Anybus® BY HMS NETWORKS

Anybus® CompactCom™ 30

SERCOS III

NETWORK GUIDE

HMSI-168-72 3.1 en-US ENGLISH

HMS Connecting Devices™

Important User Information

Liability

Every care has been taken in the preparation of this document. Please inform HMS Industrial Networks of any inaccuracies or omissions. The data and illustrations found in this document are not binding. We, HMS Industrial Networks, 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. HMS Industrial Networks 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 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 cannot assume responsibility for actual use based on these examples and illustrations.

Intellectual Property Rights

HMS Industrial Networks 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.

Table of Contents

Page

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 30 SERCOS III. The document describes the features that are specific to Anybus CompactCom 30 SERCOS III. For general information regarding Anybus CompactCom 30, consult the Anybus CompactCom 30 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 SERCOS III specific functionality is to be used, in-depth knowledge of SERCOS III networking internals and/or information from the official SERCOS III specifications may be required. In such cases, the persons responsible for the implementation of this product should either obtain the SERCOS III specification to gain sufficient knowledge or limit their implementation in such a way that this is not necessary.

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

1.2 Related Documents

Document	Author	Document ID
Anybus CompactCom 30 Software Design Guide	HMS	HMSI-168-97
Anybus CompactCom 30 Hardware Design Guide	HMS	HMSI-168-31
Anybus CompactCom Host Application Implementation Guide	HMS	HMSI-27-334
Anybus CompactCom Implementation Tutorial	HMS	HMSI-168–106
SERCOS III Specification	SERCOS International	

1.3 Document History

Version	Date	Description
1.00	2009-11-06	First official release
2.00	2010-04-15	Change of concept
2.01	2010-08-26	Added copyright notices
2.02	2011-02-10	Minor updates
2.03	2012-01-26	Minor update
2.04	2012-10-14	Minor correction
2.10	2015-02-13	Minor updates and additions
3.0	2018-10-23	First version in DOX, updated
3.1	2019-02-26	Rebranding Minor corrections

1.4 Document Conventions

Ordered lists are used for instructions that must be carried out in sequence:

1. First do this
2. Then do this

Unordered (bulleted) lists are used for:

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

...and for action-result type instructions:

► This action...

→ leads to this result

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: *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 Terminology

- 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.

1.5.1 Glossary

Word	Explanation
IDN	Identification numbers, data objects on SERCOS III
IDN/S	Identification numbers, standard data objects on SERCOS III

Word	Explanation
IDN/P	Identification numbers, manufacturer specific data objects on SERCOS III
SCP	SERCOS Communication Profile
GDP	Generic Device Profile
FTP	File Transfer Protocol
TFTP	Trivial File Transfer Protocol

1.6 Trademark Information

2 About the Anybus CompactCom 30 SERCOS III

2.1 General

The Anybus CompactCom 30 SERCOS III communication module provides instant SERCOS III 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.

The modular approach of the Anybus CompactCom 30 platform allows the Identity Object to be customized, allowing the end product to appear as a vendor-specific implementation rather than a generic Anybus module.

This product conforms to all aspects of the host interface for Active modules defined in the *Anybus CompactCom 30 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 take advantage of advanced network specific functionality, a certain degree of dedicated software support may be necessary.

The functionality of the module is described in two categories: Basic and Extended, see *Categorization of Functionality, p. 99*.

2.2 Features

- Galvanically isolated bus electronics
- Complete SERCOS III node
- Up to 256 byte of I/O data in each direction
- Up to 16319 ADIs can be accessed as manufacturer specific IDNs
- Supports GDP Basic, SCP_FixCFG, SCP_NRT, and FSP_10
- 100 Mbit/s full duplex Ethernet with integrated 2-port switch
- Up to 16383 ADIs can be accessed from the network as Manufacturer Specific Objects.
- TCP/IP Socket interface
- FTP server, TFTP server, e-mail and dynamic web server with SSI support
- Customizable Identity object
- 500 µs minimum cycle time

3 Tutorial

3.1 Introduction

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

3.2 Fieldbus Conformance Notes

- The Anybus CompactCom 30 SERCOS III has been pre-compliance tested by the Institute for Control Engineering of Machine Tools and Manufacturing Units (ISW) at the University of Stuttgart and found to comply with SERCOS interface specification, Compliance Class A. Contact HMS Industrial Networks for further information.

3.3 Conformance Test Guide

When using the default settings of all parameters, the Anybus CompactCom 30 SERCOS III is precertified for network compliance. This precertification is done to ensure that your product can be certified, but it does not mean that your product will not require certification.

Any change in the parameters in the SSDML file, supplied by HMS Industrial Networks, will require a certification. A Vendor ID can be obtained from SERCOS International and is compulsory for certification. This section provides a guide for a successful conformance testing of your product, containing the Anybus CompactCom 30 SERCOS III, to comply with the demands for network certification set by SERCOS International.

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

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

3.3.1 Reidentifying Your Product

After successful setting of attribute #5 (Setup Complete) in the Anybus Object (01h), the Anybus CompactCom 30 asks for identification data from the SERCOS III Host Object (F1h) and the Ethernet Host Object (F9h). Therefore, the attributes listed below shall be implemented and proper values returned.

Object/Instance	Attribute	Explanation	Default	Customer sample	Comment
SERCOS III Host Object (F1h), Instance 1	#1, Component name	With this attribute you set the Component name of the device.	“Communica-tion Adapter”	“Widget”	This information must match the keyword values of the “ElectronicLabel” section in the SDDML file.
SERCOS III Host Object (F1h), Instance 1	#2, Vendor code	With this attribute you set the Vendor code of the device.	005Ah	1111h	
SERCOS III Host Object (F1h), Instance 1	#3, Device name	With this attribute you set the Device name of the device	“Anybus CompactCom SERCOS III”	“Fast Widget”	
SERCOS III Host Object (F1h), Instance 1	#4, Vendor device ID	With this attribute you set the Vendor device ID of the device.	“ABCC SRC3”	“WID”	
SERCOS III Host Object (F1h), Instance 1	#5, Software revision	With this attribute you set the Software revision of the device.		1.1	
SERCOS III Host Object (F1h), Instance 1	#6, Serial number	With this attribute you set the Serial number of the device.		12345678h	
Ethernet Host Object (F9h), Instance 1	#1, MAC address	With this attribute you set the MAC address of the device.	Anybus-CC CANopen	00-11-22-33-44-55	6 byte physical address value from range obtained from IEEE

3.3.2 Factory Default Reset

Reset command to Application Object (FFh) must be supported

When Anybus CompactCom 30 SERCOS III products are delivered, they are required to be in their Factory Default state. When a Factory Default Reset command is received from the network, the Anybus CompactCom 30 will erase all non volatile information inform the host application that a reset of the Anybus CompactCom 30 is required. This is done by sending a Reset command to the Application Object (FFh) of the host (Power-on + Factory Default). For more details, please consult the Anybus CompactCom 30 Software Design Guide.

4 Basic Operation

4.1 General Information

4.1.1 Software Requirements

Generally, no additional network support code needs to be written to support the Anybus CompactCom 30 SERCOS III, however due to the nature of the SERCOS III networking system certain things must be taken into account:

- The Anybus CompactCom 30 SERCOS III supports one consuming and one producing connection.
- The flexible nature of the Anybus concept allows the application to modify the behavior on SERCOS III in ways which contradict the generic SDDML 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 SERCOS III-specific functionality may require in-depth knowledge in SERCOS III networking internals and/or information from the official SERCOS III specification. In such cases, the people responsible for the implementation of this product is expected either to obtain these specifications to gain sufficient knowledge or limit their implementation in such a way that this is not necessary.

See also...

- *Process Data, p. 13*
- *Diagnostic Object (02h), p. 52* (Anybus Module Object)
- Anybus-CompactCom Software Design Guide, ‘Application Data Object (FEh)’

4.1.2 SDDML File

HMS Industrial Networks supplies a generic Device Description 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. However, to ensure interoperability and to reduce the complexity for the end user, it is generally recommended to create a custom SDDML File to match the final implementation of the product.

The following scenarios are known to require a custom SDDML File.

- The use of a custom Vendor code
- The use of a custom Vendor device ID

Note that any deviations from the generic SDDML file requires the use of custom Vendor- and Product IDs.

See also...

- *Conformance Test Guide, p. 9*

4.2 Network Identity

By default, the module identifies itself as a generic Anybus implementation as follows:

Vendor Code	005Ah (HMS Industrial Networks, allocated by SERCOS International)
Device Type	“Communication adapter”
Vendor Device ID	“ABCC SRC3”
Device Name	“Anybus-CompactCom SERCOS III”

It is possible to customize the identity information so that the Anybus module appears as a vendor specific implementation rather than a generic Anybus product. Note however that this invalidates the standard SDDML-file an thus re-certification of the product is necessary.

