

Fieldbus Appendix

Anybus-IC PROFINET IO

Doc.Id. SCM-1200-036
Rev 2.00

Important User Information

This document is intended to provide a good understanding of the functionality offered by Anybus-IC PROFINET. The document only describes the features that are specific to the Anybus-IC PROFINET. For general information regarding the Anybus-IC, consult the Anybus-IC 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 specific functionality may require in-depth knowledge in PROFINET networking internals and/or information from the official PROFINET specifications. In such cases, the people responsible for the implementation of this product should either obtain the PROFINET 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 application 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 application meets 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

Preface	About This Document	
	How To Use This Document.....	10
	Related Documents	10
	Document History	11
	Conventions & Terminology	11
	Sales and Support	12
 Chapter 1	 About the Anybus-IC PROFINET	
	General Information	13
	Features	13
 Chapter 2	 Basic Operation	
	Network and Software Requirements	14
	Status Indicators (Fieldbus Specific Output)	15
	Switches (Fieldbus Specific Input).....	16
	Network Reset Handling.....	17
	<i>General Information</i>	17
	<i>Reset To Factory</i>	17
	Filesystem	18
	<i>General Information</i>	18
	<i>Filesystem Overview</i>	19
	<i>System Files</i>	19

Chapter 3 Network Configuration

Physical Link Settings	20
TCP/IP Settings	20
IP Access Control.....	21
Anybus IPconfig (HICP).....	22
ARP Gleaning	22

Chapter 4 PROFINET I/O Implementation

General Information	23
PROFINET Compliance	24
Electronic Data Sheet (GSD file).....	25
Device Identity.....	26
Modes of Operation.....	27
<i>Mode selection</i>	27
<i>Module Initialized with 'Module Mode' set to 'Fieldbus specific init'</i>	28
Diagnostics & Alarms	30
Initial Parameters	31
Configuration Data.....	31
Identification & Maintenance (I&M)	32
<i>Structure of the Block Header</i>	32
<i>Structure of I&M Read response</i>	32
<i>Structure of I&M Write request</i>	33
<i>Structure of I&M0 data</i>	33
<i>Structure of I&M1 data</i>	33
<i>Structure of I&M2 data</i>	33
<i>Structure of I&M3 data</i>	34
<i>Structure of I&M4 data</i>	34
Establishing a PROFINET IO connection	35
Fast Start Up	37
<i>General Information</i>	37
<i>Fast Start Up Configuration with STEP7</i>	38

Chapter 5 Modbus-TCP Implementation

General Information	40
Supported Function Codes	40
Exception Codes.....	40
Register Map.....	41
<i>Fieldbus Input Data (FB In)</i>	41
<i>Fieldbus Output Data (FB Out)</i>	41

Chapter 6 E-mail Client

General Information	42
Event-Triggered Messages	42
E-mail Definitions	43

Chapter 7 FTP Server

General Information	44
FTP Connection Example (Windows Explorer).....	45

Chapter 8 Telnet Server

General Information	46
General Commands	47
Diagnostic Commands	48
File System Operations.....	48

Chapter 9 Web Server

General Information	51
Authentication.....	52
Content Types.....	53

Chapter 10 Server Side Include (SSI)

General Information	54
Functions	55
Changing SSI output.....	63
<i>SSI Output String File</i>	63
<i>Temporary SSI Output change</i>	64

Chapter 11 Fieldbus Specific Parameters

General	65
Communication Settings	65
Server Settings	65
E-mail Client	66
File System	66
Modbus-TCP Related Parameters	66
PROFINET IO Related Parameters	67
Application Parameters	67
General	68
<i>FB Status (Parameter #100)</i>	68
<i>FB init (Parameter #101)</i>	69
<i>FB Password (Parameter #102)</i>	70
<i>DIP switch SSC (Parameter #104)</i>	70
<i>MAC address (Parameter #116)</i>	71
<i>Serial Number (Parameter #141)</i>	71
Communication Settings	72
<i>IP address cfg (Parameter #103)</i>	72
<i>IP address act (Parameter #105)</i>	72
<i>Subnet mask cfg (Parameter #106)</i>	72
<i>Subnet mask act (Parameter #107)</i>	73
<i>GW address cfg (Parameter #108)</i>	73
<i>GW address act (Parameter #109)</i>	73
<i>DHCP enable cfg (Parameter #114)</i>	74
<i>DHCP enable act (Parameter #115)</i>	74
<i>Data rate act (Parameter #118)</i>	74
<i>Duplex Act (Parameter #120)</i>	75
<i>HICP Enable (Parameter #136)</i>	75
<i>HICP Password (Parameter #137)</i>	75
<i>ARP Gleaning Enable (Parameter #144)</i>	76
Server Settings	77
<i>Web Srv Enable (Parameter #121)</i>	77
<i>FTP Srv Enable (Parameter #122)</i>	77
<i>Telnet Srv Enable (Parameter #123)</i>	78
E-mail Client	79
<i>SMTP Srv Address (Parameter #126)</i>	79
<i>Triggered Emails (Parameter #127)</i>	79
<i>SMTP Errors (Parameter #128)</i>	79
<i>Send Email (Parameter #129)</i>	80
<i>SMTP User Name (Parameter #142)</i>	80
<i>SMTP Password (Parameter #143)</i>	80
File System	81
<i>Admin Mode Cfg (Parameter #124)</i>	81
<i>Admin Mode Act (Parameter #125)</i>	81
<i>VFS Enable (Parameter #130)</i>	82
<i>RAM-Disc Path (Parameter #131)</i>	82
Modbus-TCP related Parameters	83
<i>MB/TCP Conn TO (Parameter #132)</i>	83
<i>MB/TCP Enable (Parameter #133)</i>	83
<i>In bit size (Parameter #134)</i>	84

<i>Out bit size (Parameter #135)</i>	84
PROFINET IO Related Parameters	85
<i>PRT version numb (Parameter #160)</i>	85
<i>PRT serial numb (Parameter #161)</i>	85
<i>PRT vendor id (Parameter #162)</i>	86
<i>PRT device id (Parameter #163)</i>	86
<i>PRT station type (Parameter #164)</i>	86
<i>Port 1 MAC addr (Parameter #165)</i>	87
<i>PRT station name (Parameter #166)</i>	87
<i>ArCheckInd Data (Parameter #167)</i>	87
<i>CfgMisMaIndData (Parameter #168)</i>	89
<i>ArInfoInd Data (Parameter #169)</i>	90
<i>EndOfPrmInd Data (Parameter #170)</i>	91
<i>ArOfflInd Data (Parameter #171)</i>	92
<i>ArAbortInd Data (Parameter #172)</i>	93
<i>PlugSubModFailed (Parameter #173)</i>	94
<i>PnioInitErrCode (Parameter #174)</i>	94
Application Parameters (Parameters #200-#299)	95

Chapter 12 Fieldbus Interface

General	96
Application Connector Signals	96
RJ45 Pinout	96
Typical Implementation	97
FastJack Connectors	97

Appendix A Application Parameters

General Information	98
Creating an Application Parameter	99
<i>Query - "Application Parameter Object"</i>	99
<i>Response - "Application Parameter Object"</i>	101
<i>Example</i>	102
Mapping an Application Parameter to PROFINET	103
<i>Query - "PROFINET Mapping Object"</i>	103
<i>Response - "PROFINET Mapping Object"</i>	104
<i>Example</i>	104

Appendix B HMS Object Implementation

General Information	105
Application Parameter Object (Class 85h)	106
<i>General Information</i>	106
<i>Class Attributes</i>	106
<i>Instance Attributes</i>	106
<i>Create (Class Service)</i>	107
File System Object (Class 86h)	108
<i>General Information</i>	108
<i>Class Attributes</i>	108
<i>Instance Attributes</i>	108
<i>File Open (Class Service)</i>	109
<i>File Close (Class Service)</i>	109
<i>File Delete (Class Service)</i>	110
<i>File Copy (Class Service)</i>	110
<i>File Move (Class Service)</i>	111
<i>File Rename (Class Service)</i>	111
<i>File Read (Instance Service)</i>	112
<i>File Write (Instance Service)</i>	112
Fieldbus Object (Class A0h)	113
<i>General Information</i>	113
<i>Class Attributes</i>	114
<i>Instance Attributes</i>	116
<i>API Add (Class Service)</i>	117
<i>Plug Module (Class Service)</i>	118
<i>Plug Submodule (Class Service)</i>	119
<i>Pull Module (Class Service)</i>	120
<i>Pull Submodule (Class Service)</i>	121
<i>AR Abort (Class Service)</i>	122
<i>Fieldbus Errors</i>	123
Socket Object (Class A6h)	124
<i>General Information</i>	124
<i>Class Attributes</i>	124
<i>Instance Attributes</i>	124
<i>Create Socket (Class Service)</i>	125
<i>Close Socket (Class Service)</i>	125
<i>Bind (Instance Service)</i>	125
<i>Listen (Instance Service)</i>	126
<i>Accept (Instance Service)</i>	127
<i>Connect (Instance Service)</i>	127
<i>Receive (Instance Service)</i>	127
<i>Receive From (Instance Service)</i>	128
<i>Send (Instance Service)</i>	128
<i>Send To (Instance Service)</i>	129
<i>Socket Errors</i>	130
I&M Object (Class A7h)	131
<i>General Information</i>	131
<i>Class Attributes</i>	132
<i>Instance Attributes</i>	133
PROFINET Diagnostic Object (Class A8h)	134
<i>General Information</i>	134
<i>Class Attributes</i>	134
<i>Instance Attributes</i>	134

PROFINET IO <i>Specific Coding of Alarm Specific Fields</i>	135
<i>Send Process Alarm (Class Service)</i>	136
<i>Create Channel Diagnostic Alarm (Class Service)</i>	137
<i>Create Generic Diagnostic Alarm (Class Service)</i>	139
<i>Delete Diagnostic Entry (Instance Service)</i>	140
<i>Diagnostic Errors</i>	142
PROFINET Mapping Object (Class A9h).....	143
<i>General Information</i>	143
<i>Class Attributes</i>	143
<i>Instance Attributes</i>	143
<i>Create (Class Service)</i>	144
<i>Mapping Errors</i>	145

Appendix C Conformance Test Guide

General	146
Reidentifying Your Product	147
Factory Default Reset	148
IP Address	148
Station Name.....	148
Certification in Generic Anybus Mode	149
Certification in Advanced Mode	150

Appendix C Technical Specification

Electrical Specification.....	153
<i>Protective Earth (PE) Requirements</i>	153
<i>Power Supply</i>	153
Environmental Specification	153
EMC Compliance (CE)	153

About This Document

How To Use This Document

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

Related Documents

Document	Author
Anybus-IC Design Guide	HMS
Open Modbus/TCP Specification	Schneider Automation
PROFINET Technology and Application	PROFIBUS Nutzerorganisation (PNO)
GSDML Specification for PROFINET IO	
PROFINET IO specification	
PROFIBUS Guideline, Identification & Maintenance Functions	
SIMATIC NET PROFINET IO Softwarebeschreibung	ComDec
Structure of the Diagnostic Data Records (publication: A5E00337523-01)	SIEMENS
RFC 821	Network Working Group
RFC 1918	
-	-

Document History

Summary of Recent Changes (1.02... 2.00)

Change	Page(s)
New format	-
Updated frontpage information	-
Updated sales and support information	12
Corrected website link to PROFINET	24
Added information about how to associate a bitmap to a Device Access Point using the GSD file	25
Added information about Fast Start Up	37
A more thorough description of Initial Record Data	103
Corrected values for the PROFINET Mapping Object example	104
New conformance chapter	146

Revision List

Revision	Date	Author	Chapter	Description
1.00	2008-09-11	PeP	All	Initial revision
1.01	2008-11-06	HeS	A	Minor update
1.02	2009-11-16	KeL	2, 4, B	Miscellaneous minor updates
2.00	2012-05-22	KaD	P, 2, 4, A, C	New template, new chapter, miscellaneous updates and corrections

Conventions & Terminology

The following conventions are used throughout this document:

- Numbered lists provide sequential steps
- Bulleted lists provide information, not procedural steps
- The terms ‘Anybus’ or ‘module’ refers to the Anybus-IC module.
- The term ‘application’ refers to the device that hosts the Anybus-IC module.
- Hexadecimal values are written as NNNNh or 0xNNNN, where NNNN is the actual hexadecimal value.
- In this document, Modbus-TCP register numbers are specified using the PLC convention (base 1), i.e. register 0001 equals 0000 (zero) in the actual message frame. Register numbers are specified in decimal format unless indicated otherwise using the prefix/suffix described above.

Sales and Support

Sales		Support	
HMS Sweden (Head Office)			
E-mail:	sales@hms-networks.com	E-mail:	support@hms-networks.com
Phone:	+46 (0) 35 - 17 29 56	Phone:	+46 (0) 35 - 17 29 20
Fax:	+46 (0) 35 - 17 29 09	Fax:	+46 (0) 35 - 17 29 09
Online:	www.anybus.com	Online:	www.anybus.com
HMS North America			
E-mail:	us-sales@hms-networks.com	E-mail:	us-support@hms-networks.com
Phone:	+1-312 - 829 - 0601	Phone:	+1-312-829-0601
Toll Free:	+1-888-8-Anybus	Toll Free:	+1-888-8-Anybus
Fax:	+1-312-629-2869	Fax:	+1-312-629-2869
Online:	www.anybus.com	Online:	www.anybus.com
HMS Germany			
E-mail:	ge-sales@hms-networks.com	E-mail:	ge-support@hms-networks.com
Phone:	+49 (0) 721-989777-000	Phone:	+49 (0) 721-989777-000
Fax:	+49 (0) 721-989777-010	Fax:	+49 (0) 721-989777-010
Online:	www.anybus.de	Online:	www.anybus.de
HMS Japan			
E-mail:	jp-sales@hms-networks.com	E-mail:	jp-support@hms-networks.com
Phone:	+81 (0) 45-478-5340	Phone:	+81 (0) 45-478-5340
Fax:	+81 (0) 45-476-0315	Fax:	+81 (0) 45-476-0315
Online:	www.anybus.jp	Online:	www.anybus.jp
HMS China			
E-mail:	cn-sales@hms-networks.com	E-mail:	cn-support@hms-networks.com
Phone:	+86 (0) 10-8532-3183	Phone:	+86 (0) 10-8532-3023
Fax:	+86 (0) 10-8532-3209	Fax:	+86 (0) 10-8532-3209
Online:	www.anybus.cn	Online:	www.anybus.cn
HMS Italy			
E-mail:	it-sales@hms-networks.com	E-mail:	it-support@hms-networks.com
Phone:	+39 039 59662 27	Phone:	+39 039 59662 27
Fax:	+39 039 59662 31	Fax:	+39 039 59662 31
Online:	www.anybus.it	Online:	www.anybus.it
HMS France			
E-mail:	fr-sales@hms-networks.com	E-mail:	fr-support@hms-networks.com
Phone:	+33 (0) 3 68 368 034	Phone:	+33 (0) 3 68 368 033
Fax:	+33 (0) 3 68 368 031	Fax:	+33 (0) 3 68 368 031
Online:	www.anybus.fr	Online:	www.anybus.fr
HMS UK & Eire			
E-mail:	uk-sales@hms-networks.com	E-mail:	support@hms-networks.com
Phone:	+44 (0) 1926 405599	Phone:	+46 (0) 35 - 17 29 20
Fax:	+44 (0) 1926 405522	Fax:	+46 (0) 35 - 17 29 09
Online:	www.anybus.co.uk	Online:	www.anybus.com
HMS Denmark			
E-mail:	dk-sales@hms-networks.com	E-mail:	support@hms-networks.com
Phone:	+45 (0) 35 38 29 00	Phone:	+46 (0) 35 - 17 29 20
Fax:	+46 (0) 35 17 29 09	Fax:	+46 (0) 35 - 17 29 09
Online:	www.anybus.com	Online:	www.anybus.com
HMS India			
E-mail:	in-sales@hms-networks.com	E-mail:	in-support@hms-networks.com
Phone:	+91 (0) 20 40111201	Phone:	+91 (0) 20 40111201
Fax:	+91 (0) 20 40111105	Fax:	+91 (0) 20 40111105
Online:	www.anybus.com	Online:	www.anybus.com

About the Anybus-IC PROFINET

General Information

The Anybus-IC PROFINET communication module provides instant PROFINET IO connectivity via the generic Anybus-IC application 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. Additional protocols can be implemented on top of TCP/IP or UDP using the transparent socket interface.

The data exchange can be monitored via the built-in web server, Modbus-TCP, or using event triggered e-mail messages. SSI (Server Side Include) technology enables web pages and e-mail messages to carry dynamic content such as I/O data, configuration settings etc.

This product conforms to all aspects of the application interface defined in the Anybus-IC Design Guide, which means that no dedicated software support is needed to be able to support the Anybus-IC PROFINET. However, to be able to take advantage of optional network specific functionality, a certain degree of dedicated software support may be necessary.

Features

- Shielded (FTP) unshielded (UTP) cables
- PROFINET IO Real Time (RT) communications
- Modbus-TCP server
- Up to 144 bytes of fieldbus I/O in each direction
- Flexible file system with volatile and nonvolatile storage areas
- Security framework
- Integrated FTP server provides easy access to the built-in filesystem
- Web server with dynamic content
- E-mail client capabilities
- Server Side Include (SSI) capabilities
- Device identity customization

Basic Operation

Network and Software Requirements

As a member of the Anybus concept of interchangeable network products, the Anybus-IC PROFINET is compatible with any product that supports the Anybus-IC application interface. However, due to the nature of the PROFINET networking system, certain things need to be taken into account:

- The total (cyclic) fieldbus I/O size must exceed zero.
- Advanced functionality (optional) may require additional network specific software support.
- Acyclic communication requires network-specific software support (PROFINET).
- Application parameters cannot be accessed through Modbus-TCP.
- The flexible nature of the Anybus concept allows the application to modify the behavior on PROFINET in ways which contradict the generic GSDML file or in other ways voids network certification. Those responsible for the implementation of the final product should ensure that their level of implementation matches their own requirements and policies regarding network certification and interoperability.
- The use of advanced PROFINET specific functionality may require in-depth knowledge in PROFINET networking internals and/or information from the official PROFINET specifications. In such cases, those responsible for the implementation of the product should either obtain the PROFINET specification to gain sufficient knowledge or limit their implementation in such a way that this is not necessary.

For further information about the Anybus-IC software interface, consult the general Anybus-IC Design Guide.

See also...

- “PROFINET Compliance” on page 24

Status Indicators (Fieldbus Specific Output)

The fieldbus specific output is used for fieldbus specific status indications as follows:

Bit	Color	LED no.	Function
0 - 1	Green	1	Link/Activity
2 - 3	Red		
4 - 5	Green	2	Diagnostics
6 - 7	Red		
8 - 9	Green	3	Module Status
10 - 11	Red		
12 - 13	Green	4	Network Status
14 - 15	Red		

Definitions:

LED	Indication	Meaning	Comments
Link/Activity	Off	Not initialized (or no power)	-
	Green	Link sensed	-
	Green, flickering	Exchanging packets	-
	Alternating Red/Green	Self test in progress	-
Diagnostics	Off	Not initialized (or no power)	-
	Green, flashing (1Hz)	Identification	Used by engineering tools to identify nodes
	Red, flashing (1Hz)	Diagnostic event(s) available	-
	Alternating Red/Green	Self test in progress	-
Module Status	Off	Not initialized (or no power)	-
	Green	Normal operation	-
	Green, flashing (1Hz)	Network settings error	IP address or Station Name not set
	Red	Internal error	(Contact HMS)
	Red, flashing (1Hz)	Configuration Error	- Configuration mismatch; configuration received from IO Controller doesn't match the actual configuration - Malformed configuration received from IO Controller - Required initial parameter data not set properly
	Alternating Red/Green	Self test in progress	-
Network Status	Off	Offline (or no power)	-
	Green	Online	IO Controller in RUN
	Green, flashing (1Hz)		IO Controller in STOP/CLEAR
	Red	Internal error	(Contact HMS)
	Alternating Red/Green	Self test in progress	-

See also...

- Anybus-IC Design Guide (parameter #7 'LED State')

Switches (Fieldbus Specific Input)

The fieldbus specific input register is used for fieldbus specific configuration settings and supports two types of switches/coding.

- **BCD-coded input (BCD Switches)**

This type of switch can be used to specify the IP address in the range 192.168.0.1... 192.168.0.99. Subnet mask will be fixed to 255.255.255.0, and no gateway will be set (0.0.0.0).

Switch Value	Node Address
00	IP configuration specified elsewhere
01	192.168.0.1
02	192.168.0.2
03	192.168.0.3
...	...
97	192.168.0.97
98	192.168.0.98
99	192.168.0.99

- **Binary-coded input (Hex Switches, Binary Switches)**

This type of switch can be used to specify the IP address in the range 192.168.0.1... 192.168.0.254. Subnet mask will be fixed to 255.255.255.0, and no gateway will be set (0.0.0.0).

b7	b6	b5	b4	b3	b2	b1	b0	IP Address
0	0	0	0	0	0	0	0	IP configuration specified elsewhere
0	0	0	0	0	0	0	1	192.168.0.1
0	0	0	0	0	0	1	0	192.168.0.2
0	0	0	0	0	0	1	1	192.168.0.3
...	...	...	...	...	...	...	...	...
1	1	1	1	1	1	0	1	192.168.0.253
1	1	1	1	1	1	1	0	192.168.0.254
1	1	1	1	1	1	1	1	(not valid)

Note: The switch type is specified by parameter #9 ("Switch Coding"). The default value for the Anybus-IC PROFINET is 01h (Binary Switches). Note however that there is no guarantee that the same default value is used on other networks.

See also...

- Anybus-IC Design Guide (parameter #9 "Switch Coding")
- "Communication Settings" on page 72

Network Reset Handling

General Information

The module may receive reset commands from the network or spontaneously generate a reset due to an error. The application can be notified of network reset events through the interrupt mechanisms outlined in the Anybus-IC Design Guide.

The following parameters are involved when dealing with network reset requests:

- Parameter #12 ('Interrupt Config')
- Parameter #13 ('Interrupt Cause')

For more information about network reset procedures, consult the general Anybus-IC Design Guide.

Reset To Factory

Upon receiving a 'Reset To Factory' request from the network, the module will reset basic network settings (IP configuration, PROFINET Station Name, I&M information etc.) to their factory default values.

The behavior that follows depends on the value of the 'RES' bit in parameter #13:

- **'RES' bit set**
The application is notified of the reset request using the interrupt mechanism. The module will then wait for the application to reset it.
- **'RES' bit cleared**
The application is not notified of the reset request, and the module resets itself automatically.

Filesystem

General Information

The Anybus module features a built-in filesystem, which is used to store information such as web files, network communication settings, e-mail messages etc.

The filesystem can be accessed using FTP, HTTP, and by the application via the mailbox interface.

Storage Areas

The filesystem consists of the different storage areas:

- **Nonvolatile area (approx. 780 kb)**
This section is intended for static files such as web files, configuration files etc.
- **Volatile area (approx. 1 Mb)**
This area is intended for temporary storage; data placed here will be lost in case of power loss or reset. Note that this area is not available by default, and must be mounted by the application during initialization (see “RAM-Disc Path (Parameter #131)” on page 82)

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 (plus null termination)
- A path cannot be longer than 256 characters (filename included)
- The maximum number of simultaneously open files is 40
- The maximum number of simultaneously open directories is 40

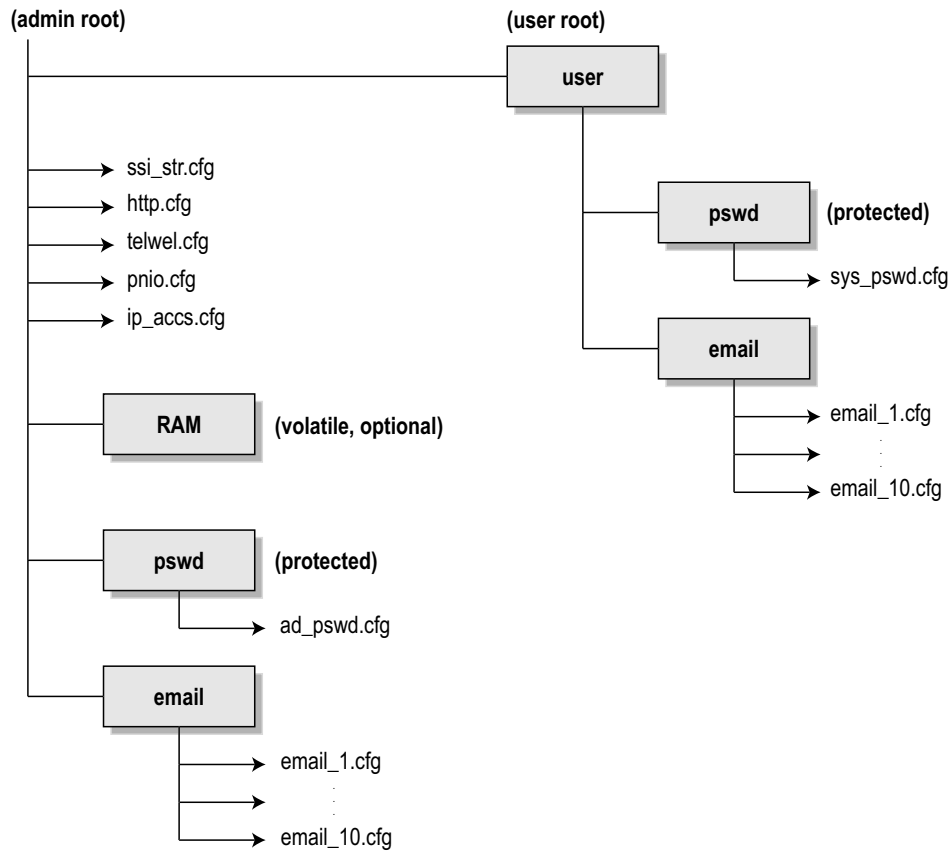
Important Notes

The nonvolatile storage is located in FLASH memory. Each FLASH segment can only be erased approximately 1000000 times due to the nature of this type of memory.

