



Anybus™ CompactCom 40

PROFINET IRT

NETWORK GUIDE

HMSI-27-226 2.1 ENGLISH

Important User Information

Liability

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

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

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

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

Intellectual Property Rights

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

Table of Contents

Page

1	Preface	5
1.1	About this document	5
1.2	Related Documents	5
1.3	Document History	6
1.4	Conventions	7
1.5	Terminology.....	7
2	About the Anybus CompactCom 40 PROFINET IRT	8
2.1	General.....	8
2.2	Features	8
2.3	Differences between the 40 and 30 series.....	9
3	Basic Operation	10
3.1	General Information	10
3.2	Network Identity.....	13
3.3	Communication Settings.....	13
3.4	Network Data Exchange.....	14
3.5	Web Interface	15
3.6	E-mail Client.....	16
3.7	File System	17
4	PROFINET Implementation Details.....	19
4.1	General Information	19
4.2	Application Process Instances (API).....	19
4.3	Application Relationships (AR).....	20
4.4	Real Identification (RI).....	20
4.5	Diagnostics	24
4.6	Identification & Maintenance (I&M).....	25
4.7	Fast Start Up	26
4.8	Address Conflict Detection (ACD)	29
4.9	PROFIsafe	30
5	FTP Server	31
5.1	General Information	31
5.2	User Accounts	31
5.3	Session Example	32

6	Web Server	33
6.1	General Information	33
6.2	Default Web Pages	33
6.3	Server Configuration	36
7	E-mail Client	39
7.1	General Information	39
7.2	How to Send E-mail Messages	39
8	Server Side Include (SSI)	40
8.1	General Information	40
8.2	Include File.....	40
8.3	Command Functions	40
8.4	Argument Functions	53
8.5	SSI Output Configuration.....	56
9	JSON	57
9.1	General Information	57
9.2	JSON Objects.....	57
9.3	Example.....	63
10	SNMP Agent	65
10.1	General.....	65
10.2	Management Information (MIB).....	65
10.3	MIB_II	65
11	Media Reduncancy Protocol (MRP)	68
11.1	General.....	68
11.2	GSDML Entries.....	68
12	Anybus Module Objects	69
12.1	General Information	69
12.2	Anybus Object (01h)	70
12.3	Diagnostic Object (02h)	71
12.4	Network Object (03h)	74
12.5	Network Configuration Object (04h).....	75
12.6	Network PROFINET IO Object (0Eh).....	82
12.7	Socket Interface Object (07h).....	99
12.8	SMTP Client Object (09h).....	116
12.9	File System Interface Object (0Ah).....	121
12.10	Network Ethernet Object (0Ch)	121
12.11	Functional Safety Module Object (11h)	122

13 Host Application Objects	126
13.1 General Information	126
13.2 Sync Object (EEh)	127
13.3 PROFINET IO Object (F6h)	131
13.4 Ethernet Host Object (F9h)	149
13.5 Functional Safety Object (E8h).....	152
A Categorization of Functionality	155
A.1 Basic.....	155
A.2 Extended.....	155
B Anybus Implementation Details	156
B.1 SUP-Bit Definition	156
B.2 Anybus State Machine	156
B.3 Application Watchdog Timeout Handling.....	156
C Flowcharts	157
C.1 Flowchart — Record Data Access	157
C.2 Flowchart — I&M Record Data Handling.....	158
C.3 Flowchart —Establishment of Real Identification (RI).....	159
C.4 Flowcharts — Handling of Configuration Mismatch	160
D Secure HICP (Secure Host IP Configuration Protocol)	162
D.1 General.....	162
D.2 Operation	162
E Technical Specification	163
E.1 Front View	163
E.2 Functional Earth (FE) Requirements.....	164
E.3 Power Supply	165
E.4 Environmental Specification.....	165
E.5 EMC Compliance.....	165
E.6 Fiber Optics Compliance (MAU type Compliance)	165

- F Conformance Test Guide 166**
 - F.1 General 166
 - F.2 Reidentifying Your Product 166
 - F.3 Factory Default Reset..... 167
 - F.4 IP Address..... 167
 - F.5 Station Name..... 168
 - F.6 Documentation Considerations 168
 - F.7 Certification in Generic Anybus Mode 168
 - F.8 Certification in Advanced Mode 169
 - F.9 Changes in GSD File for Conformance Class B 170
 - F.10 SYNC Pin Measurements for Conformance Class C Test 171

- G Copyright Notices 172**

1 Preface

1.1 About this document

This document is intended to provide a good understanding of the functionality offered by the Anybus CompactCom 40 PROFINET IRT. The document describes the features that are specific to Anybus CompactCom 40 PROFINET IRT. For general information regarding Anybus CompactCom 40, consult the Anybus CompactCom 40 design guides.

The reader of this document is expected to be familiar with high level software design and communication systems in general. The information in this network guide should normally be sufficient to implement a design. However if advanced PROFINET specific functionality is to be used, in-depth knowledge of PROFINET networking internals and/or information from the official PROFINET specifications may be required. In such cases, the persons 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.

For additional related documentation and filed downloads, please visit the support website at www.anybus.com/support.

1.2 Related Documents

Related documents

Document	Author
Anybus CompactCom 40 Software Design Guide	HMS
Anybus CompactCom 40 Hardware Design Guide	HMS
Anybus CompactCom Host Application Implementation Guide	HMS
PROFINET IO specification, rev. 2.3	Profibus International
PROFenergy Technical Specification, rev. 1.0	Profibus International
GSDML Technical Specification for PROFINET IO, rev 2.31	Profibus International
IXXAT Safe T100, Safety Manual — Original Instructions	HMS

1.3 Document History

Summary of changes in this version

Change	Where (section no.)
Updates to chapter on JSON	9
Error fixed in GSD example "How to Enable Initial Record Data"	3.1.2
Corrected description of Module Status LED	E.1.4
Ethernet Host Object (F9h) corrected.	13.4
Conformance Test appendix updated (G.3.10)	F
Added max process data numbers to feature list	2.2

Revision list

Version	Date	Author	Description
1.00	2014-06-13	KaD	First public release
1.10	2014-09-01	KeL, KaD	Major updates
1.11	2014-10-09	KaD	Added Fiber Optic information, minor updates
1.20	2014-12-15	KaD	Major updates
1.21	2015-03-06	KaD	Major updates
1.22	2015-03-XX	KaD	Minor updates
1.30	2015-11-12	KeL	Updated Configuration Mismatch information, minor updates
2.0	2016-06-22	KeL	Moved to DOX, major updates
2.1	2016-08-26	KeL	Corrections

1.4 Conventions

Unordered (bulleted) lists are used for:

- Itemized information
- Instructions that can be carried out in any order

Ordered (numbered or alphabetized) lists are used for instructions that must be carried out in sequence:

1. First do this,
2. Then open this dialog, and
 - a. set this option...
 - b. ...and then this one.

Bold typeface indicates interactive parts such as connectors and switches on the hardware, or menus and buttons in a graphical user interface.

Monospaced text is used to indicate program code and other kinds of data input/output such as configuration scripts.

This is a cross-reference within this document: [Conventions, p. 7](#)

This is an external link (URL): www.hms-networks.com



This is additional information which may facilitate installation and/or operation.



This instruction must be followed to avoid a risk of reduced functionality and/or damage to the equipment, or to avoid a network security risk.



Caution

This instruction must be followed to avoid a risk of personal injury.



WARNING

This instruction must be followed to avoid a risk of death or serious injury.

1.5 Terminology

- The terms “Anybus” or “module” refers to the Anybus CompactCom module.
- The terms “host” or “host application” refer to the device that hosts the Anybus.
- Hexadecimal values are written in the format NNNNh or 0xNNNN, where NNNN is the hexadecimal value.
- A byte always consists of 8 bits.
- The terms “basic” and “extended” are used to classify objects, instances and attributes.

2 About the Anybus CompactCom 40 PROFINET IRT

2.1 General

The Anybus CompactCom 40 PROFINET IRT communication module provides instant PROFINET Real Time connectivity via the patented Anybus CompactCom host interface. Any device that supports this standard can take advantage of the features provided by the module, allowing seamless network integration regardless of network type.

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

2.2 Features

- Ethernet connectors
- Up to 128 submodules in total
- Up to 32767 ADIs
- Max. read process data: 1308 bytes
- Max. write process data: 1308 bytes
- Max. process data (read + write, in bytes): 2616 bytes
- Generic and PROFINET specific diagnostic support
- Complies with PROFINET IO Conformance class C
- Supports up to 1440 bytes I/O data in each direction, status bytes included.
- Supports 250 µs cycle time
- SNMP agent
- FTP server
- E-mail client
- Server Side Include (SSI) functionality
- JSON functionality
- Device identity customization
- GSD file template provided by HMS
- Supports PROFINET Fast Start Up
- Supports Media Redundancy Protocol (MRP)

2.3 Differences between the 40 and 30 series

2.3.1 I&M

- I&M5-15 requests are rejected by the CompactCom 40 module in nontransparent I&M mode.
- I&M Record data transparent mode is replaced with the IM_Options command.
- I&M0 parameters IM Version and IM Supported are removed from the PROFINET Host object and set to constant values.
- Attributes #13 and #14 in the PROFINET IO Object (F6h) are only read for submodules belonging to a “non-zero API”. Constant values are used for submodules belonging to API 0.

2.3.2 Diagnostics

- Process alarms cannot be created.
- Diagnostic events can be created when not online (i.e. in PROCESS_ACTIVE or IDLE).
- The structure of network specific event information has changed. Instead of including diagnostic source information such as API, Slot and Subslot in the data field it is extracted from the extended diagnostic fields in the create command. API, Slot and Subslot is determined with the help of Slot and ADI given by the extended diagnostic mode.

2.3.3 Network Configuration Object

- Network specific instances are moved from instance number 15 and on to instance number 20 and on. This is done to increase the amount of instances in the part that is not network specific.
- The network specific instances that handled I&M data have been removed, as they are not possible to set from the application.

2.3.4 PROFINET Additional Diagnostic Object

- This object is removed. All diagnostics are handled via the standard diagnostic object.

2.3.5 PROFINET Host Object

- The instance #1 attribute PROFlenergy functionality does no longer carry any functionality. It is now set to reserved.
- The instance #1 attribute System Contact is now set to reserved and no longer used.
- The Expected_Ident_Ind command replaces Ar_Info_Ind in the [PROFINET IO Object \(F6h\)](#), p. 131 for more information.

2.3.6 LEDs

- Major internal error is now covered by the FATAL event. It is signaled by both Module and Status LEDs being solid red.

2.3.7 SNMP MIB-II

- sysContact, sysLocation and sysName are no longer connected to any other parameters/attributes. They are stored as separable readable/writeable parameters in the module's nonvolatile memory.

3 Basic Operation

3.1 General Information

3.1.1 Software Requirements

Generally, no additional network support code needs to be written to support the Anybus CompactCom 40 PROFINET IRT, however due to the nature of the PROFINET networking system certain things must be taken into account:

- Up to 32767 ADIs can be represented on PROFINET.
- ADI names, types and similar attributes cannot be accessed via PROFINET. They are however represented on the network through the built in web server.
- Up to 5 diagnostic instances can be created by the host application. An additional 6th instance may be created in event of a major fault.
- For conformance reasons, the host application must implement support for network reset types 00h (Power-on) and 02h (Power-on + Factory Default) in the Application Object (FFh).
- PROFINET in itself does not impose any particular timing demands when it comes to acyclic requests (i.e. requests towards instances in the Application Data Object), however it is generally recommended to process and respond to such requests within a reasonable time period (exactly what this means in practice depends on the implementation and the actual installation).
- The order in which ADIs are mapped to Process Data is significant and must be replicated in the IO Controller when setting up the network communication (i.e. modules must be set up in the same order, size and direction, as the mapped ADIs). In case of a configuration mismatch, see [Configuration Mismatch, p. 22](#) for more information.

See also ...

- [Application Data Instances \(ADIs\), p. 14](#)
- Diagnostic Object [Anybus Module Objects, p. 69](#)
- Anybus CompactCom 40 Software Design Guide, Application Data Object (FEh)

3.1.2 Electronic Data Sheet (GSD)

On PROFINET, the characteristics of a device is stored in an XML data file. This file, referred to as the “GSD” file, is used by PROFINET engineering tools when setting up the network configuration. HMS Industrial Networks AB provides an example GSD file, which must be adapted by the user to suit the application.

Setting Identity and Function Information

The GSD file must be adapted to your implementation. First thing is the device identity.

In the GSD file there is a section called “DeviceIdentity”. It looks like this.

```
<DeviceIdentity VendorID="0x010C" DeviceID="0x0010">
  <InfoText TextId="T_ID_DEV_DESCRIPTION"/>
  <VendorName Value="HMS Industrial Networks"/>
</DeviceIdentity>
```

The identity in the example represents HMS values.

- Replace VendorID value 0x010C with the value which corresponds to your vendor name.
If you do not have a Vendor ID you can obtain this by contacting PI.
Set with PROFINET IO Object (0xF6) attribute #2 (Vendor ID).
- Replace the DeviceID value 0x0010 with the value you have selected for this device.
Set with PROFINET IO Object (0xF6) attribute #1 (Device ID). Please note that if you change the Device ID you MUST also change the Vendor ID, as the Device ID is unique for the Vendor ID.
- Replace the VendorName value "HMS Industrial Networks" with name of your vendor.
Please note that the keyword VendorName is found not only here, but also at other places in the GSD file. Use "Search" to find all instances and replace them with the name of your vendor.

Specify the function of the device.

```
<DeviceFunction>
  <Family MainFamily="General" ProductFamily="Anybus CompactCom 40 PIR"/>
</DeviceFunction>
```

The example GSD specifies a kind of "General" device as the usage of it is unclear.

- Replace the MainFamily with the class that best describe the device. The following are the allowed values:
 - General
 - Drives
 - Switching Devices
 - I/O
 - Valves
 - Controllers
 - HMI
 - Encoders
 - NC/RC
 - Gateway
 - PLCs
 - Ident Systems
 - PA Profiles
 - Network Components
 - Sensors
- Replace the ProductFamily value "Anybus CompactCom 40 PIR" with a string which describes your device.

In addition to the above, there are a few more places where identity related information is present in the GSD file.

```
DNS_CompatibleName="ABCC40-PIR"
```

For the Device access point (DeviceAccessPointItem) there is a keyword which is called DNS_CompatibleName. Locate this by using the search function.

- Replace DNS_CompatibleName value "ABCC40-PIR" with the set Station Type. Set with PROFINET IO Object (0xF6) attribute #3 (Station Type).

The order number of the device is set with the keyword "OrderNumber".

```
<OrderNumber Value="ABCC40-PIR"/>
```

In many cases the value of the OrderNumber equals the Station Type string, but it does not necessary need to be that way.

- Replace the OrderNumber value "ABCC40-PIR" with the order number used for the device.

Set with PROFINET IO Object (0xF6) attribute #8 (IM Order ID).

How to Enable Initial Record Data

During the establishment of an IO connection between the IO device and the IO controller it is possible for the IO controller to send initial record data. This initial record data is sent using the PROFINET IO service record write. This service can be used at any time and will write data to a defined ADI. The initial record data is defined in the GSD file, and is specified for a submodule of a module. By default, the Anybus CompactCom 40 module will not make use of any initial record data, but that can be enabled if needed.

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

```
<RecordDataList>
  <ParameterRecordDataItem Index="67" Length="2" TransferSequence="0">
    <Name TextId="T_ID_EXAMPLE2"/>
    <Ref DataType="Unsigned16" ByteOffset="0" DefaultValue="0"
      AllowedValues="0..65535" TextId="T_ID_EXAMPLE2_PRM_1"/>
  </ParameterRecordDataItem>
</RecordDataList>
```

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

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

Please note that TextId's ("T_ID_xxx" above) need to be added to the "<ExternalTextList>" section of the GSD file (once for each language defined).

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

3.2 Network Identity

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

Device ID	0010h (Anybus CompactCom 40 PROFINET)
Vendor ID	010Ch (HMS Industrial Networks)
Station Type	"ABCC40-PIR"

It is recommended to customize the identity information so that the Anybus module appears as a vendor

specific implementation rather than a generic Anybus product.

See also...

- [Identification & Maintenance \(I&M\), p. 25](#)
- [PROFINET IO Object \(F6h\), p. 131](#)

3.3 Communication Settings

Network related communication settings are grouped in the Network Configuration Object (04h), and includes:

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

See also...

- [Identification & Maintenance \(I&M\), p. 25](#)
- [Web Server, p. 33](#)
- [Network Configuration Object \(04h\), p. 75](#)
- [Secure HICP \(Secure Host IP Configuration Protocol\), p. 162](#)

3.4 Network Data Exchange

3.4.1 Application Data Instances (ADIs)

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

The following parameters affect the addressing of ADIs onPROFINET:

Application Process Instance (API)	API 0 (zero) provides access to data in the Application Data Object, i.e. the ADIs. Acyclic requests towards other APIs will be forwarded to the PROFINET IO Object (F6h) , p. 131 by means of the 'Get_Record' and 'Set_Record'-commands. The remainder of this section assumes API 0 (zero).
Slot & subslot	The Slot and subslot values have no impact on the actual addressing of ADIs, except that the actual Slot and subslot needs to be populated with a module/submodule. This is always true for the DAP (Device Access Point), which occupies Slot #0/subslot #1. Other Slot/subslot values can also be used provided that the implementation populates it with a module/submodule.
Index	There is a 1:1 correlation between ADI and index as long as the index number is less than - or equal to - 7FFFh. Index 0 (zero) is not associated with an ADI and cannot be used.

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

See also...

- [Caveats](#), p. 15
- [Application Process Instances \(API\)](#), p. 19
- [PROFINET IO Object \(F6h\)](#), p. 131

3.4.2 Process Data

Mapping an ADI to Write Process Data results in PROFINET input data, and mapping an ADI to Read Process Data results in PROFINET output data. Consistency over all I/O data mapped on PROFINET is guaranteed.

See also...

- [Real Identification \(RI\), p. 20](#)



The order in which ADIs are mapped to Process Data is significant and must be replicated in the IO Controller when setting up the network communication (i.e. modules must be set up in the same order, size and direction, as the mapped ADIs).

3.4.3 Caveats

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

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

See also..

- [Application Process Instances \(API\), p. 19](#)

3.5 Web Interface

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

Web server content resides within the module's file system. This means that it can be accessed and customized as needed using a standard FTP client.

See also...

- [File System, p. 17](#)
- [FTP Server, p. 31](#)
- [Web Server, p. 33](#)
- [Server Side Include \(SSI\), p. 40](#)
- [JSON, p. 57](#)

3.6 E-mail Client

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

See also...

- [File System, p. 17](#)
- [E-mail Client, p. 39](#)
- [Server Side Include \(SSI\), p. 40](#)
- [JSON, p. 57](#)
- [SMTP Client Object \(09h\), p. 116](#)

3.7.2 General Information

The built-in file system hosts 28 Mb of nonvolatile storage, which can be accessed by the HTTP and FTP servers, the e-mail client, and the host application (through the Anybus File System Interface Object (0Ah)).

Maximum number of directories and files that can be stored in the root directory is 511, if only short filenames are used (8 bytes name + 3 bytes extension). If longer filenames are used, less than 511 directories/files can be stored. This limitation does not apply to other directories in the file system.

The file system uses the following conventions:

- \ (backslash) is used as a path separator
- Names may contain spaces, but must not begin or end with one.
- Valid characters in names are ASCII character numbers less than 127, excluding the following characters: \ / : * ? " < > |
- Names cannot be longer than 48 characters
- A path cannot be longer than 126 characters (filename included)

See also...

- [FTP Server, p. 31](#)
- [Web Server, p. 33](#)
- [E-mail Client, p. 39](#)
- [Server Side Include \(SSI\), p. 40](#)
- [File System Interface Object \(0Ah\), p. 121](#)



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

The following operations will erase one or more flash segments:

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

3.7.3 System Files

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

Example 1:

```
[Key1]
value of Key1

[Key2]
value of Key2
```

4 PROFINET Implementation Details

4.1 General Information

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

Implementation overview:

Conformance Class	The Anybus module complies to conformance class C.
Performance Characteristics	<ul style="list-style-type: none"> • 100 Mbps, full duplex with autonegotiation enabled as default • Real Time (RT) communication, 250 ?s cycle time • Isochronous Real Time (IRT) communication, 250 ?s cycle time
Device Model	<ul style="list-style-type: none"> • One IO Device instance • The IO Device instance includes an Application Process referenced by its identifier (API). API 0 (zero) is implemented by default. • The API implements one or more slots • Each Slot implements one or more subslots • Each subslot may implement one or more Channels
Slots & Subslots	Up to 128 subslots in total.
IO Data	1440 bytes of IO data in each direction, including status bytes (4 bytes for DAP submodules + 1 byte per additional submodule)

See also...

- [Electronic Data Sheet \(GSD\), p. 10](#)

4.2 Application Process Instances (API)

As mentioned previously, acyclic requests towards API 0 are forwarded to the Application Data Object.

Cyclic data exchange is by default carried out through API 0 (i.e. the Anybus associates modules and submodules with API 0).

See also...

- [Application Data Instances \(ADIs\), p. 14](#)

4.3 Application Relationships (AR)

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

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

Application Relationships are managed through the following functions:

- AR_Check_Ind (for command details see [PROFINET IO Object \(F6h\), p. 131](#))
- Expected_Ident_Ind (for command details see [PROFINET IO Object \(F6h\), p. 131](#))
- AR_Abort_Ind (for command details see [PROFINET IO Object \(F6h\), p. 131](#))
- AR_Abort (for command details see [Network PROFINET IO Object \(0Eh\), p. 82](#))

4.4 Real Identification (RI)

4.4.1 General Information

During the establishment of an IO Connection towards the Anybus CompactCom 40 PROFINET IRT, the configuration derived from the IO Controller (i.e. the Expected Identification) and the actual configuration in the Anybus CompactCom 40 PROFINET IRT (i.e. the Real Identification or RI) are compared.

The RI configuration is either handled by the module (default), or by the host application. In either case the GSD file has to be customized to correspond to the configuration.

Default Configuration (ADI Based)

By default (i.e. if the application does not issue API_Add, Plug_Module, Plug_Submodule), the Anybus CompactCom 40 PROFINET IRT handles the plugging of modules and submodules automatically in accordance with the mapped Process Data as follows:

- A DAP is plugged into Slot 0 (zero)
- Modules are added in consecutive order (based on the order of the mapping commands)
- All modules belong to API 0 (zero)

The Anybus CompactCom 40 PROFINET IRT internally creates module/submodule identifiers as described in the picture and the example below. The GSD file has to be customized to define the same modules/submodules with the same identifiers as the Anybus CompactCom 40 PROFINET IRT has created internally.

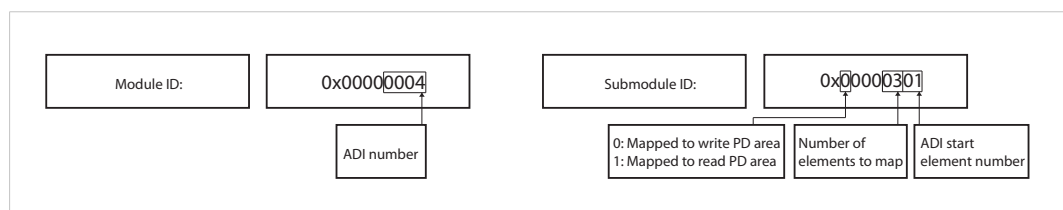


Fig. 2

Example (100BASE-TX DAP):

ADI #	Type	Resulting Real Identification		
		Module/Submodule ID	Slot/Subslot	IO Data Size (bytes)
-	-	0x80010000/ 0x00000001 0x80010000/ 0x00000002 0x80010000/ 0x00000003 0x80010000/ 0x00000003	0/1 (Device Access Point) 0/0x8000 (Interface) 0/0x8001 (Port1) 0/0x8002 (Port2)	-
6	1 SINT16 Wr	0x00000006/ 0x00000100	1/1	2/0
3	2 UINT8 Rd	0x00000003/ 0x10000200	2/1	0/2
3	1 UINT8 Rd	0x00000003/ 0x10000104	2/2	0/1
9	1 SINT32 Rd	0x00000009/ 0x10000100	3/1	0/4
10	1 SINT16 Wr	0x0000000A/ 0x00000100	4/1	2/0

See also...

- [Application Process Instances \(API\), p. 19](#)
- [Flowchart —Establishment of Real Identification \(RI\), p. 159](#)

Custom Configuration (Advanced Users)

Optionally it is possible to override the default configuration during the SETUP state by means of the PROFINET specific commands API_add, Plug_Module, Plug_Submodule, and Plug_Submodule_Ext. This way, the host application can define exactly how ADIs are represented on PROFINET by defining custom modules and submodules. These commands need to be sent after the process data is mapped, and before sending setup complete to the Anybus Compact-Com 40 PROFINET IRT.

See also...

- [Network PROFINET IO Object \(0Eh\), p. 82](#)
- [Flowchart —Establishment of Real Identification \(RI\), p. 159](#)

4.4.2 Configuration Mismatch

General

A configuration mismatch occurs when the Real Identification (RI) does not match the Expected Identification. Depending on how the RI configuration is established, the Anybus CompactCom 40 PROFINET IRT will first try to resolve the mismatch as described in the applicable section below (Resolving Mismatch for default configuration or custom configuration). If this attempt to resolve the mismatch fails, the Anybus CompactCom 40 PROFINET IRT will provide indications as described in [Further Actions to Resolve Mismatch, p. 23](#).

Resolving Mismatch (Default Configuration)

If the Real Identification has been established according to the default mode, the Anybus CompactCom 40 PROFINET IRT will try to remap the Real Identification to match the Expected Identification.



The application must have implemented support for the remap commands in the Application Data object (FEh), for remap to be possible.

As the Module ID contains the ADI number and the Submodule ID describes I/O direction and element section, all information required to perform a remap is available.



The application must be able to respond with the data type for every ADI, or a remap is not possible.

The remap is performed as follows:

- A request is sent to remap the read area. If this request is rejected no change is made to the process data map and the Real Identification.
- If the read area remap succeeds, a write remap request is sent. If this is rejected, the process data map is in an inconsistent state and new remap commands are sent that will remove all mappings.
- If both the read and the write remap requests succeed a new Real Identification will be built to match the Expected Identification.

See also ...

- [Default Configuration Mismatch, p. 160](#) (flowchart)
- Application Data Object (FEh) (see Anybus CompactCom 40 Software Design Guide)

Resolving Mismatch (Custom Configuration)

If a configuration mismatch occurs for a custom configuration, the Anybus CompactCom 40 PROFINET IRT will issue the command Expected_Ident_Ind to the host application. If the host application intends to change the Real Identification based on the Expected Identification, it responds with “Block” and performs the required Pull/Plug operations, before sending an Ident_Change_Done command to the Network PROFINET IO Object (0Eh).