See also...

- *SERCOS III Host Object (F1h), p. 93*

4.3 Communication Settings

Network related communication settings are grouped in the Network Configuration Object (04h). In case of the Anybus CompactCom 30 SERCOS III this includes:

Ethernet Interface Settings	The module is locked to 100 Mbit full duplex operation as required by SERCOS III.
TCP/IP Settings	These settings must be set in order for the module to be able to participate on the network.
SMTP Account Settings	These settings must be set in order for the module to be able to send e-mail messages.

See also...

- *Web Server, p. 23*
- *Network Configuration Object (04h), p. 54*
- *HICP (Host IP Configuration Protocol), p. 101*

4.4 Network Data Exchange

4.4.1 Application Data (ADI)

Application Data Instances (ADIs) can be accessed from the network as manufacturer specific IDNs. 4096 ADIs are accessible in each of the eight manufacturer specific parameter sets, for a total of 32767 possible ADIs. However, SERCOS limits the maximum total number of IDNs in a slave to 16383. This includes both standard IDNs and manufacturer specific IDNs. Any ADIs with instance numbers exceeding 32767 or order numbers exceeding 16383 minus the number of standard IDNs, implemented by the module, will not be accessible.

See also...

- *Manufacturer Specific IDNs, p. 19*

4.4.2 Process Data

ADIs mapped as Process Data will be exchanged cyclically. The actual map is based on the Process Data map specified during startup and cannot be changed from the network during runtime.

The module supports one consuming and one producing connection, each containing up to 256 bytes of data as shown in this table:

Device control	Process data connection content			
	Connection control	IO Control	ABCC process data	Pad byte
2 Byte	2 Byte	2 Byte	0 - 256 Byte	0 - 1 Byte

If the process data size is not even (odd number of bytes), an extra pad byte is appended at the end.

Due to limitations in the SERCOS FSP_IO profile, only a total of 255 process data mappings (total of inputs and outputs) are supported.

Device control is sent with Process data connection, but is not tied to it. The order does not have to be as shown in the table.

See also...

- *Standard IDNs, p. 16*
- *Manufacturer Specific IDNs, p. 19*

4.5 Diagnostics

The standard Diagnostic Object (02h) provides access to basic diagnostic functionality. Major unrecoverable events will cause the module to physically disconnect itself from the network, thus preventing network participation.

Major diagnostic events are translated to SERCOS Class 1 diagnostic events, if support for latching diagnostic events is included in the SERCOS III application object. Otherwise the major diagnostic events will be translated to class 2 events.

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

See also...

- *SERCOS III Host Object (F1h), p. 93*
- *Diagnostic Object (02h), p. 52*

4.6 Web Interface

The built-in web server can be used to provide rich, dynamic content, by means of SSI scripting. This enables access to information and configuration settings within the file system, as well as through the Anybus CompactCom object model.

Web server content resides within the FLASH-based file system, which means it can be accessed and customized as needed using a standard FTP-client.

See also...

- *File System, p. 15*
- *FTP Server, p. 21*
- *Web Server, p. 23*
- *Server Side Include (SSI), p. 30*

4.7 E-mail Client

The built-in email client enables the host application to send email messages stored in the file system, or defined directly within the SMTP Client Object (09h). Messages are scanned for SSI content, which means it's possible to embed dynamic information from the file system or from the Anybus CompactCom object model.

See also...

- *File System, p. 15*
- *E-Mail Client, p. 29*
- *Server Side Include (SSI), p. 30*
- *SMTP Client Object (09h), p. 76*

4.8 Network Reset Handling

4.8.1 Restore Manufacturer Parameters to Default

Upon receiving the command Load Defaults ?rocedure from the network, the module will issue a reset command to the Network Configuration Object (04h),with CmdExt[1] set to 01h (Factory default reset).

See also...

- *Network Configuration Object (04h), p. 54*, command details for Reset_Command

4.9 SERCOS Address

The SERCOS address range is 0 - 511. If defined during setup, the address can not be changed later from the network. If defined after setup, the SERCOS address can be changed from the network.

See also...

- *Network Configuration Object (04h), p. 54*, Instance #3, SERCOS Address

4.10 File System

4.10.1 General Information

The built-in file system hosts 1.18 MByte of non volatile storage, which can be accessed by the HTTP, FTP, and TFTP servers, the e-mail client, and the host application).

The file system uses the following conventions:

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

See also ...

- ref till ftpserver
- *Web Server, p. 23*
- reft till e-mail client
- *Server Side Include (SSI), p. 30*
- Filesystem interface object

> **!** *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

4.10.2 System Files

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

Example 1:

```
[Key1]
value of Key1

[Key2]
value of Key2
```

5 SERCOS III Implementation

5.1 Standard IDNs

The following standard IDNs are supported in Anybus CompactCom 30 SERCOS III.

IDN	Name	Description
S-0-0014	Interface status	
S-0-0017	IDN list of all operation data	Returns a list of all IDNs, including ADI IDNs
S-0-0021	IDN list of invalid operation data for CP2	
S-0-0025	IDN list of all procedure commands	
S-0-0095	Diagnostic message	Text string description of diagnostic message, see also *Diagnostics, p. 13* and *Diagnostic Object (02h), p. 52*
S-0-0099	Reset class 1 diagnostic	This procedure command will send a Reset diagnostic request to the SERCOS III host object, see *SERCOS III Host Object (F1h), p. 93*
S-0-0127	CP3 transition check	
S-0-0128	CP4 transition check	
S-0-0262	Load defaults procedure command	This procedure command will send a factory def al ut reset to the application object as well as restore all instances of the NC object to their default values
S-0-0265	Language selection	See *Language Codes, p. 102*
S-0-0266	List of available languages	See *Language Codes, p. 102*
S-0-0390	DiagnosticNumber	Diagnostic code of the latest diagnostic event.
S-0-1000	SCP Type & Version	List of supported SCPs (SERCOS Communication Profiles) This device supports FSP_IO, SCP_FIX_CFG (fixed configuration) and SCP_NRT (TCP/IP)
S-0-1002	Communication Cycle Time (tScyc)	
S-0-1003	Communication timeout for CP3/CP4	
S-0-1009	Device control offset in MDT	
S-0-1010	Lengths of MDTs	
S-0-1011	Device status offset in AT	
S-0-1012	Lengths of AT	
S-0-1013	SVC offset in MDT	
S-0-1014	SVC offset in AT	
S-0-1017	IPC transmission time	
S-0-1019	MAC address	Returns the current MAC ID
S-0-1020	IP address	Get/set current IP address
S-0-1021	Subnet mask	Get/set current subnet mask
S-0-1022	Gateway address	Get/set current gateway address
S-0-1026	Version of communication hardware	SERCON100S version
S-0-1027.0.1	Maximum MTU in NRT channel (requested)	
S-0-1027.0.2	Maximum MTU in NRT channel (effective)	MTU less than 68 will disable TCP/IP since TCP/IP requires an unfragmented frame size of at least 68 bytes
S-0-1035	Error counter Port1 & Port2	
S-0-1040	SERCOS address	Current SERCOS address, see instance #3, SERCOS Address, *Network Configuration Object (04h), p. 54*
S-0-1044	Device control	
S-0-1045	Device status	
S-0-1050.0.3	Telegram assignment	Telegram type (MDT or AT), telegram number (0 - 3), and C-Com offset for this connection
S-0-1050.0.5	Actual length of connection	Write process data size + 4 + pad byte if process data size is odd

