User Guide

Anybus® CompactCom 40 Common Ethernet

Doc.Id. HMSI-27-269
Rev. 1.10

Important User Information

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

The reader of this document is expected to be familiar with high level software design, and communication systems in general. The use of advanced EtherNet/IP-specific functionality may require in-depth knowledge in EtherNet/IP networking internals and/or information from the official EtherNet/IP specifications. In such cases, the people responsible for the implementation of this product should either obtain the EtherNet/IP specification to gain sufficient knowledge or limit their implementation in such a way that this is not necessary.

Liability

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

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

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

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

Intellectual Property Rights

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

Trademark Acknowledgements

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

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

Table of Contents

Appendix E Technical Specification

Appendix F Copyright Notices

P. About This Document

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

P.1 Related Documents

Document	Author
Anybus CompactCom 40 Software Design Guide	HMS
Anybus CompactCom M40 Hardware Design Guide	HMS
Anybus CompactCom 40 Network Guides	

P.2 Document History

Summary of Recent Changes (1.06... 1.10)

Change	Page(s)
Updated JSON chapter	47
Updated Ethernet Host Object (F9h)	106
Changed advanced to extended, removed category advanced	

Revision List

Revision	Date	Author(s)	Chapter(s)	Description
0.95	2014-06-17	KeL	-	Preliminary
1.00	2014-09-09	KaD	All	First release
1.01	2014-09-10	KaD	7	Minor corrections
1.05	2014-09-22	KaD	1, 2, 3, E	Major update
1.06	2015-01-22	KaD	F	Minor update
1.10	2015-11-04	KeL	8, 10	Minor update

P.3 Conventions & Terminology

The following conventions are used throughout this manual:

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

P.4 Support

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

1. About the Anybus CompactCom 40 Common Ethernet

1.1 General

The Anybus CompactCom 40 Common Ethernet communication module provides a common platform for the Industrial Ethernet networks, that are supported in the Anybus CompactCom 40 product family. The communication software for the selected network is downloaded to the module and the module will seamlessly integrate to the network. It is also possible to use the module as a general Ethernet interface, using the included TCP/IP stack (See "Socket Interface Object (07h)" on page 69).

The modular approach of the Anybus CompactCom 40 platform allows the module to be customized, allowing the end product to appear as a vendor-specific implementation rather than a generic Anybus module.

This product conforms to all aspects of the host interface for Anybus CompactCom 40 modules defined in the Anybus CompactCom 40 Hardware and Software Design Guides, making it fully interchangeable with any other device following that specification. Generally, no additional network related software support is needed, however in order to be able to take full advantage of advanced network specific functionality, a certain degree of dedicated software support may be necessary.

1.2 Features

- Two Ethernet ports
- Ethernet RJ45 connectors
- Common hardware platform for Ethernet networks
- 10/100 Mbit, full/half duplex operation
- Web server w. customizable content
- FTP server
- E-mail client
- JSON functionality
- Server Side Include (SSI) functionality
- Transparent Socket Interface

2. Basic Operation

2.1 General Information

At delivery the module supports general Ethernet functionality, such as e.g. FTP and web server. The built in socket interface allows additional protocols to be implemented on top of TCP/IP, see "Socket Interface (Advanced Users Only)" on page 11.

Firmware to enable communication with another network, can be downloaded in different ways, see "Anybus CompactCom 40 Software Design Guide, "Application Data Object (FEh)"" on page 10.

2.2 Software Requirements

Generally, no additional network support code needs to be written in order to support the Anybus CompactCom 40 Common Ethernet. However, 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.

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

See also...

- "Diagnostic Object (02h)" on page 58 (Anybus Module Objects)
- Anybus CompactCom 40 Software Design Guide, "Application Data Object (FEh)"

2.3 Device Customization

2.3.1 Network Identity

By default, the module uses the following identity settings:

- Vendor ID: 005Ah (HMS Industrial Networks)
- Device Type: 002Bh (Generic Device)
- Product Code: 00A3h (Common Ethernet)
- Product Name: 'Anybus CompactCom 40 Common Ethernet'

Depending on network downloaded, it is possible to customize the identity of the module by implementing the corresponding instance attributes in the respective network Host Objects.

2.3.2 Web Interface

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

See also...

- "File System" on page 15
- "FTP Server" on page 19
- "Web Server" on page 21
- "Server Side Include (SSI)" on page 29
- "JSON" on page 47

2.3.3 Socket Interface (Advanced Users Only)

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

See also...

- "Socket Interface Object (07h)" on page 69 (Anybus Module Object)
- "Message Segmentation" on page 125

2.4 Communication Settings

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

In this case, this includes...

- **TCP/IP settings**

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

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

- **Physical Link Settings**

  By default, the module uses auto negotiation to establish the physical link settings, however it is possible to force a specific setting if necessary.
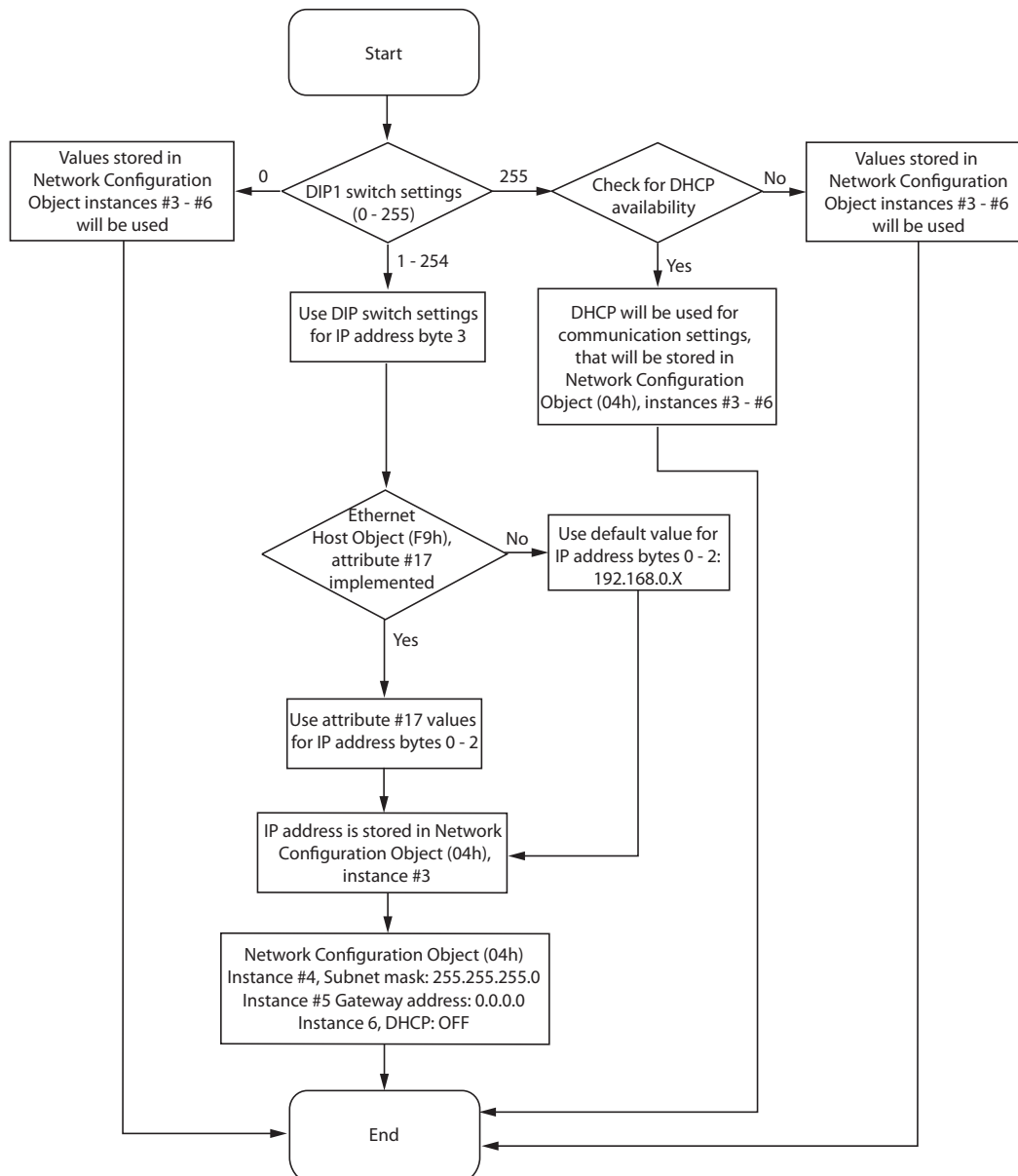
The parameters in the Network Configuration Object (04h) are available from the network through the built in web server.