See also

- [Custom Configuration mismatch, p. 161](#) (flowchart)
- Expected_Ident_Ind, see command details in [PROFINET IO Object \(F6h\), p. 131](#)
- Ident_Change_Done, see command details in [Network PROFINET IO Object \(0Eh\), p. 82](#)

Further Actions to Resolve Mismatch

If the mismatch remains unsolved, either for default or custom configuration, the following will be performed by the Anybus CompactCom 40 PROFINET IRT to find a solution that will make exchange of valid data possible:

Incomplete output mapping:	If the controller tries to connect to fewer output submodules than are plugged by the application, the controller will not be granted ownership of any output submodule. The Anybus state is set to ERROR and the LEDs will indicate configuration mismatch. Matching input submodules will be owned by controller and work normally.
Incomplete input mapping:	The controller may choose to connect to a subset of the available input submodules without any restrictions. Anybus state is set to PROCESS_ACTIVE and no error is indicated on any LED.
Mismatch of submodule(s):	As long as all of the output submodules of the Real Identification are present and matching in the Expected Identification, the Anybus state is set to PROCESS_ACTIVE. However, if there is any mismatch among the other submodules the LEDs will indicate configuration mismatch.

See also...

- The Remap_ADI_Write_Area and Remap_ADI_Read_Area commands in the Application Data Object (FEh), found in Anybus CompactCom 40 Software Design Guide.

4.5 Diagnostics

4.5.1 Standard Diagnostics

PROFINET IO uses alarms when informing the IO Controller of diagnostic entries. In the Anybus implementation, it is possible for the application to create alarms via diagnostic entries by means of the Diagnostic Object (02h).

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

Creating a diagnostic instance is done by issuing the command Create. If the module is in state IDLE or PROCESS_ACTIVE, the created instance will be communicated on the network as an “appear”-alarm. If the module is in another state, the PLC will be notified in the connect response by a module diff block.

Deleting a diagnostic instance is done by issuing the command Delete. This will trigger a “disappear”-alarm on the network. Supply the instance ID that was returned by the create-command.

Every diagnostic instance has a severity level and an event code associated to it. Major unrecoverable events will cause the module to disconnect itself from the network, thus preventing network participation. Other severity levels either produce a Channel Diagnostic alarm or a Generic Diagnostic alarm, depending on the Event Code, according to the table below.

Severity	Event code != network specific	Event code = network specific
Minor, recoverable	Channel Diagnostic Alarm	Generic Diagnostic Alarm (See Extended Diagnostics, p. 24)
Minor, unrecoverable		
Major, recoverable		
Major, unrecoverable	Anybus enters Exception state	



Process alarms can not be created.

See also..

- [Diagnostic Object \(02h\), p. 71](#)

4.5.2 Extended Diagnostics

Using the network specific event code (FFh) creates a Generic Diagnostic Alarm on the network. This type of alarm can carry extended diagnostic information and more details about the source of the problem.

Generic Diagnostic Alarm instances can be tagged with a source API and slot- and subslot number, and can also contain additional network specific diagnostic data.

For more information, see [Details: Network Specific Data, p. 73](#).

4.6 Identification & Maintenance (I&M)

4.6.1 General Information

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

The application should provide application specific I&M0 information during start-up. See [PROFINET IO Object \(F6h\), p. 131](#) for more information.

It is possible for the application to handle I&M records. Activate this using the IM_Options command. See [Network PROFINET IO Object \(0Eh\), p. 82](#) for more information.

Default I&M0 information:

IM Manufacturer ID	010Ch (HMS Industrial Networks)
IM Order ID	"ABCC40-PIR"
IM Serial Number	(unique serial number, set during manufacturing)
IM Hardware Revision	(Anybus hardware revision ID, set during manufacturing)
IM Software Revision	(Anybus software revision, set during manufacturing)
IM Revision Counter	(Revision counter)
IM Profile ID	F600h (Generic Device)
IM Profile Specific Type	0004h (No profile)
IM Version	0101h
IM Supported	For submodules belonging to a "non-zero API", the returned value is zero. For submodules belonging to API 0, the returned value is 000Eh (IM0-3 supported)

4.6.2 I&M Data Structures

The I&M records uses the following data structures.

Record	Content	Size	Description
I&M0	Manufacturer Id	2 bytes	PROFINET IO Object (F6h), attribute #2 ('Vendor ID/I&M Vendor ID')
	Order Id	20 bytes	PROFINET IO Object (F6h), attribute #8 ('I&M Order ID')
	Serial number	16 bytes	PROFINET IO Object (F6h), attribute #9 ('I&M Serial number')
	Hardware revision	2 bytes	PROFINET IO Object (F6h), attribute #10 ('I&M Hardware revision')
	Software revision	4 bytes	PROFINET IO Object (F6h), attribute #11 ('I&M Software revision')
	Revision counter	2 bytes	PROFINET IO Object (F6h), attribute #12 ('I&M Revision counter')
	Profile Id	2 bytes	PROFINET IO Object (F6h), attribute #13 ('I&M Profile ID')
	Profile specific type	2 bytes	PROFINET IO Object (F6h), attribute #14 ('I&M Profile specific type')
	IM version	2 bytes	0101h (Internal, constant value)
	IM supported	2 bytes	For submodules belonging to a "non-zero API", the returned value is zero. For submodules belonging to API 0, the returned value is 000Eh (IM0-3 supported)
I&M1	Tag Function	32 bytes	Default: All bytes set to blanks (' ')
	Tag Location	22 bytes	Default: All bytes set to blanks (' ')
I&M2	Installation date	16 bytes	Default: All bytes set to blanks (' ')
I&M3	Descriptor	54 bytes	Default: All bytes set to blanks (' ')

See also..

- [PROFINET IO Object \(F6h\), p. 131](#)

4.7 Fast Start Up

4.7.1 General Information

The Fast Start Up (FSU) function enables PROFINET IO devices, connected to the network, to power up quickly. This is useful in, for example, robot applications, where rapid retooling is necessary. With FSU activated, the module will send a DCP Hello message as soon as possible after power-on.

This function is enabled by two GSD keywords: PowerOnToCommReady and DCP_HelloSupported. The activation is made from the PLC configuration tool.

The FSU time is defined as the number of milliseconds (ms) from hardware reset (or power-on) until the module enters the PROCESS_ACTIVE state. On PROFINET, it is recommended to try to reach a FSU time <= 500 ms.

To enable FSU, set values according to the following (listed for the Device Access Point(s)):

PowerOnToComm-Ready	FSU time, in milliseconds (ms). This value must be measured by the customer.
DCP_HelloSupported	Value: true.

- 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**.

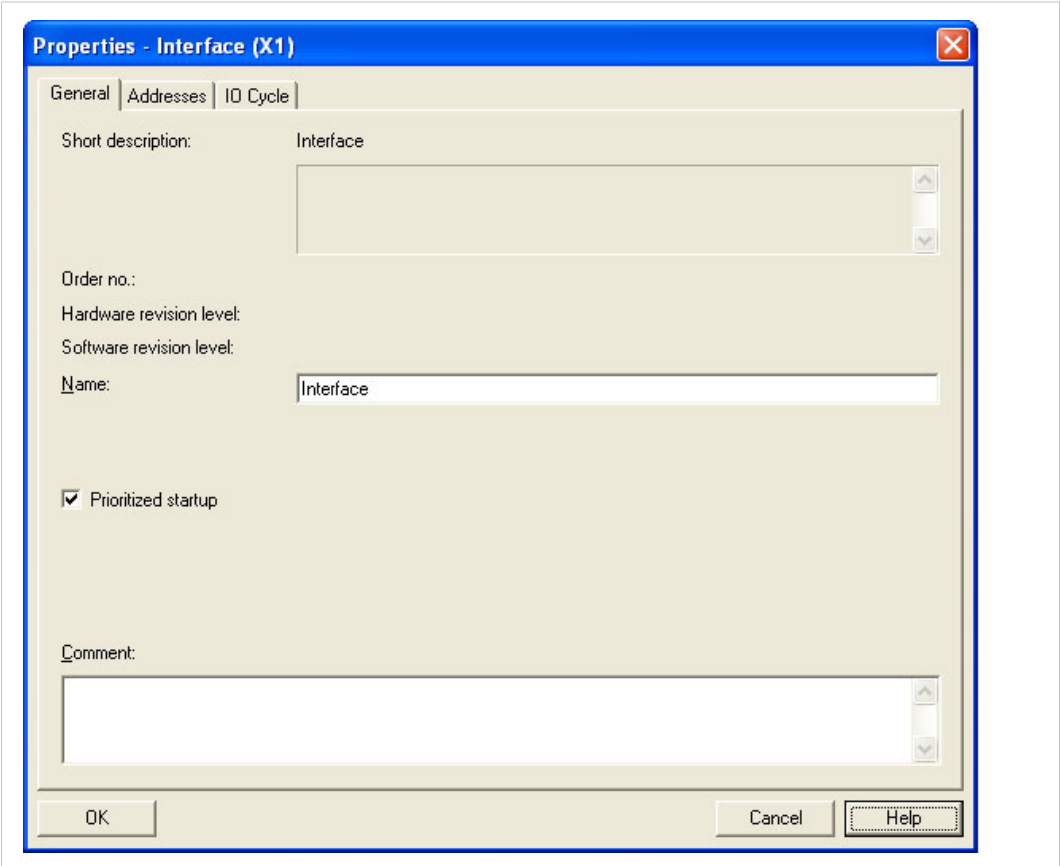


Fig. 4

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

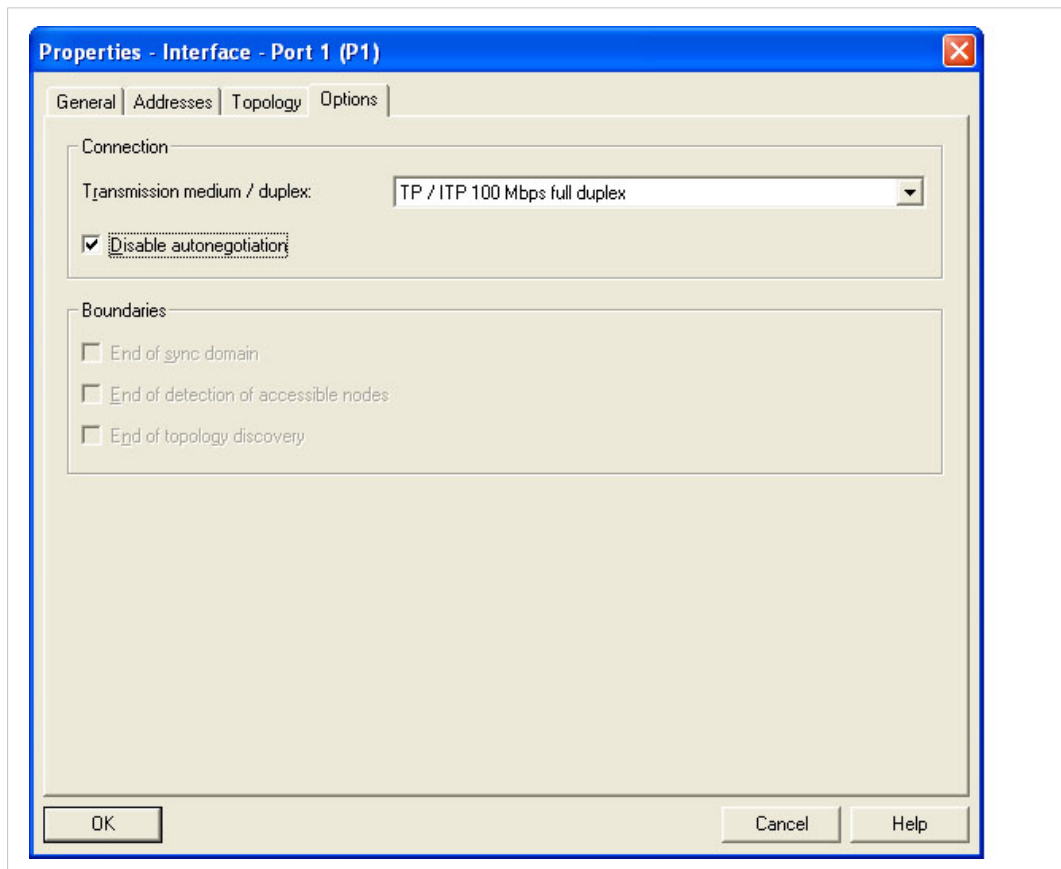


Fig. 5

- Repeat for Port 2.

4.8 Address Conflict Detection (ACD)

The Anybus CompactCom 40 PROFINET IRT supports Address Conflict Detection (ACD). This mechanism involves the following two aspects:

- Initial probing: before using an IP address, the module issues ARP probes to see if the address is already in use (three probes with a 100 ms delay).
- Address announcement: after the initial probing, the module issues ARP announcements.

If an IP address conflict is detected, IP address error will be indicated on the Network Status LED. The module will use address 0.0.0.0. A new address can be configured via the Anybus IP-config tool.



If Fast Start Up is used, ACD initial probing is automatically disabled to ensure a fast startup. Address announcement is still used, as it will not affect the actual startup time.

To enable/disable ACD, see [Ethernet Host Object \(F9h\), p. 149](#).

4.9 PROFIsafe

The Anybus CompactCom 40 PROFINET IRT supports the PROFIsafe profile. This profile makes it possible for a user to send data on a black channel interface, i.e. a safe channel over PROFINET using an add on Safety Module, e.g. the IXXAT Safe T100/PS. For more information about this module, see IXXAT Safe T100 Manual, available at www.ixxat.com.

In both general and advanced mode, the Safety Module shall be located in slot 1. The host application can specify the highest 16 bits of the module ID of the Functional Safety Module.

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

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

See also...

[Functional Safety Module Object \(11h\), p. 122](#)

[Functional Safety Object \(E8h\), p. 152](#)

[PROFINET IO Object \(F6h\), p. 131](#)

Command details: Add_Safety_Module in *[Network PROFINET IO Object \(0Eh\), p. 82](#)* .

[Anybus CompactCom Hardware Design Guide](#)

5 FTP Server

5.1 General Information

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

disabled using attribute #6 in the Ethernet Host Object (F9h).

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

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

The FTP server supports up to two concurrent clients.

5.2 User Accounts

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

File Format:

```
User1:Password1:Homedirectory1
User2:Password2:Homedirectory2
User3:Password3:Homedirectory3
```

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

File Format (ftp.cfg):

```
User1:Password1:Homedirectory1
User2:Password2:Homedirectory2
.
.
UserN:PasswordN:HomedirectoryN
\path\userlistA:HomedirectoryA
\path\userlistB:HomedirectoryB
```

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

File Format:

```
User1:Password1
User2:Password2
User3:Password3
.
.
UserN:PasswordN
```

Notes:

- Usernames must not exceed 16 characters in length.
- Passwords must not exceed 16 characters in length.
- Usernames and passwords must only contain alphabetic characters and/or numbers.

- If `\ftp.cfg` is missing or cannot be interpreted, all username/password combinations will be accepted and the home directory will be the FTP root (i.e. `\ftp\`).
- The home directory for a user must also exist in the file system, if the user shall be able to log in. It is not enough just to add the user information to the `ftp.cfg` file.
- If Admin Mode has been enabled in the Ethernet Object, all username/password combinations will be accepted and the user will have unrestricted access to the file system (i.e. the home directory will be the system root). The `vfs` folder is read-only.
- It is strongly recommended to have at least one user with root access (`\`) permission. If not, Admin Mode must be enabled each time a system file needs to be altered (including `\ftp.cfg`).

5.3 Session Example

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

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

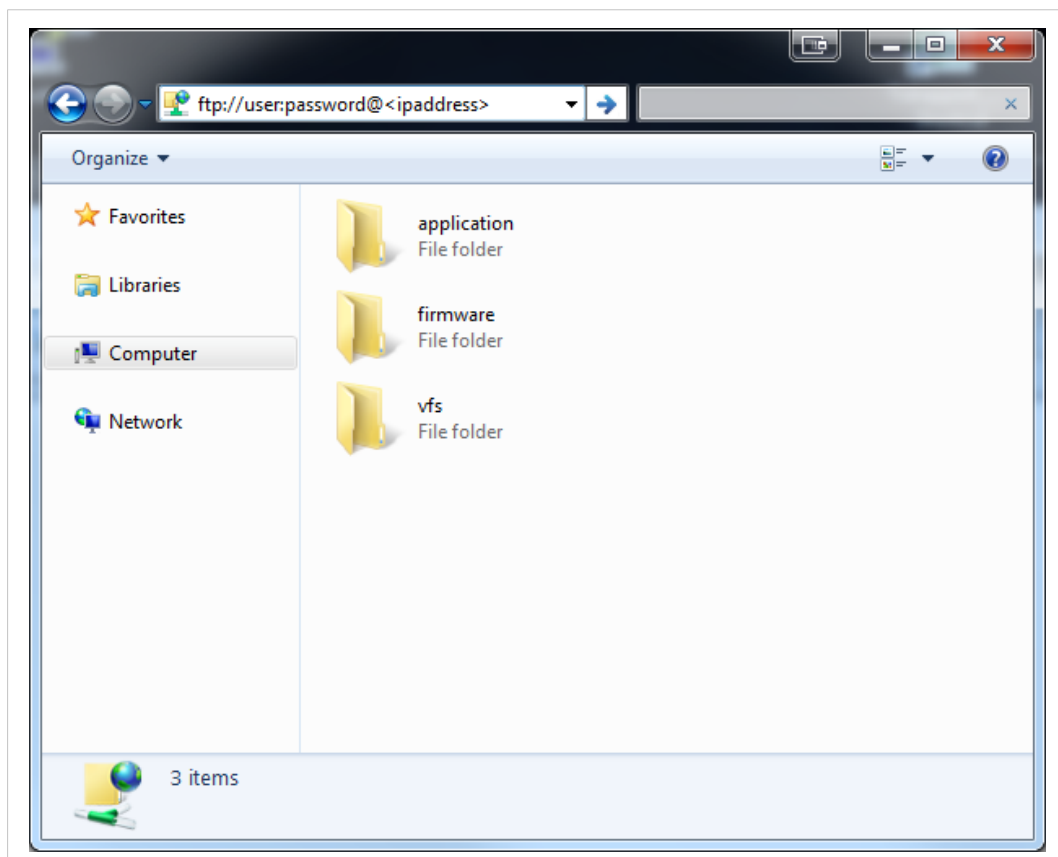


Fig. 6

6 Web Server

6.1 General Information

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

The web interfaces are stored in the file system, which can be accessed through the FTP server. If necessary, the web server can be completely disabled in the Ethernet Host Object (F9h).

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

See also...

- [FTP Server, p. 31](#)
- [Server Side Include \(SSI\), p. 40](#)
- [JSON, p. 57](#)
- [Ethernet Host Object \(F9h\), p. 149](#)

6.2 Default Web Pages

The default web pages provide access to:

- Network configuration parameters
- Network status information
- Access to the host application ADIs

The default web pages are built of files stored in a virtual file system accessible through the vfs folder. These files are read only and cannot be deleted or overwritten. The web server will first look for a file in the web root folder. If not found it will look for the file in the vfs folder, making it appear as the files are located in the web root folder. By loading files in the web root folder with exactly the same names as the default files in the vfs folder, it is possible to customize the web pages, replacing such as pictures, logos and style sheets.

If a complete customized web system is designed and no files in the vfs folder are to be used, it is recommended to turn off the virtual file system completely, see the File System Interface Object.

See also ...

- [File System, p. 17](#)
- [File System Interface Object \(0Ah\), p. 121](#)

6.2.1 Network Configuration

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

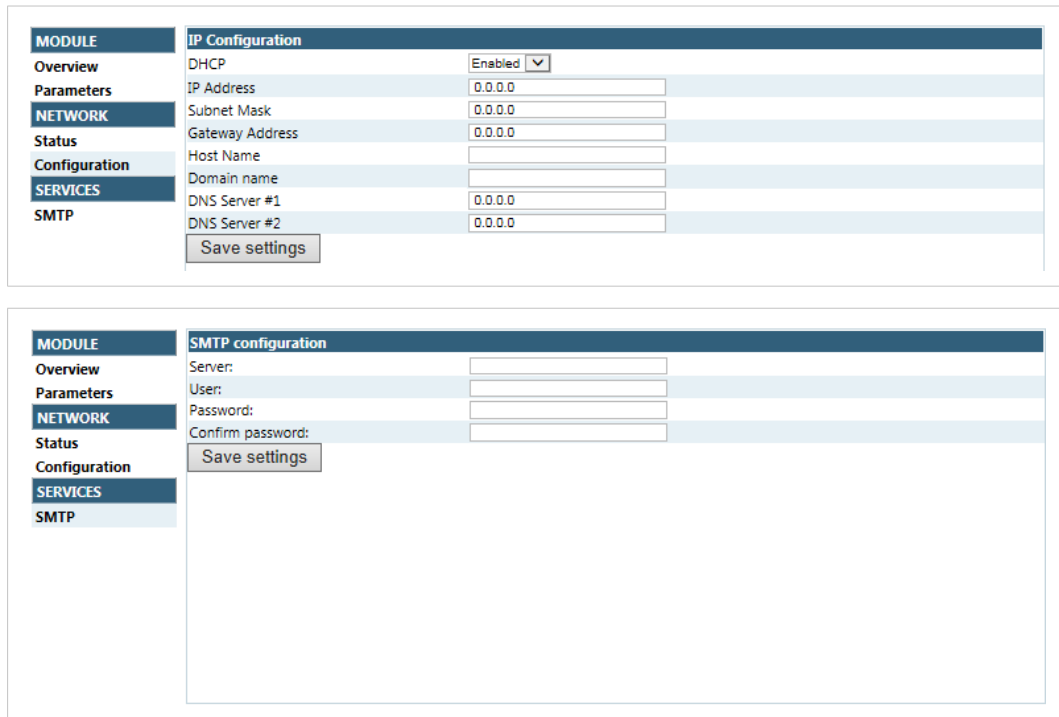


Fig. 7

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

Available IP Configuration Settings

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

Available SMTP Settings

Name	Description
Server	IP address or name Max 64 characters
User	Max 64 characters
Password	Max 64 characters

6.2.2 Ethernet Statistics Page

The Ethernet statistics web page contains the following information:

Current IP Configuration	Description
DHCP:	-
Host Name:	-
IP Address:	-
Subnet Mask:	-
Gateway Address:	-
DNS Server #1:	-
DNS Server #2:	-
Domain Name:	-

Current Ethernet Configuration	Description	
MAC Address	-	
Port 1	Speed:	The current link speed.
	Duplex:	The current duplex configuration.
Port 2	Speed:	The current link speed.
	Duplex:	The current duplex configuration.

Interface Counters	Description
In Octets:	Received bytes.
In Ucast Packets:	Received unicast packets.
In NUCast packets:	Received non-unicast packets (broadcast and multicast).
In Discards:	Received packets discarded due to no available memory buffers.
In Errors:	Received packets discarded due to reception error.
In Unknown Protos:	Received packets with unsupported protocol type.
Out Octets:	Sent bytes.
Out Ucast packets:	Sent unicast packets.
Out NUCast packets:	Sent non-unicast packets (broadcast and multicast).
Out Discards:	Outgoing packets discarded due to no available memory buffers.
Out Errors:	Transmission errors.

Media Counters	Description
Alignment Errors	Frames received that are not an integral number of octets in length.
FCS Errors	Frames received that do not pass the FCS check.
Single Collisions	Successfully transmitted frames which experienced only one collision.
Multiple Collisions	Successfully transmitted frames that experienced more than one collision.
SQE Test Errors	Number of times SQE test error messages are generated.
Deferred Transmissions	Frames for which first transmission attempt is delayed because the medium is busy.
Late Collisions	Number of times a collision is detected later than 512 bit-times into the transmission of a packet.
Excessive Collisions	Frames for which a transmission fails due to excessive collisions.
MAC Receive Errors	Frames for which reception of an interface fails due to an internal MAC sublayer receive error.
MAC Transmit Errors	Frames for which transmission fails due to an internal MAC sublayer receive error.
Carrier Sense Errors	Times that the carrier sense condition was lost or never asserted when attempted to transmit a frame.
Frame Size Too Long	Frames received that exceed the maximum permitted frame size.
Frame Size Too Short	Frames received that are shorter than lowest permitted frame size.

Fiber Optical Statistics (only available for the Anybus CompactCom 40 PROFINET IRT)	Description
Port 1 Temperature (C):	Current temperature of port 1 transceiver, in degrees Celsius.
Port 1 Power Budget (dB):	Current received power budget for port 1 transceiver, in dB.
Port 1 Power Budget Status:	Textual display of the power budget status for port 1: "OK": Power budget status > 2 dB "Maintenance required": 0 dB < power budget status < 2 dB "Maintenance demanded": Power budget = 0 dB
Port 2 Temperature (C):	Current temperature of port 2 transceiver, in degrees Celsius.
Port 2 Power Budget (dB):	Current received power budget for port 2 transceiver, in dB.
Port 2 Power Budget Status:	Textual display of the power budget status for port 2: "OK": Power budget status > 2 dB "Maintenance required": 0 dB < power budget status < 2 dB "Maintenance demanded": Power budget = 0 dB

6.3 Server Configuration

6.3.1 General Information

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

```
File Format:
  [WebRoot]
  \web

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

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

Web Root Directory [WebRoot]	The web server cannot access files outside this directory.
Content Types [FileTypes]	A list of file extensions and their reported content types. See also... Default Content Types, p. 37
SSI File Types [SSIFileTypes]	By default, only files with the extension 'shtm' are scanned for SSI. Additional SSI file types can be added here as necessary.

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

6.3.2 Index page

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

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



Substitute <WebRoot> with the web root directory specified in \http.cfg.

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

See also ...

- [Default Web Pages, p. 33](#)

6.3.3 Default Content Types

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

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

Content types can be added or redefined by adding them to the server configuration file.

6.3.4 Authorization

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

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.

