



Anybus[®] CompactCom[™] 40

EtherCAT[®]

NETWORK GUIDE

HMSI-27-220 2.1 ENGLISH

Important User Information

Liability

Every care has been taken in the preparation of this document. 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 USA 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.

Copyright © 2016 HMS Industrial Networks AB. All rights reserved.

Anybus® CompactCom™ 40 EtherCAT® Network Guide

HMSI-27-220 2.1

Table of Contents

Page

1	Preface	5
1.1	About this document	5
1.2	Related Documents	5
1.3	Document History	5
1.4	Conventions	6
1.5	Document Specific Conventions.....	6
2	About the Anybus CompactCom 40 EtherCAT	8
2.1	General	8
2.2	Features	9
3	Basic Operation	10
3.1	General Information	10
3.2	EtherCAT Implementation Details.....	13
3.3	CANopen over EtherCAT Implementation Details	15
3.4	Data exchange	16
3.5	File System	17
3.6	Communication Settings in Stand Alone Shift Register Mode	19
3.7	Network Reset Handling	20
3.8	Configured Station Alias (Node Address)	20
3.9	Device ID	20
3.10	Modular Device Profile	21
4	Object Dictionary (CANopen over EtherCAT)	22
4.1	Standard Objects	22
4.2	Manufacturer and Profile Specific Objects.....	27
5	Anybus Module Objects	33
5.1	General Information	33
5.2	Anybus Object (01h)	34
5.3	Diagnostic Object (02h)	35
5.4	Network Object (03h)	37
5.5	Network Configuration Object (04h).....	39
5.6	Socket Interface Object (07h).....	44
5.7	SMTP Client Object (09h).....	61
5.8	Network Ethernet Object (0Ch)	66

6	Host Application Objects	67
6.1	General Information	67
6.2	Assembly Mapping Object (EBh)	68
6.3	Sync Object (EEh)	69
6.4	EtherCAT Object (F5h)	72
6.5	Ethernet Host Object (F9h)	77
7	Web Server	80
7.1	General Information	80
7.2	Default Web Pages	80
7.3	Server Configuration	83
8	FTP Server	86
8.1	General Information	86
8.2	User Accounts	86
8.3	Session Example	87
9	E-mail Client	88
9.1	General Information	88
9.2	How to Send E-mail Messages	88
10	Server Side Include (SSI)	89
10.1	General Information	89
10.2	Include File	89
10.3	Command Functions	89
10.4	Argument Functions	103
10.5	SSI Output Configuration	106
11	JSON	107
11.1	General Information	107
11.2	JSON Objects	107
11.3	Example	113
A	Categorization of Functionality	115
A.1	Basic	115
A.2	Extended	115
B	Implementation Details	116
B.1	SUP-bit Definition	116
B.2	Anybus State Machine	116
B.3	Application Watchdog Timeout Handling	116

C	Technical Specification	117
C.1	Front View	117
C.2	Functional Earth (FE) Requirements.....	118
C.3	Power Supply	118
C.4	Environmental Specification.....	119
C.5	EMC Compliance.....	119
D	Timing & Performance	120
D.1	General Information	120
D.2	Internal Timing.....	120
E	Secure HICP (Secure Host IP Configuration Protocol)	123
E.1	General.....	123
E.2	Operation	123
F	Copyright Notices	124

This page intentionally left blank

1 Preface

1.1 About this document

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

The reader of this document is expected to be familiar with high level software design and communication systems in general. The information in this network guide should normally be sufficient to implement a design. However if advanced EtherCAT specific functionality is to be used, in-depth knowledge of EtherCAT networking internals and/or information from the official EtherCAT specifications may be required. In such cases, the persons responsible for the implementation of this product should either obtain the EtherCAT specification to gain sufficient knowledge or limit their implementation in such a way that this is not necessary.

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

1.2 Related Documents

Related documents

Document	Author
Anybus CompactCom 40 Software Design Guide	HMS
Anybus CompactCom M40 Hardware Design Guide	HMS
Anybus CompactCom Host Application Implementation Guide	HMS
IEC 61158-6	IEC
CiA Draft Standard 301 v4.02	CAN in Automation

1.3 Document History

Summary of changes in this version

Change	Where (section no.)
Added section about Ethernet over EtherCAT (EoE)	3.1.5
Added File System section	3.5
Added section about communication in stand alone shift register mode	3.6
Added Ethernet related instances to the Network Configuration Object	5.5
Added the Socket Interface Object	5.6
Added the SMTP Client Object	5.7
Added the Network Ethernet Object	5.8
Added attribute #17 and #18 to the EtherCAT Object	6.4
Added the Ethernet Host Object	Ethernet Host Object (F9h), p. 77
Added Web Server chapter	7
Added FTP Server chapter	8
Added E-mail Client Chapter	9
Added Server Side Include (SSI) chapter	10
Added JSON chapter	11

Revision list

Version	Date	Author	Description
1.30	2015-10-23	KeL	Last FM version.
2.0	2016	KeL	Moved from FM to XML Misc. updates
2.1	2016-09-12	KaD	Added content for EoE (Ethernet over EtherCAT)

1.4 Conventions

Unordered (bulleted) lists are used for:

- Itemized information
- Instructions that can be carried out in any order

Ordered (numbered or alphabetized) lists are used for instructions that must be carried out in sequence:


1. First do this,
2. Then open this dialog, and
 - a. set this option...
 - b. ...and then this one.

Bold typeface indicates interactive parts such as connectors and switches on the hardware, or menus and buttons in a graphical user interface.

Monospaced text is used to indicate program code and other kinds of data input/output such as configuration scripts.

This is a cross-reference within this document: [Conventions, p. 6](#)

This is an external link (URL): www.hms-networks.com

 *This is additional information which may facilitate installation and/or operation.*



This instruction must be followed to avoid a risk of reduced functionality and/or damage to the equipment, or to avoid a network security risk.

**Caution**

This instruction must be followed to avoid a risk of personal injury.

**WARNING**

This instruction must be followed to avoid a risk of death or serious injury.

1.5 Document Specific Conventions

- The terms “Anybus” or “module” refers to the Anybus CompactCom module.
- The terms “host” or “host application” refer to the device that hosts the Anybus.
- Hexadecimal values are written in the format NNNNh or 0xNNNN, where NNNN is the hexadecimal value.

-
- A byte always consists of 8 bits.
 - The terms “basic” and “extended” are used to classify objects, instances and attributes.

2 About the Anybus CompactCom 40 EtherCAT

2.1 General

The Anybus CompactCom 40 EtherCAT communication module provides instant EtherCAT conformance tested connectivity via the patented Anybus CompactCom host interface. Any device that supports this standard can take advantage of the features provided by the module, allowing seamless network integration regardless of network type.

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



EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.

2.2 Features

- CANopen over EtherCAT (CoE)
- Support for Modular Device Profile
- RJ45 or M12 connectors
- DS301 compliant
- Galvanically isolated bus electronics
- Network Identity customization
- EMCY support
- Up to 57343 ADIs can be accessed from the network as Manufacturer Specific Objects and Device Profile Specific Objects (generic mode).
- Up to 16383 ADIs can be accessed from the network as Manufacturer Specific Objects and Device Profile Specific Objects (modular device profile enabled)
- Up to 1486 bytes of fast cyclic I/O in each direction
- EtherCAT Slave Interface file provided by HMS
- Support for Sync0 functionality using distributed clocks
- Ethernet over EtherCAT (EoE)
- Web server with customizable content
- FTP server
- E-mail client
- Server Side Include (SSI) functionality
- JSON functionality
- File access over EtherCAT (FoE)
- Support for process data remap from the network
- Network cycle time down to 100 μ s
- Possible to implement DS402 drive profile, Semi device profiles, and other device profiles



If the TwinCAT 3, or a later version of 2.11, tool is used, the max amount of process data will be 1473 bytes, due to limitations in the tool.

3 Basic Operation

3.1 General Information

3.1.1 Software Requirements

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

- ADIs with instance numbers up to 57343 (DFFFh) can be accessed from the network. If the Modular Device Profile is implemented and running, instance numbers are limited to 16383 (3FFFh).
- When mapping ADIs to process data, there is a limit of 1486 elements or 1486 bytes, whichever comes first, that can be mapped in either direction.
- The flexible nature of the Anybus concept allows the application to modify the behavior on EtherCAT in ways which contradict the generic EtherCAT Slave Information file or in other ways voids network certification. Those responsible for the implementation of the final product should ensure that their level of implementation matches their own requirements and policies regarding network certification and interoperability.
- The use of advanced EtherCAT-specific functionality may require in-depth knowledge in EtherCAT networking internals and/or information from the official EtherCAT specifications. In such cases, those responsible for the implementation of the product should either obtain the EtherCAT specification to gain sufficient knowledge or limit their implementation in such a way that this is not necessary.



If the TwinCAT 3, or a later version of 2.11, tool is used, the max amount of process data will be 1473 bytes, due to limitations in the tool.

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

3.1.2 EtherCAT Slave Interface (ESI) File

Each device on EtherCAT is associated with a EtherCAT Slave Interface (ESI) file in XML format, which holds a description of the device and its functions.

HMS Industrial Networks AB supplies a generic ESI file which can serve as a basis for new implementations. However, due to the flexible nature of the Anybus CompactCom concept, it is possible to alter the functionality of the module in ways which contradicts the information in this file. This may cause trouble if the master expects the configuration stated in the file. In some cases, these problems can be rectified by the end user by manually changing I/O parameters etc. To ensure interoperability and to reduce the complexity for the end user, it is strongly recommended to create a custom ESI file to match the final implementation of the product.

The EtherCAT Technology Group (ETG) requires that the Vendor ID is changed to reflect the vendor of the end product. The following scenarios, among others, may require additional changes to the EtherCAT Slave Interface file.

- The use of a custom Product Code.
- The use of an own Vendor ID.
- Change of the product revision.
- The host application supports the Remap_ADI commands.
- The use of Ethernet over EtherCAT (EoE).
- Slow application response times. Explicit requests should be handled within 1 ms in order to comply with the generic ESI file supplied by HMS. This may not be sufficient for a slow serial link with a substantial amount of I/O (in such case, the mailbox timeout value in the file needs to be increased accordingly).



Note that deviations from the generic ESI file requires the use of custom Product Codes apart from the required custom Vendor ID.

3.1.3 Device Identity

In a generic implementation (i.e. no network specific support is implemented) the module will appear as a generic HMS device with the following identity information:

Object Entry	Value
Vendor ID	E000 001Bh (HMS Industrial Networks Secondary Vendor ID, has to be replaced by the Vendor ID of the end product vendor.)
Product Code	0000 0036h (Anybus CompactCom 40 EtherCAT)
Device Name	Anybus CompactCom 40 EtherCAT
Serial Number	(Assigned during manufacturing)

By implementing support for the EtherCAT Object (F5h), the module can be customized to appear as a vendor specific implementation rather than a generic Anybus device. For the end product to pass the ETG conformance tests and be certified, a separate Vendor ID has to be requested from ETG.

See also...

- [EtherCAT Object \(F5h\), p. 72](#)

3.1.4 File Access over EtherCAT (FoE)

The module supports File Access over EtherCAT (FoE) for downloading firmware files from a Client machine to the Server. All FoE requests not concerning files with the extension `.hiff` (HMS firmware files) or the extension `.nfo`, will be forwarded to the Application File System Interface object. Since FoE offers only very basic FTP functionality, saved files (other than `.hiff` files) will end up in the root folder of the Application File System Interface object.

If a firmware file, downloaded through FoE, is pending for update, the file with the extension `.hiff` will be possible to upload via FoE.

3.1.5 Ethernet over EtherCAT (EoE)

The module supports transparent tunneling of non-EtherCAT Ethernet frames to and from an EtherCAT slave, using Ethernet over EtherCAT (EoE). With Ethernet over EtherCAT (EoE), the following features are supported:

- Web server with customizable content
- FTP server
- E-mail client
- Server Side Include (SSI) functionality
- JSON functionality

Since the Ethernet frames are embedded in mailbox communication, the performance will be reduced compared to normal Ethernet communication. The data throughput will depend on...

- The EtherCAT process data cycle time
- The mailbox size (in bytes)



To be able to use Ethernet over EtherCAT (EoE), the Anybus CompactCom 40 device needs to be assigned a MAC address. Users with devices containing older software (prior to software version 2.00) will need to set the MAC address manually, in the Ethernet Host Object, to be able to use Ethernet over EtherCAT (EoE).

To indicate or remove support for Ethernet over EtherCAT (EoE) in the ESI file, see the following:

To support Ethernet over EtherCAT (EoE), the `<Mailbox>` element should look like this:

```
<Mailbox DataLinkLayer="1">
  <EoE IP="0" MAC="0" TimeStamp="0" />
  <CoE SdoInfo="1" CompleteAccess="1" PdoAssign="0" PdoConfig="0"
  PdoUpload="1"/>
  <FoE/>
</Mailbox>
```

To remove support for Ethernet over EtherCAT (EoE), the `<Mailbox>` element should look like this:

```
<Mailbox DataLinkLayer="1">
  <CoE SdoInfo="1" CompleteAccess="1" PdoAssign="0" PdoConfig="0"
  PdoUpload="1"/>
  <FoE/>
</Mailbox>
```

3.2 EtherCAT Implementation Details

3.2.1 General Information

The module implements a full EtherCAT slave with the following basic properties:

Application Layer:	CANopen over EtherCAT
FMMUs.	4
Sync Managers.	4
RAM Size:	16 kByte

See also...

- [CANopen over EtherCAT Implementation Details, p. 15](#)

3.2.2 EtherCAT Synchronization

EtherCAT synchronization and jitter accuracy may depend on different things:

- How often the master sends out sync frames
- Temperature variations in the environment (large impact)
- The implementation of the EtherCAT slave device
- Which Ethernet physical layer is used in the slave devices (RJ45, E-Bus etc.)

The Anybus CompactCom 40 EtherCAT modules all demonstrate less than 1 μ s synchronization accuracy. For RJ45 products the accuracy may be around 50 ns under good conditions, and for E-Bus products around 30 ns.

3.2.3 Sync Managers

The module features four Sync Managers:

Sync Manager 0	Used for mailbox write transfers (Master to Slave). The module has a configurable write mailbox size with default size of 276 bytes, corresponding to 255 bytes plus relevant protocol headers and padding.
Sync Manager 1	Used for mailbox read transfers (Slave to Master). The module has a configurable read mailbox size with default size of 276 bytes, corresponding to 255 bytes plus relevant protocol headers and padding.
Sync Manager 2	Contains the RxPDOs (in practice, Sync Manager 2 holds the Read Process Data).
Sync Manager 3	Contains the TxPDOs (in practice, Sync Manager 3 holds the Write Process Data).

3.2.4 FMMUs

There are four FMMUs. The EtherCAT master can use the FMMUs freely for any purpose.

3.2.5 Addressing Modes

As a full EtherCAT, the module supports the following addressing modes:

- position addressing
- node addressing
- logical addressing

3.2.6 Watchdog Functionality

Apart from the standard watchdog functionality, the following additional watchdog is implemented:

Output I/O Sync Manager Watchdog

If enabled, this watchdog monitors the PDO communication towards the Anybus module. If the master doesn't update the Read Process Data within the specified time period, this will trigger a timeout condition in the module, causing it to shift from OPERATIONAL to SAFE-OPERATIONAL. The supervision-bit (SUP) is also affected by this.

The sync manager watchdog is enabled by default in the ESI file, with a default time period of 100 ms.

The sync manager watchdog can always be disabled/enabled manually in the configuration tool for the master.

See also...

- [SUP-bit Definition, p. 116](#)

3.3 CANopen over EtherCAT Implementation Details

3.3.1 General Information

As mentioned previously, the module implements CANopen over EtherCAT. The object implementation is based on the DS301 communication profile.

See also...

- [Data exchange, p. 16](#)
- [Object Dictionary \(CANopen over EtherCAT\), p. 22](#)

3.3.2 Implemented Services

The module implements the following CANopen services:

Service	Description
SDO Download Expedited	Writes up to four octets to the slave
SDO Download Normal	Writes up to a negotiated number of octets to the slave
Download SDO Segment	Writes additional data if the object size exceeds the negotiated no. of octets.
SDO Upload Expedited	Reads up to four octets from the slave
SDO Upload Normal	Reads up to a negotiated number of octets from the slave§
Upload SDO Segment	Reads additional data if the object size exceeds the negotiated no. of octets
Abort SDO Transfer	Server abort of service in case of an erroneous condition
Get OD List	Reads a list of available indices
Get Object Description	Reads details of an index
Get Entry Description	Reads details of a subindex
Emergency	Reports unexpected conditions and diagnostic events.

3.4 Data exchange

3.4.1 Application Data (ADI)

Application Data Instances (ADIs) can be accessed from the network via dedicated object entries in the Manufacturer Specific range and the Profile range (2001h - FFFFh). The SDO information protocol allows nodes to retrieve the name and data type of the ADI.

