

Network Guide
Anybus[®] CompactCom 40 EtherNet/IP
Doc.Id. HMSI-27-212
Rev. 1.5



HALMSTAD • CHICAGO • KARLSRUHE • TOKYO • BEIJING • MILANO • MULHOUSE • COVENTRY • PUNE • COPENHAGEN

HMS Industrial Networks
Mailing address: Box 4126, 300 04 Halmstad, Sweden
Visiting address: Stationsgatan 37, Halmstad, Sweden

E-mail: info@hms-networks.com
Web: www.anybus.com

Important User Information

This document is intended to provide a good understanding of the functionality offered by EtherNet/IP. The document only describes the features that are specific to the Anybus CompactCom 40 EtherNet/IP. For general information regarding the 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 use of advanced EtherNet/IP-specific functionality may require in-depth knowledge in EtherNet/IP networking internals and/or information from the official EtherNet/IP specifications. In such cases, the people responsible for the implementation of this product should either obtain the EtherNet/IP specification to gain sufficient knowledge or limit their implementation in such a way that this is not necessary.

Liability

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

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

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

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

Intellectual Property Rights

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

Trademark Acknowledgements

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

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

Table of Contents

Preface	About This Document	
	Related Documents.....	8
	Document History	8
	Conventions & Terminology.....	9
	Abbreviations.....	9
	Support	9
Chapter 1	About the Anybus CompactCom 40 EtherNet/IP	
	General.....	10
	Features.....	10
	Beacon Based DLR (Device Level Ring)	11
Chapter 2	Basic Operation	
	General Information.....	12
	<i>Software Requirements</i>	12
	Device Customization	13
	<i>Network Identity</i>	13
	<i>Electronic Data Sheet (EDS)</i>	13
	<i>EtherNet/IP & CIP Implementation</i>	14
	<i>Web Interface</i>	14
	<i>Socket Interface (Advanced Users Only)</i>	14
	<i>Modular Device Functionality</i>	15
	<i>QuickConnect</i>	15
	<i>CIP Safety</i>	15
	Communication Settings	16
	<i>Communication Settings in Stand Alone Shift Register Mode</i>	17
	Diagnostics	18
	Network Data Exchange.....	19
	<i>Application Data</i>	19
	<i>Process Data</i>	19
	<i>Translation of Data Types</i>	19
	File System.....	20
	<i>Overview</i>	20
	<i>General Information</i>	21
	<i>System Files</i>	21

Chapter 3	FTP Server	
	General Information.....	22
	User Accounts	22
	Session Example.....	23
Chapter 4	Web Server	
	General Information.....	24
	Default Web Pages.....	24
	<i>Network Configuration</i>	25
	<i>Ethernet statistics page</i>	27
	Server Configuration.....	29
	<i>General Information</i>	29
	<i>Index Page</i>	29
	<i>Default Content Types</i>	30
	<i>Authorization</i>	30
Chapter 5	E-mail Client	
	General Information.....	32
	How to Send E-mail Messages.....	32
Chapter 6	Server Side Include (SSI)	
	General Information.....	33
	Include File.....	33
	Command Functions	34
	<i>General Information</i>	34
	<i>GetConfigItem()</i>	35
	<i>SetConfigItem()</i>	36
	<i>SsiOutput()</i>	38
	<i>DisplayRemoteUser</i>	38
	<i>ChangeLanguage()</i>	39
	<i>IncludeFile()</i>	40
	<i>SaveDataToFile()</i>	41
	<i>printf()</i>	42
	<i>scanf()</i>	44
	Argument Functions.....	46
	<i>General Information</i>	46
	<i>ABCCMessage()</i>	46
	SSI Output Configuration.....	50

Chapter 7	JSON	
	General Information.....	51
	JSON Objects	51
	<i>ADI</i>	51
	<i>Module</i>	54
	<i>Network</i>	55
	<i>Services</i>	59
	<i>Hex Format Explained</i>	59
	Example.....	60
Chapter 8	CIP Objects	
	General Information.....	61
	Translation of Status Codes.....	62
	Identity Object (01h)	63
	Message Router (02h).....	66
	Assembly Object (04h).....	67
	Connection Manager (06h).....	70
	Parameter Object (0Fh).....	74
	DLR Object (47h).....	77
	QoS Object (48h).....	78
	Base Energy Object (4Eh).....	79
	Power Management Object (53h).....	81
	ADI Object (A2h).....	83
	Port Object (F4h).....	85
	TCP/IP Interface Object (F5h).....	87
	Ethernet Link Object (F6h).....	90

Chapter 9	Anybus Module Objects	
	General Information.....	95
	Anybus Object (01h).....	96
	Diagnostic Object (02h).....	97
	Network Object (03h).....	98
	Network Configuration Object (04h).....	99
	Socket Interface Object (07h).....	109
	SMTP Client Object (09h).....	126
	Anybus File System Interface Object (0Ah).....	131
	<i>Examples</i>	144
	Network Ethernet Object (0Ch).....	148
	Functional Safety Module Object (11h).....	149
	CIP Port Configuration Object (0Dh).....	153
Chapter 10	Host Application Objects	
	General Information.....	155
	Functional Safety Host Object (E8h).....	156
	CIP Identity Host Object (EDh).....	158
	Sync Object (EEh).....	160
	EtherNet/IP Host Object (F8h).....	161
	Ethernet Host Object (F9h).....	171
	Application File System Interface Object (EAh).....	174
Appendix A	Categorization of Functionality	
	Basic.....	187
	Extended.....	187
Appendix B	Implementation Details	
	SUP-Bit Definition.....	188
	Anybus Statemachine.....	188
	Application Watchdog Timeout Handling.....	188
Appendix C	Message Segmentation	
	General.....	189
	Command Segmentation.....	190
	Response Segmentation.....	191

Appendix D	Secure HICP (Secure Host IP Configuration Protocol)	
	General.....	192
Appendix E	Technical Specification	
	Front View	193
	Protective Earth (PE) Requirements.....	194
	Power Supply	194
	Environmental Specification	194
	EMC Compliance.....	194
Appendix F	Timing & Performance	
	General Information.....	195
	Internal Timing.....	195
	<i>Startup Delay</i>	195
	<i>NW_INIT Handling</i>	195
	<i>Event Based WrMsg Busy Time</i>	196
	<i>Event Based Process Data Delay</i>	196
Appendix G	Copyright Notice	

P. About This Document

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

P.1 Related Documents

Document	Author
Anybus CompactCom 40 Software Design Guide	HMS
Anybus CompactCom M40 Hardware Design Guide	HMS
Anybus CompactCom B40 Hardware Design Guide	HMS
CIP specification, Volumes 1 (CIP Common) and 2 (EtherNet/IP)	ODVA

P.2 Document History

Summary of Recent Changes (1.4... 1.5)

Change	Page(s)
Added information about safety to feature section	10
Added CIP Safety section to Basic Operation chapter	15
Added Class 0 Connection details to the Connection Manager (06h)	71
Added Functional Safety Module Object (11h)	149
Added Functional Safety Host Object (E8h)	156

Revision List

Revision	Date	Author(s)	Chapter(s)	Description
1.00	2014-06-04	KeL	All	First official revision
1.10	2014-07-17	KeL	2, 6, 7, 8, E	Misc. updates
1.11	2014-08-27	KaD	2	Major update
1.20	2015-01-09	KeL	7, 8, 9, F	Misc. updates
1.30	2015-11-04	KeL	2, 7, 8, 9, 10, 7, E	Misc. updates and corrections
1.4	2016-04-04	KaD	1,7, 8, 10	Misc. updates
1.5	2016-04-22	KaD	1, 2, 9, 10	Safety updates

P.3 Conventions & Terminology

The following conventions are used throughout this manual:

- Numbered lists provide sequential steps
- Bulleted lists provide information, not procedural steps
- The terms ‘Anybus’ or ‘module’ refers to the Anybus CompactCom 40 module.
- The terms ‘host’ or ‘host application’ refers to the device that hosts the Anybus module.
- Hexadecimal values are either written in the format NNNNh or the format 0xNNNN, where NNNN is the hexadecimal value.

P.4 Abbreviations

Abbreviation	Meaning
API	assigned packet interval
RPI	requested packet interval
T	target (in this case the module)
O	origin (in this case the master)

P.5 Support

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

1. About the Anybus CompactCom 40 EtherNet/IP

1.1 General

The Anybus CompactCom 40 EtherNet/IP communication module provides instant Ethernet and EtherNet/IP connectivity via the patented Anybus CompactCom host interface. Any device that supports this standard can take advantage of the features offered by the module, allowing seamless network integration regardless of network type. The module supports both linear and ring network topology (DLR, Device Level Ring).

The modular approach of the Anybus CompactCom 40 platform allows the CIP-object implementation to be extended to fit specific application requirements. Furthermore, the Identity Object can be customized, allowing the end product to appear as a vendor-specific implementation rather than a generic Anybus module.

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.

1.2 Features

- Two EtherNet/IP ports
- Ethernet RJ45 connectors
- Beacon Based DLR (Device Level Ring) and linear network topology supported
- Black channel interface, offering a transparent channel supporting Functional Safety up to SIL3 with separate safety module¹
- 10/100 Mbit, full/half duplex operation
- Web server w. customizable content
- FTP server
- Email client
- Server Side Include (SSI) functionality
- JSON functionality
- Customizable Identity Information
- Up to 65535 ADIs
- CIP Parameter Object support
- Expandable CIP-object implementation
- Supports unconnected CIP routing
- Transparent Socket Interface
- Modular Device functionality
- QuickConnect supported
- Multiple IO assembly instances can be created

1. IXXAT Safe T100 recommended

1.3 Beacon Based DLR (Device Level Ring)

Device Level Ring (DLR) is a network technology for industrial applications that uses embedded switch functionality in automation end devices, such as programmable automation controllers and I/O modules, to enable Ethernet ring network topologies at the device level. DLR technology adds network resilience to optimize machine operation.

Beacon based DLR networks consist of a ring supervisor and a number of ring nodes, and use “beacons” to detect breaks in the ring. When a DLR network detects a break in the ring, it provides ways to alternatively route the data to recover the network. Diagnostics built into DLR products can identify the point of failure, thus helping to speed maintenance and reduce repair time.

The Anybus CompactCom 40 EtherNet/IP implements the DLR protocol, and it is enabled by default. The device is able to process and act on beacon frames sent by ring supervisors, and supports beacon rates down to 100 μ s.

If needed, the DLR functionality can be disabled. This can be done by setting attribute 31 (Enable DLR) in the EtherNet/IP Host Object to False. See “Instance Attributes (Instance #1)” on page 162.

2. Basic Operation

2.1 General Information

2.1.1 Software Requirements

Generally, no additional network support code needs to be written in order to support the Anybus CompactCom 40 EtherNet/IP. However, due to the nature of the EtherNet/IP networking system, certain restrictions must be taken into account:

- Certain functionality in the module requires that the command ‘Get_Instance_Number_By_Order’ (Application Data Object, FEh) is implemented in the host application.
- Up to 5 diagnostic instances (See “Diagnostic Object (02h)” on page 97) can be created by the host application during normal conditions. An additional 6th instance may be created in event of a major fault.¹
- EtherNet/IP in itself does not impose any specific timing demands when it comes to acyclic requests (i.e. requests towards instances in the Application Data Object), however it is generally recommended to process and respond to such requests within a reasonable time period. The application that sends the request, also decides the timeout, e.g. EIPScan employs a timeout of 10 seconds.
- The use of advanced CIP-specific functionality may require in-depth knowledge in CIP networking internals and/or information from the official CIP and EtherNet/IP specifications. In such cases, the people responsible for the implementation of this product is expected either to obtain these specifications to gain sufficient knowledge or limit their implementation in such a way that this is not necessary.

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

See also...

- “Diagnostic Object (02h)” on page 97 (Anybus Module Objects)
- Anybus CompactCom 40 Software Design Guide, “Application Data Object (FEh)”

1. This limit is set by the module, not by the network.

2.2 Device Customization

2.2.1 Network Identity

By default, the module uses the following identity settings:

- Vendor ID: 005Ah (HMS Industrial Networks)
- Device Type: 002Bh (Generic Device)
- Product Code: 0037h (Anybus CompactCom 40 EtherNet/IP)
- Product Name: ‘Anybus CompactCom 40 EtherNet/IP(TM)’

Optionally, it is possible to customize the identity of the module by implementing the corresponding instance attributes in the EtherNet/IP Host Object.

See also...

- “Identity Object (01h)” on page 63 (CIP-object)
- “EtherNet/IP Host Object (F8h)” on page 161 (Host Application Object)

IMPORTANT: *According to the CIP specification, the combination of Vendor ID and serial number must be unique. It is not permitted to use a custom serial number in combination with the HMS Vendor ID (005Ah), nor is it permitted to choose Vendor ID arbitrarily. Failure to comply to this requirement will induce interoperability problems and/or other unwanted side effects. HMS approves use of the HMS Vendor ID (005Ah), in combination with the default serial number, under the condition that the implementation requires no deviations from the standard EDS-file.*

To obtain a Vendor ID, contact the ODVA.

2.2.2 Electronic Data Sheet (EDS)

On EtherNet/IP, the characteristics of a device is stored in an ASCII data file with the suffix EDS. This file is used by configuration tools etc. when setting up the network configuration. HMS supplies a standard (generic) EDS-file, which corresponds to the default settings in the module. However, due to the flexible nature of the Anybus CompactCom concept, it is possible to alter the behavior of the product in ways which invalidate the generic EDS-file. In such case, a custom EDS-file needs to be created, which in turn invalidates the default identity information and require re-certification of the product.

Note: Since the module implements the Parameter Object, it is possible for configuration tools such as RSNetWorx to automatically generate a suitable EDS-file. Note that this functionality requires that the command ‘Get_Instance_Number_By_Order’ (Application Data Object, FEh) has been implemented in the host application.

See also...

- “Parameter Object (0Fh)” on page 74 (CIP-object)
- Anybus CompactCom 40 Software Design Guide, “Application Data Object (FEh)”

IMPORTANT: *HMS approves use of the standard EDS-file only under the condition that it matches the actual implementation and that the identity information remains unchanged.*

2.2.3 EtherNet/IP & CIP Implementation

By default, the module supports the generic CIP profile. Optionally, it is possible to re-route requests to unimplemented CIP objects to the host application, thus enabling support for other profiles etc.

To support a specific profile, perform the following steps:

- Set up the identity settings in the EtherNet/IP Host Object according to profile requirements.
- Implement the Assembly Mapping Object in the host application.
- Set up the Assembly Instance Numbers according to profile requirements.
- Enable routing of CIP messages to the host application in the EtherNet/IP Host Object.
- Implement the required CIP objects in the host application.

See also...

- “EtherNet/IP Host Object (F8h)” on page 161 (Host Application Object)
- “Command Details: Process_CIP_Object_Request” on page 166

2.2.4 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.

See also...

- “File System” on page 20
- “FTP Server” on page 22
- “Web Server” on page 24
- “Server Side Include (SSI)” on page 33
- “JSON” on page 51

2.2.5 Socket Interface (Advanced Users Only)

The built in socket interface allows additional protocols to be implemented on top of TCP/IP.

See also...

- “Socket Interface Object (07h)” on page 109 (Anybus Module Object)
- “Message Segmentation” on page 189

2.2.6 Modular Device Functionality

Modular devices consist of a backplane with a certain number of “slots”. The first slot is occupied by the “coupler” which contains the Anybus CompactCom module. All other slots may be empty or occupied by modules.

When mapping ADIs to process data the application shall map the process data of each module in slot order.

A list of modules in a Modular Device is available to the EtherNet/IP network master by a request to the CIP Identity object.

See also ...

- “Modular Device Object (ECh)” (see Anybus CompactCom 40 Software Design Guide)
- “Identity Object (01h)” on page 63

2.2.7 QuickConnect

The module supports the QuickConnect functionality. It is enabled in the EtherNet/IP Host Object. The module fulfills Class A with a startup time of less than 180 ms, with 16 bytes of I/O data mapped with parallel, SPI or shift register application interface.

See also ...

- “EtherNet/IP Host Object (F8h)” on page 161
- “TCP/IP Interface Object (F5h)” on page 87 (CIP object)

2.2.8 CIP Safety

The Anybus CompactCom 40 EtherNet/IP device supports the CIP safety profile. This profile makes it possible for a user to send data on a black channel interface, i.e. a safe channel over EtherNet/IP using an add-on safety module, e.g. the IXXAT Safe T100. For an application to support CIP safety, the Functional Safety Object (E8h) has to be implemented.

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

See “Functional Safety Host Object (E8h)” on page 156.

2.3 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...

- **TCP/IP settings**

These settings must be set properly in order for the module to be able to participate on the network.

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

- **Physical Link Settings**

By default, the module uses auto negotiation to establish the physical link settings, however it is possible to force a specific setting if necessary.

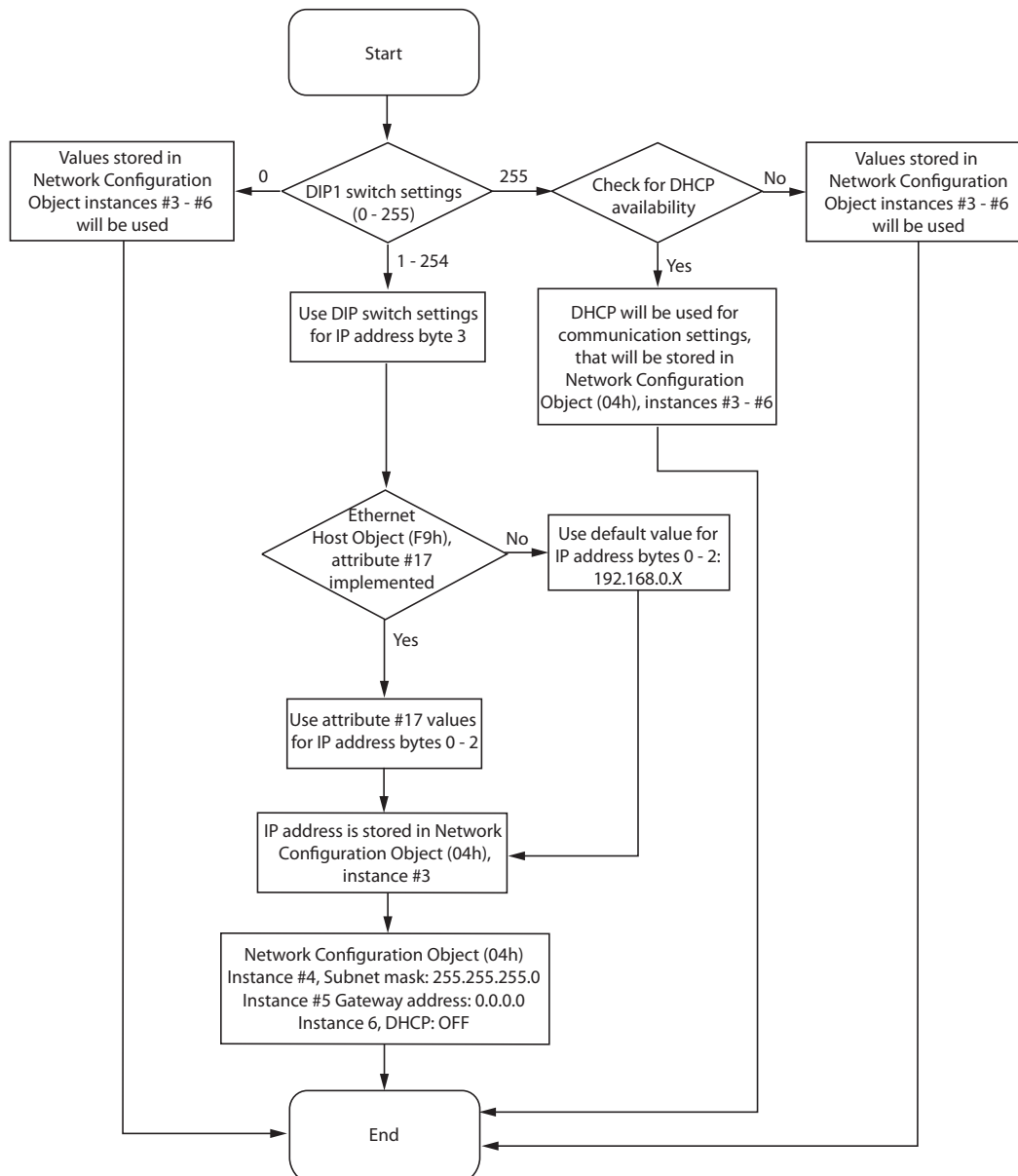
The parameters in the Network Configuration Object (04h) are available from the network through the built in web server, and through the TCP/IP Interface Object (CIP).

See also...

- “Web Server” on page 24
- “TCP/IP Interface Object (F5h)” on page 87 (CIP)
- “Ethernet Link Object (F6h)” on page 90 (CIP)
- “Network Configuration Object (04h)” on page 99 (Anybus Module Object)
- “Secure HICP (Secure Host IP Configuration Protocol)” on page 192

2.3.1 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.



See also...

- “Ethernet Host Object (F9h)” on page 171
- Anybus CompactCom M40 Hardware Design Guide
- “Network Configuration Object (04h)” on page 99

2.4 Diagnostics

The severity value of all pending events are combined (using logical OR) and copied to the corresponding bits in the 'Status'-attribute of the Identity Object (CIP).

See also...

- “Identity Object (01h)” on page 63 (CIP)
- “Diagnostic Object (02h)” on page 97 (Anybus Module Object)

2.5 Network Data Exchange

2.5.1 Application Data

Application Data Instances (ADIs) are represented through the ADI Object (CIP). Each instance within this objects corresponds directly to an instance in the Application Data Object on the host application side.

Accessible range of ADIs is 1 to 65535.

See also...

- “Parameter Object (0Fh)” on page 74 (CIP)
- “ADI Object (A2h)” on page 83 (CIP)

2.5.2 Process Data

Process Data is represented as dedicated instances in the Assembly Object (CIP).

See also...

