Network Interface Appendix

Anybus® CompactCom EtherNet/IP 2-Port Beacon Based DLR

Doc.Id. HMSI-168-35
Rev. 1.01

Connecting Devices™

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 the Anybus Compact-Com EtherNet/IP 2-Port Beacon Based DLR. The document only describes the features that are specific to the Anybus CompactCom EtherNet/IP 2-Port Beacon Based DLR. For general information regarding the Anybus CompactCom, consult the Anybus CompactCom design guides.

The reader of this document is expected to be familiar with high level software design, and communication systems in general. The use of advanced 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

Intellectual Property Rights

Trademark Acknowledgements

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

Anybus CompactCom EtherNet/IP 2-Port Beacon Based DLR Network Interface Appendix
Rev 1.01
Copyright© HMS Industrial Networks AB
May 2015 Doc Id HMSI-168-35

Table of Contents

P. About This Document

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

P.1 Related Documents

Document	Author
Anybus CompactCom Software Design Guide	HMS
Anybus CompactCom Hardware Design Guide	HMS
Anybus CompactCom Software Driver User Guide	HMS

P.2 Document History

Summary of Recent Changes (1.00... 1.01)

Change	Page(s)
Renamed Get_Instance_By_Order to Get_Instance_Number_By_Order in the document	-
Corrected connection values in the Connection Types section	56
Added Admin State attribute to the Ethernet Link Object (F6h)	69
Added the command Get_And_Clear (4Ch) to the Ethernet Link Object (F6h)	69
Fixed SMTP server data type and number of elements	83
Added QuickConnect attribute to Network Configuration Object (04h)	84
Added note to the Enable EtherNet/IP QuickConnect attribute in the EtherNet/IP Host Object	127
Added information for Process_CIP_Object_Request in the EtherNet/IP Host Object (F8h)	131
Corrected error response description for command Set_Configuration_Data in the EtherNet/IP Host Object	132
Added Port 1 State and Port 2 State attributes to Ethernet Host Object (F9h)	135

Revision List

Revision	Date	Author(s)	Chapter(s)	Description
1.00	2012-08-21	KaD	All	-
1.01	2015-05-13	KaD	All	-

P.3 Conventions & Terminology

The following conventions are used throughout this manual:

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

P.4 Sales and Support

For general contact information and support, please refer to the contact and support pages at www.hms-networks.com

1. About the Anybus CompactCom EtherNet/IP 2-port

1.1 General

The Anybus CompactCom EtherNet/IP 2-port 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 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 active modules defined in the Anybus CompactCom Hardware- and Software Design Guides, making it fully interchangeable with any other device following that specification. Generally, no additional network related software support is needed, however in order to be able to take full advantage of advanced network specific functionality, a certain degree of dedicated software support may be necessary.

1.2 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.

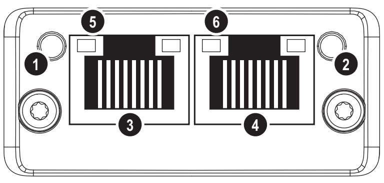
The Anybus CompactCom EtherNet/IP 2-Port Beacon Based DLR implements the DLR protocol. The device is able to process and act on beacon frames sent by ring supervisors, and supports beacon rates of 100 μs.

1.3 Features

- Two EtherNet/IP ports
- Beacon Based DLR (Device Level Ring) and linear network topology supported
- Galvanically isolated bus electronics
- 10/100 Mbit, full/half duplex operation
- Web server w. customizable content
- FTP server
- E-mail client
- Server Side Include (SSI) functionality
- Customizable identity information
- Up to 65535 ADIs
- CIP parameter object support
- Expandable CIP object implementation
- Supports unconnected CIP routing
- Transparent socket interface

1.4 Front View

#	Item
1	Network Status LED
2	Module Status LED
3	Ethernet Interface, Port 1
4	Ethernet Interface, Port 2
5	Link/Activity Port 1
6	Link/Activity Port 2



Network Status LED

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

LED State	Description
Off	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)

LINK/Activity LED 5/6

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 10/100 Mbit, full or half duplex operation.

2. Tutorial

2.1 Introduction

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

2.2 Fieldbus Conformance Notes

- ODVA requires that all manufacturers use their own Vendor ID. A Vendor ID can be applied for from ODVA.
- The Anybus CompactCom EtherNet/IP has been precompliance tested by ODVA's authorized Independent Test Lab and found to comply with the ODVA conformance test software. However, in accordance with ODVA's conformance test policy, the final product must still be compliance tested to ensure fieldbus conformance. In order to be able to do this, the vendor information in the EtherNet/IP Host Object must be customized.
- It is strongly recommended also to customize the information in the Identity Object (CIP), to enable the product to appear as a vendor specific implementation rather than a generic Anybus module.

2.3 Conformance Test Guide

When using the default settings of all parameters, the Anybus CompactCom EtherNet/IP 2-Port module is precertified for network compliance. This precertification is done to ensure that your product *can* be certified, but it does not mean that your product will not require certification.

Any change in the parameters in the EDS file, supplied by HMS, will require a certification. A Vendor ID can be obtained from ODVA and is compulsory for certification. This section provides a guide for successful conformance testing your product, containing the Anybus CompactCom EtherNet/IP 2-Port Beacon Based DLR module, to comply with the demands for network certification set by the ODVA.

Independent of selected operation mode, the actions described in this section have to be accounted for in the certification process. The identity of the product needs to be changed to match your company and device.

IMPORTANT: *This section provides guidelines and examples of what is needed for certification. Depending on the functionality of your application, there may be additional steps to take. Please contact HMS Industrial Networks at www.anybus.com for more information.*

2.3.1 Reidentifying Your Product

After successful setting of the “Setup Complete” attribute in the Anybus Object (01h), the Anybus module asks for identification data from the EtherNet/IP Host Object (F8h) and the Ethernet Host Object (F9h). Therefore, the attributes listed below shall be implemented and proper values returned.

Object/Instance	Attribute	Explanation	Default	Customer sample	Comment
EtherNet/IP Host Object (F8h), Instance 1	#1, Vendor ID	With this attribute you set the Vendor ID of the device.	Vendor ID: 005Ah	Vendor ID: 1111h	This information must match the keyword values of the “Device” section in the EDS file.
EtherNet/IP Host Object (F8h), Instance 1	#2, Device Type	With this attribute you set the Device Type of the device.	Device Type: 002Bh (Generic Device, Keyable)	Device Type: 0000h	
EtherNet/IP Host Object (F8h), Instance 1	#3, Product Code	With this attribute you set the Product Code of the device	0036h	2222h	
EtherNet/IP Host Object (F8h), Instance 1	#4, Revision	With this attribute you set the Revision of the device.		1.1	
EtherNet/IP Host Object (F8h), Instance 1	#5, Serial Number	With this attribute you set the Serial Number of the device.		12345678h	Unique number for all CIP devices produced with the same ‘Vendor ID.
EtherNet/IP Host Object (F8h), Instance 1	#6, Product Name	With this attribute you set the Product Name of the device.	Anybus-CC EIP (2-Port) BB DLR	“Widget”	This information must match the keyword values of the “Device” section in the EDS file.
Ethernet Host Object (F9h), Instance 1	#1, MAC address	With this attribute you set theMAC address of the device.		00-11-22-33-44-55	6 byte physical address value from range obtained from IEEE

2.3.2 Factory Default Reset

Reset command to Application Object (FFh) must be supported

When Anybus CompactCom EtherNet/IP 2-Port Beacon Based DLR modules are delivered, they are required to be in their factory default state. For Anybus CompactCom EtherNet/IP 2-Port Beacon Based DLR devices, this means that the IP suite is not assigned (IP 0.0.0.0). When a Factory Default Reset command is received from the network, the Anybus module will erase all IP information and inform the host application that a reset of the Anybus module is required. This is done by sending a Reset command to the Application Object (FFh) of the host (power on + factory default). For more details, please consult the Anybus CompactCom Software Design Guide.

2.3.3 IP address

Normally the IP numbers of Anybus CompactCom EtherNet/IP 2-Port Beacon Based DLR devices are assigned via a DHCP server. HMS recommends not using the Network Configuration Object (04h, instances #3 - #11) during the initialization phase for EtherNet/IP modules, unless the end user has requested the IP address to be set to a specific value (by for example using a keypad). The reason is that when a factory default reset command is received from the EtherNet/IP network (via DHCP) the node must be available after reset with the DHCP enabled.

3. Basic Operation

3.1 General Information

3.1.1 Software Requirements

Generally, no additional network support code needs to be written in order to support the Anybus CompactCom 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 76) can be created by the host application during normal conditions. An additional 6th instance may be created in event of a major fault.
- 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 (exactly what this means in practice depends on the implementation and the actual installation).
- 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 is such a way that this is not necessary.

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

See also...

