

Network Interface Appendix

Anybus[®] CompactCom 30 PROFINET IO 2-Port

Doc.Id. HMSI-168-49
Rev. 2.53

Important User Information

This document is intended to provide a good understanding of the functionality offered by Anybus CompactCom 30 PROFINET IO 2-Port. The document only describes the features that are specific to the Anybus CompactCom 30 PROFINET IO 2-Port. For general information regarding the Anybus CompactCom, consult the Anybus CompactCom design guides.

The reader of this document is expected to be familiar with high level software design, and communication systems in general. The use of advanced PROFINET IO 2-Port-specific functionality may require in-depth knowledge in PROFINET IO 2-Port networking internals and/or information from the official PROFINET IO 2-Port specifications. In such cases, the people responsible for the implementation of this product should either obtain the PROFINET IO 2-Port specification to gain sufficient knowledge or limit their implementation in such a way that this is not necessary.

Liability

Every care has been taken in the preparation of this manual. Please inform HMS Industrial Networks AB of any inaccuracies or omissions. The data and illustrations found in this document are not binding. We, HMS Industrial Networks AB, reserve the right to modify our products in line with our policy of continuous product development. The information in this document is subject to change without notice and should not be considered as a commitment by HMS Industrial Networks AB. HMS Industrial Networks AB assumes no responsibility for any errors that may appear in this document.

There are many applications of this product. Those responsible for the use of this device must ensure that all the necessary steps have been taken to verify that the applications meet all performance and safety requirements including any applicable laws, regulations, codes, and standards.

HMS Industrial Networks AB will under no circumstances assume liability or responsibility for any problems that may arise as a result from the use of undocumented features, timing, or functional side effects found outside the documented scope of this product. The effects caused by any direct or indirect use of such aspects of the product are undefined, and may include e.g. compatibility issues and stability issues.

The examples and illustrations in this document are included solely for illustrative purposes. Because of the many variables and requirements associated with any particular implementation, HMS Industrial Networks AB cannot assume responsibility for actual use based on these examples and illustrations.

Intellectual Property Rights

HMS Industrial Networks AB has intellectual property rights relating to technology embodied in the product described in this document. These intellectual property rights may include patents and pending patent applications in the US and other countries.

Trademark Acknowledgements

Anybus ® is a registered trademark of HMS Industrial Networks AB. All other trademarks are the property of their respective holders.

Warning:	This is a class A product. In a domestic environment this product may cause radio interference in which case the user may be required to take adequate measures.
ESD Note:	This product contains ESD (Electrostatic Discharge) sensitive parts that may be damaged if ESD control procedures are not followed. Static control precautions are required when handling the product. Failure to observe this may cause damage to the product.

Table of Contents

About This Document

Related Documents	7
Document History	7
Conventions & Terminology	8
Support.....	8

Chapter 1 About the Anybus CompactCom 30 PROFINET IO 2-Port

General Information	9
Features	9
<i>Compatibility with Anybus CompactCom 30 PROFINET IO module</i>	9

Chapter 2 Tutorial

Introduction	11
Fieldbus Conformance Notes	11
Certification	11

Chapter 3 Basic Operation

General Information	13
<i>Software Requirements</i>	13
<i>Electronic Data Sheet (GSD)</i>	14
Network Identity	15
Communication Settings	16
Network Data Exchange	17
<i>Application Data Instances (ADIs)</i>	17
<i>Process Data</i>	18
<i>Caveats</i>	18
Diagnostics	19
Web Interface.....	19
E-mail Client	19
File System.....	20
<i>General Information</i>	20
<i>System Files</i>	20

Chapter 4 PROFINET IO Implementation Details

General Information	21
Application Process Instances (API)	22
Application Relationships (AR).....	22
Real Identification (RI)	23
<i>General Information</i>	23
<i>Configuration Mismatch</i>	24

Channel Diagnostics & Process Alarms	24
Identification & Maintenance (I&M)	25
<i>General Information</i>	25
<i>I&M Data Structures</i>	26
Fast Start Up	27
<i>General Information</i>	27
<i>Fast Start Up Configuration with STEP7</i>	28
PROFIenergy Profile	30
<i>Implementation</i>	30
PROFIsafe	31
 Chapter 5 FTP Server	
General Information	32
User Accounts	32
Session Example	33
 Chapter 6 Web Server	
General Information	34
Default Web Pages	34
<i>Network Configuration</i>	35
<i>Ethernet statistics page</i>	36
Server Configuration	37
<i>General Information</i>	37
<i>Index Page</i>	37
<i>Default Content Types</i>	38
<i>Authorization</i>	38
 Chapter 7 E-mail Client	
General Information	40
How to Send E-mail Messages	40
 Chapter 8 Server Side Include (SSI)	
General Information	41
Include File	41
Command Functions	42
<i>General Information</i>	42
<i>GetConfigItem()</i>	43
<i>SetConfigItem()</i>	44
<i>SsiOutput()</i>	46
<i>DisplayRemoteUser</i>	46
<i>ChangeLanguage()</i>	47
<i>IncludeFile()</i>	48
<i>SaveDataToFile()</i>	49
<i>printf()</i>	50
<i>scanf()</i>	52
Argument Functions	54
<i>General Information</i>	54

<i>ABCCMessage()</i>	54
SSI Output Configuration	58
Chapter 9 SNMP Agent	
General	59
Management Information (MIB)	59
MIB-II	59
<i>System Group Variables</i>	60
<i>Interfaces Group Variables</i>	60
Chapter 10 Anybus Module Objects	
General Information	62
Anybus Object (01h)	63
Diagnostic Object (02h)	64
Network Object (03h)	67
Network Configuration Object (04h)	68
Network PROFINET IO Object (0Eh)	77
PROFINET Additional Diagnostic Object (0Fh)	89
Socket Interface Object (07h)	94
SMTP Client Object (09h)	111
File System Interface Object (0Ah)	116
Network Ethernet Object (0Ch)	129
Functional Safety Module Object (11h)	130
Chapter 11 Host Application Objects	
General Information	133
PROFINET IO Object (F6h)	134
Ethernet Host Object (F9h)	158
Energy Control Object (F0h)	161
Functional Safety Object (E8h)	166
Appendix A Categorization of Functionality	
Basic	167
Extended	167
Advanced	167
Appendix B Anybus Implementation Details	
Extended LED Functionality	168
SUP-Bit Definition	168
Anybus State Machine	168
Application Watchdog Timeout Handling	169

Appendix C Message Segmentation

General	170
Command Segmentation	170
Response Segmentation	171

Appendix D Flowcharts

Flowchart - Record Data Access	172
Flowchart - Configuration Mismatch (RI)	173
Flowchart - Establishment of Real Identification (RI)	174

Appendix E HICP (Host IP Configuration Protocol)

General	175
Operation	175

Appendix F Technical Specification

Front View	176
Network Connector, Brick Version	178
Protective Earth (PE) Requirements	179
Power Supply	179
Environmental Specification	179
EMC Compliance	179

Appendix G Timing & Performance

General Information	180
Process Data	181
<i>Overview</i>	181
<i>Anybus Read Process Data Delay (Anybus Delay)</i>	181
<i>Anybus Write Process Data Delay (Anybus Delay)</i>	181
<i>Network System Read Process Data Delay (Network System Delay)</i>	182
<i>Network System Write Process Data Delay (Network System Delay)</i>	182

Appendix H Conformance Test Guide

General	183
Reidentifying Your Product	184
Factory Default Reset	185
IP Address	185
Station Name	185
Certification in Generic Anybus Mode	186
Certification in Advanced Mode	187

Appendix I Copyright Notices

P. About This Document

For more information, documentation etc., please visit the HMS website, 'www.anybus.com'.

P.1 Related Documents

Document	Author
Anybus CompactCom Software Design Guide	www.anybus.com
Anybus CompactCom Hardware Design Guide	
Anybus CompactCom Software Driver User Guide	
Anybus CompactCom Profinet IO Fieldbus Appendix	
PROFINET IO specification, rev. 2.2	Profibus International
PROFenergy Technical Specification, rev. 1.0	Profibus International
PROFIsafe, IEC61158, SIL3 and ISO13849 PL _e	

P.2 Document History

Summary of Recent Changes (2.52 ... 2.53)

Change	Page(s)
Moved Front View and Brick Connector information to Technical Specification	176
Added note to SaveDataToFile() command	49
Added note to usage of attribute #8 in PROFINET IO Object	135
Corrected command in flowchart	187
Clarified usage of attribute #11 for conformance testing	184
Removed references to functions not available to PROFINET	170

Revision List

Revision	Date	Author(s)	Chapter(s)	Description
1.00 - 2.30	2009-05-12 - 2012-09-12	KeL, KaD	-	See earlier versions of this document
2.40	2012-12-04	KeL	1, 10	Minor update and added brick version
2.41	2013-03-22	KeL	1, 10, E	Minor update
2.42	2013-09-01	KeL	9, B	Minor updates and corrections
2.50	2014-01-08	KeL	1, 4, 10, 11, B	Safety functionality added, miscellaneous corrections
2.51	2014-05-05	KaD	P, 1, 10, 11	Minor updates
2.52	2014-07-21	KeL	6, 11	Minor updates
2.53	2015-04-01	KeL	8, 11, C, F, H	Minor updates

P.3 Conventions & Terminology

The following conventions are used throughout this manual:

- Numbered lists provide sequential steps
- Bulleted lists provide information, not procedural steps
- The terms 'Anybus' or 'module' refers to the Anybus CompactCom module
- The terms 'host' or 'host application' refers to the device that hosts the Anybus module
- Hexadecimal values are written in the format NNNNh or 0xNNNN, where NNNN is the hexadecimal value
- A byte always consists of 8 bits

P.4 Support

For general contact information and where to find support, please refer to the contact and support pages at www.anybus.com.

1. About the Anybus CompactCom 30 PROFINET IO 2-Port

1.1 General Information

The Anybus CompactCom 30 PROFINET IO 2-Port communication module provides instant PROFINET Real Time connectivity via the patented Anybus CompactCom host interface. Any device that supports this standard can take advantage of the features provided by the module, allowing seamless network integration regardless of network type.

This product conforms to all aspects of the host interface for Active modules defined in the Anybus CompactCom Hardware- and Software Design Guides, making it fully interchangeable with any other device following that specification. Generally, no additional network related software support is needed, however in order to take advantage of advanced network specific functionality, a certain degree of dedicated software support may be necessary.

1.2 Features

- PROFINET IO communication
- Ethernet or M12 connectors
- Brick version
- Supports PROFIenergy profile
- Black channel interface, offering a transparent channel with support for Functional Safety up to SIL3 with separate safety module¹
- Up to two APIs (including API 0)
- Up to 64 modules, 8 submodules each (up to 128 submodules in total)
- Up to 32767 ADIs
- Up to 256 bytes of Real Time I/O
- Generic and PROFINET-specific diagnostic support
- SNMP agent
- FTP server
- E-mail client
- Server Side Include (SSI) functionality
- Device identity customization
- Generic GSD file provided by HMS
- Supports PROFINET Fast Start Up

1.2.1 Compatibility with Anybus CompactCom 30 PROFINET IO module

Please note that the module ID of the Anybus CompactCom 30 PROFINET IO 2-Port module is different from the ID of the 1-port module. Also the GSD files differ. Depending on how the application is designed, it may not be possible to replace the 1-port module with the 2-port module without changes to the software and/or the configuration.

Also, the 2-port version supports the PROFIenergy profile.

1. Safe T100/PS from IXXAT recommended.

2. Tutorial

2.1 Introduction

This chapter is a complement to the Anybus CompactCom Implementation Tutorial. The ABCC tutorial describes and explains a simple example of an implementation with Anybus CompactCom. This chapter includes network specific settings that are needed for a host application to be up and running and possible to certify for use on PROFINET IO networks.

2.2 Fieldbus Conformance Notes

- When using the default settings of all parameters, the module is precertified for network compliance. However, any parameter changes which require deviations from the standard GSD-file supplied by HMS will require recertification. For further information, please contact HMS.
- For conformance reasons, the host application must implement support for network reset type 02h (Power-on + Factory Default) in the Application Object (FFh).

2.3 Certification

The following steps are necessary to perform to obtain a certification:

1. Change Vendor ID:

If you have obtained a unique Vendor ID from PNO (PROFIBUS Nutzerorganisation e.V.), replace the HMS Vendor ID with this. This is done by implementing the PROFINET IO object (F6h), instance 1, attribute 2, and returning the Vendor ID when receiving a Get_Attribute request.

2. Change Device ID:

Replace the HMS Device ID in the PROFINET IO object (F6h) with a Device ID of your own. Each product from a vendor must have a unique Device ID. Implement the PROFINET IO object (F6h), instance 1, attribute 1, and return the Device ID when receiving a Get_Attribute request.

3. Change Attributes Station Type, I&M Order ID and System Description (recommended):

If you change the identity of the module, please replace the Station Type (attr. 3), the I&M Order ID (attr. 8), and the System Description (attr. 19) in the PROFINET IO object (F6h), instance 1. Implement the PROFINET IO object (F6h), instance 1, attributes 3, 8, and 19. Return the Station Type, the I&M Order ID and the System Description when receiving a Get_Attribute request.

4. Change I&M Revision (optional):

To show the customer specific I&M Hardware and Software revisions, change PROFINET IO object (F6h), Instance 1, Attributes 10 and 11. Implement the PROFINET IO object (F6h), instance 1, attributes 10 and 11. Return the I&M Hardware and Software revisions when receiving a Get_Attribute request.

5. Modify the GSD-file:

Modify the PROFINET ABCC GSD file so that it corresponds to the changes made above.

6. Setting MAC-address (optional):

Set MAC-addresses in Ethernet Host Object (F9h), instance 1, attributes 1, 9 and 10.

For additional information and instructions, see “Conformance Test Guide” on page 183.

3. Basic Operation

3.1 General Information

3.1.1 Software Requirements

Generally, no additional network support code needs to be written to support the Anybus CompactCom 30 PROFINET IO 2-Port, however due to the nature of the PROFINET networking system certain things must be taken into account:

- Up to 32767 ADIs can be represented on PROFINET.
- ADI names, types and similar attributes cannot be accessed via PROFINET. They are however represented on the network through the built in web server.
- Up to 5 diagnostic instances can be created by the host application. An additional 6th instance may be created in event of a major fault.
- For conformance reasons, the host application must implement support for network reset type 02h (Power-on + Factory Default) in the Application Object (FFh).
- PROFINET in itself does not impose any particular timing demands when it comes to acyclic requests (i.e. requests towards instances in the Application Data Object), however it is generally recommended to process and respond to such requests within a reasonable time period (exactly what this means in practice depends on the implementation and the actual installation).
- The order in which ADIs are mapped to Process Data is significant and must be replicated in the IO Controller when setting up the network communication (i.e. modules must be set up in the same order, size and direction, as the mapped ADIs). If not taken into account, the network connection establishment will fail and no communication will take place.
- For technical reasons, the module will not shift from NW_INIT to WAIT_PROCESS unless there is a physical connection to a network (i.e. link sensed).
- The use of advanced PROFINET-specific functionality may require in-depth knowledge in PROFINET networking internals and/or information from the official PROFINET specification. In such cases, the people responsible for the implementation of this product is expected either to obtain these specifications to gain sufficient knowledge or limit their implementation in such a way that this is not necessary.

See also...

- “Application Data Instances (ADIs)” on page 17
- “Diagnostic Object (02h)” on page 64 (Anybus Module Object)
- Anybus CompactCom Software Design Guide, ‘Application Data Object (FFh)’

3.1.2 Electronic Data Sheet (GSD)

On PROFINET, the characteristics of a device is stored in an XML data file. This file, referred to as the 'GSD'-file, is used by PROFINET engineering tools when setting up the network configuration. HMS provides a generic GSD-file, which corresponds to the default settings in the module. However, due to the flexible nature of the Anybus CompactCom concept, it is possible to alter the behavior of the product in a way that invalidates the generic GSD-file.

See also...

- "Fieldbus Conformance Notes" on page 11

How to Associate a Bitmap to a Device Access Point

It is possible to associate a bitmap to a Device Access Point, using the GSD file.

For the Device Access Point, the following information needs to be added (add it right before the "</DeviceAccessPointItem>"):

```
<Graphics>
    <GraphicItemRef Type="DeviceSymbol" GraphicItemTarget="X"/>
</Graphics>
```

In addition to this, a list of graphics needs to be created. This list can be added directly after, for example, the "</DeviceAccessPointList>", or "</ValueList>" keywords. Please note that the "X" above and below shall be replaced with the proper value (if only one bitmap is used, replace X with 1).

```
<GraphicsList>
    <GraphicItem ID="X" GraphicFile="GSDML-VVVV-DDDD-N...N"/>
</GraphicsList>
```

The format of the name of the bitmap shall be as specified above, where VVVV corresponds to the Vendor ID (for example, "010C"), DDDD corresponds to the Device ID (for example, "0009") and "N...N" is a vendor specific extension (for example, "ABCCPRTPIC1").

How to Enable Initial Record Data

During the establishment of an IO connection between the IO device and the IO controller it is possible for the IO controller to send initial record data. This initial record data is sent using the PROFINET IO service record write. This service can be used at any time and will write data to a defined ADI. The initial record data is defined in the GSD file, and is specified for a submodule of a module. By default, the Anybus CompactCom module will not make use of any initial record data, but that can be enabled if needed.

To enable this functionality, the GSD file needs to be modified as specified below. In this example, 2 bytes are written to ADI 67 (ADI 67 corresponds to index 67) during startup of a PROFINET IO connection (the value can be configured by the end user):

```
<RecordDataList>
  <ParameterRecordDataItem Index="67" Length="1" TransferSequence="0">
    <Name TextId="T_ID_EXAMPLE2"/>
    <Ref DataType="Unsigned16" ByteOffset="0" DefaultValue="0"
      AllowedValues="0..65535" TextId="T_ID_EXAMPLE2_PRM_1"/>
  </ParameterRecordDataItem>
</RecordDataList>
```

It is recommended that the above GSD information is placed directly after the “</IOData>” keyword for the module for which the data is associated.

If more than one ADI need to be set, the keyword “ParameterRecordDataItem” is duplicated.

Please note that TextIds (“T_ID_xxx” above) need to be added to the “<ExternalTextList>” section of the GSD file (once for each language defined).

```
<Text TextId="T_ID_EXAMPLE2" Value="Config parameter 1"/>
<Text TextId="T_ID_EXAMPLE2_PRM_1" Value="Parameter value description"/>
```

3.2 Network Identity

By default, the module identifies itself as a generic Anybus implementation as follows:

Device ID	0009h (Anybus-CompactCom PROFINET IO)
Vendor ID	010Ch (HMS Industrial Networks)
Station Type	'ABCC-PRT 2-Port'

It is possible to customize the identity information so that the Anybus module appears as a vendor specific implementation rather than a generic Anybus product. Note however that this invalidates the standard GSD-file and thus recertification of the product is necessary.

See also...

- “Identification & Maintenance (I&M)” on page 25
- “PROFINET IO Object (F6h)” on page 134

3.3 Communication Settings

Network related communication settings are grouped in the Network Configuration Object (04h). In case of the Anybus CompactCom 30 PROFINET IO this includes...

- **Ethernet Interface Settings**
The module is locked to 100 Mbit full duplex operation as required by PROFINET.
- **TCP/IP Settings**
These settings must be set in order for the module to be able to participate on the network.
- **SMTP Account Settings**
These settings must be set in order for the module to be able to send e-mail messages.
- **PROFINET Station Name**
The module needs to be assigned a Station Name in order to participate on PROFINET.
- **Misc. Settings Related to IM1... IM4**
These settings specify the contents for IM1... IM4.

See also...

- “Identification & Maintenance (I&M)” on page 25
- “Web Server” on page 34
- “Network Configuration Object (04h)” on page 68
- “HICP (Host IP Configuration Protocol)” on page 175

3.4 Network Data Exchange

3.4.1 Application Data Instances (ADIs)

ADIs can be accessed acyclically from the network by means of Record Data read/write services. If addressed through a given API and Index range, the module translates the service into standard object requests towards the Application Data Object. If the host application responds with an error to such a request, that error will be translated to PROFINET standard.

The following parameters affect the addressing of ADIs on PROFINET:

- **Application Process Instance (API)**

API 0 (zero) provides access to data in the Application Data Object, i.e. the ADIs. Acyclic requests towards other APIs will be forwarded to the PROFINET IO Object (F6h) by means of the 'Get_Record' and 'Set_Record'-commands.

The remainder of this section assumes API 0 (zero).

- **Slot & subslot**

The Slot and subslot values have no impact on the actual addressing of ADIs, except that the actual Slot and subslot needs to be populated with a module/submodule. This is always true for the DAP (Device Access Point), which occupies Slot #0/subslot #1. Other Slot/subslot values can also be used provided that the implementation populates it with a module/submodule.

- **Index**

There is a 1:1 correlation between ADI and index as long as the index number is less than - or equal to - 7FFFh. Index 0 (zero) is not associated with an ADI and cannot be used.

API	Slot	Subslot	Index	ADI	Comments
0	0	1	0000h	-	(not associated with ADIs)
			0001h	1	Device Access Point (DAP)
			0002h	2	
			
			7FFFh	32767	
			8000h...FFFFh	-	(not associated with ADIs)
	X (>0)	Y	0000h		Conditional; X and Y must be populated.
			0001h	1	
			0002h	2	
			
			7FFFh	32767	
			8000h...FFFFh	-	(not associated with ADIs)
>0	-	-	-	-	See "Application Process Instances (API)" on page 22

See also...

- "Caveats" on page 18
- "Application Process Instances (API)" on page 22
- "PROFINET IO Object (F6h)" on page 134

IMPORTANT: If 'Transparent Mode' has been activated for index range 0000... 7FFFh, no requests will be forwarded to the Application Data Object (in such case, this is handled through the PROFINET IO Object).

3.4.2 Process Data

Mapping an ADI to Write Process Data results in PROFINET input data, and mapping an ADI to Read Process Data results in PROFINET output data. By default, consistency over an entire ADI can only be achieved as long as the ADI does not contain more than one element of a specific data type, since each element results in one identifier (i.e. 'module').

Optionally, advanced users may define custom modules and submodules in the GSD-file, allowing consistency over ADIs with multiple elements. Note that in such case, the host application must handle the plugging of modules and submodules.

See also...

- “Real Identification (RI)” on page 23
- “Command Details: Plug_Module” on page 79
- “Command Details: Plug_Submodule” on page 80
- “Command Details: API_Add” on page 84

IMPORTANT: *The order in which ADIs are mapped to Process Data is significant and must be replicated in the IO Controller when setting up the network communication (i.e. modules must be set up in the same order, size and direction, as the mapped ADIs). If not taken into account, the network connection establishment will fail and no communication will take place (see “Real Identification (RI)” on page 23).*

3.4.3 Caveats

The length parameter in the Record Data request specifies the number of bytes to read/write.

- When reading more data than the actual size of the ADI, the response will only contain the actual ADI data, i.e. no padding on the data is performed by the module.
- When writing to an ADI, the length parameter is not checked by the module, i.e. the host application must respond with an error if the length differs from the actual size of the requested ADI.

See also...

- “Application Process Instances (API)” on page 22

3.5 Diagnostics

The standard Diagnostic Object (02h) provides access to basic diagnostic functionality. Major unrecoverable events will cause the module to physically disconnect itself from the network, thus preventing network participation. Other severity levels either produce a Channel Diagnostic entry/alarm or a Generic Diagnostic entry/alarm, depending on the Event Code.

Up to 5 diagnostic instances can be created by the host application. An additional 6th instance may be created in event of a major unrecoverable fault.

Please note that the application can only create diagnostic instances, when the Anybus CompactCom module is in state PROCESS_ACTIVE or IDLE. Any other attempt to create a diagnostic instance will result in an error code.

See also...

- “Channel Diagnostics & Process Alarms” on page 24
- “Diagnostic Object (02h)” on page 64
- “PROFINET Additional Diagnostic Object (0Fh)” on page 89

3.6 Web Interface

The built-in web server can be used to provide rich, dynamic content, by means of SSI scripting. This enables access to information and configuration settings within the file system, as well as through the Anybus CompactCom object module.

Web server content resides within the FLASH-based file system, which means it can be accessed and customized as needed using a standard FTP client.

See also...

- “File System” on page 20
- “FTP Server” on page 32
- “Web Server” on page 34
- “Server Side Include (SSI)” on page 41

3.7 E-mail Client

The built-in email client enables the host application to send e-mail messages stored in the file system, or defined directly within the SMTP Client Object (09h). Messages are scanned for SSI content, which means it's possible to embed dynamic information from the file system or from the Anybus CompactCom object model.

See also...

- 3-20 “File System”
- 7-40 “E-mail Client”
- 8-41 “Server Side Include (SSI)”
- 10-111 “SMTP Client Object (09h)”

3.8 File System

3.8.1 General Information

The built-in file system hosts 2 MByte of non-volatile storage, which can be accessed by the HTTP and FTP servers, the e-mail client, and the host application.

The file system uses the following conventions:

- ‘\’ (backslash) is used as a path separator
- A ‘path’ originates from the system root and as such must begin with a ‘\’
- A ‘path’ must not end with a ‘\’
- Names may contain spaces (‘ ’) but must not begin or end with one.
- Names must not contain one of the following characters: ‘\ / : * ? “ < > |’
- Names cannot be longer than 48 characters
- A path cannot be longer than 255 characters (filename included)

See also...

- “FTP Server” on page 32
- “Web Server” on page 34
- “E-mail Client” on page 40
- “Server Side Include (SSI)” on page 41
- “File System Interface Object (0Ah)” on page 116

IMPORTANT: *The file system is located in flash memory. Due to technical reasons, each flash segment can be erased approximately 100000 times before failure, making it unsuitable for random access storage.*

The following operations will erase one or more flash segments:

- *Deleting, moving or renaming a file or directory*
- *Writing or appending data to an existing file*
- *Formatting the file system*

3.8.2 System Files

The file system contains a set of files used for system configuration. These files, known as “system files”, are regular ASCII files which can be altered using a standard text editor (such as the Notepad in Microsoft Windows™). The format of these files are, with some exceptions, based on the concept of ‘keys’, where each ‘key’ can be assigned a value, see below.

Example:

```
[Key1]
value of Key1

[Key2]
value of Key2
```

4. PROFINET IO Implementation Details

4.1 General Information

This chapter covers PROFINET-specific details in the Anybus implementation. Note that the use of such functionality may require in-depth knowledge in PROFINET networking internals and/or information from the official PROFINET specification. In such cases, the people responsible for the implementation of this product are expected either to obtain these specifications to gain sufficient knowledge or limit their implementation in such a way that this is not necessary.

Implementation overview:

- **Conformance Class**
The Anybus module complies to conformance class B.
- **Performance Characteristics**
 - 100 Mbps, full duplex with autonegotiation enabled as default
 - Real Time (RT) communication, 2 ms cycle time
- **Device Model**
 - One IO Device instance
 - Each IO Device instance includes one or more Application Processes referenced by its' identifier (API). API 0 (zero) is implemented by default, additional APIs may be implemented by the host application.
 - Each API implements one or more slots
 - Each Slot implements one or more subslots
 - Each subslot may implement one or more Channels
- **Slots & Subslots**
Up to 64 slots, with up to 8 subslots per slot (up to 128 subslots in total).
- **IO Data**
Up to 256 bytes of IO Data in each direction.

IMPORTANT: *The flexible nature of the Anybus CompactCom concept allows the behavior of this product to be altered in ways which deviate from the standard GSD-file. In such cases, a custom GSD-file must be created, and recertification of the product is necessary.*

See also...

- “Fieldbus Conformance Notes” on page 11
- “Electronic Data Sheet (GSD)” on page 14

4.2 Application Process Instances (API)

As mentioned previously, acyclic requests towards API 0 are forwarded to the Application Data Object. Acyclic requests towards other APIs are forwarded to the PROFINET IO Object (F6h) by means of the 'Get_Record'- and 'Set_Record'-commands.

Cyclic data exchange is by default carried out through API 0 (i.e. the Anybus associates modules and submodules with API 0). To allow specific profiles to be supported, it is possible to add custom APIs (during the 'SETUP'-state) using the PROFINET-specific command 'API_Add'. Note that in such a case, the host application must handle the plugging of modules and submodules (see "Real Identification (RI)" on page 23).

Note that APIs cannot be chosen arbitrarily, since API numbers are associated with specific profile implementations.

See also...

- "Real Identification (RI)" on page 23
- "Command Details: Get_Record" on page 138
- "Command Details: Set_Record" on page 140
- "Network PROFINET IO Object (0Eh)" on page 77 ("Command Details: API_Add" on page 84)

4.3 Application Relationships (AR)

On PROFINET, a connection between an IO Controller/Supervisor and an I/O device (in this case the Anybus) is called 'Application Relationship' (AR). The Anybus module supports multiple simultaneous Application Relationships, allowing multiple IO Supervisors to access its data and functions.

The host implementation can either ignore this functionality altogether, in which case the Anybus module will handle it automatically, or integrate the establishment and handling of Application Relationships into the host firmware.

Application Relationships are managed through the following functions:

- AR_Check_Ind (see "Command Details: AR_Check_Ind" on page 146)
- AR_Info_Ind (see "Command Details: AR_Info_Ind" on page 149)
- AR_Offline_Ind (see "Command Details: AR_Offline_Ind" on page 152)
- AR_Abort_Ind (see "Command Details: AR_Abort_Ind" on page 154)
- AR_Abort (see "Command Details: AR_Abort" on page 86)

In addition, the following commands are tagged with an AR handle, allowing the host application to keep track of requests associated with individual Application Relationships:

- Get_Record (see "Command Details: Get_Record" on page 138)
- Set_Record (see "Command Details: Set_Record" on page 140)
- Cfg_Mismatch_Ind (see "Command Details: Cfg_Mismatch_Ind" on page 148)
- End_Of_Prm_Ind (see "Command Details: End_Of_Prm_Ind" on page 151)
- Appl_State_Ready (see "Command Details: Appl_State_Ready" on page 85)

4.4 Real Identification (RI)

4.4.1 General Information

During establishment of an IO Connection towards the Anybus module, the configuration derived from the IO Controller (i.e. the 'Expected Identification') and the actual configuration in the Anybus module (i.e. the 'Real Identification' or RI) are compared.

- **Generic (Default) Configuration**

By default (i.e. if the application doesn't issue API_Add, Plug_Module, Plug_Submodule), the Anybus handles the plugging of modules and submodules automatically in accordance with the mapped Process Data as follows:

- A DAP is plugged into Slot 0 (zero)
- Modules are added in consecutive order (based on the order of the mapping commands)
- Single-element ADIs results in one module being added
- Multiple-element ADIs results in an equal number of modules being added
- One submodule per module
- All modules belong to API 0 (zero)

Example:

ADI #	Type	Order No.	Resulting Real Identification
-	-	-	Module 000000011h, Submodule 00000000h added to Slot 0 (DAP with PDEV). Note that physical submodules are not displayed.
6	SINT16	1	Module 000000003h, Submodule 00000000h added to Slot 1, Subslot 1
3	UINT8	2	Module 000000002h, Submodule 00000000h added to Slot 2, Subslot 1
9	SINT32	3	Module 000000004h, Submodule 00000000h added to Slot 3, Subslot 1

- **Advanced (Custom) Configuration (Advanced Users Only)**

Optionally, it is possible to override the default configuration during the 'SETUP'-state by means of the PROFINET-specific commands 'API_Add', 'Plug_Module' and 'Plug_Submodule'. This way, the host application can define exactly how ADIs are represented on PROFINET by defining custom modules and submodules. Note however that this also requires the use of a custom GSD-file containing the corresponding modules and submodules.

See also...