- “Assembly Object (04h)” on page 67 (CIP)
- “Connection Manager (06h)” on page 70 (CIP)

2.5.3 Translation of Data Types

The Anybus data types are translated to CIP-standard and vice versa as follows:

Anybus Data Type	CIP Data Type	Comments
BOOL	BOOL	Each ADI element of this type occupies one byte.
ENUM	USINT	
SINT8	SINT	
UINT8	USINT	
SINT16	INT	Each ADI element of this type occupies two bytes.
UINT16	UINT	
SINT32	DINT	Each ADI element of this type occupies four bytes.
UINT32	UDINT	
FLOAT	REAL	
CHAR	SHORT_STRING	SHORT_STRING consists of a single-byte length field (which in this case represents the number of ADI elements) followed by the actual character data (in this case the actual ADI elements). This means that a 10-character string occupies 11 bytes.
SINT64	LINT	Each ADI element of this type occupies eight bytes.
UINT64	ULINT	
BITS8	BYTE	Each ADI element of this type occupies one byte.
BITS16	WORD	Each ADI element of this type occupies two bytes.
BITS32	DWORD	Each ADI element of this type occupies four bytes.
OCTET	USINT	
BITS1-7	BYTE	Bit fields of size 1 - 7
PAD0-8	BYTE	Bit fields of size 0 - 8 used for padding
PAD9-16	BYTE	Bit fields of size 9 - 16 used for padding

2.6 File System

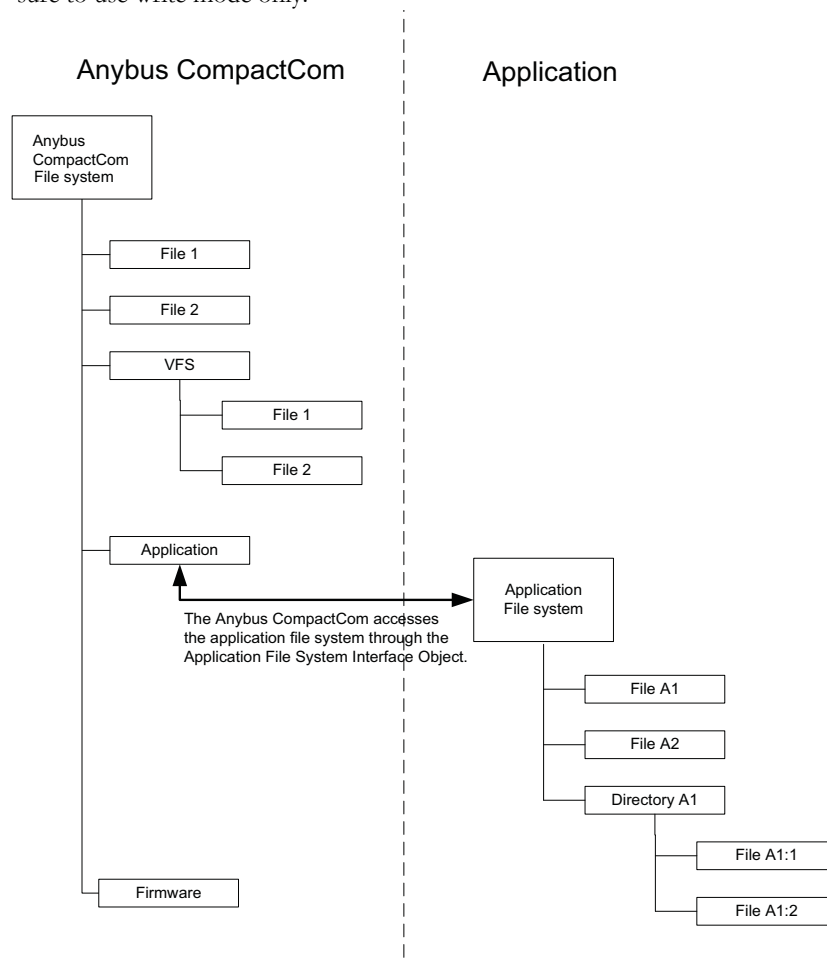
Category: Extended

2.6.1 Overview

The Anybus CompactCom 40 EtherNet/IP has an in-built 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 - Firmware updates are stored in this directory.

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



2.6.2 General Information

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

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” on page 22
- “Web Server” on page 24
- “E-mail Client” on page 32
- “Server Side Include (SSI)” on page 33
- “Anybus File System Interface Object (0Ah)” on page 131
- “Application File System Interface Object (EAh)” on page 174

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

The following operations will erase one or more flash segments:

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

2.6.3 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 WindowsTM). The format of these files are, with a few exceptions, based on the concept of ‘keys’, where each ‘key’ can be assigned a value, see below.

Example

```
[Key1]
value of Key1
```

```
[Key2]
value of Key2
```

3. FTP Server

3.1 General Information

Category: extended

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), see page 171.

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.

3.2 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:

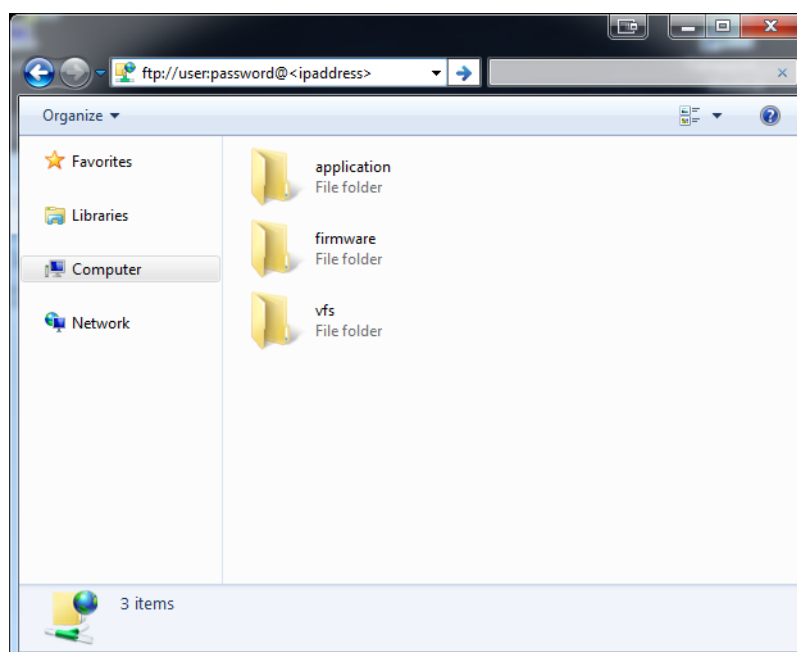
- Usernames must not exceed 16 characters in length.
- Passwords must not exceed 16 characters in length.
- All printable characters, except the separator ':', are allowed in usernames and passwords.

- If '\ftp.cfg' is missing or cannot be interpreted, all username/password combinations will be accepted and the home directory will be the system root (i.e. '\').
- The home directory for a user must also exist in the file system if they should be able to log in, just adding the user information to the 'ftp.cfg' file it is not enough.
- If 'Admin Mode' has been enabled in the Ethernet Object, all username/password combinations will be accepted and the user will have unrestricted access to the file system (i.e. the home directory will be the system root)¹.
- 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').

3.3 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.



1. Apart from the vfs folder, that is read-only.

4. Web Server

4.1 General Information

Category: extended

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

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

See also...

- “FTP Server” on page 22
- “Server Side Include (SSI)” on page 33
- “JSON” on page 51
- “Ethernet Host Object (F9h)” on page 171

4.2 Default Web Pages

The default web pages provide access to:

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

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

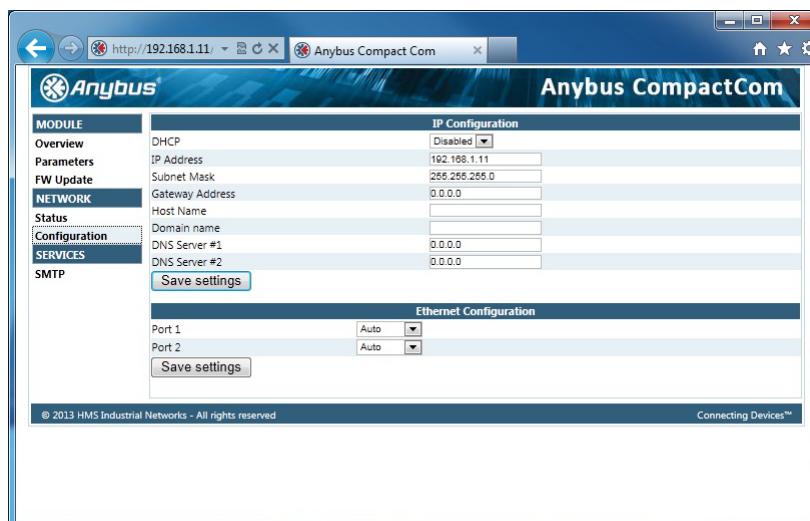
If a complete customized web system is designed and no files in the vfs folder are to be used, it is recommended to turn off the virtual file system completely, see the File System Interface Object.

See also...

- “File System” on page 20
- “Anybus File System Interface Object (0Ah)” on page 131

4.2.1 Network Configuration

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



The screenshot shows the Anybus CompactCom web interface. The browser address bar displays `http://192.168.1.11`. The page title is "Anybus CompactCom". On the left, a navigation menu lists "MODULE", "Overview", "Parameters", "FW Update", "NETWORK", "Status", "Configuration", "SERVICES", and "SMTP". The "Configuration" section is expanded, showing "IP Configuration" and "Ethernet Configuration".

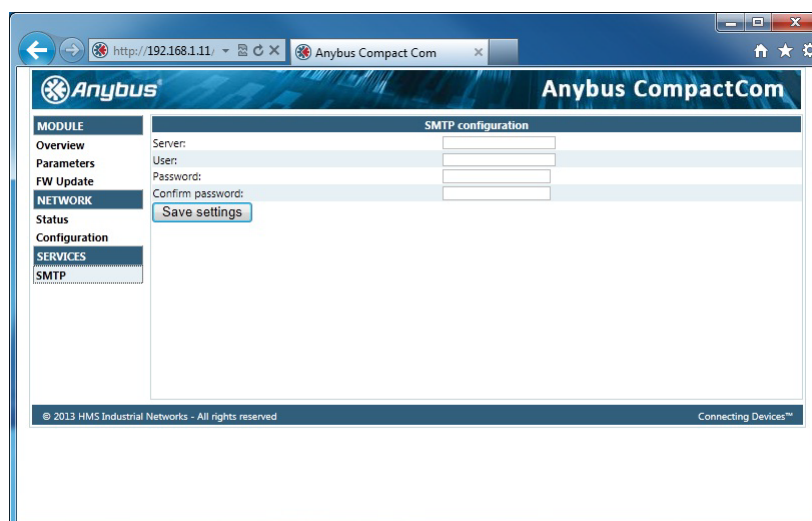
IP Configuration	
DHCP	Disabled
IP Address	192.168.1.11
Subnet Mask	255.255.255.0
Gateway Address	0.0.0.0
Host Name	
Domain name	
DNS Server #1	0.0.0.0
DNS Server #2	0.0.0.0

Save settings

Ethernet Configuration	
Port 1	Auto
Port 2	Auto

Save settings

© 2013 HMS Industrial Networks - All rights reserved Connecting Devices™



The screenshot shows the Anybus CompactCom web interface. The browser address bar displays `http://192.168.1.11`. The page title is "Anybus CompactCom". On the left, a navigation menu lists "MODULE", "Overview", "Parameters", "FW Update", "NETWORK", "Status", "Configuration", "SERVICES", and "SMTP". The "SMTP" section is expanded, showing "SMTP configuration".

SMTP configuration	
Server:	
User:	
Password:	
Confirm password:	

Save settings

© 2013 HMS Industrial Networks - All rights reserved Connecting Devices™

The module needs to be reset for the TCP/IP and SMTP settings to take effect. The Ethernet Configuration settings will take effect immediately.

Available editable settings will be explained on the next page.

IP Configuration

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

Name	Description
DHCP	Enable or disable DHCP Default value: enabled
IP address	The TCP/IP settings of the module
Subnet mask	Default values: 0.0.0.0
Gateway	Value ranges: 0.0.0.0 - 255.255.255.255
Host name	IP address or name Max 64 characters
Domain name	IP address or name Max 48 characters
DNS 1	Primary and secondary DNS server, used to resolve host name
DNS 2	Default values: 0.0.0.0 Value ranges: 0.0.0.0 - 255.255.255.255

Ethernet Configuration

Changes will take effect immediately.

Name	Description
Port 1	Ethernet speed/duplex settings
Port 2	Default value: auto

SMTP Settings

The module needs a reset before any changes take effect

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

4.2.2 Ethernet statistics page

The Ethernet statistics web page contains the following information:

Ethernet Link		Description
Port 1	Speed:	The current link speed.
	Duplex:	The current duplex configuration.
Port 2	Speed:	The current link speed.
	Duplex:	The current duplex configuration.

Ethernet/IP Statistics	Description
Established Class1 Connections	Current number of established class1 connections
Established Class3 Connections	Current number of established class3 connections
Connection Open Requests	Number of received connection open requests
Connection Open Format Rejects	Connection open requests rejected due to request format error
Connection Open Resource Rejects	Connection open requests rejected due to lack of resources
Connection Open Other Rejects	Connection open requests rejected due to other reasons
Connection Close Requests	Number of received connection open requests
Connection Close Format Rejects	Connection close requests rejected due to request format error
Connection Close Other Rejects	Connection close requests rejected due to other reasons
Connection Timeouts	Number of connection timeouts

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

Media Counters	Description
Alignment Errors	Frames received that are not an integral number of octets in length.
FCS Errors	Frames received that do not pass the FCS check.
Single Collisions	Successfully transmitted frames which experienced exactly one collision.
Multiple Collisions	Successfully transmitted frames which experienced more than one collision.
SQE Test Errors	Number of times SQE test error messages are generated. ^a
Deferred Transmissions	Frames for which first transmission attempt is delayed because the medium is busy.
Late Collisions	Number of times a collision is detected later than 512 bit-times into the transmission of a packet.
Excessive Collisions	Frames for which a transmission fails due to excessive collisions.
MAC Receive Errors	Frames for which reception of an interface fails due to an internal MAC sublayer receive error.
MAC Transmit Errors	Frames for which transmission fails due to an internal MAC sub-layer receive error.

Media Counters	Description
Carrier Sense Errors	Times that the carrier sense condition was lost or never asserted when attempted to transmit a frame.
Frame Size Too Long	Frames received that exceed the maximum permitted frame size.
Frame Size Too Short	Frames received that are shorter than lowest permitted frame size.

a. Not provided with current PHY interface.

4.3 Server Configuration

4.3.1 General Information

Category: extended

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

File Format:

```
[WebRoot]
\web

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

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

Web Root Directory

The web server cannot access files outside this directory.

Content Types

A list of file extensions and their reported content types.

See also...

- “Default Content Types” on page 30

SSI File Types

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

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

4.3.2 Index Page

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

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

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

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

See also...

- “Default Web Pages” on page 24

4.3.3 Default Content Types

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

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

Content types can be added or redefined by adding them to the server configuration file, see “General Information” on page 29.

4.3.4 Authorization

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

File Format:

```

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

```

}
 List of approved users.

```

[AuthName]
(message goes here)

```

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

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

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

Example:

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

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

```
[AuthType]
Basic
```

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

The field 'AuthType' is used to identify the authentication scheme.

Value	Description
Basic	Web authentication method using plain-text passwords.
Digest	More secure method using challenge-response authentication. Used as default if no [AuthType] field is specified.

5. E-mail Client

5.1 General Information

Category: extended

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

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

See also...

- "Network Configuration Object (04h)" on page 99
- "SMTP Client Object (09h)" on page 126

5.2 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 user name
- 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)

Note: See "SMTP Client Object (09h)" on page 126 for more information.

Sending a message based on a file in the file system is achieved using the "Send Email from File"-command. For a description of the file format, see "Command Details: Send Email From File" on page 129.

6. Server Side Include (SSI)

6.1 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.¹

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

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

6.2 Include File

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

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

Syntax:

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

filename-Source file

Default Output:

Scenario	Default Output
Success	(contents of file with any SSI tags replaced by their respective output)
Failure (e.g. file not found)	Nothing, i.e. the SSI tag is replaced by an empty string.

1. JSON offers more functionality when it comes to web pages, but is also more complex to use, see "JSON" on page 51.

6.3 Command Functions

6.3.1 General Information

Command functions executes commands and includes the result.

General Syntax:

```
<?--#exec cmd_argument='command'-->
command-Command function, see below.
```

Command Functions:

Command	Valid for Email Messages	Page
GetConfigItem()	Yes	35
SetConfigItem()	No	36
SsiOutput()	Yes	38
DisplayRemoteUser	No	38
ChangeLanguage()	No	39
IncludeFile()	Yes	40
SaveDataToFile()	No	41
printf()	Yes	42
scanf()	No	44

6.3.2 GetConfigItem()

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

File Format:

The source file must have the following format:

```
[key1]
value1

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

Syntax:

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

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

Default Output:

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

Example:

The following SSI...

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

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

```
[Apple]
Green

[Lemon]
Yellow

[Banana]
Blue
```

... returns the string Yellow.

6.3.3 SetConfigItem()

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

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

Syntax:

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

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

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

File Format:

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

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

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

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

...

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

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

Default Output:

Scenario	Default Output
Success	"Configuration stored to "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
```

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

6.3.4 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” on page 50

6.3.5 DisplayRemoteUser

This command stores returns the user name on an authentication session.

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

Syntax:

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

Default Output:

Scenario	Default Output
Success	(current user)

6.3.6 ChangeLanguage()

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

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

Syntax:

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

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

The passed value must be a single digit as follows:

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

Default Output:

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

Example:

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

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

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

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

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

</BODY>
</HTML>
```

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

6.3.7 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	(file contents)
Authentication Error	"Authentication error"
File open error	"Failed to open file "filename" "

Example:

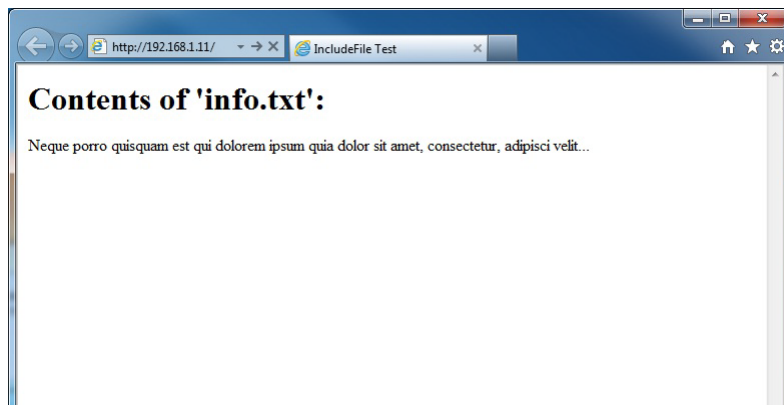
The following example demonstrates how to use this function.

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

Contents of 'info.txt':

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

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



See also...

- "Include File" on page 33

6.3.8 SaveDataToFile()

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

Note: This command cannot be used in email 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 behaviour is to store data from all form objects except the ones where the name starts with underscore ('_').
Overwrite Append	-Specifies whether to overwrite or append data to existing files.