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 filesystem

Filesystem Overview



System Files

The filesystem 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™). Note that some of these files may also be altered by the Anybus module itself, e.g. when using SSI (see “Server Side Include (SSI)” on page 54).

The format of the system files are based on the concept of ‘keys’, where each ‘key’ can be assigned a value, see example below.

Example:

```

[Key1]
value of key1

[Key2]
value of key2
  
```

The exact format of each system file is described in detail later in this document.

Network Configuration

Physical Link Settings

The module uses 100 Mbit/s full duplex.

TCP/IP Settings

The module offers three modes of operation regarding the TCP/IP settings:

- **Settings specified by Switches (Fieldbus Specific Input)**

The module will use the switch setting if the NA bit (Parameter #8 'Configuration Bits') is cleared (0) and the switches are set to a value other than zero (0).

In such case, the module will use the following settings:

IP Address: 192.168.0.x (x = switch value)

Gateway: 0.0.0.0 (no gateway)

Subnet: 255.255.255.0

DHCP: OFF

See also...

- Anybus-IC Design Guide (Parameter #8 'Configuration Bits')
- "Switches (Fieldbus Specific Input)" on page 16

- **Settings specified by Parameters**

The module will use the settings stored in the IP configuration parameters if the NA bit (Parameter #8 'Configuration Bits') is set (1) and/or the switches on the fieldbus specific input register is set to zero.

If no current settings are available (i.e. the IP configuration parameter contains invalid settings), the module will halt and indicate an error on the on-board status LEDs (the settings may however still be altered via HICP, ARP gleaning or DCP).

See also...

- Anybus-IC Design Guide (Parameter #8 'Configuration Bits')
- "Anybus IPconfig (HICP)" on page 22
- "ARP Gleaning" on page 22
- "Communication Settings" on page 72

- **Settings specified by network**

The IP configuration can be altered via PROFINET via the DCP interface.

See also...

- Anybus-IC Design Guide (Parameter #8 'Configuration Bits')

IP Access Control

It is possible to specify which IP addresses that are permitted to connect to the module. This information is stored in the system file ‘\ip_accs.cfg’.

File Format:

[Web] xxx.xxx.xxx.xxx	•	Nodes listed here may access the web server
[FTP] xxx.xxx.xxx.xxx	•	Nodes listed here may access the FTP server
[Modbus/TCP] xxx.xxx.xxx.xxx	•	Nodes listed here may access the module via Modbus/TCP
[All] xxx.xxx.xxx.xxx	•	Fallback setting, used by the module when one or several of the keys above are omitted

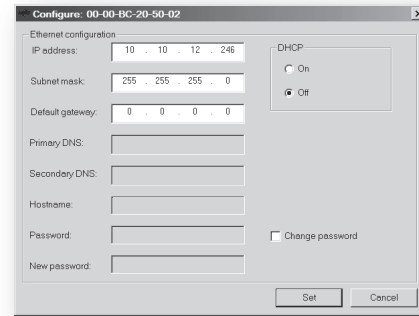
Note: ‘*’ may be used as a wildcard to select IP series. PROFINET IO traffic is not affected by the settings of this file.

Anybus IPconfig (HICP)

The module supports the HICP protocol used by the Anybus IPconfig utility from HMS, which can be downloaded free of charge from the HMS website. This utility may be used to configure the network settings of any Anybus product connected to the network. Note that if successful, this will replace the settings currently stored in the corresponding parameters.

Upon starting the program, the network is scanned for Anybus products. The network can be rescanned at any time by clicking 'Scan'. In the list of detected devices, the module will appear as 'ABIC-PRT'. To alter its network settings, double-click on its entry in the list.

A window will appear, containing the IP configuration and password settings. Validate the new settings by clicking 'Set', or click 'Cancel' to abort.



Optionally, the configuration may 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'. When protected, any changes in the configuration requires that the user supplies a valid password.

When done, click 'Set'.

ARP Gleaning

The module supports the Address Resolution Protocol (ARP), allowing the TCP/IP settings to be altered using the ARP command on a PC.

Syntax:

```
arp -s <IP address> <MAC address>
ping <IP address>
arp -d <IP address>
```

The 'arp -s' command stores the IP and MAC address in the PCs ARP table. When the 'ping' command is issued, the PC will address the module with the new IP address; the module recognizes that it was addressed with the correct MAC address and adopts the new IP address from the 'ping' message.

If successful, new settings will be stored as follows:

IP Address:	xxx.xxx.xxx.xxx(value supplied in ARP command)
Gateway:	0.0.0.0(no gateway)
Subnet:	255.255.255.0
DHCP:	OFF

Note: This functionality may cause problems if multiple devices continuously issue 'ping' messages towards the module. The reason for this lies in the very nature of this functionality; since the module adopts the IP address from all 'ping' messages, any additional 'ping' messages may cause it to change back and forth between old and new settings.

See also...

- "ARP Gleaning Enable (Parameter #144)" on page 76

PROFINET I/O Implementation

General Information

The Anybus-IC PROFINET implements Real Time (RT) PROFINET I/O communications in accordance with PROFINET Conformance Class A.

Implementation Overview:

- Up to 144 bytes of cyclic I/O in each direction
- Acyclic access to up to 100 application parameters
- 4 ms min. update time for I/O
- Supports one device instance (i.e. several IO devices cannot be accomplished)
- Supports up to two APIs (including the mandatory API 0)
- Supports up to 32 modules, with 8 submodules/module
- Supports up to 128 submodules in total (not counting port and interface submodules)
- Supports LLDP
- Supports SNMPv1 and SNMPv2 (MIB2 not supported)
- Supports PROFINET IO Conformance Class A (CCA)

The fieldbus input- and output data is exchanged as cyclic I/O data on PROFINET. Acyclic communication is possible using application parameters, which can be accessed from the network by means of the record data read/write services.

This procedure involves the following steps:

1. Define an application parameter using the application parameter object.
2. Specify its representation on PROFINET using the PROFINET IO mapping object.

The Anybus module supports up to 100 application parameters.

See also...

- “Application Parameters” on page 98
- “Application Parameter Object (Class 85h)” on page 106
- “PROFINET Mapping Object (Class A9h)” on page 143

PROFINET Compliance

Products which claim PROFINET compliance must pass conformance tests at a certified test facility. HMS embedded products for PROFINET are successfully tested for precompliance and found to comply with PROFINET specifications. Test reports etc. are available and registered at the PROFIBUS Organisation (PNO).

The use of certified PROFINET networking technology (e.g. Anybus) does not automatically make the end product certified. To be able to state PROFINET compliance for the end product, the final implementation will need to pass further certification tests at a certified test facility. Due to the vast customisation possibilities found in the Anybus concept, the use of such technology alone does in itself not constitute a discount of the certification fee.

Generic implementations¹ generally only require basic knowledge in the PROFINET networking system and are certifiable by nature as long as basic usage procedures are followed (exceptions and instructions are stated where appropriate).

With advanced implementations², the topic of network certification becomes slightly more complex. A side effect of the inherent flexibility of the Anybus concept is that the application, in theory, can alter the behavior on PROFINET in ways which makes it impossible to successfully pass network certification tests. HMS cannot possibly foresee all possible usage scenarios for this type of implementation, which mean that those responsible for the implementation of the Anybus module into the final product should take the necessary steps to ensure that the implementation operates according to PROFINET specifications. This process may involve steps which require in-depth knowledge in PROFINET networking internals and/or information from the official PROFINET specifications. Those responsible for the implementation of the final product should either obtain the PROFINET specification to gain sufficient knowledge or limit their implementation in such a way that this is not necessary.

Due to the rapid development in the industrial communication industry, and the impact this may have on network certification procedures, always make sure to use the very latest Anybus revision when certifying the final product.

In case of uncertainties, contact your nearest HMS support department. It is encouraged to always contact HMS before going to conformance testing.

The following points must be considered when doing a PROFINET IO in-design (this list is not necessarily complete):

- IP-address information and 'Station name' shall, according to the PROFINET IO specification, be set to their default values when the Anybus module is shipped from the factory, or when a 'Reset to Factory' command has been carried out. This means that if these parameters are modified from the application side, it must be on the initiative of the end user.
- During PROFINET IO conformance testing DHCP cannot be enabled.
- The application must either support the 'RES' bit functionality, or detect that the Anybus module is spontaneously restarted and reinitialize it. The reason for this is that the Anybus module must be restarted after reception of a 'Reset to factory' command.

For more information, see "Conformance Test Guide" on page 146.

Contact information:

- <http://www.profinet.com>

-
1. Includes implementations which run in normal Anybus mode and does not require changes to the generic GSD file supplied by HMS.
 2. Includes implementations which run in advanced mode and/or require deviations from the generic GSD file supplied by HMS.

Electronic Data Sheet (GSD file)

On PROFINET, the characteristics of a device is stored in an XML-format data file with the suffix GSD. This file is used by the PROFINET configuration tool when setting up the network.

HMS provides a generic GSD file, which corresponds to the default settings in the module (i.e. when operating in normal mode with the default identity settings).

The standard GSD file has the following properties:

- Complies to v2.00 of the GSDML specification for PROFINET IO.
- Features two Device Access Points (DAP):
 - 'RT', used for RT-communication ('DAP v1.0', located in the migration folder in the commissioning tool). This DAP is supplied for reverse compatibility with IO controllers not supporting extended PROFINET IO port diagnostics.
 - 'RT', used for RT-communication ('DAP v2.0').
- Available RT (DAP v1.0) modules and their block size: 1, 2, 4, 8, 16, 32, 64, 128 bytes.
- Available RT (DAP v2.0) modules and their block size: 1, 2, 4, 8, 16, 32, 64, 128 bytes.
- Modules have no assigned 'Initial Parameters'.
 - See "Initial Parameters" on page 31.
- Modules are consistent over the entire block size.
- All modules are available as input, output and bidirectional (input/output).
- Each module is associated with exactly one submodule. Multiple submodules per module requires a custom GSD file.

Network Conformance Notes:

- Any deviations from the standard GSDML file voids HMS precompliance tests.

See also...

- "Modes of Operation" on page 27

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, "ABICPRTPIC1").

Device Identity

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

Station Name^{ab}	(not assigned)
Station Type^b	'ABIC-PRT'
Vendor ID^b	010Ch (HMS Industrial Networks)
Device ID^b	0008h (Anybus-IC PROFINET)

a. Can be specified via PROFINET by means of the Discovery and basic Configuration Protocol (DCP).

b. Can be customized by the application.

Network conformance notes:

- Identity customization voids HMS network precompliance tests and requires a new customized GSDML file.

See also...

- “PROFINET Compliance” on page 24

Modes of Operation

The Anybus module offers two distinct modes of operation with regards to the PROFINET communication.

- **Normal Anybus Mode (Default)**

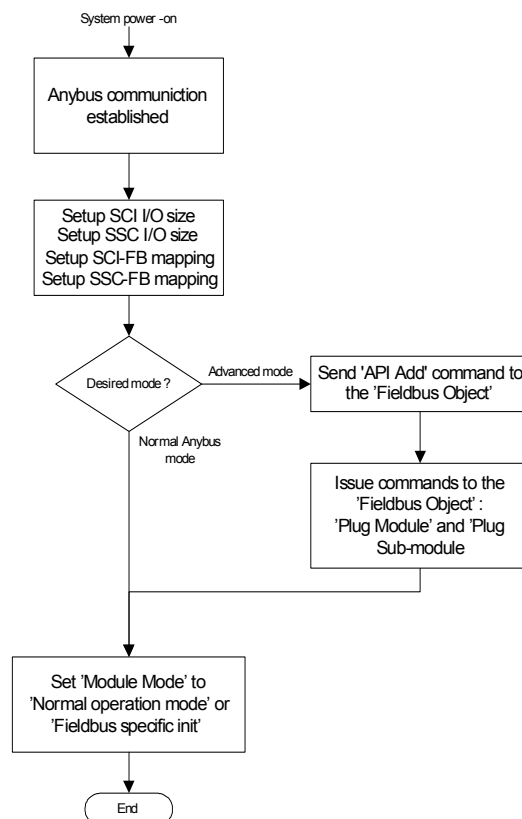
In this mode, the Anybus module creates a default configuration based on the actual I/O configuration.

- **Advanced Mode**

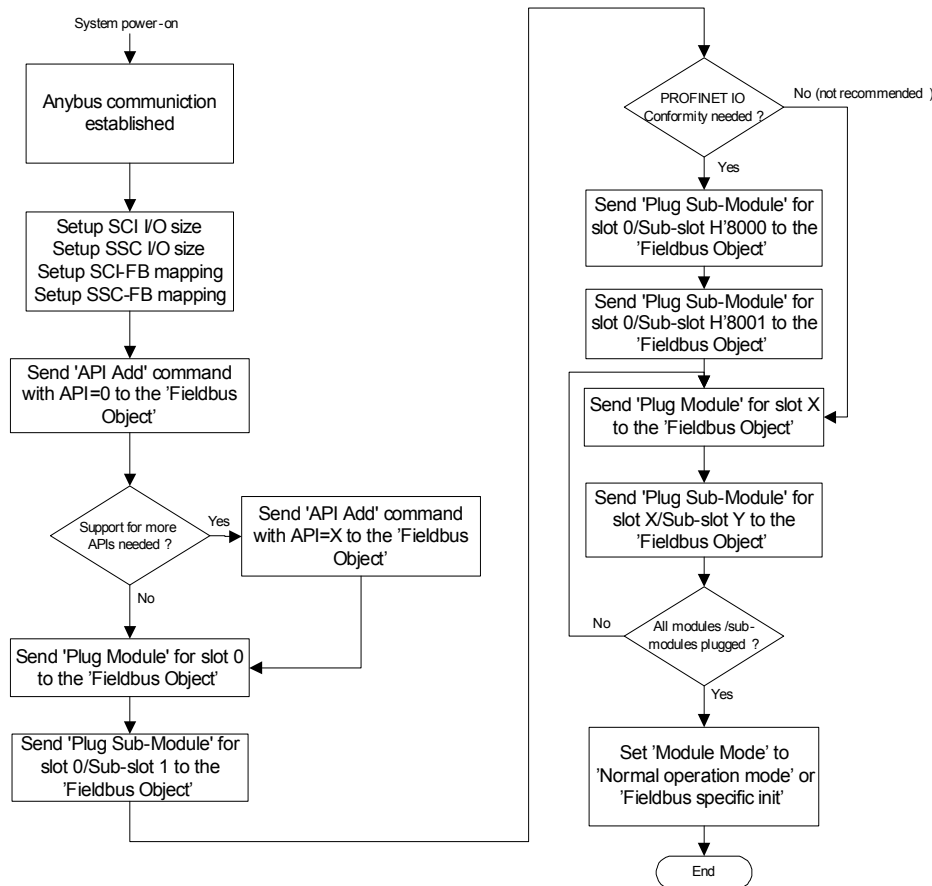
In this mode, the application creates the PROFINET IO configuration by using the network specific commands of the fieldbus object.

Mode selection

As mentioned above the distinction between 'Normal Anybus Mode' and 'Advanced Mode' is the usage of the specific services 'API Add', 'Plug Module' and 'Plug Sub-module' of the fieldbus object (see "Fieldbus Object (Class A0h)" on page 113). The figure below describes how to select the two modes.

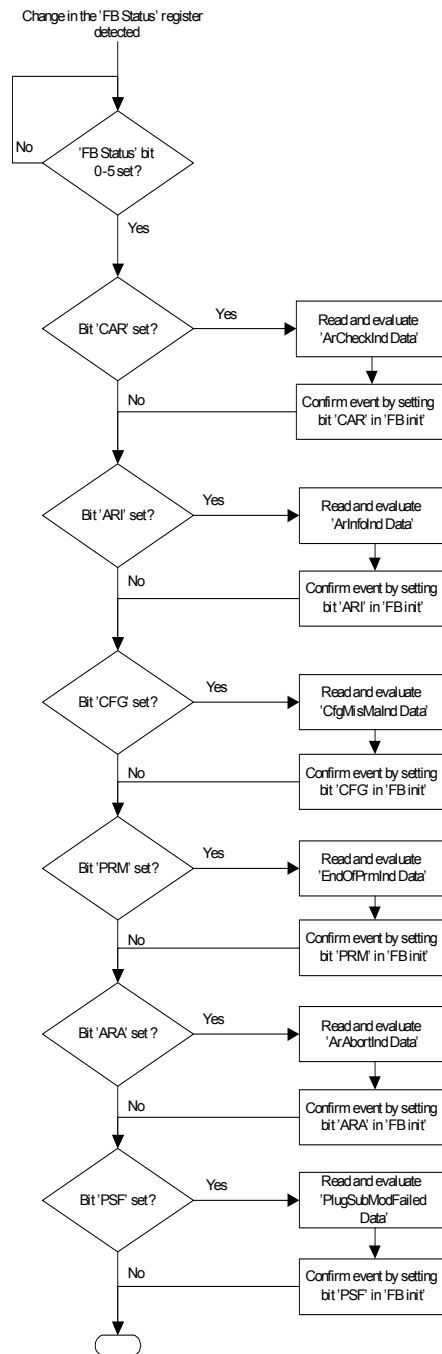


When choosing 'Advanced Mode', the following figure gives a brief overview of commands that need to be issued.



Module Initialized with 'Module Mode' set to 'Fieldbus specific init'

The host application has the opportunity to use the fieldbus specific init mode (in both 'Normal Anybus mode' and 'Advanced Mode'). In fieldbus specific init mode, the host application has the possibility to examine some network events before the Anybus module confirms them to the IO controller. The events are signaled in the 'FB Status' parameter (see "FB Status (Parameter #100)" on page 68) and are confirmed with the 'FB init' parameter (see "FB init (Parameter #101)" on page 69). The picture below describes what the application needs to do in order to proceed.



See also...

- “Establishing a PROFINET IO connection” on page 35
- “ArCheckInd Data (Parameter #167)” on page 87
- “CfgMisMaIndData (Parameter #168)” on page 89
- “ArInfoInd Data (Parameter #169)” on page 90
- “EndOfPrmInd Data (Parameter #170)” on page 91
- “ArAbortInd Data (Parameter #172)” on page 93
- “PlugSubModFailed (Parameter #173)” on page 94

Diagnostics & Alarms

In the Anybus implementation, diagnostic entries are always created in conjunction with diagnostic alarms. The following alarm types are supported:

- **Diagnostic Alarm**

There are two types of diagnostic alarms; channel- and generic diagnostic alarms. Generally, it is recommended to use channel diagnostic alarms, since the semantics are fully defined by the PROFINET specification. Alternatively, vendor specific diagnostics can be accomplished by means of generic diagnostic alarms.

Each time an event is reported as a diagnostic alarm, a corresponding diagnostic entry is created and stored by the Anybus module. The IO controller/supervisor may read the diagnostic entry using record data requests as follows:

Index	Contents
800Ah... 800Bh	Channel diagnostic data for a specific submodule
800Ch	Generic diagnostic data for a specific submodule
C00Ah... C00Bh	Channel diagnostic data for a specific module
C00Ch	Generic diagnostic data for a specific module
E00Ah... E00Bh	Channel diagnostic data for an AR
E00Ch	Generic diagnostic data for an AR
F00Ah... F00Bh	Channel diagnostic data for an API
F00Ch	Generic diagnostic data for an API

When an event has been resolved, an additional diagnostic alarm shall be issued to inform the IO controller that the event has been handled. As a result, the corresponding diagnostic entry is removed. Note that diagnostic data only can be obtained from configured modules/submodules. For more information about how to interpret the data, consult the PROFINET specification.

- **Process Alarm**

A process alarm signals the occurrence of an event related to the process, e.g. temperature exceeding limits, short circuit etc.

- **Pull/Plug Alarm**

Issued each time the application pulls/plugs modules and submodules during runtime.

See also...

- “PROFINET Diagnostic Object (Class A8h)” on page 134

Initial Parameters

During network startup, the module may optionally be loaded with initial parameters which are defined in the GSD file. This is carried out using record data requests, which means that the data will be written to the application parameters which are mapped to acyclic PROFINET IO services. To make a parameter an initial parameter, set its “Initial Record Data” attribute to true.

After the IO controller signals that the parameterization phase is completed the Anybus module will check that all entries in the ‘PROFINET Mapping Object’ set as ‘Initial Record Data’ has been correctly accessed. If not, a parameter error will be signaled.

See also...

- “Application Parameter Object (Class 85h)” on page 106
- “PROFINET Mapping Object (Class A9h)” on page 143

Configuration Data

During network startup, the IO Controller sends its expected configuration to the Anybus module for validation. The Anybus module handles this slightly differently depending on how it has been initialized:

- **Normal Anybus Mode**

The Anybus module will adopt to the expected configuration sent by the IO controller. A configuration error is triggered by the following:

- The IO controller configuration contains more output data than what is setup by the Anybus module.
- The IO controller configuration contains more input data than what is setup by the Anybus module.
- The IO controller configuration contains unknown modules, or submodules.

- **Advanced Mode (the application has created the configuration)**

In the event of a mismatch, the Anybus module will signal configuration error. In addition to this it is possible to configure the behavior slightly:

- Anybus initialized with ‘Module Mode’ = 1 (Normal Operation Mode)
In the event of a mismatch, the Anybus will reject the IO controller configuration directly, and signal configuration error.
- Anybus initialized with ‘Module Mode’ = 2 (Fieldbus Specific Init)
In the event of a mismatch, the Anybus sets the ‘CFG’ bit of the ‘FB status’ register. The application can then examine the mismatch. When the application has examined the mismatch the Anybus module will reject the configuration, and signal configuration error.

See also...

- “Mode selection” on page 27

Identification & Maintenance (I&M)

Identification & Maintenance (from now on referred to as I&M) provides a standardized way of gathering information about an IO device.

The I&M data is accessed using record data requests as follows:

Index	Contents	Comments
AFF0h	IM0	read-only
AFF1h	IM1	read/write
AFF2h	IM2	read/write
AFF3h	IM3	read/write
AFF4h	IM4	read/write

API / Slot / Subslot are 'don't care', the same I&M data is used for all combinations. IM1-4 can be disabled by the application by using the 'IM supported' attribute of the 'I&M Object'.

Structure of the Block Header

When the IO controller / IO supervisor requests to read an I&M record the response will contain two parts; a 'Block Header' and the I&M data itself. The structure of the 'Block Header' is described below.

Byte	Field	Description
0-1	'BlockType'	Describes what data is following: 0x0020 - I&M0 follows 0x0021 - I&M1 follows 0x0022 - I&M2 follows 0x0023 - I&M3 follows 0x0024 - I&M4 follows
2-3	'BlockLength'	This field contains the number of bytes of the following block and the 'Block Header', not counting the 'BlockType' and 'BlockLength'
4	'BlockVersionHigh'	Shall be set to 0x01
5	'BlockVersionLow'	Shall be set to 0x00

For more information about the 'Block Header' refer to the PROFINET IO specification.

Structure of I&M Read response

A request to read an I&M record will generate a response as described below.

