**Network Interface Appendix**

# Anybus®-CompactCom Modbus/TCP w. IT-Functionality

**Doc.Id. SCM-1200-054**
**Rev. 3.06**

**HMS** Connecting Devices™

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

# Important User Information

This document is intended to provide a good understanding of the functionality offered by Modbus/TCP w. IT. The document only describes the features that are specific to the Anybus CompactCom Modbus/TCP w. IT-Functionality. For general information regarding the Anybus CompactCom, consult the Anybus CompactCom design guides.

The reader of this document is expected to be familiar with high level software design, and communication systems in general. The use of advanced Modbus/TCP w. IT-specific functionality may require in-depth knowledge in Modbus/TCP w. IT networking internals and/or information from the official Modbus/TCP w. IT specifications. In such cases, the people responsible for the implementation of this product should either obtain the Modbus/TCP w. IT 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

**Chapter 6    E-mail Client**

**Chapter 7    Server Side Include (SSI)**

**Chapter 8    Modbus/TCP Register Implementation**

**Chapter 9    Modbus/TCP Functions**

## Appendix F Timing & Performance

## Appendix G 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 Software Design Guide | HMS |
| Anybus-CompactCom Hardware Design Guide | HMS |
| Anybus-CompactCom Software Driver User Guide | HMS |
| Modbus Application Protocol Specification (v1.1a) | www.modbus.org |
| Modbus Messaging on TCP/IP Implementation Guide (v1.0a) | www.modbus.org |
| | |
| | |
| | |
| | |

## P.2 Document History

### Summary of Recent Changes (3.05 ... 3.06)

| Change | Page(s) |
|---|---|
| Corrected attribute 9 and added attribute 11 to Modbus Host Object | 96 |
| Added attribute 11 (Enable ACD) to Ethernet Host Object | 99 |
| Added note on Address Conflict Detection | 9 |
| | |
| | |
| | |

### Revision List

| Revision | Date | Author(s) | Chapter(s) | Description |
|---|---|---|---|---|
| 2.00 | 2006-10-12 | PeP | All | 1st official release |
| 2.01 | 2006-12-15 | PeP | All | Minor update |
| 2.02 | 2007-03-07 | PeP | 4, 7, 8, 10, 10 | Misc. updates & corrections |
| 2.03 | 2007-07-12 | PeP | - | Minor update |
| 2.04 | 2008-02-08 | PeP | - | Minor update |
| 2.05 | 2008-10-28 | HeS | 1, 2, 4, 10 | Misc. updates & corrections |
| 3.00 | 2010-04-14 | KeL | All | Change of Concept, minor update |
| 3.01 | 2010-12-03 | KeL | 10 | Misc. minor corrections |
| 3.02 | 2011-02-09 | KeL | 10, P, 3 | Minor corrections |
| 3.03 | 2011-04-08 | KaD | 11 | Minor correction |
| 3.04 | 2011-08-08 | KaD | 5 | Minor addition and updates |
| 3.05 | 2011-08-26 | KaD | 3 | Minor updates |
| 3.06 | 2012-02-29 | KeL | 3, 11 | Minor update and additions |

# P.3 Conventions & Terminology

The following conventions are used throughout this manual:

- Numbered lists provide sequential steps
- Bulleted lists provide information, not procedural steps
- The terms 'Anybus' or 'module' refers to the Anybus-CompactCom module.
- The terms 'host' or 'host application' refers to the device that hosts the Anybus module.
- Hexadecimal values are written in the format NNNNh or 0xNNNN, where NNNN is the hexadecimal value.
- A byte always consists of 8 bits.

# P.4 Sales and Support

| Sales | | Support | |
|---|---|---|---|
| **HMS Sweden (Head Office)** | | | |
| E-mail: | sales@hms-networks.com | E-mail: | support@hms-networks.com |
| Phone: | +46 (0) 35 - 17 29 56 | Phone: | +46 (0) 35 - 17 29 20 |
| Fax: | +46 (0) 35 - 17 29 09 | Fax: | +46 (0) 35 - 17 29 09 |
| Online: | www.anybus.com | Online: | www.anybus.com |
| **HMS North America** | | | |
| E-mail: | us-sales@hms-networks.com | E-mail: | us-support@hms-networks.com |
| Phone: | +1-312 - 829 - 0601 | Phone: | +1-312-829-0601 |
| Toll Free: | +1-888-8-Anybus | Toll Free: | +1-888-8-Anybus |
| Fax: | +1-312-629-2869 | Fax: | +1-312-629-2869 |
| Online: | www.anybus.com | Online: | www.anybus.com |
| **HMS Germany** | | | |
| E-mail: | ge-sales@hms-networks.com | E-mail: | ge-support@hms-networks.com |
| Phone: | +49 (0) 721-989777-000 | Phone: | +49 (0) 721-989777-000 |
| Fax: | +49 (0) 721-989777-010 | Fax: | +49 (0) 721-989777-010 |
| Online: | www.anybus.de | Online: | www.anybus.de |
| **HMS Japan** | | | |
| E-mail: | jp-sales@hms-networks.com | E-mail: | jp-support@hms-networks.com |
| Phone: | +81 (0) 45-478-5340 | Phone: | +81 (0) 45-478-5340 |
| Fax: | +81 (0) 45-476-0315 | Fax: | +81 (0) 45-476-0315 |
| Online: | www.anybus.jp | Online: | www.anybus.jp |
| **HMS China** | | | |
| E-mail: | cn-sales@hms-networks.com | E-mail: | cn-support@hms-networks.com |
| Phone: | +86 (0) 10-8532-3183 | Phone: | +86 (0) 10-8532-3023 |
| Fax: | +86 (0) 10-8532-3209 | Fax: | +86 (0) 10-8532-3209 |
| Online: | www.anybus.cn | Online: | www.anybus.cn |
| **HMS Italy** | | | |
| E-mail: | it-sales@hms-networks.com | E-mail: | it-support@hms-networks.com |
| Phone: | +39 039 59662 27 | Phone: | +39 039 59662 27 |
| Fax: | +39 039 59662 31 | Fax: | +39 039 59662 31 |
| Online: | www.anybus.it | Online: | www.anybus.it |
| **HMS France** | | | |
| E-mail: | fr-sales@hms-networks.com | E-mail: | fr-support@hms-networks.com |
| Phone: | +33 (0) 3 68 368 034 | Phone: | +33 (0) 3 68 368 033 |
| Fax: | +33 (0) 3 68 368 031 | Fax: | +33 (0) 3 68 368 031 |
| Online: | www.anybus.fr | Online: | www.anybus.fr |
| **HMS UK & Eire** | | | |
| E-mail: | uk-sales@hms-networks.com | E-mail: | support@hms-networks.com |
| Phone: | +44 (0) 1926 405599 | Phone: | +46 (0) 35 - 17 29 20 |
| Fax: | +44 (0) 1926 405522 | Fax: | +46 (0) 35 - 17 29 09 |
| Online: | www.anybus.co.uk | Online: | www.anybus.com |
| **HMS Denmark** | | | |
| E-mail: | dk-sales@hms-networks.com | E-mail: | support@hms-networks.com |
| Phone: | +45 (0) 35 38 29 00 | Phone: | +46 (0) 35 - 17 29 20 |
| Fax: | +46 (0) 35 17 29 09 | Fax: | +46 (0) 35 - 17 29 09 |
| Online: | www.anybus.com | Online: | www.anybus.com |
| **HMS India** | | | |
| E-mail: | in-sales@hms-networks.com | E-mail: | in-support@hms-networks.com |
| Phone: | +91 (0) 20 40111201 | Phone: | +91 (0) 20 40111201 |
| Fax: | +91 (0) 20 40111105 | Fax: | +91 (0) 20 40111105 |
| Online: | www.anybus.com | Online: | www.anybus.com |

# 1. About the Anybus-CompactCom Modbus/TCP

## 1.1 General
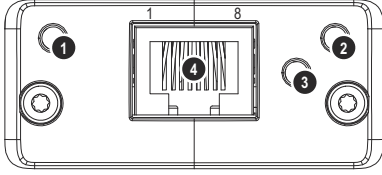
The Anybus-CompactCom Modbus/TCP communication module provides instant Ethernet and Modbus/TCP connectivity via the patented Anybus-CompactCom host interface. Any device that supports this standard can take advantage of the features offered by the module, allowing seamless network integration regardless of network type.

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

## 1.2 Features

- Galvanically isolated bus electronics
- 10/100Mbit, full/half duplex operation
- Web server w. customizable content
- FTP server
- E-mail client
- Server Side Include (SSI) functionality
- Modbus/TCP (up to 4 simultaneous connections)
- Modbus message forwarding
- Customizable Identity Information
- Transparent Socket Interface

# 1.3 Front View

| # | Item |
|---|------|
| 1 | Network Status LED |
| 2 | Module Status LED |
| 3 | Link/Activity |
| 4 | Ethernet Interface |



### Network Status LED

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

| LED State | Description |
|-----------|-------------|
| Off | No power or no IP address |
| Green | Module is in Process Active or Idle state |
| Green, flashing | Waiting for connections |
| Red | Duplicate IP address, or FATAL event |
| Red, flashing | Process Active Timeout. |

### Module Status LED

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

| LED State | Description |
|-----------|-------------|
| Off | No power |
| Green | Normal operation |
| Red | Major fault; module is in state EXCEPTION (or FATAL event) |
| Red, flashing | Minor fault in diagnostic object<br>IP conflict |

### LINK/Activity LED

| LED State | Description |
|-----------|-------------|
| Off | No link, no activity |
| Green | Link established |
| Green, flickering | Activity |

### Ethernet Interface

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

# 2. Tutorial

## 2.1 Introduction

This chapter is a complement to the Anybus CompactCom Implementation Tutorial. The ABCC tutorial describes and explains a simple example of an implementation with Anybus CompactCom.

