

Anybus[®] CompactCom[™] 40

Ethernet POWERLINK

HMSI-27-219 3.2 en-US ENGLISH

Important User Information

Liability

Every care has been taken in the preparation of this document. Please inform HMS Industrial Networks of any inaccuracies or omissions. The data and illustrations found in this document are not binding. We, HMS Industrial Networks, 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. HMS Industrial Networks 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 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 cannot assume responsibility for actual use based on these examples and illustrations.

Intellectual Property Rights

HMS Industrial Networks 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.

Table of Contents

Page

1	Preface	5
1.1	About this document	5
1.2	Related Documents	5
1.3	Document History	5
1.4	Document Conventions	5
1.5	Document Specific Conventions	6
1.6	Trademark Information	6
2	About the Anybus CompactCom 40 Ethernet POWERLINK	7
2.1	General	7
2.2	Features	8
3	Basic Operation	9
3.1	General Information	9
3.2	Device Customization	10
3.3	Communication Settings	11
3.4	Web Interface	14
3.5	Socket Interface (Advanced Users Only)	14
3.6	E-mail Client	14
3.7	Diagnostics	14
3.8	Synchronization	14
3.9	Network Data Exchange	15
3.10	File System	16
3.11	Network Reset Handling	18
4	Object Dictionary	19
4.1	Standard Objects	19
4.2	Manufacturer Specific Objects	26
5	FTP Server	30
5.1	General Information	30
5.2	User Accounts	30
5.3	Session Example	31
6	Web Server	32
6.1	General Information	32
6.2	Default Web Pages	32
6.3	Server Configuration	35

7	E-mail Client.....	38
7.1	General Information	38
7.2	How to Send E-mail Messages	38
8	JSON	39
8.1	General Information	39
8.2	JSON Objects.....	40
8.3	Example	58
9	Server Side Include (SSI)	59
9.1	General Information	59
9.2	Include File	59
9.3	Command Functions.....	59
9.4	Argument Functions	74
9.5	SSI Output Configuration	78
10	Anybus Module Objects.....	79
10.1	General Information	79
10.2	Anybus Object (01h)	80
10.3	Diagnostic Object (02h)	81
10.4	Network Object (03h)	82
10.5	Network Configuration Object (04h)	84
10.6	Socket Interface Object (07h)	90
10.7	SMTP Client Object (09h).....	107
10.8	File System Interface Object (0Ah)	112
10.9	Network Ethernet Object (0Ch)	113
11	Host Application Objects	115
11.1	General Information	115
11.2	POWERLINK Object (E9h).....	116
11.3	Application File System Interface Object (EAh)	118
11.4	SYNC Object (EEh)	119
11.5	Ethernet Host Object (F9h)	120
A	Categorization of Functionality	125
A.1	Basic.....	125
A.2	Extended	125
B	Implementation Details	126
B.1	SUP-Bit Definition	126
B.2	Anybus State Machine	127

C	Timing & Performance	128
C.1	General Information	128
C.2	Event Based Process Data Delay	128
D	Secure HICP (Secure Host IP Configuration Protocol)	129
D.1	General	129
D.2	Operation	129
E	Technical Specification.....	130
E.1	Front View	130
E.2	Functional Earth (FE) Requirements.....	131
E.3	Power Supply	131
E.4	Environmental Specification.....	131
E.5	EMC Compliance.....	131
F	Copyright Notices	132

This page intentionally left blank

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 Ethernet POWERLINK. The document describes the features that are specific to Anybus CompactCom 40 Ethernet POWERLINK. 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 Ethernet POWERLINK specific functionality is to be used, in-depth knowledge of Ethernet POWERLINK networking internals and/or information from the official Ethernet POWERLINK specifications may be required. In such cases, the persons responsible for the implementation of this product should either obtain the Ethernet POWERLINK specification to gain sufficient knowledge or limit their implementation in such a way that this is not necessary.

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

1.2 Related Documents

Document	Author	Document ID
Anybus CompactCom 40 Software Design Guide	HMS	HMSI-216–125
Anybus CompactCom M40 Hardware Design Guide	HMS	HMSI-216–126
Anybus CompactCom B40 Design Guide	HMS	HMSI-27-230
Anybus CompactCom Host Application Implementation Guide	HMS	HMSI-27-334

1.3 Document History

Version	Date	Description
1.00	2014-03-25	First official revision
1.50	2014-07-11	Major update
1.60	2015-05-28	Misc updates
1.70	2015-10-16	Minor Updates
2.0	2017	From FM to DOX
2.1	2017-07-10	Minor corrections
3.0	2018-02-16	Added IT functionality Minor corrections
3.1	2018-05-28	Minor update
3.2	2019-02-28	Rebranding Minor update

1.4 Document Conventions

Ordered lists are used for instructions that must be carried out in sequence:

1. First do this
2. Then do this

Unordered (bulleted) lists are used for:

- Itemized information

- Instructions that can be carried out in any order

...and for action-result type instructions:

- ▶ This action...
 - leads to this result

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: [Document Conventions, p. 5](#)

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 Document Specific Conventions

- 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.

1.6 Trademark Information

Anybus® is a registered trademark of HMS Industrial Networks.

All other trademarks are the property of their respective holders.

2 About the Anybus CompactCom 40 Ethernet POWERLINK

2.1 General

The Anybus CompactCom 40 Ethernet POWERLINK communication module provides instant Ethernet POWERLINK conformance tested 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 Anybus CompactCom 40 modules defined in the Anybus CompactCom 40 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 be able to take full advantage of advanced network specific functionality, a certain degree of dedicated software support may be necessary.

2.2 Features

- Two Ethernet POWERLINK ports (RJ45)
- Supports Ethernet POWERLINK V2.0 Communication Profile Specification version 1.3.0 (Controlled Node, CN)
- Integrated hub
- 100 Mbit/s, half duplex operation
- Up to 57343 ADIs
- Max. read process data: 1490 bytes
- Max. write process data: 1490 bytes
- Max. process data (read + write, in bytes): 2980 bytes
- Web server w. customizable content
- FTP server
- Email client
- Server Side Include (SSI) functionality
- JSON functionality
- Customizable Identity Information
- Transparent Socket Interface
- 200 μ s cycle time
- Supports ring redundancy
- Customizable identity information
- Supports 1 TPDO and 1 RPDO (each can hold 1490 bytes)
- Adaptable XDD file included
- Supports segmented SDO transfer
- PollResponse Chaining
- Multiplexing
- Support for SYNC functionality

3 Basic Operation

3.1 General Information

Full IT functionality is enabled in the Anybus CompactCom 40 Ethernet POWERLINK at startup. The IT functionality can be disabled in the [POWERLINK Object \(E9h\)](#), p. 116 (attribute #17). The application will then only communicate according the Ethernet POWERLINK protocol.



Full IT functionality is supported from Anybus CompactCom 40 Ethernet POWERLINK firmware version 1.12.

Full IT functionality cannot be reenabled during runtime. The application must be restarted.

3.1.1 IT Functionality States in POWERLINK

In the POWERLINK controlled node state machine there is one state and one super state where IT communication is possible, NMT_CS_BASIC_ETHERNET and NMT_CS_EPL_MODE.

In NMT_CS_BASIC_ETHERNET the standard Ethernet CSMA/CD communication is used and there are no limitations on what node that can send at what time. It is worth to consider that POWERLINK uses hubs instead of switches so if several nodes are communicating in the NMT_CS_BASIC_ETHERNET state there are risks of collisions on the network.

In the NMT_CS_EPL_MODE super state the network cycle is strictly controlled by the managing node. IT frames can only be sent in the asynchronous phase of the POWERLINK cycle, and it is the managing node that controls which node that is allowed to transmit. Due to this a normal PC cannot be connected to the POWERLINK network to do IT communication with a controlled node in NMT_CS_EPL_MODE. It is necessary to have some kind of gateway to get IT frames to and from the POWERLINK network. Normally only one Ethernet frame can be sent in a single asynchronous phase, and this includes all POWERLINK ASnd frames as well, e.g. SDO frames.

This means that the performance of the IT traffic is worse in NMT_CS_EPL_MODE compared to NMT_CS_BASIC_ETHERNET, and the performance is also negatively affected by other protocols using the asynchronous phase, e.g. SDO.



It is not possible to return to the state NMT_CS_BASIC_ETHERNET during runtime. If the Anybus CompactCom 40 Ethernet POWERLINK has entered the superstate NMT_CS_EPL_MODE, it will have to be restarted to be able to enter the state NMT_CS_BASIC_ETHERNET.

3.1.2 POWERLINK and CANopen Implementation

Ethernet POWERLINK (EPL) is a deterministic real-time protocol for standard Ethernet. It is an open protocol managed by the Ethernet POWERLINK Standardization Group (EPSG).

Ethernet POWERLINK extends Ethernet according to the IEEE 802.3 standard with mechanisms to transfer data with predictable delivery. The communication meets timing demands typical for high performance automation and motion applications.

The Ethernet POWERLINK communication profile is based on CANopen communication profiles DS301 and DS302. Based on this communication profile, the multitude of CANopen device profiles can be used in a POWERLINK environment without changes.

Ethernet POWERLINK manages the network traffic using dedicated time-slots for isochronous and asynchronous data. Only one networked device at the time gains access to the network media. Thus transmission of data will never interfere and precise communication timing is guaranteed. The mechanism is called Slot Communication Network Management (SCNM). SCNM is managed by one particular networked device - the Managing Node (MN). All other nodes are

called Controlled Nodes (CN). The module can participate as a Controlled Node in Ethernet POWERLINK networks.

3.1.3 Software Requirements

No additional network support code needs to be written in order to support the Anybus CompactCom 40 Ethernet POWERLINK, however due to the nature of the Ethernet POWERLINK networking system, certain restrictions must be taken into account:

- One diagnostic instance can be created by the host application in event of a major fault. The limit is set by the module, not by the network. The event will not show on the POWERLINK network.
- Only ADIs with instance numbers less than or equal to 57343 can be accessed from the network.

For in depth information regarding the Anybus CompactCom software interface, consult the Anybus CompactCom 40 Software Design Guide.

See also...

- [Diagnostic Object \(02h\), p. 81](#) (Anybus Module Objects)
- Anybus CompactCom 40 Software Design Guide, “Application Data Object (FEh)”

3.2 Device Customization

3.2.1 Network Identity

By default, the module uses the following identity settings:

Vendor ID	0000001Bh (HMS Industrial Networks)
Device Type	00000000h (Generic Device)
Product Code	00000028h (Anybus CompactCom 40 Ethernet POWERLINK)
Manufacturer Device Name:	“Anybus CompactCom 40 Ethernet POWERLINK”
Manufacturer Name:	“HMS Industrial Networks”

Optionally, it is possible to customize the identity of the module by implementing the corresponding instance attributes in the POWERLINK Object (E9h).

See also...

- [POWERLINK Object \(E9h\), p. 116](#) (Host Application Objects)



If the identity settings are changed, recertification of the module is needed. For the end product to pass the EPSG conformance tests and be certified, a separate Vendor ID has to be requested from EPSG (free of charge).

3.2.2 XML Device Description (XDD)

On Ethernet POWERLINK, the characteristics of a device is stored in an XML file with the suffix XDD. This file is used by configuration tools etc. when setting up the network configuration. HMS supplies a standard (generic) XDD file, which corresponds to the default settings in the module. However, all implementations will add changes to the default settings, making it necessary to create a custom XDD file where the changes are reflected. This invalidates the default identity information and requires recertification of the product.



HMS approves use of the standard XDD file only under the condition that it matches the actual implementation and that the identity information remains unchanged.

There is support for dynamic generation of the XDD file, including information about the configuration and host application ADIs, among other things. The XDD file can either be read via the network (object 1021h in the Object Dictionary) or via the Anybus File System Object (0Ah).

To be able to automatically generate an XDD file, the host application must support the command `Get_Instance_Number_By_Order` in the Application Data Object. If this command is not supported, an error code will be returned when trying to read the XDD file.

Also, it is not possible to, in the XDD file, specify the access of an ADI. The Descriptor attribute of the ADI instances must specify access as either Get or Set, otherwise an error code will be returned when trying to read the XDD file.

See also...

- [Object Entries, p. 19](#)
- [File System Interface Object \(0Ah\), p. 112](#)
- Anybus CompactCom 40 Software Design Guide, “Application Data Object (FEh)”

3.3 Communication Settings

As with other Anybus CompactCom products, network related communication settings are grouped in the Network Configuration Object (04h).

In this case, this includes:

Ethernet Interface Settings	100 Mbit, half duplex.
IP Configuration	These settings must be set properly in order for the module to be able to participate on the network, see below for more information.
Node ID Setting	The Anybus CompactCom module participates as a Controlled Node in the network, with a Node ID in the range 1 - 239.

See also...

- [Web Server, p. 32](#)
- [Network Configuration Object \(04h\), p. 84](#)
- [Secure HICP \(Secure Host IP Configuration Protocol\), p. 129](#)

3.3.1 IP Configuration

The IP configuration for the Anybus CompactCom 40 Ethernet POWERLINK is handled in different ways depending on the network state.

- In NMT_CS_BASIC_ETHERNET the network configuration set in the network configuration object is used.

The module supports DHCP, which may be used to retrieve the IP settings from a DHCP-server automatically. DHCP is enabled by default, but can be disabled if necessary.

- When entering the NMT_CS_EPL_MODE the actual network configuration is changed to a fixed configuration according to the following:
 - IP address is fixed to 192.168.100.yyy where yyy is the POWERLINK node ID.
 - Subnet mask is fixed to 255.255.255.0
 - Default gateway is changed to 192.168.100.254. This can be changed via e.g. SDO or web.
 - Hostname is set to yy-zzzzzzzz where yy is the node ID in hexadecimal form and zzzzzzzz is the vendor ID in hexadecimal form. This can be changed via e.g. SDO or web.
 - DHCP is disabled.

Note that the stored IP configuration in the NC object isn't changed, it is only the actual configuration that changes. The next time the module starts in NMT_CS_BASIC_ETHERNET the stored IP configuration will be used again.



If the IT functionality is turned off, the Anybus CompactCom 40 Ethernet POWERLINK has no IP address.