See also...

- [Manufacturer and Profile Specific Objects, p. 27](#)

3.4.2 Process Data

ADIs mapped as Process Data will be exchanged cyclically as Process Data Objects (PDOs) on the bus. The actual PDO map is based on the Process Data map specified during startup or how the application is implemented. It can be changed from the network during runtime, if the application has implemented the remap commands in the Application Data Object.

The module supports up to 6 TPDOs and up to 6 RPDOs, each supporting up to 254 SDO mappings. Each SDO equals one Process Data mapped ADI element (i.e. mapping multiple element ADIs will result in multiple SDO mappings). The number of TPDOs and RPDOs can be extended if the Assembly Mapping Object is implemented.

To gain in configurability, the Assembly Mapping Object can be used to remap and replace the Process Data map specified at startup. Each PDO will be represented by an instance in the Assembly Mapping Object. The PDOs will then be remapped when the module enters the Safe-Operational state.

If the Modular Device Object is implemented, i.e. the Modular Device Profile is enabled, the Assembly Mapping Object will be ignored.



Preferably, the EtherCAT Slave Information file should be altered to match the actual Process Data implementation. This is not a general requirement, but it has a positive impact on compatibility with 3rd party masters.

See also...

- [Standard Objects, p. 22](#)
- [Manufacturer and Profile Specific Objects, p. 27](#)
- [Assembly Mapping Object \(EBh\), p. 68](#)
- Application Data Object (see Anybus CompactCom 40 Software Design Guide)
- Modular Device Object (see Anybus CompactCom 40 Software Design Guide)

3.5 File System

3.5.1 Overview

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

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

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

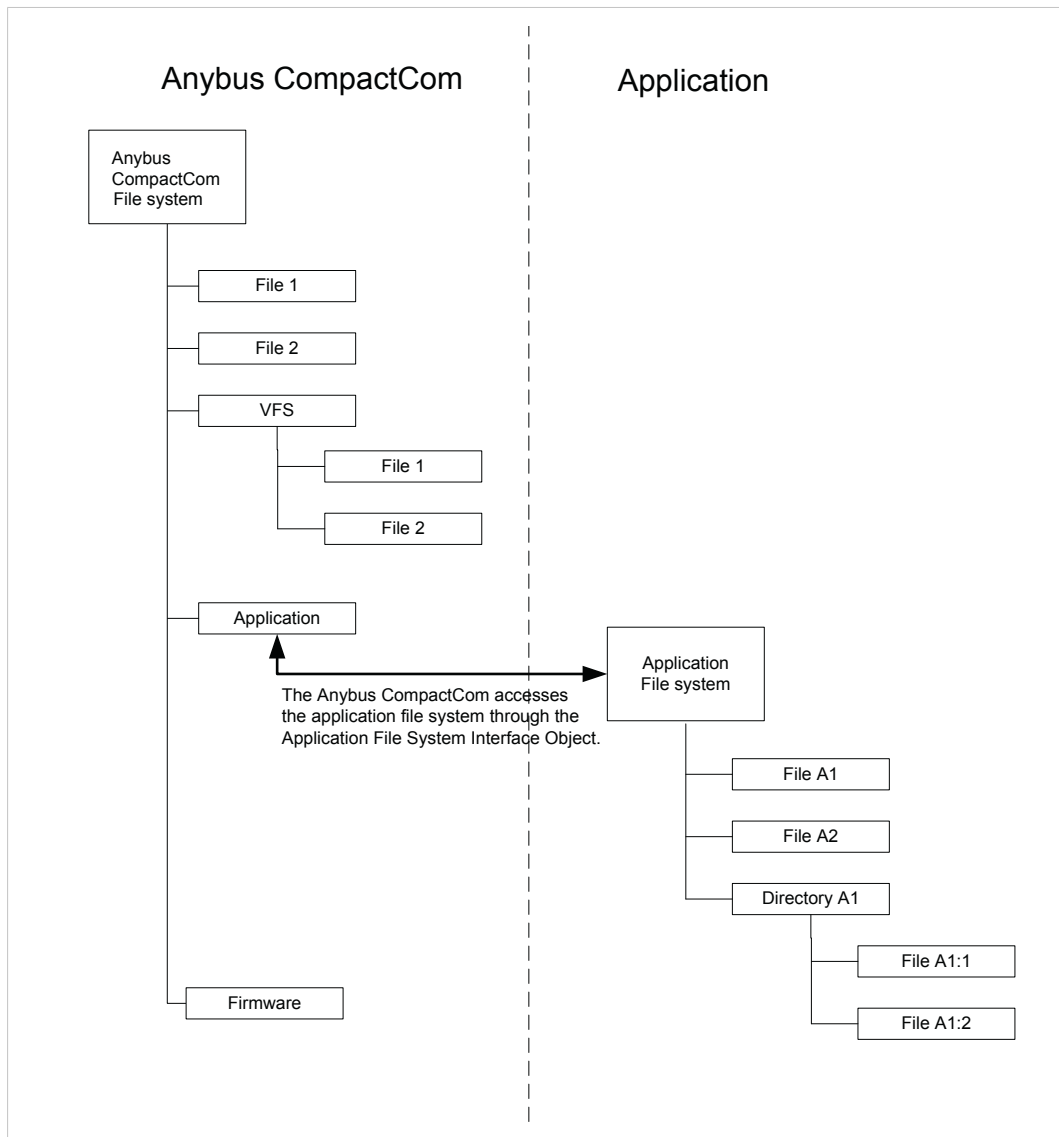


Fig. 1

3.5.2 General Information

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

Maximum number of directories and files that can be stored in the root directory is 511, if only short filenames are used (8 bytes name + 3 bytes extension). If longer filenames are used, less than 511 directories/files can be stored. This limitation does not apply to other directories in the file system.

The file system uses the following conventions:

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

See also...

- [FTP Server, p. 86](#)
- [Web Server, p. 80](#)
- [E-mail Client, p. 88](#)
- [Server Side Include \(SSI\), p. 89](#)



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

The following operations will erase one or more flash segments:

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

3.5.3 System Files

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

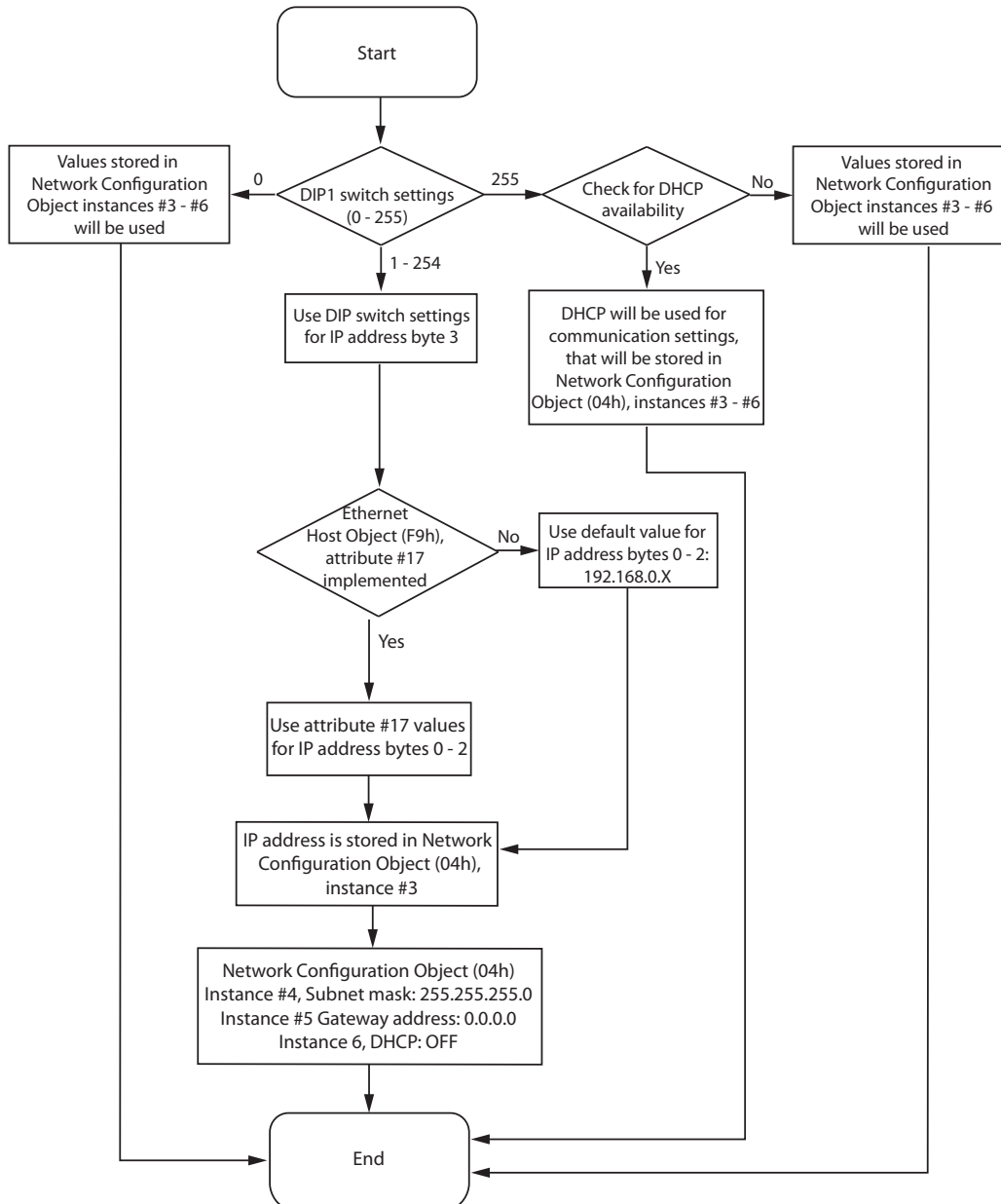
Example 1:

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

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

3.6 Communication Settings in Stand Alone Shift Register Mode

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



See also...

[Ethernet Host Object \(F9h\), p. 77](#)

[Network Configuration Object \(04h\), p. 39](#)

Anybus CompactCom M40 Hardware Design Guide

3.7 Network Reset Handling

3.7.1 Reset Node

If a valid firmware has been downloaded via FoE (File access over EtherCAT), the Anybus CompactCom 40 EtherCAT will send a reset type 00h ('Power-on reset') to the application at the transition from BOOT to INIT.

Prior to the reset command a Reset_Request command has to be sent to the host application to make sure that a reset can be performed.

3.7.2 Restore Manufacturer Parameters to Default

Upon receiving a "Restore Manufacturer Parameters to Default" request from the network, the module will issue a reset command to the Application Object (FFh) with CmdExt[1] set to 01h (Factory default reset).

A factory default reset can only be performed in the EtherCAT state PREOPERATIONAL. Performing a reset in another state than PREOPERATIONAL will generate SDO abort code 08000020h (invalid state).

See also...

- [Standard Objects, p. 22](#), entry 1011h ('Restore Parameters')

3.8 Configured Station Alias (Node Address)

The Configured Station Alias (node address) range is 1... 65535. Address 0 indicates that the device has yet to be configured. The Configured Station Alias is stored in the slave EEPROM and may be used by some masters as a node address.

For most applications it is recommended to leave the Configured Station Alias unchanged, but it is possible to assign each slave an address from the network.

3.9 Device ID

The Device ID is used by the master to explicitly identify a slave. This is e.g. useful when changing a faulty device during runtime, a so called HotConnect application. A preconfigured device can be entered into the network, and its Device ID can be set to the same Device ID as the faulty device was appointed.

It is also useful to prevent cable swapping when there are two or more identical devices on the network.

The Device ID range is 1... 65535. Address 0 indicates that the device has yet to be configured. The value can be set using the Network Configuration Object, instance 1.



In the Anybus CompactCom M30 EtherCAT, the Network Configuration Object, instance 3 was used for the Device ID

See also...

- [Network Configuration Object \(04h\), p. 39](#)

3.10 Modular Device Profile

The Anybus CompactCom 40 EtherCAT supports the Modular Device Profile, that is enabled if the Modular Device Object is implemented in the application. Running this profile, the module supports a maximum of 63 slots, including the coupler in slot 0. The maximum number of ADIs, that can be accessed from the network, is 16383.

The value of the Device Type Object (1000h) is changed to 00005001h.

Enabling the Modular Device Profile will override the settings of the Assembly Mapping Object, if this object is implemented.

See also....

- Modular Device Object (Anybus CompactCom 40 Software Design Guide)
- [Modular Device Profile, Object Entries, p. 30](#)

4 Object Dictionary (CANopen over EtherCAT)

4.1 Standard Objects

4.1.1 General

The standard object dictionary is implemented according to the DS301 communication profile. Note that certain object entries correspond to settings in the EtherCAT Object (F5h), and the Diagnostic Object (02h).

4.1.2 Object Entries

Index	Object Name	Sub-index	Description	Type	Access	Notes
1000h	Device Type	00h	Device Type	U32	RO	Default 0000 0000h (No profile). Can be managed through the EtherCAT Object, which can optionally be implemented in the host application. See EtherCAT Object (F5h) , p. 72. If the host application Modular Device Object is implemented, the default value is 0000 5001h.
1001h	Error register	00h	Error register	U8	RO	This information managed through the Diagnostic Object, see Diagnostic Object (02h) , p. 35.
1003h	Pre-defined error field	00h	Number of errors	U8	RW	
		01h...-05h	Error field	U32	RO	
1008h	Manufacturer device name	00h	Manufacturer device name	Visible string	RO	These entries are managed through the EtherCAT Object, which can optionally be implemented in the host application. See EtherCAT Object (F5h) , p. 72.
1009h	Manufacturer hardware version	00h	Manufacturer hardware version	Visible string	RO	
100Ah	Manufacturer software version	00h	Manufacturer Software version	Visible string	RO	
1011h	Restore parameters	00h	Largest sub index supported	U8	RO	01h
		01h	Restore all default parameters	U32	RW	-
1018h	Identity object	00h	Number of entries	U8	RO	Number of entries
		01h	Vendor ID	U32	RO	These entries are managed through the EtherCAT Object, which can optionally be implemented in the host application. See EtherCAT Object (F5h) , p. 72.
		02h	Product Code	U32	RO	
		03h	Revision Number	U32	RO	
	04h	Serial Number	U32	RO		
10F1h	Error Settings object	00h	Number of entries	U8	RO	
		02h	Sync error counter limit	U32	RW	
1600h - 1xxxh	Receive PDO mapping	00h	No. of mapped application objects in PDO	U8	RO/RW	No. of mapped objects (0.. 254), see Mapping ADIs on PDOs , p. 25 for more information.
		01h	Mapped object #1	U32	RO/RWa	-
		02h	Mapped object #2	U32	RO/RWa	-

Index	Object Name	Sub-index	Description	Type	Access	Notes
		-
		NNh	Mapped object #NN	U32	RO/RWa	-
1A00h - 1xxxh	Transmit PDO mapping	00h	No. of mapped application objects in PDO	U8	RO/RWa	No. of mapped objects (0.. 254), see Mapping ADIs on PDOs, p. 25 for more information.
		01h	Mapped object #1	U32	RO/RWa	-
		02h	Mapped object #2	U32	RO/RWa	-
		-
		NNh	Mapped object #NN	U32	RO/RWa	-
1C00h	Sync Manager Communication Type	00h	Number of entries	U8	RO	4
		01h	Mailbox wr	U8	RO	1
		02h	Mailbox rd	U8	RO	2
		03h	Process Data out	U8	RO	3
		04h	Process Data in	U8	RO	4
1C12h	Sync Manager Rx PDO Assign	00h	No. of assigned PDOs	U8	RO/RW	When using static PDO mapping this subindex is read only. When using dynamic PDO mapping, it is writable.
		01h - NNh	Assigned PDO	U16	RO/RW	
1C13h	Sync Manager Tx PDO Assign	00h	No. of assigned PDOs	U8	RO/RW	If more than one sync mode is supported, this entry is writable.
		01h - NNh	Assigned PDO	U16	RO/RW	
1C32h	SM output parameter	00h	Max subindex supported	U8	RO	12 (0Bh)
		01h	Sync mode	U16	RO/RW	00h: Free Run 02h: DC Sync0 See Sync Object (EEh), p. 69 .
		02h	Cycle time	U32	RW	Cycle time in nanoseconds
		03h	Shift time	U32	RW	Shift time in nanoseconds
		04h	Synchronization Types supported	U16	RO	Bit 0 set: FREE_RUN supported Bit 2 set: DC Sync0 supported. Bit 5 set: Output shift with local timer All other bits are set to 0 See Sync Object (EEh), p. 69 .
		05h	Minimum cycle time	U32	RO	Minimum cycle time in nanoseconds.
		06h	Output Calc and Copy Time	U32	RO	Output Calc and Copy Time in nanoseconds.
		09h	Delay time	U32	RO	Delay time in nanoseconds. Always set to 0.
		0Ch	Cycle Time Too Small	U16	RO	Cycle time to small
1C33h	SM input parameter	00h	Max subindex supported	U8	RO	12 (0Bh)
		01h	Sync mode	U16	RO/RW	00h: Free Run 02h: DC Sync0 See Sync Object (EEh), p. 69 .

