


Fieldbus Appendix

Anybus-IC EtherNet/IP

Doc.Id. SCM-1200-055
Rev 1.56

HMS Industrial Networks AB


Germany +49 - 721 - 96472 - 0
Japan +81 - 45 - 478 - 5340
Sweden +46 - 35 - 17 29 20
U.S.A. +1 - 312 - 829 - 0601
France +33 - 3 89 32 76 76
Italy +39 - 347 - 00894 - 70
China +86 - 10 - 8532 - 3183



ge-sales@hms-networks.com
jp-sales@hms-networks.com
sales@hms-networks.com
us-sales@hms-networks.com
fr-sales@hms-networks.com
it-sales@hms-networks.com
cn-sales@hms-networks.com



Table of Contents

Preface	About This Document
	How To Use This Document..... P-1
	Important User Information..... P-1
	Related Documents..... P-2
	Document History P-2
	Conventions & Terminology P-3
	Support..... P-3
 Chapter 1	 About the Anybus-IC EtherNet/IP
	General.....1-1
	Features.....1-1
	Fieldbus Conformance Notes1-1
 Chapter 2	 Basic Operation
	General Information2-1
	<i>Software Requirements</i>2-1
	<i>Electronic Data Sheet (EDS)</i>2-1
	Device Identity.....2-1
	General Parameters2-2
	<i>SCI WD Timeout (#24)</i>2-2
	Status Indicators (Fieldbus Specific Output)2-3
	<i>General</i>2-3
	<i>Compliance Notes</i>2-3
	Switches (Fieldbus Specific Input).....2-4
	<i>General</i>2-4
	<i>Compliance Notes</i>2-4
	EtherNet/IP Implementation2-5
	<i>General</i>2-5
	<i>Fieldbus I/O</i>2-5
	<i>Explicit Data (Application Parameters)</i>2-5
	Modbus/TCP Implementation2-6
	<i>General Information</i>2-6
	<i>Supported Function Codes</i>2-6
	<i>Supported Exception Codes</i>2-6
	<i>Register Map</i>2-7
	Filesystem2-8
	<i>General Information</i>2-8
	<i>Filesystem Overview</i>2-9
	<i>System Files</i>2-9

Chapter 3 Network Configuration

Physical Link Settings	3-1
TCP/IP Settings	3-1
IP Access Control.....	3-2
On/Off Line Configuration	3-2
Anybus IPconfig (HICP).....	3-3
ARP Gleaning	3-3

Chapter 4 FTP Server

General	4-1
FTP Connection Example (Windows Explorer).....	4-2

Chapter 5 Telnet Server

General	5-1
General Commands	5-2
Diagnostic Commands	5-3
File System Operations.....	5-3

Chapter 6 Web Server

General	6-1
Authorization	6-2
Content Types.....	6-3

Chapter 7 Email Client

General	7-1
Event-Triggered Messages	7-1
Email Definitions	7-2

Chapter 8 Server Side Include (SSI)

General Information	8-1
Functions	8-2
Changing SSI output	8-9
<i>SSI Output String File</i>	8-9
<i>Temporary SSI Output change</i>	8-10

Chapter 9 CIP Object Implementation

General Information	9-1
Identity Object (01h).....	9-2
<i>General Information</i>	9-2
<i>Class Attributes</i>	9-2
<i>Instance #1 Attributes</i>	9-2
<i>Service Details: Reset Service</i>	9-3
Message Router (02h)	9-4
<i>General Information</i>	9-4
<i>Class Attributes</i>	9-4
<i>Instance Attributes</i>	9-4
Assembly Object (04h)	9-5
<i>General Information</i>	9-5
<i>Class Attributes</i>	9-5
<i>Instance 64h Attributes (Producing Instance)</i>	9-5
<i>Instance 96h Attributes (Consuming Instance)</i>	9-5
<i>Instance C6h Attributes (Heartbeat, Input-Only)</i>	9-5
<i>Instance C7h Attributes (Heartbeat, Listen-Only)</i>	9-5
Connection Manager (06h)	9-6
<i>General Information</i>	9-6
<i>Instance Descriptions</i>	9-6
<i>Class 1 Connection Details</i>	9-6
<i>Class 3 Connection Details</i>	9-7
TCP/IP Interface Object (F5h)	9-8
<i>General Information</i>	9-8
<i>Class Attributes</i>	9-8
<i>Instance Attributes</i>	9-9
Ethernet Link Object (F6h).....	9-10
<i>General Information</i>	9-10
<i>Class Attributes</i>	9-10
<i>Instance Attributes</i>	9-10
<i>Interface Flags</i>	9-11
Vendor Specific Objects.....	9-12

Chapter 10 Fieldbus Interface

General.....	10-1
Application Connector Signals	10-1
RJ45 Pinout	10-1
Typical Implementation.....	10-2
FastJack Connectors	10-2

Chapter 11 Fieldbus Specific Parameters

General.....	11-3
<i>Serial Number (Parameter #141)</i>	11-3
<i>FB Status (Parameter #100)</i>	11-3
<i>MAC address (Parameter #116)</i>	11-4
<i>DIP switch SSC (Parameter #104)</i>	11-4
Communication Settings	11-5
<i>IP address cfg (Parameter #103)</i>	11-5
<i>IP address act (Parameter #105)</i>	11-5
<i>Subnet mask cfg (Parameter #106)</i>	11-5
<i>Subnet mask act (Parameter #107)</i>	11-6
<i>GW address cfg (Parameter #108)</i>	11-6
<i>GW address act (Parameter #109)</i>	11-6
<i>DHCP enable cfg (Parameter #114)</i>	11-7
<i>DHCP enable act (Parameter #115)</i>	11-7
<i>Data rate cfg (Parameter #117)</i>	11-8
<i>Data rate act (Parameter #118)</i>	11-8
<i>Duplex Cfg (Parameter #119)</i>	11-9
<i>Duplex Act (Parameter #120)</i>	11-9
<i>HICP Enable (Parameter #136)</i>	11-10
<i>HICP Password (Parameter #137)</i>	11-10
<i>ARP Gleaning Enable (Parameter #144)</i>	11-10
Server Settings.....	11-1
<i>Web Srv Enable (Parameter #121)</i>	11-1
<i>FTP Srv Enable (Parameter #122)</i>	11-1
<i>Telnet Srv Enable (Parameter #123)</i>	11-2
Email Client.....	11-3
<i>SMTP Srv Address (Parameter #126)</i>	11-3
<i>Triggered Emails (Parameter #127)</i>	11-3
<i>SMTP Errors (Parameter #128)</i>	11-3
<i>Send Email (Parameter #129)</i>	11-4
<i>SMTP User Name (Parameter #142)</i>	11-4
<i>SMTP Password (Parameter #143)</i>	11-4
File System.....	11-5
<i>VFS Enable (Parameter #130)</i>	11-5
<i>RAM-Disc Path (Parameter #131)</i>	11-5
<i>Admin Mode Cfg (Parameter #124)</i>	11-6
<i>Admin Mode Act (Parameter #125)</i>	11-6
Modbus/TCP.....	11-7
<i>MB/TCP Conn TO (Parameter #132)</i>	11-7
<i>MB/TCP Enable (Parameter #133)</i>	11-7
<i>In bit size (Parameter #134)</i>	11-8
<i>Out bit size (Parameter #135)</i>	11-8
<i>On / Off line trg (Parameter #138)</i>	11-8
<i>On / Off line time (Parameter #139)</i>	11-9
<i>On / Off line Cmds (Parameter #140)</i>	11-9
EtherNet/IP.....	11-10
<i>EIP Enb Cfg (Parameter #160)</i>	11-10
<i>EIP Enb Act (Parameter #161)</i>	11-10
<i>EIP Strip Status (Parameter #162)</i>	11-11
<i>FB Password (Parameter #102)</i>	11-12
<i>EIP Vendor ID (Parameter #163)</i>	11-12
<i>EIP Device Type (Parameter #164)</i>	11-13

<i>EIP Product Code (Parameter #165)</i>	11-13
<i>EIP Revision (Parameter #166)</i>	11-14
<i>EIP Product Name (Parameter #167)</i>	11-14
Application Parameters (Parameters #200... #299)	11-15

Appendix A Application Parameters

General Information	A-1
Creating an Application Parameter	A-2
<i>Query</i> - “ <i>Application Parameter Object</i> ”	A-2
<i>Response</i> - “ <i>Application Parameter Object</i> ”	A-4
<i>Example</i>	A-5
Mapping an Application Parameter to CIP	A-6
<i>Query</i> - “ <i>CIP Mapping Object</i> ”	A-6
<i>Response</i> - “ <i>CIP Mapping Object</i> ”	A-7
<i>Example</i>	A-7

Appendix B HMS Object Implementation

General Information	B-1
Application Parameter Object (Class 85h)	B-2
<i>General Information</i>	B-2
<i>Class Attributes</i>	B-2
<i>Instance Attributes</i>	B-2
<i>Create (Class Service)</i>	B-3
File System Object (Class 86h)	B-4
<i>General Information</i>	B-4
<i>Class Attributes</i>	B-4
<i>Instance Attributes</i>	B-4
<i>File Open (Class Service)</i>	B-5
<i>File Close (Class Service)</i>	B-5
<i>File Delete (Class Service)</i>	B-6
<i>File Copy (Class Service)</i>	B-6
<i>File Move (Class Service)</i>	B-7
<i>File Rename (Class Service)</i>	B-7
<i>File Read (Instance Service)</i>	B-8
<i>File Write (Instance Service)</i>	B-8
CIP Mapping Object (Class A5h)	B-9
<i>General Information</i>	B-9
<i>Class Attributes</i>	B-9
<i>Instance Attributes</i>	B-9
<i>Create (Class Service)</i>	B-10
Socket Object (Class A6h)	B-11
<i>General Information</i>	B-11
<i>Class Attributes</i>	B-11
<i>Instance Attributes</i>	B-11
<i>Create Socket (Class Service)</i>	B-12
<i>Close Socket (Class Service)</i>	B-12
<i>Bind (Instance Service)</i>	B-13
<i>Listen (Instance Service)</i>	B-13
<i>Accept (Instance Service)</i>	B-14
<i>Connect (Instance Service)</i>	B-14
<i>Receive (Instance Service)</i>	B-15
<i>Receive From (Instance Service)</i>	B-15
<i>Send (Instance Service)</i>	B-16
<i>Send To (Instance Service)</i>	B-16
<i>Socket Errors</i>	B-17

Appendix C Firmware Upgrade

Appendix D Technical Specification

Electrical Specification	D-1
<i>Protective Earth (PE) Requirements</i>	D-1
<i>Power Supply</i>	D-1
Environmental Specification	D-1
EMC Compliance (CE)	D-1

About This Document

How To Use This Document

This document is intended to be used as a supplement to the Anybus-IC Design Guide. The reader of this document is expected to have basic knowledge in modern TCP/IP communications, and communication systems in general.

Please consult the general Anybus-IC Design Guide for further information about the Anybus-IC platform.

Important User Information

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

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

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

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 or liability for actual use based on these examples and illustrations.

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

Related Documents

Document name	Author
Anybus-IC Design Guide	HMS
Open Modbus/TCP Specification	Schneider Automation
RFC 821	Network Working Group
RFC 1918	Network Working Group
ENIP Specifications	ControlNet International and ODVA
-	-

Document History

Summary of Recent Changes (v1.55... v1.56)

Change	Page(s)
Added General Parameter #24	2-2
Added Vendor ID information	1-1
Updated Create request and response services in the Application Parameter Object	B-3
Corrected Fieldbus I/O section	2-5

Revision List

Revision	Date	Author(s)	Chapter(s)	Description
<=1.12	-	-	-	(see previous releases)
1.50	2006-09-15	PeP	-	Major rewrite
1.51	2007-01-09	PeP	-	Minor updates
1.52	2007-06-13	PeP	2	Corrected LED indications
1.53	2007-12-14	PeP	2, 8, 9	Minor update
1.54	2008-06-03	PeP	2	Minor update
1.55	2008-11-07	HeS	P, 9, D	Minor update
1.56	2010-02-19	KeL	1, 2, B	Minor update

Conventions & Terminology

The following conventions are used throughout this document:

- Numbered lists provide sequential steps
- Bulleted lists provide information, not procedural steps
- The term ‘module’ is refers to the Anybus module
- The term ‘application’ refers to the device connected to the Anybus application connector
- Hexadecimal values are written in the format NNNNh, where NNNN is the hexadecimal value.
- Binary values are written in the format NNNNb, where NNNN is the binary value.
- 16/32 bit values are written in big endian Motorola format
- Floating point values are in the IEEE Standard 754 format

Support

HMS Sweden (Head Office)

E-mail:	support@hms-networks.com
Phone:	+46 (0) 35 - 17 29 20
Fax:	+46 (0) 35 - 17 29 09
Online:	www.anybus.com

HMS North America

E-mail:	us-support@hms-networks.com
Phone:	+1-312-829-0601
Toll Free:	+1-888-8-Anybus
Fax:	+1-312-738-5873
Online:	www.anybus.com

HMS Germany

E-mail:	ge-support@hms-networks.com
Phone:	+49-721-96472-0
Fax:	+49-721-964-7210
Online:	www.anybus.com

HMS Japan

E-mail:	jp-support@hms-networks.com
Phone:	+81-45-478-5340
Fax:	+81-45-476-0315
Online:	www.anybus.com

HMS China

E-mail:	cn-support@hms-networks.com
Phone:	+86 10 8532 3023
Online:	www.anybus.com

HMS Italy

E-mail:	it-support@hms-networks.com
Phone:	+39 039 59662 27
Fax:	+39 039 59662 31
Online:	www.anybus.com

HMS France

E-mail:	mta@hms-networks.com
Phone:	+33 (0) 3 89 32 76 41
Fax:	+33 (0) 3 89 32 76 31
Online:	www.anybus.com

About the Anybus-IC EtherNet/IP

General

The Anybus-IC EtherNet/IP communication module provides instant integration in any ethernet based lan via SMTP, FTP, Telnet, HTTP as well as EtherNet/IP and Modbus/TCP. Additional protocols can be implemented on top of TCP/IP or UDP using the transparent socket interface.

The exchange of data can be monitored via the built in web server, Modbus/TCP, or using event triggered email messages. SSI (Sever Side Include) technology enables web pages and email messages to carry dynamic content such as I/O data, configuration settings etc.

Features

- 10 and 100mbit operation, Full and Half Duplex
- Up to 144 bytes of fieldbus I/O in each direction
- Adapter Clarr Functionality (EtherNet/IP)
- UCMM Capable, up to 5 explicit server connections
- Modbus/TCP server
- Built-in file system
- Security framework
- Web server
- Email client
- FTP server
- Telnet server
- Server Side Include (SSI) capability
- Device identity customization
- Transparent socket interface
- Application Parameters (used for Explicit Messaging)

Fieldbus Conformance Notes

- The Anybus-IC EtherNet has been tested standalone by ODVA's authorized Independent Test Lab and found to comply with ODVA Conformance Test Software Version A12. However, in accordance with ODVA's conformance test policy, the final product must still be compliance tested to ensure fieldbus conformance. In order to be able to do this, the vendor information in the EtherNet/IP Host Object must be customized.
- It is strongly recommended to customize the information in the Identity Object (CIP), to enable the product to appear as a vendor specific implementation rather than a generic Anybus module. ODVA requires that all manufacturers use their own Vendor ID. A Vendor ID can be applied for from ODVA.

For more information, please contact HMS Industrial Networks or the ODVA.

Basic Operation

General Information

Software Requirements

Generally, no network-specific support code needs to be written in order to support the Anybus-IC EtherNet/IP. Optionally, advanced fieldbus functionality can be enabled using fieldbus-specific parameters.

Electronic Data Sheet (EDS)

On EtherNet/IP, the characteristics of a device is stored in an ASCII data file with the suffix 'EDS'. This file is used by EtherNet/IP, configuration tools etc. when setting up the network.

HMS provides a generic EDS-file, which corresponds to the default settings in the module. However, due to the flexible nature of the Anybus concept, it is possible to alter the behaviour of the product in ways that invalidates the generic EDS-file.

See also...