- "Diagnostic Object (02h)" on page 76 (Anybus Module Objects)
- Anybus CompactCom Software Design Guide, 'Application Data Object (FEh)'

3.2 Device Customization

3.2.1 Network Identity

By default, the module uses the following identity settings:

• Vendor ID:	005Ah (HMS Industrial Networks)
• Device Type:	002Bh (Generic Device, Keyable)
• Product Code:	0036h (Anybus CompactCom EtherNet/IP 2-Port BB DLR)
• Product Name:	'Anybus-CC EIP (2-Port) BB DLR'

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 49 (CIP-object)
- "EtherNet/IP Host Object (F8h)" on page 127 (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 interopability 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.

3.2.2 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 invalidates 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 recertification 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 58 (CIP-object)
- Anybus CompactCom 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.*

3.2.3 EtherNet/IP & CIP Implementation

By default, the module supports the generic CIP profile. Optionally, it is possible to reroute 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.
- 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.

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

See also...

- “EtherNet/IP Host Object (F8h)” on page 127 (Host Application Object)
- “Command Details: Process_CIP_Object_Request” on page 131

3.2.4 Web Interface

The web interface can be fully customized to suit a particular application. 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 19
- “FTP Server” on page 20
- “Web Server” on page 22

3.2.5 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 86 (Anybus Module Object)
- “Message Segmentation” on page 141

3.3 Communication Settings

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

In this case, this includes...

- **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 autonegotiation 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 22
- "TCP/IP Interface Object (F5h)" on page 66 (CIP)
- "Ethernet Link Object (F6h)" on page 69 (CIP)
- "Network Configuration Object (04h)" on page 78 (Anybus Module Object)
- "HICP (Host IP Configuration Protocol)" on page 144

3.4 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 49 (CIP)
- "Diagnostic Object (02h)" on page 76 (Anybus Module Object)

3.5 Network Data Exchange

3.5.1 Application Data (ADIs)

ADIs are represented through the ADI Object (CIP). Each instance within this object 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 58 (CIP)
- "ADI Object (A2h)" on page 62 (CIP)

3.5.2 Process Data

Process Data is represented as dedicated instances in the Assembly Object (CIP). Note that each ADI element is mapped on a byte boundary, i.e. each BOOL occupies one byte.

See also...

- "Assembly Object (04h)" on page 52 (CIP)
- "Connection Manager (06h)" on page 55 (CIP)

3.5.3 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	

3.6 File System

3.6.1 General Information

Category: Extended

The built-in file system hosts 1.43 MByte of nonvolatile storage, which can be accessed by the HTTP and FTP servers, the e-mail client, and the host application.

The file system uses the following conventions:

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

See also...

- "FTP Server" on page 20
- "Web Server" on page 22
- "E-mail Client" on page 29
- "Server Side Include (SSI)" on page 30
- "File System Interface Object (0Ah)" on page 108

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

The following operations will erase one or more flash segments:

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

3.6.2 System Files

The file system contains a set of files used for system configuration. These files, known as "system files", are regular ASCII files which can be altered using a standard text editor (such as the Notepad in Microsoft Windows™). The format of these files are, with 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
```

4. FTP Server

4.1 General Information

Category: extended

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

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

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

The FTP server supports up to 8 concurrent connections.

4.2 User Accounts

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

File Format:

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

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

File Format ('\ftp.cfg'):

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

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

File Format:

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

Notes:

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

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

4.3 Session Example

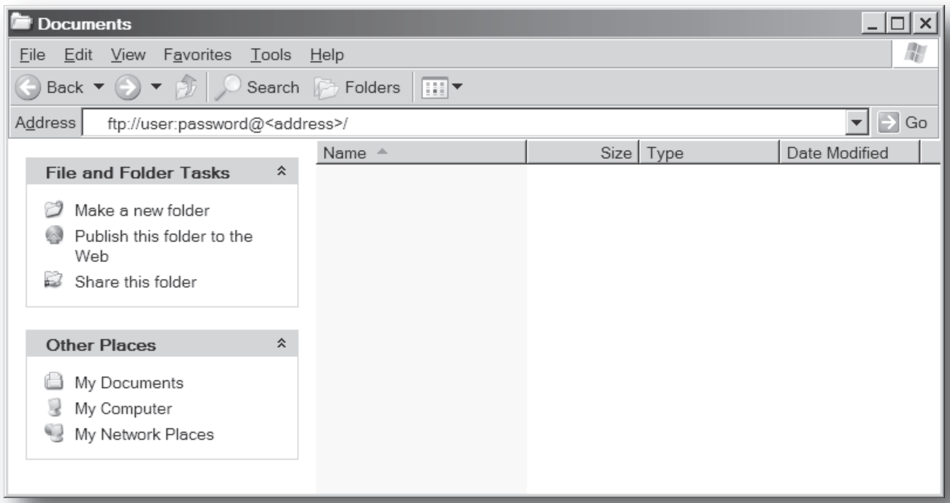
The Windows Explorer features a built-in FTP client which can easily be used to access the file system as follows:

1. Open the Windows Explorer by right-clicking on the 'Start' button and selecting 'Explorer'.

2. In the address field, type FTP://<user>:<password>@<address>.

- Substitute <address> with the IP address of the Anybus module.
- Substitute <user> with the username.
- Substitute <password> with the password.

3. Press enter. The Explorer will now attempt to connect to the Anybus module using the specified settings. If successful, the file system will be displayed in the Explorer window.



ftp://user:password@<address>/

5. Web Server

5.1 General Information

Category: extended

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

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

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

See also...

- "FTP Server" on page 20
- "Server Side Include (SSI)" on page 30
- "Ethernet Host Object (F9h)" on page 135

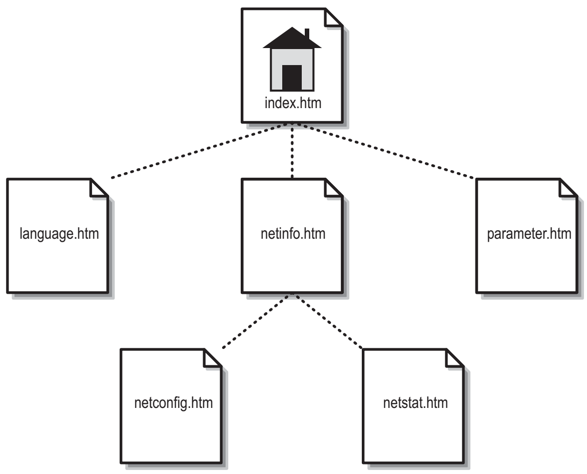
5.2 Default Web Pages

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

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

The files are:

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



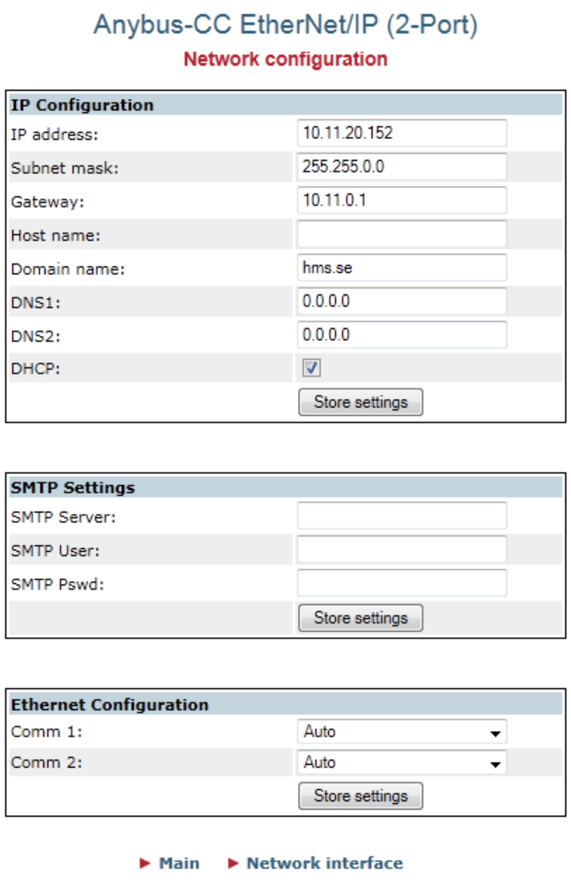
Note: If none of these files are used, it is recommended to completely disable the virtual file system altogether in the file system interface object.

See also...

- "File System" on page 19
- "File System Interface Object (0Ah)" on page 108

5.2.1 Network Configuration

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


Anybus-CC EtherNet/IP (2-Port)

Network configuration

IP Configuration	
IP address:	10.11.20.152
Subnet mask:	255.255.0.0
Gateway:	10.11.0.1
Host name:	
Domain name:	hms.se
DNS1:	0.0.0.0
DNS2:	0.0.0.0
DHCP:	✓

SMTP Settings	
SMTP Server:	
SMTP User:	
SMTP Pswd:	

Ethernet Configuration	
Comm 1:	Auto
Comm 2:	Auto

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
IP address	The TCP/IP settings of the module Default values: 0.0.0.0 Value ranges: 0.0.0.0 - 255.255.255.255
Subnet mask	
Gateway	
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 Default values: 0.0.0.0 Value ranges: 0.0.0.0 - 255.255.255.255
DNS 2	
DHCP	Checkbox for enabling or disabling DHCP Default value: enabled

SMTP Settings

The module needs a reset before any changes take effect.

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

Ethernet Configuration

Changes will take effect immediately.

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

5.2.2 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 du to request format error
Connection Close Other Rejects	Connection close requests rejected due to other reasons
Connection Timeouts	Nummer of connection timeouts

Interface Counters	Description
In Octets:	Received bytes.
In Ucast Packets:	Received unicast packets.
In NUcast packets:	Received nonunicast 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 nonunicast packets (broadcast and multicast).
Out Discards:	Outgoing packets discarded due to no available memory buffers.
Out Errors:	Transmission errors.

5.3 Server Configuration

5.3.1 General Information

Category: advanced

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

File Format:

```
[WebRoot]
\web

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

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

- **Web Root Directory**

  The web server cannot access files outside this directory.