Default Output:

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

Example:

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

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

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

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

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

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

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

</BODY>
</HTML>
```

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

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

6.3.9 printf()

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

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

Syntax:

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

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

Formatting tags are written as follows:

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

See also...

- “Formatting Tags” on page 43

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

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

See also...

- “ABCCMessage()” on page 46

Default Output:

Scenario	Default Output
Success	(printf() result)
ABCCMessage error	ABCCMessage error string (“Errors” on page 49)

Example:

See also...

- “ABCCMessage()” on page 46
- “Example (Get_Attribute):” on page 48

Formatting Tags

- **Type (Required)**

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

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

- **Flags (Optional)**

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

- **Width (Optional)**

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

- **.Precision (Optional)**

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

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

- **Modifier**

Modifier Character	Meaning
hh	Argument is interpreted as SINT8 or UIN8
h	Argument is interpreted as SINT16 or UINT16
L	Argument is interpreted as SINT32 or UINT32

6.3.10 scanf()

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

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

Note: This command cannot be used in email messages.

Syntax:

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

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

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

Formatting tags are written as follows:

```
[%[*] [width] [modifier] type
```

See also...

- “Formatting Tags” on page 45

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

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

See also...

- “ABCCMessage()” on page 46

Default Output:

Scenario	Default Output
Success	“Success”
Parsing error	“Incorrect data format ”
Too much data for argument	“Too much data ”
ABCC Message error	ABCCMessage error string (“Errors” on page 49)

Example:

See also...

- “ABCCMessage()” on page 46
- “Example (Set_Attribute):” on page 48

Formatting Tags

- **Type (Required)**

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

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

- *** (Optional)**

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

- **Width (Optional)**

Specifies the maximum number of characters to be read.

- **Modifier (Optional)**

Specifies a different data size.

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

6.4 Argument Functions

6.4.1 General Information

Argument functions are supplied as parameters to certain command functions.

General Syntax:

(Syntax depends on context)

Argument Functions:

Function	Description	Page
ABCCMessage()	-	46

6.4.2 ABCCMessage()

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

Syntax:

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

- object - Specifies the Destination Object
- instance - Specifies the Destination Instance
- command - Specifies the Command Number
- ce0 - Specifies CmdExt[0] for the command message
- ce1 - Specifies CmdExt[1] for the command message
- msgdata - Specifies the actual contents of the MsgData[] subfield in the command
 - Data can be supplied in direct form (format depends on c_type)
 - The keyword “ARG” is used when data is supplied by the parent command (e.g. scanf()).
- c_type - Specifies the data type in the command (msgdata)
 - See also...
 - “Command Data Types (c_type)” on page 47
- r_type - Specifies the data type in the response (msgdata)
 - See also...
 - “Response Data Types (r_type)” on page 47

Numeric input can be supplied in the following formats:

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

See also...

- “Example (Get_Attribute):” on page 48
- “Example (Set_Attribute):” on page 48

- **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
BITS8	Yes	8-bit field
BITS16	Yes	16-bit field
BITS32	Yes	32-bit field
OCTET	Yes	8-bit field
BIT1 - 7	Yes	1-bit to 7-bit field
PAD0 - 16	Yes	0 - 16-bit field, for filling up a string to a predefined size
NONE	No	Command holds no data, hence no data type

- **Response Data Types (r_type)**

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

Type	Supports Arrays	Comments
BOOL	Yes	Optionally, it is possible to exchange the BOOL data with a message based on the value (true or false). In such case, the actual data type returned from the function will be STRING. Syntax: BOOL<true><false> For arrays, the format will be BOOL[n]<true><false>.
SINT8	Yes	-
SINT16	Yes	-
SINT32	Yes	-
UINT8	Yes	This type can also be used when reading ENUM data types from an object. In such case, the actual ENUM value will be returned.
UINT16	Yes	-
UINT32	Yes	-
CHAR	Yes	-
STRING	No	-
ENUM	No	When using this data type, the ABCCMessage() function will first read the ENUM value. It will then issue a 'Get Enum String'-command to retrieve the actual enumeration string. The actual data type in the response will be STRING.
FLOAT	Yes	-
BITS8	Yes	-
BITS16	Yes	-
BITS32	Yes	-
OCTET	Yes	-
BIT1 - 7	Yes	-

Type	Supports Arrays	Comments
PAD0 - 16	Yes	-
NONE	No	-

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

Example (Get_Attribute):

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

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

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

See also...

- “printf()” on page 42

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

See also...

- “scanf()” on page 44

Errors

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

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

See also...

- "SSI Output Configuration" on page 50

6.5 SSI Output Configuration

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

File format:

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

Each error code corresponds to a dedicated output string, labelled from 1 to 16.
See also...
- “Errors” on page 49

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

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

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

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

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

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

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

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

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

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

See also...

- “SsiOutput()” on page 38

7. JSON

7.1 General Information

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

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

Access

The JSON resources should be password protected. Add password protection by adding a file called `web_accs.cfg` in the root directory. See “Authorization” on page 30 for more information.

Error

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

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

7.2 JSON Objects

7.2.1 ADI

info.json

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

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

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

JSON object layout:

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

data.json

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

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

JSON object layout:

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

metadata.json

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

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

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

JSON object layout:

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

enum.json

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

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

Name	Data Type	Note
string	String	-
value	Number	-

JSON object layout:

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

update.json

POST adi/update.json - form data:

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

Updates the value of an ADI for the specified ADI instance <instance>. The value, <data>, shall be hex formatted (see “Hex Format Explained” on page 59 for more information). If <element> is specified, only the value of the specified element is updated. In this case, <data> shall only update that single element value. When <element> is not specified, <data> shall represent the entire array value. Optionally, a callback may be passed to the request for JSONP output.

Name	Data Type	Note
result	Number	0 = success

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

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

7.2.2 Module

info.json

GET module/info.json.

Name	Data Type	Note
modulename	String	-
serial	String	32 bit hex ASCII
fwver	Array of Number	(major, minor, build)
uptime	Array of Number	[high, low] milliseconds (ms)
cpuload	Number	CPU load in %

JSON object layout:

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

7.2.3 Network

ethstatus.json

GET network/ethstatus.json.

“comm2” is not present if Ethernet port 2 is inactivated.

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

Comm Object Definition:

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

JSON object layout:

```
{
  "mac":          "003011FF0201",
  "comm1":       {
    "link":       1,
    "speed":      1,
    "duplex":     1
  }
  "comm2":       {
    "link":       0,
    "speed":      0,
    "duplex":     0
  }
}
```

ipstatus.json & ipconf.json

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

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

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

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

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

Example:

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

ethconf.json

GET network/ethconf.json.

“comm2” is not present if Ethernet port 2 is inactivated.

The values of "comm1" and "comm2" comes from the Network Configuration object (04h) instance 7 and instance 8. See “Instance Attributes (Instance #7, Ethernet Communication Settings 1)” on page 101 for more information.

Name	Data Type	Note
comm1	Number	-
comm2	Number	-

ifcounters.json

GET network/ifcounters.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	Note
inoctets	Number	IN: bytes
inucast	Number	IN: unicast packets
innucast	Number	IN: broadcast and multicast packets
indiscards	Number	IN: discarded packets
inerrors	Number	IN: errors
inunknown	Number	IN: unsupported protocol type
outoctets	Number	OUT: bytes
outucast	Number	OUT: unicast packets
outnucast	Number	OUT: broadcast and multicast packets
outdiscards	Number	OUT: discarded packets
outerrors	Number	OUT: errors

mediacounters.json

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

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

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

nwstats.json

GET network/nwstats.json.

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

Example output:

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

Get network specific statistics:

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

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

7.2.4 Services

smtp.json

GET services/smtp.json.

Note: Password is not returned when retrieving the settings.

Name	Data Type	Note
server	String	-
user	String	-

7.2.5 Hex Format Explained

The metadata max and min fields and the ADI values are ABP data encoded in a hex format. If the data type is an integer, the endianness used is determined by the data format field found in adi/info.json (see “info.json” on page 51).

Examples:

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

0500

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

414243

7.3 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 example below is built using the following libraries:

- jQuery JavaScript Library v1.9.1 (<http://jquery.com>)
- JavaScript Templates 2.2.0 (<https://github.com/blueimp/JavaScript-Templates>)

The file, containing this code, has to be stored in the built-in file system, see “File System” on page 20, 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>
```

8. CIP Objects

8.1 General Information

This chapter specifies the CIP-object implementation in the module. These objects can be accessed from the network, but not directly by the host application.

Mandatory Objects:

- “Identity Object (01h)” on page 63
- “Message Router (02h)” on page 66
- “Assembly Object (04h)” on page 67
- “Connection Manager (06h)” on page 70
- “Parameter Object (0Fh)” on page 74
- “DLR Object (47h)” on page 77
- “QoS Object (48h)” on page 78
- “TCP/IP Interface Object (F5h)” on page 87
- “Ethernet Link Object (F6h)” on page 90

CIP Energy Objects:

- “Base Energy Object (4Eh)” on page 79
- “Power Management Object (53h)” on page 81

Optional Objects:

- “Port Object (F4h)” on page 85 (Optional)

Vendor Specific Objects:

- “ADI Object (A2h)” on page 83

It is possible to implement additional CIP-objects in the host application using the CIP forwarding functionality, see “EtherNet/IP Host Object (F8h)” on page 161 and “Command Details: Process_CIP_Object_Request” on page 166.

Unconnected CIP routing is supported, which means that a message can be sent to a device without first setting up a connection.

8.2 Translation of Status Codes

If an error occurs when an object is requested from the application, an error code is returned. These Anybus CompactCom 40 error codes are translated to CIP status codes according to the table below.

Anybus CompactCom 40 Error Code		CIP Status Code	
Value	Error	Value	Status
00h	Reserved	1Eh	Embedded service error
01h	Reserved	1Eh	Embedded service error
02h	Invalid message format	1Eh	Embedded service error
03h	Unsupported object	05h	Path destination unknown
04h	Unsupported instance	05h	Path destination unknown
05h	Unsupported Command	08h	Service not supported
06h	Invalid CmdExt(0)	14h	Depending on Anybus CompactCom Service returning this reply, e.g. attribute not supported
07h	Invalid CmdExt(1)	-	Depending on Anybus CompactCom Service returning this reply
08h	Attribute not settable	0Eh	Attribute not settable
09h	Attribute not gettable	2Ch	Attribute not gettable
0Ah	Too Much Data	15h	Too much data
0Bh	Not Enough Data	13h	Not enough data
0Ch	Out of range	09h	Invalid attribute value
0Dh	Invalid state	0Ch	Object state conflict
0Eh	Out of resources	02h	Resource unavailable
0Fh	Segmentation failure	1Eh	Embedded service error
10h	Segmentation buffer overflow	23h	Buffer overflow
11h	Value too high	09h	Invalid attribute value
12h	Value too low	09h	Invalid attribute value
13h	Attribute controlled	0Fh	A permission/privilege check failed
14h	Message channel too small	11h	Reply data too large
FFh	Object Specific Error	1Fh	Vendor specific error. No additional error codes will be sent on EtherNet/IP
Other	-	1Eh	Embedded service error

For further information about the Anybus CompactCom error codes, please consult the Anybus CompactCom 40 Software Design Guide.

8.3 Identity Object (01h)

Category

Extended

Object Description

The Identity Object provides identification of and general information about the module.

The object supports multiple instances. Instance 1, which is the only mandatory instance, describes the whole product. It is used by applications to determine what nodes are on the network and to match an EDS file with a product on the network. The other (optional) instances describe different parts of the product, e.g. the software.

If modular device functionality is enabled, a list of the modules in the slots can be retrieved and made available to the network master by sending a get request to class attribute 100.

Instance attributes 1 - 7 can be customized by implementing the EtherNet/IP Host Object.

Additional identity instances can be registered by implementing the CIP Identity Host Object (host application object).

See also

- “EtherNet/IP Host Object (F8h)” on page 161
- “CIP Identity Host Object (EDh)” on page 158

Supported Services

Class:	Get_Attribute_Single Get_Attributes_All
Instance:	Get_Attribute_Single Set_Attribute_Single Get_Attributes_All Reset

Class Attributes

#	Name	Access	Type	Value
1	Revision	Get	UINT	0001h (Object revision)
2	Max instance	Get	UINT	Maximum instance number
3	Number of instances	Get	UINT	Number of instances
100	Module ID List	Get	Array of UINT32	If modular device functionality is enabled, a request to this attribute will generate a Get_List request to the Modular Device Object in the host application.

Instance Attributes

Extended

#	Name	Access	Type	Value
1	Vendor ID	Get	UINT	005Ah (HMS Industrial Networks AB) ^a
2	Device Type	Get	UINT	002Bh (Generic Device) ^a
3	Product Code	Get	UINT	0037h (Anybus CompactCom 40 EtherNet/IP) ^a
4	Revision	Get	Struct of: USINT USINT	Major and minor firmware revision ^a
5	Status	Get	WORD	See "Device Status" on page 65
6	Serial Number	Get	UDINT	Unique serial number (assigned by HMS) ^a
7	Product Name	Get	SHORT_STRING	"Anybus CompactCom 40 EtherNet/IP (TM)" ^a
11	Active language	Set	Struct of: USINT USINT USINT	Requests sent to this instance are forwarded to the Application Object. If the request is accepted, the module will update the language accordingly.
12	Supported Language List	Get	Array of: Struct of: USINT USINT USINT	List of languages supported by the host application. The list is read from the Application Object and translated to CIP standard. By default the only supported language is English. The application has to implement the corresponding attributes in the application object to enable more languages.

a. Can be customized by implementing the EtherNet/IP Host Object, see "EtherNet/IP Host Object (F8h)" on page 161

Device Status

bit(s)	Name
0	Module Owned
1	(reserved)
2	Configured ^a
3	(reserved)
4... 7	Extended Device Status: <u>Value:Meaning:</u> 0000b Unknown 0010b Faulted I/O Connection 0011b No I/O connection established 0100b Non volatile configuration bad 0101b Major fault 0110b Connection in Run mode 0111b Connection in Idle mode (other) (reserved)
8	Set for minor recoverable faults ^b
9	Set for minor unrecoverable faults ^b
10	Set for major recoverable faults ^b
11	Set for major unrecoverable faults ^b
12... 15	(reserved)

a. This bit shows if the product has other settings than “out-of-box”. The value is set to true if the configured attribute in the Application Object is set and/or the module’s NV storage is changed from default.

b. See “Diagnostic Object (02h)” on page 97.

Service Details: Reset Service

Note: This service is not supported if safety is enabled in the Functional Safety Object (E8h).

The module forwards reset requests from the network to the host application. For more information about network reset handling, consult the general Anybus CompactCom 40 Software Design Guide.

There are two types of network reset requests on EtherNet/IP:

- **Type 0: ‘Power Cycling Reset’**

This service emulates a power cycling of the module, and corresponds to Anybus reset type 0 (Power cycling). For further information, consult the general Anybus CompactCom 40 Software Design Guide.

- **Type 1: ‘Out of box reset’**

This service sets a “out of box” configuration and performs a reset, and corresponds to Anybus reset type 2 (Power cycling + factory default). For further information, consult the general Anybus CompactCom 40 Software Design Guide.

8.4 Message Router (02h)

Category

Extended

Object Description

The Message Router Object provides a messaging connection point through which a client may address a service to any object class or instance residing in the physical module.

In the Anybus CompactCom 40 module it is used internally to direct object requests.

Supported Services

Class: -

Instance: -

Class Attributes

-

Instance Attributes

-

8.5 Assembly Object (04h)

Category

Extended

Object Description

The Assembly object uses static assemblies and holds the Process Data sent/received by the host application. It allows data to and from each object to be sent or received over a single connection. The default assembly instance IDs used are in the vendor specific range.

It is possible for the application to create and support up to six consuming and six producing instances if the Assembly Mapping Object is implemented.

The terms “input” and “output” are defined from the network’s point of view. An input will produce data on the network and an output will consume data from the network.

See also...

- “Process Data” on page 19
- “EtherNet/IP Host Object (F8h)” on page 161
- Assembly Mapping Object (see Anybus CompactCom 40 Software Design Guide)

Supported Services

Class: Get_Attribute_Single

Instance: Get_Attribute_Single
Set_Attribute_Single

Class Attributes

#	Name	Access	Type	Value
1	Revision	Get	UINT	0002h (Object revision)
2	Max Instance	Get	UINT	(Highest instance number)

Instance 03h Attributes (Heartbeat, Input-Only)

Extended

This instance is used as heartbeat for Input-Only connections. The data size of the Heartbeat instance in the Forward_Open-request should be 0 bytes, however other values are also permitted.

#	Name	Access	Type	Value
3	Data	Set	N/A	- (The data size of this attribute is zero)
4	Size	Get	UINT	0 (Number of bytes in attribute 3)

The instance number for this instance can be changed by implementing the corresponding attribute in the EtherNet/IP Host Object.

Instance 04h Attributes (Heartbeat, Listen-Only)

Extended

This instance is used as heartbeat for listen-only connections. The data size of the Heartbeat instance in the Forward_Open-request should be 0 bytes, however other values are also permitted.

#	Name	Access	Type	Value
3	Data	Set	N/A	- (The data size of this attribute is zero)
4	Size	Get	UINT	0 (Number of bytes in attribute 3)

The instance number for this instance can be changed by implementing the corresponding attribute in the EtherNet/IP Host Object.

Instance 05h Attributes (Configuration Data)

Extended

Configuration Data that is sent through the 'Forward_Open'-service will be written to this instance.

#	Name	Access	Type	Value
3	Data	Set	N/A	- (Configuration data written to the application when the forward open command has the configuration data included)
4	Size	Get	UINT	Number of bytes in attribute 3

The instance number for this instance can be changed by implementing the corresponding attribute in the EtherNet/IP Host Object.

See also...

- “Command Details: Set_Configuration_Data” on page 167
- “Command Details: Get_Configuration_Data” on page 170

Instance 06h Attributes (Heartbeat, Input-Only Extended)

Extended

This instance is used as heartbeat for input-only extended connections, and does not carry any attributes. The state of connections made to this instance does not affect the state of the Anybus CompactCom 40 module, i.e. if the connection times out, the module does not switch to the Error state. The data size of the Heartbeat instance in the Forward_Open-request should be 0 bytes, however other values are also permitted.

The instance number for this instance can be changed by implementing the corresponding attribute in the EtherNet/IP Host Object.

#	Name	Access	Type	Value
3	Data	Set	N/A	- (The data size of this attribute is zero)
4	Size	Get	UINT	0 (Number of bytes in attribute 3)

Instance 07h Attributes (Heartbeat, Listen-Only Extended)

Extended

This instance is used as heartbeat for listen-only extended connections, and does not carry any attributes. The state of connections made to this instance does not affect the state of the Anybus CompactCom 40 module, i.e. if the connection times out, the module does not switch to the Error state. The data size of the Heartbeat instance in the Forward_Open-request should be 0 bytes, however other values are also permitted.

The instance number for this instance can be changed by implementing the corresponding attribute in the EtherNet/IP Host Object.

#	Name	Access	Type	Value
3	Data	Set	N/A	- (The data size of this attribute is zero)
4	Size	Get	UINT	0 (Number of bytes in attribute 3)

Instance 64h Attributes (Producing Instance)

Extended

The instance number for this instance can be changed by implementing the corresponding attribute in the EtherNet/IP Host Object.

#	Name	Access	Type	Value
3	Produced Data	Get	Array of BYTE	This data corresponds to the Write Process Data.
4	Size	Get	UINT	Number of bytes in attribute 3

See also...

- “Network Data Exchange” on page 19
- “EtherNet/IP Host Object (F8h)” on page 161 (Instance attribute #7)

Instance 96h Attributes (Consuming Instance)

Extended

The instance number for this instance can be changed by implementing the corresponding attribute in the EtherNet/IP Host Object.

#	Name	Access	Type	Value
3	Consumed Data	Set	Array of BYTE	This data corresponds to the Read Process Data.
4	Size	Get	UINT	Number of bytes in attribute 3

See also...

- “Network Data Exchange” on page 19
- “EtherNet/IP Host Object (F8h)” on page 161 (Instance attribute #8)

8.6 Connection Manager (06h)

Category

Extended

Object Description

This object is used for connection and connectionless communications, including establishing connections across multiple subnets.

Supported Services

Class: -

Instance: Get Attribute All
 Get Attribute Single
 Set Attribute Single
 Large_Forward_Open
 Forward_Open
 Forward_Close
 Unconnected Send (when unconnected routing is enabled)

Class Attributes

(No supported class attributes)

Instance Attributes

Extended

#	Name	Access	Type	Description
1	Open Requests	Set	UINT	Number of Forward Open service requests received.
2	Open Format Rejects	Set	UINT	Number of Forward Open service requests which were rejected due to bad format.
3	Open Resource Rejects	Set	UINT	Number of Forward Open service requests which were rejected due to lack of resources.
4	Open Other Rejects	Set	UINT	Number of Forward Open service requests which were rejected for reasons other than bad format or lack of resources.
5	Close Requests	Set	UINT	Number of Forward Close service requests received.
6	Close Format Rejects	Set	UINT	Number of Forward Close service requests which were rejected due to bad format.
7	Close Other Rejects	Set	UINT	Number of Forward Close service requests which were rejected for reasons other than bad format.
8	Connection Timeouts	Set	UINT	Total number of connection timeouts that have occurred in connections controlled by this Connection Manager.

Class 0 Connection Details

General

Class 0 connections are only supported for safety connections. The CompactCom device will act as a transparent bridge for safety connections, meaning that open and close requests for safety connections and safety I/O data will be forwarded to the safety module. Class 0 connections use UDP transport.

- Total number of supported class 0 connections: 2
- Max input connection size: 241 bytes^a
- Max output connection size: 239 bytes^a
- Supported RPI^b: 1... 20000 ms

a. Including the Mode Byte, Actual, Complement and Time stamp sections

b. Requested packet interval

Class 1 Connection Details

General

Class 1 connections are used to transfer I/O data, and can be established to instances in the Assembly Object. Each Class 1 connection will establish two data transports; one consuming and one producing. The heartbeat instances can be used for connections that shall only access inputs. Class 1 connections use UDP transport. Null forward open is supported.

- Total number of supported class 1 connections: 4
- Max input connection size: 1448 bytes with Large_Forward_Open, 509 bytes with Forward_Open
- Max output connection size: 1448 bytes with Large_Forward_Open, 505 bytes with Forward_Open
- Supported RPI^a: 1... 3200 ms
- T^b->O^c Connection type: Point-to-point, Multicast, Null
- O->T Connection type: Point-to-point, Null
- Supported trigger types: Cyclic, CoS (Change of State)
- Supported priorities: Low, High, Scheduled, Urgent

a. Requested packet interval

b. Target, in this case the module

c. Origin, in this case the master

Connection Types

- **Exclusive-Owner connection**

This type of connection controls the outputs of the Anybus module and does not depend on other connections.

- Max. no. of Exclusive-Owner connections: 1
- Connection point $O \Rightarrow T$: Assembly Object, instance 96h (Default)
- Connection point $T \Rightarrow O$: Assembly Object, instance 64h (Default)

- **Input-Only connection**

This type of connection is used to read data from the Anybus module without controlling the outputs. It does not depend on other connections.

- Max. no. of Input-Only connections: Up to 4¹
- Connection point $O \Rightarrow T$: Assembly Object, instance 03h (Default)
- Connection point $T \Rightarrow O$: Assembly Object, instance 64h (Default)

Note: If an Exclusive-Owner connection has been opened towards the module and times out, the Input-Only connection times out as well. If the Exclusive-Owner connection is properly closed, the Input-Only connection remains unaffected.

- **Input-Only Extended connection**

This connections functionality is the same as the standard Input-Only connection. However when this connection times out it does not affect the state of the application.

- Connection point $O \Rightarrow T$: Assembly Object, instance 06h (Default)
- Connection point $T \Rightarrow O$: Assembly Object, instance 64h (Default)

- **Listen-Only connection**

This type of connection requires another connection in order to exist. If that connection (Exclusive-Owner or Input-Only) is closed, the Listen-Only connection will be closed as well.

- Max. no. of Input-Only connections: Up to 4²
- Connection point $O \Rightarrow T$: Assembly Object, instance 04h (Default)
- Connection point $T \Rightarrow O$: Assembly Object, instance 64h (Default)

- **Listen-Only Extended connection**

This connections functionality is the same as the standard Listen-Only connection. However when this connection times out it does not affect the state of the application.

- Connection point $O \Rightarrow T$: Assembly Object, instance 07h (Default)
- Connection point $T \Rightarrow O$: Assembly Object, instance 64h (Default)

- **Redundant-Owner connection**

This connection type is not supported by the module.

1. Shared with Exclusive-Owner and Listen-Only connections
2. Shared with Exclusive-Owner and Input-Only connections

Class 3 Connection Details

- **Explicit message connection**

Class 3 connections are used to establish connections towards the message router. Thereafter, the connection is used for explicit messaging. Class 3 connections use TCP transport.

- No. of simultaneous Class 3 connections: 6
- Supported RPI: 100 - 10000 ms
- T->O Connection type: Point-to-point
- O->T Connection type: Point-to-point
- Supported trigger type: Application
- Supported connection size: 1448 bytes
- Supported priorities: Low

8.7 Parameter Object (0Fh)

Category

Extended

Object Description

The Parameter Object provides an interface to the configuration data of the module. It can provide all the information necessary to define and describe each of the module configuration parameters, as well as a full description of each parameter, including minimum and maximum values and a text string describing the parameter. Configuration tools, such as RSNetworkx, can extract information about the Application Data Instances (ADIs) and present them with their actual name and range to the user.