- 1-1 "Fieldbus Conformance Notes"

Device Identity

On EtherNet/IP, the module identifies itself as follows:

Information	Default Value	Comments
Vendor ID	005Ah	(HMS Industrial Networks)
Device Type	000Ch	(Communications Adapter)
Product Code	0002h	(Anybus-IC)
Revision	-	(reflects Anybus-IC product revision)
Product Name	'Anybus-IC EtherNet/IP'	-

The information above can be customized. Note that in such case, the EDS-file needs to be altered as well. Note that the identity information is protected by a password to prevent unintentional changes. This password can be obtained by contacting HMS.

Note: These settings are stored in non-volatile memory and will not be affected by a reset or a 'Set Defaults'-operation unless the correct password has been supplied using parameter #102 ('FB Password').

See also...

- 11-12 "FB Password (Parameter #102)"
- 11-12 "EIP Vendor ID (Parameter #163)"
- 11-13 "EIP Device Type (Parameter #164)"
- 11-13 "EIP Product Code (Parameter #165)"
- 11-14 "EIP Revision (Parameter #166)"
- 11-14 "EIP Product Name (Parameter #167)"

General Parameters

The general parameters are used to configure the basic settings of the module. Most of these parameters are common to all Anybus-IC modules, and are described in the Anybus-IC Design Guide. The parameter described here, is only applicable to the Anybus IC EIP. For all other general parameters, please consult the Anybus-IC Design Guide.

SCI WD Timeout (#24)

This parameter is used to disable / enable the SCI channel watchdog timeout functionality.

Enable by writing a non-zero value between 1-65535. The value will be the watchdog timeout in milliseconds.

- The setting will be saved in NVS memory and remembered after a power-cycle.
- The initial watchdog will start after the first correctly received SCI Modbus request.
- Writing 0 to the WD parameter will disable the watchdog at once.
- If a watchdog timeout has occurred a power-cycle/module restart is required to recover.

Parameter number	24
Modbus Address	0x601F
Default value	0x0000
Range	0x0000 - 0xFFFF
Size	1 byte
Stored in NV RAM	Yes
Access	R/W

Status Indicators (Fieldbus Specific Output)

General

The Anybus-IC EtherNet/IP uses bi-coloured status indications as follows:

Bit	Activates Colour	LED	Comments
0	Green	Link/Activity	-
1	Red		-
2	Green	Data Rate	-
3	Red		-
4	Green	Module Status	-
5	Red		-
6	Green	Network Status	-
7	Red		-

The standard indications are as follows:

LED	State	Status
Link/Activity	Off	Device not powered
	Green	Module connected to an Ethernet network
	Green, flashing	RX / TX Activity
	Alternating Red/Green	Self test in progress
Data Rate	Off	10 Mbps operation
	Green	100 Mbps operation
	Alternating Red/Green	Self test in progress
Module Status	Off	Device not powered
	Green	Device has an EtherNet/IP connection
	Green, flashing	Device has no EtherNet/IP connection
	Red	Major fault (unrecoverable)
	Red, flashing	Minor fault (recoverable)
	Alternating Red/Green	Self test in progress
Network Status	Off	No power or no IP address
	Green	EtherNet/IP connection(s) established
	Green, flashing	No EtherNet/IP connections established
	Red	Duplicate IP address detected
	Red, flashing	One or several EtherNet/IP connections has timed out
	Alternating Red/Green	Self test in progress

Note: To comply with EtherNet/IP interoperability guidelines, the LED indications has changed slightly from that of software revision 1.05. For further information, contact HMS.

See also...

- Anybus-IC Design Guide (parameter #7 'LED State')

Compliance Notes

The following points should be kept in mind when designing for EtherNet/IP compliance.

- Module Status LED must be labelled 'Module Status' or 'MS'
- Network Status LED must be labelled 'Network Status' or 'NS'

Switches (Fieldbus Specific Input)

General

The Fieldbus Specific Input is used for fieldbus-specific configuration settings and supports two types of switches/coding.

- **BCD-coded Switches**

Two switches are used to specify the last byte of the IP address, one for each decimal digit. Subnet mask and gateway address is fixed to the values below.

Switch A (x10)	Switch B (x1)	IP Address	Subnet Mask	Default Gateway
0	0	(set by software)	(set by software)	(set by software)
0	1	192.168.1.1	255.255.255.0	0.0.0.0
0	2	192.168.1.2	255.255.255.0	0.0.0.0
0	3	192.168.1.3	255.255.255.0	0.0.0.0
...
9	8	192.168.1.98	255.255.255.0	0.0.0.0
9	9	192.168.1.99	255.255.255.0	0.0.0.0

- **Binary Switches**

This type of switch allows a greater address range compared to BCD-coded switches. Subnet mask and gateway address is fixed to the values below.

b7	b6	b5	b4	b3	b2	b1	b0	IP Address	Subnet Mask	Default Gateway
0	0	0	0	0	0	0	0	(set by software)	(set by software)	(set by software)
0	0	0	0	0	0	0	1	192.168.1.1	255.255.255.0	0.0.0.0
0	0	0	0	0	0	1	0	192.168.1.2	255.255.255.0	0.0.0.0
0	0	0	0	0	0	1	1	192.168.1.3	255.255.255.0	0.0.0.0
...
1	1	1	1	1	1	1	0	192.168.1.254	255.255.255.0	0.0.0.0
1	1	1	1	1	1	1	1	(invalid setting)	-	-

Note: The type of switch used is specified in parameter #9 ("Switch Coding"). On the Anybus-IC EtherNet/IP, the default value for this parameter is 01h (Binary Switches). However, since this is fieldbus dependant, there is no guarantee that the same type of switch is used by default on other networks.

See also...

- Anybus-IC Design Guide (parameter #9 "Switch Coding")
- 3-1 "Network Configuration"
- 11-5 "Communication Settings"

Compliance Notes

The following issues should be kept in mind when designing for EtherNet/IP compliance.

- Switches should be placed so that the most significant digit is located to the left or to the top of the product.

EtherNet/IP Implementation

General

EtherNet/IP is based on the Common Industrial Protocol (CIP) which is also the application layer for DeviceNet and ControlNet. The module acts as an Adapter Class product on the EtherNet/IP network.

See also...

- 9-1 “CIP Object Implementation”
- 11-10 “EtherNet/IP”

Fieldbus I/O

On EtherNet/IP, the fieldbus I/O is represented through dedicated instances in the Assembly Object. The module supports Class 1 connections with Cyclic and Change of State production triggers.

See also...

- 9-5 “Assembly Object (04h)”
- 9-6 “Connection Manager (06h)”

Explicit Data (Application Parameters)

Explicit Data is exchanged by means of vendor specific objects mapped to Application Parameters.

See also...

- 9-12 “Vendor Specific Objects”
- A-1 “Application Parameters”
- B-1 “HMS Object Implementation”

Modbus/TCP Implementation

General Information

The Modbus/TCP protocol is an implementation of the standard Modbus protocol running on top of TCP/IP. The built in Modbus/TCP server provides access to the fieldbus I/O via a subset of the functions defined in the Modbus/TCP specification. Up to 8 simultaneous Modbus/TCP connections are supported.

Modbus/TCP messages are exchanged through TCP port no. 502. For detailed information regarding the Modbus/TCP protocol, consult the Open Modbus Specification.

Supported Function Codes

The following function codes are implemented:

Modbus Function	Function Code	Associated with...
Read Coil	1	Output Data (FB Out)
Read Input Discretes	2	Input Data (FB In)
Read Multiple Registers	3	Output Data (FB Out)
Read Input Registers	4	Input Data (FB In)
Write Coil	5	Output Data (FB Out)
Write Single Register	6	
Force Multiple Coils	15	
Force Multiple Registers	16	
Mask Write Register	22	
Read/Write Registers	23	

Supported Exception Codes

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

Register Map

Fieldbus Input Data (FB In)

The Input Data is mapped to Coils & Registers as follows:

Register #	Input Data (FB In) location	Comments
0001 (0513)	000h + Coil Size In	Each register corresponds to two bytes of data.
0002 (0514)	002h + Coil Size In	
0003 (0515)	004h + Coil Size In	
0004 (0516)	006h + Coil Size In	
...	...	
0071 (0583)	08Ch + Coil Size In	
0072 (0584)	08Eh + Coil Size In	

Note: A mirror of the Input Data is available at register 513... 584 to allow efficient I/O scanning by means of Read/Write Registers (function code 13)

Coil #	Input Data (FB In) location	Comments
0001... 0008	000h	Each coil corresponds a single bit of data.
0009... 0016	001h	
0017... 0024	002h	
0025... 0032	003h	
...	...	
1137... 1144	08Eh	Note: Coils are mapped MSB first, i.e. coil #1 corresponds to the most significant bit of byte 000h
1145... 1152	08Fh	

Fieldbus Output Data (FB Out)

The Output Data is mapped to Coils & Registers as follows:

Register #	Output Data (FB Out) location	Comments
0001	000h + Coil Size Out	Each register corresponds to two bytes of data.
0002	002h + Coil Size Out	
0003	004h + Coil Size Out	
0004	006h + Coil Size Out	
...	...	
0071	08Ch + Coil Size In	
0072	08Eh + Coil Size In	

Coil #	Output Data (FB Out) location	Comments
0001... 0008	000h	Each coil corresponds a single bit of data
0009... 0016	001h	
0017... 0024	002h	
0025... 0032	003h	
...	...	
1137... 1144	08Eh	Note: Coils are mapped MSB first, i.e. coil #1 corresponds to the most significant bit of byte 000h
1145... 1152	08Fh	

Filesystem

General Information

The module features a built in filesystem, which is used to store information such as web files, network communication settings, email messages etc.

The filesystem can be accessed using FTP, HTTP and Telnet.

Storage

The filesystem features two storage areas:

- **Non-volatile area (approx. 1Mb)**

This section is intended for static files such as web files, configuration files etc.

- **Volatile area (approx. 1Mb)**

This area is intended for temporary storage; data placed here will be lost in case of power loss or reset. Note that this area is not available by default, and must be mounted by the application during initialisation (see 11-5 “RAM-Disc Path (Parameter #131)”)

Conventions

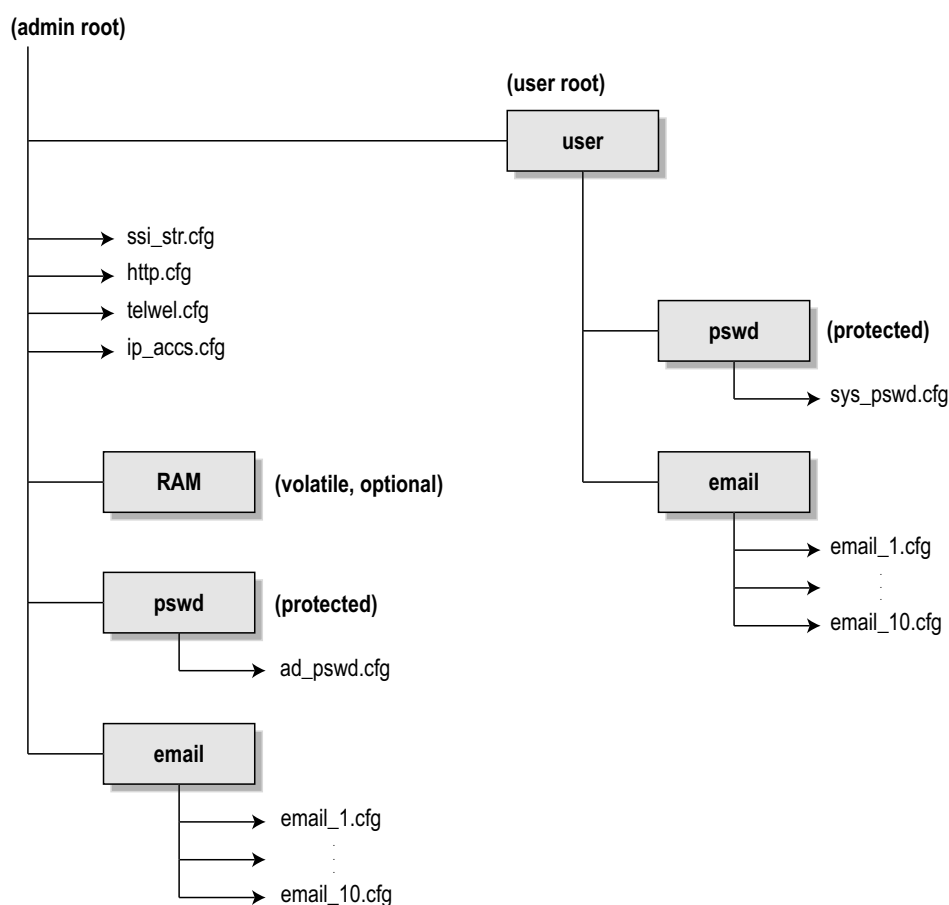
- ‘\’ (backslash) is used as a path separator
- A ‘path’ originates from the system root and as such must begin with a ‘\’
- A ‘path’ must not end with a ‘\’
- Names may contain spaces (‘ ’) but must not begin or end with one.
- Names must not contain one of the following characters: ‘\ / : * ? “ < > | ’
- Names cannot be longer than 48 characters (plus null termination)
- A path cannot be longer than 256 characters (filename included)
- The maximum number of simultaneously open files is 40
- The maximum number of simultaneously open directories is 40

IMPORTANT: *The non-volatile storage is located in FLASH memory. Each FLASH segment can only be erased approximately 1000000 times due to the nature of this type of memory.*

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 filesystem*

Filesystem Overview



System Files

The filesystem 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™). Note that some of these files may also be altered by the Anybus module itself, e.g. when using SSI (see 8-1 “Server Side Include (SSI)”).

The format of the system files are based on the concept of ‘keys’, where each ‘key’ can be assigned a value, see example below.

Example:

```
[Key1]
value of key1

[Key2]
value of key2
```

The exact format of each system file is described in detail later in this document.

Network Configuration

Physical Link Settings

By default, the module uses auto-negotiation to establish the physical link settings. Optionally, it is possible to force a particular settings using parameters #117 and #119.

See also...

- 11-8 “Data rate cfg (Parameter #117)”
- 11-9 “Duplex Cfg (Parameter #119)”

TCP/IP Settings

The module offers three modes of operation regarding the TCP/IP settings:

- **Settings specified by Switches (Fieldbus Specific Input)**

The module will use the switch setting if the NA-bit (Parameter #8 ‘Configuration Bits’) is cleared (0) and the switches are set to a value other than zero (0).

In such case, the module will use the following settings:

IP Address:	192.168.0.x	(x = switch value)
Gateway:	0.0.0.0	(no gateway)
Subnet:	255.255.255.0	
DHCP:	OFF	

See also...

- Anybus-IC Design Guide (Parameter #8 ‘Configuration Bits’)
- 2-4 “Switches (Fieldbus Specific Input)”

- **Settings specified by Parameters**

The module will use the settings stored in the IP configuration parameters if the NA-bit (Parameter #8 ‘Configuration Bits’) is set (1) and/or the switches on the fieldbus specific input register is set to zero.

If no current settings are available (i.e. the IP configuration parameter contains invalid settings), the module will halt and indicate an error on the on-board status LEDs (the settings may however still be altered via HICP or ARP gleaning).

See also...

- Anybus-IC Design Guide (Parameter #8 ‘Configuration Bits’)
- 3-3 “Anybus IPconfig (HICP)”
- 3-3 “ARP Gleaning”
- 11-5 “Communication Settings”

- **Settings specified by network**

The IP configuration can be altered via EtherNet/IP via the TCP/IP Interface Object.

See also...

- Anybus-IC Design Guide (Parameter #8 ‘Configuration Bits’)
- 9-8 “TCP/IP Interface Object (F5h)”

IP Access Control

It is possible to specify which IP addresses that are permitted to connect to the module. This information is stored in the system file ‘\ip_accs.cfg’.

File Format:

[Web] xxx.xxx.xxx.xxx	• Nodes listed here may access the web server
[FTP] xxx.xxx.xxx.xxx	• Nodes listed here may access the FTP server
[Modbus/TCP] xxx.xxx.xxx.xxx	• Nodes listed here may access the module via Modbus/TCP
[EtherNet/IP] xxx.xxx.xxx.xxx	• Nodes listed here may access the module via EtherNet/IP
[All] xxx.xxx.xxx.xxx	• Fallback setting, used by the module when one or several of the keys above are omitted

Note: ‘*’ may be used as a wildcard to select IP series.