- **Content Types**

  A list of file extensions and their reported content types.

  See also...
  - "Default Content Types" on page 27

- **SSI File Types**

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

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

5.3.2 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 22

5.3.3 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 26.

5.3.4 Authorization

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

File Format:

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

• List of approved users.

```
[AuthName]
(message goes here)
```

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

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

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

Example:

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

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

[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 web authentication method using encrypted password. Used as default if no [Authtype] field is specified.

6. E-mail Client

6.1 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 78
- "SMTP Client Object (09h)" on page 103

6.2 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).

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 106.

7. Server Side Include (SSI)

7.1 General Information

Category: advanced

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

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

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

7.2 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)

7.3 Command Functions

7.3.1 General Information

Command functions executes commands and includes the result.

General Syntax:

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

command-Command function, see below.

Command Functions:

Command	Valid for E-mail Messages	Page
GetConfigItem()	Yes	32
SetConfigItem()	No	33
SsiOutput()	Yes	35
DisplayRemoteUser	No	35
ChangeLanguage()	No	36
IncludeFile()	Yes	37
SaveDataToFile()	No	38
printf()	Yes	39
scanf()	No	41

7.3.2 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. “<br>”).
(default is CRLF).

Default Output:

Scenario	Default Output
Success	*(value of specified key)*
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’.

7.3.3 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'.

7.3.4 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 47

7.3.5 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)

7.3.6 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'.

7.3.7 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. “<br>”).

Default Output:

Scenario	Default Output
Success	*(file contents)*
Authentication Error	“Authentication error ”
File open error	“Failed to open file “*filename*” “

Example:

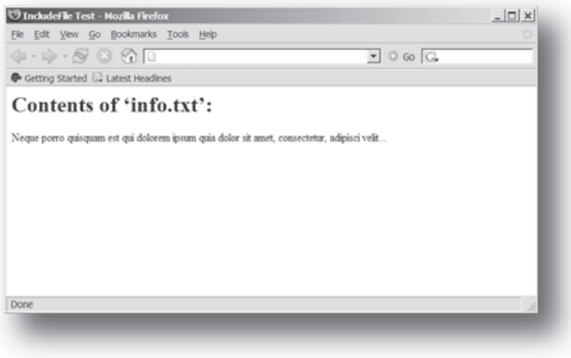
The following example demonstrates how to use this function.

```
<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:


IncludeFile Test - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
Getting Started Latest Headlines
Done

Contents of ‘info.txt’:

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

See also...

- “Include File” on page 30

7.3.8 SaveDataToFile()

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

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

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

Syntax:

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

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

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

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

Default Output:

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

Example:

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

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

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

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

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

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

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

</BODY>
</HTML>
```

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

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

7.3.9 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 40
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 43

Default Output:

Scenario	Default Output
Success	(printf() result)
ABCCMessage error	ABCCMessage error string ("Errors" on page 46)

Example:

See also...

- "ABCCMessage()" on page 43
- "Example (Get_Attribute):" on page 45

Formatting Tags

- **Type (Required)**

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

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

- **Flags (Optional)**

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

- **Width (Optional)**

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

- **Precision (Optional)**

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

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

- **Modifier**

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

7.3.10 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 e-mail messages.

Syntax:

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

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

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

Formatting tags are written as follows:

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

See also...

- “Formatting Tags” on page 42

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 43

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 46)

Example:

See also...

- “ABCCMessage()” on page 43
- “Example (Set_Attribute):” on page 45

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: Initial Characters:Format: 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 nonwhitespace characters	STRING
[scanset]	Accepts a sequence of nonwhitespace characters from a set of expected bytes specified by the scanlist (e.g ‘[0123456789ABCDEF]’) A literal ‘]’ character can be specified as the first character of the set. A caret character (‘^’) immediately following the initial ‘[’ inverts the scanlist, i.e. allows all characters except the ones that are listed.	STRING
%	Accepts a single ‘%’ input at this point; no assignment or conversion is done. The complete conversion specification should be ‘%%’.	-

- *** (Optional)**

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

- **Width (Optional)**

  Specifies the maximum number of characters to be read.

- **Modifier (Optional)**

  Specifies a different data size.

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

7.4 Argument Functions

7.4.1 General Information

Argument functions are supplied as parameters to certain command functions.

General Syntax:

(Syntax depends on context)

Argument Functions:

Function	Description	Page
ABCCMessage()	-	43

7.4.2 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 44

r_type - Specifies the data type in the response (msgdata)

See also...

- "Response Data Types (r_type)" on page 44

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 45
- "Example (Set_Attribute):" on page 45

- **Command Data Types (c_type)**

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

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

- **Response Data Types (r_type)**

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

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

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

Example (Get_Attribute):

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

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

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

See also...

- "printf()" on page 39

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 41

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 47

7.5 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"

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

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

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

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

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

Each error code corresponds to a dedicated output string, labelled from 1 to 16.

See also...

- "Errors" on page 46

Use "%s" to include the name of the file.

Use "%s" to include the name of the file.

Use "%s" to include the name of the file.

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 35

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 49
- "Message Router (02h)" on page 51
- "Assembly Object (04h)" on page 52
- "Connection Manager (06h)" on page 55
- "Parameter Object (0Fh)" on page 58
- "DLR Object (47h)" on page 60
- "QoS Object (48h)" on page 61
- "Port Object (F4h)" on page 64 (Optional)
- "TCP/IP Interface Object (F5h)" on page 66
- "Ethernet Link Object (F6h)" on page 69

Vendor Specific Objects:

- "ADI Object (A2h)" on page 62

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 127 and "Command Details: Process_CIP_-Object_Request" on page 131.

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

8.2 Identity Object (01h)

Category

Extended

Object Description

-

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	-

Instance #1 Attributes

If needed, the Anybus CompactCom EtherNet/IP 2-Port Beacon Based DLR can support multiple Identity Object instances. This is done by implementing instances in the CIP Identity Host Object. This will alter attribute 1 - 6. The remaining attributes will be treated identical for all instances. See "CIP Identity Host Object (EDh)" on page 125 for more information.

Extended

#	Name	Access	Type	Value
1	Vendor ID	Get	UINT	005Ah (HMS Industrial Networks AB)[a]
2	Device Type	Get	UINT	002Bh (Generic Device, Keyable)[a]
3	Product Code	Get	UINT	0036h (Anybus CompactCom EIP 2-Port BB DLR)[a]
4	Revision	Get	Struct of: {USINT, USINT}	Major and minor firmware revision[a]
5	Status	Get	WORD	See "Device Status" on page 50
6	Serial Number	Get	UDINT	Unique serial number (assigned by HMS)[a]
7	Product Name	Get	SHORT_STRING	"Anybus-CC EIP (2-Port) BB DLR" (Name of product)[a]
11	Active language	Set	Struct of: USINT USINT USINT	Requests sent to this instance are forwarded to the application object. The host application is then responsible for updating the language settings accordingly.

#	Name	Access	Type	Value
12	Supported Language List	Get	Array of:	List of languages supported by the host application. This list is read from the Application Object during the NW_INIT state, and translated to CIP standard.
			Struct of: USINT USINT USINT	

a. Can be customized by implementing the EtherNet/IP Host Object, see "EtherNet/IP Host Object (F8h)" on page 127.

Device Status

Bit(s)	Name
0	Module Owned
1	(reserved)
2	Configured[a]
3	(reserved)
4... 7	Extended Device Status: Value:Meaning: 0000b Unknown 0010b Faulted I/O Connection 0011b No I/O connection established 0100b Nonvolatile configuration bad 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 76.

Service Details: Reset Service

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

8.3 Message Router (02h)

Category

Extended

Object Description

-

Supported Services

Class:	-
Instance:	-

Class Attributes

-

Instance Attributes

-

8.4 Assembly Object (04h)

Category

Extended

Object Description

The assembly object uses static assemblies and holds the process data sent/received by the host application. The default assembly instance IDs used are in the vendor specific range.

See also...

- "Process Data" on page 18
- "EtherNet/IP Host Object (F8h)" on page 127

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)

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)

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	Get/Set	N/A	- (Configuration data written to the application when the forward open command has the configuration data included)

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 132

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 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.