See also...

- "Web Server" on page 21
- "Network Configuration Object (04h)" on page 60 (Anybus Module Object)
- "Secure HICP (Secure Host IP Configuration Protocol)" on page 128

2.4.1 Communication Settings in Stand Alone Shift Register Mode

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

```
                              ┌─────────────┐
                              │    Start    │
                              └──────┬──────┘
                                     │
  ┌─────────────────┐         ╱──────▼──────╲        255    ╱──────────────╲   No    ┌──────────────────┐
  │ Values stored in │   0   ╱ DIP1 switch    ╲─────────────╱ Check for DHCP ╲──────▶│ Values stored in  │
  │ Network Config.  │◀─────▶ settings         ├────────────▶ availability     │      │ Network Config.   │
  │ Object instances │       ╲ (0 - 255)      ╱              ╲                ╱       │ Object instances  │
  │ #3 - #6 will be  │        ╲──────┬────────╱                ╲──────┬───────╱        │ #3 - #6 will be   │
  │ used             │               │                               │ Yes           │ used              │
  └─────────────────┘          1 - 254                                ▼               └──────────────────┘
                                     │                   ┌──────────────────────────┐
                          ┌──────────▼─────────┐         │ DHCP will be used for     │
                          │ Use DIP switch      │         │ communication settings,   │
                          │ settings for IP     │         │ that will be stored in    │
                          │ address byte 3      │         │ Network Configuration     │
                          └──────────┬─────────┘         │ Object (04h), instances   │
                                     │                   │ #3 - #6                   │
                                     │                   └──────────────────────────┘
                          ╱──────────▼─────────╲   No    ┌──────────────────────┐
                         ╱ Ethernet Host Object ╲───────▶│ Use default value for │
                         ╲ (F9h), attribute #17  ╱        │ IP address bytes 0-2: │
                          ╲ implemented         ╱         │ 192.168.0.X           │
                           ╲─────────┬─────────╱          └──────────────────────┘
                                     │ Yes
                          ┌──────────▼─────────┐
                          │ Use attribute #17  │
                          │ values for IP      │
                          │ address bytes 0-2  │
                          └──────────┬─────────┘
                                     │
                          ┌──────────▼──────────────┐
                          │ IP address is stored in   │
                          │ Network Configuration     │
                          │ Object (04h), instance #3 │
                          └──────────┬──────────────┘
                                     │
                          ┌──────────▼───────────────────────┐
                          │ Network Configuration Object (04h) │
                          │ Instance #4, Subnet mask:          │
                          │ 255.255.255.0                      │
                          │ Instance #5 Gateway address:0.0.0.0│
                          │ Instance 6, DHCP: OFF              │
                          └──────────┬───────────────────────┘
                                     │
                              ┌──────▼──────┐
                              │     End     │
                              └─────────────┘
```

See also...

- "Ethernet Host Object (F9h)" on page 106
- Anybus CompactCom M40 Hardware Design Guide
- "Network Configuration Object (04h)" on page 60

2.5 Diagnostics

Events and their severity value are recorded in the Diagnostic Object.

See also...

- "Diagnostic Object (02h)" on page 58 (Anybus Module Object)

2.6 Network Data Exchange

The Anybus CompactCom 40 Common Ethernet does not perform any network specific handling of the process data.

For information on network data exchange for supported networks, please consult the respective network guides, available at the support pages at www.anybus.com.
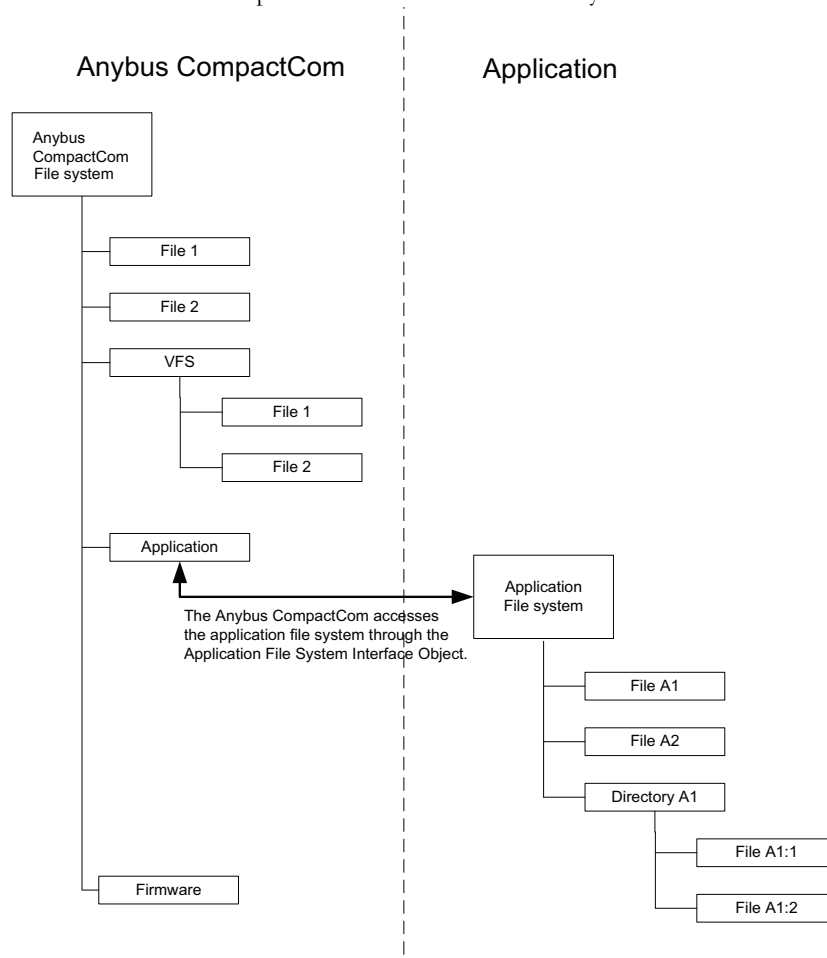
2.7 File System

Category: Extended

2.7.1 Overview

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

- VFS - The virtual file system that e.g. holds the Anybus default web pages of the module.
- Application - This directory provides access to the application file system through the Application File System Interface Object (EAh).
- Firmware - Firmware updates are stored in this directory.

2.7.2 General Information

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

The file system uses the following conventions:

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

See also...

- "FTP Server" on page 19
- "Web Server" on page 21
- "E-mail Client" on page 28
- "Server Side Include (SSI)" on page 29
- "Anybus File System Interface Object (0Ah)" on page 91
- "Application File System Interface Object (EAh)" on page 110

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

The following operations will erase one or more flash segments:

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

2.7.3 System Files

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

Example

```
[Key1]
value of Key1

[Key2]
value of Key2
```

3. Firmware Download / Firmware Upgrade

Download and upgrade of network communication firmware for a specific industrial Ethernet network can be performed in different ways.

3.1 Using Firmware Manager II

This tool is available without cost from the HMS website, www.anybus.com. It can be used to download new firmware or firmware upgrades for all Anybus CompactCom modules.
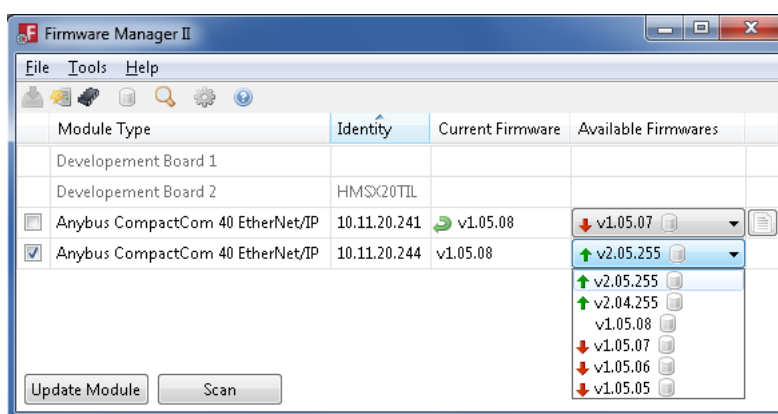
Using the tool, perform the following steps to download new firmware to the module.