On/Off Line Configuration

By default, the On/Off Line indication is triggered by the Link Status. Other triggering options can however be specified in the system file ‘\onoffln.cfg’, which looks as follows:

File Format:

[ON/OFF-line trigger] Modbus	• ON/OFF-line trigger Values: ‘Link’, ‘EIP’ and ‘Modbus’
[Timeout] 10	• Timeout Value: Timeout value. A value of 10 equals 1000ms.
[Commands] 3, 16, 23	• Commands (Optional) Selects what Modbus commands that must be received during the timeout period. If the keyword ‘ALL’ is given, the ON/OFF line functionality will trigger on all Modbus commands.

The keys ‘[Timeout]’ and ‘[Commands]’ shall only given if the ON/OFF-line Trigger value is set to ‘Modbus’.

Note: The settings in this file will be ignored when using parameters #138... 149

See also...

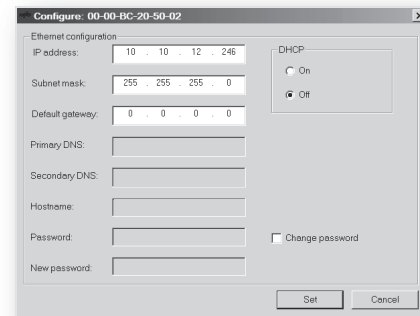
- 11-8 “On / Off line trg (Parameter #138)”
- 11-9 “On / Off line time (Parameter #139)”
- 11-9 “On / Off line Cmds (Parameter #140)”

Anybus IPconfig (HICP)

The module supports the HICP protocol used by the Anybus IPconfig utility from HMS, which can be downloaded free of charge from the HMS website. This utility may be used to configure the network settings of any Anybus product connected to the network. Note that if successful, this will replace the settings currently stored in the corresponding parameters.

Upon starting the program, the network is scanned for Anybus products. The network can be rescanned at any time by clicking 'Scan'. In the list of detected devices, the module will appear as 'ABIC-EIP'. To alter its network settings, double-click on its entry in the list.

A window will appear, containing the IP configuration and password settings. Validate the new settings by clicking 'Set', or click 'Cancel' to abort.



Optionally, the configuration may be protected from unauthorized access by a password. To enter a password, click on the 'Change password' checkbox, and enter the password under 'New password'. When protected, any changes in the configuration requires that the user supplies a valid password.

When done, click 'Set'.

ARP Gleaning

The module supports the Address Resolution Protocol (ARP), allowing the TCP/IP settings to be altered using the ARP-command on a PC.

Syntax:

```
arp -s <IP address> <MAC address>
ping <IP address>
arp -d <IP address>
```

The 'arp -s' command stores the IP and MAC address in the PC's ARP-table. When the 'ping'-command is issued, the PC will address the module with the new IP address; the module recognizes that it was addressed with the correct MAC address and adopts the new IP address from the 'ping' message.

If successful, new settings will be stored as follows:

IP Address:	xxx.xxx.xxx.xxx	(value supplied in ARP command)
Gateway:	0.0.0.0	(no gateway)
Subnet:	255.255.255.0	
DHCP:	OFF	

Note: This functionality may cause problems if multiple devices continuously issue 'ping'-messages towards the module. The reason for this lies in the very nature of this functionality; since the module adopts the IP address from all 'ping'-messages, any additional 'ping'-messages may cause it to change back and forth between old and new settings.

See also...

- 11-10 "ARP Gleaning Enable (Parameter #144)"

FTP Server

General

The built in FTP server provides a way to access the file system using a standard FTP client.

The following port numbers are used for FTP communication:

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

See also...

- 11-1 “FTP Srv Enable (Parameter #122)”

Security Levels

The FTP-server features two security levels; admin and normal.

- **Normal-level users**

The root directory will be ‘\user’.

- **Admin-level users**

The root directory will be ‘\’, i.e. the user has unrestricted access to the file system.

User Accounts

The user accounts are stored in two files, which are protected from web access:

- ‘\user\pswd\sys_pswd.cfg’

This file holds the user accounts for normal-level users.

- ‘\pswd\ad_pswd.cfg’

This file holds the user accounts for admin-level users.

File Format:

The format of these files are as follows:

```
Username1:Password1
Username2:Password2
Username3:Password3
```

Note 1: If no valid user accounts have been defined, or if running in Global Admin Mode, the module will grant Admin-level access to all users. In such case, the FTP accepts any username/password combination, and the root directory will be ‘\’.

Note 2: The FTP server shares user accounts with the Telnet server.

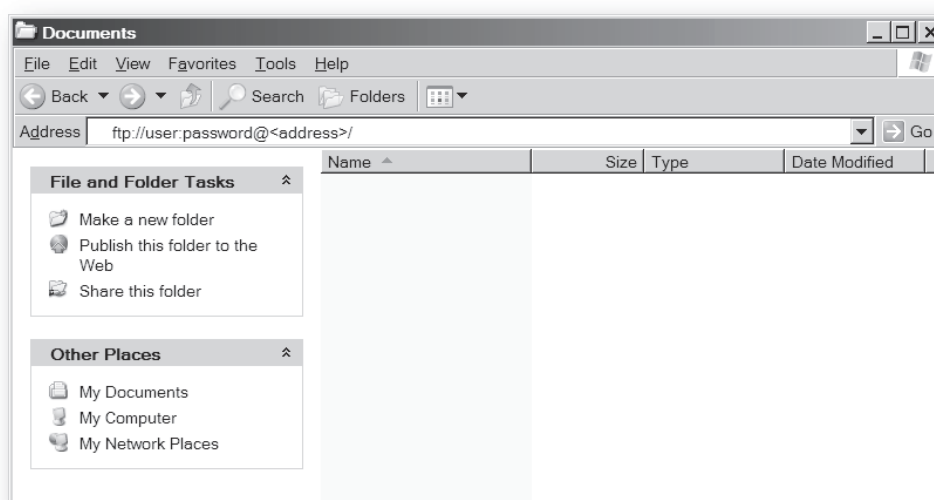
See also...

- 5-1 “Telnet Server”
- 11-6 “Admin Mode Cfg (Parameter #124)”
- 11-6 “Admin Mode Act (Parameter #125)”

FTP Connection Example (Windows Explorer)

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

1. Open the Windows Explorer by right-clicking on the 'Start' button and selecting 'Explore'.
2. In the address field, type FTP://<user>:<password>@<address>
 - Substitute <address> with the IP address of the Anybus module
 - Substitute <user> with the username
 - Substitute <password> with the password
3. Press enter. The Explorer will now attempt to connect to the module using the specified settings. If successful, the built in file system is displayed in the Explorer window.



Telnet Server

General

The built in Telnet server provides a way to access the file system using a standard Telnet client. The server communicates through TCP port 23.

See also...

- 11-2 “Telnet Srv Enable (Parameter #123)”

Security Levels

Just like the FTP server, the Telnet server features two security levels; admin and normal.

- **Normal-level users**
The root directory will be ‘\user’.
- **Admin-level users**
The root directory will be ‘\’, i.e. the user has unrestricted access to the file system.

User Accounts

The Telnet server shares user accounts with the FTP server. If no valid user accounts have been defined, or if running in Global Admin Mode startup, the module will grant Admin-level access to all users. In such case, no login is required, and the root directory will be ‘\’.

See also...

- 4-1 “User Accounts”
- 11-6 “Admin Mode Cfg (Parameter #124)”
- 11-6 “Admin Mode Act (Parameter #125)”

Welcome Message

The default telnet welcome message can be customized by adding the file ‘\telwel.cfg’. This file shall contain the desired message in pure ASCII format.

General Commands

admin

- **Syntax**

admin

- **Description**

Provided that the user can supply a valid admin username/password combination, this command provides Admin access rights to Normal-level users.

exit

- **Syntax**

exit

- **Description**

This command closes the Telnet session.

help

- **Syntax**

help [general|diagnostic|filesystem]

- **Description**

If no argument is specified, the following menu will be displayed.

General commands:

help	- Help with menus
version	- Display version information
exit	- Exit station program

Also try 'help [general|diagnostic|filesystem]'

version

- **Syntax**

version

- **Description**

This command will display version information, serial number and MAC ID of the module.

Diagnostic Commands

arps

- **Syntax**
arps
- **Description**
Display ARP stats and table

iface

- **Syntax**
iface
- **Description**
Display net interface stats

routes

- **Syntax**
routes
- **Description**
Display IP route table

sockets

- **Syntax**
sockets
- **Description**
Display socket list

File System Operations

For commands where filenames, directory names or paths shall be given as an argument the names can be written directly or within quotes. For names including spaces the filenames must be surrounded by quotes. It is also possible to use relative pathnames using '.', '\', and '..'.

append

- **Syntax**
append [file] ["The line to append"]
- **Description**
Appends a line to a file.

cd

- **Syntax**
`cd [path]`
- **Description**
Changes current directory.

copy

- **Syntax**
`copy [source] [destination]`
- **Description**
This command creates a copy of the source file at a specified location.

del

- **Syntax**
`del [file]`
- **Description**
Deletes a file.

dir

- **Syntax**
`dir [path]`
- **Description**
Lists the contents of a directory. If no path is given, the contents of the current directory is listed.

df

- **Syntax**
`df`
- **Description**
Displays filesystem info.

format

- **Syntax**
`format`
- **Description**
Formats the filesystem. This command may only be executed by admin level users.

md

- **Syntax**
`md [directory]`
- **Description**
Creates a directory. If no path is given, the directory is created in the current directory.

mkfile

- **Syntax**
`mkfile [filename]`
- **Description**
Creates an empty file.

move

- **Syntax**
`move [source] [destination]`
- **Description**
This command moves a file or directory from the source location to a specified destination.

rd

- **Syntax**
`rd [directory]`
- **Description**
Removes a directory. The directory can only be removed if it is empty.

ren

- **Syntax**
`ren [old name] [new name]`
- **Description**
Renames a file or directory.

type

- **Syntax**
`type [filename]`
- **Description**
Types the contents of a file.

Web Server

General

The Anybus module features a flexible web server with SSI capabilities. The built in web pages can be customized to fit a particular application and allow access to I/O data and configuration settings.

The web server communicates through port 80.

See also...

- 11-1 “Web Srv Enable (Parameter #121)”

Protected Files

For security reasons, the following files are protected from web access:

- Files located in ‘\user\pswd’
- Files located in ‘\pswd’
- Files located in a directory which contains a file named ‘web_accs.cfg’

Default Web Pages

The Anybus module contains a set of virtual files that can be used when building a web page for configuration of network parameters. These virtual files can be overwritten (not erased) by placing files with the same name in the root of disc 0.

This makes it possible to, for example, replace the HMS logo by uploading a new logo named ‘\logo.jpg’. It is also possible to make links from a web page to the virtual configuration page. In that case the link shall point to ‘\config.htm’.

These virtual files are:

\index.shtm	- includes the content of config.htm
\config.htm	- Configuration frame page
\configform.shtm	- Configuration form page
\store.shtm	- Configuration store page
\logo.jpg	- HMS logo
\configuration.gif	- Configuration picture
\border_bg.gif	- Picture forming a border
\border_m_bg.gif	- Picture forming a border

Note: The virtual file system can be disabled using parameter #130.

See also...

- 11-5 “VFS Enable (Parameter #130)”

Authorization

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

File Format:

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

• List of approved users.


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

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

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

Example:

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

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

[AuthName]
Yeah. Whatsda passwoid?
```

Note that when using this feature, make sure to put the user/password files in a directory that is protected from web access, see 6-1 "Protected Files".

Content Types

By default, the following content types are recognized by their file extension:

Content Type	File Extension
text/html	*.htm, *.html, *.shtm
image/gif	*.gif
image/jpeg	*.jpeg, *.jpg, *.jpe
image/x-png	*.png
application/x-javascript	*.js
text/plain	*.bat, *.txt, *.c, *.h, *.cpp, *.hpp
application/x-zip-compressed	*.zip
application/octet-stream	*.exe, *.com
text/vnd.wap.wml	*.wml
application/vnd.wap.wmlc	*.wmlc
image/vnd.wap.wbmp	*.wbmp
text/vnd.wap.wmlscript	*.wmls
application/vnd.wap.wmlscriptc	*.wmlsc
text/xml	*.xml
application/pdf	*.pdf

It is possible to configure/reconfigure the reported content types, and which files that shall be scanned for SSI. This is done in the system file ‘\http.cfg’.

File Format:

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

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

Note: Up to 50 content types and 50 SSI file types may be specified in this file.

Email Client

General

The built-in email client can send predefined email messages when instructed to do so or based on trigger-events in the fieldbus I/O data.

The client supports SSI; note however that certain SSI-functions cannot be used in email messages (this is specified separately for each SSI-function).

Note: This functionality requires a valid SMTP-server configuration. It is currently not possible to use servers which require authentication.

See also...

- 7-2 “Email Definitions”
- 8-1 “Server Side Include (SSI)”
- 11-3 “SMTP Srv Address (Parameter #126)”
- 11-3 “SMTP Errors (Parameter #128)”
- 11-4 “Send Email (Parameter #129)”

Event-Triggered Messages

The email client can issue predefined messages based on events in the fieldbus I/O data. The fieldbus I/O is scanned once every 0.5 seconds, which means that an event must be present for at least 0.5 seconds to ensure detection by the Anybus module.

In operation, this works as follows:

1. The trigger source is fetched from the fieldbus I/O data
2. A logical AND is performed between the trigger source and a mask value
3. The result is compared to a reference value according to a specified operand
4. If the end result is true, the email is sent to the specified recipient(s).

See also...

- 7-2 “Email Definitions”
- 11-3 “Triggered Emails (Parameter #127)”

Email Definitions

The email definitions are stored in the following two directories:

- **'\user\email'**
This directory holds up to 10 messages which can be altered by normal-level FTP-users.
- **'\email'**
This directory holds up to 10 messages which can be altered by admin-level FTP-users.

Email definition files must be named 'email_1.cfg', 'email_2.cfg'... 'email_10.cfg' in order to be properly recognized by the module.

File Format:

```
[Register]
Area, Offset, Type

[Register Match]
Value, Mask, Operand

[To]
recipient

[From]
sender

[Subject]
subject line

[Headers]
Optional extra headers

[Message]
message body
```

Setting	Value	Scanned for SSI
Area	Source area in DPRAM. Possible values are 'IN' or 'OUT'	No
Offset	Source offset, written in decimal or hexadecimal.	
Type	Source data type. Possible values are 'byte', 'word', and 'long'	
Value	Used as a reference value for comparison.	
Mask	Mask value, applied on the trigger source prior to comparison (logical AND).	
Operand	Possible values are '<', '=', or '>'	
To	Email recipient	Yes
From	Sender email address	
Subject	Email subject. One line only.	
Headers	Optional; may be used to provide additional headers.	
Message	The actual message.	

Note: Hexadecimal values must be written with the prefix '0x' in order to be recognized by the module.

Server Side Include (SSI)

General Information

Server Side Include functionality (from now on referred to as SSI) allows data in files and objects to be represented on web pages and in email messages.

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

Syntax

The 'X's below represents a command opcode and parameters associated with the command.

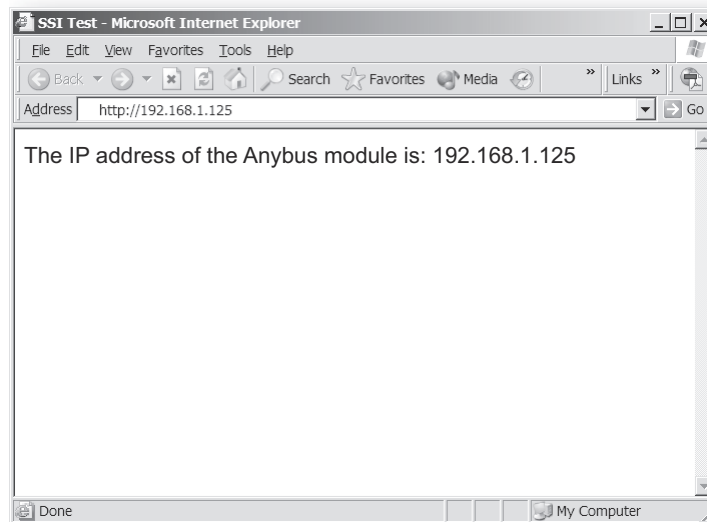
```
<?--#exec cmd_argument='XXXXXXXXXXXXXXXXXXXXX' -->
```

Example

The following example causes a web page to display the IP address of the module:

```
<HTML>
<HEAD><TITLE>SSI Test</TITLE></HEAD>
<BODY>
The IP address of the Anybus module is:
<?--#exec cmd_argument='DisplayIP' -->
</BODY>
</HTML>
```

Resulting webpage:



Functions

DisplayIP

This function returns the currently used IP address.

Syntax:

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

DisplaySubnet

This function returns the currently used Subnet mask.

Syntax:

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

DisplayGateway

This function returns the currently used Gateway address.

Syntax:

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

DisplayDhcpState

This function returns whether DHCP/BootP is enabled or disabled.

Syntax:

```
<?--#exec cmd_argument='DisplayDhcpState( "Output when ON", "Output when OFF" )'-->
```

DisplayEmailServer

This function returns the currently used SMTP server address.

Syntax:

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

StoreEtnConfig

Note: This function cannot be used in email messages.

This function stores a passed IP configuration in the corresponding parameters.

Syntax:

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

Include this line in a HTML page and pass a form with new IP settings to it.

Accepted fields in form:

```
SetIp
SetSubnet
SetGateway
SetSMTPServer
SetDhcpState- (value "on" or "off")
```

Default output:

```
Invalid IP address!
Invalid Subnet mask!
Invalid Gateway address!
Invalid IP address or Subnet mask!
Invalid Email Server address!
Invalid DHCP state!
Configuration stored correctly.
Failed to store configuration.
```

GetText

Note: This function cannot be used in email messages.

This function retrieves the text from an object and stores it in a specified parameter or in the fieldbus out area.

Syntax:

```
<?--#exec cmd_argument='GetText( "ObjName", argument, N)'-->
```

```
ObjName      - Name of object.
argument      - (see below)
n             - Number of characters to process (Optional)
```

Argument	Description
OutWriteString(<i>offset</i>)	Writes the data to position <i>offset</i> in the fieldbus out area
ParWriteString(<i>prm</i>)	Writes the data to parameter <i>prm</i> . Requires parameter to be a string.
ParWriteFormatted(<i>prm</i>)	Writes the data to parameter <i>prm</i> . Parses and converts the data according to the parameter data type (i.e. integer, string etc.).

Default output:

```
Success      - Write succeeded
Failure      - Write failed
```


printf

This function returns a formatted string, which may contain parameter value and/or data from the field-bus in/out area. The formatting of the string is similar to the standard C function printf().

Syntax:

```
<?--#exec cmd_argument='printf("String to write", Arg1, Arg2, ..., ArgN)'-->
```

Like the standard C function printf() the "String to write" for this SSI function contains two types of objects: Ordinary characters, which are copied to the output stream, and conversion specifications, each of which causes conversion and printing of the next successive argument to printf. Each conversion specification begins with the character % and ends with a conversion character. Between the % and the conversion character there may be, in order:

- Flags (in any order), which modify the specification:
 - which specifies left adjustment of the converted argument in its field.
 - + which specifies that the number will always be printed with a sign
 - (space) if the first character is not a sign, a space will be prefixed.
 - 0 for numeric conversions, specifies padding to the field with leading zeroes.
 - # which specifies an alternate output form. For o, the first digit will be zero. For x or X, 0x or 0X will be prefixed to a non-zero result. For e, E, f, g and G, the output will always have a decimal point; for g and G, trailing zeros will not be removed.
- A number specifying a minimum field width. The converted argument will be printed in a field at least this wide, and wider if necessary. If the converted argument has fewer characters than the field width it will be padded on the left (or right, if left adjustment has been requested) to make up the field width. The padding character is normally space, but can be 0 if the zero padding flag is present.
- A period, which separates the field width from the precision.
- A number, the precision, that specifies the maximum number of characters to be printed from a string, or the number of digits to be printed after the decimal point for e, E, or F conversions, or the number of significant digits for g or G conversion, or the minimum number of digits to be printed for an integer (leading 0s will be added to make up the necessary width)
- A length modifier h, l (letter ell), or L. "h" Indicates that the corresponding argument is to be printed as a short or unsigned short; "l" indicates that the argument is along or unsigned long.

The conversion characters and their meanings are shown below. If the character after the % is not a conversion character, the behaviour is undefined.

Character(s)	Argument type, Converted to
d, i	byte, short; decimal notation (For signed representation. Use signed argument)
o	byte, short; octal notation (without a leading zero).
x, X	byte, short; hexadecimal notation (without a leading 0x or 0X), using abcdef for 0x or ABCDEF for 0X.
u	byte, short; decimal notation.
c	byte, short; single character, after conversion to unsigned char.
s	char*; characters from the string are printed until a "\0" is reached or until the number of characters indicated by the precision have been printed
f	float; decimal notation of the form [-]mmm.ddd, where the number of d's is specified by the precision. The default precision is 6; a precision of 0 suppresses the decimal point.
e, E	float; decimal notation of the form [-]m.ddddd e+-xx or [-]m.ddddd E+-xx, where the number of d's specified by the precision. The default precision is 6; a precision of 0 suppresses the decimal point.
g, G	float; %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 zeros and trailing decimal point are not printed.
%	no argument is converted; print a %

The arguments that can be passed to the SSI function *printf* are:

Argument	Description
InReadSByte(<i>offset</i>)	Read a signed byte from position <i>offset</i> in the fieldbus IN area
InReadUByte(<i>offset</i>)	Read an unsigned byte from position <i>offset</i> in the fieldbus IN area
InReadSWord(<i>offset</i>)	Read a signed word from position <i>offset</i> in the fieldbus IN area
InReadUWord(<i>offset</i>)	Read an unsigned word from position <i>offset</i> in the fieldbus IN area
InReadSLong(<i>offset</i>)	Read a signed longword from position <i>offset</i> in the fieldbus IN area
InReadULong(<i>offset</i>)	Read an unsigned longword from position <i>offset</i> in the fieldbus IN area
InReadString(<i>offset</i>)	Read a string (char*) from position <i>offset</i> in the fieldbus IN area
InReadFloat(<i>offset</i>)	Read a floating point (float) value from position <i>offset</i> in the fieldbus IN area
OutReadSByte(<i>offset</i>)	Read a signed byte from position <i>offset</i> in the fieldbus OUT area
OutReadUByte(<i>offset</i>)	Read an unsigned byte from position <i>offset</i> in the fieldbus OUT area
OutReadSWord(<i>offset</i>)	Read a signed word (short) from position <i>offset</i> in the fieldbus OUT area
OutReadUWord(<i>offset</i>)	Read an unsigned word (short) from position <i>offset</i> in the fieldbus OUT area
OutReadSLong(<i>offset</i>)	Read a signed longword (long) from position <i>offset</i> in the fieldbus OUT area
OutReadULong(<i>offset</i>)	Read an unsigned longword (long) from position <i>offset</i> in the fieldbus OUT area
OutReadString(<i>offset</i>)	Read a null-terminated string from position <i>offset</i> in the fieldbus OUT area
OutReadFloat(<i>offset</i>)	Read a floating point (float) value from position <i>offset</i> in the fieldbus OUT area
ParReadSByte(<i>par</i>)	Reads parameter <i>par</i> and returns it as a signed byte
ParReadUByte(<i>par</i>)	Reads parameter <i>par</i> and returns it as an unsigned byte
ParReadSWord(<i>par</i>)	Reads parameter <i>par</i> and returns it as a signed word
ParReadUWord(<i>par</i>)	Reads parameter <i>par</i> and returns it as an unsigned word
ParReadSLong(<i>par</i>)	Reads parameter <i>par</i> and returns it as a signed longword
ParReadULong(<i>par</i>)	Reads parameter <i>par</i> and returns it as an unsigned longword
ParReadString(<i>par</i>)	Reads parameter <i>par</i> and returns it as a string
ParReadFloat(<i>par</i>)	Reads parameter <i>par</i> and returns it as a float
ParReadFormatted(<i>par</i>)	Reads parameter <i>par</i> , interprets it and returns it as a formatted string.

scanf

Note: This function cannot be used in email messages.

This SSI function reads a string passed from an object in a HTML form, interprets the string according to the specification in format, and stores the result in the OUT area according to the passed arguments. The formatting of the string is equal to the standard C function call scanf()

Syntax:

```
<?--#exec cmd_argument='scanf( "ObjName", "format", Arg1, ..., ArgN), ErrVal1, ..., ErrValN'-->
```

ObjName	- The name of the object with the passed data string
format	- Specifies how the passed string shall be formatted
Arg1 - ArgN	- Specifies where to write the data
ErrVal1 -ErrValN	- Optional; specifies the value/string to write in case of an error.

Character(s)	Input, Argument Type
d	Decimal number; byte, short
i	Number, byte, short. The number may be in octal (leading 0(zero)) or hexadecimal (leading 0x or 0X)
o	Octal number (with or without leading zero); byte, short
u	Unsigned decimal number; unsigned byte, unsigned short
x	Hexadecimal number (with or without leading 0x or 0X); byte, short
c	Characters; char*. The next input characters (default 1) are placed at the indicated spot. The normal skip over white space is suppressed; to read the next non-white space character, use %1s.
s	Character string (not quoted); char*, pointing to an array of characters large enough for the string and a terminating "\0" that will be added.
e, f, g	Floating-point number with optional sign, optional decimal point and optional exponent; float*
%	Literal %; no assignment is made.

The conversion characters d, i, o, u and x may be preceded by l (letter ell) to indicate that a pointer to 'long' appears in the argument list rather than a 'byte' or a 'short'

The arguments that can be passed to the SSI function scanf are:

Argument	Description
OutWriteByte(offset)	Write a byte to position <i>offset</i> in the fieldbus OUT area
OutWriteWord(offset)	Write a word to position <i>offset</i> in the fieldbus OUT area
OutWriteLong(offset)	Write a long to position <i>offset</i> in the fieldbus OUT area
OutWriteString(offset)	Write a string to position <i>offset</i> in the fieldbus OUT area
OutWriteFloat(offset)	Write a floating point value to position <i>offset</i> in the fieldbus OUT area
ParWriteByte(par)	Writes a byte to parameter <i>par</i>
ParWriteWord(par)	Writes a word to parameter <i>par</i>
ParWriteLong(par)	Writes a longword to parameter <i>par</i>
ParWriteString(par)	Writes a string to parameter <i>par</i>
ParWriteFloat(par)	Writes a float to parameter <i>par</i>
ParWriteFloat(par)	Writes a float to parameter <i>par</i>
ParWriteFormatted(par)	Writes a formatted value to parameter <i>par</i>

Default output:

```
Write succeeded
Write failed
```

IncludeFile

This function includes the contents of a file.

Syntax:

```
<?--#exec cmd_argument='IncludeFile( "File name" )'-->
```

Default output:

Success	- <File content>
Failure	- Failed to open <filename>

SaveToFile

Note: This function cannot be used in email messages.

This SSI function saves the contents of a passed form to a file. The passed name/value pair will be written to the file "File name" separated by the "Separator" string. The [Append|Overwrite] parameter determines if the specified file shall be overwritten, or if the data in the file shall be appended.

Syntax:

```
<?--#exec cmd_argument='SaveToFile( "File name",  
"Separator", [Append|Overwrite] )'-->
```

Default output:

Success	- Form saved to file
Failure	- Failed to save form

GetConfigItem

This function returns the value of a specific key in a configuration file. The format of the file must be as follows:

```
[key1]  
Value1  
[key2]  
Value2  
...  
[keyN]  
ValueN
```

Syntax:

```
<?--#exec cmd_argument='GetConfigItem( "file", "key", "separator" )'-->
```

file	- File to read from
key	- Key to read value from
separator	- Optional; specifies line-separator ('CRLF' (default), 'LF' or 'CR')

Default output:

Success	- (Value associated with the specified key)
Failure	- Could not get value for [key]

For information about how to change the SSI output, please see 8-9 "Changing SSI output".

SetConfig

This function stores a passed HTML-form as a configuration file. The field name will be used as key, and the field value will be stored as the value to that key. If the passed key (field name) isn't present in the file, it will be added. If the specified file does not exist, it will be created.

Field names beginning with underscore ('_') will be excluded.

Syntax:

```
<?--#exec cmd_argument='SetConfig( "File name" )'-->
```

Default output:

Success – Configuration stored to “File name”

Failure – Could not store configuration to “File name”

For information about how to change the SSI output, please see 8-9 “Changing SSI output”.

Example:

An HTML-form with the following fields...

- Name = “Speed”, value = “48”
- Name = “Temp”, value = “20”
- Name = “_B1”, value = “submit”
(Button used to submit the form, not stored.)

... generates a file with the following format:

```
[Speed]
48
[Temp]
20
```

Changing SSI output

There is two methods of changing the output strings from SSI functions:

1. Changing SSI output defaults by creating a file called "\ssi_str.cfg" containing the output strings for all SSI functions in the system
2. Temporary changing the SSI output using SsiOutput()

SSI Output String File

If the file "\ssi_str.cfg" is found in the file system and the file is correctly according to the specification below, the SSI functions will use the output strings specified in this file instead of the default strings.

The files shall have the following format:

```
[StoreEtnConfig]
Success: "String to use on success"
Invalid IP: "String to use when the IP address is invalid"
Invalid Subnet: "String to use when the Subnet mask is invalid"
Invalid Gateway: "String to use when the Gateway address is invalid"
Invalid Email server: "String to use when the SMTP address is invalid"
Invalid IP or Subnet: "String to use when the IP address and Subnet mask does
not match"
Invalid DNS1: "String to use when the primary DNS cannot be found"
Invalid DNS2: "String to use when the secondary DNS cannot be found"
Save Error: "String to use when storage fails"
Invalid DHCP state: "String to use when the DHCP state is invalid"

[scanf]
Success: "String to use on success"
Failure: "String to use on failure"

[IncludeFile]
Failure: "String to use when failure"1

[SaveToFile]
Success: "String to use on success"
Failure: "String to use on failure"1

[SaveDataToFile]
Success: "String to use on success"
Failure: "String to use on failure"1

[GetText]
Success: "String to use on success"
Failure: "String to use on failure"
```

The contents of this file can be redirected by placing the line '[File path]' on the first row, and a file path on the second.

Example:

```
[File path]
\user\ssi_strings.cfg
```

In this example, the settings described above will be loaded from the file 'user\ssi_strings.cfg'.

1. '%s' includes the filename in the string

Temporary SSI Output change

The SSI output for the next called SSI function can be changed with the SSI function “SsiOutput()” The next called SSI function will use the output according to this call. Thereafter the SSI functions will use the default outputs or the outputs defined in the file ‘\ssi_str.cfg’. The maximum size of a string is 128 bytes.

Syntax:

```
<?--#exec cmd_argument='SsiOutput( "Success string", "Failure string" )'-->
```

Example:

This example shows how to change the output strings for a scanf SSI call.