Since this process may be somewhat time consuming, especially when using the serial host interface, it is possible to disable support for this functionality in the EtherNet/IP Host Object.

Each parameter is represented by one instance. Instance numbers start at 1, and are incremented by one, with no gaps in the list. Due to limitations imposed by the CIP standard, ADIs containing multiple elements (i.e. arrays etc.) cannot be represented through this object. In such cases, default values will be returned, see “Default Values” on page 76.

See also...

- “ADI Object (A2h)” on page 83 (CIP Object)
- “EtherNet/IP Host Object (F8h)” on page 161 (Host Application Object)

Supported Services

Class: Get_Attribute_Single

Instance: Get_Attribute_Single
 Set_Attribute_Single
 Get_Attributes_All
 Get_Enum_String

Class Attributes

#	Name	Access	Type	Value
1	Revision	Get	UINT	0001h (Revision of the object)
2	Max Instance	Get	UINT	Maximum created instance number = class attribute 3 in the Application Data Object ^a
8	Parameter Class Descriptor	Get	WORD	Default: 0000 0000 0000 1011b <u>Bit:Contents:</u> 0 Supports parameter instances 1 Supports full attributes 2 Must do non-volatile storage save command 3 Parameters are stored in non-volatile storage
9	Configuration Assembly Instance	Get	UINT	0000h (Application does not support configuration data) 0005h (If the application supports configuration data, unless the configuration instance number has been changed using attribute 15 in the EtherNet/IP Host Object.)

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

Instance Attributes

Extended

#	Name	Access	Type	Value
1	Parameter Value	Get/Set	Specified in attributes 4, 5 & 6.	Actual value of parameter This attribute is read-only if bit 4 of Attribute #4 is true
2	Link Path Size	Get	USINT	0007h (Size of link path in bytes)
3	Link Path	Get	Packed EPATH	20 A2 25 nn nn 30 05h (Path to the object from where this parameter's value is retrieved, in this case the ADI Object)
4	Descriptor	Get	WORD	<u>Bit:Contents:</u> 0 Supports Settable Path (N/A) 1 Supports Enumerated Strings 2 Supports Scaling (N/A) 3 Supports Scaling Links (N/A) 4 Read only Parameter 5 Monitor Parameter (N/A) 6 Supports Extended Precision Scaling (N/A)
5	Data Type	Get	USINT	Data type code
6	Data Size	Get	USINT	Number of bytes in parameter value
7	Parameter Name String	Get	SHORT_STRING	Name of the parameter, truncated to 16 chars
8	Units String	Get	SHORT_STRING	"" (default string)
9	Help String	Get	SHORT_STRING	
10	Minimum Value	Get	(Data Type) ^a	Minimum value of parameter
11	Maximum Value	Get	(Data Type) ^a	Maximum value of parameter
12	Default Value	Get	(Data Type) ^a	Default value of parameter
13	Scaling Multiplier	Get	UINT	0001h
14	Scaling Divisor	Get	UINT	
15	Scaling Base	Get	UINT	
16	Scaling Offset	Get	INT	0000h
17	Multiplier Link	Get	UINT	
18	Divisor Link	Get	UINT	
19	Base Link	Get	UINT	
20	Offset Link	Get	UINT	
21	Decimal Precision	Get	USINT	00h

a. The Data Type is defined in attribute 5.

Default Values

In case of ADIs containing several elements, that cannot be represented through this object, default values will be returned, according to the table below.

#	Name	Value	Comments
1	Parameter Value	0	-
2	Link Path Size	0	Size of link path in bytes.
3	Link Path	-	NULL Path
4	Descriptor	0010h	Read only Parameter
5	Data type	C6h	USINT
6	Data size	1	-
7	Parameter Name String	"Reserved"	-
8	Units String	""	-
9	Help String	""	-
10	Minimum value	N/A	0
11	Maximum value	N/A	0
12	Default value	N/A	0
13	Scaling Multiplier	N/A	1
14	Scaling Divisor	N/A	1
15	Scaling Base	N/A	1
16	Scaling Offset	N/A	0
17	Multiplier Link	N/A	0
18	Divisor Link	N/A	0
19	Base Link	N/A	0
20	Offset Link	N/A	0
21	Decimal Precision	N/A	0

8.8 DLR Object (47h)

Category

Extended

Object Description

The Device Level Ring (DLR) Object provides the status information interface for the DLR protocol. This protocol enables the use of an Ethernet ring topology, and the DLR Object provides the CIP application-level interface to the protocol.

Supported Services

Class: Get_Attribute_Single
 Get_Attributes_All

Instance: Get_Attribute_Single

Class Attributes

#	Name	Access	Type	Value
1	Revision	Get	UINT	0003h (Object revision)

Instance Attributes

Extended

#	Name	Access	Type	Value
1	Network Topology	Get	USINT	<u>Bit Contents:</u> 0 "Linear" 1 "Ring"
2	Network Status	Get	USINT	<u>Bit Contents:</u> 0 "Normal" (N/A) 1 "Ring Fault" 2 "Unexpected Loop Detected" 3 "Partial Network Fault" 4 "Rapid Fault/Restore Cycle"
10	Active Supervisor Address	Get	Struct of: UDINT Array of: 6 USINTs	This attribute holds the IP address (IPv4) and/or the Ethernet Mac address of the active ring supervisor
12	Capability Flags	Get	DWORD	82h (Beacon-based ring node, Flush_Table frame capable)

8.9 QoS Object (48h)

Category

Extended

Object Description

Quality of Service (QoS) is a general term that is applied to mechanisms used to treat traffic streams with different relative priorities or other delivery characteristics. Standard QoS mechanisms include IEEE 802.1D/Q (Ethernet frame priority) and Differentiated Services (DiffServ) in the TCP/IP protocol suite.

The QoS Object provides a means to configure certain QoS related behaviors in EtherNet/IP devices.

The object is required for devices that support sending EtherNet/IP messages with nonzero DiffServ code points (DSCP), or sending EtherNet/IP messages in 802.1Q tagged frames.

Supported Services

Class: Get_Attribute_Single

Instance: Get_Attribute_Single
Set_Attribute_Single

Class Attributes

#	Name	Access	Type	Value
1	Revision	Get	UINT	0001h (Object revision)

Instance Attributes

Extended

#	Name	Access	Type	Value
1	802.1Q Tag Enable	Set	USINT	Enables or disables sending 802.1Q frames. <u>Bit Contents:</u> 0 Disabled (Default) 1 Enabled
4	DSCP Urgent	Set	USINT	CIP transport class 1 messages with priority Urgent Default: 55
5	DSCP Scheduled	Set	USINT	CIP transport class 1 messages with priority Scheduled Default: 47
6	DSCP High	Set	USINT	CIP transport class 1 messages with priority High Default: 43
7	DSCP Low	Set	USINT	CIP transport class 1 messages with priority Low Default: 31
8	DSCP Explicit	Set	USINT	CIP UCMM and CIP class 3 Default: 27

8.10 Base Energy Object (4Eh)

Category

Extended

Object Description

The Base Energy Object acts as an “Energy Supervisor” for CIP Energy implementations. It is responsible for providing a time base for energy values, provides energy mode services, and can provide aggregation services for aggregating energy values up through the various levels of an industrial facility. It also provides a standard format for reporting energy metering results. The object is energy type independent and allows energy type specific data and functionality to be integrated into an energy system in a standard way. The Anybus CompactCom 40 EtherNet/IP module supports one instance of the Base Energy Object. For instance, an electric power monitor may count metering pulse output transitions of a separate metering device. The count of such transitions, represented by a Base Energy Object instance, would reflect the energy consumption measured by the separate metering device. An instance of the Base Energy Object may exist as a stand-alone instance, or it may exist in conjunction with an Electrical and/or Non-Electrical Energy Object instance¹. If an instance of any of these objects is implemented in a device, it must be associated with a Base Energy Object instance in the device.

For this object to be able to access the network, the Energy Reporting Object (E7h) must be implemented in the host application, see the Anybus CompactCom 40 Software Design Guide for more information.

Supported Services

Class: Get_Attribute_Single

Instance: Get_Attribute_Single

Class Attributes

#	Name	Access	Type	Value
1	Revision	Get	UINT	0002h (Object revision)

1. These objects are not implemented in the Anybus CompactCom 40 EtherNet/IP

Instance Attributes

Extended

#	Name	Access	Type	Value/Description
1	Energy/ Resource Type	Get	UINT	Type of energy managed by this instance Always 0 (Generic)
2	Base Energy Object Capabili- ties	Get	UINT	Always 0 (Energy measured)
3	Energy Accu- racy	Get	UINT	Specifies the accuracy of power and energy metering results, either in 0.01 percent of reading (default) or 0.01 of other units specified in attribute #4. If 0, unknown.
4	Energy Accu- racy Basis	Get	UINT	Always 0 (Percent of reading)
7 ^a	Consumed Energy Odome- ter	Get	ODOMETER ^b	The value of the consumed energy.
8 ^a	Generated Energy Odome- ter	Get	ODOMETER ^b	The value of the generated energy.
12	Energy Type Specific Object Path	Get	Struct of: UINT (Path size) padded EPATH (Path)	NULL path

- a. Depending on whether the instance reports consumed or generated energy, either attribute #7 or attribute #8 is required.
- b. This struct data type makes it possible to represent very large values, for more information please consult the CIP specification Volume 1 (CIP Common).

8.11 Power Management Object (53h)

Category

Extended

Object Description

The Power Management Object provides standardized attributes and services to support the control of devices into and out of paused or sleep states. The Energy Control Object (F0h) has to be implemented for this object to gain access to the network.

See also ..

- Energy Control Object (F0h) (Anybus CompactCom 40 Software Design Guide)

Supported Services

Class: Get_Attribute_Single

Instance: Get_Attribute_Single
Power_Management
Set_Pass_Code
Clear_Pass_Code

Class Attributes

#	Name	Access	Type	Value
1	Revision	Get	UINT	0002h (Object revision)

Instance Attributes

Extended

#	Name	Access	Type	Value/Description
1	Power Management Command	Get	DWORD	Collection of bit fields comprising the most recent power management request.
2	Power Management Status	Get	DWORD	Collection of bit fields providing Power Management status information.
3	Client Path	Get	Struct of:	Specifies the EPATH from this instance (server) to its current owner (client).
			UINT (Path Size)	Size of path (in words)
			Padded EPATH (Path)	
4	Number of Power Management Modes	Get	UINT	Number of Power Management Mode array entries in attribute 5.

#	Name	Access	Type	Value/Description
5	Power Management Nodes	Get	Array of:	Array of low power modes
			Struct of:	Modes (Array of mode structures)
			USINT	Minimum Pause Units (Specifies the unit of Minimum Pause Time)
			UINT	Minimum Pause Time
			USINT	Resume Units (Specifies the unit of Resume Time)
			UINT	Resume Time (Required time to transition from the paused stated to the owned state.
			REAL	Power Level (Power in kW for this mode)
BOOL	Availability (Specifies whether this mode can be entered given the current device state)			
6	Sleeping State Support	Get	BOOL	0 (Sleeping state not supported)

8.12 ADI Object (A2h)

Category

Extended

Object Description

This object maps instances in the Application Data Object to EtherNet/IP. All requests to this object will be translated into explicit object requests towards the Application Data Object in the host application; the response is then translated back to CIP-format and sent to the originator of the request.

The ADI Object can be disabled using attribute 30 in the EtherNet/IP Host Object (F8h). This attribute can also be used to change the ADI Object number.

See also...

- Application Data Object (see Anybus CompactCom 40 Software Design Guide)
- “Parameter Object (0Fh)” on page 74 (CIP Object)
- “EtherNet/IP Host Object (F8h)” on page 161

Supported Services

Class: Get_Attribute_Single

Instance: Get_Attribute_Single
Set_Attribute_Single

Class Attributes

#	Name	Access	Type	Value
1	Revision	Get	UINT	Object revision (Current value = 0002h)
2	Max Instance	Get	UINT	Equals attribute #4 in the Application Data Object ^a
3	Number of instances	Get	UINT	Equals attribute #3 in the Application Data Object ^a

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

Instances Attributes

Each instance corresponds to an instance within the Application Data Object (for more information, consult the general Anybus CompactCom 40 Software Design Guide).

Extended

#	Name	Access	Type	Description
1	Name	Get	SHORT_STRING	Parameter name (Including length)
2	ABCC Data type	Get	Array of USINT	Data type of instance value
3	No. of Elements	Get	USINT	Number of elements of the specified data type
4	Descriptor	Get	Array of USINT	Bit field describing the access rights for this instance <u>Bit:Meaning:</u> 0 Set = Get Access 1 Set = Set Access
5	Value ^a	Get/Set	Determined by attribute #2	Instance value
6	Max Value ^a	Get		The maximum permitted parameter value.
7	Min Value ^a	Get		The minimum permitted parameter value.
8	Default Value ^a	Get		The default parameter value.
9	Number of subelements	Get	UINT	Number of subelements in the ADI. Default value is 1 unless implemented in the application.

a. Converted to/from CIP standard by the module

8.13 Port Object (F4h)

Category

Extended

Object Description

The Port Object describes the CIP ports present on the device. Each routable CIP port is described in a separate instance. Non-routable ports may be described. Devices with a single CIP port are not required to support this object.

The object exists only if enabled in the EtherNet/IP Host Object (Instance Attribute #17).

See also...

- “EtherNet/IP Host Object (F8h)” on page 161 (Anybus Module Object)
- “CIP Port Configuration Object (0Dh)” on page 153 (Host Application Object)

Supported Services

Class: Get_Attributes_All
 Get_Attribute_Single

Instance: Get_Attributes_All
 Get_Attribute_Single

Class Attributes

#	Name	Access	Type	Value
1	Revision	Get	UINT	Object revision (Current value = 0001h)
2	Max Instance	Get	UINT	Max. instance number
3	Number of Instances	Get	UINT	Number of ports currently created.
8	Entry Port	Get	UINT	Returns the instance of the Port Object that describes the port through which this request entered the device.
9	Port Instance Info	Get	Array of:	Array of structures containing instance attributes 1 and 2 from each instance. The array is indexed by instance number, up to the maximum number of instances. The value at index 1 (offset 0) and any non-instantiated instances will be zero.
			Struct of: UINT (Type) UINT (Number)	Enumerates the type of port (see instance attribute #1) CIP port number associated with this port (see instance attribute #2)

Instances Attributes (Instance #1)

Extended

This instance reflects the properties associated with the Ethernet interface.

#	Name	Access	Type	Value
1	Port Type	Get	UINT	0h (default) 4h (if the application registers a port)
2	Port Number	Get	UINT	2h
3	Link Object	Get	Struct of: UINT Padded EPATH	- 2h (Path Length) 20 F5 24 01h (Link Path)
4	Port Name	Get	SHORT_STRING	"EtherNet/IP"
5	Port Type Name	Get	SHORT_STRING	""
6	Port Description	Get	SHORT_STRING	""
7	Node Address	Get	Padded EPATH	-

See also...

- "CIP Port Configuration Object (0Dh)" on page 153

Instances Attributes (Instances #2... #8)

Extended

#	Name	Access	Type	Value
1	Port Type	Get	UINT	Enumerates the type of port
2	Port Number	Get	UINT	CIP port number associated with this port
3	Link Object	Get	Struct of: UINT Padded EPATH	- Path length (number of 16-bit words) Logical path segments which identify the object for this port. The path must consist of one logical class segment and one logical instance segment. The maximum size is 12 bytes.
4	Port Name	Get	SHORT_STRING	Name of port, e.g. "Port A". Max. 64 characters.
5	Port Type Name	Get	SHORT_STRING	""
6	Port Description	Get	SHORT_STRING	""
7	Node Address	Get	Padded EPATH	Node number of this device on port. The range within this data type is restricted to a Port Segment.
8	Port Node Range	Get	Struct of: UINT (Min.) UINT (Max.)	- Min. node number on port Max. node number on port

See also...

- "CIP Port Configuration Object (0Dh)" on page 153 ("Instance Attributes" on page 154)

8.14 TCP/IP Interface Object (F5h)

Category

Extended

Object Description

This object provides the mechanism to configure the TCP/IP network interface of the module. It groups the TCP/IP-related settings in one instance for each TCP/IP capable communications interface.

See also...

- “Communication Settings” on page 16
- “Network Configuration Object (04h)” on page 99 (Anybus Module Object)

Supported Services

Class services: Get_Attribute_All
 Get_Attribute_Single

Instance services: Get_Attribute_All
 Get_Attribute_Single
 Set_Attribute_Single

Class Attributes

#	Name	Access	Type	Value
1	Revision	Get	UINT	0004h (Object revision)
2	Max instance	Get	UINT	1 (Maximum instance number)
3	Number of instances	Get	UINT	1 (Number of instances)
6	Maximum ID Number Class Attributes	Get	UINT	7 (The attribute number of the last implemented class attribute)
7	Maximum ID Number Instance Attributes	Get	UINT	13 (The attribute number of the last implemented instance attribute)

Instance Attributes

Extended

#	Name	Access	Type	Value	Comments
1	Status	Get	DWORD	-	<p><u>Bit #:</u> <u>Meaning:</u></p> <p>0-3: When set to h, attribute #5 contains valid configuration from DHCP or non-volatile storage. When set to 2h, attribute #5 contains valid configuration from hardware settings. Remaining values are reserved for future use.</p> <p>4: Multicast pending if set to 1.</p> <p>5: Interface configuration pending if set to 1. A new configuration will be loaded at the next reset.</p> <p>6: AcdStatus. Set to 1 if an address conflict is detected. Address conflict detection is enabled/disabled in attribute #10.</p> <p>7 AcdFault</p> <p>8 - 31: (reserved, set to 0)</p>
2	Configuration Capability	Get	DWORD	0000 0084h - or - 0000 0094h - or - 0000 00A4h - or - 0000 00B4h	<p><u>Bit #:</u> <u>Meaning:</u></p> <p>0-3: Always 4. For more information, consult the CIP specifications.</p> <p>4: The 'Configuration Settable'-bit reflects the value of instance attribute #9 in the "EtherNet/IP Host Object (F8h)" on page 161.</p> <p>5: The module is hardware configurable when this bit is set to 1. The bit will be set if any of the address attributes is set in the Network Configuration Object (04h) during setup or if attribute #6 (Hardware configurable address) in the Application Object (FFh) is set.</p> <p>6: Always 0. For more information, consult the CIP specifications.</p> <p>7: Always 1, the device is capable of detecting address conflicts.</p> <p>8 - 31: (reserved, set to 0)</p>
3	Configuration Control ^a	Get/Set ^a	DWORD	-	<p><u>Value:</u> <u>Meaning:</u></p> <p>0: Configuration from non-volatile memory</p> <p>2: Configuration from DHCP</p>
4	Physical Link Object	Get	Struct of: UINT (Path size) Padded EPATH	- 0002h 20 F6 24 03h	- - Path to Ethernet Link Object, Instance #3
5	Interface Configuration ^a	Get/Set ^a	Struct of: UDINT (IP) UDINT (Mask) UDINT (GW) UDINT (DNS1) UDINT (DNS2) STRING (Domain)	-	- IP address Subnet mask Default gateway Primary DNS Secondary DNS Default domain
6	Host Name	Get/Set	STRING	-	Host name of Anybus module
8	TTL Value	Get/Set	USINT	1	TTL value for EtherNet/IP multicast packets

#	Name	Access	Type	Value	Comments
9	Mcast Config	Get/Set	Struct of:		IP multicast configuration.
	Alloc Control		USINT	0	<u>Value:</u> <u>Meaning:</u> 0: Use default allocation algorithm to generate multicast addresses 1: Allocate multicast addresses according to the values in the 'Num Mcast'- and 'Mcast Start Addr'-fields.
	(reserved)		USINT	0	Set to zero. Do not change.
	Num Mcast		UINT	1	Number of multicast addresses to allocate for EtherNet/IP
	Mcast Start Addr		UDINT	-	Starting multicast address from which to begin allocation
10 ^b	SelectAcd	Set	Bool	1	<u>Value:</u> <u>Meaning:</u> 0: Disable ACD 1: Enable ACD (Default) If ACD (address conflict detection) is enabled, bit 6 in attribute #1 will be set if an ACD conflict is detected. The Network Status LED will also indicate a detected conflict, see "Network Status LED" on page 193.
11 ^b	LastConflictDetected	Set	Struct of:		ACD Diagnostic parameters Related to the last conflict detected.
	AcdActiviity		USINT	-	State of ACD activity when last conflict detected.
	RemoteMAC		ARRAY of 6 USINT	-	MAC address of remote node from the ARP PDU in which a conflict was detected.
	ArpPdu		ARRAY of 28 USINT	-	Copy of the raw ARP PDU in which a conflict was detected.
12	EIP QuickConnect ^{c,d}	Set	Bool	0	<u>Value:</u> <u>Meaning:</u> 0: Disable EIP QuickConnect (Default) 1: Enable EIP QuickConnect If EIP QuickConnect is enabled, the QuickConnect feature will direct EtherNet/IP target devices to quickly power up and join an EtherNet/IP network.
13	Encapsulation inactivity timeout	Set	UINT	0 - 3600	Number of seconds of inactivity before a TCP connection is closed. 0: Disabled

- Support for configuring network settings from the network can be disabled by implementing attribute #9 in the EtherNet/IP Host Object, see "Instance Attributes (Instance #1)" on page 162.
- Attributes #10 and #11 will not be available if ACD is disabled using attribute #11 in the Ethernet Host Object (F9h).
- If the module is configured to use EIP QuickConnect functionality, the EDS file has to be changed. As the EDS file is changed, the identity of the module has to be changed and the module will require certification, see "Conformance Test Guide" on page 12.
- This attribute exists if attribute #26 in the EtherNet/IP Host Object is implemented, see "Instance Attributes (Instance #1)" on page 162.

8.15 Ethernet Link Object (F6h)

Category

Extended

Object Description

This object maintains link specific counters and status information for an IEEE 802.3 communications interface. Exactly one instance for each communications interface on the module is supported. Instances for internally accessible interfaces can also be supported.

See also...