3.3.2 Communication Settings in Stand Alone Shift Register Mode

If the Anybus CompactCom 40 is used stand alone, there is no application from which to set the IP address. The IP address is instead set using the DIP1 switches (IP address byte 3) and the virtual attributes (Ethernet Host object (F9h), attribute #17), that are written to memory during setup (IP address byte 0 - 2). A flowchart is shown below.

Please note that the flowchart is used when the Anybus CompactCom 40 Ethernet POWERLINK is in the NMT_CS_BASIC ETHERNET state. If it is in NMT_CS_EPL_MODE, the settings are fixed, according to [IP Configuration, p. 12](#)

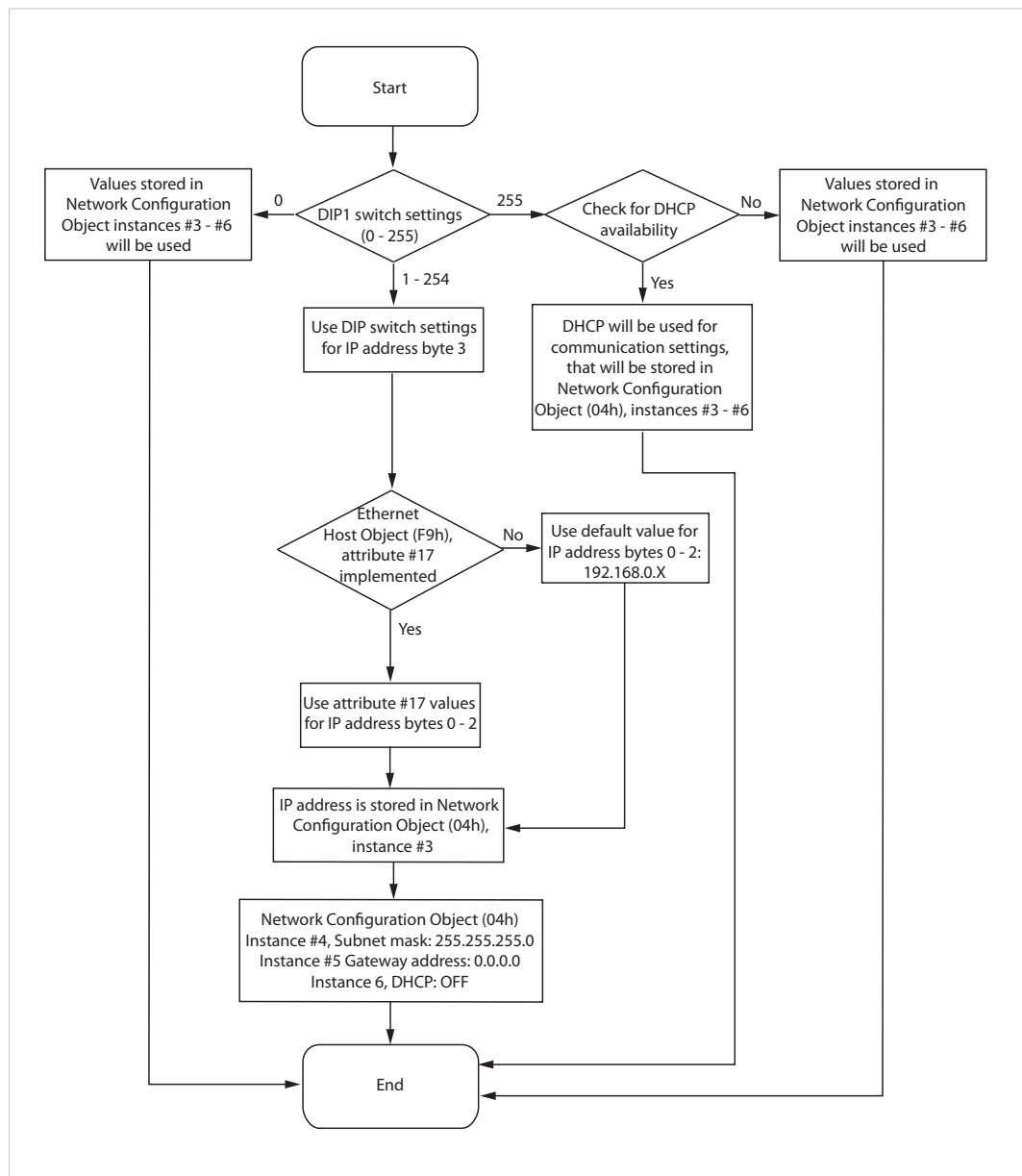


Fig. 1

See also ...

- [Ethernet Host Object \(F9h\), p. 120](#)
- Anybus CompactCom M40 Hardware Design Guide
- [Network Configuration Object \(04h\), p. 84](#)

3.4 Web Interface

The web interface can be fully customized to suit a particular application. Dynamic content can be created by means of JSON and SSI scripting. Data and web pages are stored in a FLASH-based file system, which can be accessed using any standard FTP client or the File System Interface Object.

See also...

- [File System, p. 16](#)
- [FTP Server, p. 30](#)
- [Web Server, p. 32](#)
- [Server Side Include \(SSI\), p. 59](#)
- [JSON, p. 39](#)

3.5 Socket Interface (Advanced Users Only)

The built in socket interface allows additional protocols to be implemented on top of TCP/IP. Data is structured by the application and is then embedded within the Ethernet frames. The host application can open network connections of its own to other nodes on the network, e.g. if you want to connect to another server or use a web server of your own.

See also..

- ref to socket interface object needed(Anybus Module Object)
- ref to message segmentation needed

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. 16](#)

3.7 Diagnostics

A major unrecoverable event will cause the module to enter the EXCEPTION state. This will be recorded in the Diagnostic Object, but not reported to the Ethernet POWERLINK network. The module will cease communication on the network without notice.

See [Diagnostic Object \(02h\), p. 81](#) (Anybus Module Objects)

3.8 Synchronization

The Anybus CompactCom 40 Ethernet POWERLINK module supports synchronization. To utilize SYNC in the application, implement the SYNC object. See [SYNC Object \(EEh\), p. 119](#).

If synchronous operation is supported it will be enabled for all POWERLINK NMT states with a fixed cycle length and cyclic data exchange, i.e. the POWERLINK NMT states Pre Operational 2, Ready to Operate and Operational.

If SYNC is supported, the SYNC cycle time will be written to the SYNC object as soon as the value is received from the network. If SYNC is not supported, it will be written after the NMTResetConfiguration command.

The SYNC signal must be synced in the Anybus state WAIT_PROCESS (where the SYNC signal may differ much between the configured and the actual value).

When SYNC is established, the SYNC signal is produced upon the expected reception of SoC frames, in all NMT states that transmit SoC frames.

3.9 Network Data Exchange

3.9.1 Application Data (ADIs)

Application Data Instances (ADIs) can be accessed from the network via dedicated object entries in the Manufacturer Specific Range (2001h - FFFFh), see [Manufacturer Specific Objects, p. 26](#).



The XDD file must match the actual ADI implementation in the host application.

To be able to automatically generate an XDD file, the host application must support the command Get_Instance_Number_By_Order in the Application Data Object. If this command is not supported, an error code will be returned when trying to read the XDD file.

The generated XDD file contains information about the ADIs present in the host application and specifies e.g. name, type, access rights and the PDO mapping capabilities of each ADI. If attribute #10 (Element Name) in the Application Data Object (FEh) is not implemented, it is not possible to retrieve the names of individual subindex fields in the ADIs. The subindex fields will have a default generic name, with the subindex number attached to the end. If attribute #10 (Element Name) in the Application Data Object (FEh) is implemented, each sub element will have a specific name. See Application Data Object (FEh) in Anybus CompactCom 40 Software Design Guide for more information.

3.9.2 Process Data

On POWERLINK, ADIs mapped as Process Data can be exchanged on the bus, following the predefined time scheme with slots for sent and received data. Process data is mapped as PDOs, and the module supports 1 TPDO and 1 RPDO.

Process data can be either statically or dynamically mapped. Dynamic mapping is configured by the Managing Node of the network. To support dynamic mapping, the application has to implement the Remap_ADI_Write_Area and Remap_ADI_Read_Area commands in the Application Data Object. See Application Data Object (FEh) in Anybus CompactCom 40 Software Design Guide for more information. If these commands are not implemented, the static mapping, provided by the application using the Map_Adi command of the Network Object, will be used.

There are several object entries in the Object Dictionary that have different access rights depending on whether dynamic mapping or static mapping is used. If dynamic mapping is supported, the object entries can be changed and are marked as "rw" (read/write). If the application uses static mapping, they cannot be changed and are marked as "ro" (read only).

The values in object entries 1400h, 1600h, 1800h and 1A00h cannot be changed in POWERLINK states corresponding to IDLE or PROCESS_ACTIVE (See [Anybus State Machine, p. 127](#)). If the network tries to change any of these entries when the device is in one of those states the command will be aborted by the CompactCom and an SDO abort code will be returned to the network.

See object entries 1400h, 1600h, 1800h and 1A00h in [Object Dictionary, p. 19](#).

3.10 File System

3.10.1 Overview

The Anybus CompactCom 40 Ethernet POWERLINK has a built-in file system, that can be accessed from the application and from the network. Three directories are predefined:

VFS	The virtual file system that e.g. holds the web pages of the module.
Application	This directory provides access to the application file system through the Application File System Interface Object (EAH) (optional).
Firmware	The firmware directory points to the firmware candidate area where firmware files can be uploaded.



In the firmware folder, it is not possible to use append mode when writing a file. Be sure to use write mode only.

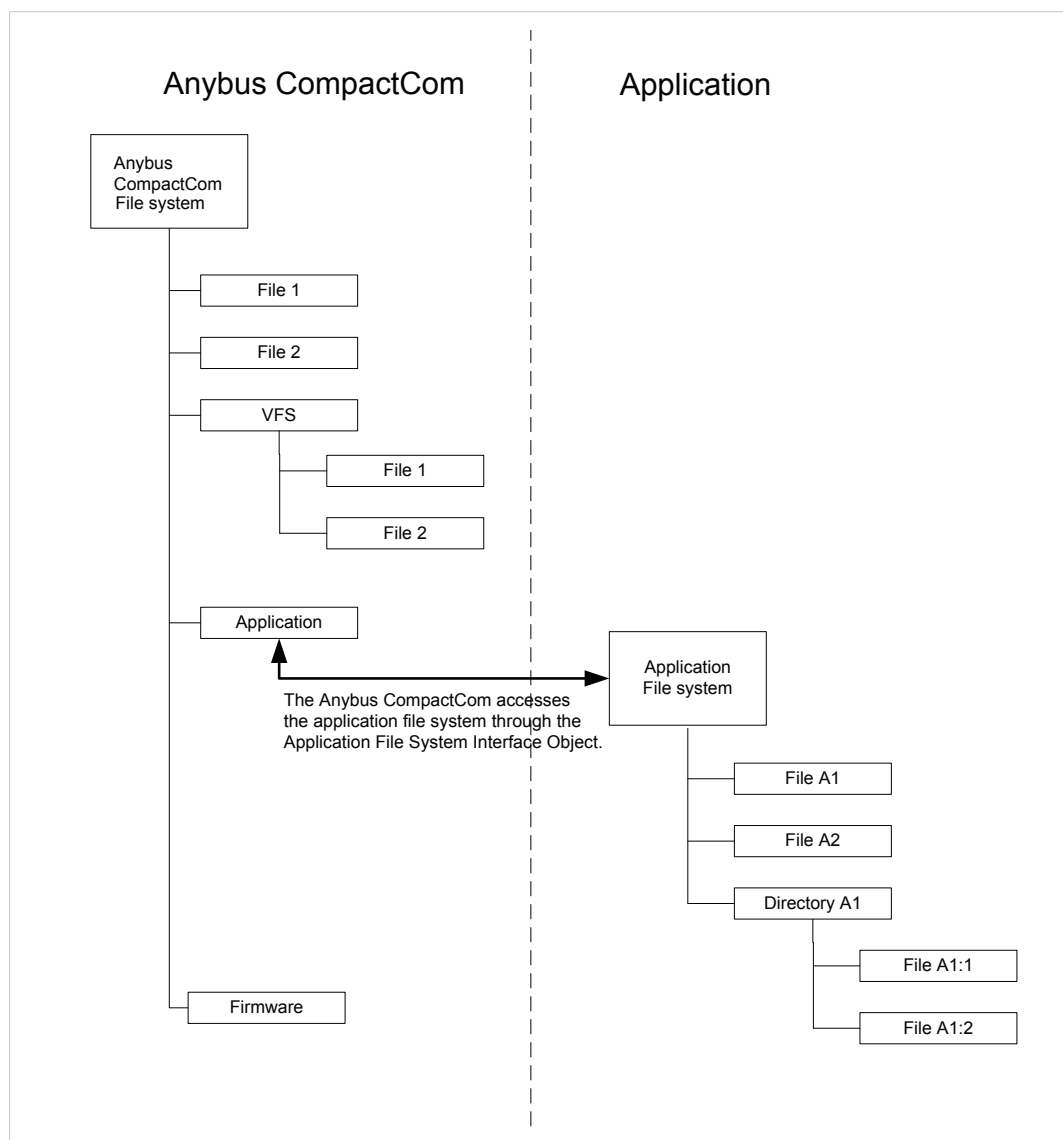


Fig. 2

3.10.2 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. 30](#)
- [Web Server, p. 32](#)
- [E-mail Client, p. 38](#)
- [Server Side Include \(SSI\), p. 59](#)