```
<?--#exec cmd_argument='SsiOutput ( "Parameter1 updated", "Error" )'-->  
<?--#exec cmd_argument="scanf( "Parameter1", "%d", OutWriteByte(0) )'-->
```

CIP Object Implementation

General Information

This chapter specifies the CIP-object implementation in the module. These objects can be accessed from the network, but not directly by the host application.

Mandatory Objects:

- 9-2 “Identity Object (01h)”
- 9-4 “Message Router (02h)”
- 9-5 “Assembly Object (04h)”
- 9-6 “Connection Manager (06h)”
- 9-8 “TCP/IP Interface Object (F5h)”
- 9-10 “Ethernet Link Object (F6h)”

Vendor Specific Objects:

- 9-12 “Vendor Specific Objects”

Identity Object (01h)

General Information

Object Description

-

Supported Services

Class: Get_Attribute_Single

Instance: Get_Attribute_Single
Get_Attributes_All
Reset

Class Attributes

#	Name	Access	Type	Value
1	Revision	Get	UINT	0001h (Object revision)

Instance #1 Attributes

#	Name	Access	Type	Value
1	Vendor ID	Get	UINT	005Ah (HMS Industrial Networks AB) ^a
2	Device Type	Get	UINT	000Ch (Communication Adapter) ^a
3	Product Code	Get	UINT	0002h (Anybus-IC Ethernet/IP) ^a
4	Revision	Get	Struct of: {USINT, USINT}	Major and minor firmware revision
5	Status	Get	WORD	See 9-3 "Device Status"
6	Serial Number	Get	UDINT	Unique serial number (assigned by HMS)
7	Product Name	Get	SHORT_STRING	"Anybus-IC EtherNet/IP" (Name of product) ^a

a. Can be customized, see 11-10 "EtherNet/IP"

Device Status

bit(s)	Name
0	Module Owned
1	(reserved)
2	Configured (always set (1) in this implementation)
3	(reserved)
4... 7	Extended Device Status: <u>Value:</u> <u>Meaning:</u> 0000b Unknown 0010b Faulted I/O Connection 0011b No I/O connection established 0100b Non-volatile configuration bad 0110b Connection in Run mode 0111b Connection in Idle mode (other) (reserved)
8	Set for minor recoverable faults
9	Set for minor unrecoverable faults
10	Set for major recoverable faults
11	Set for major unrecoverable faults
12... 15	(reserved)

Service Details: Reset Service

The module can forward reset requests from the network to the application as interrupts. For more information about network reset handling, consult the general Anybus-IC Design Guide.

There are two types of network reset requests on EtherNet/IP:

- **Type 0: ‘Power Cycling Reset’**

By default, the module performs a reset of the module. Optionally, the module can issue an interrupt to the application, which in turn is responsible for resetting itself and the Anybus module.

See also...

- Anybus-IC Design Guide (Parameter #12 ‘Interrupt Config’, ‘RES’-bit)

- **Type 1: ‘Out of box reset’**

By default, the module resets the parameters below to their default values and resets itself.

- 11-5 “IP address cfg (Parameter #103)”
- 11-5 “Subnet mask cfg (Parameter #106)”
- 11-6 “GW address cfg (Parameter #108)”
- 11-7 “DHCP enable cfg (Parameter #114)”
- 11-8 “Data rate cfg (Parameter #117)”
- 11-9 “Duplex Cfg (Parameter #119)”
- 11-3 “SMTP Srv Address (Parameter #126)”

If the ‘DEF’-bit (Parameter #12, bit 4) is set, the module will issue an interrupt to the application. The interrupt cause register will indicate ‘Set Default’ as cause. The application is then responsible for resetting configuration settings before resetting itself and the module.

If the ‘RES’-bit (Parameter #12, bit 5) is set, the module will issue an interrupt to the application. The interrupt cause register will indicate ‘Reset’ as cause. The application is then responsible for resetting itself and the module.

See also...

- Anybus-IC Design Guide (Parameter #12 ‘Interrupt Config’, ‘RES’ and ‘DES’-bits)

Message Router (02h)

General Information

Object Description

-

Supported Services

Class: -

Instance: -

Class Attributes

-

Instance Attributes

-

Assembly Object (04h)

General Information

Object Description

The Assembly Object uses static assemblies and holds the fieldbus I/O data. The assembly instance IDs used are in the vendor specific range.

See also...

- 2-5 “EtherNet/IP Implementation”

Supported Services

Class: Get_Attribute_Single

Instance: Get_Attribute_Single
Set_Attribute_Single

Class Attributes

#	Name	Access	Type	Value
1	Revision	Get	UINT	0002h
2	Max Instance	Get	UINT	0096h

Instance 64h Attributes (Producing Instance)

#	Name	Access	Type	Value
3	Produced Data	Get	Array of BYTE	Data corresponds to fieldbus input data (FB IN)

Instance 96h Attributes (Consuming Instance)

#	Name	Access	Type	Value
3	Consumed Data	Set	Array of BYTE	Data corresponds to fieldbus output data (FB OUT)

Instance C6h Attributes (Heartbeat, Input-Only)

This instance is used as heartbeat for Input-Only connections, and does not carry any attributes. The data size of the Heartbeat instance in the Forward_Open-request should be 0 bytes, however other values are also permitted.

Instance C7h Attributes (Heartbeat, Listen-Only)

This instance is used as heartbeat for listen-only connections, and does not carry any attributes. The data size of the Heartbeat instance in the Forward_Open-request should be 0 bytes, however other values are also permitted.

Connection Manager (06h)

General Information

Object Description

-

Supported Services

Class: -

Instance: Forward_Open
Forward_Close

Instance Descriptions

(No supported instance attributes)

Class 1 Connection Details

General

Class 1 connections are used to transfer I/O data, and can be established to instances in the Assembly Object. Each Class 1 connection will establish two data transports; one consuming and one producing. The heartbeat instances can be used for connections that shall only access inputs.

Class 1 connections use UDP transport.

- Total number of supported class 1 connections: 20
- Supported transports: 1
- Max input connection size: 144 bytes
- Max output connection size: 144 bytes
- Supported API: 10... 3200ms
- T->O Connection type: Point-to-point, Multicast
- O->T Connection type: Point-to-point
- Supported trigger type: Cyclic/Change of state¹

1. Only applicable from firmware revision 2.01-...

Connection Types

- **Exclusive-Owner connection**

This type of connection controls the outputs of the Anybus module and does not depend on other connections.

- Max. no. of Exclusive-Owner connections: 1
- Connection point $O \Rightarrow T$: Assembly Object, instance 64h (Default)
- Connection point $T \Rightarrow O$: Assembly Object, instance 96h (Default)

- **Input-Only connection**

This type of connection is used to read data from the Anybus module without controlling the outputs. It does not depend on other connections.

- Max. no. of Input-Only connections: Up to 4¹
- Connection point $O \Rightarrow T$: Assembly Object, instance 03h (Default)
- Connection point $T \Rightarrow O$: Assembly Object, instance 96h (Default)

Note: If an Exclusive-Owner connection has been opened towards the module and times out, the Input-Only connection times out as well. If the Exclusive-Owner connection is properly closed, the Input-Only connection remains unaffected.

- **Listen-Only connection**

This type of connection requires another connection in order to exist. If that connection (Exclusive-Owner or Input-Only) is closed, the Listen-Only connection will be closed as well.

- Max. no. of Input-Only connections: Up to 4²
- Connection point $O \Rightarrow T$: Assembly Object, instance 04h (Default)
- Connection point $T \Rightarrow O$: Assembly Object, instance 96h (Default)

- **Redundant-Owner connection**

This connection type is not supported by the module.

Class 3 Connection Details

- **Explicit message connection**

Class 3 connections are used to establish connections towards the message router. Thereafter, the connection is used for explicit messaging.

Class 3 connections use TCP transport.

- No. of simultaneous Class 3 connections: 16
- Supported API: ≥ 2 ms
- $T \rightarrow O$ Connection type: Point-to-point
- $O \rightarrow T$ Connection type: Point-to-point
- Supported trigger type: Application

1. Shared with Exclusive-Owner and Listen-Only connections
2. Shared with Exclusive-Owner and Input-Only connections

TCP/IP Interface Object (F5h)

General Information

Object Description

The object groups TCP/IP-related settings.

See also...

- 2-4 “Switches (Fieldbus Specific Input)”
- 3-1 “TCP/IP Settings”

Supported Services

Class services: Get_Attribute_All
 Get_Attribute_Single

Instance services: Get_Attribute_All
 Get_Attribute_Single
 Set_Attribute_Single

Class Attributes

#	Access	Name	Type	Value	Description
1	Get	Revision	UINT	0002h	Revision 2

Instance Attributes

#	Access	Name	Type	Value	Comments
1	Get	Status	DWORD	0000 0001h	Attribute #1 contains valid information.
2	Get	Configuration Capability	DWORD	0000 0014h	Capable of obtaining network configuration via DHCP. Configuration settable.
3	Get/Set	Configuration Control	DWORD	-	<u>Value:</u> <u>Meaning:</u> 0 Configuration from non-volatile memory 2 Configuration from DHCP
4	Get	Physical Link Object	Struct of:	-	Physical Link => Ethernet object
		Path Size	UINT	0002h	2 words
		Path	Padded EPATH	20 F6 24 01h	Path to Ethernet Link Object, Instance 1
5	Get/Set	Interface Configuration	Struct of:		
		IP Address	UDINT	-	Corresponds to IP address act (Parameter #105)
		Network Mask	UDINT	-	Corresponds to Subnet mask act (Parameter #107)
		Gateway Address	UDINT	-	Corresponds to GW address act (Parameter #109)
		Name Server 1	UDINT	0000 0000h	-
		Name Server 2	UDINT	0000 0000h	-
		Domain Name	STRING	00h	-
6	Get/Set	Host Name	STRING	00h	-
g ^a	Get/Set	TTL value	USINT	1	TTL value for EIP multicast packets
g ^a	Get/Set	Mcast Config	Struct of:	-	IP multicast address configuration
		Alloc Control	USINT	0	0 - Use default allocation algorithm to generate multicast addresses.
		1 - Multicast addresses shall be allocated according to the values in Num Mcast and Mcast Start Addr.			
		Reserved	USINT	-	Shall be 0
		Num Mcast	UINT	1	Number of multicast addresses to allocate for Ethernet/IP.

a. Only applicable from firmware revision 2.01-...

Ethernet Link Object (F6h)

General Information

Object Description

This object groups information for the physical link.

See also...

- 3-1 “Physical Link Settings”
- 11-5 “Communication Settings”

Supported Services

Class services: Get_Attribute_All
 Get_Attribute_Single

Instance services: Get_Attribute_All
 Get_Attribute_Single

Class Attributes

#	Access	Name	Type	Value	Comments
1	Get	Revision	UINT	0002h	Revision 2

Instance Attributes

#	Access	Name	Type	Value	Comments
1	Get	Interface Speed	UDINT	10 or 100	Actual ethernet interface speed
2	Get	Interface Flags	DWORD	-	See 9-11 “Interface Flags”
3	Get	Physical Address	Array of 6 USINTS	(MAC ID)	Physical network address
6	Get/Set	Interface Control	Struct:		
		Control Bits	WORD	-	Interface control bits
		Forced Interface Speed	UINT	-	Speed at which the interface shall be forced to operate. Returns ‘Object state Conflict’ if auto-negotiation is enabled.

Interface Flags

Bit	Name	Description
0	Link status	Indicates whether or not the Ethernet 802.3 communications interface is connected to an active network. <u>Value:</u> <u>Meaning:</u> 0 Inactive link 1 Active link
1	Half/full duplex	Indicates the duplex mode currently in use. <u>Value:</u> <u>Meaning:</u> 0 Half duplex 1 Full duplex
2 - 4	Negotiation Status	Indicates the status of link auto-negotiation <u>Value:</u> <u>Meaning:</u> 0 Auto-negotiation in progress. 1 Auto-negotiation and speed detection failed (using default values) 2 Auto negotiation failed but detected speed (using default duplex value) 3 Successfully negotiated speed and duplex. 4 Auto-negotiation not attempted. Forced speed and duplex.
5	Manual Setting requires Reset	<u>Value:</u> <u>Meaning:</u> 0 Interface can activate changes to link parameters during runtime 1 Reset is required in order for changes to have effect
6	Local Hardware Fault	<u>Value:</u> <u>Meaning:</u> 0 No local hardware fault detected 1 Local hardware fault detected
7-31	(reserved)	(ignore)

Vendor Specific Objects

It is possible to create application-specific parameters and map them to a Vendor Specific CIP Class, Instance and Attribute.

See also...

- 11-15 “Application Parameters (Parameters #200... #299)”
- A-1 “Application Parameters”

Fieldbus Interface

General

The physical interface requires an external isolation transformer with the following characteristics:

Turns Ratio (TX)	Turns Ratio (RX)	Common Mode Rejection (0 - 30Mhz)
1CT:1CT	1CT:1CT	-40db (-59dB recommended for EtherNet/IP conformance)

The PCB is part of the physical layer in that the characteristics of the traces and materials control impedance, capacitance, coupling and voltage withstand. This means that the PCB layout is a key factor when it comes to reducing noise ingress and emissions.

To ensure proper operation, make sure to fulfil the following recommendations:

- Earth ground planes and power planes must be properly defined and isolated
- Traces between connector and transformer must be short and equal in length
- Traces between transformer and application interface must be short and equal in length
- Traces must match the circuit impedance (100 Ohms) using micro strip layout techniques

Suggested reading:

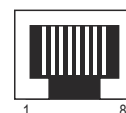
- “General Recommendations for EtherNet/IP Developers” (ODVA)
- “Successfully Designing EtherNet/IP Devices for Harsh Areas” (Rockwell Automation)

Application Connector Signals

Pin	Pin name	Signal
13	FB1	TX+
14	FB2	TX-
15	FB3	RX+
16	FB4	RX-
19	FB5	CT (RX)
20	FB6	CT (TX)

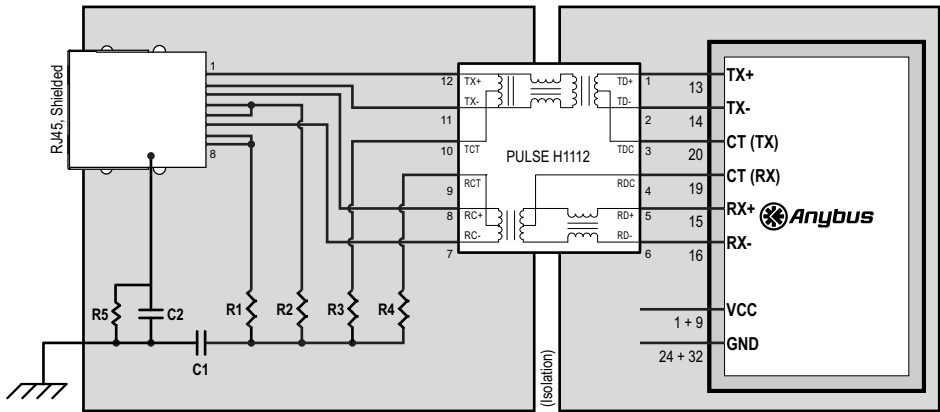
RJ45 Pinout

Ethernet Connector (RJ45)		Anybus	
Pin	Signal	Pin	Signal
1	TX+	-	-
2	TX-	-	-
3	RX+	-	-
4	-	-	-
5	-	-	-
6	RX-	-	-
7	-	-	-
8	-	-	-
Housing	Cable Shield	-	-



Typical Implementation

The following example uses the PULSE H1112. Other isolation transformers which conform to the characteristics described earlier may also be used, but may require a slightly different circuit connection than the one illustrated below.

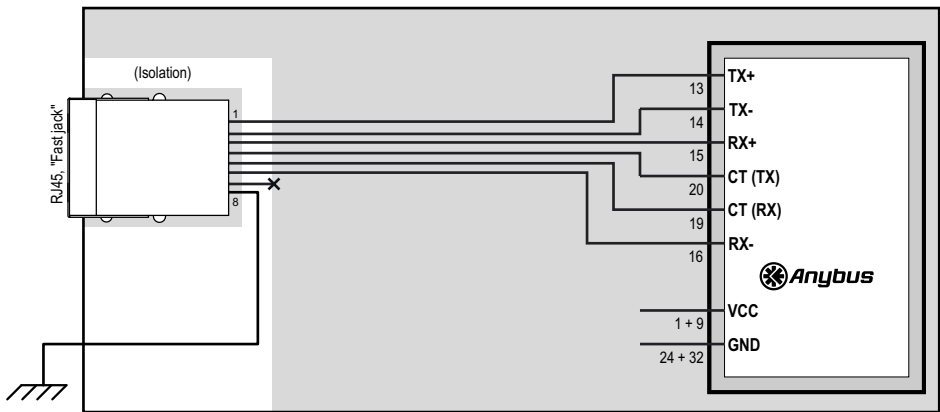


Ref.	Component	Comments
R1, R2, R3, R4	75 ohm	Network termination
R5	1M ohm	Filter to PE
C1, C2	1nF/2kV	Filter capacitors to PE
PULSE H1112	PULSE H1112	-
RJ45, Shielded	RJ45 connector	-

FastJack Connectors

FastJack connectors feature built-in isolation transformers and cable shield filters, which makes them suitable for applications where limited space is an issue.

To comply with modern EMC directives, this type of connector requires the use of shielded cables. Also note that this type of connector is generally not recommended for EtherNet/IP applications.



Ref.	Component	Comments
RJ45, Fast Jack	HFJ11-2450E	FastJack (HALO electronics Inc.)

Fieldbus Specific Parameters

General

#	R/W	Name	Size	Default Value	Modbus Address	HOS Object
141	R	Serial number	4 byte	N/A	H'704D	Device Object (0x80)