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 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.

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.

See also...

- "Network Data Exchange" on page 18
- "EtherNet/IP Host Object (F8h)" on page 127 (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.

See also...

- "Network Data Exchange" on page 18
- "EtherNet/IP Host Object (F8h)" on page 127 (Instance attribute #8)

8.5 Connection Manager (06h)

Category

Extended

Object Description

-

Supported Services

Class:	-
Instance:	Forward_Open Forward_Close

Instance Descriptions

(No supported instance attributes)

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.

- Total number of supported class 1 connections:4
- Max input connection size: 256 bytes
- Max output connection size: 256 bytes
- Supported API: 2... 3200 ms
- T->O Connection type: Point-to-point, Multicast
- O->T Connection type: Point-to-point
- Supported trigger types: Cyclic, COS

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 ⇒ T:	Assembly object, instance 96h (default)
- Connection point T ⇒ 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 4[1]
- Connection point O ⇒ T:	Assembly object, instance 03h (default)
- Connection point T ⇒ 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 ⇒ T:	Assembly object, instance 06h (default)
- Connection point T ⇒ 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 4[2]
- Connection point O ⇒ T:	Assembly object, instance 04h (default)
- Connection point T ⇒ 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 ⇒ T:	Assembly object, instance 07h (default)
- Connection point T ⇒ 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:	16
- Supported API:	2 - 10000 ms
- T->O Connection type:	Point-to-point
- O->T Connection type:	Point-to-point
- Supported trigger type:	Application

8.6 Parameter Object (0Fh)

Category

Extended

Object Description

This object allows configuration tools such as RSNetworx to 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.

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 59.

See also...

- "Default Values" on page 59
- "ADI Object (A2h)" on page 62 (CIP Object)
- "EtherNet/IP Host Object (F8h)" on page 127 (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 01011b Bit:Contents: 0 Supports parameter instances 1 Supports full attributes 2 Must do nonvolatile storage save command 3 Parameters are stored in nonvolatile storage
9	Configuration Assembly instance	Get	UINT	0000h (Configuration assembly not supported)

a. Consult the general Anybus CompactCom 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
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	Bit:Contents: 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	EPATH	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	(not supported)
9	Help String	Get	SHORT_STRING	
10	Minimum value	Get	(Data Type)	Minimum value of parameter
11	Maximum value	Get	(Data Type)	Maximum value of parameter
12	Default value	Get	(Data Type)	Default value of parameter
13	Scaling Multiplier	Get	UINT	0001h (not supported)
14	Scaling Divisor	Get	UINT	
15	Scaling Base	Get	UINT	
16	Scaling Offset	Get	INT	0000h (not supported)
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	

Default Values

#	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

8.7 DLR Object (47h)

Category

Extended

Supported Services

Class:	Get_Attribute_Single Get_Attributes_All
Instance:	Get_Attribute_Single Set_Attribute_Single

Class Attributes

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

Instance #1 Attributes

Extended

#	Name	Access	Type	Value
1	Network Topol-ogy	Get	USINT	Bit:Contents: 0 "Linear" 1 "Ring"
2	Network Status	Get	USINT	Bit:Contents: 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 8 USINTs}	This attribute holds the IP address (IPv4) and/or the Ethernet Mac address of the active ring supervisor
12	Capability Flags[a]	Get	DWORD	02h (Beacon based ring node)

a. See table below.

Capability Flags

Bits	Name
0	Announce based Ring Node
1	Beacon based ring node
2 - 4	Reserved (set to zero)
5	Supervisor capable
6	Redundant gateway capable
7	Flush_Table frame capable
8 - 31	Reserved (set to zero)

8.8 QoS Object (48h)

Category

Extended

Object Description

-

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 #1 Attributes

Extended

#	Name	Access	Type	Value
1	802.1Q Tag Enable	Set	USINT	Enables or disables sending 802.1Q frames. Bit:Contents: 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

8.9 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.

See also...

- Application data object (see Anybus CompactCom Software Design Guide)
- "Parameter Object (0Fh)" on page 58 (CIP Object)

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 = 0001h)
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 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 Software Design Guide).

Extended

#	Name	Access	Type	Description
1	Name	Get	SHORT_STRING	Parameter name (Including length)
2	ABCC Data type	Get	USINT	Data type of instance value
3	No. of elements	Get	USINT	Number of elements of the specified data type
4	Descriptor	Get	USINT	Bit field describing the access rights for this instance Bit:Meaning: 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.

a. Converted to/from CIP standard by the module

8.10 Port Object (F4h)

Category

Extended

Object Description

This object exists only if enabled in the EtherNet/IP host object (Instance Attribute #17).

See also...

- "EtherNet/IP Host Object (F8h)" on page 127 (Anybus module object)
- "CIP Port Configuration Object (0Dh)" on page 122 (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
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 noninstantiated 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 #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) 3h (if the application registers a port)
2	Port Number	Get	UINT	2h
3	Link Object	Get	Struct of: UINT Padded EPATH	- 2h 20 F5 24 01h
4	Port Name	Get	SHORT_STRING	“TCP/IP”
7	Node Address	Get	Padded EPATH	-
8	Port Node Range	Get	Struct of: UINT (Min.) UINT (Max.)	-

See also...

- “CIP Port Configuration Object (0Dh)” on page 122

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.
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 122 (“Instance Attributes” on page 123)

8.11 TCP/IP Interface Object (F5h)

Category

Extended

Object Description

The object groups TCP/IP-related settings. See also...

- "Communication Settings" on page 17
- "Network Configuration Object (04h)" on page 78 (Anybus Module Object)

Supported Services

Class services:	Get_Attribute_All Get_Attribute_Single
Instance services:	Get_Attribute_All Get_Attribute_Single Set_Attribute_Single[1]

Class Attributes

#	Access	Name	Type	Value	Description
1	Get	Revision	UINT	0003h	Revision 3

1. Support for this service can be disabled by implementing attribute #9 in the EtherNet/IP host object.

Instance Attributes

Extended

#	Access	Name	Type	Value	Comments
1	Get	Status	DWORD	-	Bit #: Meaning: 0-3: When set to 1h, attribute #5 contains valid configuration from DHCP or nonvolatile storage. When set to 2h, attribute #5 contains valid configuration from hardware settings. Remaining values are reserved for future use. 4: Multicast pending if set to 1 5: Interface configuration pending if set to 1 6: AcdStatus. Set to 1 if an address conflict is detected. Address conflict detection is enabled/disabled in attribute #10. 7 - 31: (reserved, set to 0)
2	Get	Configuration Capability	DWORD	0000 0086h - or - 0000 0096h - or - 0000 00A6h - or - 0000 00B6h	Bit #: Meaning: 0: (reserved, set to 0) 1: Always 1. The module is capable of resolving host names by querying a DNS server. 2: Always 1. The module is capable of obtaining its network configuration via DHCP. 3: (reserved, set to 0) 4: The ‘Configuration Settable’-bit reflects the value of instance attribute #9 in the “EtherNet/IP Host Object (F8h)” on page 127. 5: The module is hardware configurable when this bit is set to 1. 6: (reserved, set to 0) 7: Always 1, the device is capable of detecting address conflicts. 8 - 31: (reserved, set to 0)
3	Get/Set	Configuration Control	DWORD	-	Value: Meaning: 0: Configuration from nonvolatile memory 2: Configuration from DHCP
4	Get	Port Object	Struct of: UINT (Path size) Padded EPATH	- 0002h 20 F6 24 01h	- 2 words Path to Ethernet link object, Instance #1
5	Get/Set	Interface Configuration	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	Get/Set	Host Name	STRING	-	Host name of Anybus module
7	(not used)				
8	Get/Set	TTL Value	USINT	1	TTL value for EtherNet/IP multicast packets

#	Access	Name	Type	Value	Comments
9	Get/Set	Mcast Config	Struct of:		
		Alloc Control	USINT	0	Value: Meaning: 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	Set	SelectAcd	Bool	1	Value: Meaning: 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 11.
11	Set	LastConflictDetected	Struct of:		ACD Diagnostic parameters
		AcdActivity	USINT	-	
		RemoteMAC	ARRAY of 6 USINT	-	
		ArpPdu	ARRAY of 28 USINT	1-	
12[a]	Set	EtherNet/IP QuickConnect	BOOL	0	Value: Meaning: 0: Disable EtherNet/IP QuickConnect 1: Enable EtherNet/IP QuickConnect

a. QuickConnect shall only be supported if it is enabled from the application.

8.12 Ethernet Link Object (F6h)

Category

Extended

Object Description

This object groups diagnostic information for the Ethernet interface.

See also...