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

3.10.3 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
```

3.11 Network Reset Handling

3.11.1 Software Reset

If a software reset is requested from the network, the module will issue a Reset command to the Application Object (FFh). The module will enter the EXCEPTION state. The module then has to be power-cycled or reset using the reset pin.

3.11.2 Reset Node

If a reset node is requested from the network, the module will issue a Reset command to the Application Object (FFh). The module will enter the EXCEPTION state. The module then has to be power-cycled or reset using the reset pin.

3.11.3 Reset Communication

The module will return to state WAIT_PROCESS, and the parameters in the Communication Profile Area (1000h-1FFFh) will be reset.

3.11.4 Reset Configuration

The module will return to state WAIT_PROCESS and the configuration parameters, set in the object dictionary, will be used to generate the active configuration. COP object entries marked as “valid on reset” will be activated.

4 Object Dictionary

4.1 Standard Objects

4.1.1 General

The standard object dictionary is implemented according to the EPSG (Ethernet POWERLINK Standardization Group) DS301. Note that certain object entries correspond to settings in the POWERLINK Object (E9h).

4.1.2 Access Rights

The view point is from the network into the device.

Access	Description
RW	Read and write, the value is not stored in flash memory.
WO	Write only, the value is not stored in flash memory.
RO	Read only.
CONST	Read only, constant value.
valid on reset	Any changes to this object will be valid after the reception of an NMTResetConfiguration command.

4.1.3 Object Entries



None of these objects can be mapped as TPDO or RPDO.

The type DOMAIN is used to transfer an arbitrarily large block of data between a client and a server. The contents of the data block is application specific.

1000h, NMT_DeviceType_U32

Subindex	Description	Type	Access	Value/Notes
00h	Device Type	UNSIGNED32	const	Default 0000 0000h (No profile). Can be managed through the POWERLINK Object, which can optionally be implemented in the host application. See POWERLINK Object (E9h) , p. 116.

1001h, ERR_ErrorRegister_U8

Subindex	Description	Type	Access	Value/Notes
00h	Error register	UNSIGNED8	ro	00h (default)

1006h, NMT_CycleLen_U32

Subindex	Description	Type	Access	Value/Notes
00h	Cycle time of full POWERLINK cycle.	UNSIGNED32	rw, valid on reset	200 (default) Valid range 200 - 2147483 (μs)

1008h, NMT_ManufactDevName_VS

Subindex	Description	Type	Access	Value/Notes
00h	Manufacturer device name	VISIBLE_STRING64	const	This entry is managed through the POWERLINK Object, which can optionally be implemented in the host application. See POWERLINK Object (E9h) , p. 116.

1009h, NMT_ManufactHwVers_VS

Subindex	Description	Type	Access	Value/Notes
00h	Manufacturer hardware version	VISIBLE_STRING64	const	This entry is managed through the POWERLINK Object, which can optionally be implemented in the host application. See POWERLINK Object (E9h) , p. 116.

100Ah, NMT_ManufactSwVers_VS

Subindex	Description	Type	Access	Value/Notes
00h	Manufacturer software version	VISIBLE_STRING64	const	This entry is managed through the POWERLINK Object, which can optionally be implemented in the host application. See POWERLINK Object (E9h) , p. 116.

1018h, Identity Object

Subindex	Description	Type	Access	Value/Notes
00h	Number of entries	UNSIGNED8	const	04h
01h	VendorId_U32: Vendor ID	UNSIGNED32	const	These entries are managed through the POWERLINK Object, which can optionally be implemented in the host application. See POWERLINK Object (E9h) , p. 116.
02h	ProductCode_U32: Product Code	UNSIGNED32	const	
03h	RevisionNo_U32: Revision Number	UNSIGNED32	const	
04h	SerialNo_U32: Serial Number	UNSIGNED32	const	

1020h, CFM_VerifyConfiguration_REC

Subindex	Description	Type	Access	Value/Notes
00h	Number of entries	UNSIGNED8	const	02h
01h	ConfDate_U32: Configuration Date	UNSIGNED32	rw, valid on reset	0 (default) Valid range 0 - FFFFFFFFh
02h	ConfTime_U32: Configuration Time	UNSIGNED32	rw, valid on reset	0 (default) Valid range 0 - FFFFFFFFh

1021h, CFM_StoreDevDescrFile_DOM

Subindex	Description	Type	Access	Value/Notes
00h	The contents of an XDD file.	DOMAIN	ro	If the object 1022h has the value 00h, this object will contain an XDD file adapted to the specific host application. See XML Device Description (XDD) , p. 11

1022h, CFM_StoreDevDescrFormat_U16

Subindex	Description	Type	Access	Value/Notes
00h	Description of the format of what is stored in object 1021h.	UNSIGNED16	ro	00h: Object 1021h contains an XDD file. FFh: Object 1021h does not contain an XDD file.

1030h, NMT_InterfaceGroup_00h_REC

Subindex	Description	Type	Access	Value/Notes
00h	Number of entries	UNSIGNED8	const	09h
01h	InterfaceIndex_U16: Interface index	UNSIGNED16	ro	0001h
02h	InterfaceDescription_VSTR: Interface description	VISIBLE_STRING194	const	See POWERLINK Object (E9h) , p. 116 for default value (Manufacturer name, Manufacturer product name, and Manufacturer hardware version).
03h	InterfaceType_U8: Interface type	UNSIGNED8	const	06h
04h	InterfaceMtu_U16: Interface maximum transmission unit	UNSIGNED16	const	1500 bytes
05h	InterfacePhysAddress_OSTR: Interface Physical Address	OCTET_STRING6	const	MAC address, see description of Ethernet Host Object (F9h) , p. .
06h	InterfaceName_VSTR: Interface Name	VISIBLE_STRING11	ro	"Interface 1"
07h	InterfaceOperStatus_U8: Interface Operational Status	UNSIGNED8	ro	01h (default)
08h	InterfaceAdminState_U8: Interface Admin State	UNSIGNED8	rw	01h (default) Valid range 00h - 01h
09h	Valid_Bool: Valid	BOOLEAN	rw	01h (default) Valid range 00h - 01h

1300h, SDO_SequLayerTimeout_U32

Subindex	Description	Type	Access	Value/Notes
00h	SDO sequence layer timeout	UNSIGNED32	rw, valid on reset	15000 (default) Valid range 100 - FFFFFFFFh

1400h, PDO_RxCommParam_00h_REC

Subindex	Description	Type	Access	Value/Notes
00h	Number of entries	UNSIGNED8	const	02h
01h	NodeID_U8: Node ID	UNSIGNED8	rw	00h (default) For static mapping: 00h For dynamic mapping: set by MN
02h	MappingVersion_U8: Mapping version	UNSIGNED8	rw/ro	00h (default) For static mapping: 00h For dynamic mapping: set by MN For more info on dynamic an static mapping, see Process Data , p. 15

1600h, PDO_RxMappParam_00h_AU64

Subindex	Description	Type	Access	Value/Notes
00h	The number of ADI host application objects mapped as receive PDO	UNSIGNED8	rw	00h (default) For static mapping: decided by application For dynamic mapping: set by MN For more info on dynamic an static mapping, see Process Data, p. 15
01h - FEh	ObjectMapping	UNSIGNED64	rw/ro	For more info on dynamic an static mapping, see Process Data, p. 15

1800h, PDO_TxCommParam_00h_REC

Subindex	Description	Type	Access	Value/Notes
00h	Number of entries	UNSIGNED8	const	02h
01h	NodeID_U8: Node ID	UNSIGNED8	rw	00h (default)
02h	MappingVersion_U8: Mapping version	UNSIGNED8	rw/ro	00h (default) For static mapping: 00h For dynamic mapping: set by MN For more info on dynamic an static mapping, see Process Data, p. 15

1A00h, PDO_TxMappParam_00h_AU64

Subindex	Description	Type	Access	Value/Notes
00h	The number of ADI host application objects mapped as receive PDO	UNSIGNED8	rw	00h (default) For static mapping: decided by application For dynamic mapping: set by MN For more info on dynamic an static mapping, see Process Data, p. 15
01h - FEh	ObjectMapping	UNSIGNED64	rw/ro	For more info on dynamic an static mapping, see Process Data, p. 15

1C0Bh, DLL_CNLossSoC_REC

Subindex	Description	Type	Access	Value/Notes
00h	Number of entries	UNSIGNED8	const	03h
01h	CumulativeCnt_U32: Cumulative count	UNSIGNED32	rw	0 (default) Valid range 0 - FFFFFFFFh Note: if the module is reset, this attribute will be set to its default value.
02h	ThresholdCnt_U32: Threshold count	UNSIGNED32	ro	00h
03h	Threshold_U32: Threshold	UNSIGNED32	rw	15 (default) Valid range 0 - FFFFFFFFh

1C0Fh, DLL_CNCRCErrror_REC

Subindex	Description	Type	Access	Value/Notes
00h	Number of entries	UNSIGNED8	const	03h
01h	CumulativeCnt_U32: Cumulative count	UNSIGNED32	rw	0 (default) Valid range 0 - FFFFFFFFh Note: if the module is reset, this attribute will be set to its default value.
02h	ThresholdCnt_U32: Threshold count	UNSIGNED32	ro	00h
03h	Threshold_U32: Threshold	UNSIGNED32	rw	15 (default) Valid range 0 - FFFFFFFFh

1C14h, DLL_CNLossOfSocTolerance_U32

Subindex	Description	Type	Access	Value/Notes
00h		UNSIGNED32	rw	100 000 (default) Valid range 0 - 2147483000 (ns)

1E40h, NWL_IpAddrTable_Xh_REC

This object does not exist if the IT functionality is disabled.

Subindex	Description	Type	Access	Value/Notes
00h	Number of entries	UNSIGNED8	const	05h
01h	IfIndex_U16	UNSIGNED16	ro	1
02h	Addr_IPAD	IP_ADDRESS	ro	As the object describes an Ethernet POWERLINK interface, access is read only. Value: 192.168.100.xxx with xxx = NMT_EPLNodeID_REC_U8.
03h	NetMask_IPAD	IP_ADDRESS	ro	As the object describes an Ethernet POWERLINK interface, access is read only. Default value: 255.255.255.000.
04h	ReasmMaxSize_U16	UNSIGNED16	ro	Default value: 1500
05h	DefaultGateway_IPAD	IP_ADDRESS	rw	Default value: 192.168.100.254

1E4Ah, NWL_IpGroup_REC

This object does not exist if the IT functionality is disabled.

Subindex	Description	Type	Access	Value/Notes
00h	Number of entries	UNSIGNED8	const	02h (Default)
01h	Forwarding_BOOL	BOOLEAN	ro	0: Not forwarding Access is read only as the device is unable to forward IT frames
02h	DefaultTTL_U16	UNSIGNED16	rw	64 Any 16 bit value can be written to this entry, but the written value is ignored since the TTL value is always supplied by the TCP/IP stack.

1F50h, PDL_DownloadProgData_ADOM

Subindex	Description	Type	Access	Value/Notes
00h	Number of entries	UNSIGNED8	rw	01h
01h	Program	DOMAIN	wo	A HIFF (HMS Interchange File Format) file, used to update the firmware of the module.

1F51h, PDL_ProgCtrl_AU8

Subindex	Description	Type	Access	Value/Notes
00h	Number of entries	UNSIGNED8	rw	01h
01h	ProgCtrl: Program control	UNSIGNED8	rw	01h (Writing a value other than 1 will result in an Abort SDO Transfer message (error code 08000024h.))

1F52h, PDL_LocVerApplSw_REC

Subindex	Description	Type	Access	Value/Notes
00h	Number of entries	UNSIGNED8	const	02h
01h	ApplSwDate_U32	UNSIGNED32	rw	Number of days between 1984-01-01 and the current software build time.
02h	ApplSwTime_U32	UNSIGNED32	rw	Milliseconds since midnight of the current software build time.

1F81h, NMT_Node Assignment_AU32

Subindex	Description	Type	Access	Value/Notes
00h	Number of entries	UNSIGNED8	rw, valid on reset	FEh (default)Valid range 01h - FEh
01h	NodeAssignment	UNSIGNED32	rw, valid on reset	0 (default)This value is a bit field.

1F82h, NMT_FeatureFlags_U32

Subindex	Description	Type	Access	Value/Notes
00h	Feature flags	UNSIGNED32	const	48245h if dynamic mapping is supported, and the node ID is set by hardware.48205h if static mapping is supported and the node ID is set by hardware. 48645h if dynamic mapping is supported, and the node ID is set by software.48605h if static mapping is supported and the node ID is set by software.

1F83h, NMT_EPLVersion_U8

Subindex	Description	Type	Access	Value/Notes
00h	Ethernet POWERLINK version	UNSIGNED8	const	20h

1F8Ch, NMT_CurrNMTState_U8

Subindex	Description	Type	Access	Value/Notes
00h	Current NMT State	UNSIGNED8	ro	-

1F8Dh, NMT_PresPayloadLimitList_AU16

Subindex	Description	Type	Access	Value/Notes
00h	Number of entries	UNSIGNED8	rw, valid on reset	FEh (default) Valid range 01h - FEh
01h - FEh	PresPayloadLimit	UNSIGNED16	rw, valid on reset	36 (default) Valid range: 0, 36 - 1490, FFFFh In this object, subindices 1..254, there are two special values: If the value of subindex <i>index</i> is 0, the node shall not listen to PRes frames from the node with node ID <i>index</i> . If the value is FFFFh, the PRes payload limit of the node with node ID <i>index</i> is equal to the value of 1F98/2h. Otherwise the value at sub-index <i>index</i> defines the PRes payload limit of the node with node ID <i>index</i> .

1F93h, NMT_EPLNodeID_REC

Subindex	Description	Type	Access	Value/Notes
00h	Number of entries	UNSIGNED8	const	02h
01h	NodeID_U8: NodeID	UNSIGNED8	ro	The configured node ID, see Network Configuration Object (04h) , p. 84
02h	NodeIDByHW_BOOL: NodeID by hardware	BOOLEAN	ro	01h if the application configured the node ID in the Anybus SETUP state, otherwise 00h.

1F98h, NMT_CycleTiming_REC

Subindex	Description	Type	Access	Value/Notes
00h	Number of entries	UNSIGNED8	const	08h
01h	IsochrTxMaxPayload_U16	UNSIGNED16	const	1490
02h	IsochrRxMaxPayload_U16	UNSIGNED16	const	1490
03h	PresMaxLatency_U32	UNSIGNED32	const	1000
04h	PReqActPayloadLimit_U16	UNSIGNED16	rw, valid on reset	The mapped sized is measured in number of bytes. Valid range 36 - 1490 For dynamic mapping: 36 (default) For static mapping: The default value will match the size of the process data the module is configured to receive from the network.
05h	PResActPayloadLimit_U16	UNSIGNED16	rw, valid on reset	The mapped sized is measured in number of bytes. Valid range 36 - 1490 For dynamic mapping: 36 (default) For static mapping: The default value will match the size of the process data the module is configured to receive from the network.
06h	AsndMaxLatency_U32	UNSIGNED32	const	1000
07h	MultiplCycleCnt_U8	UNSIGNED8	rw, valid on reset	0 (default) Valid range 0 - 255
08h	AsyncMTU_U16	UNSIGNED16	rw, valid on reset	300 default) Valid range 300 - 1500

1F99h, NMT_CNBasicEthernetTimeout_U32

Subindex	Description	Type	Access	Value/Notes
00h	After booting, this is the maximum time in microseconds the module silently listens for POWERLINK traffic on the network, before it decides if it should go to the Basic Ethernet state (no EPL traffic) or Pre-operational 1 state (with EPL traffic).	UNSIGNED32	rw, valid on reset	5000000 (default)

1F9Ah, NMT_HostName_VSTR

This object does not exist if the IT functionality is disabled.

Subindex	Description	Type	Access	Value/Notes
00h	Host name	VISIBLE_STRING32	rw	<yy-zzzzzzz> where yy is the node ID and zzzzzzz is the vendor ID, both hexadecimally encoded with leading zeroes, e.g 01-0000001b when node ID is 01h and vendor ID 1Bh.

1F9Bh, NMT_MultiplCycleAssign_AU8

Subindex	Description	Type	Access	Value/Notes
00h	Number of entries	UNSIGNED8	rw, valid on reset	FEh (default) Valid range 01h - FEh
01h - FEh	Cycle number	UNSIGNED8	rw, valid on reset	0 (default) Valid range 0 - NMT_Cycletiming_REC. MultiplCycleCnt_U8 (the value of object 1F98h subindex 7).

1F9Eh, NMT_ResetCmd_U8

Subindex	Description	Type	Access	Value/Notes
00h	Reset command	UNSIGNED8	rw	FFh

4.2 Manufacturer Specific Objects

4.2.1 General

Each object entry in the manufacturer specific range (2001h...FFFFh) corresponds to an instance (a.k.a. ADI) within the Application Data Object (FEh), i.e. network accesses to these objects results in object requests towards the host application. In case of an error, the status (or error) code returned in the response from the host application will be translated into the corresponding CANopen abort code.



If ADI has > 1 elements: COP subindex = (ADI element - 1).

If ADI has 1 element: COP subindex = (ADI element (only 0 is valid)).



As any access to these object entries will result in an object access towards the host application, the time spent communicating on the host interface must be taken into account when calculating the SDO timeout value.

4.2.2 Translation of Status Codes

Status (or error codes) are translated to POWERLINK abort codes as follows:

Anybus CompactCom Status Code	Anybus CompactCom Status Code No.	POWERLINK SDO Abort Code #	POWERLINK Abort Code Description "<hostname>"
Invalid message format	02h	0604 0047h	General internal incompatibility in the device.
Unsupported object	03h	0602 0000h	Object does not exist in the object dictionary.
Unsupported instance	04h	0602 0000h	Object does not exist in the object dictionary.
Unsupported command	05h	0604 0043h	General parameter incompatibility.
Invalid CmdExt[0]	06h	0602 0000h	Object does not exist in the object dictionary. (ADI access).
Invalid CmdExt[1]	07h	0609 0011h	Subindex does not exist. (ADI access).
Attribute not settable	08h	0601 0002h	Attempt to write a read only object.
Attribute not gettable	09h	0601 0001h	Attempt to read a write only object.
Too Much Data	0Ah	0607 0012h	Data type does not match, length of service parameter too high.
Not Enough Data	0Bh	0607 0013h	Data type does not match, length of service parameter too low.
Out of range	0Ch	0609 0030h	Value range of parameter exceeded (only for write access). Note: Use only when Anybus CompactCom status codes 11h (Value too high) or 12h (Value too low) cannot be used.
Invalid state	0Dh	0800 0022h	Data cannot be transferred or stored to the application because of the present device state.
Out of resources	0Eh	0504 0005h	Out of memory.
Segmentation failure	0Fh	0800 0000h	General error.
Segmentation buffer overflow	10h	0800 0000h	General error.
Value too high	11h	0609 0031h	Value of parameter written is too high.
Value too low	12h	0609 0032h	Value of parameter written is too low.
Attribute controlled from another channel	13h	0800 0021h	Data cannot be transferred or stored to the application because of local control.
Object Specific Error	FFh	0800 0000h	General error.

The default error code will be the 'General error' code (0800 0000h) if no corresponding error meets the error definition.

4.2.3 Network Data Format

Data is translated between the native network format and the Anybus data format as follows:

Anybus Data Type	Native POWERLINK Data Type	Notes
BOOL	UNSIGNED8	An Anybus BOOL is always 8 bits, unlike a POWERLINK BOOLEAN that can be mapped as a single bit.
SINT8	INTEGER8	
SINT16	INTEGER16	
SINT32	INTEGER32	
UINT8	UNSIGNED8	
UINT16	UNSIGNED16	
UINT32	UNSIGNED32	
CHAR or array of CHAR	VISIBLE_STRING n	An array of n characters is encoded as a VISIBLE_STRING n , e.g. VISIBLE_STRING32 when n is 32. The " n " in VISIBLE_STRING n specifies the maximum allowed length: the actual length may be shorter. It is not necessary to NULL terminate the string. In the XDD file, the type is always specified as VISIBLE_STRING, without a specified maximum length.
OCTET or array of OCTET	OCTET_STRING n	An array of n octets are encoded as a OCTET_STRING n , e.g. OCTET_STRING6 when n is 6. A single OCTET is treated as an array with one OCTET. In the XDD file, the type is always specified as OCTET_STRING, without a specified maximum length.
ENUM	UNSIGNED8	
BIT1	BOOLEAN or BIT1	It is recommended to map BIT1 as BOOLEAN
BIT n ($2 \leq n \leq 7$)	BIT n	
BITS8	UNSIGNED8 or BIT8	It is recommended to map BITS8 as UNSIGNED 8
BITS16	UNSIGNED16 or BIT16	It is recommended to map BITS16 as UNSIGNED 16
BITS32	UNSIGNED32 or BIT32	It is recommended to map BITS32 as UNSIGNED 32
SINT64	INTEGER64	
UINT64	UNSIGNED64	
FLOAT	REAL32	
PAD x ($0 \leq x \leq 16$)	VOID n or BIT n	Neither VOID n nor BIT n have a data type number and cannot be specified as a type in the device XDD file

The Ethernet POWERLINK BIT(S) data types do not have data type numbers and cannot be specified as a types in the device XDD file.



Every ADI must be specified in the XDD file with their corresponding Ethernet POWERLINK data type.



Single element ADIs are represented as simple variables.

ADIs with multiple elements of the same type are represented as arrays.

ADIs with multiple elements of different types are represented as RECORD.

ADIs of data type CHAR will be represented as VISIBLE_STRING. ADIs of data type OCTET will be represented as OCTET_STRING.

4.2.4 Object Entries

Requests for objects in the range 2001h to FFFFh will be forwarded to the host Application Data Object.

Index	Object Name	Notes
2001h	ADI 0001h	Object index 2001h corresponds to ADI 0001h
2002h	ADI 0002h	Object index 2002h corresponds to ADI 0002h
...
FFFFh	ADI DFFFh	Object index FFFFh corresponds to ADI DFFFh (57343)

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 alphanumeric characters.
- 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.

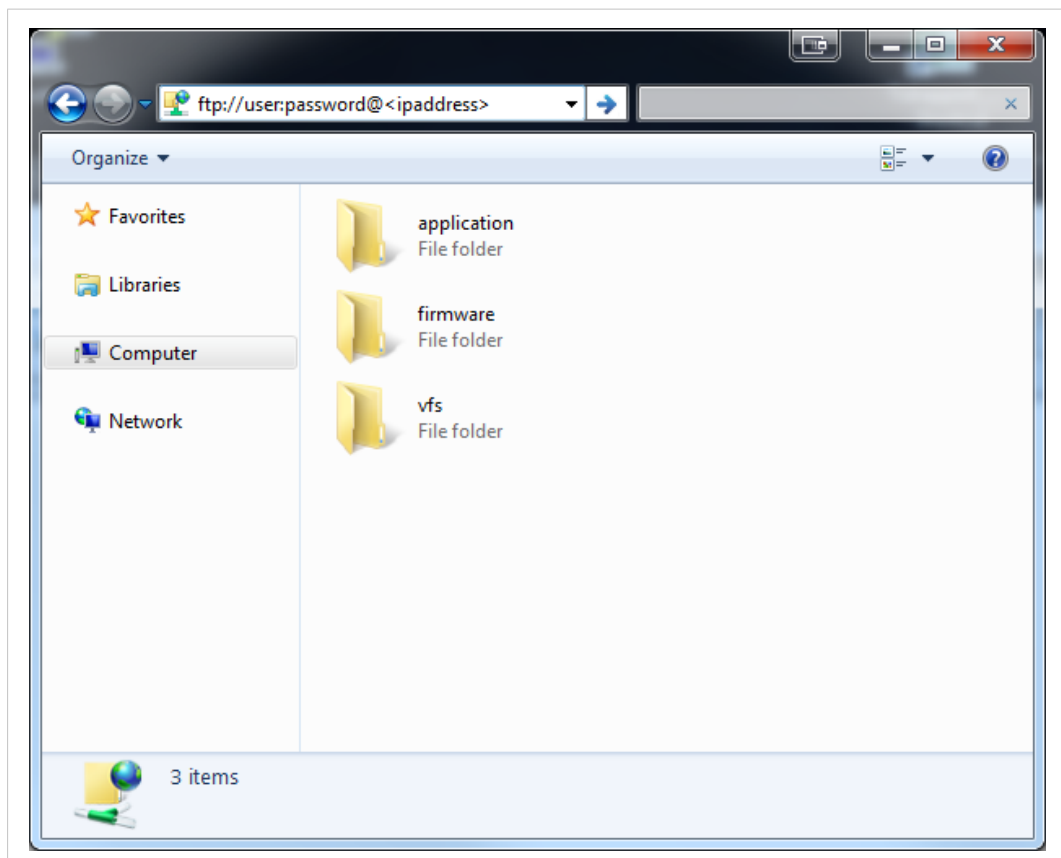


Fig. 3

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. 30](#)
- [Server Side Include \(SSI\), p. 59](#)
- [JSON, p. 39](#)
- [Ethernet Host Object \(F9h\), p. 120](#)

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.

6.2.1 Network Configuration

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

The figure displays two screenshots of the Network Configuration web interface. The top screenshot shows the 'IP Configuration' module, which includes a sidebar with navigation links (Overview, Parameters, NETWORK, Status, Configuration, SERVICES, SMTP) and a main content area with fields for DHCP (Enabled), IP Address (0.0.0.0), Subnet Mask (0.0.0.0), Gateway Address (0.0.0.0), Host Name, Domain name, DNS Server #1 (0.0.0.0), and DNS Server #2 (0.0.0.0). The bottom screenshot shows the 'SMTP configuration' module, which includes a sidebar with navigation links (Overview, Parameters, NETWORK, Status, Configuration, SERVICES, SMTP) and a main content area with fields for Server, User, Password, and Confirm password, along with a 'Save settings' button.

Fig. 4

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 Network Status Page

The Network Status 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 Status	Description
MAC Address	-
Port 1	The current link speed and duplex configuration.
Port 2	The current link speed and duplex configuration.
IT Interface Counters	Description
In Octets:	Octets received by the module on both ports (IT related communication only)
In Ucast Packets:	Unicast packets received on both ports (IT related communication only)
In NUcast packets:	Broadcast/Multicast packets received on both ports (IT related communication only)
In Discards:	Number of packets received on both ports that have been discarded (IT related communication only)
In Errors:	Number of erroneous packets received on both ports (IT related communication only)
In Unknown Protos	Number of packets discarded due to unknown protocol (IT related communication only)
Out Octets:	Octets sent by the module on both ports (IT related communication only)
Out Ucast packets:	Unicast packets sent on both ports (IT related communication only)
Out NUcast packets:	Broadcast/Multicast packets sent on both ports (IT related communication only)
Out Discards:	Number of packets sent that have been discarded (IT related communication only)
Out Errors:	Number of erroneous packets that couldn't be sent (IT related communication only)

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. 36
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. 32](#)

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 JSON

8.1 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. In Anybus CompactCom 40, it is used to transmit data objects consisting of name - value pairs between the webserver in the Anybus CompactCom 40 and a web application. The object members are unordered, thus the value pairs can appear in any order. JavaScripts are used to create dynamic web pages to present the values. Optionally, a callback may be passed to the GET-request for JSONP output.

JSON is more versatile than SSI in that you not only can read and write, but also change 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.

8.1.1 Encoding

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.

8.1.2 Access

It is recommended to password protect the JSON resources. Add password protection by adding a file called web_accs.cfg in the root directory (all web content will be protected). The file is described in the "Web Server" section in this document.

8.1.3 Error Response

If the module fails to parse or process a request, the response will contain an error object with an Anybus error code:

```
{
  "error"      : 02
}
```

The Anybus error codes are listed in the Anybus CompactCom 40 Software Design Guide.

8.2 JSON Objects

8.2.1 ADI

info.json

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

This object holds information about the ADI JSON interface. This data is static during runtime.

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

JSON response example:

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

data.json

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

These object calls fetch a sorted list of up to <count> ADIs values, starting from <offset> or <instance>. The returned values may change at any time during runtime.

Request data:

Name	Data Type	Description
offset	Number	Offset is the "order number" of the first requested ADI. The first implemented ADI will always get order number 0. <count> number of existing ADI values will be returned. I.e. non-existing ADIs are skipped.
inst	Number	Instance number of first requested ADI. <count> number of ADI values is returned. A null value will be returned for non-existing ADIs
count	String	Number of requested ADI values
callback	Number	Optional. A callback function for JSONP output.

Response data:

Name	Data Type	Description
—	Array of Strings	Sorted list of string representations of the ADI value attributes

JSON response example (using offset):

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

JSON response example (using inst):

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

metadata.json

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

These object calls fetch a sorted list of metadata objects for up to <count> ADIs, starting from <offset> or <instance>.

The returned information provided is a transparent representation of the attributes available in the host Application Data object (FEh). See the Anybus CompactCom 40 Software Design Guide for more information about the content of each attribute.

The ADI metadata is static during runtime.

Request data:

Name	Data Type	Description
offset	Number	Offset is the "order number" of the first requested ADI. The first implemented ADI will always get order number 0. Metadata objects for <count> number of existing ADI will be returned. I.e. non-existing ADIs are skipped.
inst	Number	Instance number of first requested ADI. Metadata objects for <count> number of ADI values are returned. A null object will be returned for non-existing ADIs
count	String	Number of requested ADI values
callback	Number	Optional. A callback function for JSONP output.

Response data:

Name	Data Type	Description
instance	Number	-
name	String	May be NULL if no name is present.
numelements	Number	-
datatype	Number	-
min	String	Hex formatted string, see Hex Format Explained, p. 57 for more information. May be NULL if no minimum value is present.
max	String	Hex formatted string, see Hex Format Explained, p. 57 for more information. May be NULL if no maximum value is present.
access	Number	Bit 0: Read access Bit 1: Write access

JSON response example (using offset):

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

JSON response example (using inst):

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

metadata2.json

