkanalversion	Parameter channel version	Versión de canal de parámetros	Version du canal de paramètres	Versione canale parametro	**参数通道版本**
	.03	HardwareVersion	Hardware-Version	Hardware version	Versión de hardware	Version matériel	Versione hardware	**硬件版本**
	.04	FirmwareVersion	Firmware-Version	Firmware version	Versión de firmware	Version firmware	Versione firmware	**固件版本**
003B		PDIN_Descr	Beschreibung der Eingangsprozessdat	IN process data description	Descripción de datos de proceso de	Description des données d'entrée de	Descrizione dei dati di processo in	IN过程数据描述

10 Appendix B

10.1 Translation table for the object names

Index	Subindex	Objektname (Basisprofil)	DE	EN-US	ES	FR	IT	ZH-CN
			en		entrada	process	ingresso	
	.01	Type	Typ des I/O-Datums	Type of I/O data item	Referencia de la fecha de E/S	Type de date E/S	Tipo di data I/O	I/O**数据**项的类型
	.02	ChNo	Anzahl der Kanäle	Number of channels	Número de canales	Nombre de canaux	Numero di canali	**通道数**
	.03	ChLength	Länge eines Kanals	Length of a channel	Longitud de un canal	Longueur d'un canal	Lunghezza di un canale	**通道**长度
003C		PDOUT_Descr	Beschreibung der Ausgangsprozessdaten	OUT process data description	Descripción de datos de proceso de salida	Description des données de sortie de process	Descrizione dei dati di processo in uscita	OUT过程数据描述
	.01	Type	Typ des I/O-Datums	Type of I/O data item	Referencia de la fecha de E/S	Type de date E/S	Tipo di data I/O	I/O**数据**项的类型
	.02	ChNo	Anzahl der Kanäle	Number of channels	Número de canales	Nombre de canaux	Numero di canali	**通道数**
	.03	ChLength	Länge eines Kanals	Length of a channel	Longitud de un canal	Longueur d'un canal	Lunghezza di un canale	**通道**长度
003D		WakeUpTime	Hochlaufzeit	Startup time	Tiempo de arranque	Temps d'accélération	Tempo di avviamento	启动时间
003E		EnergyMgmt	Energiemanagement	Energy management	Gestión de energía	Gestion de l'énergie	Gestione dell'energia	能源管理
	.01	ActualMode	Aktueller Energiesparmodus	Current energy-saving mode	Modo de ahorro de energía actual	Mode actuel d'économie d'énergie	Modalità di risparmio energetico attuale	当前的节能模式
	.02	Pause	Pausenzeit	Pause time	Tiempo de pausa	Durée de pause	Tempo di pausa	暂停时间
0040		ListOfObjToRestore	Liste der wiederherzustellenden Objekte	List of objects to be restored	Lista de objetos a restaurar	Liste des objets à rétablir	Elenco degli oggetti da ripristinare	待恢复的对象列表
	.01 … .N	SubModNoN IndexN SubindexN	Sub-Modul-Nummer, Index und Subindex des N-ten Objekts	Submodule number, index and subindex of the Nth object	Número de submódulo, índice y subíndice del objeto nº n	Numéro du sous-module, index et sous-index du Nième objet	Numero modulo subordinato, indice e indice subordinato dell'oggetto Nesimo	子模块编号，第N个对象的索引和子索引
0041		RefSubBusStructure	Sub-Bus-Sollkonfiguration	Desired sub-bus configuration	Configuración nominal del subbus	Configuration consigne de sous-bus	Configurazione nominale del bus subordinato	所需的子总线组态
0042		ModuleStatus	Modulstatus im Sub-Bus	Module status on the sub-bus	Estado del módulo en el subbus	Statut de module dans le sous-bus	Stato modulo nel bus subordinato	子总线上的模块状态
	.01 … .N	ModuleN	Status des N-ten Moduls	Status of the Nth module	Estado del módulo nº n	Statut du Nième module	Stato del Nesimo modulo	第N个模块的状态
0043		SubBusBehaviour	Sub-Bus-Verhalten	Sub-bus behavior	Comportamiento del subbus	Comportement sous-bus	Comportamento del bus subordinato	子总线特性
	.01	AutoSetup	Automatischer Anlauf	Automatic startup	Secuencia de activación automática	Démarrage automatique	Avvio automatico	自动重启
	.02	AutoRestart	Automatischer Wiederanlauf	Automatic restart	Reinicio automático	Redémarrage automatique	Riavvio automatico	自动重启