Index	Object Name	Sub-index	Description	Type	Access	Notes
		02h	Cycle time	U32	RW	Cycle time in nanoseconds, same value as 1C32h, subindex 2.
		03h	Shift time	U32	RW	Shift time in nanoseconds.
		04h	Synchronization Types supported	U16	RO	Bit 0 set: FREE_RUN supported Bit 2 set: DC Sync0 supported. Bit 5 set: Input shift with local timer All other bits are set to 0 See Sync Object (EEh) , p. 69.
		05h	Minimum cycle time	U32	RO	Minimum cycle time in nanoseconds, same value as 1C32h, subindex 5.
		06h	Input Calc and Copy Time	U32	RO	Input Calc and Copy Time in nanoseconds.
		0Ch	Cycle Time Too Small	U16	RO	Cycle time too small, same value as 1C32h, subindex 12 (0Bh).

Receive PDO mapping and Transmit PDO mapping are writable when dynamic process data is supported by the application (remap commands).

Mapping ADIs on PDOs

The Receive PDO mapping objects (1600h - 1xxxh) and the Transmit PDO mapping objects (1A00h - 1xxxh) are configured depending on how the host application is set up:

Mode	Access	Number of objects (in each direction)	Number of sub indices per object	Notes
Generic, static mapping	RO	1 - 6 Depends on how many ADI mapping items that are mapped by the application during setup. Each PDO can hold 254 ADI mapping items.	1 - 254 Depends on how many ADI mapping items that are mapped by the application during setup. One PDO mapping object at the time will be filled with mapped items.	
Generic, dynamic mapping	RW	1 - 6 Depends on how many ADI mapping items that are mapped by the application during setup. Each PDO can hold 254 ADI mapping items.	254 (except the 6th object, that has 216 sub indices as the maximum number of entries is 1486)	If the TwinCAT 3, or a later version of 2.11, tool is used, the maximal number of entries will be 1473 bytes, due to limitations in the tool.
Assembly Mapping Object implemented in host	RO/RW	Number of assembly mapping instances in that direction (max 63)	1486/(number of objects) (max 254)	See Assembly Mapping Object (EBh), p. 68 for more information. Access is RO if the corresponding assembly instance is static, RW if it is dynamic If the TwinCAT 3, or a later version of 2.11, tool is used, the maximal number of entries will be 1473 bytes, due to limitations in the tool.
Modular device, static mapping	RO	Same as the number of modules that have objects mappable in that direction (max 63)	Same as the number of ADIs mapped in that direction during setup	
Modular device, dynamic mapping	RW	Same as the number of modules that have objects mappable in that direction (max 63)	1486/(number of objects) (max 254)	If the TwinCAT 3, or a later version of 2.11, tool is used, the maximal number of entries will be 1473 bytes, due to limitations in the tool.

Please note that in Generic mode and in Modular Device Profile mode, the ADI to PDO mapping is performed by the application at startup. Also note that if both the Assembly Mapping Object and the Modular Device Object are implemented in the host, the Modular Device Profile mode will be enabled, overriding the settings of the Assembly Mapping Object.

The PDO assignment objects (1C12h and 1C13h) are configured depending on how the host application is set up:

Mode	Access	Number of sub indices per object	Content
Generic, static mapping	RO	Same as the number of PDO mapping objects in that direction.	All PDO mapping objects in that direction.
Generic, dynamic mapping	RW	Same as the number of PDO mapping objects in that direction.	All PDO mapping objects in that direction.
Assembly Mapping Object implemented in host	RW	Same as the number of PDO mapping objects in that direction.	The first PDO in that direction
Modular device, static mapping	RO	Same as the number of PDO mapping objects in that direction.	All PDO mapping objects in that direction.
Modular device, dynamic mapping	RW	Same as the number of PDO mapping objects in that direction.	All PDO mapping objects in that direction.

4.2 Manufacturer and Profile Specific Objects

4.2.1 General

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



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

4.2.2 Network Data Format

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

Anybus Data Type	Network Data Type
BOOL	UNSIGNED8
SINT8	INTEGER8
SINT16	INTEGER16
SINT32	INTEGER32
UINT8	UNSIGNED8
UINT16	UNSIGNED16
UINT32	UNSIGNED32
CHAR	VISIBLE_STRING
ENUM	UNSIGNED8 or ENUM
BITS8	BITARR8
BITS16	BITARR16
BITS32	BITARR32
OCTET	OCTET_STRING
SINT64	INTEGER64
UINT64	UNSIGNED64
FLOAT	REAL32
PAD0-16	NULL
BIT1 - BIT7	BIT1 - BIT7

ADIs with multiple elements are represented either as arrays (all elements share the same data type) or as records (the elements may have different data types). Exceptions to this are CHAR which will always be represented as VISIBLE_STRING, and OCTET which will always be represented as OCTET_STRING.

Single element ADIs are represented as a simple variable, with the exception of CHAR which will always be represented as VISIBLE_STRING, and OCTET which will always be represented as OCTET_STRING.

4.2.3 Error Codes

If an error occurs when an object in the application is requested from the module, the error code returned is translated to an CANopen abort code as follows:

Anybus CompactCom Error Code	CANopen Abort Code	Description (CANopen)
Reserved	N/A	-If an error occurs when an object in the application is requested from the module, the error code returned is translated to an CANopen abort code as follows:
Fragmentation error (serial mode)	N/A	-
Invalid message format	N/A	-
Unsupported object	0602 0000h	Object does not exist in the object dictionary.
Unsupported instance	0602 0000h	Object does not exist in the object dictionary.
Unsupported command	0604 0043h	General parameter incompatibility reason.
Invalid CmdExt[0]	0602 0000h	Object does not exist in the object dictionary. (ADI access)
Invalid CmdExt[1]	0609 0011h	Subindex does not exist. (ADI access]
Attribute not settable	0601 0002h	Attempt to write a read only object.
Attribute not gettable	0601 0001h	Attempt to read a write only object.
Too much data	0607 0012h	Data type does not match, length of service parameter too long.
Not enough data	0607 0013h	Data type does not match, length of service parameter too short.
Out of range	0609 0030h	Value range of parameter exceeded (only for write access).
Invalid state	0800 0022h	Data cannot be transferred or stored to the application because of the present device state.
Out of resources	0504 0005h	Out of memory
Value too high	0609 0031h	Value of parameter higher than upper limit (only for write access).
Value too low	0609 0032h	Value of parameter lower than lower limit (only for write access).
Write access to a read process data mapped ADI	0601 0006h	Object mapped to RxPDO, SDO download blocked.
Object Specific Error	0800 0000h	General error

If no corresponding error code can be defined on CANopen, the default error code will be General error (0800 000h).

4.2.4 Object Entries

The exact representation of an ADI depends on its number of elements. In the following example, ADIs no. 0002h and 0004h only contain one element each, causing them to be represented as simple variables rather than arrays. The other ADIs have more than 1 element (of the same data type), causing them to be represented as arrays. If an ADI has more than 1 element, of different data types, it will be represented as a record.

Index	Object Name	Subindex	Description	Type	Access
2001h	ADI 0001h	00h	Number of entries (NNh)	U8	RO
		01h	ADI value(s) (Attribute #5)	-	-
		02h	ADIs with multiple elements (i.e. arrays) are represented as multiple subindexes.		
		...	The data type and access rights of the ADI values are determined by the ADI itself.		
		NNh			
2002h	ADI 0002h	00h	ADI value (Attribute #5)	-	-
2003h	ADI 0003h	00h	Number of entries (NNh)	U8	RO
		01h	ADI value(s) (Attribute #5)	-	-
		02h	ADIs with multiple elements (i.e. arrays) are represented as multiple subindexes.		
		...			
		NNh			
2004h	ADI 0004h	00h	ADI value (Attribute #5)	-	-
2005h	ADI 0005h	00h	Number of entries (NNh)	U8	RO
		01h	ADI value(s) (Attribute #5)	-	-
		02h	ADIs with multiple elements (i.e. arrays) are represented as multiple subindexes.		
		...			
		NNh			
...
5FFFh	ADI 3FFFh	00h	Number of entries (NNh)	U8	RO
		01h	ADI value(s) (Attribute #5)	-	-
		02h	ADIs with multiple elements (i.e. arrays) are represented as multiple subindexes.		
		...			
		NNh			

4.2.5 Modular Device Profile, Object Entries

The objects listed in the table below, shall be implemented if the Modular Device Profile mode is enabled.

Index	Object Name	Sub-index	Description	Type	Access	Notes
6000h - 6FFFh	Input data	Any	ADIs for all modules, except the coupler, that are write process data mappable will be represented in this range.	Any	R, RW	For more information, see ADI to SDO Translation, p. 31 .
7000h - 7FFFh	Output data	Any	ADIs for all modules, except the coupler, that are read process data mappable will be represented in this range.	Any	W, RW	For more information, see ADI to SDO Translation, p. 31 .
9nnnh	Information data	Any	Information objects, one for each module, occupying a slot, except the coupler.	Any	RW	For more information, see Module Identification Objects, p. 31 .
F000h	Modular Device Profile	00h	Number of entries (NNh)	U8	R	Value: 5
		01h	Index distance	U16	R	This value decides how many objects are assigned to each slot. The value is the same for all modules, and thus gives the index distance between two slots. Value: "Number of ADIs per slot", attribute #12 in the Modular Device Object. See Anybus CompactCom 40 Software Design Guide for more information.
		02h	Maximum number of modules	U16	R	Value: "Number of slots", Attribute 11 in the Modular Device Object. See Anybus CompactCom 40 Software Design Guide for more information.
		04h	General Information	U32	R	Value: 0000 0700h (Subindices 9, 10, and 11 are supported in the 9nnnh module identification objects)
		05h	Module PDO group of the device	U16	R	Set to 0 to force the coupler process data to be positioned ahead of the process data. This allows for better integration towards the modular device host object.
F030h	Configured Module Ident List	00h	Number of Entries (Number of slots-1)	U8	R	The master writes the configured module list to these objects, so that the slave can compare the expected module configuration to the actual configuration.
		01h	Module identity of the module configured on position 1 (slot 1).	U32	RW	
		0nh	Module identity of the module configured on position n (slot n).	U32	RW	
F050h	Detected Module Ident List	00h	Number of Entries (Number of slots-1)	U8	R	This object contains information about the

Index	Object Name	Sub-index	Description	Type	Access	Notes
		01h	Module identity of the module configured on position 1 (slot 1).	U32	RW	modules, in the occupied slots, scanned from the application.
				
		0nh	Module identity of the module configured on position n (slot n).	U32	RW	
F600h - F6FFh	Input data area for the coupler	Any	ADIs for the coupler that are write process mappable will be represented in this range.	Any	R, RW	-
F700h - F7FFh	Output data area for the coupler	Any	ADIs for the coupler that are read process mappable will be represented in this range.	Any	W, RW	-

If the Configured Module Ident List (F030h) does not match the Detected Module Ident List (F050h), the module will indicate a mismatch configuration by setting the ALStatusCode register to 0070h. The module will not enter SAFE-OPERATIONAL state.

ADI to SDO Translation

In the Modular Device Profile, all ADIs have to be mapped in numbering order. The number of ADIs mapped per slot is defined in the Modular Device Object, where the same number of objects is assigned to each slot. Depending on whether the ADIs are write or read mappable, they will be mapped to different object ranges. An ADI that is both read and write mappable will be mapped to both ranges. Please note that the SDOs are assigned in number order, but occupy different ranges, depending on type.

The ADIs, that are neither read nor write mappable, will not be mapped to an SDO, resulting in “empty SDOs” as shown in the table below.

Module	ADI	Type	SDO
0 (Coupler)	1	Write mappable	F600h
	2	Read mappable	F701h
	3	Write mappable	F602h
	4	Read mappable	F703h
	5	Not mappable	-
1	6	Read mappable	7000h
	7	Write mappable	6001h
	8	Writable	-
	9	Read only	-
	10	Read mappable	7004h
2	11	Writeable	-
	12	Read only	-
	-	-	-
	14	Write mappable	6008h
	15	Write and Read mappable	6009h and 7009h

Module Identification Objects

The first SDO in the 9nnnh range for each module, shall be predefined according to the table below:

Subindex	Type	Access	Name and Description
00h (0)	U8	R	Highest sub-index supported. Value: 11 (0Bh)
09h (9)	U16	R	Module PDO group. Value: 1. (The PDO group is set to 1 for all modules except the coupler to allow coupler data to be put before module data.)
0Ah (10)	U32	R	Module Identity (Module identity for the module according to the host application.)
0Bh (11)	U16	r	Slot (Module number)

PDO Mapping

The Receive PDO mapping objects and the Transmit PDO mapping objects are configured depending on how the host application is set up. One object in the 16xxh series is created for each module, that holds at least one read mappable ADI. The object numbers will be 1600h + slot number -1. One object in the 1Axxh series is created for each module, that holds at least one write mappable ADI. The object numbers will be 1A00h + slot number -1.

If the coupler holds any write or read mappable ADIs, objects will be created for these. Any objects for the coupler are created after all other mapping objects have been created.

For more information, see [Mapping ADIs on PDOs, p. 25](#).

5 Anybus Module Objects

5.1 General Information

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

Standard Objects:

- [Anybus Object \(01h\), p. 34](#)
- [Diagnostic Object \(02h\), p. 35](#)
- [Network Object \(03h\), p. 37](#)
- [Network Configuration Object \(04h\), p. 39](#)
- [Socket Interface Object \(07h\), p. 44](#)
- [SMTP Client Object \(09h\), p. 61](#)
- File System Interface Object (0Ah), see Anybus CompactCom 40 Software Design Guide
- [Network Ethernet Object \(0Ch\), p. 66](#)

Network Specific Objects:

(none)

5.2 Anybus Object (01h)

Category

Basic

Object Description

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

Supported Commands

Object:	Get_Attribute
Instance:	Get_Attribute
	Set_Attribute
	Get_Enum_String

Object Attributes (Instance #0)

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

Instance Attributes (Instance #1)

Basic

#	Name	Access	Type	Value
1	Module type	Get	UINT16	0403h (Anybus CompactCom 40)
2... 11	-	-	-	Consult the general Anybus CompactCom 40 Software Design Guide for further information.
12	LED colors	Get	struct of: UINT8 (LED1A) UINT8 (LED1B) UINT8 (LED2A) UINT8 (LED2B)	Value: Color: 01h Green 02h Red 01h Green 02h Red
13... 16	-	-	-	Consult the general Anybus CompactCom 40 Software Design Guide for further information.
17	Virtual attributes	Get/Set		
18	Black list/White list	Get/Set		
19	Network time	Get	UINT64	64-bit value expressed in nanoseconds. Base: 12:00 AM, January 1, 2000. The Network time attribute contains the value of the DC system time register of the EtherCAT slave controller.

5.3 Diagnostic Object (02h)

Category

Basic

Object Description

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

An EMCY Object (Emergency Object) is sent on the network each time a diagnostic instance is created or deleted.

Supported Commands

Object:	Get_Attribute
	Create
	Delete
Instance:	Get_Attribute

Object Attributes (Instance #0)

#	Name	Access	Type	Value
1... 4	-	-	-	Consult the general Anybus CompactCom 40 Software Design Guide for further information.
11	Max no. of instances	Get	UINT16	5 + 1 (one instance is reserved for a major unrecoverable event)
12	Supported functionality	Get	BITS32	Bit 0: 0 (The module does not support latching events) Bits 1 - 31: 0

Instance Attributes (Instance #1)

Basic

#	Name	Access	Type	Value
1	Severity	Get	UINT8	See Anybus CompactCom 40 Software Design Guide
2	Event Code	Get	UINT8	
3	NW specific extension	Get	Array of UINT8	CANopen specific EMCY code (2 bytes)
4 -7	(not used)			

When an instance is created (i.e. a diagnostic event is entered), the following actions are performed:

1. A new entry will be created in object entry 1003h (pre-defined error field) in one of two possible ways:

- If the Event Code is 00h — FEh:

High byte	(UINT32)	Low byte
(Not used)	Event Code	00h

- If the Event Code is FFh (network specific):

High byte	(UINT32)	Low byte
(Not used)	Network specific information	

2. The Error Register (object entry 1001h) is set with the corresponding bit information

Bit	Description	Condition for setting bit
0	Generic error	Always set when another error bit in this object is set.
1	Current	Event code is 20h - 23h OR Event code is FFh AND the high byte in NW specific information is 20h - 23h.
2	Voltage	Event code is 30h - 33h OR Event code is FFh AND the high byte in NW specific information is 30h - 33h.
3	Temperature	Event code is 40h - 42h OR Event code is FFh AND the high byte in NW specific information is 40h - 42h.
4	Communication error	Event code is 80h - 82h OR Event code is FFh AND the high byte in NW specific information is 80h - 82h OR Anybus state equals ERROR.
5	Device profile specific	Always 0
6	Reserved	Always 0
7	Manufacturer specific	Event code is FFh AND the high byte in NW specific information is FFh.

