User Manual

Anybus® Communicator™

for EtherNet/IP™ / Modbus-TCP

Doc. Id. HMSI-27-314
Rev. 3.10

Connecting Devices™

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

HMS Industrial Networks
Mailing address: Box 4126, 300 04 Halmstad, Sweden
Visiting address: Stationsgatan 37, Halmstad, Sweden

E-mail: info@hms-networks.com
Web: www.anybus.com

Important User Information

This document contains a general introduction as well as a description of the technical features provided by the Anybus Communicator, including the PC-based configuration software.

The reader of this document is expected to be familiar with PLC and software design, as well as communication systems in general. The reader is also expected to be familiar with the Microsoft® Windows® operating system.

Liability

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

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

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

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

Intellectual Property Rights

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

Trademark Acknowledgements

Anybus® is a registered trademark of HMS Industrial Networks AB. Microsoft® and Windows® are registered trademarks of Microsoft, Inc. EtherNet/IP™ and ODVA™ are trademarks of ODVA, Inc. All other trademarks are the property of their respective holders.

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

Anybus Communicator EtherNet/IP / Modbus-TCP User Manual
Copyright© HMS Industrial Networks AB
Doc: HMSI-27-314

Table of Contents

Preface About This Document

Chapter 1 About the Anybus Communicator for EtherNet/IP

Chapter 2 Basic Operation

Chapter 3 File System

Chapter 4 FTP Server

Chapter 5 Telnet Server

Chapter 6 Basic Network Configuration

Chapter 7 Web Server

Chapter 8 Server Side Include (SSI)

Chapter 9 E-mail Client

Chapter 10 Navigating ACM

Chapter 22 CIP Object Implementation

Chapter 23 Advanced Fieldbus Configuration

Appendix A Parameter Data Initialization (Explicit Data)

Appendix B Connector Pin Assignments

Appendix C Technical Specification

Appendix D Troubleshooting

Appendix E ASCII Table

Appendix F Copyright Notices

P. About This Document

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

P.1 Related Documents

Document name	Author
ABC-EIP Installation Leaflet	HMS
DF1 Protocol and Command Set - Reference Manual, 1770-6.5.16, October 1996	Allen-Bradley
Open Modbus-TCP Specification, Release 1.0	Schneider Electric
RFC 821	Network Working Group
RFC 1918	Network Working Group
ENIP Specifications	ODVA

P.2 Document History

Summary of Recent Changes (3.03... 3.10)

Change	Page(s)
Screenshots and descriptions of ABC Tool updated for Anybus Configuration Manager	Multiple
Changed “ABC” to “Communicator RS232/422/485”	Multiple
Amended description of “Update time” parameter	71, 72
Added description for Consume/Response to “Object Delimiter” parameter	79
Changed “Maximum Data Length” limit	79
Removed obsolete “Start Bits” parameter	88
Removed obsolete “ABCC ExtLink Wizard” entry	100
Replaced “Sales and Support” info with link to website	8
Added parameters to checksum object description	80
Minor text edits, typo corrections	Multiple
Updated screenshots in examples	120, 122

Revision List

Revision	Date	Author	Chapter	Description
2.00	2006-03-27	PeP	All	1st release
2.01	2006-12-22	PeP	All	Misc. minor corrections
2.02	2008-02-08	PeP	2, 8, A	Minor update
2.03	2008-11-03	HeS	1	Minor update
2.04	2009-04-24	KeL	All	Misc. minor corrections and updates
3.00	2011-02-01	KaD	All	Misc. corrections, new template and DF1 functionality
3.01	2011-09-30	KaD	All	Misc corrections and updates, new Anybus Configuration Manager name
3.02	2011-11-15	KaD	P, 2, 3, 6, 8	Minor corrections and updates
3.03	2012-06-08	KaD	P, 8, 22	Minor updates
3.10	March 2015	ThN	All	Misc. corrections and updates, new Doc. ID.

P.3 Conventions & Terminology

The following conventions are used throughout this document:

- Numbered lists provide sequential steps
- Bulleted lists provide information, not procedural steps
- The term 'user' refers to the person or persons responsible for installing the Anybus Communicator in a network.
- The term 'ABC' refers to the Anybus Communicator.
- Hexadecimal values are written in the format 0xNNNN, where NNNN is the hexadecimal value.
- Decimal values are represented as NNNN where NNNN is the decimal value
- As in all communication systems, the terms "input" and "output" can be ambiguous, because their meaning depend on which end of the link is being referenced. The convention in this document is that "input" and "output" are always being referenced to the master/scanner end of the link.

P.3.1 Glossary

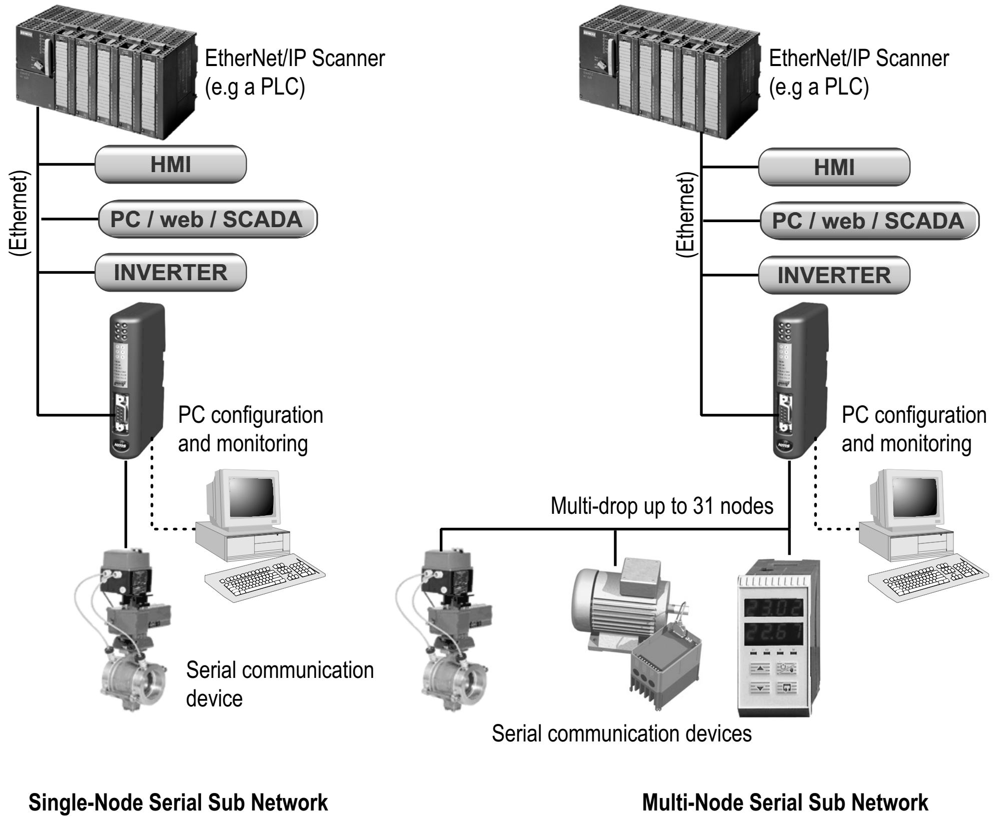
Term	Meaning
ABC	Anybus Communicator
ACM	Anybus Configuration Manager
EIP	EtherNet/IP
Broadcaster	A protocol specific node in the sub-network scan that hold transactions destined to all nodes
Command	A protocol specific transaction.
Configuration	List of configured nodes with transactions on the sub-network
Fieldbus	The network to which the communicator is connected.
Frame	Higher level series of bytes forming a complete telegram on the sub-network
Monitor	A tool for debugging the Anybus Communicator and the network connections
Node	A device in the scan-list that defines the communication with a slave on the sub-network
Scan list	List of configured slaves with transactions on the sub-network
sub-network	The network that logically is located on a subsidiary level with respect to the fieldbus and to which the Anybus Communicator acts as a gateway
Transaction	A generic building block that is used in the sub-network scan-list and defines the data that is sent out the sub-network
Fieldbus Control System	Fieldbus master
Higher Level Network	In this case, Ethernet (including EtherNet/IP and Modbus-TCP)
Network	
Fieldbus	

P.4 Sales and Support

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

1. About the Anybus Communicator for EtherNet/IP

The Anybus Communicator for EtherNet/IP acts as a gateway between virtually any serial application protocol and an EtherNet/IP-based network. Integration of industrial devices is enabled with no loss of functionality, control and reliability, both when retro-fitting to existing equipment as well as when setting up new installations.



Sub-network

The Anybus Communicator can address up to 31 nodes, and supports the following physical standards:

- RS-232
- RS-422
- RS-485

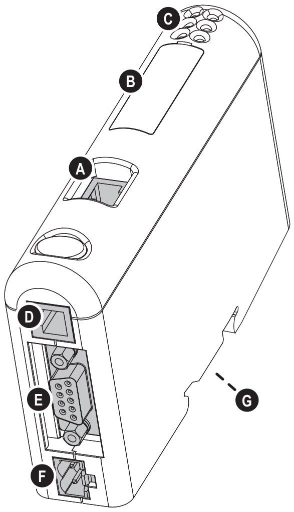
Ethernet Interface

Ethernet connectivity is provided through the patented Anybus technology; a proven industrial communication solution used all over the world by leading manufacturers of industrial automation products.

- EtherNet/IP group 2 and 3 server
- Modbus-TCP slave functionality
- Server Side Include (SSI) functionality
- Web server and E-mail client capabilities
- FTP & Telnet servers
- 10/100 Mbit/s, twisted pair

1.1 External View

For wiring and pin assignments, see "Connector Pin Assignments" on page 124.



A: Ethernet Connectors

These connectors are used to connect the Anybus Communicator to the network.

See also...

- "Ethernet Connector" on page 124

B: Configuration Switches

See also...

- "Configuration Switches" on page 14

C: Status LEDs

See also...

- "Status LEDs" on page 13

D: PC-connector

This connector is used to connect the gateway to a PC for configuration and monitoring purposes.

See also...

- "PC Connector" on page 125

E: Sub-network Connector

This connector is used to connect the gateway to the serial sub-network.

See also...

- "Sub-network Interface" on page 126

F: Power Connector

This connector is used to apply power to the gateway.

See also...

- "Power Connector" on page 124

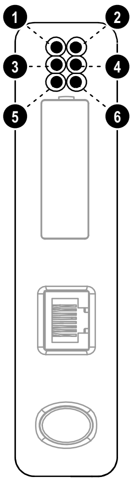
G: DIN-rail Connector

The DIN-rail mechanism connects the gateway to PE (Protective Earth).

See also...

- "Configuration Switches" on page 14

1.2 Status LEDs

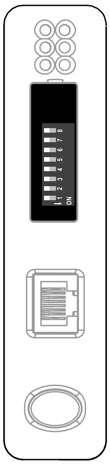


#	State	Status
1 - Module Status *(EtherNet/IP only)*	Off	No power
	Green	Controlled by a scanner in run state
	Green, flashing	Not configured, or scanner in idle state
	Red	Major fault (unrecoverable)
	Red, flashing	Minor fault (recoverable)
	Alternating Red/Green	Self-test
2 - Network Status *(EtherNet/IP only)*	Off	No IP address (or no power)
	Green	Online, EtherNet/IP connection(s) established
	Green, flashing	Online, no EtherNet/IP connections established
	Red	Duplicate IP address detected, fatal error
	Red, flashing	One or more connections timed out
	Alternating Red/Green	Self-test
3 - Link	Off	No link (or no power)
	Green	Connected to an ethernet network
4 - Activity	Off	No ethernet activity (or no power)
	Green	Receiving or transmitting ethernet packet
5 - Subnet Status[a]	Off	(no power)
	Green, flashing	Running correctly, but one or more transaction error(s) have occurred
	Green	Running
	Red	Transaction error/timeout or subnet stopped
6 - Device Status	Off	(no power)
	Alternating Red/Green	Invalid or missing configuration
	Green	Initializing
	Green, flashing	Running
	Red	Bootloader mode[b]
	Red, flashing	If the Device Status LED is flashing in a sequence starting with one or more red flashes, please note the sequence pattern and contact HMS support.

[a] This LED shows green when all transactions have been active at least once. This includes any transactions using “change of state” or “change of state on trigger”. If a timeout occurs on a transaction, this LED will show red.

[b] The gateway is in bootloader mode, and firmware must be restored in order for it to work properly. Start up the Anybus Configuration Manager and connect to the Anybus Communicator. Select **Tools/Options/Module**. Click **Factory Restore** to restore firmware. See “Tools” on page 61.

1.3 Configuration Switches



If set to a nonzero value, the configuration switches forces the Anybus Communicator to use an IP address in the range 192.168.0.1 - 192.168.0.255.

If set to zero, these settings are specified by the system file ‘ethcfg.cfg’, or by settings in Anybus Configuration Manager.

Note that the switches are read once during startup; any changes require a reset in order to have effect.

See also...

- “Basic Network Configuration” on page 37
- “Basic Settings” on page 64

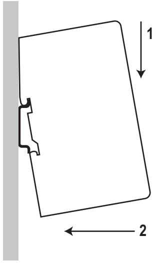
SW1	SW2	SW3	SW4	SW5	SW6	SW7	SW8	DHCP	Subnet	Gateway	IP
OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	(settings determined by ‘ethcfg.cfg’)			
OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF	255.255.255.0	192.168.0.255	192.168.0.1
OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF	OFF	255.255.255.0	192.168.0.255	192.168.0.2
...	...	...	...	...	...	...	...	...	...	...	...
ON	ON	ON	ON	ON	ON	ON	OFF	OFF	255.255.255.0	192.168.0.255	192.168.0.254
ON	ON	ON	ON	ON	ON	ON	ON	(invalid setting)			

1.4 Hardware Installation

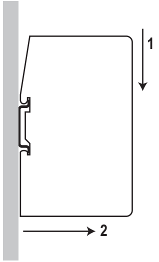
Perform the following steps to install the Anybus Communicator module:

1. Snap the gateway on to the DIN-rail.

   The DIN-rail mechanism works as follows:



To snap the ABC *on*, first press it downwards (1) to compress the spring in the DIN-rail mechanism, then push it against the DIN-rail as to make it snap on (2)



To snap the ABC *off*, push it downwards (1) and pull it out from the DIN-rail (2), as to make it snap off from the DIN-rail

2. Connect the gateway to an Ethernet network.
3. Connect the gateway to the serial sub-network.
4. Connect the gateway to the PC via the configuration cable.
5. Connect the power cable and apply power.
6. Start the Anybus Configuration Manager program on the PC.

   (The Anybus Configuration Manager software will automatically attempt to detect the serial port. If unsuccessful, select the correct port manually in the “Port”-menu)
7. Configure the gateway using the Anybus Configuration Manager and download the configuration.
8. Set up the EtherNet/IP communication according to the gateway configuration.

1.5 Software Installation

1.5.1 Anybus Configuration Manager

System requirements

- Pentium 133 MHz or higher
- 650 MB of free space on the hard drive
- 32 MB RAM
- Screen resolution 800 x 600 (16 bit color) or higher
- Microsoft Windows® 2000 / XP / Vista / 7 (32- or 64-bit)
- Internet Explorer 4.01 SP1 or newer (or any equivalent browser)

Installation

- **Anybus Communicator resource CD**
  - Insert the CD and follow the on-screen instructions.
  - If the installation does not start automatically: right-click on the CD drive icon and select "Explore" to show the contents of the CD. Locate the installation executable and double-click on it to start the installation, then follow the on-screen instructions.
- **From HMS website**
  - Download the latest version of Anybus Configuration Manager from www.anybus.com.
  - Unzip the archive on your computer and double-click on the installation executable.

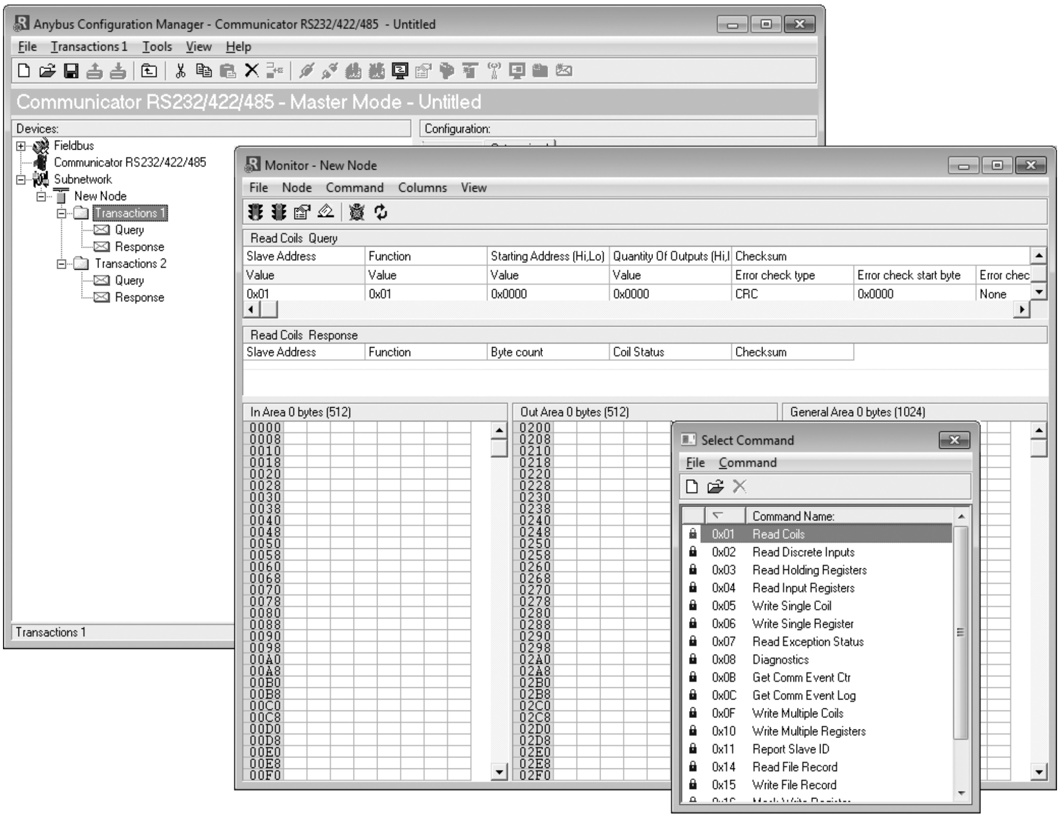
2. Basic Operation

2.1 General

The Anybus Communicator is designed to exchange data between a serial sub-network and a higher level network. Unlike most other similar devices, the Anybus Communicator has no fixed protocol for the sub-network, and consequently can be configured to handle almost any form of serial communication.