## 2.2 Fieldbus Conformance Notes

- HMS do not pre-certify this product.
  For further information, please contact HMS.

# 3. Basic Operation

## 3.1 General Information

### 3.1.1 Software Requirements

Generally, no additional network support code needs to be written to support the Anybus-Compact-Com Modbus/TCP, however due to the nature of the Modbus/TCP networking system certain restrictions must be taken into account:

- The total number of ADIs that can be represented on the network depends on their size. By default, ADIs with instance numbers 1...4063 can be accessed from the network, each with a size of up to 32 bytes.

- ADI names, types and similar attributes cannot be accessed via Modbus/TCP. They are however represented on the network through the built in web server.

- A network write access of an ADI mapped to process data will result in a corresponding write access of the process data buffer of the Anybus CompactCom Modbus/TCP w. IT-Functionality. Such access will therefore not result in a Set_Attribute command towards the application.

- A network read access of an ADI, even if it is mapped to process data, will result in a corresponding Get_Attribute command towards the application.

- Network reset requests are not supported.

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

- Modbus/TCP in itself does not impose any particular timing demands when it comes to acyclic requests (i.e. requests towards instances in the Application Data Object), however it is generally recommended to process and respond to such requests within a reasonable time period (exactly what this means in practice depends on the implementation and the actual installation).

- The use of advanced Modbus/TCP-specific functionality may require in-depth knowledge in Modbus/TCP networking internals and/or information from the official Modbus/TCP specification. In such cases, the people responsible for the implementation of this product is expected either to obtain these specifications to gain sufficient knowledge or limit their implementation is such a way that this is not necessary.

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

See also...

- "Application Data (ADIs)" on page 11
- "Diagnostic Object (02h)" on page 51 (Anybus Module Object)
- Anybus-CompactCom Software Design Guide, 'Application Data Object (FEh)'

# 3.2 Device Customization

## 3.2.1 Web Interface

The web interface can be fully customized to suit a particular application. Data and web pages are stored in a FLASH-based file system, which can be accessed using any standard FTP-client.

See also...

- "File System" on page 13
- "FTP Server" on page 14
- "Web Server" on page 16

## 3.2.2 Modbus/TCP Implementation

By default, a 'Read Device Identification'-request returns the following information:

- Vendor Name:              "HMS"
- Product Code:             "ABCC-ETN"
- Major Minor Rev.:         (no information returned by default)
- Vendor URL:               (no information returned by default)
- Product Name:             (no information returned by default)
- Model Name:               (no information returned by default)
- User Application Name:  (no information returned by default)

It is possible to customize this information by implementing the Modbus Host Object.

See also...

- "Modbus Host Object (FAh)" on page 95 (Host Application Object)

## 3.2.3 Socket Interface (Advanced Users Only)

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

See also...

- "Socket Interface Object (07h)" on page 60 (Anybus Module Object)
- "Message Segmentation" on page 103

## 3.3 Communication Settings

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

In this case, this includes...

- **Ethernet Interface settings**

    By default, the module is set to auto negotiate the network interface settings, however it is possible to force the module to use a specific setting if necessary.

- **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.

- **Modbus/TCP Connection Timeout**

    This setting specifies how long a Modbus/TCP connection may be idle before it is closed by the module (default is 60 seconds).

- **Process Active Timeout**

    This value specifies how long the module shall stay in the 'PROCESS_ACTIVE'-state after receiving a Modbus/TCP request. Note that this affects the behaviour of the SUP-bit, see B-104 "Extended LED Functionality".

The parameters in the Network Configuration Object (04h) are available from the network as dedicated entries in the Modbus register map, through the built in web server, and via HICP.

**Note:** If an IP conflict occurs and Address Conflict Detection (ACD) is enabled, the IP address will be set to [ 0,0,0,0 ]. A new IP address can be set using e.g. HICP.

See also...

## 3.4 Diagnostics

Each instance within the Diagnostic Object (02h) is represented on the network as a dedicated entry in the Modbus register map (see "Input Registers (3x)" on page 42).

Note that since each entry corresponds *directly* to a specific diagnostic instance, it is possible to have "empty" diagnostic entries in the register map (when read, such entries will return zeroes).

See also...

# 3.5 Network Data Exchange

## 3.5.1 General

It is important to notice that various register areas might have different response times. Generally queries directed at the process data registers will be answered more quickly than those directed at the ADI-related registers since the former are directly processed by the module itself whereas the latter are forwarded to the application, which must respond before the module can respond to the master. In the latter case this will have repercussions on the allowable timeout time for the master to use against these registers.

## 3.5.2 Translation of Data Types

On Modbus/TCP, all network data (ADIs and Process Data) will be represented as follows:

| Anybus Type | Conversion Details |
|---|---|
| BOOL | Single BOOLS and arrays of BOOLS are converted to bit fields and padded with zeroes to the closest 16bit multiple. This enables Booleans to be accessed using the Modbus discrete and coil functions. |
| SINT8 | Padded with zeroes to fill a 16bit register. Arrays of SINT8 are packed and padded with zeroes to the closest 16bit multiple. |
| SINT16 | Occupies one 16bit Modbus register |
| SINT32 | Occupies two 16bit Modbus registers |
| SINT64 | Occupies four 16bit Modbus registers |
| UINT8 | Padded with zeroes to fill a 16bit register. Arrays of UINT8 are packed and padded with zeroes to the closest 16bit multiple. |
| UINT16 | Occupies one 16bit Modbus register |
| UINT32 | Occupies two 16bit Modbus registers |
| UINT64 | Occupies four 16bit Modbus registers |
| CHAR | Padded with zeroes to fill a 16bit register. Arrays of CHAR are packed and padded with zeroes to the closest 16bit multiple. |
| ENUM | Padded with zeroes to fill a 16bit register. Arrays of 8bit ENUM are packed and padded with zeroes to the closest 16bit multiple. |
| FLOAT | Occupies two 16bit Modbus registers |

### 3.5.3 Application Data (ADIs)

As mentioned previously, the total number of ADIs that can be represented on the network depends on their size. By default, ADIs with instance numbers 1...4063 can be accessed from the network, each with a size of up to 32 bytes. It is possible to alter this ratio by changing the number of ADI indexing bits (attribute #9, Modbus Host Object (FAh)).

Example 1 (Default settings)

In this example, attribute #9 in the Modbus Host Object (FAh) is set to its default value (04h).

| Holding Register # | ADI No. |
|---|---|
| 0210h... 021Fh | 1 |
| 0220h... 022Fh | 2 |
| 0230h... 023Fh | 3 |
| 0240h... 024Fh | 4 |
| ... | ... |
| FFE0h... FFEFh | 4062 |
| FFF0h... FFFFh | 4063 |

Each ADI is represented using 16 Modbus registers, which means that in theory up to 32 bytes of an ADI can be accessed from the network. Note however that this number may be less due to the data conversion process, see "Translation of Data Types" on page 10.

Example 2 (Customized implementation)

In this example, attribute #9 in the Modbus Host Object (FAh) is set to 05h.

| Holding Register # | ADI No. |
|---|---|
| 0210h... 022Fh | 1 |
| 0230h... 024Fh | 2 |
| 0250h... 026Fh | 3 |
| 0270h... 028Fh | 4 |
| ... | ... |
| FFB0h... FFCFh | 2030 |
| FFD0h... FFEFh | 2031 |

Each ADI is represented using 32 Modbus registers, which means that in theory up to 64 bytes of an ADI can be accessed from the network. Note however that this number may be less due to the data conversion process, see "Translation of Data Types" on page 10.

See also...

### 3.5.4 Process Data

Modbus does not feature a dedicated cyclic data channel in the same sense as many other networks. In the Anybus-CompactCom implementation, Process Data can however still be accessed from the network via dedicated entries in the Modbus register map. Just as with regular ADIs, the Process Data is converted to a format suitable for Modbus.

Process Data can be accessed on a bit by bit basis (as Coils & Discrete Inputs) - *or* - as 16bit entities (Holding Registers & Input Registers).

For example, reading Discrete Inputs 0000h-000Fh will return the same data as reading Input Register 0000h or Holding Register 0100h.

**Note:** For natural reasons, writing to the Write Process Data register area has no effect, and reading unused register locations will return zeroes.

Example:

| Application Process Data | | Corresponding Modbus Register Layout | | | Comment |
|---|---|---|---|---|---|
| Byte no. | Type | Register no. | High Byte | Low Byte | |
| 0 | UINT8 | 1 | | UINT8 | Padded to a full 16bit register |
| 1 | UINT8 | 2 | | UINT8 | Padded to a full 16bit register |
| 2 | ENUM | 3 | | ENUM | Padded to a full 16bit register |
| 3 | SINT16 | 4 | SINT16 | | - |
| 4 | | | | | |
| 5 | BOOL | 5 | | b0 | Converted to a bit field and padded to a full 16bit register |
| 6 | UINT32 | 6 | UINT32 | | Two 16 bit registers |
| 7 | | | | | |
| 8 | | 7 | | | |
| 9 | | | | | |
| 10 | BOOL[3] | 8 | | b2 b1 b0 | Packed to a bit field and padded to a full 16bit register |
| 11 | | | | | |
| 12 | | | | | |
| 13 | SINT8[3] | 9 | SINT8[1] | SINT8[0] | Padded to two 16bit registers |
| 14 | | 10 | | SINT8[2] | |
| 15 | | | | | |
| 16...37 | BOOL[23] | 11 | b15...0 | | Packed to a bit field and padded to two 16bit registers |
| | | 12 | | b22...b16 | |

**Note:** The example above assumes that the Process Data is accessed as Holding Registers or Input Registers.

See also...