1. Connect a computer with the Firmware Manager II software installed to the network containing the module.

2. Start the Firmware Manager II tool.

3. Scan the network and find the module.

4. Click the Firmware Repository icon in the menu, to open the Firmware Repository window. Drag the firmware folder into the window to add the new firmware to the repository. Close the Firmware Repository window.

5. In the scan window, under the "Available Networks" tab, select the appropriate firmware for the module. Click the "Change Network" button. A confirmation window will appear. Clicking "Yes" will start the download of the new firmware.

6. After download, a restart of the module is needed to install the new firmware. If the application allows it, it is possible to restart the module via the "Restart Module" button in the Firmware Manager II tool. If the application does not allow restart from the network, a manual restart of the module is needed.

For more information, see the help file in the Firmware Manager II software.

3.2 Via the Internal File System

The internal file system of the module includes a directory (/firmware), where the new firmware or the upgrade is stored. The file system can be accessed through FTP or via the File System Interface Object. The next time the module is started, the new firmware will be installed. After the firmware is installed, the firmware file is deleted from the /firmware directory.

See also ...

- "File System" on page 15
- "FTP Server" on page 19
- "Application File System Interface Object (EAh)" on page 110

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. It can be disabled using attribute #6 in the Ethernet Host Object (F9h), see page 106.

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

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

The FTP server supports up to two concurrent clients.

4.2 User Accounts

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

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

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

File Format ('\ftp.cfg'):
```
User1:Password1:Homedirectory1
User2:Password2:Homedirectory2
.
.
UserN:PasswordN:HomedirectoryN
\path\userlistA:HomedirectoryA
\path\userlistB:HomedirectoryB
```

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

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

Notes:

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

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

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

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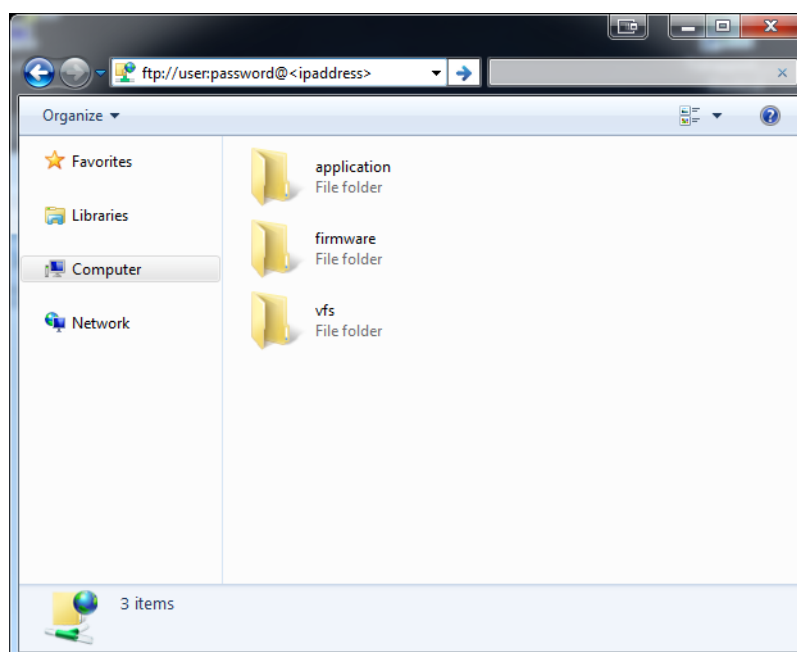
1. Apart from the vfs folder, that is read-only.

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 JSON, SSI, and client-side scripting allows access to objects and file system data, enabling the creation of advanced graphical user interfaces.

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

See also...

- "FTP Server" on page 19
- "JSON" on page 47
- "Server Side Include (SSI)" on page 29
- "Ethernet Host Object (F9h)" on page 106

5.2 Default Web Pages

The default web pages provide access to:

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

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

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

See also...

- "File System" on page 15
- "Anybus File System Interface Object (0Ah)" on page 91

5.2.1 Network Configuration

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

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

Available editable settings will be explained on the next page.

IP Configuration

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

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

Ethernet Configuration

Changes will take effect immediately.

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

SMTP Settings

The module needs a reset before any changes take effect

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

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.

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

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

a. Not provided with current PHY interface.

5.3 Server Configuration

5.3.1 General Information

Category: extended

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

File Format:

```
[WebRoot]
\web

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

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

Web Root Directory

The web server cannot access files outside this directory.

Content Types

A list of file extensions and their reported content types.

See also...

- "Default Content Types" on page 26

SSI File Types

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

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

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 21

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 25.

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


[AuthName]
(message goes here)
```

List of approved users.

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

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

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

Example:

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

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

[AuthType]
Basic

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

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

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

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 60
- "SMTP Client Object (09h)" on page 86

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)

Note: See "SMTP Client Object (09h)" on page 86 for more information.

Sending a message based on a file in the file system is achieved using the 'Send Email from File'-command. For a description of the file format, see "Command Details: Send Email From File" on page 89.

7. Server Side Include (SSI)

7.1 General Information

Category: extended

Server Side Include functionality, or SSI, allows data from files and objects to be represented on web pages and in e-mail messages.[1]

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

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

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

---

1. JSON offers more functionality when it comes to web pages, but is also more complex to use, see "JSON" on page 47.

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 Email Messages	Page
GetConfigItem()	Yes	31
SetConfigItem()	No	32
SsiOutput()	Yes	34
DisplayRemoteUser	No	34
ChangeLanguage()	No	35
IncludeFile()	Yes	36
SaveDataToFile()	No	37
printf()	Yes	38
scanf()	No	40

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 behavior 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...

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 <u>not</u> 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:
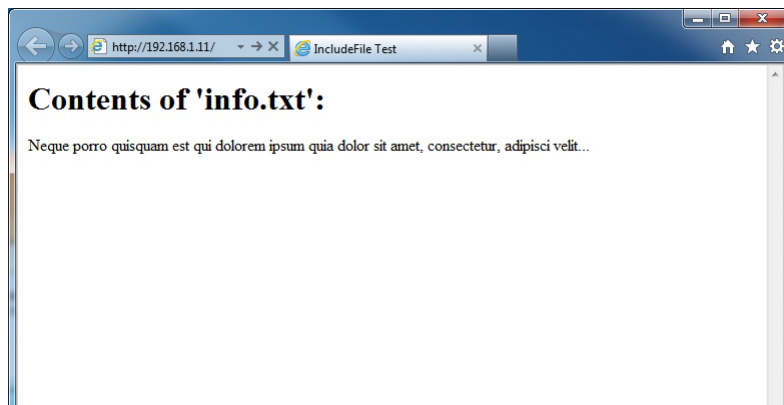
See also...

•   "Include File" on page 29

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: This command cannot be used in email messages.