The gateway can issue serial telegrams cyclically, on change of state, or based on trigger events issued by the control system in the higher level network (i.e. the fieldbus master or PLC). It can also monitor certain aspects of the sub-network communication and notify the higher level network when data has changed.

An essential part of the Anybus Communicator package is Anybus Configuration Manager (ACM), a Windows-based application used to supply the gateway with a description of the sub-network protocol. No programming skills are required; instead, a visual protocol description-system is used to specify the different parts of the serial communication.


Anybus Configuration Manager - Communicator RS232/422/485 - Untitled
File Transactions 1 Tools View Help
Communicator RS232/422/485 - Master Mode - Untitled
Devices:
Configuration:
Fieldbus
Communicator RS232/422/485
Subnetwork
New Node
Transactions 1
Query
Response
Transactions 2
Query
Response
Transactions 1
Monitor - New Node
File Node Command Columns View
Select Command
File Command

Read Coils Query

Slave Address	Function	Starting Address (Hi,Lo)	Quantity Of Outputs (Hi,l	Checksum		
Value	Value	Value	Value	Error check type	Error check start byte	Error chec
0x01	0x01	0x0000	0x0000	CRC	0x0000	None

Read Coils Response

Slave Address	Function	Byte count	Coil Status	Checksum

In Area 0 bytes (512)

Out Area 0 bytes (512)

General Area 0 bytes (1024)

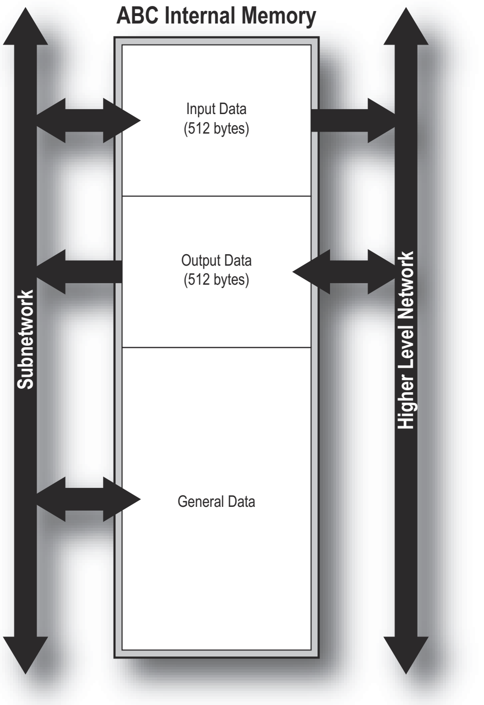
		Command Name:
🔒	0x01	Read Coils
🔒	0x02	Read Discrete Inputs
🔒	0x03	Read Holding Registers
🔒	0x04	Read Input Registers
🔒	0x05	Write Single Coil
🔒	0x06	Write Single Register
🔒	0x07	Read Exception Status
🔒	0x08	Diagnostics
🔒	0x0B	Get Comm Event Ctr
🔒	0x0C	Get Comm Event Log
🔒	0x0F	Write Multiple Coils
🔒	0x10	Write Multiple Registers
🔒	0x11	Report Slave ID
🔒	0x14	Read File Record
🔒	0x15	Write File Record

2.2 Data Exchange Model

Internally, data exchanged on the sub-network and on the higher level network all resides in the same memory.

This means that in order to exchange data with the sub-network, the higher level network simply reads and writes data to the memory locations specified using the Anybus Configuration Manager. The very same memory locations can then be exchanged on the sub-network.

The internal memory buffer is divided into three areas, based on function:



- **Input Data (512 bytes)**

  This area can be read by the higher level network, the web server and the E-mail client.

  (Data representation on the higher level network is described later in this chapter).

- **Output Data (512 bytes)**

  This area can be read/written to by the higher level network, the web server and the E-mail client.

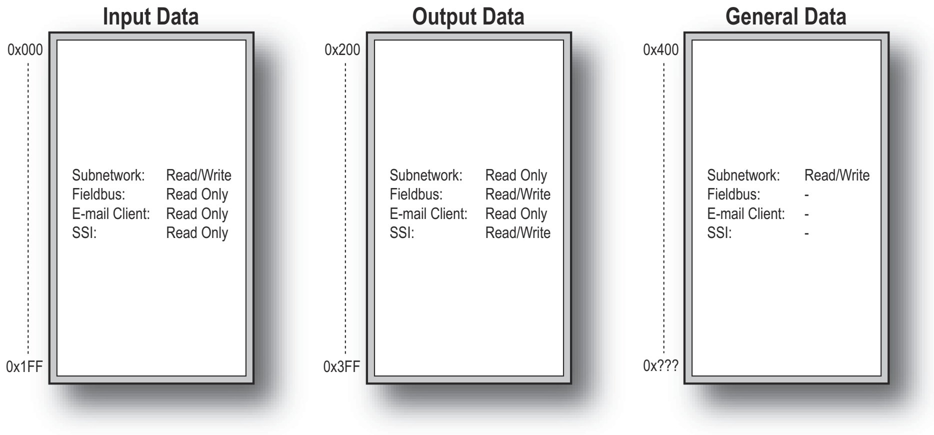
  (Data representation on the higher level network is described later in this chapter).

- **General Data (up to 1024 bytes)**

  This area cannot be accessed from the higher level network, but can be used for transfers between individual nodes on the sub-network, or as a general "scratch pad" for data. The actual size of this area depends on the amount of data that is exchanged on the sub-network. The gateway can handle up to 1024 bytes of general data.

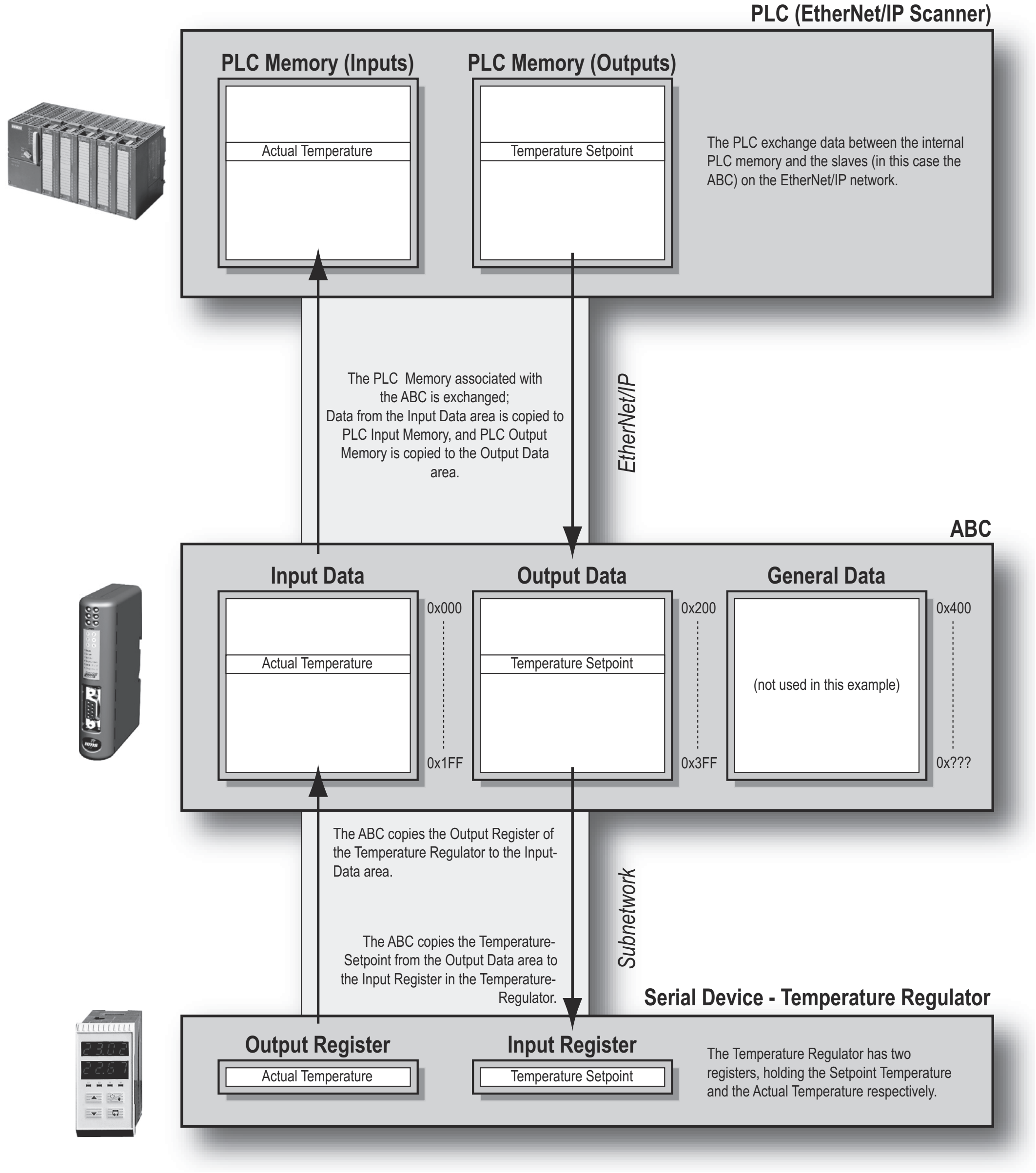
2.2.1 Memory Map

When building the sub-network configuration using the Anybus Configuration Manager, the different areas described above are mapped to the memory locations (addresses) specified below.



2.2.2 Data Exchange Example

In the following example, a temperature regulator on the sub-network exchanges information with a PLC on the higher level network, via the internal memory buffers in the Anybus Communicator.



The PLC exchange data between the internal PLC memory and the slaves (in this case the ABC) on the EtherNet/IP network.

The PLC Memory associated with the ABC is exchanged; Data from the Input Data area is copied to PLC Input Memory, and PLC Output Memory is copied to the Output Data area.

The ABC copies the Output Register of the Temperature Regulator to the Input-Data area.

The ABC copies the Temperature-Setpoint from the Output Data area to the Input Register in the Temperature-Regulator.

The Temperature Regulator has two registers, holding the Setpoint Temperature and the Actual Temperature respectively.

2.3 Sub-network Protocol

2.3.1 Protocol Modes

The Anybus Communicator features three distinct operating modes for sub-network communication: ‘Master Mode’, ‘DF1 Master Mode’ and ‘Generic Data Mode’. Note that the protocol mode only specifies the basic communication model, not the actual sub-network protocol.

- **Master Mode**

  In this mode, the gateway acts as a master on the sub-network, and the serial communication is query-response based. The nodes on the network are not permitted to issue messages unless first addressed by the gateway .

  For more information about this mode, see “Master Mode” on page 21.

- **DF1 Master Mode**

  In this mode, the gateway acts as a master on the sub-network, using the DF1 protocol. The serial communication is query-response based. For more information about this mode, see “DF1 Protocol Mode” on page 86.

- **Generic Data Mode**

  In this mode, there is no master-slave relationship between the sub-network nodes and the gateway; any node on the sub-network, including the gateway, may spontaneously produce or consume messages.

  For more information about this mode, see “Generic Data Mode” on page 22.

2.3.2 Protocol Building Blocks

The following building blocks are used in Anybus Configuration Manager to describe the sub-network communication. How these blocks apply to the three protocol modes is described later in this document.

- **Node**

  A ‘node’ represents a single device on the sub-network. Each node can be associated with a number of transactions, see below.

- **Transaction**

  A ‘transaction’ represents a complete serial telegram, and consists of a number of frame objects (see below). Each transaction is associated with a set of parameters controlling how and when to use it on the sub-network.

- **Commands**

  A ‘command’ is simply a predefined transaction stored in a list in the Anybus Configuration Manager. This simplifies common operations by allowing transactions to be stored and reused.

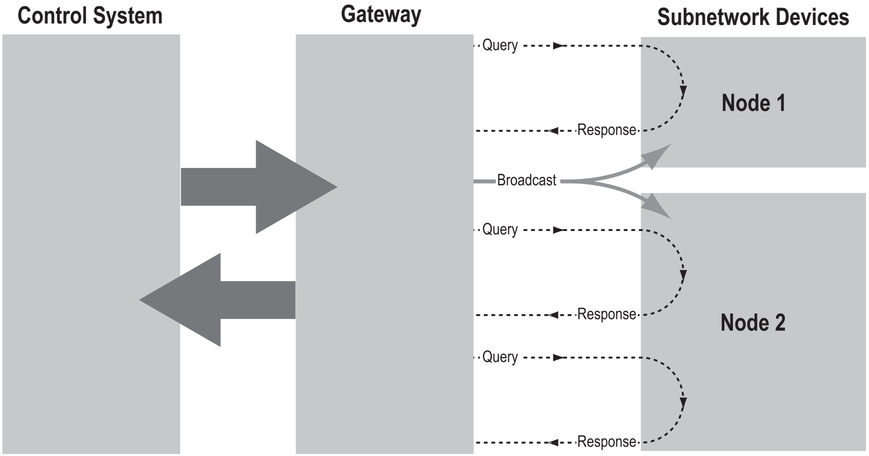
- **Frame Object**

  ‘Frame objects’ are low level entities used to compose a transaction (see above). A frame object can represent a fixed value (a constant), a range of values (limit objects), a block of data or a calculated checksum.

2.3.3 Master Mode

In this mode, the communication is based on a query-response scheme; when the gateway issues a query on the sub-network, the addressed node is expected to issue a response. Nodes are not permitted to issue responses/messages spontaneously, i.e. without first receiving a query.

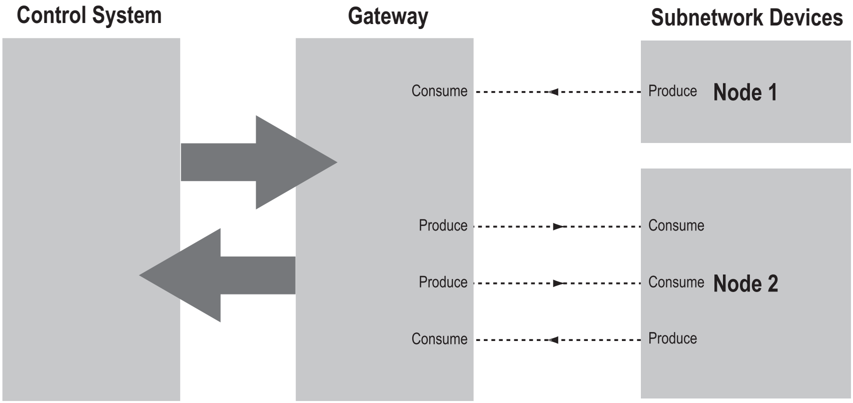
There is, however, one exception to this rule; the broadcaster. Most protocols offer some way of broadcasting messages to all nodes on the network, without expecting them to respond to the broadcasted message. This is also reflected in the gateway, which features a dedicated broadcaster node.



In Master Mode, Anybus Configuration Manager comes pre-loaded with the most commonly used Modbus RTU commands, which can be conveniently reached by right-clicking on a node in the Anybus Configuration Manager and selecting 'Insert New Command'. Note, however, that this in no way prevents other protocols based on the same query-response message-scheme from also being implemented.

2.3.4 Generic Data Mode

In this mode, there is no master-slave relationship between the nodes on the sub-network and the gateway. Any node (including the gateway) may spontaneously produce or consume a message. Nodes are not obliged to respond to messages, nor do they need to wait for a query in order to send a message.



In the figure above, the Anybus Communicator 'consumes' data 'produced' by a node on the sub-network. This 'consumed' data can then be accessed from the higher level network. This also works the other way around; the data received from the higher level network is used to 'produce' a message on the sub-network, for 'consumtion' by a node.

2.3.5 DF1 Master Mode

Please refer to "DF1 Protocol Mode" on page 86.

2.4 EtherNet/IP

2.4.1 General

EtherNet/IP is based on the Control and Information Protocol (CIP), which is also the application layer for DeviceNet and ControlNet. The Anybus Communicator acts as a Group 2 or 3 server on the EtherNet/IP network.

Input and output data is accessed using I/O connections or explicit messages towards the assembly object and the parameter input/output mapping objects.

See also...

- "CIP Object Implementation" on page 107

2.4.2 Data Types

The input and output data hold two types of data; I/O data and parameter data. I/O data is exchanged on change of value, and can be accessed using I/O connections towards the assembly object.

Parameter data can be accessed acyclically via the parameter input and output mapping objects. Note, however, that each instance attribute within these objects must be created manually using the Anybus Configuration Manager.

For more information see "Parameter Data Initialization (Explicit Data)" on page 119.

See also...