Byte	Field	Description
0-5	'Block Header'	Header describing the data following after the header and the length of it. See "Structure of the Block Header" on page 32
6-	I&Mx data	The I&M data record

Structure of I&M Write request

A request to write an I&M record shall contain the data as specified below.

Byte	Field	Description
0-5	'Block Header'	Header describing the data following after the header and the length of it. See "Structure of the Block Header" on page 32
6-	I&Mx data	The I&M data record

Structure of I&M0 data

The structure of the I&M0 data is composed as follows.

Byte	Field	Description
0-1	'Manufacturer Id'	See "I&M Object (Class A7h)" on page 131
2-21	'Order Id'	
22-37	'Serial number'	
38-39	'Hardware revision'	
40-43	'Software revision'	
44-45	'Revision counter'	
46-47	'Profile Id'	
48-49	'Profile specific type'	
50-51	'IM version'	
52-53	'IM supported'	

Structure of I&M1 data

The structure of the I&M1 data is composed as follows.

Byte	Field	Description
0-31	'Tag Function'	See "I&M Object (Class A7h)" on page 131
32-53	'Tag Location'	

Structure of I&M2 data

The structure of the I&M2 data is composed as follows.

Byte	Field	Description
0-15	'Installation date'	See "I&M Object (Class A7h)" on page 131

Structure of I&M3 data

The structure of the I&M3 data is composed as follows.

Byte	Field	Description
0-53	'Descriptor'	See "I&M Object (Class A7h)" on page 131

Structure of I&M4 data

The structure of the I&M4 data is composed as follows.

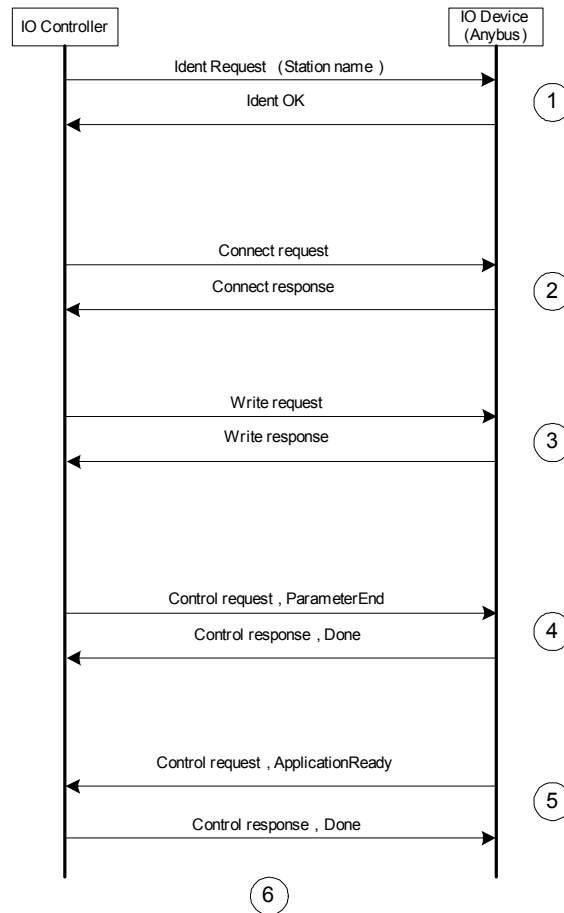
Byte	Field	Description
0-53	'Signature'	See "I&M Object (Class A7h)" on page 131

See also...

- "I&M Object (Class A7h)" on page 131

Establishing a PROFINET IO connection

When a PROFINET IO connection is established the following steps are carried out on the network and in the Anybus module. The drawing below is simplified, but should give a rough idea of what the flow looks like.



1. The IO controller sends out an 'Identification' request asking the node with the configured station name to respond. The Anybus module responds automatically to this request.
2. When the IO controller has identified the correct device it sends a 'Connect request' to the slave. The Anybus module uses this information to generate information to the application host (if initialized with 'Module Mode' set to 'Fieldbus specific init'): information about the application relationship and possible configuration mismatch indication(s). The following events are / can be triggered with the 'FB status' parameter: 'CAR', 'ARI' and 'CFG' ('CFG' can be signaled more than once). Once all events are confirmed by the host application the 'Connect response' is sent.
3. After reception of the 'Connect response', the IO controller will send 'Initial Parameters' by means of one, or several, 'Write requests', if this is configured with the GSD file (with the default GSD file this is not configured).
4. After completion of the 'Initial Parameters' the IO controller will send a 'ParameterEnd' request. The Anybus module will either respond directly to this request, or, if configured with 'Module Mode'

set to 'Fieldbus specific init', trigger an 'ARA' event in the 'FB status' parameter. Once the host application has confirmed the event the response is sent.

5. The Anybus module will signal to the IO controller that the Anybus module is ready for 'Data Exchange' by sending the 'ApplicationReady' telegram.
6. Normal 'Data Exchange' is started (in fact, 'Data Exchange' has been running all the time, but the status bytes related to the IO data has been set to 'BAD', meaning that there is no valid IO data).

See also...

- "Modes of Operation" on page 27
- "FB Status (Parameter #100)" on page 68
- "FB init (Parameter #101)" on page 69

Fast Start Up

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-IC 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: 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:

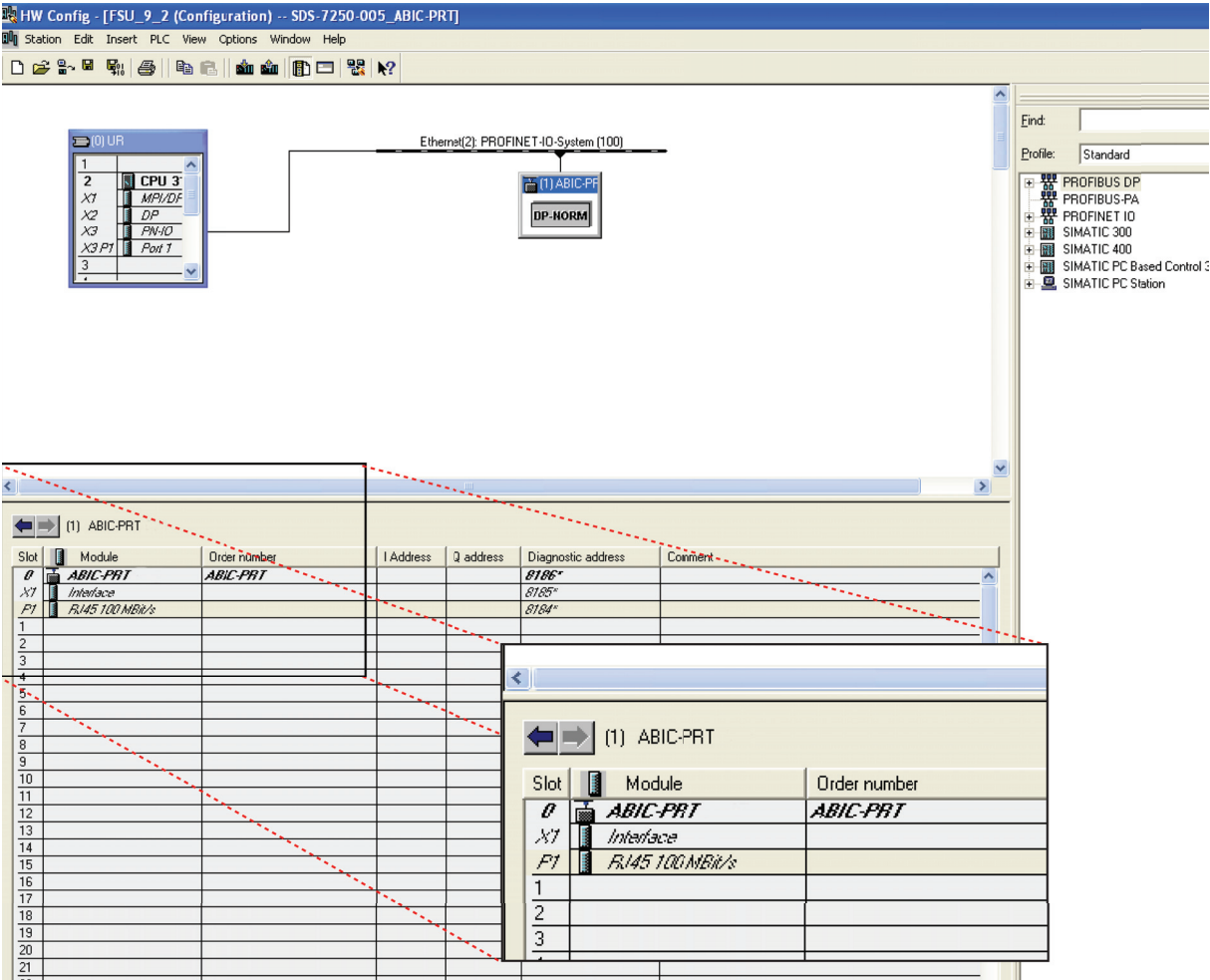
- **PowerOnToCommReady**
Remove this keyword from the GSD file.
- **DCP_HelloSupported**
Value: false.

Fast Start Up Configuration with STEP7

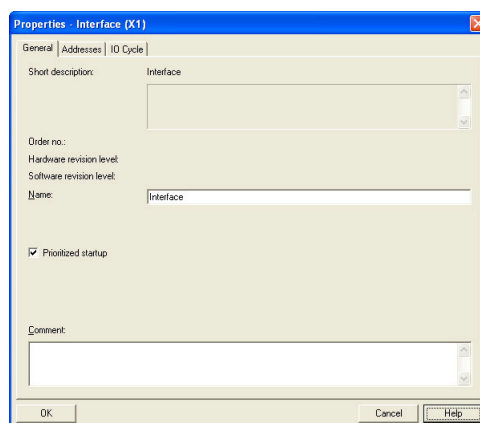
The example below shows the procedure when the Siemens tool STEP7 is used for configuration.

Activation of Fast Start Up

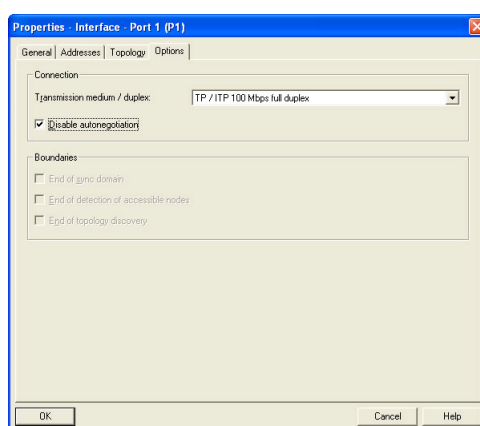
1. Start the configuration tool. The figure below shows the HW Config window of the STEP7 tool. The enlarged part from the Module column is used when activating Fast Start Up.



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 “P1” in the Module column. The window shown to the right will appear. Choose the Options tag. To configure fastest possible up-start, choose transmission rate “100 Mbps, full duplex” and check the “Disable autonegotiation” box.



Modbus-TCP Implementation

General Information

The built-in Modbus-TCP server provides access to the fieldbus I/O via a subset of the functions defined in the Modbus-TCP specification. For detailed information regarding the Modbus-TCP protocol, consult the Open Modbus Specification.

Implementation Overview:

- Acyclic access to the Fieldbus I/O buffers (up to 144 bytes in each direction)
- Coil and Register access
- Up to 8 concurrent Modbus-TCP connections
- Communicates over TCP port 502

Note: Application Parameters cannot be accessed via Modbus-TCP.

See also...

- “PROFINET I/O Implementation” on page 23

Supported Function Codes

The following function codes are implemented:

Modbus Function	Function Code	Associated with...
Read Coil	1	Output Data (FB Out)
Read Input Discretes	2	Input Data (FB In)
Read Multiple Registers	3	Output Data (FB Out)
Read Input Registers	4	Input Data (FB In)
Write Coil	5	Output Data (FB Out)
Write Single Register	6	
Force Multiple Coils	15	
Force Multiple Registers	16	
Mask Write Register	22	
Read/Write Registers	23	

Exception Codes

Code	Name	Description
01h	Illegal function	The function code in the query is not supported
02h	Illegal data address	The data address received in the query is outside the initialized memory area
03h	Illegal data value	The data in the request is illegal

Register Map

Note: In this document, Modbus-TCP registers are specified using the PLC convention (base 1), i.e. register 0001 equals 0000 (zero) in the actual message frame.

Fieldbus Input Data (FB In)

The fieldbus input data is mapped to coils & registers as follows:

Register #	Input Data (FB In) Location	Comments
0001 (0513)	000h + Coil Size In	Each register corresponds to two bytes of data.
0002 (0514)	002h + Coil Size In	
0003 (0515)	004h + Coil Size In	
...	...	
0071 (0583)	08Ch + Coil Size In	
0072 (0584)	08Eh + Coil Size In	

Note: A mirror of the input data is available at register 513... 584 to allow efficient I/O scanning by means of Read/Write registers (function code 13)

Coil #	Input Data (FB In) Location	Comments
0001... 0008	000h	Each coil corresponds a single bit of data.
0009... 0016	001h	
0017... 0024	002h	
...	...	
1137... 1144	08Eh	Note: Coils are mapped MSB first, i.e. coil #1 corresponds to the most significant bit of byte 000h
1145... 1152	08Fh	

Fieldbus Output Data (FB Out)

The fieldbus output data is mapped to coils & registers as follows:

Register #	Output Data (FB Out) Location	Comments
0001	000h + Coil Size Out	Each register corresponds to two bytes of data.
0002	002h + Coil Size Out	
0003	004h + Coil Size Out	
...	...	
0071	08Ch + Coil Size In	
0072	08Eh + Coil Size In	

Coil #	Output Data (FB Out) Location	Comments
0001... 0008	000h	Each coil corresponds a single bit of data
0009... 0016	001h	
0017... 0024	002h	
0025... 0032	003h	
...	...	Note: Coils are mapped MSB first, i.e. coil #1 corresponds to the most significant bit of byte 000h
1137... 1144	08Eh	
1145... 1152	08Fh	

E-mail Client

General Information

The built-in e-mail client can send predefined e-mail messages when instructed to do so or based on trigger-events in the fieldbus I/O data.

The client supports SSI; note however that certain SSI functions cannot be used in e-mail messages (this is specified separately for each SSI function). SSI functions can be used to alter recipient, sender, subject line, headers, and message body.

Note: This functionality requires a valid SMTP server configuration. It is currently not possible to use servers which require authentication.

See also...

- “E-mail Definitions” on page 43
- “Server Side Include (SSI)” on page 54
- “SMTP Srv Address (Parameter #126)” on page 79
- “SMTP Errors (Parameter #128)” on page 79
- “Send Email (Parameter #129)” on page 80

Event-Triggered Messages

The e-mail client can issue predefined messages based on events in the fieldbus I/O data. The Anybus module scans the fieldbus I/O buffers for events every 0.5 seconds, which means that an event must be present for at least 0.5 seconds to ensure detection.

In operation, this works as follows:

1. The trigger source is fetched from the fieldbus I/O data
2. A logical AND is performed between the trigger source and a mask value
3. The result is compared to a reference value according to a specified operand
4. If the end result is true, the e-mail is sent to the specified recipient(s)

See also...

- “E-mail Definitions” on page 43
- “Triggered Emails (Parameter #127)” on page 79

E-mail Definitions

The e-mail definitions are stored in the following two directories:

- **'\user\email'**
This directory holds up to 10 messages which can be altered by normal-level FTP users.
- **'\email'**
This directory holds up to 10 messages which can be altered by admin-level FTP users.

E-mail definition files must be named 'email_1.cfg', 'email_2.cfg'... 'email_10.cfg' in order to be properly recognized by the module.

File Format:

```
[Register]
Area, Offset, Type

[Register Match]
Value, Mask, Operand

[To]
recipient

[From]
sender

[Subject]
subject line

[Headers]
Optional extra headers

[Message]
message body
```

Field	Value	Scanned for SSI
Area	Source buffer. Possible values are 'IN' or 'OUT'	No
Offset	Source offset, written in decimal or hexadecimal	
Type	Source data type. Possible values are 'byte', 'word', and 'long'	
Value	Used as a reference value for comparison	
Mask	Mask value, applied on the trigger source prior to comparison (logical AND)	
Operand	Possible values are '<', '=', or '>'	
To	E-mail recipient	Yes
From	Sender e-mail address	
Subject	E-mail subject. One line only	
Headers	Optional; may be used to provide additional headers	
Message	The actual message	

Note: Hexadecimal values must be written with the prefix '0x' in order to be recognized by the module.

FTP Server

General Information

The built-in FTP server provides a way to access the file system using a standard FTP client.

The following port numbers are used for FTP communication:

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

See also...

- “FTP Srv Enable (Parameter #122)” on page 77

Security Levels

The FTP server features two security levels; admin and normal.

- **Normal-level users**
The root directory will be ‘\user’.
- **Admin-level users**
The root directory will be ‘\’, i.e. the user has unrestricted access to the file system.

User Accounts

The user accounts are stored in two files, which are protected from web access:

- ‘\user\pswd\sys_pswd.cfg’
This file holds the user accounts for normal-level users.
- ‘\pswd\ad_pswd.cfg’
This file holds the user accounts for admin-level users.

File Format:

The format of these files are as follows:

```
Username1:Password1
Username2:Password2
Username3:Password3
```

Note 1: If no valid user accounts have been defined, or if running in global admin mode, the module will grant admin-level access to all users. In such cases, the FTP accepts any username/password combination, and the root directory will be ‘\’.

Note 2: The FTP server shares user accounts with the Telnet server.

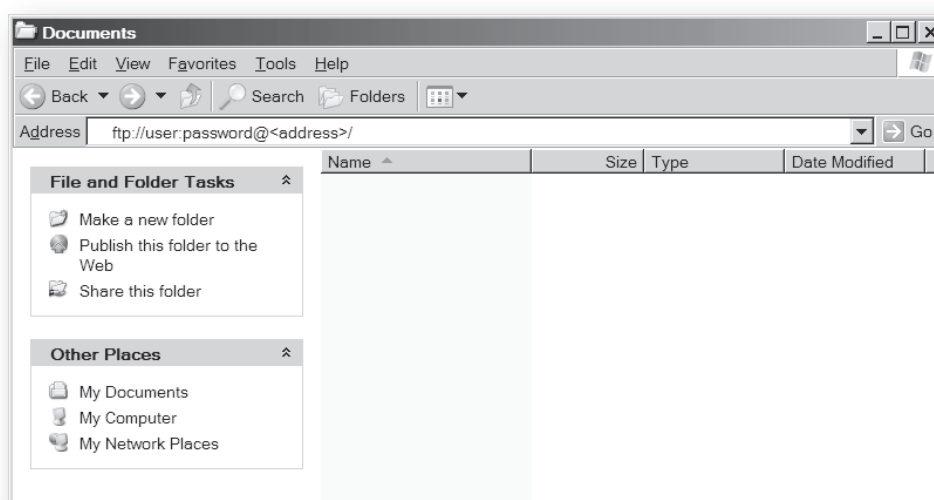
See also...

- “Telnet Server” on page 46
- “Admin Mode Cfg (Parameter #124)” on page 81
- “Admin Mode Act (Parameter #125)” on page 81

FTP Connection Example (Windows Explorer)

The built-in FTP client in Windows Explorer 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 'Explore'.
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 module using the specified settings. If successful, the built-in file system is displayed in the Explorer window.



Telnet Server

General Information

The built-in Telnet server provides a way to access the file system using a standard Telnet client. The server communicates through TCP port 23.

See also...

- “Telnet Srv Enable (Parameter #123)” on page 78

Security Levels

Just like the FTP server, the Telnet server features two security levels; admin and normal.

- **Normal-level users**
The root directory will be ‘\user’.
- **Admin-level users**
The root directory will be ‘\’, i.e. the user has unrestricted access to the file system.

User Accounts

The Telnet server shares user accounts with the FTP server. If no valid user accounts have been defined, or if running in global admin mode startup, the module will grant admin-level access to all users. In such case, no login is required, and the root directory will be ‘\’.

See also...

- “User Accounts” on page 44
- “Admin Mode Cfg (Parameter #124)” on page 81
- “Admin Mode Act (Parameter #125)” on page 81

Welcome Message

The default Telnet welcome message can be customized by adding the file ‘\telwel.cfg’. This file shall contain the desired message in pure ASCII format.

General Commands

admin

- **Syntax**

admin

- **Description**

Provided that the user can supply a valid admin username/password combination, this command provides admin access rights to normal-level users.

exit

- **Syntax**

exit

- **Description**

This command closes the Telnet session.

help

- **Syntax**

help [general|diagnostic|filesystem]

- **Description**

If no argument is specified, the following menu will be displayed.

General commands:

help	- Help with menus
version	- Display version information
exit	- Exit station program

Also try 'help [general|diagnostic|filesystem]'

version

- **Syntax**

version

- **Description**

This command will display version information, serial number and MAC ID of the module.

Diagnostic Commands

arps

- **Syntax**
arps
- **Description**
Display ARP stats and table

iface

- **Syntax**
iface
- **Description**
Display net interface stats

routes

- **Syntax**
routes
- **Description**
Display IP route table

sockets

- **Syntax**
sockets
- **Description**
Display socket list

File System Operations

For commands where filenames, directory names or paths shall be given as an argument the names can be written directly or within quotes. For names including spaces the filenames must be surrounded by quotes. It is also possible to use relative pathnames using '.', '\', and '..'.

append

- **Syntax**
append [file] ["The line to append"]
- **Description**
Appends a line to a file.

cd

- **Syntax**
`cd [path]`
- **Description**
Changes current directory.

copy

- **Syntax**
`copy [source] [destination]`
- **Description**
This command creates a copy of the source file at a specified location.

del

- **Syntax**
`del [file]`
- **Description**
Deletes a file.

dir

- **Syntax**
`dir [path]`
- **Description**
Lists the contents of a directory. If no path is given, the contents of the current directory is listed.

df

- **Syntax**
`df`
- **Description**
Displays filesystem info.

format

- **Syntax**
`format`
- **Description**
Formats the filesystem. This command may only be executed by admin level users.

md

- **Syntax**
`md [directory]`
- **Description**
Creates a directory. If no path is given, the directory is created in the current directory.

mkfile

- **Syntax**
`mkfile [filename]`
- **Description**
Creates an empty file.

move

- **Syntax**
`move [source] [destination]`
- **Description**
This command moves a file or directory from the source location to a specified destination.

rd

- **Syntax**
`rd [directory]`
- **Description**
Removes a directory. The directory can only be removed if it is empty.

ren

- **Syntax**
`ren [old name] [new name]`
- **Description**
Renames a file or directory.

type

- **Syntax**
`type [filename]`
- **Description**
Types the contents of a file.

Web Server

General Information

The Anybus module features a flexible web server with SSI capabilities. The built-in web pages can be customized to fit a particular application and allow access to I/O data and configuration settings.

The web server communicates through port 80.

See also...

- “Web Srv Enable (Parameter #121)” on page 77

Protected Files

For security reasons, the following files are protected from web access:

- Files located in ‘\user\pswd’
- Files located in ‘\pswd’
- Files located in a directory which contains a file named ‘web_accs.cfg’

Default Web Pages

The Anybus module contains a set of virtual files that can be used when building a web page for configuration of network parameters. These virtual files can be overwritten (not erased) by placing files with the same name in the root of disc 0.

This makes it possible to, for example, replace the HMS logo by uploading a new logo named ‘\logo.jpg’. It is also possible to make links from a web page to the virtual configuration page. In that case the link shall point to ‘\config.htm’.

These virtual files are:

\index.shtm	- includes the content of config.htm
\config.htm	- Configuration frame page
\configform.shtm	- Configuration form page
\store.shtm	- Configuration store page
\logo.jpg	- HMS logo
\configuration.gif	- Configuration picture
\border_bg.gif	- Picture forming a border
\border_m_bg.gif	- Picture forming a border

Note: The virtual file system can be disabled using parameter #130.

See also...

- “VFS Enable (Parameter #130)” on page 82

Authentication

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.