- "Application Process Instances (API)" on page 22
- "Command Details: Plug_Module" on page 79
- "Command Details: Plug_Submodule" on page 80
- "Command Details: API_Add" on page 84
- "Flowchart - Establishment of Real Identification (RI)" on page 174

4.4.2 Configuration Mismatch

In case of a mismatch, the Anybus notifies the IO Controller about this through the ‘module diff block’ in the service response, and issues ‘Cfg_Mismatch_Ind’ to inform the host application of each offending block.

The Anybus will unconditionally reject a configuration mismatch for any slot/subslot (i.e. the module will not shift to the ‘PROCESS ACTIVE’-state in case of a mismatch). However the host application can adapt to the Expected Identification in one of the following ways:

- Restart the Anybus module and adjust the configuration to match the Expected Identification that was handed over to the application through the ‘Cfg_Mismatch_Ind’-command.
- Adjust the configuration at a later stage (preferable between the commands ‘End_Of_Prm_Ind’ and ‘Appl_State_Ready’ by removing the offending modules/submodules and inserting new ones to match the Expected Identification.

See also...

- “Command Details: Cfg_Mismatch_Ind” on page 148
- “Command Details: End_Of_Prm_Ind” on page 151
- “Command Details: Appl_State_Ready” on page 85
- “Flowchart - Establishment of Real Identification (RI)” on page 174

4.5 Channel Diagnostics & Process Alarms

PROFINET IO uses Alarms when informing the IO Controller of diagnostic entries. In the Anybus implementation, it is possible for the application to create diagnostic entries by means of either the Diagnostic Object (02h) or the PROFINET Additional Diagnostic Object (0Fh). The former only provides rudimentary diagnostic support, while the latter can create detailed Channel Diagnostics as well as Process Alarms.

Instances are created in a fashion similar to that of the standard Diagnostic Object (02h), and can be tagged with a source API, slot- and subslot number, channel number, and channel type. Each instance corresponds to a Channel Diagnostic entry, and can be removed at a later stage using the ‘delete’-command.

Please note that the application can only create diagnostic instances, when the Anybus CompactCom module is in state PROCESS_ACTIVE or IDLE. Any other attempt to create a diagnostic instance will result in an error code.

Process Alarms are created by means of the object-specific ‘Process_Alarm’-command. Each alarm will be tagged with a source API, slot and subslot number, a structure specifying the nature of the alarm, and data associated with the alarm. Note however that issuing this command does not result in an instance being created, and consequently no ‘delete’-command can be used to remove it.

See also...

- “Diagnostics” on page 19
- “Diagnostic Object (02h)” on page 64
- “PROFINET Additional Diagnostic Object (0Fh)” on page 89
- “Command Details: Process_Alarm” on page 92

IMPORTANT: *Users are expected to be familiar with the concepts used for this functionality, or resort to using the functionality provided through the standard Diagnostic Object (02h).*

4.6 Identification & Maintenance (I&M)

4.6.1 General Information

Identification & Maintenance (I&M) provides a standard way of gathering information about an I/O device. The I&M information can be accessed by the IO Controller by means of acyclic Record Data Read/Write services.

By default, the Anybus module implements support for I&M0 as follows:

Default I&M0 Information:

IM Manufacturer ID	010Ch (HMS Industrial Networks)
IM Order ID	'ABCC-PRT (2-Port)'
IM Serial Number	(unique serial number, set during manufacturing)
IM Hardware Revision	(Anybus hardware revision ID, set during manufacturing)
IM Software Revision	(Anybus software revision, set during manufacturing)
IM Revision Counter	(Revision counter)
IM Profile ID	F600h (Generic Device)
IM Profile Specific Type	0004h (No profile)
IM Version	0101h
IM Supported	0000h (IM0 supported)

Optionally, the host application can customize the information for these I&M entries, or implement the support for the 'Get_IM_Record'- and 'Set_IM_Record'-commands to support all I&M record for all slots.

See also...

- "PROFINET IO Object (F6h)" on page 134
- "Command Details: Get_IM_Record" on page 142
- "Command Details: Set_IM_Record" on page 144
- "Network Configuration Object (04h)" on page 68

4.6.2 I&M Data Structures

The I&M records uses the following data structures.

Record	Content	Size	Description
I&M0	Manufacturer Id	2 bytes	PROFINET IO Object (F6h), attribute #2 ('Vendor ID/I&M Vendor ID')
	Order Id	20 bytes	PROFINET IO Object (F6h), attribute #8 ('I&M Order ID')
	Serial number	16 bytes	PROFINET IO Object (F6h), attribute #9 ('I&M Serial number')
	Hardware revision	2 bytes	PROFINET IO Object (F6h), attribute #10 ('I&M Hardware revision')
	Software revision	4 bytes	PROFINET IO Object (F6h), attribute #11 ('I&M Software revision')
	Revision counter	2 bytes	PROFINET IO Object (F6h), attribute #12 ('I&M Revision counter')
	Profile Id	2 bytes	PROFINET IO Object (F6h), attribute #13 ('I&M Profile ID')
	Profile specific type	2 bytes	PROFINET IO Object (F6h), attribute #14 ('I&M Profile specific type')
	IM version	2 bytes	PROFINET IO Object (F6h), attribute #15 ('I&M Version')
	IM version	2 bytes	PROFINET IO Object (F6h), attribute #15 ('I&M Version')
I&M1	Tag Function	32 bytes	Network Configuration Object (04h), attribute #16 ('I&M Tag Function')
	Tag Location	22 bytes	Network Configuration Object (04h), attribute #17 ('I&M Tag Location')
I&M2	Installation date	16 bytes	Network Configuration Object (04h), attribute #18 ('I&M Installation Date')
I&M3	Descriptor	54 bytes	Network Configuration Object (04h), attribute #19 ('I&M Descriptor')
I&M4 ^a	Signature	54 bytes	Default: All bytes set to zero (00h)

a. Data of this field must only be accessed from the network by the IO Controller/Supervisor.

See also...

- “PROFINET IO Object (F6h)” on page 134

4.7 Fast Start Up

4.7.1 General Information

The Fast Start Up (FSU) function enables PROFINET IO devices, connected to the network, to power up quickly. This is useful in, for example, robot applications, where rapid retooling is necessary. This function has to be activated when configuring the Anybus CompactCom module.

In the GSD file a few keywords for this functionality are used. The FSU time is defined as the number of milliseconds from hardware reset (or power-on) until establishment of PROFINET IO Communication. If the FSU-time is measured to be larger than approximately 1500 ms it is recommended that this functionality is disabled.

The following keywords are used for this functionality (listed for the Device Access Point(s)):

- **PowerOnToCommReady**
FSU time, in milliseconds (ms). Default value is 0 ms.
- **DCP_HelloSupported**
Keyword stating whether or not the device will transfer “Hello” messages at power on. Default value: true.

To disable FSU, set the keywords to the following values:

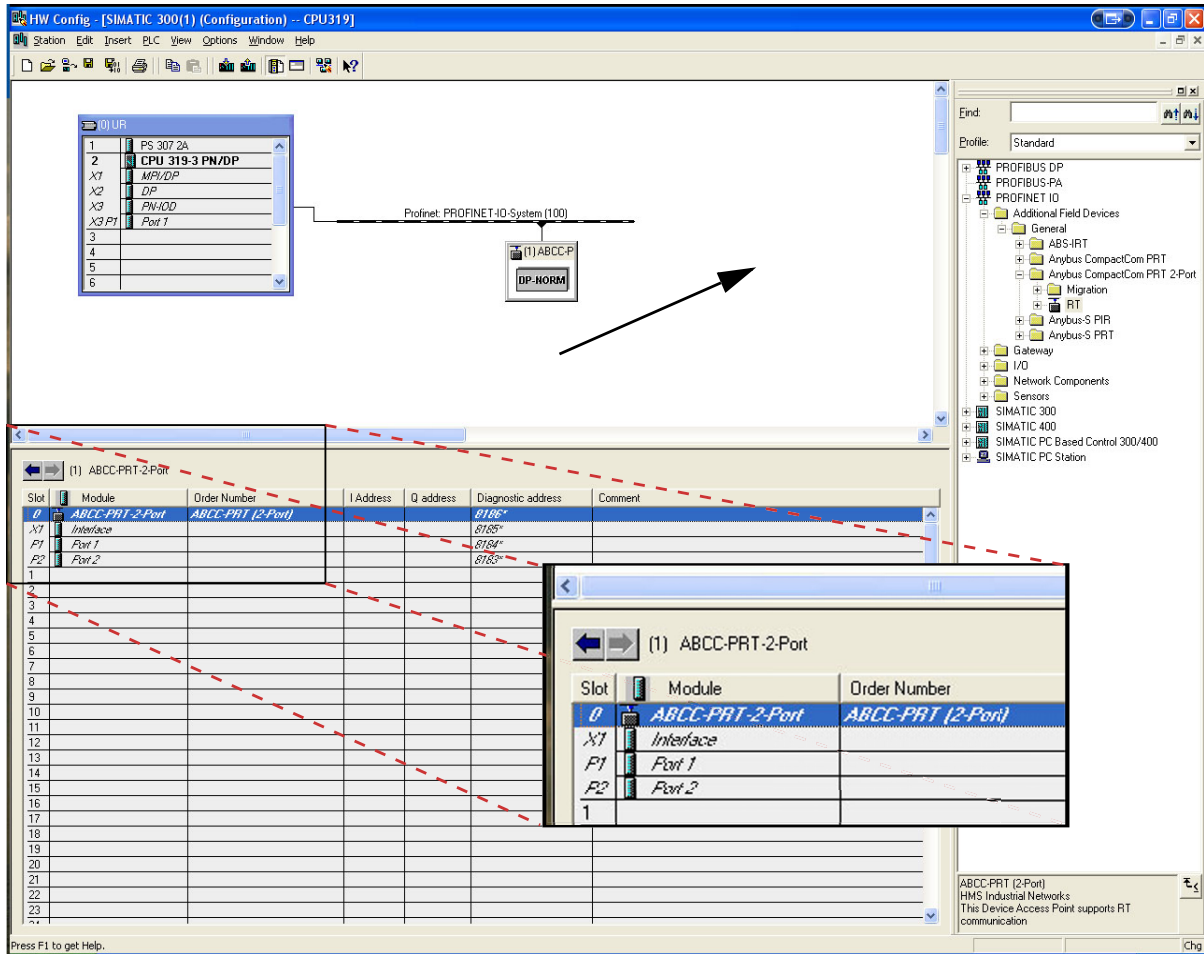
- **PowerOnToCommReady**
Remove this keyword from the GSD file.
- **DCP_HelloSupported**
Value: false.

4.7.2 Fast Start Up Configuration with STEP7

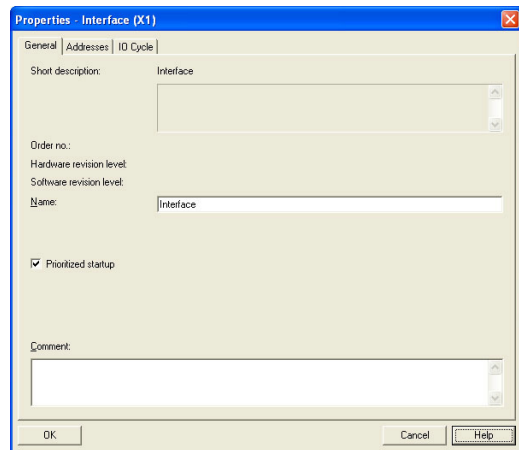
The example below shows the procedure when the Siemens tool STEP7 is used for configuration.

Activation of Fast Start Up

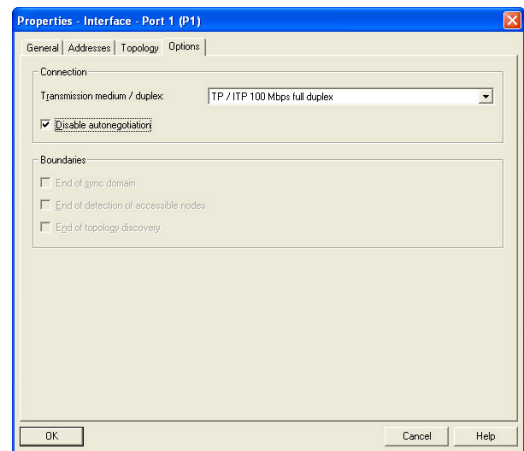
1. Start the configuration tool. The figure below shows the HW Config window of the STEP7 tool. The enlarged part from the Module column is used when activating Fast Start Up.



2. Double click on “Interface” in the Module column. The window shown to the right will appear. Choose the “General” tab and check the box “Prioritized startup”.



3. Return to the HW Config window. Double click on “Port 1” in the Module column. The window shown to the right will appear. Choose the Options tag. To configure fastest possible startup, choose transmission rate “100 Mbps, full duplex” and check the “Disable autonegotiation” box.



4. Repeat for Port 2.

4.8 PROFIenergy Profile

The Anybus CompactCom 30 PROFINET IO 2-Port module supports the PROFIenergy profile, according to the PROFIenergy Technical Specification, rev. 1.0. This profile makes it possible for a user to temporarily put a device in energy saving mode., e.g. during a lunch break or during weekends. The amount of power used by machines, when they are not in active use is thus reduced. Each device can be set individually to the energy saving mode that is the most optimal depending on the length of the production stop. Operators in factories, for example, can easily set all devices at the same time, in the, for each device, optimal energy saving mode.

The application defines the time for how long the device will stay in energy saving mode, and the device decides which mode will be the most appropriate. Transitions between the “Ready to operate” mode and all saving modes are mandatory, while transitions between different energy saving modes are optional. The transition from “Ready to operate mode” to “Power off mode” is not mandatory, as re-powering the device may mean hands on restart.

4.8.1 Implementation

The PROFIenergy profile is implemented in the Anybus CompactCom 30 PROFINET IO 2-Port module according to the state machine described in the PROFIenergy Technical Specification (available from PROFIBUS International). PROFIenergy commands arriving from the network will be translated into the Anybus CompactCom implementation as follows:

PROFIenergy command	Sub-command	Anybus CompactCom Implementation
Start_Pause	-	Translated into a StartPause command to instance 0 in the Energy Control Object, see page 164
End_Pause	-	Translated into a EndPause command to instance 0 in the Energy Control Object, see page 165
Query_Modes ^a	List_Energy_Saving_Modes	Instances present in the Energy object will be listed. The maximum number of instances/modes in the Energy Control Object for the PROFIenergy profile is 254.
	Get_Mode	Supported attributes for the requested mode/instance will be read. Unsupported attributes will be set to default.
PEM_Status ^a	-	Several reads of instance 0 and the currently energy saving mode instance in the Energy object will be performed
PE_Identify ^a	-	Will read the “PROFIenergy functionality” attribute (no. 23) in the PROFINET IO Object (F6h)
Query_Measurement	Get_Measurement_List	Not supported
	Get_Measurement_Values	

a. This command is enabled/disabled by the PROFIenergy attribute (no 23) in the PROFINET IO Object (F6h)

The PROFIenergy profile is valid for all sub-slots and is accessed through index 80A0h. The functionality is enabled/disabled by attribute 23 in the PROFINET IO Object (F6h). This attribute contains a bit mask that will enable or disable different parts of the PROFIenergy functionality. Maximum number of instances/modes in the Energy Control Object is 254.

See also...

- “Energy Control Object (F0h)” on page 161
- “PROFINET IO Object (F6h)” on page 134

4.9 PROFI-safe

The Anybus CompactCom 30 PROFINET IO 2-Port module supports the PROFI-safe profile. This profile makes it possible for a user to send data on a black channel interface, i.e. a safe channel over PROFINET using an add on Safety Module, e.g. Safe T100/PS from IXXAT.

In generic (default) mode the Safety Module shall be located in slot 1. In advanced mode any slot can be used for the Safety Module, and the host application can specify the highest 16 bits of the module ID of the Functional Safety Module.

For an application to support PROFI-safe, the Functional Safety Object in the application have to be implemented. Slots are assigned using the Command Add_Safety_Module to the Network PROFINET IO object (0Eh). The safe communication is enabled in the host application Functional Safety Object (E8h).

The Anybus CompactCom serial channel is used for functional safety communication. When this channel is used for the host application, a second separate serial channel, is implemented for the functional safety communication, see Anybus CompactCom Hardware Design Guide.

See also...

- “Functional Safety Module Object (11h)” on page 130
- “Functional Safety Object (E8h)” on page 166
- “PROFINET IO Object (F6h)” on page 134
- “Network PROFINET IO Object (0Eh)” on page 77 (“Command Details: Add_Safety_Module” on page 87)
- Anybus CompactCom Hardware Design Guide

5. FTP Server

5.1 General Information

Category: extended

The built-in FTP-server makes it easy to manage the file system using a standard FTP client.

By default, the following port numbers are used for FTP communication:

- TCP, port 20 (FTP data port)
- TCP, port 21 (FTP command port)

The FTP server supports up to 8 concurrent connections.

5.2 User Accounts

User accounts are stored in the configuration file '`\ftp.cfg`'. This file holds the usernames, passwords, and home directory for all users. Users are not able to access files outside of their home directory.

File Format:

```
User1:Password1:Homedir1
User2:Password2:Homedir2
User3:Password3:Homedir3
```

Optionally, the `UserN:PasswordN`-section can be replaced by a path to a file containing a list of users as follows:

File Format ('`\ftp.cfg`')

```
User1:Password1:Homedir1
User2:Password2:Homedir2
\path\userlistA:HomedirA
\path\userlistB:HomedirB
```

The files containing the user lists shall have the following format:

File Format:

```
User1:Password1
User2:Password2
User3:Password3
```

Notes:

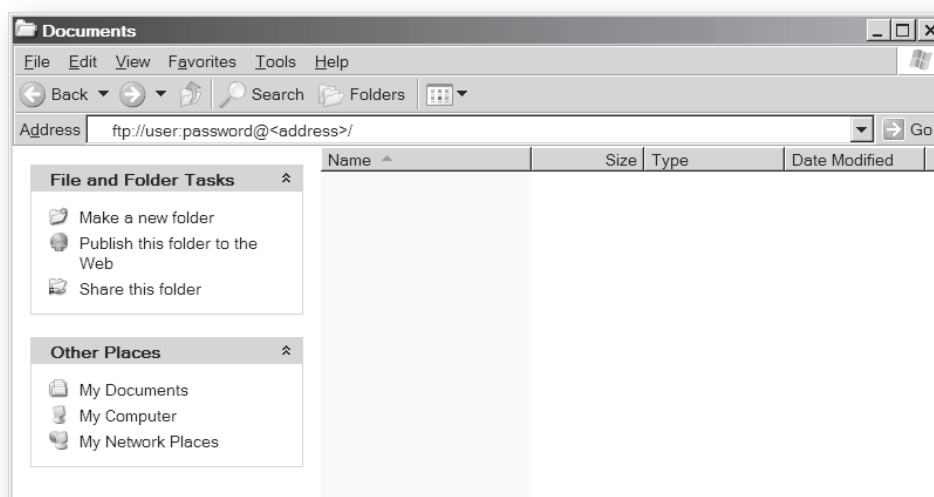
- usernames must not exceed 15 characters in length.
- Passwords must not exceed 15 characters in length.
- usernames and passwords must only contain alphabetic characters and/or numbers.
- If '`\ftp.cfg`' is missing or cannot be interpreted, all username/password combinations will be accepted and the home directory will be the FTP root (i.e. '`\ftp\`').
- The home directory for a user must also exist in the file system if they should be able to log in, just adding the user information to the '`ftp.cfg`' file it is not enough.

- If 'Admin Mode' has been enabled in the Ethernet Object, all username/password combinations will be accepted and the user will have unrestricted access to the file system (i.e. the home directory will be the system root).
- It is strongly recommended to have at least one user with root access ('\') permission. If not, 'Admin Mode' must be enabled each time a system file needs to be altered (including 'ftp.cfg').

5.3 Session Example

The Windows Explorer features a built-in FTP client which can easily be used to access the file system as follows:

1. Open the Windows Explorer by right-clicking on the 'Start'-button and selecting 'Explorer'
2. In the address field, type `FTP://<user>:<password>@<address>`
 - Substitute <address> with the IP address of the Anybus module
 - Substitute <user> with the username
 - Substitute <password> with the password
3. Press enter. The Explorer will now attempt to connect to the Anybus module using the specified settings. If successful, the file system will be displayed in the Explorer window.



6. Web Server

6.1 General Information

Category: extended

The built-in web server provides a flexible environment for end-user interaction and configuration purposes. The powerful combination of SSI and client-side scripting allows access to objects and file system data, enabling the creation of advanced graphical user interfaces.

The web interfaces is stored in the file system, which can be accessed through the FTP server. If necessary, the web server can be completely disabled in the Ethernet Host Object.

The web server supports up to 20 concurrent connections and communicates through port 80.

See also...

- “FTP Server” on page 32
- “Server Side Include (SSI)” on page 41
- “Ethernet Host Object (F9h)” on page 147

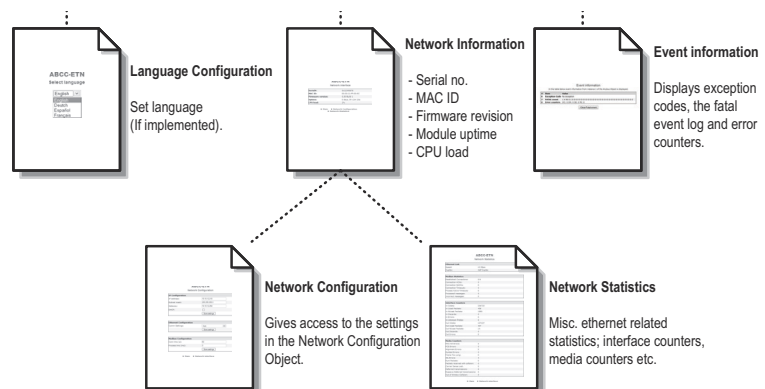
6.2 Default Web Pages

The default web interface consists of a set of virtual files; these virtual files may be replaced, but not permanently erased, by placing files with the same name in the same location (i.e. the web root).

The files can be used as-is or called from a customized web environment.

The files are:

```
<WebRoot>\style.css
<WebRoot>\arrow_red.gif
<WebRoot>\index.htm
<WebRoot>\netinfo.htm
<WebRoot>\netconfig.htm
<WebRoot>\netstat.htm
<WebRoot>\parameter.htm
<WebRoot>\eventinfo.htm
<WebRoot>\language.htm
```



Note: If none of these files are used, it is recommended to completely disable the virtual file system altogether in the File System Interface Object.

See also...

- “File System” on page 17
- “File System Interface Object (0Ah)” on page 116

6.2.1 Network Configuration

The network configuration page provides an interface for changing TCP/IP and SMTP settings in the Network Configuration Object.

ABCC-PRT (2-Port)
Network configuration

IP Configuration	
IP address:	<input type="text" value="10.11.20.227"/>
Subnet mask:	<input type="text" value="255.255.0.0"/>
Gateway:	<input type="text" value="0.0.0.0"/>
DHCP:	<input type="checkbox"/>
Host name:	<input type="text"/>
Domain name:	<input type="text"/>
<input type="button" value="Store settings"/>	

SMTP Settings	
SMTP Server:	<input type="text"/>
SMTP User:	<input type="text"/>
SMTP Pswd:	<input type="text"/>
<input type="button" value="Store settings"/>	

Safety Module Settings	
F address:	<input type="text" value="200"/> <input type="button" value="↑"/>
<input type="button" value="Store F-Address"/>	

▶ Main ▶ Network interface

The module needs a reset for the changes to take effect.

Available editable settings:

IP Configuration

Name	Description
IP address	The TCP/IP settings of the module
Subnet mask	Default values: 0.0.0.0
Gateway	Value ranges: 0.0.0.0 - 255.255.255.255
DHCP	Checkbox for enabling or disabling DHCP Default value: enabled
Host name	IP address or name Max 64 characters
Domain name	IP address or name Max 48 characters

SMTP Settings

Name	Description
SMTP Server	IP address or name Max 64 characters
SMTP User	Max 64 characters
SMTP Pswd	Max 64 characters

Safety Module Settings

Name	Description
F-address	The F-address used for the Safety Module as PROFIsafe address

6.2.2 Ethernet statistics page

The Ethernet statistics web page contains the following information:

Ethernet Link		Description
Port 1	Speed:	The current link speed.
	Duplex:	The current duplex configuration.
Port 2	Speed:	The current link speed.
	Duplex:	The current duplex configuration.
Interface Counters		Description
In Octets:		Received bytes.
In Ucast Packets:		Received unicast packets.
In NUCast packets:		Received non-unicast packets (broadcast and multicast).
In Discards:		Received packets discarded due to no available memory buffers.
In Errors:		Received packets discarded due to reception error.
In Unknown Protos:		Received packets with unsupported protocol type.
Out Octets:		Sent bytes.
Out Ucast packets:		Sent unicast packets.
Out NUCast packets:		Sent non-unicast packets (broadcast and multicast).
Out Discards:		Outgoing packets discarded due to no available memory buffers.
Out Errors:		Transmission errors.

6.3 Server Configuration

6.3.1 General Information

Category: advanced

Basic web server configuration settings are stored in the system file ‘\http.cfg’. This file holds the root directory for the web interface, content types, and a list of file types which shall be scanned for SSI.

File Format:

```
[WebRoot]
\web

[FileTypes]
FileType1:ContentType1
FileType2:ContentType2
...
FileTypeN:ContentTypeN

[SSIFileTypes]
FileType1
FileType2
...
FileTypeN
```

Web Root Directory

The web server cannot access files outside this directory.

Content Types

A list of file extensions and their reported content types.

See also...

- “Default Content Types” on page 38

SSI File Types

By default, only files with the extension ‘shtm’ are scanned for SSI. Additional SSI file types can be added here as necessary.

The web root directory determines the location of all files related to the web interface. Files outside of this directory and its subdirectories *cannot* be accessed by the web server.

6.3.2 Index Page

The module searches for possible index pages in the following order:

1. <WebRoot>\index.htm
2. <WebRoot>\index.html
3. <WebRoot>\index.shtm
4. <WebRoot>\index.wml

Note 1: Substitute <WebRoot> with the web root directory specified in ‘\http.cfg’.

Note 2: If no index page is found, the module will default to the virtual index file (if enabled).

See also...

- “Default Web Pages” on page 34

6.3.3 Default Content Types

By default, the following content types are recognized by their file extension:

File Extension	Reported Content Type
htm, html, shtm	text/html
gif	image/gif
jpeg, jpg, jpe	image/jpeg
png	image/x-png
js	application/x-javascript
bat, txt, c, h, cpp, hpp	text/plain
zip	application/x-zip-compressed
exe, com	application/octet-stream
wml	text/vnd.wap.wml
wmlc	application/vnd.wap.wmlc
wbmp	image/vnd.wap.wbmp
wmls	text/vnd.wap.wmlscript
wmlsc	application/vnd.wap.wmlscriptc
xml	text/xml
pdf	application/pdf
css	text/css

Content types can be added or redefined by adding them to the server configuration file, see “General Information” on page 37.

6.3.4 Authorization

Directories can be protected from web access by placing a file called ‘web_accs.cfg’ in the directory to protect. This file shall contain a list of users that are allowed to access the directory and its subdirectories.

File Format:

```

Username1:Password1
Username2:Password2
...
UsernameN:PasswordN

```

• List of approved users.


```

[AuthName]
(message goes here)

```

• Optionally, a login message can be specified by including the key [AuthName]. This message will be displayed by the web browser upon accessing the protected directory.

The list of approved users can optionally be redirected to one or several other files.

Note: If the list of approved users is put in another file, be aware that this file can be accessed and read from the network.

Example:

In this example, the list of approved users will be loaded from 'here.cfg' and 'too.cfg'.

```
[File path]
\i\put\some\over\here.cfg
\i\actually\put\some\of\it\here\too.cfg
```

```
[AuthName]
Howdy. Password, please.
```

7. E-mail Client

7.1 General Information

Category: extended

The built-in e-mail client allows the application to send e-mail messages through an SMTP-server. Messages can either be specified directly in the SMTP Client Object, or retrieved from the file system. The latter may contain SSI, however note that for technical reasons, certain commands cannot be used (specified separately for each SSI command).

The client supports authentication using the 'LOGIN' method. Account settings etc. are stored in the Network Configuration Object.

See also...

- "Network Configuration Object (04h)" on page 72
- "SMTP Client Object (09h)" on page 111

7.2 How to Send E-mail Messages

To be able to send e-mail messages, the SMTP-account settings must be specified.

This includes...

- A valid SMTP-server address
- A valid username
- A valid password

To send an e-mail message, perform the following steps:

1. Create a new e-mail instance using the 'Create'-command (03h)
2. Specify the sender, recipient, topic and message body in the e-mail instance
3. Issue the 'Send Instance Email'-command (10h) towards the e-mail instance
4. Optionally, delete the e-mail instance using the 'Delete'-command (04h)

Sending a message based on a file in the file system is achieved using the 'Send Email from File'-command. For a description of the file format, see "Command Details: Send Email From File" on page 114.

8. Server Side Include (SSI)

8.1 General Information

Category: advanced

Server Side Include functionality, or SSI, allows data from files and objects to be represented on web pages and in e-mail messages.

SSI are special commands embedded within the source document. When the Anybus module encounters such a command, it will execute it, and replace it with the result specified operation (if applicable).

By default, only files with the extension 'shtml' are scanned for SSI.

8.2 Include File

This function includes the contents of a file. The content is scanned for SSI.

Note: This function cannot be used in e-mail messages.

Syntax:

```
<?--#include file="filename"-->
```

filename-Source file

Default Output:

Scenario	Default Output
Success	(contents of file)

8.3 Command Functions

8.3.1 General Information

Command functions executes commands and includes the result.

General Syntax:

```
<?--#exec cmd_argument='command'-->
```

command-Command function, see below.

Command Functions:

Command	Valid for Email Messages	Page
GetConfigItem()	Yes	43
SetConfigItem()	No	44
SsiOutput()	Yes	46
DisplayRemoteUser	No	46
ChangeLanguage()	No	47
IncludeFile()	Yes	48
SaveDataToFile()	No	49
printf()	Yes	50
scanf()	No	52

8.3.2 GetConfigItem()

This command returns specific information from a file in the file system.

File Format:

The source file must have the following format:

```
[key1]
value1

[key2]
value2
...
[keyN]
valueN
```

Syntax:

```
<!--exec cmd_argument='GetConfigItem("filename", "key"[,"separator"])'-->
```

filename- Source file to read from.
 key - Source [key] in file.
 separator- Optional; specifies line separation characters (e.g. "
").
 (default is CRLF).

Default Output:

Scenario	Default Output
Success	<i>(value of specified key)</i>
Authentication Error	"Authentication error "
File open error	"Failed to open file "filename" "
Key not found	"Tag (key) not found "

Example:

The following SSI...

```
<!--exec cmd_argument='GetConfigItem("\fruit.cnf", "Lemon")'-->
```

... in combination with the following file ("fruit.cnf")...

```
[Apple]
Green

[Lemon]
Yellow

[Banana]
Blue
```

... returns the string 'Yellow'.

8.3.3 SetConfigItem()

This function stores an HTML-form as a file in the file system.

Note: This function cannot be used in e-mail messages.

Syntax:

```
<?--#exec cmd_argument='SetConfigItem("filename" [, Overwrite])'-->
```

filename- Destination file. If the specified file does not exist, it will be created (provided that the path is valid).

Overwrite -Optional; forces the module to create a new file each time the command is issued. The default behavior is to modify the existing file.

File Format:

Each form object is stored as a [tag], followed by the actual value.

```
[form object name 1]
form object value 1
```

```
[form object name 2]
form object value 2
```

```
[form object name 3]
form object value 3
```

...

```
[form object name N]
form object value N
```

Note: Form objects with names starting with underscore ('_') will not be stored.