- "Assembly Object, Class 04h" on page 110
- "Parameter Data Input Mapping Object, Class B0h" on page 112
- "Parameter Data Output Mapping Object, Class B1h" on page 113
- "Fieldbus Settings" on page 64

2.4.3 Memory Layout

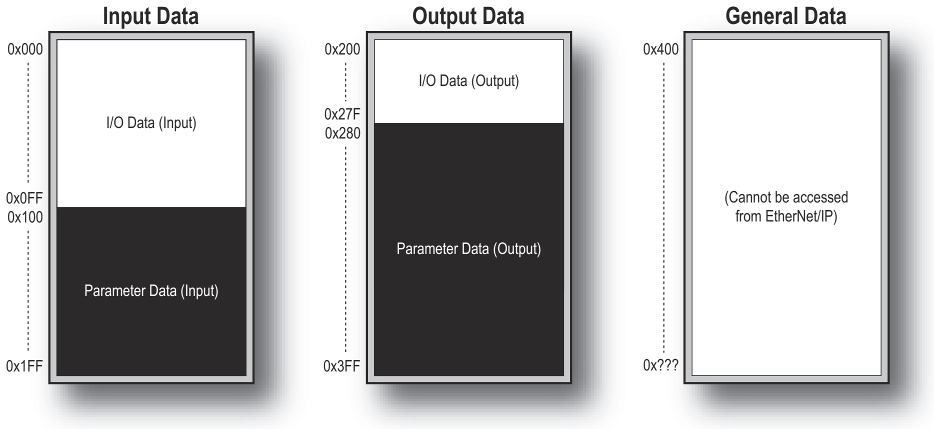
I/O sizes are specified using the Anybus Configuration Manager and correlate to the Anybus Communicator memory as follows:

Example:

I/O Sizes for the gateway set to the following values:

IO Size In= 256 bytes (0x0100)
IO Size Out= 128 bytes (0x0080)

Resulting memory layout:



2.5 Modbus-TCP

2.5.1 General

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 input and output data areas via a subset of the functions defined in the Modbus-TCP specification.

The server supports up to 8 simultaneous connections and communicates over TCP port 502. For detailed information regarding the Modbus-TCP protocol, consult the Open Modbus Specification.

2.5.2 Addressing Modes

The Anybus Communicator features two different modes of operation regarding the Modbus communication:

- **Modbus Addressing Mode (Default)**

  In this mode, the input and output data areas are mapped to different function codes.

  Note that coil addressing is not possible in this mode.

  See also...

  - "Modbus Addressing Mode" on page 26

- **Anybus Addressing Mode**

  Compared to Modbus Addressing Mode, this mode allows data to be addressed in a more flexible way. Note however that several function codes can be used to access the same data in the gateway. While this may appear confusing at first, it allows data to be manipulated in ways not possible in Modbus Addressing Mode (e.g. it is possible to manipulate individual bits of a register by accessing coils associated with the same memory location).

  See also...

  - "Anybus Addressing Mode" on page 27

2.5.3 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

2.5.4 Modbus Addressing Mode

Supported Function Codes

The following function codes can be used in this mode:

Modbus Function	Function Code	Associated with Area	No. of I/Os or Data Points per Command
Read Holding Registers	3	Output Data area (0x200...0x3FF)	1 - 125 registers
Read Input Registers	4	Input Data area (0x000....0x1FF)	1 - 125 registers
Write Single Register	6	Output Data area (0x200...0x3FF)	1 register
Force Multiple Registers	16		1 - 800 registers
Mask Write Register	22		1 register
Read/Write Registers	23		125 registers read / 100 registers write

Input Register Map

The input data area is mapped to input registers as follows:

Register #	Memory Location in the gateway	Comments
1	0x000... 0x001	Each register corresponds to two bytes in the input data area.
2	0x002... 0x003	
3	0x004... 0x005	
4	0x006... 0x007	
5	0x008... 0x009	
6	0x00A... 0x00B	
...	...	
255	0x1FC... 0x1FD	
256	0x1FE... 0x1FF	

Holding Register Map

The output data area is mapped to holding registers as follows:

Register #	Memory Location in the gateway	Comments
1	0x200... 0x201	Each register corresponds to two bytes in the output data area.
2	0x202... 0x203	
3	0x204... 0x205	
4	0x206... 0x207	
5	0x208... 0x209	
6	0x20A... 0x20B	
...	...	
255	0x3FC... 0x3FD	
256	0x3FE... 0x3FF	

2.5.5 Anybus Addressing Mode

Supported Function Codes

The following function codes can be used in this mode:

Modbus Function	Function Code	Associated with Area(s)	No. of I/Os or Data Points per Command
Read Coil	1	Input and Output Data Area (0x000... 0x3FF)	1 - 2000 bits
Read Input Discretes	2		1 - 2000 bits
Read Holding Registers	3		1 - 125 registers
Read Input Registers	4		1 - 125 registers
Write Coil	5	Output Data Area (0x200... 0x3FF)	1 bit
Write Single Register	6		1 register
Force Multiple Coils	15		1 - 800 bits
Force Multiple Registers	16		1 - 100 registers
Mask Write Register	22		1 register
Read/Write Registers	23	Input and Output Data Area (0x000... 0x3FF)	125 registers read/100 registers write

Coil & Register Map

The input and output data areas are mapped to coils and registers as follows:

Register #	Coil #	Memory Location in ABC	Area	Comments
1	1... 16	0x000... 0x001	Input Data area	-
2	17... 32	0x002... 0x003		
3	33... 48	0x004... 0x005		
4	49... 64	0x006... 0x007		
...	...	...		
255	4065... 4080	0x1FC... 0x1FD		
256	4081... 4096	0x1FE... 0x1FF		
257 ... 1024	4097... 4112 ... 16369... 16384	-	-	(reserved)
1025	16385... 16400	0x200... 0x201	Output Data area	-
1026	16401... 16416	0x202... 0x203		
1027	16417... 16432	0x204... 0x205		
1028	16433... 16448	0x206... 0x207		
...	...	...		
1279	20449... 20464	0x3FC... 0x3FD		
1280	20465... 20480	0x3FE... 0x3FF		

Note 1: The table above applies to all function codes.

Note 2: Coils are mapped MSB first, i.e. coil 0 corresponds to bit 15 of register 0.

3. File System

3.1 General

General

The Anybus Communicator features a built-in file system, which is used to store information such as web files, network communication settings, e-mail messages etc.

Storage Areas

The file system consists of the different storage areas:

- **Non-volatile area (approx. 1.4 Mb)**
  This section is intended for static files such as web files, configurations files etc.

- **Volatile area (approx. 1 Mb)**
  This area is intended for temporary storage; data placed here will be lost in case of power loss or reset.

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 may not contain 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

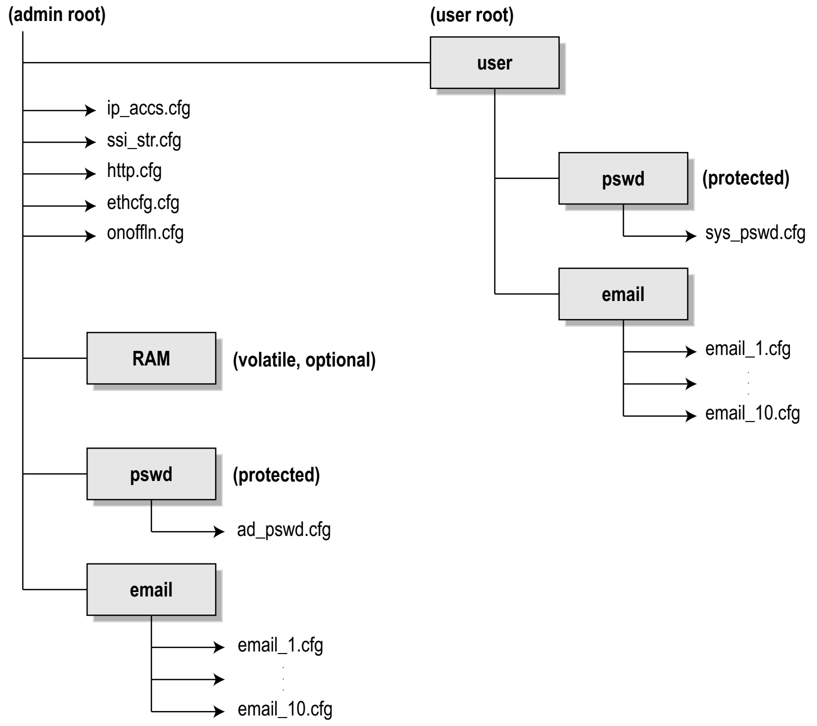
Important Note:

The non-volatile storage is located in FLASH memory. Each FLASH segment can be erased approximately 100 000 times.

The following operations will erase one or more FLASH segments:

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

3.2 File System Overview



3.3 System Files

The file system contains a set of files used for system configuration. These files, known as "system files", are regular ASCII files that 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 gateway itself, e.g. when using SSI (see "Server Side Include (SSI)" on page 45).

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.

The contents of the above files can be redirected:

Example:

In this example, the contents will be loaded from the file 'here.cfg'.

```
[file path]
\i\put\it\over\here.cfg
```

Note: Any directory in the file system can be protected from web access by placing the file web-accs.cfg in the directory, see "Authorization" on page 43.

4. FTP Server

4.1 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)

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, the gateway will grant admin level access to all users. In such cases, 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.

4.2 FTP Connection Example (Windows Explorer)

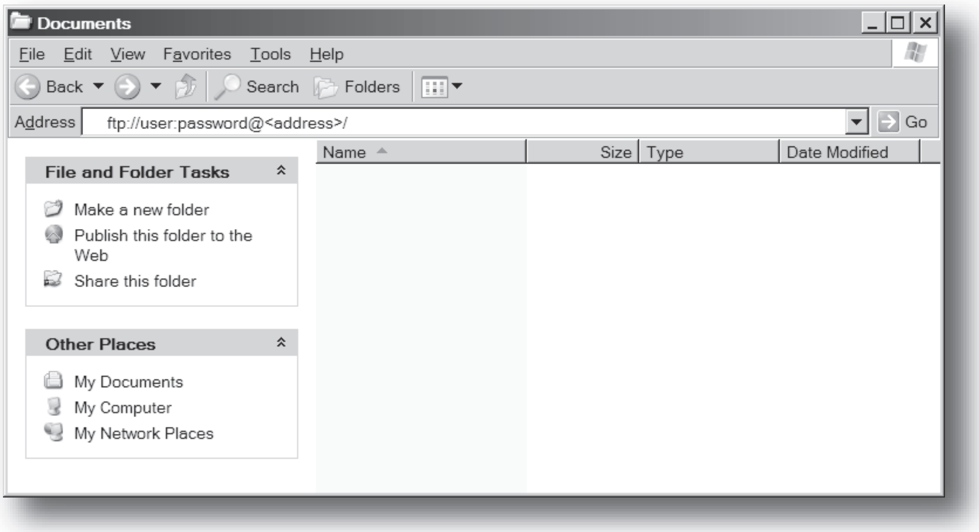
The built-in FTP client in Windows Explorer can easily be used to access the file system as follows:

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 Communicator
- Substitute <user> with the username
- Substitute <password> with the password

3. Press enter. The Explorer will now attempt to connect to the gateway using the specified settings. If successful, the built in file system is displayed in the Explorer window.



Address: ftp://user:password@<address>/

5. Telnet Server

5.1 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.

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, the gateway will grant admin level access to all users. In such case, no login is required, and the root directory will be '\'.

For more information, see "User Accounts" on page 30

5.2 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 Ethernet-module, in the Communicator.

5.3 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

5.4 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 is a privileged command and can only be called in administration mode.

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.

6. Basic Network Configuration

6.1 General Information

The Anybus Communicator offers two modes of operation regarding the network settings:

- **Settings specified by Configuration Switches**

  If the on-board switches are set to a non-zero value, the ABC is locked to the following settings:

  IP Address:192.168.0.x(x = switch value)
  Gateway:255.255.255.0
  Subnet:255.255.255.0
  DHCP:OFF

  See also...

  - "Configuration Switches" on page 14

- **Settings specified in Anybus Configuration Manager**

  When valid settings have been specified in Anybus Configuration Manager ('TCP/IP Settings' = enabled), then these are the settings the gateway will use.

  When settings have been specified in Anybus Configuration Manager, the contents of the system file 'ethcfg.cfg' will be ignored completely, causing the following behavior:

  - DNS services will not be available
  - Domain and Host name cannot be set
  - E-mail services will not be available
  - Network settings received via HICP or DCP) will be lost in the event of a power loss or a reset.

- **Settings specified in 'ethcfg.cfg'**

  If no settings are specified in Anybus Configuration Manager (i.e. 'TCP/IP Settings' = disabled), the gateway will use the settings stored in the system file 'ethcfg.cfg'.

  If this file is missing, the gateway will attempt to retrieve the settings via DHCP or HICP for 30 seconds. If no configuration has been received within this period, the gateway will halt and indicate an error on the on-board LEDs.

EtherNet/IP

The TCP/IP settings can be accessed from EtherNet/IP through the TCP/IP Interface Object.

See also...

- "TCP/IP Interface Object, Class F5h" on page 115

DHCP/BootP

The Anybus Communicator can retrieve the TCP/IP settings from a DHCP or BootP server. If no DHCP server is found, the gateway will default to the current settings in '\ethcfg.cfg'.

If no current settings are available ('ethcfg.cfg' is missing, or contains invalid settings), the gateway will halt and indicate an error on the on-board status LEDs (the network configuration may however still be accessed via HICP, see "Anybus IPconfig (HICP)" on page 41.

6.2 Ethernet Configuration File ('ethcfg.cfg')

6.2.1 General

To exist on the network, the Anybus Communicator needs a valid TCP/IP configuration. These settings are stored in the system file '\ethcfg.cfg'. Note that if TCP/IP settings are enabled in Anybus Configuration Manager, then the IP address, gateway and subnet settings in ethcfg.cfg will be overwritten every time the module is restarted. All other settings are unaffected.

File Format:

```
[IP address]
xxx.xxx.xxx.xxx

[Subnet mask]
xxx.xxx.xxx.xxx

[Gateway address}
xxx.xxx.xxx.xxx

[DHCP/BOOTP]
ON or OFF

[SMTP address]
xxx.xxx.xxx.xxx

[SMTP username]
username

[SMTP password]
password

[DNS1 address]
xxx.xxx.xxx.xxx

[DNS2 address]
xxx.xxx.xxx.xxx

[Domain name]
domain

[Host name]
anybus

[HICP password]
password
```

- **IP address**
- **Subnet mask**
- **Gateway address**
- **DHCP/BootP**
  ON - Enabled
  OFF - Disabled
- **SMTP server/login settings**
  Username and Password is only necessary if required by the server.
- **Primary and Secondary DNS**
  Needed to be able to resolve host names
- **Default domain name for not fully qualified host names**
- **Host name**
- **HICP password**

The settings in this file may also be affected by...

- EtherNet/IP (See "EtherNet/IP" on page 37).
- HICP (See "Anybus IPconfig (HICP)" on page 41)
- SSI (See "Server Side Include (SSI)" on page 45)

See also...

- "FTP Server" on page 30
- "Fieldbus Settings" on page 64

6.3 IP Access Control

It is possible to specify which IP addresses are permitted to connect to the Anybus Communicator. 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 gateway via Modbus-TCP

`[EtherNet/IP]`
`xxx.xxx.xxx.xxx`
	Nodes listed here may access the gateway via EtherNet/IP

`[All]`
`xxx.xxx.xxx.xxx`
	Fallback setting, used by the gateway when one or several of the keys above are omitted

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

6.4 On/Offline Configuration

By default, the On/Offline indication is triggered by the link status. Other triggering options can however be specified in the optional system file '\onoffln.cfg', which should be placed in the module root and looks as follows:

File Format:

```
[ON/OFF-line trigger]
Modbus

[Timeout]
10

[Commands]
3, 16, 23

[ON-line method]
1
```

- On/Offline trigger source

  Values: 'Link' (default), 'EIP', 'Modbus' or a combination

- Timeout Value

  Range: 1... 65535 (default = 1).

  A value of 10 equals 1000 ms.

- Commands (Optional)

  Selects what Modbus commands that must be received during the timeout period.

  If the keyword 'ALL' is given (default), the On/Offline functionality will trigger on all Modbus commands.

- Online method (Optional)

  Defines how to handle data in the OUT I/O area when going from Offline to Online.

  If "1" (default), "old data" is restored
  If "2", "Offline" data is kept until overwritten by master.

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.

File example:

```
[File path]
\my_settings\on-off-line_configuration.cfg
```

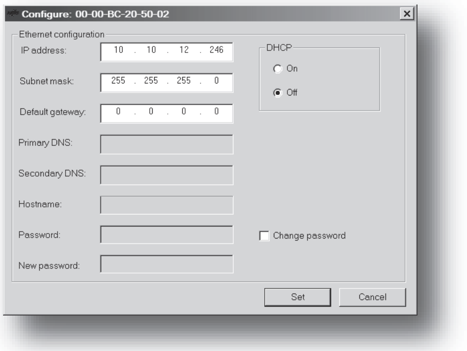
In this example, the settings described above will be loaded from the file '\my_settings\on-off-line_-configuration.cfg'.

Note 1: The keys '[Timeout]' and '[Commands]' shall only be given if the On/Offline Trigger value is set to 'Modbus'.

Note 2: The settings in this file will be ignored if the application has issued the mailbox message MB_ON_OFF_LINE_CONFIG. See "Advanced Fieldbus Configuration" on page 118.

6.5 Anybus IPconfig (HICP)

The Anybus Communicator 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 configuration file (‘ethcfg.cfg’).



Configure: 00-00-BC-20-50-02

Ethernet configuration

IP address: 10 . 10 . 12 . 246

Subnet mask: 255 . 255 . 255 . 0

Default gateway: 0 . 0 . 0 . 0

Primary DNS:

Secondary DNS:

Hostname:

Password:

New password:

DHCP ○ On ● Off

☐ Change password

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 gateway will appear as ‘ABC-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’. The new IP configuration will now be stored in the configuration file (‘ethcfg.cfg’).

Note that if ‘TCP/IP Settings’ has been enabled in the Anybus Configuration Manager, any settings received via HICP will be lost in the event of a power loss or reset.

7. Web Server

7.1 General

The Anybus Communicator 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...