Example:

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

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

[AuthName]
Yeah. Whatsda passwoid?
```

Note that when using this feature, make sure to put the user/password files in a directory that is protected from web access, see "Protected Files" on page 51 .

Content Types

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

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

It is possible to configure/reconfigure the reported content types, and what files to scan for SSI. This is done in the system file ‘\http.cfg’.

File Format:

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

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

Note: Up to 50 content types and 50 SSI file types may be specified in this file.

Server Side Include (SSI)

General Information

Server Side Include functionality (from now on referred to as SSI) allows data in files and objects to be represented on web pages and in e-mail messages.

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

Syntax

The 'X's below represents a command opcode and parameters associated with the command.

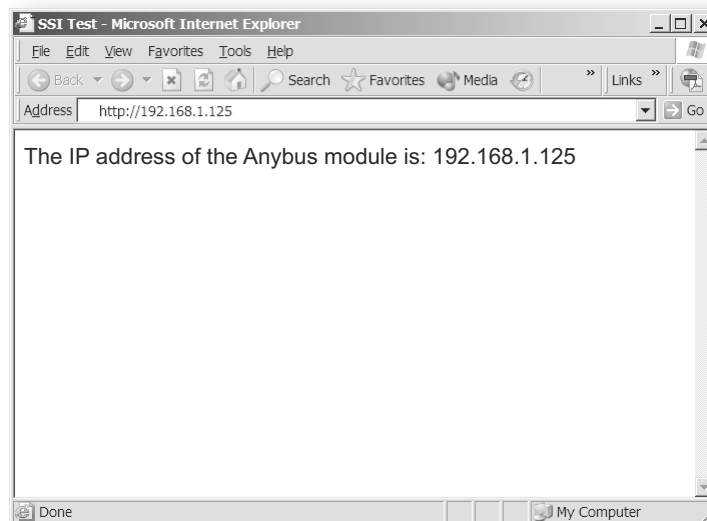
```
<?--#exec cmd_argument='XXXXXXXXXXXXXXXXXXXXXXX'-->
```

Example

The following example causes a web page to display the IP address of the module:

```
<HTML>
<HEAD><TITLE>SSI Test</TITLE></HEAD>
<BODY>
The IP address of the Anybus module is:
<?--#exec cmd_argument='DisplayIP'-->
</BODY>
</HTML>
```

Resulting webpage:



Functions

DisplayIP

This function returns the currently used IP address.

Syntax:

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

DisplaySubnet

This function returns the currently used Subnet mask.

Syntax:

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

DisplayGateway

This function returns the currently used Gateway address.

Syntax:

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

DisplayDhcpState

This function returns whether DHCP/BootP is enabled or disabled.

Syntax:

```
<?--#exec cmd_argument='DisplayDhcpState( "Output when ON", "Output when OFF" )'-->
```

DisplayEmailServer

This function returns the currently used SMTP server address.

Syntax:

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

StoreEtnConfig

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

This function stores a passed IP configuration in the corresponding parameters.

Syntax:

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

Include this line in a HTML page and pass a form with new IP settings to it.

Accepted fields in form:

```
SetIp
SetSubnet
SetGateway
SetSMTPServer
SetDhcpState      - (value "on" or "off")
```

Default output:

```
Invalid IP address!
Invalid Subnet mask!
Invalid Gateway address!
Invalid IP address or Subnet mask!
Invalid Email Server address!
Invalid DHCP state!
Configuration stored correctly.
Failed to store configuration.
```

GetText

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

This function retrieves the text from an object and stores it in a specified parameter or in the fieldbus out area.

Syntax:

```
<?--#exec cmd_arbgment='GetText( "ObjName", argument, N) '-->
```

ObjName- Name of object.

argument- (see below)

n - Number of characters to process (Optional)

Argument	Description
OutWriteString(<i>offset</i>)	Writes the data to position <i>offset</i> in the fieldbus out area
ParWriteString(<i>prm</i>)	Writes the data to parameter <i>prm</i> . Requires parameter to be a string.
ParWriteFormatted(<i>prm</i>)	Writes the data to parameter <i>prm</i> . Parses and converts the data according to the parameter data type (i.e. integer, string etc.).

Default output:

```
Success          - Write succeeded
Failure          - Write failed
```

printf

This function returns a formatted string, which may contain parameter value and/or data from the field-bus in/out area. The formatting of the string is similar to the standard C function printf().

Syntax:

```
<?--#exec cmd_argument='printf("String to write", Arg1, Arg2, ..., ArgN) '-->
```

Like the standard C function printf() the "String to write" for this SSI function contains two types of objects: Ordinary characters, which are copied to the output stream, and conversion specifications, each of which causes conversion and printing of the next successive argument to printf. Each conversion specification begins with the character % and ends with a conversion character. Between the % and the conversion character there may be, in order:

- Flags (in any order), which modify the specification:
 - which specifies left adjustment of the converted argument in its field.
 - + which specifies that the number will always be printed with a sign
 - (space) if the first character is not a sign, a space will be prefixed.
 - 0 for numeric conversions, specifies padding to the field with leading zeroes.
 - # which specifies an alternate output form. For o, the first digit will be zero. For x or X, 0x or 0X will be prefixed to a non-zero result. For e, E, f, g and G, the output will always have a decimal point; for g and G, trailing zeros will not be removed.
- A number specifying a minimum field width. The converted argument will be printed in a field at least this wide, and wider if necessary. If the converted argument has fewer characters than the field width it will be padded on the left (or right, if left adjustment has been requested) to make up the field width. The padding character is normally space, but can be 0 if the zero padding flag is present.
- A period, which separates the field width from the precision.
- A number, the precision, that specifies the maximum number of characters to be printed from a string, or the number of digits to be printed after the decimal point for e, E, or F conversions, or the number of significant digits for g or G conversion, or the minimum number of digits to be printed for an integer (leading 0s will be added to make up the necessary width)
- A length modifier h, l (letter ell), or L. "h" indicates that the corresponding argument is to be printed as a short or unsigned short; "l" indicates that the argument is a 'long' or 'unsigned long'.

The conversion characters and their meanings are shown below. If the character after the % is not a conversion character, the behavior is undefined.

Character(s)	Argument type, Converted to
d, i	byte, short; decimal notation (For signed representation. Use signed argument)
o	byte, short; octal notation (without a leading zero).
x, X	byte, short; hexadecimal notation (without a leading 0x or 0X), using abcdef for 0x or ABCDEF for 0X.
u	byte, short; decimal notation.
c	byte, short; single character, after conversion to unsigned char.
s	char*; characters from the string are printed until a "\0" is reached or until the number of characters indicated by the precision have been printed
f	float; decimal notation of the form [-]mmm.ddd, where the number of d's is specified by the precision. The default precision is 6; a precision of 0 suppresses the decimal point.
e, E	float; decimal notation of the form [-]m.ddddd e+-xx or [-]m.dddddE+-xx, where the number of d's specified by the precision. The default precision is 6; a precision of 0 suppresses the decimal point.
g, G	float; %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 zeros and trailing decimal point are not printed.
%	no argument is converted; print a %

The arguments that can be passed to the SSI function *printf* are:

Argument	Description
InReadSByte(<i>offset</i>)	Read a signed byte from position <i>offset</i> in the fieldbus IN area
InReadUByte(<i>offset</i>)	Read an unsigned byte from position <i>offset</i> in the fieldbus IN area
InReadSWord(<i>offset</i>)	Read a signed word from position <i>offset</i> in the fieldbus IN area
InReadUWord(<i>offset</i>)	Read an unsigned word from position <i>offset</i> in the fieldbus IN area
InReadSLong(<i>offset</i>)	Read a signed longword from position <i>offset</i> in the fieldbus IN area
InReadULong(<i>offset</i>)	Read an unsigned longword from position <i>offset</i> in the fieldbus IN area
InReadString(<i>offset</i>)	Read a string (char*) from position <i>offset</i> in the fieldbus IN area
InReadFloat(<i>offset</i>)	Read a floating point (float) value from position <i>offset</i> in the fieldbus IN area
OutReadSByte(<i>offset</i>)	Read a signed byte from position <i>offset</i> in the fieldbus OUT area
OutReadUByte(<i>offset</i>)	Read an unsigned byte from position <i>offset</i> in the fieldbus OUT area
OutReadSWord(<i>offset</i>)	Read a signed word (short) from position <i>offset</i> in the fieldbus OUT area
OutReadUWord(<i>offset</i>)	Read an unsigned word (short) from position <i>offset</i> in the fieldbus OUT area
OutReadSLong(<i>offset</i>)	Read a signed longword (long) from position <i>offset</i> in the fieldbus OUT area
OutReadULong(<i>offset</i>)	Read an unsigned longword (long) from position <i>offset</i> in the fieldbus OUT area
OutReadString(<i>offset</i>)	Read a null-terminated string from position <i>offset</i> in the fieldbus OUT area
OutReadFloat(<i>offset</i>)	Read a floating point (float) value from position <i>offset</i> in the fieldbus OUT area
ParReadSByte(<i>par</i>)	Reads parameter <i>par</i> and returns it as a signed byte
ParReadUByte(<i>par</i>)	Reads parameter <i>par</i> and returns it as an unsigned byte
ParReadSWord(<i>par</i>)	Reads parameter <i>par</i> and returns it as a signed word
ParReadUWord(<i>par</i>)	Reads parameter <i>par</i> and returns it as an unsigned word
ParReadSLong(<i>par</i>)	Reads parameter <i>par</i> and returns it as a signed longword
ParReadULong(<i>par</i>)	Reads parameter <i>par</i> and returns it as an unsigned longword
ParReadString(<i>par</i>)	Reads parameter <i>par</i> and returns it as a string
ParReadFloat(<i>par</i>)	Reads parameter <i>par</i> and returns it as a float
ParReadFormatted(<i>par</i>)	Reads parameter <i>par</i> , interprets it and returns it as a formatted string.

scanf

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

This SSI function reads a string passed from an object in a HTML form, interprets the string according to the specified in-format, and stores the result in the OUT area according to the passed arguments. The formatting of the string is equal to the standard C function call scanf()

Syntax:

```
<?--#exec cmd_argument='scanf( "ObjName", "format", Arg1, ..., ArgN), ErrVal1, ..., ErrValN'-->
```

ObjName - The name of the object with the passed data string
 format - Specifies how the passed string shall be formatted
 Arg1 - ArgN - Specifies where to write the data
 ErrVal1 -ErrValN - Optional; specifies the value/string to write in case of an error.

Character(s)	Input, Argument Type
d	Decimal number; byte, short
i	Number, byte, short. The number may be in octal (leading 0(zero)) or hexadecimal (leading 0x or 0X)
o	Octal number (with or without leading zero); byte, short
u	Unsigned decimal number; unsigned byte, unsigned short
x	Hexadecimal number (with or without leading 0x or 0X); byte, short
c	Characters; char*. The next input characters (default 1) are placed at the indicated spot. The normal skip over white space is suppressed; to read the next non-white space character, use %1s.
s	Character string (not quoted); char*, pointing to an array of characters large enough for the string and a terminating "\0" that will be added.
e, f, g	Floating-point number with optional sign, optional decimal point and optional exponent; float*
%	Literal %; no assignment is made.

The conversion characters d, i, o, u and x may be preceded by l (letter ell) to indicate that a pointer to 'long' appears in the argument list rather than a 'byte' or a 'short'

The arguments that can be passed to the SSI function scanf are:

Argument	Description
OutWriteByte(offset)	Write a byte to position <i>offset</i> in the fieldbus OUT area
OutWriteWord(offset)	Write a word to position <i>offset</i> in the fieldbus OUT area
OutWriteLong(offset)	Write a long to position <i>offset</i> in the fieldbus OUT area
OutWriteString(offset)	Write a string to position <i>offset</i> in the fieldbus OUT area
OutWriteFloat(offset)	Write a floating point value to position <i>offset</i> in the fieldbus OUT area
ParWriteByte(par)	Writes a byte to parameter <i>par</i>
ParWriteWord(par)	Writes a word to parameter <i>par</i>
ParWriteLong(par)	Writes a longword to parameter <i>par</i>
ParWriteString(par)	Writes a string to parameter <i>par</i>
ParWriteFloat(par)	Writes a float to parameter <i>par</i>
ParWriteFloat(par)	Writes a float to parameter <i>par</i>
ParWriteFormatted(par)	Writes a formatted value to parameter <i>par</i>

Default output:

```
Write succeeded
Write failed
```

IncludeFile

This function includes the contents of a file.

Syntax:

```
<?--#exec cmd_argument='IncludeFile( "File name" )'-->
```

Default output:

Success	- <File content>
Failure	- Failed to open <filename>

SaveToFile

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

This SSI function saves the contents of a passed form to a file. The passed name/value pair will be written to the file "File name" separated by the "Separator" string. The [Append|Overwrite] parameter determines if the specified file shall be overwritten, or if the data in the file shall be appended.

Syntax:

```
<?--#exec cmd_argument='SaveToFile( "File name", "Separator", [Append|Overwrite] )'-->
```

Default output:

Success	- Form saved to file
Failure	- Failed to save form

GetConfigItem

This function returns the value of a specific key in a configuration file. The format of the file must be as follows:

```
[key1]
Value1
[key2]
Value2
...
[keyN]
ValueN
```

Syntax:

```
<?--#exec cmd_argument='GetConfigItem( "file", "key", "separator" )'-->
```

file	- File to read from
key	- Key to read value from
separator-	Optional; specifies line-separator ('CRLF' (default), 'LF' or 'CR')

Default output:

Success	- (Value associated with the specified key)
Failure	- Could not get value for [key]

For information about how to change the SSI output, please see "Changing SSI output" on page 63 .

SetConfig

This function stores a passed HTML-form as a configuration file. The field name will be used as key, and the field value will be stored as the value to that key. If the passed key (field name) isn't present in the file, it will be added. If the specified file does not exist, it will be created.

Field names beginning with underscore ('_') will be excluded.

Syntax:

```
<?--#exec cmd_argument='SetConfig( "File name" )'-->
```

Default output:

Success – Configuration stored to “File name”

Failure – Could not store configuration to “File name”

For information about how to change the SSI output, please see
“Changing SSI output” on page 63 .

Example:

An HTML-form with the following fields...

- Name = “Speed”, value = “48”
- Name = “Temp”, value = “20”
- Name = “_B1”, value = “submit”
(Button used to submit the form, not stored.)

... generates a file with the following format:

```
[Speed]
48
[Temp]
20
```

DisplayStationName

This function returns the PROFINET Station Name.

Syntax:

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

Default output:

```
<PROFINET Station Name>
```

DisplayStationType

This function returns the PROFINET Station Type.

Syntax:

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

Default output:

```
<PROFINET Station Type>
```

DisplayVendorId

This function returns the PROFINET Vendor ID.

Syntax:

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

Default output:

```
<PROFINET Vendor ID>
```

DisplayDeviceId

This function returns the PROFINET Device ID.

Syntax:

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

Default output:

```
<PROFINET Device ID>
```

Changing SSI output

There are two methods of changing the output strings from SSI functions:

1. Changing SSI output defaults by creating a file called "\ssi_str.cfg" containing the output strings for all SSI functions in the system
2. Temporary changing the SSI output using SsiOutput()

SSI Output String File

If the file "\ssi_str.cfg" is found in the file system and the file is correct according to the specification below, the SSI functions will use the output strings specified in this file instead of the default strings.

The files shall have the following format:

```
[StoreEtnConfig]
Success: "String to use on success"
Invalid IP: "String to use when the IP address is invalid"
Invalid Subnet: "String to use when the Subnet mask is invalid"
Invalid Gateway: "String to use when the Gateway address is invalid"
Invalid Email server: "String to use when the SMTP address is invalid"
Invalid IP or Subnet: "String to use when the IP address and Subnet mask does
not match"
Invalid DNS1: "String to use when the primary DNS cannot be found"
Invalid DNS2: "String to use when the secondary DNS cannot be found"
Save Error: "String to use when storage fails"
Invalid DHCP state: "String to use when the DHCP state is invalid"

[scanf]
Success: "String to use on success"
Failure: "String to use on failure"

[IncludeFile]
Failure: "String to use when failure"1

[SaveToFile]
Success: "String to use on success"
Failure: "String to use on failure"1

[SaveDataToFile]
Success: "String to use on success"
Failure: "String to use on failure"1

[GetText]
Success: "String to use on success"
Failure: "String to use on failure"
```

The contents of this file can be redirected by placing the line '[File path]' on the first row, and a file path on the second.

Example:

```
[File path]
\user\ssi_strings.cfg
```

In this example, the settings described above will be loaded from the file 'user\ssi_strings.cfg'.

1. '%s' includes the filename in the string

Temporary SSI Output change

The SSI output for the next called SSI function can be changed with the SSI function “SsiOutput()”. The next called SSI function will use the output according to this call. Thereafter the SSI functions will use the default outputs or the outputs defined in the file ‘\ssi_str.cfg’. The maximum size of a string is 128 bytes.

Syntax:

```
<?--#exec cmd_argument='SsiOutput ( "Success string", "Failure string" )'-->
```

Example:

This example shows how to change the output strings for a scanf SSI call.