Default Output:

Scenario	Default Output
Success	"Configuration stored to " <i>filename</i> " "
Authentication Error	"Authentication error "
File open error	"Failed to open file " <i>filename</i> " "
File write error	"Could not store configuration to " <i>filename</i> " "

Example:

The following example demonstrates how to use this function. The resulting page sends a form to itself, which is then evaluated by the SetConfigItem command.

```
<HTML>
<HEAD><TITLE>SetConfigItem Test</TITLE></HEAD>
<BODY>

<?--#exec cmd_argument='SetConfigItem("\food.txt")'-->

<FORM action="test.shtm">
  <P>
    <LABEL for="Name">Name: </LABEL><BR>
    <INPUT type="text" name="Name"><BR><BR>

    <LABEL for="_Age">Age: </LABEL><BR>
    <INPUT type="text" name="_Age"><BR><BR>

    <LABEL for="Food">Food: </LABEL><BR>
    <INPUT type="radio" name="Food" value="Cheese"> Cheese<BR>
    <INPUT type="radio" name="Food" value="Sausage"> Sausage<BR><BR>

    <LABEL for="Drink">Drink: </LABEL><BR>
    <INPUT type="radio" name="Drink" value="Wine"> Wine<BR>
    <INPUT type="radio" name="Drink" value="Beer"> Beer<BR><BR>

    <INPUT type="submit" name="_submit">
    <INPUT type="reset" name="_reset">
  </P>
</FORM>

</BODY>
</HTML>
```

The resulting file ('food.txt') may look somewhat as follows:

```
[Name]
Cliff Barnes

[Food]
Cheese

[Drink]
Beer
```

Note: In order for this example to work, the HTML-file must be named 'test.shtm'.

8.3.4 SsiOutput()

This command temporarily modifies the SSI output of the following command function.

Syntax:

```
<?--#exec cmd_argument='SsiOutput("success", "failure")'-->
```

success- String to use in case of success

failure - String to use in case of failure

Default Output:

(this command produces no output on it's own)

Example:

The following example illustrates how to use this command.

```
<?--#exec cmd_argument='SsiOutput ("Parameter stored", "Error")'-->
<?--#exec cmd_argument='SetConfigItem("File.cfg", Overwrite)'-->
```

See also...

- “SSI Output Configuration” on page 58

8.3.5 DisplayRemoteUser

This command stores returns the username on an authentication session.

Note: This command cannot be used in e-mail messages.

Syntax:

```
<?--#exec cmd_argument='DisplayRemoteUser'-->
```

Default Output:

Scenario	Default Output
Success	(current user)

8.3.6 ChangeLanguage()

This command changes the language setting based on an HTML form object.

Note: This command cannot be used in e-mail messages.

Syntax:

```
<?--#exec cmd_argument='ChangeLanguage( "source" )'-->
```

source -Name of form object which contains the new language setting.

The passed value must be a single digit as follows:

Form value	Language
"0"	English
"1"	German
"2"	Spanish
"3"	Italian
"4"	French

Default Output:

Scenario	Default Output
Success	"Language changed"
Error	"Failed to change language"

Example:

The following example demonstrates how to use this function. The resulting page sends a form to itself, which is then evaluated by the ChangeLanguage() command.

```
<HTML>
<HEAD><TITLE>ChangeLanguage Test</TITLE></HEAD>
<BODY>

<?--#exec cmd_argument='ChangeLanguage("lang")'-->

<FORM action="test.shtm">
  <P>
    <LABEL for="lang">Language (0-4) : </LABEL><BR>
    <INPUT type="text" name="lang"><BR><BR>

    <INPUT type="submit" name="_submit">
  </P>
</FORM>

</BODY>
</HTML>
```

Note: In order for this example to work, the HTML-file must be named 'test.shtm'.

8.3.7 IncludeFile()

This command includes the content of a file. Note that the content is not scanned for SSI.

Syntax:

```
<?--#exec cmd_argument='IncludeFile("filename" [, separator])'-->
```

filename- Source file

separator- Optional; specifies line separation characters (e.g. "
").

Default Output:

Scenario	Default Output
Success	<i>(file contents)</i>
Authentication Error	"Authentication error "
File open error	"Failed to open file "filename" "

Example:

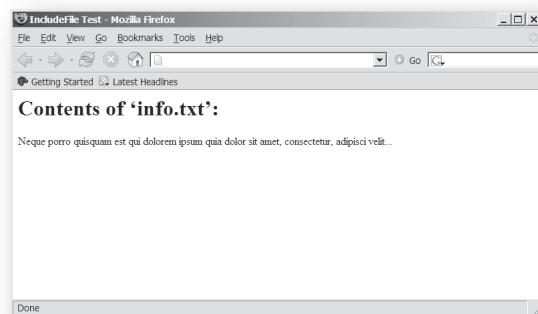
The following example demonstrates how to use this function.

```
<HTML>
<HEAD><TITLE>IncludeFile Test</TITLE></HEAD>
<BODY>
  <H1> Contents of 'info.txt':</H1>
  <P>
    <?--#exec cmd_argument='IncludeFile("info.txt")'-->.
  </P>
</BODY>
</HTML>
```

Contents of 'info.txt':

```
Neque porro quisquam est qui dolorem ipsum quia dolor sit amet,
consectetur, adipisci velit...
```

When viewed in a browser, the resulting page should look somewhat as follows:



See also...

- "Include File" on page 41

8.3.8 SaveDataToFile()

This command stores data from an HTML-form as a file in the file system. Content from the different form objects are separated by a blank line (2*CRLF).

Note 1: This command cannot be used in e-mail messages.

Note 2: The power to the module must not be recycled during the execution of this command. As there is no indication to confirm that the function has been fully executed, the function has to be used with care to avoid corruption of the file system.

Syntax:

```
<?--#exec cmd_argument='SaveDataToFile("filename" [, "source"],
                                         Overwrite|Append) '-->
```

filename- Destination file. If the specified file does not exist, it will be created (provided that the path is valid).

source - Optional; by specifying a form object, only data from that particular form object will be stored. Default behavior is to store data from all form objects except the ones where the name starts with underscore ('_').

Overwrite|Append- Specifies whether to overwrite or append data to existing files.

Default Output:

Scenario	Default Output
Success	"Configuration stored to "filename" "
Authentication Error	"Authentication error "
File write error	"Could not store configuration to "filename" "

Example:

The following example demonstrates how to use this function. The resulting page sends a form to itself, which is then evaluated by the SaveDataToFile command.

```
<HTML>
<HEAD><TITLE>SaveDataToFile Test</TITLE></HEAD>
<BODY>

<?--#exec cmd_argument='SaveDataToFile("\stuff.txt", "Meat", Overwrite) '-->

<FORM action="test.shtm">
  <P>
    <LABEL for="Fruit">Fruit: </LABEL><BR>
    <INPUT type="text" name="Fruit"><BR><BR>

    <LABEL for="Meat">Meat: </LABEL><BR>
    <INPUT type="text" name="Meat"><BR><BR>

    <LABEL for="Bread">Bread: </LABEL><BR>
    <INPUT type="text" name="Bread"><BR><BR>

    <INPUT type="submit" name="_submit">
  </P>
</FORM>

</BODY>
</HTML>
```

The resulting file ('stuff.txt') will contain the value specified for the form object called 'Meat'.

Note: In order for this example to work, the HTML-file must be named 'test.shtm'.

8.3.9 printf()

This function returns a formatted string which may contain data from the Anybus module and/or application. The formatting syntax used is similar to that of the standard C-function printf().

The function accepts a template string containing zero or more formatting tags, followed by a number of arguments. Each formatting tag corresponds to a single argument, and determines how that argument shall be converted to human readable form.

Syntax:

```
<?--#exec cmd_argument='printf("template" [, argument1, ..., argumentN])'-->
```

template- Template which determines how the arguments shall be represented. May contain any number of formatting tags which are substituted by subsequent arguments and formatted as requested. The number of format tags must match the number of arguments; if not, the result is undefined.

Formatting tags are written as follows:

```
%[Flags] [Width] [.Precision] [Modifier] type
```

See also...

- “Formatting Tags” on page 51

argument- Source arguments; optional parameters which specify the actual source of the data that shall be inserted in the template string. The number of arguments must match the number of formatting tags; if not, the result is undefined.

At the time of writing, the only allowed argument is ABCCMessage().

See also...

- “ABCCMessage()” on page 54

Default Output:

Scenario	Default Output
Success	(printf() result)
ABCCMessage error	ABCCMessage error string (8-57 “Errors”)

Example:

See also...

- “ABCCMessage()” on page 54
- “Example (Get_Attribute):” on page 56

Formatting Tags

- Type (Required)**

The Type-character is required and determines the basic representation as follows:

Type Character	Representation	Example
c	Single character	b
d, i	Signed decimal integer.	565
e, E	Floating-point number in exponential notation.	5.6538e2
f	Floating-point number in normal, fixed-point notation.	565.38
g, G	%e or %E is used if the exponent is less than -4 or greater than or equal to the precision; otherwise %f is used. Trailing zeroes/decimal point are not printed.	565.38
o	Unsigned octal notation	1065
s	String of characters	Text
u	Unsigned decimal integer	4242
x, X	Hexadecimal integer	4e7f
%	Literal %; no assignment is made	%

- Flags (Optional)**

Flag Character	Meaning
-	Left-justify the result within the give width (default is right justification)
+	Always include a '+' or '-' to indicate whether the number is positive or negative
(space)	If the number does not start with a '+' or '-', prefix it with a space character instead.
0 (zero)	Pad the field with zeroes instead of spaces
#	For %e, %E, and %f, forces the number to include a decimal point, even if no digits follow. For %x and %X, prefixes 0x or 0X, respectively.

- Width (Optional)**

Width	Meaning
number	Specifies the minimum number of characters to be printed. If the value to be printed is shorter than this number, the result is padded to make up the field width. The result is never truncated even if the result is larger.
*	The width is not specified in the format string, it is specified by an integer value preceding the argument that has to be formatted.

- .Precision (Optional)**

The exact meaning of this field depends on the type character:

Type Character	Meaning
d, i, o, u, x, X	Specifies the minimum no. of decimal digits to be printed. If the value to be printed is shorter than this number, the result is padded with space. Note that the result is never truncated, even if the result is larger.
e, E, f	Specifies the no. of digits to be printed after the decimal point (default is 6).
g, G	Specifies the max. no. of significant numbers to be printed.
s	Specifies the max. no. of characters to be printed
c	(no effect)

- Modifier**

Modifier Character	Meaning
h	Argument is interpreted as SINT8, SINT16, UINT8 or UINT16
l	Argument is interpreted as SINT32 or UINT32

8.3.10 scanf()

This function is very similar to the `printf()` function described earlier, except that it is used for input rather than output. The function reads a string passed from an HTML form object, parses the string as specified by a template string, and sends the resulting data to the specified argument. The formatting syntax used is similar to that of the standard C-function `scanf()`.

The function accepts a source, a template string containing zero or more formatting tags, followed by a number of arguments. Each argument corresponds to a formatting tag, which determines how the data read from the HTML form shall be interpreted prior sending it to the destination argument.

Note: This command cannot be used in email messages.

Syntax:

```
<?--#exec cmd_argument='scanf("source", "template" [,
                                argument1, ..., argumentN])'-->
```

source - Name of the HTML form object from which the string shall be extracted.

template- Template which specifies how to parse and interpret the data. May contain any number of formatting tags which determine the conversion prior to sending the data to subsequent arguments. The number of formatting tags must match the number of arguments; if not, the result is undefined.

Formatting tags are written as follows:

```
%[*] [Width] [Modifier] type
```

See also...

- “Formatting Tags” on page 53

argument- Destination argument(s) specifying where to send the interpreted data. The number of arguments must match the number of formatting tags; if not, the result is undefined.

At the time of writing, the only allowed argument is `ABCCMessage()`.

See also...

- “`ABCCMessage()`” on page 54

Default Output:

Scenario	Default Output
Success	“Success”
Parsing error	“Incorrect data format ”
Too much data for argument	“Too much data ”
ABCC Message error	ABCCMessage error string (“Errors” on page 57)

Example:

See also...

- “`ABCCMessage()`” on page 54
- “Example (Set_Attribute):” on page 56

Formatting Tags

- **Type (Required)**

The Type-character is required and determines the basic representation as follows:

Type	Input	Argument Data Type
c	Single character	CHAR
d	Accepts a signed decimal integer	SINT8 SINT16 SINT32
i	Accepts a signed or unsigned decimal integer. May be given as decimal, hexadecimal or octal, determined by the initial characters of the input data: <u>Initial Characters:Format:</u> 0x Hexadecimal 0 Octal 1... 9 Decimal	SINT8/UINT8 SINT16/UINT16 SINT32/UINT32
u	Accepts an optionally signed decimal integer.	UINT8 UINT16 UINT32
o	Accepts an optionally signed octal integer.	SINT8/UINT8 SINT16/UINT16 SINT32/UINT32
x, X	Accepts an optionally signed hexadecimal integer.	SINT8/UINT8 SINT16/UINT16 SINT32/UINT32
e, E, f, g, G	Accepts an optionally signed floating point number. The input format for floating-point numbers is a string of digits, with some optional characteristics: - It can be a signed value - It can be an exponential value, containing a decimal rational number followed by an exponent field, which consists of an 'E' or an 'e' followed by an integer.	FLOAT
n	Consumes no input; the corresponding argument is an integer into which scanf writes the number of characters read from the object input.	SINT8/UINT8 SINT16/UINT16 SINT32/UINT32
s	Accepts a sequence of non-whitespace characters	STRING
[scanset]	Accepts a sequence of non-whitespace characters from a set of expected bytes specified by the scanlist (e.g '[0123456789ABCDEF]') A literal '[' character can be specified as the first character of the set. A caret character (^) immediately following the initial '[' inverts the scanlist, i.e. allows all characters except the ones that are listed.	STRING
%	Accepts a single '%' input at this point; no assignment or conversion is done. The complete conversion specification should be '%%'.	-

- *** (Optional)**

Data is read but ignored. It is not assigned to the corresponding argument.

- **Width (Optional)**

Specifies the maximum number of characters to be read.

- **Modifier (Optional)**

Specifies a different data size.

Modifier	Meaning
h	SINT8, SINT16, UINT8 or UINT16
l	SINT32 or UINT32

8.4 Argument Functions

8.4.1 General Information

Argument functions are supplied as parameters to certain command functions.

General Syntax:

(Syntax depends on context)

Argument Functions:

Function	Description	Page
ABCCMessage()	-	54

8.4.2 ABCCMessage()

This function issues an object request towards an object in the module or in the host application.

Syntax:

```
ABCCMessage(object, instance, command, ce0, ce1,
            msgdata, c_type, r_type)
```

- object - Specifies the Destination Object
- instance- Specifies the Destination Instance
- command- Specifies the Command Number
- ce0 - Specifies CmdExt[0] for the command message
- ce1 - Specifies CmdExt[1] for the command message
- msgdata- Specifies the actual contents of the MsgData[] subfield in the command
 - Data can be supplied in direct form (format depends on c_type)
 - The keyword “ARG” is used when data is supplied by the parent command (e.g. scanf()).
- c_type - Specifies the data type in the command (msgdata)

See also...

 - “Command Data Types (c_type)” on page 55
- r_type - Specifies the data type in the response (msgdata)

See also...

 - “Response Data Types (r_type)” on page 55

Numeric input can be supplied in the following formats:

- Decimal (e.g. 50)- (no prefix)
- Octal (e.g. 043)- Prefix 0 (zero)
- Hex (e.g. 0x1f)- Prefix 0x

See also...

- “Example (Get_Attribute):” on page 56
- “Example (Set_Attribute):” on page 56

- **Command Data Types (c_type)**

For types which support arrays, the number of elements can be specified using the suffix '[n]', where 'n' specifies the number of elements. Each data element must be separated by space.

Type	Supports Arrays	Data format (as supplied in msgdata)
BOOL	Yes	1
SINT8	Yes	-25
SINT16	Yes	2345
SINT32	Yes	-2569
UINT8	Yes	245
UINT16	Yes	40000
UINT32	Yes	32
CHAR	Yes	A
STRING	No	"abcde" Note: Quotes can be included in the string if preceded by backslash('\') Example: "We usually refer to it as \"the Egg\""
FLOAT	Yes	5.6538e2
NONE	No	Command holds no data, hence no data type

- **Response Data Types (r_type)**

For types which support arrays, the number of elements can be specified using the suffix '[n]', where 'n' specifies the number of elements.

Type	Supports Arrays	Comments
BOOL	Yes	Optionally, it is possible to exchange the BOOL data with a message based on the value (true or false). In such case, the actual data type returned from the function will be STRING. Syntax: BOOL<true><false> For arrays, the format will be BOOL[n]<true><false>.
SINT8	Yes	-
SINT16	Yes	-
SINT32	Yes	-
UINT8	Yes	This type can also be used when reading ENUM data types from an object. In such case, the actual ENUM value will be returned.
UINT16	Yes	-
UINT32	Yes	-
CHAR	Yes	-
STRING	No	-
ENUM	No	When using this data type, the ABCCMessage() function will first read the ENUM value. It will then issue a 'Get Enum String'-command to retrieve the actual enumeration string. The actual data type in the response will be STRING.
FLOAT	Yes	-
NONE	No	Response holds no data, hence no data type

IMPORTANT: It is important to note that the message will be passed transparently to the addressed object. The SSI engine performs no checks for violations of the object addressing scheme, e.g. a malformed Get_Attribute request which (wrongfully) includes message data will be passed unmodified to the object, even though this is obviously wrong. Failure to observe this may cause loss of data or other undesired side effects.

Example (Get_Attribute):

This example shows how to retrieve the IP address using `printf()` and `ABCCMessage()`.

```
<?--#exec cmd_argument='printf( "%u.%u.%u.%u",
                                ABCCMessage(4,3,1,5,0,0,NONE,UINT8[4] ) )'-->
```

Variable	Value	Comments
object	4	Network Configuration Object (04h)
instance	3	Instance #3 (IP address)
command	1	Get_attribute
ce0	5	Attribute #5
ce1	0	-
msgdata	0	-
c_type	NONE	Command message holds no data
r_type	UINT8[4]	Array of 4 unsigned 8-bit integers

See also...

- 8-50 “`printf()`”

Example (Set_Attribute):

This example shows how to set the IP address using `scanf()` and `ABCCMessage()`. Note the special parameter value ‘ARG’, which instructs the module to use the passed form data (parsed by `scanf()`).

```
<?--#exec cmd_argument='scanf("IP", "%u.%u.%u.%u",
                                ABCCMessage(4,3,2,5,0,ARG,UINT8[4],NONE ) )'-->
```

Variable	Value	Comments
object	4	Network Configuration Object (04h)
instance	3	Instance #3 (IP address)
command	2	Set_attribute
ce0	5	Attribute #5
ce1	0	-
msgdata	ARG	Use data parsed by <code>scanf()</code> call
c_type	UINT8[4]	Array of 4 unsigned 8-bit integers
r_type	NONE	Response message holds no data

See also...

- “`scanf()`” on page 52

Errors

In case an object request results in an error, the error code in the response will be evaluated and translated to human readable form as follows:

Error Code	Output
0	"Unknown error"
1	"Unknown error"
2	"Invalid message format"
3	"Unsupported object"
4	"Unsupported instance"
5	"Unsupported command"
6	"Invalid CmdExt[0]"
7	"Invalid CmdExt[1]"
8	"Attribute access is not set-able"
9	"Attribute access is not get-able"
10	"Too much data in msg data field"
11	"Not enough data in msg data field"
12	"Out of range"
13	"Invalid state"
14	"Out of resources"
15	"Segmentation failure"
16	"Segmentation buffer overflow"
17... 255	"Unknown error"

See also...

- "SSI Output Configuration" on page 58

8.5 SSI Output Configuration

Optionally, the SSI output can be permanently changed by adding the file ‘\output.cfg’.

File format:

```
[ABCCMessage_X]
0:"Success string"
1:"Error string 1"
2:"Error string 2"
...
16:"Error string 16"
```

Each error code corresponds to a dedicated output string, labelled from 1 to 16.
See also...
- “Errors” on page 57

```
[GetConfigItem_X]
0:"Success string"
1:"Authentication error string"
2:"File open error string"
3:"Tag not found string"
```

Use “%s” to include the name of the file.

```
[SetConfigItem_X]
0:"Success string"
1:"Authentication error string"
2:"File open error string"
3:"File write error string"
```

Use “%s” to include the name of the file.

```
[IncludeFile_X]
0:"Success string"
1:"Authentication error string"
2:"File readS error string"
```

Use “%s” to include the name of the file.

```
[scanf_X]
0:"Success string"
1:"Parsing error string"
```

```
[ChangeLanguage_X]
0:"Success string"
1:"Change error string"
```

All content above can be included in the file multiple times changing the value ‘X’ in each tag for different languages. The module will then select the correct output string based on the language settings. If no information for the selected language is found, it will use the default SSI output.

Value of X	Language
0	English
1	German
2	Spanish
3	Italian
4	French

See also...

- “SsiOutput()” on page 46

9. SNMP Agent

9.1 General

Simple Network Management Protocol (SNMP, see RFC1157 standard) is used in network management systems to monitor network-attached devices for conditions that warrant administrative attention. A management agent is installed in the managing station, and exchanges data via get and set requests.

9.2 Management Information (MIB)

A MIB is a device data base that is accessed by an SNMP agent. The Anybus CompactCom 30 Profinet IO 2-Port module supports standardized MIBs: LLDP-MIB and MIB-II. Standardized MIBs are defined in RFC standards and contain variables that are divided into so called groups. The host application can change the values of some of the variables for the MIB-II.

9.3 MIB-II

The MIB-II of the ABCC-PRT IO 2-Port module contains the system- and interfaces group. The following tables show the variables according to the MIB-II standard (RFC1213) for monitoring the device status. The access authorizations refer to access via the SNMP protocol.

9.3.1 System Group Variables

Variable	Access authorizations	Description	Source of origin
sysDescr	Read only	Description of the device. Data type: Display-String(only printable ASCII characters). Max 255 characters. Factory default setting: "HMS Industrial Networks, Anybus CompactCom PROFINET IO 2-port"	PROFINET Object; Instance attribute 19 - System Description See 11-135 "Instance Attributes (Instance #1)"
sysObjectID	Read only	N/A. Value=0	Internal
sysUpTime	Read only	Time since last power up (in hundredths of a second)	Internal
sysContact	Read only	Identification of the contact person for the device, including contact information. Data type: Displaystring. Max 255 characters. Factory default setting: "www.anybus.com"	PROFINET Object; Instance attribute 22 - System Contact See 11-135 "Instance Attributes (Instance #1)"
sysName	Read only	Name of the device (Profinet Station name). Data type: Displaystring. Max 255 characters. Factory default setting: empty string	Network configuration object; Instance attribute 15 - Station name
sysLocation	Read only	Physical location of the device (IM Tag Location). Data type: (DisplayString). Max 22 characters. Factory default setting: 22 blanks	Network configuration object; Instance attribute 17 - IM Tag Location
sysServices	Read only	Functionality of the device. Value=74, which indicates that the device has functionality that represents layers 2(switch), 4(TCP) and 7(Application) in the OSI model.	Internal

9.3.2 Interfaces Group Variables

Access authorizations for all variables are read only with values from internal sources. The number in brackets refers to the port number (1 - Port 1, 2 - Port 2, 3 - Virtual port)

Variable	Data type	Value ^a	Description
ifNumber	integer	3	Number of network interfaces present. Constant
ifIndex(1..3)	integer	ifIndex(1) = 1 ifIndex(2) = 2 ifIndex(3) = 3	Unique value for each interface. Constant
ifDescr(1..3)	octetstring	ifDescr(1) = "port-001" ifDescr(2) = "port-002" ifDescr(3) = "port-virtual"	Information about the interface. ifDescr(1) must equal "port-001" and ifDescr(2) = "port-002" to be compatible with the STEP7 topology scanner.
ifType(1..3)	integer	6 ("Ethernet-csmacd")	Type of interface
ifMtu(1..3)	integer	1500	Size of largest datagram that can be sent/received on the interface, specified in octets
ifSpeed(1..3)	gauge	0 or 100 000 000	Data transfer rate of the Ethernet port in bits per second. The speed is only shown for ports where the link status is "up".
ifPhysAddress(1..3)	octetstring		MAC address for the ports
ifAdminStatus(1..3)	integer	1 ("up")	Desired state of the Ethernet port
ifOperStatus(1..3)	integer	1 ("up") or 2 ("down")	Current operating state of the Ethernet port. (Link = "up", No link = "down").
ifLastChange(1..3)	timeticks	Time when state changed, except ifLastChange(3) = 0	Time (since start-up) when the port changed to its current state, see previous variable. Indicated in multiples of hundredths of a second
ifInOctets(1..3)	counter	ifInOctets(1..3) = Number of octets	Total number of octets received on the interface, including framing characters

Variable	Data type	Value ^a	Description
ifInUcastPkts(1..3)	counter	ifInUcastPkts(1..3) = Number of unicast packets	Number of subnetwork-unicast packets delivered to a higher-layer protocol
ifInNUcastPkts(1..3)	counter	ifInNUcastPkts(1..3) = Number of non-unicast packets	Number of non-unicast (i.e. subnetwork-broadcast or subnetwork-multicast) delivered to a higher-layer protocol.
ifInDiscards(1..3)	counter	ifInDiscards(1..3) = number of discarded packets	Number of inbound packets which were discarded, without any error detected, not to be delivered to a higher-layer protocol. (One reason to discard packages might be to free up buffer space)
ifInErrors(1..3)	counter	ifInErrors(1..3) = number of error packets	Number of inbound packets with errors
ifInUnknownProtos(1..3)	counter	ifInUnknownProtos(1..3) = Number of unknown packets	Number of packets received via the interface, discarded because of an unknown or unsupported protocol.
ifOutOctets(1..3)	counter	ifOutOctets(1..3) = Number of octets	Total number of octets transmitted out from the interface, including framing characters
ifOutUcastPkts(1..3)	counter	ifOutUcastPkts(1..3) = Number of unicast packets	Total number of packets that higher-level protocols requested to be transmitted to a subnetwork-unicast address, including those that were discarded or not sent.
ifOutNUcastPkts(1..3)	counter	ifOutNUcastPkts(1..3) = Number of non-unicast packets	Total number of packets that higher-level protocols requested be transmitted to a non-unicast (i.e. a subnetwork-broadcast or subnetwork-multicast) address, including those that were discarded or not sent.
ifOutDiscards(1..3)	counter	ifOutDiscards(1..3) = Number of discarded packets	Number of outbound packets which were discarded without any error detected, not to be transmitted. (One reason to discard packages might be to free up buffer space)
ifOutErrors(1..3)	counter	ifOutErrors(1..3) = Number of error packets	Number of outbound packets that could not be transmitted due to errors
ifOutQLen(1..3)	gauge	ifOutQLen(1..3) = Number of packets in queue	Length of the output packet queue (in packets),
ifSpecific(1..3)	objid	.0.0	Reference to MIB definitions specific to the particular media being used to realize the interface. Here no reference is available, so a fixed value is used for all ports.

a. If nothing else is specified, the value of a variable is 0

10. Anybus Module Objects

10.1 General Information

This chapter specifies the Anybus Module Object implementation in the module.

Standard Objects:

- “Anybus Object (01h)” on page 63
- “Diagnostic Object (02h)” on page 64
- “Network Object (03h)” on page 67
- “Network Configuration Object (04h)” on page 68

Network Specific Objects:

- “Socket Interface Object (07h)” on page 94
- “SMTP Client Object (09h)” on page 111
- “File System Interface Object (0Ah)” on page 116
- “Network Ethernet Object (0Ch)” on page 123
- “Network PROFINET IO Object (0Eh)” on page 77
- “PROFINET Additional Diagnostic Object (0Fh)” on page 89
- “Functional Safety Module Object (11h)” on page 130

10.2 Anybus Object (01h)

Category

Basic

Object Description

This object assembles all common Anybus data, and is described thoroughly in the general Anybus CompactCom 30 Software Design Guide.

Supported Commands

Object: Get_Attribute
 Instance: Get_Attribute
 Set_Attribute
 Get_Enum_String

Object Attributes (Instance #0)

(Consult the general Anybus CompactCom 30 Software Design Guide for further information.)

Instance Attributes (Instance #1)

Basic

#	Name	Access	Type	Value
1	Module type	Get	UINT16	0401h (Standard Anybus CompactCom 30)
2... 11	-	-	-	Consult the general Anybus CompactCom 30 Software Design Guide for further information.
12	LED colors	Get	struct of: UINT8(LED1A) UINT8(LED1B) UINT8(LED2A) UINT8(LED2B)	<u>Value:Color:</u> 01h Green 02h Red 01h Green 02h Red
13... 16	-	-	-	Consult the general Anybus CompactCom 30 Software Design Guide for further information.

10.3 Diagnostic Object (02h)

Category

Extended, advanced

Object Description

This object provides a standardised way of handling host application events & diagnostics, and is thoroughly described in the general Anybus CompactCom Software Design Guide.

Please note that the application can only create diagnostic instances, when the Anybus CompactCom module is in state PROCESS_ACTIVE or IDLE. Any other attempt to create a diagnostic instance will result in an error code.

Supported Commands

Object: Get_Attribute
 Create
 Delete

Instance: Get_Attribute

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1... 4	-	-	-	Consult the general Anybus CompactCom Software Design Guide for further information.
11	Max no. of instances	Get	UINT16	5+1

Instance Attributes

Extended

#	Name	Access	Type	Value
1	Severity	Get	UINT8	Consult the general Anybus CompactCom Software
2	Event Code	Get	UINT8	Design Guide for further information.