Syntax:

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

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

Default Output:

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

Example:

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

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

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

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

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

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

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

</BODY>
</HTML>
```

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

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

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 39

| 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 42

Default Output:

Scenario	Default Output
Success	(printf() result)
ABCCMessage error	ABCCMessage error string ("Errors" on page 45)

Example:

See also...

- "ABCCMessage()" on page 42
- "Example (Get_Attribute):" on page 44

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	Meaning
hh	Argument is interpreted as SINT8 or UINT8
h	Argument is interpreted as SINT16 or UINT16
l	Argument is interpreted as SINT32 or UINT32

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 email messages.

Syntax:

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

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

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

Formatting tags are written as follows:

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

See also...

  - • "Formatting Tags" on page 41

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 42

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 45)

Example:

See also...

  - "ABCCMessage()" on page 42
  - "Example (Set_Attribute):" on page 44

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

- *** (Optional)**

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

- **Width (Optional)**

    Specifies the maximum number of characters to be read.

- **Modifier (Optional)**

    Specifies a different data size.

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

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()	-	42

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 43

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

See also...

- "Response Data Types (r_type)" on page 43

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...

- **Command Data Types (c_type)**

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

Type	Supports Arrays	Data format (as supplied in msgdata)
BOOL	Yes	1
SINT8	Yes	-25
SINT16	Yes	2345
SINT32	Yes	-2569
UINT8	Yes	245
UINT16	Yes	40000
UINT32	Yes	32
CHAR	Yes	A
STRING	No	"abcde" **Note:** Quotes can be included in the string if preceded by backslash('\') Example: *"We usually refer to it as \"the Egg\" "*
FLOAT	Yes	5.6538e2
BITS8	Yes	8-bit field
BITS16	Yes	16-bit field
BITS32	Yes	32-bit field
OCTET	Yes	8-bit field
BIT1 - 7	Yes	1-bit to 7-bit field
PAD0 - 16	Yes	0 - 16-bit field, for filling up a string to a predefined size
NONE	No	Command holds no data, hence no data type

- **Response Data Types (r_type)**

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

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

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

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

Example (Get_Attribute):

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

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

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

See also...

- "printf()" on page 38

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 40

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...

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 45

• 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 34

8. JSON

8.1 General Information

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

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

Access

The JSON resources should be password protected. Add password protection by adding a file called web_accs.cfg in the root directory. See "Authorization" on page 26 for more information.

8.2 JSON Objects

8.2.1 ADI

info.json

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

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

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

JSON object layout:

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

data.json

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

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

JSON object layout:

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

metadata.json

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

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

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

JSON object layout:

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

enum.json

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

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

Name	Data Type	Note
string	String	-
value	Number	-

JSON object layout:

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

update.json

POST adi/update.json - form data:
inst=<instance>&value=<data>[&elem=<element>][&callback=<function>].

Updates the value of an ADI for the specified ADI instance <instance>. The value, <data>, shall be hex formatted (see "Hex Format Explained" on page 55 for more information). If <element> is specified, only the value of the specified element is updated. In this case, <data> shall only update that single element value. When <element> is not specified, <data> shall represent the entire array value. Optionally, a callback may be passed to the request for JSONP output.

Name	Data Type	Note
result	Number	0 = success

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

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

8.2.2 Module

info.json

GET module/info.json.

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

JSON object layout:

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

8.2.3 Network

ethstatus.json

GET network/ethstatus.json.

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

Comm Object Definition:

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

JSON object layout:

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

ipstatus.json & ipconf.json

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

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

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

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

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

Example:

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

ethconf.json

GET network/ethconf.json.

Name	Data Type	Note
comm1	Number	-
comm2	Number	-

ifcounters.json

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

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

mediacounters.json

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

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

nwstats.json

GET network/nwstats.json.

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

Example output:

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

Get network specific statistics:

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

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

8.2.4 Services

smtp.json

GET services/smtp.json.

Note: Password is not returned when retrieving the settings.

Name	Data Type	Note
server	String	-
user	String	-

8.2.5 Hex Format Explained

The metadata max and min fields and the ADI values are ABP data encoded in a hex format. If the data type is an integer, the endianness used is determined by the data format field found in adi/info.json (see "info.json" on page 47).

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