3. If the diagnostic instance is created in the state WAIT_PROCESS or higher, an EMCY object is sent to the network with the following information:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
00h	Event Code	Error Register (1001h)	Manufacturer Specific Field (Not used)				

No EMCY object is sent if the instance is created in either of the states SETUP or NW_INIT.

When creating a Major unrecoverable event, this will not end up as an EMCY message on the bus, since this effectively forces the Anybus module to enter the EXCEPTION state.

Bytes 0 and 1 (00h + Event Code) will be replaced by the value of attribute 3 if implemented.

An EMCY object with error code 0000h (“error reset”) is sent when a diagnostic instance is deleted.

5.4 Network Object (03h)

Category

Basic

Object Description

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

Supported Commands

Object:	Get_Attribute
Instance:	Get_Attribute
	Set_Attribute
	Get_Enum_String
	Map_ADI_Write_Area
	Map_ADI_Read_Area
	Map_ADI_Write_Ext_Area
	Map_ADI_Read_Ext_Area

Object Attributes (Instance #0)

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

Instance Attributes (Instance #1)

Basic

#	Name	Access	Type	Value
1	Network type	Get	UINT16	0087h
2	Network type string	Get	Array of CHAR	'EtherCAT'
3	Data format	Get	ENUM	00h (LSB first)
4	Parameter data support	Get	BOOL	True
5	Write process data size	Get	UINT16	Current write process data size (in bytes). Updated on every successful Map_ADI_Write_Area, Map_ADI_Write_Ext_Area and Remap_ADI_Write_Area. Consult the general Anybus CompactCom 40 Software Design Guide for further information.
6	Read process data size	Get	UINT16	Current read process data size (in bytes). Updated on every successful Map_ADI_Read_Area, Map_ADI_Read_Ext_Area and Remap_ADI_Read_Area. Consult the general Anybus CompactCom 40 Software Design Guide for further information.
7	Exception Information	Get	UINT8	Additional information may be provided here when the module has entered the EXCEPTION state, see exception information in table below. Consult the general Anybus CompactCom 40 Software Design Guide for further information.
8... 10	(reserved)	-	-	

Exception Information

Value	Description
00h	No additional information available.
01h	(reserved)
02h	
03h	
04h	
05h	
06h	The implementation of the Assembly Mapping Host Object is incorrect, e.g. the attribute 11 or 12 is not supported.
07h	The application supports the Remap ADI commands, but returned an error response when requesting object attributes 11 or 12 of the Application Data Object ("No. of read process data mappable instances" or "No of write process data mappable instances") or when issuing the Get_Instance_Numbers command towards the Application Data Object.
08h	The implementation of the Modular Device Object in the host application is not correct, e.g. an error response is received on the Get_List command.

5.5 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.

Supported Commands

Object:	Get_Attribute
	Reset
Instance:	Get_Attribute
	Set_Attribute

Object Attributes (Instance #0)

#	Name	Access	Type	Value
1	Name	Get	Array of CHAR	"Network configuration"
2	Revision	Get	UINT8	01h
3	Number of instances	Get	UINT16	12
4	Highest instance no.	Get	UINT16	15

Instance Attributes (Instance #1, Device ID)

Extended

See also [Device ID, p. 20](#).

Changes have immediate effect.

#	Name	Access	Type	Value
1	Name	Get	Array of CHAR	"Device ID" Multilingual, see Multilingual Strings, p. 43
2	Data type	Get	UINT8	05h (= UINT16)
3	Number of elements	Get	UINT8	01h (one element)
4	Descriptor	Get	UINT8	03h (read/write access)
5	Value	Get/Set	UINT16	1...65535: Valid network address 0: Device not configured (Default)
6	Configured Value	Get	UINT16	Configured value for Device ID. The value always equals the value of attribute #5.

Instance Attributes (Instance #3, IP Address)

Value is used after module reset.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"IP address" (Multilingual, see Multilingual Strings, p. 43)
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)



This attribute should not be set by the application at every power on, as this would cause certification problems.

Instance Attributes (Instance #4, Subnet Mask)

Value is used after module reset.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"Subnet mask" (Multilingual, see Multilingual Strings, p. 43)
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)



This attribute should not be set by the application at every power on, as this would cause certification problems.

Instance Attributes (Instance #5, Gateway)

Value is used after module reset.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"Gateway" (Multilingual, see Multilingual Strings, p. 43)
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)



This attribute should not be set by the application at every power on, as this would cause certification problems.

Instance Attributes (Instance #6, DHCP)

Value is used after module reset.

#	Name	Access	Data Type	Description									
1	Name	Get	Array of CHAR	"DHCP" (Multilingual, see Multilingual Strings, p. 43)									
2	Data type	Get	UINT8	08h (= ENUM)									
3	Number of elements	Get	UINT8	01h (one element)									
4	Descriptor	Get	UINT8	07h (read/write/shared access)									
5	Value	Get/Set	ENUM	<table border="1"> <thead> <tr> <th>Value</th> <th>String</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>00h</td> <td>"Disable"</td> <td>DHCP disabled (default)</td> </tr> <tr> <td>01h</td> <td>"Enable"</td> <td>DHCP enabled (Multilingual, see Multilingual Strings, p. 43)</td> </tr> </tbody> </table>	Value	String	Meaning	00h	"Disable"	DHCP disabled (default)	01h	"Enable"	DHCP enabled (Multilingual, see Multilingual Strings, p. 43)
Value	String	Meaning											
00h	"Disable"	DHCP disabled (default)											
01h	"Enable"	DHCP enabled (Multilingual, see Multilingual Strings, p. 43)											

Instance Attributes (Instance #9, DNS1)

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

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"DNS1" (Multilingual, see Multilingual Strings, p. 43)
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)

Instance Attributes (Instance #10, DNS2)

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

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"DNS2" (Multilingual, see Multilingual Strings, p. 43)
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)

Instance Attributes (Instance #11, Host name)

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

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"Host name" (Multilingual, see Multilingual Strings, p. 43)
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	40h (64 elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of CHAR	Host name, 64 characters (pad with space to full length)

Instance Attributes (Instance #12, Domain name)

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

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"Host name" (Multilingual, see Multilingual Strings, p. 43)
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	30h (48 elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of CHAR	Domain name, 48 characters (pad with space to full length)

Instance Attributes (Instance #13, SMTP Server)

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

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"SMTP server" (Multilingual, see Multilingual Strings, p. 43)
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	40h (64 elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of CHAR	SMTP server address, 64 characters. Dotted decimal format or server name(pad with space to full length)

Instance Attributes (Instance #14, SMTP User)

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

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"SMTP user" (Multilingual, see Multilingual Strings, p. 43)
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	40h (64 elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of CHAR	SMTP account user name, 64 characters (pad with space to full length)

Instance Attributes (Instance #15, SMTP Password)

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

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"SMTP Pswd" (Multilingual, see Multilingual Strings, p. 43)
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	40h (64 elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of CHAR	Host name, 64 characters (pad with space to full length)

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
1	Device ID	Geräteadresse	ID Dispos.	ID Dispos.	ID appareil

Reset

When a factory default (reset) command is issued to this object, the configured Device ID will be set to 0 (default value).

5.6 Socket Interface Object (07h)

Category

Extended

Object Description

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

Note that some of the commands used when accessing this object may require segmentation. A message will be segmented if the amount of data sent or received is larger than the message channel can handle. For more information, see [Message Segmentation, p. 59](#).



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

Supported Commands

Object:	Get_Attribute Create (See below) Delete (See below)
Instance:	Get_Attribute Set_Attribute Bind (See below) Shutdown (See below) Listen (See below) Accept (See below) Connect (See below) Receive (See below) Receive_From (See below) Send (See below) Send_To (See below) IP_Add_membership (See below) IP_Drop_membership (See below) DNS_Lookup (See below)

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1	Name	Get	Array of CHAR	"Socket interface"
2	Revision	Get	UINT8	01h
3	Number of instances	Get	UINT16	Number of opened sockets
4	Highest instance no.	Get	UINT16	Highest created instance number
11	Max. no. of instances	Get	UINT16	0014h (20 instances)

Instance Attributes (Sockets #1...20)

Extended

#	Name	Access	Data Type	Description
1	Socket Type	Get	UINT8	<p><u>Value:</u> <u>Socket Type</u></p> <p>00h SOCK_STREAM, NONBLOCKING (TCP)</p> <p>01h SOCK_STREAM, BLOCKING (TCP)</p> <p>02h SOCK_DGRAM, NONBLOCKING (UDP)</p> <p>03h SOCK_DGRAM, BLOCKING (UDP)</p>
2	Port	Get	UINT16	Local port that the socket is bound to
3	Host IP	Get	UINT32	Host IP address, or 0 (zero) if not connected
4	Host port	Get	UINT16	Host port number, or 0 (zero) if not connected
5	TCP State	Get	UINT8	<p>State (TCP sockets only):</p> <p><u>Value</u> <u>State/Description</u></p> <p>00h CLOSED Closed</p> <p>01h LISTEN Listening for connection</p> <p>02h SYN_SENT Active, have sent and received SYN</p> <p>03h SYN_RECEIVED Have sent and received SYN</p> <p>04h ESTABLISHED Established.</p> <p>05h CLOSE_WAIT Received FIN, waiting for close</p> <p>06h FIN_WAIT_1 Have closed, sent FIN</p> <p>07h CLOSING Closed exchanged FIN; await FIN ACK</p> <p>08h LAST_ACK Have FIN and close; await FIN ACK</p> <p>09h FIN_WAIT_2 Have closed, FIN is acknowledged</p> <p>Ah TIME_WAIT Quiet wait after close</p>
6	TCP RX bytes	Get	UINT16	Number of bytes in RX buffers (TCP sockets only)
7	TCP TX bytes	Get	UINT16	Number of bytes in TX buffers (TCP sockets only)
8	Reuse address	Get/Set	BOOL	<p>Socket can reuse local address</p> <p><u>Value</u> <u>Meaning</u></p> <p>1 Enabled</p> <p>0 Disabled (default)</p>
9	Keep alive	Get/Set	BOOL	<p>Protocol probes idle connection (TCP sockets only).</p> <p>If the Keep alive attribute is set, the connection will be probed for the first time after it has been idle for 120 minutes. If a probe attempt fails, the connection will continue to be probed at intervals of 75s. The connection is terminated after 8 failed probe attempts.</p> <p><u>Value</u> <u>Meaning</u></p> <p>1 Enabled</p> <p>0 Disabled (default)</p>
10	IP Multicast TTL	Get/Set	UINT8	IP Multicast TTL value (UDP sockets only). Default = 1.
11	IP Multicast Loop	Get/Set	BOOL	<p>IP multicast loop back (UDP sockets only)</p> <p>Must belong to group in order to get the loop backed message</p> <p><u>Value</u> <u>Meaning</u></p> <p>1 Enabled (default)</p> <p>0 Disabled</p>

#	Name	Access	Data Type	Description						
12	Ack delay time	Get/Set	UINT16	Time for delayed ACKs in ms (TCP sockets only) Default = 200ms. Resolution is 50ms, i.e. 50...99 = 50ms, 100...149 = 100ms, 199 = 150ms etc.						
13	TCP No Delay	Get/Set	BOOL	Don't delay send to coalesce packets (TCP). <table border="1"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Delay (default)</td> </tr> <tr> <td>0</td> <td>Don't delay (turn off Nagle's algorithm on socket)</td> </tr> </tbody> </table>	Value	Meaning	1	Delay (default)	0	Don't delay (turn off Nagle's algorithm on socket)
Value	Meaning									
1	Delay (default)									
0	Don't delay (turn off Nagle's algorithm on socket)									
14	TCP Connect Timeout	Get/Set	UINT16	TCP Connect timeout in seconds (default = 75s)						

Command Details: Create

Category

Extended

Details

Command Code	03h
Valid for:	Object Instance

Description

This command creates a socket.

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

- Command Details

Field	Contents										
CmdExt[0]	(reserved, set to zero)										
CmdExt[1]	<table border="1"> <thead> <tr> <th>Value:</th> <th>Socket Type:</th> </tr> </thead> <tbody> <tr> <td>00h</td> <td>SOCK_STREAM, NON-BLOCKING (TCP)</td> </tr> <tr> <td>01h</td> <td>SOCK_STREAM, BLOCKING (TCP)</td> </tr> <tr> <td>02h</td> <td>SOCK_DGRAM, NON-BLOCKING (UDP)</td> </tr> <tr> <td>03h</td> <td>SOCK_DGRAM, BLOCKING (UDP)</td> </tr> </tbody> </table>	Value:	Socket Type:	00h	SOCK_STREAM, NON-BLOCKING (TCP)	01h	SOCK_STREAM, BLOCKING (TCP)	02h	SOCK_DGRAM, NON-BLOCKING (UDP)	03h	SOCK_DGRAM, BLOCKING (UDP)
Value:	Socket Type:										
00h	SOCK_STREAM, NON-BLOCKING (TCP)										
01h	SOCK_STREAM, BLOCKING (TCP)										
02h	SOCK_DGRAM, NON-BLOCKING (UDP)										
03h	SOCK_DGRAM, BLOCKING (UDP)										

- Response Details

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

Command Details: Delete

Category

Extended

Details

Command Code	04h
Valid for:	Object Instance

Description

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

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

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

- Response Details
(no data)

Command Details: Bind**Category**

Extended

Details

Command Code 10h
Valid for: Instance

Description

This command binds a socket to a local port.

- Command Details

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

- Response Details

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

Command Details: Shutdown**Category**

Extended

Details

Command Code 11h
Valid for: Instance

Description

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

- Command Details

Field	Contents		
CmdExt[0]	(reserved, set to zero)		
CmdExt[1]	<table border="0"> <tr> <td style="vertical-align: top;"><u>Value:</u> 00h 01h 02h</td> <td style="vertical-align: top;"><u>Mode:</u> Shutdown receive channel Shutdown send channel Shutdown both receive- and send channel</td> </tr> </table>	<u>Value:</u> 00h 01h 02h	<u>Mode:</u> Shutdown receive channel Shutdown send channel Shutdown both receive- and send channel
<u>Value:</u> 00h 01h 02h	<u>Mode:</u> Shutdown receive channel Shutdown send channel Shutdown both receive- and send channel		

- Response Details

(no data)

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

Application initiates shutdown:

1. Send shutdown with CmdExt[1] set to 01h. This will send FIN-flag to host shutting down the send channel, note that the receive channel will still be operational.
2. Receive data on socket until error message Object specific error (EPIPE (13)) is received, indicating that the host closed the receive channel. If host does not close the receive channel use a timeout and progress to step 3.
3. Delete the socket instance. If step 2 timed out, RST-flag will be sent to terminate the socket.

Application initiates shutdown:

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

Command Details: Listen

Category

Extended

Details

Command Code 12h
Valid for: Instance

Description

This command puts a TCP socket in listening state.

- Command Details

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

- Response Details
(no data)

Command Details: Accept

Category

Extended

Details

Command Code	13h
Valid for:	Instance

Description

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

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

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

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

- Command Details
(no data)
- Response Details

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

Command Details: Connect

Category

Extended

Details

Command Code	14h
Valid for:	Instance

Description

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

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

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

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

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

- Command Details

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

- Response Details
(no data)

Command Details: Receive

Category

Extended

Details

Command Code	15h
Valid for:	Instance

Description

This command receives data from a connected socket. Message segmentation may be used to receive up to 1472 bytes (for more information, see [Message Segmentation, p. 59](#)).

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

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

NON-BLOCKING mode:	If no data is available on the socket, the error code 0006h (EWOULDBLOCK) will be returned.
BLOCKING mode:	The module will not issue a response until the operation has finished.

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

- Command Details

Field	Contents	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control bits	For more information, see Message Segmentation, p. 59
Data[0]	Receive data size (low)	Only used in the first segment
Data[1]	Receive data size (high)	

- Response Details

The data in the response may be segmented (For more information, see [Message Segmentation, p. 59](#)).

Field	Contents	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control bits	For more information, see Message Segmentation, p. 59
Data[0...n]	Received data	-

Command Details: Receive_From

Category

Extended

Details

Command Code	16h
Valid for:	Instance

Description

This command receives data from an unconnected SOCK_DGRAM-socket. Message segmentation may be used to receive up to 1472 bytes (For more information, see [Message Segmentation, p. 59](#)).