```
File Format:
  Username1:Password1
  Username2:Password2
  ...
  UsernameN:PasswordN

  [AuthName]
  (message goes here)
```

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



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

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

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

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


7 E-mail Client

7.1 General Information

The built-in e-mail client allows the application to send e-mail messages through an SMTP-server. Messages can either be specified directly in the SMTP Client Object (04h), or retrieved from the file system. The latter may contain SSI, however note that for technical reasons, certain commands cannot be used (specified separately for each SSI command).

The client supports authentication using the 'LOGIN' method. Account settings etc. are stored in the Network Configuration Object (04h).

7.2 How to Send E-mail Messages

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

This includes:

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

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

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

Sending a message based on a file in the file system is achieved using the Send Email from File command. This command is described in the SMTP Client Object (04h).

8 Server Side Include (SSI)

8.1 General Information

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

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

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

8.2 Include File

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



This function cannot be used in e-mail messages.

Syntax:

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

filename: Source file

Scenario	Default Output
Success	(contents of file)

8.3 Command Functions

8.3.1 General Information

Command functions executes commands and includes the result.

General Syntax

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

command: Command function, see below

Command Functions

Command	Valid for E-mail Messages	Page
GetConfigItem()	Yes	
SetConfigItem()	No	
SsiOutput()	Yes	
DisplayRemoteUser	No	
ChangeLanguage()	No	
IncludeFile()	Yes	
SaveDataToFile()	No	
printf()	Yes	
scanf()	No	

8.3.2 GetConfigItem()

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

File Format

The source file must have the following format:

```
[key1]
value1

[key2]
value2

...

[keyN]
valueN
```

Syntax:

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

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

Default Output

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

Example

The following SSI...

```
<?--exec cmd_argument='GetConfigItem("\example.cnf", "B")'-->
```


... in combination with the following file (\example.cnf)...

```
[A]
First
[B]
Second
[C]
Third
```

... returns the string 'Third'.

8.3.3 SetConfigItem()

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

 *This function cannot be used in e-mail messages.*

File Format

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

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

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

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

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

 *Form objects with names starting with underscore will not be stored.*

Syntax:

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

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

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

Default Output

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

Example

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

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

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

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

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

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

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

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

</BODY>
</HTML>
```

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

```
[Name]
Cliff Barnes

[Food]
Cheese

[Drink]
Beer
```



In order for this example to work, the HTML file must be named "test.shtm".

8.3.4 SsiOutput()

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

Syntax:

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

success: String to use in case of success
failure: String to use in case of failure

Default Output

(this command produces no output on its own)

Example

The following example illustrates how to use this command.

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

See also...

- [SSI Output Configuration, p. 56](#)

8.3.5 DisplayRemoteUser

This command stores returns the username on an authentication session.



This command cannot be used in e-mail messages.

Syntax:


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

Default Output

Scenario	Default Output
Success	(current user)

8.3.6 ChangeLanguage()

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

 *This function cannot be used in e-mail messages.*

Syntax:

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

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

The passed value must be a single digit as follows:

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

Default Output

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

Example

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


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

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

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

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

</BODY>
</HTML>
```

 *In order for this example to work, the HTML file must be named "test.shtm".*

8.3.7 IncludeFile()

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

Syntax:

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

filename: Source file
separator: Optional; specifies line separation characters (e.g. "
").

Default Output

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

Example

The following example demonstrates how to use this function.

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

Contents of 'info.txt':

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

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

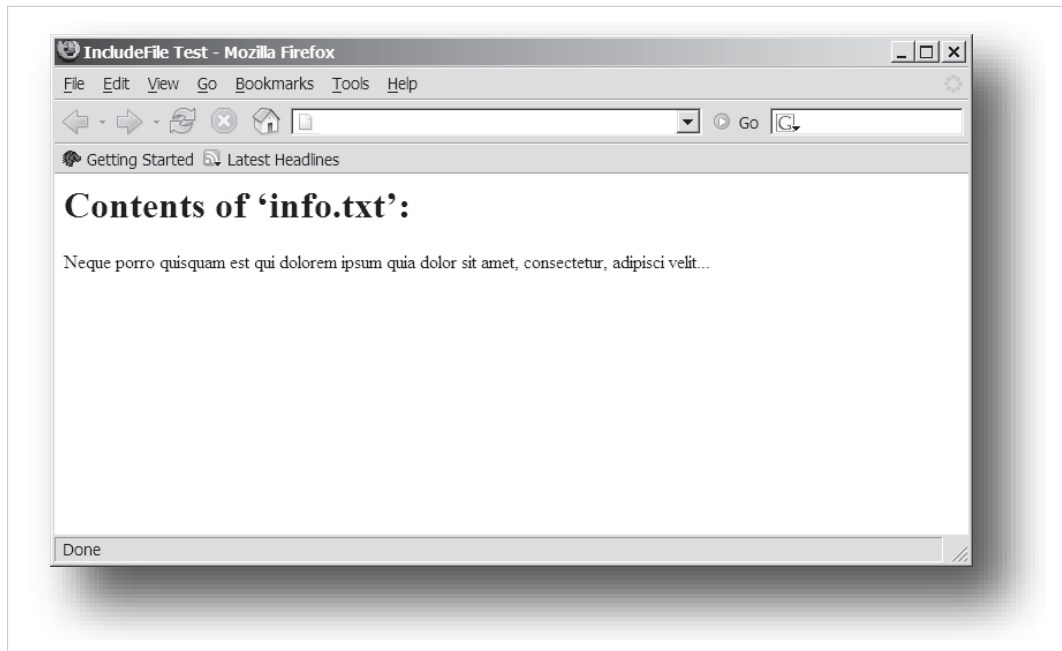



Fig. 8

See also...

- [Include File, p. 40](#)

8.3.8 SaveDataToFile()

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

 This function cannot be used in e-mail messages.

Syntax:

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

filename	Destination file. If the specified file does not exist, it will be created (provided that the path is valid).
source:	Optional; by specifying a form object, only data from that particular form object will be stored. Default behavior is to store data from all form objects except the ones where the name starts with underscore.
Overwrite Append	Specifies whether to overwrite or append data to existing files.

Default Output

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

Example

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

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

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

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

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

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

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

</BODY>
</HTML>
```

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



In order for this example to work, the HTML file must be named "test.shtml".

8.3.9 printf()

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

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

Syntax:

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

template:	Template which determines how the arguments shall be represented. May contain any number of formatting tags which are substituted by subsequent arguments and formatted as requested. The number of format tags must match the number of arguments; if not, the result is undefined. See section "Formatting Tags" below for more information.
argument:	Source arguments; optional parameters which specify the actual source of the data that shall be inserted in the template string. The number of arguments must match the number of formatting tags; if not, the result is undefined. At the time of writing, the only allowed argument is ABCCMessage(). See also... <ul style="list-style-type: none"> • ABCCMessage(), p. 53

Default Output

Scenario	Default Output
Success	(printf() result)
ABCCMessage error	ABCCMessage error string (Errors, p. 55)

Example

See ..

- [ABCCMessage\(\), p. 53](#)
- [Example \(Get_Attribute\);, p. 54](#)

Formatting Tags

Formatting tags are written as follows:

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

- Type (Required)

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

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

- Flags (Optional)

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

- Width (Optional)

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

- Precision (Optional)

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

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

- Modifier

Modifier Character	Meaning
hh	Argument is interpreted as SINT8 or UINT8
h	Argument is interpreted as SINT16 or UINT16
L	Argument is interpreted as SINT32 or UINT32

8.3.10 scanf()

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

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



This command cannot be used in e-mail messages.

Syntax:

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

source	Name of the HTML form object from which the string shall be extracted.
template:	Template which specifies how to parse and interpret the data. May contain any number of formatting tags which determine the conversion prior to sending the data to subsequent arguments. The number of formatting tags must match the number of arguments; if not, the result is undefined. See section "Formatting Tags" below for more information.
argument:	Destination argument(s) specifying where to send the interpreted data. The number of arguments must match the number of formatting tags; if not, the result is undefined. At the time of writing, the only allowed argument is ABCCMessage(). See also... <ul style="list-style-type: none"> • ABCCMessage(), p. 53

Default Output

Scenario	Default Output
Success	"Success"
Parsing error	"Incorrect data format"
Too much data for argument	"Too much data"
ABCCMessage error	ABCCMessage error string (Errors, p. 55)

Example

See also...

[ABCCMessage\(\), p. 53](#)

[Example \(Set_Attribute\);, p. 55](#)

Formatting Tags

Formatting tags are written as follows:

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

- Type (Required)

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

Type	Input	Argument Data Type
c	Single character	CHAR
d	Accepts a signed decimal integer	SINT8 SINT16 SINT32
i	Accepts a signed or unsigned decimal integer. May be given as decimal, hexadecimal or octal, determined by the initial characters of the input data: Initial Characters: Format: 0x Hexadecimal 0: Octal 1... 9: Decimal	SINT8/UINT8 SINT16/UINT16 SINT32/UINT32
u	Accepts an unsigned decimal integer.	UINT8 UINT16 UINT32
o	Accepts an optionally signed octal integer.	SINT8/UINT8 SINT16/UINT16 SINT32/UINT32
x, X	Accepts an optionally signed hexadecimal integer.	SINT8/UINT8 SINT16/UINT16 SINT32/UINT32
e, E, f, g, G	Accepts an optionally signed floating point number. The input format for floating-point numbers is a string of digits, with some optional characteristics: <ul style="list-style-type: none"> – It can be a signed value – It can be an exponential value, containing a decimal rational number followed by an exponent field, which consists of an 'E' or an 'e' followed by an integer. 	FLOAT
n	Consumes no input; the corresponding argument is an integer into which scanf writes the number of characters read from the object input.	SINT8/UINT8 SINT16/UINT16 SINT32/UINT32
s	Accepts a sequence of nonwhitespace characters	STRING
[scanset]	Accepts a sequence of nonwhitespace characters from a set of expected bytes specified by the scanlist (e.g '[0123456789ABCDEF]') A literal '[' character can be specified as the first character of the set. A caret character (^) immediately following the initial '[' inverts the scanlist, i.e. allows all characters except the ones that are listed.	STRING
%	Accepts a single '%' input at this point; no assignment or conversion is done. The complete conversion specification should be '%%'.	-

- * (Optional)

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

- Width (Optional)

Specifies the maximum number of characters to be read

- Modifier (Optional)

Specifies a different data size.

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

8.4 Argument Functions

8.4.1 General Information

Argument functions are supplied as parameters to certain command functions.

General Syntax:

(Syntax depends on context)

Argument Functions:

Function	Description
ABCCMessage()	-

8.4.2 ABCCMessage()

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

Syntax

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

object	Specifies the Destination Object
instance	Specifies the Destination Instance
command	Specifies the Command Number
ce0	Specifies CmdExt[0] for the command message
ce1	Specifies CmdExt[1] for the command message
msgdata	Specifies the actual contents of the MsgData[] subfield in the command <ul style="list-style-type: none"> • Data can be supplied in direct form (format depends on c_type) • The keyword "ARG" is used when data is supplied by the parent command (e.g. scanf()).
c_type:	Specifies the data type in the command (msgdata), see below.
r_type:	Specifies the data type in the response (msgdata), see below.

Numeric input can be supplied in the following formats:

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

- Command Data Types (c_type)

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

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

- Response Data Types (r_type)

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

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



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

Example (Get_Attribute):

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


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

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

Example (Set_Attribute):

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

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

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

Errors

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

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

Error Code	Output
16	"Segmentation buffer overflow"
17... 255	"Unknown error"

See also...

[SSI Output Configuration, p. 56](#)

8.5 SSI Output Configuration

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

File format:

[ABCCMessage_X] 0:"Success string" 1:"Error string 1" 2:"Error string 2" ... 16:"Error string 16"	Each error code corresponds to a dedicated output string, labelled from 1 to 16. See Errors, p. 55
[GetConfigItem_X] 0: "Success string" 1:"Authentication error string" 2:"File open error string" 3:"Tag not found string"	Use "%s" to include the name of the file.
[SetConfigItem_X] 0: "Success string" 1:"Authentication error string" 2:"File open error string" 3:"File write error string"	Use "%s" to include the name of the file.
[IncludeFile_X] 0: "Success string" 1:"Authentication error string" 2:"File read error string"	Use "%s" to include the name of the file.
[scanf_X] 0: "Success string" 1:"Parsing error string"	-
[ChangeLanguage_X] 0: "Success string" 1:"Change error string"	-

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

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

See also...

•

[SsiOutput\(\), p. 44](#)

9 JSON

9.1 General Information

JSON is an acronym for JavaScript Object Notation and an open standard format for storing and exchanging data in an organized and intuitive way. It is used as an alternative to XML, to transmit data objects consisting of attribute - value pairs between a server and a web application. JavaScripts are used to create dynamic web pages to present the values.

JSON is more versatile than SSI in that you not only can change the values on a web page, but also the size and the look of the web page dynamically. A simple example of how to create a web page is added at the end of this chapter.

JSON requests shall be UTF-8 encoded. The module will interpret JSON requests as UTF-8 encoded, while all other HTTP requests will be interpreted as ISO-8859-1 encoded. All JSON responses, sent by the module, are UTF-8 encoded, while all other files sent by the web server are encoded as stored in the file system.

9.1.1 Access

The JSON resources should be password protected. Add password protection by adding a file called `web_accs.cfg` in the root directory.

9.2 JSON Objects

9.2.1 ADI

info.json

GET `adi/info.json[?callback=<function>]`.

This object holds data common to all ADIs that are static during runtime. Optionally, a callback may be passed to the GET-request for JSONP output.

Name	Data Type	Note
<code>dataformat</code>	Number	0 = Little endian 1 = Big endian (Affects value, min and max representations)
<code>numadis</code>	Number	Total number of ADIs
<code>webversion</code>	Number	Web/JSON API version

JSON object layout:

```
{
  "dataformat": 0,
  "numadis": 123,
  "webversion": 1
}
```

data.json

GET adi/data.json?offset=<offset>&count=<count>[&callback=<function>].

This object call fetches values for up to <count> ADIs, starting from <offset> in a list sorted by ADI order number. The values may change at any time during runtime. Optionally, a callback may be passed to the GET-request for JSONP output.

JSON object layout:

```
[
  "FF",
  "A201",
  "01FAC105"
]
```

metadata.json

GET adi/metadata.json?offset=<offset>&count=<count>[&callback=<function>].

This object call fetches metadata for up to <count> ADIs, starting from <offset> in a list sorted by ADI order number. This data is static during runtime. Optionally, a callback may be passed to the GET-request for JSONP output.

Name	Data Type	Note
instance	Number	-
name	String	May be NULL if no name is present.
numelements	Number	-
datatype	Number	-
min	String	Minimum value. May be NULL if no minimum value is present.
max	String	Maximum value. May be NULL if no maximum value is present.
access	Number	Bit 0: Read accessBit 1: Write access

JSON object layout:

```
[
  {
    "instance": 1,
    "name": "Temperature threshold",
    "numelements": 1,
    "datatype": 0,
    "min": "00",
    "max": "FF",
    "access": 0x03
  },
  {
    "nine more..."
  }
]
```

enum.json

GET adi/enum.json?inst=<instance>[&value=<element>][&callback=<function>].

This object call fetches enum strings for the instance <instance>. If an <element> is specified, only the enum string for that value is returned. If no enum strings are available, an empty list is returned. Optionally, a callback may be passed to the GET-request for JSONP output.

Name	Data Type	Note
string	String	-
value	Number	-

JSON object layout:

```
[
  {
    "string": "String for value 1",
    ...."value": 1
  },
  {
    "string": "String for value 1",
    ...."value": 1
  },
  ...
]
```

update.json

POST adi/update.json - form data:

inst=<instance>&value=<data>[&elem=<element>][&callback=<function>].

Updates the value of an ADI for the specified ADI instance <instance>. The value, <data>, shall be hex formatted (see [Hex Format Explained, p. 63](#) for more information). If <element> is specified, only the value of the specified element is updated. In this case, <data> shall only update that single element value. When <element> is not specified, <data> shall represent the entire array value. Optionally, a callback may be passed to the request for JSONP output

Name	Data Type	Note
result	Number	0 = success

POST adi/update.json - form data: inst=15&value=FF01

```
{
  "result" : 0
}
```

9.2.2 Module

info.json

GET module/info.json

Name	Data Type	Note
modulename	String	-
serial	String	32 bit hex ASCII
fwver	Array of Number	(major, minor, build)

Name	Data Type	Note
uptime	Array of Number	[high, low] milliseconds (ms)
cpuload	Number	CPU load in %

JSON object layout:

```
{
  "modulename": "ABCC M40",
  "serial": "ABCDEF00",
  "fwver": [ 1, 5, 0 ],
  "uptime": [ 5, 123456 ],
  "cpuload": 55
}
```

9.2.3 Network

ethstatus.json

GET network/ethstatus.json.

Name	Data Type	Note
mac	String	6 byte hex
comm1	Object	See object definition in the table below
comm2	Object	See object definition in the table below

Comm Object Definition:

Name	Data Type	Note
link	Number	0: No link 1: Link
speed	Number	0: 10 Mbit 1: 100 Mbit
duplex	Number	0: Half 1: Full

JSON object layout:

```
{
  "mac": "003011FF0201",
  "comm1": {
    "link": 1,
    "speed": 1,
    "duplex": 1
  },
  "comm2": {
    "link": 1,
    "speed": 1,
    "duplex": 1
  },
  ...
}
```

ipstatus.json & ipconf.json

These two object share the same data format. The object ipconf.json returns the configured IP settings, and ipstatus.json returns the actual values that are currently used. ipconf.json can also be used to alter the IP settings.

GET network/ipstatus.json, or GET network/ipconf.json.

Name	Data Type	Note
dhcp	Number	-
addr	String	-
subnet	String	-
gateway	String	-
dns1	String	-
dns2	String	-
hostname	String	-
domainname	String	-

```
{
  "dhcp":      0,
  "addr":      "192.168.0.55",
  "subnet":    "255.255.255.0",
  "gateway":   "192.168.0.1",
  "dns1":      "10.10.55.1",
  "dns2":      "10.10.55.2",
  "hostname":  "<hostname>",
  "domainname": "hms.se"
}
```

To change IP settings, use network/ipconf.json. It accepts any number of arguments from the list above. Values should be in the same format.

Example:

GET ipconf.json?dhcp=0&addr=10.11.32.2&hostname=abcc123&domainname=hms.se

ethconf.json

GET network/ethconf.json

Name	Data Type	Note
comm1	Number	-
comm2	Number	-

ifcounters.json

GET network/ifcounters.json?port=<port>. The argument <port> is either 1 or 2.

Name	Data Type	Note
inoctets	Number	IN: bytes
inucast	Number	IN: unicast packets
innucast	Number	IN: broadcast and multicast packets
indiscards	Number	IN: discarded packets
inerrors	Number	IN: errors
inunknown	Number	IN: unsupported protocol type
outoctets	Number	OUT: bytes
outucast	Number	OUT: unicast packets
outnucast	Number	OUT: broadcast and multicast packets
outdiscards	Number	OUT: discarded packets
outerrors	Number	OUT: errors

mediacounters.json

GET network/mediacounters.json?port=<port>. The argument <port> is either 1 or 2.

Name	Data Type	Note
align	Number	Frames received that are not an integral number of octets in length
fcs	Number	Frames received that do not pass the FCS check
singlecoll	Number	Successfully transmitted frames which experienced exactly one collision
multicoll	Number	Successfully transmitted frames which experienced more than one collision
latecoll	Number	Number of collisions detected later than 512 bit times into the transmission of a packet
excesscoll	Number	Frames for which transmissions fail due to excessive collisions
sqetest	Number	Number of times SQE test error is generated
deferredtrans	Number	Frames for which the first transmission attempt is delayed because the medium is busy
macrecerr	Number	Frames for which reception fails due to an internal MAC sublayer receive error
mactranserr	Number	Frames for which transmission fails due to an internal MAC sublayer transmit error
cserr	Number	Times that the carrier sense was lost or never asserted when attempting to transmit a frame
toolong	Number	Frames received that exceed the maximum permitted frame size
tooshort	Number	Frames received that are shorter than the lowest permitted frame size

nwstats.json

GET network/nwstats.json.

This object lists available statistics data. The data available depends on the product.

Example output:

```
[
  or
  [ { "identifier": "eip", "title": "EtherNet/IP Statistics" } ]
  or
  [
    { "identifier": "bacnet", "title": "BACnet/IP Statistics" },
    { "identifier": "bacnetae", "title": "BACnet Alarm and Event" },
    { "identifier": "bacnetapl", "title": "BACnet APL Statistics" }
  ]
]
```

Get network specific statistics:

GET network/nwstats.json?get=<ID>. <ID> is an "identifier" value returned from the previous command ("eip", for example)

```
[
  { "name": "Established Class1 Connections", "value": 0 },
  { "name": "Established Class3 Connections", "value": 1 }
]
```


9.2.4 Services

smtp.json

GET services/smtp.json.



Password is not returned when retrieving the settings.

Name

Data Type

Note

server

String

-

user

String

-

9.2.5 Hex Format Explained

The metadata max and min fields and the ADI values are ABP data encoded in a hex format. If the data type is an integer, the endianness used is determined by the data format field found in `adi/info.json`.

Examples:

The value 5 encoded as a UINT16, with data format = 0 (little endian):

```
0500
```

The character array "ABC" encoded as CHAR[3] (data format is not relevant for CHAR):

```
414243
```

9.3 Example

This example shows how to create a web page that fetches Module Name and CPU load from the module and presents it on the web page. The file, containing this code, has to be stored in the built-in file system, and the result can be seen in a common browser.

```
<html>
  <head>
    <title>Anybus CompactCom</title>

    <!-- Imported libs -->
    <script type="text/javascript" src="vfs/js/jquery-1.9.1.js"></script>
    <script type="text/javascript" src="vfs/js/tmpl.js"></script>
  </head>
  <body>
    <div id="info-content"></div>
    <script type="text/x-tmpl" id="tmpl-info">
      <b>From info.json</b><br><br>
      Module name:
      {%=o.modulename%}<br>

      CPU Load:
```

```
        {%=o.cpuload%}%<br>
    </script>
    <script type="text/javascript">
        $.getJSON( "/module/info.json", null, function(data){
            $("#info-content").html( tmpl("tmpl-info", data ) );
        });
    </script>
</body>
</html>
```

10 SNMP Agent

10.1 General

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

10.2 Management Information (MIB)

A MIB is a device database that is accessed by an SNMP agent. The Anybus CompactCom 40 PROFINET IRT supports standardized MIBs: LLDP-MIB and MIB-II. Standardized MIBs are defined in RFC standards and contain variables that are divided into so called groups. The host application can change the values of some of the variables for the MIB-II.

10.3 MIB_II

The MIB-II of the Anybus CompactCom 40 PROFINET IRT contains the system- and interfaces group. The following tables show the variables according to the MIB-II standard (RFC1213) for monitoring the device status. The access authorizations refer to access via the SNMP protocol.

10.3.1 System Group Variables

Variable	Access Authorizations	Description	Source of Origin
sysDescr	Read only	Description of the device. Data type: DisplayString (only printable ASCII characters). Max 255 characters. Factory default setting: "HMS Industrial Networks Anybus-CompactCom 40"	PROFINET IO Object; Instance attribute 19 - System Description See PROFINET IO Object (F6h) , p. 131
sysObjectID	Read only	N/A. Value=0	Internal
sysUpTime	Read only	Time since last power up (in hundredths of a second)	Internal
sysContact	Read/Write	Identification of the contact person for the device, including contact information. Data type: Displaystring. Max 255 characters. Factory default setting: empty string	Internal
sysName	Read/Write	Name of the device. Data type: Displaystring. Max 255 characters. Factory default setting: empty string	Internal
sysLocation	Read/Write	Physical location of the device (IM Tag Location). Data type: (DisplayString). Max 255 characters. Factory default setting: empty string	Internal
sysServices	Read only	Functionality of the device. Value=74, which indicates that the device has functionality that represents layers 2 (switch), 4(TCP) and 7(Application) in the OSI model.	Internal

10.3.2 Interfaces Group Variables

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

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

Variable	Data Type	Value	Description
ifNumber	integer	3	Number of network interfaces present. Constant
ifIndex(1..3)	integer	ifIndex(1) = 1 ifIndex(2) = 2 ifIndex(3) = 3	Unique value for each interface. Constant
ifDescr(1..3)	octetstring	ifDescr(1) = "port-001" ifDescr(2) = "port-002" ifDescr(3) = "port-internal"	Information about the interface. ifDescr(1) must equal "port-001" and ifDescr(2) = "port-002" to be compatible with the STEP7 topology scanner.
ifType(1..3)	integer	6 ("Ethernet-csmacd")	Type of interface
ifMtu(1..3)	integer	1500	Size of largest datagram that can be sent/received on the interface, specified in octets
ifSpeed(1..3)	gauge	0 or 100 000 000	Data transfer rate of the Ethernet port in bits per second. The speed is only shown for ports where the link status is "up".
ifPhysAddress(1..3)	octetstring		MAC address for the ports
ifAdminStatus(1..3)	integer	1 ("up")	Desired state of the Ethernet port
ifOperStatus(1..3)	integer	1 ("up") or 2 ("down")	Current operating state of the Ethernet port. (Link = "up", No link = "down".)
ifLastChange(1..3)	timeticks	Time when state changed, except ifLastChange(3) = 0	Time (since start-up) when the port changed to its current state, see previous variable. Indicated in multiples of hundredths of a second
ifInOctets(1..3)	counter	ifInOctets(1..3) = Number of octets	Total number of octets received on the interface, including framing characters
ifInUcastPkts(1..3)	counter	ifInUcastPkts(1..3) = Number of unicast packets	Number of subnetwork-unicast packets delivered to a higher-layer protocol
ifInNUcastPkts(1..3)	counter	ifInNUcastPkts(1..3) = Number of non-unicast packets	Number of non-unicast (i.e. subnetwork-broadcast or subnetwork-multicast) delivered to a higher-layer protocol.
ifInDiscards(1..3)	counter	ifInDiscards(1..3) = number of discarded packets	Number of inbound packets which were discarded, without any error detected, not to be delivered to a higher-layer protocol. (One reason to discard packages might be to free up buffer space)
ifInErrors(1..3)	counter	ifInErrors(1..3) = number of error packets	Number of inbound packets with errors
ifInUnknownProtos(1..3)	counter	ifInUnknownProtos(1..3) = Number of unknown packets	Number of packets received via the interface, discarded because of an unknown or unsupported protocol.
ifOutOctets(1..3)	counter	ifOutOctets(1..3) = Number of octets	Total number of octets transmitted out from the interface, including framing characters
ifOutUcastPkts(1..3)	counter	ifOutUcastPkts(1..3) = Number of unicast packets	Total number of packets that higher-level protocols requested to be transmitted to a subnetwork-unicast address, including those that were discarded or not sent.
ifOutNUcastPkts(1..3)	counter	ifOutNUcastPkts(1..3) = Number of non-unicast packets	Total number of packets that higher-level protocols requested be transmitted to a non-unicast (i.e. a subnetwork-broadcast or subnetwork-multicast) address, including those that were discarded or not sent.