100	RW	FB status	2 byte	N/A	H'7000	Fieldbus Object (0xA0)
102	W	FB Password	2 byte	N/A	H'7003	
116	R	MAC address	6 byte	N/A	H'701B - H'701D	
104	R	DIP switch SSC	1 byte	N/A	H'7006	SSC Object (0xA2)

Network Configuration

#	R/W	Name	Size	Default Value	Modbus Address	HOS Object
103	RW	IP address cfg	4 byte	0.0.0.0	H'7004 – H'7005	Fieldbus Object (0xA0)
105	R	IP address act	4 byte	N/A	H'7007 - H'7008	
106	RW	Subnet mask cfg	4 byte	0.0.0.0	H'7009 - H'700A	
107	R	Subnet mask act	4 byte	N/A	H'700B - H'700C	
108	RW	GW address cfg	4 byte	0.0.0.0	H'700D - H'700E	
109	R	GW address act	4 byte	N/A	H'700F - H'7010	
114	RW	DHCP enable cfg	1 byte	0x01	H'7019	
115	R	DHCP enable act	1 byte	N/A	H'701A	
117	RW	Data rate cfg	1 byte	0x00	H'701E	
118	R	Data rate act	1 byte	N/A	H'701F	
119	RW	Duplex cfg	1 byte	0x00	H'7020	
120	R	Duplex act	1 byte	N/A	H'7021	
136	RW	HICP enable	1 byte	0x01	H'7039	
137	RW	HICP password	30 byte	-	H'703A-H'7048	

Server Settings

#	R/W	Name	Size	Default Value	Modbus Address	HOS Object
121	RW	Web srv enable	1 byte	0x01	H'7022	Fieldbus Object (0xA0)
122	RW	FTP srv enable	1 byte	0x01	H'7023	
123	RW	Telnet enable	1 byte	0x01	H'7024	

Email Client

#	R/W	Name	Size	Default Value	Modbus Address	HOS Object
126	RW	SMTP srv address	4 byte	0.0.0.0	H'7027 - H'7028	Fieldbus Object (0xA0)
127	R	Triggered emails	2 byte	N/A	H'7029	
128	R	SMTP errors	2 byte	N/A	H'702A	
129	W	Send email	2 byte	0x0000	H'702B	

File System

#	R/W	Name	Size	Default Value	Modbus Address	HOS Object
124	RW	Admin mode cfg	1 byte	0x00	H'7025	Fieldbus Object (0xA0)
125	RW	Admin mode act	1 byte	N/A	H'7026	
130	RW	VFS enable	1 byte	0x01	H'702C	
131	RW	RAM disc path	16 byte	"\RAM"	H'702D-H'7034	

Modbus/TCP & EtherNet/IP Related Parameters

#	R/W	Name	Size	Default Value	Modbus Address	Object
132	RW	MB/TCP conn TO	2 byte	60	H'7035	Fieldbus Object (0xA0)
133	RW	MB/TCP enable	1 byte	0x01	H'7036	
134	RW	In bit size	2 byte	0x30	H'7037	
135	RW	Out bit size	2 byte	0x30	H'7038	
139	RW	On/Off line time	1 byte	0x0A	H'704A	
140	RW	On/Off line cmds	4 byte	0xFFFFFFFF	H'704B- H'704C	
138	RW	On/Off line trg	1 byte	0x01	H'7049	
160	RW	EIP enable cfg	1 byte	0x01	H'7100	
161	R	EIP enable act	1 byte	N/A	H'7101	
162	RW	EIP strip status	1 byte	1	H'7102	
163	R(W)	EIP vendor ID	2 byte	0x005A	H'7103	
164	R(W)	EIP device type	2 byte	0x000C	H'7104	
165	R(W)	EIP product code	2 byte	0x0002	H'7105	
166	R(W)	EIP revision	2 byte	N/A	H'7106	
167	R(W)	EIP product name	33 byte	-	H'7107 – H'7117	

Application Parameters

#	R/W	Name	Size	Default Value	Modbus Address	Object
200	a	Application Parameter #1	a	a	H'8000	Application Parameter Object (0x85)
201	a	Application Parameter #2	a	a	a	
...	a	Application Parameter #3-99	a	a	a	
299	a	Application Parameter #100	a	a	a	

a. Parameter Dependant

General

Serial Number (Parameter #141)

This parameter holds the production serial number of the module.

Parameter Name	'Serial Number'
Parameter no.	141
Modbus Address	704Dh
Default Value	-
Range	0h... FFFFFFFFh
Size	4 bytes
Stored in NV RAM	No
Access	RW

FB Status (Parameter #100)

This parameter holds returns information about the current network status.

Parameter Name	'FB Status'
Parameter no.	100
Modbus Address	7000h
Default Value	-
Range	0000h... FFFFh
Size	2 bytes
Stored in NV RAM	No
Access	R

Bit layout

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
(reserved)						IDLE	BUS	(reserved)							

- **BUS**
 - 1: Bus is on line
 - 0: Bus is off line
- **IDLE**
 - 1: Scanner is in idle / program mode
 - 0: Scanner is in run mode'

MAC address (Parameter #116)

This parameter holds the ethernet MAC address of the module.

Parameter Name	'MAC address'
Parameter no.	116
Modbus Address	701Bh... 701Dh
Default Value	-
Range	-
Size	6 bytes
Stored in NV RAM	No
Access	R

DIP switch SSC (Parameter #104)

This parameter holds the auto configured fieldbus node address from Fieldbus Specific Input register on the SSC interface. Note that in order for this value to be valid, the 'NA'-bit (parameter #8 'Configuration Bits') must be cleared (0).

Parameter Name	'DIP switch SSC'
Parameter no.	104
Modbus Address	7006h
Default Value	-
Range	0h... FFh
Size	1 byte
Stored in NV RAM	No
Access	R

Communication Settings

IP address cfg (Parameter #103)

This parameter holds the manually configured IP address.

See also...

- 3-1 “TCP/IP Settings”

Parameter Name	'IP address cfg'
Parameter no.	103
Modbus Address	7004h... 7005h
Default Value	0.0.0.0
Range	0.0.0.0... 255.255.255.255
Size	4 bytes
Stored in NV RAM	No
Access	RW

IP address act (Parameter #105)

This parameter holds the actual IP address.

See also...

- 3-1 “TCP/IP Settings”

Parameter Name	'IP address act'
Parameter no.	105
Modbus Address	7007h... 7008h
Default Value	-
Range	0.0.0.0... 255.255.255.255
Size	4 bytes
Stored in NV RAM	No
Access	R

Subnet mask cfg (Parameter #106)

With this parameter it is possible to configure the Subnet mask.

Note: The module must be restarted in order for changes to have effect.

See also...

- 3-1 “TCP/IP Settings”

Parameter Name	'Subnet mask cfgt'
Parameter no.	106
Modbus Address	7009h... 700Ah
Default Value	-
Range	0.0.0.0... 255.255.255.255
Size	4 bytes
Stored in NV RAM	No
Access	RW

Subnet mask act (Parameter #107)

This parameter holds the currently used Subnet mask.

See also...

- 3-1 “TCP/IP Settings”

Parameter Name	'Subnet mask act'
Parameter no.	107
Modbus Address	700Bh... 700Ch
Default Value	-
Range	0.0.0.0... 255.255.255.255
Size	4 bytes
Stored in NV RAM	No
Access	R

GW address cfg (Parameter #108)

With this parameter it is possible to configure the Gateway address.

Note: The module must be restarted in order for changes to have effect.

See also...

- 3-1 “TCP/IP Settings”

Parameter Name	'GW address cfg'
Parameter no.	108
Modbus Address	700Dh... 700Eh
Default Value	-
Range	0.0.0.0... 255.255.255.255
Size	4 bytes
Stored in NV RAM	No
Access	RW

GW address act (Parameter #109)

This parameter holds the currently used Gateway address.

Note: The module must be restarted in order for changes to have effect.

See also...

- 3-1 “TCP/IP Settings”

Parameter Name	'GW address act'
Parameter no.	109
Modbus Address	700Fh... 7010h
Default Value	-
Range	0.0.0.0... 255.255.255.255
Size	4 bytes
Stored in NV RAM	No
Access	R

DHCP enable cfg (Parameter #114)

This parameter enables/disables DHCP support.

Note: The module must be restarted in order for changes to have effect.

See also...

- 3-1 “TCP/IP Settings”

Parameter Name	'GW address act'
Parameter no.	114
Modbus Address	7019h
Default Value	1
Range	0: Disable 1: Enable
Size	1 bytes
Stored in NV RAM	No
Access	RW

DHCP enable act (Parameter #115)

This parameter holds the current DHCP state.

See also...

- 3-1 “TCP/IP Settings”

Parameter Name	'DHCP enable act'
Parameter no.	115
Modbus Address	701Ah
Default Value	-
Range	0: Disabled 1: Enabled
Size	1 byte
Stored in NV RAM	No
Access	R

Data rate cfg (Parameter #117)

This parameter specifies the data rate for the physical link.

Note: The module must be restarted in order for changes to have effect.

See also...

- 3-1 “Physical Link Settings”

Parameter Name	'Data rate cfg'
Parameter no.	117
Modbus Address	701Eh
Default Value	0
Range	0: Auto Negotiate 1: 10Mbps 2: 100Mbps)
Size	1 bytes
Stored in NV RAM	No
Access	RW

Data rate act (Parameter #118)

This parameter holds the currently used data rate of the physical link.

See also...

- 3-1 “Physical Link Settings”

Parameter Name	'Data rate cftt'
Parameter no.	118
Modbus Address	701Fh
Default Value	-
Range	1: 10Mbps 2: 100Mbps
Size	1 bytes
Stored in NV RAM	No
Access	R

Duplex Cfg (Parameter #119)

This parameter specifies the duplex mode for the physical link.

Note: The module must be restarted for changes to have effect.

See also...

- 3-1 “Physical Link Settings”

Parameter Name	'Duplex Cfg'
Parameter no.	119
Modbus Address	7020h
Default Value	0
Range	0: Auto Negotiate 1: Half Duplex 2: Full Duplex
Size	1 byte
Stored in NV RAM	No
Access	RW

Duplex Act (Parameter #120)

This parameter holds the currently used duplex mode of the physical link.

See also...

- 3-1 “Physical Link Settings”

Parameter Name	'Duplex Act'
Parameter no.	120
Modbus Address	7021h
Default Value	-
Range	1: Half Duplex 2: Full Duplex
Size	1 byte
Stored in NV RAM	No
Access	R

HICP Enable (Parameter #136)

This parameter enables/disables support for the Anybus IPconfig (HICP) protocol.

See also...

- 3-3 “Anybus IPconfig (HICP)”

Parameter Name	'HICP Enable'
Parameter no.	136
Modbus Address	7039h
Default Value	1
Range	0: Disable 1: Enable
Size	1 byte
Stored in NV RAM	No
Access	RW

HICP Password (Parameter #137)

The Anybus-IPconfig (HICP) protocol allows the IP configuration to be protected from unauthorized changes via a password, which can be specified through this parameter.

See also...

- 3-3 “Anybus IPconfig (HICP)”

Parameter Name	'HICP Password'
Parameter no.	137
Modbus Address	703Ah... 7048h
Default Value	-
Range	-
Size	30 bytes
Stored in NV RAM	No
Access	RW

ARP Gleaning Enable (Parameter #144)

See also...

- 3-1 “Network Configuration”

Parameter Name	'HICP Password'
Parameter no.	144
Modbus Address	706Fh
Default Value	1
Range	0: Disable 1: Enable
Size	1 byte
Stored in NV RAM	No
Access	RW

Server Settings

Web Srv Enable (Parameter #121)

This parameter enables/disables the built-in web server.

See also...

- 6-1 “Web Server”

Parameter Name	'Web Srv Enable'
Parameter no.	121
Modbus Address	7022h
Default Value	1
Range	0: Disable 1: Enable
Size	1 byte
Stored in NV RAM	No
Access	RW

FTP Srv Enable (Parameter #122)

This parameter enables/disables the built-in FTP server.

See also...

- 4-1 “FTP Server”

Parameter Name	'FTP Srv Enable'
Parameter no.	122
Modbus Address	7023h
Default Value	1
Range	0: Disable 1: Enable
Size	1 byte
Stored in NV RAM	No
Access	RW

Telnet Srv Enable (Parameter #123)

This parameter enables/disables the built-in Telnet server.

See also...

- 5-1 “Telnet Server”

Parameter Name	'Telnet Srv Enable'
Parameter no.	123
Modbus Address	7024h
Default Value	1
Range	0: Disable 1: Enable
Size	1 byte
Stored in NV RAM	No
Access	RW

Email Client

SMTP Srv Address (Parameter #126)

With this parameter it is possible to configure the SMTP server address.

See also...

- 7-1 “Email Client”

Parameter Name	'SMTP Srv Address'
Parameter no.	126
Modbus Address	7027h... 7028h
Default Value	0.0.0.0
Range	0.0.0.0... 255.255.255.255
Size	4 bytes
Stored in NV RAM	No
Access	RW

Triggered Emails (Parameter #127)

This parameter indicates how many email trigger events that has been detected by the module.

See also...

- 7-1 “Email Client”

Parameter Name	'Triggered Emails'
Parameter no.	127
Modbus Address	7029h
Default Value	0000h
Range	0000h... FFFFh
Size	2 bytes
Stored in NV RAM	No
Access	R

SMTP Errors (Parameter #128)

This parameter indicates how many emails that the module failed to send to the SMTP server.

See also...

- 7-1 “Email Client”

Parameter Name	'Triggered Emails'
Parameter no.	128
Modbus Address	702Ah
Default Value	0000h
Range	0000h... FFFFh
Size	2 bytes
Stored in NV RAM	No
Access	R

Send Email (Parameter #129)

This parameter is sends a predefined email stored in the file system.

The MSB in the data word determines the email type (0: Admin, 1: Normal), and the LSB specifies the number of the message.

Example:

The value 0004h issues Admin email no. #4, and 010Ah issues Normal user email no. #10.

See also...

- 7-1 “Email Client”

Parameter Name	'Send Email'
Parameter no.	129
Modbus Address	702Bh
Default Value	0000h
Range	0000h... 010Ah
Size	2 bytes
Stored in NV RAM	No
Access	W

SMTP User Name (Parameter #142)

See also...

- 7-1 “Email Client”

Parameter Name	'SMTP User Name'
Parameter no.	142
Modbus Address	704Fh... 705Eh
Default Value	""
Range	String, NULL-terminated
Size	Up to 32 bytes, including NULL-termination
Stored in NV RAM	No
Access	RW

SMTP Password (Parameter #143)

See also...

- 7-1 “Email Client”

Parameter Name	'SMTP Password'
Parameter no.	143
Modbus Address	705Fh... 706E
Default Value	""
Range	String, NULL-terminated
Size	Up to 32 bytes, including NULL-termination
Stored in NV RAM	No
Access	RW

File System

VFS Enable (Parameter #130)

This parameter enables/disables the virtual file system (VFS).

See also...

- 6-1 “Default Web Pages”

Parameter Name	'VFS Enable'
Parameter no.	130
Modbus Address	702Ch
Default Value	1
Range	0: Disable 1: Enable
Size	2 bytes
Stored in NV RAM	No
Access	RW

RAM-Disc Path (Parameter #131)

This parameter specifies where to mount the volatile part of the file system (i.e. the RAM disc). The specified directory must be empty or non-existing; an empty value (“”) disables the RAM-disc.

See also...

- 2-8 “Filesystem”
- 2-8 “Storage”

Parameter Name	'RAM Disc Path'
Parameter no.	131
Modbus Address	702Dh... 7034h
Default Value	"\RAM"
Range	String, null terminated
Size	Up to 16 bytes (null termination included)
Stored in NV RAM	No
Access	RW

Admin Mode Cfg (Parameter #124)

This parameter enables/disabled Global Admin Mode.

Note: The module must be restarted for changes to have effect.

See also...

- 4-1 “FTP Server” (4-1 “Security Levels”, 4-1 “User Accounts”)
- 5-1 “Telnet Server” (5-1 “Security Levels”, 5-1 “User Accounts”)

Parameter Name	'Admin Mode Cfg'
Parameter no.	124
Modbus Address	7025h
Default Value	0
Range	0: Disable Global Admin Mode 1: Enable Global Admin Mode
Size	1 byte
Stored in NV RAM	No
Access	RW

Admin Mode Act (Parameter #125)

This parameter indicates whether the module is operating in Global Admin Mode or not.

See also...

- 4-1 “FTP Server” (4-1 “Security Levels”, 4-1 “User Accounts”)
- 5-1 “Telnet Server” (5-1 “Security Levels”, 5-1 “User Accounts”)

Parameter Name	'Admin Mode Act'
Parameter no.	125
Modbus Address	7026h
Default Value	-
Range	0: Disabled 1: Enabled
Size	1 byte
Stored in NV RAM	No
Access	RW

Modbus/TCP

MB/TCP Conn TO (Parameter #132)

This parameter specifies the Modbus/TCP connection timeout in seconds. If no Modbus/TCP query has been received within the specified time period, the Modbus/TCP connection will be closed.

See also...

- 2-6 “Modbus/TCP Implementation”

Parameter Name	'MB/TCP Conn TO'
Parameter no.	132
Modbus Address	7035h
Default Value	60
Range	0... 65535 (0 = Timeout disabled)
Size	2 bytes
Stored in NV RAM	No
Access	RW

MB/TCP Enable (Parameter #133)

This parameter enables/disables support for the Modbus/TCP protocol.

See also...

- 2-6 “Modbus/TCP Implementation”

Parameter Name	'MB/TCP Enable'
Parameter no.	133
Modbus Address	7036h
Default Value	1
Range	0: Disable 1: Enable
Size	1 byte
Stored in NV RAM	No
Access	RW

In bit size (Parameter #134)

This parameter specifies how many bytes of the fieldbus input data that shall be represented as coils/discretes. The remainder will be represented as registers.