- "Communication Settings" on page 17
- "Network Configuration Object (04h)" on page 78 (Anybus Module Object)

Supported Services

Class services:	Get_Attribute_All (01h) Get_Attribute_Single (0Eh)
Instance services:	Get_Attribute_All (01h) Get_Attribute_Single (0Eh) Set_Attribute_Single (10h) Get_And_Clear (4Ch)

Class Attributes

#	Access	Name	Type	Value	Comments
1	Get	Revision	UINT	3	Revision 3
2	Get	Max Instance	UINT	3	Instance 3 is the max instance
3	Get	Number of instances	UINT	3	3 instances

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 72.
3	Get	Physical Address	Array of 6 USINTS	(MAC ID)	Physical network address
4	Get	Interface Counters	Struct:		
		In Octets	UDINT	-	Octets received on the interface
		In Ucast Packets	UDINT	-	Unicast packets received on the interface
		In NUcast Packets	UDINT	-	Nonunicast packets received on the interface
		In Discards	UDINT	-	Inbound packets with unknown protocol
		In Errors	UDINT	-	Inbound packets that contain errors (does not include discards)
		In Unknown Protos	UDINT	-	Inbound packets with unknown protocol
		Out Octets	UDINT	-	Octets sent on the interface
		Out Ucast Packets	UDINT	-	Unicast packets sent on the interface
		Out NUcast Packets	UDINT	-	Nonunicast packets sent on the interface
		Out Discards	UDINT	-	Outbound packets with unknown protocol
		Out Errors	UDINT	-	Outbound packets that contain errors (does not include discards)

#	Access	Name	Type	Value	Comments
5	Get	Media Counters	Struct:		
		Alignment Errors	UDINT	-	Frames received that are not an integral number of octets in length
		FCS Errors	UDINT	-	Frames received that do not pass the FCS check
		Single Collisions	UDINT	-	Successfully transmitted frames which experienced exactly one collision
		Multiple Collisions	USINT	-	Successfully transmitted frames which experienced more than one collision
		SQE Test Errors	UDINT	0	-
		Deferred Transmissions	UDINT	-	Frames for which first transmission attempt is delayed because the medium is busy
		Late Collisions	UDINT	-	Number of times a collision is detected later than 512 bit-times into the transmission of a packet
		Excessive Collisions	UDINT	-	Frames for which a transmission fails due to excessive collisions
		MAC Transmit Errors	UDINT	-	Frames for which transmission fails due to an internal MAC sublayer receive error
		Carrier Sense Errors	UDINT	-	Times that the carrier sense condition was lost or never asserted when attempted to transmit a frame
		Frame Too Long	UDINT	-	Frames received that exceed the maximum permitted frame size
		MAC Receive Errors	UDINT	-	Frames for which reception on an interface fails due to an internal MAC sublayer receive error
6	Get/Set[a]	Interface Control	Struct:		
		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 Type” on page 72
9	Get/Set	Admin State	USINT	-	Controls the administrative setting of the interface state. 1: Enabled 2: Disabled 0, 3 - 255: Reserved
10	Get	Interface Label	SHORT_STRING	-	See “Interface Label” on page 72

a. Support for this attribute can be disabled by implementing attribute #9 in the EtherNet/IP host object.

Interface Flags

Bit	Name	Description
0	Link status	Indicates whether or not the Ethernet 802.3 communications interface is connected to an active network. Value:Meaning: 0 Inactive link 1 Active link
1	Half/full duplex	Indicates the duplex mode currently in use. Value:Meaning: 0 Half duplex 1 Full duplex
2 - 4	Negotiation Status	Indicates the status of link autonegotiation Value:Meaning: 0 Autonegotiation in progress. 1 Autonegotiation and speed detection failed (using default values) 2 Autonegotiation failed but detected speed (using default duplex value) 3 Successfully negotiated speed and duplex. 4 Autonegotiation not attempted. Forced speed and duplex.
5	Manual Setting requires Reset	Value:Meaning: 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	Value:Meaning: 0 No local hardware fault detected 1 Local hardware fault detected
7-31	(reserved)	(ignore)

Interface Type

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

Interface Label

Instance	Value
1	Port 1
2	Port 2
3	Internal

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 EtherNet/IP.

Standard Objects:

- "Anybus Object (01h)" on page 74
- "Diagnostic Object (02h)" on page 76
- "Network Object (03h)" on page 77
- "Network Configuration Object (04h)" on page 78

Network Specific Objects:

- "Socket Interface Object (07h)" on page 86
- "SMTP Client Object (09h)" on page 103
- "File System Interface Object (0Ah)" on page 108
- "Network Ethernet Object (0Ch)" on page 19
- "CIP Port Configuration Object (0Dh)" on page 122

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 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 Software Design Guide for further information.)

Instance Attributes (Instance #1)

Basic

#	Name	Access	Type	Value
1	Module type	Get	UINT16	0401h (Standard Anybus CompactCom)
2... 11	-	-	-	Consult the general Anybus CompactCom Software Design Guide for further information.

12	LED colors	Get	struct of:
			UINT8(LED1A)
			UINT8(LED1B)
			UINT8(LED2A)
			UINT8(LED2B)
				Value:Color:
				01h Green
				02h Red
				01h Green
				02h Red

13... 15	-	-	-	Consult the general Anybus CompactCom Software Design Guide for further information.

Extended

#	Name	Access	Type	Value
16	GPIO configuration	Get/Set[a]	UINT16	Configuration of the host interface GPIO pins. See the table below.

a. Set access of attribute GPIO configuration is only valid in state SETUP.

GPIO Configuration Settings

Value	Functionality	Description
0x0000	Standard	GIP[0..1] and GOP[0..1] are used as general input/output pins
0x0001	Extended LED functionality	GIP1 is used for Link/Activity LED (port 1). The 10/100 MHz indication is combined into one signal, connected to a green LED, that is lit when there is a link and flickering on activity. GOP1 is used for Link/Activity LED (port 2). The 10/100 MHz indication is combined into one signal, connected to a green LED, that is lit when there is a link and flickering on activity.

For more information, see

- “Extended LED Functionality” in “Appendix B” on page 139.

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 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 Software Design Guide for further information.
11	Max no. of instances	Get	UINT16	5+1

Instance Attributes

Basic

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

In this implementation, the severity level of all instances are combined (using logical 'OR') and represented on the network through the CIP identity object. The event code cannot be represented on the network and is thus ignored by the module.

See also...

- "Diagnostics" on page 17
- "Identity Object (01h)" on page 49 (CIP-object)

9.4 Network Object (03h)

Category

Basic

Object Description

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

Supported Commands

Object:	Get_Attribute
Instance:	Get_Attribute Set_Attribute Get_Enum_String Map_ADI_Write_Area Map_ADI_Read_Area

Object Attributes (Instance #0)

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

Instance Attributes(Instance #1)

Basic

#	Name	Access	Type	Value
1	Network type	Get	UINT16	009Bh
2	Network type string	Get	Array of CHAR	'EtherNet/IP (2-Port) BB DLR'
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	ValueMeaning 0: No information available 1: Invalid assembly instance mapping
8... 10	-	-	-	Consult the general Anybus CompactCom Software Design Guide for further information.

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

9.5 Network Configuration Object (04h)

Category

Extended, Advanced

Object Description

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

See also...

- "Communication Settings" on page 17
- "TCP/IP Interface Object (F5h)" on page 66 (CIP-object)
- "Ethernet Link Object (F6h)" on page 69 (CIP-object)
- "E-mail Client" on page 29

Note: In order to ensure fieldbus conformance, the recommendations stated in the Anybus Compact-Com Software Design Guide regarding this object must be followed.

Supported Commands

Object:	Get_Attribute Reset
Instance:	Get_Attribute Set_Attribute Get_Enum_String

Object Attributes (Instance #0)

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

Instance Attributes (Instance #3, IP Address)

Changes are valid after reset.

Advanced

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

a. Multilingual, see “Multilingual Strings” on page 85.

Instance Attributes (Instance #4, Subnet Mask)

Changes are valid after reset.

Advanced

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

a. Multilingual, see “Multilingual Strings” on page 85.

Instance Attributes (Instance #5, Gateway)

Changes are valid after reset.

Advanced

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

a. Multilingual, see “Multilingual Strings” on page 85.

Instance Attributes (Instance #6, DHCP)

Changes are valid after reset.

Extended

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

a. Multilingual, see "Multilingual Strings" on page 85.

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

Changes have immediate effect.

Advanced

#	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	Value:Enum. String:Meaning: 00h 'Auto' Autonegotiation (default) 01h '10 HDX' 10 Mbit, half duplex 02h '10 FDX' 10 Mbit, full duplex 03h '100 HDX' 100 Mbit, half duplex 04h '100 HDX' 100 Mbit, full duplex

a. Multilingual, see "Multilingual Strings" on page 85.

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

Changes have immediate effect.

Advanced

#	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' Autonegotiation (default) 01h '10 HDX' 10 Mbit, half duplex 02h '10 FDX' 10 Mbit, full duplex 03h '100 HDX' 100 Mbit, half duplex 04h '100 HDX' 100 Mbit, full duplex

a. Multilingual, see "Multilingual Strings" on page 85.

Instance Attributes (Instance #9, DNS1)

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

Advanced

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

a. Multilingual, see "Multilingual Strings" on page 85.