Variable	Data Type	Value	Description
ifOutDiscards(1..3)	counter	ifOutDiscards(1..3) = Number of discarded packets	Number of outbound packets which were discarded without any error detected, not to be transmitted. (One reason to discard packages might be to free up buffer space)
ifOutErrors(1..3)	counter	ifOutErrors(1..3) = Number of error packets	Number of outbound packets that could not be transmitted due to errors
ifOutQLen(1..3)	gauge	ifOutQLen(1..3) = Number of packets in queue	Length of the output packet queue (in packets),
ifSpecific(1..3)	objid	.0.0	Reference to MIB definitions specific to the particular media being used to realize the interface. Here no reference is available, so a fixed value is used for all ports.

11 Media Redundancy Protocol (MRP)

11.1 General

Media Redundancy Protocol (MRP) is a PROFINET specific ring protocol ensuring redundancy in the network, which can significantly decrease network downtime. It is a token based ring protocol with a master-slave hierarchy.

All the nodes in the PROFINET network part of the ring are connected using ring topology (that is, the last node is connected directly to the first node). If, at any point, the connection between two nodes would break, the data will flow the other way instead, guaranteeing that data can be sent to/from the IO Controller to the IO Device(s). The self-healing time is approximately 200 ms.

The Media Redundancy Master (MRM) is responsible for checking the functional capability of the ring network, by sending out cyclic tokens. The Media Redundancy Clients (MRC) basically work as switches that pass on the tokens. The Anybus CompactCom module supports acting as a Media Redundancy Client (MRC). It also supports propagating link change to the Media Redundancy Master.

11.2 GSDML Entries

MRP functionality is enabled by default in the GSD file. The settings for MRP is located at the Device Access Point (DAP). Within the `<InterfaceSubmoduleItem ...>` the role the Anybus module can play for MRP is defined. This shall be set to "Client" with the keyword `<MediaRedundancy SupportedRole="Client"/>`.

For each physical port there are two keywords in the `<PortSubmoduleItem ...>` section `<PortSubmoduleItem ... SupportsRingportConfig="true" IsDefaultRingport="true" ... />`. These are set to "true" by default. To disable MRP, these two shall be set to "false".

12 Anybus Module Objects

12.1 General Information

Standard Objects:

- [Anybus Object \(01h\), p. 70](#)
- [Diagnostic Object \(02h\), p. 71](#)
- [Network Object \(03h\), p. 74](#)
- [Network Configuration Object \(04h\), p. 75](#)

Network Specific Objects:

- [Network PROFINET IO Object \(0Eh\), p. 82](#)
- [Socket Interface Object \(07h\), p. 99](#)
- [SMTP Client Object \(09h\), p. 116](#)
- [File System Interface Object \(0Ah\), p. 121](#)
- [Network Ethernet Object \(0Ch\), p. 121](#)
- [Functional Safety Module Object \(11h\), p. 122](#)

12.2 Anybus Object (01h)

Category

Basic

Object Description

This object assembles all common Anybus data, and is described thoroughly in the general *Anybus CompactCom 40 Software Design Guide*.

Supported Commands

Object:	Get_Attribute
Instance:	Get_Attribute
	Set_Attribute
	Get_Enum_String

Object Attributes (Instance #0)

(Consult the general *Anybus CompactCom 40 Software Design Guide* for further information.)

Instance Attributes (Instance #1)

Basic

#	Name	Access	Type	Value
1	Module type	Get	UINT16	0403h (Standard Anybus Compact-Com 40)
2... 11	-	-	-	Consult the general Anybus Compact-Com 40 Software Design Guide for further information.
12	LED colors	Get	struct of: UINT8 (LED1A) UINT8 (LED1B) UINT8 (LED2A) UINT8 (LED2B)	<u>Value:</u> <u>Color:</u> 01h Green 02h Red 01h Green 02h Red
13... 16	-	-	-	Consult the general Anybus Compact-Com 40 Software Design Guide for further information.
17	Virtual attributes	Get/Set		
18	Black list/White list	Get/Set		
19	Network time	Get	UINT64	0 (Not supported)

12.3 Diagnostic Object (02h)

Category

Extended

Object Description

This object provides a standardized way of handling host application events & diagnostics, and is

thoroughly described in the general *Anybus CompactCom 40 Software Design Guide*.

Supported Commands

Object:	Get_Attribute
	Create
	Delete
Instance:	Get_Attribute

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1... 4	-	-	-	Consult the general Anybus CompactCom 40 Software Design Guide for further information.
11	Max no. of instances	Get	UINT16	5+1 (Of the maximum number of instances there should always be one instance reserved for an event of severity level 'Major, unrecoverable', to force the module into the 'EXCEPTION'-state.)
12	Supported functionality	Get	BITS32	Bit 0: "0" (Latching events are not supported) Bit 1 - 31: reserved (shall be "0")

Instance Attributes (Instance #1)

Extended

#	Name	Access	Data Type	Value
1	Severity	Get	UINT8	Consult the general Anybus CompactCom 40 Software Design Guide for further information.
2	Event Code	Get	UINT8	
3	NW specific data	Get	Array of UINT8	Optional network specific information, see below.

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

Event Code	Result																																																																								
0...FEh	<p>Module issues a Channel Diagnostic entry/alarm. The Event Code will be translated and represented as the Channel Error Type as follows:</p> <table border="1"> <thead> <tr> <th><u>Code:</u></th> <th><u>Event (Anybus):</u></th> <th><u>Channel Error Type (PROFINET):</u></th> </tr> </thead> <tbody> <tr><td>10h</td><td>Generic Error</td><td>Error</td></tr> <tr><td>20h</td><td>Current</td><td>Short circuit</td></tr> <tr><td>21h</td><td>Current, device input side</td><td>Short circuit</td></tr> <tr><td>22h</td><td>Current, inside the device</td><td>Short circuit</td></tr> <tr><td>23h</td><td>Current, device output side</td><td>Short circuit</td></tr> <tr><td>30h</td><td>Voltage</td><td>Overvoltage</td></tr> <tr><td>31h</td><td>Mains Voltage</td><td>Overvoltage</td></tr> <tr><td>32h</td><td>Voltage inside the device</td><td>Overvoltage</td></tr> <tr><td>33h</td><td>Output Voltage</td><td>Overvoltage</td></tr> <tr><td>40h</td><td>Temperature</td><td>Overtemperature</td></tr> <tr><td>41h</td><td>Ambient Temperature</td><td>Overtemperature</td></tr> <tr><td>42h</td><td>Device Temperature</td><td>Overtemperature</td></tr> <tr><td>50h</td><td>Device Hardware</td><td>Error</td></tr> <tr><td>60h</td><td>Device Software</td><td>Error</td></tr> <tr><td>61h</td><td>Internal Software</td><td>Error</td></tr> <tr><td>62h</td><td>User Software</td><td>Error</td></tr> <tr><td>63h</td><td>Data Set</td><td>Error</td></tr> <tr><td>70h</td><td>Additional Modules</td><td>Error</td></tr> <tr><td>80h</td><td>Monitoring</td><td>Error</td></tr> <tr><td>81h</td><td>Communication</td><td>Error</td></tr> <tr><td>82h</td><td>Protocol Error</td><td>Error</td></tr> <tr><td>90h</td><td>External Error</td><td>Error</td></tr> <tr><td>F0h</td><td>Additional Functions</td><td>Error</td></tr> </tbody> </table>	<u>Code:</u>	<u>Event (Anybus):</u>	<u>Channel Error Type (PROFINET):</u>	10h	Generic Error	Error	20h	Current	Short circuit	21h	Current, device input side	Short circuit	22h	Current, inside the device	Short circuit	23h	Current, device output side	Short circuit	30h	Voltage	Overvoltage	31h	Mains Voltage	Overvoltage	32h	Voltage inside the device	Overvoltage	33h	Output Voltage	Overvoltage	40h	Temperature	Overtemperature	41h	Ambient Temperature	Overtemperature	42h	Device Temperature	Overtemperature	50h	Device Hardware	Error	60h	Device Software	Error	61h	Internal Software	Error	62h	User Software	Error	63h	Data Set	Error	70h	Additional Modules	Error	80h	Monitoring	Error	81h	Communication	Error	82h	Protocol Error	Error	90h	External Error	Error	F0h	Additional Functions	Error
<u>Code:</u>	<u>Event (Anybus):</u>	<u>Channel Error Type (PROFINET):</u>																																																																							
10h	Generic Error	Error																																																																							
20h	Current	Short circuit																																																																							
21h	Current, device input side	Short circuit																																																																							
22h	Current, inside the device	Short circuit																																																																							
23h	Current, device output side	Short circuit																																																																							
30h	Voltage	Overvoltage																																																																							
31h	Mains Voltage	Overvoltage																																																																							
32h	Voltage inside the device	Overvoltage																																																																							
33h	Output Voltage	Overvoltage																																																																							
40h	Temperature	Overtemperature																																																																							
41h	Ambient Temperature	Overtemperature																																																																							
42h	Device Temperature	Overtemperature																																																																							
50h	Device Hardware	Error																																																																							
60h	Device Software	Error																																																																							
61h	Internal Software	Error																																																																							
62h	User Software	Error																																																																							
63h	Data Set	Error																																																																							
70h	Additional Modules	Error																																																																							
80h	Monitoring	Error																																																																							
81h	Communication	Error																																																																							
82h	Protocol Error	Error																																																																							
90h	External Error	Error																																																																							
F0h	Additional Functions	Error																																																																							
FFh	<p>Module issues a Generic Diagnostic entry/alarm based on network specific data. See details below.</p>																																																																								

Details: Network Specific Data

Network specific diagnostic data serves as the payload in the PROFINET diagnostic alarm. The data contains an identifier (UserStructureIdentifier) that describes the structure of the data.

The following identifier values are supported:

- 8000h (Channel Diagnostic)
- 8002h (Extended Channel Diagnostic)
- 8003h (Qualified Channel Diagnostic)

Byte	Contents
1	UserStructureIdentifier, low byte
2	UserStructureIdentifier, high byte
3... 16	Data

Object Error Codes

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

12.4 Network Object (03h)

Category

Basic

Object Description

For more information regarding this object, consult the general *Anybus CompactCom 40 Software Design Guide*.

Supported Commands

Object:	Get_Attribute
Instance:	Get_Attribute
	Set_Attribute
	Get_Enum_String
	Map_ADI_Write_Area
	Map_ADI_Read_Area

Object Attributes (Instance #0)

(Consult the general *Anybus CompactCom 40 Software Design Guide* for further information.)

Instance Attributes (Instance #1)

Basic

#	Name	Access	Type	Value
1	Network type	Get	UINT16	0089h (PROFINET IRT) or 009Dh (PROFINET IRT Fiber Optic)
2	Network type string	Get	Array of CHAR	"PROFINET IRT" or "PROFINET IRT Fiber Optic"
3	Data format	Get	ENUM	01h (MSB first)
4	Parameter data support	Get	BOOL	True
5	Write process data size	Get	UINT16	Current write process data size (in bytes) Updated on every successful Map_ADI_Write_Area. (Consult the general <i>Anybus CompactCom 40 Software Design Guide</i> for further information.)
6	Read process data size	Get	UINT16	Current read process data size (in bytes) Updated on every successful Map_ADI_Read_Area. (Consult the general <i>Anybus CompactCom 40 Software Design Guide</i> for further information.)
7	Exception Information	Get	UINT8	Additional information available if the module has entered the EXCEPTION state. <u>Value:</u> <u>Meaning:</u> 00h No information 01h Illegal value 02h Wrong data size 03h Illegal response 04h Missing MAC address (Only valid for Anybus IP)

12.5 Network Configuration Object (04h)

Category

Extended

Object Description

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

See also...

- [Communication Settings, p. 13](#)
- [E-mail Client, p. 39](#)



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

Supported Commands

Object:	Get_Attribute Reset
Instance:	Get_Attribute Set_Attribute Get_Enum_String

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value	Description
3	Number of instances	Get	UINT16	17	Supported number of instances
4	Highest instance number	Get	UINT16	21	Highest instance number

(Consult the general *Anybus CompactCom 40 Software Design Guide* for further information.)

Instance Attributes (Instance #3, IP Address)

Value is used after module reset.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"IP address" (Multilingual, see page 80)
2	Data type	Get	UINT8	04h (= UINT8)
3	Number of elements	Get	UINT8	04h (four elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of UINT8	Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)



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

Instance Attributes (Instance #4, Subnet Mask)

Value is used after module reset.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"Subnet mask" (Multilingual, see page 80)
2	Data type	Get	UINT8	04h (= UINT8)
3	Number of elements	Get	UINT8	04h (four elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of UINT8	Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)



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

Instance Attributes (Instance #5, Gateway)

Value is used after module reset.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"Gateway" (Multilingual, see page 80)
2	Data type	Get	UINT8	04h (= UINT8)
3	Number of elements	Get	UINT8	04h (four elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of UINT8	Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)



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

Instance Attributes (Instance #6, DHCP)

Value is used after module reset.

#	Name	Access	Data Type	Description									
1	Name	Get	Array of CHAR	"DHCP" (Multilingual, see page 80)									
2	Data type	Get	UINT8	08h (= ENUM)									
3	Number of elements	Get	UINT8	01h (one element)									
4	Descriptor	Get	UINT8	07h (read/write/shared access)									
5	Value	Get/Set	ENUM	<table border="0"> <thead> <tr> <th><u>Value</u></th> <th><u>String</u></th> <th><u>Meaning</u></th> </tr> </thead> <tbody> <tr> <td>00h</td> <td>"Disable"</td> <td>DHCP disabled (default)</td> </tr> <tr> <td>01h</td> <td>"Enable"</td> <td>DHCP enabled (Multilingual, see page 80)</td> </tr> </tbody> </table>	<u>Value</u>	<u>String</u>	<u>Meaning</u>	00h	"Disable"	DHCP disabled (default)	01h	"Enable"	DHCP enabled (Multilingual, see page 80)
<u>Value</u>	<u>String</u>	<u>Meaning</u>											
00h	"Disable"	DHCP disabled (default)											
01h	"Enable"	DHCP enabled (Multilingual, see page 80)											



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

Instance Attributes (Instance #9, DNS1)

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

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"DNS1" (Multilingual, see page 80)
2	Data type	Get	UINT8	04h (= UINT8)
3	Number of elements	Get	UINT8	04h (four elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of UINT8	Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)

Instance Attributes (Instance #10, DNS2)

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

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"DNS2" (Multilingual, see page 80)
2	Data type	Get	UINT8	04h (= UINT8)
3	Number of elements	Get	UINT8	04h (four elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of UINT8	Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)

Instance Attributes (Instance #11, Host name)

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

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"Host name" (Multilingual, see page 80)
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	40h (64 elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of CHAR	Host name, 64 characters (pad with space to full length)

Instance Attributes (Instance #12, Domain name)

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

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"Host name" (Multilingual, see page 80)
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	30h (48 elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of CHAR	Domain name, 48 characters (pad with space to full length)

Instance Attributes (Instance #13, SMTP Server)

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

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"SMTP server" (Multilingual, see page 80)
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	40h (64 elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of CHAR	SMTP server address, 64 characters. Dotted decimal format or server name(pad with space to full length)

Instance Attributes (Instance #14, SMTP User)

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

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"SMTP user" (Multilingual, see page 80)
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	40h (64 elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of CHAR	SMTP account user name, 64 characters (pad with space to full length)

Instance Attributes (Instance #15, SMTP Password)

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

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"SMTP Pswd" (Multilingual, see page 80)
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	40h (64 elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of CHAR	Host name, 64 characters (pad with space to full length)

Instance Attributes (Instance #20, Station Name)

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

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

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



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

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	“Station name” (Multilingual, see page 80)
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	F0h (240 elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of CHAR	The current Station name
6	Configured value	Get/Set	Array of CHAR	The configured value that will be used after restart



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

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

Instance Attributes (Instance #21, F-Address)

This instance holds the F-Address, which is the PROFIsafe address for the safety module. If this value is changed by the host application during runtime, a reset is required in order for changes to have effect.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"F-Address" (Multilingual, see page 80)
2	Data type	Get	UINT8	05h (= UINT16)
3	Number of elements	Get	UINT8	01h (one elements)
4	Descriptor	Get	UINT8	03h (read/write access)
5	Value	Get/Set	Array of CHAR	F-Address set by the host application. Range: 1 - 65534 (Default: 1)
6	Configured value	Get/Set	Array of CHAR	The configured value that will be used after restart

Multilingual Strings

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

Instance	English	German	Spanish	Italian	French
3	IP address	IP-Adresse	Dirección IP	Indirizzo IP	Adresse IP
4	Subnet mask	Subnetz- maske	Masac. subred	Sottorete	Sous-réseau
5	Gateway	Gateway	Pasarela	Gateway	Passerelle
6	DHCP	DHCP	DHCP	DHCP	DHCP
	Enable	Einschalten	Activado	Abilitato	Activé
	Disable	Ausschalten	Desactivado	Disabilitato	Désactivé
7	Comm 1	Komm 1	Comu 1	Connessione 1	Comm 1
8	Comm 2	Komm 2	Comu 2	Connessione 2	Comm 2
9	DNS1	DNS 1	DNS Primaria	DNS1	DNS1
10	DNS2	DNS 2	DNS Secundia.	DNS2	DNS2
11	Host name	Host name	Nombre Host	Nome Host	Nom hôte
12	Domain name	Domain name	Nobre Domain	Nome Dominio	Dom Domaine
13	SMTP Server	SMTP Server	Servidor SMTP	Server SMTP	SMTP serveur
14	SMTP User	SMTP User	Usuario SMTP	Utente SMTP	SMTP utiliza.
15	SMTP Pswd	SMTP PSWD	Clave SMTP	Password SMTP	SMTP mt passe
20	Station name	Stationsname	Nom. Estacion	Nome Stazione	Nom Station
21	F-Address	F-Adresse	Dirección-F	Indirizzo-F	F-Adresse

Command Details: Reset

Category

Details

Command Code: 05h
Valid for: Object Instance

Description

A reset command to this object will result in that all instances are set to their default values.

It is optional to implement support for this command.

- Command Details

Field	Comments
CmdExt[0]	00h (Reserved)
CmdExt[1]	01h (Factory default reset)

- Response Details
(No data)

12.6 Network PROFINET IO Object (0Eh)

Category

Extended

Object Description

When the application maps ADIs to process data during start-up, the Anybus CompactCom 40 PROFINET IRT will create the module configuration as described in [Real Identification \(RI\), p. 20](#). The modules in the GSDML file must then be described in the same way. The GSDML file provided by HMS provides a few examples based on this way of describing modules.

If the end-user wishes to define modules in another way the application must provide the module configuration to the Anybus CompactCom 40 PROFINET IRT. This is achieved by using the following commands:

- API_Add
- Plug_Module
- Plug_Submodule
- Plug_Submodule_Ext

These commands need to be sent after the process data is mapped, and before sending setup complete to the Anybus CompactCom 40 PROFINET IRT.

Example:

Initially, the application maps ADIs as process data by calling all or some of the functions below:

- Map_ADI_Write_Area (10h)
- Map_ADI_Read_Area (11h)
- Map_ADI_Write_Ext_Area (12h)
- Map_ADI_Read_Ext_Area (13h)

Modules and submodules are now created based on this information as described in [Real Identification \(RI\), p. 20](#).

1. Call API_Add to add an API.
2. Call Plug_Module to add a module to the API.
3. Call Plug_Submodule one or more times to add submodules to the module.
4. Repeat steps 2 and 3 to add modules to the API.

After the configuration is complete, call setup complete.

See also ...

[Flowchart —Establishment of Real Identification \(RI\), p. 159](#)

Removing and Exchanging Modules and Submodules

If the RI has been created by the host application through custom configuration, there are ways of removing modules and plug new modules during runtime.

The application will be notified by the command `Cfg_Mismatch_Ind` for every submodule that does not match. This information will also be provided in the command `Expected_Ident_Ind`. The application can then decide to remove the plugged module by issuing the command `Pull_Module`. This will remove the whole module and its submodules. Then, based on the information received from either `Cfg_Mismatch_Ind` or `Expected_Ident_Ind`, the application can adopt to the PLC configuration by issuing new `Plug_Module`, `Plug_Submodule` and `Plug_Submodule_Ext` commands.

See also ...

- [Configuration Mismatch, p. 22](#)
- [Custom Configuration mismatch, p. 161](#) (flowchart)

Supported Commands

Object:	<code>Get_Attribute</code>
	<code>Plug_Module</code> (see below)
	<code>Plug_Submodule</code> (see below)
	<code>Plug_Submodule_Ext</code> (see below)
	<code>Pull_Module</code> (see below)
	<code>Pull_Submodule</code> (see below)
	<code>API_Add</code> (see below)
	<code>AppI_State_Ready</code> (see below)
	<code>AR_Abort</code> (see below)
	<code>IM_Options</code> (see below)
	<code>Ident_Change_Done</code> (see below)
	<code>Add_Safety_Module</code> (see below)
Instance:	<code>Get_Attribute</code>

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1	Name	Get	Array of Char	"Network PROFINET IO"
2	Revision	Get	UINT8	01h
3	Number of instances	Get	UINT16	0001h
4	Highest instance number	Get	UINT16	0001h

Instance Attributes (Instance #1)

Extended

#	Name	Access	Type	Value
1	OnLineTrans	Get	UINT32	Diagnostic counters; keeps track of the number of on-line transitions
2	OffLineTrans	Get	UINT32	Diagnostic counters; keeps track of the number of off-line transitions
3	(reserved)			
4	Last AbortInd ReasonCode	Get	UINT16	Reason code for most recent Abort indication See also... - "Command Details: AR_Abort" on page 98
5	AddedApis	Get	UINT16	Returns the number of APIs added (including API 0)
6	ApiList	Get	Array of UINT32	First element will always be zero and the second element will contain an additional API. Length of the array is determined by parameter "AddedApis".
7	EstablishedArs	Get	UINT16	The number of Application Relationships currently established
8	ArList	Get	Array of UINT16	Array of Application Relationship handles. Length of array is determined by parameter "EstablishedArs".
9	-	-	-	-
10	Port 1 MAC Address	Get	Array of UINT8	6 Byte PROFINET Port 1 MAC address See also... - "Ethernet Host Object (F9h)" on page 147
11	Port 2 MAC Address	Get	Array of UINT8	6 Byte PROFINET Port 2 MAC address See also... - "Ethernet Host Object (F9h)" on page 147

Command Details: Plug_Module

Category

Extended

Details

Command Code: 10h
Valid for: Object Instance

Description

This command may be called during start-up to specify the Real Identification. It may also be called during runtime in case there are changes to the Real Identification.



*It is only permitted to issue this command if **API_Add** has been issued first.*

- Command Details

Field	Contents	Comments
CmdExt[0]	(reserved, set to zero)	(reserved for future use)
CmdExt[1]		
Data[0]	API (low word, low byte)	Application Process Instance (API) See also... - Application Data Instances (ADIs) , p. 14
Data[1]	API (low word, high byte)	
Data[2]	API (high word, low byte)	
Data[3]	API (high word, high byte)	
Data[4]	SlotNr (low byte)	Destination slot for module. Range: 0... 0x7FFF
Data[5]	SlotNr (high byte)	
Data[6]	ModIdent (low word, low byte)	Module identified as state in the GSD-file
Data[7]	ModIdent (low word, high byte)	
Data[8]	ModIdent (high word, low byte)	
Data[9]	ModIdent (high word, high byte)	

See also...

[Real Identification \(RI\)](#), p. 20 ([Configuration Mismatch](#), p. 22)

Command Details: Plug_Submodule

Category

Extended

Details

Command Code:	11h
Valid for:	Object Instance

Description

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

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

The Anybus CompactCom 40 PROFINET IRT supports up to 128 submodules in total.



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

*It is only permitted to issue this command if **API_Add** has been issued first.*

*The **Interface**- and **Port** submodules have to be plugged in order to pass certification tests.*

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

- Command Details

Field	Contents	Comments
CmdExt[0]	(reserved, set to zero)	(reserved for future use)
CmdExt[1]		
Data[0]	ADI number (low byte), Read	Reference to the ADI previously mapped with Map_ADI_Read_Area.
Data[1]	ADI number (high byte), Read	
Data[2]	ADI element, Read	Reference to the element of the ADI mapped with Map_ADI_Read_Area for the specified SlotNr (See Data[10... 11]). Range: 1... 255 ADI element associated with the submodule 0 Entire ADI is associated with the submodule
Data[3]	ADI number (low byte), Write	Reference to the ADI previously mapped with Map_ADI_Write_Area.
Data[4]	ADI number (high byte), Write	
Data[5]	ADI element, Read	Reference to the element of the ADI mapped with Map_ADI_Write_Area for the specified SlotNr (See Data[10... 11]). Range: 1... 255 ADI element associated with the submodule 0 Entire ADI is associated with the submodule
Data[6]	API (low word, low byte)	Application Process Instance (API) See also... - "Application Process Instances (API)" on page 22
Data[7]	API (low word, high byte)	
Data[8]	API (high word, low byte)	
Data[9]	API (high word, high byte)	
Data[10]	SlotNr (low byte)	Destination slot for submodule. Range: 0... 0x7FFF
Data[11]	SlotNr (high byte)	
Data[12]	SubSlotNr (low byte)	Destination subslot for submodule. Range: For API 0: 1... 0x8002 For API >0: 1... 0x7FFF
Data[13]	SubSlotNr (high byte)	
Data[14]	SubModIdent (low word, low byte)	Submodule identifier as stated in the GSD-file
Data[15]	SubModIdent (low word, high byte)	
Data[16]	SubModIdent (high word, low byte)	
Data[17]	SubModIdent (high word, high byte)	

See also...

[Real Identification \(RI\), p. 20](#) ([Configuration Mismatch, p. 22](#))

Command Details: Plug_Submodule_Ext

Category

Extended

Details

Command Code:	19h
Valid for:	Object Instance

Description

This is an extended version of the **Plug_Submodule** command. This command may be called during start-up to specify the Real Identification. It additionally features the possibility to associate a submodule with several consecutive ADI elements. (The **Plug_Submodule** command only allows association with one ADI element or all ADI elements.)

This command can also be called during operation if there are changes to the Real Identification. A **Plug** or **Plug Wrong Submodule**-alarm is automatically sent to the master as a result of this action.

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

The Anybus CompactCom 40 PROFINET IRT supports up to 128 submodules in total.