See also...

- 2-6 “Modbus/TCP Implementation” (2-7 “Fieldbus Input Data (FB In)”)

Parameter Name	'In bit size'
Parameter no.	134
Modbus Address	7037h
Default Value	0000h
Range	0000h... 0030h
Size	2 bytes
Stored in NV RAM	No
Access	RW

Out bit size (Parameter #135)

This parameter specifies how many bytes of the fieldbus output data that shall be represented as coils/discretes. The remainder will be represented as registers.

See also...

- 2-6 “Modbus/TCP Implementation” (2-7 “Fieldbus Output Data (FB Out)”)

Parameter Name	'Out bit size'
Parameter no.	135
Modbus Address	7038h
Default Value	0000h
Range	0000h... 0030h
Size	2 byte
Stored in NV RAM	No
Access	RW

On / Off line trg (Parameter #138)

This parameter specifies the trigger source for on/off line events.

See also...

- 11-9 “On / Off line time (Parameter #139)”

Parameter Name	'On / Off line trg'
Parameter no.	138
Modbus Address	7049h
Default Value	1
Range	0: None 1: Link 2: Modbus/TCP 3: EtherNet/IP
Size	1 bytes
Stored in NV RAM	No
Access	RW

On / Off line time (Parameter #139)

If the On/Off line trigger source (Parameter #138) is set to Modbus/TCP, this parameter specifies the maximum allowed time between Modbus/TCP commands; if this time is exceeded, the module triggers an off line event. The time is set in steps of 100ms (10 = 1000ms)

Note: This parameter has no effect if the On/Off line trigger source isn't set to Modbus/TCP

See also...

- 11-8 "On / Off line trg (Parameter #138)"
- 11-9 "On / Off line Cmds (Parameter #140)"

Parameter Name	'On / Off line time'
Parameter no.	139
Modbus Address	704Ah
Default Value	10 (1 second)
Range	1... 255
Size	1 byte
Stored in NV RAM	No
Access	704Ah

On / Off line Cmds (Parameter #140)

If the On / Off line trigger source (#138) is set to Modbus/TCP, this parameter specifies which Modbus commands that shall trigger an on-line event. The parameter value represents a bit field where each bit specifies if a particular Modbus/TCP command shall trigger the on-line event or not.

Note: This parameter has no effect if the On/Off line trigger source isn't set to Modbus/TCP

See also...

- 11-8 "On / Off line trg (Parameter #138)"
- 11-9 "On / Off line time (Parameter #139)"

Parameter Name	'On / Off line Cmds'
Parameter no.	140
Modbus Address	704Bh... 704Ch
Default Value	FFFFFFFFh
Range	Bit field, 32 bits
Size	4 bytes
Stored in NV RAM	No
Access	RW

- **Value**

Each bit in the value represents a Modbus/TCP function code, see below.

b31	b30	b29	...	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
FC32	FC31	FC30	...	FC12	FC11	FC10	FC9	FC8	FC7	FC6	FC5	FC4	FC3	FC2	FC1

FCxx = Modbus/TCP function code xx

EtherNet/IP

EIP Enb Cfg (Parameter #160)

This parameter enables/disables support for the EtherNet/IP protocol.

Note: The module must be restarted for changes to have effect.

See also...

- 2-5 “EtherNet/IP Implementation”

Parameter Name	'EIP Enb Cfg'
Parameter no.	160
Modbus Address	7100h
Default Value	1
Range	0: Disable 1: Enable
Size	1 byte
Stored in NV RAM	No
Access	RW

EIP Enb Act (Parameter #161)

This parameter indicates whether support for the EtherNet/IP protocol is enabled or disabled.

See also...

- 2-5 “EtherNet/IP Implementation”

Parameter Name	'EIP Enb Act'
Parameter no.	161
Modbus Address	7101h
Default Value	-
Range	0: Disabled 1: Enabled
Size	1 byte
Stored in NV RAM	No
Access	R

EIP Strip Status (Parameter #162)

This parameter specifies how the first four bytes of a consumed EtherNet/IP connection shall be treated by the module.

See also...

- 11-3 “FB Status (Parameter #100)”
- 9-1 “CIP Object Implementation” (9-5 “Assembly Object (04h)”)

Parameter Name	'EIP Strip Status'
Parameter no.	162
Modbus Address	7102h
Default Value	1
Range	0: Don't strip status data 1: Strip status data; FB Status (Parameter #100) reflects the master's run/idle mode.
Size	1 byte
Stored in NV RAM	No
Access	RW

FB Password (Parameter #102)

This parameter grants write access to the following parameters provided that a valid password is supplied:

- EIP Vendor ID (Parameter #163)
- EIP Device Type (Parameter #164)
- EIP Product Code (Parameter #165)
- EIP Revision (Parameter #166)
- EIP Product Name (Parameter #167)

(The password can be obtained by contacting HMS)

Parameter Name	'FB Password'
Parameter no.	102
Modbus Address	7003
Default Value	-
Range	0000h... FFFFh
Size	2 bytes
Stored in NV RAM	No
Access	W

EIP Vendor ID (Parameter #163)

This parameter holds the EtherNet/IP Vendor ID.

See also...

- 2-1 "Device Identity"
- 9-2 "Identity Object (01h)"
- 11-12 "FB Password (Parameter #102)"

Parameter Name	'EIP Vendor ID'
Parameter no.	163
Modbus Address	7103h
Default Value	005Ah (HMS Networks AB)
Range	0000h... FFFFh
Size	2 byte
Stored in NV RAM	No
Access	R(W)

EIP Device Type (Parameter #164)

This parameter holds the EtherNet/IP Device Type.

See also...

- 2-1 “Device Identity”
- 9-2 “Identity Object (01h)”
- 11-12 “FB Password (Parameter #102)”

Parameter Name	'EIP Device Type'
Parameter no.	164
Modbus Address	7104h
Default Value	000Ch (Communication Adapter)
Range	0000h... FFFFh
Size	2 byte
Stored in NV RAM	No
Access	R(W)

EIP Product Code (Parameter #165)

This parameter holds the EtherNet/IP Product Code.

See also...

- 2-1 “Device Identity”
- 9-2 “Identity Object (01h)”
- 11-12 “FB Password (Parameter #102)”

Parameter Name	'EIP Product Code'
Parameter no.	165
Modbus Address	7105h
Default Value	0002h (Anybus-IC)
Range	0000h... FFFFh
Size	2 byte
Stored in NV RAM	No
Access	R(W)

EIP Revision (Parameter #166)

This parameter holds the EtherNet/IP Revision.

See also...

- 2-1 “Device Identity”
- 9-2 “Identity Object (01h)”
- 11-12 “FB Password (Parameter #102)”

Parameter Name	‘EIP Product Code’
Parameter no.	166
Modbus Address	7106h
Default Value	Anybus-IC Product Revision
Range	0000h... FFFFh
Size	2 byte
Stored in NV RAM	No
Access	R(W)

EIP Product Name (Parameter #167)

This parameter holds the EtherNet/IP Product Name.

See also...

- 2-1 “Device Identity”
- 9-2 “Identity Object (01h)”
- 11-12 “FB Password (Parameter #102)”

Parameter Name	‘EIP Product Name’
Parameter no.	167
Modbus Address	7107h... 7116h
Default Value	‘Anybus-IC EtherNet/IP’
Range	String, null terminated
Size	32 byte (null termination included)
Stored in NV RAM	No
Access	R(W)

Application Parameters (Parameters #200... #299)

See also...

- 9-12 “Vendor Specific Objects”
- A-1 “Application Parameters”

Application Parameters

General Information

An Application Parameter is an application-specific parameter created by the application during startup. Application Parameters works just like normal parameters and can be accessed via the MIF and SCI interfaces. In addition, Application Parameters can be accessed from the network by mapping them to CIP objects.

See also...

- 9-1 “CIP Object Implementation” (9-12 “Vendor Specific Objects”)
- A-2 “Creating an Application Parameter”
- A-6 “Mapping an Application Parameter to CIP”
- B-1 “HMS Object Implementation”

Creating an Application Parameter

Query - “Application Parameter Object”

To create a new Application Parameter, send the following message using Modbus Object Messaging. (Consult the Anybus-IC Design Guide for more information about the Object Message Sub Field)

Object Message Sub Field.

Fragment byte count	Fragment protocol	Class ID	Instance ID	Service Code	Attribute	Data Field
(size)	02h	0085h	0000h	0005h	0000h	(See below)
Parameter Size (WORD)	Descriptor (DWORD)		Parameter Info (Size varies)		Extension Word (Optional, WORD)	

Parameter Size

This value depends on the type of data specified in the Descriptor (see below).

Data Type	Valid Parameter Size values
UINT, INT, BITSTRING	1, 2, 4
FLOAT	4
STRING	1... 32 (String length including NULL termination)
BYTE_ARRAY	1... 1024

Descriptor¹

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
(reserved)						Data Format		Data Type			(reserved)			Write	Read

Write	Meaning
0	Write access not allowed
1	Write access allowed

Read	Meaning
0	Read access not allowed
1	Read access allowed

Data Type	Meaning
0 (0000)	UINT
1 (0001)	INT
2 (0010)	BITSTRING
3 (0011)	STRING
4 (0100)	FLOAT
5 (0101)	BYTE ARRAY

Data Format	Meaning
0 (00)	Dec.
1 (01)	Hex
2 (10)	Bin
3 (11)	Dotted decimal

1. Note that the upper 16 bits of the Descriptor are reserved for future use and must be set to 0000h.

Parameter Info

The size and contents of this field depends on the Data Type specified in the Descriptor block.

- **Data types UINT, INT, BITSTRING & FLOAT**

Min Value (size varies)	Max Value (size varies)	Init Value (size varies)	Name ^a (String, 16 bytes)	Unit ^a (String, 16 bytes)
----------------------------	----------------------------	-----------------------------	---	---

Field	Type / Size	Description
Min. Value	Specified in 'Parameter Size'	Minimum allowed parameter value
Max. Value	Specified in 'Parameter Size'	Maximal allowed parameter value
Init Value	Specified in 'Parameter Size'	Initial parameter value
Name	String (16 byte, null terminated)	Name of parameter, e.g "Speed" ^a
Unit	String (16 byte, null terminated)	Unit, e.g "RPM" ^a

a. These fields are optional. (However, if used, both fields must be present)

- **Data type STRING**

Init Value (size varies)	Name ^a (STRING, 16 bytes)	Unit ^a (STRING, 16 bytes)
-----------------------------	---	---

Field	Type / Size	Description
Init Value	Specified in 'Parameter Size'	Initial value
Name	String (16 byte, null terminated)	Name of parameter ^a
Unit	String (16 byte, null terminated)	Unit ^a

a. These fields are optional. (However, if used, both fields must be present)

- **Data type BYTE_ARRAY**

Min. Value (BYTE)	Max. Value (BYTE)	Init Value (BYTE)	Name ^a (String, 16 bytes)	Unit ^a (String, 16 bytes)
----------------------	----------------------	----------------------	---	---

Field	Type / Size	Description
Min. Value	Byte	Min. allowed value of each element in the array
Max. Value	Byte	Max. allowed value of each element in the array
Init Value	Byte	Initial value of all elements in the array
Name	String (16 byte, null terminated)	Name of parameter ^a
Unit	String (16 byte, null terminated)	Unit ^a

a. These fields are optional. (However, if used, both fields must be present)

Extension Word (Optional)

This word is optional and specifies whether the response message should contain the Modbus address of the created Application Parameter or not.

Value	Description
0x0000	-
0x0001	Request Modbus Address
Other values	(Reserved for future use)

Response - “Application Parameter Object”

The Anybus-IC module will respond with the following message. (Consult the Anybus-IC Design Guide for more information about the Object Message Sub Field)

Object Message Sub Field

Fragment byte count	Fragment protocol	Class ID	Instance ID	Service Code	Error Code	Data Field
(size)	02h	0085h	0000h	0006h	0000h	(See below)
HOS Instance (WORD)			Parameter Number (WORD)		Modbus Address ^a (WORD)	

a. This field is only present if the Extension Word of the query is set to 0001h

HOS Instance

If the Error Code is 0 (Success), this field contains the HOS Instance of the created Application Parameter.

Parameter Number

If the Error Code is 0 (Success), this field contains the parameter number of the created Application Parameter.

Modbus Address

If the Error Code is 0 (Success), this field contains the Modbus Address of the created Application Parameter.

Note: This field is only present if the Extension Word of the query is set to 0001h.

Example

The example below creates an Application Parameter with the following properties:

- Parameter Name “Speed”, unit “rpm”
- Type 16 bit unsigned INT, range 0 - 65535, initial parameter value 32768.
- R/W access

Query			
	01h	5Bh	
	37h	02h	
Class	0085h		Application Parameter Class
Instance	0000h		
Service Code	0005h		Create
Attribute	0000h		
Parameter Size	0002h		Parameter Size = 2 bytes
Descriptor MSW	0000h		
Descriptor LSW	0003h		UINT, DEC, R/W
Min value	0000h		Minimum allowed value: 0
Max value	FFFFh		Maximum allowed value: 65535
Init value	8000h		Initial value: 8000h
Name	53h ('S')	70h ('p')	"Speed"
	65h ('e')	65h ('e')	
	64h ('d')	00h	
	-	-	
	-	-	
	-	-	
	-	-	
	-	-	
Unit	72h ('r')	70h ('p')	"rpm"
	6Dh ('m')	00h	
	-	-	
	-	-	
	-	-	
	-	-	
	-	-	
	-	-	
Extension Word	0001h		Request Modbus Address
	CRC		

Response			
	01h	5Bh	
	0Fh	02h	
Class	0085h		Application Parameter Object Class
Instance	0000h		
Service Code	0006h		Create Response
Error Code	0000h		Success
HOS Instance	0001h		HOS Instance 1
Parameter no.	00C8h		Parameter no. = 200
Modbus Address	8000h		Modbus Address = 8000h
	CRC		

Mapping an Application Parameter to CIP

Acyclic data on the Anybus-IC module is exchanged by means of Application Parameters mapped to Vendor Specific CIP Objects.

The mapping procedure consists of two steps:

- **Creating the Application Parameter**
(See A-2 “Creating an Application Parameter”)
- **Mapping the created Application Parameter to a CIP Object**
This is done by creating a new instance in the Anybus-IC CIP Mapping Object Class (A5h). This class is used to map a vendor specific CIP Object Attribute onto an Anybus-IC Object Attribute.

Query - “CIP Mapping Object”

(Consult the Anybus-IC Design Guide for more information about the Object Message Sub Field)

Object Message Sub Field

Fragment byte count	Fragment protocol	Class ID	Instance ID	Service Code	Data Field	
(size)	02h	00A5h	0000h	0005h	(See below)	
CIP Class (WORD)	CIP Instance (WORD)	CIP Attribute (WORD)	HOS Class (WORD)	HOS Instance (WORD)	HOS Attribute (WORD)	Attribute Size (WORD)

CIP Class

CIP Class to map

CIP Instance

CIP Instance to map

CIP Attribute

CIP Attribute to map

Attribute Size

Size of attribute. This value should match the Parameter Size value in the Application Parameter request.

HOS Class

HOS Class to map.
(In this case 85h “Application Parameter Object Class”)

HOS Instance

HOS Instance to map
(In this case, use the HOS Instance value returned from the Application Parameter Object request when the Application Parameter was created.)

HOS Attribute

HOS Attribute to map
In this case, the 0001h (=Parameter Value)

Response - “CIP Mapping Object”

The response contains no additional data. (Consult the Anybus-IC Design Guide for more information about the Object Message Sub Field)

Object Message Sub Field

Fragment byte count	Fragment protocol	Class ID	Instance ID	Service Code	Error Code
(8 bits)	02h	00A5h	0000h	0006h	(16 bits)

Example

This example will map the Application Parameter created earlier in this chapter to CIP Class 144, Instance 1, Attribute 1.

Query		
	01h	5Bh
	17h	02h
Class	00A5h	
Instance	0000h	
Service Code	0005h	
Attribute	0000h	
CIP Class	0090h	
CIP Instance	0001h	
CIP Attribute	0001h	
HOS Class	0085h	