- “Server Side Include (SSI)” on page 45
- “IP Access Control” on page 39

Protected Files

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

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

Default Web Pages

The Anybus Communicator contains a set of virtual files which 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 such case the link shall point to ‘\config.htm’.

These virtual files are:

```
\index.htm                    - Points to the contents of config.htm
\config.htm                   - Configuration frame page
\configform.htm               - Configuration form page
\configform2.htm              - Configuration form page
\store.htm                    - Configuration store page
\logo.jpg                     - HMS logo
\configuration.gif            - Configuration picture
\boarder.bg.gif               - picture
\boarder_m_bg.gif             - picture
\index.htm        l           - Points to the contents of config.htm
\eth_stat.html                - Configuration frame page
\cip_stat.html                - Configuration form page
\ip_config.shtm               - Configuration form page
\smtp_config.shtm             - Configuration store page
\style.css                    - HMS logo
\arrow_red.gif                - Configuration picture
```

7.2 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]
Please enter password
```

Note that when using this feature, make sure to put the user/password files in a directory that is protected from web access, see “Protected Files” on page 42.

7.3 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.

8. Server Side Include (SSI)

General

Server Side Include (from now on referred to as SSI) functionality enables dynamic content to be used on web pages and in e-mail 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 (when applicable).

Syntax

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

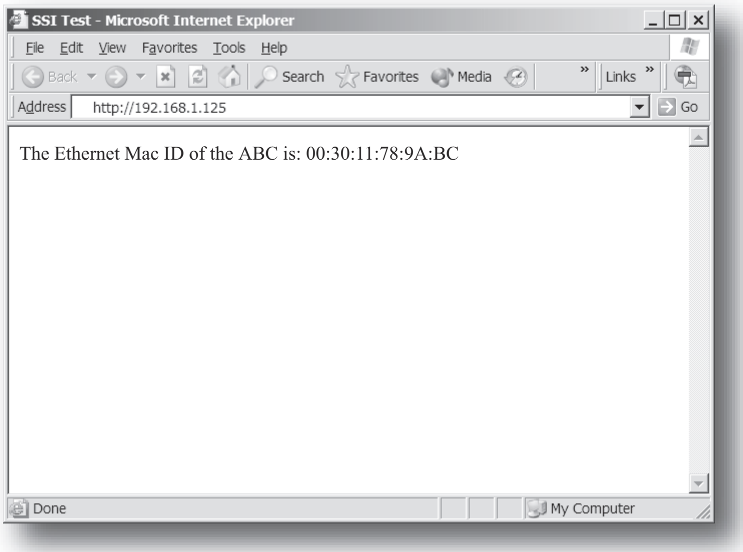
```
<?--#exec cmd_argument='XXXXXXXXXXXXXXXXXXXXXX'-->
```

Example

The following example causes a web page to display the Ethernet Mac ID of the module:

```
<HTML>
<HEAD><TITLE>SSI Test</TITLE></HEAD>
<BODY>
The Ethernet Mac ID of the Anybus module is:
<?--#exec cmd_argument='DisplayMacID'-->
</BODY>
</HTML>
```

Resulting webpage:


SSI Test - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Media
Links
Address
http://192.168.1.125
Go
Done
My Computer

The Ethernet Mac ID of the ABC is: 00:30:11:78:9A:BC

8.1 Functions

DisplayMacID

This function returns the MAC ID in format xx:xx:xx:xx:xx:xx.

Syntax:

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

DisplaySerial

This function returns the serial number of the Anybus module.

Syntax:

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

DisplayFWVersion

This function returns the main firmware revision of the Anybus module.

Syntax:

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

DisplayBLVersion

This function returns the bootloader firmware revision of the Anybus module.

Syntax:

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

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'-->
```

DisplayDNS1

This function returns the address of the primary DNS server.

Syntax:

```
<?--#exec cmd_argument='DisplayDNS1'-->
```

DisplayDNS2

This function returns the address of the secondary DNS server.

Syntax:

```
<?--#exec cmd_argument='DisplayDNS2'-->
```

DisplayHostName

This function returns the hostname.

Syntax:

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

DisplayDomainName

This function returns the default domain name.

Syntax:

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

DisplayDhcpState

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

Syntax:

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

DisplayDhcpSupport

This function returns ‘Arg1’ if it’s enabled and ‘Arg2’ if it’s disabled.

Syntax:

```
<?--#exec cmd_argument='DisplayDhcpSupport( "Arg1", "Arg2" )'-->
```

DisplayEmailServer

This function returns the currently used SMTP server address.

Syntax:

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

DisplaySMTPUser

This function returns the username used for SMTP authentication.

Syntax:

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

DisplaySMTPPswd

This function returns the password used for SMTP authentication.

Syntax:

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

DisplayStationName

This function returns the PROFINET Station Name.

Syntax:

```
<?--#exec cmd:argument='DisplayStationName'-->
```

DisplayStationType

This function returns the PROFINET Station Type.

Syntax:

```
<?--#exec cmd:argument='DisplayStationType'-->
```

DisplayVendorID

This function returns the PROFINET Vendor ID.

Syntax:

```
<?--#exec cmd:argument='DisplayVendorId'-->
```

DisplayDeviceID

This function returns the PROFINET DeviceID.

Syntax:

```
<?--#exec cmd:argument='DisplayDeviceId'-->
```

StoreEtnConfig

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

This SSI function stores a passed IP configuration in the configuration file 'ethcfgIP.cfg'.

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
SetEmailServer
SetDhcpState - value "on" or "off"
SetDNS1
SetDNS2
SetHostName
SetDomainName
SetSMTPUser
SetSMTPPswd
```

Default output:

```
Invalid IP address!
Invalid Subnet mask!
Invalid Gateway address!
Invalid IP address or Subnet mask!
Invalid Email Server IP address!
Invalid DHCP state!
Invalid DNS1!
Invalid DNS2!
Configuration stored correctly.
Failed to store configuration.
```

GetText

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

This SSI function gets the text from an object and stores it in the OUT area.

Syntax:

```
<?--#exec cmd_argument='GetText( "ObjName", OutWriteString ( offset ), n )'-->
```

ObjName- Name of object.
offset - Specifies the offset from the beginning of the OUT area.
n - Specifies maximum number of characters to read (Optional)

Default output:

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

printf

This SSI function includes a formatted string, which may contain data from the Anybus IN/OUT area, on a web page. The formatting of the string is equal 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 behavior is undefined.

Character	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.dddddd e+-xx or[-]m.ddddddE+-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(*offset*)	Read a signed byte from position *offset* in the IN area
InReadUByte(*offset*)	Read an unsigned byte from position *offset* in the IN area
InReadSWord(*offset*)	Read a signed word from position *offset* in the IN area
InReadUWord(*offset*)	Read an unsigned word from position *offset* in the IN area
InReadSLong(*offset*)	Read a signed longword from position *offset* in the IN area
InReadULong(*offset*)	Read an unsigned longword from position *offset* in the IN area
InReadString(*offset*)	Read a string (char*) from position *offset* in the IN area
InReadFloat(*offset*)	Read a floating point (float) value from position *offset* in the IN area
OutReadSByte(*offset*)	Read a signed byte from position *offset* in the OUT area
OutReadUByte(*offset*)	Read an unsigned byte from position *offset* in the OUT area
OutReadSWord(*offset*)	Read a signed word (short) from position *offset* in the OUT area
OutReadUWord(*offset*)	Read an unsigned word (short) from position *offset* in the OUT area
OutReadSLong(*offset*)	Read a signed longword (long) from position *offset* in the OUT area
OutReadULong(*offset*)	Read an unsigned longword (long) from position *offset* in the OUT area
OutReadString(*offset*)	Read a null-terminated string from position *offset* in the OUT area
OutReadFloat(*offset*)	Read a floating point (float) value from position *offset* in the OUT area
MbReadSByte(*id*)	Read a signed byte (short) from the application via the mailbox interface
MbReadUByte(*id*)	Read an unsigned byte (short) from the application via the mailbox interface
MbReadSWord(*id*)	Read a signed word from the application via the mailbox interface
MbReadUWord(*id*)	Read an unsigned word from the application via the mailbox interface
MbReadSLong(*id*)	Read a signed longword from the application via the mailbox interface
MbReadULong(*id*)	Read an unsigned longword from the application via the mailbox interface
MbReadString(*id*)	Read a null-terminated string from the application via the mailbox interface
MbReadFloat(*id*)	Read a floating point (float) value from the application via the mailbox interface
CipReadSByte(*class*, *inst*, *attr*)	Read a signed byte from a CIP-object
CipReadUByte(*class*, *inst*, *attr*)	Read an unsigned byte from a CIP-object
CipReadSWord(*class*, *inst*, *attr*)	Read a signed word from a CIP-object

Argument	Description
CipReadUWord(*class*, *inst*, *attr*)	Read an unsigned word from a CIP-object
CipReadSLong(*class*, *inst*, *attr*)	Read a signed longword from a CIP-object
CipReadULong(*class*, *inst*, *attr*)	Read an unsigned longword from a CIP-object
CipReadFloat(*class*, *inst*, *attr*)	Read a floating point value from a CIP-object
CipReadShortString(*class*, *inst*, *attr*)	Read a short string from a CIP-object
CipReadString(*class*, *inst*, *attr*)	Read a null-terminated string from a CIP-object
CipReadUByteArray(*class*, *inst*, *attr*)	Read an unsigned byte-array from a CIP-object
CipReadUWordArray(*class*, *inst*, *attr*)	Read an unsigned word-array from a CIP-object
CipReadULongArray(*class*, *inst*, *attr*)	Read an unsigned longword-array from a CIP-object

scanf

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

This SSI function reads a string passed from an object in a HTML form, interprets the string according to the specified 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	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	Unsinged 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 *offset* in the OUT area
OutWriteWord(*offset*)	Write a word to position *offset* in the OUT area
OutWriteLong(*offset*)	Write a long to position *offset* in the OUT area
OutWriteString(*offset*)	Write a string to position *offset* in the OUT area
OutWriteFloat(offset)	Write a floating point value to position *offset* in the OUT area
MbWriteByte(*id*)	Write a byte to the application via the mailbox interface
MbWriteWord(*id*)	Write a word to the application via the mailbox interface
MbWriteLong(*id*)	Write a longword to the application via the mailbox interface
MbWriteString(*id*)	Write a string to the application via the mailbox interface
MbWriteFloat(*id*)	Write a floating point value to the application via the mailbox interface
CipWriteByte(*class, inst, attr*)	Write a byte value to a CIP-object
CipWriteWord(*class, inst, attr*)	Write a word value to a CIP-object
CipWriteLong(*class, inst, attr*)	Write a longword to a CIP-object
CipWriteFloat(*class, inst, attr*)	Write a floating point value to a CIP-object

Default output:

```
Write succeeded
Write failed
```

IncludeFile

This SSI function includes the contents of a file on a web page.

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 e-mail 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|Over-
write] )'-->
```

Default output:

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

SaveDataToFile

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

This SSI function saves the data of a passed form to a file. The “Object name” parameter is optional, if specified, only the data from that object will be stored. If not, the data from all objects in the form will be stored.

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='SaveDataToFile( "File name", "Object name",[Ap-
pend|Overwrite] )'-->
```

Default output:

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

DisplayRemoteUser

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

This SSI function returns the user name on an authentication session.

Syntax:

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

8.2 Changing SSI output

There are 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 by calling the SSI function "SsiOutput()".

8.2.1 SSI Output String File

If the file "\ssi_str.cfg" is found in the filesystem and the file is correct 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

8.2.2 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) )'-->
```

9. E-mail Client

9.1 General

The built-in e-mail client can send predefined e-mail messages based on trigger-events in input and output data areas. The client supports SSI, however note that some SSI functions cannot be used in e-mail messages (specified separately for each SSI function).

See also...

- "Server Side Include (SSI)" on page 45

Server Settings

The Anybus Communicator needs a valid SMTP server configuration in order to be able to send e-mail messages. These settings are stored in the system file '\ethcfg.cfg'.

See also...

- "Ethernet Configuration File ('ethcfg.cfg')" on page 38

Event-Triggered Messages

As mentioned previously, the e-mail client can send predefined messages based on events in the input and output data areas. In operation, this works as follows:

1. The trigger source is fetched from a specified location
2. A logical AND is performed between the trigger source and a mask value
3. The result is compared to a reference value
4. If the result is true, the e-mail is sent to the specified recipient(s).

Which events that shall cause a particular message to be sent, is specified separately for each message. For more information, see "E-mail Definitions" on page 58.

Note that the input and output data areas are scanned twice per second, i.e. to ensure that an event is detected by the gateway, it must be present longer than 0.5 seconds.

9.2 E-mail Definitions

The e-mail 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.

E-mail definition files must be named ‘email_1.cfg’, ‘email_2.cfg’... ‘email_10.cfg’ in order to be properly recognized by the gateway.