IDN	Name	Description
S-0-1050.0.8	Connection control	
S-0-1050.1.3	Telegram assignment	Telegram type (MDT or AT), telegram number (0 - 3), and C-Com offset for this connection
S-0-1050.1.5	Actual length of connection	Read process data size + 4 + pad byte if process data size is odd
S-0-1050.1.8	Connection control	
S-0-1300.0.01	Component Name	See *SERCOS III Host Object (F1h), p. 93*
S-0-1300.0.03	Vendor Code	
S-0-1300.0.04	Device Name	
S-0-1300.0.05	Vendor Device ID	
S-0-1300.0.09	Software Revision	
S-0-1300.0.12	Serial Number	
S-0-1301	GDP Type & Version	
S-0-1302.0.1	FSP Type & Version	
S-0-1302.0.2	Function groups	List of all instanced FSP_IO function groups
S-0-1302.0.3	Application Type	Current application type, see instance #15, Application Type, *Network Configuration Object (04h), p. 54*
S-0-1399.0.1	Diagnostic Event	Hidden IDN. Used by conformance test
S-0-1500.0.1	IO Control	High bit activates PROCESS_ACTIVE state
S-0-1500.0.2	IO Status	Bit 15 set when in PROCESS_ACTIVE state Bit 14 always set Bit 12-13 indicate any DI instances
S-0-1500.0.3	List of module type codes	List of FSP_IO module type codes: 1500: “bus coupler” 1502: digital outputs 1503: digital inputs
S-0-1500.0.11	List of replaced function groups	List of FSP_IO function group replacements. The module only supports one-to-one replacements. Function groups 1502, 1503, 1504, and 1505 are supported. Attempts at replacement with other function groups will be refused.
S-0-1500.0.12	Rearrangement of IO resource	Procedure command used to perform the replacement according to IDN/S-0-1500.0.11
S-0-1502.x.3	Digital output, channel quantity	One 1502 IDN is generated for every read process data mapping, and one 1503 IDN is generated for every write process data mapping. Quantity and width are set to match the number of elements and data type size of the mapping. The structure instance number starts at 1 and is incremented for each process data mapping. For this reason the module will only support a maximum of 255 process data mappings.
S-0-1502.x.4	Digital output, channel width	
S-0-1503.x.7	Digital input, channel quantity	
S-0-1503.x.8	Digital input, channel width	
S-0-1504.x.3	Analog output, channel quantity	Normally there are no 1504 or 1505 IDNs available, but IDN/S-0-1500.0.11 and procedure command IDN/S-0-1500.0.12 can be used to replace digital 1502 and 1503 IDNs with analog 1504 and 1505 IDNs respectively. The replacement is purely a change in representation of the process data, and has no effect on the actual raw process data exchange.
S-0-1504.x.4	Analog output, channel width	
S-0-1505.x.7	Analog input, channel quantity	
S-0-1505.x.8	Analog input, channel width	

Anybus CompactCom 30 SERCOS III does not support synchronization of process data, but due to limitations in the SERCON100S, early versions of this product implements several SCP_SYNC related IDNs as well, see table below:

IDN	Name
S-0-1005	Minimum feedback processing time
S-0-1006	AT transmission starting time
S-0-1007	Feedback acquisition capture point
S-0-1008	Command value valid time
S-0-1015	Ring delay
S-0-1016	Slave delay
S-0-1023	SYNC jitter
S-0-1024	SYNC delay measuring procedure command
S-0-1028	Error counter MST-P/S
S-0-1041	AT Command value valid time
S-0-1050.0.1	Connection setup
S-0-1050.1.1	Connection setup
S-0-1050.0.10	Producer Cycle Time
S-0-1050.1.10	Producer Cycle Time
S-0-1050.0.11	Allowed Data Losses
S-0-1050.1.11	Allowed Data Losses
S-0-1050.0.12	Error Counter Data Losses
S-0-1050.1.12	Error Counter Data Losses

These IDNs may be removed from this product when and if the SERCON100S is updated to support devices without synchronization.

5.2 Manufacturer Specific IDNs

5.2.1 Translation of ADIs to IDNs

ADIs are accessible from the network as manufacturer specific IDNs. 4096 ADIs are accessible in each of the eight manufacturer specific parameter sets, for at total of 32767 possible ADIs (IDN/P-0-0000 is unused).

An IDN is addressed with 32 bits and an ADI with 16. ADIs with numbers up to 32767 are possible to address from the network, but only up to total of 16319 can be accessed. Translation between IDN numbers and ADI numbers is done as follows:

Structure Instance								Structure Element								S/P	Parameter Set			Data Block Number												
31							24	23							16	15	14		12	11												0
0								0								1	ADI number															

Structure of IDN number:

IDN/(S/P)-(Parameter Set)-(Data Block Number).(Structure Instance).(Structure Element)

Examples:

ADI number	IDN number
1	IDN/P-0-0001.0.0
2	IDN/P-0-0002.0.0
3 : 4095	IDN/P-0-0002.0.0 : IDN/P-0-4095.0.0
4096	IDN/P-1-0000.0.0
4097 : 32767	IDN/P-1-0001.0.0 : IDN/P-7-4095.0.0

S	Standard IDN
P	Manufacturer specific IDN

5.2.2 Translation of Adi Properties

ADI attributes are translated to SERCOS IDN elements.

IDN element		Description	
No.	**Name**		
1	IDN		
2	Name	The name of the ADI translated to UTF8, and truncated to 60 bytes. String length information is prepended.	
3	Attribute	Bits:	Description:
		28-30	set for read-only ADIs
		24-27	0000h (decimal point)
		20-22	see data type translation, *Data Format, p. 102*
		19 0	(not a procedure command)
		16-18	see data type translation, *Data Format, p. 102*
		0-15	0001h (conversion factor)
4	Unit	(not supported)	
5	Minimum input value	ADI Min value converted to SERCOS type. Not supported for ADIs of SERCOS list type.	

IDN element		Description
No.	**Name**	
6	Maximum input value	ADI Max value converted to SERCOS type. Not supported for ADIs of SERCOS list type.
7	Operation data	Value of the ADI translated according to data type translation, page *Data Format, p. 102*

6 FTP Server

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 eight concurrent clients.

The module also supports TFTP.

6.1 User Accounts

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

File Format:

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

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

File Format (\ftp.cfg):

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

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

File Format:

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

Notes:

- Usernames must not exceed 15 characters in length.
- Passwords must not exceed 15 characters in length.
- Usernames and passwords must only contain alphanumeric characters.
- 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).

6.2 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 CompactCom 30
   – - Substitute <user> with the username
   – - Substitute <password> with the password
3. Press **Enter**. The Explorer will now attempt to connect to the Anybus CompactCom 30 using the specified settings. If successful, the file system will be displayed in the Explorer window.

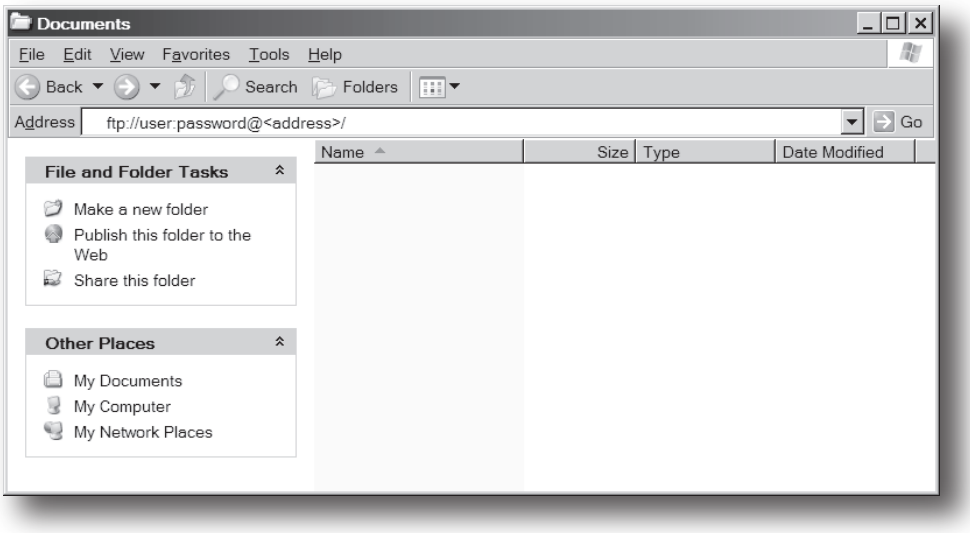


Fig. 1

7 Web Server

7.1 General Information

The built-in web server provides a flexible environment for end-user interaction and configuration purposes. 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. 21*
- *Server Side Include (SSI), p. 30*
- *Ethernet Host Object (F9h), p. 96*

7.2 Default Web Pages

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

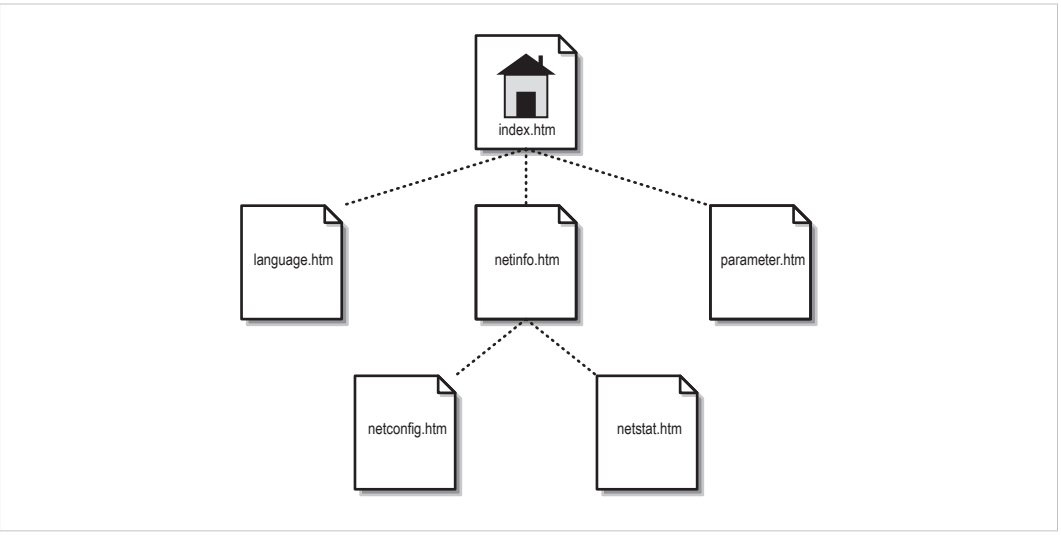


Fig. 2

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

The files are:

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

If none of these files are used, it is recommended to completely disable the virtual file system altogether in the File System Interface Object.