Advanced

#	Name	Access	Type	Value
3	NW specific data	Get	Array of UINT8	Optional network specific information. See also... - "Details: Network Specific Data" on page 66

Major unrecoverable events cause the module to physically disconnect itself from the network, thus preventing network participation. Other severity levels either produce a Channel Diagnostic entry/alarm or a Generic Diagnostic entry/alarm, depending on the Event Code:

Event Code	Result																																																																								
0...FEh	<p>Module issues a Channel Diagnostic entry/alarm. The Event Code will be translated and represented as the Channel Error Type as follows:</p> <table><tr><th>Code:</th><th>Event (Anybus):</th><th>Channel Error Type (PROFINET):</th></tr><tr><td>10h</td><td>Generic Error</td><td>Error</td></tr><tr><td>20h</td><td>Current</td><td>Short circuit</td></tr><tr><td>21h</td><td>Current, device input side</td><td>Short circuit</td></tr><tr><td>22h</td><td>Current, inside the device</td><td>Short circuit</td></tr><tr><td>23h</td><td>Current, device output side</td><td>Short circuit</td></tr><tr><td>30h</td><td>Voltage</td><td>Over voltage</td></tr><tr><td>31h</td><td>Mains Voltage</td><td>Over voltage</td></tr><tr><td>32h</td><td>Voltage inside the device</td><td>Over voltage</td></tr><tr><td>33h</td><td>Output Voltage</td><td>Over voltage</td></tr><tr><td>40h</td><td>Temperature</td><td>Over temperature</td></tr><tr><td>41h</td><td>Ambient Temperature</td><td>Over temperature</td></tr><tr><td>42h</td><td>Device Temperature</td><td>Over temperature</td></tr><tr><td>50h</td><td>Device Hardware</td><td>Error</td></tr><tr><td>60h</td><td>Device Software</td><td>Error</td></tr><tr><td>61h</td><td>Internal Software</td><td>Error</td></tr><tr><td>62h</td><td>User Software</td><td>Error</td></tr><tr><td>63h</td><td>Data Set</td><td>Error</td></tr><tr><td>70h</td><td>Additional Modules</td><td>Error</td></tr><tr><td>80h</td><td>Monitoring</td><td>Error</td></tr><tr><td>81h</td><td>Communication</td><td>Error</td></tr><tr><td>82h</td><td>Protocol Error</td><td>Error</td></tr><tr><td>90h</td><td>External Error</td><td>Error</td></tr><tr><td>F0h</td><td>Additional Functions</td><td>Error</td></tr></table>	Code:	Event (Anybus):	Channel Error Type (PROFINET):	10h	Generic Error	Error	20h	Current	Short circuit	21h	Current, device input side	Short circuit	22h	Current, inside the device	Short circuit	23h	Current, device output side	Short circuit	30h	Voltage	Over voltage	31h	Mains Voltage	Over voltage	32h	Voltage inside the device	Over voltage	33h	Output Voltage	Over voltage	40h	Temperature	Over temperature	41h	Ambient Temperature	Over temperature	42h	Device Temperature	Over temperature	50h	Device Hardware	Error	60h	Device Software	Error	61h	Internal Software	Error	62h	User Software	Error	63h	Data Set	Error	70h	Additional Modules	Error	80h	Monitoring	Error	81h	Communication	Error	82h	Protocol Error	Error	90h	External Error	Error	F0h	Additional Functions	Error
Code:	Event (Anybus):	Channel Error Type (PROFINET):																																																																							
10h	Generic Error	Error																																																																							
20h	Current	Short circuit																																																																							
21h	Current, device input side	Short circuit																																																																							
22h	Current, inside the device	Short circuit																																																																							
23h	Current, device output side	Short circuit																																																																							
30h	Voltage	Over voltage																																																																							
31h	Mains Voltage	Over voltage																																																																							
32h	Voltage inside the device	Over voltage																																																																							
33h	Output Voltage	Over voltage																																																																							
40h	Temperature	Over temperature																																																																							
41h	Ambient Temperature	Over temperature																																																																							
42h	Device Temperature	Over temperature																																																																							
50h	Device Hardware	Error																																																																							
60h	Device Software	Error																																																																							
61h	Internal Software	Error																																																																							
62h	User Software	Error																																																																							
63h	Data Set	Error																																																																							
70h	Additional Modules	Error																																																																							
80h	Monitoring	Error																																																																							
81h	Communication	Error																																																																							
82h	Protocol Error	Error																																																																							
90h	External Error	Error																																																																							
F0h	Additional Functions	Error																																																																							
FFh	<p>Module issues a Generic Diagnostic entry/alarm based on network specific data.</p> <p>See also...</p> <ul style="list-style-type: none">- "Details: Network Specific Data" on page 66																																																																								

Details: Network Specific Data

Offset	Contents
0	API, low word, low byte
1	API, low word, high byte
2	API, high word, low byte
3	API, high word, high byte
4	Slot number, low byte
5	Slot number, high byte
6	Subslot number, low byte
7	Subslot number, high byte
8	Channel number, low byte
9	Channel number, high byte
10	Channel properties, Data type: <u>Value:Meaning:</u> 00h Shall be used if 'Channel number' equals 8000h, or if none of the types below are appropriate. 01h 1 bit 02h 2 bits 03h 4 bits 04h 8 bits 05h 16 bits 06h 32 bits 07h 64 bits
11	Channel properties, Direction: <u>Value:Meaning:</u> 00h Manufacturer specific 01h Input 02h Output 03h Input/Output
12	UserStructIdent, low byte
13	UserStructIdent, high byte
14...n	Data

Object Specific Error Codes

Code	Error
03h	API does not exist
04h	No module inserted in the specified slot
05h	No submodule inserted in the specified subslot
06h	Slot number specified is out-of-range
07h	Subslot number specified is out-of-range
08h	Failed to add the channel diagnostic entry to the PROFINET IO stack
09h	Failed to send the channel diagnostic alarm to the PROFINET IO stack
0Ah	Channel number out-of-range
0Bh	ChannelPropType out-of-range
0Ch	ChannelPropDir out-of-range
0Dh	ChannelPropAcc out-of-range
0Eh	ChannelPropMaintReq out-of-range
0Fh	ChannelPropMaintDem out-of-range
10h	UserStructIdent out-of-range
11h	ChannelErrType out-of-range
FFh	Unknown error

10.4 Network Object (03h)

Category

Basic

Object Description

For more information regarding this object, consult the general Anybus CompactCom Software Design Guide.

Supported Commands

Object: Get_Attribute
 Instance: Get_Attribute
 Set_Attribute
 Get_Enum_String
 Map_ADI_Write_Area
 Map_ADI_Read_Area

Object Attributes (Instance #0)

(Consult the general Anybus CompactCom Software Design Guide for further information.)

Instance Attributes (Instance #1)

Basic

#	Name	Access	Type	Value
1	Network type	Get	UINT16	0096h
2	Network type string	Get	Array of CHAR	'PROFINET IO 2-Port'
3	Data format	Get	ENUM	01h (MSB first)
4	Parameter data support	Get	BOOL	True
5	Write process data size	Get	UINT16	Current write process data size (in bytes) Updated on every successful Map_ADI_Write_Area ^a
6	Read process data size	Get	UINT16	Current read process data size (in bytes) Updated on every successful Map_ADI_Read_Area ^a
7	Exception Information	Get	UINT8	Additional information available if the module has entered the EXCEPTION state. <u>Value:Meaning:</u> 01h Illegal value 02h Wrong data size 03h Illegal response
8... 10	-	-	-	Consult the general Anybus CompactCom Software Design Guide for further information.

a. Consult the general Anybus CompactCom Software Design Guide for further information.

10.5 Network Configuration Object (04h)

Category

Extended, advanced

Object Description

This object holds network specific configuration parameters that may be set by the end user. A reset command (factory default) issued towards this object will result in all instances being set to their default values.

See also...

- “Communication Settings” on page 15
- “E-mail Client” on page 40

Note: Allowing the following instances to be set by the host application during start-up will inhibit the possibility to pass conformance tests.

Supported Commands

Object: Get_Attribute
 Reset

Instance: Get_Attribute
 Set_Attribute
 Get_Enum_String

Object Attributes (Instance #0)

(Consult the general Anybus CompactCom Software Design Guide for further information.)

Instance Attributes (Instance #3, IP Address)

Value is used after module reset.

Advanced

#	Name	Access	Type	Description
1	Name ^a	Get	Array of CHAR	'IP address'
2	Data type	Get	UINT8	04h (= UINT8)
3	Number of elements	Get	UINT8	04h (four elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of UINT8	Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)

a. Multilingual, see “Multilingual Strings” on page 76.

Note: This attribute should not be set by the application at every power on, as this would cause certification problems.

Instance Attributes (Instance #4, Subnet Mask)

Value is used after module reset.

Advanced

#	Name	Access	Type	Description
1	Name ^a	Get	Array of CHAR	'Subnet mask'
2	Data type	Get	UINT8	04h (= UINT8)
3	Number of elements	Get	UINT8	04h (four elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of UINT8	Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)

a. Multilingual, see "Multilingual Strings" on page 76.

Note: This attribute should not be set by the application at every power on, as this would cause certification problems.

Instance Attributes (Instance #5, Gateway)

Value is used after module reset.

Advanced

#	Name	Access	Type	Description
1	Name ^a	Get	Array of CHAR	'Gateway'
2	Data type	Get	UINT8	04h (= UINT8)
3	Number of elements	Get	UINT8	04h (four elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of UINT8	Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)

a. Multilingual, see "Multilingual Strings" on page 76.

Note: This attribute should not be set by the application at every power on, as this would cause certification problems.

Instance Attributes (Instance #6, DHCP)

Value is used after module reset.

Extended

#	Name	Access	Type	Description
1	Name ^a	Get	Array of CHAR	'DHCP'
2	Data type	Get	UINT8	08h (= ENUM)
3	Number of elements	Get	UINT8	01h (one element)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value ^a	Get/Set	ENUM	Value:Enum. String:Meaning: 00h 'Disable' DHCP disabled (default) 01h 'Enable' DHCP enabled

a. Multilingual, see "Multilingual Strings" on page 76.

Note: Do not set this unless the end user explicitly would like to turn DHCP on. Normally the PROFINET IO Controller assigns the IP address.

Instance Attributes (Instance #8, DNS1)

Advanced

This instance holds the address to the primary DNS server. Changes are valid after reset.

#	Name	Access	Type	Description
1	Name ^a	Get	Array of CHAR	'DNS1'
2	Data type	Get	UINT8	04h (= UINT8)
3	Number of elements	Get	UINT8	04h
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of UINT8	Valid range: 0.0.0.0. - 255.255.255.255 (Default =0.0.0.0)

a. Multilingual, see "Multilingual Strings" on page 76.

Instance Attributes (Instance #9, DNS2)

This instance holds the address to the secondary DNS server. Changes are valid after reset.

Advanced

#	Name	Access	Type	Description
1	Name ^a	Get	Array of CHAR	'DNS2'
2	Data type	Get	UINT8	04h (= UINT8)
3	Number of elements	Get	UINT8	04h
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of UINT8	Valid range: 0.0.0.0. - 255.255.255.255 (Default =0.0.0.0)

a. Multilingual, see "Multilingual Strings" on page 76.

Instance Attributes (Instance #10, Host name)

This instance holds the host name of the module. Changes are valid after reset.

Advanced

#	Name	Access	Type	Description
1	Name ^a	Get	Array of CHAR	'Host name'
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	40h
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of CHAR	Host name, 64 characters (pad with space to full length)

a. Multilingual, see "Multilingual Strings" on page 76.

Instance Attributes (Instance #11, Domain name)

This instance holds the domain name. Changes are valid after reset.

Advanced

#	Name	Access	Type	Description
1	Name ^a	Get	Array of CHAR	'Domain name'
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	30h
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of CHAR	Domain name, 48 characters (pad with space to full length)

a. Multilingual, see "Multilingual Strings" on page 76.

Instance Attributes (Instance #12, SMTP Server)

This instance holds the SMTP server address. Changes are valid after reset.

Advanced

#	Name	Access	Type	Description
1	Name ^a	Get	Array of CHAR	'SMTP Server'
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	40h
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of UINT8	SMTP server address, 64 characters (pad with space to full length)

a. Multilingual, see "Multilingual Strings" on page 76.

Instance Attributes (Instance #13, SMTP User)

This instance holds user name for the SMTP account. Changes are valid after reset.

Advanced

#	Name	Access	Type	Description
1	Name ^a	Get	Array of CHAR	'SMTP User'
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	40h
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of UINT8	SMTP account user name, 64 characters (pad with space to full length)

a. Multilingual, see "Multilingual Strings" on page 76.

Instance Attributes (Instance #14, SMTP Password)

This instance holds the password for the SMTP account. Changes are valid after reset.

Advanced

#	Name	Access	Type	Description
1	Name ^a	Get	Array of CHAR	'SMTP Pswd'
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	40h
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of UINT8	SMTP account password, 64 characters (pad with space to full length)

a. Multilingual, see "Multilingual Strings" on page 76.

Instance Attributes (Instance #15, Station Name)

The Station Name identifies the Anybus module on PROFINET. If this value is changed by the host application during runtime, a reset is required in order for changes to have effect. Changes made through DCP will have immediate effect, however.

The Station Name field shall be coded as data type CHAR with 1 to 240 characters. The definition of RFC 5890 and the following syntax applies:

- 1 or more labels, separated by [-]
- Total length is 1 to 240
- Label length is 1 to 63
- Labels consist of [a-z, 0-9, -]
- Labels do not start with [-]
- Labels do not end with [-]
- The first label must not have the form “port-xyz” or “port-xyz-abcde”, where a, b, c, d, e, x, y, z = 0...9, to avoid similarity with the field AliasNameValue
- Station names must not have the form n.n.n.n, where n = 0...999

Note: Be sure to verify that the Station Name parameter value is correct, according to the criteria above. No verification checks will be made by the module, until after the application has issued “setup complete”. A faulty Station Name will then be discarded (set to an empty string) without any warning.

Extended

#	Name	Access	Type	Description
1	Name ^a	Get	Array of CHAR	'Station name'
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get/Set	UINT8	F0h
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of UINT8	Station name

a. Multilingual, see “Multilingual Strings” on page 76.

Note 1: This attribute shall normally not be set by the application. The station name is normally set by the end user via the network. The host application shall use this attribute when the end user has the possibility to edit the station name through the application, and chooses to do so.

Note 2: This attribute should not be set by the application at every power on, as this would cause certification problems.

Instance Attributes (Instance #16, Function Tag)

This instance holds the I&M1 parameter 'Function Tag' for slot 0, subslot 1. The value can be specified either by the host application or from the network, and is saved in nonvolatile memory.

Advanced

#	Name	Access	Type	Description
1	Name ^a	Get	Array of CHAR	'Function tag'
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	20h
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of UINT8	String describing the functionality or task of the device. (pad with space to full length, default = '')

a. Multilingual, see "Multilingual Strings" on page 76.

Note: This attribute should not be set by the application at every power on, as this would cause certification problems.

Instance Attributes (Instance #17, Location Tag)

This instance holds the I&M1 parameter 'Location Tag' for slot 0, subslot 1. The value can be specified either by the host application or from the network, and is saved in nonvolatile memory.

Advanced

#	Name	Access	Type	Description
1	Name ^a	Get	Array of CHAR	'Location tag'
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	16h
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of UINT8	String describing the location of the device. (pad with space to full length, default = '')

a. Multilingual, see "Multilingual Strings" on page 76.

Note: This attribute should not be set by the application at every power on, as this would cause certification problems.

Instance Attributes (Instance #18, Installation Date)

This instance holds the I&M2 parameter 'Installation Date' for slot 0, subslot 1. The value can be specified either by the host application or from the network, and is saved in nonvolatile memory.

Advanced

#	Name	Access	Type	Description
1	Name ^a	Get	Array of CHAR	'Install. date'
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	10h
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of UINT8	String in format 'YYYY-MM-DD hh:mm' Example: '2007-06-15 22:00' (pad with space to full length, default = ' ')

a. Multilingual, see "Multilingual Strings" on page 76.

Note: This attribute should not be set by the application at every power on, as this would cause certification problems.

Instance Attributes (Instance #19, Description)

This instance holds the I&M3 parameter 'Description' for slot 0, subslot 1. The value can be specified either by the host application or from the network, and is saved in nonvolatile memory.

Advanced

#	Name	Access	Type	Description
1	Name ^a	Get	Array of CHAR	'Description'
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	36h
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of UINT8	String describing the device; free for end-user use. (pad with space to full length, default = ' ')

a. Multilingual, see "Multilingual Strings" on page 76.

Note: This attribute should not be set by the application at every power on, as this would cause certification problems.

Instance Attributes (Instance #20, F-Address)

This instance holds the 'F-Address' used for the Safety Module as PROFIsafe address. Data written to the attribute "Value" is always saved in nonvolatile memory.

Advanced

#	Name	Access	Type	Description
1	Name ^a	Get	Array of CHAR	'F-Address'
2	Data type	Get	UINT8	05h (= UINT16)
3	Number of elements	Get	UINT8	01h
4	Descriptor	Get	UINT8	03h (read/write access)

#	Name	Access	Type	Description
5	Value	Get/Set	UINT16	F-Address set by the host application Range: 1-65534 Default: 1

a. Multilingual, see “Multilingual Strings” on page 76.

Multilingual Strings

The instance names and enumeration strings in this object are multilingual, and are translated based on the current language settings as follows:

Instance	English	German	Spanish	Italian	French
3	IP address	IP-Adresse	Dirección IP	Indirizzo IP	Adresse IP
4	Subnet mask	Subnetzmaske	Masac. subred	Sottorete	Sous-réseau
5	Gateway	Gateway	Pasarela	Gateway	Passerelle
6	DHCP	DHCP	DHCP	DHCP	DHCP
	Enable	Einschalten	Activado	Abilitato	Activé
	Disable	Ausschalten	Desactivado	Disabilitato	Désactivé
8	DNS1	DNS 1	DNS Primaria	DNS1	DNS1
9	DNS2	DNS 2	DNS Secundia.	DNS2	DNS2
10	Host name	Host name	Nombre Host	Nome Host	Nom hôte
11	Domain name	Domain name	Nobre Domain	Nome Dominio	Dom Domaine
12	SMTP Server	SMTP Server	Servidor SMTP	Server SMTP	SMTP serveur
13	SMTP User	SMTP User	Usuario SMTP	Utente SMTP	SMTP utilise.
14	SMTP Pswd	SMTP PSWD	Clave SMTP	Password SMTP	SMTP mt passe
15	Station name	Stationsname	Nom. Estacion	Nome Stazione	Nom Station
16	Function tag	Funktion	Tag Funcion	Descr. Funz.	Var. Fonction
17	Location tag	Position	Tag Locacion	Descr. Locaz.	Var. Location
18	Install. date	Install. Tag	Fecha Instal.	Data Install.	Date Install.
19	Description	Beschreibung	Descripcion	Descrizione	Description
20	F-Address	F-Adresse	Dirección-F	Indirizzo-F	F-adresse

10.6 Network PROFINET IO Object (0Eh)

General Information

Extended, advanced

Object Description

-

Supported Commands

Object: Get_Attribute
 Plug_Module (see “Command Details: Plug_Module” on page 79)
 Plug_Submodule (see “Command Details: Plug_Submodule” on page 80)
 Pull_Module (see “Command Details: Pull_Module” on page 82)
 Pull_Submodule (see “Command Details: Pull_Submodule” on page 83)
 API_Add (see “Command Details: API_Add” on page 84)
 Appl_State_Ready (see “Command Details: Appl_State_Ready” on page 85)
 AR_Abort (see “Command Details: AR_Abort” on page 86)
 Add_Safety_Module (see “Command Details: Add_Safety_Module” on page 87)

Instance: Get_Attribute

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1	Name	Get	Array of CHAR	'Network PROFINET IO'
2	Revision	Get	UINT8	01h
3	Number of instances	Get	UINT16	0001h
4	Highest instance no.	Get	UINT16	0001h

Instance Attributes (Instance #1)

Extended

#	Name	Access	Type	Value
1	OnLineTrans	Get	UINT32	Diagnostic counters; keeps track of the number of on-line transitions
2	OffLineTrans	Get	UINT32	Diagnostic counters; keeps track of the number of off-line transitions
3	Last OffLineInd ReasonCode	Get	UINT16	Reason code for most recent Off-line indication See also... - "Command Details: AR_Offline_Ind" on page 152
4	Last AbortInd ReasonCode	Get	UINT16	Reason code for most recent Abort indication See also... - "Command Details: AR_Offline_Ind" on page 152
5	AddedApis	Get	UINT16	Returns the number of APIs added (including API 0)
6	ApiList	Get	Array of UINT32	First element will always be zero and the second element will contain an additional API. Length of the array is determined by parameter "AddedApis".
7	EstablishedArs	Get	UINT16	The number of Application Relationships currently established
8	ArList	Get	Array of UINT16	Array of Application Relationship handles. Length of array is determined by parameter "EstablishedArs".
9	ProfinetIoStack Init ErrorCode	Get	UINT16	If the PROFINET IO stack, for some reason, would not acknowledge the current configuration the returned error code can be read with this attribute.
10	Port 1 MAC Address	Get	Array of UINT8	6 Byte PROFINET Port 1 MAC address See also... - "Ethernet Host Object (F9h)" on page 158
11	Port 2 MAC Address	Get	Array of UINT8	6 Byte PROFINET Port 2 MAC address See also... - "Ethernet Host Object (F9h)" on page 158

Command Details: Plug_Module

Category

Advanced

Details

Command Code.: 10h

Valid for: Object Instance

Description

This command may be called during start-up to specify the Real Identification. It may also be called during runtime in case there are changes to the Real Identification. In such case, the Anybus will automatically issue a 'Plug'-alarm to the IO Controller.

Note: It is only permitted to issue this command if 'API_Add' has been issued first.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	(reserved, set to zero)	(reserved for future use)
CmdExt[1]		
Data[0]	API (low word, low byte)	Application Process Instance (API) See also... - "Application Process Instances (API)" on page 22
Data[1]	API (low word, high byte)	
Data[2]	API (high word, low byte)	
Data[3]	API (high word, high byte)	
Data[4]	SlotNr (low byte)	Destination slot for module.
Data[5]	SlotNr (high byte)	Range: 0... N(defined in API_Add)
Data[6]	ModIdent (low word, low byte)	Module identified as state in the GSD-file
Data[7]	ModIdent (low word, high byte)	
Data[8]	ModIdent (high word, low byte)	
Data[9]	ModIdent (high word, high byte)	

- **Response Details**

See "Object Specific Error Codes" on page 88.

See also...

- "Real Identification (RI)" on page 23 ("Configuration Mismatch" on page 24)
- "Command Details: Plug_Submodule" on page 80
- "Command Details: Pull_Module" on page 82
- "Command Details: Pull_Submodule" on page 83
- "Command Details: API_Add" on page 84
- "Flowchart - Establishment of Real Identification (RI)" on page 174

Command Details: Plug_Submodule

Category

Advanced

Details

Command Code.: 11h

Valid for: Object Instance

Description

This command may be called during start-up to specify the Real Identification. It may also be called during runtime in case there are changes to the Real Identification. In such case, the Anybus will automatically issue a 'Return of Submodule'-alarm to the IO Controller.

A submodule plugged with this command can hold IO data to the master, from the master or data in both directions. It is also possible to plug submodules which do not carry any data at all.

The Anybus supports up to 128 submodules in total.

Note 1: In case the slot number in the command is set to 0 (zero), the ADI number must also be 0 (zero), since slot 0 cannot hold any actual data.

Note 2: It is only permitted to issue this command if 'API_Add' has been issued first.

Note 3: The 'Interface'- and 'Port'- submodules have to be plugged in order to pass certification tests.

Note 4: The interface and port submodule can only be plugged during the SETUP-state. Any attempt to plug these submodules during runtime will result in error.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	(reserved, set to zero)	(reserved for future use)
CmdExt[1]		
Data[0]	ADI number (low byte), Read	Reference to the ADI previously mapped with Map_ADI_Read_Area.
Data[1]	ADI number (high byte), Read	
Data[2]	ADI element, Read	Reference to the element of the ADI mapped with Map_ADI_Read_Area for the specified SlotNr (See Data[10... 11]). Range: 1... 255ADI element associated with the submodule 0 Entire ADI is associated with the submodule
Data[3]	ADI number (low byte), Write	Reference to the ADI previously mapped with Map_ADI_Write_Area.
Data[4]	ADI number (high byte), Write	
Data[5]	ADI element, Read	Reference to the element of the ADI mapped with Map_ADI_Write_Area for the specified SlotNr (See Data[10... 11]). Range: 1... 255ADI element associated with the submodule 0 Entire ADI is associated with the submodule
Data[6]	API (low word, low byte)	Application Process Instance (API) See also... - "Application Process Instances (API)" on page 22
Data[7]	API (low word, high byte)	
Data[8]	API (high word, low byte)	
Data[9]	API (high word, high byte)	
Data[10]	SlotNr (low byte)	Destination slot for submodule.
Data[11]	SlotNr (high byte)	Range: 0... N(defined in API_Add)
Data[12]	SubSlotNr (low byte)	Destination subslot for submodule. Range:
Data[13]	SubSlotNr (high byte)	0... P(defined in API_Add) 8000... 8002h(slot 0 only)
Data[14]	SubModIdent (low word, low byte)	Module identified as stated in the GSD-file
Data[15]	SubModIdent (low word, high byte)	
Data[16]	SubModIdent (high word, low byte)	
Data[17]	SubModIdent (high word, high byte)	

- **Response Details**

See "Object Specific Error Codes" on page 88.

See also...

- "Real Identification (RI)" on page 23 ("Configuration Mismatch" on page 24)
- "Command Details: Plug_Submodule_Failed" on page 156
- "Command Details: Plug_Module" on page 79
- "Command Details: Pull_Module" on page 82
- "Command Details: Pull_Submodule" on page 83
- "Command Details: API_Add" on page 84
- "Flowchart - Establishment of Real Identification (RI)" on page 174

Command Details: Pull_Module

Category

Advanced

Details

Command Code.: 12h

Valid for: Object Instance

Description

This command removes a module from the configuration. Can be issued at any time.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	(reserved, set to zero)	(reserved for future use)
CmdExt[1]		
Data[0]	API (low word, low byte)	Application Process Instance (API) See also... - "Application Process Instances (API)" on page 22
Data[1]	API (low word, high byte)	
Data[2]	API (high word, low byte)	
Data[3]	API (high word, high byte)	
Data[4]	SlotNr (low byte)	Slot number of module.
Data[5]	SlotNr (high byte)	Range: 0... N(defined in API_Add)

- **Response Details**

See "Object Specific Error Codes" on page 88.

See also...

- "Command Details: Plug_Module" on page 79
- "Command Details: Pull_Submodule" on page 83

Command Details: Pull_Submodule

Category

Advanced

Details

Command Code.: 13h

Valid for: Object Instance

Description

This command removes a submodule from the configuration. Can be issued at any time.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	(reserved, set to zero)	(reserved for future use)
CmdExt[1]		
Data[0]	API (low word, low byte)	Application Process Instance (API) See also... - "Application Process Instances (API)" on page 22
Data[1]	API (low word, high byte)	
Data[2]	API (high word, low byte)	
Data[3]	API (high word, high byte)	
Data[4]	SlotNr (low byte)	Slot number of submodule.
Data[5]	SlotNr (high byte)	Range: 0... N(defined in API_Add)
Data[6]	SubSlotNr (low byte)	Subslot number of submodule. Range: 0... P(defined in API_Add) 8000... 8002h(slot 0 only)
Data[7]	SubSlotNr (high byte)	

- **Response Details**

See "Object Specific Error Codes" on page 88.

See also...

- "Command Details: Plug_Submodule" on page 80
- "Command Details: Pull_Module" on page 82

Command Details: API_Add

Category

Advanced

Details

Command Code.: 14h

Valid for: Object Instance

Description

By default, the module only supports API 0 (zero). If additional APIs are to be supported, or if the host application shall handle plugging/unplugging of modules and submodules, this command must be used to specify the API implementation. Note that if using this command, it is mandatory to declare API 0 (zero) prior to defining other APIs or plugging/unplugging modules/submodules. API numbers are assigned by (PROFIBUS & PROFINET International (PI)).

Note 1: This command may only be issued prior to setting the ‘Setup Complete’-attribute in the Anybus Object.

Note 2: This command clears the generic (default) Real Identification created by the Anybus module while mapping ADIs to Process Data. Therefore, issuing this command effectively makes it mandatory to specify the actual Real Identification by means of the ‘Plug_Module’ and ‘Plug_Submodule’-commands.

Note 3: When this command has been issued, any attempt to map ADIs to Process Data will result in an error (‘Invalid State’).

• Command Details

Field	Contents	Comments
CmdExt[0]	(reserved, set to zero)	(reserved for future use)
CmdExt[1]		
Data[0]	API (low word, low byte)	Application Process Instance (API) See also... - “Application Process Instances (API)” on page 22
Data[1]	API (low word, high byte)	
Data[2]	API (high word, low byte)	
Data[3]	API (high word, high byte)	
Data[4]	MaxSlots (low byte)	Max. number of slots (MNS) for the API.
Data[5]	MaxSlots (high byte)	Range: 0... 64
Data[6]	MaxSubSlots (low byte)	Max. number of subslots per slot for the API. Range: 0... 8
Data[7]	MaxSubSlots (high byte)	

• Response Details

See “Object Specific Error Codes” on page 88.

See also...

- “Application Process Instances (API)” on page 22
- “Flowchart - Establishment of Real Identification (RI)” on page 174

Command Details: Appl_State_Ready

Category

Advanced

Details

Command Code.: 15h

Valid for: Object Instance

Description

This command is only applicable if the host application implements support for 'End_Of_Prm_Ind', and signals to the module (and in turn the I/O Controller) that the host application is ready for data exchange.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	AR handle (low byte)	Handle for the Application Relationship (AR) (see "Application Process Instances (API)" on page 22)
CmdExt[1]	AR handle (high byte)	

- **Response Details**

See "Object Specific Error Codes" on page 88.

See also...

- "Application Relationships (AR)" on page 22
- "Command Details: End_Of_Prm_Ind" on page 151

Command Details: AR_Abort

Category

Advanced

Details

Command Code.: 16h

Valid for: Object Instance

Description

This command indicates to the ABCC-PRT that the current application relationship shall be aborted, and that the ABCC-PRT shall go off-line.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	AR handle (low byte)	Handle for the Application Relationship (AR) (see "Application Process Instances (API)" on page 22)
CmdExt[1]	AR handle (high byte)	

- **Response Details**

See "Object Specific Error Codes" on page 88.

See also...

- "Application Relationships (AR)" on page 22
- "Command Details: AR_Check_Ind" on page 146
- "Command Details: AR_Info_Ind" on page 149
- "Command Details: AR_Offline_Ind" on page 152
- "Command Details: AR_Abort_Ind" on page 154

Command Details: Add_Safety_Module

Category

Extended

Details

Command Code.: 17h

Valid for: Object Instance

Description

This command may be called during start-up to specify the slot number where the Safety Module shall be located. In addition to the slot number, the 16 most significant bits of the 32 bit module identification number for the Safety Module are specified. The 16 least significant bits are specified by the Safety Module itself. The command shall be called after successful enabling of the Safety Module.

Note: It is only permitted to issue this command if 'API_Add' has been issued first.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	(reserved, set to zero)	(reserved for future use)
CmdExt[1]		
Data[0]	SlotNr (low byte)	Number of the slot where to insert the Safety Module. Range: 1 to N
Data[1]	SlotNr (high byte)	
Data[2]	ModIdent_High (low byte)	Module identifier as stated in the GSD file (highest 16 bits).
Data[3]	ModIdent_High (high byte)	

- **Response Details**

See "Object Specific Error Codes" on page 88.

See also...

- "PROFINET IO Object (F6h)" on page 134

Object Specific Error Codes

Code	Meaning
01h	The ADI has not been mapped with command Map_ADI_Write_Area
02h	The ADI has not been mapped with command Map_ADI_Read_Area
03h	Element does not exist for the ADI
04h	This ADI/element is already mapped
05h	API 0 must be added first
06h	API does not exist
07h	Trying to add an API already present
08h	There is no room for any more APIs
09h	Module in slot 0 cannot have any IO data
0Ah	Prior to plugging the requested module/submodule, slot 0 must be populated with a module and a submodule (in subslot 1)
0Bh	Slot occupied
0Ch	subslot occupied
0Dh	No module inserted in the specified slot
0Eh	No submodule inserted in the specified subslot
0Fh	Slot number specified is out-of-range
10h	subslot number specified is out-of-range
11h	The AR handle provided is not valid
12h	There is no application ready pending
13h	Unknown error (PROFINET IO stack denied the request)
14h	Max number of submodules have already been plugged
15h	The Safety Module has not been enabled.

10.7 PROFINET Additional Diagnostic Object (0Fh)

Category

Extended

Object Description

This object provides advanced PROFINET-specific diagnostic support. Each instance in this object corresponds to a diagnostic entry on the network.

Please note that the application can only create diagnostic instances, when the Anybus CompactCom module is in state PROCESS_ACTIVE or IDLE. Any other attempt to create a diagnostic instance will result in an error code.

Supported Commands

Object:	Get Attribute
	Set Attribute
	Create (See “Command Details: Create” on page 90)
	Delete (See “Command Details: Delete” on page 91)
	Process_Alarm (See “Command Details: Process_Alarm” on page 92)
Instance:	Get Attribute

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1	Name	Get	Array of CHAR	'Additional diagnostic PROFINET IO'
2	Revision	Get	UINT8	02h
3	Number of instances	Get	UINT16	Corresponds to no. of pending events
4	Highest instance no.	Get	UINT16	Highest created instance no.
1	Max.no of instances	Get	UINT16	6

Instance Attributes (Instance #1... n)

Extended

#	Name	Access	Data Type	Description
1	Type	Get	UINT8	Type of instance: <u>Value:Meaning:</u> 00h Channel Diagnostics (other) Reserved for future use

Command Details: Create

Category

Extended

Details

Command Code: 03h

Valid for: Object

Description

This command creates a channel diagnostic entry and causes the module to issue a channel diagnostic alarm. It is only possible to create a new instance when the Anybus CompactCom module is in the PRO-CESS_ACTIVE state. If the command is executed when the module is in any other state, an error will be returned (0Dh, "The command is not supported in the current state").