- “Communication Settings” on page 16
- “Network Configuration Object (04h)” on page 99 (Anybus Module Object)

Supported Services

Class services: Get_Attribute_All
 Get_Attribute_Single

Instance services: Get_Attribute_All
 Get_Attribute_Single
 Set_Attribute_Single
 Get_And_Clear

Class Attributes

By default, three instances (port 1, port 2 and the internal port) are implemented, meaning that two ports are activated.

If port 2 is inactivated in the Port 2 State attribute of the Ethernet Host Object (F9h), only one instance (port 1) should be implemented.

#	Name	Access	Type	Value
1	Revision	Get	UINT	0004h (Object revision)
2	Max instance	Get	UINT	1 or 3 (Maximum instance number)
3	Number of instances	Get	UINT	1 or 3 (Number of instances)
6	Maximum ID Number Class Attributes	Get	UINT	7 (The attribute number of the last implemented class attribute.)
7	Maximum ID Number Instance Attributes	Get	UINT	11 (The attribute number of the last implemented instance attribute.)

Instance Attributes

Extended

#	Access	Name	Type	Value	Comments
1	Get	Interface Speed	UDINT	10 or 100	Actual Ethernet interface speed.
2	Get	Interface Flags	DWORD	-	See "Interface Flags" on page 92.
3	Get	Physical Address	Array of 6 USINTs	(MAC ID)	Physical network address, i.e. assigned MAC address.
4	Get	Interface Counters	Struct of:		
		In Octets	UDINT	N/A	Octets received on the interface
		In Ucast Packets	UDINT	N/A	Unicast packets received on the interface
		In NUcast Packets	UDINT	N/A	Nonunicast packets received on the interface
		In Discards	UDINT	N/A	Inbound packets with unknown protocol
		In Errors	UDINT	N/A	Inbound packets that contain errors (does not include In discards)
		In Unknown Protos	UDINT	N/A	Inbound packets with unknown protocol
		Out Octets	UDINT	N/A	Octets sent on the interface
		Out Ucast Packets	UDINT	N/A	Unicast packets sent on the interface
		Out NUcast Packets	UDINT	N/A	Nonunicast packets sent on the interface
		Out Discards	UDINT	N/A	Outbound packets with unknown protocol
Out Errors	UDINT	N/A	Outbound packets that contain errors (does not include Out discards)		
5	Get	Media Counters	Struct of:		Media specific counters
		Alignment Errors	UDINT	N/A	Frames received that are not an integral number of octets in length
		FCS Errors	UDINT	N/A	Frames received that do not pass the FCS check
		Single Collisions	UDINT	N/A	Successfully transmitted frames that have experienced exactly one collision
		Multiple Collisions	UDINT	N/A	Successfully transmitted frames that have experienced more than one collision
		SQE Test Errors	UDINT	0	The number of times the SQE test error message is generated (Counter not provided with current PHY interface)
		Deferred Transmissions	UDINT	N/A	Frames for which the first transmission attempt is delayed because the medium is busy
		Late Collisions	UDINT	N/A	The number of times a collision is detected later than 512 bit-times into the transmission of a packet
		Excessive Collisions	UDINT	N/A	Frames for which a transmission fails due to excessive collisions
		MAC Transmit Errors	UDINT	N/A	Frames for which a transmission fails due to an internal MAC sublayer receive error
		Carrier Sense Errors	UDINT	N/A	The number of times that the carrier sense condition was lost or never asserted when attempting to transmit a frame
Frame Too Long	UDINT	N/A	Frames received that exceed the maximum permitted frame size		
MAC Receive Errors	UDINT	N/A	Frames for which reception on an interface fails due to an internal MAC sublayer receive error		

#	Access	Name	Type	Value	Comments
6 ^a	Get/Set	Interface Control	Struct of:		
		Control Bits	WORD	-	Interface control bits
		Forced Interface Speed	UINT	-	Speed at which the interface shall be forced to operate. Returns 'Object state Conflict' if auto-negotiation is enabled
7	Get	Interface Type	USINT	-	See "Interface State" on page 93
8	Get	Interface State	USINT	-	See "Interface State" on page 93
9 ^b	Get/Set	Admin State	USINT	-	See "Admin State" on page 93
10	Get	Interface Label	SHORT_STRING	-	See "Interface Label" on page 93
11	Get	Interface Capability	Struct of:	-	Indication of the capabilities of the interface
		Capability Bits	DWORD	-	Interface capabilities, other than speed/duplex See "Interface Capability" on page 94
		Speed/Duplex Options	Struct of:	-	Indicates speed/duplex pairs supported in the Interface Control Attribute
			USINT	-	Speed/duplex array count
			Array of Struct of:	-	Speed/duplex array
			UINT	-	Interface speed
USINT	-	Interface Duplex Mode 0 = half duplex 1 = full duplex 2 - 255 = Reserved			

- a. Support for this attribute can be disabled by implementing attribute #9 in the EtherNet/IP Host Object (F8h).
b. Support for this attribute can be disabled by implementing the port state attributes (#12 or #13) in the Ethernet Host object (F9h).

Interface Flags

Bit	Name	Description
0	Link status	Indicates whether or not the Ethernet 802.3 communications interface is connected to an active network. <u>Value:Meaning:</u> 0 Inactive link 1 Active link
1	Half/full duplex	Indicates the duplex mode currently in use. <u>Value:Meaning:</u> 0 Half duplex 1 Full duplex
2 - 4	Negotiation Status	Indicates the status of link auto-negotiation. <u>Value:Meaning:</u> 0 Auto-negotiation in progress. 1 Auto-negotiation and speed detection failed (using default values ^a) 2 Auto negotiation failed but detected speed (using default duplex value) 3 Successfully negotiated speed and duplex. 4 Auto-negotiation not attempted. Forced speed and duplex.
5	Manual Setting requires Reset	<u>Value:Meaning:</u> 0 Interface can activate changes to link parameters during runtime 1 Reset is required in order for changes to have effect
6	Local Hardware Fault	<u>Value:Meaning:</u> 0 No local hardware fault detected 1 Local hardware fault detected
7-31	(reserved)	Set to 0.

- a. Recommended default values are 10 Mbps, half duplex.

Interface State

This attribute indicates the current operational state of the interface.

Value	Description
0	Unknown interface state.
1	The interface is enabled and is ready to send and receive data.
2	The interface is disabled.
3	The interface is testing.

Admin State

This attribute controls the administrative setting of the interface state.

Value	Description
0	(reserved)
1	Enable the interface.
2	Disable the interface.
3-255	(reserved)

Interface Label

Instance	Value
1	Port 1
2	Port 2
3	Internal

Interface Type

Instance	Value	Description
1	2	Twisted-pair
2	2	Twisted-pair
3	1	Internal interface

Interface Capability

Bit	Name	Description	Implementation
0	Manual setting requires reset	Indicates whether or not the device requires a reset to apply changes made to the Interface Control attribute (#6). 0 = Indicates that the device automatically applies changes made to the Interface Control attribute (#6) and, therefore, does not require a reset in order for changes to take effect. This bit shall have this value when the Interface Control attribute (#6) is not implemented. 1 = Indicates that the device does not automatically apply changes made to the Interface Control attribute (#6) and, therefore, will require a reset in order for changes to take effect. Note: this bit shall also be replicated in the Interface Flags attribute (#2), in order to retain backwards compatibility with previous object revisions.	Return 0
1	Auto-negotiate	0 = Indicates that the interface does not support link auto-negotiation 1 = Indicates that the interface supports link auto-negotiation	0 for internal interface, 1 for external interfaces
2	Auto-MDIX	0 = Indicates that the interface does not support auto MDIX operation 1 = Indicates that the interface supports auto MDIX operation	0 for internal interface, 1 for external interfaces
3	Manual speed/duplex	0 = Indicates that the interface does not support manual setting of speed/duplex. The Interface Control attribute (#6) shall not be supported. 1 = Indicates that the interface supports manual setting of speed/duplex via the Interface Control attribute (#6)	0 for internal interface, 1 for external interfaces
4 - 31	Reserved	Shall be set to 0	Return 0

9. Anybus Module Objects

9.1 General Information

This chapter specifies the Anybus Module Object implementation and how they correspond to the functionality in the Anybus CompactCom 40 EtherNet/IP.

Standard Objects:

- “Anybus Object (01h)” on page 96
- “Diagnostic Object (02h)” on page 97
- “Network Object (03h)” on page 98
- “Network Configuration Object (04h)” on page 99

Network Specific Objects:

- “Socket Interface Object (07h)” on page 109
- “SMTP Client Object (09h)” on page 126
- “Anybus File System Interface Object (0Ah)” on page 131
- “Network Ethernet Object (0Ch)” on page 19
- “CIP Port Configuration Object (0Dh)” on page 153
- “Functional Safety Module Object (11h)” on page 149

9.2 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 40 CompactCom)
2... 11	-	-	-	Consult the general Anybus CompactCom 40 Software Design Guide for further information.
12	LED colours	Get	struct of: UINT8(LED1A) UINT8(LED1B) UINT8(LED2A) UINT8(LED2B)	<u>Value:Colour:</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	UINT64	Not used (Always 0)

9.3 Diagnostic Object (02h)

General Information

Basic

Object Description

This object provides a standardised way of handling host application events & diagnostics, and is thoroughly described in the general Anybus CompactCom 40 Software Design Guide.

Supported Commands

Object: Get_Attribute
 Create
 Delete

Instance: Get_Attribute

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1... 4	-	-	-	Consult the general Anybus CompactCom 40 Software Design Guide for further information.
11	Max no. of instances	Get	UINT16	5+1 (One instance is reserved for major diagnostic events.)
12	Supported functionality	Get	BITS32	00 00 00 00h (Latching events are not supported.)

Instance Attributes

Basic

#	Name	Access	Type	Value	
1	Severity	Get	UINT8	Consult the general Anybus CompactCom 40 Software Design Guide for further information.	
2	Event Code ^a	Get	UINT8		
3	-	-	-		Not implemented in product.
4	Slot ^a	Get	UINT16		
5	ADJ ^a	Get	UINT16		
6	Element ^a	Get	UINT8		
7	Bit ^a	Get	UINT8		

a. This attribute can not be represented on the network and is thus ignored by the module.

In this implementation, the severity level of all instances are combined (using logical ‘OR’) and represented on the network through the CIP Identity Object.

See also...

- “Diagnostics” on page 18
- “Identity Object (01h)” on page 63 (CIP-object)

9.4 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)

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

Instance Attributes (Instance #1)

Basic

#	Name	Access	Type	Value
1	Network type	Get	UINT16	009Bh
2	Network type string	Get	Array of CHAR	'Ethernet/IP(TM)'
3	Data format	Get	ENUM	00h (LSB first)
4	Parameter data support	Get	BOOL	True
5	Write process data size	Get	UINT16	Current write process data size (in bytes) Updated on every successful Map_ADI_Write_Area ^a
6	Read process data size	Get	UINT16	Current read process data size (in bytes) Updated on every successful Map_ADI_Read_Area ^a
7	Exception Information	Get	UINT8	<u>ValueMeaning</u> 0: No information available 1: Invalid assembly instance mapping 2: Missing MAC address ^b

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

b. Exception information only available to Anybus IP

9.5 Network Configuration Object (04h)

Category

Extended

Object Description

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

If the settings in this object do not match the configuration used, the Module Status LED will flash red to indicate a minor error.

The object is described in further detail in the Anybus CompactCom 40 Software Design Guide.

See also...

- “Communication Settings” on page 16
- “TCP/IP Interface Object (F5h)” on page 87 (CIP-object)
- “Ethernet Link Object (F6h)” on page 90 (CIP-object)
- “E-mail Client” on page 32

Supported Commands

Object: Get_Attribute
 Reset

Instance: Get_Attribute
 Set_Attribute
 Get_Enum_String

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1	Name	Get	Array of CHAR	'Network Configuration'
2	Revision	Get	UINT8	01h
3	Number of instances	Get	UINT16	000Fh (15)
4	Highest instance no.	Get	UINT16	0013h (19)

Instance Attributes (Instance #3, IP Address)

Extended

#	Name	Access	Type	Description
1	Name ^a	Get	Array of CHAR	'IP address'
2	Data type	Get	UINT8	04h (= UINT8)
3	Number of elements	Get	UINT8	04h (four elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of UINT8	Any change is valid after 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)

a. Multilingual, see "Multilingual Strings" on page 107.

Instance Attributes (Instance #4, Subnet Mask)

Extended

#	Name	Access	Type	Description
1	Name ^a	Get	Array of CHAR	'Subnet mask'
2	Data type	Get	UINT8	04h (= UINT8)
3	Number of elements	Get	UINT8	04h (four elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of UINT8	Any change is valid after 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)

a. Multilingual, see "Multilingual Strings" on page 107.

Instance Attributes (Instance #5, Gateway Address)

Extended

#	Name	Access	Type	Description
1	Name ^a	Get	Array of CHAR	'Gateway'
2	Data type	Get	UINT8	04h (= UINT8)
3	Number of elements	Get	UINT8	04h (four elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of UINT8	Any change is valid after 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)

a. Multilingual, see "Multilingual Strings" on page 107.

Instance Attributes (Instance #6, DHCP Enable)

Extended

#	Name	Access	Type	Description
1	Name ^a	Get	Array of CHAR	'DHCP'
2	Data type	Get	UINT8	08h (= ENUM)
3	Number of elements	Get	UINT8	01h (one element)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value ^a	Get/Set	ENUM	Any change is valid after reset. <u>Value:Enum. String:Meaning:</u> 00h 'Disable' DHCP disabled 01h 'Enable' DHCP enabled (default)
6	Configured Value	Get	ENUM	Holds the configured value, which will be written to attribute #5 after the module has been reset. <u>Value:Enum. String:Meaning:</u> 00h 'Disable' DHCP disabled 01h 'Enable' DHCP enabled (default)

a. Multilingual, see "Multilingual Strings" on page 107.

Instance Attributes (Instance #7, Ethernet Communication Settings 1)

Changes have immediate effect.

Extended

#	Name	Access	Type	Description
1	Name ^a	Get	Array of CHAR	'Comm 1'
2	Data type	Get	UINT8	08h (= ENUM)
3	Number of elements	Get	UINT8	01h (one element)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value ^a	Get/Set	ENUM	<u>Value:Enum. String:Meaning:</u> 00h 'Auto' Auto negotiation (default) 01h '10 HDX' 10Mbit, half duplex 02h '10 FDX' 10Mbit, full duplex 03h '100 HDX' 100Mbit, half duplex 04h '100 FDX' 100Mbit, full duplex
6	Configured Value	Get	ENUM	Holds the configured value, which will be written to attribute #5. <u>Value:Enum. String:Meaning:</u> 00h 'Auto' Auto negotiation (default) 01h '10 HDX' 10Mbit, half duplex 02h '10 FDX' 10Mbit, full duplex 03h '100 HDX' 100Mbit, half duplex 04h '100 FDX' 100Mbit, full duplex

a. Multilingual, see "Multilingual Strings" on page 107.

Instance Attributes (Instance #8, Ethernet Communication Settings 2)

Changes have immediate effect.

Extended

#	Name	Access	Type	Description
1	Name ^a	Get	Array of CHAR	'Comm 2'
2	Data type	Get	UINT8	08h (= ENUM)
3	Number of elements	Get	UINT8	01h (one element)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value ^a	Get/Set	ENUM	Value:Enum. String:Meaning: 00h 'Auto' Auto negotiation (default) 01h '10 HDX' 10Mbit, half duplex 02h '10 FDX' 10Mbit, full duplex 03h '100 HDX' 100Mbit, half duplex 04h '100 FDX' 100Mbit, full duplex
6	Configured Value	Get	ENUM	Holds the configured value, which will be written to attribute #5. Value:Enum. String:Meaning: 00h 'Auto' Auto negotiation (default) 01h '10 HDX' 10Mbit, half duplex 02h '10 FDX' 10Mbit, full duplex 03h '100 HDX' 100Mbit, half duplex 04h '100 FDX' 100Mbit, full duplex

a. Multilingual, see "Multilingual Strings" on page 107.

Instance Attributes (Instance #9, DNS1)

This instance holds the address to the primary DNS server.

Extended

#	Name	Access	Type	Description
1	Name ^a	Get	Array of CHAR	'DNS1'
2	Data type	Get	UINT8	04h (= UINT8)
3	Number of elements	Get	UINT8	04h
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of UINT8	Any change is valid after 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)

a. Multilingual, see "Multilingual Strings" on page 107.

Instance Attributes (Instance #10, DNS2)

This instance holds the address to the secondary DNS server.

Extended

#	Name	Access	Type	Description
1	Name ^a	Get	Array of CHAR	'DNS2'
2	Data type	Get	UINT8	04h (= UINT8)
3	Number of elements	Get	UINT8	04h
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of UINT8	Any change is valid after 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)

a. Multilingual, see "Multilingual Strings" on page 107.

Instance Attributes (Instance #11, Host name)

This instance holds the host name of the module.

Extended

#	Name	Access	Type	Description
1	Name ^a	Get	Array of CHAR	'Host name'
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	40h
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of CHAR	Any change is valid after reset. Host name, 64 characters
6	Configured Value	Get	Array of UINT8	Holds the configured value, which will be written to attribute #5 after the module has been reset. Host name, 64 characters

a. Multilingual, see "Multilingual Strings" on page 107.

Instance Attributes (Instance #12, Domain name)

This instance holds the domain name.

Extended

#	Name	Access	Type	Description
1	Name ^a	Get	Array of CHAR	'Domain name'
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	30h
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of CHAR	Any change is valid after 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. Host name, 48 characters

a. Multilingual, see "Multilingual Strings" on page 107.

Instance Attributes (Instance #13, SMTP Server)

This instance holds the SMTP server address.

Extended

#	Name	Access	Type	Description
1	Name ^a	Get	Array of CHAR	'SMTP Server'
2	Data type	Get	UINT8	04h (= UINT8)
3	Number of elements	Get	UINT8	40h
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of CHAR	Any change is valid after 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

a. Multilingual, see "Multilingual Strings" on page 107.

Instance Attributes (Instance #14, SMTP User)

This instance holds user name for the SMTP account.

Extended

#	Name	Access	Type	Description
1	Name ^a	Get	Array of CHAR	'SMTP User'
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	40h
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of CHAR	Any change is valid after 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

a. Multilingual, see "Multilingual Strings" on page 107.

Instance Attributes (Instance #15, SMTP Password)

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

Extended

#	Name	Access	Type	Description
1	Name ^a	Get	Array of CHAR	'SMTP Pswd'
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	40h
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of CHAR	Any change is valid after 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

a. Multilingual, see "Multilingual Strings" on page 107.

Instance Attributes (Instance #16, MDI 1 settings)

This instance holds the settings for MDI/MDIX 1. Changes have immediate effect.

Extended

#	Name	Access	Type	Description
1	Name	Get	Array of CHAR	'MDI 1'
2	Data type	Get	UINT8	08h (= ENUM)
3	Number of elements	Get	UINT8	01h
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	ENUM	<u>Value:Enum. String:Meaning:</u> 00h 'Auto' (default) 01h 'MDI' 02h 'MDIX'
6	Configured Value	Get	ENUM	Holds the configured value, which will be written to attribute #5. <u>Value:Enum. String:Meaning:</u> 00h 'Auto' (default) 01h 'MDI' 02h 'MDIX'

Instance Attributes (Instance #17, MDI 2 settings)

This instance holds the settings for MDI/MDIX 2. Changes have immediate effect.

Extended

#	Name	Access	Type	Description
1	Name	Get	Array of CHAR	'MDI 2'
2	Data type	Get	UINT8	08h (= ENUM)
3	Number of elements	Get	UINT8	01h
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	ENUM	<u>Value:Enum. String:Meaning:</u> 00h 'Auto' (default) 01h 'MDI' 02h 'MDIX'
6	Configured Value	Get	ENUM	Holds the configured value, which will be written to attribute #5. <u>Value:Enum. String:Meaning:</u> 00h 'Auto' (default) 01h 'MDI' 02h 'MDIX'

Instance Attributes (Instances #18 and #19)

These instances are reserved for future attributes.

Instance Attributes (Instance #20, QuickConnect)

This instance enables or disables the QuickConnect functionality from the application. Changes are valid after reset or power cycle. The value of the QuickConnect attribute (#12) in the TCP/IP Interface object (F5h), will change immediately.

QuickConnect has to be enabled in the Ethernet Host object (F9h) for this instance to be available.

See also ...

- “TCP/IP Interface Object (F5h)” on page 87
- “Ethernet Host Object (F9h)” on page 171

Extended

#	Name	Access	Type	Description
1	Name	Get	Array of CHAR	'QuickConnect'
2	Data type	Get	UINT8	08h (= ENUM)
3	Number of elements	Get	UINT8	01h
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	ENUM	<u>Value Meaning:</u> 00h Disable (default) 01h Enable
6	Configured Value	Get	ENUM	Holds the configured value, which will be written to attribute #5. <u>Value Meaning:</u> 00h Disable (default) 01h Enable

Multilingual Strings

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

Instance	English	German	Spanish	Italian	French
3	IP address	IP-Adresse	Dirección IP	Indirizzo IP	Adresse IP
4	Subnet mask	Subnetzmaske	Masac. subred	Sottorete	Sous-réseau
5	Gateway	Gateway	Pasarela	Gateway	Passerelle
6	DHCP	DHCP	DHCP	DHCP	DHCP
	Enable	Einschalten	Activado	Abilitato	Activé
	Disable	Ausschalten	Desactivado	Disabilitato	Désactivé
7	Comm 1	Komm 1	Comu 1	Connessione 1	Comm 1
	Auto	Auto	Auto	Auto	Auto
	10 HDX	10 HDX	10 HDX	10 HDX	10 HDX
	10 FDX	10 FDX	10 FDX	10 FDX	10 FDX
	100 HDX	100 HDX	100 HDX	100 HDX	100 HDX
	100 FDX	100FDX	100 FDX	100 FDX	100 FDX

Instance	English	German	Spanish	Italian	French
8	Comm 2	Komm 2	Comu 2	Conessione 2	Comm 2
	Auto	Auto	Auto	Auto	Auto
	10 HDX	10 HDX	10 HDX	10 HDX	10 HDX
	10 FDX	10 FDX	10 FDX	10 FDX	10 FDX
	100 HDX	100 HDX	100 HDX	100 HDX	100 HDX
	100 FDX	100FDX	100 FDX	100 FDX	100 FDX
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	Nom 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

9.6 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 sent over TCP/UDP.