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

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

NON-BLOCKING mode:	If no data is available on the socket, the error code 0006h (EWOULDBLOCK) will be returned.
BLOCKING mode:	The module will not issue a response until the operation has finished.

- Command Details

Field	Contents	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control bits	For more information, see Message Segmentation, p. 59
Data[0]	Receive data size (low byte)	Only used in the first segment
Data[1]	Receive data size (high byte)	

- Response Details

The data in the response may be segmented (For more information, see [Message Segmentation, p. 59](#)).

Field	Contents	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control bits	For more information, see Message Segmentation, p. 59
Data[0]	Host IP address byte 4	The host address/port information is only included in the first segment. All data thereafter will start at Data[0]
Data[1]	Host IP address byte 3	
Data[2]	Host IP address byte 2	
Data[3]	Host IP address byte 1	
Data[4]	Host port number (low byte)	
Data[5]	Host port number (high byte)	
Data[6...n]	Received data	

Command Details: Send**Category**

Extended

Details

Command Code	17h
Valid for:	Instance

Description

This command sends data on a connected socket. Message segmentation may be used to send up to 1472 bytes (For more information, see [Message Segmentation, p. 59](#)).

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

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

- Command Details

To allow larger amount of data (i.e. >255 bytes) to be sent, the command data may be segmented (For more information, see [Message Segmentation, p. 59](#)).

Field	Contents	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control	(For more information, see Message Segmentation, p. 59)
Data[0...n]	Data to send	-

- Response Details

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

Command Details: Send_To

Category

Extended

Details

Command Code	18h
Valid for:	Instance

Description

This command sends data to a specified host on an unconnected SOCK-DGRAM-socket. Message segmentation may be used to send up to 1472 bytes (For more information, see appendix For more information, see [Message Segmentation, p. 59](#)).

- Command Details

To allow larger amount of data (i.e. >255 bytes) to be sent, the command data may be segmented (For more information, see [Message Segmentation, p. 59](#)).

Field	Contents	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control	For more information, see Message Segmentation, p. 59
Data[0]	Host IP address byte 4	The host address/port information shall only be included in the first segment. All data thereafter must start at Data[0]
Data[1]	Host IP address byte 3	
Data[2]	Host IP address byte 2	
Data[3]	Host IP address byte 1	
Data[4]	Host port number (low byte)	
Data[5]	Host port number (high byte)	
Data[6...n]	Data to send	

- Response Details

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

Command Details: IP_Add_Membership

Category

Extended

Details

Command Code 19h
Valid for: Instance

Description

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

- Command Details

Field	Contents
CmdExt[0]	(reserved, set to zero)
CmdExt[1]	
Data[0]	Group IP address byte 4
Data[1]	Group IP address byte 3
Data[2]	Group IP address byte 2
Data[3]	Group IP address byte 1

- Response Details
(no data)

Command Details: IP_Drop_Membership

Category

Extended

Details

Command Code 1Ah
Valid for: Instance

Description

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

- Command Details

Field	Contents
CmdExt[0]	(reserved, set to zero)
CmdExt[1]	
Data[0]	Group IP address byte 4
Data[1]	Group IP address byte 3
Data[2]	Group IP address byte 2
Data[3]	Group IP address byte 1

- Response Details
(no data)

Command Details: DNS_Loopup

Category

Extended

Details

Command Code 1Bh
Valid for: Instance

Description

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

- Command Details

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

- Response Details (Success)

Field	Contents	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]		
Data[0]	IP address byte 4	IP address of the specified host
Data[1]	IP address byte 3	
Data[2]	IP address byte 2	
Data[3]	IP address byte 1	

Socket Interface Error Codes (Object Specific)

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

Error Code	Name	Meaning
1	ENOBUFS	No internal buffers available
2	ETIMEDOUT	A timeout event occurred
3	EISCONN	Socket already connected
4	EOPNOTSUPP	Service not supported
5	ECONNABORTED	Connection was aborted
6	EWOULDBLOCK	Socket cannot block because unblocking socket type
7	ECONNREFUSED	Connection refused
8	ECONNRESET	Connection reset
9	ENOTCONN	Socket is not connected
10	EALREADY	Socket is already in requested mode
11	EINVAL	Invalid service data
12	EMSGSIZE	Invalid message size
13	EPIPE	Error in pipe
14	EDESTADDRREQ	Destination address required
15	ESHUTDOWN	Socket has already been shutdown
16	(reserved)	-
17	EHAVEEOB	Out of band data available
18	ENOMEM	No internal memory available

Error Code	Name	Meaning
19	EADDRNOTAVAIL	Address is not available
20	EADDRINUSE	Address already in use
21	(reserved)	-
22	EINPROGRESS	Service already in progress
28	ETOOMANYREFS	Too many references
101	Command aborted	If a command is blocking on a socket, and that socket is closed using the Delete command, this error code will be returned to the blocking command.
102	DNS name error	Failed to resolve the host name (name error response from DNS server).
103	DNS timeout	Timeout when performing a DNS lookup.
104	DNS command failed	Other DNS error.

Message Segmentation

General

Category: Extended

The maximum message size supported by the Anybus CompactCom 40 is normally 1524 bytes. In some applications a maximum message size of 255 bytes is supported, e.g. if an Anybus CompactCom 40 is to replace an Anybus CompactCom 30 without any changes to the application. The maximum socket message size is 1472. To ensure support for socket interface messages larger than 255 bytes a segmentation protocol is used.



The segmentation bits have to be set for all socket interface messages, in the commands where segmentation can be used, whether the messages have to be segmented or not.

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

The module supports 1 (one) segmented message per instance

Command Segmentation

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

Command segmentation is used for the following commands (Socket Interface Object specific commands):

- Send
- Send To

When issuing a segmented command, the following rules apply:

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

Segmentation Control Bits (Command)

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

Segmentation Control Bits (Response)

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

Response Segmentation

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

Response segmentation is used for responses to the following commands (Socket Interface Object specific commands):

- Receive
- Receive From

When receiving a segmented response, the following rules apply:

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

Segmentation Control bits (Command)

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

Segmentation Control bits (Response)

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

5.7 SMTP Client Object (09h)

Category

Extended

Object Description

This object groups functions related to the SMTP client.

Supported Commands

Object:	Get_Attribute
	Create
	Delete
	Send e-mail from file (see below)
Instance:	Get_Attribute
	Set_Attribute
	Send e-mail (see below)

Object Attributes (Instance #0)

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

Instance Attributes (Instance #1)

Instances are created dynamically by the application.

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

Command Details: Create

Category

Extended

Details

Command Code 03h

Valid for: Object

Description

This command creates an e-mail instance.

- Command Details

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

- Response Details

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

Command Details: Delete

Category

Extended

Details

Command Code 04h

Valid for: Object

Description

This command deletes an e-mail instance.

- Command Details

Field	Contents	Comments
CmdExt[0]	E-mail instance number	low byte
CmdExt[1]		high byte

- Response Details
(no data)

Command Details: Send E-mail From File

Category

Extended

Details

Command Code 11h
Valid for: Object

Description

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

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

```
[To]
recipient

[From]
sender

[Subject]
email subject

Se [Headers]
extra headers, optional

[Message]
actual email message
```

- **Command Details**

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

- **Response Details**
(no data)

Command Details: Send E-mail

Category

Extended

Details

Command Code	10h
Valid for:	Object

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)

5.8 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)

#	Name	Access	Data Type	Description
1	MAC Address	Get	Array of UINT8	Current MAC address. See also "Ethernet Host Object (F9h)"
2	Port 1 MAC Address	Get	Array of UINT8	MAC address for port 1 (mandatory for the LLDP protocol) See also "Ethernet Host Object (F9h)"
3	Port 2 MAC Address	Get	Array of UINT8	MAC address for port 2 (mandatory for the LLDP protocol) See also "Ethernet Host Object (F9h)"

6 Host Application Objects

6.1 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 EtherCAT implementation.

Standard Objects:

- Application Object (see Anybus CompactCom 40 Software Design Guide)
- Application File System Interface Object (see Anybus CompactCom 40 Software Design Guide)
- [Assembly Mapping Object \(EBh\), p. 68](#)
- [Sync Object \(EEh\), p. 69](#)
- Modular Device Object (see Anybus CompactCom 40 Software Design Guide)
- Application Data Object (see Anybus CompactCom 40 Software Design Guide)
- [Ethernet Host Object \(F9h\), p. 77](#)

Network Specific Objects:

- [EtherCAT Object \(F5h\), p. 72](#)

6.2 Assembly Mapping Object (EBh)

Category

Extended

Object Description

If the application has implemented this object, the object will replace the PDO mapping created when the application is started. The original mapping will be replaced during the transition from PRE-OPERATIONAL state to SAFE-OPERATIONAL state. The application must support the Remap_ADI commands, if this object is to be implemented.

Each instance in the Assembly Mapping Object corresponds to one PDO. The first read assembly is mapped to object 1600h in the object dictionary, the second to 1601h and so on. Similarly, the first write assembly is mapped to object 1A00h, the second to 1A01h and so on. Up to 64 each of read and write assembly instances are supported.

The table below illustrates an example on how PDO mapping object numbers are assigned for different assembly mapping object instances.

Instance no.	Direction	PDO mapping object number
1	Write	1A00h
2	Read	1600h
3	Write	1A01h
4	Read	1601h
5	Read	1602h

Each assembly mapping instances supports up to 254 ADI elements, corresponding to one full PDO on EtherCAT.

If the Modular Device Object is implemented in the host application, i.e. modular device profile is enabled, the settings of this objects will be ignored.

See also ..

- Anybus CompactCom 40 Software Design Guide, "Assembly Mapping Object"
[Standard Objects, p. 22](#) for assembly to PDO mapping.

6.3 Sync Object (EEh)

Category

Extended

Object Description

This object implements the host application SYNC settings.

The implementation of this object is optional; if it is not implemented, the module only supports the EtherCAT Free Run mode.

If there is any problem with the configuration of the sync functionality as a whole, the application must indicate this in the application status register. The module will then change EtherCAT states to SafeOp and indicate the problem in the ALStatusCode register, see [Application Status Register, p. 70](#)

See also ...

- Anybus CompactCom 40 Software Design Guide, “Sync”
- Anybus CompactCom 40 Software Design Guide, “Sync Object”

Supported Commands

Object:	Get_Attribute
Instance:	Get_Attribute Set_Attribute

Object Attributes (Instance #0)

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

Instance Attributes (Instance #1)

Extended

The attributes are represented on EtherCAT as follows:

#	Name	Access	Type	Default Value	Comment
1	Cycle time	Get/Set	UINT32		Application cycle time in nanoseconds. Replaces the setting in object entry 1C32h, subindex 2. (SM Output Parameter, Cycle time)
2	Output valid	Get/Set	UINT32	0	Output valid point relative to SYNC events, in nanoseconds. Replaces the setting in object entry 1C32h, subindex 3. (SM Output Parameter, Shift time)
3	Input capture	Get/Set	UINT32	0	Input capture point relative to SYNC events, in nanoseconds. Replaces the setting in object entry 1C33, subindex 3. (SM Input Parameter, Shift time)
4	Output processing	Get	UINT32		Minimum required time, in nanoseconds, between RDPDI interrupt and valid output.

#	Name	Access	Type	Default Value	Comment
					Specifies the value of object entry 1C32h, subindex 6. (SM Output Parameter, Output Calc and Copy Time) The Anybus latency is added to this value before it is presented on EtherCAT.
5	Input processing	Get	UINT32		Maximum required time, in nanoseconds, from "Input capture" until write process data has been completely written to the Anybus CompactCom module. Specifies the value of object entry 1C33h, subindex 6. (SM Input Parameter, Input Calc and Copy Time) The Anybus latency is added to this value before it is presented on EtherCAT.
6	Min cycle time	Get	UINT32		Minimum cycle time supported by the application. Specifies the values of object entries 1C32h and 1C33h, subindex 5. (SM Output and SM Input Parameters, Minimum cycle time)
7	Sync mode	Get/Set	UINT16		Selection of synchronization mode. The attribute enumerates the bits in attribute 8. 0: Free Run (no sync) 1: Synced with DC Specifies the values of object entries 1C32h and 1C33h, subindex 1. (SM Output and SM Input Parameters, Synchronization type).
8	Supported sync modes	Get	UINT16		A list of the synchronization modes the application supports. Each bit corresponds to a mode in attribute 7. Bit 0: 1 = Free run supported Bit 1: 1 = DC supported Specifies the values of object entries 1C32h and 1C33h, subindex 4. (SM Output and Input Parameters, Synchronization types supported)

Application Status Register

If the application sets an error status to the application status register (H_APPSTATUS), the module sets the EtherCAT state to SafeOp. The H_APPSTATUS is translated to the ALStatus-Code register as shown in the table below.

H_APPSTATUS	Error	ALStatusCode: ALSTATUSCODE_XXX (#)	Comment
0000h	No error	-	Application can operate in state PROCESS_ACTIVE
0001h	Not yet synchronized	NOSYNCERROR (002Dh)	Application is not synchronized to the SYNC signal and not ready to go to PROCESS_ACTIVE.
0002h	Sync config error	INVALIDSYNCCFG (0030h)	A problem with the configuration of the Sync host object prevents the application from going to PROCESS_ACTIVE.
0003h	Read process data configuration error	INVALIDOUTPUTMAPPING (0025h)	A problem with the current read process data mapping is prevents the application from going to PROCESS_ACTIVE.
0004h	Write process data configuration error	INVALIDINPUTMAPPING (0024h)	A problem with the current write process data mapping

H_APPSTATUS	Error	ALStatusCode: ALSTATUSCODE_XXX (#)	Comment
			is prevents the application from going to PROCESS_ACTIVE.
0005h	Synchronization loss	FATALSYNCERROR (002Ch)	Application has lost the lock to the synchronization. If the module is in state PROCESS_ACTIVE, it will go to ERROR.
0006h	Excessive data loss	NOVALIDINPUTSANDOUTPUTS (002Bh)	The application has detected a significant loss of process data frames from the network. If the module is in state PROCESS_ACTIVE, it will go to ERROR.
0007h	Output error	DCSYNCIOERROR (0033h)	A problem in the application prevents it from acting on outputs. If the module is in state PROCESS_ACTIVE, it will go to ERROR.

6.4 EtherCAT Object (F5h)

Category

Basic, extended

Object Description

This object implements EtherCAT specific 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 startup; 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]").



Support for this object is optional. If implemented, it is highly recommended to support all attributes in the range 1... 6.

To pass conformance tests, the end product has to have a Vendor ID valid for the end product vendor.

See also...

- Anybus CompactCom 40 Software Design Guide, "Error Codes"

Supported Commands

Object: Get_Attribute

Instance: Get_Attribute

Object Attributes (Instance #0)

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

Instance Attributes (Instance #1)

Basic

#	Name	Access	Type	Default Value	Comment
1	Vendor ID	Get	UINT32	0000 001Bh	These values replace the settings in object entry 1018h. (Identity Object)

Extended

#	Name	Access	Type	Default Value	Comment
2	Product Code	Get	UINT32	0000 0034h	These values replace the settings in object entry 1018h. (Identity Object)
3	Major revision	Get	UINT16	Major revision	
4	Minor revision	Get	UINT16	Minor revision	
5	Serial Number	Get	UINT32	Unique number, assigned at production	
6	Manufacturer Device Name	Get	Array of CHAR (Max. 64 bytes)	Product specific	Replaces object entry 1008h (Manufacturer Device Name)
7	Manufacturer Hardware Version	Get	Array of CHAR (Max. 64 bytes)	X.YY (major version.minor version)	Specifies the value of object entry 1009h (Manufacturer Hardware Version)
8	Manufacturer Software Version	Get	Array of CHAR (Max. 64 bytes)	X.YY.ZZ (major version.minor version. build)	Specifies the value of object entry 100Ah (Manufacturer Software Version)
9	ENUM ADIs	Get	Array of UINT16	-	By default, ENUMs will be translated to UNSIGNED8 on EtherCAT. By implementing this attribute, ENUMs will be translated to ENUMs on the bus as well. The attributes shall contain a sorted list of ADI instance numbers which are of type ENUM. If this attribute is implemented, also implement the optional Application Data Instance attribute #6 ('Max. Value') of all ENUM ADIs, since this improves performance and functionality of ENUMs on the bus significantly.
10	Device Type	Get	UINT32	Product specific	If implemented, this value replaces the default value for object entry 1000h (Device type).
11	Write PD assembly instance translation	Get	Array of UINT16	Empty	This attribute can be used by the application to change the default TxPDO mapping object of Write PD instances in the Assembly Mapping Object. It corresponds to attribute 11 in the Assembly Mapping Object, "Write PD Instance List". Each index in the array contains the TxPDO mapping object number that is used for the instance on the same index in the "Write PD Instance List" attribute. Valid values: 1A00h - 1BFFh.
12	Read PD assembly instance translation	Get	Array of UINT16	Empty	This attribute can be used by the application to change the default RxPDO mapping object of Read PD instances in the Assembly Mapping Object. It corresponds to attribute 12 in the Assembly Mapping Object, "Read PD Instance List". Each index in the array contains the RxPDO mapping object number

#	Name	Access	Type	Default Value	Comment
					that is used for the instance on the same index in the "Read PD Instance List" attribute. Valid values: 1600h - 17FFh.
13	ADI translation	Get	Array of Struct { UINT16 UINT16 }	Empty	This attribute can be used by the application to implement objects in the communication profile specific CoE object area (1000h - 1FFFh). Objects already implemented in the module cannot be replaced by ADIs. The attribute is implemented as an array of packed structs of two UINT16. The first UINT16 contains the ADI instance number, the second contains the object index that the ADI shall correspond to. See ADI Translation, Example, p. 76
14	(Reserved)	-	-	-	(Reserved for future use)
15	Object subindex translation	Get	Array of Struct { UINT16 UINT16 UINT8 }	Empty	This attribute can be used by the application to implement subindices of objects in the profile specific CoE object area (0x1000-0x1FFF). Subindices already implemented in the module cannot be replaced by ADIs and this attribute can only be used to add subindices to objects explicitly defined in the module to be extendable. The attribute is implemented as an array of packed Structs of two UINT16 and one UINT8. The first UINT16 contains the ADI instance number, the second contains the object index that the ADI shall correspond to. The UINT8 contains the subindex of the latter object that the ADI shall correspond to. An object dictionary index/subindex entry may only be translated to an ADI of type VAR. Translating the entry to an ADI of type ARRAY or RECORD is not supported. See: Object Subindex Translation, Example, p. 76
16	Enable FoE	Get	BOOL	TRUE (=1)	This attribute enables/disables functionality related to File access over EtherCAT. If FoE is disabled it is not possible to upgrade firmware via EtherCAT or access the Application File System Interface Object (EAh) via EtherCAT.
17	Enable EoE	Get	BOOL	TRUE (=1)	Enables/Disables functionality related to Ethernet over EtherCAT. If EoE is disabled the module will not accept any mailbox requests of EoE type and no IT functionality in the module will be usable.