See also ...

- *File System, p. 15*
- *File System Interface Object (0Ah), p. 81*

7.2.1 Network Statistics Page

The Ethernet statistics web page contains the following information:

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

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

Media Counters	Description
Alignment Errors	Packets with an invalid CRC on the last full octet, and an odd number of nibbles have been received (Dribble nibble condition with a bad CRC).
FCS Errors	This counter is incremented for each packet received with a Frame Check Sequence error (bad CRC).
Frame Too Long	This counter is incremented for each packet received with greater than the 802.3 standard maximum length of 1518 bytes.
Runt Packets:	The size of the received packet was less than 64 bytes (inc. CRC).

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 Server Name **`[ServerName]`**	Configures the web server name included in the HTTP header of the responses from the module.
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 below
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. 24*

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.

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.
```

The field “AuthType” is used to identify the authentication scheme.

8 E-Mail Client

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

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

See also...

- *Network Configuration Object (04h), p. 54*, Instance #3, SERCOS Address
- *SMTP Client Object (09h), p. 76*

8.1 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 command Create (03h) in the SMTP Client Object.
2. Specify the sender, recipient, topic and message body in the e-mail instance.
3. Issue the command Send Instance Email (10h) towards the e-mail instance.
4. Delete the e-mail instance using the command Delete (04h) (optional).

Sending a message based on a file in the file system is achieved using the command Send Email from File. For a description of the file format, see the command details for the command Send Email from File in the *SMTP Client Object (09h), p. 76*.

9 Server Side Include (SSI)

9.1 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.

9.2 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)

9.3 Command Functions

9.3.1 General Information

Command functions executes commands and includes the result.

General Syntax

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

command: Command function, see below

“command” is limited to a maximum of 500 characters.

Command Functions

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

Command	Valid for E-mail Messages
printf()	Yes
scanf()	No

9.3.2 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	*(value of specified key)*
Authentication Error	“Authentication error”
File open error	“Failed to open file ‘*filename*’”
Key not found	“Tag (*key*) not found”

Example

The following SSI...

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

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

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

... returns the string ‘Third’.

9.3.3 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'*filename*'"
Authentication Error	"Authentication error"
File open error	"Failed to open file '*filename*'"
File write error	"Could not store configuration to '*filename*'"

Example

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

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

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

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

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

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

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

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

</BODY>
</HTML>
```

The resulting file (‘\food.txt’) may look somewhat as follows:

```
[Name]
Cliff Barnes

[Food]
Cheese

[Drink]
Beer
```

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

9.3.4 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. 49*

9.3.5 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)

9.3.6 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”.

9.3.7 IncludeFile()

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

Syntax:

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

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

Default Output

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

Example

The following example demonstrates how to use this function.

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

Contents of ‘info.txt’:

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

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

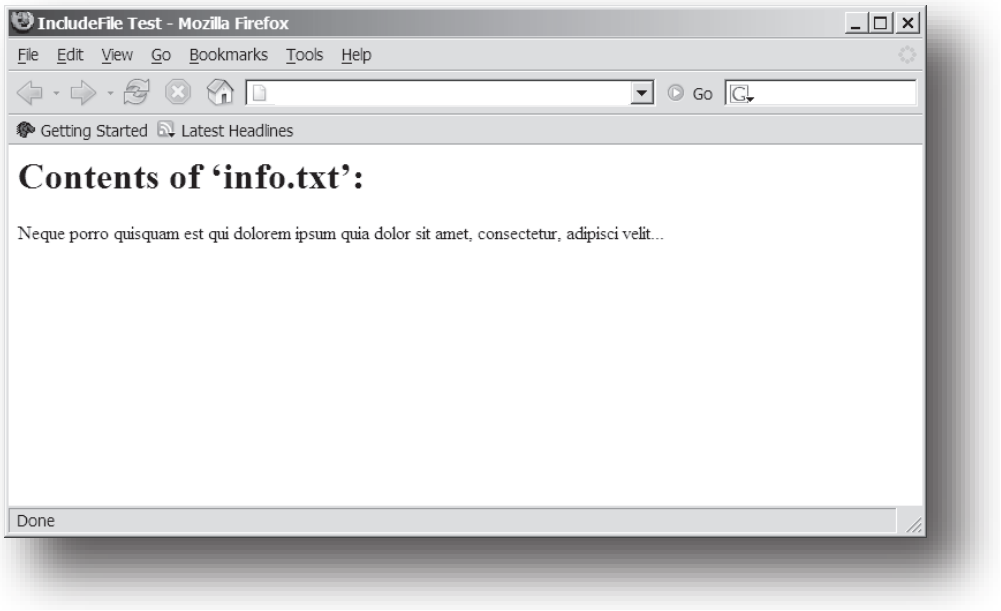


Fig. 3

See also...

- *Include File, p. 30*

9.3.8 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 '*filename*'"
Authentication Error	"Authentication error"
File Write Error	"Could not store configuration to '*filename*'"

Example

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

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

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

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

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

      <LABEL for="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.shtm”.

9.3.9 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... • *ABCCMessage(), p. 45*

Default Output

Scenario	Default Output
Success	(printf() result)
ABCCMessage error	ABCCMessage error string (*Errors, p. 48*)

Example

See ..

- *ABCCMessage(), p. 45*
- *Example (Get_Attribute):, p. 47*

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
h	Argument is interpreted as SINT16 or UINT16
l	Argument is interpreted as SINT32 or UINT32

9.3.10 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... • *ABCCMessage(), p. 45*

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. 48*)

Example

See also...

ABCCMessage(), p. 45

Example (Set_Attribute):, p. 47

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: – 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

9.4 Argument Functions

9.4.1 General Information

Argument functions are supplied as parameters to certain command functions.

General Syntax:

(Syntax depends on context)

Argument Functions:

Function	Description
ABCCMessage()	-

9.4.2 ABCCMessage()

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

Syntax

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

object	Specifies the Destination Object
instance	Specifies the Destination Instance
command	Specifies the Command Number
ce0	Specifies CmdExt[0] for the command message
ce1	Specifies CmdExt[1] for the command message
msgdata	Specifies the actual contents of the MsgData[] subfield in the command • Data can be supplied in direct form (format depends on c_type) • The keyword “ARG” is used when data is supplied by the parent command (e.g. scanf()).
c_type:	Specifies the data type in the command (msgdata), see 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”
16	“Segmentation buffer overflow”
17... 255	“Unknown error”

See also...

SSI Output Configuration, p. 49

9.5 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. 48*
[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. 35*

10 Anybus Module Objects

10.1 General Information

This chapter specifies the Anybus Module Object implementation and how they correspond to the functionality in the Anybus CompactCom 30 SERCOS III.

Standard Objects:

Network Specific Objects:

10.2 Anybus Object (01h)

Category

Basic

Object Description

This object assembles all common Anybus data, and is described thoroughly in the general *Anybus CompactCom 30 Software Design Guide*.

Supported Commands

Object:	Get_Attribute
Instance:	Get_Attribute
	Set_Attribute
	Get_Enum_String

Object Attributes (Instance #0)

(Consult the general *Anybus CompactCom 30 Software Design Guide* for further information.)

Instance Attributes (Instance #1)

Basic

#	Name	Access	Type	Value
1	Module type	Get	UINT16	0401h (Standard Anybus CompactCom 30)
2... 11	-	-	-	Consult the general Anybus CompactCom 30 Software Design Guide for further information.
12	LED colors	Get	struct of: UINT8 (LED1A) UINT8 (LED1B) UINT8 (LED2A) UINT8 (LED2B)	Value: Color: 01h Green 02h Red 01h Green 02h Red
13... 15	-	-	-	Consult the general Anybus CompactCom 30 Software Design Guide for further information.