Note that some of the commands used when accessing this object may require segmentation. For more information, see “Message Segmentation” on page 189.

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

Supported Commands

Object: Get_Attribute
 Create (See “Command Details: Create” on page 111)
 Delete (See “Command Details: Delete” on page 112)

Instance: Get_Attribute
 Set_Attribute
 Bind (See “Command Details: Bind” on page 113)
 Shutdown (See “Command Details: Shutdown” on page 114)
 Listen (See “Command Details: Listen” on page 115)
 Accept (See “Command Details: Accept” on page 116)
 Connect (See “Command Details: Connect” on page 117)
 Receive (See “Command Details: Receive” on page 118)
 Receive_From (See “Command Details: Receive_From” on page 119)
 Send (See “Command Details: Send” on page 120)
 Send_To (See “Command Details: Send_To” on page 121)
 IP_Add_membership (See “Command Details: IP_Add_Membership” on page 122)
 IP_Drop_membership (See “Command Details: IP_Drop_Membership” on page 123)
 DNS_Lookup (See “Command Details: DNS_Lookup” on page 124)

Object Attributes (Instance #0)

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

Instance Attributes (Sockets #1...8)

Extended

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

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

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

Command Details: Create

Category

Extended

Details

Command Code.: 03h

Valid for: Object Instance

Description

This command creates a socket.

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

- **Command Details**

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

- **Response Details**

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

Command Details: Delete

Category

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 “Command Details: Shutdown” on page 114) 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]	<u>Value:Mode:</u> 00h Shutdown receive channel 01h Shutdown send channel 02h Shutdown both receive- and send channel

- **Response Details**

(no data)

The recommended sequence, performed by the application, to gracefully shut down a TCP connection is described below.

Application initiates shutdown:

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

A remote host initiates shutdown:

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

Command Details: Listen

Category

Extended

Details

Command Code.: 12h

Valid for: Instance

Description

This command puts a TCP socket in listening state.

- **Command Details**

Field	Contents	Comments
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.

NON-BLOCKING mode:

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

BLOCKING mode:

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

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

- **Command Details**

(no data)

- **Response Details**

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

Command Details: Connect

Category

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

- **Response Details**

(no data)

Command Details: Receive

Category

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 (see “Message Segmentation” on page 189).

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

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

NON-BLOCKING mode:

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

BLOCKING mode:

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

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

- **Command Details**

Field	Contents	Notes
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control bits	see “Command Segmentation” on page 190
Data[0]	Receive data size (low)	Only used in the first segment
Data[1]	Receive data size (high)	

- **Response Details**

Note: The data in the response may be segmented (see “Message Segmentation” on page 189).

Field	Contents	Notes
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control bits	see “Response Segmentation” on page 191
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 (see “Message Segmentation” on page 189).

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

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

NON-BLOCKING mode:

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

BLOCKING mode:

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

- **Command Details**

Field	Contents	Notes
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control bits	see “Command Segmentation” on page 190
Data[0]	Receive data size (low)	Only used in the first segment
Data[1]	Receive data size (high)	

- **Response Details**

Note: The data in the response may be segmented (see “Message Segmentation” on page 189).

Field	Contents	Notes
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control bits	see “Response Segmentation” on page 191
Data[0]	Host IP address byte 3 (low)	The host address/port information is only included in the first segment. All data thereafter will start at Data[0]
Data[1]	Host IP address byte 2	
Data[2]	Host IP address byte 1	
Data[3]	Host IP address byte 0 (high)	
Data[4]	Host port number (low)	
Data[5]	Host port number (high)	
Data[6...n]	Received data	

Command Details: Send

Category

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 (see “Message Segmentation” on page 189).

NON-BLOCKING mode:

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

BLOCKING mode:

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

- **Command Details**

Note: To allow larger amount of data (i.e. >255 bytes) to be sent, the command data may be segmented (see “Message Segmentation” on page 189).

Field	Contents	Notes
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control	see “Command Segmentation” on page 190
Data[0...n]	Data to send	-

- **Response Details**

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

Command Details: Send_To

Category

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 (see “Message Segmentation” on page 189).

- **Command Details**

Note: To allow larger amount of data (i.e. >255 bytes) to be sent, the command data may be segmented (see “Message Segmentation” on page 189).

Field	Contents	Notes
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control	see “Command Segmentation” on page 190
Data[0]	Host IP address byte 3 (low)	The host address/port information shall only be included in the first segment. All data thereafter must start at Data[0]
Data[1]	Host IP address byte 2	
Data[2]	Host IP address byte 1	
Data[3]	Host IP address byte 0 (high)	
Data[4]	Host port number (low)	
Data[5]	Host port number (high)	
Data[6...n]	Data to send	

- **Response Details**

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

Command Details: IP_Add_Membership

Category

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

- **Response Details**

(no data)

Command Details: IP_Drop_Membership

Category

Extended

Details

Command Code.: 1Ah

Valid for: Instance

Description

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

- **Command Details**

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

- **Response Details**

(no data)

Command Details: DNS_Lookup

Category

Extended

Details

Command Code.: 1Bh

Valid for: Object Instance

Description

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

- **Command Details**

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

- **Response Details (Success)**

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

Socket Interface Error Codes (Object Specific)

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

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

9.7 SMTP Client Object (09h)

Category

Extended

Object Description

This object groups functions related to the SMTP-client.

See also...

- “File System” on page 20
- “E-mail Client” on page 32
- “Instance Attributes (Instance #13, SMTP Server)” on page 104
- “Instance Attributes (Instance #14, SMTP User)” on page 105
- “Instance Attributes (Instance #15, SMTP Password)” on page 105

Supported Commands

Object: Get_Attribute
 Create
 Delete
 Send email from file(“Command Details: Send Email From File” on page 129)

Instance: Get_Attribute
 Set_Attribute
 Send email(“Command Details: Send Email” on page 130)

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

Extended

Instances are created dynamically by the application.

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

Command Details: Create

Category

Extended

Details

Command Code.: 03h

Valid for: Object

Description

This command creates an email instance.

- **Command Details**

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

- **Response Details**

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

Command Details: Delete

Category

Extended

Details

Command Code.: 04h

Valid for: Object

Description

This command deletes an email instance.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	(reserved, ignore)	-
CmdExt[1]		

- **Response Details**

(no data)

Command Details: Send Email From File

Category

Extended

Details

Command Code.: 11h

Valid for: Object

Description

This command sends an email based on a file in the file system.

File format:

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

```
[To]
recipient

[From]
sender

[Subject]
email subject

[Headers]
extra headers, optional

[Message]
actual email message
```

- **Command Details**

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

- **Response Details**

(no data)

Command Details: Send Email

Category

Extended

Details

Command Code.: 10h

Valid for: Instance

Description

This command sends the specified email 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 email file
255	Unspecified SMTP error
(other)	(reserved)

9.8 Anybus File System Interface Object (0Ah)

Category

Extended

Object Description

This object provides an interface to the built-in file system. Each instance represents a handle to a file stream and contains services for file system operations. This provides the host application with access to the built-in file system of the module, e.g. when application specific web pages are to be installed.

Instances are created and deleted dynamically during runtime.

The object is structurally identical to the “Application File System Interface Object (EAh)” on page 174.

Supported Commands

Object:	Get_Attribute Create(“Command Details: Create” on page 133) Delete(“Command Details: Delete” on page 134) Format Disc(“Command Details: Format Disc” on page 143)
Instance:	Get_Attribute File Open(“Command Details: File Open” on page 134) File Close(“Command Details: File Close” on page 135) File Delete(“Command Details: File Delete” on page 135) File Copy(“Command Details: File Copy” on page 136) File Rename(“Command Details: File Rename” on page 137) File Read(“Command Details: File Read” on page 138) File Write(“Command Details: File Write” on page 139) Directory Open(“Command Details: Directory Open” on page 139) Directory Close(“Command Details: Directory Close” on page 140) Directory Delete(“Command Details: Directory Delete” on page 140) Directory Read(“Command Details: Directory Read” on page 141) Directory Create(“Command Details: Directory Create” on page 142) Directory Change(“Command Details: Directory Change” on page 142)

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 no.	Get	UINT16	-
11	Max. no. of instances	Get	UINT16	0004h
12	Disable virtual file system	Get	BOOL	False
13	Total disc size	Get	Array of UINT32	-
14	Free space	Get	Array of UINT32	-
15	Disc CRC	Get	Array of UINT32	-

Instance Attributes

Extended

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

Command Details: Create

Category

Extended

Details

Command Code:: 03h

Valid for: Object

Description

This command creates a file operation instance.

- **Command Details**

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

- **Response Details**

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

Command Details: Delete

Category

Extended

Details

Command Code: 04h

Valid for: Object

Description

This command deletes a file operation instance.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	(reserved, ignore)	-
CmdExt[1]		

- **Response Details**

(no data)

Command Details: File Open

Category

Extended

Details

Command Code: 10h

Valid for: Instance

Description

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

- **Command Details**

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

- **Response Details**

(no data)

Command Details: File Close

Category

Extended

Details

Command Code: 11h

Valid for: Instance

Description

This command closes a previously opened file.

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

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

Command Details: File Delete

Category

Extended

Details

Command Code: 12h

Valid for: Instance

Description

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

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

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

Command Details: File Copy

Category

Extended

Details

Command Code: 13h

Valid for: Instance

Description

This command makes a copy of a file.

- **Command Details**

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

- **Response Details**

(no data)

Command Details: File Rename

Category

Extended

Details

Command Code: 14h

Valid for: Instance

Description

This command renames or moves a file.

- **Command Details**

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

- **Response Details**

(no data)

Command Details: File Read

Category

Extended

Details

Command Code: 15h

Valid for: Instance

Description

Reads data from a file previously opened for reading.

- **Command Details**

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

- **Response Details**

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

Command Details: File Write

Category

Extended

Details

Command Code: 16h

Valid for: Instance

Description

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

- **Command Details**

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

- **Response Details**

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

Command Details: Directory Open

Category

Extended

Details

Command Code: 20h

Valid for: Instance

Description

This command opens a directory.

- **Command Details**

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

- **Response Details**

(no data)

Command Details: Directory Close

Category

Extended

Details

Command Code: 21h

Valid for: Instance

Description

This command closes a previously opened directory.

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

Command Details: Directory Delete

Category

Extended

Details

Command Code: 22h

Valid for: Instance

Description

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

- **Command Details**

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

- **Response Details**
(no data)

Command Details: Directory Read

Category

Extended

Details

Command Code: 23h

Valid for: Instance

Description

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

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

- **Command Details**

(no data)

- **Response Details**

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

Command Details: Directory Create

Category

Extended

Details

Command Code: 24h

Valid for: Instance

Description

This command creates a directory.

- **Command Details**

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

- **Response Details**

(no data)

Command Details: Directory Change

Category

Extended

Details

Command Code: 25h

Valid for: Instance

Description

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

- **Command Details**

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

- **Response Details**

(no data)

Command Details: Format Disc

Category

Extended

Details

Command Code: 30h

Valid for: Object

Description

This command formats the file system.

- **Command Details**

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

- **Response Details**

(no data)

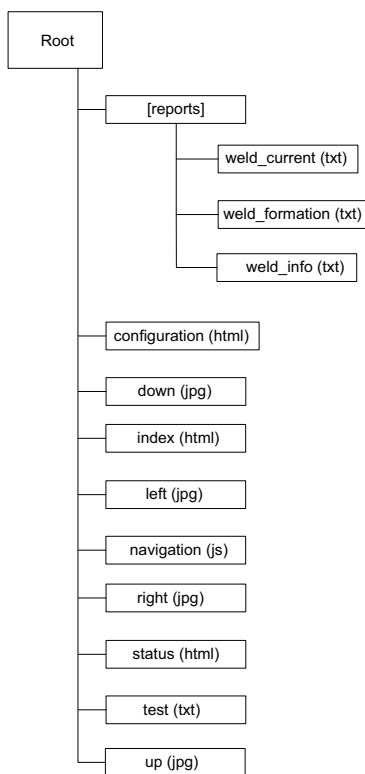
Object Specific Error Codes

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

9.8.1 Examples

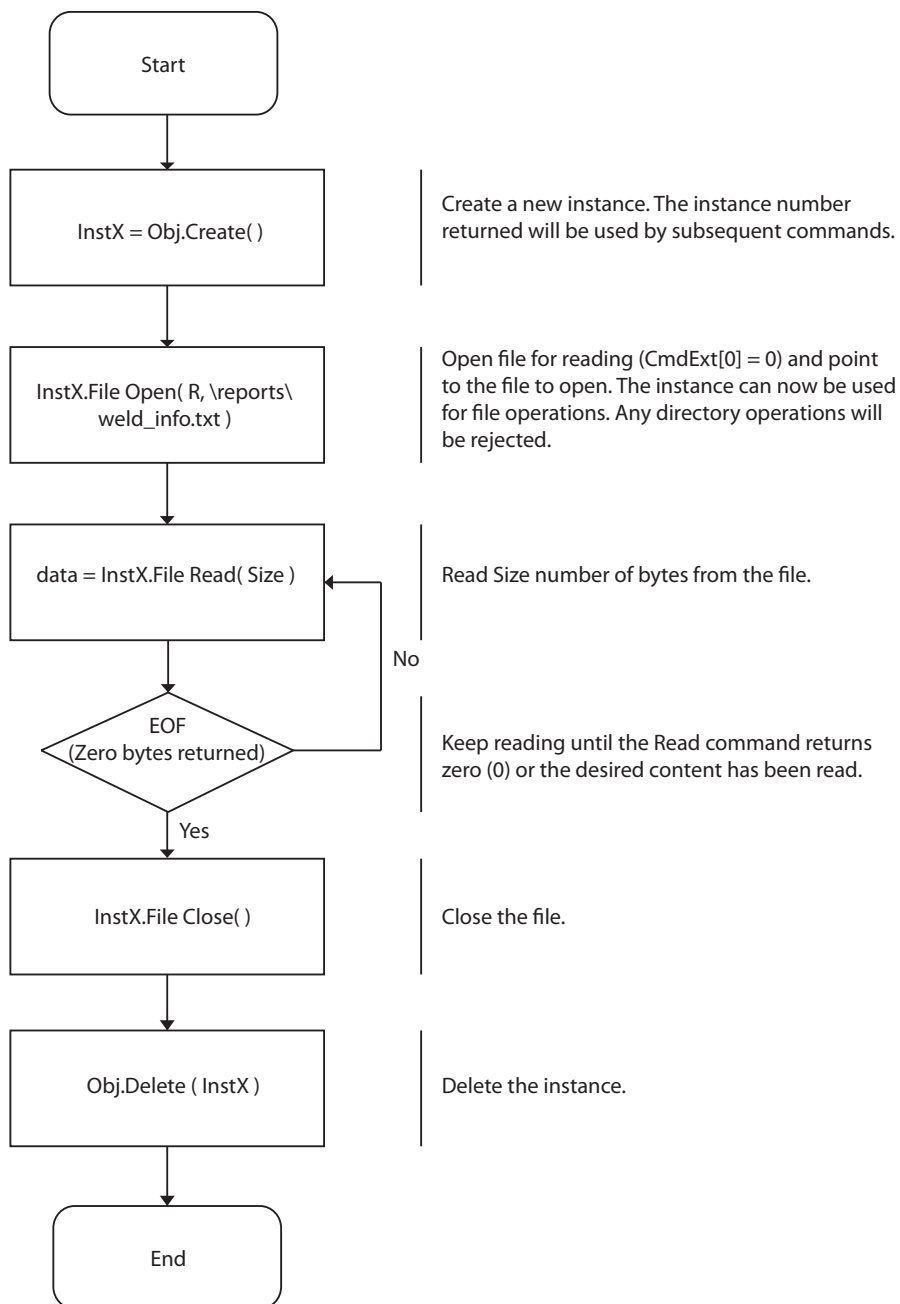
In this section are presented examples for a couple of common cases where the end user would use the File System Interface Object.

An imaginary folder structure will be used in the example, with the following files in the root folder:



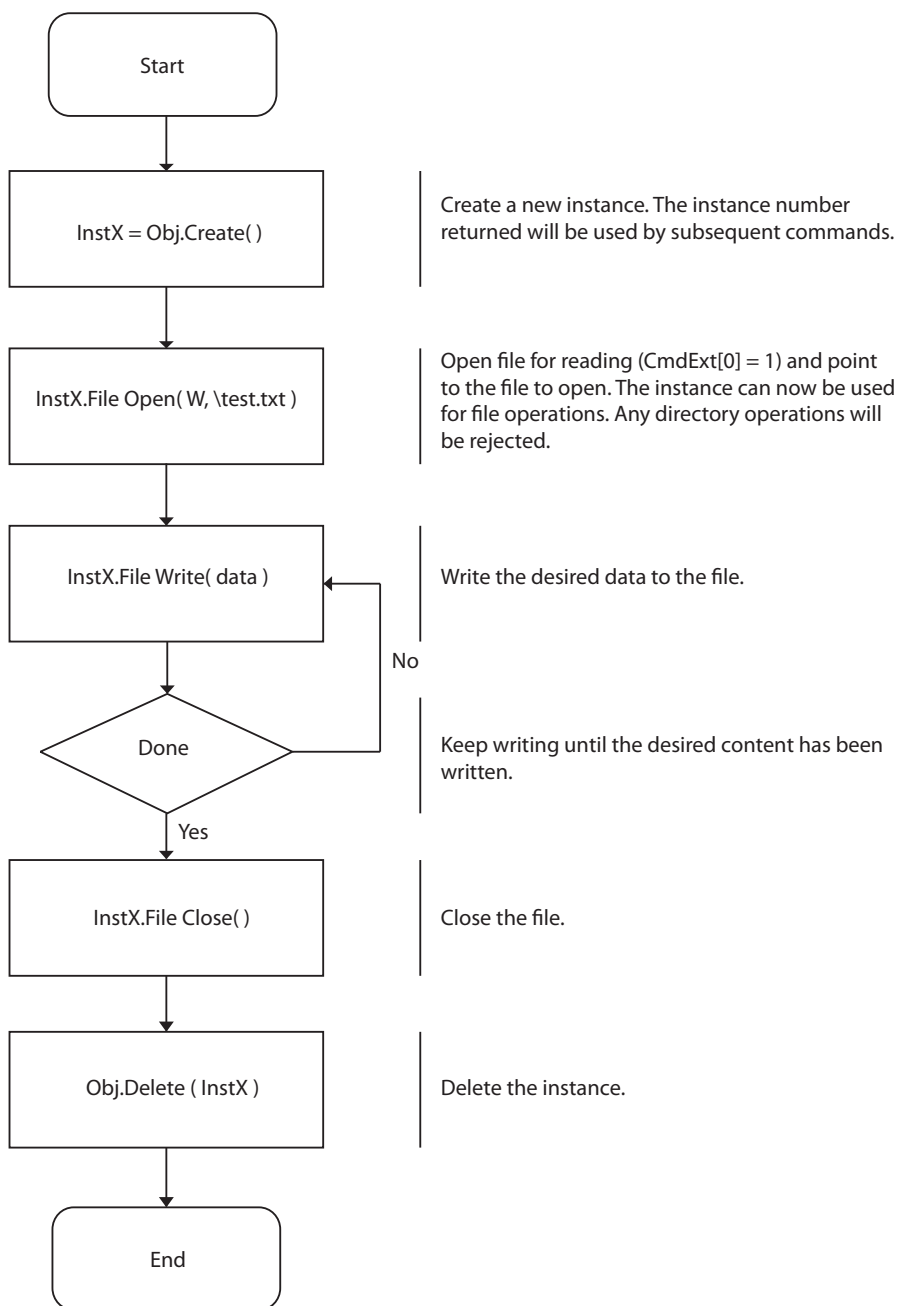
Read a File

The following example opens weld_info.txt in the reports folder and read data from the file.



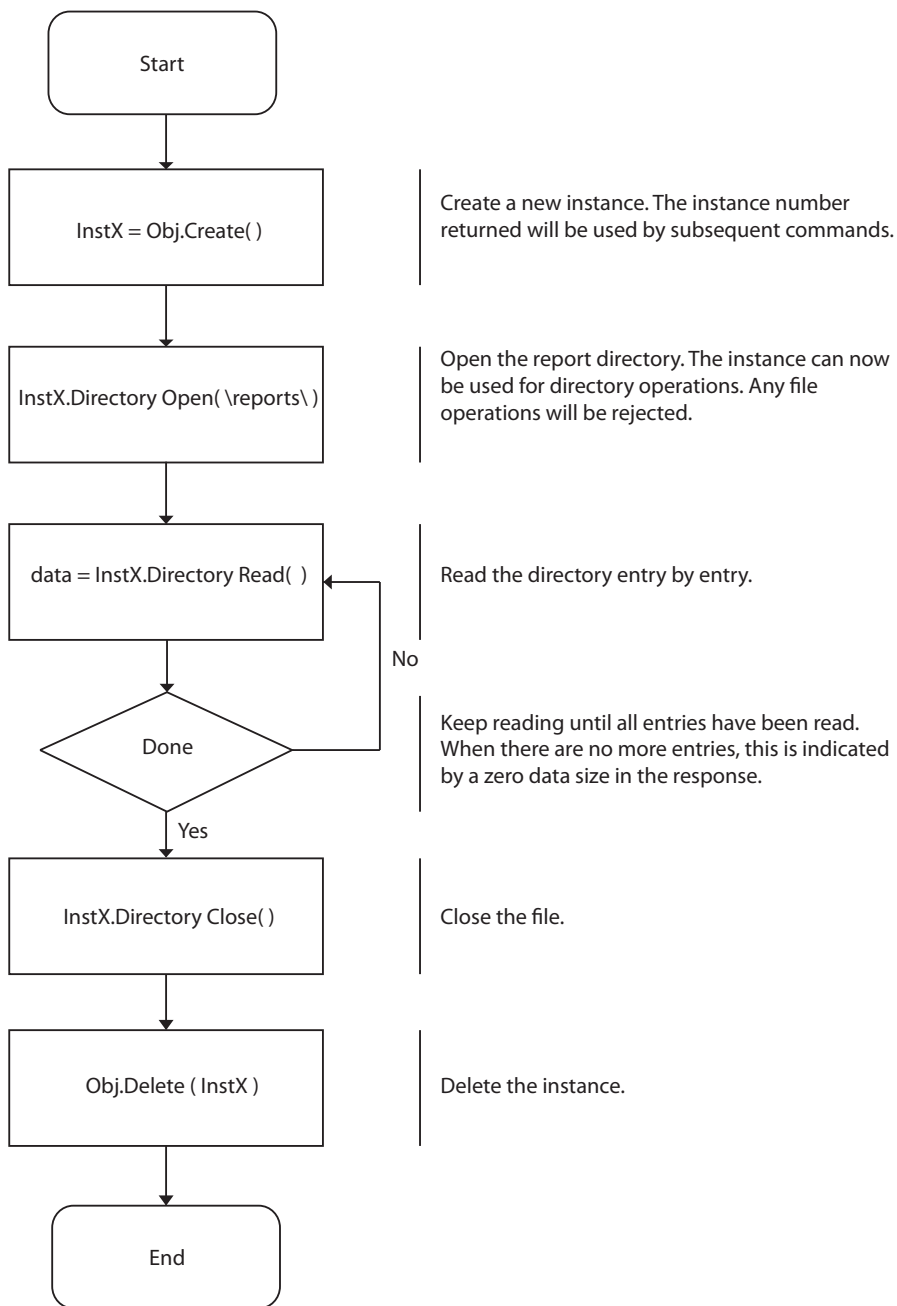
Write a File

The following example opens up the test.txt file for writing.