```
GET adi/metadata2.json?offset=<offset>&count=<count>[&callback=<function>]
GET adi/metadata2.json?inst=<instance>&count=<count>[&callback=<function>]
```

This is an extended version of the metadata function that provides complete information about the ADIs. This extended version is needed to describe more complex data types such as Structures.

The information provided is a transparent representation of the attributes available in the host Application Data object (FEh). See the Anybus CompactCom 40 Software Design Guide for more information about the content of each attribute.

The ADI metadata is static during runtime.

Request data:

Name	Data Type	Description
offset	Number	Offset is the “order number” of the first requested ADI. The first implemented ADI will always get order number 0. Metadata objects for <count> number of existing ADI will be returned. I.e. non-existing ADIs are skipped.
inst	Number	Instance number of first requested ADI. Metadata objects for <count> number of ADI values are returned. A null object will be returned for non-existing ADIs
count	String	Number of requested ADI values
callback	Number	Optional. A callback function for JSONP output.

Response data:

Name	Data Type	Description
instance	Number	-
numelements	Array of umbers	-
datatype	Array of Numbers	Array of datatypes. For Structures and Variables, each array element defines the data type of the corresponding element of the instance value. For Arrays, one array element defines the data type for all elements of the instance value.
descriptor		Array of descriptors. For Structures and Variables, each array element defines the descriptor of the corresponding element of the instance value. For Arrays, one array element defines the descriptor for all elements of the instance value.
name		May be NULL if no name is present.
min	String	Hex formatted string, see Hex Format Explained, p. 57 for more information. May be NULL if no minimum value is present.
max	String	Hex formatted string, see Hex Format Explained, p. 57 for more information. May be NULL of no maximum value is present.
default	String	Hex formatted string, see Hex Format Explained, p. 57 for more information. May be NULL of no default value is present.
numsubelements	Array of Numbers	For Structures and Variables each array element defines the number of subelements of the corresponding element of the instance value. May be NULL if not present.
elementname	Array of Strings	Array of names, one for each instance value element. May be NULL if not present.

JSON response example (using offset):

```
[
{
  "instance": 1,
  "numelements": 1,
  "datatype": [0 ],
  "descriptor": [9 ],
  "name": "Temperature threshold",
  "max": "FF",
  "min": "00",
  "default": "00",
  "numsubelements": null
  "elementname": null
},
{
  ...
}
]
```

JSON response example (instance):

```
[
{
  "instance": 1,
  "numelements": 1,
  "datatype": [0 ],
  "descriptor": [9 ],
  "name": "Temperature threshold",
  "max": "FF",
  "min": "00",
  "default": "00",
  "numsubelements": null
  "elementname": null
},
null,
null
{
  ...
}
]
```

enum.json

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

This object call fetches a list of enumeration strings for a specific instance.

The ADI enum strings are static during runtime.

Request data:

Name	Data Type	Description
inst	Number	Instance number of the ADI to get enum string for.
value	Number	Optional. If given only the enumstring for the requested <value> is returned.
callback	String	Optional. A callback function for JSONP output.

Response data:

Name	Data Type	Description
string	String	String representation for the corresponding value.
value	Number	Value corresponding to the string representation.

JSON response example:

```
[
  {
    "string": "String for value 1",
    "value": 1
  },
  {
    "string": "String for value 2",
    "value": 2
  },
  {
    ...
  }
]
```

update.json

```
POST adi/update.json
```

Form data:

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

Updates the value attribute of an ADI.

Request data:

Name	Data Type	Description
inst	Number	Instance number of the ADI
value	String	Value to set. If the value attribute is a number it shall be hex formatted, see Hex Format Explained, p. 57 for more information.
elem	Number	Optional. If specified only a single element of the ADI value is set. Then <data> shall only contain the value of the specified <element>.
callback	String	Optional. A callback function for JSONP output.

Response data:

Name	Data Type	Note
result	Number	0 = success The Anybus CompactCom error codes are used. Please see the Anybus CompactCom 40 Software Design Guide.

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


8.2.2 Module

info.json

```
GET module/info.json
```

Response data:

Name	Data Type	Description
modulename	String	-
serial	String	32 bit hex ASCII
fwver	Array of Number	(major, minor, build)
uptime	Array of Number	[high, low] milliseconds (ms)
cpuload	Number	CPU load in %
fwvertext	String	Firmware version in text
vendorname	String	Vender name (Application Object (FFh), instance attribute #8)
hwvertext	String	Hardware version in text
networktype	Number	Network type (Network Object (03h), instance attribute #1)

JSON response example:

```
{
  "modulename": "ABCC M40",
  "serial": "ABCDEF00",
  "fwver": [ 1, 5, 0 ],
  "uptime": [ 5, 123456 ],
  "cpuload": 55
  "fwvertext": "1.05.02",
  "vendorname": "HMS Industrial Networks",
  "hwvertext": "2",
  "networktype": "0085",
}
```

8.2.3 Network

ethstatus.json

```
GET network/ethstatus.json.
```

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

JSON response example:

```
{
  "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":  "abcc123",
  "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
mac	String	-
comm1	Number	-
comm2	Number	Only present if two Ethernet ports are activated in the module.

The values of “comm1” and “comm2” are read from the Network Configuration object, instances #7 and #8.

```
{
  "mac":      [00, 30, 11, FF, 02, 01],
  "comm1":    0,
  "comm2":    4
}
```

The parameters “comm1” and “comm2” are configurable by adding them as arguments to the GET request:

```
GET network/ethconf.json?comm1=0&comm2=4
```

The parameters “comm1” and “comm2” may hold an error object with Anybus error code if the module fails processing the request:

```
{
  "mac":      [00, 30, 11, FF, 02, 01],
  "comm1":    0,
  "comm2":    { error: 14 },
}
```

The Anybus CompactCom error codes are used. Please see the Anybus CompactCom 40 Software Design Guide.

ifcounters.json

```
GET network/ifcounters.json?port=<port>
```

Valid values for the argument <port> are 0, 1, and 2.

- Valid values for the argument <port> are 0, 1, and 2.
- Port number 0 option refers to the internal port (CPU port).
- Port number 2 option is only valid if two Ethernet ports are activated in the module.

Name	Data Type	Description
inotets	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
outotets	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.

Port number 2 option is only valid if two Ethernet ports are activated in the module.

Name	Data Type	Description
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

nwstats.json

```
GET network/nwstats.json
```

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

Example output:

```
[
  or
  [ { "identifier": "eipstats", "title": "EtherNet/IP Statistics" } ]
  or
  [ { "identifier": "eitstats", "title": "Modbus TCP Statistics" } ]
  or
  [
    { "identifier": "bacnetipstats",
      "title": "BACnet/IP Statistics" },
    { "identifier": "bacnetaplserverstats",
      "title": "BACnet Application Layer Server Statistics" },
    { "identifier": "bacnetaplclientstats",
      "title": "BACnet Application Layer Client Statistics" },
    { "identifier": "bacnetalarmstats",
      "title": "BACnet Alarm and Event Module Statistics" }
  ]
  or
  [ { "identifier": "eplifcounters", "title": "IT Interface Counters" } ]
  or
  [
    { "identifier": "ectstats", "title": "EtherCAT Statistics" },
    { "identifier": "eoeifcounters", "title": "EoE Interface Counters" },
  ]
  or
  [ { "identifier": "pnpof", "title": "Fiber Optical Statistics" } ]
```

Get network specific statistics (<ID> is an “identifier” value returned from the previous command):

```
GET network/nwstats.json?get=<ID>
```

“eipstats”

```
[
  { "name": "Established Class1 Connections", "value": 0 },
  { "name": "Established Class3 Connections", "value": 1 },
  { "name": "Connection Open Request", "value": 0 },
  { "name": "Connection Open Format Rejects", "value": 0 },
  { "name": "Connection Open Resource Rejects", "value": 0 },
  { "name": "Connection Open Other Rejects", "value": 0 },
  { "name": "Connection Close Requests", "value": 0 },
  { "name": "Connection Close Format Rejects", "value": 0 },
  { "name": "Connection Other Rejects", "value": 0 },
  { "name": "Connection Timeouts", "value": 0 },
]
```

“eitstats”

```
[
  { "name": "Modbus Connections", "value": 0 },
  { "name": "Connection ACKs", "value": 1 },
  { "name": "Connection NACKs", "value": 0 },
  { "name": "Connection Timeouts", "value": 0 },
  { "name": "Process Active Timeouts", "value": 0 },
  { "name": "Processed messages", "value": 0 },
  { "name": "Incorrect messages", "value": 0 },
]
```

“bacnetipstats”

```
[
  { "name": "Unconfirmed server requests received", "value": 0 },
  { "name": "Unconfirmed server requests sent", "value": 1 },
  { "name": "Unconfirmed client requests sent", "value": 0 },
]
```

“bacnetaplserversstats”

```
[
  { "name": "Active transactions", "value": 0 },
  { "name": "Max Active transactions", "value": 1 },
  { "name": "Tx segments sent", "value": 0 },
  { "name": "Tx segment ACKs received", "value": 0 },
  { "name": "Tx segment NAKs received", "value": 0 },
  { "name": "Rx segments received", "value": 0 },
  { "name": "Rx segment ACKs sent", "value": 0 },
  { "name": "Duplicate Rx segment ACKs sent", "value": 0 },
  { "name": "Rx segment NAKs sent", "value": 0 },
  { "name": "Confirmed transactions sent", "value": 0 },
  { "name": "Confirmed transactions received", "value": 0 },
  { "name": "Tx segment timeouts", "value": 0 },
  { "name": "Rx segment timeouts", "value": 0 },
  { "name": "Implicit deletes", "value": 0 },
  { "name": "Tx timeout deletes", "value": 0 },
  { "name": "Rx timeout deletes", "value": 0 },
  { "name": "Tx aborts received", "value": 0 },
  { "name": "Rx aborts received", "value": 0 },
  { "name": "Transaction aborts sent", "value": 0 },
  { "name": "Transaction rejects sent", "value": 0 },
  { "name": "Transaction errors sent", "value": 0 },
]
```


“bacnetaplcclientstats”

```
[
  { "name": "Active transactions", "value": 0 },
  { "name": "Max Active transactions", "value": 1 },
  { "name": "Tx segments sent", "value": 0 },
  { "name": "Tx segment ACKs received", "value": 0 },
  { "name": "Tx segment NAKs received", "value": 0 },
  { "name": "Rx segments received", "value": 0 },
  { "name": "Rx segment ACKs sent", "value": 0 },
  { "name": "Duplicate Rx segment ACKs sent", "value": 0 },
  { "name": "Rx segment NAKs sent", "value": 0 },
  { "name": "Confirmed transactions sent", "value": 0 },
  { "name": "Confirmed transactions received", "value": 0 },
  { "name": "Tx segment timeouts", "value": 0 },
  { "name": "Rx segment timeouts", "value": 0 },
  { "name": "Implicit deletes", "value": 0 },
  { "name": "Tx timeout deletes", "value": 0 },
  { "name": "Rx timeout deletes", "value": 0 },
  { "name": "Tx aborts received", "value": 0 },
  { "name": "Rx aborts received", "value": 0 },
  { "name": "Transaction aborts sent", "value": 0 },
  { "name": "Transaction rejects sent", "value": 0 },
  { "name": "Transaction errors sent", "value": 0 },
]
```

“bacnetalarmstats”

```
[
  { "name": "COV Active subscriptions", "value": 0 },
  { "name": "COV Max active subscriptions", "value": 1 },
  { "name": "COV Lifetime subscriptions", "value": 0 },
  { "name": "COV Confirmed resumes", "value": 0 },
  { "name": "COV Unconfirmed resumes", "value": 0 },
  { "name": "COV Confirmed notifications sent", "value": 0 },
  { "name": "COV Unconfirmed notifications sent", "value": 0 },
  { "name": "COV Confirmed notification errors", "value": 0 },
  { "name": "AE Active events", "value": 0 },
  { "name": "AE Active NC recipients", "value": 0 },
  { "name": "AE Confirmed resumes", "value": 0 },
  { "name": "AE UnConfirmed resumes", "value": 0 },
  { "name": "AE Confirmed notifications sent", "value": 0 },
  { "name": "AE UnConfirmed notifications sent", "value": 0 },
  { "name": "AE Confirmed notification errors", "value": 0 },
  { "name": "AE DAB lookup errors", "value": 0 },
]
```

“eplifcounters”

```
[
  { "name": "In Octets", "value": 22967 },
  { "name": "In Ucast Packets", "value": 121 },
  { "name": "In NUcast Packets", "value": 31 },
  { "name": "In Discards", "value": 0 },
  { "name": "In Errors", "value": 0 },
  { "name": "In Unknown Protos", "value": 0 },
  { "name": "Out Octets", "value": 169323 },
  { "name": "Out Ucast Packets", "value": 168 },
  { "name": "Out NUcast Packets", "value": 16 },
  { "name": "Out Discards", "value": 0 },
  { "name": "Out Errors", "value": 0 },
]
```

“ectstats”

```
[
  { "name": "Logical EoE port link", "value": "Yes" },
  { "name": "Invalid frame counter IN port", "value": 1 },
  { "name": "Rx error counter IN port", "value": 1 },
  { "name": "Forwarded error counter IN port", "value": 1 },
  { "name": "Lost link counter IN port", "value": 1 },
  { "name": "Invalid frame counter OUT port", "value": 1 },
  { "name": "Rx error counter OUT port", "value": 1 },
  { "name": "Forwarded error counter OUT port", "value": 1 },
  { "name": "Lost link counter OUT port", "value": 1 },
]
```

“eoeifcounters”

```
[
  { "name": "In Octets", "value": 22967 },
  { "name": "In Ucast Packets", "value": 121 },
  { "name": "In NUcast Packets", "value": 31 },
  { "name": "In Discards", "value": 0 },
  { "name": "In Errors", "value": 0 },
  { "name": "In Unknown Protos", "value": 0 },
  { "name": "Out Octets", "value": 169323 },
  { "name": "Out Ucast Packets", "value": 168 },
  { "name": "Out NUcast Packets", "value": 16 },
  { "name": "Out Discards", "value": 0 },
  { "name": "Out Errors", "value": 0 },
]
```

“pnpof”

```
[
  { "name" : "Port 1 Temperature (C)", "value" : "41.37" },
  { "name" : "Port 1 Power Budget (dB)", "value" : "23.0" },
  { "name" : "Port 1 Power Budget Status", "value" : "OK" },
  { "name" : "Port 2 Temperature (C)", "value" : "40.57" },
  { "name" : "Port 2 Power Budget (dB)", "value" : "0.0" },
  { "name" : "Port 2 Power Budget Status", "value" : "OK" }
]
```

8.2.4 Services

smtp.json

```
GET services/smtp.json
```



Password is not returned when retrieving the settings.