```
0500
```

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

```
414243
```

8.3 Example

This example shows how to create a web page that fetches Module Name and CPU load from the module and presents it on the web page. The file, containing this code, has to be stored in the built-in file system, see "File System" on page 15, and the result can be seen in a common browser.

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

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

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

9. Anybus Module Objects

9.1 General Information

This chapter specifies the Anybus Module Object implementation.

Standard Objects:

- "Anybus Object (01h)" on page 57
- "Diagnostic Object (02h)" on page 58
- "Network Object (03h)" on page 59
- "Network Configuration Object (04h)" on page 60

Network Specific Objects:

- "Socket Interface Object (07h)" on page 69
- "SMTP Client Object (09h)" on page 86
- "Anybus File System Interface Object (0Ah)" on page 91
- "Network Ethernet Object (0Ch)" on page 104

9.2 Anybus Object (01h)

Category

Basic

Object Description

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

Supported Commands

Object: Get_Attribute

Instance: Get_Attribute
                 Set_Attribute
                 Get_Enum_String

Object Attributes (Instance #0)

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

Instance Attributes (Instance #1)

Basic

#	Name	Access	Type	Value
1	Module type	Get	UINT16	0403h (Standard Anybus 40 CompactCom)
2... 11	-	-	-	Consult the general Anybus CompactCom 40 Software Design Guide for further information.
12	LED colors	Get	struct of: UINT8(LED1A) UINT8(LED1B) UINT8(LED2A) UINT8(LED2B)	Value:Color: 01h Green 02h Red 01h Green 02h Red
13... 16	-	-	-	Consult the general Anybus CompactCom 40 Software Design Guide for further information.

Extended

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

9.3 Diagnostic Object (02h)

General Information

Basic

Object Description

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

Supported Commands

Object: Get_Attribute
                 Create
                 Delete

Instance: Get_Attribute

Object Attributes (Instance #0)

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

Instance Attributes

Basic

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

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

See also...

- "Diagnostics" on page 13

9.4 Network Object (03h)

Category

Basic

Object Description

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

Supported Commands

Object: Get_Attribute

Instance: Get_Attribute
                 Set_Attribute
                 Get_Enum_String
                 Map_ADI_Write_Area
                 Map_ADI_Read_Area
                 Map_ADI_Write_Ext_Area
                 Map_ADI_Read_Ext_Area

Object Attributes (Instance #0)

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

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

Instance Attributes (Instance #1)

#	Name	Access	Type	Value
1	Network type	Get	UINT16	00A3h
2	Network type string	Get	Array of CHAR	'Common Ethernet"
3	Data format	Get	ENUM	00h (LSB first)
4	Parameter data support	Get	BOOL	0 (False)
5 - 7	-	-	-	Consult the general Anybus CompactCom 40 Software Design Guide for further information.

9.5 Network Configuration Object (04h)

Category

Extended

Object Description

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

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

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

See also...

- • "Communication Settings" on page 12
- • "E-mail Client" on page 28

Supported Commands

Object:	Get_Attribute
	Reset
Instance:	Get_Attribute
	Set_Attribute
	Get_Enum_String

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1	Name	Get	Array of CHAR	'Network Configuration'
2	Revision	Get	UINT8	01h
3	Number of instances	Get	UINT16	000Fh (15)
4	Highest instance no.	Get	UINT16	0011h (17)

Instance Attributes (Instance #3, IP Address)

Extended

#	Name	Access	Type	Description
1	Name[a]	Get	Array of CHAR	'IP address'
2	Data type	Get	UINT8	04h (= UINT8)
3	Number of elements	Get	UINT8	04h (four elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of UINT8	Any change is valid after reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)
6	Configured Value	Get	Array of UINT8	Holds the configured value, which will be written to attribute #5 after the module has been reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)

a. Multilingual, see "Multilingual Strings" on page 68.

Instance Attributes (Instance #4, Subnet Mask)

Extended

#	Name	Access	Type	Description
1	Name[a]	Get	Array of CHAR	'Subnet mask'
2	Data type	Get	UINT8	04h (= UINT8)
3	Number of elements	Get	UINT8	04h (four elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of UINT8	Any change is valid after reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)
6	Configured Value	Get	Array of UINT8	Holds the configured value, which will be written to attribute #5 after the module has been reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)

a. Multilingual, see "Multilingual Strings" on page 68.

Instance Attributes (Instance #5, Gateway Address)

Extended

#	Name	Access	Type	Description
1	Name[a]	Get	Array of CHAR	'Gateway'
2	Data type	Get	UINT8	04h (= UINT8)
3	Number of elements	Get	UINT8	04h (four elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of UINT8	Any change is valid after reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)
6	Configured Value	Get	Array of UINT8	Holds the configured value, which will be written to attribute #5 after the module has been reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)

a. Multilingual, see "Multilingual Strings" on page 68.

Instance Attributes (Instance #6, DHCP Enable)

Extended

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

a. Multilingual, see "Multilingual Strings" on page 68.

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

Changes have immediate effect.

Extended

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

a. Multilingual, see "Multilingual Strings" on page 68.

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

Changes have immediate effect.

Extended

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

a. Multilingual, see "Multilingual Strings" on page 68.

Instance Attributes (Instance #9, DNS1)

This instance holds the address to the primary DNS server.

Extended

#	Name	Access	Type	Description
1	Name[a]	Get	Array of CHAR	'DNS1'
2	Data type	Get	UINT8	04h (= UINT8)
3	Number of elements	Get	UINT8	04h
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of UINT8	Any change is valid after reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)
6	Configured Value	Get	Array of UINT8	Holds the configured value, which will be written to attribute #5 after the module has been reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)

a. Multilingual, see "Multilingual Strings" on page 68.

Instance Attributes (Instance #10, DNS2)

This instance holds the address to the secondary DNS server.

Extended

#	Name	Access	Type	Description
1	Name[a]	Get	Array of CHAR	'DNS2'
2	Data type	Get	UINT8	04h (= UINT8)
3	Number of elements	Get	UINT8	04h
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of UINT8	Any change is valid after reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)
6	Configured Value	Get	Array of UINT8	Holds the configured value, which will be written to attribute #5 after the module has been reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)

a. Multilingual, see "Multilingual Strings" on page 68.

Instance Attributes (Instance #11, Host name)

This instance holds the host name of the module.

Extended

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

a. Multilingual, see "Multilingual Strings" on page 68.

Instance Attributes (Instance #12, Domain name)

This instance holds the domain name.

Extended

#	Name	Access	Type	Description
1	Name[a]	Get	Array of CHAR	'Domain name'
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	30h
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of CHAR	Any change is valid after reset. Domain name, 48 characters
6	Configured Value	Get	Array of CHAR	Holds the configured value, which will be written to attribute #5 after the module has been reset. Host name, 48 characters

a. Multilingual, see "Multilingual Strings" on page 68.

Instance Attributes (Instance #13, SMTP Server)

This instance holds the SMTP server address.

Extended

#	Name	Access	Type	Description
1	Name[a]	Get	Array of CHAR	'SMTP Server'
2	Data type	Get	UINT8	04h (= UINT8)
3	Number of elements	Get	UINT8	40h
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of CHAR	Any change is valid after reset. SMTP server address, 64 characters
6	Configured Value	Get	Array of CHAR	Holds the configured value, which will be written to attribute #5 after the module has been reset. SMTP server address, 64 characters

a. Multilingual, see "Multilingual Strings" on page 68.

Instance Attributes (Instance #14, SMTP User)

This instance holds user name for the SMTP account.

Extended

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

a. Multilingual, see "Multilingual Strings" on page 68.

Instance Attributes (Instance #15, SMTP Password)

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

Extended

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

a. Multilingual, see "Multilingual Strings" on page 68.

Instance Attributes (Instance #16, MDI 1 settings)

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

Extended

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

Instance Attributes (Instance #17, MDI 2 settings)

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

Extended

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

Instance Attributes (Instances #18 and #19)

These instances are reserved for future attributes.

Multilingual Strings

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

Instance	English	German	Spanish	Italian	French
3	IP address	IP-Adresse	Dirección IP	Indirizzo IP	Adresse IP
4	Subnet mask	Subnetzmaske	Masac. subred	Sottorete	Sous-réseau
5	Gateway	Gateway	Pasarela	Gateway	Passerelle
6	DHCP	DHCP	DHCP	DHCP	DHCP
	Enable	Einschalten	Activado	Abilitato	Activé
	Disable	Ausschalten	Desactivado	Disabilitato	Désactivé
7	Comm 1	Komm 1	Comu 1	Connessione 1	Comm 1
	Auto	Auto	Auto	Auto	Auto
	10 HDX	10 HDX	10 HDX	10 HDX	10 HDX
	10 FDX	10 FDX	10 FDX	10 FDX	10 FDX
	100 HDX	100 HDX	100 HDX	100 HDX	100 HDX
	100 FDX	100FDX	100 FDX	100 FDX	100 FDX
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
	100 FDX	100FDX	100 FDX	100 FDX	100 FDX
9	DNS1	DNS 1	DNS Primaria	DNS1	DNS1
10	DNS2	DNS 2	DNS Secundia.	DNS2	DNS2
11	Host name	Host name	Nombre Host	Nome Host	Nom hôte
12	Domain name	Domain name	Nobre Domain	Nome Dominio	Nom Domaine
13	SMTP Server	SMTP Server	Servidor SMTP	Server SMTP	SMTP serveur
14	SMTP User	SMTP User	Usuario SMTP	Utente SMTP	SMTP utilisa.
15	SMTP Pswd	SMTP PSWD	Clave SMTP	Password SMTP	SMTP mt passe

9.6 Socket Interface Object (07h)

Category

Extended

Object Description

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

Note that some of the commands used when accessing this object may require segmentation. For more information, see "Message Segmentation" on page 125.

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 71)
	Delete (See "Command Details: Delete" on page 72)
Instance:	Get_Attribute
	Set_Attribute
	Bind (See "Command Details: Bind" on page 73)
	Shutdown (See "Command Details: Shutdown" on page 74)
	Listen (See "Command Details: Listen" on page 75)
	Accept (See "Command Details: Accept" on page 76)
	Connect (See "Command Details: Connect" on page 77)
	Receive (See "Command Details: Receive" on page 78)
	Receive_From (See "Command Details: Receive_From" on page 79)
	Send (See "Command Details: Send" on page 80)
	Send_To (See "Command Details: Send_To" on page 81)
	IP_Add_membership (See "Command Details: IP_Add_Membership" on page 82)
	IP_Drop_membership (See "Command Details: IP_Drop_Membership" on page 83)
	DNS_Lookup (See "Command Details: DNS_Lookup" on page 84)

Object Attributes (Instance #0)

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

Instance Attributes (Sockets #1...8)

Extended

#	Name	Access	Type	Description
1	Socket type	Get	UINT8	Value:Socket Type: 00h SOCK_STREAM, NON-BLOCKING (TCP) 01h SOCK_STREAM, BLOCKING (TCP) 02h SOCK_DGRAM, NON-BLOCKING (UDP) 03h SOCK_DGRAM, BLOCKING (UDP)
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 50 ms, i.e. 50...99 = 50 ms, 100...149 = 100 ms, 199 = 150 ms etc.

Command Details: Create

Category

Extended

Details

Command Code.:   03h

Valid for: Object Instance

Description

This command creates a socket.

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

- **Command Details**

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

CmdExt[1]	Value:Socket Type:
	    00h SOCK_STREAM, NON-BLOCKING (TCP)
	    01h SOCK_STREAM, BLOCKING (TCP)
	    02h SOCK_DGRAM, NON-BLOCKING (UDP)
	    03h SOCK_DGRAM, BLOCKING (UDP)

- **Response Details**

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

Command Details: Delete

Category

Extended

Details

Command Code.:   04h

Valid for: Object Instance

Description

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

- If the socket is of TCP-type and a connection is established, the connection is terminated with the RST-flag.
- To gracefully terminate a TCP-connection, it is recommended to use the 'Shutdown'-command (see "Command Details: Shutdown" on page 74) before deleting the socket, causing the connection to be closed with the FIN-flag instead.

- **Command Details**

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

- **Response Details**

  (no data)

Command Details: Bind

Category

Extended

Details

Command Code.:   10h

Valid for: Instance

Description

This command binds a socket to a local port.

- **Command Details**

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

- **Response Details**

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

Command Details: Shutdown

Category

Extended

Details

Command Code.:   11h

Valid for: Instance

Description

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

- **Command Details**

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

CmdExt[1]	Value:Mode:
	   00h Shutdown receive channel
	   01h Shutdown send channel
	   02h Shutdown both receive- and send channel

- **Response Details**

  (no data)

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

Application initiates shutdown:

1. Send shutdown with CmdExt[1] set to 01h. This will send FIN-flag to host shutting down the send channel, note that the receive channel will still be operational.

2. Receive data on socket until error message Object specific error (EDESTADDRREQ (14)) is received, indicating that the host closed the receive channel. If host does not close the receive channel use a timeout and progress to step 3.

3. Delete the socket instance. If step 2 timed out, RST-flag will be sent to terminate the socket.

A remote host initiates shutdown:

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

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

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

4. Delete the socket instance.

Command Details: Listen

Category

Extended

Details

Command Code.:   12h

Valid for: Instance

Description

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

Extended

Details

Command Code.:   13h

Valid for: Instance

Description

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

NON-BLOCKING mode:

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

BLOCKING mode:

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

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

- **Command Details**

  (no data)

- **Response Details**

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

Command Details: Connect

Category

Extended

Details

Command Code.:   14h

Valid for: Instance

Description

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

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

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

NON-BLOCKING mode:

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

BLOCKING mode:

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

- **Command Details**

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

- **Response Details**

(no data)

Command Details: Receive

Category

Extended

Details

Command Code.:   15h

Valid for: Instance

Description

This command receives data from a connected socket. Message segmentation may be used to receive up to 1472 bytes (see "Message Segmentation" on page 125).

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

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

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

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

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

- **Command Details**

Field	Contents	Notes
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control bits	see "Command Segmentation" on page 126
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 125).

Field	Contents	Notes
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control bits	see "Response Segmentation" on page 127
Data[0...n]	Received data	-

Command Details: Receive_From

Category

Extended

Details

Command Code.:   16h

Valid for: Instance

Description

This command receives data from an unconnected SOCK_DGRAM-socket. Message segmentation may be used to receive up to 1472 bytes (see "Message Segmentation" on page 125).

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

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

NON-BLOCKING mode:

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

BLOCKING mode:

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

- **Command Details**

Field	Contents	Notes
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control bits	see "Command Segmentation" on page 126
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 125).

Field	Contents	Notes
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control bits	see "Response Segmentation" on page 127
Data[0]	Host IP address byte 3 (low)	The host address/port information is only included
Data[1]	Host IP address byte 2	in the first segment. All data thereafter will start at
Data[2]	Host IP address byte 1	Data[0]
Data[3]	Host IP address byte 0 (high)	
Data[4]	Host port number (low)	
Data[5]	Host port number (high)	
Data[6...n]	Received data	

Command Details: Send

Category

Extended

Details

Command Code.:   17h

Valid for: Instance

Description

This command sends data on a connected socket. Message segmentation may be used to send up to 1472 bytes (see "Message Segmentation" on page 125).

NON-BLOCKING mode:

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

BLOCKING mode:

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

- **Command Details**

    Note: To allow larger amount of data (i.e. >255 bytes) to be sent, the command data may be segmented (see "Message Segmentation" on page 125).

Field	Contents	Notes
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control	see "Command Segmentation" on page 126
Data[0...n]	Data to send	-

- **Response Details**

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

Command Details: Send_To

Category

Extended

Details

Command Code.:   18h

Valid for: Instance

Description

This command sends data to a specified host on an unconnected SOCK_DGRAM-socket. Message seg-
mentation may be used to send up to 1472 bytes (see "Message Segmentation" on page 125).

- **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 125).

Field	Contents	Notes
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control	see "Command Segmentation" on page 126
Data[0]	Host IP address byte 3 (low)	The host address/port information shall only be
Data[1]	Host IP address byte 2	included in the first segment. All data thereafter
Data[2]	Host IP address byte 1	must start at Data[0]
Data[3]	Host IP address byte 0 (high)	
Data[4]	Host port number (low)	
Data[5]	Host port number (high)	
Data[6...n]	Data to send	

- **Response Details**

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

Command Details: IP_Add_Membership

Category

Extended

Details

Command Code.:   19h

Valid for: Instance

Description

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

- **Command Details**

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

- **Response Details**

  (no data)

Command Details: IP_Drop_Membership

Category

Extended

Details

Command Code.:   1Ah

Valid for: Instance

Description

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

- **Command Details**

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

- **Response Details**

(no data)

Command Details: DNS_Lookup

Category

Extended

Details

Command Code.:   1Bh

Valid for: Object Instance

Description

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

- **Command Details**

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

- **Response Details (Success)**

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

Socket Interface Error Codes (Object Specific)

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

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

9.7 SMTP Client Object (09h)

Category

Extended

Object Description

This object groups functions related to the SMTP-client.

See also...

- "File System" on page 15
- "E-mail Client" on page 28
- "Instance Attributes (Instance #13, SMTP Server)" on page 65
- "Instance Attributes (Instance #14, SMTP User)" on page 66
- "Instance Attributes (Instance #15, SMTP Password)" on page 66

Supported Commands

Object: Get_Attribute
                 Create
                 Delete
                 Send email from file("Command Details: Send Email From File" on page 89)

Instance: Get_Attribute
                 Set_Attribute
                 Send email("Command Details: Send Email" on page 90)

Object Attributes (Instance #0)

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

Instance Attributes

Extended

Instances are created dynamically by the application.

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

Command Details: Create

Category

Extended

Details

Command Code.:   03h

Valid for: Object

Description

This command creates an 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

Extended

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

Extended

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]
e-mail subject

[Headers]
extra headers, optional

[Message]
actual e-mail message
```