10 Appendix B

10.1 Translation table for the object names

Index	Subindex	Objektname (Basisprofil)	DE	EN-US	ES	FR	IT	ZH-CN
	.03	Remaining	Restsystem	Residual system	Sistema residual	Système résiduel	Sistema residuo	剩余系统
0044		SubBusControl	Steuerung des Sub-Busses	Sub-bus control	Sistema de control del subbus	Commande du sous-bus	Controllo del bus subordinato	子总线控制
	.01	SubBusControlCMD	Sub-Bus Kontrolle	Sub-bus control	Control del subbus	Contrôle du sous-bus	Controllo del bus subordinato	子总线控制
0045		InitFWDownload	Firmware-Download-Initialisierung	Initializing firmware download	Inicialización de descarga de firmware	Initialisation du téléchargement du firmware	Inizializzazione download firmware	初始化固件下载
	.01	Status	Status	Status	Estado	Etat	Stato	状态
	.02	Control	Steuerung	Controller	Sistema de control	Automate	Comando	控制器
	.03	HeaderVersion	Header-Information	Header information	Información de cabecera	Information en-tête	Informazione header	标题版本
	.04	UpdateVersion	Update-Version	Update version	Versión de actualización	Version d'actualisation	Versione update	更新版本
	.05	BaseData	Basisdaten	Basic data	Datos básicos	Données de base	Dati essenziali	基本数据
0047		AddInfo	Zusätzliche Informationen	Additional information	Informaciones adicionales	Informations complémentaires	Informazioni aggiuntive	更多信息
	.01	AddInfo.SafetyProtType	Safety-Protokolltyp	Safety protocol type	Tipo de protocolo de seguridad	Type de protocole Safety	Tipo di protocollo safety	安全协议类型
	.02	AddInfo.SafetyProtVers	Safety-Protokollversion	Safety protocol version	Versión de protocolo de seguridad	Version de protocole Safety	Versione di protocollo safety	安全协议版本
	.03	AddInfo.LegacyInfo	Information zu Vorgängerversionen	Information about previous version	Información sobre versiones anteriores	Informations relatives aux versions précédentes	Informazione sulle versioni precedenti	有关之前版本的信息
C000 … C07F		ProjBasProf	Projektion des Basisprofils für Sub-Busse	Projection of basic profile for sub-buses	Proyección del perfil básico para subbus	Projection du profil de base pour sous-bus	Proiezione del profilo di base per bus subordinati	**子**总线基本配置文件的映射
E800		DiagStateLong	Diagnosezustand (Langform)	Diagnostic state (long form)	Estado de diagnóstico (formato largo)	Etat de diagnostic (forme longue)	Stato di diagnosi (forma estesa)	诊断状态（长格式）
E801		DiagHistory	Diagnosearchiv	Diagnostics archive	Archivo de diagnóstico	Archive de diagnostic	Archivio diagnosi	诊断归档
	.01	DiagState1	Ältester Diagnosezustand in der Kurzform	Oldest diagnostic state in short form	Estado de diagnóstico más antiguo en el formato corto	Etat de diagnostic le plus ancien, forme courte	Stato di diagnosi più vecchio nella forma breve	**短格式中最旧的**诊断状态
	…							
	N	DiagStateN	Neuester Diagnosezustand in der Kurzform	Newest diagnostic state in short form	Estado de diagnóstico más nuevo en el formato corto	Etat de diagnostic le plus récent, forme courte	Stato di diagnosi più recente nella forma breve	**短格式中最新的**诊断状态

Index	Subindex	Objektname (Basisprofil)	DE	EN-US	ES	FR	IT	ZH-CN
E802		DiagHistoryLong	Diagnosearchiv (Langform)	Diagnostics archive (long form)	Archivo de diagnóstico (formato largo)	Archive de diagnostic (forme longue)	Archivio diagnosi (forma estesa)	诊断归档（长格式）
	.01	DiagState1	Ältester Diagnosezustand in der Langform	Oldest diagnostic state in long form	Estado de diagnóstico más antiguo en el formato largo	Etat de diagnostic le plus ancien, forme longue	Stato di diagnosi più vecchio nella forma estesa	长格式中最旧的诊断状态
	…							
	N	DiagStateN	Neuester Diagnosezustand in der Langform	Newest diagnostic state in long form	Estado de diagnóstico más nuevo en el formato largo	Etat de diagnostic le plus récent, forme longue	Stato di diagnosi più recente nella forma estesa	长格式中最新的诊断状态
E803		TraceBuffer	Puffer für Protokollierung	Logging buffer	Búfer para registro	Tampon d'établissement de protocole	Buffer per registrazione	记录缓冲区
E804		LanguageAvailable	Verfügbare Sprachen	Available languages	Idiomas disponibles	Langues disponibles	Lingue disponibili	**可用的**语言
E805		ObjDescrLong	Objektbeschreibung (Langform)	Object description (long form)	Descripción de objeto (formato largo)	Description d'objet (forme longue)	Descrizione oggetto (forma estesa)	对象描述（长格式）
E806		ComplDiagState	Vollständiger aktueller Diagnosezustand	Complete current diagnostic state	Estado de diagnóstico actual completo	Etat de diagnostic actuel, complet	Stato di diagnosi attuale completo	完整的当前诊断状态
	.01	DiagState1	Ältester Diagnosezustand in der Kurzform	Oldest diagnostic state in short form	Estado de diagnóstico más antiguo en el formato corto	Etat de diagnostic le plus ancien, forme courte	Stato di diagnosi più vecchio nella forma breve	**短格式中最旧的**诊断状态
	…							
	N	DiagStateN	Neuester Diagnosezustand in der Kurzform	Newest diagnostic state in short form	Estado de diagnóstico más nuevo en el formato corto	Etat de diagnostic le plus récent, forme courte	Stato di diagnosi più recente nella forma breve	**短格式中最新的**诊断状态
E807		Device FW	Firmware-File	Firmware file	Archivo de firmware	Fichier firmware	File firmware	**固件文件**
E808		OnBoardDeviceDescFile	OnBoard Gerätebeschreibungsdatei (komprimiert)	Onboard device description file (compressed)	Archivo de descripción de dispositivos Onboard (comprimido)	Fichier de description de l'appareil embarqué (comprimé)	Nome del file di descrizione del dispositivo OnBoard (compresso)	**机**载设备描述文件名称（已压缩）
E809		BackUpDataCompr	Komprimierte Daten für Geräte-Backup	Compressed data for device backup	Datos comprimidos para respaldo de dispositivos	Données comprimées pour sauvegarde d'appareils	Dati compressi per backup dispositivo	设备备份的压缩数据

.

.

.

.

- End -