• "Translation of Data Types" on page 10.

# 3.6 File System

## 3.6.1 General Information

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

The file system uses the following conventions:

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

See also...

- "FTP Server" on page 14
- "Web Server" on page 16
- "E-mail Client" on page 23
- "Server Side Include (SSI)" on page 24
- "File System Interface Object (0Ah)" on page 82

---

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

*The following operations will erase one or more flash segments:*

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

## 3.6.2 System Files

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

Example:

```
[Key1]
value of Key1

[Key2]
value of Key2
```

# 4. FTP Server

## 4.1 General Information

**Category**: extended

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

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

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

The FTP server supports up to 8 concurrent connections.

## 4.2 User Accounts

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

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

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

File Format ('\ftp.cfg'):
```
User1:Password1:Homedir1
User2:Password2:Homedir2
\path\userlistA:HomedirA
\path\userlistB:HomedirB
```

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

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

**Notes:**

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

- If 'Admin Mode' has been enabled in the Ethernet Object, all username/password combinations will be accepted and the user will have unrestricted access to the file system (i.e. the home directory will be the system root).

- It is strongly recommended to have at least one user with root access ('\') permission. If not, 'Admin Mode' must be enabled each time a system file needs to be altered (including '\ftp.cfg').

# 4.3 Session Example

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

1. Open the Windows Explorer by right-clicking on the 'Start'-button and selecting 'Explorer'
2. In the address field, type FTP://<user>:<password>@<address>
   - Substitute <address> with the IP address of the Anybus module
   - Substitute <user> with the username
   - Substitute <password> with the password

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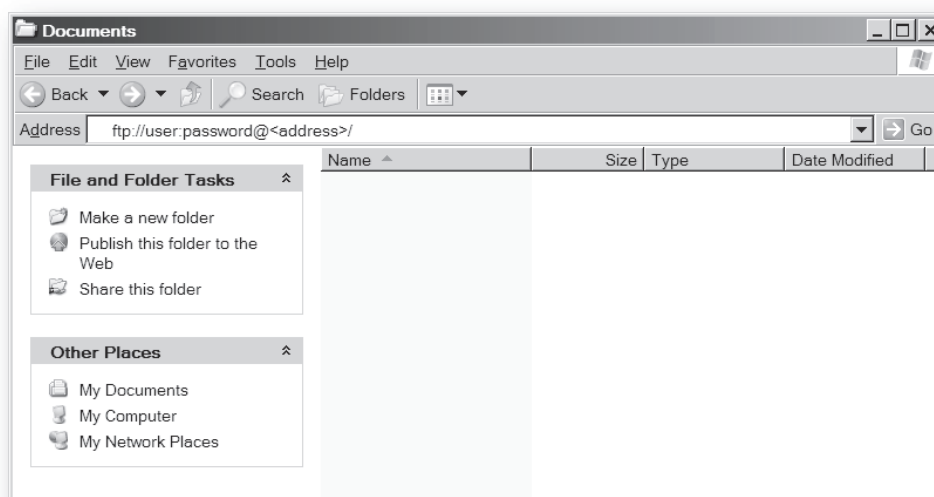
# 5. Web Server

## 5.1 General Information

**Category**: extended

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

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

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

See also...

- "FTP Server" on page 14
- "Server Side Include (SSI)" on page 23
- "Ethernet Host Object (F9h)" on page 121

## 5.2 Default Web Pages

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

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

The files are:

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



**Network Information**
- Serial no.
- MAC ID
- Firmware revision
- Module uptime
- CPU load

**Language Conf**
Set language
(If implemented)

**Network Configuration**
Gives access to the settings in the Network Configuration Object.

**Network Statistics**
Misc. ethernet related statistics; interface counters, media counters etc.
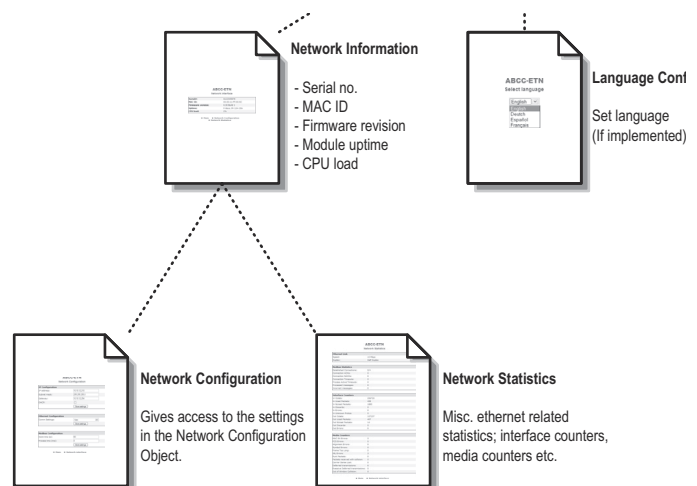
**Note:** If none of these files are used, it is recommended to completely disable the virtual file system altogether in the File System Interface Object.

See also...

- "File System" on page 13
- "File System Interface Object (0Ah)" on page 80

## 5.2.1 Network Configuration

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



Available editable settings will be explained on the next page.

### IP Configuration

| Name | Description |
|------|-------------|
| 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<br>The module needs a reset for any changes to take effect |
| Host name | IP address or name<br>Max 64 characters<br>Changes take effect immediately |
| Domain name | IP address or name<br>Max 48 characters<br>Changes take effect immediately |
| DNS 1 | Primary and secondary DNS server, used to resolve host name |
| DNS 2 | Default values: 0.0.0.0<br>Value ranges: 0.0.0.0 - 255.255.255.255<br>Changes take effect immediately |
| DHCP | Checkbox for enabling or disabling DHCP<br>Default value: enabled<br>The module needs a reset for any changes to take effect |

### SMTP Settings

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

| Name | Description |
|------|-------------|
| SMTP Server | IP address or name<br>Max 64 characters |
| SMTP User | Max 64 characters |
| SMTP Pswd | Max 64 characters |

### Ethernet Configuration

Changes will take effect immediately.

| Name | Description |
|------|-------------|
| Comm settings | Ethernet speed/duplex settings<br>Default value: auto |

### Modbus Configuration

| Name | Description |
|------|-------------|
| Conn tmo (s) | Using the value 0 will disable the timeout option<br>Default value: 60<br>Changes will take effect immediately |
| Process tmo (ms) | Default value: 0<br>Changes will take effect immediately |
| Word order | Modbus word order for types larger than 16 bits<br>Default value: Little-endian<br>The module needs a reset for any changes to take effect |