- **Command Details**

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

- **Response Details**

(no data)

Command Details: Send Email

Category

Extended

Details

Command Code.:   10h

Valid for: Instance

Description

This command sends the specified e-mail instance.

- **Command Details**

  (no data)

- **Response Details**

  (no data)

Object Specific Error Codes

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

9.8 Anybus File System Interface Object (0Ah)

Category

Extended

Object Description

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

Instances are created and deleted dynamically during runtime.

The object is structurally identical to the "Application File System Interface Object (EAh)" on page 110.

Supported Commands

Object:	Get_Attribute
	Create("Command Details: Create" on page 93)
	Delete("Command Details: Delete" on page 94)
	Format Disc("Command Details: Format Disc" on page 103)
Instance:	Get_Attribute
	File Open("Command Details: File Open" on page 94)
	File Close("Command Details: File Close" on page 95)
	File Delete("Command Details: File Delete" on page 95)
	File Copy("Command Details: File Copy" on page 96)
	File Rename("Command Details: File Rename" on page 97)
	File Read("Command Details: File Read" on page 98)
	File Write("Command Details: File Write" on page 99)
	Directory Open("Command Details: Directory Open" on page 99)
	Directory Close("Command Details: Directory Close" on page 100)
	Directory Delete("Command Details: Directory Delete" on page 100)
	Directory Read("Command Details: Directory Read" on page 101)
	Directory Create("Command Details: Directory Create" on page 102)
	Directory Change("Command Details: Directory Change" on page 102)

Object Attributes (Instance #0)

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

Instance Attributes

Extended

#	Name	Access	Type	Description
1	Instance type	Get	UINT8	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

Extended

Details

Command Code.:   03h

Valid for: Object

Description

This command creates a file operation instance.

- **Command Details**

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

- **Response Details**

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

Command Details: Delete

Category

Extended

Details

Command Code.:   04h

Valid for: Object

Description

This command deletes a file operation instance.

- **Command Details**

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

- **Response Details**

  (no data)

Command Details: File Open

Category

Extended

Details

Command Code.:   10h

Valid for: Instance

Description

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

- **Command Details**

Field	Contents	Comments

CmdExt[0]	Mode	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

Extended

Details

Command Code.:   11h

Valid for: Instance

Description

This command closes a previously opened file.

- **Command Details**

  (no data)

- **Response Details**

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

Command Details: File Delete

Category

Extended

Details

Command Code.:   12h

Valid for: Instance

Description

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

- **Command Details**

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

Extended

Details

Command Code.:   13h

Valid for: Instance

Description

This command makes a copy of a file.

- **Command Details**

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

- **Response Details**

  (no data)

Command Details: File Rename

Category

Extended

Details

Command Code.:   14h

Valid for: Instance

Description

This command renames or moves a file.

- **Command Details**

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

- **Response Details**

   (no data)

Command Details: File Read

Category

Extended

Details

Command Code.: 15h

Valid for: Instance

Description

Reads data from a file previously opened for reading.

- **Command Details**

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

- **Response Details**

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

Command Details: File Write

Category

Extended

Details

Command Code.: 16h

Valid for: Instance

Description

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

- **Command Details**

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

- **Response Details**

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

Command Details: Directory Open

Category

Extended

Details

Command Code.: 20h

Valid for: Instance

Description

This command opens a directory.

- **Command Details**

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

- **Response Details**

  (no data)

Command Details: Directory Close

Category

Extended

Details

Command Code.:   21h

Valid for: Instance

Description

This command closes a previously opened directory.