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
```

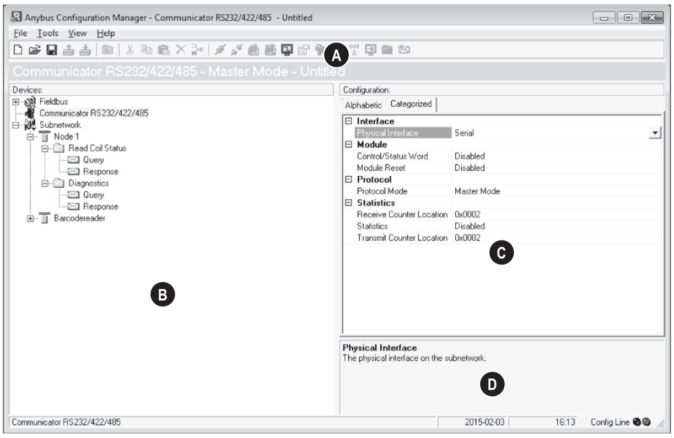
Key	Value	Scanned for SSI
Area	Source area. Possible values: ‘IN’ (Input Data area) or ‘OUT’ (Output Data area)	No
Offset	Source offset, written in decimal or hexadecimal.	
Type	Source data type. Possible values are ‘byte’, ‘word’, and ‘long’	
Value	Used as a reference value for comparison.	
Mask	Mask value, applied on the trigger source prior to comparison (logical AND).	
Operand	Possible values are ‘<‘, ‘=’ or ‘>’	
To	E-mail recipient	Yes
From	Sender e-mail address	
Subject	E-mail 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 Anybus Communicator.

10. Navigating ACM

10.1 Main Window

The main window in ACM can be divided into 4 sections as follows:

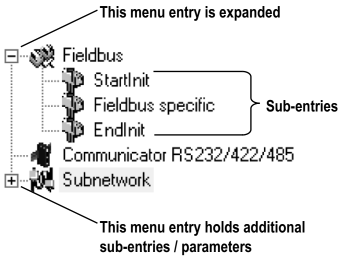


Interface	
Physical Interface	Serial
Module	
Control/Status Word	Disabled
Module Reset	Disabled
Protocol	
Protocol Mode	Master Mode
Statistics	
Receive Counter Location	0x0002
Statistics	Disabled
Transmit Counter Location	0x0002

Physical Interface
The physical interface on the subnetwork.

- **A: Drop-down Menus & Tool Bar**

  The second drop-down menu from the left will change depending on the current context. The Tool Bar provides quick access to the most frequently used functions.


This menu entry is expanded
Fieldbus
StartInit
Fieldbus specific
EndInit
Sub-entries
Communicator RS232/422/485
Subnetwork
This menu entry holds additional sub-entries / parameters

- **B: Navigation Section**

  This section is the main tool for selecting and altering different levels of the sub-network configuration.

  Entries preceded by a “+” holds further configuration parameters or “sub menus”. To gain access to these parameters, the entry must be expanded by clicking “+”.

  There are three main levels in the navigation window, namely Fieldbus, Communicator RS232/422/485, and Subnetwork.

  Right-clicking on entries in this section brings out additional selections related to that particular entry.

- **C: Parameter Section**

  This section holds a list of parameters or options related to the currently selected entry in the Navigation Section.

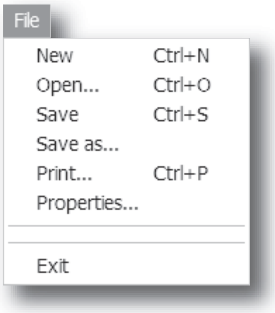
  The parameter value may be specified either using a selection box or manually, depending on the parameter itself. Values can be specified in decimal form (e.g. “42”), or in hexadecimal format (e.g. “0x2A”).

- **D: Information Section**

  This section holds information related to the currently selected parameter.

10.1.1 Drop-down Menus

File



- **New**

  Create a new configuration.

  See also “Configuration Wizards” on page 64.

- **Open...**

  Open a previously created configuration.

- **Save**

  Save the current configuration.

- **Save As...**

  Save the current configuration under a new name.

- **Print...**

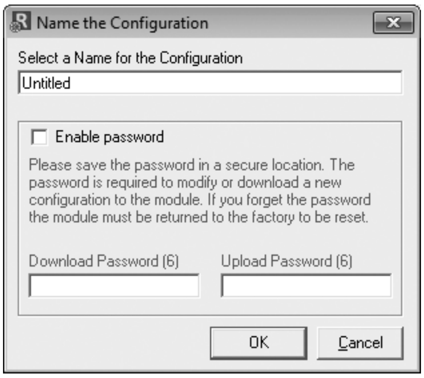
  Send details about the current configuration to a printer.

- **Properties...**

  Set the name and (optional) passwords for the configuration.

Item	Description
Select a Name for the Configuration	Enter a descriptive name for the new configuration
Enable Password	Enables password protection
Download Password(6)	Set passwords for downloading and uploading the configuration (max. 6 characters)
Upload Password(6)	

CAUTION: Always keep a copy of the password in a safe place. A lost password cannot be retrieved!


Name the Configuration

Select a Name for the Configuration

Untitled

☐ Enable password

Please save the password in a secure location. The password is required to modify or download a new configuration to the module. If you forget the password the module must be returned to the factory to be reset.

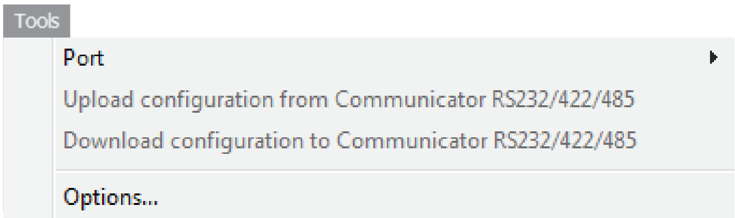
Download Password (6)

Upload Password (6)

- **Exit**

  Close ACM.

Tools


Tools
Port
Upload configuration from Communicator RS232/422/485
Download configuration to Communicator RS232/422/485
Options...

- **Port**

  Select the COM-port used for the configuration of the gateway.

- **Upload configuration from Communicator RS232/422/485**

  Upload the configuration from the gateway to ACM.

- **Download configuration to Communicator RS232/422/485**

  Download the current configuration to the gateway.

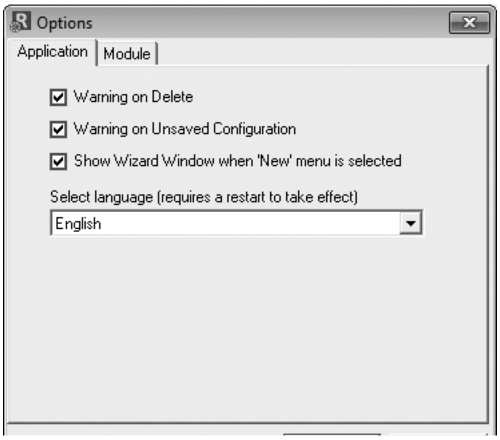
- **Start Logging**

  Start the Data Logger (see “Data Logger” on page 98).

  Note that when the Data Logger is active, this menu entry is changed to “Stop Logging”.

- **Options**

  This will open the following window:



☑ Warning on Delete

☑ Warning on Unsaved Configuration

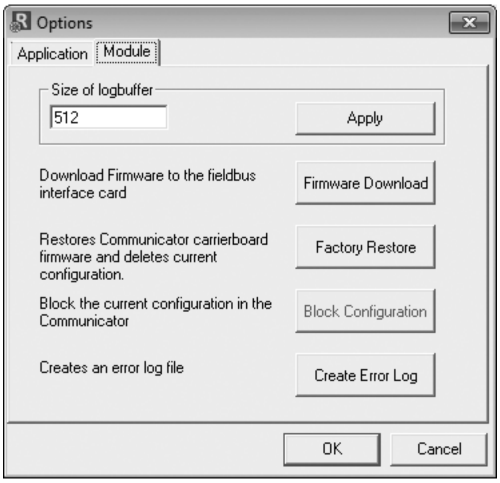
☑ Show Wizard Window when 'New' menu is selected

Select language (requires a restart to take effect)

English

Item	Description
Warning on Delete	A confirmation dialog is displayed each time something is deleted.
Warning on Unsaved Configuration	A confirmation dialog is displayed when closing ACM with unsaved data.
Show Wizard when “New” menu is selected	The Wizard is displayed each time a new configuration is created.
Select language	Selects which language to use. The new setting will be active the next time the program is launched.

Selecting the “Module” tab will reveal additional properties:



Options

Application | Module

Size of logbuffer: 512 [Apply]

Download Firmware to the fieldbus interface card [Firmware Download]

Restores Communicator carrierboard firmware and deletes current configuration. [Factory Restore]

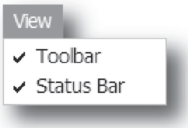
Block the current configuration in the Communicator [Block Configuration]

Creates an error log file [Create Error Log]

[OK] [Cancel]

Item	Description
Size of logbuffer	By default, the Data Logger can log up to 512 entries in each direction. If necessary, it is possible to specify a different number of entries (valid settings range from 1...512). Click “Apply” to validate the new settings. See also “Data Logger” on page 98.
Firmware Download	Download firmware to the embedded fieldbus interface. **Warning: Use with caution.**
Factory Restore	Restores the gateway firmware to the original state (does not affect the embedded fieldbus interface).
Block Configuration	When selected, the downloaded configuration will not be executed by the gateway. **Warning: Use with caution.**
Create Error log	Creates an error log file

View



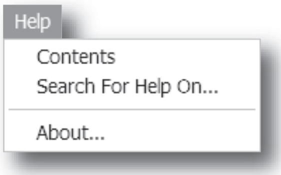
- **Toolbar**

  Enables/disables the toolbar icons at the top of the main window.

- **Status Bar**

  Enables/disables the status bar at the bottom of the main window.

Help



- **Contents/Search For Help On...**

  Opens a built-in browser window with a link to the Anybus support website.

- **About...**

  Displays general information about the gateway and the current version of ACM.

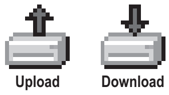
10.1.2 Toolbar Icons

The toolbar features icons for the most commonly used functions.

- **New, Open & Save**
  See “File” on page 60.

New
Open
Save

- **Upload from ABC & Download to ABC**
  See “Tools” on page 61.



- **Up one Level**
  Clicking on this icon will move the selection in the navigation section.

Up one Level

- **Cut, Copy, Paste, Delete, Insert**
  These icons are used for common editing functions in the navigation section.

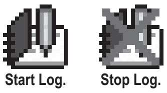
Cut
Copy
Paste
Delete
Insert

- **Connect**
  Clicking on this icon will cause ACM to attempt to connect to the gateway.

- **Disconnect**
  Clicking on this icon will cause ACM to disconnect from the gateway.

Disconnect

- **Start Logging & Stop Logging**
  See “Tools” on page 61 & “Data Logger” on page 98.



- **Sub-network Monitor**
  Clicking on this icon will launch the sub-network Monitor (see “Sub-network Monitor” on page 93).

Sub-Network Monitor

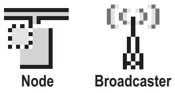
- **Add Command**
  This icon is used to add commands to the currently selected node.

Add Command

- **Add Mailbox**
  (Advanced functionality, see “Mailbox Editor” on page 118)

Add Mailbox

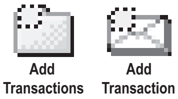
- **Add Node & Add Broadcaster**
  These icons are used to add nodes to the configuration.



- **Node Monitor**
  Clicking on this icon will launch the Node Monitor (see “Node Monitor” on page 94)

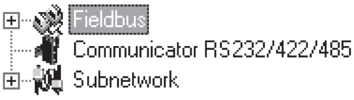
Node Monitor

- **Add Transaction(s)**
  These icons are used to add transactions to the currently selected node.



11. Basic Settings

11.1 Fieldbus Settings



(Select ‘Fieldbus’ in the Navigation Section to gain access to the parameters described in this section).

General

During start-up the fieldbus interface of the Anybus Communicator is initialized to fit the configuration created in the Anybus Configuration Manager. Optionally, some initialization parameters can be set manually to provide better control over how the data shall be treated by the gateway.

Fieldbus Settings

To be able to participate on the network, the following settings must be correctly made:

- **Fieldbus Type**

  Anybus Configuration Manager supports a wide range of networking systems. Make sure this parameter is set to “EtherNet/IP 2Port”.

- **Modbus Address Mode**

  Enabled - Use Modbus Address Mode
  Disabled - Use Anybus Address Mode

  See also “Modbus-TCP” on page 25

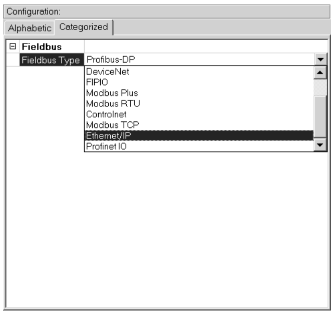
- **Communicator IP-address, Gateway, Subnet Mask**

  See “Basic Network Configuration” on page 37.

- **TCP/IP Settings**

  Enabled - Use settings in Anybus Configuration Manager
  Disabled - Use settings stored in ‘ethcfg.cfg’

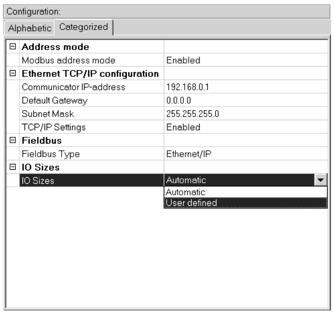
  See also “Basic Network Configuration” on page 37



Fieldbus	
Fieldbus Type	Profibus-DP

DeviceNet
FIPIO
Modbus Plus
Modbus RTU
Controlnet
Modbus TCP
Ethernet/IP
Profinet IO

Fieldbus Type



Address mode	
Modbus address mode	Enabled
Ethernet TCP/IP configuration	
Communicator IP-address	192.168.0.1
Default Gateway	0.0.0.0
Subnet Mask	255.255.255.0
TCP/IP Settings	Enabled
Fieldbus	
Fieldbus Type	Ethernet/IP
IO Sizes	
IO Sizes	Automatic

Automatic
User defined

IO Sizes

I/O Sizes

These parameters specify how data from the internal memory buffer will be exchanged over EtherNet/IP. This can either be handled automatically based on the sub-network configuration, or it can be specified manually.

- **Automatic**

  All data will be represented as I/O Data on EtherNet/IP.

- **User defined**

  Additional parameter properties appear; ‘IO Size In’ and ‘IO Size Out’. The specified amount, starting at address 0x0000 of the respective memory buffers, will be reserved for and represented as I/O Data. The remainder will be reserved for Parameter Data.

  See also “EtherNet/IP” on page 23

11.2 Communicator Parameters

Interface

Only serial communication is currently supported.

Control/Status Word

See "Control and Status Registers" on page 102.

Value	Description
Enabled	Enable the Control and Status Registers. The "Data Valid"-bit in the Control Register must be set to start the sub-network communication.
Enabled but no startup lock	This setting is similar to "Enabled", except that the control system is not required to set the "Data Valid"-bit to start the sub-network communication.
Disabled	This setting completely disables the Control and Status Registers.

Module Reset

This parameter specifies how the gateway will behave in the event of a fatal error.

Value	Description
Enabled	The gateway will be restarted, and no error will be indicated to the user.
Disabled	The gateway will halt and indicate an error.

Protocol Mode

This parameter specifies which protocol mode to use for the sub-network. See "Protocol Modes" on page 17.

Value	Description
Generic Data Mode	This mode is primarily intended for Produce & Consume-based protocols, where there are no Master-Slave relationship between the gateway and the nodes on the sub-network.
Master Mode	This mode is intended for "Query & Response"-based protocols, where a single Master exchanges data with a number of Slaves.
DF1	This mode is intended for the DF1 protocol. The Anybus Communicator can only be configured as a Master with half-duplex communication. **Note:** This is the only mode available if you intend to configure an ABC module for DF1.

Statistics

The Transmit- and Receive Counters indicate how many transactions that have successfully been exchanged on the sub-network. This feature is primarily intended for debugging purposes.

- **Receive Counter Location**
  Specifies the location of the Receive Counter in the internal memory buffer.

- **Transmit Counter Location**
  Specifies the location of the Transmit Counter in the internal memory buffer.

- **Statistics**
  Enables/disables the Receive and Transmit Counters.

11.3 Sub-network Parameters

Fieldbus
Communicator RS232/422/485
Subnetwork

Communication

These parameters specify the actual communication settings used for the sub-network.

Parameter	Description	Master Mode and Generic Mode
Bitrate (bits/s)	Selects the bit rate	1200 2400 4800 9600 19200 35700 38400 57600
Data bits	Selects the number of data bits	7, 8
Parity	Selects the parity mode	None, Odd, Even
Physical standard	Selects the physical interface type	RS232, RS422, RS485
Stop bits	Number of stop bits.	1, 2

Start- and End Character

Note: These parameters are only available in Generic Data Mode.

Start and end characters are used to indicate the beginning and end of a serial message. For example, a message may be initiated with <ESC> and terminated with <LF>. In this case, the Start character would be 0x1B (ASCII code for <ESC>) and the End character 0x0A (ASCII code for <LF>)

Parameter	Description	Valid settings
End character value	End character for the message, ASCII	0x00–0xFF
Use End character	Determines if the End character shall be used or not	Enable / Disable
Start character value	Start character for the message, ASCII	0x00–0xFF
Use Start character	Determines if the Start character shall be used or not	Enable / Disable

Timing (Message Delimiter)

The parameters in this category differs slightly between the different protocol modes.

- **Master Mode**

  The Message Delimiter specifies the time that separates two messages in steps of 10 ms. If set to 0 (zero), the gateway will use the standard Modbus delimiter of 3.5 characters (the actual number of ms will be calculated automatically based on the currently used communication settings).

- **Generic Data Mode**

  The Message Delimiter specifies the time that separates two messages in steps of 10 μs.

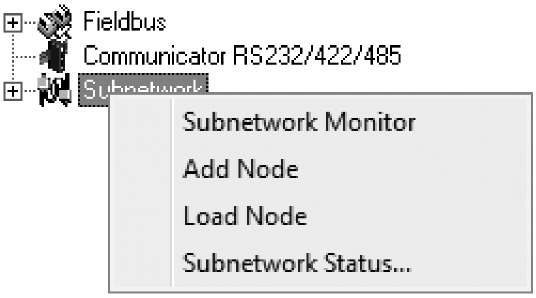
12. Nodes

12.1 General

In ACM, a node represents a single device on the network. Although the gateway does not feature a scan list in the traditional sense, all nodes and their transactions will be processed in the order they were defined in ACM.

The maximum number of nodes that can be created in ACM is 31.

12.2 Adding & Managing Nodes

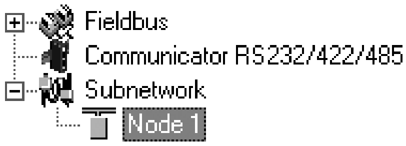


Function	Description
Paste	Paste a node from the clipboard
Subnetwork Monitor	Launch the subnet monitor (see “Sub-network Monitor” on page 93)
Add Node	Add a node to the configuration
Add Broadcaster[a]	Add a broadcaster node to the configuration