## 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. |

| 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. |

# 5.3 Server Configuration

## 5.3.1 General Information

**Category**: advanced

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

*File Format:*
```
[WebRoot]
\web

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

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

**Web Root Directory**

The web server cannot access files outside this directory.

**Content Types**

A list of file extensions and their reported content types.

See also...

- "Default Content Types" on page 21

**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 16

### 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 5-20 "General Information".

### 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.

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

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

# 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 68
- "SMTP Client Object (09h)" on page 77

## 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 username
- A valid password

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

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

Sending a message based on a file in the file system is achieved using the 'Send Email from File'-command. For a description of the file format, see "Command Details: Send Email From File" on page 80.

# 7. Server Side Include (SSI)

## 7.1 General Information

**Category**: advanced

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

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

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

## 7.2 Include File

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

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

Syntax:

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

filename-Source file

Default Output:

| Scenario | Default Output |
|----------|----------------|
| Success | (contents of file) |

# 7.3 Command Functions

## 7.3.1 General Information

Command functions executes commands and includes the result.

General Syntax:

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

Command Functions:

| Command | Valid for Email Messages | Page |
|---|---|---|
| GetConfigItem() | Yes | 26 |
| SetConfigItem() | No | 27 |
| SsiOutput() | Yes | 29 |
| DisplayRemoteUser | No | 29 |
| ChangeLanguage() | No | 30 |
| IncludeFile() | Yes | 31 |
| SaveDataToFile() | No | 32 |
| printf() | Yes | 33 |
| scanf() | No | 35 |

## 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 it's 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 username 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 24

### 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 e-mail messages.

Syntax:

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

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

Default Output:

| Scenario | Default Output |
|---|---|
| Success | "Configuration stored to "*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 34

| | |
|---|---|
| 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 37

Default Output:

| Scenario | Default Output |
|---|---|
| Success | (printf() result) |
| ABCCMessage error | ABCCMessage error string (7-40 "Errors") |

Example:

See also...

- - "ABCCMessage()" on page 37
- - "Example (Get_Attribute):" on page 39

### Formatting Tags

- **Type (Required)**

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

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

- **Flags (Optional)**

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

- **Width (Optional)**

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

- **.Precision (Optional)**

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

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

- **Modifier**

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

### 7.3.10 scanf()

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

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

**Note:** This command cannot be used in 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 36

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 37

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 40) |

Example:

See also...

  -  "ABCCMessage()" on page 37
  -  "Example (Set_Attribute):" on page 39

### 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() | - | 37 |

## 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 38

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

See also...

- "Response Data Types (r_type)" on page 38

Numeric input can be supplied in the following formats:

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

See also...

- "Example (Get_Attribute):" on page 39
- "Example (Set_Attribute):" on page 39

- **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" <br> **Note:** Quotes can be included in the string if preceded by backslash('\') <br> Example: *"We usually refer to it as \"the Egg\" "* |
| FLOAT | Yes | 5.6538e2 |
| NONE | No | Command holds no data, hence no data type |

- **Response Data Types (r_type)**

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

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

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

Example (Get_Attribute):

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

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

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

See also...

- 7-33 "printf()"

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 35

### Errors

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

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

See also...

- "SSI Output Configuration" on page 41

# 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 40

• 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 29

# 8. Modbus/TCP Register Implementation

## 8.1 Holding Registers (4x)

| Range | Contents | Notes |
|---|---|---|
| 0000h...00FFh | Read Process Data | See 3-12 "Process Data" |
| 0100h...01FFh | Write Process Data | |
| 0200h...0202h | (reserved) | - |
| 0203h | Process Active Timeout | See 10-51 "Network Configuration Object (04h)" (instance #9) |
| 0204h | Enter/Exit Idle Mode[a] | 0: Not Idle, >0: Idle |
| 0205h...020Fh | (reserved) | - |
| 0210h...FFFFh | ADI No. 1....nn | See 3-11 "Application Data (ADIs)" |

a. See B-104 "Anybus State Machine"

## 8.2 Input Registers (3x)

| Range | Contents | Notes |
|---|---|---|
| 0000h...00FFh | Write Process Data | See 3-12 "Process Data" |
| 0100h | Diagnostic Event Count | Number of pending diagnostic events |
| 0101h | Diagnostic Event #1 | These registers corresponds to instances in the Diagnostic Object (02h), see 11-95 "Modbus Host Object (FAh)". |
| 0102h | Diagnostic Event #2 | |
| 0103h | Diagnostic Event #3 | |
| 0104h | Diagnostic Event #4 | High byte = Severity |
| 0105h | Diagnostic Event #5 | Low byte = Event Code |
| 0106h | Diagnostic Event #6 | |
| 0107h...FFFFh | (reserved) | - |

## 8.3 Coils (0x)

| Range | Contents | Notes |
|---|---|---|
| 0000h...0FFFh | Read Process Data | See 3-12 "Process Data" |

## 8.4 Discrete Inputs (1x)

| Range | Contents | Notes |
|---|---|---|
| 0000h...0FFFh | Write Process Data | See 3-12 "Process Data" |

# 9. Modbus/TCP Functions

The following Modbus/TCP functions are implemented in the module:

| # | Function | Page |
|---|----------|------|
| 1 | Read Coils | 44 |
| 2 | Read Discrete Inputs | 44 |
| 3 | Read Holding Registers | 45 |
| 4 | Read Input Registers | 45 |
| 5 | Write Single Coil | 45 |
| 6 | Write Single Register | 46 |
| 15 | Write Multiple Coils | 46 |
| 16 | Write Multiple Registers | 47 |
| 23 | Read/Write Multiple Registers | 47 |
| 43 | Read Device Identification | 47 |

## Supported Exception Codes

| Code | Name | Description |
|------|------|-------------|
| 0x01 | Illegal function | The function code in the query is not supported |
| 0x02 | Illegal data address | The data address received in the query is outside the initialized memory area |
| 0x03 | Illegal data value | The data in the request is illegal |

**Note:** If Modbus message forwarding has been enabled in the Modbus Object (FAh), all Modbus messages will be forwarded to the host application.

See also...

- "Modbus Host Object (FAh)" on page 95
- "Command Details: Process Modbus Message" on page 97

# 9.1 Read Coils

Function Code:      1
Register Type:      0x (Coils)

### Details

This function is mapped to the Read Process data part of the Holding Registers as follows:

| Coil # | Holding Register # | Bit # |
|---|---|---|
| 0000h | 0000h | 0 |
| 0001h |  | 1 |
| 0002h |  | 2 |
| 0003h |  | 3 |
| ... |  | ... |
| 000Fh |  | 15 |
| 0010h | 0001h | 0 |
| 0011h |  | 1 |
| 0012h |  | 2 |
| 0013h |  | 3 |
| ... |  | ... |
| 001Fh |  | 15 |
| ... | ... | ... |
| 1FF0h | 01FFh | 0 |
| 1FF1h |  | 1 |
| 1FF2h |  | 2 |
| 1FF3h |  | 3 |
| ... |  | ... |
| 1FFFh |  | 15 |

# 9.2 Read Discrete Inputs

Function Code:      2
Register Type:      1x (Discrete Inputs)

### Details

This function is mapped to the Write Process data part of the Input Registers; the mapping is otherwise identical to that of the 'read coils' function described above.

# 9.3 Read Holding Registers

Function Code:          3
Register Type:          4x (Holding Registers)

### Details

Mapped to Read- and Write Process Data, ADIs, and configuration registers. It is allowed to read parts of a larger ABCC data type; it is also allowed to read multiple ADIs using a single request.

# 9.4 Read Input Registers

Function Code:          4
Register Type:          3x (Input Registers)

### Details

Mapped to Write Process Data and diagnostic registers.

# 9.5 Write Single Coil

Function Code:          5
Register Type:          0x (Coils)

### Details

This function is mapped to the Read Process data part of the Holding Registers as follows:

| Coil # | Holding Register # | Bit # |
|---|---|---|
| 0000h | 0000h | 0 |
| 0001h | | 1 |
| 0002h | | 2 |
| 0003h | | 3 |
| ... | | ... |
| 000Fh | | 15 |
| 0010h | 0001h | 0 |
| 0011h | | 1 |
| 0012h | | 2 |
| 0013h | | 3 |
| ... | | ... |
| 001Fh | | 15 |
| ... | ... | ... |
| 1FF0h | 01FFh | 0 |
| 1FF1h | | 1 |
| 1FF2h | | 2 |
| 1FF3h | | 3 |
| ... | | ... |
| 1FFFh | | 15 |

## 9.6 Write Single Register

Function Code:          6
Register Type:          4x (Discrete Inputs)

### Details

Mapped to Read- and Write Process Data, ADIs and configuration registers. ADIs must be written as a whole, however the Process Data area accepts writes of any size.

## 9.7 Write Multiple Coils

Function Code:          15
Register Type:          0x (Coils)

### Details

This function is mapped to the Read Process data part of the Holding Registers; the mapping is identical to that of the 'read coils' function described above.

## 9.8 Write Multiple Registers

Function Code:     16
Register Type:     4x (Holding Registers)

### Details

Mapped to Read- and Write Process Data, ADIs and configuration registers.

**Note:** ADIs must be written as a whole, but the Process Data area accepts writes of any size.

## 9.9 Read/Write Multiple Registers

Function Code:     23
Register Type:     4x (Holding Registers)