- **Command Details**

  (no data)

- **Response Details**

  (no data)

Command Details: Directory Delete

Category

Extended

Details

Command Code.:   22h

Valid for: Instance

Description

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

- **Command Details**

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

- **Response Details**

  (no data)

Command Details: Directory Read

Category

Extended

Details

Command Code.:   23h

Valid for: Instance

Description

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

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

- **Command Details**

  (no data)

- **Response Details**

Field	Contents	Comments
CmdExt[0]	(reserved, ignore)	-
CmdExt[1]		
MsgData[0]	Size of entry	Low byte, low word
MsgData[1]		-
MsgData[2]		-
MsgData[3]		High byte, high word
MsgData[4]	Flags	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

Extended

Details

Command Code.:   24h

Valid for: Instance

Description

This command creates a directory.

- **Command Details**

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

- **Response Details**

   (no data)

Command Details: Directory Change

Category

Extended

Details

Command Code.:   25h

Valid for: Instance

Description

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

- **Command Details**

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

- **Response Details**

   (no data)

Command Details: Format Disc

Category

Extended

Details

Command Code.:   30h

Valid for: Object

Description

This command formats the file system.

- **Command Details**

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

- **Response Details**

  (no data)

Object Specific Error Codes

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

9.9 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 106

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 implementation.

Standard Objects:

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

Network Specific Objects:

- "Ethernet Host Object (F9h)" on page 106
- "Application File System Interface Object (EAh)" on page 110

10.2 Ethernet Host Object (F9h)

Category

Basic, extended

Object Description

This object implements Ethernet features in the host application.

Supported Commands

Object:	Get_Attribute
Instance:	Get_Attribute
	Set_Attribute

Object Attributes (Instance #0)

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

Instance Attributes (Instance #1)

Basic

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

a. If an attribute is not implemented, the module will use this value instead
b. The module is pre-programmed with a valid Mac address. To use that address, do *not* implement this attribute.

Extended

#	Name	Access	Type	Default[a]	Comment
2	Enable HICP	Get	BOOL	True	Value:Meaning: True HICP enabled False HICP disabled (see "Secure HICP (Secure Host IP Configuration Protocol)" on page 128)
3	Enable Web Server	Get	BOOL	True	Value:Meaning: True web server enabled False web server disabled (see "Web Server" on page 21)
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 21)
6	Enable FTP server	Get	BOOL	True	Value:Meaning: True FTP server enabled False FTP server disabled (see "FTP Server" on page 19)
7	Enable admin mode	Get	BOOL	False	Value:Meaning: True FTP Admin mode enabled False FTP Admin mode disabled (see "FTP Server" on page 19)
8	Network Status	Set	UINT16	-	See "Network Status" on page 109
9	Port 1 MAC address	Get	Array of UINT8	-	MAC address for Ethernet port 1, 6 bytes
10	Port 2 MAC address	Get	Array of UINT8	-	MAC address for Ethernet port 2, 6 bytes
11	Enable ACD	Get	BOOL	True	Value:Meaning: True ACD enabled False ACD disabled
12	Port 1 State	Get	ENUM	Enable	State of Ethernet port 1, see "Port State" on page 109
13	Port 2 State	Get	ENUM	Enable	State of Ethernet port 2, see "Port State" on page 109
14	Reserved				
15	Enable reset from HICP	Get	BOOL	False	Value:Meaning: True Possible to reset the module from HICP False Not possible to reset the module from HICP

#	Name	Access	Type	Default[a]	Comment
16	IP configuration	Set	Struct of: UINT32 (IP address) UINT32 (Subnet mask) UINIT32 (Gateway)	N/A	The Anybus CompactCom writes the IP configuration (IP address, Subnet mask, Gateway) to this attribute whenever the configuration is assigned or changed.
17	IP address byte 0 - 2	Get	Array of UINT8[3]	[0] : 192 [1] : 168 [2] : 0	This attributes holds the first three bytes of the IP address. The attribute is used in Shift Register Mode if the configuration switch value is set to 1 - 245. The first three bytes of the IP address will be given by the values in the attribute and the last byte will be given by the configuration switch value.

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

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 detected
		  False No link

1	IP in use	Value:Meaning:
		  True IP address in use (no address conflict detected)
		  False No IP address in use

2	IP conflict	Value:Meaning:
		  True IP address conflict detected
		  False No IP address conflict detected

3	Link port 1	Value:Meaning:
		  True Valid link on port 1
		  False No valid link on port 1

4	Link port 2	Value:Meaning:
		  True Valid link on port 2
		  False No valid link on port 2

5... 15	(reserved)	(mask off and ignore)

Port State

The attributes Port 1 State and Port 2 State tells wether the corresponding port is enabled or not.

Value	State	Description
00h	Enable	The Ethernet port is enabled.
01h	Disable	The Ethernet port is disabled.

10.3 Application File System Interface Object (EAh)

Category

Extended

Object Description

This object provides an interface to the built-in file system. Each instance represents a handle to a file stream and contains services for file system operations. This allows the user to download software through the FTP server to the application. The application decides the available memory space.

Instances are created and deleted dynamically during runtime.

The object is structurally identical to the "Anybus File System Interface Object (0Ah)" on page 91.

Supported Commands

Object:	Get_Attribute
	Create("Command Details: Create" on page 112)
	Delete("Command Details: Delete" on page 113)
Instance:	Get_Attribute
	File Open("Command Details: File Open" on page 113)
	File Close("Command Details: File Close" on page 114)
	File Delete("Command Details: File Delete" on page 114)
	File Copy("Command Details: File Copy" on page 115)
	File Rename("Command Details: File Rename" on page 116)
	File Read("Command Details: File Read" on page 117)
	File Write("Command Details: File Write" on page 118)
	Directory Open("Command Details: Directory Open" on page 118)
	Directory Close("Command Details: Directory Close" on page 119)
	Directory Delete("Command Details: Directory Delete" on page 119)
	Directory Read("Command Details: Directory Read" on page 120)
	Directory Create("Command Details: Directory Create" on page 121)
	Directory Change("Command Details: Directory Change" on page 121)

Object Attributes (Instance #0)

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

Instance Attributes

Extended

#	Name	Access	Type	Description
1	Instance type	Get	UINT8	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

Extended

Details

Command Code.:   03h

Valid for: Object

Description

This command creates a file operation instance.

- **Command Details**

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

- **Response Details**

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

Command Details: Delete

Category

Extended

Details

Command Code.:   04h

Valid for: Object

Description

This command deletes a file operation instance.

- **Command Details**

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

- **Response Details**

(no data)

Command Details: File Open

Category

Extended

Details

Command Code.:   10h

Valid for: Instance

Description

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

- **Command Details**

Field	Contents	Comments

CmdExt[0]	Mode	Value:Mode:
		  00h Read mode
		  01h Write mode
		  02h Append mode

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

- **Response Details**

(no data)

Command Details: File Close

Category

Extended

Details

Command Code.:   11h

Valid for: Instance

Description

This command closes a previously opened file.

- **Command Details**

  (no data)

- **Response Details**

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

Command Details: File Delete

Category

Extended

Details

Command Code.:   12h

Valid for: Instance

Description

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

- **Command Details**

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

- **Response Details**

  (no data)

Command Details: File Copy

Category

Extended

Details

Command Code.:   13h

Valid for: Instance

Description

This command makes a copy of a file.

- **Command Details**

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

- **Response Details**

(no data)

Command Details: File Rename

Category

Extended

Details

Command Code.:   14h

Valid for: Instance

Description

This command renames or moves a file.

- **Command Details**

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

- **Response Details**

  (no data)

Command Details: File Read

Category

Extended

Details

Command Code.:   15h

Valid for: Instance

Description

Reads data from a file previously opened for reading.

- **Command Details**

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

- **Response Details**

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

Command Details: File Write

Category

Extended

Details

Command Code.:   16h

Valid for: Instance

Description

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

- **Command Details**

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

- **Response Details**

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

Command Details: Directory Open

Category

Extended

Details

Command Code.:   20h

Valid for: Instance

Description

This command opens a directory.