Name	Data Type	Note
server	String	IP address or name of mail server, e.g. "mail.hms.se"
user	String	-

```
[
  { "server": "192.168.0.55" },
  { "user": "test" }
]
```

Set:

Form data:

```
[
  [server=192.168.0.56]&[user=test2]&[password=secret],
]
```

8.2.5 Hex Format Explained

The metadata max, min, and default fields and the ADI values are ASCII hex encoded binary data. If the data type is an integer, the endianness used is determined by the dataformat field found in adi/info.json.

Examples:

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

```
0500
```

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

```
414243
```

8.3 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>
      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>
```

9 Server Side Include (SSI)

9.1 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.

9.2 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)

9.3 Command Functions

9.3.1 General Information

Command functions executes commands and includes the result.

General Syntax

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

command: Command function, see below



"command" is limited to a maximum of 500 characters.

Command Functions

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

Command	Valid for E-mail Messages
printf()	Yes
scanf()	No

9.3.2 GetConfigItem()

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

File Format

The source file must have the following format:

```
[key1]
value1

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

Syntax:

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

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

Default Output

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

Example

The following SSI...

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

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

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

... returns the string 'Third'.

9.3.3 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 'filename'"
Authentication Error	"Authentication error"
File open error	"Failed to open file 'filename'"
File write error	"Could not store configuration to 'filename'"

Example

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

9.3.4 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. 78](#)

9.3.5 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)

9.3.6 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".

9.3.7 IncludeFile()

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

Syntax:

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

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

Default Output

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

Example

The following example demonstrates how to use this function.

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

Contents of 'info.txt':

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

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

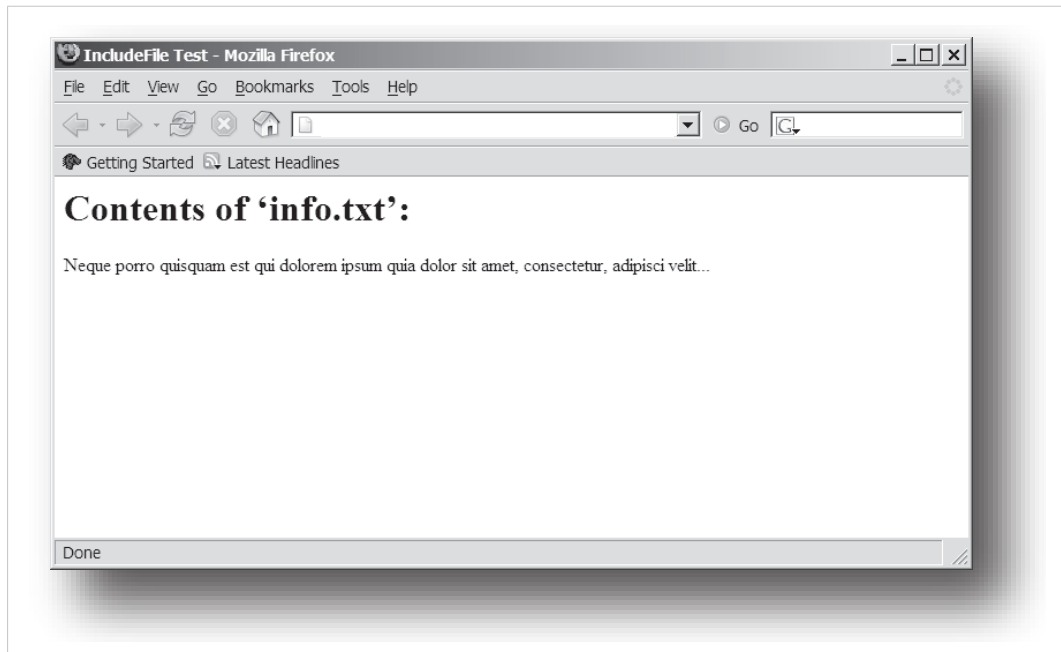


Fig. 5

See also...

- [Include File, p. 59](#)

9.3.8 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.shtm”.

9.3.9 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. 74 |

Default Output

Scenario	Default Output
Success	(printf() result)
ABCCMessage error	ABCCMessage error string (Errors, p. 77)

Example

See ..

- [ABCCMessage\(\), p. 74](#)
- [Example \(Get_Attribute\);, p. 76](#)

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.

- **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

9.3.10 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. 74

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. 77)

Example

See also...

[ABCCMessage\(\), p. 74](#)

[Example \(Set_Attribute\);, p. 76](#)

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: – 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

9.4 Argument Functions

9.4.1 General Information

Argument functions are supplied as parameters to certain command functions.

General Syntax:

(Syntax depends on context)

Argument Functions:

Function	Description
ABCCMessage()	-

9.4.2 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 ABCMessage() 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"
16	"Segmentation buffer overflow"
17... 255	"Unknown error"

See also...

[SSI Output Configuration, p. 78](#)

9.5 SSI Output Configuration

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

File format:

<pre>[ABCCMessage_X] 0:"Success string" 1:"Error string 1" 2:"Error string 2" ... 16:"Error string 16"</pre>	Each error code corresponds to a dedicated output string, labelled from 1 to 16. See Errors, p. 77
<pre>[GetConfigItem_X] 0:"Success string" 1:"Authentication error string" 2:"File open error string" 3:"Tag not found string"</pre>	Use "%s" to include the name of the file.
<pre>[SetConfigItem_X] 0:"Success string" 1:"Authentication error string" 2:"File open error string" 3:"File write error string"</pre>	Use "%s" to include the name of the file.
<pre>[IncludeFile_X] 0:"Success string" 1:"Authentication error string" 2:"File read error string"</pre>	Use "%s" to include the name of the file.
<pre>[scanf_X] 0:"Success string" 1:"Parsing error string"</pre>	-
<pre>[ChangeLanguage_X] 0:"Success string" 1:"Change error string"</pre>	-

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. 64](#)

10 Anybus Module Objects

10.1 General Information

This chapter specifies the Anybus Module Object implementation and how they correspond to the functionality in the Anybus CompactCom 40 Ethernet POWERLINK.

Standard Objects:

- [Anybus Object \(01h\), p. 80](#)
- [Diagnostic Object \(02h\), p. 81](#)
- [Network Object \(03h\), p. 82](#)
- [Network Configuration Object \(04h\), p. 84](#)

Network Specific Objects:

- [Socket Interface Object \(07h\), p. 90](#)
- [SMTP Client Object \(09h\), p. 107](#)
- [File System Interface Object \(0Ah\), p. 112](#)
- [Network Ethernet Object \(0Ch\), p. 113](#)

10.2 Anybus Object (01h)

Category

Basic

Object Description

This object assembles all common Anybus data, and is described thoroughly in the general *Anybus CompactCom 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 CompactCom 40)
2... 11	-	-	-	Consult the general Anybus CompactCom 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 CompactCom 40 Software Design Guide for further information.

Extended

#	Name	Access	Type	Value
17	Virtual attributes	Get/Set	-	Consult the general Anybus CompactCom 40 Software Design Guide for further information.
18	Black list/White list	Get/Set		
19	Network time	Get		

10.3 Diagnostic Object (02h)

Category

Basic

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*.

The module supports one instance of this object, reserved for a major unrecoverable diagnostic event. For POWERLINK, a major unrecoverable diagnostic event always causes the module to enter the state EXCEPTION. Thus these events will not be visible on the POWERLINK network.

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	1 (Major unrecoverable diagnostic event)
12	Supported functionality	Get	BITS32	0 (Latching events not supported)

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... 7	-	-	-	Consult the general Anybus CompactCom 40 Software Design Guide for further information.

In this implementation, only unrecoverable diagnostic events are detected, and no information is forwarded to the network.

10.4 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
	Map_ADI_Write_Ext_Area
	Map_ADI_Read_Ext_Area

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1	Name	Get	Array of CHAR	"Network"
2	Revision	Get	UINT8	02h
3	Number of instances	Get	UINT16	01h
4	Highest instance number	Get	UINT16	01h

Instance Attributes (Instance #1)

Basic

#	Name	Access	Type	Description
1	Network type	Get	UINT16	009Fh
2	Network type string	Get	Array of CHAR	"POWERLINK"
3	Data format	Get	ENUM	00h (LSB first; POWERLINK is a Little Endian network)
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, Remap_ADI_Write_Area and Map_ADI_Write_Ext_Area. Consult the general Anybus CompactCom 40 Software Design Guide 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, Remap_ADI_Read_Area and Map_ADI_Read_Ext_Area.
7	Exception Information	Get	UINT8	00h: No information 08h: Set if no MAC address is implemented in the Ethernet Host Object when running Anybus IP. (There is no default MAC address defined for Anybus IP)

10.5 Network Configuration Object (04h)

Category

Basic

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 the Node ID being set to its default value.

As soon as the used combination of IP address, Subnet mask and Gateway is changed, the module informs the application by writing the new set to instance #1, attribute #16 in the Ethernet Host Object (F9h).

See also...

- [Communication Settings, p. 11](#)
- [E-mail Client, p. 38](#)
- [Ethernet Host Object \(F9h\), p. 120](#)

Supported Commands

Object:	Get_Attribute
	Reset
Instance:	Get_Attribute
	Set_Attribute
	Get_Enum_String

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
3	Number of instances	Get	UINT16	12
4	Highest instance number	Get	UINT16	15

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

Instance Attributes (Instance #1, Node ID)

Value is used after module reset.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"Node ID" (Multilingual, see page 89)
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: 1 - 239 (Default = 0) As the default value is not in the valid range, the host application must specify a valid node ID before the module can communicate on the network. If the host application has not specified a valid node ID when the first POWERLINK frame is received, the Anybus CompactCom will continue to the Anybus state EXCEPTION, when trying to enter the super state NMT_CS_EPL_MODE. A value written in the Anybus state SETUP will take effect immediately. A value written in a later Anybus state will take effect the next time the module is booted. Writing to this attribute will always update attribute 6, "Configured Value", immediately.
6	Configured Value	Get	Array of UINT8	The configured value is equal to the last value written to attribute 5, "Value".

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 89)
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	If read, the actual value will be received. If written, the written value is reflected in attribute #6 until a reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)
6	Configured Value	Get	Array of UINT8	Holds the configured value, which will be written to attribute #5 after the module has been reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)

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 89)
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	If read, the actual value will be received. If written, the written value is reflected in attribute #6 until a reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)
6	Configured Value	Get	Array of UINT8	Holds the configured value, which will be written to attribute #5 after the module has been reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)

Instance Attributes (Instance #5, Gateway Address)

Value is used after module reset.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"Gateway" (Multilingual, see page 89)
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	If read, the actual value will be received. If written, the written value is reflected in attribute #6 until a reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)
6	Configured Value	Get	Array of UINT8	Holds the configured value, which will be written to attribute #5 after the module has been reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)

Instance Attributes (Instance #6, DHCP Enable)

Value is used after module reset.

#	Name	Access	Data Type	Description									
1	Name	Get	Array of CHAR	“DHCP” (Multilingual, see page 89)									
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	<div>If read, the actual value will be received. If written, the written value is reflected in attribute #6 until a reset. (Multilingual, see page 89)</div> <table><tr><th><u>Value</u></th><th><u>String</u></th><th><u>Meaning</u></th></tr><tr><td>00h</td><td>“Disable”</td><td>DHCP disabled</td></tr><tr><td>01h</td><td>“Enable”</td><td>DHCP enabled (default)</td></tr></table>	<u>Value</u>	<u>String</u>	<u>Meaning</u>	00h	“Disable”	DHCP disabled	01h	“Enable”	DHCP enabled (default)
<u>Value</u>	<u>String</u>	<u>Meaning</u>											
00h	“Disable”	DHCP disabled											
01h	“Enable”	DHCP enabled (default)											
6	Configured Value	Get	ENUM	<div>Holds the configured value, which will be written to attribute #5 after the module has been reset.</div> <table><tr><th><u>Value</u></th><th><u>String</u></th><th><u>Meaning</u></th></tr><tr><td>00h</td><td>“Disable”</td><td>DHCP disabled</td></tr><tr><td>01h</td><td>“Enable”</td><td>DHCP enabled</td></tr></table>	<u>Value</u>	<u>String</u>	<u>Meaning</u>	00h	“Disable”	DHCP disabled	01h	“Enable”	DHCP enabled
<u>Value</u>	<u>String</u>	<u>Meaning</u>											
00h	“Disable”	DHCP disabled											
01h	“Enable”	DHCP enabled											

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 89)
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	If read, the actual value will be received. If written, the written value is reflected in attribute #6 until a reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)
6	Configured Value	Get	Array of UINT8	Holds the configured value, which will be written to attribute #5 after the module has been reset. 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 89)
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	If read, the actual value will be received. If written, the written value is reflected in attribute #6 until a reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)
6	Configured Value	Get	Array of UINT8	Holds the configured value, which will be written to attribute #5 after the module has been reset. 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 89)
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	20h (32 elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of CHAR	If read, the actual value will be received. If written, the written value is reflected in attribute #6 until a reset. Host name, 32 characters
6	Configured Value	Get	Array of CHAR	Holds the configured value, which will be written to attribute #5 after the module has been reset. Host name, 32 characters

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 89)
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	If read, the actual value will be received. If written, the written value is reflected in attribute #6 until a reset. Domain name, 48 characters
6	Configured Value	Get	Array of CHAR	Holds the configured value, which will be written to attribute #5 after the module has been reset. Domain name, 48 characters

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 89)
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	If read, the actual value will be received. If written, the written value is reflected in attribute #6 until a reset. SMTP server address, 64 characters.
6	Configured Value	Get	Array of CHAR	Holds the configured value, which will be written to attribute #5 after the module has been reset. SMTP server address, 64 characters.