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

*It is only permitted to issue this command if **API_Add** has been issued first.*

*The **Interface**- and **Port** submodules have to be plugged in order to pass certification tests.*

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

*It is not recommended to mix **Plug_Submodule** and **Plug_Submodule_Ext** commands.*

- Command Details

Field	Contents	Comments
CmdExt[0]	(reserved, set to zero)	(reserved for future use)
CmdExt[1]		
Data[0]	ADI number (low byte), Read	Reference to the ADI previously mapped with Map_ADI_Read_Area.
Data[1]	ADI number (high byte), Read	
Data[2]	First ADI element, Read	Reference to the first ADI element associated with the submodule. Range: 0 to 255 - ADI element (0 is the first element)
Data[3]	Number of consecutive ADI elements, Read	Number of consecutive elements associated with the submodule. Range: 1 to 255 (Number of elements)
Data[4]	ADI number (low byte), Write	Reference to the ADI previously mapped with Map_ADI_Write_Area.
Data[5]	ADI number (high byte), Write	
Data[6]	ADI element, Write	Reference to the first ADI element associated with the submodule. Range: 0 to 255 - ADI element (0 is the first element)
Data[7]	Number of consecutive ADI elements, Write	Number of consecutive elements associated with the submodule. Range: 1 to 255 (Number of elements)
Data[8]	API (low word, low byte)	Application Process Instance (API) See also... - "Application Process Instances (API)" on page 22
Data[9]	API (low word, high byte)	
Data[10]	API (high word, low byte)	
Data[11]	API (high word, high byte)	
Data[12]	SlotNr (low byte)	Destination slot for submodule. Range: 0... 0x7FFF
Data[13]	SlotNr (high byte)	
Data[14]	SubSlotNr (low byte)	Destination subslot for submodule. Range: For API 0: 1... 0x8002 For API >0: 1... 0x7FFF
Data[15]	SubSlotNr (high byte)	
Data[16]	SubModIdent (low word, low byte)	Submodule identifier as stated in the GSD file
Data[17]	SubModIdent (low word, high byte)	
Data[18]	SubModIdent (high word, low byte)	
Data[19]	SubModIdent (high word, high byte)	

See also...

[Real Identification \(RI\), p. 20 \(Configuration Mismatch, p. 22\)](#)

Command Details: Pull_Module

Category

Extended

Details

Command Code: 12h
Valid for: Object Instance

Description

This command removes a module from the configuration. Can be issued at any time. During runtime, it can be called in case there are changes to the Real Identification. The Anybus CompactCom 40 PROFINET IRT then automatically issues a **Pull** or **Pull Module** alarm to the master.

- Command Details

Field	Contents	Comments
CmdExt[0]	(reserved, set to zero)	(reserved for future use)
CmdExt[1]		
Data[0]	API (low word, low byte)	Application Process Instance (API) See also... – Application Process Instances (API) , p. 19
Data[1]	API (low word, high byte)	
Data[2]	API (high word, low byte)	
Data[3]	API (high word, high byte)	
Data[4]	SlotNr (low byte)	Slot number of module. Range: 0... 0x7FFF
Data[5]	SlotNr (high byte)	

Command Details: Pull_Submodule

Category

Extended

Details

Command Code: 13h
Valid for: Object Instance

Description

This command removes a submodule from the configuration and can be issued at any time. During runtime, it can be called in case there are changes to the Real Identification. The Anybus CompactCom 40 PROFINET IRT0 then automatically issues a **Pull** alarm to the master.

- Command Details

Field	Contents	Comments
CmdExt[0]	(reserved, set to zero)	(reserved for future use)
CmdExt[1]		
Data[0]	API (low word, low byte)	Application Process Instance (API) See also... – Application Process Instances (API) , p. 19
Data[1]	API (low word, high byte)	
Data[2]	API (high word, low byte)	
Data[3]	API (high word, high byte)	
Data[4]	SlotNr (low byte)	Slot number of submodule. Range: 0... 0x7FFF
Data[5]	SlotNr (high byte)	
Data[6]	SubSlotNr (low byte)	Subslot number of submodule. Range: For API 0: 1... 0x8002 For API >0: 1... 0x7FFF
Data[7]	SubSlotNr (high byte)	

Command Details: API_Add

Category

Extended

Details

Command Code: 14h
Valid for: Object Instance

Description

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



*This command may only be issued prior to setting the **Setup Complete**-attribute in the Anybus Object.*

*This command clears the default Real Identification created by the Anybus module while mapping ADIs to Process Data. Therefore, issuing this command effectively makes it mandatory to specify the actual Real Identification by means of the **Plug_Module** and **Plug_Submodule**-commands.*

- Command Details

Field	Contents	Comments
CmdExt[0]	(reserved, set to zero)	(reserved for future use)
CmdExt[1]		
Data[0]	API (low word, low byte)	Application Process Instance (API) See also... – Application Process Instances (API), p. 19
Data[1]	API (low word, high byte)	
Data[2]	API (high word, low byte)	
Data[3]	API (high word, high byte)	
Data[4]	(reserved, set to zero)	(reserved)
Data[5]		
Data[6]		
Data[7]		

See also...

[Application Process Instances \(API\), p. 19](#)

Command Details: Appl_State_Ready

Category

Extended

Details

Command Code: 15h
Valid for: Object Instance

Description

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

- Command Details

Field	Contents	Comments
CmdExt[0]	AR handle (low byte)	Handle for the Application Relationship (AR)
CmdExt[1]	AR handle (high byte)	

See also...

[Application Process Instances \(API\), p. 19](#)

- End_Of_Prm_Ind, command details in [PROFINET IO Object \(F6h\), p. 131](#)

Command Details: AR_Abort

Category

Extended

Details

Command Code: 16h
Valid for: Object Instance

Description

This command indicates to the module that the current application relationship shall be aborted.

- Command Details

Field	Contents	Comments
CmdExt[0]	AR handle (low byte)	Handle for the Application Relationship (AR)
CmdExt[1]	AR handle (high byte)	

See also...

- [Application Process Instances \(API\), p. 19](#)
- AR_Check_Ind, command details in [PROFINET IO Object \(F6h\), p. 131](#)
- Expected_Ident_Ind, command details in [PROFINET IO Object \(F6h\), p. 131](#)
- Appl_State_Ready, command details in [PROFINET IO Object \(F6h\), p. 131](#)

Command Details: IM_Options

Category

Extended

Details

Command Code:	18h
Valid for:	Object Instance

Description

During startup, this command can be called to specify if I&M data for Slot0 (DAP) and/or Slot > 0 should be forwarded transparently by the Anybus CompactCom module. Additionally, it provides a way for the application to specify the I&M0 Filter Data.

I&M0 Filter Data is composed of three blocks: I&M0 Carrier Data, Module Representative Data and Device Representative Data (see table below). A submodule can belong to several blocks.

I&M0 Filter Data Contents	
Content	Description
I&M0 Carrier Data	List of all submodules being a carrier of discrete I&M data Block is mandatory to support according to the PROFINET specification
Module Representative Data	List of all submodules acting as module representative Block is optional to support according to the PROFINET specification
Device Representative Data	List of at least one submodule where I&M1, I&M2, and I&M3 can be written Block is mandatory to support according to the PROFINET specification

When transparent I&M data for slot > 0 is enabled, the application must store I&M data for modules located in slots > 0 to nonvolatile memory. In this case, all modules that carry discrete I&M data shall be included in the **I&M0 Carrier Data**. (The Anybus CompactCom 40 PROFINET IRT will include the DAP submodule (located in slot 0, subslot 1) in the **I&M0 Carrier Data** and **Device Representative Data**.)

When transparent I&M data for slot 0 is enabled, the application must store I&M data for slot 0 to nonvolatile memory. The DAP submodule is by default included in the **I&M0 Carrier Data** and **Device Representative Data** but may be removed from any of the blocks using the **IM_Options** command.

See also...

- [Flowchart — I&M Record Data Handling, p. 158.](#)
- Get Record, command details in [PROFINET IO Object \(F6h\), p. 131](#)
- Set_Record, command details in [PROFINET IO Object \(F6h\), p. 131](#)

- Command Details

If the I&M0 Filter Data is of no interest, the Data Field is left out (command length = 0).

The command may contain one or several I&M0 Filter Data entries. The maximum amount of entries depends on the application. For a 256 bytes message channel the maximum amount of entries is 51 (256 / 5 = 51). For a 1524 bytes message channel, the maximum amount of entries is 304 (1524 / 5 = 304).

! For submodules to be listed in the I&M0 Filter data, the command must be sent when the Real Identification has been determined. This means that for the “ADI Based RI” method, the command must be sent when the module has shifted to WAIT_PROCESS state. For the “Application specific RI” method the command can be sent in SETUP state but after the plugging of modules/submodules is finished (Plug_Module/Plug_Submodule).

The table below contains two entries as an example.

Field	Contents	Comments	Example
CmdExt[0]	IM_Transparent	Bitmask for enabling transparent I&M data See table below	02h
CmdExt[1]	Reserved		
Data[0]	SlotNr (low byte)	Location of the submodule that shall be part of the I&M0 Filter Data	Entry #1 - module with discrete I/M data in slot 1, subslot 1
Data[1]	SlotNr (high byte)		
Data[2]	SubSlotNr (low byte)		
Data[3]	SubSlotNr (high byte)		
Data[4]	IM0_Filter_Data	Bitmask specifying which I&M0 Filter Data block(s) the submodule shall belong to See table below	01h
Data[5]	SlotNr (low byte)		Entry #2 - module with discrete I&M data in slot 2, subslot 1
Data[6]	SlotNr (high byte)		
Data[7]	SubSlotNr (low byte)		
Data[8]	SubSlotNr (high byte)		
Data[9]	IM0_Filter_Data		01h

IM_Transparent		
Bit	Value	Description
0	0	
	1	Transparent I&M data for SlotNr = 0 (DAP)
1	0	
	1	Transparent I&M data for SlotNr not equal to 0
2 - 7	-	Reserved

IM0_Filter_Data		
Bit	Value	Description
0	0	
	1	Submodule will be included in the I&M0 Carrier Data block
1	0	
	1	Submodule will be included in the Device Representative Data block
2	0	
	1	Submodule will be included in the Module Representative Data block
3 - 7	-	Reserved

Command Details: Ident_Change_Done

Category

Extended

Details

Command Code: 1Ah

Valid for: Object Instance

Description

This command shall be sent to the module when the host application has finished its adaptations of the Real Identification.

It is optional to implement support for this command, except for that it must be issued if the host application previously has responded with **Block** to the command Expected_Ident_Ind (1Bh).

- Command Details

Field	Contents	Comments
CmdExt[0]	AR handle (low byte)	Handle for the Application Relationship (AR)
CmdExt[1]	AR handle (high byte)	

- Response Details
(No data)

Command Details: Add_Safety_Module

Category

Extended

Details

Command Code: 17h
Valid for: Object Instance

Description

If the PROFIsafe functionality is to be used, this command must be called during start-up, before setup is complete. The Safety Module will always be located in slot 1. In addition to the slot number, the 16 most significant bits of the 32 bit module identification number for the Safety Module are specified. The 16 least significant bits are specified by the Safety Module itself. The command shall be called after successful enabling of the Safety Module.



*It is only permitted to issue this command if **API_Add** has been issued first.*

- Command Details

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

- Response Details

See“ Object Specific Error Codes” below.

Object Specific Error Codes

Code	Meaning
01h	The ADI has not been mapped with command Map_ADI_Write_Area
02h	The ADI has not been mapped with command Map_ADI_Read_Area
03h	Element does not exist for the ADI
04h	This ADI/element is already mapped
05h	API 0 must be added first
06h	API does not exist
07h	Trying to add an API already present
08h	There is no room for any more APIs
09h	Module in slot 0 cannot have any IO data
0Ah	Prior to plugging the requested module/submodule, slot 0 must be populated with a module and a submodule (in subslot 1)
0Bh	Slot occupied
0Ch	subslot occupied
0Dh	No module inserted in the specified slot
0Eh	No submodule inserted in the specified subslot
0Fh	Slot number specified is out-of-range
10h	subslot number specified is out-of-range
11h	The AR handle provided is not valid
12h	There is no application ready pending
13h	Unknown error (PROFINET IO stack denied the request)
14h	Max number of submodules have already been plugged
15h	Safety module has not been plugged
16h	ADI data type constraint

12.7 Socket Interface Object (07h)

Category

Extended

Object Description

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

Note that some of the commands used when accessing this object may require segmentation. A message will be segmented if the amount of data sent or received is larger than the message channel can handle. For more information, see [Message Segmentation, p. 114](#).



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

Supported Commands

Object:	Get_Attribute
	Create (See below)
	Delete (See below)
Instance:	Get_Attribute
	Set_Attribute
	Bind (See below)
	Shutdown (See below)
	Listen (See below)
	Accept (See below)
	Connect (See below)
	Receive (See below)
	Receive_From (See below)
	Send (See below)
	Send_To (See below)
	IP_Add_membership (See below)
	IP_Drop_membership (See below)
	DNS_Lookup (See below)

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1	Name	Get	Array of CHAR	"Socket interface"
2	Revision	Get	UINT8	01h
3	Number of instances	Get	UINT16	Number of opened sockets
4	Highest instance no.	Get	UINT16	Highest created instance number
11	Max. no. of instances	Get	UINT16	0014h (20 instances)

Instance Attributes (Sockets #1...20)

Extended

#	Name	Access	Data Type	Description
1	Socket Type	Get	UINT8	<p><u>Value:</u> <u>Socket Type</u></p> <p>00h SOCK_STREAM, NONBLOCKING (TCP)</p> <p>01h SOCK_STREAM, BLOCKING (TCP)</p> <p>02h SOCK_DGRAM, NONBLOCKING (UDP)</p> <p>03h SOCK_DGRAM, BLOCKING (UDP)</p>
2	Port	Get	UINT16	Local port that the socket is bound to
3	Host IP	Get	UINT32	Host IP address, or 0 (zero) if not connected
4	Host port	Get	UINT16	Host port number, or 0 (zero) if not connected
5	TCP State	Get	UINT8	<p>State (TCP sockets only):</p> <p><u>Value</u> <u>State/Description</u></p> <p>00h CLOSED Closed</p> <p>01h LISTEN Listening for connection</p> <p>02h SYN_SENT Active, have sent and received SYN</p> <p>03h SYN_RECEIVED Have sent and received SYN</p> <p>04h ESTABLISHED Established.</p> <p>05h CLOSE_WAIT Received FIN, waiting for close</p> <p>06h FIN_WAIT_1 Have closed, sent FIN</p> <p>07h CLOSING Closed exchanged FIN; await FIN ACK</p> <p>08h LAST_ACK Have FIN and close; await FIN ACK</p> <p>09h FIN_WAIT_2 Have closed, FIN is acknowledged</p> <p>Ah TIME_WAIT Quiet wait after close</p>
6	TCP RX bytes	Get	UINT16	Number of bytes in RX buffers (TCP sockets only)
7	TCP TX bytes	Get	UINT16	Number of bytes in TX buffers (TCP sockets only)
8	Reuse address	Get/Set	BOOL	<p>Socket can reuse local address</p> <p><u>Value</u> <u>Meaning</u></p> <p>1 Enabled</p> <p>0 Disabled (default)</p>
9	Keep alive	Get/Set	BOOL	<p>Protocol probes idle connection (TCP sockets only).</p> <p>If the Keep alive attribute is set, the connection will be probed for the first time after it has been idle for 120 minutes. If a probe attempt fails, the connection will continue to be probed at intervals of 75s. The connection is terminated after 8 failed probe attempts.</p> <p><u>Value</u> <u>Meaning</u></p> <p>1 Enabled</p> <p>0 Disabled (default)</p>
10	IP Multicast TTL	Get/Set	UINT8	IP Multicast TTL value (UDP sockets only). Default = 1.
11	IP Multicast Loop	Get/Set	BOOL	<p>IP multicast loop back (UDP sockets only)</p> <p>Must belong to group in order to get the loop backed message</p> <p><u>Value</u> <u>Meaning</u></p> <p>1 Enabled (default)</p> <p>0 Disabled</p>

#	Name	Access	Data Type	Description						
12	Ack delay time	Get/Set	UINT16	Time for delayed ACKs in ms (TCP sockets only) Default = 200ms. Resolution is 50ms, i.e. 50...99 = 50ms, 100...149 = 100ms, 199 = 150ms etc.						
13	TCP No Delay	Get/Set	BOOL	Don't delay send to coalesce packets (TCP). <table border="1"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Delay (default)</td> </tr> <tr> <td>0</td> <td>Don't delay (turn off Nagle's algorithm on socket)</td> </tr> </tbody> </table>	Value	Meaning	1	Delay (default)	0	Don't delay (turn off Nagle's algorithm on socket)
Value	Meaning									
1	Delay (default)									
0	Don't delay (turn off Nagle's algorithm on socket)									
14	TCP Connect Timeout	Get/Set	UINT16	TCP Connect timeout in seconds (default = 75s)						

Command Details: Create

Category

Extended

Details

Command Code	03h
Valid for:	Object Instance

Description

This command creates a socket.

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

- Command Details

Field	Contents										
CmdExt[0]	(reserved, set to zero)										
CmdExt[1]	<table border="1"> <thead> <tr> <th>Value:</th> <th>Socket Type:</th> </tr> </thead> <tbody> <tr> <td>00h</td> <td>SOCK_STREAM, NON-BLOCKING (TCP)</td> </tr> <tr> <td>01h</td> <td>SOCK_STREAM, BLOCKING (TCP)</td> </tr> <tr> <td>02h</td> <td>SOCK_DGRAM, NON-BLOCKING (UDP)</td> </tr> <tr> <td>03h</td> <td>SOCK_DGRAM, BLOCKING (UDP)</td> </tr> </tbody> </table>	Value:	Socket Type:	00h	SOCK_STREAM, NON-BLOCKING (TCP)	01h	SOCK_STREAM, BLOCKING (TCP)	02h	SOCK_DGRAM, NON-BLOCKING (UDP)	03h	SOCK_DGRAM, BLOCKING (UDP)
Value:	Socket Type:										
00h	SOCK_STREAM, NON-BLOCKING (TCP)										
01h	SOCK_STREAM, BLOCKING (TCP)										
02h	SOCK_DGRAM, NON-BLOCKING (UDP)										
03h	SOCK_DGRAM, BLOCKING (UDP)										

- Response Details

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

Command Details: Delete

Category

Extended

Details

Command Code	04h
Valid for:	Object Instance

Description

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

- If the socket is of TCP-type and a connection is established, the connection is terminated with the RST-flag.
- To gracefully terminate a TCP-connection, it is recommended to use the 'Shutdown'-command (see below) before deleting the socket, causing the connection to be closed with the FIN-flag instead.
- Command Details

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

- Response Details
(no data)

Command Details: Bind

Category

Extended

Details

Command Code	10h
Valid for:	Instance

Description

This command binds a socket to a local port.

- Command Details

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

- Response Details

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

Command Details: Shutdown

Category

Extended

Details

Command Code	11h
Valid for:	Instance

Description

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

- Command Details

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

- Response Details
(no data)

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

Application initiates shutdown:

1. Send shutdown with CmdExt[1] set to 01h. This will send FIN-flag to host shutting down the send channel, note that the receive channel will still be operational.
2. Receive data on socket until error message Object specific error (EPIPE (13)) is received, indicating that the host closed the receive channel. If host does not close the receive channel use a timeout and progress to step 3.
3. Delete the socket instance. If step 2 timed out, RST-flag will be sent to terminate the socket.

Application initiates shutdown:

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

Command Details: Listen

Category

Extended

Details

Command Code 12h
Valid for: Instance

Description

This command puts a TCP socket in listening state.

- Command Details

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

- Response Details
(no data)

Command Details: Accept

Category

Extended

Details

Command Code	13h
Valid for:	Instance

Description

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

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

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

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

- Command Details
(no data)
- Response Details

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

Command Details: Connect

Category

Extended

Details

Command Code	14h
Valid for:	Instance

Description

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

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

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

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

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

- Command Details

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

- Response Details
(no data)

Command Details: Receive

Category

Extended

Details

Command Code	15h
Valid for:	Instance

Description

This command receives data from a connected socket. Message segmentation may be used to receive up to 1472 bytes (for more information, see [Message Segmentation, p. 114](#)).

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

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

NON-BLOCKING mode:	If no data is available on the socket, the error code 0006h (EWOULDBLOCK) will be returned.
BLOCKING mode:	The module will not issue a response until the operation has finished.

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

- Command Details

Field	Contents	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control bits	For more information, see Message Segmentation, p. 114
Data[0]	Receive data size (low)	Only used in the first segment
Data[1]	Receive data size (high)	

- Response Details

The data in the response may be segmented (For more information, see [Message Segmentation, p. 114](#)).

Field	Contents	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control bits	For more information, see Message Segmentation, p. 114
Data[0...n]	Received data	-

Command Details: Receive_From

Category

Extended

Details

Command Code 16h
Valid for: Instance

Description

This command receives data from an unconnected SOCK_DGRAM-socket. Message segmentation may be used to receive up to 1472 bytes (For more information, see [Message Segmentation, p. 114](#)).

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

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

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

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

- Command Details

Field	Contents	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control bits	For more information, see Message Segmentation, p. 114
Data[0]	Receive data size (low byte)	Only used in the first segment
Data[1]	Receive data size (high byte)	

- Response Details

The data in the response may be segmented (For more information, see [Message Segmentation, p. 114](#)).

Field	Contents	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control bits	For more information, see Message Segmentation, p. 114
Data[0]	Host IP address byte 4	The host address/port information is only included in the first segment. All data thereafter will start at Data[0]
Data[1]	Host IP address byte 3	
Data[2]	Host IP address byte 2	
Data[3]	Host IP address byte 1	
Data[4]	Host port number (low byte)	
Data[5]	Host port number (high byte)	
Data[6...n]	Received data	

Command Details: Send

Category

Extended

Details

Command Code	17h
Valid for:	Instance

Description

This command sends data on a connected socket. Message segmentation may be used to send up to 1472 bytes (For more information, see [Message Segmentation, p. 114](#)).

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

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

- Command Details

To allow larger amount of data (i.e. >255 bytes) to be sent, the command data may be segmented (For more information, see [Message Segmentation, p. 114](#)).

Field	Contents	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control	(For more information, see Message Segmentation, p. 114)
Data[0...n]	Data to send	-

- Response Details

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

Command Details: Send_To

Category

Extended

Details

Command Code	18h
Valid for:	Instance

Description

This command sends data to a specified host on an unconnected SOCK-DGRAM-socket. Message segmentation may be used to send up to 1472 bytes (For more information, see appendix For more information, see [Message Segmentation, p. 114](#)).

- Command Details

To allow larger amount of data (i.e. >255 bytes) to be sent, the command data may be segmented (For more information, see [Message Segmentation, p. 114](#)).

Field	Contents	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control	For more information, see Message Segmentation, p. 114
Data[0]	Host IP address byte 4	The host address/port information shall only be included in the first segment. All data thereafter must start at Data[0]
Data[1]	Host IP address byte 3	
Data[2]	Host IP address byte 2	
Data[3]	Host IP address byte 1	
Data[4]	Host port number (low byte)	
Data[5]	Host port number (high byte)	
Data[6...n]	Data to send	

- Response Details

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

Command Details: IP_Add_Membership

Category

Extended

Details

Command Code 19h
Valid for: Instance

Description

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

- Command Details

Field	Contents
CmdExt[0]	(reserved, set to zero)
CmdExt[1]	
Data[0]	Group IP address byte 4
Data[1]	Group IP address byte 3
Data[2]	Group IP address byte 2
Data[3]	Group IP address byte 1

- Response Details
(no data)

Command Details: IP_Drop_Membership

Category

Extended

Details

Command Code 1Ah
Valid for: Instance

Description

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

- Command Details

Field	Contents
CmdExt[0]	(reserved, set to zero)
CmdExt[1]	
Data[0]	Group IP address byte 4
Data[1]	Group IP address byte 3
Data[2]	Group IP address byte 2
Data[3]	Group IP address byte 1

- Response Details
(no data)

Command Details: DNS_Loopup

Category

Extended

Details

Command Code 1Bh
Valid for: Instance

Description

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

- Command Details

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

- Response Details (Success)

Field	Contents	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]		
Data[0]	IP address byte 4	IP address of the specified host
Data[1]	IP address byte 3	
Data[2]	IP address byte 2	
Data[3]	IP address byte 1	

Socket Interface Error Codes (Object Specific)

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

Error Code	Name	Meaning
1	ENOBUFS	No internal buffers available
2	ETIMEDOUT	A timeout event occurred
3	EISCONN	Socket already connected
4	EOPNOTSUPP	Service not supported
5	ECONNABORTED	Connection was aborted
6	EWOULDBLOCK	Socket cannot block because unblocking socket type
7	ECONNREFUSED	Connection refused
8	ECONNRESET	Connection reset
9	ENOTCONN	Socket is not connected
10	EALREADY	Socket is already in requested mode
11	EINVAL	Invalid service data
12	EMSGSIZE	Invalid message size
13	EPIPE	Error in pipe
14	EDESTADDRREQ	Destination address required
15	ESHUTDOWN	Socket has already been shutdown
16	(reserved)	-
17	EHAVEOOB	Out of band data available

Error Code	Name	Meaning
18	ENOMEM	No internal memory available
19	EADDRNOTAVAIL	Address is not available
20	EADDRINUSE	Address already in use
21	(reserved)	-
22	EINPROGRESS	Service already in progress
28	ETOOMANYREFS	Too many references
101	Command aborted	If a command is blocking on a socket, and that socket is closed using the Delete command, this error code will be returned to the blocking command.
102	DNS name error	Failed to resolve the host name (name error response from DNS server).
103	DNS timeout	Timeout when performing a DNS lookup.
104	DNS command failed	Other DNS error.

Message Segmentation

General

Category: Extended

The maximum message size supported by the Anybus CompactCom 40 is normally 1524 bytes. In some applications a maximum message size of 255 bytes is supported, e.g. if an Anybus CompactCom 40 is to replace an Anybus CompactCom 30 without any changes to the application. The maximum socket message size is 1472. To ensure support for socket interface messages larger than 255 bytes a segmentation protocol is used.



The segmentation bits have to be set for all socket interface messages, in the commands where segmentation can be used, whether the messages have to be segmented or not.

The segmentation protocol is implemented in the message layer and must not be confused with the fragmentation protocol used on the serial host interface. Consult the general *Anybus CompactCom 40 Software Design Guide* for further information.

The module supports 1 (one) segmented message per instance

Command Segmentation

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

Command segmentation is used for the following commands (Socket Interface Object specific commands):

- Send
- Send To

When issuing a segmented command, the following rules apply:

- When issuing the first segment, FS must be set.
- When issuing subsequent segments, both FS and LS must be cleared.
- When issuing the last segment, the LF-bit must be set.
- For single segment commands (i.e. size less or equal to the message channel size), both FS and LS must be set.
- The last response message contains the actual result of the operation.
- The command initiator may at any time abort the operation by issuing a message with AB set.
- If a segmentation error is detected during transmission, an error message is returned, and the current segmentation message is discarded. Note however that this only applies to the current segment; previously transmitted segments are still valid.