### Details

Mapped to read and write process data, ADIs and configuration registers.

**Note 1:** ADIs must be written as a whole, but the process data area accepts writes of any size.

**Note 2:** It is allowed to read parts of larger data types, and to read multiple ADIs using a single request.

## 9.10 Read Device Identification

Function Code:     43 (Subcode 14)
Register Type:     -

### Details

Basic and regular device identification objects are supported according to the Modbus specification. Extended device identification objects are not supported.

Identification strings are extracted from the host application via the Modbus Object (FAh). If this object is not implemented, the default identification strings will be returned.

# 10. Anybus Module Objects

## 10.1 General Information

This chapter specifies the Anybus Module Object implementation in the module.

Standard Objects:

Network Specific Objects:

# 10.2 Anybus Object (01h)

## Category

Basic

## Object Description

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

## Supported Commands

Object:          Get_Attribute

Instance:        Get_Attribute
                 Set_Attribute
                 Get_Enum_String

## Object Attributes (Instance #0)

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

## Instance Attributes (Instance #1)

### Basic

| # | Name | Access | Type | Value |
|---|------|--------|------|-------|
| 1 | Module type | Get | UINT16 | 0401h (Standard Anybus-CompactCom) |
| 2... 11 | - | - | - | Consult the general Anybus-CompactCom Software Design Guide for further information. |
| 12 | LED colors | Get | struct of:<br>　UINT8(LED1A)<br>　UINT8(LED1B)<br>　UINT8(LED2A)<br>　UINT8(LED2B) | Value:Color:<br>　01h  Green<br>　02h  Red<br>　01h  Green<br>　02h  Red |
| 13... 15 | - | - | - | Consult the general Anybus-CompactCom Software Design Guide for further information. |

## 10.3 Diagnostic Object (02h)

### Category

Extended

### Object Description

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

### Supported Commands

| | |
|---|---|
| Object: | Get_Attribute |
| | Create |
| | Delete |
| Instance: | Get_Attribute |

### Object Attributes (Instance #0)

| # | Name | Access | Data Type | Value |
|------|---------------------|--------|-----------|-------------------------------------------------------------------------------|
| 1... 4 | - | - | - | Consult the general Anybus-CompactCom Software Design Guide for further information. |
| 11 | Max no. of instances | Get | UINT16 | 5+1 |

### Instance Attributes

#### Extended

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

Each diagnostic instance is represented as a dedicated entry in the Modbus/TCP register map.

See also...

# 10.4 Network Object (03h)

## Category

Extended

## Object Description

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

## Supported Commands

Object:          Get_Attribute

Instance:        Get_Attribute
                 Set_Attribute
                 Get_Enum_String
                 Map_ADI_Write_Area
                 Map_ADI_Read_Area

## Object Attributes (Instance #0)

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

## Instance Attributes (Instance #1)

### Extended

| # | Name | Access | Type | Value |
|---|------|--------|------|-------|
| 1 | Network type | Get | UINT16 | 0080h |
| 2 | Network type string | Get | Array of CHAR | 'Modbus-TCP' |
| 3 | Data format | Get | ENUM | 00h (LSB first) |
| 4 | Parameter data support | Get | BOOL | True |
| 5 | Write process data size | Get | UINT16 | Current write process data size (in bytes)<br>Updated on every successful Map_ADI_Write_Area[a] |
| 6 | Read process data size | Get | UINT16 | Current read process data size (in bytes)<br>Updated on every successful Map_ADI_Read_Area[a] |
| 7 | Exception Information | Get | UINT8 | (No network specific exception information available) |
| 8... 10 | - | - | - | Consult the general Anybus-CompactCom Software Design Guide for further information. |

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

# 10.5 Network Configuration Object (04h)

## Category

Extended

## Object Description

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

See also...

- "Communication Settings" on page 9
- "E-mail Client" on page 22

## Supported Commands

Object:          Get_Attribute
                 Reset

Instance:        Get_Attribute
                 Set_Attribute
                 Get_Enum_String

## Object Attributes (Instance #0)

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

## Instance Attributes (Instance #3, IP Address)

### Extended

Value is used after module reset.

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

a. Multilingual, see "Multilingual Strings" on page 57.

### Instance Attributes (Instance #4, Subnet Mask)

#### Extended

Value is used after module reset.

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

  a. Multilingual, see "Multilingual Strings" on page 57.

### Instance Attributes (Instance #5, Gateway)

#### Extended

Value is used after module reset.

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

  a. Multilingual, see "Multilingual Strings" on page 57.

### Instance Attributes (Instance #6, DHCP)

#### Extended

Value is used after module reset.

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

  a. Multilingual, see "Multilingual Strings" on page 57.

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

### Extended

Changes have immediate effect.

| # | Name | Access | Type | Description |
|---|------|--------|------|-------------|
| 1 | Name[a] | Get | Array of CHAR | 'Comm settings' |
| 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: <br> 00h 'Auto' Auto negotiation (default) <br> 01h '10 HDX' 10Mbit, half duplex <br> 02h '10 FDX' 10Mbit, full duplex <br> 03h '100 HDX' 100Mbit, half duplex <br> 04h '100 HDX' 100Mbit, full duplex |

a. Multilingual, see "Multilingual Strings" on page 57.

## Instance Attributes (Instance #8, Modbus connection timeout)

### Extended

Changes have immediate effect.

| # | Name | Access | Type | Description |
|---|------|--------|------|-------------|
| 1 | Name[a] | Get | Array of CHAR | 'Conn tmo' |
| 2 | Data type | Get | UINT8 | 05h (= UINT16) |
| 3 | Number of elements | Get | UINT8 | 01h (one element) |
| 4 | Descriptor | Get | UINT8 | 07h (read/write/shared access) |
| 5 | Value | Get/Set | UINT16 | Modbus connection timeout in seconds. <br> Value:Meaning: <br> 0 Disabled <br> (other) Timeout in seconds (default = 60) |

a. Multilingual, see "Multilingual Strings" on page 57.

## Instance Attributes (Instance #9, Process active timeout)

### Extended

This instance specifies the process active timeout in milliseconds. For more information regarding this parameter, see "Anybus State Machine" on page 102.

| # | Name | Access | Type | Description |
|---|------|--------|------|-------------|
| 1 | Name[a] | Get | Array of CHAR | 'Process tmo' |
| 2 | Data type | Get | UINT8 | 05h (= UINT16) |
| 3 | Number of elements | Get | UINT8 | 01h (one element) |
| 4 | Descriptor | Get | UINT8 | 07h (read/write/shared access) |
| 5 | Value | Get/Set | UINT16 | Process active timeout in milliseconds. <br> Value:Meaning: <br> 0 Disabled (default) <br> (other) Timeout in milliseconds. |

a. Multilingual, see "Multilingual Strings" on page 57.

## Instance Attributes (Instance #10, DNS1)

### Extended

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

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

a. Multilingual, see "Multilingual Strings" on page 57.

## Instance Attributes (Instance #11, DNS2)

### Extended

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

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

a. Multilingual, see "Multilingual Strings" on page 57.

## Instance Attributes (Instance #12, Host name)

### Extended

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

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

a. Multilingual, see "Multilingual Strings" on page 57.

## Instance Attributes (Instance #13, Domain name)

### Extended

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

| # | 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 | Domain name, 48 characters (pad with space to full length) |

a. Multilingual, see "Multilingual Strings" on page 57.

## Instance Attributes (Instance #14, SMTP Server)

### Extended

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

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

a. Multilingual, see "Multilingual Strings" on page 57.

## Instance Attributes (Instance #15, SMTP User)

### Extended

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

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

a. Multilingual, see "Multilingual Strings" on page 57.

## Instance Attributes (Instance #16, SMTP Password)

### Extended

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

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

a. Multilingual, see "Multilingual Strings" on page 57.

## Instance Attributes (Instance #17, Word Order)

### Extended

This instance defines Modbus word order for data types larger than 16 bits. Changes are valid after reset.

| # | Name | Access | Type | Description |
|---|------|--------|------|-------------|
| 1 | Name[a] | Get | Array of CHAR | 'Word order' |
| 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 | Word order <br> Value:Meaning: <br> 0 "Little-endian" (default) <br> 1 "Big-endian" <br> all other values will set default |

a. Multilingual, see "Multilingual Strings" on page 57.

## 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 Settings | Komm Einstell | Comu Config | Impos. Comun. | Comm Config |
|  | 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 | 100 FDX | 100 FDX | 100 FDX | 100 FDX |
| 8 | Conn tmo | Verb. tmo | Tout Conexion | Tout Conn. | Conn tmo |
| 9 | Process tmo | Prozess tmo | Tout Proceso | Tout Processo | Process tmo |
| 10 | DNS1 | DNS 1 | DNS Primaria | DNS1 | DNS1 |
| 11 | DNS2 | DNS 2 | DNS Secundia. | DNS2 | DNS2 |
| 12 | Host name | Host name | Nombre Host | Nome Host | Nom hôte |
| 13 | Domain name | Domain name | Nobre Domain | Nome Dominio | Dom Domaine |
| 14 | SMTP Server | SMTP Server | Servidor SMTP | Server SMTP | SMTP serveur |
| 15 | SMTP User | SMTP User | Usuario SMTP | Utente SMTP | SMTP utilisa. |
| 16 | SMTP Pswd | SMTP PSWD | Clave SMTP | Password SMTP | SMTP mt passe |
| 17 | Word order | Wortfolge | Orden palabra | Ordine "word" | Ordre-mots |

# 10.6 Socket Interface Object (07h)

## Category

Advanced

## Object Description

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

Note that some of the commands used when accessing this object may require segmentation. For more information, see "Message Segmentation" on page 103.

---

**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 62)
              Delete (See "Command Details: Delete" on page 63)

Instance:    Get_Attribute
              Set_Attribute
              Bind (See "Command Details: Bind" on page 64)
              Shutdown (See "Command Details: Shutdown" on page 65)
              Listen (See "Command Details: Listen" on page 66)
              Accept (See "Command Details: Accept" on page 67)