Load Node	Add a previously saved node
Subnetwork Status...	View diagnostic information about the sub-network

[a] This function is only available in Master Mode.

12.3 Node Parameters

12.3.1 Master Mode and Generic Data Mode



To gain access to the parameters described in this section, select a node in the Navigation Section.

Parameter	Description
Slave Address	The value entered here may be used to set the node address in certain commands. For more information, see “The Command Editor” on page 83.

13. Transactions

13.1 General

As mentioned previously, transactions are representations of the actual serial telegrams exchanged on the serial sub-network. Although the gateway does not feature a scan list in the traditional sense, all nodes and their transactions will be processed in the order they were defined in ACM.

Transactions are handled slightly differently in the three protocol modes:

- **Master Mode**

  For regular nodes, transactions always come in pairs; a query and a response. The query is issued by the gateway, while responses are issued by the slaves on the sub-network. The Broadcaster can only send transactions.

- **Generic Data Mode**

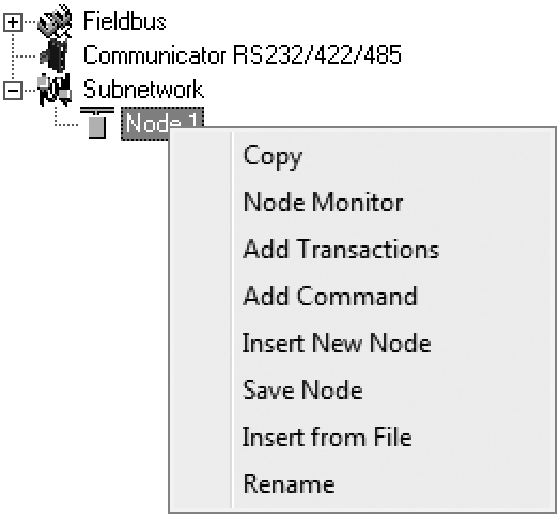
  Transactions can be added as desired for both directions. Transactions sent to the sub-network are called “Transaction Produce”, and transactions issued by other nodes are called “Transaction Consume”.

- **DF1 Master Mode**

  Please refer to “DF1 Protocol Mode” on page 86.

Theoretically, the gateway supports up to 150 transactions. The actual number may however be less depending on the memory requirements of the defined transactions.

13.2 Adding & Managing Transactions

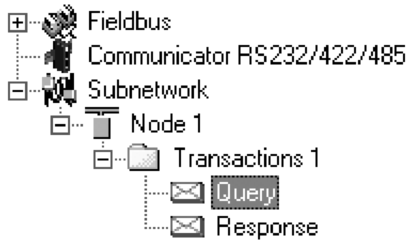


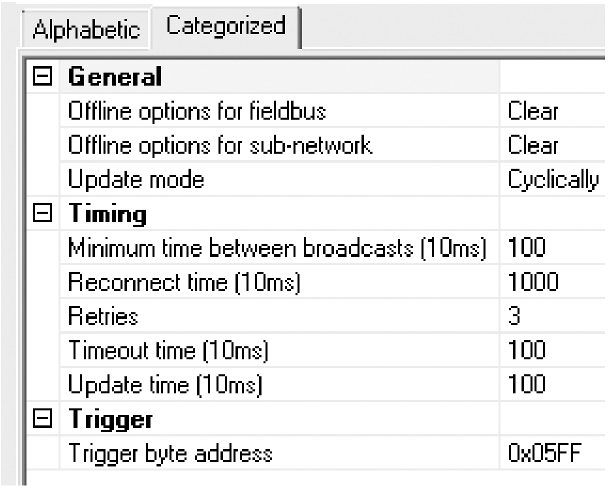
Function	Description
Copy	Copy a node to the clipboard
Delete[a]	Delete a node
Node Monitor	Launch the node monitor (see “Node Monitor” on page 94)
Add Transaction(s)[b]	On regular nodes, this adds a Query and a Response. The two transactions will be grouped in order to increase readability. On the Broadcaster, a single transaction will be added.
Add Transaction Consume[c]	Add a “Consume”-transaction
Add transaction Produce[c]	Add a “Produce”-transaction
Add Command	Add predefined transactions to the node
Insert New Node	Insert a new node above the currently selected one
Save Node	Save the selected node
Insert from File	Insert a previously saved node above the currently selected node
Rename	To increase readability, each node can be given a unique name using this function

a. Only available if more than one node exists
b. Only available in Master Mode
c. Only available in Generic Data Mode

13.3 Transaction Parameters (Master Mode)

13.3.1 Parameters (Query & Broadcast)





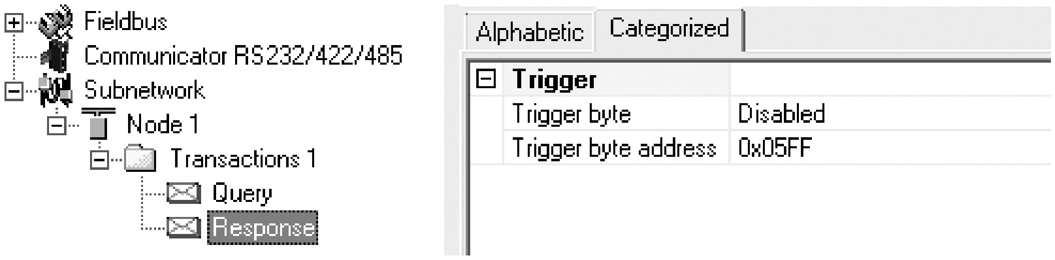
Alphabetic | Categorized

General	
Offline options for fieldbus	Clear
Offline options for sub-network	Clear
Update mode	Cyclically
Timing	
Minimum time between broadcasts (10ms)	100
Reconnect time (10ms)	1000
Retries	3
Timeout time (10ms)	100
Update time (10ms)	100
Trigger	
Trigger byte address	0x05FF

Parameter	Description
Minimum time between broadcasts (10 ms)	This parameter specifies how long the gateway shall wait after transmitting a broadcast transaction before processing the next entry in the scanlist. The value should be set high enough to allow the slave devices time to finish the handling of the broadcast. The entered value is multiplied by 10. An entered value of 5 will result in 50 ms. **Note:** This setting is only relevant for the Broadcaster node.
Offline options for fieldbus	This parameter specifies the action to take for this transaction if the higher level network goes offline. This affects the data that is sent to the sub-network. • **Clear** - The data destined for the slave-devices is cleared (set to zero) • **Freeze** - The data destined for the slave-device is frozen • **NoScanning** -The updating of the sub-network is stopped
Offline options for sub-network	This parameter specifies the action to take for this transaction if the sub-network goes offline. This affects the data that is reported to the control system. • **Clear** - Data is cleared (0) on the higher level network if the sub-network goes offline • **Freeze** - Data is frozen on the higher level network if the sub-network goes offline
Reconnect time (10 ms)	This parameter specifies how long the gateway shall wait before attempting to reconnect a disconnected node. A node will be disconnected in case the maximum number of retries (below) has been reached. The entered value is multiplied by 10. An entered value of 5 will result in 50 ms. **Note:** This setting is not relevant for the Broadcaster node.
Retries	This parameter specifies how many times a timeout may occur in sequence before the node is disconnected.
Timeout time (10 ms)	This parameter specifies how long the gateway will wait for a response from a node. If this time is exceeded, the gateway will retransmit the Query until the maximum number of retries (see above) has been reached. The entered value is multiplied by 10. An entered value of 5 will result in 50 ms.
Trigger byte address	This parameter specifies the location of the trigger byte in internal memory (only relevant when “Update mode” is set to “Change of state on trigger”). Valid settings range from 0x200 to 0x3FF and 0x400 to 0xFFF

Parameter	Description
Update mode	This parameter is used to specify when the transaction shall be sent to the slave: • **Cyclically** The transaction is issued cyclically at the interval specified in the "Update time" parameter. • **On data change** The data area is polled for changes at the time interval defined by Update time. A transaction is issued when a change in data is detected. • **Single shot** The Query is issued once at start up. • **Change of state on trigger** The Query is issued when the trigger byte value has changed. This feature enables the control system to notify the gateway when to issue a particular Query. To use this feature correctly, the control system must first update the data area associated with the Query/transaction, then increase the trigger byte by one. The location of the trigger byte is specified by the "Trigger byte address" parameter. The trigger byte is checked at the interval specified in the "Update time" parameter.
Update time (10 ms)	This parameter specifies how often the transaction will be issued in steps of 10 ms (relevant only when "Update mode" is set to "Cyclically", "On data change" or "Change of state on trigger"). The entered value is multiplied by 10. An entered value of 5 will result in 50 ms.

13.3.2 Parameters (Response)

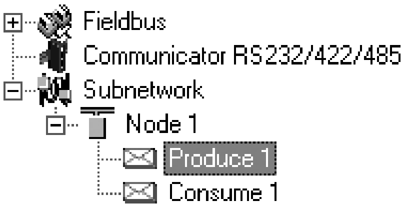


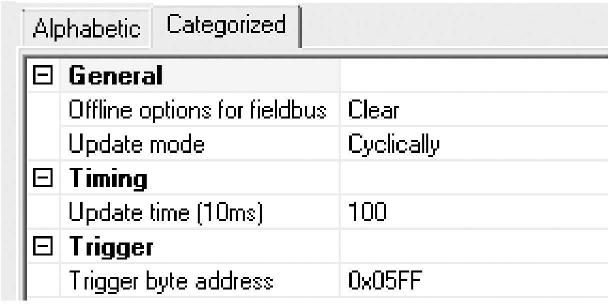
Trigger	
Trigger byte	Disabled
Trigger byte address	0x05FF

Parameter	Description
Trigger byte	This parameter is used to enable/disable the trigger functionality for the response. If enabled, the gateway will increase the trigger byte by one when the gateway receives new data from the sub-network. This can be used to notify the control system of the updated data. The location of the trigger byte is specified by the "Trigger byte address" parameter below.
Trigger byte address	This parameter specifies the location of the trigger byte in the internal memory buffer. Valid settings range from 0x000 to 0x1FF and 0x400 to 0xFFF

13.4 Transaction Parameters (Generic Data Mode)

13.4.1 Produce Transactions



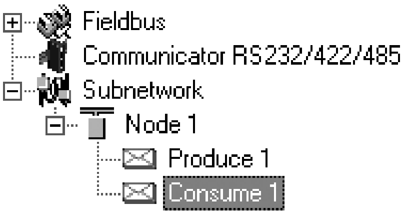


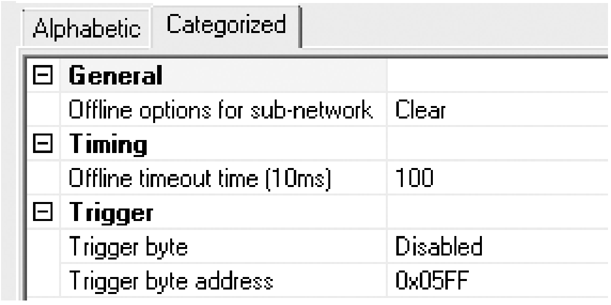
General	
Offline options for fieldbus	Clear
Update mode	Cyclically
Timing	
Update time (10ms)	100
Trigger	
Trigger byte address	0x05FF

Parameter	Description
Offline options for fieldbus	This parameter specifies the action to take for this transaction if the higher level network goes offline. This affects the data that is sent to the sub-network. • **Clear** Data is cleared (0) on the sub-network if the higher level network goes offline • **Freeze** Data is frozen on the sub-network if the higher level network goes offline • **NoScanning** Stop subnet scanning for this transaction if the higher level network goes offline
Update mode	The update mode for the transaction: • **Cyclically** The transaction is sent cyclically at the interval specified in “Update Time”. • **On data change** The data area is polled for changes at the time interval defined by Update time. A transaction is issued when a change in data is detected. • **Single shot** The transaction is sent once at startup. • **Change of state on trigger** The transaction is sent when the trigger byte has changed. This feature enables the control system to notify the gateway when to issue a particular transaction. To use this feature correctly, the control system must first update the data area associated with the transaction, then increase the trigger byte by one. The location of the trigger byte is specified by the “Trigger byte address” parameter. The trigger byte is checked at the interval specified in the “Update time” parameter.
Update time (10 ms)	This parameter specifies how often the transaction will be issued in steps of 10ms (relevant only when “Update mode” is set to “Cyclically”, “On data change” or “Change of state on trigger”). The entered value is multiplied by 10. An entered value of 5 will result in 50 ms.

Parameter	Description
Trigger byte address	This parameter specifies location of the trigger byte in the internal memory buffer. If "Update mode" is set to "Change of state on trigger", the memory location specified by this parameter is monitored by the gateway. Whenever the trigger byte is updated, the gateway will produce the transaction on the sub-network. This way, the control system can instruct the gateway to produce a specific transaction on the sub-network by updating the corresponding trigger byte. The trigger byte should be incremented by one for each activation. Please note that the trigger byte address must be unique to each transaction. It can not be shared by two or more transactions. **Note:** This parameter has no effect unless the "Update mode" parameter is set to "Change of state on trigger". Valid settings range from 0x200 to 0x3FF and 0x400 to 0xFFF

13.4.2 Consume Transactions



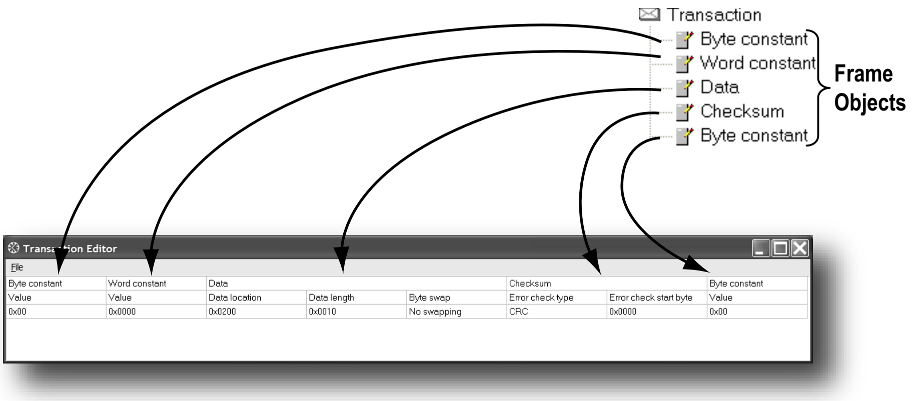


⊟ **General**	
Offline options for sub-network	Clear
⊟ **Timing**	
Offline timeout time (10ms)	100
⊟ **Trigger**	
Trigger byte	Disabled
Trigger byte address	0x05FF

Parameter	Description
Offline options for sub-network	This parameter specifies the action to take for this transaction if the sub-network goes offline. This affects the data that is sent to the higher level network. • **Clear** Data is cleared (0) on the higher level network if the sub-network goes offline • **Freeze** Data is frozen on the higher level network if the sub-network goes offline
Offline timeout time (10 ms)	This parameter specifies the maximum allowed time between two incoming messages in steps of 10ms. If this time is exceeded, the sub-network is considered to be offline. A value of 0 disables this feature, i.e. the sub-network can never go offline. The entered value is multiplied by 10. An entered value of 5 will result in 50 ms.
Trigger byte	• **Enable** Enables the trigger byte. The location of the trigger byte must be specified in "Trigger byte address". The trigger byte value will be increased each time a valid transaction has been consumed by the gateway. The trigger byte will also be increased if the offline option is set to "Clear" and the offline timeout time value is reached. This feature enables the control system to be notified each time new data has been consumed on the sub-network. • **Disable** Disables the trigger byte functionality.
Trigger byte address	This parameter specifies the location of the trigger byte in the internal memory buffer. Valid settings range from 0x000 to 0x1FF and 0x400 to 0xFFF. Please note that the trigger byte address must be unique to each transaction. It can not be shared by two or more transactions.

13.5 Transaction Editor

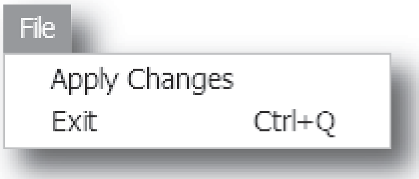
The Transaction Editor can be used to edit the individual frame objects of a transaction. The same settings are also available in the parameter section of the main window, however the Transaction Editor presents the frame objects in a more visual manner.



Byte constant	Word constant	Data			Checksum		Byte constant
Value	Value	Data location	Data length	Byte swap	Error check type	Error check start byte	Value
0x00	0x0000	0x0200	0x0010	No swapping	CRC	0x0000	0x00

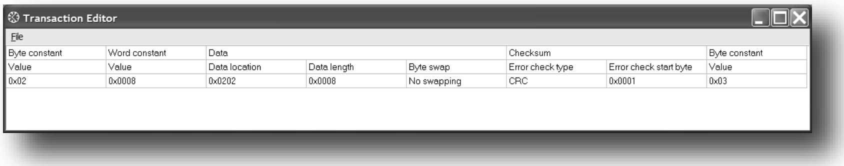
To edit the value of a parameter, click on it and enter a new value using the keyboard. When editing transactions which are based on predefined commands, certain parts of the transaction may not be editable.

The File menu features the following entries:



- **Apply Changes**
  This will save any changes and exit to the main window.
- **Exit**
  Exit without saving.

Example:



Byte constant	Word constant	Data			Checksum		Byte constant
Value	Value	Data location	Data length	Byte swap	Error check type	Error check start byte	Value
0x02	0x0008	0x0202	0x0008	No swapping	CRC	0x0001	0x03

The transaction created in this example are built up as follows:

The first byte holds the STX (0x02) followed by two bytes specifying the length of the data field (in this case 8). The next 8 bytes are data and since this is a “query”-transaction, the data is to be fetched from the Output Area which starts at address location 0x202. No swapping will be performed on the data. This is followed by a two-byte checksum. The checksum calculation starts with the second byte in the transaction.

The transaction ends with a byte constant, the ETX (0x03).

14. Frame Objects

14.1 General

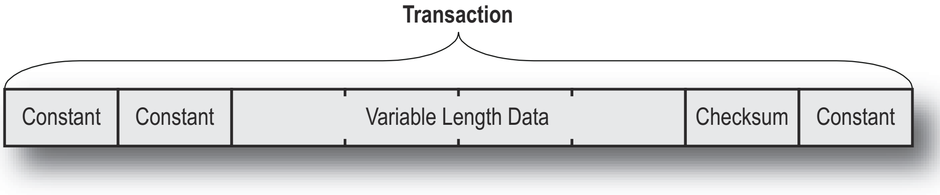
Each transaction consists of Frame Objects which makes up the serial telegram frame. Each Frame Object specifies how the gateway shall interpret or generate a particular part of the telegram.

There are 5 types of frame objects, which are described in detail later in this chapter:

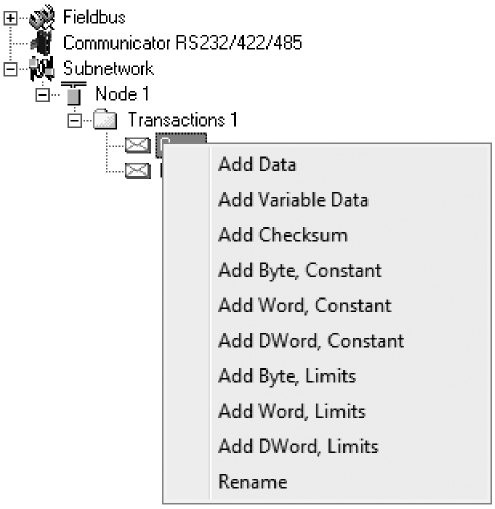
- Constant Objects
- Limit Objects
- Data Objects
- Variable Data Objects
- Checksum Objects

Example:

The following Transaction consists of several frame objects; three constants, a data object, and a checksum object.



14.2 Adding and Editing Frame Objects



To add a frame object to a Transaction, right-click on the Transaction in the Navigation Section and select one of the entries in the menu that appears.

The entry called "Transaction Editor" will launch the Transaction Editor, which is used to edit transactions and frame objects in a more visual manner. For more information, see "Transaction Editor" on page 74.

To edit parameters associated with a particular frame object, select the frame object in the Navigation Section. The settings for that frame object will be displayed in the Parameter Section.

It is also possible to edit the frame objects in a transaction in a more visual manner using the Transaction Editor, see "Transaction Editor" on page 74.