10.3 Diagnostic Object (02h)

Category

Basic

Object Description

This object provides a standardized way of handling host application events & diagnostics, and is thoroughly described in the general *Anybus CompactCom 30 Software Design Guide*.

The Anybus CompactCom 30 SERCOS III supports up to 6 events/instances. One instance is reserved for major unrecoverable events.

Supported Commands

Object:	Get_Attribute
	Create
	Delete
Instance:	Get_Attribute

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1... 4	-	-	-	Consult the general Anybus CompactCom 30 Software Design Guide for further information.
11	Max no. of instances	Get	UINT16	5+1 (Of the maximum number of instances there should always be one instance reserved for an event of severity level 'Major, unrecoverable', to force the module into the 'EXCEPTION'-state.)

Instance Attributes (Instance #1)

Basic

#	Name	Access	Data Type	Value
1	Severity	Get	UINT8	Consult the general Anybus CompactCom 30 Software Design Guide for further information.
2	Event Code	Get	UINT8	

Major diagnostic events are translated to SERCOS Class 1 diagnostic events if support for latching diagnostic events is included in the SERCOS III application object. Otherwise the major diagnostic events will be translated to class 2 events.

The most recent diagnostic event is presented in IDN/S-0-0390 with the value C10yzz00h, where "zz" is the Anybus CompactCom event code, and "y" is "F" for major events and "E" for minor events.

See also...

- *Diagnostics, p. 13*

10.4 Network Object (03h)

Category

Basic

Object Description

For more information regarding this object, consult the general *Anybus CompactCom 30 Software Design Guide*.

Supported Commands

Object:	Get_Attribute
Instance:	Get_Attribute
	Set_Attribute
	Get_Enum_String
	Map_ADI_Write_Area
	Map_ADI_Read_Area

Object Attributes (Instance #0)

(Consult the general *Anybus CompactCom 30 Software Design Guide* for further information.)

Instance Attributes (Instance #1)

Basic

#	Name	Access	Type	Value
1	Network type	Get	UINT16	009Bh
2	Network type string	Get	Array of CHAR	" SERCOS III"
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. (Consult the general *Anybus CompactCom 30 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. (Consult the general *Anybus CompactCom 30 Software Design Guide* for further information.)
7	Exception Information	Get	UINT8	Additional information available if the module has entered the EXCEPTION state. Value: Meaning: 00h No information available
8... 10	-	-	-	Consult the general *Anybus CompactCom 30 Software Design Guide* for further information.

10.5 Network Configuration Object (04h)

Category

Extended

Object Description

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

The object is described in further detail in the Anybus CompactCom 30 Software Design Guide.

See also...

- *Communication Settings, p. 12*
- Ref till E-mail Client

Supported Commands

Object:	Get_Attribute
	Reset
Instance:	Get_Attribute
	Set_Attribute
	Get_Enum_String

Object Attributes (Instance #0)

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	“Network configuration”
2	Revision	Get	UINT8	01h (first revision)
3	Number of instances	Get	UINT8	0Ch (supported number of instances)
4	Highest instance number	Get	UINT8	0Eh (highest instance number)

(Consult the general *Anybus CompactCom 30 Software Design Guide* for further information.)

Instance Attributes (Instance #3, SERCOS Address)

Changes are valid after reset. IDN/S-0-1040 will be write-protected if this instance is written during SETUP state. It will otherwise be writable in CP2.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	“SERCOS ADR” (Multilingual, see page 58)
2	Data type	Get	UINT8	05h (= UINT16)
3	Number of elements	Get	UINT8	01h (one element)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	UINT16	Valid range: 0 - 511 (Default = 0) The seven most significant bits will be unconditionally masked to zero when this instance is set.

Instance Attributes (Instance #4, IP Address)

Changes are valid after reset.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	“IP address” (Multilingual, see page 58)
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 #5, Subnet Mask)

Changes are valid after reset.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	“Subnet mask” (Multilingual, see page 58)
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 #6, Gateway Address)

Changes are valid after reset.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	“Gateway” (Multilingual, see page 58)
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 #7, DHCP Enable)

Changes are valid after reset.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	“DHCP” (Multilingual, see page 58)
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	(Multilingual, see page 58) Value String Meaning 00h “Disable” DHCP disabled 01h “Enable” DHCP enabled (default)

Instance Attributes (Instance #8, 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 page 58)
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 #9, 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 page 58)
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, 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 page 58)
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 #11, 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 page 58)
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 #12, 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 page 58)
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, dotted decimal format or server name, 64 characters (pad with space to full length)

Instance Attributes (Instance #13, 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 page 58)
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 #14, 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 page 58)
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 password, 64 characters (pad with space to full length)

Instance Attributes (Instance #15, Application Type)

This instance holds the application type according to Sercos III standard IDN S-0-1302.0.3. Changes take effect immediately.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	“Application Type” (Multilingual, see page 58)
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	3Ch (60 elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of CHAR	Application type according to IDN S-0-1302.0.3

Multilingual Strings

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

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

10.6 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. 74*.

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)
	DNS_Lookup (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)
	P_Add_membership (See below)
	IP_Drop_membership (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	0008h (8 instances):

Instance Attributes (Sockets #1...Max. no. of instances)

Extended

#	Name	Access	Data Type	Description
1	Socket Type	Get	UINT8	Value: Socket Type 00h SOCK_STREAM, NONBLOCKING (TCP) 01h SOCK_STREAM, BLOCKING (TCP) 02h SOCK_DGRAM, NONBLOCKING (UDP) 03h SOCK_DGRAM, BLOCKING (UDP)
2	Port	Get	UINT16	Local port that the socket is bound to
3	Host IP	Get	UINT32	Host IP address, or 0 (zero) if not connected
4	Host port	Get	UINT16	Host port number, or 0 (zero) if not connected
5	TCP State	Get	UINT8	State (TCP sockets only): Value State/Description 00h CLOSED Closed 01h LISTEN Listening for connection 02h SYN_SENT Active, have sent and received SYN 03h SYN_RECEIVED Have sent and received SYN 04h ESTABLISHED Established. 05h CLOSE_WAIT Received FIN, waiting for close 06h FIN_WAIT_1 Have closed, sent FIN 07h CLOSING Closed exchanged FIN; await FIN ACK 08h LAST_ACK Have FIN and close; await FIN ACK 09h FIN_WAIT_2 Have closed, FIN is acknowledged 0Ah TIME_WAIT Quiet wait after close
6	TCP RX bytes	Get	UINT16	Number of bytes in RX buffers (TCP sockets only)
7	TCP TX bytes	Get	UINT16	Number of bytes in TX buffers (TCP sockets only)
8	Reuse address	Get/Set	BOOL	Socket can reuse local address Value Meaning 1 Enabled 0 Disabled (default)
9	Keep alive	Get/Set	BOOL	Protocol probes idle connection (TCP sockets only). 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 75 s. The connection is terminated after 8 failed probe attempts. Value Meaning 1 Enabled 0 Disabled (default)
10	IP Multicast TTL	Get/Set	UINT8	IP Multicast TTL value (UDP sockets only). Default = 1.
11	IP Multicast Loop	Get/Set	BOOL	IP multicast loop back (UDP sockets only) Must belong to group in order to get the loop backed message Value Meaning 1 Enabled (default) 0 Disabled
12	Ack delay time	Get/Set	UINT16	Time for delayed ACKs in ms (TCP sockets only) Default: 200 ms Resolution is 50 ms, i.e. 50...99 = 50 ms, 100...149 = 100 ms, 199 = 150 ms etc

#	Name	Access	Data Type	Description
13	TCP No Delay	Get/Set	BOOL	Don't delay send to coalesce packets (TCP). Value Meaning 1 Delay (default) 0 Don't delay (turn off Nagle's algorithm on socket)
14	TCP Connect Timeout	Get/Set	UINT16	TCP Connect time out in seconds (default = 75 s)

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

- Response Details

  (no data)

The recommended sequence to 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.

Host initiates shutdown:

1. Receive data on socket, if zero bytes received it indicates that the host closed the receive channel of the socket.
2. Try to send any unsent data to the host.
3. Send shutdown with CmdExt[1] set to 01h. This will send FIN-flag to host shutting down the send 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. 74*).

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. 74*
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. 74*).

Field	Contents	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control bits	For more information, see *Message Segmentation, p. 74*
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. 74*).

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. 74*
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. 74*).

Field	Contents	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control bits	For more information, see *Message Segmentation, p. 74*
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. 74*).

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. 74*).

Field	Contents	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control	(For more information, see *Message Segmentation, p. 74*)
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. 74*).

- 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. 74*).