Instance Attributes (Instance #10, DNS2)

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

Advanced

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

a. Multilingual, see “Multilingual Strings” on page 85.

Instance Attributes (Instance #11, Host name)

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

Advanced

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

a. Multilingual, see “Multilingual Strings” on page 85.

Instance Attributes (Instance #12, Domain name)

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

Advanced

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

a. Multilingual, see “Multilingual Strings” on page 85.

Instance Attributes (Instance #13, SMTP Server)

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

Advanced

#	Name	Access	Type	Description
1	Name[a]	Get	Array of CHAR	'SMTP Server'
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	40h
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of UINT8	Valid range: 0.0.0.0 - 255.255.255.255 (Default = 0.0.0.0)

a. Multilingual, see "Multilingual Strings" on page 85.

Instance Attributes (Instance #14, SMTP User)

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

Advanced

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

a. Multilingual, see "Multilingual Strings" on page 85.

Instance Attributes (Instance #15, SMTP Password)

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

Advanced

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

a. Multilingual, see "Multilingual Strings" on page 85.

Instance Attributes (Instance #16 - #19, Reserved)

These instances are reserved for future use.

Instance Attributes (Instance #20, QuickConnect Enable/Disable)

This instance enables or disables the QuickConnect functionality. Changes to the parameter will immediately change the value of the QuickConnect attribute in the TCP/IP Interface object, see page 66. Changes are valid after reset.

This network configuration instance is not available during SETUP or NW-INIT states. It is only available when QuickConnect feature is enabled in the EtherNet/IP Host Object (F8h) and the module is in the WAIT_PROCESS or above.

Advanced

#	Name	Access	Type	Description
1	Name[a]	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	Value:Enum. String:Meaning: 00h DisableQuickConnect (default) 01h EnableQuickConnect

a. Multilingual, see "Multilingual Strings" on page 85.

Multilingual Strings

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

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

9.6 Socket Interface Object (07h)

Category

Advanced

Object Description

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

Note that some of the commands used when accessing this object may require segmentation. For more information, see “Message Segmentation” on page 141.

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 88) Delete (See “Command Details: Delete” on page 89)
Instance:	Get_Attribute Set_Attribute Bind (See “Command Details: Bind” on page 90) Shutdown (See “Command Details: Shutdown” on page 91) Listen (See “Command Details: Listen” on page 92) Accept (See “Command Details: Accept” on page 93) Connect (See “Command Details: Connect” on page 94) Receive (See “Command Details: Receive” on page 95) Receive_From (See “Command Details: Receive_From” on page 96) Send (See “Command Details: Send” on page 97) Send_To (See “Command Details: Send_To” on page 98) IP_Add_membership (See “Command Details: IP_Add_Membership” on page 99) IP_Drop_membership (See “Command Details: IP_Drop_Membership” on page 100) DNS_Lookup (See “Command Details: DNS_Loopup” on page 101)

Object Attributes (Instance #0)

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

Instance Attributes (Sockets #1...8)

Advanced

#	Name	Access	Type	Description
1	Socket type	Get	UINT8	Value:Socket Type: 00h SOCK_STREAM, NONBLOCKING (TCP) 01h SOCK_STREAM, BLOCKING (TCP) 02h SOCK_DGRAM, NONBLOCKING (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): Value:State:Description: 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 Value:Meaning: 1 Enabled 0 Disabled (default)
9	Keep alive	Get/Set	BOOL	Protocol probes idle connection (TCP sockets only) Value:Meaning: 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] Value:Meaning: 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). Value:Meaning: 1 Delay (default) 0 Don’t delay (turn off Nagle’s algorithm on socket)
14	TCP Connect Timeout	Get/Set	UINT16	TCP Connect timeout in seconds (default = 75s)

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

Advanced

Details

Command Code.:	03h
Valid for:	Object Instance

Description

This command creates a socket.

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

- **Command Details**

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

- **Response Details**

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

Command Details: Delete

Category

Advanced

Details

Command Code.:	04h
Valid for:	Object Instance

Description

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

- If the socket is of TCP type and a connection is established, the connection is terminated with the RST flag.
- To terminate a TCP connection, it is recommended to use the 'Shutdown' command (see "Command Details: Shutdown" on page 91) before deleting the socket, causing the connection to be closed with the FIN flag instead.

- **Command Details**

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

- **Response Details**

  (no data)

Command Details: Bind

Category

Advanced

Details

Command Code.:	10h
Valid for:	Instance

Description

This command binds a socket to a local port.

- **Command Details**

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

- **Response Details**

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

Command Details: Shutdown

Category

Advanced

Details

Command Code.:	11h
Valid for:	Instance

Description

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

- **Command Details**

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

- **Response Details**

  (no data)

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

Application initiates shutdown:

1. Send shutdown with CmdExt[1] set to 01h. This will send the FIN flag to the 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 the 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, the RST flag will be sent to terminate the socket.

Host initiates shutdown:

1. Receive data on socket, if zero bytes received it indicates that the host closed the receive channel of the socket.

2. Try to send any unsent data to the host.

3. Send shutdown with CmdExt[1] set to 01h. This will send the FIN flag to the host shutting down the receive channel.

4. Delete the socket instance.

Command Details: Listen

Category

Advanced

Details

Command Code.:	12h
Valid for:	Instance

Description

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

- **Command Details**

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

- **Response Details**

  (no data)

Command Details: Accept

Category

Advanced

Details

Command Code.:	13h
Valid for:	Instance

Description

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

NONBLOCKING mode:

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

BLOCKING mode:

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

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

- **Command Details**

  (no data)

- **Response Details**

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

Command Details: Connect

Category

Advanced

Details

Command Code.:	14h
Valid for:	Instance

Description

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

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

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

NONBLOCKING mode:

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

BLOCKING mode:

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

- **Command Details**

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

- **Response Details**

(no data)

Command Details: Receive

Category

Advanced

Details

Command Code.:	15h
Valid for:	Instance

Description

This command receives data from a connected socket. Message segmentation may be used to receive up to 1472 bytes (see “Message Segmentation” on page 141).

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.

NONBLOCKING 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 142
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 141).

Field	Contents	Notes
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control bits	see “Response Segmentation” on page 143
Data[0...n]	Received data	-

Command Details: Receive_From

Category

Advanced

Details

Command Code.:	16h
Valid for:	Instance

Description

This command receives data from an unconnected SOCK_DGRAM socket. Message segmentation may be used to receive up to 1472 bytes (see "Message Segmentation" on page 141).

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.

NONBLOCKING 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 142
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 141).

Field	Contents	Notes
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control bits	see "Response Segmentation" on page 143
Data[0]	Host IP address byte 3 (low)	The host address/port information is only included in the first segment. All data thereafter will start at Data[0]
Data[1]	Host IP address byte 2	
Data[2]	Host IP address byte 1	
Data[3]	Host IP address byte 0 (high)	
Data[4]	Host port number (low)	
Data[5]	Host port number (high)	
Data[6...n]	Received data	

Command Details: Send

Category

Advanced

Details

Command Code.:	17h
Valid for:	Instance

Description

This command sends data on a connected socket. Message segmentation may be used to send up to 1472 bytes (see "Message Segmentation" on page 141).

NONBLOCKING 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 141).

Field	Contents	Notes
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control	see "Command Segmentation" on page 142
Data[0...n]	Data to send	-

- **Response Details**

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

Command Details: Send_To

Category

Advanced

Details

Command Code.:	18h
Valid for:	Instance

Description

This command sends data to a specified host on an unconnected SOCK-DGRAM socket. Message segmentation may be used to send up to 1472 bytes (see "Message Segmentation" on page 141).

- **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 141).

Field	Contents	Notes
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control	see "Command Segmentation" on page 142
Data[0]	Host IP address byte 3 (low)	The host address/port information shall only be included in the first segment. All data thereafter must start at Data[0]
Data[1]	Host IP address byte 2	
Data[2]	Host IP address byte 1	
Data[3]	Host IP address byte 0 (high)	
Data[4]	Host port number (low)	
Data[5]	Host port number (high)	
Data[6...n]	Data to send	

- **Response Details**

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

Command Details: IP_Add_Membership

Category

Advanced

Details

Command Code.:	19h
Valid for:	Instance

Description

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

- **Command Details**

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

- **Response Details**

  (no data)

Command Details: IP_Drop_Membership

Category

Advanced

Details

Command Code.: 1Ah

Valid for: Instance

Description

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

- **Command Details**

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

- **Response Details**

  (no data)

Command Details: DNS_Loopup

Category

Advanced

Details

Command Code.:	1Bh
Valid for:	Object Instance

Description

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

- **Command Details**

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

- **Response Details (Success)**

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

Socket Interface Error Codes (Object Specific)

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

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

9.7 SMTP Client Object (09h)

Category

Advanced

Object Description

This object groups functions related to the SMTP client.

See also...