#	Name	Access	Type	Default Value	Comment
18	Change shift register switch functionality	Get	BOOL	FALSE (=0)	Normally when running shift register operation mode, switch 1 is used for the last octet of the IP address and switch 2 is used for the Device ID. If this attribute is set to TRUE this behavior is changed so switch 1 is used for Device ID and switch 2 is used for the last octet of the IP address.
19	Set Device ID as Configured station alias	Get		FALSE (=0)	Normally the Configured station alias value can only be set from the EtherCAT configuration tool. This is the case when this attribute is either false or not implemented. If this attribute is set to True, the value set to instance 1 of the network configuration object (Device ID) will be set to the configured station alias as well (both the register 0x0012 and the EEPROM).

ADI Translation, Example

The host application wants to implement the diagnostic object (10F3h) and the timestamp object (10F8h). To do this it needs to create two ADIs that match the CoE implementation of these objects, e.g. ADI F0F3h for the diagnostic object and F0F8 for the timestamp object. It then needs to implement the following data for the ADI translation attribute:

Example 2:

```
[
  {
    F0F3h
    10F3h
  }
  {
    F0F8h
    10F8h
  }
]
```

SDO requests towards these CoE objects will then be forwarded to the corresponding ADI. If a CoE object present in this attribute is implemented by the module, the module will handle all requests to that object by itself, and nothing is forwarded to the host application.

Object Subindex Translation, Example

The host application wants to implement the Sync Error subindex (subindex 32) of the 0x1C32 and 0x1C33 objects. To do this it needs to create two ADIs that match the CoE implementation of these entries. Let's say it creates ADI 0xF0FD for entry 0x1C32:32 and ADI 0xF0FE for entry 0x1C33:32. It then needs to implement the following data for the "Object subindex translation" attribute:

Example 3:

```
[
  {
    0xF0FD
    0x1C32
    32
  }
  {
    0xF0FE
    0x1C33
    32
  }
]
```

SDO requests towards these CoE object/subindex entries will then be forwarded to the corresponding ADI.

If a CoE entry present in this attribute is implemented by the module, the module will handle all requests to that entry by itself, as it will if the object does not support being extended with more subindices, and nothing is forwarded to the host application.

6.5 Ethernet Host Object (F9h)

Object Description

This object implements Ethernet features in the host application.

Supported Commands

Object:	Get_Attribute
Instance:	Get_Attribute Set_Attribute

Object Attributes (Instance #0)

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

Instance Attributes (Instance #1)

- If an attribute is not implemented, the default value will be used.
- The module is preprogrammed with a valid MAC address. To use that address, do not implement attribute #1.
- Do not implement attributes #9 and #10, only used for PROFINET devices, if the module shall use the preprogrammed MAC addresses.
- If new MAC addresses are assigned to a PROFINET device, these addresses (in attributes #1, #9, and #10) have to be consecutive, e.g. (xx:yy:zz:aa:bb:01), (xx:yy:zz:aa:bb:02), and (xx:yy:zz:aa:bb:03) with the first five octets not changing.

#	Name	Access	Data Type	Default Value	Comment
1	MAC address	Get	Array of UINT8	-	6 byte physical address value; overrides the preprogrammed Mac address. Note that the new Mac address value must be obtained from the IEEE. Do not implement this attribute if the preprogrammed Mac address is to be used.
2	Enable HICP	Get	BOOL	True (Enabled)	Enable/Disable HICP
3	Enable Web Server	Get	BOOL	True (Enabled)	Enable/Disable Web Server
4	(reserved)				Reserved for Anybus CompactCom 30 applications.
5	Enable Web ADI access	Get	BOOL	True (Enabled)	Enable/Disable Web ADI access
6	Enable FTP server	Get	BOOL	True (Enabled)	Enable/Disable FTP server
7	Enable admin mode	Get	BOOL	False (Disabled)	Enable/Disable FTP admin mode
8	Network Status	Set	UINT16	-	See below.

#	Name	Access	Data Type	Default Value	Comment
9	Port 1 MAC address	Get	Array of UINT8	-	Note: This attribute is only valid for PROFINET devices. 6 byte MAC address for port 1 (mandatory for the LLDP protocol). This setting overrides any Port MAC address in the host PROFINET IO Object. Do not implement this attribute if the pre-programmed Mac address is to be used.
10	Port 2 MAC address	Get	Array of UINT8	-	Note: This attribute is only valid for PROFINET devices. 6 byte MAC address for port 2 (mandatory for the LLDP protocol). This setting overrides any Port MAC address in the host PROFINET IO Object. Do not implement this attribute if the pre-programmed Mac address is to be used.
11	Enable ACD	Get	BOOL	1 = True (Enabled)	Enable/Disable ACD protocol. If ACD functionality is disabled using this attribute, the ACD attributes in the CIP TCP/IP object (F5h) are not available.
12	Port 1 State	Get	ENUM	0 (Enabled)	The state of Ethernet port 1. This attribute is not read by EtherCAT devices, where Port 1 is always enabled. 00h: Enabled 01h: Disabled. The port is treated as existing. References to the port can exist, e.g. in network protocol or on website.
13	Port 2 State	Get	ENUM	0 (Enabled)	The state of Ethernet port 2. This attribute is not read by EtherCAT devices, where Port 1 is always enabled. 00h: Enabled 01h: Disabled. The port is treated as existing. References to the port can exist, e.g. in network protocol or on website. 02h: Inactive. The attribute is set to this value for a device that only has one physical port. All two-port functionality is disabled. No references can be made to this port. Note: This functionality is only available for Ethernet(IP devices at the moment.
14	(reserved)				
15	Enable reset from HICP	Get	BOOL	0 = False	Enables the option to reset the module from HICP.
16	IP configuration	Set	Struct of: UINT32 (IP address) UINT32 (Subnet mask) UINT32 (Gateway)	N/A	Whenever the configuration is assigned or changed, the Anybus CompactCom module will update this attribute.

#	Name	Access	Data Type	Default Value	Comment
17	IP address byte 0-2	Get	Array of UINT8[3]	[0] = 192 [1] = 168 [2] = 0	First three bytes in IP address. Used in "shift register mode" if the configuration switch value is set to 1-245. In that case the IP address will be set to: Y[0].Y[1].Y[2].X Where Y0-2 is configured by this attribute and the last byte X by the configuration switch.
18	Get	Ethernet PHY Configuration	Array of BITS16	0x0000 for each port	Ethernet PHY configuration bit field. The length of the array shall equal the number of Ethernet ports of the product. Each element represents the configuration of one Ethernet port (element #0 maps to Ethernet port #1, element #1 maps to Ethernet port #2 and so on). Note: Only valid for EtherNet/IP and Modbus-TCP devices. Bit 0: Auto negotiation fallback duplex 0 = Half duplex 1 = Full duplex Bit 1-15: Reserved

Network Status

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

Bit	Contents	Description
0	Link	Current global link status 1= Link sensed 0= No link
1	IP established	1 = IP address established 0 = IP address not established
2	(reserved)	(mask off and ignore)
3	Link port 1	Current link status for port 1 1 = Link sensed 0 = No link
4	Link port 2	Current link status for port 2 1 = Link sensed 0 = No link
5... 15	(reserved)	(mask off and ignore)

7 Web Server

7.1 General Information

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

The web interfaces are stored in the file system, which can be accessed through the FTP server. If necessary, the web server can be completely disabled in the Ethernet Host Object (F9h).

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

See also...

- [FTP Server, p. 86](#)
- [Server Side Include \(SSI\), p. 89](#)
- [JSON, p. 107](#)
- [Ethernet Host Object \(F9h\), p. 77](#)

7.2 Default Web Pages

The default web pages provide access to:

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

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

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

7.2.1 Network Configuration

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

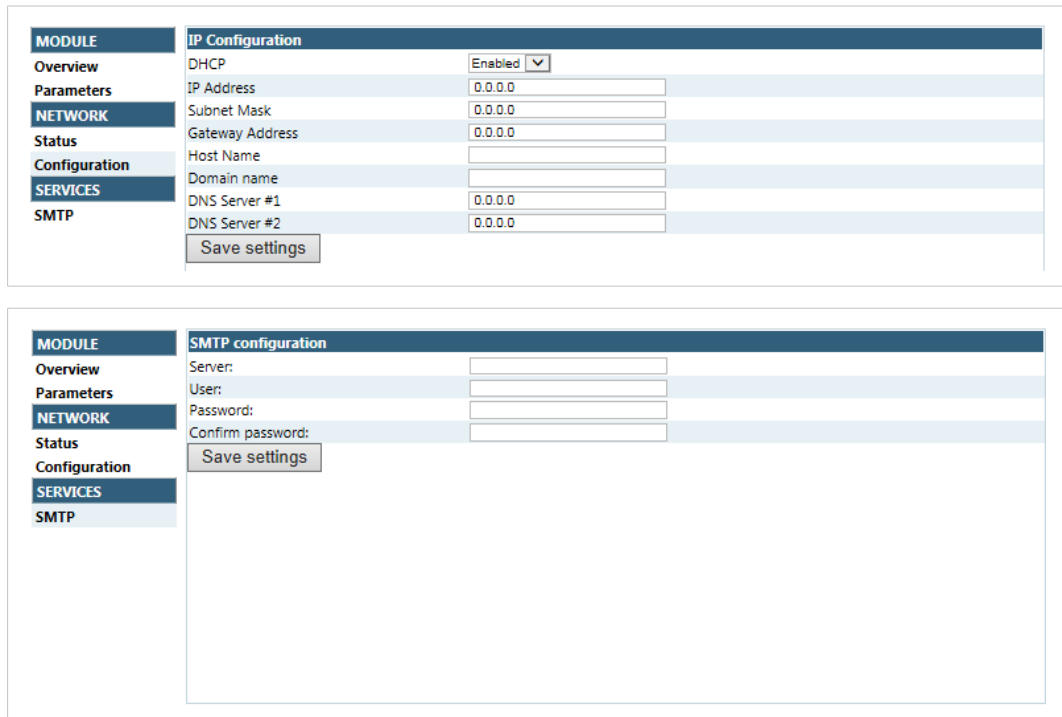


Fig. 2

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

Available IP Configuration Settings

Name	Description
DHCP	Checkbox for enabling or disabling DHCP Default value: disabled
IP address	The TCP/IP settings of the module Default values: 0.0.0.0 Value ranges: 0.0.0.0 - 255.255.255.255
Subnet mask	
Gateway address	
Host name	IP address or name Max 64 characters
Domain name	IP address or name Max 48 characters

Available SMTP Settings

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

7.2.2 Network Status Page

The Network Status web page contains the following information:

Current IP Configuration	Description
DHCP:	-
Host Name:	-
IP Address:	-
Subnet Mask:	-
Gateway Address:	-
DNS Server #1:	-
DNS Server #2:	-
Domain Name:	-

Current Ethernet Status	Description
MAC Address	-
Port 1	The current link speed and duplex configuration.
Port 2	The current link speed and duplex configuration.

EtherCAT Statistics	Description
Logical EoE port link:	Link on one of the physical Ethernet ports is not enough to be able to communicate using EoE. It is also necessary for the network state to be at least pre-operational and mailbox communication to be active. "Logical EoE port link" indicates if these requirements are fulfilled in order for the device to be able to send and receive EoE frames.
Invalid frame counter IN port:	Value of ESC register 0x300. Number of invalid frames received.
Rx error counter IN port:	Value of ESC register 0x301. Number of Rx errors reported by the PHY.
Forwarded error counter IN port:	Value of ESC register 0x308. Number of forwarded Rx errors.
Lost link counter IN port:	Value of ESC register 0x310. Number of times link has been lost on the port.
Invalid frame counter OUT port:	Value of ESC register 0x302. Number of invalid frames received.
Rx error counter OUT port:	Value of ESC register 0x303. Number of Rx errors reported by the PHY.
Forwarded error counter OUT port:	Value of ESC register 0x309. Number of forwarded Rx errors.
Lost link counter OUT port:	Value of ESC register 0x311. Number of times link has been lost on the port.

EoE 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.

7.3 Server Configuration

7.3.1 General Information

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

```
File Format:
  [WebRoot]
  \web

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

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

Web Root Directory [WebRoot]	The web server cannot access files outside this directory.
Content Types [FileTypes]	A list of file extensions and their reported content types. See also... Default Content Types, p. 84
SSI File Types [SSIFileTypes]	By default, only files with the extension 'shtm' are scanned for SSI. Additional SSI file types can be added here as necessary.

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

7.3.2 Index page

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

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



Substitute <WebRoot> with the web root directory specified in \http.cfg.

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

See also ...

- [Default Web Pages, p. 80](#)

7.3.3 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.

7.3.4 Authorization

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

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

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

  [AuthName]
  (message goes here)
```

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



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

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

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

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

8 FTP Server

8.1 General Information

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

disabled using attribute #6 in the Ethernet Host Object (F9h).

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

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

The FTP server supports up to two concurrent clients.

8.2 User Accounts

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

File Format:

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

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

File Format (\ftp.cfg):

```
User1:Password1:Homedirectory1
User2:Password2:Homedirectory2
.
.
UserN:PasswordN:HomedirectoryN
\path\userlistA:HomedirectoryA
\path\userlistB:HomedirectoryB
```

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

File Format:

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

Notes:

- Usernames must not exceed 16 characters in length.
- Passwords must not exceed 16 characters in length.
- Usernames and passwords must only contain 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 the user shall be able to log in. It is not enough just to add the user information to the `ftp.cfg` file.
- If Admin Mode has been enabled in the Ethernet Object, all username/password combinations will be accepted and the user will have unrestricted access to the file system (i.e. the home directory will be the system root). The `vfs` folder is read-only.
- It is strongly recommended to have at least one user with root access (`\`) permission. If not, Admin Mode must be enabled each time a system file needs to be altered (including `\ftp.cfg`).