Field	Contents	Comments
CmdExt[0]	(reserved)	(set to zero)
CmdExt[1]	Segmentation Control	For more information, see *Message Segmentation, p. 74*
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_Lookup

Category

Extended

Details

Command Code	1Bh
Valid for:	Object

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	EHAVEOOB	Out of band data available
18	ENOMEM	No internal memory available
19	EADDRNOTAVAIL	Address is not available
20	EADDRINUSE	Address already in use
21	(reserved)	-
22	EINPROGRESS	Service already in progress
28	ETOOMANYREFS	Too many references
101	Command aborted	If a command is blocking on a socket, and that socket is closed using the Delete command, this error code will be returned to the blocking command.
102	DNS name error	Failed to resolve the host name (name error response from DNS server.
103	DNS timeout	Timeout when performing a DNS lookup.
104	DNS command failed	Other DNS error.

Message Segmentation

General

Category: Extended

The maximum message size supported by the Anybus CompactCom 30 is 255 bytes. To ensure support for socket interface messages larger than 255 bytes a segmentation protocol is used.

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 30 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 255 bytes), both FS and LS must be set.
- The last response message contains the actual result of the operation.
- The command initiator may at any time abort the operation by issuing a message with AB set.
- If a segmentation error is detected during transmission, an error message is returned, and the current segmentation message is discarded. Note however that this only applies to the current segment; previously transmitted segments are still valid.

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)

10.7 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:	Instance

Description

This command sends the specified e-mail instance.

- Command Details

  (no data)

- Response Details

  (no data)

Object Specific Error Codes

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

10.8 File System Interface Object (0Ah)

Category

Extended

Object Description

This object provides an interface to the built-in file system. Each instance represents a handle to a file stream and contains services for file system operations. This provides the host application with access to the built-in file system of the module. Instances are created and deleted dynamically during runtime.

Supported Commands

Object:	Get_Attribute (01h)
	Create (03h)
	Delete (04h)
	FormatDisc (30h)
Instance:	Get_Attribute (01h)
	File Open (10h)
	File Close (11h)
	File Delete (12h)
	File Copy (13h)
	File Rename (14h)
	File Read (15h)
	File Write (16h)
	Directory Open (20h)
	Directory Close (21h)
	Directory Delete (22h)
	Directory Read (23h)
	Directory Create (24h)
	Directory Change (25h)

Object Attributes (Instance #0)

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

Instance Attributes (Instance #1... 4)

#	Name	Access	Category	Type	Description
1	Instance Type	Get	Extended	UINT8	Value: Meaning: 0: Reserved 1: File instance 2: Directory instance
2	File size	Get	Extended	UINT32	File size in bytes (0 for a directory)
3	Path	Get	Extended	Array of CHAR	The file path to where the instance operates

File System Errors

In case of errors for services calling the file system interface object, the module will return FFh (object specific error). A descriptive file system error will be returned in the error response data field.

#	Name	Description
1	FILE_OPEN_FAILED	Could not open file
2	FILE_CLOSE_FAILED	Could not close file
3	FILE_DELETE_FAILED	Could not delete file
4	DIRECTORY_OPEN_FAILED	Could not open directory
5	DIRECTORY_CLOSE_FAILED	Could not close directory
6	DIRECTORY_CREATE_FAILED	Could not create directory
7	DIRECTORY_DELETE_FAILED	Could not delete directory
8	DIRECTORY_CHANGE_FAILED	Could not change directory
9	FILE_COPY_OPEN_READ_FAILED	Could not open file for copy
10	FILE_COPY_OPEN_WRITE_FAILED	Could not open file for destination
11	FILE_COPY_WRITE_FAILED	Could not write file when copying
12	FILE_RENAME_FAILED	Could not rename file

Command Details: File Open

Details

Command Code:	10h
Valid for:	Instance

Description

Opens a file for reading, writing or appending.

- Command details:

Field	Contents	Comment
CmdExt[0]	00h - Read mode	Opens a file for read only access.
	01h - Write mode	Opens a file for write only access. If the specified file does not exist, it will be created. If the specified file already exists, it will be overwritten.
	02h - Append mode	Opens a file for writing at end-of-file. If the specified file does not exist, it will be created. If the specified file exists, any data written to the file will be appended at end-of-file.
CmdExt[1]	(reserved, 0)	-
MsgData[0...n]	Path + filename of the file to open relative to current path	-

- Response details:

  (No data)

Command Details: File Close

Details

Command Code:	11h
Valid for:	Instance

Description

Closes an open file.

- Command details:

  (No data)

- Response details:

Field	Contents	Comment
CmdExt[0]	Reserved	(ignore)
CmdExt[1]	Reserved	(ignore)
MsgData[0]	File size (low byte)	The size of the closed file
MsgData[1]	File size	
MsgData[2]	File size	
MsgData[3]	File size (high byte)	

Command Details: File Delete

Details

Command Code:	12h
Valid for:	Instance

Description

Deletes the specified file.

- Command details:

Field	Contents
CmdExt[0]	(reserved, 0)
CmdExt[1]	(reserved, 0)
MsgData[0...n]	Path + filename of the file to delete relative to current path

- Response details:

  (No data)

Command Details: File Copy

Details

Command Code:	13h
Valid for:	Instance

Description

Makes a copy of a file.

- Command details:

Field	Contents
CmdExt[0]	(reserved, 0)
CmdExt[1]	(reserved, 0)
MsgData[0]...	Path + filename of the source file, relative to the current path
MsgData[x]	NULL (00h)
MsgData[y]...	Path + filename of the destination file, relative to the current path

- Response details:

  (No data)

Command Details: File Rename

Details

Command Code:	14h
Valid for:	Instance

Description

Renames or moves a file.

- Command details:

Field	Contents
CmdExt[0]	(reserved, 0)
CmdExt[1]	(reserved, 0)
MsgData[0]...	Old path + filename, relative to the current path
MsgData[x]	NULL (00h)
MsgData[y]...	New path + filename, relative to the current path

- Response details:

  (No data)

Command Details: File Read

Details

Command Code:	15h
Valid for:	Instance

Description

Reads data from a file open for reading.

- Command details:

Field	Contents
CmdExt[0]	The number of bytes to read
CmdExt[1]	(reserved, set to zero)

- Response details:

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

Command Details: File Write

Details

Command Code:	16h
Valid for:	Instance

Description

Writes data to a file open for writing or appending.

- Command details:

Field	Contents
CmdExt[0]	(reserved, 0)
CmdExt[1]	(reserved, 0)
MsgData[0]...	Data to write from the file

- Response details:

Field	Contents
CmdExt[0]	No of bytes written
CmdExt[1]	(reserved, set to zero)

Command Details: Directory Open

Details

Command Code:	20h
Valid for:	Instance

Description

Opens a directory.

- Command details:

Field	Contents
CmdExt[0]	(reserved, 0)
CmdExt[1]	(reserved, 0)
MsgData[0]...	Path + name to the directory to open relative to the current path

- Response details:

  (No data)

Command Details: Directory Close

Details

Command Code:	21h
Valid for:	Instance

Description

Opens a directory.

- Command details:

  (No data)
- Response details:

  (No data)

Command Details: Directory Delete

Details

Command Code:	22h
Valid for:	Instance

Description

Deletes a directory in the file system. The directory must be empty to be deleted. An attempt to delete a directory that is not empty will result in an error.

- Command details:

Field	Contents
CmdExt[0]	(reserved, 0)
CmdExt[1]	(reserved, 0)
MsgData[0]...	Path + name to the directory to delete, relative to the current path

- Response details:

  (No data)

Command Details: Directory Read

Details

Command Code:	23h
Valid for:	Instance

Description

This command reads data from a directory previously opened for reading by the Directory Open command.

For each command sent the next directory entry (file or directory) is returned. When all entries in the directory have been read, the response data size will be set to zero (0) and no message data will be returned, to indicate that no more entries exist in the directory.

- Command details:

  (No data)

- Response details:

Field	Contents
CmdExt[0]	Reserved (0)
CmdExt[1]	Reserved (0)
MsgData[0]	Size of object (low byte)
MsgData[1]	Size of object
MsgData[2]	Size of object
MsgData[3]	Size of object (high byte)
MsgData[4]	Object flags
MsgData[5]...	Object name (file or directory)

- Object Flags

Field	Contents
01h	The object is a directory
02h	The object is read only
04h	The object is hidden
08h	The object is a system object

Command Details: Directory Create

Details

Command Code:	24h
Valid for:	Instance

Description

Creates a directory in the file system.

- Command details:

Field	Contents
CmdExt[0]	(reserved, 0)
CmdExt[1]	(reserved, 0)
MsgData[0]...	Path + name to the directory to create, relative to the current path

- Response details:

  (No data)

Command Details: Directory Change

Details

Command Code:	25h
Valid for:	Instance

Description

Change directory/path of the instance.