              Connect (See "Command Details: Connect" on page 68)
              Receive (See "Command Details: Receive" on page 69)
              Receive_From (See "Command Details: Receive_From" on page 70)
              Send (See "Command Details: Send" on page 71)
              Send_To (See "Command Details: Send_To" on page 72)
              IP_Add_membership (See "Command Details: IP_Add_Membership" on page 73)
              IP_Drop_membership (See "Command Details: IP_Drop_Membership" on page 74)
              DNS_Lookup (See "Command Details: DNS_Loopup" on page 75)

## Object Attributes (Instance #0)

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

## Instance Attributes (Sockets #1...8)

### Advanced

| # | Name | Access | Type | Description |
|---|------|--------|------|-------------|
| 1 | Socket type | Get | UINT8 | Value:Socket Type:<br>  00h  SOCK_STREAM, NON-BLOCKING (TCP)<br>  01h  SOCK_STREAM, BLOCKING (TCP)<br>  02h  SOCK_DGRAM, NON-BLOCKING (UDP)<br>  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):<br><br>Value:State:Description:<br>  00h  CLOSED  Closed<br>  01h  LISTEN  Listening for connection<br>  02h  SYN_SENT  Active, have sent SYN<br>  03h  SYN_RECEIVED  Have sent and received SYN<br>  04h  ESTABLISHED  Established.<br>  05h  CLOSE_WAIT  Received FIN, waiting for close<br>  06h  FIN_WAIT_1  Have closed, sent FIN<br>  07h  CLOSING  Closed exchanged FIN; await FIN ACK<br>  08h  LAST_ACK  Have FIN and close; await FIN ACK<br>  09h  FIN_WAIT_2  Have closed, FIN is acknowledged<br>  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<br><br>Value:Meaning:<br>  1  Enabled<br>  0  Disabled (default) |
| 9 | Keep alive | Get/Set | BOOL | Protocol probes idle connection (TCP sockets only)<br><br>Value:Meaning:<br>  1  Enabled<br>  0  Disabled (default) |
| 10 | IP Multicast TTL | Get/Set | UINT8 | IP Multicast TTL value (UDP sockets only).<br>Default = 1. |
| 11 | IP Multicast Loop | Get/Set | BOOL | IP multicast loop back (UDP sockets only)[a]<br><br>Value:Meaning:<br>  1  Enable (default)<br>  0  Disable |
| 12 | Ack delay time | Get/Set | UINT16 | Time for delayed ACKs in ms (TCP sockets only)<br><br>Default = 200ms[b] |
| 13 | TCP No Delay | Get/Set | BOOL | Don't delay send to coalesce packets (TCP).<br><br>Value:Meaning:<br>  1  Delay (default)<br>  0  Don't delay (turn off Nagle's algorithm on socket) |
| 14 | TCP Connect Timeout | Get/Set | UINT16 | TCP Connect timeout in seconds (default = 75s) |

a. Must belong to group in order to get the loop backed message
b. Resolution is 50ms, i.e. 50...99 = 50ms, 100...149 = 100ms, 199 = 150ms etc.

# Command Details: Create

## Category

Advanced

## Details

Command Code.:    03h

Valid for:             Object Instance

## Description

This command creates a socket.

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

- **Command Details**

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

- **Response Details**

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

## Command Details: Delete

### Category

Advanced

### Details

Command Code.:     04h

Valid for:             Object Instance

### Description

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

- If the socket is of TCP-type and a connection is established, the connection is terminated with the RST-flag.
- To gracefully terminate a TCP-connection, it is recommended to use the 'Shutdown'-command (see "Command Details: Shutdown" on page 65) before deleting the socket, causing the connection to be closed with the FIN-flag instead.

- **Command Details**

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

- **Response Details**

  (no data)

## Command Details: Bind

### Category

Advanced

### Details

Command Code.:     10h

Valid for:             Instance

### Description

This command binds a socket to a local port.

- **Command Details**

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

- **Response Details**

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

## Command Details: Shutdown

### Category

Advanced

### Details

Command Code.: 11h

Valid for: Instance

### Description

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

- **Command Details**

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

- **Response Details**

    (no data)

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

Application initiates shutdown:

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

**2.** Receive data on socket until error message Object specific error (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.

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

Advanced

### Details

Command Code.:      12h

Valid for:               Instance

### Description

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

- **Command Details**

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

- **Response Details**

  (no data)

## Command Details: Accept

### Category

Advanced

### Details

Command Code.:     13h

Valid for:          Instance

### Description

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

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

Advanced

### Details

Command Code.:     14h

Valid for:               Instance

### Description

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

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

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

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

Advanced

### Details

Command Code.:     15h

Valid for:             Instance

### Description

This command receives data from a connected socket. Message segmentation may be used to receive up to 1472 bytes (see "Message Segmentation" on page 103).

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 103 |
| 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 103).

| Field | Contents | Notes |
|---|---|---|
| CmdExt[0] | (reserved) | (set to zero) |
| CmdExt[1] | Segmentation Control bits | see "Response Segmentation" on page 104 |
| Data[0...n] | Received data | - |

## Command Details: Receive_From

### Category

Advanced

### Details

Command Code.:     16h

Valid for:              Instance

### Description

This command receives data from an unconnected SOCK_DGRAM-socket. Message segmentation may be used to receive up to 1472 bytes (see "Message Segmentation" on page 103).

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 103 |
| 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 103).

| Field | Contents | Notes |
|---|---|---|
| CmdExt[0] | (reserved) | (set to zero) |
| CmdExt[1] | Segmentation Control bits | see "Response Segmentation" on page 104 |
| Data[0] | Host IP address byte 3 (low) | The host address/port information is only included in the first segment. All data thereafter will start at Data[0] |
| Data[1] | Host IP address byte 2 | |
| Data[2] | Host IP address byte 1 | |
| Data[3] | Host IP address byte 0 (high) | |
| Data[4] | Host port number (low) | |
| Data[5] | Host port number (high) | |
| Data[6...n] | Received data | |

## Command Details: Send

### Category

Advanced

### Details

Command Code.:     17h

Valid for:              Instance

### Description

This command sends data on a connected socket. Message segmentation may be used to send up to 1472 bytes (see "Message Segmentation" on page 103).

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 103).

  | Field | Contents | Notes |
  |---|---|---|
  | CmdExt[0] | (reserved) | (set to zero) |
  | CmdExt[1] | Segmentation Control | see "Command Segmentation" on page 103 |
  | Data[0...n] | Data to send | - |

- **Response Details**

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

# Command Details: Send_To

## Category

Advanced

## Details

Command Code.:    18h

Valid for:          Instance

## Description

This command sends data to a specified host on an unconnected SOCK-DGRAM-socket. Message segmentation may be used to send up to 1472 bytes (see "Message Segmentation" on page 103).

- **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 153).

| Field | Contents | Notes |
|-------|----------|-------|
| CmdExt[0] | (reserved) | (set to zero) |
| CmdExt[1] | Segmentation Control | see "Command Segmentation" on page 103 |
| 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

Advanced

### Details

Command Code.:     19h

Valid for:             Instance

### Description

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

- **Command Details**

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

- **Response Details**

   (no data)

## Command Details: IP_Drop_Membership

### Category

Advanced

### Details

Command Code.:     1Ah

Valid for:                 Instance

### Description

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

- **Command Details**

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

- **Response Details**

(no data)

## Command Details: DNS_Loopup

### Category

Advanced

### Details

Command Code.:    1Bh

Valid for:             Object Instance

### Description

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

- **Command Details**

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

- **Response Details (Success)**

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

## Socket Interface Error Codes (Object Specific)

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

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

# 10.7 SMTP Client Object (09h)

## Category

Advanced

## Object Description

This object groups functions related to the SMTP-client.

See also...

- "File System" on page 13
- "E-mail Client" on page 23
- "Instance Attributes (Instance #12, SMTP Server)" on page 72
- "Instance Attributes (Instance #13, SMTP User)" on page 73
- "Instance Attributes (Instance #14, SMTP Password)" on page 73

## Supported Commands

Object:         Get_Attribute
                Create
                Delete
                Send email from file( "Command Details: Send Email From File" on page 80)

Instance:       Get_Attribute
                Set_Attribute
                Send email( "Command Details: Send Email" on page 81)

## Object Attributes (Instance #0)

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

## Instance Attributes

### Advanced

Instances are created dynamically by the application.

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

# Command Details: Create

### Category

Advanced

### Details

Command Code.:     03h

Valid for:             Object

### Description

This command creates an e-mail instance.

- **Command Details**

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

- **Response Details**

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

## Command Details: Delete

### Category

Advanced

### Details

Command Code.:    04h

Valid for:        Object

### Description

This command deletes an e-mail instance.

- **Command Details**

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

- **Response Details**

  (no data)

## Command Details: Send Email From File

### Category

Advanced

### Details

Command Code.:     11h

Valid for:             Object

### Description

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

File format:

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