- "File System" on page 19
- "E-mail Client" on page 29
- "Instance Attributes (Instance #13, SMTP Server)" on page 83
- "Instance Attributes (Instance #14, SMTP User)" on page 83
- "Instance Attributes (Instance #15, SMTP Password)" on page 83

Supported Commands

Object:	Get_Attribute Create Delete Send email from file("Command Details: Send Email From File" on page 106)
Instance:	Get_Attribute Set_Attribute Send email("Command Details: Send Email" on page 107)

Object Attributes (Instance #0)

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

Instance Attributes

Advanced

Instances are created dynamically by the application.

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

Command Details: Create

Category

Advanced

Details

Command Code.:	03h
Valid for:	Object

Description

This command creates an e-mail instance.

- **Command Details**

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

- **Response Details**

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

Command Details: Delete

Category

Advanced

Details

Command Code.:	04h
Valid for:	Object

Description

This command deletes an e-mail instance.

- **Command Details**

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

- **Response Details**

  (no data)

Command Details: Send Email From File

Category

Advanced

Details

Command Code.:	11h
Valid for:	Object

Description

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

File format:

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

```
[To]
recipient

[From]
sender

[Subject]
email subject

[Headers]
extra headers, optional

[Message]
actual email message
```

- **Command Details**

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

- **Response Details**

  (no data)

Command Details: Send Email

Category

Advanced

Details

Command Code.:	10h
Valid for:	Instance

Description

This command sends the specified e-mail instance.

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

Object Specific Error Codes

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

9.8 File System Interface Object (0Ah)

Category

Advanced

Object Description

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

Supported Commands

Object:	Get_Attribute
	Create("Command Details: Create" on page 110)
	Delete("Command Details: Delete" on page 111)
	Format Disc("Command Details: Format Disc" on page 120)

Instance:	Get_Attribute File Open("Command Details: File Open" on page 111) File Close("Command Details: File Close" on page 112) File Delete("Command Details: File Delete" on page 112) File Copy("Command Details: File Copy" on page 113) File Rename("Command Details: File Rename" on page 114) File Read("Command Details: File Read" on page 115) File Write("Command Details: File Write" on page 116) Directory Open("Command Details: Directory Open" on page 116) Directory Close("Command Details: Directory Close" on page 117) Directory Delete("Command Details: Directory Delete" on page 117) Directory Read("Command Details: Directory Read" on page 118) Directory Create("Command Details: Directory Create" on page 119) Directory Change("Command Details: Directory Change" on page 119)

Object Attributes (Instance #0)

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

Instance Attributes

Advanced

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

Command Details: Create

Category

Advanced

Details

Command Code.:	03h
Valid for:	Object

Description

This command creates a file operation instance.

- **Command Details**

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

- **Response Details**

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

Command Details: Delete

Category

Advanced

Details

Command Code.:	04h
Valid for:	Object

Description

This command deletes a file operation instance.

- **Command Details**

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

- **Response Details**

  (no data)

Command Details: File Open

Category

Advanced

Details

Command Code.:	10h
Valid for:	Instance

Description

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

- **Command Details**

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

- **Response Details**

  (no data)

Command Details: File Close

Category

Advanced

Details

Command Code.:	11h
Valid for:	Instance

Description

This command closes a previously opened file.

- **Command Details**
  (no data)

- **Response Details**

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

Command Details: File Delete

Category

Advanced

Details

Command Code.:	12h
Valid for:	Instance

Description

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

- **Command Details**

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

- **Response Details**
  (no data)

Command Details: File Copy

Category

Advanced

Details

Command Code.:	13h
Valid for:	Instance

Description

This command makes a copy of a file.

- **Command Details**

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

- **Response Details**
  (no data)

Command Details: File Rename

Category

Advanced

Details

Command Code:	14h
Valid for:	Instance

Description

This command renames or moves a file.

- **Command Details**

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

- **Response Details**
  (no data)

Command Details: File Read

Category

Advanced

Details

Command Code.:	15h
Valid for:	Instance

Description

Reads data from a file previously opened for reading.

- **Command Details**

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

- **Response Details**

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

Command Details: File Write

Category

Advanced

Details

Command Code.: 16h

Valid for: Instance

Description

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

- **Command Details**

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

- **Response Details**

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

Command Details: Directory Open

Category

Advanced

Details

Command Code.: 20h

Valid for: Instance

Description

This command opens a directory.

- **Command Details**

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

- **Response Details**

  (no data)

Command Details: Directory Close

Category

Advanced

Details

Command Code.:	21h
Valid for:	Instance

Description

This command closes a previously opened directory.

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

Command Details: Directory Delete

Category

Advanced

Details

Command Code.:	22h
Valid for:	Instance

Description

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

- **Command Details**

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

- **Response Details**
  (no data)

Command Details: Directory Read

Category

Advanced

Details

Command Code.:	23h
Valid for:	Instance

Description

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

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

- **Command Details**
  (no data)

- **Response Details**

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

Command Details: Directory Create

Category

Advanced

Details

Command Code.:	24h
Valid for:	Instance

Description

This command creates a directory.

- **Command Details**

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

- **Response Details**
  (no data)

Command Details: Directory Change

Category

Advanced

Details

Command Code.:	25h
Valid for:	Instance

Description

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

- **Command Details**

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

- **Response Details**
  (no data)

Command Details: Format Disc

Category

Advanced

Details

Command Code.:	30h
Valid for:	Object

Description

This command formats the file system.

- **Command Details**

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

- **Response Details**
  (no data)

Object Specific Error Codes

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

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 135)

9.10 CIP Port Configuration Object (0Dh)

Category

Advanced

Object Description

This object is used to populate and enumerate the CIP port object (see “Port Object (F4h)” on page 64) 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 127, 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.

Instance attribute values can only be changed while the module is in setup state, with the exception of attribute #7. Attribute #7 can be changed at runtime as well. 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 64 (CIP)
- “EtherNet/IP Host Object (F8h)” on page 127 (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

Advanced

#	Name	Access	Type	Description
1	Port Type	Set	UINT16	Enumerates the port (see CIP specification).
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.

See also...

- "Port Object (F4h)" on page 64 ("Instances Attributes (Instance #1)" on page 65)
- "Port Object (F4h)" on page 64 ("Instances Attributes (Instances #2... #8)" on page 65)

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

Network Specific Objects:

- "CIP Identity Host Object (EDh)" on page 125
- "EtherNet/IP Host Object (F8h)" on page 127
- "Ethernet Host Object (F9h)" on page 135

10.1 CIP Identity Host Object (EDh)

Category

Advanced

Object Description

This object allows for applications to support additional CIP identity instances. It is used to provide additional product identity information.

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.

Supported Commands

Instance:	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	Value	Comment
1	Vendor ID	Get	UINT16	-	These values replace the default values for the CIP Identity object.
2	Device type	Get	UINT16	-	
3	Product code	Get	UINT16	-	
4	Revision	Get	Struct of UINT8	-	
	Major revision		UINT8	-	
	Minor revision		UINT8	-	
5	Status	Get	UINT16	-	
6	Serial number	Get	UINT32	-	
7	Product name	Get	Array of CHAR	-	

Command Details: Get_Attribute_All

Category

Advanced

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**

Byte	Parameter	Data type	Description
0	Source ID	UINT8	Selected by the Anybus module
1	Destination object	UINT8	CIP identity host object
2, 3	Instance	UINT16	Instance number
4	Command	UINT8	50h = Get_Attribute_All (command bit set)
5	Data field size (in bytes)	UINT8	0
6	CmdExt[0]	UINT8	0
7	CmdExt[1]	UINT8	0

- **Response Details**

Byte	Parameter	Data type	Description
0	Source ID	UINT8	Selected by the Anybus module
1	Destination object	UINT8	CIP identity host object
2, 3	Instance	UINT16	Instance number
4	Command	UINT8	10h = Get_Attribute_All response
5	Data field size (in bytes)	UINT8	Number of bytes in the data field
6	CmdExt[0]	UINT8	0
7	CmdExt[1]	UINT8	0
0, 1	Vendor ID	UINT16	CIP identity host data
2, 3	Device type	UINT16	
4, 5	Product code	UINT16	
6	Major revision	UINT8	
7	Minor revision	UINT8	
8, 9	Status	UINT16	
10 - 13	Serial number	UINT32	
14 - n	Product name	Array of CHAR	

10.2 EtherNet/IP Host Object (F8h)

Category

Basic, extended, advanced

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 135.

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 141.

If the module is configured to use EtherNet/IP 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 recertification. See "Conformance Test Guide" on page 12.

See also...