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 89)
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	If read, the actual value will be received. If written, the written value is reflected in attribute #6 until a reset. SMTP account user name, 64 characters
6	Configured Value	Get	Array of CHAR	Holds the configured value, which will be written to attribute #5 after the module has been reset. SMTP account user name, 64 characters

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 89)
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	If read, the actual value will be received. If written, the written value is reflected in attribute #6 until a reset. SMTP account password, 64 characters
6	Configured Value	Get	Array of CHAR	Holds the configured value, which will be written to attribute #5 after the module has been reset. SMTP account password, 64 characters

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
1	Node ID	Knoten ID	ID Nodo	ID Node	ID Neoud
3	IP address	IP-Adresse	Dirección IP	Indirizzo IP	Adresse IP
4	Subnet mask	Subnetzmaske	Masac. subred	Sottorete	Sous-réseau
5	Gateway	Gateway	Pasarela	Gateway	Passerelle
6	DHCP	DHCP	DHCP	DHCP	DHCP
	Enable	Einschalten	Activado	Abilitato	Activé
	Disable	Ausschalten	Desactivado	Disabilitato	Désactivé
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

10.6 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. 105](#).



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)
	DNS_Lookup (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)
	P_Add_membership (See below)
	IP_Drop_membership (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	0008h (8 instances): BACnet/IP 0014h (20 instances): All other industrial Ethernet networks

Instance Attributes (Sockets #1...Max. no. of instances)

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	<p>IP Multicast TTL value (UDP sockets only).</p> <p>Default = 1.</p>
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>
12	(reserved)			
13	TCP No Delay	Get/Set	BOOL	<p>Don't delay send to coalesce packets (TCP).</p> <p><u>Value</u> <u>Meaning</u></p> <p>1 Delay (default)</p> <p>0 Don't delay (turn off Nagle's algorithm on socket)</p>
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><tr><td><u>Value:</u></td><td><u>Socket Type:</u></td></tr><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></table>	<u>Value:</u>	<u>Socket Type:</u>	00h	SOCK_STREAM, NON-BLOCKING (TCP)	01h	SOCK_STREAM, BLOCKING (TCP)	02h	SOCK_DGRAM, NON-BLOCKING (UDP)	03h	SOCK_DGRAM, BLOCKING (UDP)
<u>Value:</u>	<u>Socket Type:</u>										
00h	SOCK_STREAM, NON-BLOCKING (TCP)										
01h	SOCK_STREAM, BLOCKING (TCP)										
02h	SOCK_DGRAM, NON-BLOCKING (UDP)										
03h	SOCK_DGRAM, BLOCKING (UDP)										

- Response Details

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

Command Details: Delete

Category

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> 00h 01h 02h</td><td><u>Mode:</u> Shutdown receive channel Shutdown send channel Shutdown both receive- and send channel</td></tr></table>	<u>Value:</u> 00h 01h 02h	<u>Mode:</u> Shutdown receive channel Shutdown send channel Shutdown both receive- and send channel
<u>Value:</u> 00h 01h 02h	<u>Mode:</u> Shutdown receive channel Shutdown send channel 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.

Host initiates shutdown:

1. Receive data on socket, if zero bytes received it indicates that the host closed the receive channel of the socket.
2. Try to send any unsent data to the host.
3. Send shutdown with CmdExt[1] set to 01h. This will send FIN-flag to host shutting down the send 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. 105](#)).

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. 105
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. 105](#)).

Field	Contents	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control bits	For more information, see Message Segmentation, p. 105
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. 105](#)).

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. 105
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. 105](#)).

Field	Contents	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control bits	For more information, see Message Segmentation, p. 105
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. 105](#)).

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. 105](#)).

Field	Contents	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control	(For more information, see Message Segmentation, p. 105)
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. 105](#)).

- 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. 105](#)).

Field	Contents	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control	For more information, see Message Segmentation, p. 105
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_Lookup

Category

Extended

Details

Command Code	1Bh
Valid for:	Object

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
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)

10.7 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:	Instance

Description

This command sends the specified e-mail instance.

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

Object Specific Error Codes

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

10.8 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. Available directories are the “firmware” directory and the “XDD” directory.

A dynamically generated XDD file can be read via this object.

See the Anybus CompactCom 40 Software Design Guide for more information.

Supported Commands

See the Anybus CompactCom 40 Software Design Guide for information about object and instance commands.

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1	Name	Get	Array of CHAR	“Anybus File System Interface”
2	Revision	Get	UINT8	01h
3	Number of instances	Get	UINT16	–
4	Highest instance number	Get	UINT16	–
11	Max. no. of instances	Get	UINT16	0001h
12	Disable virtual file system	Get	BOOL	False
13	Total disc size	Get	Array of UINT32	-
14	Free space	Get	Array of UINT32	-
15	Disc CRC	Get	Array of UINT32	-

Instance Attributes (Instance #1)

Extended

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

10.9 Network Ethernet Object (0Ch)

Category

Extended

Object Description

This object provides Ethernet-specific information to the application.

The object has three instances, each corresponding to a port:

Instance #	Port
1	Internal port
2	Port 1
3	Port 2

Each instance provides statistic counters for the port. This information can e.g be presented on internal web pages, if present, using the JSON script language.



Instance attribute #1 is reserved and used for backwards compatibility with earlier applications.

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	3
4	Highest instance no.	Get	UINT16	3

Instance Attributes (Instance #1)

#	Name	Access	Data Type	Description
1	MAC Address	Get	Array of UINT8	Reserved, used for backwards compatibility. (Device MAC address.) (See also Ethernet Host Object (F9h) , p. 120)
2	(Reserved)			
3	(Reserved)			
4	MAC Address	Get	Array of UINT8	Device MAC address
5	Interface Counters	Get	Array of UINT32	Array containing IT interface counters (No hardware interface counters are implemented) See table below for array indices.
6	(Reserved)			

Instance Attributes (Instances #2 - #3)

(no attributes available)

Interface Counters

: Array indices of Interface Counters attribute (#5)

Index	Name	Description
0	In octets	Octets received on the interface
1	In Unicast Packets	Unicast packets received on the interface
2	In Non-Unicast Packets	Non-unicast packets (multicast/broadcast) packets received on the interface
3	In Discards	Inbound packets received on the interface but discarded
4	In Errors	Inbound packets that contain errors (does not include In Discards)
5	In Unknown Protos	Inbound packets with unknown protocol
6	Out Octets	Octets transmitted on the interface
7	Out Unicast packets	Unicast packets transmitted on the interface
8	Out Non-Unicast Packets	Non-unicast (multicast/broadcast) packets transmitted on the interface
9	Out Discards	Outbound packets discarded
10	Out Errors	Outbound packets that contain errors

11 Host Application Objects

11.1 General Information

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

Standard Objects:

- Application Object (see Anybus CompactCom 40 Software Design Guide)
- Application Data Object (see Anybus CompactCom 40 Software Design Guide)

Network Specific Objects:

- *POWERLINK Object (E9h), p. 116*
- *Application File System Interface Object (EAh), p. 118*
- *SYNC Object (EEh), p. 119*
- *Ethernet Host Object (F9h), p. 120*

11.2 POWERLINK Object (E9h)

Category

Extended

Object Description

This object implements POWERLINK features and IT functionality in the host application. The application can support none, some, or all attributes specified.

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	"POWERLINK"
2	Revision	Get	UINT8	01h
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.

#	Name	Access	Data Type	Value If Not Implemented	Comment
1	Vendor ID	Get	UINT32	0000001Bh	These values replace the default values for CANopen identity object (1018h, see Object Entries, p. 19) Note: Changing any of these attributes requires a new Vendor ID.
2	Product Code	Get	UINT32	00000028h	
3	Revision high word	Get	UINT16	0000h	
4	Revision low word	Get	UINT16	(hardware revision)	
5	Serial Number	Get	UINT32	(set at production)	
6	Manufacturer device name	Get	Array of CHAR (max 64 bytes)	"Anybus CompactCom 40 Ethernet POWERLINK"	Corresponds to Manufacturer Device Name object (1008h, see Object Entries, p. 19).
7	Manufacturer hardware version	Get	Array of CHAR (max 64 bytes)	Hardware version	Corresponds to Manufacturer hardware version object (1009h, see Object Entries, p. 19).
8	Manufacturer software version	Get	Array of CHAR (max 64 bytes)	Anybus CompactCom Software Version	Corresponds to Manufacturer software version object (100Ah, see Object Entries, p. 19).
10	Device type	Get	UINT32	00000000h	Corresponds to Device type object (1000h, see Object Entries, p. 19).
14	Manufacturer name	Get	Array of CHAR (max 64 bytes)	"HMS Industrial Networks"	Will be used as part of the Interface Description string in the Interface Group object (1030h, see Object Entries, p. 19).
15 - 16	(reserved)				

#	Name	Access	Data Type	Value If Not Implemented	Comment
17	Enable IT Functionality	Get	BOOL	True (= 1)	Enables/disables support for IT functionality (e.g. WEB, FTP, HICP, SMTP client, socket interface etc.). When disabled the module will only receive and transmit Ethernet frames of POWERLINK type.
18	(reserved)				
19	SDO/IT frame ratio	Get	UINT8	3	See below
20	(reserved)				

Attribute #19

Both SDO frames and IT frames are sent in the asynchronous phase. SDO frames are sent with higher priority than IT frames. To avoid that IT frames would never be sent, due to the SDO channel being fully used, an SDO/IT frame flow control is implemented. If an IT frame is queued and 3 (default value) SDO frames are sent without the IT frame being sent, the module will force the IT frame to be sent on the next unspecified invite from the managing node, even if another SDO frame is queued.

Attribute #19 (SDO/IT frame ratio) is by default 3, but can be set by the host application to influence how many SDO frames that can be sent before the IT frame is forced. Valid values are 0 - 15.

0 means that the flow control is disabled, meaning the frame with the highest priority is always sent. This can cause a complete interruption of IT functionality if the SDO frames take all bandwidth.

If a value higher than 15 is set in the attribute the module will use the value 15.

11.3 Application File System Interface Object (EAh)

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 allows the user to download software through the FTP server to the application. The application decides the available memory space.

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

11.4 SYNC Object (EEh)

Category

Extended

Object Description

This object contains the host application sync settings. The module will generate a sync pulse to the application if the following conditions are fulfilled: This object is implemented, sync mode is supported and the NMT state is NMT_CS_READY_TO_OPERATE, NMT_CS_PRE_OPERATIONAL_2, NMT_CS_OPERATIONAL or NMT_CS_STOPPED.

See also...

- [Synchronization, p. 14](#)
- Anybus CompactCom 40 Software Design Guide, "SYNC"
- Anybus CompactCom 40 Software Design Guide, "Error Codes"

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)

Extended

#	Name	Access	Type	Comment
1	Cycle time	Get/Set	UINT32	This attribute contains the cycle time on the Ethernet POWERLINK network. It corresponds to the value of the Object Dictionary Object #1006h (NMT Cycle Length Object), see Object Dictionary, p. 19 .
2 - 6	-	-	-	Not used
7	Sync mode	Get/Set	UINT16	Set to 1 (synchronous operation) when the POWERLINK network cycle is isochronous and bit 1 in attribute #8 is set; otherwise, set to 0 (nonsynchronous operation).
8	Supported sync modes	Get	UINT16	Bit 0: Nonsynchronous operation (default value if nonsynchronous operation is supported) Bit 1: Synchronous operation is supported Bit 2 - 15: Reserved. Set to zero.

11.5 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 (Not used if Transparent Ethernet is enabled.)
4	(reserved)				Reserved for Anybus CompactCom 30 applications.
5	Enable Web ADI access	Get	BOOL	True (Enabled)	Enable/Disable Web ADI access (Not used if Transparent Ethernet is enabled.)
6	Enable FTP server	Get	BOOL	True (Enabled)	Enable/Disable FTP server (Not used if Transparent Ethernet is enabled.)
7	Enable admin mode	Get	BOOL	False (Disabled)	Enable/Disable FTP admin mode (Not used if Transparent Ethernet is enabled.)
8	Network Status	Set	UINT16	-	See below.