- **Command Details**

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

- **Response Details**

  (no data)

Command Details: Directory Close

Category

Extended

Details

Command Code.:   21h

Valid for: Instance

Description

This command closes a previously opened directory.

- **Command Details**

  (no data)

- **Response Details**

  (no data)

Command Details: Directory Delete

Category

Extended

Details

Command Code.:   22h

Valid for: Instance

Description

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

- **Command Details**

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

- **Response Details**

  (no data)

Command Details: Directory Read

Category

Extended

Details

Command Code.:   23h

Valid for: Instance

Description

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

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

- **Command Details**

  (no data)

- **Response Details**

Field	Contents	Comments
CmdExt[0]	(reserved, set to zero)	-
CmdExt[1]		
MsgData[0]	Size of entry	Low byte, low word
MsgData[1]		-
MsgData[2]		-
MsgData[3]		High byte, high word

MsgData[4]	Flags	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

Extended

Details

Command Code.:   24h

Valid for: Instance

Description

This command creates a directory.

- **Command Details**

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

- **Response Details**

  (no data)

Command Details: Directory Change

Category

Extended

Details

Command Code.:   25h

Valid for: Instance

Description

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

- **Command Details**

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

- **Response Details**

  (no data)

Object Specific Error Codes

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

A. Categorization of Functionality

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

A.1 Basic

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

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

A.2 Extended

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

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

B. Implementation Details

B.1 SUP-Bit Definition

Not used.

B.2 Anybus Statemachine

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

Anybus State	Implementation	Comment
WAIT_PROCESS	The module stays in this state as long as it is running. From here it can only change to ERROR or EXCEPTION.	-
ERROR	Duplicate IP address detected	-
PROCESS_ACTIVE	-	The module shall never enter any of these states.
IDLE	-	
EXCEPTION	Unexpected error, e.g. watchdog timeout etc.	MS LED turns red (to indicate a major fault) NS LED is off

B.3 Application Watchdog Timeout Handling

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

C. Message Segmentation

C.1 General

Category: Extended

The maximum message size supported by the Anybus CompactCom 40 is 1524 bytes. If the host application implements a data message size of 1524 bytes, a message will always fit into one segment. The host application can implement a shorter message size (256 bytes for backwards compatibility with the 30-series).

No service requires messages larger than what is supported by the Anybus CompactCom 40 series 1524 bytes messaging interface. If this interface is used by the application, it allows very basic segmentation handling. The first segment bit (FS) and the last segment bit (LS) shall always be set in each segmented command or response. Some commands in the Socket Interface Object (page 69) use segmentation.

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

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

The module supports 20 simultaneous segmented messages.

C.2 Command Segmentation

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

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

Command segmentation is used for the following commands:

- Send (see "Command Details: Send" on page 80)
- Send To (see "Command Details: Send_To" on page 81)

Segmentation Control bits (Command)

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

Segmentation Control bits (Response)

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

When issuing a segmented command, the following rules apply:

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

C.3 Response Segmentation

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

Response segmentation is used for responses to the following commands:

- Receive (object specific, see "Command Details: Receive" on page 78)
- Receive From (object specific, see "Command Details: Receive_From" on page 79)

Segmentation Control bits (Command)

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

Segmentation Control bits (Response)

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

When receiving a segmented response, the following rules apply:

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

D. Secure HICP (Secure Host IP Configuration Protocol)

D.1 General

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

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

D.2 Operation

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

To alter the network settings of a module, double-click on its entry in the list. A window will appear, containing the settings for the module.
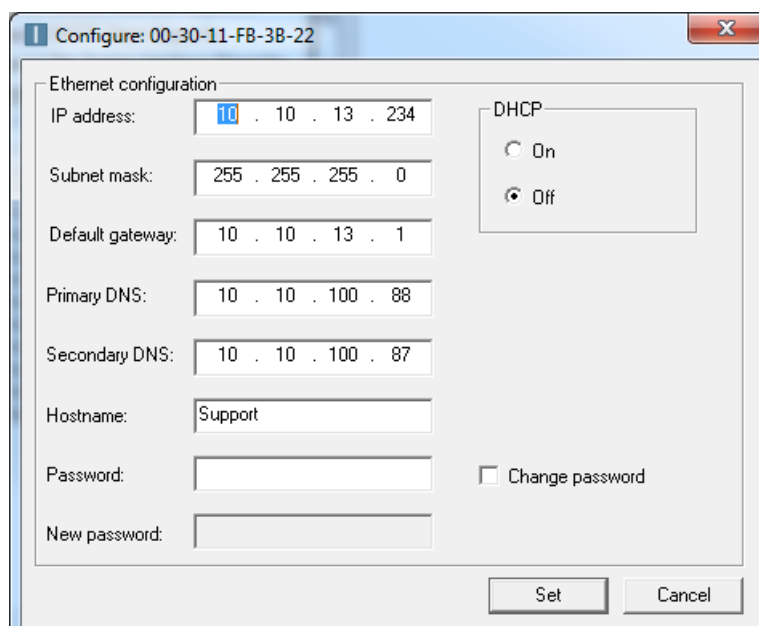
Validate the new settings by clicking "Set", or click "Cancel" to cancel all changes. Optionally, the configuration can be protected from unauthorized access by a password. To enter a password, check the "Change password" check box and enter the password in the "New password" text field.
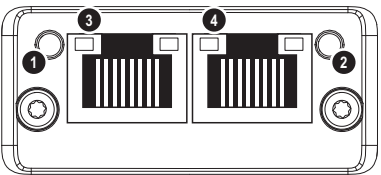
E. Technical Specification

E.1 Front View

Please note that the LED indications may change, once specific network communication software has been downloaded to the module. When the network communication software has been downloaded, the Network Status LED (labelled L1) and the Module Status LED (labelled L2) must in many cases be relabelled to conform to specific network certification requirements.

Relabelling can be done by the customer or with interchangeable network specific fronts provided by HMS. Contact HMS for more information.

#	Item
1	L1 (Network Status LED[a])
2	L2 (Module Status LED[a])
3	Link/Activity LED (port 1)
4	Link/Activity LED (port 2)

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

E.1.1 Network Status LED

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

LED State	Description
Off	No power
Green	IP address assigned
Green, flashing	No IP address has been assigned
Red	IP address conflict detected, ERROR

E.1.2 Module Status LED

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

LED State	Description
Off	Not in EXCEPTION or WAIT_PROCESS state
Green	In WAIT_PROCESS state
Red	Major fault (EXCEPTION state)

E.1.3 LINK/Activity LED 3/4

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

Fatal Error

If both the Network Status LED and the Module Status LED are red, a fatal error has occurred.

Ethernet Interface

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

E.2 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 40 Hardware Design Guide.

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

E.3 Power Supply

Supply Voltage

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

Power Consumption

The Anybus CompactCom 40 Common Ethernet is designed to fulfil the requirements of a Class B module. For more information about the power consumption classification used on the Anybus CompactCom 40 platform, consult the general Anybus CompactCom 40 Hardware Design Guide.

E.4 Environmental Specification

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

E.5 EMC Compliance

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

F. Copyright Notices

Copyright 2013 jQuery Foundation and other contributors

http://jquery.com/

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE
LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

****

rsvp.js

Copyright (c) 2013 Yehuda Katz, Tom Dale, and contributors

Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies
of the Software, and to permit persons to whom the Software is furnished to do
so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
****

lwIP is licenced under the BSD licence:

Copyright (c) 2001-2004 Swedish Institute of Computer Science.
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice,
this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice,
this list of conditions and the following disclaimer in the documentation
and/or other materials provided with the distribution.
3. The name of the author may not be used to endorse or promote products
derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS AND ANY EXPRESS OR IMPLIED
WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT
SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT
OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY
OF SUCH DAMAGE.

****

MD5 routines

Copyright (C) 1999, 2000, 2002 Aladdin Enterprises. All rights reserved.

This software is provided 'as-is', without any express or implied
warranty. In no event will the authors be held liable for any damages
arising from the use of this software.Permission is granted to anyone to use this software for any purpose,
including commercial applications, and to alter it and redistribute it
freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not
   claim that you wrote the original software. If you use this software
   in a product, an acknowledgment in the product documentation would be
   appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be
   misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.

L. Peter Deutsch
ghost@aladdin.com