- Command details:

Field	Contents
CmdExt[0]	(reserved, 0)
CmdExt[1]	(reserved, 0)
MsgData[0]...	Path + name to the directory to change to, relative to the current path

- Response details:

  (No data)

Command Details: Format Disc

Details

Command Code:	30h
Valid for:	Object

Description

Formats a disc in the file system (will erase all data on the disc).

- Command details:

Field	Contents
CmdExt[0]	Disc to format. Set to zero (0)
CmdExt[1]	(reserved, 0)

- Response details:

  (No data)

10.9 Network Ethernet Object (0Ch)

Category

Extended

Object Description

This object provides Ethernet-specific information to the application.

Supported Commands

Object:	Get_Attribute
Instance:	Get_Attribute

Object Attributes (Instance #0)

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

Instance Attributes (Instance #1)

#	Name	Access	Data Type	Description
1	MAC Address	Get	Array of UINT8	Current MAC address. See also “Ethernet Host Object (F9h)”

11 Host Application Objects

11.1 General Information

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

Standard Objects:

- Application Object (FFh) - (see Anybus CompactCom 30 Software Design Guide)
- Application Data Object (FEh) - (see Anybus CompactCom 30 Software Design Guide)

Network Specific Objects:

- *SERCOS III Host Object (F1h), p. 93*
- *Ethernet Host Object (F9h), p. 96*

11.2 SERCOS III Host Object (F1h)

Category

Basic, Extended

Object Description

This object implements SERCOS specific features 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]”).

Note that some of the commands used when accessing this object may require segmentation. For more information, see *Message Segmentation, p. 74*.

See also ...

- *Anybus Object (01h), p. 51*
- *Network Configuration Object (04h), p. 54* (CIP object)
- Anybus CompactCom 30 Software Design Guide, “Error Codes”

Supported Commands

Object:	Reset_Diagnostic
Instance:	-

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1	Name	Get	Array of CHAR	“SERCOS III”
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	Data Type	Default Value	Comment
1	Component name	Get	Array of CHAR	“Communication adapter”	Type of device
2	Vendor code	Get	UINT16	005Ah	Vendor code allocated by SERCOS international
3	Device name	Get	Array of CHAR	“Anybus CompactCom 30 SERCOS III”	
4	Vendor device ID	Get	Array of CHAR	“ABCC SRC3”	
5	Software revision	Get	Array of CHAR	(revision)	
6	Serial number	Get	Array of CHAR	(set at production)	

Extended

#	Name	Access	Data Type	Default Value	Comment
7	Major diagnostic events are latching	Get	BOOL	FALSE	This attribute must be set to TRUE if major diagnostic events shall be translated into SERCOS class 1 events. When the cause for a major diagnostic event is gone, the application shall not delete the diagnostic event. Instead the module will send a Reset_Diagnostic request to the application to ask permission to delete the diagnostic event.

All attributes of type CHAR will be translated to UTF8 and truncated to 32 bytes, which may not be 32 UTF characters.

Command Details: Reset_Diagnostic

Category

Extended

Details

Command Code:	10h
Valid for:	Object

Description

This command is sent to the application when the Anybus CompactCom 30 SERCOS III has received a Reset C1D command from the SERCOS III network.

- Command Details

Field	Contents	Comments
MsgData[0... n]	List of diagnostic instances. (UINT16)	List of diagnostic instance numbers which the Anybus module requests permission to delete. The list may be zero in size.

- Response Details

Field	Contents	Comments
MsgData[0... n]	List of diagnostic instances.(UINT16)	List of diagnostic instance numbers which the Anybus module is permitted to delete.

See also...

- *Diagnostic Object (02h), p. 52*

11.3 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	01h
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.

#	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)				
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.

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	IP conflict	1 = IP address conflict detected 0 = IP address conflict not detected
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)

This page intentionally left blank

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 SERCOS III, this bit is set when the SERCOS III slave state machine is in “operating level”, meaning state is CP2 .. CP4 and “operation state outputs” bit in IO Control word is set.

B.2 Anybus State Machine

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

Anybus State	Corresponding SERCOS III State	Comment
WAIT_PROCESS	NRT CP0, CP1 or CP2 HP0, HP1 or HP2	-
ERROR	NRT or CP0 (and bit 15 in DeviceStatus is set)	Communication timeout or similar
PROCESS_ACTIVE	CP4 and "Producer ready" in C-Con is set and the "operation state outputs" bit in the IO Control word is set	
IDLE	CP3 or CP4 and "Producer ready" in C-Con or "operation state outputs" bit in the IO Control word are cleared	
EXCEPTION	Fieldbus hardware disabled	

C HICP (Host IP Configuration Protocol)

The Anybus CompactCom 30 SERCOS III supports the HICP protocol used by the Anybus IPconfig utility for changing settings, e.g. IP address, Subnet mask, and enable/disable DHCP. Anybus IPconfig can be downloaded free of charge from www.anybus.com/support. This utility may be used to access the network settings of any Anybus product connected to the network via UDP port 3250.

C.1 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.

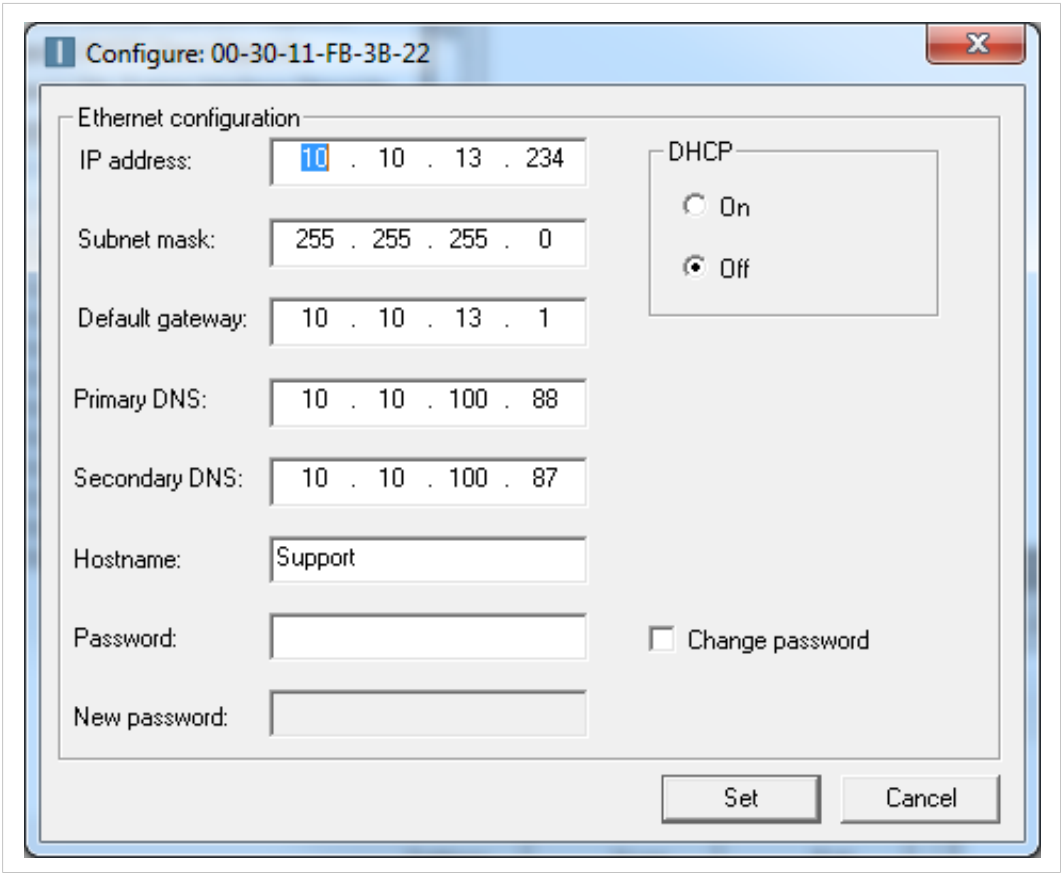


Fig. 4

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.