List Directory Contents

The following example lists the contents of the reports directory



Create a new instance. The instance number returned will be used by subsequent commands.

Open the report directory. The instance can now be used for directory operations. Any file operations will be rejected.

Read the directory entry by entry.

Keep reading until all entries have been read. When there are no more entries, this is indicated by a zero data size in the response.

Close the file.

Delete the instance.

9.9 Network Ethernet Object (0Ch)

Category

Extended

Object Description

This object provides ethernet-specific information to the application.

Supported Commands

Object: Get_Attribute

Instance: Get_Attribute

Object Attributes (Instance #0)

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

Instance Attributes (Instance #1)

Extended

#	Name	Access	Type	Description
1	MAC Address	Get	Array of UINT8	Current MAC address. See also "Ethernet Host Object (F9h)" on page 171)

9.10 Functional Safety Module Object (11h)

Category

Extended

Object Description

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

Supported Commands

Object: Get_Attribute
 Error_Confirmation

Instance: Get_Attribute

Object Attributes (Instance #0)

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

Instance Attributes (Instance #1)

Extended

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

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

a. Values depend on which Safety Module is used.

Exception Information

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

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

Command Details: Error_Confirmation

Category

Extended

Details

Command Code: 10h

Valid for: Object Instance

Description

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

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

Object Specific Error Codes

Error Code	Description	Comments
0x01	The safety module rejected a message.	Error code sent by safety module is found in MsgData[2] and MsgData[3].
0x02	Message response from the safety module has the wrong format (for example, wrong length).	-

9.11 CIP Port Configuration Object (0Dh)

Category

Extended

Object Description

This object is used to populate and enumerate the CIP Port Object (see “Port Object (F4h)” on page 85) on the network side. Basically, this is a matter of creating and updating instances and attributes which shall represent a CIP Port within the host application. This process is necessary in case support for Unconnected CIP Routing has been enabled (see “EtherNet/IP Host Object (F8h)” on page 161, Instance Attribute #17).

Each instance within this object corresponds to an instance in the CIP Port Object. The object supports up to 8 instances, where instance #1 is dedicated to the local TCP port, enabling the host application to implement up to 7 additional ports. Instance #1 will automatically be populated with default values, however it is possible for the host application to customize instance attributes #2 and #4.

Apart from attribute #7, it is possible to write to the instance attributes only during setup. The host application is responsible for keeping instance attribute #7 updated for all ports located within the host application.

See also...

- “Port Object (F4h)” on page 85 (CIP)
- “EtherNet/IP Host Object (F8h)” on page 161 (Instance Attribute #17)

IMPORTANT: *Note that the module does not take over the host application responsibility for error control; the module will not verify that the data set by the host application is correct.*

Supported Commands

Object:	Get_Attribute Create Delete
Instance:	Get_Attribute Set_Attribute

Object Attributes (Instance #0)

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

Instance Attributes

Extended

#	Name	Access	Type	Description
1	Port Type	Set	UINT16	Enumerates the port ^a
2	Port Number	Set	UINT16	CIP port number associated with this port
3	Link Path	Set	Array of UINT8	Logical path segments which identify the object for this port.
4	Port Name	Set	Array of CHAR	String (max. no. of characters is 64) which names the port.
5	-	-	-	(reserved)
6	-	-	-	(reserved)
7	Node Address	Set	Array of UINT8	Node number of this device on port. The data type restricts the range to a Port Segment. The encoded port number must match the value specified in attribute #2. A device which does not have a node number on the port can specify a zero length node address within the Port Segment (i.e. 10h 00h). In case the node address changes during runtime, the host application is responsible for updating this attribute as well.
8	Port Node Range	Set	Struct of: UINT16 (Min) UINT16 (Max)	Minimum and maximum node number on port. Support for this attribute is conditional; the attribute shall be supported provided that the node number can be reported within the range of the data type (e.g. DeviceNet). If not (as is the case with networks such as EtherNet/IP which uses a 4 byte IP address), the attribute shall not be supported.

a. See CIP specification, available from www.odva.org.

See also...

- “Port Object (F4h)” on page 85 (“Instances Attributes (Instance #1)” on page 86)
- “Port Object (F4h)” on page 85 (“Instances Attributes (Instances #2... #8)” on page 86)

10. Host Application Objects

10.1 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/IP implementation.

Standard Objects:

- Application Object (FFh) - (see Anybus CompactCom 40 Software Design Guide)
- Application Data Object (FEh) - (see Anybus CompactCom 40 Software Design Guide)
- Energy Control Object (F0h) - (see Anybus CompactCom 40 Software Design Guide)
- Assembly Mapping Object (EBh) - (see Anybus CompactCom 40 Software Design Guide)
- Modular Device Object (ECh) - (see Anybus CompactCom 40 Software Design Guide)
- “Sync Object (EEh)” on page 160

Network Specific Objects:

- “CIP Identity Host Object (EDh)” on page 158
- “EtherNet/IP Host Object (F8h)” on page 161
- “Ethernet Host Object (F9h)” on page 171
- “Application File System Interface Object (EAh)” on page 174
- “Functional Safety Host Object (E8h)” on page 156

10.2 Functional Safety Host Object (E8h)

Category

Extended

Object Description

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

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

Supported Commands

Object: Get_Attribute

Instance: Get_Attribute

Object Attributes (Instance #0)

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

Instance Attributes (Instance #1)

Extended

#	Name	Access	Type	Description
1	Safety enabled	Get	BOOL	<p>When TRUE, communication with the safety module is enabled.</p> <p>Note: If functional safety is not supported, this attribute must be set to FALSE.</p>
2	Baud Rate	Get	UINT32	<p>Optional attribute^a. Sets the baud rate for the communication between the Anybus CompactCom module and the safety module.</p> <p>Valid values:</p> <ul style="list-style-type: none"> • 625000 (625 kbit/s) • 1000000 (1000 kbit/s) • 1020000 (1020 kbit/s) <p>If not implemented, the default value 1020 kbit/s will be used.</p> <p>If other values are set in this attribute the Anybus CompactCom module will go into Exception state.</p>
3	IO Configuration	Get	Array of UINT8	<p>Optional attribute. Manufacturer specific settings of the digital I/O of the safety module.</p> <p>See the manual of the safety module used for information.</p>
4	Cycle Time	Get	UINT8	<p>Optional attribute^a. Communication cycle time between the Anybus CompactCom and the safety module in milliseconds.</p> <p>Valid values:</p> <ul style="list-style-type: none"> • 2 ms • 4 ms • 8 ms • 16 ms <p>If another value is set, the CompactCom will enter Exception state.</p> <p>If not implemented, the minimum cycle time for the chosen baud rate will be used:</p> <ul style="list-style-type: none"> • 2 ms for 1020 kbit/s • 2 ms for 1000 kbit/s • 4 ms for 625 kbit/s <p>The CompactCom validates the cycle time according to the minimum values above. If e.g. baud rate is 625 kbit/s and the cycle time is set to 2 ms the CompactCom will enter Exception state. If the cycle time is set to 8 ms or 16 ms, any baud rate is valid.</p>

a. The host application shall never implement this attribute when using the IXXAT Safe T100.

10.3 CIP Identity Host Object (EDh)

Category

Extended

Object Description

This object allows for applications to support additional CIP identity instances. It is used to provide additional product identity information, e.g. concerning the software installed.

The first instance in the CIP identity object will not change its behavior. When implementing instances in the CIP identity host object, they will be mapped to the CIP identity object starting at instance 2. Instance no. 1 in the CIP identity host object will be mapped to instance no. 2 in the CIP identity object and so on.

See also...

- “Identity Object (01h)” on page 63 (CIP)

Supported Commands

Object: Get_Attribute

Instance: Get_Attribute
 Get_Attribute_All

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value	Description
1	Name	Get	STRING	'CIP Identity'	Object name
2	Revision	Get	UINT8	01h	Object revision
3	Number of instances	Get	UINT16	Depends on application	Supported number of instances
4	Highest instance no.	Get	UINT16	Depends on application	Highest implemented instance

Instance Attributes (Instance #1)

Extended

#	Name	Access	Type	Default Value	Comment
1	Vendor ID	Get	UINT16	-	These values replace the values for the CIP identity object instance #2 and upwards. See also... - “Identity Object (01h)” on page 63 (CIP-object)
2	Device Type	Get	UINT16	-	
3	Product Code	Get	UINT16	-	
4	Revision	Get	struct of: UINT8 Major UINT8 Minor	-	
5	Status	Get	UNIT16	-	
6	Serial Number	Get	UINT32	-	
7	Product Name	Get	Array of CHAR	-	

Command Details: Get_Attribute_All

Category

Extended

Details

Command Code: 10h

Valid for: Object Instance

Description

This service must be implemented by the application for all instances that exist in the CIP identity host object. If identity data is requested from the network the Anybus module will issue this command to the application. The application will then respond with a message containing a struct of all attributes in the requested instance.

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

Field	Contents	Notes
MsgData[0, 1]	Vendor ID	ABCC CIP identity data
MsgData[2,3]	Device type	
MsgData[4,5]	Product code	
MsgData[6]	Major revision	
MsgData[7]	Minor revision	
MsgData[8,9]	Status	
MsgData[10 ... 13]	Serial number	
MsgData[14 n]	Product name	

10.1 Sync Object (EEh)

Category

Extended

Object Description

The Anybus CompactCom 40 EIP does not support CIP Sync. This object is only used to store the cycle time for the last established IO connection that consumes data.

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

The attributes are represented on EtherNet/IP as follows:

#	Name	Access	Type	Default Value	Comment
1	Cycle time	Get/Set	UINT32		The RPI for the last established IO connection that consumes data (O->T RPI)
2-8	(not implemented)				

10.2 EtherNet/IP Host Object (F8h)

Category

Basic, extended

Object Description

This object implements EtherNet/IP specific features in the host application. Note that this object must not be confused with the Ethernet Host Object, see “Ethernet Host Object (F9h)” on page 171.

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

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

Note that some of the commands used when accessing this object may require segmentation. For more information, see “Message Segmentation” on page 189.

If the module is configured to use EIP QuickConnect functionality, the EDS file has to be changed. As the EDS file is changed, the identity of the module has to be changed and the module will require certification, see “Conformance Test Guide” on page 12.

See also...

- “Identity Object (01h)” on page 63 (CIP)
- “Assembly Object (04h)” on page 67 (CIP)
- “Port Object (F4h)” on page 85 (CIP)
- “CIP Port Configuration Object (0Dh)” on page 153 (Anybus Module Object)
- Anybus CompactCom 40 Software Design Guide, “Error Codes”

Supported Commands

Object:	Get_Attribute Process_CIP_Object_Request (See “Command Details: Process_CIP_Object_Request” on page 166) Set_Configuration_Data (See “Command Details: Set_Configuration_Data” on page 167) Process_CIP_Routing_Request (See “Command Details: Process_CIP_Routing_Request” on page 169) Get_Configuration_Data (See “Command Details: Get_Configuration_Data” on page 170)
Instance:	Get_Attribute

Object Attributes (Instance #0)

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

Instance Attributes (Instance #1)

Basic

#	Name	Access	Type	Default Value	Comment
1	Vendor ID	Get	UINT16	005Ah	These values are set in the Identity Object (CIP) at startup.
2	Device Type	Get	UINT16	002Bh	
3	Product Code	Get	UINT16	0037h	
4	Revision	Get	struct of: UINT8 Major UINT8 Minor	(software revision)	See also... - "Device Customization" on page 13 - "Identity Object (01h)" on page 63 (CIP-object)
5	Serial Number	Get	UINT32	(set at production)	Note: Changing any of these attributes requires a new Vendor ID.
6	Product Name	Get	Array of CHAR	'Anybus CompactCom 40 EtherNet/IP(TM)'	

Extended

#	Name	Access	Type	Default Value	Comment
7	Producing Instance No.	Get	Array of UINT16	-	The values in this array are the EtherNet/IP Assembly instance numbers that matches the host application Assembly Mapping Object instances that are listed in attribute #11 (Write PD Instance List). If the Assembly Mapping Object is not implemented, one element in this array is allowed, to set the producing instance number. The maximum number of entries in the array is 6. See "Multiple Assembly Instances" on page 165 for an example.
8	Consuming Instance No.	Get	Array of UINT16	-	The values in this array are the EtherNet/IP Assembly instance numbers that matches the host application Assembly Mapping Object instances that are listed in attribute #12 (Read PD Instance List). If the Assembly Mapping Object is not implemented, one element in this array is allowed, to set the consuming instance number. See "Multiple Assembly Instances" on page 165 for an example. The maximum number of entries in the array is 6.

#	Name	Access	Type	Default Value	Comment
9	Enable communication settings from Net	Get	BOOL	True	<u>Value:Meaning:</u> True Can be set from network False Cannot be set from network See also... - "TCP/IP Interface Object (F5h)" on page 87 (CIP-object) - "Ethernet Link Object (F6h)" on page 90 (CIP-object) - "Network Configuration Object (04h)" on page 99 (Anybus Module Object)
11	Enable CIP forwarding	Get	BOOL	False	<u>Value:Meaning:</u> True Requests to unknown CIP objects and unknown assembly object instances are routed to the application. False Requests to unknown CIP objects and unknown assembly object instances are not routed to the application. See also... - "Command Details: Process_CIP_Object_Request" on page 166.
12	Enable Parameter Object	Get	BOOL	True	<u>Value:Meaning:</u> True Enable CIP Parameter Object False Disable CIP Parameter Object
13	Input-Only heartbeat instance number	Get	UINT16	0003h	See also... - "Instance 03h Attributes (Heartbeat, Input-Only)" on page 67 (CIP-instance)
14	Listen-Only heartbeat instance number	Get	UINT16	0004h	See also... - "Instance 04h Attributes (Heartbeat, Listen-Only)" on page 68 (CIP-instance)
15	Assembly object Configuration instance number	Get	UINT16	0005h	See also... - "Instance 05h Attributes (Configuration Data)" on page 68 (CIP-instance)
16	Disable Strict IO Match	Get	BOOL	False	If true, the module will accept Class1 connection requests that have sizes that's less than or equal to the configured IO sizes.
17	Enable unconnected routing	Get	BOOL	False	If true, the module enables unconnected CIP routing. This also triggers an initial upload of the contents of the CIP Port Mapping object.
18	Input-Only extended heartbeat instance number	Get	UINT16	0006h	See also... - "Instance 06h Attributes (Heartbeat, Input-Only Extended)" on page 68 (CIP-instance)
19	Listen-Only extended heartbeat instance number	Get	UINT16	0007h	See also... - "Instance 07h Attributes (Heartbeat, Listen-Only Extended)" on page 69 (CIP-instance)
20	Interface label port 1	Get	Array of CHAR	Port 1	The value of this attribute is used to change the interface label for Ethernet Link Object Instance #1
21	Interface label port 2	Get	Array of CHAR	Port 2	The value of this attribute is used to change the interface label for Ethernet Link Object Instance #2
22	Interface label internal port	Get	Array of CHAR	Internal	The value of this attribute is used to change the interface label for Ethernet Link Object Instance #3

#	Name	Access	Type	Default Value	Comment
26	Enable EtherNet/IP QuickConnect ^a	Get	BOOL	False	<u>Value:Meaning:</u> True EtherNet/IP QuickConnect functionality enabled. False EtherNet/IP QuickConnect functionality disabled.
29	Ignore Sequence Count Check	Get	BOOL	False	Setting this attribute to "true" makes the module ignore the Sequence Count Check for consumed Class 1 data. This means that all data, not just changed/new data, received from the Originator, will be copied to the application. Copying all data and not just changed data is a violation of the CIP specification. It will also affect the performance of the module. Use precaution when setting this flag to "true". HMS will do NO performance measurements and states NO guarantees about how performance will be affected when copying all data.
30	ABCC ADI Object Number	Get	UINT16	00A2h	This attribute either changes the object number of the ADI Object or disables the ADI Object (see page 83). Valid object numbers are within the vendor specific ranges (0064h - 00C7h and 0300h - 04FFh). Any other value will disable the ADI object.
31	Enable DLR	Get	BOOL	True	True = DLR functionality enabled False= DLR functionality disabled

- a. If the module is configured to use EIP QuickConnect functionality, the EDS file has to be changed. As the EDS file is changed, the identity of the module has to be changed and the module will require certification, see "Conformance Test Guide" on page 12.

Multiple Assembly Instances

The Assembly Mapping Object has two arrays on class level (Write PD Instance List and Read PD Instance List) listing instances defined by the application. The arrays of attributes 7 and 8 in the EtherNet/IP host object (Producing Instance Number and Consuming Instance number) are bound to the instance lists in the Assembly Mapping Object. The arrays list the corresponding CIP instance numbers representing each assembly instance defined by the application.

The example below shows how the EtherNet/IP assembly instances are bound to host application assembly instances.

The length of the arrays must match, otherwise the module enters exception.

Assembly Mapping Object (EBh) Instances	
1	Read PD
2	Read PD
10	Write PD
11	Write PD
100	Read PD
101	Write PD

Assembly Mapping Object Attribute	Value		Value	EtherNet/IP Host Object Instance Attribute
11 - Write PD Instance List	1	<--->	70	7 - Producing Instance Number
	2	<--->	71	
	100	<--->	150	
12 - Read PD Instance List	10	<--->	20	8 - Consuming Instance Number
	11	<--->	21	
	101	<--->	100	

See also ...

- Assembly Mapping Object (see Anybus CompactCom 40 Software Design Guide)

Command Details: Process_CIP_Object_Request

Category

Extended

Details

Command Code: 10h

Valid for: Object Instance

Description

By setting the 'Enable CIP Request Forwarding'-attribute (#11), all requests to unimplemented CIP-objects and unknown assembly object instances, will be forwarded to the host application through this command. The application then has to evaluate the request and return a proper response. The module supports one CIP-request; additional requests will be rejected by the module.

Note that since the telegram length on the host interface is limited, the request data size must not exceed 255 bytes. If it does, the module will send a 'resource unavailable' response to the originator of the request and the message will not be forwarded to the host application.

Note: This command is similar - but not identical - to the 'Process_CIP_Request'-command in the Anybus CompactCom 40 DeviceNet.

- **Command Details**

Field	Contents	Notes
CmdExt[0]	CIP Service Code	CIP service code from original CIP request
CmdExt[1]	Request Path Size	Number of 16-bit words in the Request Path field
MsgData[0... m]	Request Path	CIP EPATH (Class, Instance, Attr. etc.)
MsgData[m... n]	Request Data	Service-specific data

- **Response Details**

Field	Contents	Notes
CmdExt[0]	CIP Service Code	(Reply bit set)
CmdExt[1]	00h	(reserved, set to zero)
MsgData[0]	General Status	CIP General Status Code
MsgData[1]	Size of Additional Status	Number of 16-bit words in Additional Status array
MsgData[2... m]	Additional Status	Additional Status, if applicable
MsgData[m... n]	Response data	Actual response data, if applicable

IMPORTANT: *When using this functionality, make sure to implement the common CIP Class Attribute (attribute #1, 'Revision') for all objects in the host application firmware. Failure to observe this will prevent the module from successfully passing conformance tests.*

Command Details: Set_Configuration_Data

Category

Extended

Details

Command Code: 11h

Valid for: Object Instance

Description

If the data segment in the CIP 'Forward_Open' service contains Configuration Data, this will be forwarded to the host application through this command. If implemented, the host application should evaluate the request and return a proper response.

Segmentation is used, see "Message Segmentation" on page 189 for more information. The maximum total amount of configuration data that will be accepted by the module is 458 bytes.

Note: This command must be implemented in order to support Configuration Data. If not implemented, the CIP 'Forward_Open'-request will be rejected by the module.

- **Command Details**

Field	Contents	Notes
CmdExt[0]	-	(reserved, ignore)
CmdExt[1]	Segmentation Control bits	See "Message Segmentation" on page 189
MsgData[0 - 1] ^a	Producing connection point	Producing connection point, requested by the originator.
MsgData[2 - 3] ^a	Consuming connection point	Consuming connection point, requested by the originator.
MsgData[0... n]	Data	Actual configuration data

- a. MsgData[0 - 1] and MsgData[2 - 3] can both be 0. Normally, the Set_Configuration_Data command is sent to the application when an I/O connection is setup on the network. Producing connection point and consuming connection point are available and will be forwarded with the command. But if the configuration data originates from a set attribute single request or a not matching NULL forward open request, there is no information on the connection points and 0 (zero) will be forwarded to the application.

- **Response Details (Success)**

Field	Contents	Notes
CmdExt[0]	00h	(reserved, set to zero)
CmdExt[1]	00h	(reserved, set to zero)

- **Response Details (Error)**

Field	Contents	Notes
CmdExt[0]	00h	(reserved, set to zero)
CmdExt[1]	00h	(reserved, set to zero)
MsgData[0]	Error code	Anybus error code
MsgData[1]	Extended error code	If the Anybus error code is set to FFh, the extended error code shall be translated as shown in "Extended Error Code" on page 168.
MsgData[2 - 3]	Index	If the Extended error code is set to 02h (invalid configuration), this parameter points to the attribute that failed.