```
<?--#exec cmd_argument='SsiOutput ( "Parameter1 updated", "Error" )'-->  
<?--#exec cmd_argument="scanf ( "Parameter1", "%d", OutWriteByte(0) )'-->
```

Fieldbus Specific Parameters

General

#	R/W	Name	Size	Default Value	Modbus Address	HOS Object
100	R	FB status	2 byte	N/A	0x7000	Fieldbus Object (0xA0)
101	W	FB init	1 byte	0x00	0x7001	
102	W	FB Password	2 byte	N/A	0x7003	
104	R	DIP switch SSC	1 byte	N/A	0x7006	SSC Object (0xA2)
116	R	MAC address	6 byte	N/A	0x701B-0x701D	Fieldbus Object (0xA0)
141	R	Serial number	4 byte	N/A	0x704D-0x704E	Device Object (0x80)

Communication Settings

#	R/W	Name	Size	Default Value	Modbus Address	HOS Object
103	RW	IP address cfg	4 byte	0.0.0.0	0x7004-0x7005	Fieldbus Object (0xA0)
105	R	IP address act	4 byte	N/A	0x7007-0x7008	
106	RW	Subnet mask cfg	4 byte	0.0.0.0	0x7009-0x700A	
107	R	Subnet mask act	4 byte	N/A	0x700B-0x700C	
108	RW	GW address cfg	4 byte	0.0.0.0	0x700D-0x700E	
109	R	GW address act	4 byte	N/A	0x700F-0x7010	
114	RW	DHCP enable cfg	1 byte	0x00	0x7019	
115	R	DHCP enable act	1 byte	N/A	0x701A	
118	R	Data rate act	1 byte	N/A	0x701F	
120	R	Duplex act	1 byte	N/A	0x7021	
136	RW	HICP enable	1 byte	0x01	0x7039	
137	RW	HICP password	30 byte	-	0x703A-0x7048	
144	RW	Gleaning enable	1 byte	0x01	0x706F	

Server Settings

#	R/W	Name	Size	Default Value	Modbus Address	HOS Object
121	RW	Web srv enable	1 byte	0x01	0x7022	Fieldbus Object (0xA0)
122	RW	FTP srv enable	1 byte	0x01	0x7023	
123	RW	Telnet enable	1 byte	0x01	0x7024	

E-mail Client

#	R/W	Name	Size	Default Value	Modbus Address	HOS Object
126	RW	SMTP srv address	4 byte	0.0.0.0	0x7027-0x7028	Fieldbus Object (0xA0)
127	R	Trigged emails	2 byte	N/A	0x7029	
128	R	SMTP errors	2 byte	N/A	0x702A	
129	W	Send email	2 byte	0x0000	0x702B	
142	RW	SMTP user name	32 byte	""	0x704F-0x705E	
143	RW	SMTP password	32 byte	""	0x705F-0x706E	

File System

#	R/W	Name	Size	Default Value	Modbus Address	HOS Object
124	RW	Admin mode cfg	1 byte	0x00	0x7025	Fieldbus Object (0xA0)
125	R	Admin mode act	1 byte	N/A	0x7026	
130	RW	VFS enable	1 byte	0x01	0x702C	
131	RW	RAM disc path	16 byte	"\RAM"	0x702D-0x7034	

Modbus-TCP Related Parameters

#	R/W	Name	Size	Default Value	Modbus Address	HOS Object
132	RW	MB/TCP conn TO	2 byte	60	0x7035	Fieldbus Object (0xA0)
133	RW	MB/TCP enable	1 byte	0x01	0x7036	
134	RW	In bit size	2 byte	0x00	0x7037	
135	RW	Out bit size	2 byte	0x00	0x7038	

PROFINET IO Related Parameters

#	R/W	Name	Size	Default Value	Modbus Address	HOS Object
160	R(W)	PRT version numb	2 byte	N/A	0x7100	Fieldbus Object (0xA0)
161	R(W)	PRT serial numb	4 byte	N/A	0x7101-0x7102	
162	R(W)	PRT vendor id	2 byte	0x010C	0x7103	
163	R(W)	PRT device id	2 byte	0x0008	0x7104	
164	R(W)	PRT station type	30 byte	'ABIC-PRT'	0x7105-0x7113	
165	R	Port 1 MAC addr	6 byte	N/A	0x7184-0x7186	
166	RW	PRT station name	240 byte	"	0x7187-0x71FE	
167	R	ArCheckInd Data	12 byte	N/A	0x71FF-0x7204	
168	R	CfgMisMalInd Data	18 byte	N/A	0x7205-0x720D	
169	R	ArInfoInd Data	2048 byte	N/A	0x720E-0x760D	
170	R	EndOfPrmInd Data	10 byte	N/A	0x760E-0x7612	
171	R	ArOfflineInd Data	4 byte	N/A	0x7613-0x7614	
172	R	ArAbortInd Data	4 byte	N/A	0x7615-0x7616	
173	R	PlugSubModFailed	4 byte	N/A	0x7617-0x761A	
174	R	PniInitErrCode	2 byte	0x0000	0x761B	

Application Parameters

#	R/W	Name	Size	Default Value	Modbus Address	HOS Object
200	a	Application Parameter #1	a	a	0x8000	Application Parameter Object (0x85)
201	a	Application Parameter #2	a	a	a	
...	a	Application Parameter #3-99	a	a	a	
299	a	Application Parameter #100	a	a	a	

a. Parameter dependent.

General

FB Status (Parameter #100)

This parameter holds information about the current bus status.

Parameter Name	'FB Status'
Parameter no.	100
Modbus Address	0x7000
Default Value	-
Range	0x0000-0xFFFF
Size	2 bytes
Stored in NV RAM	No
Access	R

Bit Layout

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
(reserved)						IDLE	BUS	(reserved)	PSF	ARA	PRM	CFG	ARI	CAR	

- **CAR**
 - 1: A PROFINET application relation (AR) received
 - 0: -
- **ARI**
 - 1: Information about the application relation (AR) has been received
 - 0: -
- **CFG**
 - 1: Module / Submodule configuration mismatch data received
 - 0: -
- **PRM**
 - 1: Parameter end indication data received
 - 0: -
- **ARA**
 - 1: Application relation (AR) has been aborted
 - 0: -
- **PSF**
 - 1: Plug submodule failed (can only be triggered if a submodule is plugged at runtime by the application)
 - 0: -
- **BUS**
 - 1: Bus is online
 - 0: Bus is offline

- **IDLE**
 - 1: IO Controller is in idle / clear mode
 - 0: IO Controller is in run mode

FB init (Parameter #101)

If the Anybus module has been initialized by the application by setting parameter 'Module Mode' to 'Fieldbus Specific Init' (0x02), this parameter will contain indications to events triggered by the PROFINET IO network. The application needs to confirm the events signaled with register 'FB status' by writing to this register. Please note that events can only be confirmed (that is, a received event can never be negatively acknowledged).

Parameter Name	'FB init'
Parameter no.	101
Modbus Address	0x7001
Default Value	0x00
Range	0x00-0xFF
Size	1 byte
Stored in NV RAM	No
Access	W

Bit Layout

b7	b6	b5	b4	b3	b2	b1	b0
(reserved)	PSF	ARA	PRM	CFG	ARI	CAR	

- **CAR**
 - 1: Confirm application relation (AR) received event
 - 0: -
- **ARI**
 - 1: Confirm reception of information about the application relation (AR) event
 - 0: -
- **CFG**
 - 1: Confirm reception of a module / submodule configuration mismatch event
 - 0: -
- **PRM**
 - 1: Confirm reception of parameter end indication event
 - 0: -
- **ARA**
 - 1: Confirm reception of application relation (AR) abort event
 - 0: -
- **PSF**
 - 1: Confirm reception of plug submodule failed event
 - 0: -

FB Password (Parameter #102)

This parameter grants write access to the following parameters provided that a valid password is supplied.

- PRT version numb (Parameter #160)
- PRT serial numb (Parameter #161)
- PRT vendor id (Parameter #162)
- PRT device id (Parameter #163)
- PRT station type (Parameter #164)
- System description (Fieldbus Object, Attribute 56)
- Interface description (Fieldbus Object, Attribute 57)

(The password can be obtained by contacting HMS)

Parameter Name	'FB Password'
Parameter no.	102
Modbus Address	0x7003
Default Value	-
Range	0x0000-0xFFFF
Size	2 bytes
Stored in NV RAM	No
Access	W

DIP switch SSC (Parameter #104)

This parameter holds the automatically configured fieldbus node address from fieldbus specific input register on the SSC interface. Note that in order for this value to be valid, the 'NA' bit (parameter #8 'Configuration Bits') must be cleared (0).

Parameter Name	'DIP switch SSC'
Parameter no.	104
Modbus Address	0x7006
Default Value	-
Range	0x00... 0xFF
Size	1 byte
Stored in NV RAM	No
Access	R

MAC address (Parameter #116)

This parameter holds the Ethernet MAC address of the module.

Parameter Name	'MAC address'
Parameter no.	116
Modbus Address	0x701B...0x 701D
Default Value	-
Range	-
Size	6 bytes
Stored in NV RAM	No
Access	R

Serial Number (Parameter #141)

This parameter holds the production serial number of the module.

Parameter Name	'Serial Number'
Parameter no.	141
Modbus Address	0x704D-0x704E
Default Value	-
Range	0x0... 0xFFFFFFFF
Size	4 bytes
Stored in NV RAM	No
Access	RW

Communication Settings

IP address cfg (Parameter #103)

This parameter holds the manually configured IP address.

See also...

- “TCP/IP Settings” on page 20

Parameter Name	'IP address cfg'
Parameter no.	103
Modbus Address	0x7004... 0x7005
Default Value	0.0.0.0
Range	0.0.0.0... 255.255.255.255
Size	4 bytes
Stored in NV RAM	No
Access	RW

IP address act (Parameter #105)

This parameter holds the actual IP address.

See also...

- “TCP/IP Settings” on page 20

Parameter Name	'IP address act'
Parameter no.	105
Modbus Address	0x7007... 0x7008
Default Value	-
Range	0.0.0.0... 255.255.255.255
Size	4 bytes
Stored in NV RAM	No
Access	R

Subnet mask cfg (Parameter #106)

With this parameter it is possible to configure the Subnet mask.

Note: The module must be restarted in order for changes to have effect.

See also...

- “TCP/IP Settings” on page 20

Parameter Name	'Subnet mask cfg'
Parameter no.	106
Modbus Address	0x7009... 0x700A
Default Value	-
Range	0.0.0.0... 255.255.255.255
Size	4 bytes
Stored in NV RAM	No
Access	RW

Subnet mask act (Parameter #107)

This parameter holds the currently used Subnet mask.

See also...

- “TCP/IP Settings” on page 20

Parameter Name	'Subnet mask act'
Parameter no.	107
Modbus Address	0x700B...0x700C
Default Value	-
Range	0.0.0.0... 255.255.255.255
Size	4 bytes
Stored in NV RAM	No
Access	R

GW address cfg (Parameter #108)

With this parameter it is possible to configure the Gateway address.

Note: The module must be restarted in order for changes to have effect.

See also...

- “TCP/IP Settings” on page 20

Parameter Name	'GW address cfg'
Parameter no.	108
Modbus Address	0x700D... 0x700E
Default Value	-
Range	0.0.0.0... 255.255.255.255
Size	4 bytes
Stored in NV RAM	No
Access	RW

GW address act (Parameter #109)

This parameter holds the currently used Gateway address.

Note: The module must be restarted in order for changes to have effect.

See also...

- “TCP/IP Settings” on page 20

Parameter Name	'GW address act'
Parameter no.	109
Modbus Address	0x700F... 0x7010
Default Value	-
Range	0.0.0.0... 255.255.255.255
Size	4 bytes
Stored in NV RAM	No
Access	R

DHCP enable cfg (Parameter #114)

This parameter enables/disables DHCP support.

Note: The module must be restarted in order for changes to have effect.

See also...

- “TCP/IP Settings” on page 20

Parameter Name	'DHCP enable cfg'
Parameter no.	114
Modbus Address	0x7019
Default Value	0
Range	0: Disable 1: Enable
Size	1 bytes
Stored in NV RAM	No
Access	RW

DHCP enable act (Parameter #115)

This parameter holds the current DHCP state.

See also...

- “TCP/IP Settings” on page 20

Parameter Name	'DHCP enable act'
Parameter no.	115
Modbus Address	0x701A
Default Value	-
Range	0: Disabled 1: Enabled
Size	1 byte
Stored in NV RAM	No
Access	R

Data rate act (Parameter #118)

This parameter holds the currently used data rate of the physical link.

See also...

- “TCP/IP Settings” on page 20

Parameter Name	'Data rate act'
Parameter no.	118
Modbus Address	0x701F
Default Value	-
Range	2: 100Mbps
Size	1 bytes
Stored in NV RAM	No
Access	R

Duplex Act (Parameter #120)

This parameter holds the currently used duplex mode of the physical link.

See also...

- “Physical Link Settings” on page 20

Parameter Name	'Duplex Act'
Parameter no.	120
Modbus Address	0x7021
Default Value	-
Range	2: Full Duplex
Size	1 byte
Stored in NV RAM	No
Access	R

HICP Enable (Parameter #136)

This parameter enables/disables support for the Anybus IPconfig (HICP) protocol.

See also...

- “Anybus IPconfig (HICP)” on page 22

Parameter Name	'HICP Enable'
Parameter no.	136
Modbus Address	0x7039
Default Value	1
Range	0: Disable 1: Enable
Size	1 byte
Stored in NV RAM	No
Access	RW

HICP Password (Parameter #137)

The Anybus IPconfig (HICP) protocol allows the IP configuration to be protected from unauthorized changes via a password, which can be specified through this parameter.

See also...

- “Anybus IPconfig (HICP)” on page 22

Parameter Name	'HICP Password'
Parameter no.	137
Modbus Address	0x703A... 0x7048
Default Value	-
Range	-
Size	30 bytes
Stored in NV RAM	No
Access	RW

ARP Gleaning Enable (Parameter #144)

See also...

- “ARP Gleaning” on page 22

Parameter Name	'ARP Gleaning Enable'
Parameter no.	144
Modbus Address	0x706F
Default Value	1
Range	0: Disable 1: Enable
Size	1 byte
Stored in NV RAM	No
Access	RW

Server Settings

Web Srv Enable (Parameter #121)

This parameter enables/disables the built-in web server.

See also...

- “Web Server” on page 51

Parameter Name	'Web Srv Enable'
Parameter no.	121
Modbus Address	0x7022
Default Value	1
Range	0: Disable 1: Enable
Size	1 byte
Stored in NV RAM	No
Access	RW

FTP Srv Enable (Parameter #122)

This parameter enables/disables the built-in FTP server.

See also...

- “FTP Server” on page 44

Parameter Name	'FTP Srv Enable'
Parameter no.	122
Modbus Address	0x7023
Default Value	1
Range	0: Disable 1: Enable
Size	1 byte
Stored in NV RAM	No
Access	RW

Telnet Srv Enable (Parameter #123)

This parameter enables/disables the built-in Telnet server.

See also...

- “Telnet Server” on page 46

Parameter Name	'Telnet Srv Enable'
Parameter no.	123
Modbus Address	0x7024
Default Value	1
Range	0: Disable 1: Enable
Size	1 byte
Stored in NV RAM	No
Access	RW

E-mail Client

SMTP Srv Address (Parameter #126)

With this parameter it is possible to configure the SMTP server address.

See also...

- “E-mail Client” on page 42

Parameter Name	'SMTP Srv Address'
Parameter no.	126
Modbus Address	0x7027... 0x7028
Default Value	0.0.0.0
Range	0.0.0.0... 255.255.255.255
Size	4 bytes
Stored in NV RAM	No
Access	RW

Triggered Emails (Parameter #127)

This parameter indicates how many e-mail trigger events that has been detected by the module.

See also...

- “E-mail Client” on page 42

Parameter Name	'Triggered Emails'
Parameter no.	127
Modbus Address	0x7029
Default Value	0x0000
Range	0x0000... 0xFFFF
Size	2 bytes
Stored in NV RAM	No
Access	R

SMTP Errors (Parameter #128)

This parameter indicates how many e-mails that the module failed to send to the SMTP server.

See also...

- “E-mail Client” on page 42

Parameter Name	'SMTP Errors'
Parameter no.	128
Modbus Address	0x702A
Default Value	0x0000
Range	0x0000... 0xFFFF
Size	2 bytes
Stored in NV RAM	No
Access	R

Send Email (Parameter #129)

This parameter sends a predefined e-mail stored in the file system.

The MSB in the data word determines the e-mail type (0: admin, 1: normal), and the LSB specifies the number of the message.

Example:

The value 0x0004 issues admin e-mail no. #4, and 0x010A issues normal user e-mail no. #10.

See also...

- “E-mail Client” on page 42

Parameter Name	'Send Email'
Parameter no.	129
Modbus Address	0x702B
Default Value	0x0000
Range	0x0000...0x010A
Size	2 bytes
Stored in NV RAM	No
Access	W

SMTP User Name (Parameter #142)

See also...

- “E-mail Client” on page 42

Parameter Name	'SMTP User Name'
Parameter no.	142
Modbus Address	0x704F... 0x705E
Default Value	""
Range	String, NULL-terminated
Size	Up to 32 bytes, including NULL-termination
Stored in NV RAM	No
Access	RW

SMTP Password (Parameter #143)

See also...

- “E-mail Client” on page 42

Parameter Name	'SMTP Password'
Parameter no.	143
Modbus Address	0x705F... 0x706E
Default Value	""
Range	String, NULL-terminated
Size	Up to 32 bytes, including NULL-termination
Stored in NV RAM	No
Access	RW

File System

Admin Mode Cfg (Parameter #124)

This parameter enables/disables global admin mode.

Note: The module must be restarted for changes to have effect.

See also...

- “FTP Server” on page 44 (“Security Levels” on page 44 , “User Accounts” on page 44)
- “Telnet Server” on page 46 (“Security Levels” on page 46 , “User Accounts” on page 46)

Parameter Name	'Admin Mode Cfg'
Parameter no.	124
Modbus Address	0x7025
Default Value	0
Range	0: Disable global admin mode 1: Enable global admin mode
Size	1 byte
Stored in NV RAM	No
Access	RW

Admin Mode Act (Parameter #125)

This parameter indicates whether the module is operating in global admin mode or not.

See also...

- “FTP Server” on page 44 (“Security Levels” on page 44 , “User Accounts” on page 44)
- “Telnet Server” on page 46 (“Security Levels” on page 46 , “User Accounts” on page 46)

Parameter Name	'Admin Mode Act'
Parameter no.	125
Modbus Address	0x7026
Default Value	-
Range	0: Disabled 1: Enabled
Size	1 byte
Stored in NV RAM	No
Access	RW

VFS Enable (Parameter #130)

This parameter enables/disables the virtual file system (VFS).

See also...

- “Default Web Pages” on page 51

Parameter Name	'VFS Enable'
Parameter no.	130
Modbus Address	0x702C
Default Value	1
Range	0: Disable 1: Enable
Size	2 bytes
Stored in NV RAM	No
Access	RW

RAM-Disc Path (Parameter #131)

This parameter specifies where to mount the volatile part of the file system (i.e. the RAM disc). The specified directory must be empty or nonexisting; an empty value ("") disables the RAM-disc.

See also...

- “Filesystem” on page 18
- “Storage Areas” on page 18

Parameter Name	'RAM Disc Path'
Parameter no.	131
Modbus Address	0x702D... 0x7034
Default Value	"\\RAM"
Range	String, NULL-terminated
Size	Up to 16 bytes (NULL-termination included)
Stored in NV RAM	No
Access	RW

Modbus-TCP related Parameters

MB/TCP Conn TO (Parameter #132)

This parameter specifies the Modbus-TCP connection timeout in seconds. If no Modbus-TCP query has been received within the specified time period, the Modbus-TCP connection will be closed.

See also...

- “Modbus-TCP Implementation” on page 40

Parameter Name	'MB/TCP Conn TO'
Parameter no.	132
Modbus Address	0x7035
Default Value	60
Range	0... 65535 (0 = Timeout disabled)
Size	2 bytes
Stored in NV RAM	No
Access	RW

MB/TCP Enable (Parameter #133)

This parameter enables/disables support for the Modbus-TCP protocol.

See also...

- “Modbus-TCP Implementation” on page 40

Parameter Name	'MB/TCP Enable'
Parameter no.	133
Modbus Address	0x7036
Default Value	1
Range	0: Disable 1: Enable
Size	1 byte
Stored in NV RAM	No
Access	RW

In bit size (Parameter #134)

This parameter specifies how many bytes of the fieldbus input data that shall be represented as coils/ discretes. The remainder will be represented as registers.

See also...

- “Modbus-TCP Implementation” on page 40 (“Fieldbus Input Data (FB In)” on page 41)

Parameter Name	'In bit size'
Parameter no.	134
Modbus Address	0x7037
Default Value	0x0000
Range	0x0000... 0x0090
Size	2 bytes
Stored in NV RAM	No
Access	RW

Out bit size (Parameter #135)

This parameter specifies how many bytes of the fieldbus output data that shall be represented as coils/ discretes. The remainder will be represented as registers.

See also...

- “Modbus-TCP Implementation” on page 40 (“Fieldbus Output Data (FB Out)” on page 41)

Parameter Name	'Out bit size'
Parameter no.	135
Modbus Address	0x7038
Default Value	0x0000
Range	0x0000... 0x0090
Size	2 byte
Stored in NV RAM	No
Access	RW

PROFINET IO Related Parameters

PRT version numb (Parameter #160)

This parameter holds the PROFINET IO version number. By altering this parameter the version number of the PROFINET IO Object is changed. In order to alter the version number of the I&M data also the version number(s) of the I&M Object needs to be changed.

Note: The module must be restarted for changes to have effect.

If this value is set to 0xFFFF, the default value will be used (the Anybus-IC Product Revision).

Parameter Name	'PRT version numb'
Parameter no.	160
Modbus Address	0x7100
Default Value	Anybus-IC Product Revision
Range	0x0000-0xFFFF
Size	2 byte
Stored in NV RAM	Yes
Access	R(W)

PRT serial numb (Parameter #161)

This parameter holds the PROFINET IO serial number. By altering this parameter the serial number of the PROFINET IO Object is changed. In order to alter the serial number of the I&M data also the serial number of the I&M Object needs to be changed.

Note: The module must be restarted for changes to have effect.

If this value is set to 0xFFFFFFFF, the default value will be used (the Anybus-IC Serial Number).

Parameter Name	'PRT serial numb'
Parameter no.	161
Modbus Address	0x7101-0x7102
Default Value	Anybus-IC Serial Number
Range	0x00000000-0xFFFFFFFF
Size	4 byte
Stored in NV RAM	Yes
Access	R(W)

PRT vendor id (Parameter #162)

This parameter holds the PROFINET vendor ID of the device.

Note: The module must be restarted for changes to have effect.

Parameter Name	'PRT vendor id'
Parameter no.	162
Modbus Address	0x7103
Default Value	0x010C
Range	0x0000-0xFFFF
Size	2 byte
Stored in NV RAM	Yes
Access	R(W)

PRT device id (Parameter #163)

This parameter holds the PROFINET device ID of the device.

Note: The module must be restarted for changes to have effect.

Parameter Name	'PRT device id'
Parameter no.	163
Modbus Address	0x7104
Default Value	0x0008
Range	0x0000-0xFFFF
Size	2 byte
Stored in NV RAM	Yes
Access	R(W)

PRT station type (Parameter #164)

This parameter holds the PROFINET station type of the device.

Note: The module must be restarted for changes to have effect.

See also...

- FB Password (Parameter #102)

Parameter Name	'PRT station type'
Parameter no.	164
Modbus Address	0x7105-0x7113
Default Value	'ABIC-PRT'
Range	Null-terminated string (do not pad with spaces)
Size	30 byte
Stored in NV RAM	Yes
Access	R(W)

Port 1 MAC addr (Parameter #165)

This parameter holds the MAC address for port 1 of the device. Set at production of the Anybus module.

Parameter Name	'Port1 MAC addr'
Parameter no.	165
Modbus Address	0x7184-0x7186
Default Value	-
Range	-
Size	6 byte
Stored in NV RAM	Yes
Access	R

PRT station name (Parameter #166)

This parameter holds the PROFINET Station name of the Anybus module. Normally this parameter is assigned via DCP (via PROFINET).

Note: The module must be restarted for changes to have effect.

Parameter Name	'PRT station name'
Parameter no.	166
Modbus Address	0x7187-0x71FE
Default Value	" (empty string)
Range	Null-terminated string (do not pad with spaces)
Size	240 byte
Stored in NV RAM	Yes
Access	RW

ArCheckInd Data (Parameter #167)

This parameter holds the information handed over by the IO controller in the most recent attempt to establish an IO connection between the IO controller and the Anybus module.

See also...

- "FB Status (Parameter #100)" on page 68

Parameter Name	'ArCheckInd Data'
Parameter no.	167
Modbus Address	0x71FF-0x7204
Default Value	N/A
Range	-
Size	12 byte
Stored in NV RAM	No
Access	R

Data Interpretation