8.3 Session Example

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

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

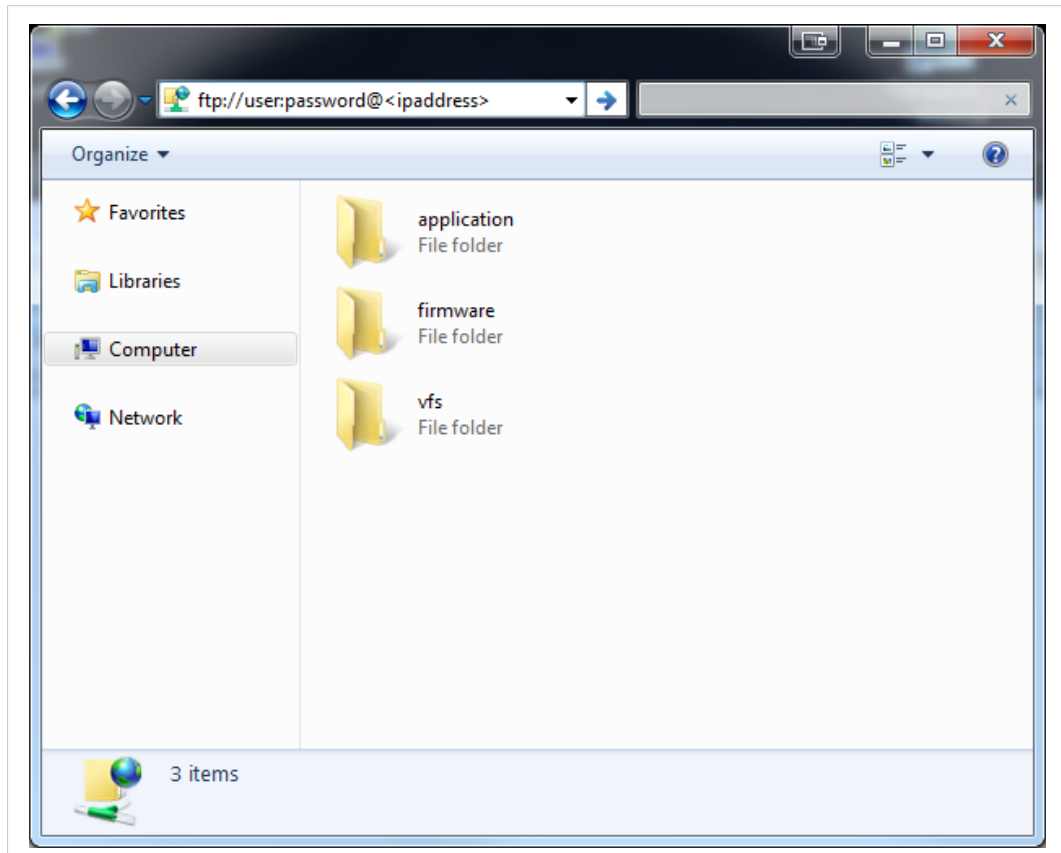


Fig. 3

9 E-mail Client

9.1 General Information

The built-in e-mail client allows the application to send e-mail messages through an SMTP-server. Messages can either be specified directly in the SMTP Client Object (04h), or retrieved from the file system. The latter may contain SSI, however note that for technical reasons, certain commands cannot be used (specified separately for each SSI command).

The client supports authentication using the 'LOGIN' method. Account settings etc. are stored in the Network Configuration Object (04h).

9.2 How to Send E-mail Messages

To be able to send e-mail messages, the SMTP-account settings must be specified.

This includes:

- A valid SMTP-server address
- A valid username
- A valid password

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

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

Sending a message based on a file in the file system is achieved using the Send Email from File command. This command is described in the SMTP Client Object (04h).

10 Server Side Include (SSI)

10.1 General Information


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

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

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

10.2 Include File

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

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

Syntax:

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

filename: Source file

Scenario	Default Output
Success	(contents of file)

10.3 Command Functions

10.3.1 General Information

Command functions executes commands and includes the result.

General Syntax

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

command: Command function, see below

Command Functions

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

10.3.2 GetConfigItem()

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

File Format

The source file must have the following format:

```
[key1]
value1

[key2]
value2

...

[keyN]
valueN
```

Syntax:

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

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

Default Output

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

Example

The following SSI...

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


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

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

... returns the string 'Third'.

10.3.3 SetConfigItem()

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

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

File Format

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

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

[form object name 2]
form object value 2

[form object name 3]
form object value 3

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

 *Form objects with names starting with underscore will not be stored.*

Syntax:

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

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

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

Default Output

Scenario	Default Output
Success	"Configuration stored to " <i>filename</i> "
Authentication Error	"Authentication error"
File open error	"Failed to open file " <i>filename</i> "
File write error	"Could not store configuration to " <i>filename</i> "

Example

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

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

<?--#exec cmd_argument='SetConfigItem("\food.txt")'-->

<FORM action="test.shtm">
```

```
<P>
  <LABEL for="Name">Name: </LABEL><BR>
  <INPUT type="text" name="Name"><BR><BR>

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

  <LABEL for="Food">Food: </LABEL><BR>
  <INPUT type="radio" name="Food" value="Cheese"> Cheese<BR>
  <INPUT type="radio" name="Food" value="Sausage"> Sausage<BR><BR>

  <LABEL for="Drink">Drink: </LABEL><BR>
  <INPUT type="radio" name="Drink" value="Wine"> Wine<BR>
  <INPUT type="radio" name="Drink" value="Beer"> Beer<BR><BR>

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

</BODY>
</HTML>
```

The resulting file ('food.txt') may look somewhat as follows:

```
[Name]
Cliff Barnes

[Food]
Cheese

[Drink]
Beer
```



In order for this example to work, the HTML file must be named "test.shtml".

10.3.4 SsiOutput()

This command temporarily modifies the SSI output of the following command function.

Syntax:

```
<?--#exec cmd_argument='SsiOutput("success", "failure")'-->
```

success: String to use in case of success
failure: String to use in case of failure

Default Output

(this command produces no output on its own)

Example

The following example illustrates how to use this command.

```
<?--#exec cmd_argument='SsiOutput ("Parameter stored", "Error")'-->
<?--#exec cmd_argument='SetConfigItem("File.cfg", Overwrite)'-->
```

See also...

- [SSI Output Configuration, p. 106](#)

10.3.5 DisplayRemoteUser

This command stores returns the username on an authentication session.



This command cannot be used in e-mail messages.

Syntax:


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

Default Output

Scenario	Default Output
Success	(current user)

10.3.6 ChangeLanguage()

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

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

Syntax:

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

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

The passed value must be a single digit as follows:

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

Default Output

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

Example

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


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

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

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

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

</BODY>
</HTML>
```

 *In order for this example to work, the HTML file must be named "test.shtm".*

10.3.7 IncludeFile()

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

Syntax:

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

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

Default Output

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

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, adipiscing elit...
```

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

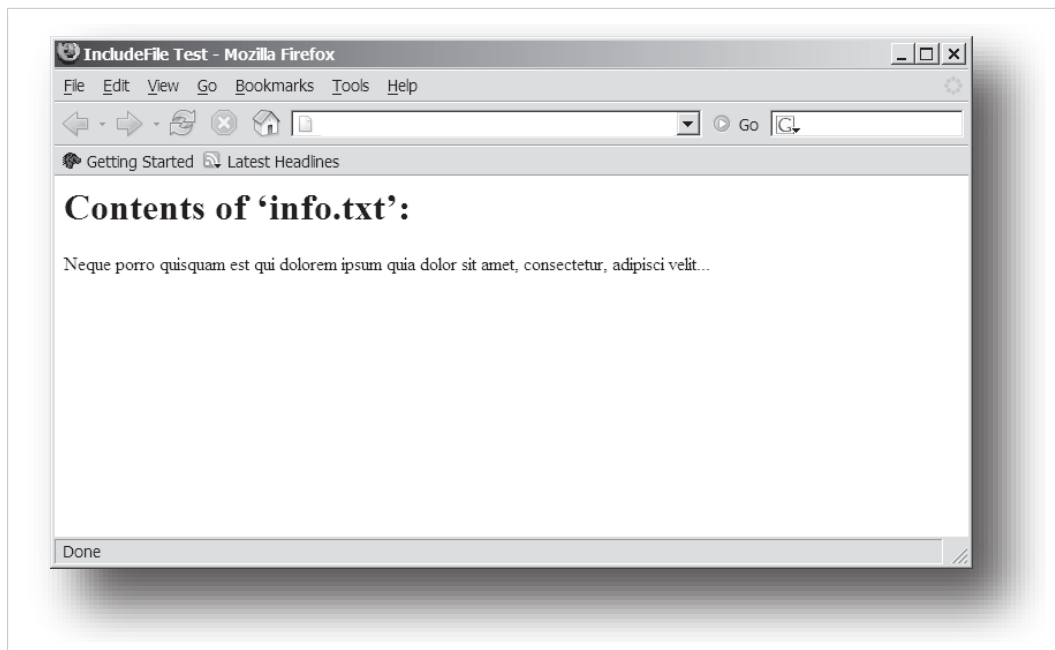



Fig. 4

See also...

- [Include File, p. 89](#)

10.3.8 SaveDataToFile()

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

 This function cannot be used in e-mail messages.

Syntax:

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

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

Default Output

Scenario	Default Output
Success	"Configuration stored to <i>filename</i> "
Authentication Error	"Authentication error"
File Write Error	"Could not store configuration to <i>filename</i> "

Example

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

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

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

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

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

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

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

</BODY>
</HTML>
```

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



In order for this example to work, the HTML file must be named "test.shtml".

10.3.9 printf()

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

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

Syntax:

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

template:	Template which determines how the arguments shall be represented. May contain any number of formatting tags which are substituted by subsequent arguments and formatted as requested. The number of format tags must match the number of arguments; if not, the result is undefined. See section "Formatting Tags" below for more information.
argument:	Source arguments; optional parameters which specify the actual source of the data that shall be inserted in the template string. The number of arguments must match the number of formatting tags; if not, the result is undefined. At the time of writing, the only allowed argument is ABCCMessage(). See also... <ul style="list-style-type: none"> • ABCCMessage(), p. 103

Default Output

Scenario	Default Output
Success	(printf() result)
ABCCMessage error	ABCCMessage error string (Errors, p. 105)

Example

See ..

- [ABCCMessage\(\), p. 103](#)
- [Example \(Get_Attribute\);, p. 104](#)

Formatting Tags

Formatting tags are written as follows:

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

- Type (Required)

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

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

- Flags (Optional)

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

- Width (Optional)

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

- Precision (Optional)

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

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

- Modifier

Modifier Character	Meaning
hh	Argument is interpreted as SINT8 or UINT8
h	Argument is interpreted as SINT16 or UINT16
L	Argument is interpreted as SINT32 or UINT32

10.3.10 scanf()

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

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



This command cannot be used in e-mail messages.

Syntax:

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

source	Name of the HTML form object from which the string shall be extracted.
template:	Template which specifies how to parse and interpret the data. May contain any number of formatting tags which determine the conversion prior to sending the data to subsequent arguments. The number of formatting tags must match the number of arguments; if not, the result is undefined. See section "Formatting Tags" below for more information.
argument:	Destination argument(s) specifying where to send the interpreted data. The number of arguments must match the number of formatting tags; if not, the result is undefined. At the time of writing, the only allowed argument is ABCCMessage(). See also... <ul style="list-style-type: none"> • ABCCMessage(), p. 103

Default Output

Scenario	Default Output
Success	"Success"
Parsing error	"Incorrect data format"
Too much data for argument	"Too much data"
ABCCMessage error	ABCCMessage error string (Errors, p. 105)

Example

See also...

[ABCCMessage\(\), p. 103](#)

[Example \(Set_Attribute\);, p. 105](#)

Formatting Tags

Formatting tags are written as follows:

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

- Type (Required)

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

Type	Input	Argument Data Type
c	Single character	CHAR
d	Accepts a signed decimal integer	SINT8 SINT16 SINT32
i	Accepts a signed or unsigned decimal integer. May be given as decimal, hexadecimal or octal, determined by the initial characters of the input data: Initial Characters: Format: 0x Hexadecimal 0: Octal 1... 9: Decimal	SINT8/UINT8 SINT16/UINT16 SINT32/UINT32
u	Accepts an unsigned decimal integer.	UINT8 UINT16 UINT32
o	Accepts an optionally signed octal integer.	SINT8/UINT8 SINT16/UINT16 SINT32/UINT32
x, X	Accepts an optionally signed hexadecimal integer.	SINT8/UINT8 SINT16/UINT16 SINT32/UINT32
e, E, f, g, G	Accepts an optionally signed floating point number. The input format for floating-point numbers is a string of digits, with some optional characteristics: <ul style="list-style-type: none"> – It can be a signed value – It can be an exponential value, containing a decimal rational number followed by an exponent field, which consists of an 'E' or an 'e' followed by an integer. 	FLOAT
n	Consumes no input; the corresponding argument is an integer into which scanf writes the number of characters read from the object input.	SINT8/UINT8 SINT16/UINT16 SINT32/UINT32
s	Accepts a sequence of nonwhitespace characters	STRING
[scanset]	Accepts a sequence of nonwhitespace characters from a set of expected bytes specified by the scanlist (e.g '[0123456789ABCDEF]') A literal '[' character can be specified as the first character of the set. A caret character (^) immediately following the initial '[' inverts the scanlist, i.e. allows all characters except the ones that are listed.	STRING
%	Accepts a single '%' input at this point; no assignment or conversion is done. The complete conversion specification should be '%%'.	-

- * (Optional)

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

- Width (Optional)

Specifies the maximum number of characters to be read

- Modifier (Optional)

Specifies a different data size.

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

10.4 Argument Functions

10.4.1 General Information

Argument functions are supplied as parameters to certain command functions.

General Syntax:

(Syntax depends on context)

Argument Functions:

Function	Description
ABCCMessage()	-

10.4.2 ABCCMessage()

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

Syntax

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

object	Specifies the Destination Object
instance	Specifies the Destination Instance
command	Specifies the Command Number
ce0	Specifies CmdExt[0] for the command message
ce1	Specifies CmdExt[1] for the command message
msgdata	Specifies the actual contents of the MsgData[] subfield in the command <ul style="list-style-type: none"> • Data can be supplied in direct form (format depends on c_type) • The keyword "ARG" is used when data is supplied by the parent command (e.g. scanf()).
c_type:	Specifies the data type in the command (msgdata), see below.
r_type:	Specifies the data type in the response (msgdata), see below.

Numeric input can be supplied in the following formats:

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

- Command Data Types (c_type)

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

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

- Response Data Types (r_type)

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

Type	Supports Arrays	Data format (as supplied in msgdata)
BOOL	Yes	Optionally, it is possible to exchange the BOOL data with a message based on the value (true or false). In such case, the actual data type returned from the function will be STRING. Syntax: BOOL<true><false> For arrays, the format will be BOOL[n]<true><false>.
SINT8	Yes	-
SINT16	Yes	-
SINT32	Yes	-
UINT8	Yes	This type can also be used when reading ENUM data types from an object. In such case, the actual ENUM value will be returned.
UINT16	Yes	-
UINT32	Yes	-
CHAR	Yes	-
STRING	No	-
ENUM	No	When using this data type, the 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



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

Example (Get_Attribute):

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

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

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

Example (Set_Attribute):

This example shows how to set the IP address using scanf() and ABCCMessage(). Note the special parameter value "ARG", which instructs the module to use the passed form data (parsed by scanf()).

```
<!--#exec cmd_argument='scanf("IP", "%u.%u.%u.%u",
ABCCMessage(4,3,2,5,0,ARG,UINT8[4],NONE ) )'-->
```

Variable	Value	Comments
object	4	Network Configuration Object (04h)
instance	3	Instance #3 (IP address)
command	2	Set_attribute
ce0	5	Attribute #5
ce1	0	-
msgdata	ARG	Use data parsed by scanf() call
c_type	UINT8[4]	Array of 4 unsigned 8-bit integers
r_type	NONE	Response message holds no data

Errors

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

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

Error Code	Output
16	"Segmentation buffer overflow"
17... 255	"Unknown error"

See also...

[SSI Output Configuration, p. 106](#)

10.5 SSI Output Configuration

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

File format:

[ABCCMessage_X] 0:"Success string" 1:"Error string 1" 2:"Error string 2" ... 16:"Error string 16"	Each error code corresponds to a dedicated output string, labelled from 1 to 16. See Errors, p. 105
[GetConfigItem_X] 0: "Success string" 1:"Authentication error string" 2:"File open error string" 3:"Tag not found string"	Use "%s" to include the name of the file.
[SetConfigItem_X] 0: "Success string" 1:"Authentication error string" 2:"File open error string" 3:"File write error string"	Use "%s" to include the name of the file.
[IncludeFile_X] 0: "Success string" 1:"Authentication error string" 2:"File read error string"	Use "%s" to include the name of the file.
[scanf_X] 0: "Success string" 1:"Parsing error string"	-
[ChangeLanguage_X] 0: "Success string" 1:"Change error string"	-

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

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

See also...

•

[SsiOutput\(\), p. 93](#)

11 JSON

11.1 General Information

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

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

JSON requests shall be UTF-8 encoded. The module will interpret JSON requests as UTF-8 encoded, while all other HTTP requests will be interpreted as ISO-8859-1 encoded. All JSON responses, sent by the module, are UTF-8 encoded, while all other files sent by the web server are encoded as stored in the file system.

11.1.1 Access

The JSON resources should be password protected. Add password protection by adding a file called `web_accs.cfg` in the root directory.