- **Command Details**

Field	Contents	Comments
CmdExt[0]	(reserved, set to zero)	(reserved for future use)
CmdExt[1]	SubCommand specifier	Type of instance to create: <u>Value:</u> <u>Meaning:</u> 0: Channel Diagnostics (other) (reserved for future use)
Data[0]	API (low word, low byte)	Application Process Instance (API) See also... - "Application Process Instances (API)" on page 22
Data[1]	API (low word, high byte)	
Data[2]	API (high word, low byte)	
Data[3]	API (high word, high byte)	
Data[4]	SlotNr (low byte)	Slot number associated with the diagnostic entry
Data[5]	SlotNr (high byte)	
Data[6]	SubSlotNr (low byte)	Subslot number associated with the diagnostic entry
Data[7]	SubSlotNr (high byte)	
Data[8]	ChannelNr (low byte)	Channel number: <u>Value:</u> <u>Meaning:</u> 0...7FFFh: Manufacturer specific 8000h Refers to the submodule itself, not a specific channel
Data[9]	ChannelNr (high byte)	
Data[10]	ChannelPropType	Channel type <u>Value:</u> <u>Meaning:</u> 0: Other 1: 1-bit 2: 2-bit 3: 4-bit 4: Byte 5: Word (2 bytes) 6: Double Word (4 bytes) 7: Long Word (8 bytes)
Data[11]	ChannelPropDir	Channel direction: <u>Value:</u> <u>Meaning:</u> 0: Manufacturer specific 1: Input (data to I/O controller) 2: Output (data from I/O controller) 3: Input/Output (data to/from I/O controller)

Field	Contents	Comments
Data[12]	ChannelPropAcc	(currently not supported, set to zero)
Data[13]	ChannelPropMaintReq	
Data[14]	ChannelPropMaintDem	
Data[15]	ChannelErrorType (low byte)	Channel error type <u>Value:</u> <u>Meaning:</u> 1: Short circuit 2: Under voltage 3: Over voltage 4: Overload 5: Over temperature 6: Line break 7: Upper limit value exceeded 8: Lower limit value exceeded 9: Error (other) (consult the PROFINET IO specification)
Data[16]	ChannelErrorType (high byte)	

- **Response Details (Success)**

Field	Contents	Comments
Data[0]	Instance number (low byte)	The number of the created instance
Data[1]	Instance number (high byte)	

See also...

- Anybus CompactCom Software Design Guide (Command Details: Create, Diagnostic Object)

Command Details: Delete

Category

Extended

Details

Command Code: 04h

Valid for: Instance

Description

This command deletes a previously created diagnostic event / instance.

- **Command Details**

Field	Contents
CmdExt[0]	The number of the instance to delete (low byte)
CmdExt[1]	The number of the instance to delete (high byte)

- **Response Details**

-

Command Details: Process_Alarm

Category

Extended

Details

Command Code: 10h

Valid for: Object

Description

This command issues a Process Alarm on the network. No instance is created, and consequently no 'delete'-can be issued to remove it.

- Command Details

Field	Contents	Comments
CmdExt[0]	(reserved, set to zero)	(reserved for future use)
CmdExt[1]		
Data[0]	API (low word, low byte)	Application Process Instance (API) See also... - "Application Process Instances (API)" on page 22
Data[1]	API (low word, high byte)	
Data[2]	API (high word, low byte)	
Data[3]	API (high word, high byte)	
Data[4]	SlotNr (low byte)	Slot number associated with the diagnostic entry
Data[5]	SlotNr (high byte)	
Data[6]	SubSlotNr (low byte)	subslot number associated with the diagnostic entry
Data[7]	SubSlotNr (high byte)	
Data[8]	UserStructIdent (low byte)	Specifies the nature of the passed alarm data <u>Value:</u> <u>Meaning:</u> 0...7FFFh: Manufacturer specific 8000h: Channel diagnostic data 8001h: Multiple manufacturer specific (other) (reserved)
Data[9]	UserStructIdent (high byte)	
Data[10...]	Data	Data to pass with the alarm

- Response Details

-

Object Specific Error Codes

Code	Error
03h	API does not exist
04h	No module inserted in the specified slot
05h	No submodule inserted in the specified subslot
06h	Slot number specified is out-of-range
07h	subslot number specified is out-of-range

Code	Error
08h	Failed to add the channel diagnostic entry to the PROFINET IO stack
09h	Failed to send the channel diagnostic alarm to the PROFINET IO stack
0Ah	Channel number out-of-range
0Bh	ChannelPropType out-of-range
0Ch	ChannelPropDir out-of-range
0Dh	ChannelPropAcc out-of-range
0Eh	ChannelPropMaintReq out-of-range
0Fh	ChannelPropMaintDem out-of-range
10h	UserStructIdent out-of-range
11h	ChannelErrType out-of-range
12h	Failed to remove the diagnostic entry from the PROFINET IO stack
FFh	Unknown error

10.8 Socket Interface Object (07h)

Category

Advanced

Object Description

This object provides direct access to the TCP/IP stack socket interface, enabling custom protocols to be implemented over TCP/UDP.

Note that some of the commands used when accessing this object may require segmentation. For more information, see “Message Segmentation” on page 146.

IMPORTANT: *The use of functionality provided by this object should only be attempted by users who are already familiar with socket interface programming and who fully understands the concepts involved in TCP/IP programming.*

Supported Commands

Object: Get_Attribute
 Create (See “Command Details: Create” on page 96)
 Delete (See “Command Details: Delete” on page 97)

Instance: Get_Attribute
 Set_Attribute
 Bind (See “Command Details: Bind” on page 98)
 Shutdown (See “Command Details: Shutdown” on page 99)
 Listen (See “Command Details: Listen” on page 100)
 Accept (See “Command Details: Accept” on page 101)
 Connect (See “Command Details: Connect” on page 102)
 Receive (See “Command Details: Receive” on page 103)
 Receive_From (See “Command Details: Receive_From” on page 104)
 Send (See “Command Details: Send” on page 105)
 Send_To (See “Command Details: Send_To” on page 106)
 IP_Add_membership (See “Command Details: IP_Add_Membership” on page 107)
 IP_Drop_membership (See “Command Details: IP_Drop_Membership” on page 108)
 DNS_Lookup (See “Command Details: DNS_Lookup” on page 109)

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1	Name	Get	Array of CHAR	'Socket interface'
2	Revision	Get	UINT8	01h
3	Number of instances	Get	UINT16	-
4	Highest instance no.	Get	UINT16	-
11	Max. no. of instances	Get	UINT16	0008h

Instance Attributes (Sockets #1...8)

Advanced

#	Name	Access	Type	Description
1	Socket type	Get	UINT8	<u>Value:Socket Type:</u> 00h SOCK_STREAM, NON-BLOCKING (TCP) 01h SOCK_STREAM, BLOCKING (TCP) 02h SOCK_DGRAM, NON-BLOCKING (UDP) 03h SOCK_DGRAM, BLOCKING (UDP)
2	Port	Get	UINT16	Local port that the socket is bound to
3	Host IP	Get	UINT32	Host IP address, or 0 (zero) if not connected
4	Host port	Get	UINT16	Host port number, or 0 (zero) if not connected
5	TCP State	Get	UINT8	State (TCP sockets only): <u>Value:State:Description:</u> 00h CLOSED Closed 01h LISTEN Listening for connection 02h SYN_SENT Active, have sent SYN 03h SYN_RECEIVED Have sent and received SYN 04h ESTABLISHED Established. 05h CLOSE_WAIT Received FIN, waiting for close 06h FIN_WAIT_1 Have closed, sent FIN 07h CLOSING Closed exchanged FIN; await FIN ACK 08h LAST_ACK Have FIN and close; await FIN ACK 09h FIN_WAIT_2 Have closed, FIN is acknowledged 0Ah TIME_WAIT Quiet wait after close
6	TCP RX bytes	Get	UINT16	Number of bytes in RX buffers (TCP sockets only)
7	TCP TX bytes	Get	UINT16	Number of bytes in TX buffers (TCP sockets only)
8	Reuse address	Get/Set	BOOL	Socket can reuse local address <u>Value:Meaning:</u> 1 Enabled 0 Disabled (default)
9	Keep alive	Get/Set	BOOL	Protocol probes idle connection (TCP sockets only). ^a <u>Value:Meaning:</u> 1 Enabled 0 Disabled (default)
10	IP Multicast TTL	Get/Set	UINT8	IP Multicast TTL value (UDP sockets only). Default = 1.
11	IP Multicast Loop	Get/Set	BOOL	IP multicast loop back (UDP sockets only) ^b <u>Value:Meaning:</u> 1 Enable (default) 0 Disable
12	Ack delay time	Get/Set	UINT16	Time for delayed ACKs in ms (TCP sockets only) Default = 200ms ^c
13	TCP No Delay	Get/Set	BOOL	Don't delay send to coalesce packets (TCP). <u>Value:Meaning:</u> 1 Delay (default) 0 Don't delay (turn off Nagle's algorithm on socket)
14	TCP Connect Timeout	Get/Set	UINT16	TCP Connect timeout in seconds (default = 75s)

a. If the Keep alive attribute is set, the connection will be probed for the first time after it has been idle for 120 minutes. If a probe attempt fails, the connection will continue to be probed at intervals of 75 s. The connection is terminated after 8 failed probe attempts.

b. Must belong to group in order to get the loop backed message

c. Resolution is 50ms, i.e. 50...99 = 50ms, 100...149 = 100ms, 199 = 150ms etc.

Command Details: Create

Category

Advanced

Details

Command Code.: 03h

Valid for: Object Instance

Description

This command creates a socket.

Note: This command is only allowed in WAIT_PROCESS, IDLE and PROCESS_ACTIVE states.

- **Command Details**

Field	Contents
CmdExt[0]	(reserved, set to zero)
CmdExt[1]	<u>Value:Socket Type:</u> 00h SOCK_STREAM, NON-BLOCKING (TCP) 01h SOCK_STREAM, BLOCKING (TCP) 02h SOCK_DGRAM, NON-BLOCKING (UDP) 03h SOCK_DGRAM, BLOCKING (UDP)

- **Response Details**

Field	Contents	Comments
Data[0]	Instance number (low)	Instance number of the created socket.
Data[1]	Instance number (high)	

Command Details: Delete

Category

Advanced

Details

Command Code.: 04h

Valid for: Object Instance

Description

This command deletes a previously created socket and closes the connection (if connected).

- If the socket is of TCP-type and a connection is established, the connection is terminated with the RST-flag.
- To gracefully terminate a TCP-connection, it is recommended to use the ‘Shutdown’-command (see “Command Details: Shutdown” on page 99) before deleting the socket, causing the connection to be closed with the FIN-flag instead.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	Instance number to delete (low)	Instance number of socket that shall be deleted.
CmdExt[1]	Instance number to delete (high)	

- **Response Details**

(no data)

Command Details: Bind

Category

Advanced

Details

Command Code.: 10h

Valid for: Instance

Description

This command binds a socket to a local port.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	Requested port number (low)	Set to 0 (zero) to request binding to any free port.
CmdExt[1]	Requested port number (high)	

- **Response Details**

Field	Contents	Comments
CmdExt[0]	Bound port number (low)	Actual port that the socket was bound to.
CmdExt[1]	Bound port number (high)	

Command Details: Shutdown

Category

Advanced

Details

Command Code.: 11h

Valid for: Instance

Description

This command closes a TCP-connection using the FIN-flag. Note that the response does not indicate if the connection actually shut down, which means that this command cannot be used to poll non-blocking sockets, nor will it block for blocking sockets.

- Command Details**

Field	Contents
CmdExt[0]	(reserved, set to zero)
CmdExt[1]	<u>Value:Mode:</u> 00h Shutdown receive channel 01h Shutdown send channel 02h Shutdown both receive- and send channel

- Response Details**

(no data)

The recommended sequence to gracefully shut down a TCP connection is described below.

Application initiates shutdown:

1. Send shutdown with CmdExt[1] set to 01h. This will send FIN-flag to host shutting down the send channel, note that the receive channel will still be operational.
2. Receive data on socket until error message Object specific error (EDESTADDRREQ (14)) is received, indicating that the host closed the receive channel. If host does not close the receive channel use a timeout and progress to step 3.
3. Delete the socket instance. If step 2 timed out, RST-flag will be sent to terminate the socket.

Host initiates shutdown:

1. Receive data on socket, if zero bytes received it indicates that the host closed the receive channel of the socket.
2. Try to send any unsent data to the host.
3. Send shutdown with CmdExt[1] set to 01h. This will send FIN-flag to host shutting down the receive channel.
4. Delete the socket instance.

Command Details: Listen

Category

Advanced

Details

Command Code.: 12h

Valid for: Instance

Description

This command puts a TCP socket in listening state. Backlog queue length is the number of unaccepted connections allowed on the socket. When backlog queue is full, further connections will be refused with RST-flag.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Value:Backlog queue length: 00h 1 01h 2 02h 4	-

- **Response Details**

(no data)

Command Details: Accept

Category

Advanced

Details

Command Code.: 13h

Valid for: Instance

Description

This command accepts incoming connections on a listening TCP socket. A new socket instance is created for each accepted connection. The new socket is connected with the host and the response returns its instance number.

NON-BLOCKING mode:

This command must be issued repeatedly (polled) for incoming connections. If no incoming connection request exists, the module will respond with error code 0006h (EWOULDBLOCK).

BLOCKING mode:

This command will block until a connection request has been detected.

Note: This command will only be accepted if there is a free instance to use for accepted connections. For blocking connections, this command will reserve an instance.

- **Command Details**

(no data)

- **Response Details**

Field	Contents
Data[0]	Instance number for the connected socket (low)
Data[1]	Instance number for the connected socket (high)
Data[2]	Host IP address byte 3 (low)
Data[3]	Host IP address byte 2
Data[4]	Host IP address byte 1
Data[5]	Host IP address byte 0 (high)
Data[6]	Host port number (low)
Data[7]	Host port number (high)

Command Details: Connect

Category

Advanced

Details

Command Code.: 14h

Valid for: Instance

Description

For SOCK_DGRAM-sockets, this command specifies the peer with which the socket is to be associated (to which datagrams are sent and the only address from which datagrams are received).

For SOCK_STREAM-sockets, this command attempts to establish a connection to a host.

SOCK_STREAM-sockets may connect successfully only once, while SOCK_DGRAM-sockets may use this service multiple times to change their association. SOCK_DGRAM-sockets may dissolve their association by connecting to IP address 0.0.0.0, port 0 (zero).

NON-BLOCKING mode:

This command must be issued repeatedly (polled) until a connection is connected, rejected or timed out. The first connect-attempt will be accepted, thereafter the command will return error code 22 (EINPROGRESS) on poll requests while attempting to connect.

BLOCKING mode:

This command will block until a connection has been established or the connection request is cancelled due to a timeout or a connection error.

- **Command Details**

Field	Contents	Contents
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]		
Data[0]	Host IP address byte 3 (low)	-
Data[1]	Host IP address byte 2	
Data[2]	Host IP address byte 1	
Data[3]	Host IP address byte 0 (high)	
Data[4]	Host port number (low)	
Data[5]	Host port number (high)	

- **Response Details**

(no data)

Command Details: Receive

Category

Advanced

Details

Command Code.: 15h

Valid for: Instance

Description

This command receives data from a connected socket. Message segmentation may be used to receive up to 1472 bytes (see “Message Segmentation” on page 146).

For SOCK_DGRAM-sockets, the module will return the requested amount of data from the next received datagram. If the datagram is smaller than requested, the entire datagram will be returned in the response message. If the datagram is larger than requested, the excess bytes will be discarded.

For SOCK_STREAM-sockets, the module will return the requested number of bytes from the received data stream. If the actual data size is less than requested, all available data will be returned.

NON-BLOCKING mode:

If no data is available on the socket, the error code 0006h (EWOULDBLOCK) will be returned.

BLOCKING mode:

The module will not issue a response until the operation has finished.

If the module responds successfully with 0 (zero) bytes of data, it means that the host has closed the connection. The send channel may however still be valid and must be closed using ‘Shutdown’ and/or ‘Delete’.

- **Command Details**

Field	Contents	Notes
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control bits	see “Command Segmentation” on page 146
Data[0]	Receive data size (low)	Only used in the first segment
Data[1]	Receive data size (high)	

- **Response Details**

Note: The data in the response may be segmented (see “Message Segmentation” on page 146).

Field	Contents	Notes
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control bits	see “Response Segmentation” on page 147
Data[0...n]	Received data	-

Command Details: Receive_From

Category

Advanced

Details

Command Code.: 16h

Valid for: Instance

Description

This command receives data from an unconnected SOCK_DGRAM-socket. Message segmentation may be used to receive up to 1472 bytes (see “Message Segmentation” on page 146).

The module will return the requested amount of data from the next received datagram. If the datagram is smaller than requested, the entire datagram will be returned in the response message. If the datagram is larger than requested, the excess bytes will be discarded.

The response message contains the IP address and port number of the sender.

NON-BLOCKING mode:

If no data is available on the socket, the error code 0006h (EWOULDBLOCK) will be returned.

BLOCKING mode:

The module will not issue a response until the operation has finished.

- **Command Details**

Field	Contents	Notes
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control bits	see “Command Segmentation” on page 146
Data[0]	Receive data size (low)	Only used in the first segment
Data[1]	Receive data size (high)	

- **Response Details**

Note: The data in the response may be segmented (see “Message Segmentation” on page 146).

Field	Contents	Notes
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control bits	see “Response Segmentation” on page 147
Data[0]	Host IP address byte 3 (low)	The host address/port information is only included in the first segment. All data thereafter will start at Data[0]
Data[1]	Host IP address byte 2	
Data[2]	Host IP address byte 1	
Data[3]	Host IP address byte 0 (high)	
Data[4]	Host port number (low)	
Data[5]	Host port number (high)	
Data[6...n]	Received data	

Command Details: Send

Category

Advanced

Details

Command Code.: 17h

Valid for: Instance

Description

This command sends data on a connected socket. Message segmentation may be used to send up to 1472 bytes (see “Message Segmentation” on page 146).

NON-BLOCKING mode:

If there isn't enough buffer space available in the send buffers, the module will respond with error code 0006h (EWOULDBLOCK)

BLOCKING mode:

If there isn't enough buffer space available in the send buffers, the module will block until there is.

- Command Details**

Note: To allow larger amount of data (i.e. >255 bytes) to be sent, the command data may be segmented (see “Message Segmentation” on page 146).

Field	Contents	Notes
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control	see “Command Segmentation” on page 146
Data[0...n]	Data to send	-

- Response Details**

Field	Contents	Notes
CmdExt[0]	(reserved)	(ignore)
CmdExt[1]		
Data[0]	Number of sent bytes (low)	Only valid in the last segment
Data[1]	Number of sent bytes (high)	

Command Details: Send_To

Category

Advanced

Details

Command Code.: 18h

Valid for: Instance

Description

This command sends data to a specified host on an unconnected SOCK-DGRAM-socket. Message segmentation may be used to send up to 1472 bytes (see “Message Segmentation” on page 146).

- **Command Details**

Note: To allow larger amount of data (i.e. >255 bytes) to be sent, the command data may be segmented (see “Message Segmentation” on page 153).

Field	Contents	Notes
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control	see “Command Segmentation” on page 146
Data[0]	Host IP address byte 3 (low)	The host address/port information shall only be included in the first segment. All data thereafter must start at Data[0]
Data[1]	Host IP address byte 2	
Data[2]	Host IP address byte 1	
Data[3]	Host IP address byte 0 (high)	
Data[4]	Host port number (low)	
Data[5]	Host port number (high)	
Data[6...n]	Data to send	

- **Response Details**

Field	Contents	Notes
CmdExt[0]	(reserved)	(ignore)
CmdExt[1]		
Data[0]	Number of sent bytes (low)	Only valid in the last segment
Data[1]	Number of sent bytes (high)	

Command Details: IP_Add_Membership

Category

Advanced

Details

Command Code.: 19h

Valid for: Instance

Description

This command assigns the socket an IP multicast group membership. The module always joins the 'All hosts group' automatically, however this command may be used to specify up to 20 additional memberships.

- **Command Details**

Field	Contents	Notes
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]		
Data[0]	Group IP address byte 3 (low)	-
Data[1]	Group IP address byte 2	
Data[2]	Group IP address byte 1	
Data[3]	Group IP address byte 0 (high)	

- **Response Details**

(no data)

Command Details: IP_Drop_Membership

Category

Advanced

Details

Command Code.: 1Ah

Valid for: Instance

Description

This command removes the socket from an IP multicast group membership.

- **Command Details**

Field	Contents	Notes
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]		
Data[0]	Group IP address byte 3 (low)	-
Data[1]	Group IP address byte 2	
Data[2]	Group IP address byte 1	
Data[3]	Group IP address byte 0 (high)	

- **Response Details**

(no data)

Command Details: DNS_Loopup

Category

Advanced

Details

Command Code.: 1Bh

Valid for: Object Instance

Description

This command resolves the given host name and returns the IP address.

- **Command Details**

Field	Contents	Notes
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]		
Data[0... N]	Host name	Host name to resolve

- **Response Details (Success)**

Field	Contents	Notes
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]		
Data[0]	IP address byte 3 (low)	IP address of the specified host
Data[1]	IP address byte 2	
Data[2]	IP address byte 1	
Data[3]	IP address byte 0 (high)	

Socket Interface Error Codes (Object Specific)

The following object-specific error codes may be returned by the module when using the socket interface object.

Error Code	Name	Meaning
1	ENOBUFS	No internal buffers available
2	ETIMEDOUT	A timeout event occurred
3	EISCONN	Socket already connected
4	EOPNOTSUPP	Service not supported
5	ECONNABORTED	Connection was aborted
6	EWouldBlock	Socket cannot block because unblocking socket type
7	ECONNREFUSED	Connection refused
8	ECONNRESET	Connection reset
9	ENOTCONN	Socket is not connected
10	EALREADY	Socket is already in requested mode
11	EINVAL	Invalid service data
12	EMSGSIZE	Invalid message size
13	EPIPE	Error in pipe
14	EDESTADDRREQ	Destination address required
15	ESHUTDOWN	Socket has already been shutdown
16	(reserved)	-
17	EHAVEOOB	Out of band data available
18	ENOMEM	No internal memory available
19	EADDRNOTAVAIL	Address is not available
20	EADDRINUSE	Address already in use
21	(reserved)	-
22	EINPROGRESS	Service already in progress
28	ETOOMANYREFS	Too many references
101	Command aborted	If a command is blocking on a socket, and that socket is closed using the Delete command, this error code will be returned to the blocking command.

10.9 SMTP Client Object (09h)

Category

Advanced

Object Description

This object groups functions related to the SMTP-client.

See also...

- “File System” on page 17
- “E-mail Client” on page 40
- “Instance Attributes (Instance #12, SMTP Server)” on page 76
- “Instance Attributes (Instance #13, SMTP User)” on page 77
- “Instance Attributes (Instance #14, SMTP Password)” on page 77

Supported Commands

Object:	Get_Attribute Create Delete Send email from file(“Command Details: Send Email From File” on page 114)
Instance:	Get_Attribute Set_Attribute Send email(“Command Details: Send Email” on page 115)

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1	Name	Get	Array of CHAR	'SMTP Client'
2	Revision	Get	UINT8	01h
3	Number of instances	Get	UINT16	-
4	Highest instance no.	Get	UINT16	-
11	Max. no. of instances	Get	UINT16	0006h
12	Success count	Get	UINT16	Reflects the no. of successfully sent messages
13	Error count	Get	UINT16	Reflects the no. of messages that could not be delivered

Instance Attributes

Advanced

Instances are created dynamically by the application.

#	Name	Access	Type	Description
1	From	Get/Set	Array of CHAR	e.g. "someone@somewhere.com"
2	To	Get/Set	Array of CHAR	e.g. "someone.else@anywhere.net"
3	Subject	Get/Set	Array of CHAR	e.g. "Important notice"
4	Message	Get/Set	Array of CHAR	e.g. "Duck and cover"

Command Details: Create

Category

Advanced

Details

Command Code.: 03h

Valid for: Object

Description

This command creates an e-mail instance.

- Command Details

Field	Contents	Comments
CmdExt[0]	(reserved, set to zero)	-
CmdExt[1]		

- Response Details

Field	Contents	Comments
CmdExt[0]	(reserved, ignore)	-
CmdExt[1]		
MsgData[0]	Instance number	low byte
MsgData[1]		high byte

Command Details: Delete

Category

Advanced

Details

Command Code.: 04h

Valid for: Object

Description

This command deletes an e-mail instance.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	E-mail instance number	low byte
CmdExt[1]		high byte

- **Response Details**

(no data)

Command Details: Send Email From File

Category

Advanced

Details

Command Code.: 11h

Valid for: Object

Description

This command sends an e-mail based on a file in the file system.

File format:

The file must be a plain ASCII-file in the following format:

```
[To]
recipient

[From]
sender

[Subject]
email subject

[Headers]
extra headers, optional

[Message]
actual email message
```

- **Command Details**

Field	Contents	Comments
CmdExt[0]	(reserved, set to zero)	-
CmdExt[1]		
MsgData[0... n]	Path + filename of message file	-

- **Response Details**

(no data)

Command Details: Send Email

Category

Advanced

Details

Command Code.: 10h

Valid for: Instance

Description

This command sends the specified e-mail instance.

- **Command Details**
(no data)
- **Response Details**
(no data)

Object Specific Error Codes

Error Codes	Meaning
1	SMTP server not found
2	SMTP server not ready
3	Authentication error
4	SMTP socket error
5	SSI scan error
6	Unable to interpret e-mail file
255	Unspecified SMTP error
(other)	(reserved)

10.10 File System Interface Object (0Ah)

Category

Advanced

Object Description

This object provides an interface to the built-in file system. Each instance represents a handle to a file stream and contains services for file system operations.

Supported Commands

Object:	Get_Attribute Create("Command Details: Create" on page 118) Delete("Command Details: Delete" on page 119) Format Disc("Command Details: Format Disc" on page 128)
Instance:	Get_Attribute File Open("Command Details: File Open" on page 119) File Close("Command Details: File Close" on page 120) File Delete("Command Details: File Delete" on page 120) File Copy("Command Details: File Copy" on page 121) File Rename("Command Details: File Rename" on page 122) File Read("Command Details: File Read" on page 123) File Write("Command Details: File Write" on page 124) Directory Open("Command Details: Directory Open" on page 124) Directory Close("Command Details: Directory Close" on page 125) Directory Delete("Command Details: Directory Delete" on page 125) Directory Read("Command Details: Directory Read" on page 126) Directory Create("Command Details: Directory Create" on page 127) Directory Change("Command Details: Directory Change" on page 127)

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1	Name	Get	Array of CHAR	'File System Interface'
2	Revision	Get	UINT8	01h
3	Number of instances	Get	UINT16	-
4	Highest instance no.	Get	UINT16	-
11	Max. no. of instances	Get	UINT16	0004h
12	Disable virtual file system	Get	BOOL	False
13	Total disc size	Get	Array of UINT32	-
14	Free space	Get	Array of UINT32	-
15	Disc CRC	Get	Array of UINT32	-

Instance Attributes

Advanced

#	Name	Access	Type	Description
1	Instance type	Get	UINT8	<u>Value.Type:</u> 00h Reserved 01h File instance 02h Directory instance
2	File size	Get	UINT32	File size in bytes (zero for directories)
3	Path	Get	Array of CHAR	Path where instance operates

Command Details: Create

Category

Advanced

Details

Command Code.: 03h

Valid for: Object

Description

This command creates a file operation instance.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	(reserved, set to zero)	-
CmdExt[1]		

- **Response Details**

Field	Contents	Comments
CmdExt[0]	(reserved, ignore)	-
CmdExt[1]		
MsgData[0]	Instance number	low byte
MsgData[1]		high byte

Command Details: Delete

Category

Advanced

Details

Command Code.: 04h

Valid for: Object

Description

This command deletes a file operation instance.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	File operation instance number	low byte
CmdExt[1]		high byte

- **Response Details**

(no data)

Command Details: File Open

Category

Advanced

Details

Command Code.: 10h

Valid for: Instance

Description

This command opens a file for reading, writing, or appending.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	Mode	Value:Mode: 00h Read mode 01h Write mode 02h Append mode
CmdExt[1]	(reserved, set to zero)	-
MsgData[0... n]	Path + filename	Relative to current path

- **Response Details**

(no data)

Command Details: File Close

Category

Advanced

Details

Command Code.: 11h

Valid for: Instance

Description

This command closes a previously opened file.

- **Command Details**

(no data)

- **Response Details**

Field	Contents	Comments
CmdExt[0]	(reserved, ignore)	-
CmdExt[1]		-
MsgData[0]	File size	low byte, low word
MsgData[1]		-
MsgData[2]		-
MsgData[3]		high byte, high word

Command Details: File Delete

Category

Advanced

Details

Command Code.: 12h

Valid for: Instance

Description

This command permanently deletes a specified file from the file system.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	(reserved, set to zero)	-
CmdExt[1]		-
MsgData[0... n]	Path + filename	Relative to current path

- **Response Details**

(no data)

Command Details: File Copy

Category

Advanced

Details

Command Code.: 13h

Valid for: Instance

Description

This command makes a copy of a file.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	(reserved, set to zero)	-
CmdExt[1]		
MsgData[0... n]	Source path + filename	
	NULL	Relative to current path, separated by NULL
	Destination path + filename	

- **Response Details**

(no data)

Command Details: File Rename

Category

Advanced

Details

Command Code.: 14h

Valid for: Instance

Description

This command renames or moves a file.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	(reserved, set to zero)	-
CmdExt[1]		
MsgData[0... n]	Old path + filename	Relative to current path, separated by NULL
	NULL	
	New path + filename	

- **Response Details**

(no data)

Command Details: File Read

Category

Advanced

Details

Command Code.: 15h

Valid for: Instance

Description

Reads data from a file previously opened for reading.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	Bytes	no. of bytes to read
CmdExt[1]	(reserved, set to zero)	-

- **Response Details**

Field	Contents	Comments
CmdExt[0]	(reserved, ignore)	-
CmdExt[1]		
MsgData[0... n]	Data	Data read from file

Command Details: File Write

Category

Advanced

Details

Command Code.: 16h

Valid for: Instance

Description

Writes data to a file previously opened for writing or appending.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	(reserved, set to zero)	-
CmdExt[1]		
Data[0... n]	Data	Data to write to file

- **Response Details**

Field	Contents	Comments
CmdExt[0]	Bytes	no. of bytes written
CmdExt[1]	(reserved, ignore)	-

Command Details: Directory Open

Category

Advanced

Details

Command Code.: 20h

Valid for: Instance

Description

This command opens a directory.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	(reserved, set to zero)	-
CmdExt[1]		
Data[0... n]	Path + name of directory	Relative to current path

- **Response Details**

(no data)

Command Details: Directory Close

Category

Advanced

Details

Command Code.: 21h

Valid for: Instance

Description

This command closes a previously opened directory.

- **Command Details**
(no data)
- **Response Details**
(no data)

Command Details: Directory Delete

Category

Advanced

Details

Command Code.: 22h

Valid for: Instance

Description

This command permanently deletes an empty directory from the file system.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	(reserved, set to zero)	-
CmdExt[1]		
MsgData[0... n]	Path + name of directory	Relative to current path

- **Response Details**
(no data)

Command Details: Directory Read

Category

Advanced

Details

Command Code.: 23h

Valid for: Instance

Description

This command reads the contents of a directory previously opened for reading.