Byte	Contents	Description
0	AR handle (high byte)	Handle for the application relationship (AR)
1	AR handle (low byte)	
2	IP address (high word, high byte)	IP address of the remote station (IO Controller)
3	IP address (high word, low byte)	
4	IP address (low word, high byte)	
5	IP address (low word, low byte)	
6	AR Type (high byte)	Indicates the type of application relationship: 1 - IO_AR_SINGLE 2 - IO_AR_CIR 3 - IO_AR_REDUNDANT_CONTROLLER 4 - IO_AR_REDUNDANT_DEVICE 5 - SUPERVISOR_AR
7	AR Type (low byte)	
8	AR Properties (high word, high byte)	Bit field indicating the properties of the AR: Bit 0-2, State: 0 - Backup 1 - Primary Bit 3, Supervisor takeover allowed: 0 - Not allowed 1 - Allowed Bit 4, Parameterization server: 0 - External parameter server 1 - Connection manager initiator Bit 5-6, Data rate: 0 - At least 100 Mbps 1 - 100 Mbps 2 - 1 Gbps 3 - 10 Gbps Bit 8, Device access: 0 - AR context 1 - Device context Bit 9-10, Companion AR: 0 - Single AR 1 - First AR 2 - Companion AR
9	AR Properties (high word, low byte)	
10	AR Properties (low word, high byte)	
11	AR Properties (low word, low byte)	

CfgMisMalndData (Parameter #168)

This parameter holds the information about the most recent configuration mismatch, between the IO controller and the Anybus module.

See also...

- “FB Status (Parameter #100)” on page 68

Parameter Name	'CfgMisMalnd Data'
Parameter no.	168
Modbus Address	0x7205-0x720D
Default Value	N/A
Range	-
Size	18 byte
Stored in NV RAM	No
Access	R

Data Interpretation

Byte	Contents	Description
0	AR handle (high byte)	Handle for the application relationship
1	AR handle (low byte)	
2	API (high word, high byte)	Application process interface
3	API (high word, low byte)	
4	API (low word, high byte)	
5	API (low word, low byte)	
6	Slot (high byte)	Slot number of mismatch
7	Slot (low byte)	
8	Subslot (high byte)	Subslot number of mismatch
9	Subslot (low byte)	
10	Expected Module Identifier (high word, high byte)	Module identifier (as stated in the GSD-file) derived from the IO Controller configuration
11	Expected Module Identifier (high word, low byte)	
12	Expected Module Identifier (low word, high byte)	
13	Expected Module Identifier (low word, low byte)	
14	Expected Submodule Identifier (high word, high byte)	Submodule identifier (as stated in the GSD-file) derived from the IO Controller configuration
15	Expected Submodule Identifier (high word, low byte)	
16	Expected Submodule Identifier (low word, high byte)	
17	Expected Submodule Identifier (low word, low byte)	

ArlInfolnd Data (Parameter #169)

This parameter holds the information about the most recent information of the application relationship (AR) handed over by the IO controller during connection establishment.

See also...

- “FB Status (Parameter #100)” on page 68

Parameter Name	'ArlInfolnd Data'
Parameter no.	169
Modbus Address	0x720E-0x760D
Default Value	N/A
Range	-
Size	2048 byte
Stored in NV RAM	No
Access	R

Data Interpretation

Byte	Contents	Description
0	Total length (high byte)	Length of the data field
1	Total length (low byte)	
2	AR handle (high byte)	Handle for the application relationship
3	AR handle (low byte)	
4-2047	Data field	The actual data of the AR information received. Coded as described below.

Data Field Description

The 'Data field' from above is coded as follows:

Data type	Name			Description	
UINT16	iNbrApi			Number of APIs	
UINT32		IApiNbr		API number	
UINT16		iNbrMod		Initial block (2 bytes)	
UINT16			iSlotNbr	Module block (8 bytes)	
UINT16			iNbrSubMod		
UINT32			IModIdent		
UINT16					iSubSlotNbr
UINT32				iSubModIdent	
UINT16				iInDataLength	
UINT16				iOutDa- taLength	

EndOfPrmInd Data (Parameter #170)

This parameter holds the information about the most recent end-of-parameterization data handed over by the IO controller during connection establishment.

See also...

- “FB Status (Parameter #100)” on page 68

Parameter Name	'EndOfPrmInd Data'
Parameter no.	170
Modbus Address	0x720E-0x760D
Default Value	N/A
Range	-
Size	2048 byte
Stored in NV RAM	No
Access	R

Data Interpretation

Byte	Contents	Description
0	AR handle (high byte)	Handle for application relationship
1	AR handle (low byte)	
2	API (high word, high byte)	Application process instance, only valid if Subslot > 0
3	API (high word, low byte)	
4	API (low word, high byte)	
5	API (low word, low byte)	
6	Slot (high byte)	Slot number affected by the command, only valid if Subslot > 0
7	Slot (low byte)	
8	Subslot (high byte)	Subslot number affected by the command 0 - Command applies to all modules of the configuration > 0 - Command applies the specified slot/subslot
9	Subslot (low byte)	

When this command is acknowledged the Anybus module will issue a “Ready for data exchange” to the IO controller.

ArOfflInd Data (Parameter #171)

This parameter holds the information about the most recent offline indication data.

See also...

- “FB Status (Parameter #100)” on page 68

Parameter Name	'ArOfflInd Data'
Parameter no.	171
Modbus Address	0x7613-0x7614
Default Value	N/A
Range	-
Size	4 byte
Stored in NV RAM	No
Access	R

Data Interpretation

Byte	Contents	Description
0	AR handle (high byte)	Handle for the application relationship
1	AR handle (low byte)	
2	Reason code (high byte)	Reason code for the offline transaction: 0 - No reason (unknown reason) 3 - Out of memory 4 - Add provider or consumer failed 5 - Consumer miss 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 - Rerun 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/plug 26 - AP removed 27 - Link down 28 - Could not register multicast MAC 29 - Not synchronized 30 - Wrong topology 31 - DCP Station Name changed 32 - DCP Reset to factory settings 33 - Cannot start companion AR
3	Reason code (low byte)	

ArAbortInd Data (Parameter #172)

This parameter holds the information about the most recent offline indication data.

See also...

- “FB Status (Parameter #100)” on page 68

Parameter Name	'ArAbortInd Data'
Parameter no.	172
Modbus Address	0x7615-0x7616
Default Value	N/A
Range	-
Size	4 byte
Stored in NV RAM	No
Access	R

Data Interpretation

Byte	Contents	Description
0	AR handle (high byte)	Handle for the application relationship
1	AR handle (low byte)	
2	Reason code (high byte)	Reason code for the offline transaction: 0 - No reason (unknown reason) 3 - Out of memory 4 - Add provider or consumer failed 5 - Consumer miss 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 - Rerun 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/plug 26 - AP removed 27 - Link down 28 - Could not register multicast MAC 29 - Not synchronized 30 - Wrong topology 31 - DCP Station Name changed 32 - DCP Reset to factory settings 33 - Cannot start companion AR
3	Reason code (low byte)	

PlugSubModFailed (Parameter #173)

This parameter holds the information about the failure reason when the Anybus module has failed plugging a requested submodule.

See also...

- “FB Status (Parameter #100)” on page 68

Parameter Name	'PlugSubModFailed'
Parameter no.	173
Modbus Address	0x7617-0x761A
Default Value	N/A
Range	-
Size	8 byte
Stored in NV RAM	No
Access	R

Data Interpretation

Byte	Contents	Description
0	API (high word, high byte)	Application process instance
1	API (high word, low byte)	
2	API (low word, high byte)	
3	API (low word, low byte)	
4	Slot (high byte)	Slot number affected by the command
5	Slot (low byte)	
6	Subslot (high byte)	Subslot number affected by the command
7	Subslot (low byte)	

PniolInitErrCode (Parameter #174)

This parameter holds the information about the failure reason when the Anybus module failed to initialize the PROFINET IO stack. The value of this parameter can be handed over to HMS support for investigation.

Parameter Name	'PniolInitErrCode'
Parameter no.	174
Modbus Address	0x761B
Default Value	0x0000
Range	0x0000...0xFFFF
Size	2 byte
Stored in NV RAM	No
Access	R

Application Parameters (Parameters #200-#299)

See also...

- “PROFINET Mapping Object (Class A9h)” on page 143
- “Application Parameters” on page 98

Fieldbus Interface

General

The physical interface requires an external isolation transformer with the following characteristics:

Turns Ratio (TX)	Turns Ratio (RX)	Common Mode Rejection (0 - 30Mhz)
1CT:1CT	1CT:1CT	-40db

The PCB is part of the physical layer in that the characteristics of the traces and materials control impedance, capacitance, coupling and voltage withstand. This means that the PCB layout is a key factor when it comes to reducing noise ingress and emissions.

To ensure proper operation, make sure to fulfil the following recommendations:

- Earth ground planes and power planes must be properly defined and isolated
- Traces between connector and transformer must be short and equal in length
- Traces between transformer and application interface must be short and equal in length
- Traces must match the circuit impedance (100 Ohms) using micro strip layout technique

Application Connector Signals

Pin	Pin name ^a	Signal
13	FB1	TX+
14	FB2	TX-
15	FB3	RX+
16	FB4	RX-
19	FB5	CT (RX)
20	FB6	CT (TX)

a. FB7 and FB8 should be left unconnected.

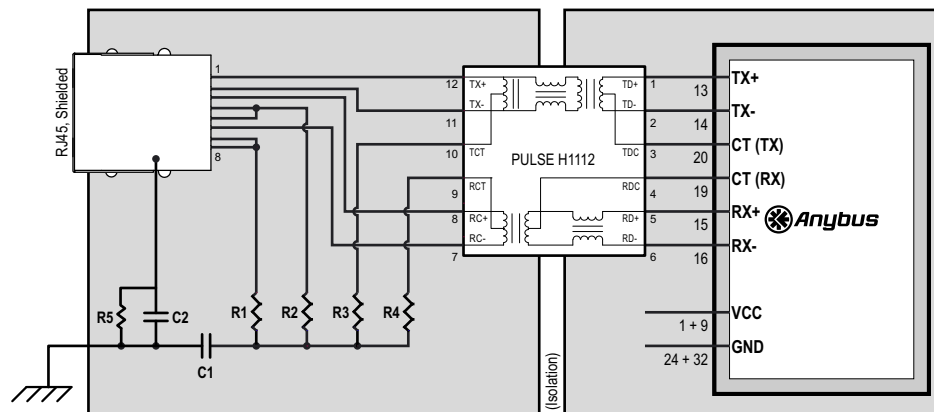
RJ45 Pinout

Ethernet Connector (RJ45)	
Pin	Signal
1	TX+
2	TX-
3	RX+
4	-
5	-
6	RX-
7	-
8	-
Housing	Cable Shield



Typical Implementation

The following example uses the PULSE H1112. Other isolation transformers which conform to the characteristics described earlier may also be used, but may require a slightly different circuit connection than the one illustrated below.

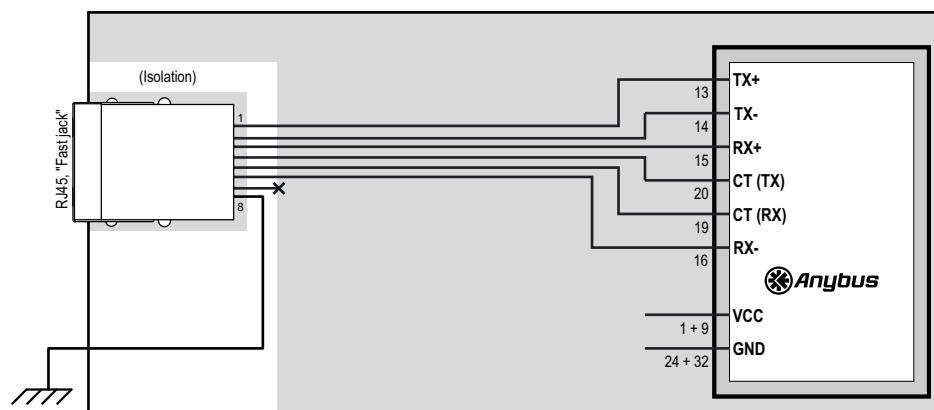


Ref.	Component	Comments
R1, R2, R3, R4	75 ohm	Network termination
R5	1M ohm	Filter to PE
C1, C2	1nF/2kV	Filter capacitors to PE
PULSE H1112	PULSE H1112	-
RJ45, Shielded	RJ45 connector	-

FastJack Connectors

FastJack connectors feature built-in isolation transformers and cable shield filters, which makes them suitable for applications where limited space is an issue.

To comply with modern EMC directives, this type of connector requires the use of shielded cables.



Ref.	Component	Comments
RJ45, Fast Jack	HFJ11-2450E	FastJack (HALO electronics Inc.)

Application Parameters

General Information

An application parameter is an application specific parameter created by the application during startup. Application parameters works just like normal parameters and can be accessed via the MIF and SCI interfaces. In addition, application parameters can be accessed from the network by mapping them to PROFINET IO services.

See also...

- “PROFINET Mapping Object (Class A9h)” on page 143
- “Creating an Application Parameter” on page 99
- “Mapping an Application Parameter to PROFINET” on page 103
- “HMS Object Implementation” on page 105

Creating an Application Parameter

Query - “Application Parameter Object”

To create a new application parameter, send the following message using Modbus Object Messaging. (Consult the Anybus-IC Design Guide for more information about the Object Message Subfield)

Object Message Subfield

Fragment byte count	Fragment protocol	Class ID	Instance ID	Service Code	Attribute	Data Field
(size)	02h	0085h	0000h	0005h	0000h	(See below)

Parameter Size (WORD)	Descriptor (DWORD)	Parameter Info (Size varies)	Extension Word (Optional, WORD)
-----------------------	--------------------	------------------------------	---------------------------------

Parameter Size

This value depends on the type of data specified in the descriptor (see below).

Data Type	Valid Parameter Size Values
UINT, INT, BITSTRING	1, 2, 4
FLOAT	4
STRING	1... 32 (String length including NULL-termination)
BYTE_ARRAY	1... 1024

Descriptor¹

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	
(reserved)					Data Format		Data Type			(reserved)					Write	Read

Write	Meaning
0	Write access not allowed
1	Write access allowed

Read	Meaning
0	Read access not allowed
1	Read access allowed

Data Type	Meaning
0 (0000)	UINT
1 (0001)	INT
2 (0010)	BITSTRING
3 (0011)	STRING
4 (0100)	FLOAT
5 (0101)	BYTE ARRAY

Data Format	Meaning
0 (00)	Dec
1 (01)	Hex
2 (10)	Bin
3 (11)	Dotted decimal

1. Note that the upper 16 bits of the descriptor are reserved for future use and must be set to 0000h.

Parameter Info

The size and contents of this field depends on the data type specified in the descriptor block.

- **Data types UINT, INT, BITSTRING & FLOAT**

Min Value (size varies)	Max Value (size varies)	Init Value (size varies)	Name ^a (String, 16 bytes)	Unit ^a (String, 16 bytes)
----------------------------	----------------------------	-----------------------------	---	---

Field	Type / Size	Description
Min. Value	Specified in 'Parameter Size'	Minimum allowed parameter value
Max. Value	Specified in 'Parameter Size'	Maximal allowed parameter value
Init Value	Specified in 'Parameter Size'	Initial parameter value
Name	String (16 byte, null terminated)	Name of parameter, e.g "Speed" ^a
Unit	String (16 byte, null terminated)	Unit, e.g "RPM" ^a

a. These fields are optional. (However, if used, both fields must be present)

- **Data type STRING**

Init Value (size varies)	Name ^a (STRING, 16 bytes)	Unit ^a (STRING, 16 bytes)
-----------------------------	---	---

Field	Type / Size	Description
Init Value	Specified in 'Parameter Size'	Initial value
Name	String (16 byte, null terminated)	Name of parameter ^a
Unit	String (16 byte, null terminated)	Unit ^a

a. These fields are optional. (However, if used, both fields must be present)

- **Data type BYTE_ARRAY**

Min. Value (BYTE)	Max. Value (BYTE)	Init Value (BYTE)	Name ^a (String, 16 bytes)	Unit ^a (String, 16 bytes)
----------------------	----------------------	----------------------	---	---

Field	Type / Size	Description
Min. Value	Byte	Min. allowed value of each element in the array
Max. Value	Byte	Max. allowed value of each element in the array
Init Value	Byte	Initial value of all elements in the array
Name	String (16 byte, null terminated)	Name of parameter ^a
Unit	String (16 byte, null terminated)	Unit ^a

a. These fields are optional. (However, if used, both fields must be present)

Extension Word (Optional)

This word is optional and specifies whether the response message should contain the Modbus address of the created application parameter or not.

Value	Description
0x0000	-
0x0001	Request Modbus Address
Other values	(Reserved for future use)

Response - “Application Parameter Object”

The Anybus-IC module will respond with the following message. (Consult the Anybus-IC Design Guide for more information about the Object Message Subfield)

Object Message Subfield

Fragment byte count	Fragment protocol	Class ID	Instance ID	Service Code	Error Code	Data Field
(size)	02h	0085h	0000h	0006h	0000h	(See below)

HOS Instance (WORD)	Parameter Number (WORD)	Modbus Address ^a (WORD)
------------------------	----------------------------	---------------------------------------

a. This field is only present if the extension word of the query is set to 0001h

HOS Instance

If the error code is 0 (success), this field contains the HOS Instance of the created application parameter.

Parameter Number

If the error code is 0 (success), this field contains the parameter number of the created application parameter.

Modbus Address

If the error code is 0 (success), this field contains the Modbus address of the created application parameter.

Note: This field is only present if the extension word of the query is set to 0001h.

Example

The example below creates an application parameter with the following properties:

- Parameter Name “Speed”, unit “rpm”
- Type 16 bit unsigned INT, range 0 - 65535, initial parameter value 32768
- R/W access

Query			
	01h	5Bh	
	37h	02h	
Class	0085h		Application Parameter Class
Instance	0000h		
Service Code	0005h		Create
Attribute	0000h		
Parameter Size	0002h		Parameter Size = 2 bytes
Descriptor MSW	0000h		
Descriptor LSW	0003h		UINT, DEC, R/W
Min value	0000h		Minimum allowed value: 0
Max value	FFFFh		Maximum allowed value: 65535
Init value	8000h		Initial value: 8000h
Name	53h ('S')	70h ('p')	"Speed"
	65h ('e')	65h ('e')	
	64h ('d')	00h	
	-	-	
	-	-	
	-	-	
	-	-	
	-	-	
Unit	72h ('r')	70h ('p')	"rpm"
	6Dh ('m')	00h	
	-	-	
	-	-	
	-	-	
	-	-	
	-	-	
	-	-	
Extension Word	0001h		Request Modbus Address
	CRC		

Response			
	01h	5Bh	
	0Fh	02h	
Class	0085h		Application Parameter Object Class
Instance	0000h		
Service Code	0006h		Create Response
Error Code	0000h		Success
HOS Instance	0001h		HOS Instance 1
Parameter no.	00C8h		Parameter no. = 200
Modbus Address	8000h		Modbus Address = 8000h
	CRC		

Mapping an Application Parameter to PROFINET

Acyclic data on the Anybus-IC module is exchanged by means of application parameters mapped to acyclic PROFINET services.

The mapping procedure consists of two steps:

- **Creating the Application Parameter**
(See “Creating an Application Parameter” on page 99)
- **Mapping the created Application Parameter to PROFINET**

This is done by creating a new instance in the Anybus-IC PROFINET Mapping Object Class (A9h). This class is used to map a PROFINET ‘Record Read’ / ‘Record Write’ onto an Anybus-IC Object Attribute.

Query - “PROFINET Mapping Object”

(Consult the Anybus-IC Design Guide for more information about the Object Message Subfield)

Object Message Subfield

Fragment byte count	Fragment protocol	Class ID	Instance ID	Service Code	Data Field
(size)	02h	00A9h	0000h	0005h	(See below)

Instance (WORD)	Slot number (WORD)	Subslot number (WORD)	Index (WORD)	Initial Record Data (BYTE)
--------------------	-----------------------	--------------------------	-----------------	-------------------------------

Slot number

PROFINET slot number to map

Subslot number

PROFINET subslot number to map

Index

PROFINET index number to map

Initial Record Data

Flag to tell whether or not the map shall be considered as initial record data or not

Values: 0 (FALSE) or 1 (TRUE)

See “Initial Parameters” on page 31 for more information about initial record data

Instance

HOS Instance to map

(In this case, use the HOS Instance value returned from the application parameter object request when the application parameter was created.)

Response - “PROFINET Mapping Object”

The response contains no additional data. (Consult the Anybus-IC Design Guide for more information about the Object Message Subfield)

Object Message Subfield

Fragment byte count	Fragment protocol	Class ID	Instance ID	Service Code	Error Code
(8 bits)	02h	00A9h	0000h	0006h	(16 bits)

Example

This example will map the application parameter created earlier in this chapter to PROFINET slot 7, subslot 1, index 100.

Query		
	01h	5Bh
	12h	02h
Class	00A9h	
Instance	0000h	
Service Code	0005h	
Attribute	0000h	
Instance	0001h	
Slot number	0007h	
Subslot number	0001h	
Index	0064h	
Initial Record Data	01h	00h (stuff byte)
	CRC	
	PROFINET Mapping Object	
	Create	
	HOS Instance 1	
	PROFINET Slot number = 7	
	PROFINET Subslot number = 1	
	PROFINET Index = 100	
	Initial Record Data = 1 (TRUE)	

Response		
	01h	5Bh
	09h	02h
Class	00A9h	
Instance	0000h	
Service Code	0006h	
Error Code	0000h	
	CRC	
	Create Response	
	Success	

HMS Object Implementation

General Information

The objects described in this chapter can be accessed using the Modbus Object Messaging protocol, and provides access to advanced fieldbus-specific functionality.

The module features the following HOS objects:

- “Application Parameter Object (Class 85h)” on page 106
- “File System Object (Class 86h)” on page 108
- “Fieldbus Object (Class A0h)” on page 113
- “Socket Object (Class A6h)” on page 124
- “I&M Object (Class A7h)” on page 131
- “PROFINET Diagnostic Object (Class A8h)” on page 134
- “PROFINET Mapping Object (Class A9h)” on page 143

For in-depth information about the HMS Object Software (HOS) and the Object Messaging protocol, consult the general Anybus-IC Design Guide.

Application Parameter Object (Class 85h)

General Information

Object Description

This object manages application parameters.

See also...

- “Application Parameters” on page 98
- “Creating an Application Parameter” on page 99
- “PROFINET Mapping Object (Class A9h)” on page 143

Supported Services

Class: Get_Attribute
 Create(see “Create (Class Service)” on page 107)

Instance: -

Class Attributes

#	Name	Access	Type	Value
1	Revision	Get	Byte	01h
2	No. of Instances	Get	Word	Current no. of instances in class.

Instance Attributes

#	Name	Access	Type	Value
1	(varies)	(varies)	(varies)	(specified on 'create')
2	Parameter Size	Get	Word	
3	Descriptor	Get	DWord	

Create (Class Service)

Service Description

This service creates a new application parameter instance.

Service Request (05h)

#	Contents	Type	Comments
1	Parameter Size	Word	-
2	Descriptor	DWord	Determines access rights, types etc.
3	Min. Value	(varies)	Type depends on descriptor
4	Max. Value		
5	Initial Value		
6	Name	String[16]	(optional)
7	Unit	String[16]	(optional)

Service Response (06h)

#	Contents	Type	Comments
1	Instance Number	Word	HOS instance number
2	Parameter Number	Word	Parameter no. of the created parameter

File System Object (Class 86h)

General Information

Object Description

The File System Object class can create and delete file system instances dynamically during runtime. Each instance is a unique handle to a file stream and contains services for file system operations.

See also...

- “Filesystem” on page 18

Supported Services

Class:	Get_Attribute File Open(see “File Open (Class Service)” on page 109) File Close(see “File Close (Class Service)” on page 109) File Delete(see “File Delete (Class Service)” on page 110) File Copy(see “File Copy (Class Service)” on page 110) File Move(see “File Move (Class Service)” on page 111) File Rename(see “File Rename (Class Service)” on page 111)