Segmentation Control Bits (Command)

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

Segmentation Control Bits (Response)

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

Response Segmentation

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

Response segmentation is used for responses to the following commands (Socket Interface Object specific commands):

- Receive
- Receive From

When receiving a segmented response, the following rules apply:

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

Segmentation Control bits (Command)

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

Segmentation Control bits (Response)

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

12.8 SMTP Client Object (09h)

Category

Extended

Object Description

This object groups functions related to the SMTP client.

Supported Commands

Object:	Get_Attribute
	Create
	Delete
	Send e-mail from file (see below)
Instance:	Get_Attribute
	Set_Attribute
	Send e-mail (see below)

Object Attributes (Instance #0)

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

Instance Attributes (Instance #1)

Instances are created dynamically by the application.

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

Command Details: Create

Category

Extended

Details

Command Code 03h

Valid for: Object

Description

This command creates an e-mail instance.

- Command Details

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

- Response Details

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

Command Details: Delete

Category

Extended

Details

Command Code 04h

Valid for: Object

Description

This command deletes an e-mail instance.

- Command Details

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

- Response Details
(no data)

Command Details: Send E-mail From File

Category

Extended

Details

Command Code 11h
Valid for: Object

Description

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

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

```
[To]
recipient

[From]
sender

[Subject]
email subject

Se [Headers]
extra headers, optional

[Message]
actual email message
```

- **Command Details**

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

- **Response Details**
(no data)

Command Details: Send E-mail

Category

Extended

Details

Command Code	10h
Valid for:	Object

Description

This command sends the specified e-mail instance.

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

Object Specific Error Codes

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

12.9 File System Interface Object (0Ah)

Category

Extended

Object Description

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

This object is thoroughly described in *Anybus CompactCom 40 Software Design Guide*.

12.10 Network Ethernet Object (0Ch)

Category

Extended

Object Description

This object provides Ethernet-specific information to the application.

Supported Commands

Object: Get_Attribute

Instance: Get_Attribute

Object Attributes (Instance #0)

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

Instance Attributes (Instance #1)

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

12.11 Functional Safety Module Object (11h)

Category

Extended

Object Description

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

Supported Commands

Object: Get_Attribute
Error_Confirmation

Instance: Get_Attribute

Object Attributes (Instance #0)

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

Instance Attributes (Instance #1)

#	Name	Access	Data Type	Description
1	State	Get	UINT8	Current state of the Safety Module Please consult the manual for the Safety Module used.
2	Vendor ID	Get	UINT16	Identifies vendor of the Safety Module. E.g. 0001h (HMS Industrial Networks) Please consult the manual for the Safety Module used.
3	IO Channel ID	Get	UINT16	Describes the IO Channels that the Safety Module is equipped with. Please consult the manual for the Safety Module used.
4	Firmware version	Get	Struct of UINT8 (Major) UINT8 (Minor) UINT8 (Build)	Safety Module firmware version.
5	Serial number	Get	UINT32	32 bit number, assigned to the Safety Module at production. Please consult the manual for the Safety Module used.
6	Output data	Get	Array of UINT8	Current value of the Safety Module output data, i.e. data FROM the network Note: This data is unsafe, since it is provided by the Anybus CompactCom module.
7	Input data	Get	Array of UINT8	Current value of the Safety Module input data, i.e. data sent TO the network. Note: This data is unsafe, since it is provided by the Anybus CompactCom module.
8	Error counters	Get	Struct of UINT16 (ABCC DR) UINT16 (ABCC SE) UINT16 (SM DR) UINT16 (SM SE)	Error counters (each counter stops counting at FFFFh) ABCC DR: Responses (unexpected) from the Safety Module, discarded by the Anybus CompactCom module. ABCC SE: Serial reception errors detected by the Anybus CompactCom module. SM DR: Responses (unexpected) from the Anybus CompactCom module, discarded by the Safety Module. SM SE: Serial reception errors detected by the Safety Module.
9	Event log	Get	Array of UINT8	Latest Safety Module event information (if any) is logged to this attribute. Any older event information is erased when a new event is logged. For evaluation by HMS support.
10	Exception information	Get	UINT8	If the Exception Code in the Anybus object is set to "Safety communication error" (09h), additional exception information is presented here, see table below.
11	Bootloader version	Get	Struct of UINT8 Major UINT8 Minor	Safety Module bootloader version.

Exception Information

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

Value	Exception Information
00h	No information
01h	Baud rate not supported
02h	No start message
03h	Unexpected message length
04h	Unexpected command in response
05h	Unexpected error code
06h	Safety application not found
07h	Invalid safety application CRC
08h	No flash access
09h	Answer from wrong safety processor during boot loader communication
0Ah	Boot loader timeout
0Bh	Network specific parameter error
0Ch	Invalid IO configuration string
0Dh	Response differed between the safety microprocessors (e.g. different module types)
0Eh	Incompatible module (e.g. supported network)
0Fh	Max number of retransmissions performed (e.g. due to CRC errors)
10h	Firmware file error
11h	The cycle time value in attribute #4 in the Functional Safety Host Object can not be used with the current baud rate
12h	Invalid SPDU input size in start-up telegram
13h	Invalid SPDU output size in start-up telegram
14h	Badly formatted input SPDU
15h	Anybus CompactCom to safety module initialization failure

Command Details: Error_Confirmation

Category

Extended

Details

Command Code	10h
Valid for:	Object

Description

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

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

Object Specific Error Codes

Error Code	Description	Comments
01h	The safety module rejected a message.	Error code sent by safety module is found in MsgData [2] and MsgData[3].
02h	Message response from the safety module has incorrect format (for example, wrong length).	-

13 Host Application Objects

13.1 General Information

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

Standard Objects

- “Application Object (FFh)” (see Anybus CompactCom 40 Software Design Guide)
- “Application Data Object (FEh)” (see Anybus CompactCom 40 Software Design Guide)
- [Sync Object \(EEh\), p. 127](#)
- “Application File System Object (EAh)” (see Anybus CompactCom 40 Software Design Guide)

Network Specific Objects:

- [PROFINET IO Object \(F6h\), p. 131](#)
- [Ethernet Host Object \(F9h\), p. 149](#)
- [Functional Safety Object \(E8h\), p. 152](#)

13.2 Sync Object (EEh)

Category

Extended

Object Description

This object contains the host object sync settings.

How to Use Sync Functionality

Please note that it is up to the application to make sure that the product is synchronous. The module will only ask the application for:

Output processing time	The time in ns it takes from that the application is notified of new process data until valid output.
Input processing time	The time in ns from input capture time until the process data has been copied to the module.

The module will notify the application of:

Output valid	This is the time in ns relative to the sync event, for when the outputs shall be used.
Input capture	This is the time in ns relative to the sync event, for when the input process data shall be captured.

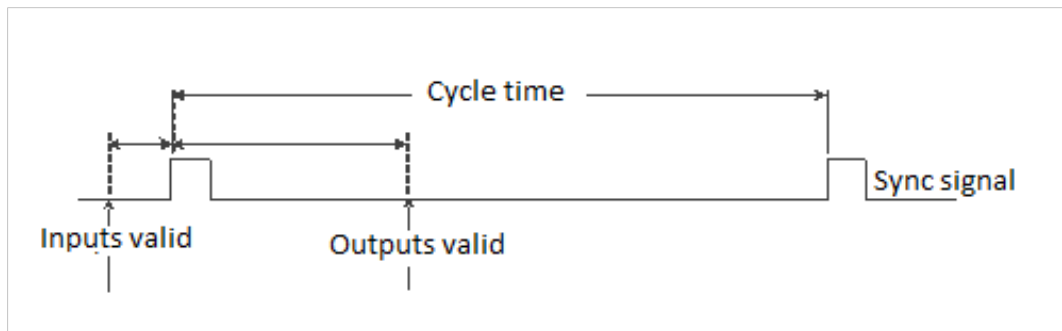


Fig. 9

All modules have the sync signal in the connector and as an interrupt to the host application.

Output Processing Time

This needs to be calculated and measured by the application designer. The time consists of two delays:

- The delay added by the module, from when message is available on the network until it is available for the host application. For copper based PROFINET, this delay is zero, for fiber optic PROFINET the formula below can be used to calculate the delay.
- The delay added by the application. This is the time it takes for the application from it is notified that new process data has arrived, to when the process data is copied and output is valid.

To calculate the delay added by the module the following formula shall be used:

$$t_{ABCC} = 15600ns + (t_{tout} * 734ns) + \sum_{n=1}^{ttout} (\text{data length of } n * 35ns)$$

Fig. 10

t_{tout} = total number of output submodules

Input Processing Time

This needs to be calculated and measured by the application designer. The time consists of two delays:

- The delay added by the module
The time from when the message is available in the module until it is available on the network. For copper based PROFINET, this delay is 12 ?s, for fiber optic PROFINET the formula below can be used to calculate the delay.
- The delay added by the application
This is the time it takes for the application from inputs are captured until the inputs are available for the module.

To calculate the delay added by the module the following formula shall be used:

$$t_{ABCC} = 12980ns + (t_{tin} * 789ns) + \sum_{n=1}^{ttin} (\text{data length of } n * 35ns) + (t_{tout} * 240ms)$$

Fig. 11

t_{tout} = total number of output submodules

t_{tin} = total number of input submodules

GSDML Entries

When the total time has been calculated for every module, the following must be added to the GSDML file:

```
IsochroneModeInRT_Classes="RT_CLASS_3" in the <InterfacesubmoduleItem>
<IsochroneMode T_DC_Base="8" T_DC_Min="1" T_DC_Max="16" T_IO_Base="1000" T_IO_Ou
```

X = output processing time

Y = input processing time

Using the Sync Attributes

- Nonisochronous mode** In this mode, the isochronous mode is not enabled in the PLC configuration tool. The application does not have to implement any of the attributes in the Sync object. The module will still write to attribute #1, Cycle time, but it will disregard the answer from the application.
- Isochronous mode** In this mode the isochronous mode is enabled in the PLC configuration tool. The application must then implement all attributes of the Sync object. The first attribute read by the module will be #8, supported sync modes. The application must respond with "1" for the isochronous AR to be established.

By start-up: attribute #8, supported sync modes, is read:

1. For every submodule where ISOM is activated in the PLC:
 - Read OutputProcessing
 - Read InputProcessing
 - Read MinCycleTime
2. Write sync mode
 - NACK && ISOM enabled in PLC: NACK
3. Write cycle time
 - NACK && ISOM enabled in PLC: NACK
4. If ISOM is activated:
 - Set OutputValid
 - If NACK: NACK
 - Set InputCapture
 - If NACK: NACK

Supported Commands

Object:	Get_Attribute
Instance:	Get_Attribute Set_Attribute

Object Attributes (Instance #0)

(Consult the general *Anybus CompactCom 40 Software Design Guide* for further information.)

Instance Attributes (Instance #1)

#	Name	Access	Corresponding term for PROFINET IRT	Data Type	Value
1	Cycle time	Get/Set	RR*SCF	UINT32	The module supports cycle times as low as 250 μ s
2	Output valid	Get/Set	T_IO_Output	UINT32	Output valid point relative to SYNC events, in nanoseconds Default value: 0
3	Input capture	Get/Set	T_IO_Input	UINT32	Input capture point relative to SYNC events, in nanoseconds Default value: 0
4	Output processing	Get	T_IO_OutputMin	UINT32	Minimum required time, in nanoseconds, between RDPDI interrupt and "Output valid"
5	Input processing	Get	T_IO_InputMin	UINT32	Maximum required time, in nanoseconds, from "Input capture" until write process data has been completely written to the Anybus CompactCom module
6	Min cycle time	Get	T_DC_Min	UINT32	The module supports cycle times as low as 250 μ s
7	Sync mode	Get/Set	-	UINT16	This attribute is used to select synchronization mode. It enumerates the bits in attribute 8 0: Nonsynchronous operation. (Default value if nonsynchronous operation is supported) 1: Synchronous operation 2 - 65535: Reserved. Any attempt to set sync mode to an unsupported value shall generate an error response
8	Supported sync modes	Get	-	UINT16	A list of the synchronization modes the application supports. Each bit corresponds to a mode in attribute 7 Bit 0: 1 = Nonsynchronous mode supported Bit 1: 1 = Synchronous mode supported Bit 2 - 15: Reserved (0)

13.3 PROFINET IO Object (F6h)

Category

Basic, extended

Object Description

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

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

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

See also...

- [Network PROFINET IO Object \(0Eh\), p. 82](#)
- [Flowchart — Record Data Access, p. 157](#)
- *Anybus CompactCom 40 Software Design Guide*, "Error Codes"

Supported Commands

Object:	Get_Attribute Get_Record (See below) Set_Record (See below) AR_Check_Ind (See below) Cfg_Mismatch_Ind (See below) Expected_Ident_Ind (See below) End_Of_Prm_Ind (See below) AR_Abort_Ind (See below) Indicate_Device (See below)
Instance:	Get_Attribute

Object Attributes (Instance #0)

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

Instance Attributes (Instance #1)

Basic

If an attribute is not implemented, the default value will be used.

#	Name	Access	Data Type	Default value/Comment
1	Device ID	Get	UINT16	0010h Identifies the device. (Assigned by manufacturer)
2	Vendor ID (I&M Manufacturer ID)	Get	UINT16	010Ch (HMS Industrial Networks AB) Characterizes the device. (Assigned by manufacturer); up to 25 characters.
3	Station Type	Get	Array of CHAR	"ABCC40-PIR" I&M0 Parameter: Order ID of device; up to 20 characters.
8	I&M Order ID	Get	Array of CHAR	"ABCC40-PIR"
9	I&M Serial Number	Get	Array of CHAR	Assigned during manufacturing I&M0 Parameter: Serial number of device; up to 16 characters.
19	System Description	Get	Array of CHAR	"HMS Industrial Networks Anybus Compact-Com 40" Up to 255 characters.

GSD Entries

The GSDML entries below must match the values of the corresponding attributes in the PROFINET IO object.

- Attributes #1 and #2 correspond to the following entry in the GSDML file:

```
<DeviceIdentity VendorID="0x010C" DeviceID="0x0010">
```

- Attribute #3 corresponds to the following entry in the GSDML file:

```
DNS_CompatibleName="ABCC40-PIR"
```

- Attribute #8 correspond to the following entry in the GSDML file:

```
<OrderNumber Value="ABCC40-PIR"/>
```

Extended

- If an attribute is not implemented, the default value will be used.
- The Anybus module in itself does not alter its behavior based on the value of attributes #13 and #14. The host application has to implement the corresponding functionality..
- The module is preprogrammed with a valid Mac address. To use that address, do not implement attributes #17 and #18.

#	Name	Access	Data Type	Default Value	Comment
4	MaxAr	Get	UINT32	0003h	Max. no.of simultaneous ARs. (Range 1... 3)
5	(Reserved)	-	-	-	Reserved for future use
6					
7	Record Data Mode	Get	UINT8	00h	This setting affects how Record Data requests are treated, and holds a bit field as follows: <u>Bit 0:Index 0... 7FFFh:</u> 0: Normal Mode 1: Transparent Mode <u>Bit 1:Index AFF0h... AFFFh:</u> Reserved (replaced with the command IM_Options)
10	I&M Hardware Revision	Get	UINT16	Hardware Rev.	I&M0 Parameter: Hardware revision of device; FFFFh indicates availability of profile specific information
11	I&M Software Revision	Get	Array of CHAR	Software Rev.	I&M0 Parameter: Software revision of device. <u>Byte:Value:Meaning:</u> 0: 'V' Official release 'R' Revision 'P' Prototype 'U' Under test 'T' Test device 1: 0... 255 Major Version 2: 0... 255 Minor Version 3: 0... 255 Internal Change Please note that version V255.255.255 indicates availability of profile specific information.
12	I&M Revision Counter	Get	UINT16	0000h	I&M0 Parameter: Revision counter of device; a changed value marks a change of the hardware or its parameters.
13	I&M Profile ID	Get	UINT16	F600h (Generic Device)	I&M0 Parameter: If the application supports a specific profile, the Profile ID must be defined here. It shall equal the API number for the profile.. Please not that: this will not change the behavior of the module, since it does not handle profiles.
14	I&M Profile Specific Type	Get	UINT16	Generic Profile: 0004h (Communication Module)	I&M0 Parameter: If the application supports a specific profile, the profile specific type must be defined here (value is defined by the profile). Please not that: this will not change the behavior of the module, since it does not handle profiles.
15	(Reserved)	-	-	-	Reserved, not used

#	Name	Access	Data Type	Default Value	Comment
16					
17	Port 1 MAC address	Get	Array of UINT8	-	6 byte physical address value; overrides the preprogrammed Mac address. Do not implement this attribute if the preprogrammed Mac address is to be used.
18	Port 2 MAC address	Get	Array of UINT8	-	6 byte physical address value; overrides the preprogrammed Mac address. Do not implement this attribute if the preprogrammed Mac address is to be used.
20	(Reserved)	-	-	-	Reserved, not used
21					
22					
23					
24	Custom Station Name	Get	Array of CHAR	Product specific. (e.g. "abcc40-pir")	When using shift register mode, this value is used to create the station name. If for example the attribute has the value abcc40-pir, the station name will be abcc40-pir-XYZ where XYZ is the value of DIP2. Valid characters are letters "a" to "z", numbers "0" to "9", hyphens and periods. The attribute value must begin with a letter. Maximum number of elements in array: 58 For more information, see the <i>Anybus CompactCom M40 Hardware Design Guide</i> .

Command Details: Get_Record

Category

Extended

Details

Command Code	10h
Valid for:	Object Instance

Description

The module issues this command in the following situations:

- Module receives a Record Data Read request towards an API other than 0 (zero).
- Module receives a Record Data Read request towards API 0, but the record in question is handled in Transparent Mode.

See instance attribute #7 for more information about Transparent Mode.

See [Flowchart — Record Data Access, p. 157](#) for more information.

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

- Command Details

Field	Contents	Comments
CmdExt[0]	AR handle (low byte)	Handle for Application Relationship (AR)
CmdExt[1]	AR handle (high byte)	
MsgData[0]	API (low word, low byte)	Application Process Instance (API)
MsgData[1]	API (low word, high byte)	
MsgData[2]	API (high word, low byte)	
MsgData[3]	API (high word, high byte)	
MsgData[4]	Slot (low byte)	Slot number of request
MsgData[5]	Slot (high byte)	
MsgData[6]	Subslot (low byte)	Subslot number of request
MsgData[7]	Subslot (high byte)	
MsgData[8]	Index (low byte)	Index of request
MsgData[9]	Index (high byte)	
MsgData[10]	Length (low byte)	Range: 1 - 1524 MsgData[11] is only available if the length value exceeds 255.
MsgData[11]	Length (high byte)	

- Response Details (Success)

Field	Contents	Comments
CmdExt[0...1]	(reserved)	(set to zero)
MsgData[0...n]	Data (up to 1524 bytes)	Data to be returned in the Record Data Read response.

- Response Details (Error)

Field	Contents	Comments
CmdExt[0...1]	(reserved)	(set to zero)
MsgData[0]	FFh	Object specific error
MsgData[1]	Error Code 1	See details below “Details: Error Code 1” on page 146
MsgData[2]	Error Code 2	User specific error code
MsgData[3]	Additional Data 1	API specific. Set to zero if no Additional Data 1 is defined.
MsgData[4]	Additional Data 2	User specific. Set to zero if no Additional Data 2 is defined.

See also...

- [“Command Details: Set Record” on page 137](#)
- [“Flowchart - Record Data Access” on page 154](#)

Command Details: Set_Record

Category

Extended

Details

Command Code	11h
Valid for:	Object Instance

Description

The module issues this command in the following situations:

- Module receives a Record Data Write request towards an API other than 0 (zero).
- Module receives a Record Data Write request towards API 0, but the record in question is handled in Transparent Mode

See instance attribute #7 for more information about Transparent Mode.

See [“Flowchart - Record Data Access” on page 154](#) for more information.

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

- Command Details

Field	Contents	Comments
CmdExt[0]	AR handle (low byte)	Handle for Application Relationship (AR)
CmdExt[1]	AR handle (high byte)	
MsgData[0]	API (low word, low byte)	Application Process Instance (API)
MsgData[1]	API (low word, high byte)	
MsgData[2]	API (high word, low byte)	
MsgData[3]	API (high word, high byte)	
MsgData[4]	Slot (low byte)	Slot number of request
MsgData[5]	Slot (high byte)	
MsgData[6]	Subslot (low byte)	Subslot number of request
MsgData[7]	Subslot (high byte)	
MsgData[8]	Index (low byte)	Index of request
MsgData[9]	Index (high byte)	
MsgData[10]	(reserved)	(set to zero)
MsgData[11..n]	Data (up to 1512 bytes)	Data from the Record Data Write request.

- Response Details (Success)

(no data)

- Response Details (Error)

Field	Contents	Comments
CmdExt[0...1]	(reserved)	(set to zero)
MsgData[0]	FFh	Object specific error
MsgData[1]	Error Code 1	See Details: Error Code 1, p. 148
MsgData[2]	Error Code 2	User specific error code
MsgData[3]	Additional Data 1	API specific. Set to zero if no Additional Data 1 is defined.
MsgData[4]	Additional Data 2	User specific. Set to zero if no Additional Data 2 is defined.

See also...

- Command details for “Get_Record”
- [Flowchart — Record Data Access, p. 157](#)

Command Details: AR_Check_Ind**Category**

Extended

Details

Command Code	14h
Valid for:	Object Instance

Description

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

- Command Details

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

- Response Details

(no data)

Command Details: Cfg_Mismatch_Ind

Category

Extended

Details

Command Code	15h
Valid for:	Object Instance

Description

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

It is optional to implement support for this command.

- Command Details

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

- Response Details
(no data)

Command Details: Expected_Ident_Ind

Category

Extended

Details

Command Code	1Bh
Valid for:	Object Instance

Description

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

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

Expected_Ident_Ind is similar to AR_Info_Ind but uses a different segmentation protocol that shall be used for the 40 series concept.

For very large configurations the Expected Identification can not fit into one message (max message data size = 1524 bytes). If this happens the message will be truncated.

It is optional to implement support for this command.

- Command Details

Field	Contents	Comments
CmdExt[0]	Reserved	For segmented messages the CmdExt[1] byte has been reserved for segment bits.
CmdExt[1]	Cmd segment bits	
MsgData[0]	AR handle (low byte)	Handle for the Application Relationship.
MsgData[1]	AR handle (high byte)	
MsgData[2..n]	Data field	The first two bytes in the initial block of the Data field indicates the number of modules in the configuration. Each module is represented by a Module block, followed by a number of Submodule blocks (provided that the module in question contains submodules). See "Data Format" below for coding of the data field.

- Response Details

Field	Contents	Comments
CmdExt[0]	Reserved	(set to zero)
CmdExt[1]		
MsgData[0]	Continue/Block	<p><u>Value:</u> <u>Meaning:</u></p> <p>0 Continue. The host application will not perform any changes to the Real Identification, A connect response may be sent on the network immediately following this response.</p> <p>1 Block. The host application will perform changes in the Real identification in response to the received Expected identification. A connect response will not be sent on the network until Ident_Change_Done is sent by the host application</p> <p>Returning any error response or providing no data/ too much data will give the same result as responding with Continue (0). The timeout is 300 seconds. A "Continue" response or the Ident_Change_Done command should be sent well within this time limit.</p>

Data Format

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

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

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

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

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

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

Example

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

Fig. 12

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

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

Command Details: End_Of_Prm_Ind

Category

Extended

Details

Command Code	17h
Valid for:	Object Instance

Description

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

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

- Command Details

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

- Response Details

Field	Contents	Comments						
CmdExt[0]	(reserved)	(set to zero)						
CmdExt[1]	AR handle (high byte)							
MsgData[0]	Subslot (high byte)	<table> <tr> <td><u>Value:</u></td> <td><u>Meaning:</u></td> </tr> <tr> <td>0:</td> <td>Ready for Data Exchange</td> </tr> <tr> <td>1:</td> <td>Not ready for data exchange (Appl_State_Ready must be issued at a later stage)</td> </tr> </table>	<u>Value:</u>	<u>Meaning:</u>	0:	Ready for Data Exchange	1:	Not ready for data exchange (Appl_State_Ready must be issued at a later stage)
<u>Value:</u>	<u>Meaning:</u>							
0:	Ready for Data Exchange							
1:	Not ready for data exchange (Appl_State_Ready must be issued at a later stage)							

See also...

- Appl_State_Ready, details in [Network PROFINET IO Object \(0Eh\)](#), p. 82

Command Details: AR_Abort_Ind

Category

Extended

Details

Command Code 19h
Valid for: Object Instance

Description

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

- Command Details

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

- Response Details (no data)

Command Details: Indicate_Device

Category

Extended

Details

Command Code 1Eh
Valid for: Object Instance

Description

This command is optional. The module issues this command to inform the application that the DCP command Set Control/Signal has been received on the network. This is used by engineering tools to identify the node on the network. The application should flash its dedicated Network Status LED for 3 seconds after receiving this command.

- Command Details

Field	Contents	Comments
CmdExt[0]	(reserved)	
CmdExt[1]		

- Response Details
(no data)

Details: Error Code 1

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

- Get_Record
- Set_Record
- Get_IM_Record
- Set_IM_Record

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

See also...

- Command details for Get_Record
- Command details for Set_Record

13.4 Ethernet Host Object (F9h)

Object Description

This object implements Ethernet features in the host application.

Supported Commands

Object:	Get_Attribute
Instance:	Get_Attribute Set_Attribute

Object Attributes (Instance #0)

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

Instance Attributes (Instance #1)

- If an attribute is not implemented, the default value will be used.
- The module is preprogrammed with a valid MAC address. To use that address, do not implement attribute #1.
- Do not implement attributes #9 and #10, only used for PROFINET devices, if the module shall use the preprogrammed MAC addresses.
- If new MAC addresses are assigned to a PROFINET device, these addresses (in attributes #1, #9, and #10) have to be consecutive, e.g. (xx:yy:zz:aa:bb:01), (xx:yy:zz:aa:bb:02), and (xx:yy:zz:aa:bb:03) with the first five octets not changing.