The command returns information about a single directory entry, which means that the command must be issued multiple times to retrieve the complete contents of a directory. When the last entry has been read, the command returns an “empty” response (i.e. a response where the data size is zero).

- **Command Details**

(no data)

- **Response Details**

Field	Contents	Comments
CmdExt[0]	(reserved, ignore)	-
CmdExt[1]		-
MsgData[0]	Size of entry	Low byte, low word
MsgData[1]		-
MsgData[2]		-
MsgData[3]		High byte, high word
MsgData[4]	Flags	<u>Bit:Meaning:</u>
		0 Entry is a directory
		1 Entry is read-only
		2 Entry is hidden
		3 Entry is a system entry
MsgData[5... n]	Name of entry	-

Command Details: Directory Create

Category

Advanced

Details

Command Code.: 24h

Valid for: Instance

Description

This command creates a directory.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	(reserved, set to zero)	-
CmdExt[1]		
MsgData[0... n]	Path + name of directory	Relative to current path

- **Response Details**

(no data)

Command Details: Directory Change

Category

Advanced

Details

Command Code.: 25h

Valid for: Instance

Description

This command changes the current directory/path for an instance.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	(reserved, set to zero)	-
CmdExt[1]		
MsgData[0... n]	Path + name of directory	Relative to current path

- **Response Details**

(no data)

Command Details: Format Disc

Category

Advanced

Details

Command Code.: 30h

Valid for: Object

Description

This command formats the file system.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	(reserved, set to zero)	-
CmdExt[1]		

- **Response Details**

(no data)

Object Specific Error Codes

Error Codes	Meaning
1	Failed to open file
2	Failed to close file
3	Failed to delete file
4	Failed to open directory
5	Failed to close directory
6	Failed to create directory
7	Failed to delete directory
8	Failed to change directory
9	Copy operation failure (could not open source)
10	Copy operation failure (could not open destination)
11	Copy operation failure (write failed)
12	Unable to rename file

10.11 Network Ethernet Object (0Ch)

Category

Advanced

Object Description

This object provides Ethernet-specific information to the application.

Supported Commands

Object: Get_Attribute

Instance: Get_Attribute

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1	Name	Get	Array of CHAR	'Network Ethernet'
2	Revision	Get	UINT8	01h
3	Number of instances	Get	UINT16	-
4	Highest instance no.	Get	UINT16	-

Instance Attributes (Instance #1)

Advanced

#	Name	Access	Type	Description
1	MAC Address	Get	Array of UINT8	Current MAC address. See also "Ethernet Host Object (F9h)" on page 158)
2	Port 1 MAC Address	Get	Array of UINT8	MAC address for port 1 (mandatory for the LLDP protocol) See also "Ethernet Host Object (F9h)" on page 158)
3	Port 2 MAC Address	Get	Array of UINT8	MAC address for port 2 (mandatory for the LLDP protocol) See also "Ethernet Host Object (F9h)" on page 158)

10.12 Functional Safety Module Object (11h)

Category

Extended

Object Description

This object contains information provided by the Safety Module connected to the Anybus CompactCom module. Please consult the manual for the Safety Module used, for values of the attributes below.

Supported Commands

Object: Get_Attribute
 Error_Confirmation

Instance: Get_Attribute

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1	Name	Get	Array of CHAR	'Functional Safety Module'
2	Revision	Get	UINT8	01h
3	Number of instances	Get	UINT16	0001h
4	Highest instance no.	Get	UINT16	0001h

Instance Attributes (Instance #1)

Extended

#	Name	Access	Type	Description
1	State	Get	UINT8	Current state of the Safety Module ^a
2	Vendor ID	Get	UINT16	Identifies vendor of the Safety Module. ^a E.g. 0001h (HMS Industrial Networks)
3	IO Channel ID	Get	UINT16	Describes the IO Channels that the Safety Module is equipped with. ^a
4	Firmware version	Get	Struct of UINT8 (Major) UINT8 (Minor) UINT8 (Build)	Safety Module firmware version. Version 2.18.3 would be represented as: first byte: 02h second byte: 12h third byte: 03h
5	Serial number	Get	UINT32	32 bit number, assigned to the Safety Module at production. ^a
6	Output data	Get	Array of UINT8	Current value of the Safety Module output data, i.e. data FROM the network Note: This data is unsafe, since it is provided by the Anybus CompactCom module.

#	Name	Access	Type	Description
7	Input data	Get	Array of UINT8	Current value of the Safety Module input data, i.e. data sent TO the network. Note: This data is unsafe, since it is provided by the Anybus CompactCom module.
8	Error counters	Get	Struct of UINT16 (ABCC DR) UINT16 (ABCC SE) UINT16 (SM DR) UINT16 (SM SE)	Error counters (each counter stops counting at FFFFh) ABCC DR: Responses (unexpected) from the Safety Module, discarded by the Anybus CompactCom module. ABCC SE: Serial reception errors detected by the Anybus CompactCom module. SM DR: Responses (unexpected) from the Anybus CompactCom module, discarded by the Safety Module. SM SE: Serial reception errors detected by the Safety Module.
9	Event log	Get	Array of UINT8	Latest Safety Module event information (if any) is logged to this attribute. Any older event information is erased when a new event is logged. For evaluation by HMS support.
10	Exception information	Get	UINT8	If the Exception Code in the Anybus object is set to "Safety communication error" (09h), additional exception information is presented here, see table below.
11	Bootloader version	Get	Struct of UINT8 Major UINT8 Minor	Safety Module bootloader version. Format: version "2.12" would be represented as: First byte = 0x02 Second byte = 0x0C

a. Values depend on which Safety Module is used.

Exception Information

If Exception Code 09h is set in the Anybus object, there is an error regarding the functional safety module in the application. Exception information is presented in instance attribute #10 according to this table:

Value	Exception Information
00h	No information
01h	Baud rate not supported
02h	No start message
03h	Unexpected message length
04h	Unexpected command in response
05h	Unexpected error code
06h	Safety application not found
07h	Invalid safety application CRC
08h	No flash access
09h	Answer from wrong safety processor during boot loader communication
0Ah	Boot loader timeout
0Bh	Network specific parameter error
0Ch	Invalid IO configuration string
0Dh	Response differed between the safety microprocessors (e.g. different module types)
0Eh	Incompatible module (e.g. supported network)
0Fh	Max number of retransmissions performed (e.g. due to CRC errors)

Command Details: Error_Confirmation

Category

Extended

Details

Command Code: 10h

Valid for: Object Instance

Description

When the Safety Module has entered the Safe State, for any reason, it must receive an error confirmation from the application, before it can leave the safe state. The application sends this command to the Anybus CompactCom module, that forwards it to the Safety Module.

- **Command Details**
(No data)
- **Response Details**
(No data)

11. Host Application Objects

11.1 General Information

This chapter specifies the host application object implementation in the module. The objects listed here may optionally be implemented within the host application firmware to expand the PROFINET implementation.

Standard Objects:

- Application Object (FFh) (see Anybus CompactCom Software Design Guide)
- Application Data Object (FEh) (see Anybus CompactCom Software Design Guide)
- “Functional Safety Object (E8h)” on page 166

Network Specific Objects:

- “PROFINET IO Object (F6h)” on page 134
- “Ethernet Host Object (F9h)” on page 158
- “Energy Control Object (F0h)” on page 161

11.2 PROFINET IO Object (F6h)

Categories

Basic, extended, advanced

Object Description

This object implements PROFINET IO-related settings in the host application.

The implementation of this object is optional; the host application can support none, some, or all of the attributes specified below. The module will attempt to retrieve the values of these attributes during start-up; if an attribute is not implemented in the host application, simply respond with an error message (06h, "Invalid CmdExt[0]"). In such a case, the module will use its default value.

If the module attempts to retrieve a value of an attribute not listed below, respond with an error message (06h, "Invalid CmdExt[0]").

See also...

- "Network PROFINET IO Object (0Eh)" on page 77
- "Flowchart - Record Data Access" on page 172
- Anybus CompactCom Software Design Guide, "Error Codes"

Supported Commands

Object:	Get Attribute	
	Get_Record	(See "Command Details: Get_Record" on page 138)
	Set_Record	(See "Command Details: Set_Record" on page 140)
	Get_IM_Record	(See "Command Details: Get_IM_Record" on page 142)
	Set_IM_Record	(See "Command Details: Set_IM_Record" on page 144)
	AR_Check_Ind	(See "Command Details: AR_Check_Ind" on page 146)
	Cfg_Mismatch_Ind	(See "Command Details: Cfg_Mismatch_Ind" on page 148)
	AR_Info_Ind	(See "Command Details: AR_Info_Ind" on page 149)
	End_Of_Prm_Ind	(See "Command Details: End_Of_Prm_Ind" on page 151)
	AR_Offline_Ind	(See "Command Details: AR_Offline_Ind" on page 152)
	AR_Abort_Ind	(See "Command Details: AR_Abort_Ind" on page 154)
	Plug_Submodule_Failed	(See "Command Details: Plug_Submodule_Failed" on page 156)

Instance: Get Attribute

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1	Name	Get	Array of CHAR	'PROFINET IO'
2	Revision	Get	UINT8	01h
3	Number of instances	Get	UINT16	0001h
4	Highest instance no.	Get	UINT16	0001h

Instance Attributes (Instance #1)

Basic

#	Name	Access	Type	Default Value ^a	Comment
1	Device ID	Get	UINT16	0009h	Identifies the device. (Assigned by manufacturer)
2	Vendor ID (I&M Manufacturer ID)	Get	UINT16	010Ch	Identifies the device manufacturer. (Assigned by the PNO)
3	Station Type	Get	Array of CHAR	'ABCC-PRT (2-Port)'	Characterizes the device (Assigned by manufacturer); up to 25 characters.
8	I&M Order ID	Get	Array of CHAR	'ABCC-PRT (2-Port)'	I&M0 Parameter: Order ID of device; up to 20 characters. ^b
10	I&M Hardware Revision	Get	UINT16	(Hardware Rev.)	I&M0 Parameter: Hardware revision of device; FFFFh indicates availability of profile specific information
11	I&M Software Revision	Get	Array of CHAR	(Software Rev.)	I&M0 Parameter: Software revision of device. <u>Byte:Value:Meaning:</u> 0: 'V' Official release 'R' Revision 'P' Prototype 'U' Under test 'T' Test device 1: 0... 255 Major Version 2: 0... 255 Minor Version 3: 0... 255 Internal Change Note: Version V255.255.255 indicates availability of profile specific information.
19	System Description	Get	Array of CHAR	'HMS Industrial Networks, Anybus CompactCom PROFINET IO 2-Port'	System description of the PROFINET IO interface. Up to 255 characters.
20	Interface Description	Get	Array of CHAR	'PROFINET IO interface'	Name of the Ethernet interface of the IO device. Up to 255 characters.
22	System Contact	Get	Array of CHAR	"www.anybus.com"	Identification of the contact person for this managed node, together with information on how to contact this person. Up to 255 characters.

a. If an attribute is not implemented, this value will be used instead.

b. If the application will use both 1-port and 2-port PROFINET modules, pad unused characters with ' ' (space), to ensure compability.

Extended

#	Name	Access	Type	Default Value ^a	Comment
7	Record Data Mode	Get	UINT8	00h	This setting affects how Record Data requests are treated, and constitutes a bit field as follows: <u>Bit 0: Index 0... 7FFFh:</u> 0: Normal Mode 1: Transparent Mode <u>Bit 1: Index AFF0h... AFFFh:</u> 0: Normal Mode 1: Transparent Mode
17	Port 1 MAC address ^b	Get	Array of UINT8	-	6 byte physical address value; overrides the preprogrammed Mac address.
18	Port 2 MAC address ^b	Get	Array of UINT8	-	6 byte physical address value; overrides the preprogrammed Mac address.
23	PROFenergy functionality	Get	UINT8	00h	Bit mask for enabling/disabling different parts of the PROFenergy functionality. See table below

a. If an attribute is not implemented, this value will be used instead.

b. The module is preprogrammed with a valid Mac address. To use that address, do *not* implement this attribute.

The table below describes the bit mask (attr. #23) for enabling/disabling different parts of the PROFenergy functionality.

Bit	Value	Description
0	0	PROFenergy functionality disabled. Ignore bits 1 - 7.
	1	PROFenergy functionality enabled
1	0	Query_Modes disabled
	1	Query_modes supported
2	0	PEM_Status disabled
	1	PEM_Status enabled
3	0	PE_Identify disabled
	1	PE_Identify enabled
4 - 7	(reserved)	

Advanced

#	Name	Access	Type	Default Value ^a	Comment
4	MaxAr	Get	UINT32	0003h	Max. no. of simultaneous ARs. (Range 1... 3)
5	(reserved)	-	-	-	(reserved for future use)
6	(reserved)	-	-	-	(reserved for future use)
9	I&M Serial Number	Get	Array of CHAR	(assigned during manufacturing)	I&M0 Parameter: Serial number of device; up to 16 characters, pad unused characters with ' ' (space).
12	I&M Revision Counter	Get	UINT16	0000h	I&M0 Parameter: Revision counter of device; a changed value marks a change of the hardware or its parameters.
13	I&M Profile ID ^b	Get	UINT16	F600h (Generic Device)	I&M0 Parameter: If the application supports a specific profile, this can be specified here.

#	Name	Access	Type	Default Value ^a	Comment
14	I&M Profile Specific Type ^b	Get	UINT16	0004h (Communication Module)	I&M0 Parameter: If the application supports a specific profile, the profile specific types is specified here.
15	I&M Version	Get	Struct of: UINT8, UINT8	0101h	I&M0 Parameter: This parameter must only be used if the host application supports an I&M version other than 1.1. If this is the case, the host application must also be prepared to handle commands Get_IM_Record and Set_IM_Record, since requests towards unknown I&M records will be forwarded to the host application.
16	I&M Supported	Get	UINT16	001Eh	I&M0 Parameter: Bit field defining which I&M parameters that are supported by the device for slot 0, subslot 1. <u>Bit:Meaning</u> 0: Profile specific 1: I&M1 supported 2: I&M2 supported 3: I&M3 supported 4: I&M4 supported 5... 15: I&M[5... 15] supported Note: I&M0 is mandatory and cannot be disabled.
21	Module ID Assignment Mode	Get	UINT8	0x00	0 - Default. Module Identification numbers are generated according to HMS standard GSD file. 1 - Incremental. Module Identification numbers are generated sequential. Starting at 0x000000100 for slot 1. Note 1: This parameter has no effect if the host application has issued the Api_Add command. Note 2: The Device Access Point located in slot 0 is not affected by this parameter. Note 3: If sequential generation is selected all modules of the GSD file MUST be unique.

a. If an attribute is not implemented, this value will be used instead.

b. The host application has to implement the corresponding functionality; the Anybus module in itself does not alter its behavior based on these parameters.

Command Details: Get_Record

Category

Extended

Details

Command Code: 10h

Valid for: Object Instance

Description

The module issues this command in the following situations:

- Module receives a Record Data Read request towards an API other than 0 (zero)
- Module receives a Record Data Read request towards a slot index other than 0 (zero)
- Module receives a Record Data Read request and I&M Version is set to a value other than 1.1
- Module receives a Record Data Read request towards API 0, but the record in question is handled in Transparent Mode

It is optional to implement support for this command. If not implemented, the original network request will be rejected and an error is returned to the IO Controller/Supervisor.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	AR handle (low byte)	Handle for Application Relationship (AR)
CmdExt[1]	AR handle (high byte)	
MsgData[0]	API (low word, low byte)	Application Process Instance (API)
MsgData[1]	API (low word, high byte)	
MsgData[2]	API (high word, low byte)	
MsgData[3]	API (high word, high byte)	
MsgData[4]	Slot (low byte)	Slot number of request
MsgData[5]	Slot (high byte)	
MsgData[6]	Subslot (low byte)	Subslot number of request
MsgData[7]	Subslot (high byte)	
MsgData[8]	Index (low byte)	Index of request
MsgData[9]	Index (high byte)	
MsgData[10]	Length	Range: 1... 244

- **Response Details (Success)**

Field	Contents	Comments
CmdExt[0... 1]	(reserved)	(set to zero)
MsgData[0... n]	Data (up to 244 bytes)	Data to be returned in the Record Data Read response.

- **Response Details (Error)**

Field	Contents	Comments
CmdExt[0... 1]	(reserved)	(set to zero)
MsgData[0]	FFh	Object specific error
MsgData[1]	Error Code 1	See "Details: Error Code 1" on page 157
MsgData[2]	Error Code 2	User specific error code
MsgData[3]	Additional Data 1	API specific. Set to zero if no Additional Data 1 is defined.
MsgData[4]	Additional Data 2	User specific. Set to zero if no Additional Data 2 is defined.

See also...

- "Command Details: Set_Record" on page 140
- "Command Details: Get_IM_Record" on page 142
- "Command Details: Set_IM_Record" on page 144
- "Flowchart - Record Data Access" on page 172

Command Details: Set_Record

Category

Extended

Details

Command Code: 11h

Valid for: Object Instance

Description

The module issues this command in the following situations:

- Module receives a Record Data Write request towards an API other than 0 (zero)
- Module receives a Record Data Write request towards a slot index other than 0 (zero)
- Module receives a Record Data Write request and I&M Version is set to a value other than 1.1
- Module receives a Record Data Write request towards API 0, but the record in question is handled in Transparent Mode

It is optional to implement support for this command. If not implemented, the original network request will be rejected and an error is returned to the IO Controller/Supervisor.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	AR handle (low byte)	Handle for Application Relationship (AR)
CmdExt[1]	AR handle (high byte)	
MsgData[0]	API (low word, low byte)	Application Process Instance (API)
MsgData[1]	API (low word, high byte)	
MsgData[2]	API (high word, low byte)	
MsgData[3]	API (high word, high byte)	
MsgData[4]	Slot (low byte)	Slot number of request
MsgData[5]	Slot (high byte)	
MsgData[6]	Subslot (low byte)	Subslot number of request
MsgData[7]	Subslot (high byte)	
MsgData[8]	Index (low byte)	Index of request
MsgData[9]	Index (high byte)	
MsgData[10]	(reserved)	Reserved; set to zero.
MsgData[11...n]	Data (up to 244 bytes)	Data from the Record Data Write request.

- **Response Details (Success)**

(no data)

- **Response Details (Error)**

Field	Contents	Comments
CmdExt[0... 1]	(reserved)	(set to zero)
MsgData[0]	FFh	Object specific error
MsgData[1]	Error Code 1	See "Details: Error Code 1" on page 157
MsgData[2]	Error Code 2	User specific error code
MsgData[3]	Additional Data 1	API specific. Set to zero if no Additional Data 1 is defined.
MsgData[4]	Additional Data 2	User specific. Set to zero if no Additional Data 2 is defined.

See also...

- "Command Details: Get_Record" on page 138
- "Command Details: Get_IM_Record" on page 142
- "Command Details: Set_IM_Record" on page 144
- "Flowchart - Record Data Access" on page 172

Command Details: Get_IM_Record

Category

Extended

Details

Command Code: 12h

Valid for: Object Instance

Description

This command retrieves I&M information from the host application, and may be issued in the following situations:

- The module receives a request towards an unknown I&M Index
- The module receives an I&M request and Transparent Mode is enabled (Index AFF0h... AFFFh)

It is optional to implement support for this command. If not implemented, the original network request will be rejected and an error is returned to the IO Controller/Supervisor.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	IM Record Index	Valid range: 0... 15 <u>Value:Meaning:</u> 0: I&M0 1: I&M1 2: I&M2 3: I&M3 4: I&M4 5...15: Reserved for additional I&M functions
CmdExt[1]	(reserved)	(ignore)
MsgData[0]	Slot (low byte)	Slot number of request
MsgData[1]	Slot (high byte)	
MsgData[2]	Subslot (low byte)	Subslot number of request
MsgData[3]	Subslot (high byte)	

- **Response Details (Success)**

Field	Contents	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]		
MsgData[0...9]	(reserved)	-
MsgData[10...n]	I&M Data	See "I&M Data Structures" on page 26

- **Response Details (Error)**

Field	Contents	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]		
MsgData[0]	FFh	Object specific error
MsgData[1]	Error Code 1	See "Details: Error Code 1" on page 157
MsgData[2]	Error Code 2	User specific error code
MsgData[3]	Additional Data 1	API specific. Set to zero if no Additional Data 1 is defined.
MsgData[4]	Additional Data 2	User specific. Set to zero if no Additional Data 2 is defined.

See also...

- "Command Details: Get_Record" on page 138
- "Command Details: Set_Record" on page 140
- "Command Details: Set_IM_Record" on page 144
- "Flowchart - Record Data Access" on page 172

Command Details: Set_IM_Record

Category

Extended

Details

Command Code: 13h

Valid for: Object Instance

Description

This command specifies I&M information to the host application, and may be issued in the following situations:

- The module receives a request towards an unknown I&M Index
- The module receives an I&M request and Transparent Mode is enabled (Index AFF0h... AFFFh)

It is optional to implement support for this command. If not implemented, the original network request will be rejected and an error is returned to the IO Controller/Supervisor.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	IM Record Index	Valid range: 0... 15 <u>Value:Meaning:</u> 0: I&M0 1: I&M1 2: I&M2 3: I&M3 4: I&M4 5...15: Reserved for additional I&M functions
CmdExt[1]	(reserved)	(ignore)
MsgData[0]	Slot (low byte)	Slot number of request
MsgData[1]	Slot (high byte)	
MsgData[2]	Subslot (low byte)	Subslot number of request
MsgData[3]	Subslot (high byte)	
MsgData[4...13]	(reserved)	(ignore)
MsgData[14...n]	I&M Data	See "I&M Data Structures" on page 26

- **Response Details (Success)**

(no data)

- **Response Details (Error)**

Field	Contents	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]		
MsgData[0]	FFh	Object specific error
MsgData[1]	Error Code 1	See "Details: Error Code 1" on page 157
MsgData[2]	Error Code 2	User specific error code
MsgData[3]	Additional Data 1	API specific. Set to zero if no Additional Data 1 is defined.
MsgData[4]	Additional Data 2	User specific. Set to zero if no Additional Data 2 is defined.

See also...

- "Command Details: Get_Record" on page 138
- "Command Details: Set_Record" on page 140
- "Command Details: Get_IM_Record" on page 142
- "Flowchart - Record Data Access" on page 172

Command Details: AR_Check_Ind

Category

Advanced

Details

Command Code: 14h

Valid for: Object Instance

Description

The module issues this command to inform the host application that an Application Relationship (AR) is to be established. It is optional to implement support for this command.

• Command Details

Field	Contents	Comments
CmdExt[0]	AR handle (low byte)	Handle for the Application Relationship (AR)
CmdExt[1]	AR handle (high byte)	
MsgData[0]	IP address (low word, low byte)	IP address of the remote station (IO Controller/Supervisor)
MsgData[1]	IP address (low word, high byte)	
MsgData[2]	IP address (high word, low byte)	
MsgData[3]	IP address (high word, high byte)	
MsgData[4]	AR Type (low byte)	Indicates the type of AR as follows: <u>Value:Meaning:</u> 1: IO_AR_SINGLE 3: IO_AR_CIR 4: IO_AR_REDUNDANT_CONTROLLER 5: IO_AR_REDUNDANT_DEVICE 6: SUPERVISOR_AR
MsgData[5]	AR Type (high byte)	
MsgData[6]	AR Properties (low word, low byte)	Bit-field indicating the properties of the AR as follows: <u>Bit 0-2:State:</u> 0: Backup 1: Primary <u>Bit 3:Supervisor take over allowed:</u> 0: Not allowed 1: Allowed <u>Bit 4:Parameterization server:</u> 0: EXTERNAL_PRM_SERVER 1: CM_INITIATOR <u>Bit 5-6:Data rate:</u> 0: AT_LEAST_100 Mbps 1: 100 Mbps 2: 1 Gbps 3: 10 Gbps <u>Bit 8:Device Access:</u> 0: AR_CONTEXT 1: DEVICE_CONTEXT <u>Bit 9-10:Companion AR:</u> 0: SINGLE_AR 1: FIRST_AR 2: COMPANION_AR
MsgData[7]	AR Properties (low word, high byte)	
MsgData[8]	AR Properties (high word, low byte)	
MsgData[9]	AR Properties (high word, high byte)	
MsgData[10]	Remote station name length	Length of remote station name, in bytes
MsgData[11...n]	Remote station name	Remote station name (IO Controller/Supervisor)

- **Response Details**

(No data)

Command Details: Cfg_Mismatch_Ind

Category

Advanced

Details

Command Code: 15h

Valid for: Object Instance

Description

The module issues this command to inform the host application that the configuration in the IO Controller (i.e. the Expected Identification) does not match the configuration defined by the host application (i.e. the Real Identification).

It is optional to implement support for this command.

• Command Details

Field	Contents	Comments
CmdExt[0]	AR handle (low byte)	Handle for the Application Relationship (AR)
CmdExt[1]	AR handle (high byte)	
MsgData[0]	API (low word, low byte)	
MsgData[1]	API (low word, high byte)	
MsgData[2]	API (high word, low byte)	
MsgData[3]	API (high word, high byte)	Application Process Instance (API)
MsgData[4]	Slot (low byte)	
MsgData[5]	Slot (high byte)	
MsgData[6]	Subslot (low byte)	
MsgData[7]	Subslot (high byte)	
MsgData[8]	Expected Module Identifier (low word, low byte)	Slot number of mismatch
MsgData[9]	Expected Module Identifier (low word, high byte)	
MsgData[10]	Expected Module Identifier (high word, low byte)	
MsgData[11]	Expected Module Identifier (high word, high byte)	
MsgData[12]	Expected Submodule Identifier (low word, low byte)	Subslot number of mismatch
MsgData[13]	Expected Submodule Identifier (low word, high byte)	
MsgData[14]	Expected Submodule Identifier (high word, low byte)	
MsgData[15]	Expected Submodule Identifier (high word, high byte)	

• Response Details

(No data)

See also...

- “Real Identification (RI)” on page 23 (“Configuration Mismatch” on page 24)
- “Flowchart - Configuration Mismatch (RI)” on page 173

Command Details: AR_Info_Ind

Category

Advanced

Details

Command Code: 16h

Valid for: Object Instance

Description

The module issues this command to inform the host application of the Expected Identification (Module/Submodule List) that the IO Controller will use for the established AR.

Note that this information may be split in multiple segments, which means that this command will be issued multiple times by the module, each time containing different parts of the configuration.

It is optional to implement support for this command.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	AR handle (low byte)	Handle for the Application Relationship (AR)
CmdExt[1]	AR handle (high byte)	
MsgData[0]	Current segment (low byte)	Current segment number; when this word equals 'Total segments' (below), all data has been transferred.
MsgData[1]	Current segment (high byte)	
MsgData[2]	Total segments (low byte)	
MsgData[3]	Total segments (high byte)	
MsgData[4...n]	Data field	The first two bytes in the initial block of the Data field indicates the number of APIs in the configuration. Each module is represented by a Module block, followed by a number of Submodule blocks (provided that the module in question contains submodules). See "Data Format" on page 150 for coding of the data field.

- **Response Details**

(No data)

Data Format

When all data has been received, the resulting data shall be interpreted as follows:

Type	Name		Description
UINT16	iNbrApi		Number of APIs in configuration.
UINT32		iApiNbr	Initial module block including API number and number of module blocks in the API.
UINT16		iNbrMod	
UINT16		iSlotNbr	Module block (8 bytes), see below.
UINT16		iNbrSubMod	
UINT32		IModIdent	
UINT16		iSubSlotNbr	Submodule block (10 bytes), see below.
UINT32		ISubModIdent	
UINT16		iInDataLength	
UINT16		iOutDataLength	

The initial API block (iNbrApi) defines the number of APIs in the configuration.

Each API has an initial module block, that includes information on the API number (iApiNbr) and the number of modules (or slots) in the API.

Each module starts with a module block, which holds the slot number, the number of submodules (or subslots) and the module identity number.

Finally each submodule block holds subslot number, submodule identification number, input and output data lengths.

Example:

In this example, the configuration contains two APIs with the following properties:

- API #1 contains two modules, the first with two submodules, the second with one submodule
- API #2 contains one module with one submodule

Initial API Block	No. of APIs	0002h
Initial Module Block (API #0)	API no.	00 00 00 00h
	No. of Modules	0002h
Module Block (Module #1)	Slot no.	0001h
	No. of Submodules	0003h
	Module ID	4A 6F 48 62h
Submodule Block (Module #1)	Subslot no.	0001h
	Submodule ID	65 6C 69 65h
	Input Data Length	0004h
	Output Data Length	0010h
Submodule Block (Module #1)	Subslot no.	0002h
	Submodule ID	76 65 73 69h
	Input Data Length	0008h
	Output Data Length	0002h
Submodule Block (Module #1)	Subslot no.	0003h
	Submodule ID	6E 53 61 6Eh
	Input Data Length	0008h
	Output Data Length	0002h
Module Block (Module #2)	Slot no.	0002h
	No. of Submodules	0001h
	Module ID	74 61 43 6Ch
Submodule Block (Module #2)	Subslot no.	0001h
	Submodule ID	61 75 73 21h
	Input Data Length	0010h
	Output Data Length	0001h
Initial Module Block (API #2)	API no.	00 00 00 02h
	No. of Modules	0001h
Module Block (Module #1)	Slot no.	0001h
	No. of Submodules	0002h
	Module ID	4A 6F 48 82h
Submodule Block (Module #1)	Subslot no.	0001h
	Submodule ID	65 6C 67 65h
	Input Data Length	0004h
	Output Data Length	0010h
Submodule Block (Module #1)	Subslot no.	0002h
	Submodule ID	76 65 74 69h
	Input Data Length	0008h
	Output Data Length	0002h

Command Details: End_Of_Prm_Ind

Category

Advanced

Details

Command Code: 17h

Valid for: Object Instance

Description

The module may issue this command to indicate to the host application that the parameterization phase is completed. It is optional to implement support for this command.

If implemented, the host application may, depending on the response issued to this command, be required to issue 'Appl_State_Ready' at a later stage to indicate that it is ready for data exchange. If not implemented, this is handled automatically by the module.

• Command Details

Field	Contents	Comments
CmdExt[0]	AR handle (low byte)	Handle for the Application Relationship (AR)
CmdExt[1]	AR handle (high byte)	
MsgData[0]	API (low word, low byte)	Application Process Instance (API) - Only valid if subslot > 0
MsgData[1]	API (low word, high byte)	
MsgData[2]	API (high word, low byte)	
MsgData[3]	API (high word, high byte)	
MsgData[4]	Slot (low byte)	Slot number affected by the command - Only valid if subslot > 0
MsgData[5]	Slot (high byte)	
MsgData[6]	Subslot (low byte)	Subslot number affected by the command <u>Value:Meaning:</u> 0: Command applies to all modules in the configuration other: Command applies only to the specified slot/subslot
MsgData[7]	Subslot (high byte)	

• Response Details

Field	Contents	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]		
MsgData[0]	Application State	<u>Value:Meaning:</u> 0: Ready for Data Exchange 1: Not ready for data exchange (Appl_State_Ready must be issued at a later stage)

See also...