Instance:	File Read(see “File Read (Instance Service)” on page 112) File Write(see “File Write (Instance Service)” on page 112)

Class Attributes

#	Name	Access	Type	Value
1	Revision	Get	Byte	01h
2	No. of Instances	Get	Word	Current no. of instances in class.
3	Max. Instance	Get	Word	Max. no. of allowed instances; equals the max. number of files that can be open simultaneously.

Instance Attributes

-

File Open (Class Service)

Service Description

Creates a file instance and opens a file for reading, writing, or appending.

Service Request (80h)

#	Contents	Type	Comments
1	Mode	Byte	00h: Read mode 01h: Write mode 02h: Append mode
2	Filename	Byte[x]	Path + filename + NULL termination

- **Read mode**
Opens a file for read-only access
- **Write mode**
Opens a file for write-only access; if the specified file does not exist, it will be created. If the specified file already exists, it will be overwritten.
- **Read mode**
Opens a file for writing at end-of-file; if the specified file does not exist, it will be created. If the specified file already exists, data will be appended at end-of-file.

Service Response (81h)

#	Contents	Type	Comments
1	File instance number	Word	Used for all further operations on the file

File Close (Class Service)

Service Description

Closes a previously opened file and deletes the instance.

Service Request (82h)

#	Contents	Type	Comments
1	File instance number	Word	Acquired on 'File Open'

Service Response (83h)

#	Contents	Type	Comments
1	File size	DWord	Size of closed file

File Delete (Class Service)

Service Description

Deletes a file.

Service Request (84h)

#	Contents	Type	Comments
1	Filename	Byte[x]	Path + filename + NULL termination

Service Response (85h)

#	Contents	Type	Comments
-	-	-	(no data)

File Copy (Class Service)

Service Description

Creates a copy of a file.

Service Request (86h)

#	Contents	Type	Comments
1	Filename (source)	Byte[x]	Source path + filename + NULL termination
2	Filename (destination)	Byte[x]	Destination path + filename + NULL termination

Service Response (87h)

#	Contents	Type	Comments
-	-	-	(no data)

File Move (Class Service)

Service Description

Moves a file.

Service Request (88h)

#	Contents	Type	Comments
1	Filename	Byte[x]	Source path + filename + NULL termination
2	Destination	Byte[x]	Destination path + NULL termination

Service Response (89h)

#	Contents	Type	Comments
-	-	-	(no data)

File Rename (Class Service)

Service Description

Renames a file.

Service Request (8Ah)

#	Contents	Type	Comments
1	Old filename	Byte[x]	Old path + filename + NULL termination
2	New filename	Byte[x]	New path + filename + NULL termination

Service Response (8Bh)

#	Contents	Type	Comments
-	-	-	(no data)

File Read (Instance Service)

Service Description

Reads data from a file opened for reading.

Service Request (A0h)

#	Contents	Type	Comments
1	Length	Word	Number of bytes to read (up to 0400h)

Service Response (A1h)

#	Contents	Type	Comments
1	Length	Word	Number of bytes read
2	Data	Byte[x]	Actual data read from file

File Write (Instance Service)

Service Description

Writes data to a file opened for writing or appending.

Service Request (A2h)

#	Contents	Type	Comments
1	Length	Word	Number of bytes to write
2	Data	Byte[x]	Data that shall be written to the file

Service Response (A3h)

#	Contents	Type	Comments
1	Length	Word	Actual number of bytes written

Fieldbus Object (Class A0h)

General Information

Object Description

The Fieldbus Object class holds all network related configuration. Also, if the application wants to create a tailor-made configuration this object is used.

See also...

- “PROFINET I/O Implementation” on page 23
- “I&M Object (Class A7h)” on page 131
- “PROFINET Diagnostic Object (Class A8h)” on page 134
- “PROFINET Mapping Object (Class A9h)” on page 143

Supported Services

Class:	Get_Attribute
	Set_Attribute
	API Add(see “API Add (Class Service)” on page 117)
	Plug Module(see “Plug Module (Class Service)” on page 118)
	Plug Sub-module(see “Plug Submodule (Class Service)” on page 119)
	Pull Module(see “Pull Module (Class Service)” on page 120)
	Pull Sub-module(see “Pull Submodule (Class Service)” on page 121)
	AR Abort(see “AR Abort (Class Service)” on page 122)
Instance:	-

Class Attributes

#	Name	Access	Type	Parameter number	Comments
1	Class revision	R	Byte	N/A	Revision number of the class
2	Number of Instances	R	Word	N/A	Number of instances in the class
3	FB status	R	Word	100	Current network status
4	FB init	W	Word	101	Action in response to FB status
5	MAC address	R	Byte[6]	116	Module MAC address
6	IP address cfg	RW	Byte[4]	103	Configured IP address
7	(reserved)	-	-	-	-
8	IP address act	R	Byte[4]	105	Currently used IP address
9	Byte order	RW	Byte	40	The byte order in data area
10	In data size	R	Word	43	Size of in-data
11	Out data size cfg	RW	Word	41	Configured out data size
12	Out data size act	R	Word	42	Actual used out data size
13	In IO Object instance	RW	Word	N/A	Instance number for in-data in the IO object
14	Out IO Object instance	R	Word	N/A	Instance number for out-data in the IO object
15	(reserved)	-	-	-	-
16	FB password	W	Word	102	Protected parameter password
17	(reserved)	-	-	-	(Data rate cfg)
18	Data rate act	R	Byte	118	Current Ethernet communication speed
19	(reserved)	-	-	-	(Duplex cfg)
20	Duplex act	R	Byte	120	Current Ethernet communication duplex mode
21	Subnet mask cfg	RW	Byte[4]	106	Configured Subnet mask
22	Subnet mask act	R	Byte[4]	107	Currently used Subnet mask
23	GW address cfg	RW	Byte[4]	108	Configured Gateway address
24	GW address act	R	Byte[4]	109	Currently used Gateway address
25	DHCP enable cfg	RW	Byte	114	Configured DHCP enable status
26	DHCP enable act	R	Byte	115	Currently used DHCP enable status
27	WEB srv enable	RW	Byte	121	Web Server enable state
28	FTP srv enable	RW	Byte	122	FTP Server enable state
29	Telnet enable	RW	Byte	123	Telnet Server enable state
30	Admin mode cfg	RW	Byte	124	Configured Global admin mode state
31	Admin mode act	R	Byte	125	Actual Global admin mode state
32	SMTP srv address	RW	Byte[4]	126	E-mail server address state
33	Triggered emails	R	Word	127	Number of triggered emails
34	SMTP errors	R	Word	128	Number of SMTP errors
35	Send email	W	Word	129	Send a predefined e-mail
36	VFS enable	RW	Byte	130	VFS Enable state
37	RAM disc path	RW	Byte[16]	131	Where to mount the RAM disc, Null terminated
38	MB/TCP connection TO	RW	Word	132	Modbus/TCP connection timeout
39	MB/TCP enable	RW	Byte	133	Modbus/TCP enable state
40	In bit size	RW	Word	134	Size of input area that is modbus bit area
41	Out bit size	RW	Word	135	Size of output area that is modbus bit area
42	HICP enable	RW	Byte	136	HICP protocol enabled
43	HICP password	RW	Byte[30]	137	HICP Password
44	(reserved)	-	-	-	-
45	(reserved)	-	-	-	-
46	(reserved)	-	-	-	-
47	SMTP user name	RW	Byte[32]	142	SMTP user name
48	SMTP password	RW	Byte[32]	143	SMTP password

#	Name	Access	Type	Parameter number	Comments
49	Gleaning enable	RW	Byte	144	ARP Gleaning enable
50	PRT version numb	R(W)	Word	160	Version number which can be set by the host application. If set to 0xFFFF the default version of the firmware is used.
51	PRT serial numb	R(W)	Double Word	161	Serial number which can be set by the host application. If set to 0xFFFFFFFF the default serial number is used.
52	PRT vendor id	R(W)	Word	162	Identifies the manufacturer of the device. Will also be reflected in the I&M object. This ID is assigned by the PNO. Default value is 0x010C (HMS).
53	PRT device id	R(W)	Word	163	Identifies the device. Assigned by the manufacturer. Default value is 0x0008.
54	PRT station type	R(W)	Byte[30]	164	Characterizes the device. Assigned by the manufacturer. This information will be set also to the SNMP parameter "System Name". String shall be null terminated. Default value is "ABIC-PRT".
55	Port 1 MAC addr	R	Byte[6]	165	PROFINET IO port 1 MAC address. Default value is assigned during production.
56	System description	R(W)	Byte[256]	-	System description for the PROFINET IO interface. Null terminated. Default value is "HMS Industrial Networks Anybus-IC".
57	Interface description	R(W)	Byte[256]	-	Name of the Ethernet interface of the IO device. Null terminated. Default value is "PROFINET IO interface"
58	PRT station name	RW	Byte[240]	166	The station name of the IO device. Null terminated. Default value is "".
59	ArCheckInd Data	R	Byte[12]	167	Array of bytes containing the most recent information of the AR. See "ArCheckInd Data (Parameter #167)" on page 87 .
60	CfgMisMalInd Data	R	Byte[18]	168	Array of bytes containing the most recent information about a configuration mismatch. See "CfgMisMalIndData (Parameter #168)" on page 89 .
61	ArInfolnd Data	R	Byte[2048]	169	Array of bytes containing the most recent information of how the AR is built up. See "ArInfolnd Data (Parameter #169)" on page 90 .
62	EndOfPrmlnd Data	R	Byte[10]	170	Array of bytes containing the most recent information of an end of parameter indication. See "EndOfPrmlnd Data (Parameter #170)" on page 91
63	ArOfflnd Data	R	Byte[4]	171	Most recent offline indication data (including AR and Reason code). See "ArOfflnd Data (Parameter #171)" on page 92 .
64	ArAbortInd Data	R	Byte[4]	172	Most recent abort indication data (including AR and Reason code). See "ArAbortInd Data (Parameter #172)" on page 93
65	PlugSubModFailed	R	Byte[8]	173	Information about the submodule which failed to be plugged. See "PlugSubModFailed (Parameter #173)" on page 94 .
66	PniolnitErrCode	R	Word	174	If the PROFINET IO stack, for some reason, would not accept the current configuration, the returned error code can be read with this attribute.

Instance Attributes

-

API Add (Class Service)

Service Description

This service assigns an Application Process Instance (API) to the configuration. After having issued this service the application is responsible for setting up the configuration.

This service can only be issued while 'Module Mode' equals 'Start-up mode' (0x0000). If used, API 0 must always be added first.

Please note that if this service has been issued, commands 'Plug Module' and 'Plug Sub-module' must be used.

Service Request (0x80)

#	Type	Name	Description
1	DWord	API	Number of the API
2	Word	Max No. Slots	Maximum number of slots for this API Range: 1-33 Please note that the Device Access Point (DAP, slot 0) is included in this number
3	Word	Max No. Sub-slots	Maximum number of subslots per slot, for this API Range: 1-8 Please note that if interface- and port submodules are to be plugged in slot 0, these are included in this figure (that is, set to at least 3)

Service Response (0x81)

#	Type	Name	Description
1	Word	Fieldbus Error	See "Fieldbus Errors" on page 123

Plug Module (Class Service)

Service Description

This service plugs a module in a slot of an already added API.

This service can only be issued while 'Module Mode' equals 'Start-up mode' (0x0000).

Service 'API Add' must be used before using this service.

Service Request (0x82)

#	Type	Name	Description
1	DWord	API	Number of the API
2	Word	Slot number	Number of the slot to which the module should be plugged
3	DWord	Module Identification number	Module identification number, as stated in the GSD-file.

Service Response (0x83)

#	Type	Name	Description
1	Word	Fieldbus Error	See "Fieldbus Errors" on page 123

Plug Submodule (Class Service)

Service Description

This service plugs a submodule in a subslot in a slot of an already added API.

This service can only be issued while 'Module Mode' equals 'Start-up mode' (0x0000).

Services 'API Add' and 'Plug Module' must be used before this service can be used.

Service Request (0x84)

#	Type	Name	Description
1	DWord	API	Number of the API
2	Word	Slot number	Number of the slot to which the module should be plugged
3	Word	Sub-slot number	Number of the subslot to which the submodule should be plugged
4	DWord	Sub-module Identification number	Submodule identification number, as stated in the GSD-file.
5	Byte	Nw IO direction	Indicates the direction of the IO data associated with this submodule: 0 - No IO data associated with this submodule 1 - In IO data associated with this submodule 2 - Out IO data associated with this submodule 3 - Both In and Out IO data associated with this module
6	Byte	Nw In IO offset	In IO offset of total IO area
7	Byte	Nw In IO length	Length of In IO data used for this submodule
8	Byte	Nw Out IO offset	Out IO offset of total IO area
9	Byte	Nw Out IO length	Length of Out IO data used for this submodule

Service Response (0x85)

#	Type	Name	Description
1	Word	Fieldbus Error	See "Fieldbus Errors" on page 123

Note:

- 1). When this service is received the Anybus module carries out only limited feasibility checks of the parameters 'Nw In IO offset' / 'Nw In IO length' and 'Nw Out IO offset' / 'Nw Out IO length'. Instead, the Anybus module carries out these checks once the Anybus module is initialized. If the Anybus module at that point finds any inconsistencies in the setup the IOPS / IOCS related to these submodules will be set to state 'BAD', signaling to the IO Controller that the data is not valid.
- 2). It is highly recommended that submodules 0x8000 (interface submodule) and 0x8001 (Port 1 submodule) are plugged into slot 0. If this is not implemented it is likely that conformance tests will not be passed successfully.
- 3). If this service is issued after the Anybus module has been initialized, the response to this command will be generated immediately, before handing over this command to the PROFINET IO stack. This means that, even if a successful response to this command has been received, a failure can occur. If this should happen the Anybus module will trigger the event 'PSI' in the 'FB status register' if 'Module Mode' parameter is set to 'Fieldbus specific init'. If not, the host application cannot detect that the command has failed. If the command does not fail, the completion of the command will be signaled by triggering a 'PRM' event in the 'FB status' parameter.

Pull Module (Class Service)

Service Description

This service pulls a module out of a slot of an already added API.

Services 'API Add' and 'Plug Module' must be used before this service can be used.

Service Request (0x86)

#	Type	Name	Description
1	DWord	API	Number of the API
2	Word	Slot number	Number of the slot to which the module should be pulled

Service Response (0x87)

#	Type	Name	Description
1	Word	Fieldbus Error	See "Fieldbus Errors" on page 123

Pull Submodule (Class Service)

Service Description

This service pulls a submodule out of a subslot in a slot of an already added API.

Services 'API Add', 'Plug Module' and 'Plug Submodule' must be used before this service can be used.

Service Request (0x88)

#	Type	Name	Description
1	DWord	API	Number of the API
2	Word	Slot number	Number of the slot
3	Word	Sub-slot number	Number of the subslot to which the submodule should be pulled

Service Response (0x89)

#	Type	Name	Description
1	Word	Fieldbus Error	See "Fieldbus Errors" on page 123

AR Abort (Class Service)

Service Description

This service aborts the specified application relationship (AR).

Service Request (0x8A)

#	Type	Name	Description
1	Word	AR handle	AR handle of the application relationship to abort

Service Response (0x8B)

#	Type	Name	Description
1	Word	Fieldbus Error	See "Fieldbus Errors" on page 123

Note: The IO controller might try to reestablish the IO connection again when noticing it has been closed.

Fieldbus Errors

Fieldbus Error	Name
0	No error
1	The AR handle provided is not valid
2	API 0 must be added first
3	API does not exist
4	API is already present
5	There is no room for any more APIs
6	Module in slot 0 cannot have any I/O data
7	Before plugging the requested module / submodule, slot 0 must be properly plugged with a module and a submodule (in subslot 1)
8	Slot occupied
9	Subslot occupied
10	No module inserted in the specified slot
11	No submodule inserted in the specified subslot
12	Slot number is out-of-range
13	Subslot number is out-of-range
14	The maximum number of submodules have already been plugged
15	Invalid state
16	Addressing error, In I/O parameter(s)
17	Addressing error, Out I/O parameter(s)
18	Unknown error

Socket Object (Class A6h)

General Information

Object Description

The Socket Object Class can create and delete socket instances dynamically during runtime. Each socket instance contains services to establish communication and communicate over TCP or UDP channels.

Supported Services

Class:	Get_Attribute Create Socket(see “Create Socket (Class Service)” on page 125) Close Socket(see “Close Socket (Class Service)” on page 125)
Instance:	Bind(see “Bind (Instance Service)” on page 125) Listen(see “Listen (Instance Service)” on page 126) Accept(see “Accept (Instance Service)” on page 127) Connect(see “Connect (Instance Service)” on page 127) Receive(see “Receive (Instance Service)” on page 127) Receive From(see “Receive From (Instance Service)” on page 128) Send(see “Send (Instance Service)” on page 128) Send to(see “Send To (Instance Service)” on page 129)

Class Attributes

#	Type	Access	Name	Req	Description
1	Byte	R	Class revision	R	Revision number of the class.
2	Word	R	Number of Instances	R	Number of instances in the class.
3	Word	R	Max Instances	R	Maximal allowed number of instances.

Instance Attributes

-

Create Socket (Class Service)

Service Description

This service creates a socket.

Service Request (0x80)

#	Type	Name	Description
1	Byte	Socket type	1: SOCK_STREAM (TCP socket) 2: SOCK_DGRAM (UDP socket)

Service Response (0x81)

#	Type	Name	Description
1	Word	Socket Error	See "Socket Errors" on page 130
2	Word	Instance Number	The number of the created socket instance

Close Socket (Class Service)

Service Description

This service causes a connected socket to shut down and deletes the socket instance.

Service Request (0x82)

#	Type	Name	Description
1	Word	Instance	Socket Instance number to close

Service Response (0x83)

#	Type	Name	Description
1	Word	Socket Error	See "Socket Errors" on page 130

Bind (Instance Service)

Service Description

Binds a socket to a local port. Port 0 = any free port.

Service Request (0x84)

#	Type	Name	Description
1	Word	Port	The port to bind the socket to

Service Response (0x85)

#	Type	Name	Description
1	Word	Socket Error	See "Socket Errors" on page 130
2	Word	Port	Local port number

Listen (Instance Service)

Service Description

Sets a socket to listening state.

Service Request (0x86)

#	Type	Name	Description	
1	Byte	Backlog	Backlog for incoming connections:	
			Backlog	Queue length
			0	1
			1	2
			2	4
			3	5
			4	7
			5	8

Service Response (0x87)

#	Type	Name	Description
1	Word	Socket Error	See "Socket Errors" on page 130

Accept (Instance Service)

Service Description

Accepts connections on a socket in listening state. A new socket-instance is created for each accepted connection. The new socket is connected with the host and the listen response returns its instance number.

Service Request (0x88)

#	Type	Name	Description
-	-	-	-

Accept Response (0x89)

#	Type	Name	Description
1	Word	Socket Error	See "Socket Errors" on page 130
2	Word	Instance Number	The number of the new instance for the connected socket.
3	DWord	IP address	Host IP address
4	Word	Port	Host port number

Connect (Instance Service)

Service Description

If the socket type is SOCK_DGRAM (UDP) this service specifies the peer with which the socket is to be associated. This is to which datagrams are sent and the only address from which datagrams are received.

If the socket type is SOCK_STREAM (TCP) this service attempts to establish a connection to another socket.

Stream sockets may connect successfully only once while datagram sockets may use connect multiple times to change their association. Datagram sockets may dissolve the association by connection to the invalid address: IP=0.0.0.0, Port=0.

Service Request (0x8A)

#	Type	Name	Description
1	Dword	IP address	IP address to connect to
2	Word	Port	Port number to connect to

Service Response (0x8B)

#	Type	Name	Description
1	Word	Socket Error	See "Socket Errors" on page 130

Receive (Instance Service)

Service Description

This service receives data from a connected socket.

Service Request (0x8C)

#	Type	Name	Description
1	Word	Length	The number of bytes to receive. Maximum value is 1460 bytes

Service Response (0x8D)

#	Type	Name	Description
1	Word	Socket Error	See "Socket Errors" on page 130
2	Word	Length	Bytes received
3	Byte[x]	Data	Received data

Receive From (Instance Service)**Service Description**

This service receives the next received datagram from an unconnected socket.

Service Request (0x8E)

#	Type	Name	Description
1	Word	Length	The number of bytes to receive. Maximum value is 1460 bytes

Service Response (0x8F)

#	Type	Name	Description
1	Word	Socket Error	See "Socket Errors" on page 130
2	Dword	IP address	Host IP address
3	Word	Port	Host port number
4	Word	Length	Bytes received
5	Byte[x]	Data	Received data

Send (Instance Service)**Service Description**

This service sends data on a connected socket.

Service Request (0x90)

#	Type	Name	Description
1	Word	Length	Number of bytes to send. Maximum value is 1460 bytes
2	Byte[x]	Data	Data to send

Service Response (0x91)

#	Type	Name	Description
1	Word	Socket Error	See "Socket Errors" on page 130
2	Word	Length	Number of bytes sent

Send To (Instance Service)

Service Description

This service sends data on an unconnected socket.

Service Request (0x92)

#	Type	Name	Description
1	Dword	IP address	Host IP address
2	Word	Port	Host port number
3	Word	Length	Number of bytes to send. Maximum value is 1460 bytes