D Conversion Tables

D.1 Data Format

Data is translated between the native network format and the Anybus data format according to this table:

Anybus Data Type	Corresponding SERCOS Data Type	
	Elements == 1	**Elements != 1**
BOOL	1 octet unsigned decimal list	
SINT8	1 octet signed decimal list	
SINT16	2 octet signed decimal	2 octet signed decimal list
SINT32	4 octet signed decimal	4 octet signed decimal list
UINT8	1 octet unsigned decimal list	
UINT16	2 octet unsigned decimal	2 octet unsigned decimal list
UINT32	4 octet unsigned decimal	4 octet unsigned decimal list
CHAR (Character set ISO 8859-1 (Latin-1))	1 octet list text (UTF8)	
ENUM	1 octet unsigned decimal list	
SINT64	8 octet signed decimal	8 octet signed decimal list
UINT64	8 octet unsigned decimal	8 octet unsigned decimal list
FLOAT	4 octet float	4 octet float list

All ADI fields of type CHAR will automatically be translated to UTF8 on SERCOS III.

IDN “maximum length” field for ADI of type CHAR will be set to the ADI number of elements times two and truncated to 255, to account for the effects of Latin1->UTF8 conversion.

D.2 Language Codes

SERCOS Language Code	Language	ABCC Language Code
0	German	1
1	English	0
2	French	4
3	Spanish	2
4	Italian	3

D.3 Error Codes

Error codes are translated between SERCOS and the Anybus CompactCom status code according to this table:

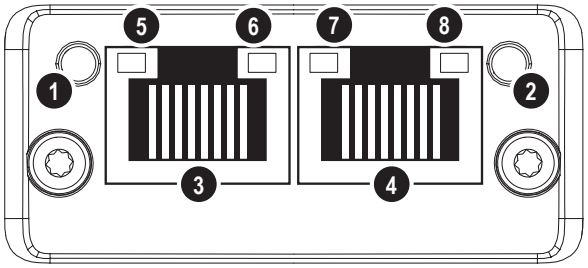
ABCC#	ABCC Status Code	SERCOS Code	SERCOS Description
00h	(reserved)	X001	No X
01h	(reserved)	X001	No X
02h	Invalid message format	X001	No X
03h	Unsupported object	X001	No X
04h	Unsupported instance	X001	No X
05h	Unsupported command	X001	No X
06h	Invalid CmdExt[0]	X001	No X
07h	Invalid CmdExt[1]	X001	No X
08h	Attribute not settable	X004	X cannot be changed (read only)
09h	Attribute not gettable	X001	No X
0Ah	Too much data	X003	X transmission too long
0Bh	Not enough data	X002	X transmission too short
0Ch	Out of range	X008	X invalid operation data
0Dh	Invalid state	X005	X is write-protected at this communication phase
0Eh	Out of resources	X001	No X
0Fh	Segmentation failure	X001	No X
10h	Segmentation buffer overflow	X001	No X
FFh	Object specific error	X001	No X
Other	-	X001	No X

E Technical Specification

E.1 Front View

E.1.1 Front View (Ethernet Connector)

#	Item
1	Sercos III LED
2	Module Status LED
3	SERCOS III (port 1)
4	SERCOS III (port 2)
5	Link (port 1)
6	Activity (port 1)
7	Link (port 2)
8	Activity (port 2)



E.1.2 SERCOS III LED

This LED reflects the status of the network communication

State	Indication
Off	Not online / No power / EXCEPTION
Flashing Orange (4 Hz)	Identification
Orange	CP0 .. CP3
Flashing Red (4 Hz)	Communication error
Red	Class 1 diagnostic or FATAL event
Flashing Green (4 Hz)	Loopback
Green	CP4, On-line
Alternating Green/Red (4 Hz)	Communication error

Please contact HMS support in case of FATAL event (Both SERCOS III LED and Module Status LED are red)

E.1.3 Module Status LED

This LED indicates the status of the Anybus module.

State	Indication
Off	Operating in normal condition
Red	EXCEPTION or FATAL event

Please contact HMS support in case of FATAL event (Both SERCOS III LED and Module Status LED are red)

E.1.4 Link LEDs

Link (port 1) and Link (port 2) LEDs show the status of the respective links.

State	Indication
Off	No link
Red	Link

E.1.5 Activity LEDs

Activity (port 1) and Activity (port 2) LEDsshow the status of the respective links.

State	Indication
Off	No link
Red	Link

E.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 *Anybus CompactCom 30 Hardware Design Guide*. Proper EMC behavior is not guaranteed unless these FE requirements are fulfilled.

E.3 Power Supply

E.3.1 Supply Voltage

The Anybus CompactCom 30 SERCOS III requires a regulated 3.3 V power source as specified in the general *Anybus CompactCom 30 Hardware Design Guide*.

E.3.2 Power Consumption

TheAnybus CompactCom 30 SERCOS III is designed to fulfil the requirements of a ClassB module. For more information about the power consumption classification used on the Anybus CompactCom platform, consult the general Anybus-CompactCom 30 Hardware Design Guide.

The current hardware design consumes up to 400 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 30 Hardware Design Guide, and not on the exact power requirements of a single product.

In line with HMS Industrial Networks policy of continuous product development, we reserve the right to change the exact power requirements of this product without prior notification. However, in any case, the Anybus CompactCom 30 SERCOS III will remain as a Class B module.

E.4 Environmental Specification

Consult the *Anybus CompactCom 30 Hardware Design Guide* for further information.

E.5 EMC Compliance

Consult the *Anybus CompactCom 30 Hardware Design Guide* for further information.

F Timing & Performance

F.1 General Information

This chapter specifies timing and performance parameters that are verified and documented for the Anybus CompactCom 30 SERCOS III.

Category	Parameters	Comment
Startup Delay	T1, T2	Please consult the Anybus CompactCom Software 30 Design Guide, App. B.
NW_INIT Delay	T3	
Telegram Delay	T4	
Command Delay	T5	
Anybus Read Process Data Delay (Anybus Delay)	T6, T7, T8	
Anybus Write Process Data Delay (Anybus Delay)	T12, T13, T14	
Network System Read Process Data Delay (Network System Delay)	T9, T10, T11	
Network System Write Process Data Delay (Network System Delay)	T15, T16, T17	

For further information, please consult the Anybus CompactCom 30 Software Design Guide.

F.2 Process Data

F.2.1 Overview

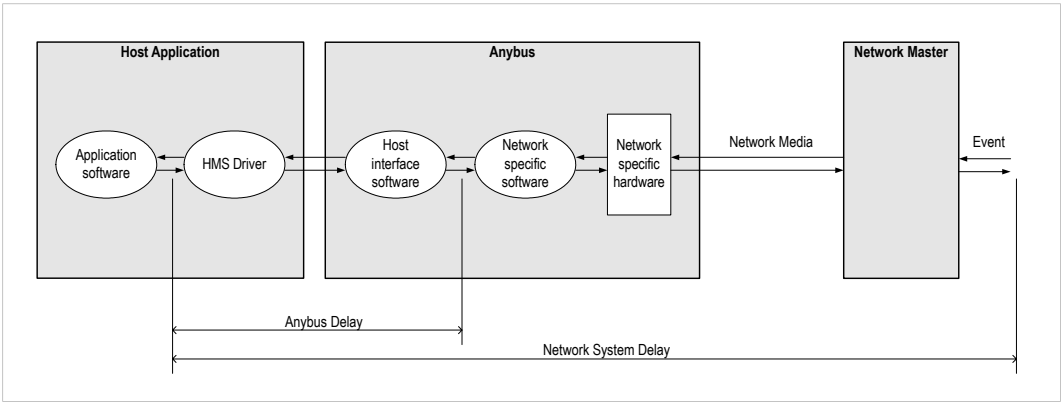


Fig. 5

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

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

Please consult the Anybus CompactCom Software 30 Design Guide, Appendix B, for more information.

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

The Write Process Data Delay (labelled “Anybus delay” in the figure) is defined as the time measured from the point the data is available from the host application (just before the data is

written from the host application to the driver), to the point where the new data has been forwarded to the network buffer by the Anybus host interface software.

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

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

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

Parameter	Description	Typ.	Max.	Unit.
T9	Network System Read Process Data delay, 8 ADIs (single UINT8)	3.3	4.35	ms
T10	Network System Read Process Data delay, 16 ADIs (single UINT8)	3.3	5.1	ms
T11	Network System Read Process Data delay, 32 ADIs (single UINT8)	3.3	5.95	ms

Conditions:

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

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

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

Parameter	Description	Min.	Max.	Unit.
T15	Network System Write Process Data delay, 8 ADIs (single UINT8)	3.3	4.35	ms
T16	Network System Write Process Data delay, 16 ADIs (single UINT8)	3.3	5.1	ms
T17	Network System Write Process Data delay, 32 ADIs (single UINT8)	3.3	5.95	ms

Conditions:

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

G Copyright Notices

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

****

Copyright 1986 by Carnegie Mellon.

****

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

****

Copyright (c) 1988 Stephen Deering.

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

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

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

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

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

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

****

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

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

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

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

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

****

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

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

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

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

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

Box 4126
300 04 Halmstad, Sweden

info@hms.se