HOS Instance	0001h	
HOS Attribute	0001h	
Attribute Size	0002h	
	CRC	

Response		
	01h	5Bh
	09h	02h
Class	00A5h	
Instance	0000h	
Service Code	0006h	
Error Code	0000h	
	CRC	

CIP Mapping Object
Create Response
Success

HMS Object Implementation

General Information

The objects described in this chapter can be accessed using the Modbus Object Messaging protocol, and provides access to advanced fieldbus-specific functionality.

The module features the following HOS objects:

- B-2 “Application Parameter Object (Class 85h)”
- B-4 “File System Object (Class 86h)”
- B-9 “CIP Mapping Object (Class A5h)”
- B-11 “Socket Object (Class A6h)”

Application Parameter Object (Class 85h)

General Information

Object Description

This object manages Application Parameters.

See also...

- A-1 “Application Parameters”
- A-2 “Creating an Application Parameter”
- B-9 “CIP Mapping Object (Class A5h)”

Supported Services

Class: Get_Attribute
 Create (see B-3 “Create (Class Service)”)

Instance: -

Class Attributes

#	Name	Access	Type	Value
1	Revision	Get	Byte	01h
2	No. of Instances	Get	Word	Current no. of instances in class.

Instance Attributes

#	Name	Access	Type	Value
1	(varies)	(varies)	(varies)	(specified on 'create')
2	Parameter Size	Get	Word	
3	Descriptor	Get	DWord	

Create (Class Service)

Service Description

This service creates a new Application Parameter instance.

Service Request (05h)

#	Contents	Type	Comments
1	Parameter Size	Word	-
2	Descriptor	DWord	Determines access rights, types etc.
3	Min. Value	(varies)	Type depends on Descriptor
4	Max. Value		
5	Initial Value		
6	Name	String[16]	(optional)
7	Unit	String[16]	(optional)
8	ReqExtResp	Wprd	Request extended response (optional)

Service Response (06h)

#	Contents	Type	Comments
1	Instance Number	Word	HOS instance number
2	Parameter Number	Word	Parameter no. of the created parameter
3	Modbus Address	Word	(optional)

File System Object (Class 86h)

General Information

Object Description

The File System Object class can create and delete file system instances dynamically during runtime. Each instance is a unique handle to a file stream and contains services for file system operations.

See also...

- 2-8 “Filesystem”

Supported Services

Class:	Get_Attribute	
	File Open	(see B-5 “File Open (Class Service)”)
	File Close	(see B-5 “File Close (Class Service)”)
	File Delete	(see B-6 “File Delete (Class Service)”)
	File Copy	(see B-6 “File Copy (Class Service)”)
	File Move	(see B-7 “File Move (Class Service)”)
	File Rename	(see B-7 “File Rename (Class Service)”)
Instance:	File Read	(see B-8 “File Read (Instance Service)”)
	File Write	(see B-8 “File Write (Instance Service)”)

Class Attributes

#	Name	Access	Type	Value
1	Revision	Get	Byte	01h
2	No. of Instances	Get	Word	Current no. of instances in class.
3	Max. Instance	Get	Word	Max. no. of allowed instances; equals the max. number of files that can be open simultaneously.

Instance Attributes

-

File Open (Class Service)

Service Description

Creates a file instance and opens a file for reading, writing, or appending.

Service Request (80h)

#	Contents	Type	Comments
1	Mode	Byte	00h: Read mode 01h: Write mode 02h: Append mode
2	Filename	Byte[x]	Path + filename + NULL termination

- **Read mode**
Opens a file for read-only access
- **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.
- **Read 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 already exists, data will be appended at end-of-file.

Service Response (81h)

#	Contents	Type	Comments
1	File instance number	Word	Used for all further operations on the file

File Close (Class Service)

Service Description

Closes a previously opened file and deletes the instance.

Service Request (82h)

#	Contents	Type	Comments
1	File instance number	Word	Acquired on 'File Open'

Service Response (83h)

#	Contents	Type	Comments
1	File size	DWord	Size of closed file

File Delete (Class Service)

Service Description

Deletes a file.

Service Request (84h)

#	Contents	Type	Comments
1	Filename	Byte[x]	Path + filename + NULL termination

Service Response (85h)

#	Contents	Type	Comments
-	-	-	(no data)

File Copy (Class Service)

Service Description

Creates a copy of a file.

Service Request (86h)

#	Contents	Type	Comments
1	Filename (source)	Byte[x]	Source path + filename + NULL termination
2	Filename (destination)	Byte[x]	Destination path + filename + NULL termination

Service Response (87h)

#	Contents	Type	Comments
-	-	-	(no data)

File Move (Class Service)

Service Description

Moves a file.

Service Request (88h)

#	Contents	Type	Comments
1	Filename	Byte[x]	Source path + filename + NULL termination
2	Destination	Byte[x]	Destination path + NULL termination

Service Response (89h)

#	Contents	Type	Comments
-	-	-	(no data)

File Rename (Class Service)

Service Description

Renames a file.

Service Request (8Ah)

#	Contents	Type	Comments
1	Old filename	Byte[x]	Old path + filename + NULL termination
2	New filename	Byte[x]	New path + filename + NULL termination

Service Response (8Bh)

#	Contents	Type	Comments
-	-	-	(no data)

File Read (Instance Service)

Service Description

Reads data from a file opened for reading.

Service Request (A0h)

#	Contents	Type	Comments
1	Length	Word	Number of bytes to read (up to 0400h)

Service Response (A1h)

#	Contents	Type	Comments
1	Length	Word	Number of bytes read
2	Data	Byte[x]	Actual data read from file

File Write (Instance Service)

Service Description

Writes data to a file opened for writing or appending.

Service Request (A2h)

#	Contents	Type	Comments
1	Length	Word	Number of bytes write
2	Data	Byte[x]	Data that shall be written to the file

Service Response (A3h)

#	Contents	Type	Comments
1	Length	Word	Actual number of bytes written

CIP Mapping Object (Class A5h)

General Information

Object Description

Maps Application Parameter instances to CIP-attributes.

See also...

- 2-5 “EtherNet/IP Implementation” (2-5 “Explicit Data (Application Parameters)”)
- 9-1 “CIP Object Implementation” (9-12 “Vendor Specific Objects”)
- A-1 “Application Parameters” (A-6 “Mapping an Application Parameter to CIP”)
- B-2 “Application Parameter Object (Class 85h)”

Supported Services

Class: Get_Attribute
 Create (see B-10 “Create (Class Service)”)

Instance: -

Class Attributes

#	Name	Access	Type	Value
1	Revision	Get	Byte	01h
2	No. of Instances	Get	Word	Current no. of instances in class.
3	Max. Instance	Get	Word	Max. no. of allowed instances.

Instance Attributes

#	Name	Access	Type	Value
1	CIP Class	Get	Word	Mapping information
2	CIP Instance	Get	Word	
3	CIP Attribute	Get	Word	
4	HOS Class	Get	Word	
5	HOS Instance	Get	Word	
6	HOS Attribute	Get	Word	
7	Size	Get	Word	Size of attribute data

Create (Class Service)

Service Description

This service creates a new mapping instance.

Service Request (05h)

#	Contents	Type	Comments
1	CIP Class	Word	CIP instance target attribute; Class must be in vendor-specific range
2	CIP Instance	Word	
3	CIP Attribute	Word	
4	HOS Class	Word	Source Application Parameter instance attribute to map
5	HOS Instance	Word	
6	HOS Attribute	Word	
7	Size	Word	Size of attribute

Service Response (06h)

#	Contents	Type	Comments
-	-	-	-

Socket Object (Class A6h)

General Information

Object Description

The Socket Object Class can create and delete Socket Instances dynamically during runtime. Each socket instance contains services to establish and communicate over TCP or UDP channels.

Supported Services

Class:	Get_Attribute	
	Create Socket	(see B-12 "Create Socket (Class Service)")
	Close Socket	(see B-12 "Close Socket (Class Service)")
Instance:	Bind	(see B-13 "Bind (Instance Service)")
	Listen	(see B-13 "Listen (Instance Service)")
	Accept	(see B-14 "Accept (Instance Service)")
	Connect	(see B-14 "Connect (Instance Service)")
	Receive	(see B-15 "Receive (Instance Service)")
	Receive From	(see B-15 "Receive From (Instance Service)")
	Send	(see B-16 "Send (Instance Service)")
	Send to	(see B-16 "Send To (Instance Service)")

Class Attributes

#	Type	Access	Name	Req	Description
1	Byte	R	Class revision	R	Revision number of the Class.
2	Word	R	Number of Instances	R	Number of instances in the Class.
3	Word	R	Max Instances	R	Maximal allowed number of instances.

Instance Attributes

-

Create Socket (Class Service)

Service Description

This service creates a socket.

Service Request (0x80)

#	Type	Name	Description
1	Byte	Socket type	1: SOCK_STREAM (TCP socket) 2: SOCK_DGRAM (UDP socket)

Service Response (0x81)

#	Type	Name	Description
1	Word	Socket Error	See B-17 "Socket Errors"
2	Word	Instance Number	The number of the created socket instance.

Close Socket (Class Service)

Service Description

This service causes a connected socket to shut down and deletes the socket instance.

Service Request (0x82)

#	Type	Name	Description
1	Word	Instance	Socket Instance number to close

Service Response (0x83)

#	Type	Name	Description
1	Word	Socket Error	See B-17 "Socket Errors"

Bind (Instance Service)

Service Description

Binds a socket to a local port. Port 0 = any free port.

Service Request (0x84)

#	Type	Name	Description
1	Word	Port	The port to bind the socket to.

Service Response (0x85)

#	Type	Name	Description
1	Word	Socket Error	See B-17 "Socket Errors"
2	Word	Port	Local port number

Listen (Instance Service)

Service Description

Sets a socket to listening state.

Service Request (0x86)

#	Type	Name	Description	
1	Byte	Backlog	Backlog for incoming connections:	
			Backlog	Queue length
			0	1
			1	2
			2	4
			3	5
			4	7
			5	6

Service Response (0x87)

#	Type	Name	Description
1	Word	Socket Error	See B-17 "Socket Errors"

Accept (Instance Service)

Service Description

Accepts connections on a socket in listening state. A new socket-instance is created for each accepted connection. The new socket is connected with the host and the Listen Response returns its instance number.

Service Request (0x88)

#	Type	Name	Description
-	-	-	-

Accept Response (0x89)

#	Type	Name	Description
1	Word	Socket Error	See B-17 "Socket Errors"
2	Word	Instance Number	The number of the new instance for the connected socket.
3	DWord	IP address	Host IP address
4	Word	Port	Host port number

Connect (Instance Service)

Service Description

If the socket type is SOCK_DGRAM (UDP) this service specifies the peer with which the socket is to be associated. This is to which datagrams are sent and the only address from which datagrams are received.

If the socket type is SOCK_STREAM (TCP) this service attempts to establish a connection to another socket.

Stream sockets may connect successfully only once while datagram sockets may use connect multiple times to change their association. Datagram sockets may dissolve the association by connection to the invalid address: IP=0.0.0.0 Port=0

Service Request (0x8A)

#	Type	Name	Description
1	Dword	IP address	IP address to connect to
2	Word	Port	Port number to connect to

Service Response (0x8B)

#	Type	Name	Description
1	Word	Socket Error	See B-17 "Socket Errors"

Receive (Instance Service)

Service Description

This service receives data from a connected socket.

Service Request (0x8C)

#	Type	Name	Description
1	Word	Length	How many bytes to receive. Maximum value is 1460 bytes.

Service Response (0x8D)

#	Type	Name	Description
1	Word	Socket Error	See B-17 "Socket Errors"
2	Word	Length	Bytes received
3	Byte[x]	Data	Received data

Receive From (Instance Service)

Service Description

This service receives the next received datagram from an unconnected socket.

Service Request (0x8E)

#	Type	Name	Description
1	Word	Length	How many bytes to receive. Maximum value is 1460 bytes.

Service Response (0x8F)

#	Type	Name	Description
1	Word	Socket Error	See B-17 "Socket Errors"
2	Dword	IP address	Host IP address
3	Word	Port	Host port number
4	Word	Length	Bytes received
5	Byte[x]	Data	Received data

Send (Instance Service)

Service Description

This service sends data on a connected socket.

Service Request (0x90)

#	Type	Name	Description
1	Word	Length	Number of bytes to send. Maximum value is 1460 bytes.
2	Byte[x]	Data	Data to send

Service Response (0x91)

#	Type	Name	Description
1	Word	Socket Error	See B-17 "Socket Errors"
2	Word	Length	Number of sent bytes.

Send To (Instance Service)

Service Description

This service sends data on an unconnected socket.

Service Request (0x92)

#	Type	Name	Description
1	Dword	IP address	Host IP address
2	Word	Port	Host port number
3	Word	Length	Number of bytes to send. Maximum value is 1460 bytes.
4	Byte[x]	Data	Data to send

Service Response (0x93)

#	Type	Name	Description
1	Word	Socket Error	See B-17 "Socket Errors"
2	Word	Length	Number of sent bytes.

Socket Errors

Socket Error	Name
0	NOERROR
1	ENOBUFS
2	ETIMEDOUT
3	EISCONN
4	EOPNOTSUPP
5	ECONNABORTED
6	EWouldBlock
7	ECONNREFUSED
8	ECONNRESET
9	ENOTCONN
10	EALREADY
11	EINVAL
12	EMSGSIZE
13	EPIPE
14	EDESTADDRREQ
15	ESHUTDOWN
16	ENOPROTOOPT
17	EHAVEOOB
18	ENOMEM
19	EADDRNOTAVAIL
20	EADDRINUSE
21	EAFNOSUPPORT
22	EINPROGRESS
23	ELOWER

Firmware Upgrade

The module supports the “standard” Anybus-IC firmware update procedures:

- **Standard Firmware Upgrade**
(Consult the general Anybus-IC Design Guide for more information)
- **Firmware Upgrade using Bootloader Switch**
(Consult the general Anybus-IC Design Guide for more information)

In addition to this, it also supports firmware updates via FTP. To update the firmware using this method, follow the steps below:

1. As a precaution, make a backup copy of the filesystem contents.
2. Upload the firmware file to the system root (“\”), or to the user root (“\user\”) of the module.
3. Perform a module reset
During startup, the module will check for a new firmware file. If a valid file is found, the module will reprogram the flash. The file will be deleted automatically after programming.
4. Done.

Technical Specification

Electrical Specification

Protective Earth (PE) Requirements

See 10-1 “Fieldbus Interface” and 10-2 “Typical Implementation”.

Power Supply

Supply Voltage

The module requires a regulated $5V \pm 5\%$ DC power supply as specified in the Anybus-IC Design Guide.

Power Consumption

The maximum power consumption is 250mA.

Environmental Specification

- **Temperature**

Test performed according to IEC-68-2-1 and IEC 68-2-2.

Operating: -40 to +85 °C (-40 to 185°F)

Storage: -40 to +85 °C (-40 to 185°F)

- **Humidity**

The product is designed for a relative humidity of 5 to 95% non-condensing.

Test performed according to IEC 68-2-30.

EMC Compliance (CE)

EMC pre-compliance testing has been conducted according to the following standards:

- **Emission:** EN 50 081-2:1993

Tested per EN 55022:1997, class A

- **Immunity:** EN 61000-6-2: 1999

Tested per
EN 61000-4-2:1995
EN 61000-4-3:1996
EN 61000-4-4:1995
EN 61000-4-5:1995
EN 61000-4-6:1996