#	Name	Access	Data Type	Default Value	Comment
1	MAC address	Get	Array of UINT8	-	6 byte physical address value; overrides the preprogrammed Mac address. Note that the new Mac address value must be obtained from the IEEE. Do not implement this attribute if the preprogrammed Mac address is to be used.
2	Enable HICP	Get	BOOL	True (Enabled)	Enable/Disable HICP
3	Enable Web Server	Get	BOOL	True (Enabled)	Enable/Disable Web Server
4	(reserved)				Reserved for Anybus CompactCom 30 applications.
5	Enable Web ADI access	Get	BOOL	True (Enabled)	Enable/Disable Web ADI access
6	Enable FTP server	Get	BOOL	True (Enabled)	Enable/Disable FTP server
7	Enable admin mode	Get	BOOL	False (Disabled)	Enable/Disable FTP admin mode
8	Network Status	Set	UINT16	-	See below.

#	Name	Access	Data Type	Default Value	Comment
9	Port 1 MAC address	Get	Array of UINT8	-	Note: This attribute is only valid for PROFINET devices. 6 byte MAC address for port 1 (mandatory for the LLDP protocol). This setting overrides any Port MAC address in the host PROFINET IO Object. Do not implement this attribute if the pre-programmed Mac address is to be used.
10	Port 2 MAC address	Get	Array of UINT8	-	Note: This attribute is only valid for PROFINET devices. 6 byte MAC address for port 2 (mandatory for the LLDP protocol). This setting overrides any Port MAC address in the host PROFINET IO Object. Do not implement this attribute if the pre-programmed Mac address is to be used.
11	Enable ACD	Get	BOOL	1 = True (Enabled)	Enable/Disable ACD protocol. If ACD functionality is disabled using this attribute, the ACD attributes in the CIP TCP/IP object (F5h) are not available.
12	Port 1 State	Get	ENUM	0 (Enabled)	The state of Ethernet port 1. Note: This attribute is not read by EtherCAT devices, where Port 1 is always enabled. 00h: Enabled 01h: Disabled. The port is treated as existing. References to the port can exist, e.g. in network protocol or on website.
13	Port 2 State	Get	ENUM	0 (Enabled)	The state of Ethernet port 2. Note: This attribute is not read by EtherCAT devices, where Port 1 is always enabled. 00h: Enabled 01h: Disabled. The port is treated as existing. References to the port can exist, e.g. in network protocol or on website. 02h: Note: EIP only. A port shall only be configured to this state if it does not exist physically. Inactive. The port is treated as non-existing. No references to the port exists. Functionality requiring two ports is disabled.
14	(reserved)				
15	Enable reset from HICP	Get	BOOL	0 = False	Enables the option to reset the module from HICP.
16	IP configuration	Set	Struct of: UINT32 (IP address) UINT32 (Subnet mask) UINT32 (Gateway)	N/A	Whenever the configuration is assigned or changed, the Anybus CompactCom module will update this attribute.

#	Name	Access	Data Type	Default Value	Comment
17	IP address byte 0-2	Get	Array of UINT8[3]	[0] = 192 [1] = 168 [2] = 0	First three bytes in IP address. Used in "shift register mode" if the configuration switch value is set to 1-245. In that case the IP address will be set to: Y[0].Y[1].Y[2].X Where Y0-2 is configured by this attribute and the last byte X by the configuration switch.
18	Get	Ethernet PHY Configuration	Array of BITS16	0x0000 for each port	Ethernet PHY configuration bit field. The length of the array shall equal the number of Ethernet ports of the product. Each element represents the configuration of one Ethernet port (element #0 maps to Ethernet port #1, element #1 maps to Ethernet port #2 and so on). Note: Only valid for EtherNet/IP and Modbus-TCP devices. Bit 0: Auto negotiation fallback duplex 0 = Half duplex 1 = Full duplex Bit 1-15: Reserved

Network Status

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

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

13.5 Functional Safety Object (E8h)

Category

Extended

Object Description



Do not implement this object if a safety module is not used.

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

Supported Commands

Object: Get_Attribute

Instance: Get_Attribute

Object Attributes (Instance #0)

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

Instance Attributes (Instance #1)

#	Name	Access	Data Type	Default Value	Comment
1	Safety enabled	Get	BOOL	-	When TRUE, enables communication with the Safety Module. Note: If functional safety is not supported, this attribute must be set to FALSE.
2	Baud Rate	Get	UINT32	1020 kbit/s	This attribute sets the baud rate of the communication in bits/s between the Anybus CompactCom and the Safety Module. Valid values: <ul style="list-style-type: none"> • 625 kbit/s • 1000 kbit/s • 1020 kbit/s (default) Any other value set to this attribute, will cause the module to enter the EXCEPTION state. The attribute is optional. If not implemented, the default value will be used. Note: The host application shall never implement this attribute when using the IXXAT Safe T100.

#	Name	Access	Data Type	Default Value	Comment
3	IO Configuration	Get	Array of UINT8	-	<p>Optional attribute. Manufacturer specific settings of the digital I/O of the Safety Module. See the manual of the Safety Module used for information.</p> <p>Note: The host application shall never implement this attribute when using the IXXAT Safe T100.</p>
4	Cycle Time	Get	UINT8	-	<p>Communication cycle time between the Anybus and the Safety module in milliseconds.</p> <p>Note: The host application shall never implement this attribute when using the IXXAT Safe T100.</p> <p>Valid values:</p> <ul style="list-style-type: none"> • 2 ms • 4 ms • 8 ms • 16 ms <p>If another value is set in this attribute the Anybus will enter Exception state.</p> <p>Optional attribute; If not implemented the minimum cycle time for the chosen baud rate will be used:</p> <ul style="list-style-type: none"> • 2 ms for 1020 kbit/s • 2 ms for 1000 kbit/s • 4 ms for 625 kbit/s <p>The ABCC validates the cycle time according to the minimum values above. If e.g. baud rate is 625 kbit/s and the cycle time is set to 2 ms the ABCC will enter the EXCEPTION state.</p>

This page intentionally left blank

A Categorization of Functionality

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

A.1 Basic

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

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

A.2 Extended

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

Some of the functionality offered may be specialized and/or seldom used. As most of the available network functionality is enabled and accessible, access to the specification of the industrial network may be required.

B Anybus Implementation Details

B.1 SUP-Bit Definition

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

B.2 Anybus State Machine

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

Anybus State	Implementation	Comment
WAIT_PROCESS	The Anybus stays in this state until an IO connection with an IO controller is opened.	-
ERROR	Configuration data mismatch or initial parameter error.	-
PROCESS_ACTIVE	IO connection established with IO controller and valid output data has been provided at least once.	-
IDLE	IO controller with which an IO connection is established is in STOP mode or the IO controller has not provided valid output at least once.	Some IO controllers will not provide valid output data in the first cycles following a successful connection.
EXCEPTION	Turn module status LED red, to indicate major fault, turn network status LED off, and hold Ethernet MAC in reset.	Some kind of unexpected behavior, for example watchdog timeout. See also instance 1, attribute 7 in Network Object (03h) , p. 74.

B.3 Application Watchdog Timeout Handling

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

C Flowcharts

C.1 Flowchart — Record Data Access

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

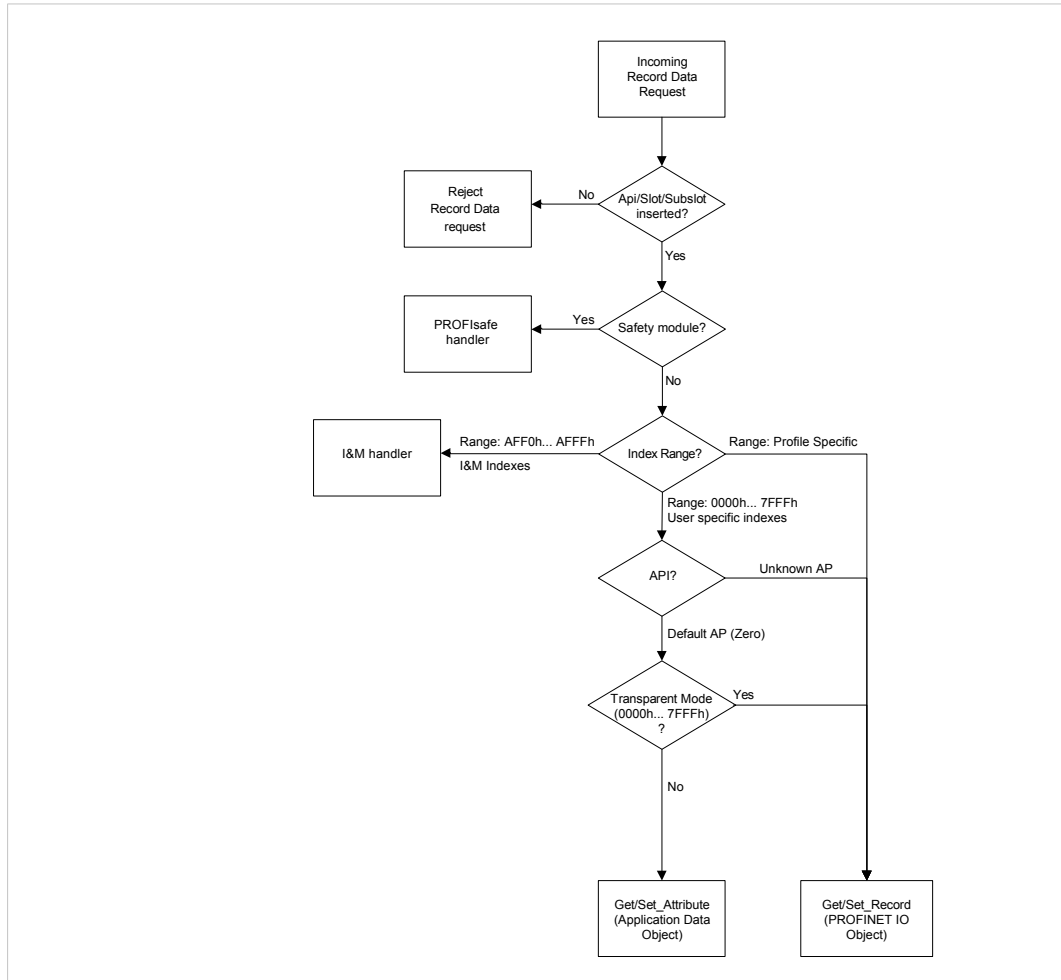


Fig. 13

See also...

- [Application Data Instances \(ADIs\), p. 14](#)
- [PROFINET IO Object \(F6h\), p. 131](#)
- Details for command Get_Record in the [PROFINET IO Object \(F6h\), p. 131](#)
- Details for command Set_Record in the [PROFINET IO Object \(F6h\), p. 131](#)

C.2 Flowchart — I&M Record Data Handling

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

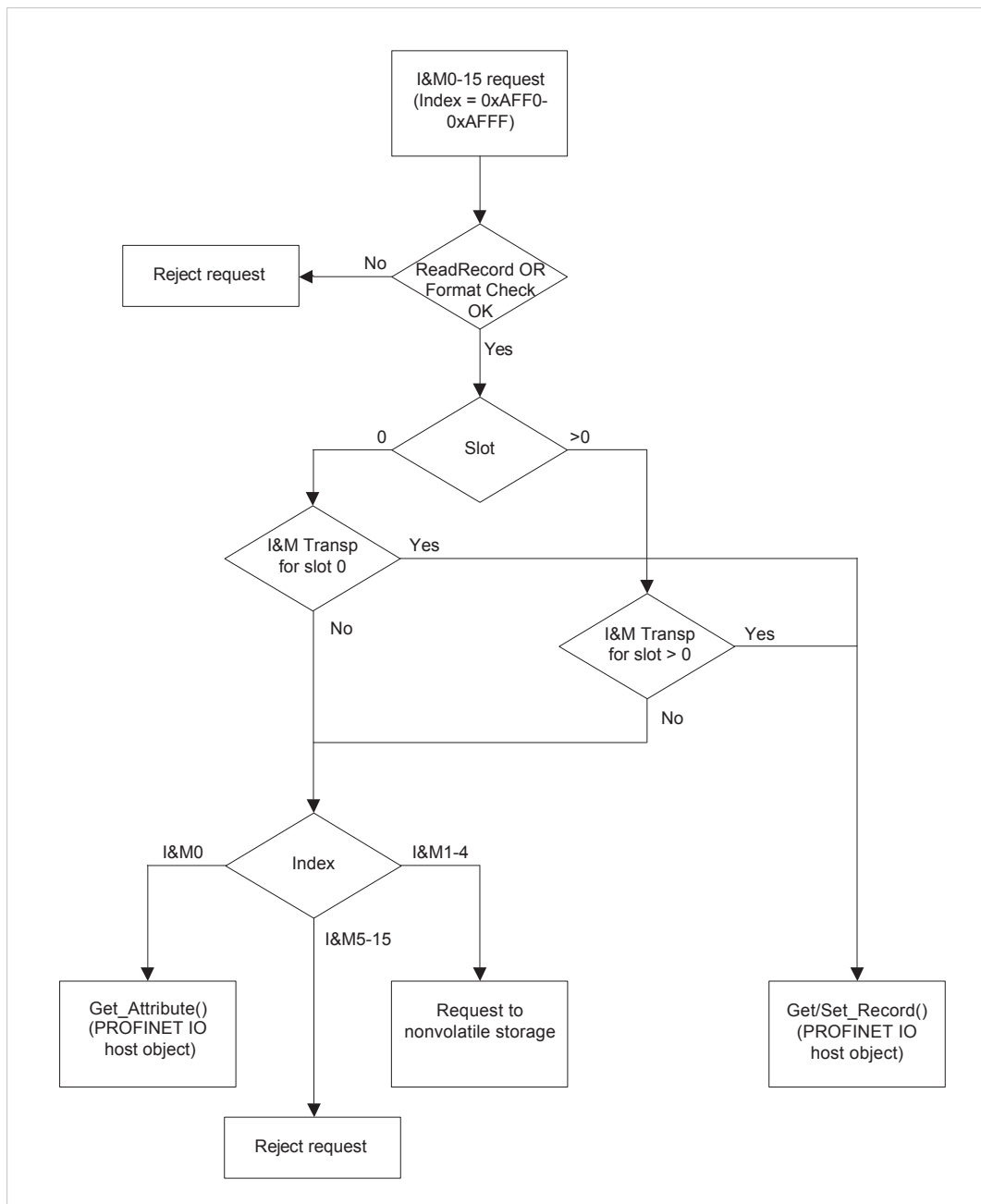


Fig. 14

- [PROFINET IO Object \(F6h\), p. 131](#)
- Details for command Get_Record in the [PROFINET IO Object \(F6h\), p. 131](#)
- Details for command Set_Record in the [PROFINET IO Object \(F6h\), p. 131](#)

C.3 Flowchart —Establishment of Real Identification (RI)

This flowchart illustrates the establishment of the Real Identification.

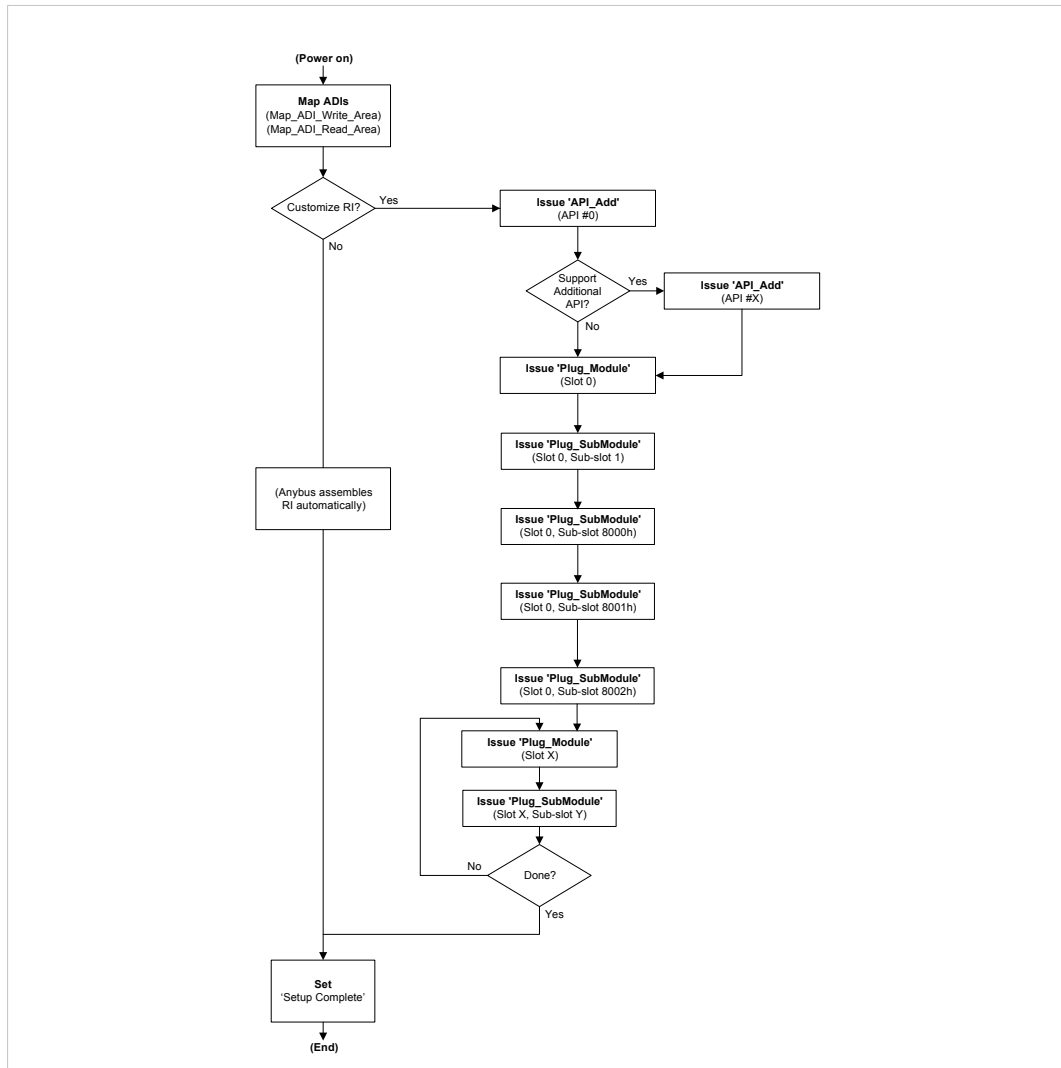


Fig. 15

- [Process Data, p. 15](#)
- [Real Identification \(RI\), p. 20](#)
- Details for command Set_Record in the [PROFINET IO Object \(F6h\), p. 131](#)

C.4 Flowcharts — Handling of Configuration Mismatch

C.4.1 Default Configuration Mismatch

This flowchart shows how the Anybus CompactCom automatically handles a configuration mismatch when the Real Identification has been established by the default configuration method.

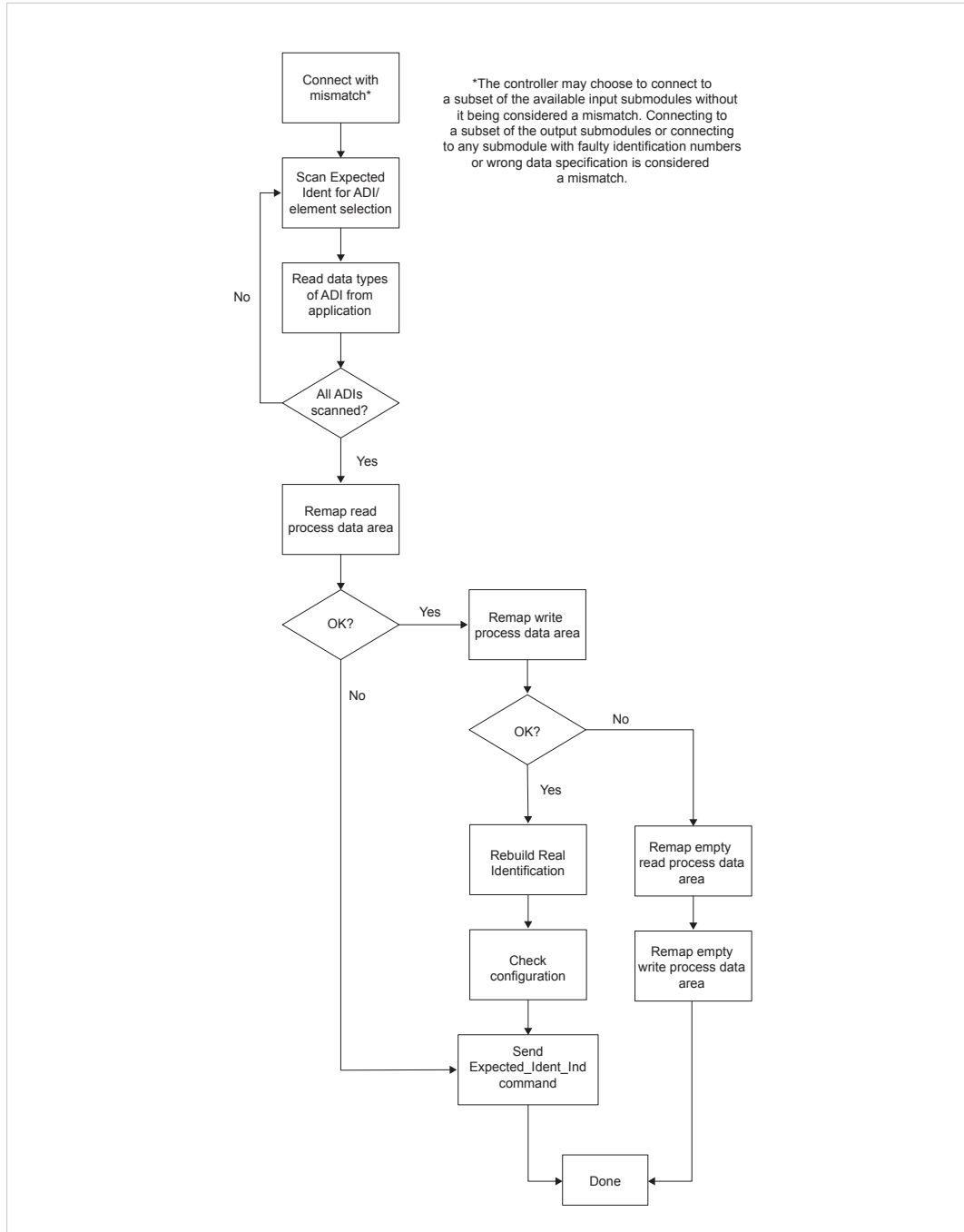


Fig. 16

C.4.2 Custom Configuration mismatch

This flowchart shows how to handle a configuration mismatch when the Real Identification has been established by the host application (custom configuration).

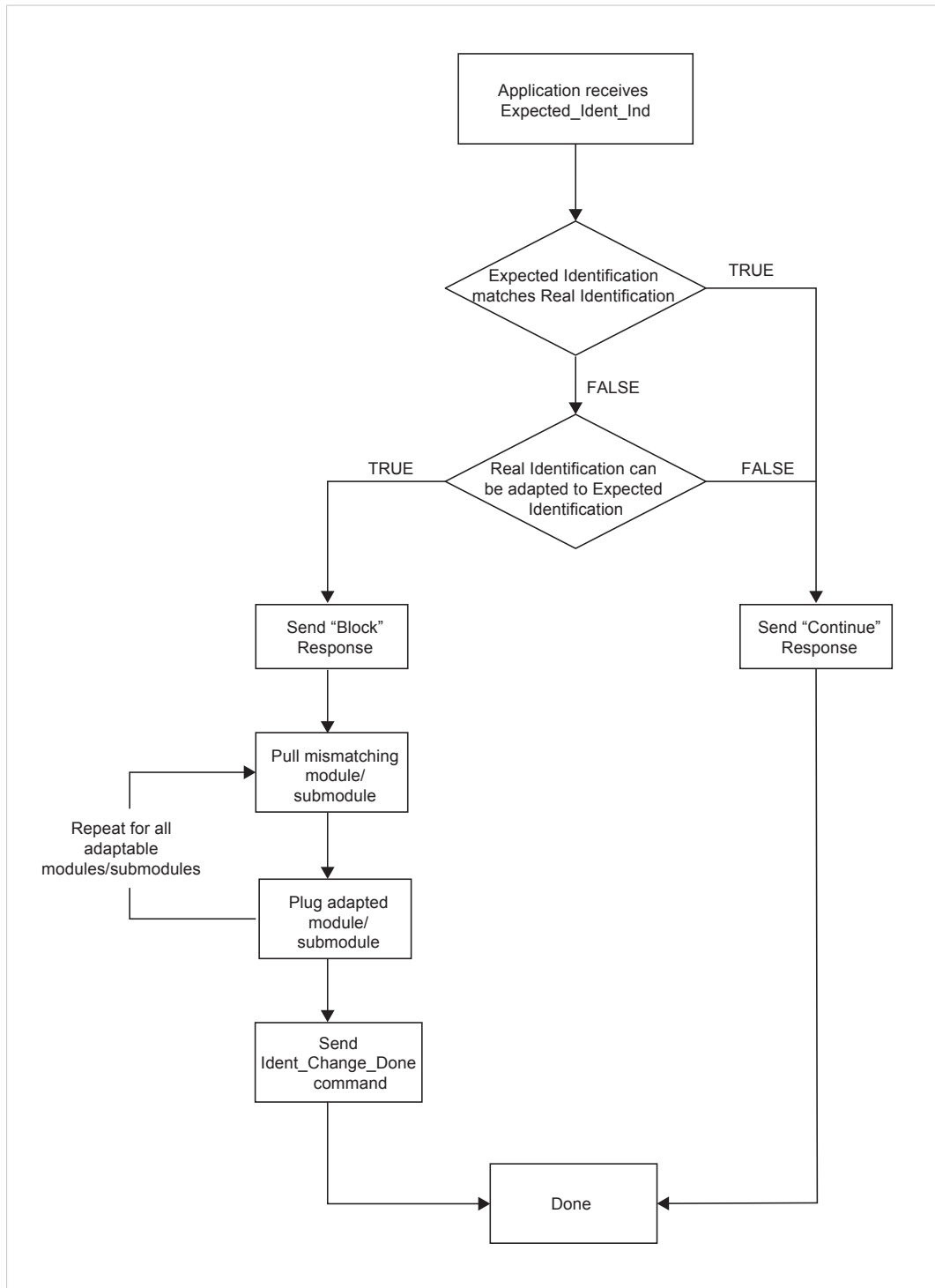


Fig. 17

D Secure HICP (Secure Host IP Configuration Protocol)

D.1 General

The Anybus CompactCom 40 PROFINET IRT supports the Secure HICP protocol used by the Anybus IPconfig utility for changing settings, e.g. IP address, Subnet mask, and enable/disable DHCP. Anybus IPconfig can be downloaded free of charge from the HMS website, www.anybus.com. This utility may be used to access the network settings of any Anybus product connected to the network via UDP port 3250.