#	Name	Access	Data Type	Default Value	Comment
9	Port 1 MAC address	Get	Array of UINT8	-	<p>Note: This attribute is only valid for PROFINET devices. 6 byte MAC address for port 1 (mandatory for the LLDP protocol).</p> <p>This setting overrides any Port MAC address in the host PROFINET IO Object.</p> <p>Do not implement this attribute if the preprogrammed Mac address is to be used.</p>
10	Port 2 MAC address	Get	Array of UINT8	-	<p>Note: This attribute is only valid for PROFINET devices. 6 byte MAC address for port 2 (mandatory for the LLDP protocol).</p> <p>This setting overrides any Port MAC address in the host PROFINET IO Object.</p> <p>Do not implement this attribute if the preprogrammed Mac address is to be used.</p>
11	Enable ACD	Get	BOOL	True (Enabled)	<p>Enable/Disable ACD protocol.</p> <p>If ACD functionality is disabled using this attribute, the ACD attributes in the CIP TCP/IP object (F5h) are not available.</p>
12	Port 1 State	Get	ENUM	0 (Enabled)	<p>The state of Ethernet port 1.</p> <ul style="list-style-type: none"> This attribute is not read by EtherCAT and Ethernet POWERLINK devices, where Port 1 is always enabled. This attribute is not used by PROFINET and Ethernet POWERLINK <p>00h: Enabled 01h: Disabled.</p> <p>The port is treated as existing. References to the port can exist, e.g. in network protocol or on website.</p>
13	Port 2 State	Get	ENUM	0 (Enabled)	<p>The state of Ethernet port 2.</p> <ul style="list-style-type: none"> This attribute is not read by EtherCAT and Ethernet POWERLINK devices, where Port 2 is always enabled. This attribute is not used by PROFINET <p>00h: Enabled 01h: Disabled.</p> <p>The port is treated as existing. References to the port can exist, e.g. in network protocol or on website.</p> <p>02h: Inactive.</p> <p>The attribute is set to this value for a device that only has one physical port. All two-port functionality is disabled. No references can be made to this port.</p> <p>Note: This functionality is available for Ethernet/IP and Modbus-TCP devices.</p>
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 standalone 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	Ethernet PHY Configuration	Get	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
20	SNMP read-only community string	Get	Array of CHAR	“public”	Note: This attribute is only valid for PROFINET devices. Sets the SNMP read-only community string. Max length is 32.
21	SNMP read-write community string	Get	Array of CHAR	“private”	Note: This attribute is only valid for PROFINET devices. Sets the SNMP read-write community string. Max length is 32.
22	DHCP Option 61 source	Get	ENUM	0 (Disabled)	Note: This attribute is currently only valid for Ethernet/IP devices. See below (DHCP Option 61, Client Identifier)
23	DHCP Option 61 generic string	Get	Array of UINT8	N/A	Note: This attribute is currently only valid for Ethernet/IP devices. See below (DHCP Option 61, Client Identifier)
24	Enable DHCP Client	Get	BOOL	1 = True	Note: This attribute is currently valid for Ethernet/IP and PROFINET devices. Enable/disable DHCP Client functionality 0: DHCP Client functionality is disabled 1: DHCP Client functionality is enabled

Network Status

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

Bit	Contents	Description	Comment
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	EtherCAT only: This link status indicates whether the Anybus CompactCom is able to communicate using Ethernet over EtherCAT (EoE) or not. That is, it indicates the status of the logical EoE port link and is not related to the link status on the physical EtherCAT ports.
4	Link port 2	Current link status for port 2 1 = Link sensed 0 = No link	Not used for EtherCAT
5... 15	(reserved)	(mask off and ignore)	

DHCP Option 61 (Client Identifier)



Only valid for EtherNet/IP devices

The DHCP Option 61 (Client Identifier) allow the end-user to specify a unique identifier, which has to be unique within the DHCP domain.

Attribute #22 (DHCP Option 61 source) is used to configure the source of the Client Identifier. The table below shows the definition for the Client identifier for different sources and their description.

Value	Source	Description
0	Disable	The DHCP Option 61 is disabled. This is the default value if the attribute is not implemented in the application.
1	MACID	The MACID will be used as the Client Identifier
2	Host Name	The configured Host Name will be used as the Client Identifier
3	Generic String	Attribute #23 will be used as the Client Identifier

Attribute #23 (DHCP Option 61 generic string) is used to set the Client Identifier when Attribute #22 has been set to 3 (Generic String). Attribute #23 contains the Type field and Client Identifier and shall comply with the definitions in RFC 2132. The allowed max length that can be passed to the module via attribute #23 is 64 octets.

Example:

If Attribute #22 has been set to 3 (Generic String) and Attribute #23 contains 0x01, 0x00, 0x30, 0x11, 0x33, 0x44, 0x55, the Client Identifier will be represented as an Ethernet Media Type with MACID 00:30:11:33:44:55.

Example 2:

If Attribute #22 has been set to 2 (Host Name) Attribute #23 will be ignored and the Client Identifier will be the same as the configured Host Name.

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 Implementation Details

B.1 SUP-Bit Definition

The supervised bit (SUP) in the status register indicates that the network participation is supervised by another network device.

On POWERLINK, a slave (Controlled Node, CN) has no real knowledge if it is supervised by another network device. It is considered supervised if it enters the NMT_CS_OPERATIONAL state where exchange of valid PDO data takes place. If the Managing Node (MN) of the network is lost, the device will leave the operational state and the supervised bit will be cleared

Supervised Bit State	Description
0	The module is not in NMT_CS_OPERATIONAL
1	The module is in NMT_CS_OPERATIONAL

B.2 Anybus State Machine

The table below describes how the Anybus state machine relates to the Ethernet POWERLINK network

Anybus State	POWERLINK NMT state(s)	Implementation	Comment
WAIT_PROCESS	NMT_GS_INITIALISATION (superstate) <ul style="list-style-type: none"> NMT_GS_INITIALISING NMT_GS_RESET_APPLICATION NMT_GS_RESET_COMMUNICATION NMT_GS_RESET_CONFIGURATION NMT_CS_NOT_ACTIVE NMT_CS_BASIC_ETHERNET NMT_CS_PRE_OPERATIONAL_1 (if no error) NMT_CS_PRE_OPERATIONAL_2 NMT_CS_STOPPED	The module stays in this state until the setup is finalized and the CN and the MN together initiates a transition to POWERLINK state NMT_CS_READY_TO_OPERATE, which equals Anybus state IDLE. If the MN orders the CN to stop, the POWERLINK state will switch to NMT_CS_STOPPED and the Anybus state will switch to WAIT_PROCESS.	In case synchronous mode is supported, sync signals will start being produced in this state. The response of a PReq (POWERLINK message), will be disabled for all these NMT states, except for PRE_OPERATIONAL_2.
ERROR		There is no designated ERROR state on POWERLINK.	Errors result in a transition to POWERLINK state NMT_CS_PRE_OPERATIONAL_1. A SoC frame, received from the network, will cause a transition from NMT_CS_PRE_OPERATIONAL_1 to the state NMT_CS_PRE_OPERATIONAL_2 (corresponding to Anybus state WAIT_PROCESS). Read process data should be regarded as not valid by the application.
PROCESS_ACTIVE	NMT_CS_OPERATIONAL (after reception of the first valid PDO data)	Corresponds to POWERLINK state NMT_CS_OPERATIONAL	Received PDO data is valid only in this state.
IDLE		POWERLINK state NMT_CS_READY_TO_OPERATE and NMT_CS_OPERATIONAL before receiving the first valid PDO data. Transition to PROCESS_ACTIVE can only be initiated by MN.	-
EXCEPTION		POWERLINK state NMT_GS_OFF. No network traffic is sent or received, though the hub continues to operate. Unexpected errors, e.g. watchdog timeouts, causes a transition to this state.	ERROR and STATUS LEDs turn red (to indicate a major fault). The module has to be reset to leave this state.

C Timing & Performance

C.1 General Information

This chapter specifies timing and performance parameters that are verified and documented for the Anybus CompactCom 40 Ethernet POWERLINK.

Category
Event Based Process Data Delay

For general timing information, see the Anybus CompactCom 40 Software Design Guide.

C.2 Event Based Process Data Delay

“Read process data delay” is defined as the time from when the last bit of the network frame enters the module, to when the RDPDI interrupt is asserted to the application.

“Write process data delay” is defined in two different ways, depending on network type.

- For software stack based cyclic/pollled networks, it is defined as the time from when the module exchanges write process data buffers, to when the first bit of the new process data frame is sent out on the network.
- For COS (Change Of State) networks, it is defined as the time from when the application exchanges write process data buffers, to when the first bit of the new process data frame is sent out on the network.

Parameter	Description	Measured Time	Unit	Comment
T101	Read process data delay	617	ns	Time from last input bit received until nIRQ goes active on the chip
T102	Write process data delay	2	μs	Time from when the Anybus CompactCom module exchanges write process data buffers, to when the first bit of the new process data frame is sent out on the network interface towards the PHY.

D Secure HICP (Secure Host IP Configuration Protocol)

D.1 General

The Anybus CompactCom 40 Ethernet POWERLINK 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.

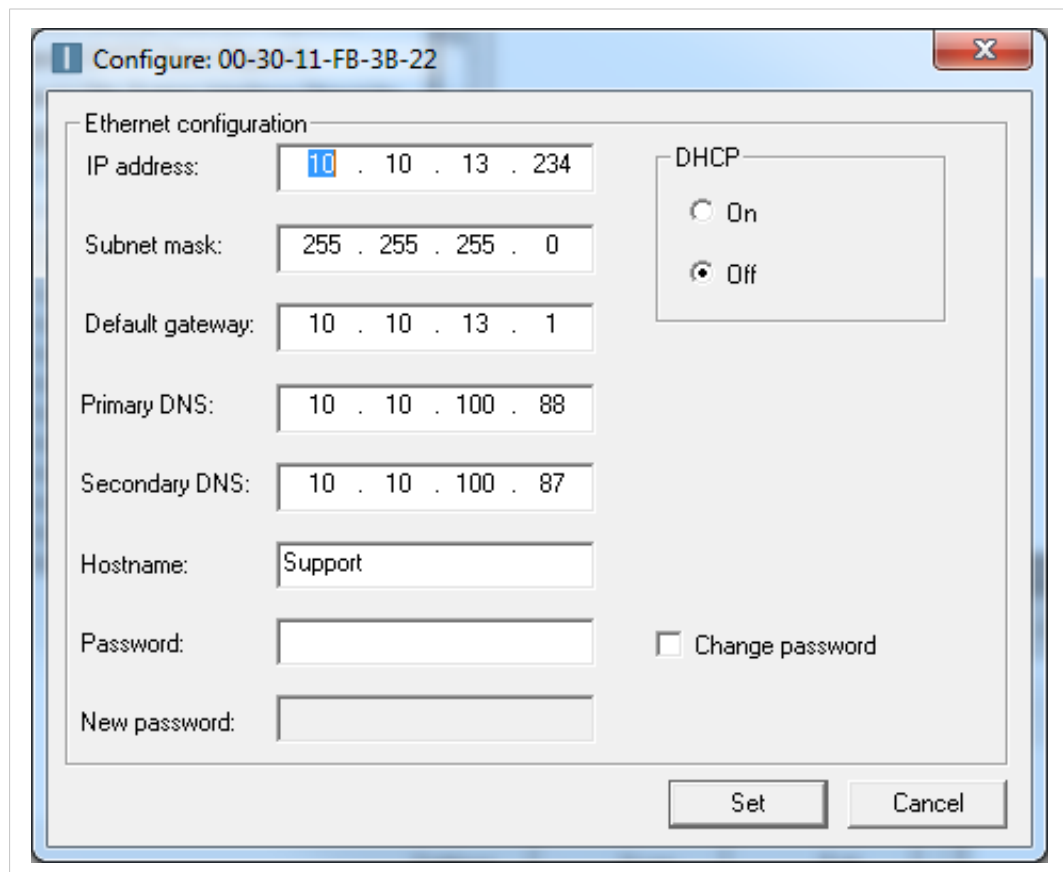


Fig. 6

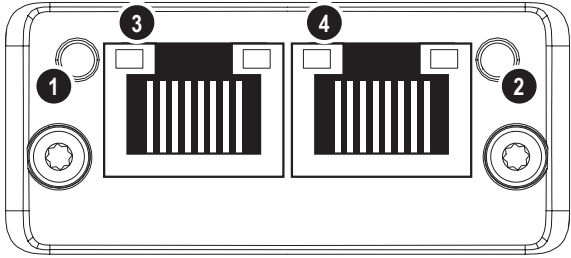
Validate the new settings by clicking **Set**, or click **Cancel** to cancel all changes. Optionally, the configuration can be protected from unauthorized access by a password. To enter a password, check the **Change password** checkbox and enter the password in the **New password** text field.

E Technical Specification

E.1 Front View

E.1.1 Ethernet Connectors

#	Item	Connector
1	STATUS LED	Ethernet, RJ45
2	ERROR 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 STATUS LED

LED State	Description
Off	Module is off, initializing, or not active.
Green, fast flashing (on 50 ms, off 50 ms)	NMT_CS_BASIC_ETHERNET Basic Ethernet state: no POWERLINK traffic has been detected.
Green, single flash	NMT_CS_PRE_OPERATIONAL_1. Only asynchronous data.
Green, double flash	NMT_CS_PRE_OPERATIONAL_2. Asynchronous and synchronous data. No PDO data. Any process data sent is declared not valid and received process data must be ignored in this state.
Green, triple flash	NMT_CS_READY_TO_OPERATE. Ready to operate. Asynchronous and synchronous data. No PDO data. Any process data sent is declared not valid and received process data must be ignored in this state.
Green	NMT_CS_OPERATIONAL. Fully operational. Asynchronous and synchronous data. PDO data is sent and received.
Green, slow flashing (on 200 ms, off 200 ms)	NMT_CS_STOPPED Module stopped (for controlled shutdown, for example). Asynchronous and synchronous data. No PDO data. Any process data sent is declared not valid and received process data must be ignored in this state.
Red	If the ERROR LED also is red, a fatal event was encountered.

E.1.3 ERROR LED

LED State	Description
Off	No error
Red	If the STATUS LED is not red, a non-fatal error has been detected. If the STATUS LED is red, a fatal event was encountered.

E.1.4 LINK/Activity LED 3/4

LED State	Description
Off	No link.
Green	Link, no traffic.
Green, flashing	Link and traffic.

E.1.5 Ethernet Interface

The Ethernet interface supports 100 Mbit/s, half duplex operation.

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*.

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

E.3 Power Supply

E.3.1 Supply Voltage

The Anybus CompactCom 40 Ethernet POWERLINK 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 Ethernet POWERLINK is designed to fulfil the requirements of a Class B module.

For more information about the power consumption classification used on the Anybus CompactCom 40 platform, consult the general *Anybus CompactCom 40 Hardware Design Guide*.

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.

F Copyright Notices

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.

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

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.

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.

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.

The "inih" library is distributed under the New BSD license:

Copyright (c) 2009, Ben Hoyt
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 Ben Hoyt 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 BEN HOYT "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 BEN HOYT 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

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.

Copyright (c) 2016 The MINIX 3 Project.

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.

Author: David van Moolenbroek <david@minix3.org>

This page intentionally left blank