```
[To]
recipient

[From]
sender

[Subject]
email subject

[Headers]
extra headers, optional

[Message]
actual email message
```

- **Command Details**

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

- **Response Details**

(no data)

## Command Details: Send Email

### Category

Advanced

### Details

Command Code.:     10h

Valid for:              Instance

### Description

This command sends the specified e-mail instance.

- **Command Details**

  (no data)

- **Response Details**

  (no data)

## Object Specific Error Codes

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

## 10.8 File System Interface Object (0Ah)

### Category

Advanced

### Object Description

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

### Supported Commands

Object:            Get_Attribute
Create( "Command Details: Create" on page 84)
Delete( "Command Details: Delete" on page 85)
Format Disc( "Command Details: Format Disc" on page 94)

Instance:        Get_Attribute
File Open( "Command Details: File Open" on page 85)
File Close( "Command Details: File Close" on page 86)
File Delete( "Command Details: File Delete" on page 86)
File Copy( "Command Details: File Copy" on page 87)
File Rename( "Command Details: File Rename" on page 88)
File Read( "Command Details: File Read" on page 89)
File Write( "Command Details: File Write" on page 90)
Directory Open( "Command Details: Directory Open" on page 90)
Directory Close( "Command Details: Directory Close" on page 91)
Directory Delete( "Command Details: Directory Delete" on page 91)
Directory Read( "Command Details: Directory Read" on page 92)
Directory Create( "Command Details: Directory Create" on page 93)
Directory Change( "Command Details: Directory Change" on page 93)

## Object Attributes (Instance #0)

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

## Instance Attributes

### Advanced

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

## Command Details: Create

### Category

Advanced

### Details

Command Code.:     03h

Valid for:               Object

### Description

This command creates a file operation instance.

- **Command Details**

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

- **Response Details**

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

# Command Details: Delete

## Category

Advanced

## Details

Command Code.:     04h

Valid for:            Object

## Description

This command deletes a file operation instance.

- **Command Details**

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

- **Response Details**

  (no data)

# Command Details: File Open

## Category

Advanced

## Details

Command Code.:     10h

Valid for:            Instance

## Description

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

- **Command Details**

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

- **Response Details**

  (no data)

## Command Details: File Close

### Category

Advanced

### Details

Command Code.: 11h

Valid for: Instance

### Description

This command closes a previously opened file.

- **Command Details**

  (no data)

- **Response Details**

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

## Command Details: File Delete

### Category

Advanced

### Details

Command Code.: 12h

Valid for: Instance

### Description

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

- **Command Details**

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

- **Response Details**

  (no data)

## Command Details: File Copy

### Category

Advanced

### Details

Command Code.:     13h

Valid for:          Instance

### Description

This command makes a copy of a file.

- **Command Details**

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

- **Response Details**

  (no data)

## Command Details: File Rename

### Category

Advanced

### Details

Command Code.:     14h

Valid for:              Instance

### Description

This command renames or moves a file.

- **Command Details**

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

- **Response Details**

  (no data)

## Command Details: File Read

### Category

Advanced

### Details

Command Code.:     15h

Valid for:              Instance

### Description

Reads data from a file previously opened for reading.

- **Command Details**

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

- **Response Details**

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

## Command Details: File Write

### Category

Advanced

### Details

Command Code.:     16h

Valid for:              Instance

### Description

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

- **Command Details**

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

- **Response Details**

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

## Command Details: Directory Open

### Category

Advanced

### Details

Command Code.:     20h

Valid for:              Instance

### Description

This command opens a directory.

- **Command Details**

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

- **Response Details**

(no data)

## Command Details: Directory Close

### Category

Advanced

### Details

Command Code.:     21h

Valid for:          Instance

### Description

This command closes a previously opened directory.

- **Command Details**

    (no data)

- **Response Details**

    (no data)

## Command Details: Directory Delete

### Category

Advanced

### Details

Command Code.:     22h

Valid for:          Instance

### Description

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

- **Command Details**

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

- **Response Details**

    (no data)

## Command Details: Directory Read

### Category

Advanced

### Details

Command Code.:     23h

Valid for:             Instance

### Description

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

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

- **Command Details**

   (no data)

- **Response Details**

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

## Command Details: Directory Create

### Category

Advanced

### Details

Command Code.:     24h

Valid for:            Instance

### Description

This command creates a directory.

- **Command Details**

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

- **Response Details**

  (no data)

## Command Details: Directory Change

### Category

Advanced

### Details

Command Code.:     25h

Valid for:            Instance

### Description

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

- **Command Details**

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

- **Response Details**

  (no data)

## Command Details: Format Disc

### Category

Advanced

### Details

Command Code.: 30h

Valid for: Object

### Description

This command formats the file system.

- **Command Details**

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

- **Response Details**

  (no data)

## Object Specific Error Codes

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

# 10.9 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.<br>See also "Ethernet Host Object (F9h)" on page 121 |

# 11. Host Application Objects

## 11.1 General Information

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

Standard Objects:

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

Network Specific Objects:

- "Modbus Host Object (FAh)" on page 95
- "Ethernet Host Object (F9h)" on page 98

# 11.2 Modbus Host Object (FAh)

## Category

Extended, advanced

## Object Description

This object implements Modbus related settings in the host application.

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

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

See also...

- Anybus CompactCom Software Design Guide, "Error Codes"

## Supported Commands

Object:           Get Attribute
                  Process Modbus Message
                      (See  "Command Details: Process Modbus Message" on page 97)

Instance:         Get Attribute

## Object Attributes (Instance #0)

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

## Instance Attributes (Instance #1)

### Extended

| # | Name | Access | Type | Default Value[a] | Comment |
|---|------|--------|------|-----------------|---------|
| 1 | Vendor name | Get | Array of CHAR | 'HMS' | These settings can (if implemented) be viewed on the built in web pages, and will also be returned in response to a 'Read Device Identification'-request. The maximum allowed length of each string is 244 bytes; strings exceeding this length will be truncated. |
| 2 | Product Code | Get | Array of CHAR | 'ABCC-ETN' | |
| 3 | Major Minor Revision | Get | Array of CHAR | (firmware rev.) | |
| 4 | Vendor URL | Get | Array of CHAR | - | |
| 5 | Product name | Get | Array of CHAR | - | |
| 6 | Model name | Get | Array of CHAR | - | |
| 7 | User Application Name | Get | Array of CHAR | - | See also...<br>- "Web Server" on page 16<br>- "Read Device Identification" on page 46 |
| 8 | Device ID | Get | Array of UINT8 | - | (not used, included for compatibility only) |
| 9 | No. of ADI indexing bits | Get | UINT8 | 04h | Value:Meaning:<br>00h each ADI = 1 Modbus register<br>01h each ADI = 2 Modbus registers<br>02h each ADI = 4 Modbus registers<br>03h each ADI = 8 Modbus registers<br>04h each ADI = 16 Modbus registers<br>05h each ADI = 32 Modbus registers<br>06h each ADI = 64 Modbus registers<br>07h each ADI = 128 Modbus registers<br>(other) (invalid, default value will be used)<br>(see "Application Data (ADIs)" on page 11) |

a. If an attribute is not implemented, this value will be used instead (with the exception of attribute #8 'Device ID').

### Advanced

| # | Name | Access | Type | Default Value[a] | Comment |
|---|------|--------|------|-----------------|---------|
| 10 | Enable Modbus Message forwarding | Get | BOOL | False | Value:Meaning:<br>True Enabled<br>False Disabled<br>(see "Command Details: Process Modbus Message" on page 97) |
| 11 | Modbus Read/Write registers command offsets | Get | SINT16[ 2 ] | [ 0000h, 0000h ] | These settings gives the offsets for the Read/Write Registers command.<br>[READ, WRITE] |

a. If an attribute is not implemented, this value will be used instead (with the exception of attribute #8 'Device ID').

## Command Details: Process Modbus Message

### Category

Advanced

### Details

Command Code.:        10h

Valid for:            Object Instance

### Description

If enabled, this command routes Modbus/TCP communication to the host application.

- **Command Details**

| Field | Contents | Comments |
|---|---|---|
| CmdExt[0] | (reserved) | (ignore) |
| CmdExt[1] | | |
| MsgData[0... n] | Modbus message frame (Query) | - |

- **Response Details**

| Field | Contents | Comments |
|---|---|---|
| CmdExt[0] | (reserved) | (set to zero) |
| CmdExt[1] | | |
| MsgData[0... n] | Modbus message frame (Response) | - |

**Note 1:** The response data size must not exceed 254 bytes.

**Note 2:** If the response contains not data, no Modbus response will be sent to the originator of the request.

# 11.3 Ethernet Host Object (F9h)

## Category

Extended, advanced

## 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 | 01h |
| 3 | Number of instances | Get | UINT16 | 0001h |
| 4 | Highest instance no. | Get | UINT16 | 0001h |

## Instance Attributes (Instance #1)

### Extended

| # | Name | Access | Type | Default[a] | Comment |
|---|------|--------|------|---------|---------|
| 2 | Enable HICP | Get | BOOL | True | Value:Meaning:<br>  True  HICP enabled<br>  False  HICP disabled<br>(see "HICP (Anybus IPconfig)" on page 105) |
| 3 | Enable Web Server | Get | BOOL | True | Value:Meaning:<br>  True  web server enabled<br>  False  web server disabled<br>(see "Web Server" on page 16) |
| 5 | Enable Web ADI access | Get | BOOL | True | Value:Meaning:<br>  True  web ADI access enabled<br>  False  web ADI access disabled<br>(see "Web Server" on page 16) |
| 6 | Enable FTP server | Get | BOOL | True | Value:Meaning:<br>  True  FTP server enabled<br>  False  FTP server disabled<br>(see "FTP Server" on page 14) |
| 7 | Enable admin mode | Get | BOOL | False | Value:Meaning:<br>  True  FTP Admin mode enabled<br>  False  FTP Admin mode disabled<br>(see "FTP Server" on page 14) |
| 8 | Network Status | Set | UINT16 | - | See "Network Status" on page 100 |
| 11 | Enable ACD | Get | BOOL | True | Value:Meaning:<br>  True  Address Conflict Detection enabled<br>  False  Address Conflict Detection disabled<br>(see "Communication Settings" on page 9) |

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

### Advanced

| # | 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. |
| 4 | Enable Modbus/TCP | Get | BOOL | True | Value:Meaning:<br>  True  Modbus/TCP enabled<br>  False  Modbus/TCP disabled |

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.

## Network Status

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

| Bit | Contents | Description |
|---|---|---|
| 0 | Link | Value:Meaning:<br>True  Link sensed<br>False  No link |
| 1 | IP established | Value:Meaning:<br>True  IP address established<br>False  IP address not established |
| 2... 15 | (reserved) | (mask off and ignore) |

# A. Categorization of Functionality

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