- "Identity Object (01h)" on page 49 (CIP)
- "Assembly Object (04h)" on page 52 (CIP)
- "Port Object (F4h)" on page 64 (CIP)
- "CIP Port Configuration Object (0Dh)" on page 122 (Anybus Module Object)
- Anybus CompactCom Software Design Guide, "Error Codes"

Supported Commands

Object:	Process_CIP_Request (See "Command Details: Process_CIP_Object_Request" on page 131) Set_Configuration_Data (See "Command Details: Set_Configuration_Data" on page 132) Process_CIP_Routing_Request (See "Command Details: Process_CIP_Routing_Request" on page 133) Get_Configuration_Data (See "Command Details: Get_Configuration_Data" on page 134
Instance:	-

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 forwarded to the identity object (CIP). See also... - “Device Customization” on page 15 - “Identity Object (01h)” on page 49 (CIP-object) **Note:** Changing any of these attributes requires a new Vendor ID.
2	Device Type	Get	UINT16	002Bh	
3	Product Code	Get	UINT16	0036h	
4	Revision	Get	struct of: UINT8 Major UINT8 Minor	(software revision)	
5	Serial Number	Get	UINT32	(set at production)	
6	Product Name	Get	Array of CHAR	‘Anybus-CC EIP (2-Port) BB DLR’	

Extended

#	Name	Access	Type	Default Value	Comment
7	Producing Instance No.	Get	UINT16	0064h	See also... - “Instance 64h Attributes (Producing Instance)” on page 54 (CIP instance)
8	Consuming Instance No.	Get	UINT16	0096h	See also... - “Instance 96h Attributes (Consuming Instance)” on page 54 (CIP instance)
9	Enable communication settings from Net	Get	BOOL	True	Value:Meaning: True Can be set from network False Cannot be set from network See also... - “TCP/IP Interface Object (F5h)” on page 66 (CIP object) - “Ethernet Link Object (F6h)” on page 69 (CIP object) - “Network Configuration Object (04h)” on page 78 (Anybus Module Object)
12	Enable Parameter Object	Get	BOOL	True	Value:Meaning: 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 52 (CIP instance)
14	Listen-Only heartbeat instance number	Get	UINT16	0004h	See also... - “Instance 04h Attributes (Heartbeat, Listen-Only)” on page 53 (CIP instance)

#	Name	Access	Type	Default Value	Comment
15	Assembly object Configuration instance number	Get	UINT16	0005h	See also... - “Instance 05h Attributes (Configuration Data)” on page 53 (CIP instance)
16	Disable Strict IO Match	Get	BOOL	False	If true, the module will accept Class1 connection requests that have sizes that are less than or equal to the configured IO sizes.
18	Input-Only extended heart-beat instance number	Get	UINT16	0006h	See also... - “Instance 06h Attributes (Heartbeat, Input-Only Extended)” on page 53 (CIP instance)
19	Listen-Only extended heart-beat instance number	Get	UINT16	0007h	See also... - “Instance 07h Attributes (Heartbeat, Listen-Only Extended)” on page 54 (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
26	Enable EtherNet/IP QuickConnect[a]	Get	BOOL	False	Value:Meaning: True EtherNet/IP QuickConnect enabled False EtherNet/IP QuickConnect 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 which will require a certification. See “Conformance Test Guide” on page 12.

Advanced

#	Name	Access	Type	Default Value	Comment
11	Enable CIP forwarding	Get	BOOL	False	Value:Meaning: True Enable CIP forwarding False Disable CIP forwarding See also... - “Command Details: Process_CIP_Object_Request” on page 131.
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.
27	Producing Instance Map	Get	Array of UINT		List of producing assembly instances and their sizes. This list can hold a maximum of 6 entries. Using this attribute splits up the process data area into areas as defined by this attribute. The application is responsible for mapping and maintaining the process data map. This attribute will be ignored if attribute #7 is implemented.
28	Consuming Instance Map	Get	Array of UINT		List of consuming assembly instances and their sizes. This list can hold a maximum of 6 entries. Using this attribute splits up the process data area into areas as defined by this attribute. The application is responsible for mapping and maintaining the process data map. This attribute will be ignored if attribute #8 is implemented.
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.

Command Details: Process_CIP_Object_Request

Category

Advanced

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 up to 6 pending CIP requests; 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 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 Padded 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

Advanced

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.

Since the telegram length on the host interface is limited, segmentation is needed for data sizes larger than 255 bytes. 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 141
MsgData[0 - 1][a]	Producing connection point	Producing connection point, requested by the originator.
MsgData[2 - 3][a]	Consuming connection point	Consuming connecition 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)
MsgData[0]	Status	00h: Accepted

- **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 133.
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 55 (CIP)
- "Message Segmentation" on page 141

Command Details: Process_CIP_Routing_Request

Category

Advanced

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 Unconneced_Send service with updated route path information.

The module supports up to 6 pending requests. 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]	Timeout_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 64 (CIP)
- "CIP Port Configuration Object (0Dh)" on page 122

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.

Since the telegram length on the host interface is limited, segmentation is needed for data sizes larger than 255 bytes. 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 141
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 141
MsgData[0]	Status	Anybus protocol error code

See also...

"Message Segmentation" on page 141

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 preprogrammed 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 preprogrammed 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	Value:Meaning: True HICP enabled False HICP disabled (see "HICP (Host IP Configuration Protocol)" on page 144)
3	Enable Web Server	Get	BOOL	True	Value:Meaning: True web server enabled False web server disabled (see "Web Server" on page 22)
5	Enable Web ADI access	Get	BOOL	True	Value:Meaning: True web ADI access enabled False web ADI access disabled (see "Web Server" on page 22)
6	Enable FTP server	Get	BOOL	True	Value:Meaning: True FTP server enabled False FTP server disabled (see "FTP Server" on page 20)
7	Enable admin mode	Get	BOOL	False	Value:Meaning: True FTP Admin mode enabled False FTP Admin mode disabled (see "FTP Server" on page 20)
8	Network Status	Set	UINT16	-	See "Network Status" on page 136
12	Port 1 State	Get	ENUM	0 (Enabled)	State of Ethernet port 1 0: Enabled 1: Disabled
13	Port 2 State	Get	ENUM	0 (Enabled)	State of Ethernet port 2 0: Enabled 1: Disabled

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

Network Status

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

Bit	Contents	Description
0	Link	Value:Meaning: True Link sensed False No link

Bit	Contents	Description
1	IP established	<u>Value:Meaning:</u> True IP address established False IP address not established
2... 15	(reserved)	(mask off and ignore)

A. Categorization of Functionality

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

A.1 Basic

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

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

A.2 Extended

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

A.3 Advanced

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

B. Implementation Details

B.1 Extended LED Functionality

On the Anybus CompactCom EtherNet/IP module, there is the possibility to use GIP1 and GOP1 for indicating link/activity on the 2 ports of the module.

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

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

GPIO mode description

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

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

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

Note 3: LED behavior is described in chapter 1. See “LINK/Activity LED 5/6” on page 11

B.2 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.3 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.	

Anybus State	Implementation	Comment
EXCEPTION	Unexpected error, e.g. watchdog timeout etc.	MS LED turns red (to indicate a major fault) NS LED is off

B.4 Application Watchdog Timeout Handling

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

C. Message Segmentation

C.1 General

Category: Advanced

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

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

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

C.2 Command Segmentation

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

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

Command segmentation is used for the following commands:

- Set_Configuration_Data (see "Command Details: Set_Configuration_Data" on page 132)
- Get_Configuration_Data (see "Command Details: Get_Configuration_Data" on page 134)
- Send (see "Command Details: Send" on page 97)
- Send To (see "Command Details: Send_To" on page 98)

Segmentation Control bits (Command)

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

Segmentation Control bits (Response)

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

When issuing a segmented command, the following rules apply:

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

C.3 Response Segmentation

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

Response segmentation is used for responses to the following commands:

- Receive (object specific, see "Command Details: Receive" on page 95)
- Receive From (object specific, see "Command Details: Receive_From" on page 96)

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.

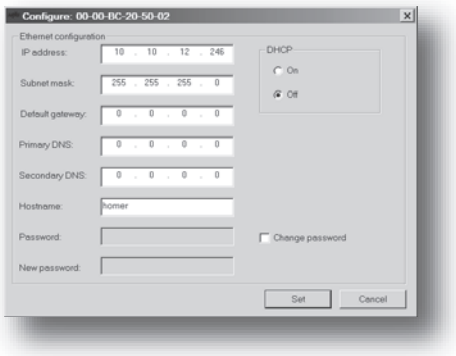
D. HICP (Host IP Configuration Protocol)

D.1 General

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

D.2 Operation

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


Configure: 00-00-BC-20-50-02

Ethernet configuration

IP address:	10 . 10 . 12 . 246
Subnet mask:	255 . 255 . 255 . 0
Default gateway:	0 . 0 . 0 . 0
Primary DNS:	0 . 0 . 0 . 0
Secondary DNS:	0 . 0 . 0 . 0
Hostname:	homer
Password:	
New password:	

DHCP: On [✓ Off]

☐ Change password

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

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

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

E. Technical Specification

E.1 Protective Earth (PE) Requirements

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

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

E.2 Power Supply

Supply Voltage

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

Power Consumption

The Anybus CompactCom 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 platform, consult the general Anybus CompactCom Hardware Design Guide.

The current hardware design consumes up to 380 mA[1].

E.3 Environmental Specification

Consult the Anybus CompactCom Hardware Design Guide for further information.

E.4 EMC Compliance

Consult the Anybus CompactCom Hardware Design Guide for further information.

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

F. Copyright Notices