- “Command Details: Appl_State_Ready” on page 85

Command Details: AR_Offline_Ind

Category

Advanced

Details

Command Code: 18h

Valid for: Object Instance

Description

The module issues this command to indicate to the host application that the module enters an offline state. It is optional to implement support for this command.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	AR handle (low byte)	Handle for the Application Relationship (AR)
CmdExt[1]	AR handle (high byte)	
MsgData[0]	Reason code (low byte)	Reason code for the offline transition
MsgData[1]	Reason code (high byte)	<u>Value:Reason:</u> 0: No reason (unknown reason) 3: Out of mem 4: Add provider or consumer failed 5: Miss (consumer) 6: Cmi timeout 7: Alarm-open failed 8: Alarm-send.cnf(-) 9: Alarm-ack-send.cnf(-) 10: Alarm-data too long 11: Alarm.ind(err) 12: Rpc-client call.cnf(-) 13: Ar-abort.req 14: Re-run aborts existing 15: Got release.ind 16: Device passivated 17: Device/Ar removed 18: Protocol violation 19: NARE error 20: RPC-Bind error 21: RPC-Connect error 22: RPC-Read error 23: RPC-Write error 24: RPC-Control error 25: Forbidden pull or plug after check.rsp and before in-data.ind 26: AP removed 27: Link down 28: Could not register multicast-mac 29: Not synchronized (cannot start companion-ar) 30: Wrong topology (cannot start companion-ar) 31: Dcp, station-name changed 32: Dcp, reset to factory-settings 33: Cannot start companion AR because of parameter error

- **Response Details**
(no data)

Command Details: AR_Abort_Ind

Category

Advanced

Details

Command Code: 19h

Valid for: Object Instance

Description

The module issues this command to indicate to the host application that an Application Relationship (AR) is aborted (by the application or any other source).

It is optional to implement support for this command.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	AR handle (low byte)	Handle for the Application Relationship (AR)
CmdExt[1]	AR handle (high byte)	
MsgData[0]	Reason code (low byte)	Reason code for the offline transition
MsgData[1]	Reason code (high byte)	<u>Value:Reason:</u> 0: No reason (unknown reason) 3: Out of mem 4: Add provider or consumer failed 5: Miss (consumer) 6: Cmi timeout 7: Alarm-open failed 8: Alarm-send.cnf(-) 9: Alarm-ack-send.cnf(-) 10: Alarm-data too long 11: Alarm.ind(err) 12: Rpc-client call.cnf(-) 13: Ar-abort.req 14: Re-run aborts existing 15: Got release.ind 16: Device passivated 17: Device/Ar removed 18: Protocol violation 19: NARE error 20: RPC-Bind error 21: RPC-Connect error 22: RPC-Read error 23: RPC-Write error 24: RPC-Control error 25: Forbidden pull or plug after check.rsp and before in-data.ind 26: AP removed 27: Link down 28: Could not register multicast-mac 29: Not synchronized (cannot start companion-ar) 30: Wrong topology (cannot start companion-ar) 31: Dcp, station-name changed 32: Dcp, reset to factory-settings 33: Cannot start companion AR because of parameter error

- **Response Details**

(no data)

Command Details: Plug_Submodule_Failed

Category

Advanced

Details

Command Code: 1Ah

Valid for: Object Instance

Description

The module issues this command to indicate that a Plug_Submodule towards the Network PROFINET IO Object failed.

It is optional to implement support for this command.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	(reserved)	(ignore)
CmdExt[1]		
MsgData[0]	API (low word, low byte)	Application Process Instance (API)
MsgData[1]	API (low word, high byte)	
MsgData[2]	API (high word, low byte)	
MsgData[3]	API (high word, high byte)	
MsgData[4]	Slot (low byte)	Slot number of request
MsgData[5]	Slot (high byte)	
MsgData[6]	Subslot (low byte)	Subslot number of request
MsgData[7]	Subslot (high byte)	

- **Response Details**

(No data)

See also...

- “Command Details: Plug_Module” on page 79
- “Command Details: Plug_Submodule” on page 80
- “Command Details: Pull_Module” on page 82
- “Command Details: Pull_Submodule” on page 83

Details: Error Code 1

The error codes below shall be used when providing error responses to the following commands:

- Get_Record
- Set_Record
- Get_IM_Record
- Set_IM_Record

High nibble (bits 4... 7)		Low nibble (bits 0... 3)		Comments
ErrorClass	Meaning	ErrorCode	Meaning	
0... 9	Reserved	(reserved)	(reserved)	-
10	Application	0	Read error	-
		1	Write error	-
		2	Module error	-
		3... 6	(reserved)	-
		7	Busy	-
		8	Version conflict	-
		9	Feature not supported	-
		10... 15	User specific	-
11	Access	0	Invalid index	-
		1	Write length error	-
		2	Invalid slot/subslot	-
		3	Type conflict	-
		4	Invalid area	-
		5	State conflict	-
		6	Access denied	-
		7	Invalid range	-
		8	Invalid parameter	-
		9	Invalid type	-
		10	Backup	-
		11... 15	User specific	-
12	Resource	0	Read constrain conflict	-
		1	Write constrain conflict	-
		2	Resource busy	-
		3	Resource unavailable	-
		4... 7	(reserved)	-
		8... 15	User specific	-
13... 15	User specific	(user specific)	User specific	-

See also...

- “Command Details: Get_Record” on page 138
- “Command Details: Set_Record” on page 140
- “Command Details: Get_IM_Record” on page 142
- “Command Details: Set_IM_Record” on page 144

11.3 Ethernet Host Object (F9h)

Category

Extended, advanced

Object Description

This object implements Ethernet features in the host application.

Supported Commands

Object:	Get_Attribute
Instance:	Get_Attribute Set_Attribute

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1	Name	Get	Array of CHAR	'Ethernet'
2	Revision	Get	UINT8	02h
3	Number of instances	Get	UINT16	0001h
4	Highest instance no.	Get	UINT16	0001h

Instance Attributes (Instance #1)

Extended

#	Name	Access	Type	Default ^a	Comment
2	Enable HICP	Get	BOOL	True	Enable/Disable HICP ^b (see "HICP (Host IP Configuration Protocol)" on page 175)
3	Enable Web Server	Get	BOOL	True	Enable/Disable Web Server ^b (see "Web Server" on page 34)
5	Enable Web ADI access	Get	BOOL	True	Enable/Disable Web ADI access ^b (see "Web Server" on page 34)
6	Enable FTP server	Get	BOOL	True	Enable/Disable FTP server ^b (see "FTP Server" on page 32)
7	Enable admin mode	Get	BOOL	False	Enable/Disable FTP admin mode ^b (see "FTP Server" on page 32)
8	Network Status	Set	UINT16	-	See "Network Status" on page 160

a. If an attribute is not implemented, the module will use this value instead

b. True=Enable/False=Disable

Advanced

#	Name	Access	Type	Default ^a	Comment
1	MAC address ^b	Get	Array of UINT8	-	6 byte physical address value; overrides the pre-programmed Mac address. Note that the new Mac address value must be obtained from the IEEE.
9	Port 1 MAC address ^b	Get	Array of UINT8	-	6 byte MAC address for port 1 (mandatory for the LLDP protocol). Note: This setting overrides any Port MAC address in the host PROFINET IO Object.
10	Port 2 MAC address ^b	Get	Array of UINT8	-	6 byte MAC address for port 2 (mandatory for the LLDP protocol). Note: This setting overrides any Port MAC address in the host PROFINET IO Object.

a. If an attribute is not implemented, the module will use this value instead

b. The module is preprogrammed with a valid Mac address. To use that address, do *not* implement this attribute.

Network Status

This attribute holds a bit field which indicates the overall network status as follows:

Bit	Contents	Description
0	Link	Current global link status True=Link sensed/False=No link
1	IP established	True=IP address established/False= IP address not established
2	(reserved)	(mask off and ignore)
3	Link port 1	Current link status for port 1 True=Link sensed/False=No link
4	Link port 2	Current link status for port 2 True=Link sensed/False=No link
5... 15	(reserved)	(mask off and ignore)

11.4 Energy Control Object (F0h)

Category

Extended

Object Description

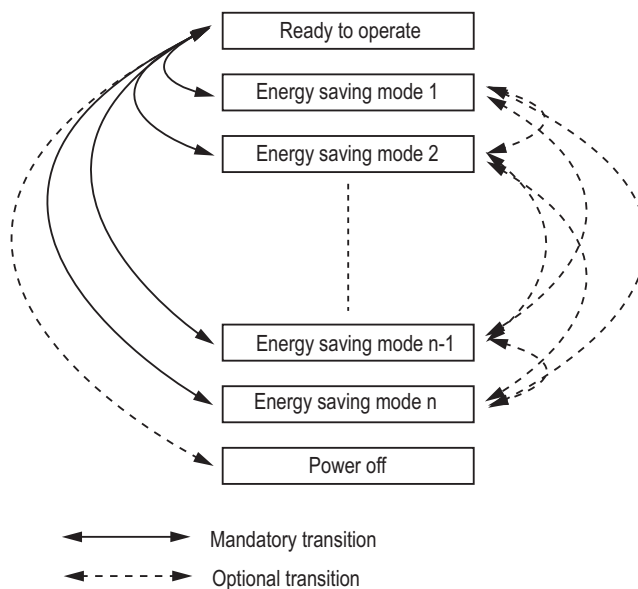
This object implements energy control functionality, i.e. energy specific settings, in the host application. The implementation of this object is optional; the host application can support none, some, or all of the attributes specified below.

Each enabled instance in the object corresponds to an Energy saving mode. The number of available modes is device specific, and must be defined by the application. The higher the instance number, the more energy is saved. The instance with the highest number always corresponds to the “Power off” mode, i.e. the state where the device is essentially shut down. Instance 1 of the object represents “Ready to operate”, i.e. the mode where the device is fully functional and does not save energy at all. Consequently a meaningful implementation always contains at least two instances, one for energy saving and one for operating. It is recommended not to use a higher instance number than 9.

Please note that these states are always present, they are not dynamically created or deleted.

See also...

- “PROFINET IO Object (F6h)” on page 134
- “PROFIenergy Profile” on page 30



Supported Commands

Object: Get Attribute
StartPause
EndPause

Instance: Get Attribute

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value	Description
1	Name	Get	Array of CHAR	'Energy'	Object Name
2	Revision	Get	UINT8	01h	First Revision of object
3	Number of instances ^a	Get	UINT16	-	Number of instances in object
4	Highest instance no. ^a	Get	UINT16	-	Highest created instance number (max. 65534)
11	Current Energy Saving mode	Get	UINT16	-	Instance number of the currently used Energy Saving mode. "Ready to operate" will equal 1 and "Power off" will equal Highest instance number.
12	RemaingTimeToDestination ^b	Get	UINT32	FFFFFFFFh (Default)	When changing mode this parameter will reflect the actual time (in milliseconds) remaining until the shift is completed. ^c
13	EnergyConsumption-ToDestination	Get	FLOAT	0.0 (Default, also used if the value is undefined.)	When changing mode this parameter will reflect the actual power (in kWh) that will be consumed until the shift is completed. ^c

- a. Attribute 3 and 4 will hold the same value, as the highest created instance number always equals the number of instances.
- b. If the value of this attribute is infinite or unknown, the maximum value of FFFFFFFFh shall be used. If the value is zero, 00000000h shall be used.
- c. If a dynamic value can not be generated, the static value for the transition from the source to destination mode shall be used.

Instance Attributes (Instance #n)

Extended

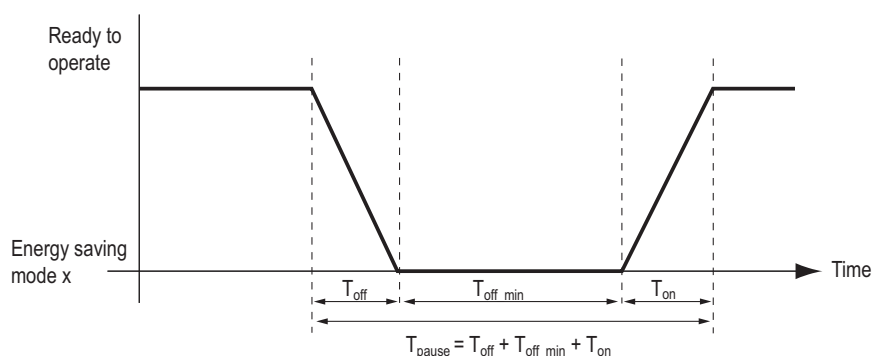
#	Name	Access	Type	Default Value ^a	Comment
1	ModeAttributes	Get	UINT16	0	Bit field defining whether static or dynamic values are available. <u>Bit 0: Meaning</u> 0: Only static time and energy values 1: Dynamic time and energy values Bit 1-15: reserved
2	TimeMinPause ^b	Get	UINT32	0	Minimum pause time, t_{pause} (ms) ^c
3	TimeToPause ^b	Get	UINT32	0	Expected time to go to this energy saving state, t_{off} (ms) ^c
4	TimeToOperate ^b	Get	UINT32	0	Time needed to go the "Ready to operate" state, t_{on} (ms) ^c
5	TimeMinLengthOfStay ^b	Get	UINT32	0	The minimum time that the device must stay in this state, $t_{\text{off_min}}$ (ms) ^c
6	TimeMaxLengthOfStay ^b	Get	UINT32	FFFFFFFFh	The maximum time that it is allowed to stay in this state (ms)
7	ModePowerConsumption	Get	FLOAT	0.0 ^d	Amount of energy consumed in this state (kW)
8	EnergyConsumptionTo-Pause	Get	FLOAT	0.0 ^d	Amount of energy required to go to this state (kWh)
9	EnergyConsumptionTo-Operate	Get	FLOAT	0.0 ^d	Amount of energy required to go to the "Ready to operate" state from this state (kWh)

a. If an attribute is not implemented, this value will be used instead.

b. If the value of this attribute is infinite or unknown, the maximum value of FFFFFFFFh shall be used. If the value is zero, 00000000h shall be used.

c. $t_{\text{off}} + t_{\text{off_min}} + t_{\text{on}} = t_{\text{pause}}$, see picture below.

d. The value 0.0 is also used if the value of the attribute is undefined.



Command Details: StartPause

Category

Extended

Details

Command Code: 10h

Valid for: Object Instance

Description

The module issues this command to the host application when the system wants to initialize a pause of the system. The length of the pause is specified in milliseconds. The response contains the destination mode, i.e. the instance number of the selected energy saving state. The command is always issued towards the object itself.

- **Command Details**

Field	Contents	Comments
CmdExt[0]		(not used)
CmdExt[1]		
MsgData[0]	Pause time (low word, low byte)	Pause time (ms)
MsgData[1]	Pause time (low word, high byte)	
MsgData[2]	Pause time (high word, low byte)	
MsgData[3]	Pause time (high word, high byte)	

- **Response Details**

Field	Contents	Comments
CmdExt[0... 1]	(reserved)	(set to zero)
MsgData[0]	Instance number (low byte)	Instance number of the selected Energy mode
MsgData[1]	Instance number (high byte)	

If the application does not change modes, the error code “ABP_ERR_OUT_OF_RANGE” (0Ch) is returned.

See also...

- “Command Details: EndPause” on page 165

Command Details: EndPause

Category

Extended

Details

Command Code: 11h

Valid for: Object Instance

Description

The module issues this command to the host application when the system wants to return the system from a pause mode back to “Ready to operate” mode. The response contains the number of milliseconds needed to return to “Ready to operate” mode. The command is always issued towards the object itself.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	-	(not used)
CmdExt[1]		

- **Response Details**

Field	Contents	Comments
CmdExt[0... 1]	(reserved)	(set to zero)
MsgData[0]	Time to operate (low word, low byte)	Time needed to switch to “Ready to operate” mode (ms)
MsgData[1]	Time to operate (low word, high byte)	
MsgData[2]	Time to operate (high word, low byte)	
MsgData[3]	Time to operate (high word, high byte)	

If the application is unable to end the pause, the error code “ABP_ERR_INV_STAT” (0Dh) shall be returned.

11.5 Functional Safety Object (E8h)

Category

Extended

Object Description

This object specifies the safety settings of the application. It is mandatory if Functional Safety is to be supported and a Safety Module is connected to the Anybus CompactCom module.

Supported Commands

Object: Get_Attribute

Instance: Get_Attribute

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1	Name	Get	Array of CHAR	'Functional Safety'
2	Revision	Get	UINT8	01h
3	Number of instances	Get	UINT16	0001h
4	Highest instance no.	Get	UINT16	0001h

Instance Attributes (Instance #1)

Extended

#	Name	Access	Type	Description
1	Safety enabled	Get	BOOL	When TRUE, enables communication with the Safety Module. Note: If functional safety is not supported, this attribute must not be set to TRUE.
2	Baud Rate	Get	UINT32	Optional attribute. Sets the baud rate for the communication between the Anybus CompactCom module and the Safety Module. If not implemented, the default value 1020 kbit/s will be used. If the generated baud rate differs more than 5% from the value of this attribute, the Anybus CompactCom module will go into EXCEPTION.
3	IO Configuration	Get	Array of UINT8	Optional attribute. Manufacturer specific settings of the digital I/O of the Safety Module. See the manual of the Safety Module used for information.

A. Categorization of Functionality

The objects, including attributes and services, of the Anybus CompactCom and the application are divided into three categories: basic, advanced and extended.

A.1 Basic

This category includes objects, attributes and services that are mandatory to implement or to use. They will be enough for starting up the Anybus CompactCom and sending/receiving data with the chosen network protocol. The basic functions of the industrial network are used.

Additional objects etc, that will make it possible to certify the product also belong to this category.

A.2 Extended

Use of the objects in this category extends the functionality of the application. Access is given to the more specific characteristics of the industrial network, not only the basic moving of data to and from the network. Extra value is given to the application.

A.3 Advanced

The objects, attributes and services that belong to this group offer specialized and/or seldom used functionality. Most of the available network functionality is enabled and accessible. Access to the specification of the industrial network is normally required.

B. Anybus Implementation Details

B.1 Extended LED Functionality

On the Anybus CompactCom 30 PROFINET IO 2-Port module, there is the possibility to use GIP1 and GOP1 for indicating link/activity on the 2 ports of the module.

To enable the extended LED functionality, the application needs to set the Anybus Object Instance 1 attribute 16 (GPIO configuration) to 0x0001 during state SETUP.

See the Anybus CompactCom Hardware Design Guide for Host Interface Signals.

GPIO mode description

		Signal	
		GIP1	GOP1
GPIO Configuration	Value: 0x0000 (Default)	General purpose input	General purpose output
	Value: 0x0001 (Extended LED functionality)	The 10/100 Mbit indication for port 1 is combined into one signal, connected to a green LED, that is lit when there is a link and flickering on activity.	The 10/100 Mbit indication for port 2 is combined into one signal, connected to a green LED, that is lit when there is a link and flickering on activity.

Note 1: Enabling the extended LED functionality will cause both GIP[0..1] and GOP[0..1] to function as outputs.

Note 2: Enabling the extended LED functionality will define both GIP[0..1] and GOP[0..1] as active low. This means that LEDs will be lit when the corresponding pin is low.

Note 3: LED behavior is described in section “Network Status LED” on page 176

B.2 SUP-Bit Definition

The supervised bit (SUP) indicates that the network participation is supervised by another network device. In the case of PROFINET IO, this bit is set when one or more IO connections are established.

B.3 Anybus State Machine

The table below describes how the Anybus State Machine relates to the PROFINET IO network.

Anybus State	Implementation	Comment
WAIT_PROCESS	The Anybus stays in this state until an IO connection with an IO controller is opened.	Note that the module will not shift from NW_INIT to WAIT_PROCESS unless there is a physical connection to a network (i.e. link sensed).
ERROR	Configuration data mismatch or initial parameter error.	-
PROCESS_ACTIVE	IO connection established with IO controller and valid output data has been provided at least once.	-

Anybus State	Implementation	Comment
IDLE	IO controller with which an IO connection is established is in STOP mode or the IO controller has not provided valid output at least once.	Some IO controllers will not provide valid output data int the first cycles following a successful connection.
EXCEPTION	Turn module status LED red, to indicate major fault, turn network status LED off, and hold Ethernet MAC in reset.	Some kind of unexpected behavior, for example watchdog timeout. See also instance 1, attribute 7 in "Network Object (03h)" on page 67.

B.4 Application Watchdog Timeout Handling

Upon detection of an application watchdog timeout, the module will cease network participation and shift to state 'EXCEPTION'. No other network specific actions are performed.

C. Message Segmentation

C.1 General

Category: Advanced

The maximum message size supported by the Anybus CompactCom 30 is 255 bytes. To provide support for longer messages (needed when using the socket interface), a segmentation protocol is used.

The segmentation protocol is implemented in the message layer and must not be confused with the fragmentation used on the serial host interface. Consult the general Anybus CompactCom 30 Software Design Guide for further information.

The module supports 1 (one) simultaneous segmented message per instance.

C.2 Command Segmentation

When a command message is segmented, the command initiator sends the same command header multiple times. For each message, the data field is exchanged with the next data segment.

Please note that some commands can't be used concurrently on the same instance, since they both need access to the segmentation buffer for that instance.

Command segmentation is used for the following commands:

- Send (see "Command Details: Send" on page 105)
- Send To (see "Command Details: Send_To" on page 106)

Segmentation Control bits (Command)

Bit	Contents	Meaning
0	FS	Set if the current segment is the first segment
1	LS	Set if the current segment is the last segment
2	AB	Set if the segmentation shall be aborted
3...7	(reserved)	Set to 0 (zero)

Segmentation Control bits (Response)

Bit	Contents	Meaning
0...7	(reserved)	Ignore

When issuing a segmented command, the following rules apply:

- When issuing the first segment, FS must be set.
- When issuing subsequent segments, both FS and LS must be cleared.
- When issuing the last segment, the LF-bit must be set.
- For single segment commands (i.e. size less or equal to 255 bytes), both FS and LS must be set.
- The last response message contains the actual result of the operation.
- The command initiator may at any time abort the operation by issuing a message with AB set.

- If a segmentation error is detected during transmission, an error message is returned, and the current segmentation message is discarded. Note however that this only applies to the current segment; previously transmitted segments are still valid.

C.3 Response Segmentation

When a response is segmented, the command initiator requests the next segment by sending the same command multiple times. For each response, the data field is exchanged with the next data segment.

Response segmentation is used for responses to the following commands:

- Receive (object specific, see “Command Details: Receive” on page 103)
- Receive From (object specific, see “Command Details: Receive_From” on page 104)

Segmentation Control bits (Command)

Bit	Contents	Meaning
0	(reserved)	(set to zero)
1		
2	AB	Set if the segmentation shall be aborted
3...7	(reserved)	(set to zero)

Segmentation Control bits (Response)

Bit	Contents	Meaning
0	FS	Set if the current segment is the first segment
1	LS	Set if the current segment is the last segment
2...7	(reserved)	(set to zero)

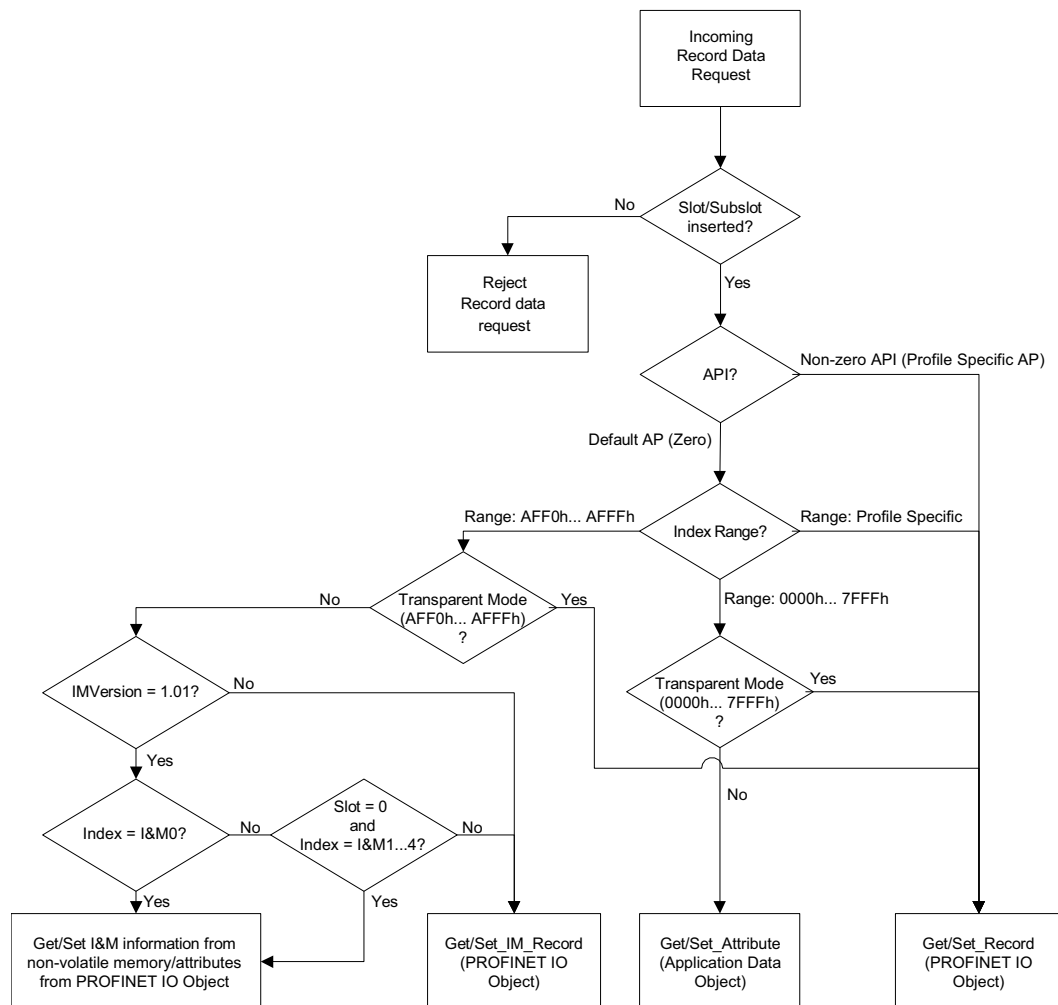
When receiving a segmented response, the following rules apply:

- In the first segment, FS is set
- In all subsequent segment, both FS and LS are cleared
- In the last segment, LS is set
- For single segment responses (i.e. size less or equal to 255 bytes), both FS and LS are set.
- The command initiator may at any time abort the operation by issuing a message with AB set.

D. Flowcharts

D.1 Flowchart - Record Data Access

This flowchart illustrates how Record Data requests are handled by the Anybus module.

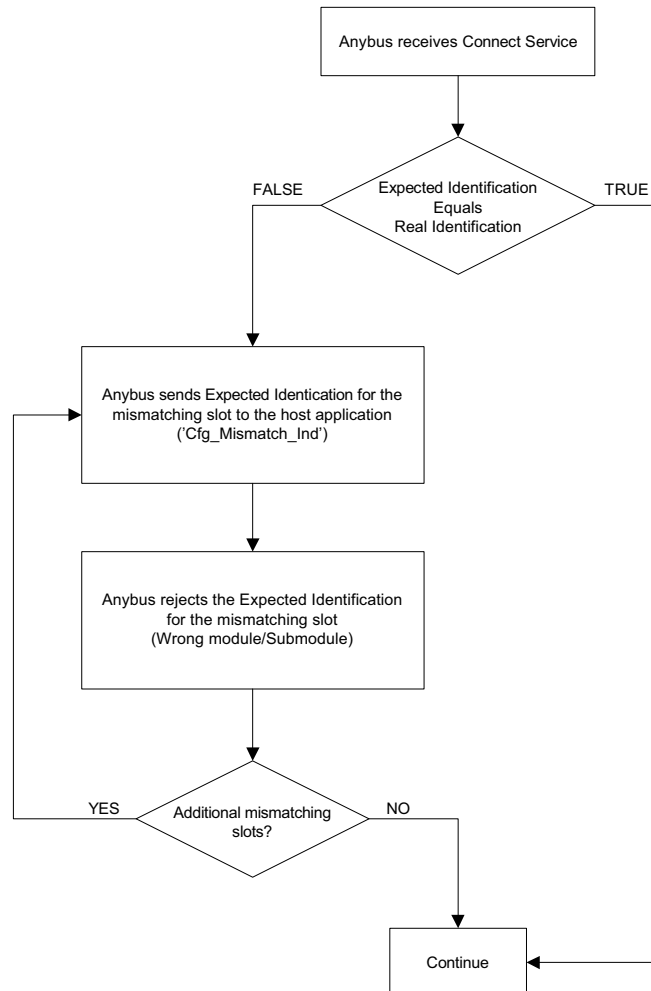


See also...

- “Application Data Instances (ADIs)” on page 17
- “PROFINET IO Object (F6h)” on page 134
- “Command Details: Get_Record” on page 138
- “Command Details: Set_Record” on page 140
- “Command Details: Get_IM_Record” on page 142
- “Command Details: Set_IM_Record” on page 144

D.2 Flowchart - Configuration Mismatch (RI)

This flowchart illustrates how the Anybus module handles a configuration mismatch.

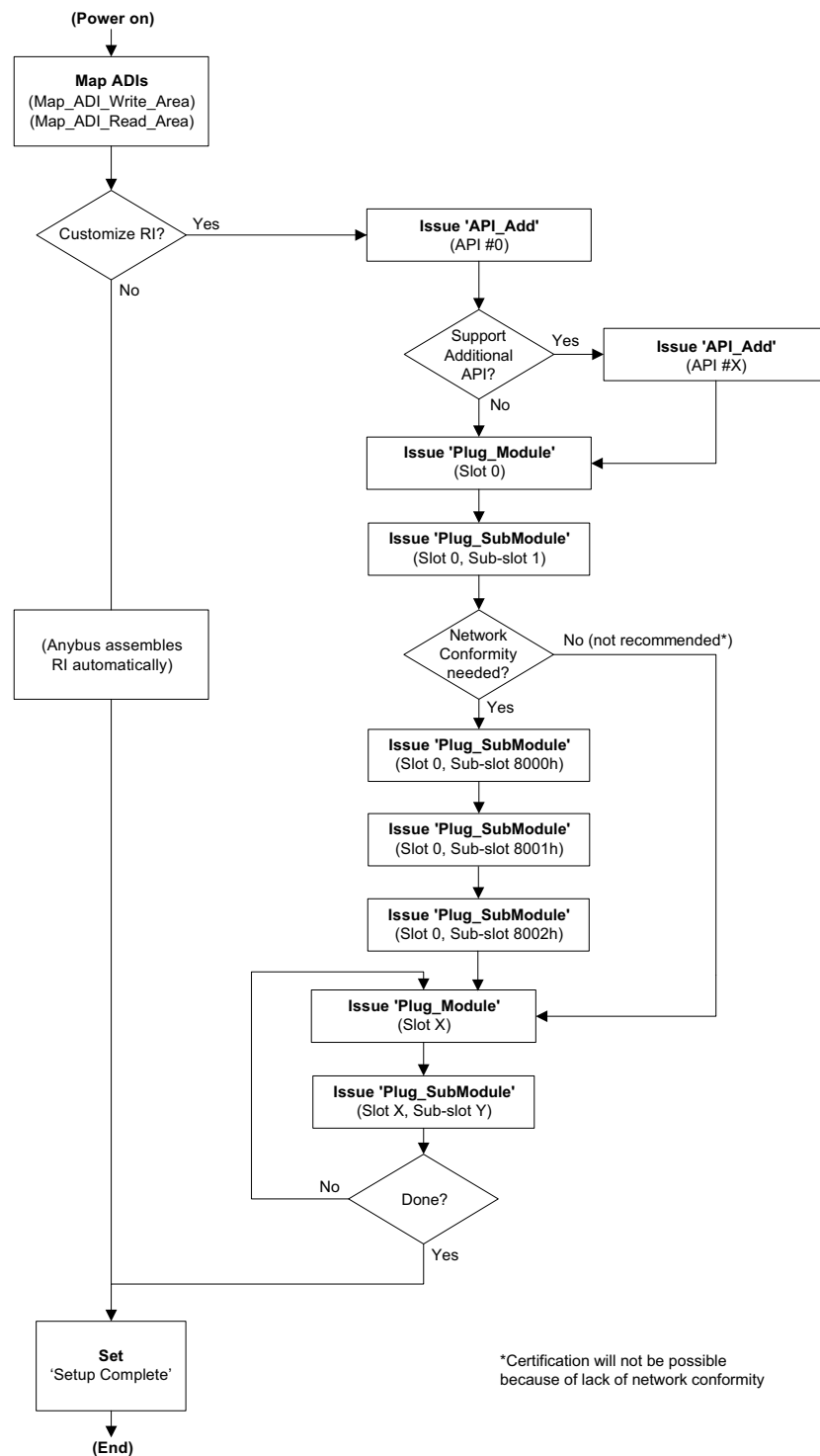


See also...