Extended Error Code

If the Error code equals FFh (Object specific error), the extended code will be translated as below:

Code	Contents	CIP no.	CIP status code	Additional Information
01h	Ownership conflict	01h	Connection failure	The configuration data was supplied in a forward open request.
		10h	Device State conflict	The configuration data was supplied in a set request to the Assembly object.
02h	Invalid configuration	09h	Bad attribute data	CIP extended error code: Use value from MsgData[2 - 3]. The extended error code shall only be used if the request originated from a Forward Open request, not for explicit set requests.

See also...

- “Connection Manager (06h)” on page 70 (CIP)
- “Message Segmentation” on page 189

Command Details: Process_CIP_Routing_Request

Category

Extended

Details

Command Code: 12h

Valid for: Object Instance

Description

The module will strip the first path within the Unconnected_Send service and evaluate whether or not it's possible to continue with the routing (e.g. check that the requested port exists within the port object). If the stripped path was the last path the contents delivered to the application will be the CIP request sent to the destination node, otherwise it will be an Unconnected_Send service with updated route path information.

The module supports one pending request. Additional requests will be rejected by the module.

Note: Since the telegram length on the host interface is limited, the data must not exceed 255 bytes in length. If it does, the module will reject the originator of the request ("Resource unavailable"), and this command will not be issued towards the host application.

- **Command Details**

Field	Contents	Notes
CmdExt[0]	-	(reserved, ignore)
CmdExt[1]	-	(reserved, ignore)
MsgData[0... n]	Destination Path	Destination path encoded as an EPATH.
MsgData[n+1]	Time_tick	Valid after timeout parameters have been updated
MsgData[n+2]	Time-out_ticks	Valid after timeout parameters have been updated
MsgData[n+3... m]	CIP message	CIP message to route

- **Response Details**

Field	Contents	Notes
CmdExt[0]	00h	(reserved, set to zero)
CmdExt[1]	00h	(reserved, set to zero)
MsgData[0]	CIP Service	Actual CIP service code, response bit set
MsgData[1]	00h	(reserved, set to zero)
MsgData[2]	General Status	Actual CIP General status code
MsgData[3]	Size of Additional Status	No. of 16-bit words in Additional Status Array
MsgData[4... n]	Additional Status Array	Additional status, if applicable
MsgData[n+1... m]	Response Data	Actual response data

See also...

- "Port Object (F4h)" on page 85 (CIP)
- "CIP Port Configuration Object (0Dh)" on page 153

Command Details: Get_Configuration_Data

Details

Command Code: 13h

Valid for: Object Instance

Description

If the configuration data is requested from the network, the Anybus will issue this command to the application. The application shall send the stored configuration data in the response message.

Segmentation is used since the telegram length on the host interface is limited. The maximum total amount of configuration data that will be accepted by the module is 458 bytes.

Note: This command must be implemented in order to support Configuration Data. If not implemented, the request will be rejected by the Anybus module.

- **Command Details**

Field	Contents	Notes
CmdExt[0]	00h	-
CmdExt[1]	00h	-
MsgData[0... n]	-	No extended message data

- **Response Details (Success)**

Field	Contents	Notes
CmdExt[0]	00h	(reserved, set to zero)
CmdExt[1]	Segmentation Control bits	See "Message Segmentation" on page 189
MsgData[0 - n]	Status	Configuration data from the application

- **Response Details (Error)**

Field	Contents	Notes
CmdExt[0]	00h	(reserved, set to zero)
CmdExt[1]	Segmentation Control bits	See "Message Segmentation" on page 189
MsgData[0]	Status	Anybus protocol error code

See also...

"Message Segmentation" on page 189

10.3 Ethernet Host Object (F9h)

Category

Basic, extended

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)

Basic

#	Name	Access	Type	Default ^a	Comment
1	MAC address ^b	Get	Array of UINT8	-	6 byte physical address value; overrides the pre-programmed Mac address. Note that the new Mac address value must be obtained from the IEEE.

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

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

Extended

#	Name	Access	Type	Default ^a	Comment
2	Enable HICP	Get	BOOL	True	<u>Value:Meaning:</u> True HICP enabled False HICP disabled (see "Secure HICP (Secure Host IP Configuration Protocol)" on page 192)
3	Enable Web Server	Get	BOOL	True	<u>Value:Meaning:</u> True web server enabled False web server disabled (see "Web Server" on page 24)
5	Enable Web ADI access	Get	BOOL	True	<u>Value:Meaning:</u> True web ADI access enabled False web ADI access disabled (see "Web Server" on page 24)
6	Enable FTP server	Get	BOOL	True	<u>Value:Meaning:</u> True FTP server enabled False FTP server disabled (see "FTP Server" on page 22)
7	Enable admin mode	Get	BOOL	False	<u>Value:Meaning:</u> True FTP Admin mode enabled False FTP Admin mode disabled (see "FTP Server" on page 22)
8	Network Status	Set	UINT16	-	See "Network Status" on page 173
9	Port 1 MAC address	Get	Array of UINT8	-	MAC address for Ethernet port 1, 6 bytes
10	Port 2 MAC address	Get	Array of UINT8	-	MAC address for Ethernet port 2, 6 bytes
11 ^b	Enable ACD	Get	BOOL	True	<u>Value:Meaning:</u> True ACD enabled False ACD disabled
12	Port 1 State	Get	ENUM	Enable	State of Ethernet port 1, see "Port State" on page 173
13	Port 2 State	Get	ENUM	Enable	State of Ethernet port 2, see "Port State" on page 173
14	Reserved				
15	Enable reset from HICP	Get	BOOL	False	<u>Value:Meaning:</u> True Possible to reset the module from HICP False Not possible to reset the module from HICP
16	IP configuration	Set	Struct of: UINT32 (IP address) UINT32 (Subnet mask) UINT32 (Gateway)	N/A	The Anybus CompactCom writes the IP configuration (IP address, Subnet mask, Gateway) to this attribute whenever the configuration is assigned or changed.
17	IP address byte 0 - 2	Get	Array of UINT8[3]	[0] : 192 [1] : 168 [2] : 0	This attributes holds the first three bytes of the IP address. The attribute is used in Shift Register Mode if the configuration switch value is set to 1 - 245. The first three bytes of the IP address will be given by the values in the attribute and the last byte will be given by the configuration switch value.

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

b. If ACD functionality is disabled using this attribute, the ACD attributes in the CIP TCP/IP object (F5h) are not available.

Network Status

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

Bit	Contents	Description
0	Link	<u>Value:Meaning:</u> True Link detected False No link
1	IP in use	<u>Value:Meaning:</u> True IP address in use (no address conflict detected) False No IP address in use
2	IP conflict	<u>Value:Meaning:</u> True IP address conflict detected False No IP address conflict detected
3	Link port 1	<u>Value:Meaning:</u> True Valid link on port 1 False No valid link on port 1
4	Link port 2	<u>Value:Meaning:</u> True Valid link on port 2 False No valid link on port 2
5... 15	(reserved)	(mask off and ignore)

Port State

The attributes Port 1 State and Port 2 State tells whether the corresponding port is enabled, disabled or inactivated.

Value	State	Description
00h	Enable	The Ethernet port is enabled.
01h	Disable	The Ethernet port is disabled. The port will be treated as existing, i.e. references to the port can exist (in network protocol, website etc.).
02h	Inactivate	The Ethernet port is inactivated The port will be treated as non-existing, i.e. no references to the port will exist (in network protocol, website etc.). Modules with two ports will disable functionality requiring two ports if one port is set to this state. NOTE: This state is only valid to use for Port 2 State (attribute #13). The module will enter exception with exception code 0x07 (Invalid application response), if Port 1 State is configured to this state. NOTE: A port shall only be configured to this state if it does not exist physically. If using an M40 module with two Ethernet ports, use state 01h (Disable) instead.

If any of these attributes are implemented, the admin state attribute (#9) in the CIP Ethernet Link object (F6h) will not be available, see page 90.

10.4 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.

Instances are created and deleted dynamically during runtime.

The object is structurally identical to the “Anybus File System Interface Object (0Ah)” on page 131.

Supported Commands

Object:	Get_Attribute Create(“Command Details: Create” on page 176) Delete(“Command Details: Delete” on page 177)
Instance:	Get_Attribute File Open(“Command Details: File Open” on page 177) File Close(“Command Details: File Close” on page 178) File Delete(“Command Details: File Delete” on page 178) File Copy(“Command Details: File Copy” on page 179) File Rename(“Command Details: File Rename” on page 180) File Read(“Command Details: File Read” on page 181) File Write(“Command Details: File Write” on page 182) Directory Open(“Command Details: Directory Open” on page 182) Directory Close(“Command Details: Directory Close” on page 183) Directory Delete(“Command Details: Directory Delete” on page 183) Directory Read(“Command Details: Directory Read” on page 184) Directory Create(“Command Details: Directory Create” on page 185) Directory Change(“Command Details: Directory Change” on page 185)

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1	Name	Get	Array of CHAR	'Application File System Interface'
2	Revision	Get	UINT8	01h
3	Number of instances	Get	UINT16	-
4	Highest instance no.	Get	UINT16	-
11	Max. no. of instances	Get	UINT16	Max number of instances supported by the application.
13	Total disc size	Get	Array of UINT32	-
14	Free space	Get	Array of UINT32	-

Instance Attributes

Extended

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

Command Details: Create

Category

Extended

Details

Command Code.: 03h

Valid for: Object

Description

This command creates a file operation instance.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	(reserved, ignore)	-
CmdExt[1]		

- **Response Details**

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

Command Details: Delete

Category

Extended

Details

Command Code.: 04h

Valid for: Object

Description

This command deletes a file operation instance.

- **Command Details**

Field	Contents	Comments
CmdExt[0]	(reserved, ignore)	-
CmdExt[1]		

- **Response Details**

(no data)

Command Details: File Open

Category

Extended

Details

Command Code.: 10h

Valid for: Instance

Description

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

- **Command Details**

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

- **Response Details**

(no data)

Command Details: File Close

Category

Extended

Details

Command Code.: 11h

Valid for: Instance

Description

This command closes a previously opened file.

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

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

Command Details: File Delete

Category

Extended

Details

Command Code.: 12h

Valid for: Instance

Description

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

- **Command Details**

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

- **Response Details**
(no data)

Command Details: File Copy

Category

Extended

Details

Command Code.: 13h

Valid for: Instance

Description

This command makes a copy of a file.

- **Command Details**

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

- **Response Details**

(no data)

Command Details: File Rename

Category

Extended

Details

Command Code.: 14h

Valid for: Instance

Description

This command renames or moves a file.

- **Command Details**

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

- **Response Details**

(no data)

Command Details: File Read

Category

Extended

Details

Command Code.: 15h

Valid for: Instance

Description

Reads data from a file previously opened for reading.

- **Command Details**

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

- **Response Details**

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

Command Details: File Write

Category

Extended

Details

Command Code.: 16h

Valid for: Instance

Description

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

- **Command Details**

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

- **Response Details**

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

Command Details: Directory Open

Category

Extended

Details

Command Code.: 20h

Valid for: Instance

Description

This command opens a directory.

- **Command Details**

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

- **Response Details**

(no data)

Command Details: Directory Close

Category

Extended

Details

Command Code.: 21h

Valid for: Instance

Description

This command closes a previously opened directory.

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

Command Details: Directory Delete

Category

Extended

Details

Command Code.: 22h

Valid for: Instance

Description

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

- **Command Details**

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

- **Response Details**
(no data)

Command Details: Directory Read

Category

Extended

Details

Command Code.: 23h

Valid for: Instance

Description

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

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

- **Command Details**

(no data)

- **Response Details**

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

Command Details: Directory Create

Category

Extended

Details

Command Code.: 24h

Valid for: Instance

Description

This command creates a directory.

- **Command Details**

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

- **Response Details**

(no data)

Command Details: Directory Change

Category

Extended

Details

Command Code.: 25h

Valid for: Instance

Description

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

- **Command Details**

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

- **Response Details**

(no data)

Object Specific Error Codes

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

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) indicates that the network participation is supervised by another network device. In the case of EtherNet/IP, this means that the SUP-bit is set when one or more CIP (Class 1 or Class 3) connections has been opened towards the module.

B.2 Anybus Statemachine

The table below describes how the Anybus Statemachine relates to the EtherNet/IP network.

Anybus State	Implementation	Comment
WAIT_PROCESS	The module stays in this state until a Class 1 connection has been opened.	-
ERROR	1. Class 1 connections errors 2. Duplicate IP address detected	-
PROCESS_ACTIVE	Error free Class 1 connection active (RUN-bit set in the 32-bit Run/Idle header of an Exclusive-Owner connection)	Only valid for consuming connections.
IDLE	Class 1 connection idle.	
EXCEPTION	Unexpected error, e.g. watchdog timeout etc.	MS LED turns red (to indicate a major fault) NS LED is off

B.3 Application Watchdog Timeout Handling

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

C. Message Segmentation

C.1 General

Category: Extended

The maximum message size supported by the Anybus CompactCom 40 is 1524 bytes. If the host application implements a data message size of 1524 bytes, a message will always fit into one segment. The host application can implement a shorter data message size (255 bytes for backwards compatibility with the 30-series).

No service requires messages larger than what is supported by the Anybus CompactCom 40 series 1524 bytes messaging interface. If this interface is used by the application, it allows very basic segmentation handling. The first segment bit (FS) and the last segment bit (LS) shall always be set in each segmented command or response. In the Anybus CompactCom 40 series, some commands in the Socket Interface Object (page 109) and in the EtherNet/IP Host Object (161) use segmentation.

If a shorter message size is implemented, segmentation has to be used, setting the FS bit in the first segment of the message sent, and setting the LS bit in the last segment sent.

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

The module supports 20 simultaneous segmented messages.

C.2 Command Segmentation

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

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

Command segmentation is used for the following commands:

- Set_Configuration_Data (see “Command Details: Set_Configuration_Data” on page 167)
- Send (see “Command Details: Send” on page 120)
- Send To (see “Command Details: Send_To” on page 121)

Segmentation Control bits (Command)

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

Segmentation Control bits (Response)

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

When issuing a segmented command, the following rules apply:

- When issuing the first segment, FS must be set.
- When issuing subsequent segments, both FS and LS must be cleared.
- When issuing the last segment, the LS-bit must be set.
- For single segment commands (i.e. size less or equal to 255 bytes), both FS and LS must be set.
- The last response message contains the actual result of the operation.
- The command initiator may at any time abort the operation by issuing a message with AB set.
- If a segmentation error is detected during transmission, an error message is returned, and the current segmentation message is discarded. Note however that this only applies to the current segment; previously transmitted segments are still valid.

C.3 Response Segmentation

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

Response segmentation is used for responses to the following commands:

- Receive (object specific, see “Command Details: Receive” on page 118)
- Receive From (object specific, see “Command Details: Receive_From” on page 119)
- Get_Configuration_Data (see “Command Details: Get_Configuration_Data” on page 170)

Segmentation Control bits (Command)

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

Segmentation Control bits (Response)

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

When receiving a segmented response, the following rules apply:

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

D. Secure HICP (Secure Host IP Configuration Protocol)

D.1 General

The module 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.

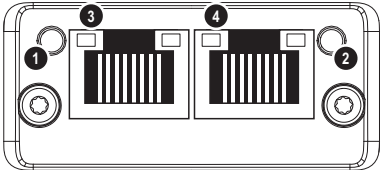
The protocol offers secure authentication and the ability to restart/reboot the device(s).

E. Technical Specification

E.1 Front View

Ethernet Connector

#	Item
1	Network Status LED ^a
2	Module Status LED ^a
3	Link/Activity LED (port 1)
4	Link/Activity LED (port 2)



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

Network Status LED

Note: A test sequence is performed on this LED during startup.

LED State	Description
Off	No power or no IP address
Green	Online, one or more connections established (CIP Class 1 or 3)
Green, flashing	Online, no connections established
Red	Duplicate IP address, FATAL error
Red, flashing	One or more connections timed out (CIP Class 1 or 3)

Module Status LED

Note: A test sequence is performed on this LED during startup.

LED State	Description
Off	No power
Green	Controlled by a Scanner in Run state
Green, flashing	Not configured, or Scanner in Idle state
Red	Major fault (EXCEPTION-state, FATAL error etc.)
Red, flashing	Recoverable fault(s). Module is configured, but stored parameters differ from currently used parameters.

LINK/Activity LED 3/4

LED State	Description
Off	No link, no activity
Green	Link (100 Mbit/s) established
Green, flickering	Activity (100 Mbit/s)
Yellow	Link (10 Mbit/s) established
Yellow, flickering	Activity (10 Mbit/s)

Ethernet Interface

The Ethernet interface supports 10/100Mbit, full or half duplex operation.

E.2 Protective Earth (PE) Requirements

In order to ensure proper EMC behaviour, the module must be properly connected to protective earth via the PE pad / PE mechanism described in the general Anybus CompactCom 40 Hardware Design Guide.

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

E.3 Power Supply

Supply Voltage

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

Power Consumption

The Anybus CompactCom 40 EtherNet/IP 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.

The current hardware design consumes up to 360 mA¹.

Note: It is strongly advised to design the power supply in the host application based on the power consumption classifications described in the general Anybus CompactCom Hardware Design Guide, and not on the exact power requirements of a single product.

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.

-
1. Note that in line with HMS policy of continuous product development, we reserve the right to change the exact power requirements of this product without prior notification. Note however that in any case, the Anybus CompactCom 40 EtherNet/IP will remain as a Class B module.

F. Timing & Performance

F.1 General Information

This chapter specifies timing and performance parameters that are verified and documented for the Anybus CompactCom 40 EtherNet/IP.

The following timing aspects are measured:

Category	Parameters	Page
Startup Delay	T1, T2	195
NW_INIT Handling	T100	195
Event Based WrMsg Busy Time	T103	196
Event Based Process Data Delay	T101, T102	196

For further information, please consult the Anybus CompactCom 40 Software Design Guide.

F.2 Internal Timing

F.2.1 Startup Delay

The following parameters are defined as the time measured from the point where /RESET is released to the point where the specified event occurs.

Parameter	Description	Max.	Unit.
T1	The Anybus CompactCom 40 EtherNet/IP module generates the first application interrupt (parallel mode)	64	ms
T2	The Anybus CompactCom 40 EtherNet/IP module is able to receive and handle the first application telegram (serial mode)	64	ms

F.2.2 NW_INIT Handling

This test measures the time required by the Anybus CompactCom 40 EtherNet/IP module to perform the necessary actions in the NW_INIT-state.

Parameter	Conditions
No. of network specific commands	Max.
No. of ADIs (single UINT8) mapped to Process Data in each direction	32 ^a
Event based application message response time	> 1 ms
Ping-pong application response time	> 10 ms
No. of simultaneously outstanding Anybus commands that the application can handle	1

a. Or maximum amount in case the network specific maximum is less.

Parameter	Description	Communication	Max.	Unit.
T100	NW_INIT handling	Event based modes	58	ms

F.2.3 Event Based WrMsg Busy Time

The Event based WrMsg busy time is defined as the time it takes for the module to return the H_WRMSG area to the application after the application has posted a message.

Parameter	Description	Min.	Max.	Unit.
T103	H_WRMSG area busy time	6	9	µs

F.2.4 Event Based Process Data Delay

“Read process data delay” is defined as the time from when the last bit of the network frame has been received by the network interface, to when the RDPDI interrupt is asserted to the application.

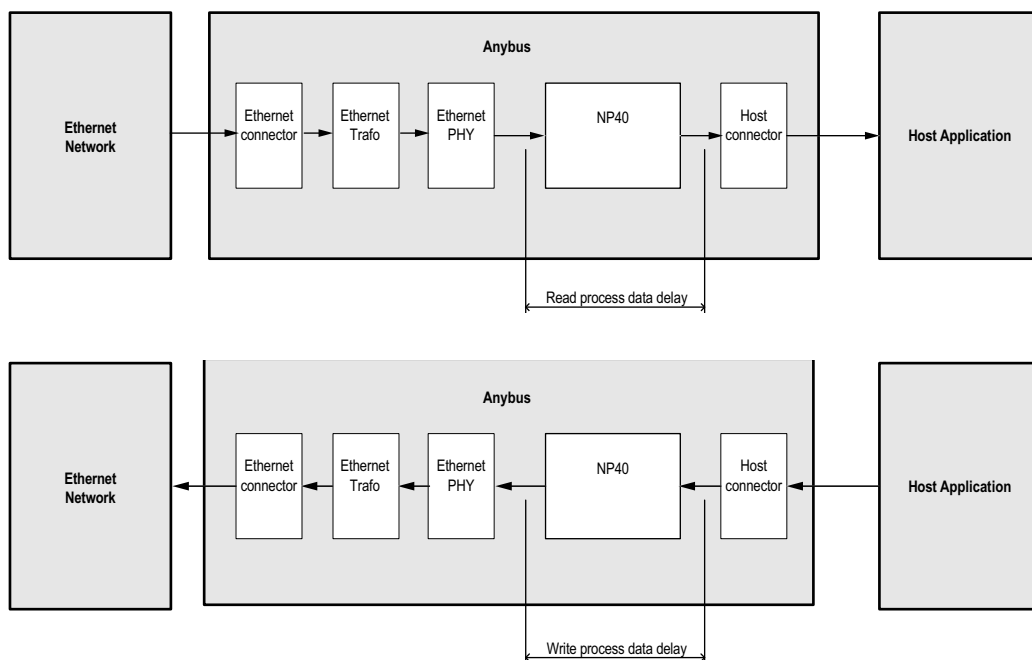
“Write process data delay” is defined as the time from when the application exchanges write process data buffers, to when the first bit of the new process data frame is sent out on the network.

The test was run in 16-bit parallel event mode, with interrupts triggered only for new process data events.

The delay added by the PHY circuit has not been included, as this delay is insignificant compared to the total process data delay.

Parameter	Description ^a	Delay (min.)	Delay (typ.)	Delay (max.)	Unit
T101	Read process data delay	45	50	84	µs
T102	Write process data delay	66	69	106	µs

a. Measured at an IO size of 32 bytes



G. Copyright Notices

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.

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.

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

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 NON-INFRINGEMENT. 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 NON-INFRINGEMENT. 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.