14.3 Constant Objects (Byte, Word, Dword)

Constant Objects have a fixed value and come in three sizes:

- **Byte**
  8 bits
- **Word**
  16 bits
- **Dword**
  32 bits

Constants are handled differently depending on the direction of the transaction:

- **Produce/Query Transactions**
  The gateway will send the value as it is without processing it.
- **Consume/Response Transactions**
  The gateway will check if the received byte/word/dword matches the specified value. If not, the message will be discarded.

To set the value of the object, select it in the Navigation Section and enter the desired value in the Parameter section.

Parameter	Description
Value	Constant value

14.4 Limit Objects (Byte, Word, Dword)

Limit Objects have a fixed range and come in three sizes:

- **Byte**
  8 bits
- **Word**
  16 bits
- **Dword**
  32 bits

Limit Objects are handled differently depending on the direction of the transaction:

- **Produce/Query Transactions**
  This object shall not be used for such transactions (value will be undefined).
- **Consume/Response Transactions**
  The gateway will check if the received byte/word/dword fits inside the specified boundaries. If not, the message will be discarded.

There are 3 types of interval objects:

- **Byte**
  8 bit interval
- **Word**
  16 bit interval
- **Dword**
  32 bit interval

To set the range of the object, select it in the Navigation Section and enter the desired range in the Parameter section as follows:

Parameter	Description
Maximum Value	This is the largest allowed value for the range. Range:0x00 to 0xFFh(byte) 0x0000 to 0xFFFFh(word) 0x00000000 to 0xFFFFFFFFh(dword) **Note:** The value must be larger than the Minimum Value.
Minimum Value	This is the smallest allowed value for the range. Range:0x00 to 0xFEh(byte) 0x0000 to 0xFFFEh(word) 0x00000000 to 0xFFFFFFFEh(dword) **Note:** The value must be less than the Maximum Value.

14.5 Data Object

Data Objects are used to represent raw data as follows:

- **Produce/Query Transactions**

  The specified data block is forwarded from the higher level network to the sub-network.

- **Consume/Response Transactions**

  The specified data block is forwarded from the sub-network to the higher level network.

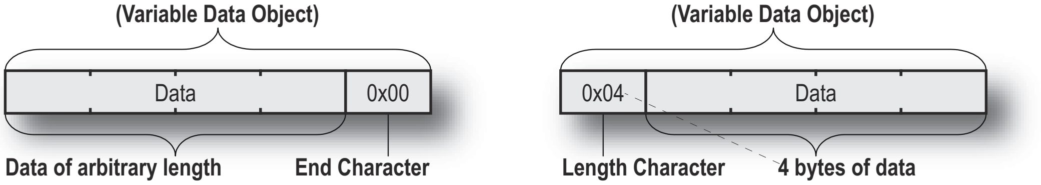
To specify the properties of the object, select it in the Navigation Section and enter the desired settings in the Parameter section as follows:

Parameter	Description
Byte Swapping	• **No Swapping** No swapping is performed on the data • **Swap 2 bytes** A, B, C, D becomes B, A, D, C • **Swap 4 bytes** A, B, C, D becomes D, C, B, A
Data Length	The length of the data block, in bytes. In case of a Response or Consume transaction, incoming messages where the data size differs from the value specified here will be discarded. Maximum data length allowed for one frame is 300 bytes.
Data Location	The location of the data block in the internal memory buffer.

14.6 Variable Data Object

Note: Only one Variable Data Object is permitted for each transaction.

This object is similar to the Data Object, except that it has no predefined length. Instead, an End or Length-character specifies the size of the data block as follows:



- **Produce/Query Transactions**

  The specified data block will be forwarded from the higher level network to the sub-network. The control system must supply an End or Length character in order for the gateway to know the size of the data block.

  The End- or Length-character itself may either be forwarded to the sub-network or discarded.

- **Consume/Response Transactions**

  The specified data block is forwarded from the sub-network to the higher level network. The End- or Length-character will be generated by the gateway automatically (if applicable).

  The End- or Length-character itself may either be forwarded to the higher level network or discarded.

To specify the properties of the object, select it in the Navigation Section enter the desired settings in the Parameter section as follows:

Parameter	Description
Byte Swapping	• **No Swapping** No swapping will be performed on the data • **Swap 2 bytes** A, B, C, D becomes B, A, D, C • **Swap 4 bytes** A, B, C, D becomes D, C, B, A
Fill unused bytes	• **Enabled**[a] Fill unused data with the value specified in “Filler byte”. • **Disabled** Don’t fill
Filler byte	Filler byte value. Only used if “Fill unused bytes” has been enabled.
Data Location	The offset in the internal memory buffer where the data shall be read from / written to
Object Delimiter (Produce/Query)	• **Length Character** Length character visible in internal memory buffer but *not* sent out on the sub-network • **Length Character Visible** Length character visible in internal memory buffer *and* sent out on the sub-network • **End Character** End character visible in internal memory buffer but *not* sent out on the sub-network • **End Character Visible** End character visible in the internal memory buffer *and* sent out on the sub-network • **No Character** No end- or length-character generated in the internal memory buffer
Object Delimiter (Consume/Response)	• **Length Character** Length character visible in internal memory buffer but *not* received from the sub-network • **Length Character Visible** Length character visible in internal memory buffer *and* received from the sub-network • **End Character** End character visible in internal memory buffer but *not* received from the sub-network • **End Character Visible** End character visible in the internal memory buffer *and* received from the sub-network • **No Character** No end or length characters included in the received string or generated in the internal memory buffer
End Character Value	End Character value[b]
Maximum Data Length	The maximum allowed length (in bytes) of the variable data object. If the actual length of the data exceeds this value, the message will be discarded. The value must not exceed 256 bytes, which is the maximum data length allowed for one frame.

[a] Only relevant for Consume/Response transactions

[b] Only used if “Object Delimiter” is set to “End Character” or “End Character Visible”

14.7 Checksum Object

Most serial protocols features some way of verifying that the data has not been corrupted during transfer. The Checksum Object calculates and includes a checksum in a transaction.

Parameter	Description
Error Check Start byte	Specifies the byte offset in the transaction to start checksum calculations on.[a]
Error Check Type	This parameter specifies which type of algorithm to use: • **CRC (2 bytes)** CRC-16 with 0xA001 polynome (Modbus RTU standard) • **LRC (1 byte)** All bytes are added together as unsigned 8-bit values. The two's complement of the result will be used as a checksum. (Modbus ASCII standard with Error Check Start Byte = 0x01 and Representation = ASCII) • **XOR (1 byte)** All bytes are logically XOR:ed together. The resulting byte will be used as a checksum. • **ADD (1 byte)** All bytes are added together as unsigned 16-bit values. The lowest 8 bits in the result will be used as a checksum.
Error check type combined with	The binary value can be converted to its one's or two's complement. This conversion is carried out before ASCII formatting (see next parameter). • **None** The checksum binary value is transmitted without conversion. • **One's complement** The checksum value will be converted to its one's complement (inverse code). Example: 00001100 will be transmitted as 11110011 • **Two's complement** The checksum value will be converted to its two's complement (complement code). Example: 00001100 will be transmitted as 11110100
Representation	• **Binary** The checksum is transmitted in binary format. • **ASCII** All characters in the checksum are converted to ASCII values.

a. In Generic Data Mode the Start character (if used) will not be included in the checksum calculation.

15. Commands

This information is only valid for the Master and Generic Data modes. For DF1 master mode, please refer to "Services" on page 89.

15.1 General

As mentioned previously, commands are actually predefined transactions that can be stored and reused. Just like regular transactions, commands consist of frame objects and are representations of the actual serial telegrams exchanged on the serial sub-network.

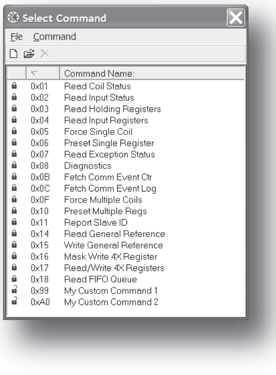
Adding a command to a node actually results in (a) transaction(s) being added according to the directions specified in the command. The frame objects in such a transaction may retrieve their values not only from parameters in the parameter section, but also from other sources such as the "SlaveAddress"-parameter (see "Node Parameters" on page 67). In such case, the parameters in the parameter section will be greyed out and cannot be edited directly.

In Master Mode, ACM comes preloaded with commands for most common Modbus RTU functions. Additional commands can easily be added using the Command Editor (see "The Command Editor" on page 83). For DF1 Master Mode, see "Services" on page 89. In Generic Data Mode, no predefined commands exist, but custom ones may be implemented as desired.

15.2 Adding & Managing Commands

To add a command to a node, right-click on the node in the Navigation Section and select "Add Command".

A list of commands will appear:



		Command Name:
🔒	0x01	Read Coil Status
🔒	0x02	Read Input Status
🔒	0x03	Read Holding Registers
🔒	0x04	Read Input Registers
🔒	0x05	Force Single Coil
🔒	0x06	Preset Single Register
🔒	0x07	Read Exception Status
🔒	0x08	Diagnostics
🔒	0x0B	Fetch Comm Event Ctr
🔒	0x0C	Fetch Comm Event Log
🔒	0x0F	Force Multiple Coils
🔒	0x10	Preset Multiple Regs
🔒	0x11	Report Slave ID
🔒	0x14	Read General Reference
🔒	0x15	Write General Reference
🔒	0x16	Mask Write 4X Register
🔒	0x17	Read/Write 4X Registers
🔒	0x18	Read FIFO Queue
🔓	0x99	My Custom Command 1
🔓	0xA0	My Custom Command 2

Select the desired command in the list, and select "Add Command" in the "Command"-menu. The specified command will be added to the node.

Just like other transactions, the frame objects of added command may be edited in the Navigation/Parameter Section or using the Transaction Editor. Note however that certain frame objects may be locked for editing.

15.2.1 Drop-down Menu

File

This menu features the following entries:

- **Select**
  Add the currently selected Command to the node.
- **Exit**
  Exit without adding a command to the node.

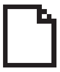
Command

This menu is used to manage the commands in the list:

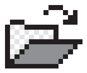
- **Add Command**
  Add a custom command to the list, and open the new command in the Command Editor.
  See also "The Command Editor" on page 83.
- **Edit Command**
  Edit the currently selected command using the Command Editor.
  See also "The Command Editor" on page 83.
- **Delete Command**
  Delete the currently selected command from the list. Note that some commands are fixed and cannot be deleted.

15.2.2 Toolbar Icons

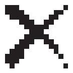
The toolbar features icons for the Add, Edit and Delete Command functions.



Add Command



Edit Command



Delete Command

15.3 The Command Editor

15.3.1 General

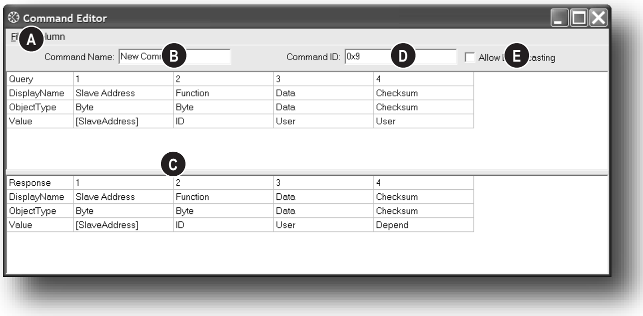
The Command Editor is used to define new commands and edit existing ones. This makes it possible to build a library of commands, which can be stored and reused at a later stage.

Note that the Command Editor is somewhat protocol-dependent in the sense that certain frame objects may not be deleted or altered.

The examples in this section use Master Mode. The procedures involved are similar in Generic Data Mode, but without the limitations imposed by the Modbus RTU protocol.

15.3.2 Basic Navigation

Open the Command Editor by selecting “Edit Command” or “Add Command” from the “Command”-menu.



Command Name: New Com

Command ID: 0x9

□ Allow casting

Query	1	2	3	4
DisplayName	Slave Address	Function	Data	Checksum
ObjectType	Byte	Byte	Data	Checksum
Value	[SlaveAddress]	ID	User	User

Response	1	2	3	4
DisplayName	Slave Address	Function	Data	Checksum
ObjectType	Byte	Byte	Data	Checksum
Value	[SlaveAddress]	ID	User	Depend

A: Drop-down Menu

See “Drop-down Menu” on page 84.

B: Name of Command

Actual name of the command, in text form.

C: Command Transactions

This section holds the actual transactions associated with the command. This can either be a query-response pair, or a single transaction, depending on the protocol mode etc.

D: Command ID

This can be used as desired when building the command, e.g. to specify the function code.

E: Other Settings

Setting	Description
Allow Broadcasting	Specifies if it is allowed to broadcast the command (only relevant in Master Mode)
Produce	The command is producing data (Generic Data Mode only)
Consume	The command is consuming data (Generic Data Mode only)

15.3.3 Drop-down Menu

File

This menu features the following entries:

- **Apply Changes**
  Save changes and exit to the main window.
- **Exit**
  Exit without saving.

Column

The functions in this menu alters the structure of the command.

- **Append Column**
  Add another column to the command.
- **Insert Column**
  Insert a column at the selected position.
- **Delete Column**
  Delete the column at the selected position.

15.3.4 Editing a Command

As mentioned previously, the transaction section in the Command Editor represents the actual transactions associated with the command. Each column represents a frame object within the transaction.

Each column features four rows with the following parameters:

- **Query/Response/Produce/Consume**
  The upper right cell indicates the direction of the transaction.
- **DisplayName**
  Each column can be named so that the different parts of the command appears in a more user friendly manner when editing its settings in the Transaction Editor or in the Parameter Section of the Main Window.
- **ObjectType**
  This row specifies the type of frame object that shall be used for the column.
- **Value**
  This row specifies where the frame object shall retrieve its value/settings.

Value	Description
Depend	This setting is only relevant for Responses in Master Mode. The value will be retrieved from the corresponding part of the “Query”-transaction.
Id	Value will be retrieved from the “Command ID”-setting (see “Basic Navigation” on page 83).
User	Settings associated with the object can be edited by the user.
[SlaveAddress]	Value will be retrieved from the “SlaveAddress”-parameter (see “Node Parameters” on page 67).
(other settings)	Other settings are no longer supported.

15.3.5 Example: Specifying a Modbus-RTU Command in Master Mode

In the following example, a Modbus-RTU command is created in Master Mode. In Modbus-RTU, a transaction always feature the following parts:

- Slave Address (1 byte)
- Function Code (1 bytes)
- A data field
- CRC (CRC-16)

Furthermore, each command always consists of a query and a response.

- **Example Query**

Query	**1**	**2**	**3**	**4**
DisplayName	Slave Address	Function	Data	Checksum