## A.1 Basic

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

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

## A.2 Extended

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

## A.3 Advanced

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

# B. Implementation Details

## B.1 Extended LED Functionality

On the Anybus CompactCom Modbus/TCP w. IT module, only one of the two network status LEDs is available through the application interface connector (LED1[A..B]). If needed, there is the possibility to use both network status LEDs by enabling the extended LED functionality. Doing so will disable LED1[A..B] and instead use GIP[0..1] and GOP[0..1] for the two network LEDs.

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

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

### GPIO mode description

| | | Signal | | | |
|---|---|---|---|---|---|
| | | GIP[0..1] | GOP[0..1] | LED1[A..B] | LED2[A..B] |
| **GPIO Configuration** | **Value: 0x0000** (Default) | General purpose input | General purpose output | Network Status LED A | Module Status LED |
| | **Value: 0x0001** (Extended LED functionality) | Network Status LED A GIP0 (yellow) GIP1 (green) | Network Status LED B GOP0 (yellow) GOP1 (green) | Disabled (set to low) | Module Status LED |

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

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

**Note 3:** LED behavior is described in chapter 1. See "Front View" on page 5.

## B.2 SUP-Bit Definition

The supervised bit (SUP) indicates that the network participation is supervised by another network device. In the case of Modbus-TCP, this bit is set when the module is in PROCESS_ACTIVE state and Process Active Timeout is set to a value other than zero.

## B.3 Anybus State Machine

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

| Anybus State | Implementation | Comment |
|---|---|---|
| WAIT_PROCESS | Awaiting Modbus-TCP requests | - |
| ERROR | - | - |
| PROCESS_ACTIVE | A Modbus-TCP request addressed to this node has been received within the specified 'Process Active Timeout' time. | If no timeout value is specified, the module will stay in this state after the first received Modbus-TCP request. |

| Anybus State | Implementation | Comment |
|---|---|---|
| IDLE | This state can be entered as desired by writing to Holding Register no. 0204 (See "Holding Registers (4x)" on page 42). | This is a network specific implementation which can be used as desired to put the module in Idle state. |
| EXCEPTION | Further Modbus-TCP requests will be ignored | - |

# B.4 Application Watchdog Timeout Handling

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

# C. Message Segmentation

## C.1 General

**Category**: Advanced

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

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

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

## C.2 Command Segmentation

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

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

Command segmentation is used for the following commands:

- Get_Configuration_Data (see "Command Details: Get_Configuration_Data" on page 120)
- Send (see "Command Details: Send" on page 69)
- Send To (see "Command Details: Send_To" on page 70)

### Segmentation Control bits (Command)

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

### Segmentation Control bits (Response)

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

When issuing a segmented command, the following rules apply:

- When issuing the first segment, FS must be set.
- When issuing subsequent segments, both FS and LS must be cleared.
- When issuing the last segment, the LF-bit must be set.
- For single segment commands (i.e. size less or equal to 255 bytes), both FS and LS must be set.
- The last response message contains the actual result of the operation.
- The command initiator may at any time abort the operation by issuing a message with AB set.

- If a segmentation error is detected during transmission, an error message is returned, and the current segmentation message is discarded. Note however that this only applies to the current segment; previously transmitted segments are still valid.

# C.3 Response Segmentation

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

Response segmentation is used for responses to the following commands:

- Receive (object specific, see "Command Details: Receive" on page 67)
- Receive From (object specific, see "Command Details: Receive_From" on page 68)

**Segmentation Control bits (Command)**

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

**Segmentation Control bits (Response)**

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

When receiving a segmented response, the following rules apply:

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

# D. HICP (Anybus IPconfig)

## D.1 General

The module supports the HICP protocol used by the Anybus IPconfig utility, which can be downloaded free of charge from the HMS website. This utility may be used to access the network settings of any Anybus product connected to the network via UDP port 3250.

The HICP protocol can be disabled using attribute 2, instance 1, in the EtherNet Host Object.

## D.2 Operation

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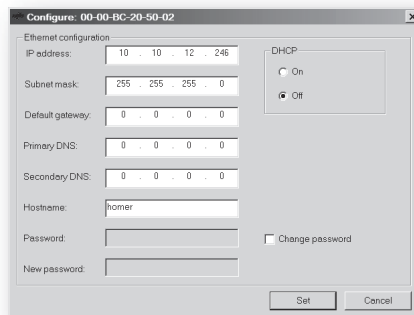


To alter the network settings of the module, double-click on its entry in the list. A window will appear, containing the settings for the module.

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

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

# E. Technical Specification

## E.1 Protective Earth (PE) Requirements

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

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

## E.2 Power Supply

### Supply Voltage

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

### Power Consumption

The Anybus-CompactCom Modbus/TCP is designed to fulfil the requirements of a Class A module. For more information about the power consumption classification used on the Anybus-CompactCom platform, consult the general Anybus-CompactCom Hardware Design Guide.

The current hardware design consumes up to 200mA[1].

## E.3 Environmental Specification

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

## E.4 EMC Compliance

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

---

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

# F. Timing & Performance

## F.1 General Information

This chapter specifies timing and performance parameters that are verified and documented for the Anybus CompactCom Modbus/TCP w. IT-Functionality.

The following timing aspects are measured:

| Category | Parameters | Page |
|---|---|---|
| Startup Delay | T1, T2 | Please consult the Anybus CompactCom Software Design Guide, App. B. |
| NW_INIT Delay | T3 | |
| Telegram Delay | T4 | |
| Command Delay | T5 | |
| Anybus Read Process Data Delay (Anybus Delay) | T6, T7, T8 | |
| Anybus Write Process Data Delay (Anybus Delay) | T12, T13, T14 | |
| Network System Read Process Data Delay (Network System Delay) | T9, T10, T11 | 109 |
| Network System Write Process Data Delay (Network System Delay) | T15, T16, T17 | 109 |

# F.2 Process Data

## F.2.1 Overview
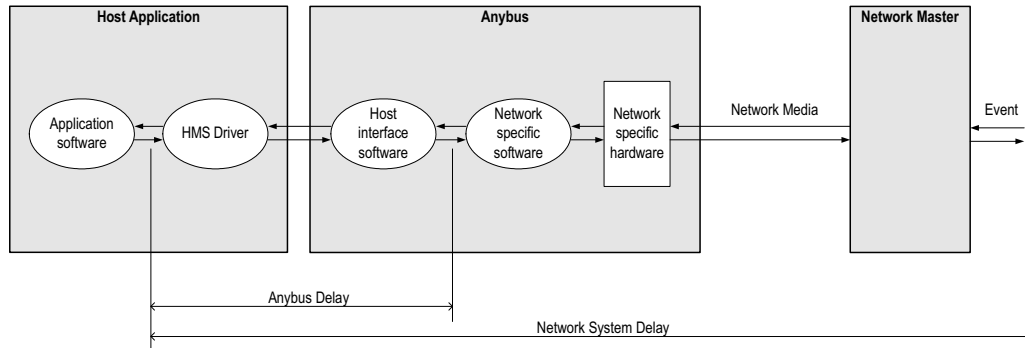


## F.2.2 Anybus Read Process Data Delay (Anybus Delay)

The Read Process Data Delay (labelled 'Anybus delay' in the figure above) is defined as the time measured from just before new data is buffered and available to the Anybus host interface software, to when the data is available to the host application (just after the new data has been read from the driver).

Please consult the Anybus CompactCom Software Design Guide, Appendix B, for more information.

## F.2.3 Anybus Write Process Data Delay (Anybus Delay)

The Write Process Data Delay (labelled 'Anybus delay' in the figure) is defined as the time measured from the point the data is available from the host application (just before the data is written from the host application to the driver), to the point where the new data has been forwarded to the network buffer by the Anybus host interface software.

Please consult the Anybus CompactCom Software Design Guide, Appendix B, for more information.

### F.2.4 Network System Read Process Data Delay (Network System Delay)

The Network System Read Process Data Delay (labelled 'Network System Delay in the figure), is defined as the time measured from the point where an event is generated at the network master to when the corresponding data is available to the host application (just after the corresponding data has been read from the driver).

| Parameter | Description | Typ. | Avg. | Max. | Unit. |
|---|---|---|---|---|---|
| T9 | Network System Read Process Data delay, 8 ADIs (single UINT8) | 19.6 | 15.0 | 20.8 | ms |
| T10 | Network System Read Process Data delay, 16 ADIs (single UINT8) | 16.4 | 13.0 | 18.6 | ms |
| T11 | Network System Read Process Data delay, 32 ADIs (single UINT8) | 18.8 | 13.8 | 19.8 | ms |

#### Conditions:

| Parameter | Conditions |
|---|---|
| Application CPU | - |
| Timer system call interval | 1 ms |
| Driver call interval | 0.2... 0.3 ms |
| No.of ADIs (single UINT8) mapped to Process Data in each direction. | 8, 16 and 32 |
| Communication | Parallel |
| Telegram types during measurement period | Process Data only |
| Bus load, no. of nodes, baud rate etc. | Normal |

### F.2.5 Network System Write Process Data Delay (Network System Delay)

The Network System Write Process Data Delay (labelled 'Network System Delay in the figure), is defined as the time measured from the time after the new data is available from the host application (just before the data is written to the driver) to when this data generates a corresponding event at the network master.

| Parameter | Description | Typ. | Avg. | Max. | Unit. |
|---|---|---|---|---|---|
| T15 | Network System Write Process Data delay, 8 ADIs (single UINT8) | 12.7 | 9.3 | 15.2 | ms |
| T16 | Network System Write Process Data delay, 16 ADIs (single UINT8) | 13.3 | 9.4 | 15.2 | ms |
| T17 | Network System Write Process Data delay, 32 ADIs (single UINT8) | 13.7 | 9.9 | 15.5 | ms |

Conditions: as in "Network System Read Process Data Delay (Network System Delay)" on page 109.

# G. Copyright Notices

This product includes software developed by Carnegie Mellon, the Massachusetts Institute of Technology, the University of California, and RSA Data Security:

*********************************************************************************

Copyright 1986 by Carnegie Mellon.

*********************************************************************************

Copyright 1983,1984,1985 by the Massachusetts Institute of Technology

*********************************************************************************

Copyright (c) 1988 Stephen Deering.

Copyright (c) 1982, 1985, 1986, 1992, 1993

The Regents of the University of California.  All rights reserved.

This code is derived from software contributed to Berkeley by Stephen Deering of Stanford University.

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

- • Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- • Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- • Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

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

*********************************************************************************

Copyright (C) 1990-2, RSA Data Security, Inc. All rights reserved.

License to copy and use this software is granted provided that it is identified as the "RSA Data Security, Inc. MD4 Message-Digest Algorithm" in all material mentioning or referencing this software or this function.

License is also granted to make and use derivative works provided that such works are identified as "derived from the RSA Data Security, Inc. MD4 Message-Digest Algorithm" in all material mentioning or referencing the derived work.

RSA Data Security, Inc. makes no representations concerning either the merchantability of this software or the suitability of this software for any particular purpose. It is provided "as is" without express or implied warranty of any kind.

These notices must be retained in any copies of any part of this documentation and/or software.

*********************************************************************************

Copyright (C) 1991-2, RSA Data Security, Inc. Created 1991. All rights reserved.

License to copy and use this software is granted provided that it is identified as the "RSA Data Security, Inc. MD5 Message-Digest Algorithm" in all material mentioning or referencing this software or this function.

License is also granted to make and use derivative works provided that such works are identified as "derived from the RSA Data Security, Inc. MD5 Message-Digest Algorithm" in all material mentioning or referencing the derived work.

RSA Data Security, Inc. makes no representations concerning either the merchantability of this software or the suitability of this software for any particular purpose. It is provided "as is" without express or implied warranty of any kind.

These notices must be retained in any copies of any part of this documentation and/or software.