- “Process Data” on page 18
- “Configuration Mismatch” on page 24
- “Command Details: Cfg_Mismatch_Ind” on page 148
- “Flowchart - Establishment of Real Identification (RI)” on page 174

D.3 Flowchart - Establishment of Real Identification (RI)

This flowchart illustrates the establishment of the Real Identification.



See also...

- “Process Data” on page 18
- “Real Identification (RI)” on page 23
- “Flowchart - Configuration Mismatch (RI)” on page 173

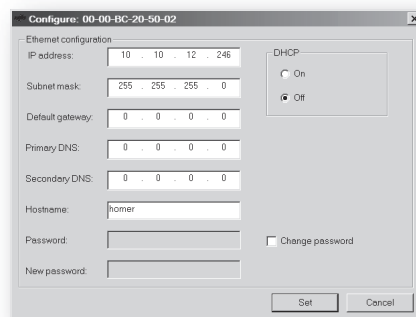
E. HICP (Host IP Configuration Protocol)

E.1 General

The module supports the HICP protocol used by the Anybus IPconfig utility for changing settings, e.g. IP address, Subnet mask, and enable/disable DHCP. Anybus IPconfig can be downloaded free of charge from the HMS website, www.anybus.com. This utility may be used to access the network settings of any Anybus product connected to the network via UDP port 3250.

E.2 Operation

Upon starting the program, the network is scanned for Anybus products. The network can be re-scanned at any time by clicking 'Scan'.



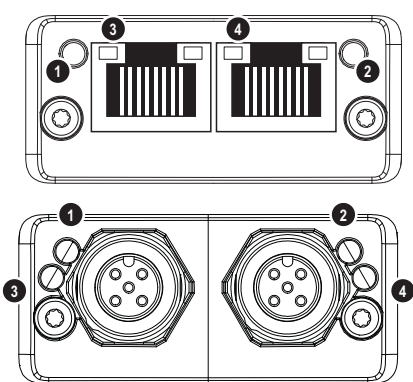
To alter the network settings of the module, double-click on its entry in the list. A window will appear, containing the settings for the module.

Validate the new settings by clicking 'Set', or click 'Cancel' to cancel all changes.

Optionally, the configuration can be protected from unauthorized access by a password. To enter a password, click on the 'Change password' checkbox, and enter the password under 'New password'.

F. Technical Specification

F.1 Front View

#	Item		Connectors
1	Network Status LED ^a		Ethernet, RJ45
2	Module Status LED ^a		
3	Link/Activity LED (port 1)		M12, female
4	Link/Activity LED (port 2)		

a. Test sequences are performed on the Network and Module Status LEDs during startup

Network Status LED

LED State	Description	Comments
Off	Offline	- No power - No connection with IO Controller
Green	Online (RUN)	- Connection with IO Controller established - IO Controller in RUN state
Green, flashing	Online (STOP)	- Connection with IO Controller established - IO Controller in STOP state

Module Status LED

LED State	Description	Comments
Off	Not Initialized	No power - <i>or</i> - Module in 'SETUP' or 'NW_INIT' state
Green	Normal Operation	Module has shifted from the 'NW_INIT' state
Green, 1 flash	Diagnostic Event(s)	Diagnostic event(s) present
Green, 1 Hz	DCP Flash	Used by engineering tools to identify the node on the network
Red	Exception Error	Module in state 'EXCEPTION'
Red, 1 flash	Configuration Error	Expected Identification differs from Real Identification
Red, 2 flashes	IP Address Error	IP address not set
Red, 3 flashes	Station Name Error	Station Name not set
Red, 4 flashes	Internal Error	Module has encountered a major internal error

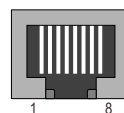
LINK/Activity LED

LED State	Description	Comments
Off	No Link	No link, no communication present
Green	Link	Ethernet link established, no communication present
Green, flickering	Activity	Ethernet link established, communication present

Ethernet Interface

The Ethernet interface operates at 100 Mbit, full duplex, as required by PROFINET.

Pin no	Description
1	TD+
2	TD-
3	RD+
4, 5, 7, 8	Connected to chassis ground over serial RC circuit
6	RD-
Housing	Cable Shield

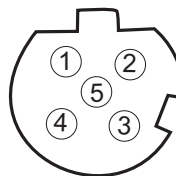


IMPORTANT:

For information on how to connect the PROFINET cable, see “Protective Earth (PE) Requirements” on page 179.

M12 Connectors, Code D, Female

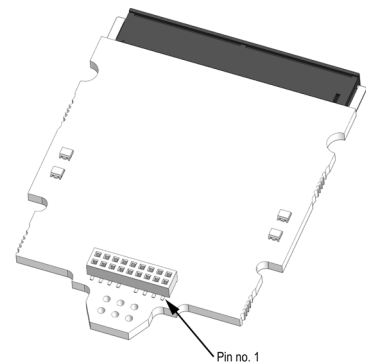
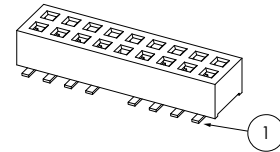
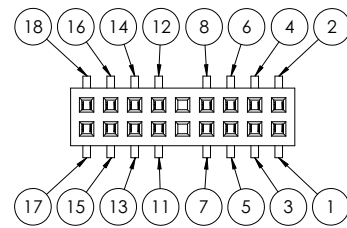
Pin	Name	Description
1	TXD+	Transmit positive
2	RXD+	Receive positive
3	TXD-	Transmit negative
4	RXD-	Receive negative
5 (Thread)	Shield	Shield



F.2 Network Connector, Brick Version

The Anybus CompactCom 30 PROFINET IO 2-Port can also be acquired in a brick version, without a fieldbus connector, but instead with a pin connector to the carrier board (the host device). The concept and assembly are described in the Anybus CompactCom Brick and without Housing Design Guide (Doc. Id. HMSI-168-30).

Pin no.	Signal	Port no.
1	Shield	2
2	TXD+	
3	TXD-	
4	Shield	
5	Shield	
6	RXD-	
7	RXD+	
8	Shield	
11	Shield	1
12	TXD+	
13	TXD-	
14	Shield	
15	Shield	
16	RXD-	
17	RXD+	
18	Shield	



The differential signal pairs must have a controlled differential impedance of 100 Ohm (+/-10%). The two traces of each pair should have controlled lengths, so that TXD+ is 1.27 mm longer than TXD-, while RXD- is 1.27 mm longer than RXD+. The shields must be used as reference planes for the respective port. Connect the shields to the CompactCom brick header so that the onboard RC filters work properly. The brick has individual filters for the two ports and a minimum isolation distance of 1.4 mm between the ports.

PROFINET - apart from other Ethernet based networks - allows direct connection of the shields to PE/FE. This can for example be done by letting the network connectors contact the product chassis.

F.3 Protective Earth (PE) Requirements

In order to ensure proper EMC behavior, the module must be properly connected to protective earth via the PE pad/PE mechanism described in the general Anybus CompactCom Hardware Design Guide.

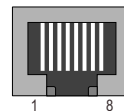
HMS Industrial Networks does not guarantee proper EMC behavior unless these PE requirements are fulfilled.

Note: The shield of the RJ45 connector is not connected directly to PE. As all nodes in a PROFINET network have to share chassis ground connection, the PROFINET cable shield has to be connected to the chassis ground at each node in the network.

For further information, see “PROFINET Installation Guideline for Cabling and Assembly”, order no. 8.072, available for download at www.PROFINET.com.

RJ45 Connector for PROFINET

Pin no	Description
1	TD+
2	TD-
3	RD+
4, 5, 7, 8	Connected to chassis ground over serial RC circuit
6	RD-
Housing	Cable Shield



F.4 Power Supply

Supply Voltage

The module requires a regulated 3.3 V power source as specified in the general Anybus CompactCom Hardware Design Guide.

Power Consumption

The Anybus CompactCom 30 PROFINET IO 2-Port is designed to fulfil the requirements of a Class B module. For more information about the power consumption classification used on the Anybus CompactCom platform, consult the general Anybus CompactCom Hardware Design Guide.

The current hardware design consumes up to 380 mA¹.

F.5 Environmental Specification

Consult the Anybus CompactCom Hardware Design Guide for further information.

F.6 EMC Compliance

Consult the Anybus CompactCom Hardware Design Guide for further information.

1. Note that in line with HMS policy of continuous product development, we reserve the right to change the exact power requirements of this product without prior notification. Note however that in any case, the Anybus CompactCom 30 PROFINET IO 2-Port will remain as a Class B module.

G. Timing & Performance

G.1 General Information

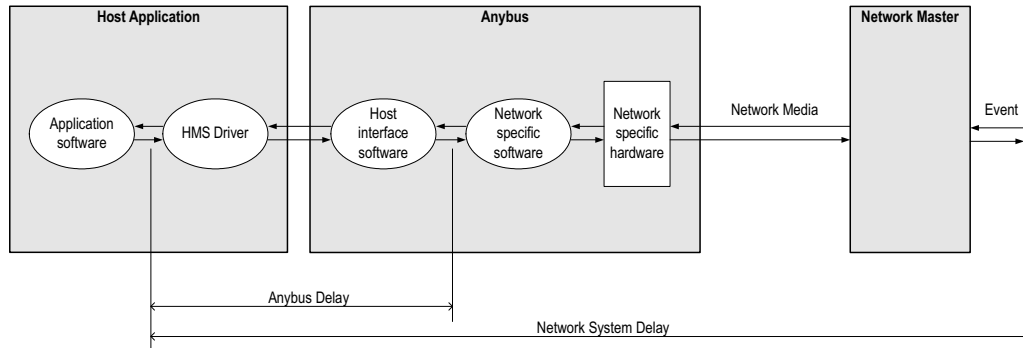
This chapter specifies timing and performance parameters that are verified and documented for the Anybus CompactCom 30 PROFINET IO 2-Port.

The following timing aspects are measured:

Category	Parameters	Page
Startup Delay	T1, T2	Please consult the Anybus CompactCom Software Design Guide, App. B.
NW_INIT Delay	T3	
Telegram Delay	T4	
Command Delay	T5	
Anybus Read Process Data Delay (Anybus Delay)	T6, T7, T8	
Anybus Write Process Data Delay (Anybus Delay)	T12, T13, T14	
Network System Read Process Data Delay (Network System Delay)	T9, T10, T11	182
Network System Write Process Data Delay (Network System Delay)	T15, T16, T17	182

G.2 Process Data

G.2.1 Overview



G.2.2 Anybus Read Process Data Delay (Anybus Delay)

The Read Process Data Delay (labelled ‘Anybus delay’ in the figure above) is defined as the time measured from just before new data is buffered and available to the Anybus host interface software, to when the data is available to the host application (just after the new data has been read from the driver).

Please consult the Anybus CompactCom Software Design Guide, Appendix B, for more information.

G.2.3 Anybus Write Process Data Delay (Anybus Delay)

The Write Process Data Delay (labelled ‘Anybus delay’ in the figure) is defined as the time measured from the point the data is available from the host application (just before the data is written from the host application to the driver), to the point where the new data has been forwarded to the network buffer by the Anybus host interface software.

Please consult the Anybus CompactCom Software Design Guide, Appendix B, for more information.

G.2.4 Network System Read Process Data Delay (Network System Delay)

The Network System Read Process Data Delay (labelled 'Network System Delay' in the figure), is defined as the time measured from the point where an event is generated at the network master to when the corresponding data is available to the host application (just after the corresponding data has been read from the driver).

Parameter	Description	Avg.	Max.	Unit.
T9	Network System Read Process Data delay, 8 ADIs (single UINT8)	3.3	4.64	ms
T10	Network System Read Process Data delay, 16 ADIs (single UINT8)	5.0	7.5	ms
T11	Network System Read Process Data delay, 32 ADIs (single UINT8)	4.5	8.0	ms

Conditions:

Parameter	Conditions
Application CPU	-
Timer system call interval	1 ms
Driver call interval	0.2... 0.3 ms
No. of ADIs (single UINT8) mapped to Process Data in each direction	8, 16 and 32
Communication	Parallel
Telegram types during measurement period	Process Data only
Bus load, no. of nodes, baud rate etc.	Normal

G.2.5 Network System Write Process Data Delay (Network System Delay)

The Network System Write Process Data Delay (labelled 'Network System Delay' in the figure), is defined as the time measured from the time after the new data is available from the host application (just before the data is written to the driver) to when this data generates a corresponding event at the network master.

Parameter	Description	Avg.	Max.	Unit.
T15	Network System Write Process Data delay, 8 ADIs (single UINT8)	3.3	5.0	ms
T16	Network System Write Process Data delay, 16 ADIs (single UINT8)	5.1	7.2	ms
T17	Network System Write Process Data delay, 32 ADIs (single UINT8)	4.25	7.5	ms

Conditions: as in "Network System Read Process Data Delay (Network System Delay)" on page 182.

H. Conformance Test Guide

H.1 General

When using the default settings of all parameters, the Anybus CompactCom 30 PROFINET IO module is precertified for network compliance. This precertification is done to ensure that your product *can* be certified, but it does not mean that your product will not require certification.

Any change in the parameters in the GSD file, supplied by HMS, will require a certification. A vendor ID can be obtained from PNO and is compulsory for certification. This chapter provides a guide for successful conformance testing your product, containing the Anybus CompactCom 30 PROFINET IO module, to comply with the demands for network certification set by the PNO.

Independent of selected operation mode, the actions described in this appendix have to be accounted for in the certification process. The identity of the product needs to be changed to match your company and device.

IMPORTANT: *This appendix provides guidelines and examples of what is needed for certification. Depending on the functionality of your application, there may be additional steps to take. Please contact HMS Industrial Networks at www.anybus.com for more information.*

H.2 Reidentifying Your Product

After successful setting of the “Setup Complete” attribute in the Anybus Object (01h), the Anybus module asks for identification data from the host PROFINET IO Object (F6h). Therefore, the attributes listed below shall be implemented and proper values returned.

Object/Instance	Attribute	Explanation	Default	Customer sample	Comment
PROFINET IO Object (F6h), Instance 1	#1, Device ID	With this attribute you set the Device ID of the device	Device ID: 0009h	Device ID: YYYYh	This information must match the keys of the “Devicelidentity” of the GSD-file.
PROFINET IO Object (F6h), Instance 1	#2, Vendor ID	With this attribute you set the Vendor ID of the device	Vendor ID: 010Ch (HMS)	Vendor ID: XXXXh	Note that the GSD file keyword “VendorName” must correspond to the Vendor ID value.
PROFINET IO Object (F6h), Instance 1	#3, Station Type	With this attribute you set the station type of the device	“ABCC-PRT (2-Port)”	“Cust-PNIO-Dev”	This information matches, in the case of Anybus CompactCom PROFINET IO, GSD keywords “DNS_CompatibleName” and “OrderNumber”. The Station Type must be equal to the “DNS_CompatibleName”, but it is allowed to have a completely different “OrderNumber”, see also I&M Order ID below.
PROFINET IO Object (F6h), Instance 1	#8, I&M Order ID	With this attribute you set the Order ID that is used in the I&M data	“ABCC-PRT (2-Port)”	“Cust-PNIO-Dev”	This information must match the keys of the “OrderNumber” of the GSD-file.
PROFINET IO Object (F6h), Instance 1	#10, I&M Hardware Revision	With this attribute you set the I&M Hardware Revision	(Hardware Rev.)	“0002h”	This information must match the keys of the “HardwareRelease” of the GSD-file.
PROFINET IO Object (F6h), Instance 1	#11, I&M Software Revision	With this attribute you set the I&M Software Revision	(Software Rev.)	“V2.5.3”	This information must match the keys of the “SoftwareRelease” of the GSD-file. The attribute is a 4 byte array in the PROFINET IO object, see page 135 for more information.
PROFINET IO Object (F6h), Instance 1	#19, System Description	With this attribute you set the description of the system	“HMS Industrial Networks Anybus- CompactCom”	“Customer HMI Interface Module”	This information can be read by means of SNMP from the network side.
PROFINET IO Object (F6h), Instance 1	#20, Interface Description	With this attribute you set the description of the interface	“PROFINET IO Interface”	“PROFINET IO Interface”	
PROFINET IO Object (F6h), Instance 1	#22, System Contact	With this attribute you set the system contact information	“www.anybus.com”	“www.customer.com”	This information can be read by means of SNMP from the network side, via the MIB-II.

Additional GSD File Information

The GSD file keyword “ProductFamily” shall correspond to the vendor’s name of the device.

The GSD file keyword “MainFamily” lists the kinds of devices for which the product shall be listed. As of GSD specification v2.25, the following “families” are available:

“General”, “Drives”, “Switching Devices”, “I/O”, “Valves”, “Controllers”, “HMI”, “Encoders”, “NC/RC”, “Gateway”, “PLCs”, “Ident Systems”, “PA Profiles”, “Network Components”, “Sensors”.

H.3 Factory Default Reset

Reset command to Application Object (FFh) must be supported

When PROFINET IO modules are delivered, they are required to be in their “Factory Default” state. For PROFINET devices this means that their Station Name is empty (“”) and that the IP-suite is not assigned (IP 0.0.0.0). When a Factory Default Reset command is received from the network, the Anybus module will erase all IP and Station Name information and inform the host application that hardware or software reset of the Anybus module is required. This is done by sending a Reset command to the Application Object (FFh) of the host (Power-on + Factory Default). For more details, please consult the Anybus CompactCom Software Design Guide.

H.4 IP Address

Normally the IP numbers of PROFINET IO devices are assigned via the PROFINET network via DCP (Discovery and Configuration Protocol). HMS recommends not using the Network Configuration Object (04h, instances #3 - #6) during the initialization phase for PROFINET modules, unless the end user has requested the IP address to be set to a specific value (by for example using a keypad). The reason is that when a factory default reset command is received from the PROFINET network (via DCP) the node must be available after a hardware or software reset with the default IP-address (0.0.0.0).

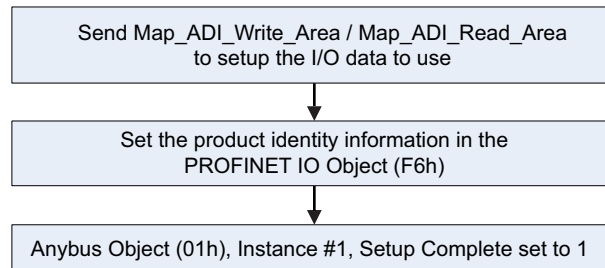
H.5 Station Name

Normally the Station Name of a PROFINET device is assigned by the end user via the PROFINET DCP protocol. HMS recommends not using the Network Configuration Object, instance #15 during the initialization phase for PROFINET modules. If this attribute is used, it is recommended that it is sent explicitly when the end user changes the Station Name with e.g. a keypad. The reason is that when a factory default reset command is received from the PROFINET network (via DCP), the node must be available after a hardware or software reset with the default Station Name (“”).

IMPORTANT: *The Anybus module will forward all information about the connection being established to the IO Controller, as commands to the host PROFINET IO Object (F6h). Even though the host application might not need this information, a response must always be generated (such as 05h, “Unsupported command”).*

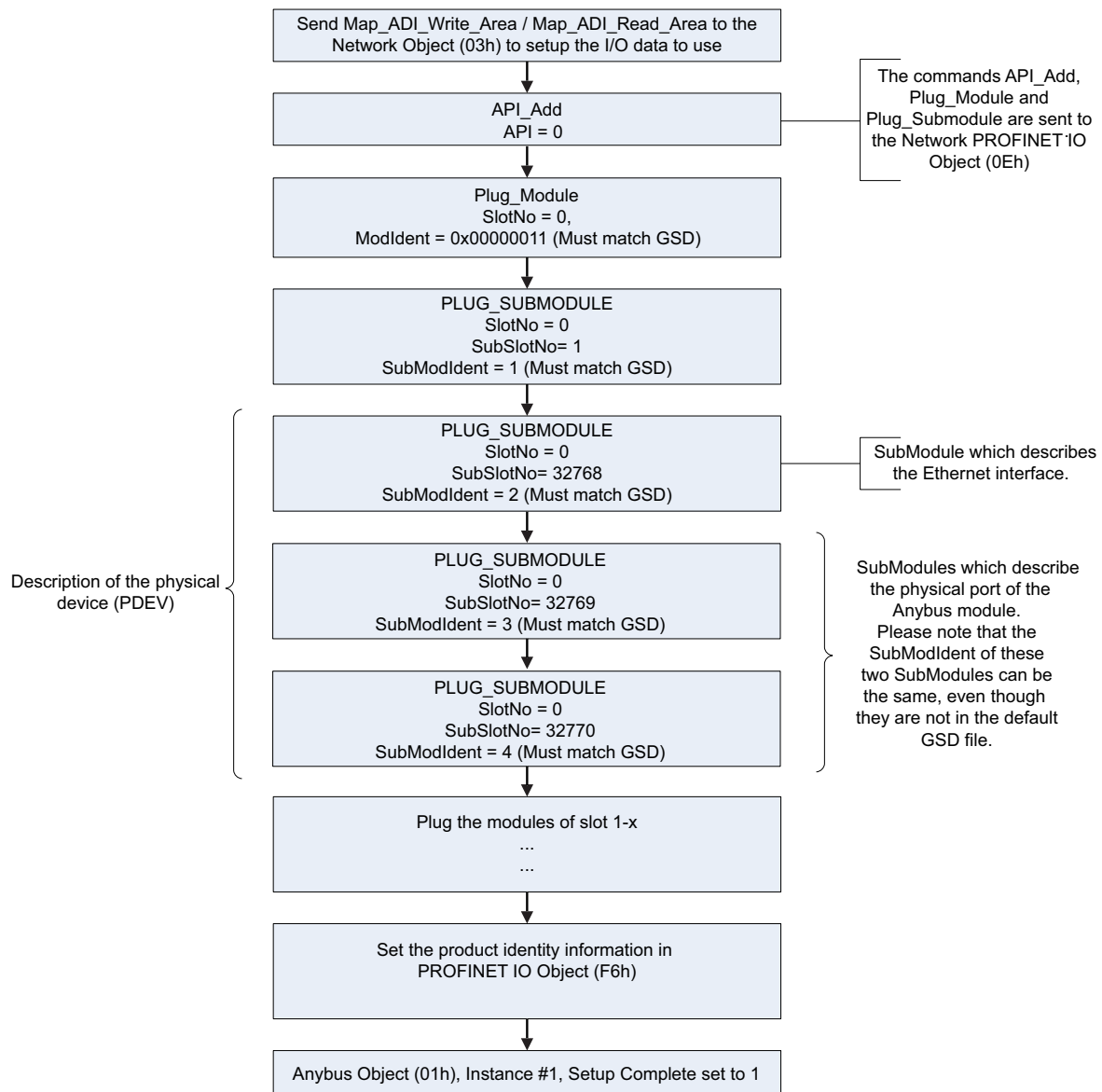
H.6 Certification in Generic Anybus Mode

In Generic Anybus Mode (when the command API_add in the Network PROFINET IO Object (0Eh) is not used) there is normally nothing that needs to be considered apart from what is mentioned earlier in this appendix. The default HMS GSD file has to be modified with respect to the identity of the product and this requires a certification of the product.



H.7 Certification in Advanced Mode

In advanced mode (Network PROFINET IO Object (0Eh) is used), the most important thing is to use a Device Access Point (DAP) that conform to PROFINET IO Specification v2.0 or later (DAP2). From specification version 2.0 it is possible to describe the physical Ethernet interface and its ports (PDEV, or Physical Device) with a special mechanism. This is done with special submodules at slot 0 (the module at slot 0 is the access point for the device). HMS recommends following the flow below for setting up a DAP2.

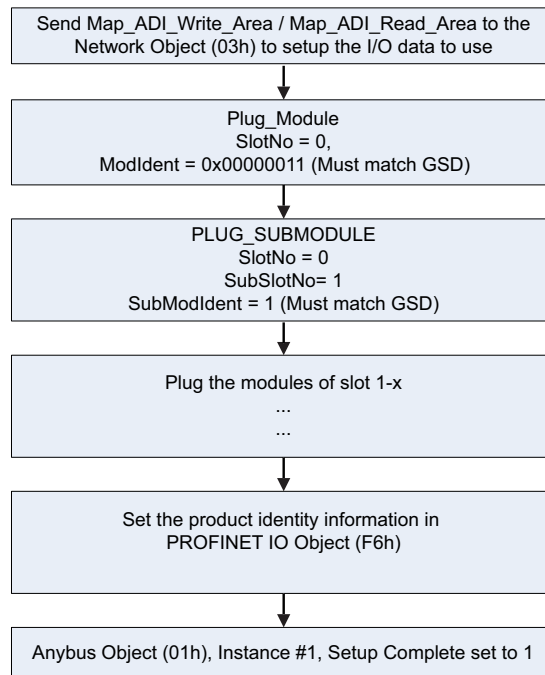


The figure shows how to set up a PROFINET compatible DAP. Please note that for some commands only the relevant parameters are shown.

Please note that the values of “SubModIdent” in the above flowchart are the values of the default HMS GSD file. They can be changed if necessary, but there is no real need for it, the important thing is that it matches the GSD file. To be able to pass the PROFINET conformance test a “DAP2” is mandatory. On the market there still are some PROFINET IO controllers not supporting PROFINET IO specification v2.0 or later. These controllers cannot use a DAP2. Therefore, it might be necessary to support also a DAP containing no PDEV (i.e. the three last PLUG_SUBMODULE commands are not issued).

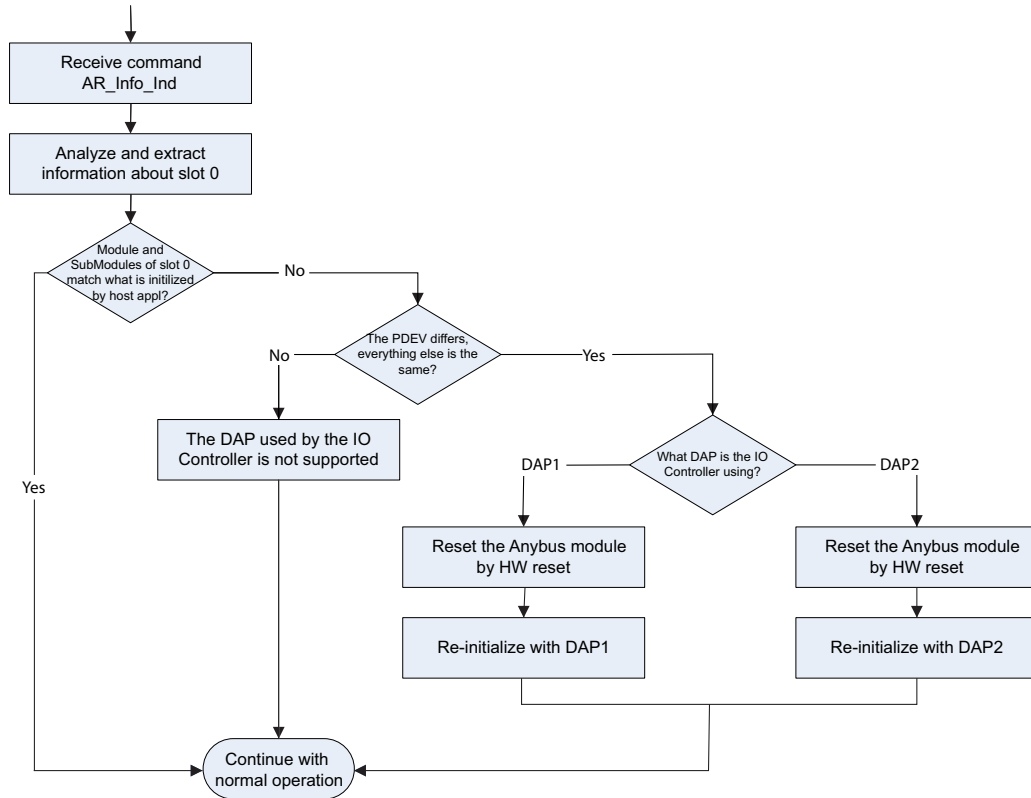
This is called a “migration” DAP. In the default GSD file there is such functionality. In the case of advanced mode this can be implemented in either of these two ways:

1. The end user decides that reverse compatibility is necessary and selects this, for example with a parameter on a hand panel. The host application performs a hardware or a software reset of the Anybus module and skips the last three Plug_Submodules as shown in the figure on page 187, resulting in the flow shown in the figure below:



The figure shows a DAP without a PDEV, for reverse compatibility only (please note that for some commands only the relevant parameters are shown.)

2. The host application uses the AR_Info_Ind command sent from the Anybus module to the host PROFIBUS IO Object (F6h) and can thus analyze the connection which is being established by the IO Controller. If the IO Controller is trying to use the DAP which has not been plugged the host application can do a hardware or software reset of the Anybus module and reinitialize the Anybus module with the correct DAP (with or without PDEV), as described in the figure below:



Note: The command AR_Info_Ind is always sent to the host PROFINET IO Object (F6h), if it is not needed/used the host application responds with 05h, "Unsupported command".

The figure shows a flowchart of the functionality to swap DAPs depending on what the IO Controller is using.

Once the DAP has been plugged into slot 0, the other slots can be populated. Of some importance with these other modules, is that the Module Identification Number must uniquely define the kind of module (for example, a digital input module must not have the same module identification number as a digital output module). There is one exception to this rule for the DAP. It is allowed to have a DAP with or without a PDEV, but with the same module identification number.

HMS recommends that the host application, if possible, store, in nonvolatile memory, the DAP used last time and uses that DAP after power-cycle. The reason for doing so is to reduce time for connection establishment. If no DAP is stored DAP2 shall be used. If it is not possible for the host application to store the most recently used DAP, the host application should always plug DAP2 initially.

I. Copyright Notices

This product includes software developed by Carnegie Mellon, the Massachusetts Institute of Technology, the University of California, and RSA Data Security:

Copyright 1986 by Carnegie Mellon.

Copyright 1983,1984,1985 by the Massachusetts Institute of Technology

Copyright (c) 1988 Stephen Deering.

Copyright (c) 1982, 1985, 1986, 1992, 1993

The Regents of the University of California. All rights reserved.

This code is derived from software contributed to Berkeley by Stephen Deering of Stanford University.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Copyright (C) 1990-2, RSA Data Security, Inc. All rights reserved.

License to copy and use this software is granted provided that it is identified as the "RSA Data Security, Inc. MD4 Message-Digest Algorithm" in all material mentioning or referencing this software or this function.

License is also granted to make and use derivative works provided that such works are identified as "derived from the RSA Data Security, Inc. MD4 Message-Digest Algorithm" in all material mentioning or referencing the derived work.

RSA Data Security, Inc. makes no representations concerning either the merchantability of this software or the suitability of this software for any particular purpose. It is provided "as is" without express or implied warranty of any kind.

These notices must be retained in any copies of any part of this documentation and/or software.

Copyright (C) 1991-2, RSA Data Security, Inc. Created 1991. All rights reserved.

License to copy and use this software is granted provided that it is identified as the "RSA Data Security, Inc. MD5 Message-Digest Algorithm" in all material mentioning or referencing this software or this function.

License is also granted to make and use derivative works provided that such works are identified as "derived from the RSA Data Security, Inc. MD5 Message-Digest Algorithm" in all material mentioning or referencing the derived work.

RSA Data Security, Inc. makes no representations concerning either the merchantability of this software or the suitability of this software for any particular purpose. It is provided "as is" without express or implied warranty of any kind.

These notices must be retained in any copies of any part of this documentation and/or software.