4	Byte[x]	Data	Data to send

Service Response (0x93)

#	Type	Name	Description
1	Word	Socket Error	See "Socket Errors" on page 130
2	Word	Length	Number of bytes sent

Socket Errors

Socket Error	Name
0	NOERROR
1	ENOBUFS
2	ETIMEDOUT
3	EISCONN
4	EOPNOTSUPP
5	ECONNABORTED
6	EWouldBlock
7	ECONNREFUSED
8	ECONNRESET
9	ENOTCONN
10	EALREADY
11	EINVAL
12	EMSGSIZE
13	EPIPE
14	EDESTADDRREQ
15	ESHUTDOWN
16	ENOPROTOOPT
17	EHAVEOOB
18	ENOMEM
19	EADDRNOTAVAIL
20	EADDRINUSE
21	EAFNOSUPPORT
22	EINPROGRESS
23	ELOWER

I&M Object (Class A7h)

General Information

Object Description

The I&M Object Class can be used to customize the Identification & Maintenance data which can be read via the PROFINET network.

Note: I&M object attributes have to be set during initialization. They can not be set once the Anybus module has been initialized.

Supported Services

Class:	Get_Attribute Set_Attribute
--------	--------------------------------

Instance:	-
-----------	---

Class Attributes

#	Type	Access	Name	Default value	Description
1	Byte	R	Class revision	1	Revision number of the class.
2	Word	R	Number of Instances	0	Number of instances in the class.
3	Word	R	Manufacturer Id	0x010C	Manufacturer Id. Same value as 'Vendor Id' in the Fieldbus Object Class.
4	20 Char	RW	Order Id	'ABIC-PRT'	Order number of the product. If the length is shorter than 20 characters the remaining characters shall be set to blank (' ' / 0x20).
5	16 Char	RW	Serial Number	-	Serial number of the product. If the serial number contains less than 16 characters, the remaining characters shall be set to blank (' ' / 0x20).
6	Word	RW	Hardware revision	0x0001	Hardware revision of the product. The number is a running number and can only be used to indicate a "major" version number, such as 1, or 2 (V1 or V2). This number cannot be used to represent for example V2.34.
7	4 Byte	RW	Software revision	'V.X.Y.Z'	Software version number. Byte 1 coding: 'V' - Official released version 'R' - Revision 'P' - Prototype 'U' - Under test (field test) 'T' - Test device Byte 2, Major version number (Functional enhancement) Byte 3, Minor version (Bug fix) Byte 4, Internal change (No impact on function) Version "V255.255.255" indicates presence of profile specific information.
8	Word	RW	Revision counter	0x0000	A changed value indicates a change of the hardware or its parameters. For more information see "Guideline, Identification & Maintenance functions".
9	Word	RW	Profile Id	0xF600	Profile Id
10	Word	RW	Profile Specific Type	0x0004	Profile Specific Type
11	Word	R	IM Version	0x0101	I&M Version (only 1.01 is supported)
12	Word	RW	IM Supported	0x001E	I&M supported: Bit 0, Profile specific Bit 1, I&M 1 enabled Bit 2, I&M 2 enabled Bit 3, I&M 3 enabled Bit 4, I&M 4 enabled Bit 5-15, reserved Note: As of version 2.3 of the PROFINET specification, it is mandatory to support I&M 0-4. It is not recommended to change the default value of 0x001E.

#	Type	Access	Name	Default value	Description
13	32 Char	RW	Tag Function	32 blanks	Tag function. If the tag function contains less than 32 characters, the remaining characters shall be set to blank (' ' / 0x20).
14	22 Char	RW	Tag Location	22 blanks	Tag location. If the tag location contains less than 22 characters, the remaining characters shall be set to blank (' ' / 0x20).
15	16 Char	RW	Installation date	16 blanks	Installation date. If the tag function contains less than 16 characters, the remaining characters shall be set to blank (' ' / 0x20).
16	54 Char	RW	Descriptor	54 blanks	Descriptor. If the descriptor contains less than 54 characters, the remaining characters shall be set to blank (' ' / 0x20).
17	54 Byte	RW	Signature	54 zeroes	Signature.

Attributes 4-10 and 12 are not stored in nonvolatile memory. If the application would like to change the default values these attributes must be set before initialization on every start-up.

Attributes 13-17 are stored in nonvolatile memory. These are normally set from the network side. Setting them from the application side may cause certification problems.

Instance Attributes

-

PROFINET Diagnostic Object (Class A8h)

General Information

Object Description

The PROFINET Diagnostic Object Class enables the application to create application specific diagnostic entries and alarms.

Supported Services

Class: Get_Attribute
 Send Process Alarm
 (see “Send Process Alarm (Class Service)” on page 136)
 Create Channel diagnostic alarm
 (see “Create Channel Diagnostic Alarm (Class Service)” on page 137)
 Create Generic diagnostic alarm
 (see “Create Generic Diagnostic Alarm (Class Service)” on page 139)

Instance: Remove (see “Application Example, Send Generic Diagnostics Alarm Failure” on page 140)

Class Attributes

#	Type	Access	Name	Req	Description
1	Byte	R	Class revision	R	Revision number of the class (0x0001).
2	Word	R	Number of Instances	R	Number of instances in the class.
3	Word	R	Max Instances	R	Maximum allowed number of instances (0x000A).

Instance Attributes

#	Type	Access	Name	Req	Description
1	Byte	R	Type	R	Type of instance. 0 - Channel Diagnostics 1 - Generic Diagnostics 2-255 - Reserved

PROFINET IO Specific Coding of Alarm Specific Fields

Channel Properties

The channel properties of a diagnostic entry / alarm is a bit-field containing several different fields. Each subfield is explained below.

Channel Properties - Type (ChannelPropType)

Value	Enumeration	Description
0	Other	Data type other than the ones specified with this table
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

Channel Properties - Specifier (ChannelPropSpec)

Value	Enumeration	Description
1	APP	Error appears
2	DISAPP	Error disappears
3	DISAPP_MORE	Error disappears, but there are more errors

Channel Properties - Direction (ChannelPropDir)

Value	Enumeration	Description
0	Manufacturer specific	Channel data direction is manufacturer specific / Unknown
1	Input	Channel data direction is of input type (direction to the IO Controller)
2	Output	Channel data direction is of output type (direction to the IO Device)
3	Input/Output (Bidirectional)	Channel data direction is of bidirectional type

Channel Properties - Accumulative (ChannelPropAcc)

This parameter indicates if the diagnosis is related to a certain channel, or to a group of channels.

Channel Properties - Maintenance Required (ChannelPropMaintReq)

This parameter indicates that a user activity for the channel is recommended in order to keep the application running.

Channel Properties - Maintenance Demanded (ChannelPropMaintDem)

This parameter indicates that a user activity for the channel is highly recommended in order to keep the application running.

Channel Error Type (ChannelErrType)

Value	Enumeration	Description
0x0001	Short circuit	-
0x0002	Under voltage	-
0x0003	Over voltage	-
0x0004	Overload	-
0x0005	Over temperature	-
0x0006	Line break	-
0x0007	Upper limit value exceeded	-
0x0008	Lower limit value exceeded	-
0x0009	Error	-
0x000A ... 0xFFFF	-	Refer to PROFINET IO specification

User Structure identifier (UserStructIdent)

Value	Enumeration	Description
0x0000... 0x7FFF	Manufacturer specific data	The data passed is manufacturer specific
0x8000	Channel diagnostic data	The data passed is coded according to diagnostic data specification This is not applicable for Process- and Generic alarms
0x8001	Multiple manufacturer specific data	The data passed is manufacturer specific This is not applicable for Process- and Generic alarms

Send Process Alarm (Class Service)**Service Description**

This service sends a process alarm to the IO controller. This service will not create a new instance in the PROFINET Diagnostic Object. This service is not available unless an IO connection with an IO controller is established.

Service Request (0x80)

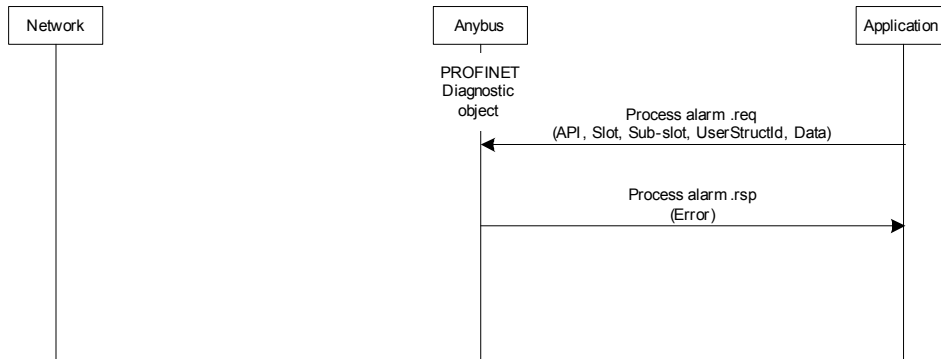
#	Type	Name	Description
1	DWord	API	Number of the API
2	Word	Slot	Slot number to which the alarm shall be sent
3	Word	Sub-slot	Subslot number to which the alarm shall be sent A process alarm cannot be sent to Subslot 0.
4	Word	UserStructIdent	User structure identifier. See "User Structure identifier (UserStructIdent)" on page 136
5	Byte[x]	Data	Data to send (max. 128 bytes)

Service Response (0x81)

#	Type	Name	Description
1	Word	Diagnostic Error	See "Diagnostic Errors" on page 142

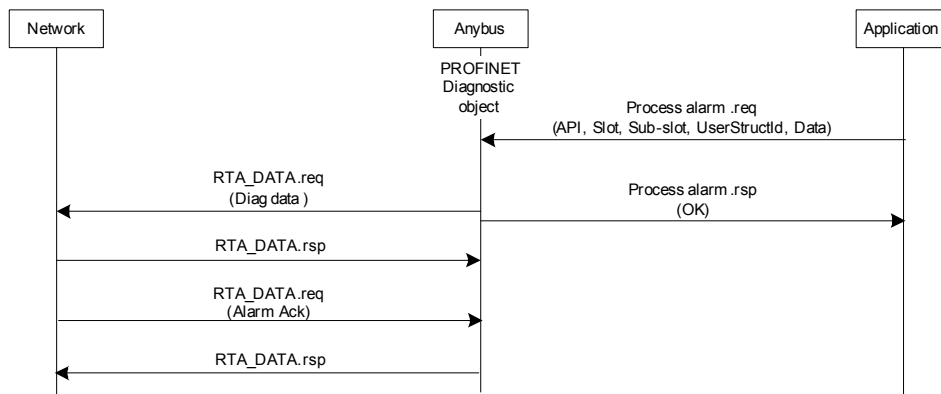
Application Example, Send Process Alarm Failure

In this example the host application tries to send a malformed request.



Application Example, Send Process Alarm

In this example the host application sends a process alarm to the IO controller.



Create Channel Diagnostic Alarm (Class Service)

Service Description

This service creates, and sends, a channel alarm to the IO controller. This service will create a new instance in the PROFINET Diagnostic Object. This service is not available unless an IO connection with an IO controller is established.

Service Request (0x82)

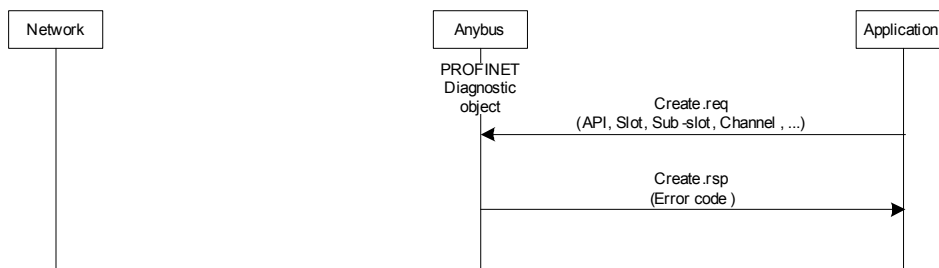
#	Type	Name	Description
1	DWord	API	Number of the API
2	Word	Slot	Slot number to which the alarm shall be sent
3	Word	Sub-slot	Subslot number to which the alarm shall be sent A process alarm cannot be sent to subslot 0
4	Word	Channel number	Channel number 0x0000 ... 0x7FFF - Manufacturer specific 0x8000 - Refers to the submodule itself (not a specific channel)
5	Word	ChannelErrorType	See "Channel Error Type (ChannelErrType)" on page 136
6	Byte	ChannelPropType	See "Channel Properties" on page 135
7	Byte	ChannelPropDir	
8	Byte	ChannelPropAcc	
9	Byte	ChannelProp-MaintReq	Not currently supported, set to zero.
10	Byte	ChannelPropMaint-Dem	

Service Response (0x83)

#	Type	Name	Description
1	Word	Diagnostic Error	See "Diagnostic Errors" on page 142
2	Word	Instance	Created instance number

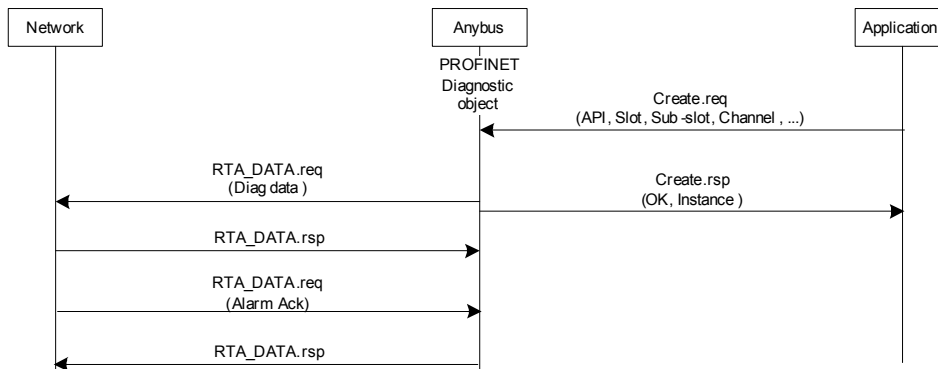
Application Example, Send Channel Diagnostic Alarm Failure

In this example the host application tries to send a malformed request.



Application Example, Send Channel Diagnostic Alarm

In this example the host application sends a channel diagnostic alarm to the IO controller.



Create Generic Diagnostic Alarm (Class Service)

Service Description

This service creates, and sends, a generic alarm to the IO controller. This service will create a new instance in the PROFINET Diagnostic Object. This service is not available unless an IO connection with an IO controller is established.

Service Request (0x84)

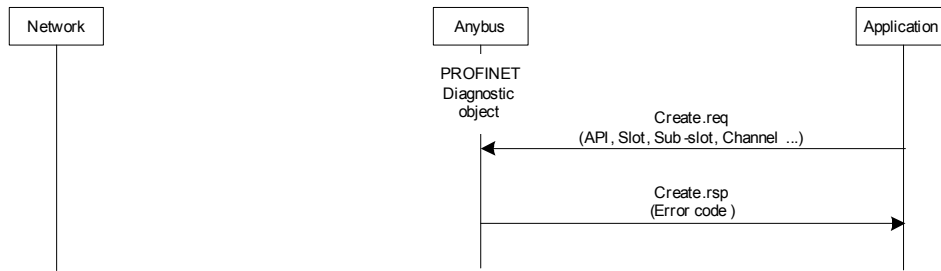
#	Type	Name	Description
1	DWord	API	Number of the API
2	Word	Slot	Slot number to which the alarm shall be sent
3	Word	Sub-slot	Subslot number to which the alarm shall be sent A process alarm cannot be sent to subslot 0.
4	Word	Channel number	Channel number 0x0000 ... 0x7FFF - Manufacturer specific 0x8000 - Refers to the submodule itself (not a specific channel)
5	Byte	ChannelPropType	See "Channel Properties" on page 135 0x00 shall be used if the 'Channel Number' equals 0x8000, or if none of the types specified are appropriate.
6	Byte	ChannelPropDir	See "Channel Properties" on page 135
7	Word	UserStructIdent	See "User Structure identifier (UserStructIdent)" on page 136
8	Byte[x]	Data	Data to send (max. 128 bytes)

Service Response (0x85)

#	Type	Name	Description
1	Word	Diagnostic Error	See "Diagnostic Errors" on page 142
2	Word	Instance	Created instance number

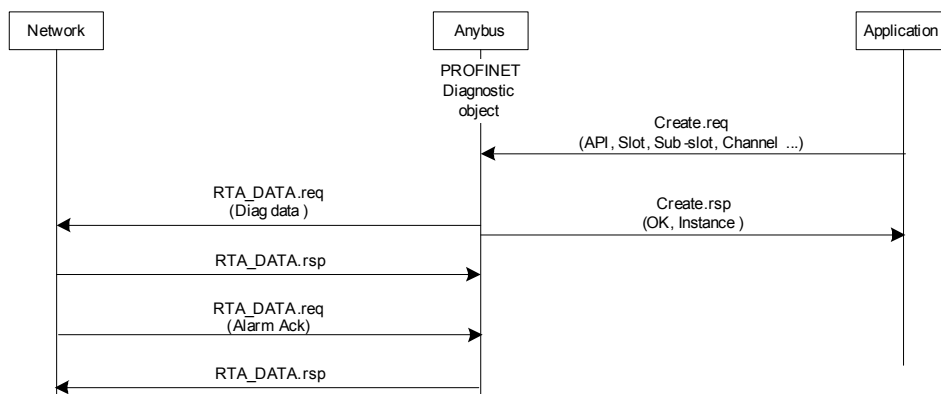
Application Example, Send Generic Diagnostics Alarm Failure

In this example the host application tries to send a malformed request.



Application Example, Send Generic Diagnostics Alarm

In this example the host application sends a generic diagnostic alarm to the IO controller.

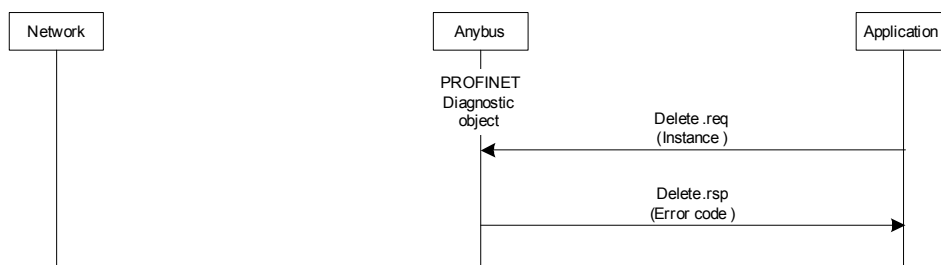


Delete Diagnostic Entry (Instance Service)

A diagnostic entry in the PROFINET Diagnostic Object is removed by sending this command (0x0007) to the instance to be deleted.

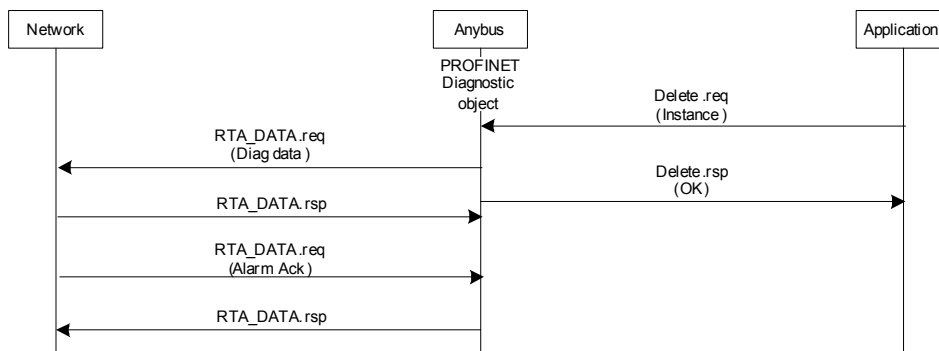
Application Example, Send Delete Diagnostic Entry (Fail).

In this example the host application tries to delete, for example, a nonexistent diagnostic entry.



Application Example, Send Delete Diagnostic Entry

In this example the host application removes a diagnostic entry.



Diagnostic Errors

Diagnostic Error	Name
0	No error
1	API does not exist
2	No module inserted in the specified slot
3	No submodule inserted in the specified slot
4	Slot number specified is out-of-range
5	Subslot number specified is out-of-range
6	Failed to add the diagnostic entry to the PROFINET IO stack
7	Failed to send the diagnostic alarm to the PROFINET IO stack
8	Channel number is out-of-range
9	ChannelPropType is out-of-range
10	ChannelPropDir is out-of-range
11	ChannelPropAcc is out-of-range
12	ChannelPropMaintReq is out-of-range
13	ChannelPropMaintDem is out-of-range
14	UserStructident is out-of-range
15	ChannelErrType is out-of-range
16	Failed to remove the diagnostic entry from the PROFINET IO stack
17	No resources (too many diagnostic entries created)
18	Unknown error

PROFINET Mapping Object (Class A9h)

General Information

Object Description

The PROFINET Mapping Object enables the application to make application specific parameters available from the PROFINET IO network, via the acyclic services 'Record Read' and 'Record Write'.

Supported Services

Class: Get_Attribute
 Create (see "Create (Class Service)" on page 144)

Instance: -

Class Attributes

#	Type	Access	Name	Req	Description
1	Byte	R	Class revision	R	Revision number of the class (0x0001)
2	Word	R	Number of Instances	R	Number of instances in the class

Instance Attributes

#	Type	Access	Name	Req	Description
1	Word	R	Instance	-	Application Parameter Object Class instance number
2	Word	R	Slot number	-	Number of the slot for access
3	Word	R	Sub-slot number	-	Number of the subslot for access Note: The combination slot=0, subslot=0 means that the slot/subslot number is not applicable (don't care)
4	Word	R	Index	-	User specific index for the access
5	Byte	R	Initial Record Data	-	Initial record data flag: 0 - This parameter is not considered to be initial record data. 1 - This parameter is considered as initial record data and must be properly set by the IO controller during commissioning (otherwise a parameter fault will be signaled)

Create (Class Service)

Service Description

This service creates a mapping of one 'Application Parameter' to the PROFINET network. The command is only valid before the application changes the 'Module Mode' to 'Normal Operation Mode' or 'Fieldbus specific init'. Note that the Anybus module will not check the mappings added, therefore it is possible to create mappings that cannot be accessed from the PROFINET network (for example, if the application creates a mapping of an application parameter to slot number 32 and this slot is not plugged - thus this application parameter cannot be accessed from PROFINET).

Service Request (0x05)

#	Type	Name	Description
1	Word	Instance	Application Parameter Object Class instance number
2	Word	Slot number	Slot number to which the map shall be tied Range: 0 ... 32
3	Word	Sub-slot number	Subslot number to which the map shall be tied Range: 0 ... 8
4	Word	Index	Index: 0x0000 ... 0x7FFF - Manufacturer specific
5	Byte	Initial Record Data	Initial record data flag: 0 - This parameter is not considered to be initial record data. 1 - This parameter is considered as initial record data and must be properly set by the IO controller during commissioning (otherwise a parameter fault will be signaled). If set to '1', parameter 'Sub-slot number' cannot be set to zero, unless also parameter 'Slot number' is set to zero.

Service Response (0x06)

#	Type	Name	Description
1	Word	Diagnostic Error	See "Diagnostic Errors" on page 142
2	Word	Instance	Created instance number

Note:

- 1). It is possible to map the same application parameter instance to several PROFINET mappings.
- 2). If the 'Slot number' and 'Sub-slot number' are both set to zero the mapping applies to all combinations of slot and subslot numbers.

Mapping Errors

Diagnostic Error	Name
0	No error
1	Application parameter instance not available
2	Slot number is out-of-range
3	Subslot number is out-of-range
4	Index is out-of-range
5	Initial record data parameter is out-of-range
6	Invalid combination ('Initial Record Data' and 'Slot number' / 'Sub-slot number')

Conformance Test Guide

General

When using the default settings of all parameters, the Anybus-IC 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 the PROFIBUS Nutzerorganisation (PNO) and is compulsory for certification. This chapter provides a guide for successful conformance testing your product, containing the Anybus-IC 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.*

Reidentifying Your Product

After successful setting of the FB init parameter, the Anybus module asks for identification data. Therefore, the parameters and attributes listed below shall be implemented and proper values returned.

Parameters

Parameter	Explanation	Default	Customer Sample	Comment
PRT device id (#163)	With this attribute you set the Device ID of the device	Device ID: 0008h	Device ID: YYYYh	This information must match the keys of the "DeviceIdentity" of the GSD file. Note that the GSD file keyword "VendorName" must correspond to the Vendor ID value.
PRT vendor id (#162)	With this attribute you set the Vendor ID of the device	Vendor ID: 010Ch (HMS)	Vendor ID: XXXXh	
PRT station type (#164)	With this attribute you set the station type of the device	"ABIC-PRT"	"Cust-PNIO-Dev"	This information matches, in the case of Anybus-IC 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.

Attributes

Object	Attribute	Explanation	Default	Customer Sample	Comment
I&M Object (A7h)	#4, I&M Order ID	With this attribute you set the Order ID that is used in the I&M data.	"ABIC-PRT "	"Cust-PNIO-Dev "	This information must match the keys of the "OrderNumber" of the GSD file.
I&M Object (A7h)	#6, 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.
I&M Object (A7h)	#7, 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.
Fieldbus Object (A0h)	#56, System Description	With this attribute you set the description of the system.	"HMS Industrial Networks Anybus-IC"	"Customer HMI Interface Module"	This information can be read by means of SNMP from the network side
Fieldbus Object (A0h)	#57, Interface Description	With this attribute you set the description of the interface.	"PROFINET IO Interface"	"PROFINET IO Interface"	

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".

Factory Default Reset

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. HMS recommends using the RES bit for this behavior.

IP Address

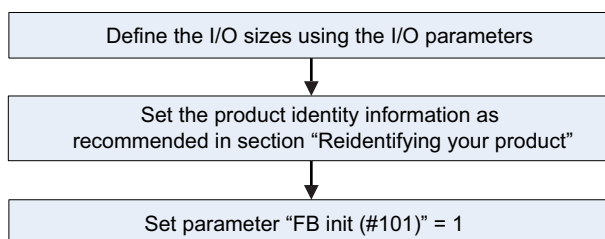
Normally the IP numbers of PROFINET IO devices are assigned via the PROFINET network via DCP (Discovery and Configuration Protocol). HMS recommends not using parameters “IP address cfg (#103)”, “Subnet mask cfg (#106)” and “GW address cfg (#108)” during the initialization phase for PROFINET modules, 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).

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 parameter “PRT station name (#166)” 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 (“”).

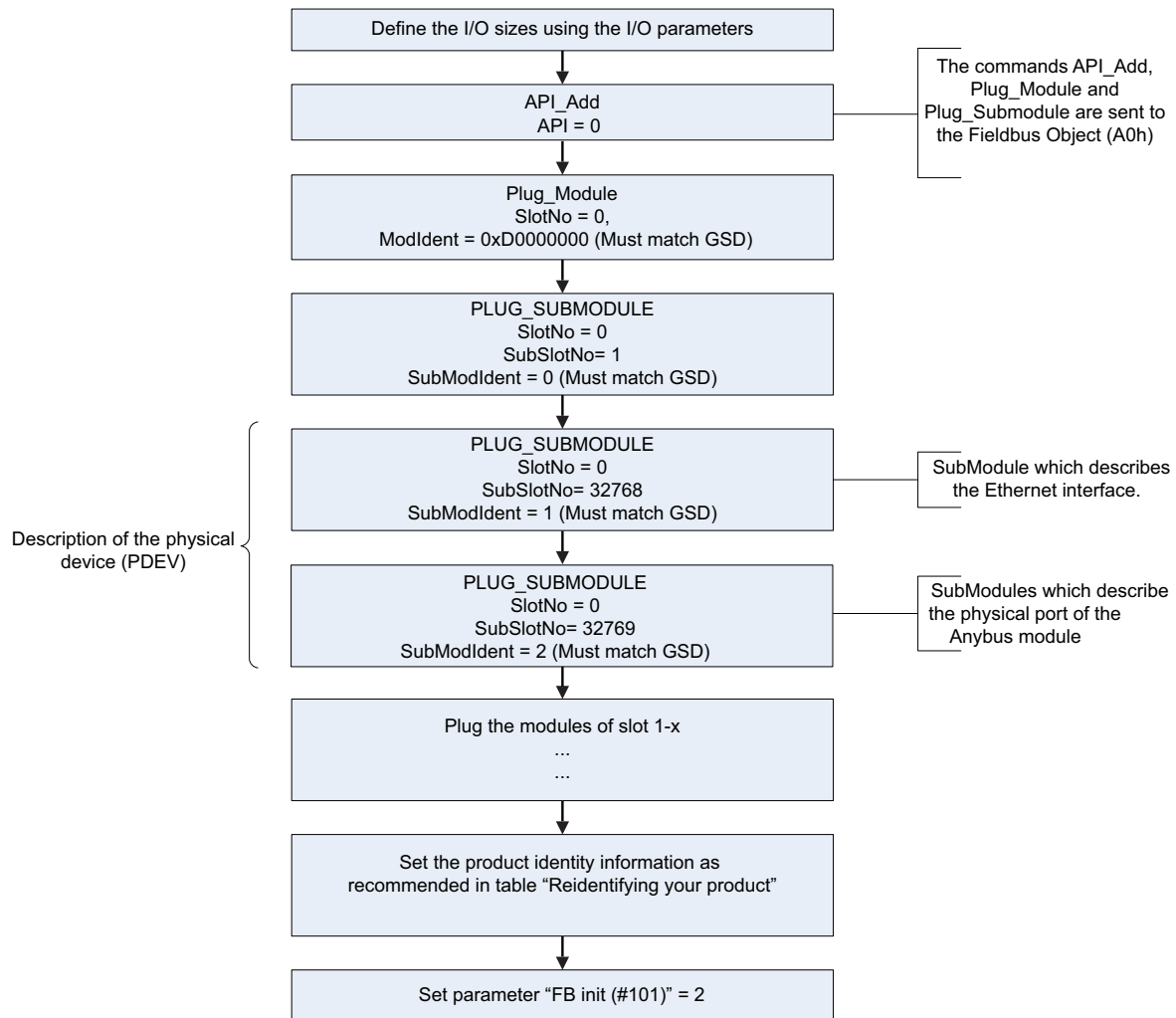
Certification in Generic Anybus Mode

In generic Anybus mode 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.



Certification in Advanced Mode

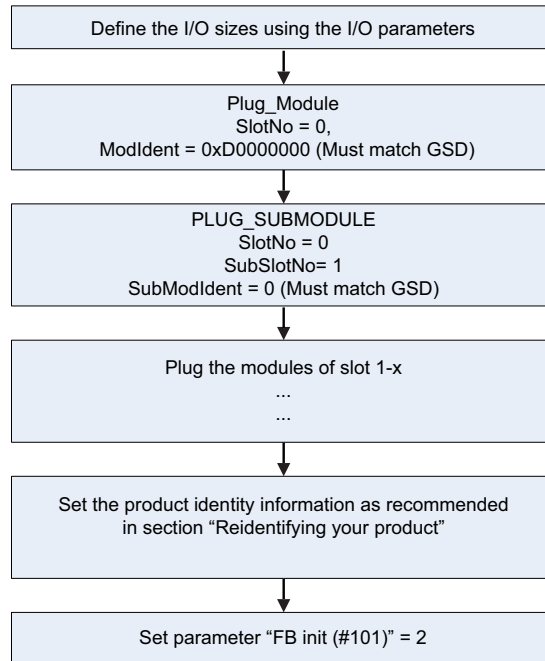
In advanced mode the most important thing is to use a Device Access Point (DAP) that conform to PROFINET IO Specification v2.0 or later (DAP2). From specification version 2.0 it is possible to describe the physical Ethernet interface and its ports (PDEV, or Physical Device) with a special mechanism. 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 flowchart 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.

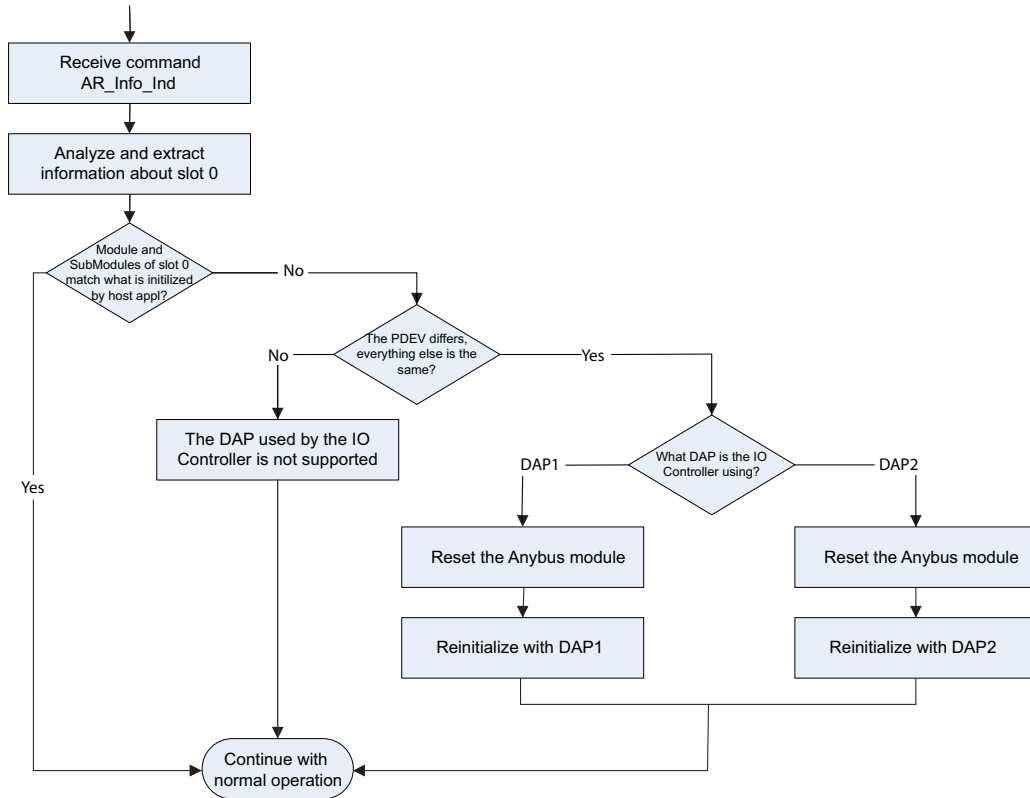
Please note that the values of "SubModIdent" in the above flowchart are the values of the default HMS GSD file. They can be changed if necessary, but there is no real need for it, the important thing is that it matches the GSD file. To be able to pass the PROFINET conformance test a "DAP2" is mandatory. 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 two 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 two Plug_Submodules as shown in the figure on page 150, 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 parameter ArInfoInd Data (#169) 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 parameter ArInfoInd (#169) always contains the most recent information about the application relationship, handed over by the IO controller during connection establishment.

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.

Technical Specification

Electrical Specification

Protective Earth (PE) Requirements

See “Fieldbus Interface” on page 96 .

Power Supply

Supply Voltage

The module requires a regulated 5 V $\pm$ 5% DC power supply as specified in the Anybus-IC Design Guide.

Power Consumption

The maximum power consumption is 250 mA.

Environmental Specification

- **Temperature**

Test performed according to IEC-68-2-1 and IEC 68-2-2.

Operating: -40 to +85°C (-40 to +185 °F)

Storage: -40 to +85°C (-40 to +185 °F)

- **Humidity**

The product is designed for a relative humidity of 5 to 95% noncondensing.

Test performed according to IEC 68-2-30.

EMC Compliance (CE)

EMC precompliance testing has been conducted according to the following standards:

- **Emission:** EN 61000-6-4

Tested per EN 55016-2-3

- **Immunity:** EN 61000-6-2

Tested per EN 61000-4-2

EN 61000-4-3

EN 61000-4-4

EN 61000-4-5

EN 61000-4-6