11.2 JSON Objects

11.2.1 ADI

info.json

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

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

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

JSON object layout:

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

data.json

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

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

JSON object layout:

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

metadata.json

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

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

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

JSON object layout:

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

enum.json

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

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

Name	Data Type	Note
string	String	-
value	Number	-

JSON object layout:

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

update.json

POST adi/update.json - form data:

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

Updates the value of an ADI for the specified ADI instance <instance>. The value, <data>, shall be hex formatted (see [Hex Format Explained, p. 113](#) for more information). If <element> is specified, only the value of the specified element is updated. In this case, <data> shall only update that single element value. When <element> is not specified, <data> shall represent the entire array value. Optionally, a callback may be passed to the request for JSONP output

Name	Data Type	Note
result	Number	0 = success

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

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

11.2.2 Module

info.json

GET module/info.json

Name	Data Type	Note
modulename	String	-
serial	String	32 bit hex ASCII
fwver	Array of Number	(major, minor, build)

Name	Data Type	Note
uptime	Array of Number	[high, low] milliseconds (ms)
cpuload	Number	CPU load in %

JSON object layout:

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

11.2.3 Network

ethstatus.json

GET network/ethstatus.json.

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

Comm Object Definition:

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

JSON object layout:

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

ipstatus.json & ipconf.json

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

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

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

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

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

Example:

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

ethconf.json

GET network/ethconf.json

Name	Data Type	Note
comm1	Number	-
comm2	Number	-

ifcounters.json

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

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

mediacounters.json

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

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

nwstats.json

GET network/nwstats.json.

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

Example output:

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

Get network specific statistics:

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

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

11.2.4 Services

smtp.json

GET services/smtp.json.



Password is not returned when retrieving the settings.

Name

Data Type

Note

server

String

-

user

String

-

11.2.5 Hex Format Explained

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

Examples:

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

```
0500
```

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

```
414243
```

11.3 Example

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

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

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

      CPU Load:
```

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

A Categorization of Functionality

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

A.1 Basic

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

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

A.2 Extended

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

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

B Implementation Details

B.1 SUP-bit Definition

The supervised bit (SUP) indicates that the network participation is supervised by another network device. In the case of EtherCAT, this functionality is mapped to the SyncManager watchdog, which can be used to detect loss of communication with the master. The SyncManager watchdog is enabled by the master.

EtherCAT-specific interpretation:

SUP-bit	Interpretation
0	SyncManager Watchdog is disabled or not running.
1	SyncManager Watchdog is enabled and running.



The watchdog and supervised bit (SUP) will not be available if the Read Process Data size is zero.

B.2 Anybus State Machine

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

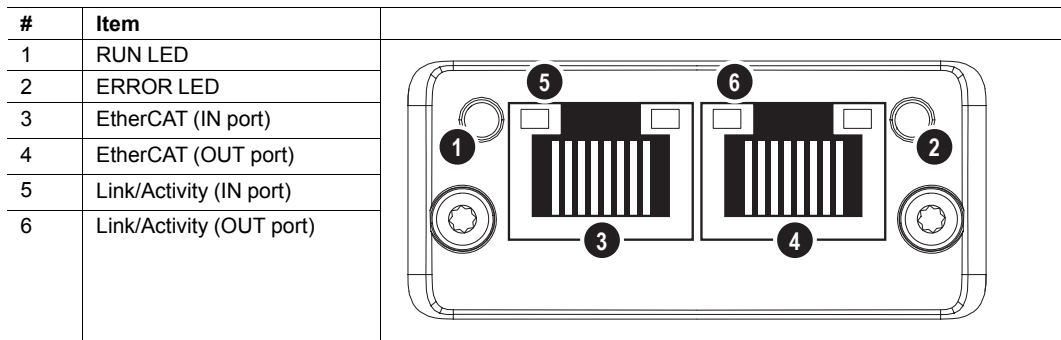
Anybus State	Corresponding EtherCAT State
WAIT_PROCESS	INIT, BOOTSTRAP or PRE-OPERATIONAL
ERROR	('Error Ind'-bit in 'AL-Status' is set)
PROCESS_ACTIVE	OPERATIONAL
IDLE	SAFE-OPERATIONAL
EXCEPTION	(EtherCAT interface is forced to INIT state, and the master is informed that a power cycle is required to resume communication)

B.3 Application Watchdog Timeout Handling

The Anybus CompactCom 40 EtherCAT module will enter the EXCEPTION state if the application watchdog times out.

C Technical Specification

C.1 Front View



The flash sequences for the RUN LED and the ERROR LED are defined in ETG1300_S_R_V1i1i0_IndicatorLabelingSpecification.pdf (ETG).

C.1.1 RUN LED

This LED reflects the status of the EtherCAT device.

LED State	Indication	Description
Off	INIT	EtherCAT device in 'INIT'-state (or no power)
Green	OPERATIONAL	EtherCAT device in 'OPERATIONAL'-state
Green, blinking	PRE-OPERATIONAL	EtherCAT device in 'PRE-OPERATIONAL'-state
Green, single flash	SAFE-OPERATIONAL	EtherCAT device in 'SAFE-OPERATIONAL'-state
Flickering	BOOT	The EtherCAT device is in 'BOOT' state
Red	(Fatal Event)	If RUN and ERR turn red, this indicates a fatal event, forcing the bus interface to a physically passive state. Contact HMS technical support.

C.1.2 ERR LED

This LED indicates EtherCAT communication errors etc.

LED State	Indication	Description
Off	No error	No error (or no power)
Red, blinking	Invalid configuration	State change received from master is not possible due to invalid register or object settings.
Red, single flash	Unsolicited state change	Slave device application has changed the EtherCAT state autonomously.
Red, double flash	Application watchdog timeout	Sync manager watchdog timeout.
Red	Application controller failure	Anybus module in EXCEPTION. If RUN and ERR turn red, this indicates a fatal event, forcing the bus interface to a physically passive state. Contact HMS technical support.
Flickering	Booting error detected	E.g. due to firmware download failure.

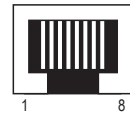
C.1.3 Link/Activity

These LEDs indicate the EtherCAT link status and activity.

LED State	Indication	Description
Off	No link	Link not sensed (or no power)
Green	Link sensed, no activity	Link sensed, no traffic detected
Green, flickering	Link sensed, activity	Link sensed, traffic detected

C.1.4 Ethernet Connector (RJ45)

Pin	Signal	Notes
1	Tx+	-
2	Tx-	-
3	Rx+	-
4	-	Normally left unused; to ensure signal integrity, these pins are tied together and terminated to PE via a filter circuit in the module.
5	-	
6	Rx-	-
7	-	Normally left unused; to ensure signal integrity, these pins are tied together and terminated to PE via a filter circuit in the module.
8	-	



C.2 Functional Earth (FE) Requirements

In order to ensure proper EMC behavior, the module must be properly connected to functional earth via the FE pad / FE mechanism described in the general Anybus CompactCom M40 Hardware Design Guide.

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

C.3 Power Supply

C.3.1 Supply Voltage

The module requires a regulated 3.3V power source as specified in the general Anybus CompactCom M40 Hardware Design Guide.

C.3.2 Power Consumption

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

The current hardware design consumes up to 430 mA.



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

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

C.4 Environmental Specification

Consult the Anybus CompactCom Hardware M40 Design Guide for further information.

C.5 EMC Compliance

Consult the Anybus CompactCom Hardware M40 Design Guide for further information.

D Timing & Performance

D.1 General Information

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

The following timing aspects are measured:

Category	Parameters	Page
Startup Delay	T1, T2	120
NW_INIT Handling	T100	120
Event Based WrMsg Busy Time	T103	120
Event Based Process Data Delay	T101, T102	121

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

D.2 Internal Timing

D.2.1 Startup Delay

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

Parameter	Description	Max.	Unit.
T1	The Anybus CompactCom 40 EtherCAT module generates the first application interrupt (parallel mode)	11	ms
T2	The Anybus CompactCom 40 EtherCAT module is able to receive and handle the first application telegram (serial mode)	11	ms

D.2.2 NW_INIT Handling

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

Parameter	Conditions
No. of network specific commands	Max.
No. of ADIs (single UINT8) mapped to Process Data in each direction. (If the network specific maximum is less than the value given here, the network specific value will be used.)	32
Event based application message response time	> 1 ms
Ping-pong application response time	> 10 ms
No. of simultaneously outstanding Anybus commands that the application can handle	1

Parameter	Description	Communication	Max.	Unit.
T100	NW_INIT handling	Event based modes	3.6	ms

D.2.3 Event Based WrMsg Busy Time

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

Parameter	Description	Min.	Max.	Unit.
T103	H_WRMSG area busy time	2.8	7.2	µs

D.2.4 Event Based Process Data Delay

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

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

The tests were run in 16-bit parallel event mode, with interrupts triggered only for new process data events. Eight different IO sizes (2, 16, 32, 64, 128, 256, 512 and 1024 bytes) were used in the tests, all giving the same test results.

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

Parameter	Description	Delay (min.)	Delay (typ.)	Delay (max.)	Unit
T101	Read process data delay	-	-	228	ns
T102	Write process data delay	-	-	170	ns

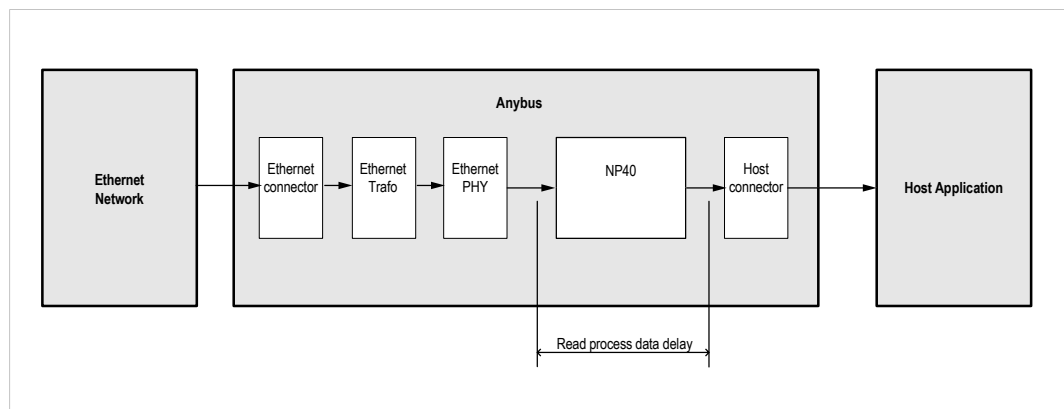


Fig. 5

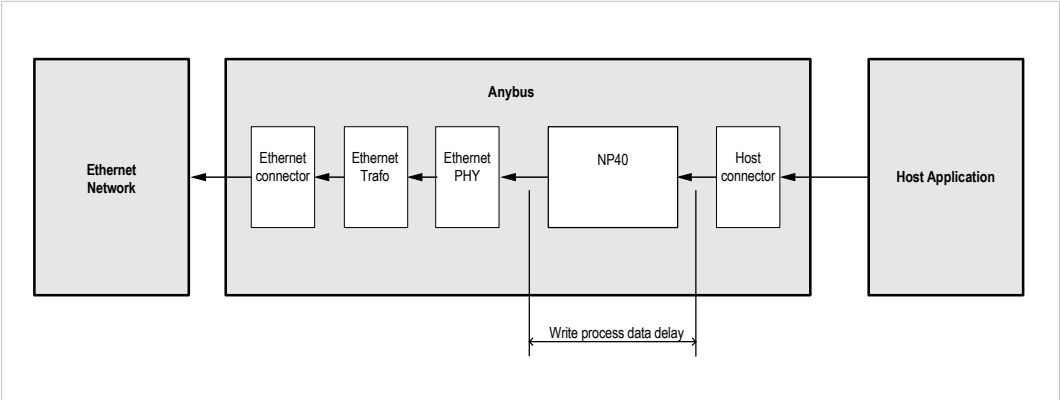


Fig. 6

E Secure HICP (Secure Host IP Configuration Protocol)

E.1 General

The Anybus CompactCom 40 EtherCAT supports the Secure HICP protocol used by the Anybus IPconfig utility for changing settings, e.g. IP address, Subnet mask, and enable/disable DHCP. Anybus IPconfig can be downloaded free of charge from the HMS website, www.anybus.com. This utility may be used to access the network settings of any Anybus product connected to the network via UDP port 3250.

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

E.2 Operation

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

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

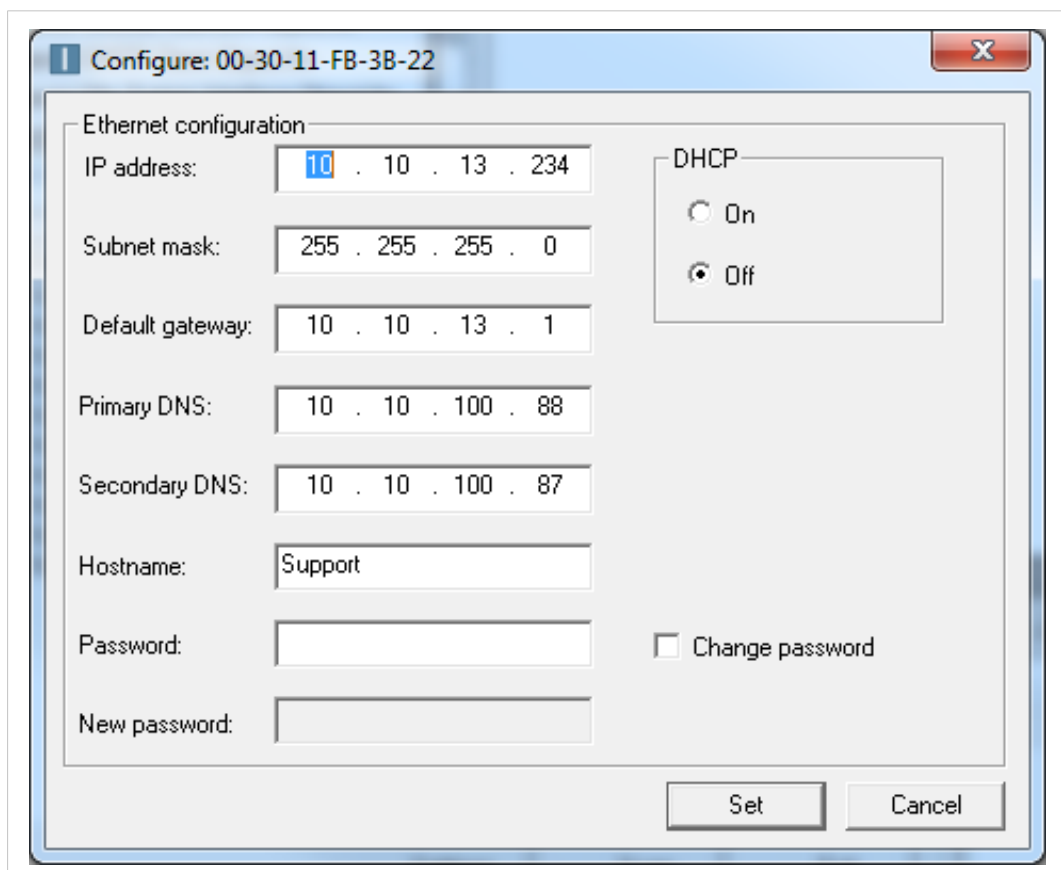


Fig. 7

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

F Copyright Notices

Print formatting routines

Copyright (C) 2002 Michael Ringgaard. All rights reserved.

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

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

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

Copyright (c) 2002 Florian Schulze.
All rights reserved.

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

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

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

ftpd.c - This file is part of the FTP daemon for lwIP

FatFs - FAT file system module R0.09b (C)ChaN, 2013

FatFs module is a generic FAT file system module for small embedded systems. This is a free software that opened for education, research and commercial developments under license policy of following terms.

Copyright (C) 2013, ChaN, all right reserved.

The FatFs module is a free software and there is NO WARRANTY. No restriction on use. You can use, modify and redistribute it for personal, non-profit or commercial products UNDER YOUR RESPONSIBILITY. Redistributions of source code must retain the above copyright notice.

lwIP is licenced under the BSD licence:

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

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

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

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

Copyright 2013 jQuery Foundation and other contributors
<http://jquery.com/>

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

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

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

rsvp.js

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

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

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

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

libb (big.js)

The MIT Expat Licence.

Copyright (c) 2012 Michael Mclaughlin

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

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

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

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

Copyright (c) 2009, Ben Hoyt
All rights reserved.

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

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

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

MD5 routines

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

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

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

L. Peter Deutsch
ghost@aladdin.com

Format - lightweight string formatting library.
Copyright (C) 2010-2013, Neil Johnson
All rights reserved.

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

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

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

HMS Industrial Networks AB
Box 4126
300 04 Halmstad, Sweden

info@hms.se

© 2016 HMS Industrial Networks AB
HMSI-27-220 2.1.1913 / 2016-09-12 08:11