The protocol offers secure authentication and the ability to restart/reboot the device(s).

D.2 Operation

When the application is started, the network is automatically scanned for Anybus products. The network can be rescanned at any time by clicking **Scan**.

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

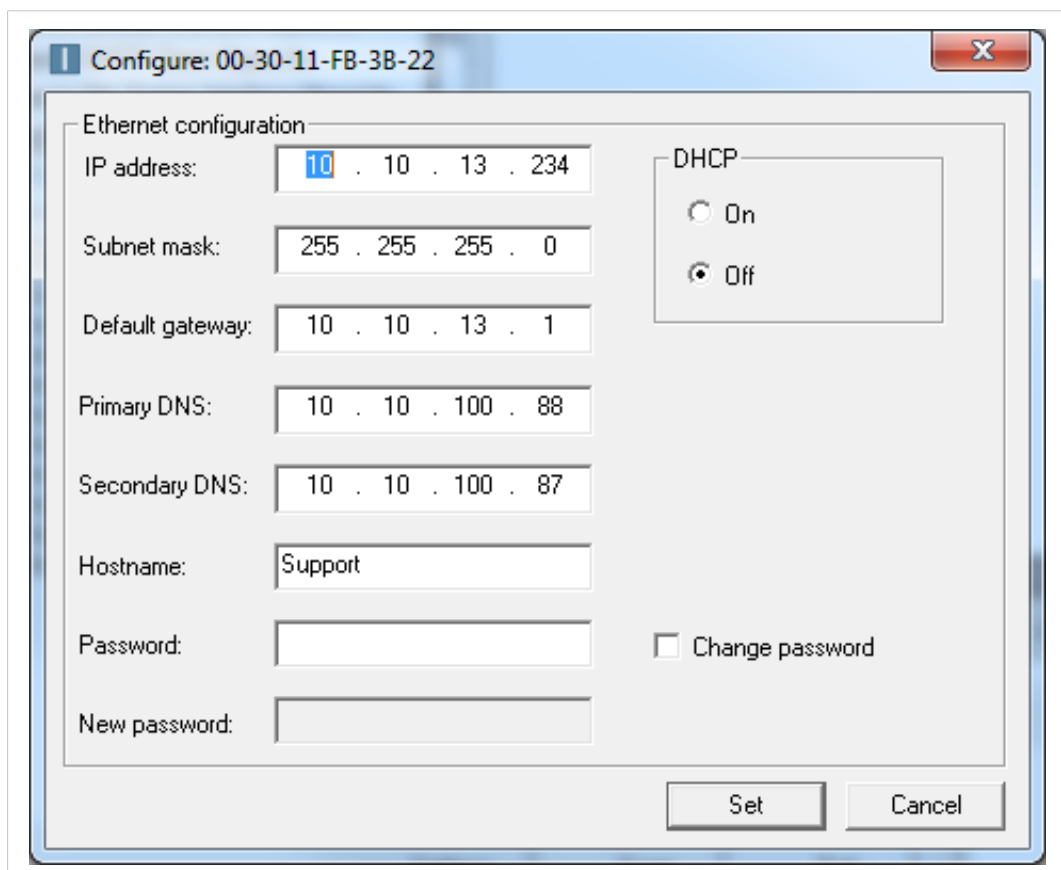


Fig. 18

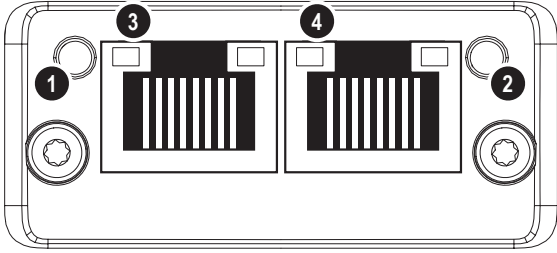
Validate the new settings by clicking **Set**, or click **Cancel** to cancel all changes. Optionally, the configuration can be protected from unauthorized access by a password. To enter a password, check the **Change password** checkbox and enter the password in the **New password** text field.

E Technical Specification

E.1 Front View

E.1.1 Front View (PROFINET IRT)

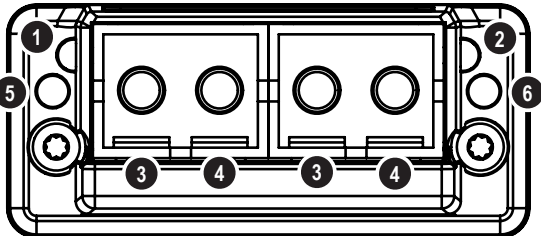
#	Item	Connector
1	Network Status LED	Ethernet, 45
2	Module Status LED	
3	Link/Activity LED (port 1)	
4	Link/Activity LED (port 2)	



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

E.1.2 Front View (PROFINET IRT Fiber Optic)

#	Item	Connector
1	Network Status LED	SconnectorC-RJ Fiber Optic C
2	Module Status LED	
3	Optical signal from the Anybus CompactCom module	
4	Optical signal to the Anybus CompactCom module	
5	Link/Activity LED (port 1)	
6	Link/Activity LED (port 2)	



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

E.1.3 Network Status LED

LED State	Description	Comments
Off	Offline	<ul style="list-style-type: none"> No power No connection with IO Controller
Green	Online (RUN)	<ul style="list-style-type: none"> Connection with IO Controller established IO Controller in RUN state
Green, 1 flash	Online (STOP)	<ul style="list-style-type: none"> Connection with IO Controller established IO Controller in STOP state or IO data bad RT synchronization not finished
Green, blinking	Blink	Used by engineering tools to identify the node on the network
Red	Fatal event	Major internal error (this indication is combined with a red module status LED)
Red, 1 flash	Station Name error	Station Name not set
Red, 2 flashes	IP address error	IP address not set
Red, 3 flashes	Configuration error	Expected Identification differs from Real Identification

E.1.4 Module Status LED

LED State	Description	Comments
Off	Not Initialized	No power OR Module in SETUP or NW_INIT state.
Green	Normal Operation	Module has shifted from the NW_INIT state.
Green, 1 flash	Diagnostic Event(s)	Diagnostic event(s) present
Red	Exception error	Device in state EXCEPTION.
	Fatal event	Major internal error (this indication is combined with a red network status LED)
Alternating Red/Green	Firmware update	Do NOT power off the module. Turning the module off during this phase could cause permanent damage.

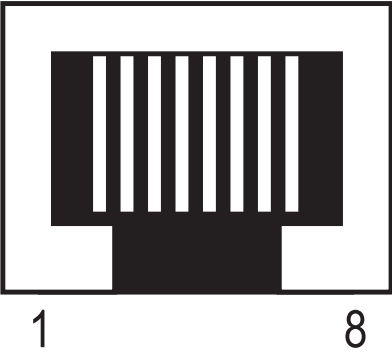
E.1.5 LINK/Activity LED

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

E.1.6 Ethernet Interface

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

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



For information on how to connect the PROFINET cable, see [Functional Earth \(FE\) Requirements, p. 164](#)

E.2 Functional Earth (FE) Requirements

In order to ensure proper EMC behavior, the module must be properly connected to functional earth via the FE pad/FE mechanism described in the *Anybus CompactCom 40 Hardware Design Guide*. Proper EMC behavior is not guaranteed unless these FE requirements are fulfilled.




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

For further information, see *PROFINET Installation Guideline for Cabling and Assembly*, available for download at www.profinet.com.

E.2.1 RJ45 Connector for PROFINET

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



E.3 Power Supply

E.3.1 Supply Voltage

The Anybus CompactCom 40 PROFINET IRT requires a regulated 3.3 V power source as specified in the general *Anybus CompactCom 40 Hardware Design Guide*.

E.3.2 Power Consumption

The Anybus CompactCom 40 PROFINET IRT is designed to fulfil the requirements of a Class B module. The current hardware design consumes up to 390 mA

The Anybus CompactCom 40 PROFINET IRT FO is designed to fulfil the requirements of a Class C module. The current hardware design consumes up to 740 mA

In line with HMS policy of continuous product development, we reserve the right to change the exact power requirements of this product without prior notification. However, in any case, the Anybus CompactCom 40 PROFINET IO will remain as a Class B module.

E.4 Environmental Specification

Consult the *Anybus CompactCom 40 Hardware Design Guide* for further information.

E.5 EMC Compliance

Consult the *Anybus CompactCom 40 Hardware Design Guide* for further information.

E.6 Fiber Optics Compliance (MAU type Compliance)

The optical interface of the Anybus CompactCom 40 PROFINET IRT is compliant with the non IEEE802.3 MAU type POF.

The supported cables are stated in the PI document *PROFINET Cabling and Interconnection Technology*;

for (SI-POF/PCF):

Physical Layer Medium Dependent Sublayer on 650 nm Fiber Optics

PROFIBUS & PROFINET International Order No: 2.432

F Conformance Test Guide

F.1 General

When using the default settings of all parameters, the Anybus CompactCom 40 PROFINET IRT is precertified for network compliance. This precertification is done to ensure that your product *can* be certified.

Changes in the parameters in the example GSD file, supplied by HMS Industrial Networks AB, will require a certification. A vendor ID can be obtained from PNO and is compulsory for certification. This chapter provides a guide for successful conformance testing your product, containing the Anybus CompactCom 40 PROFINET IRT, 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.

For multiple-API implementations, e.g. PROFIdrive, one of the submodules belonging to the “non-zero API” must be inserted in the I&M0 Filter Data by the application, using the command IP_Options.



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 AB at www.anybus.com/support for more information.

F.2 Reidentifying Your Product

After successful setting of the **Setup Complete** attribute in the Anybus Object (01h), the Anybus module asks for identification data from the host PROFINET IO Object (F6h). Therefore, the attributes listed below shall be implemented and proper values returned.

Object/Instance	Attribute	Explanation	Default	Customer sample	Comment
PROFINET IO Object (F6h), Instance 1	#1, Device ID	With this attribute you set the Device ID of the device	Device ID: 0010h	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.
PROFINET IO Object (F6h), Instance 1	#2, Vendor ID	With this attribute you set the Vendor ID of the device	Vendor ID: 010Ch (HMS)	Vendor ID: XXXXh	
PROFINET IO Object (F6h), Instance 1	#3, Station Type	With this attribute you set the station type of the device	“ABCC40-PIR”	“Cust-PNIO-Dev”	This information matches, in the case of Anybus CompactCom 40 PROFINET IRT, GSD keywords “DNS-CompatibleName” and “OrderNumber”. The Station Type must be equal to the “DNS-CompatibleName”, but it is allowed to have a completely different “OrderNumber”, see also I&M Order ID below.
PROFINET IO Object (F6h), Instance 1	#8, I&M Order ID	With this attribute you set the Order ID that is used in the I&M data	“ABCC40-PIR”	“Cust-PNIO-Dev”	This information must match the keys of “theOrderNumber” of the GSD-file.

Object/ Instance	Attribute	Explanation	Default	Customer sample	Comment
PROFINET IO Object (F6h), Instance 1	#10, I&M Hardware Revision	With this attribute you set the I&M Hardware Revision	(Hardware Rev.)	"0002h"	Optional. This information must match the keys of the "HardwareRelease" of the GSD-file.
PROFINET IO Object (F6h), Instance 1	#11, I&M Software Revision	With this attribute you set the I&M Software Revision	(Software Rev.)	"V2.5.3"	Optional. This information must match the keys of the "SoftwareRelease" of the GSD-file.
PROFINET IO Object (F6h), Instance 1	#13, I&M Profile ID	With this attribute you set the I&M Profile ID	0xF600 (Generic device)	API number of profile.	Optional. Only relevant for multi-API implementations like PROFIdrive.
PROFINET IO Object (F6h), Instance 1	#14, I&M Profile Specific type	With this attribute you set the I&M Profile Specific type	0x0004 (Communication Module)	Profile specific type, defined by profile.	Optional. Only relevant for multi-API implementations like PROFIdrive.
PROFINET IO Object (F6h), Instance 1	#19, System Description	With this attribute you set the description of the system	"HMS Industrial Networks Anybus CompactCom 40"	"Customer HMI Interface Module"	This information can be read by means of SNMP from the network side.

F.2.1 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 v2.3, 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".

F.3 Factory Default Reset

Reset command to Application Object (FFh) must be supported

When PROFINET IO modules are delivered, they are required to be in their "Factory Default" state. For PROFINET devices this means that their Station Name is empty ("") and that the IP-suite is not assigned (IP 0.0.0.0). When a Factory Default Reset command is received from the network, the Anybus module will erase all IP and Station Name information and inform the host application that a hardware reset of the Anybus module is required. This is done by sending a Reset command to the Application Object (FFh) of the host. For more details, please consult the *Anybus CompactCom Software Design Guide*.

F.4 IP Address

Normally the IP numbers of PROFINET IO devices are assigned via the PROFINET network via DCP (Discovery and Configuration Protocol). HMS recommends not using the Network Configuration Object (04h, instances #3 - #6) during the initialization phase for PROFINET modules, unless the end user has requested the IP address to be set to a specific value (by for example using a keypad). The reason is that when a factory default reset command is received from the PROFINET network (via DCP) the node must be available after a hardware reset with the default IP-address (0.0.0.0).

F.5 Station Name

Normally the Station Name of a PROFINET device is assigned by the end user via the PROFINET DCP protocol. HMS recommends not using the Station Name instance in the Network Configuration Object 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 reset with the default Station Name (“”).



The Anybus module will forward all information about the connection being established to the IO Controller, as commands to the host PROFINET IO Object (F6h). Even though the host application might not need this information, a response must always be generated (such as 05h, “Unsupported command”)

F.6 Documentation Considerations

To obtain a certificate, the following information must be present in the customer's user manual:

1. Behavior of the outputs if IOPS=BAD.
2. Behavior of the outputs if connection is aborted.
3. Behavior of the outputs at power on.

The Anybus CompactCom handles these situations in the following ways:

1. State change to IDLE. The network is informed that the I/O data of the submodule with IOPS=BAD is substituted with zeros (clear). No read process data is updated in the host interface.
2. State change to WAIT_PROCESS. The network is informed that the I/O data of all submodules is substituted with zeros (clear). No process data is updated in the host interface.
3. The network is informed that the I/O data of all submodules is substituted with zeros (clear). No process data is updated in the host interface.
4. A shift register application must use the PA signal to clear outputs when the Anybus CompactCom 40 PROFINET IRT is not in the state PROCESS_ACTIVE.

Also Information about the compliance of the optical interface with the non IEE 802.3 MAU-type POF should be provided in the documentation, see [Fiber Optics Compliance \(MAU type Compliance\)](#), p. 165.

F.7 Certification in Generic Anybus Mode

In Generic Anybus Mode (when the command API_add in the Network PROFINET IO Object (0Eh) is not used) there is normally nothing that needs to be considered apart from what is mentioned earlier in this appendix. The example HMS GSD file has to be modified with respect to the process mapping and identity of the product and this requires a certification of the product

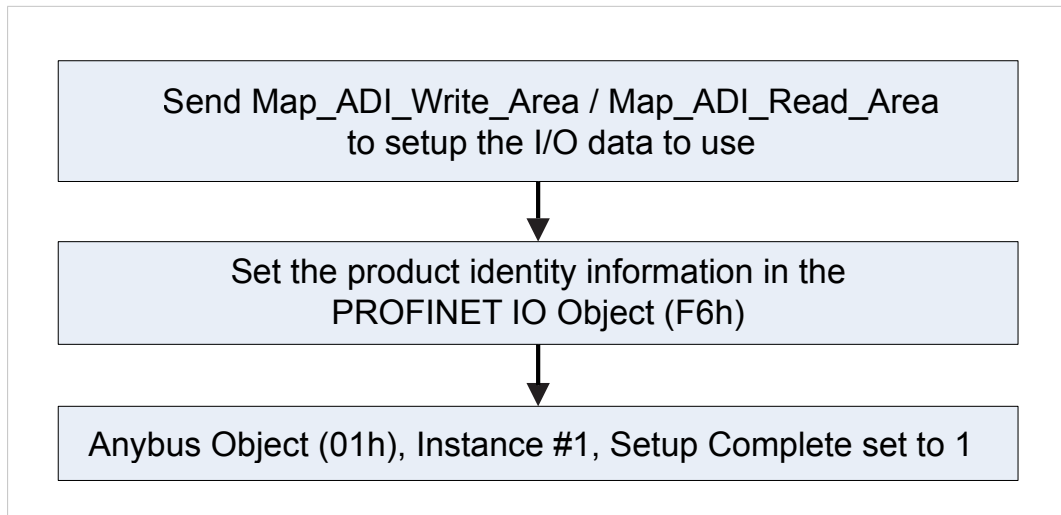


Fig. 19

F.8 Certification in Advanced Mode

In advanced mode (Network PROFINET IO Object (0Eh) is used), the most important thing is to use a Device Access Point (DAP) that conform to PROFINET IO Specification v2.0 or later (DAP2). From specification version 2.0 it is possible to describe the physical Ethernet interface and its ports (PDEV, or Physical Device) with a special mechanism. This is done with special submodules at slot 0 (the module at slot 0 is the access point for the device). HMS recommends following the flow below for setting up a DAP2.

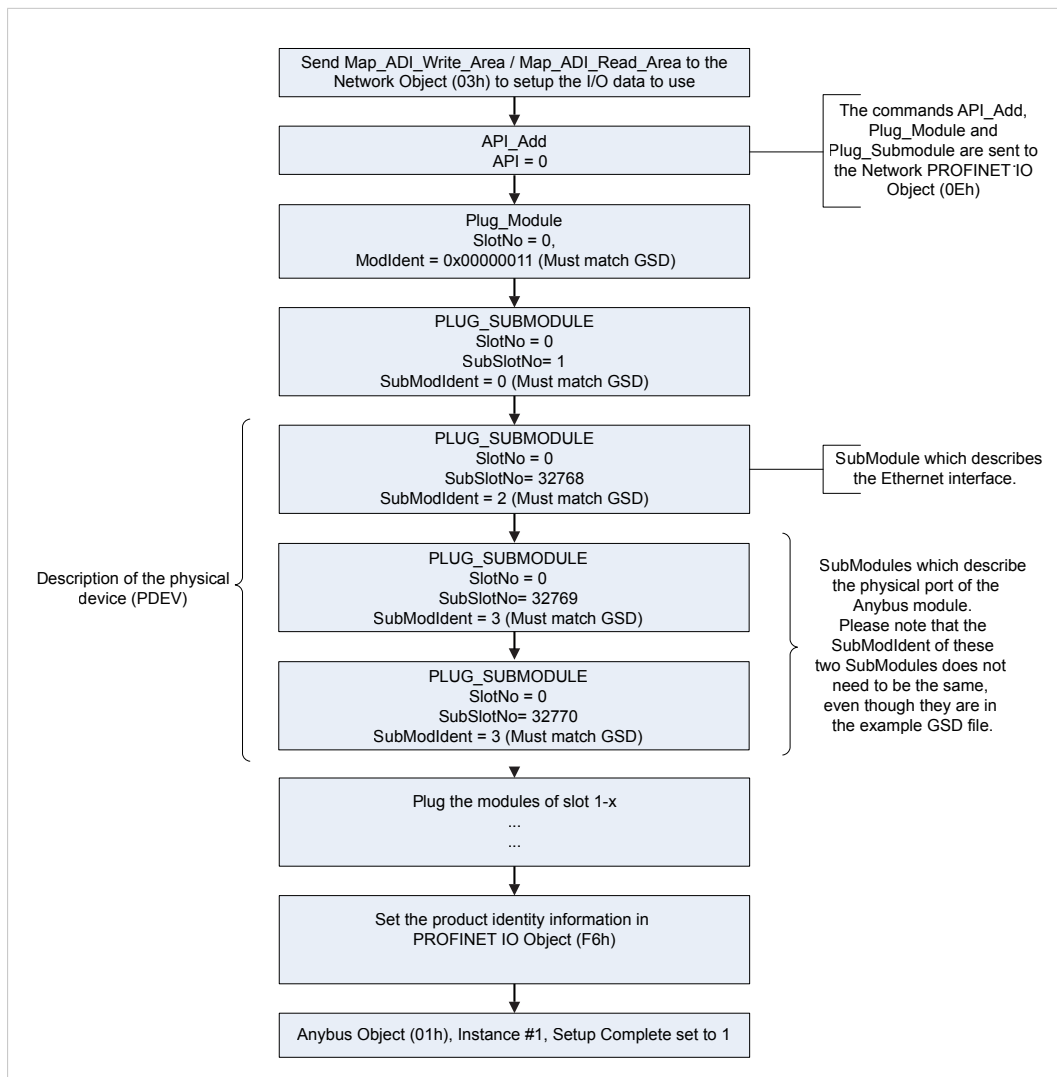


Fig. 20

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 example 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.

F.9 Changes in GSD File for Conformance Class B

The example GSD file, supplied by HMS Industrial Networks AB, is adapted for testing a Anybus CompactCom 40 PROFINET IRT for conformance class C. If the implementation does not need the isochronous features of the device, the GSD file can be modified to mirror this. The implementation can then be conformance tested for conformance class B instead. The list below describe the changes needed in the example GSD file to accomplish this.

1. The value of the `ConformanceClass` attribute in the `<CertificationInfo...>` element in each DAP should be changed from “C” to “B”:

```

<CertificationInfo ConformanceClass="B" ApplicationClass=""
NetloadClass="III"/>
  
```

2. The value of the `SupportedRT_Classes` attribute in the `<InterfaceSubmoduleItem ...>` element in each DAP should be "RT_CLASS_1". I.e. the "RT_CLASS_2" and "RT_CLASS_3" values should be removed:

```
<InterfaceSubmoduleItem ID="Interface" SubslotNumber="32768"
SubmoduleIdentNumber="0x00000002"
SupportedRT_Classes="RT_CLASS_1" TextId="T_ID_INTERFACE"
SupportedProtocols="SNMP;LLDP" SupportedMibs="MIB2" DCP_
HelloSupported="true"
PTP_BoundarySupported="true" DCP_BoundarySupported="true"
DelayMeasurementSupported="true">
```

3. The elements `<RT_Class3Properties ...>`, `<SynchronisationMode ...>`, and `<RT_Class3TimingProperties ...>` should be removed from each DAP

F.10 SYNC Pin Measurements for Conformance Class C Test

For a conformance class C (IRT) test, access to the SYNC pin must be provided to the test lab.

G Copyright Notices

This product includes software developed by Carnegie Mellon, the Massachusetts Institute of Technology, the University of California, and RSA Data Security:

Copyright 1986 by Carnegie Mellon.

Copyright 1983, 1984, 1985 by the Massachusetts Institute of Technology

Copyright (c) 1988 Stephen Deering.

Copyright (c) 1982, 1985, 1986, 1992, 1993

The Regents of the University of California. All rights reserved.

This code is derived from software contributed to Berkeley by Stephen Deering of Stanford University.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Copyright (C) 1990-2, RSA Data Security, Inc. All rights reserved.

License to copy and use this software is granted provided that it is identified as the "RSA Data Security, Inc. MD4 Message-Digest Algorithm" in all material mentioning or referencing this software or this function.

License is also granted to make and use derivative works provided that such works are identified as "derived from the RSA Data Security, Inc. MD4 Message-Digest Algorithm" in all material mentioning or referencing the derived work.

RSA Data Security, Inc. makes no representations concerning either the merchantability of this software or the suitability of this software for any particular purpose. It is provided "as is" without express or implied warranty of any kind.

These notices must be retained in any copies of any part of this documentation and/or software.

Copyright (C) 1991-2, RSA Data Security, Inc. Created 1991. All rights reserved.

License to copy and use this software is granted provided that it is identified as the "RSA Data Security, Inc. MD5 Message-Digest Algorithm" in all material mentioning or referencing this software or this function.

License is also granted to make and use derivative works provided that such works are identified as "derived from the RSA Data Security, Inc. MD5 Message-Digest Algorithm" in all material mentioning or referencing the derived work.

RSA Data Security, Inc. makes no representations concerning either the merchantability of this software or the suitability of this software for any particular purpose. It is provided "as is" without express or implied warranty of any kind.

These notices must be retained in any copies of any part of this documentation and/or software.

Copyright 2013 jQuery Foundation and other contributors

<http://jquery.com/>

Permission is hereby granted, free of charge, to any person obtaining

a copy of this software and associated documentation files (the

"Software"), to deal in the Software without restriction, including

without limitation the rights to use, copy, modify, merge, publish,

distribute, sublicense, and/or sell copies of the Software, and to

permit persons to whom the Software is furnished to do so, subject to

the following conditions:

The above copyright notice and this permission notice shall be

included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,

EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF

MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND

NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE

LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION

OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION

WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

rsvp.js

Copyright (c) 2013 Yehuda Katz, Tom Dale, and contributors

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR

IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE

AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER

LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,

OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE

SOFTWARE.

libb (big.js)

The MIT Expat Licence.

Copyright (c) 2012 Michael Mclaughlin

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the 'Software'), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,

EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

FatFs - FAT file system module R0.09b (C)ChaN, 2013

FatFs module is a generic FAT file system module for small embedded systems.

This is a free software that opened for education, research and commercial developments under license policy of following terms.

Copyright (C) 2013, ChaN, all right reserved.

The FatFs module is a free software and there is NO WARRANTY.

No restriction on use. You can use, modify and redistribute it for personal, non-profit or commercial products UNDER YOUR RESPONSIBILITY.

Redistributions of source code must retain the above copyright notice.

Copyright (c) 2002 Florian Schulze.

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the authors nor the names of the contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHORS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE

ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHORS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

ftpd.c - This file is part of the FTP daemon for lwIP

Format - lightweight string formatting library.

Copyright (C) 2010-2013, Neil Johnson

All rights reserved.

Redistribution and use in source and binary forms,

with or without modification,

are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice,

this list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice,

this list of conditions and the following disclaimer in the

documentation and/or other materials provided with the distribution.

* Neither the name of nor the names of its contributors

may be used to endorse or promote products derived from this software

without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS

"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT

LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR

A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER

OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,

EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,

PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR

PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF

LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Print formatting routines

Copyright (C) 2002 Michael Ringgaard. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the project nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND

ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE

ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE

FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS

OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)

HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT

LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

lwIP is licenced under the BSD licence:

Copyright (c) 2001-2004 Swedish Institute of Computer Science.

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The name of the author may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS AND ANY EXPRESS OR IMPLIED

WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT

SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT

OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS

INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING

IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY

OF SUCH DAMAGE.

MD5 routines

Copyright (C) 1999, 2000, 2002 Aladdin Enterprises. All rights reserved.

This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it

freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.

L. Peter Deutsch

ghost@aladdin.com