Object Type	Byte Object	Byte Object	Data Object	Checksum Object
Value	[SlaveAddress]	ID	User	User
	The value of this byte constant will be set using the "SlaveAddress" parameter (see "Node Parameters" on page 67).	*The value of this byte constant will be set using the "Command ID"-field.*	*The size and location of the data associated with this object is determined by the user.*	*The checksum type etc can be selected by the user. By default, this is set to match the Modbus-RTU standard.*

- **Example Response**

Response	**1**	**2**	**3**	**4**
DisplayName	Slave Address	Function	Data	Checksum
Object Type	Byte Object	Byte Object	Data Object	Checksum Object
Value	[SlaveAddress]	ID	User	Depend
	This value is linked to the "SlaveAddress" parameter in the parameter window.	*The value of this byte constant will be set using the "Command ID"-field.*	*The size and location of the data associated with this object is determined by the user.*	*This object will retrieve its settings from the corresponding object in the Query.*

By default, the Modbus-RTU-specific frame objects are already in place, and a data object is inserted between the function code and the CRC. These objects cannot be moved or deleted, however it is possible to add additional objects between the function code and the CRC as desired.

Name the new command by entering its name in the "Command Name" field, and enter a suitable function code in the "Command ID"-field. If the command is allowed to be broadcasted, check the "Allow Broadcasting" checkbox.

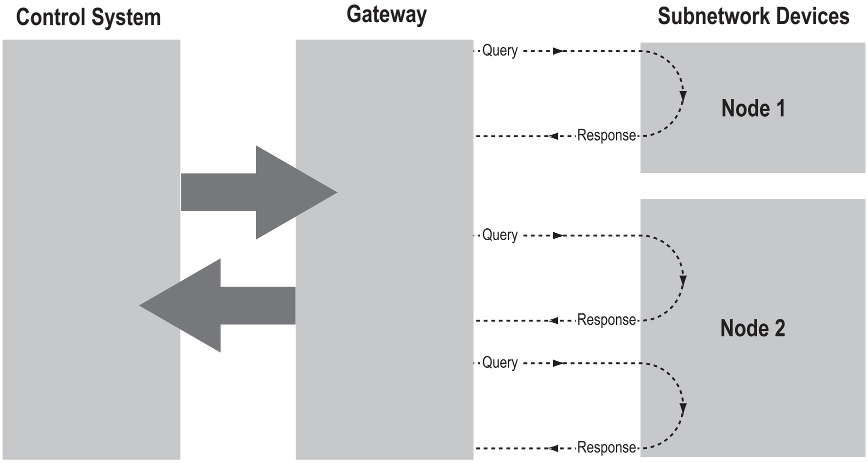
16. DF1 Protocol Mode

This mode makes the Anybus Communicator act as a DF1 protocol master on the sub-network.

16.1 General

In DF1 master mode, communication is based on “services”. A “service” represents a set of commands and operations on the sub-network, that is predefined in the Anybus Communicator. Each service is associated with a set of parameters controlling how and when to use it on the sub-network.

The communication is based on a query-response scheme, where the gateway issues a query on the sub-network. The addressed node on the sub-network is expected to issue a response to that query. Nodes are not permitted to issue responses spontaneously, i. e. without first receiving a query.



In DF1 Master Mode, ACM comes preloaded with a number of services, that can be selected by the user. The actual DF1 commands, that perform the services during runtime, are predefined in the Anybus Communicator. The configuration of the services is performed by right-clicking on a node in the ACM and selecting “Add Command”.

16.2 Communicator Parameters

Interface

Currently, only serial communication is supported.

Control/Status Word

(See “Control and Status Registers” on page 102).

Value	Description
Enabled	Enable the Control and Status Registers. The “Data Valid”-bit in the Control Register must be set to start the sub-network communication.
Enabled but no startup lock	This setting is similar to “Enabled”, except that the control system is not required to set the “Data Valid”-bit to start the sub-network communication.
Disabled	This setting completely disables the Control and Status Registers.

Module Reset

This parameter specifies how the gateway will behave in the event of a fatal error.

Value	Description
Enabled	The gateway will be restarted, and no error will be indicated to the user.
Disabled	The gateway will halt and indicate an error.

Protocol Mode

This parameter specifies which protocol mode to use for the sub-network.

Value	Description
DF1	This mode is intended for the DF1 protocol. The Anybus Communicator can only be configured as a Master with half-duplex communication. **Note:** This is the only mode available if you intend to configure an ABC module for DF1.

See also “Protocol Modes” on page 17.

Statistics

The Transmit- and Receive Counters indicate how many transactions that have successfully been exchanged on the sub-network. This feature is primarily intended for debugging purposes.

- **Receive Counter Location**

  Specifies the location of the Receive Counter in the internal memory buffer.

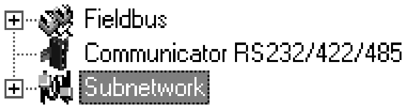
- **Transmit Counter Location**

  Specifies the location of the Transmit Counter in the internal memory buffer.

- **Statistics**

  Enables/disables the Receive and Transmit Counters.

16.3 Sub-network Parameters



Communication

These parameters specify the actual communication settings used for the sub-network.

Parameter	Description	Valid Settings
Bitrate (bits/s)	Selects the bit rate	2400 4800 9600 19200 38400 (Default)
Data bits	Selects the number of data bits	8
Parity	Selects the parity mode	None, Odd, Even
Physical standard	Selects the physical interface type	RS232, RS422, RS485
Stop bits	Number of stop bits.	1

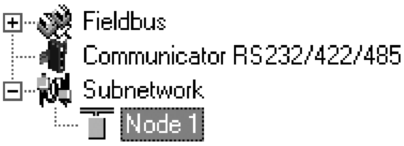
DF1 Settings

Parameter	Description	Default
Master Node Address	Node address of the master, valid values: 0–254	1
Poll time, active slaves (10 ms)	Determines how often the slave shall be polled in steps of 10 ms	100 ms[a]
Poll time, inactive slaves (10 ms)	Determines how often the slave shall be polled in steps of 10 ms	1000 ms[b]

[a] The default value is given as 10 in the parameter window. Each change of 10 ms either increases or decreases this value by 1, i.e. 9 represents a poll time of 90 ms and 11 represents a poll time of 110 ms.

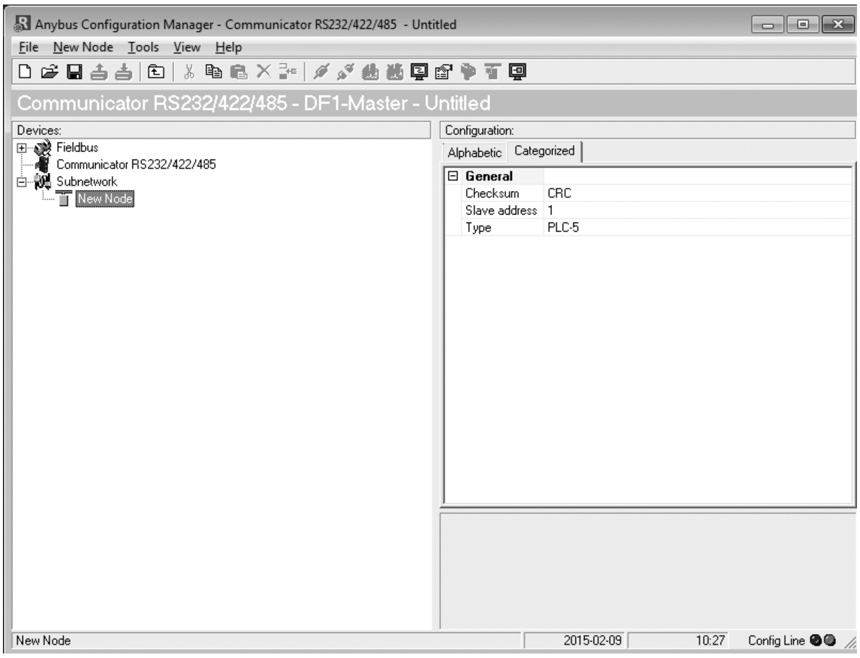
[b] The default value is given as 100 in the parameter window. Each change of 10 ms either increases or decreases this value by 1, i.e. 99 represents a poll time of 990 ms and 101 represents a poll time of 1010 ms.

16.4 Node Parameters



Fieldbus
Communicator RS232/422/485
Subnetwork
Node 1

To gain access to the parameters described in this section, select a node in the navigation section. For more information about nodes, see “Nodes” on page 67.



Anybus Configuration Manager - Communicator RS232/422/485 - Untitled

Communicator RS232/422/485 - DF1-Master - Untitled

Devices:
Fieldbus
Communicator RS232/422/485
Subnetwork
New Node

Configuration:

General
Checksum CRC
Slave address 1
Type PLC-5

New Node 2015-02-09 10:27 Config Line

Parameter	Description	Valid Settings
Checksum	Selects the type of checksum on the network.	BCC CRC (default)
Slave Address	The value entered here sets the node address.	0-254
Type	The PLC type of the slave	PLC-5 SLC500 MicroLogix

16.5 Services

Services are commands that can be stored and reused. The user configures each slave with services that can be issued from the master. A total of 50 services are allowed.

The Anybus Communicator supports a selection of DF1 commands. When the gateway is going to execute a service, it automatically chooses the appropriate DF1 command(s) that are used to perform the service on the selected DF1 node type.

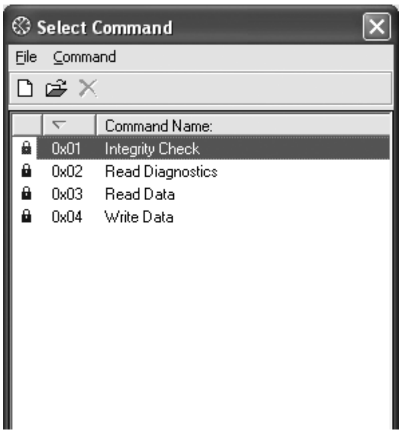
16.5.1 Available Services

Right click on the node, and choose Add Command. A pop-up window will show the four different services that are available:

- Integrity check
- Read diagnostics
- Read data
- Write data

A maximum of 50 services in total (for all nodes) can be selected.

The predefined services can be configured to suit the application. Select a service to show the parameters.



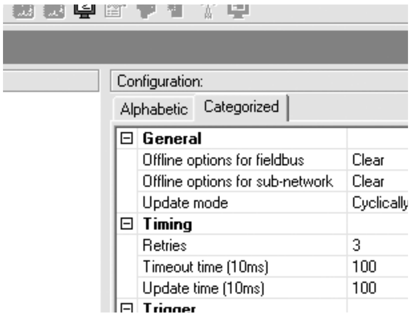
Select Command

File Command

	Command Name:
0x01	Integrity Check
0x02	Read Diagnostics
0x03	Read Data
0x04	Write Data

General Configuration Parameters

These parameters are common to all services, but the settings are individual to each instance of a service.



Configuration:

Alphabetic Categorized

General	
Offline options for fieldbus	Clear
Offline options for sub-network	Clear
Update mode	Cyclically
Timing	
Retries	3
Timeout time (10ms)	100
Update time (10ms)	100
Trigger	

General:

Parameter	Description	Valid settings
Offline options for fieldbus	The action to take for this service if the fieldbus goes offline. This option affects the data that is sent out to the sub-network.	Clear Freeze Noscanning
Offline options for sub-network	The action to take for this service if the sub-network goes offline. This option affects the data that is reported to the fieldbus master.	Clear Freeze
Update mode	The update mode for this service	Cyclically On data change Single shot Change of state on trigger

Timing:

Parameter	Description	Default
Retries	The number of times to resend this service before the node is disconnected	3
Timeout time (10 ms)	The time to wait before resending this service (in steps of 10 ms)[a]	1000 ms
Update time (10 ms)	The minimum time between two services of this kind (in steps of 10 ms)[a]	1000 ms

[a] The default value is given as 100 in the parameter window. Each change of 10 ms either increases or decreases this value by 1, i.e. 99 represents a poll time of 990 ms and 101 represents a poll time of 1010 ms.

Trigger:

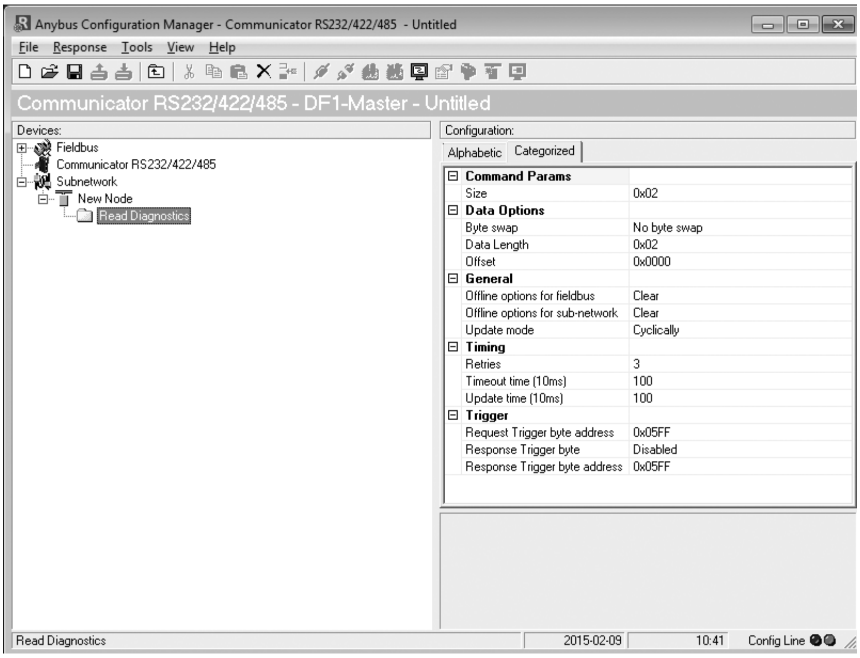
Parameter	Description	Default
Request Trigger byte address	The memory location of the trigger byte this service uses for updates on trigger byte changes	0x05FF
Response Trigger byte	Enables/disables the trigger byte	Disabled
Response Trigger byte address	The memory location of the trigger byte this service uses for updates on trigger byte changes Valid settings range from 0x200 to 0x3FF and 0x400 to 0xFFF	0x05FF

16.6 Integrity Check

This service checks that a node is up and running correctly. A telegram is sent to the node. The node mirrors and returns the telegram. No configuration is needed, apart from the general parameters, common to all services.

16.7 Read Diagnostics

This service reads diagnostic information from the module.


Anybus Configuration Manager - Communicator RS232/422/485 - Untitled
File Response Tools View Help
Communicator RS232/422/485 - DF1-Master - Untitled
Devices:
Fieldbus
Communicator RS232/422/485
Subnetwork
New Node
Read Diagnostics
Configuration:
Alphabetic
Categorized
Read Diagnostics
2015-02-09
10:41
Config Line

Command Params	
Size	0x02
Data Options	
Byte swap	No byte swap
Data Length	0x02
Offset	0x0000
General	
Offline options for fieldbus	Clear
Offline options for sub-network	Clear
Update mode	Cyclically
Timing	
Retries	3
Timeout time (10ms)	100
Update time (10ms)	100
Trigger	
Request Trigger byte address	0x05FF
Response Trigger byte	Disabled
Response Trigger byte address	0x05FF

Command parameters

The command parameter Size decides the amount of data that can be read. The size is given in bytes which means that it always has to be an even number as only whole elements can be read from the slave. One bit/integer element is 2 bytes and one float element is 4 bytes. The range of the size differs, depending on node type:

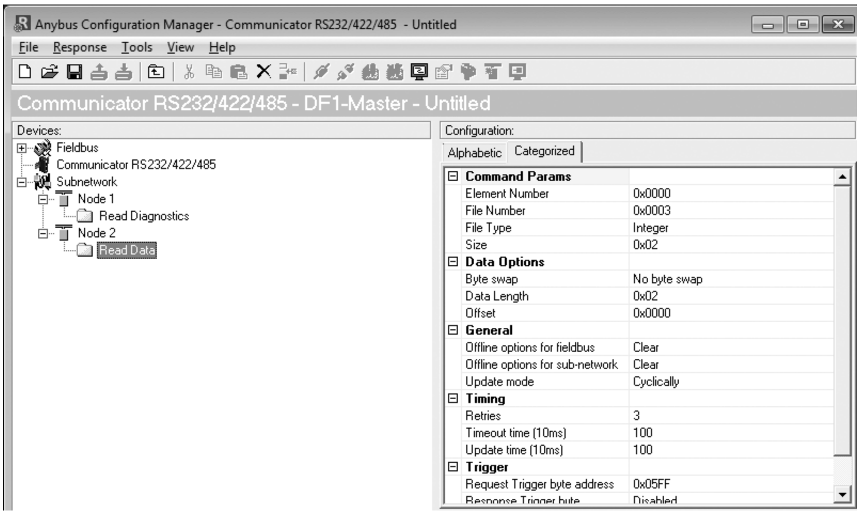
	PLC-5	SLC500	MicroLogix
Size range (in bytes)	1–26	1–28	1–26

Data options:

Parameter	Description	Valid settings
Byte swap	Determines if the data shall be swapped	No byte swap Swap words Swap double words
Data length	The number of bytes, read from the DF1 network, to write to the area determined by the Offset parameter	≤ Size
Offset	The offset in the internal memory buffer in the module, where the data shall be read.	

16.8 Read Data

This service is used to read data from the nodes in the sub-network.



Anybus Configuration Manager - Communicator RS232/422/485 - Untitled
File Response Tools View Help
Communicator RS232/422/485 - DF1-Master - Untitled
Devices:
Fieldbus
Communicator RS232/422/485
Subnetwork
Node 1
Read Diagnostics
Node 2
Read Data
Configuration:
Alphabetic
Categorized

Command Params	
Element Number	0x0000
File Number	0x0003
File Type	Integer
Size	0x02
Data Options	
Byte swap	No byte swap
Data Length	0x02
Offset	0x0000
General	
Offline options for fieldbus	Clear
Offline options for sub-network	Clear
Update mode	Cyclically
Timing	
Retries	3
Timeout time (10ms)	100
Update time (10ms)	100
Trigger	
Request Trigger byte address	0x05FF
Response Trigger byte	Disabled

Command Parameters

Parameter	Description	Valid settings
Element Number	The element number of the data file to be accessed within the slave.	PLC-5: 0–999 SLC500: 0–255 MicroLogix: 0–255
File number	The file number of the data file to be accessed.	PLC-5: 3, 7, 8, 10–999 SLC500: 3, 7, 8, 10–255 MicroLogix: 3, 7, 8, 10–255
File type	The file type of the data to be accessed.	Integer Bit Float
Size	The number of bytes to read from the slave. One bit/integer element is 2 bytes and one float element is 4 bytes. The parameter must have an even value as only whole elements can be read from the slave.	PLC-5: 2–240 SLC500: 2–236 MicroLogix: 2–242

Data Options

Parameter	Description	Valid settings
Byte swap	Determines if the data shall be swapped.	No byte swap Swap words Swap double words
Data length	The number of bytes, read from the DF1 network, to write to the area determined by the Offset parameter	≤ Size
Offset	The offset in the internal memory buffer in the module, where the data shall be read. See “Memory Map” on page 15. **Note**: If the control and status registers are enabled (default), first available data location will be: Input area 0x002, Output area 0x202.	-

16.9 Write Data

This service is used to write data to the nodes in the sub-network. The parameters to be configured are the same as for the service Read Data. The only difference is that data is read from the internal memory buffer in the Anybus Communicator and written to the sub-network bus, instead of being written to the internal memory buffer.

17. Sub-network Monitor

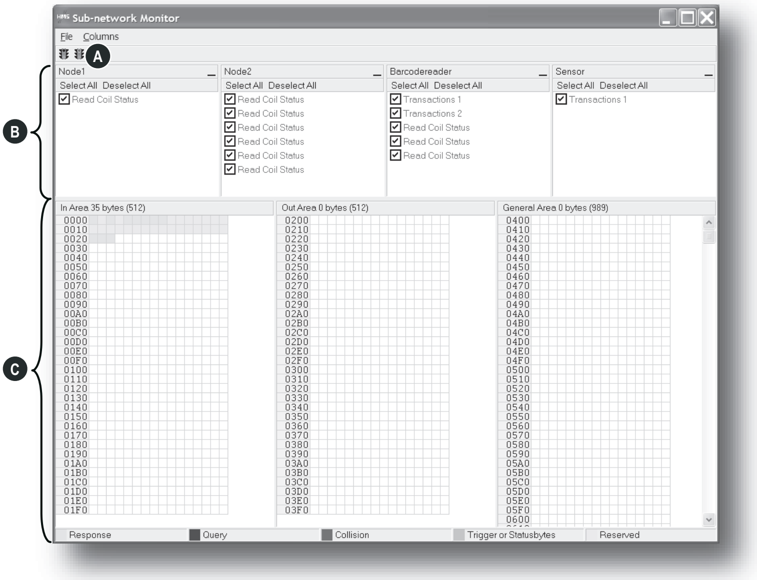
17.1 General

The sub-network Monitor is intended to simplify configuration and troubleshooting of the sub-network. Its main function is to display the data allocated for sub-network communication and detect if any area has been allocated twice (i.e if a collision has occurred).

All configured nodes, and their transactions, are listed in the middle of the screen (B). Selecting and deselecting single transactions makes it possible to view any combination of allocated data.

Note: The sub-network monitor has a negative influence on the overall performance of the gateway. Therefore the monitor functionality should be used with care.

17.2 Operation



Sub-network Monitor

File Columns

Node1
Select All Deselect All
☑ Read Coil Status

Node2
Select All Deselect All
☑ Read Coil Status
☑ Read Coil Status
☑ Read Coil Status
☑ Read Coil Status
☑ Read Coil Status
☑ Read Coil Status

Barcodereader
Select All Deselect All
☑ Transactions 1
☑ Transactions 2
☑ Read Coil Status
☑ Read Coil Status
☑ Read Coil Status

Sensor
Select All Deselect All
☑ Transactions 1

In Area 35 bytes (512)

Out Area 0 bytes (512)

General Area 0 bytes (989)

Response ■ Query ■ Collision ■ Trigger or Statusbytes ■ Reserved

A: Start Network & Stop Network Icons

These icons controls the sub-network activity. To stop all activity, click on the red light. To start the sub-network again, click on the green light.

Start Stop

B: Nodes / Transactions

To view data blocks associated with a transaction, select the transaction in the list. The corresponding data will then appear in the Monitor Section (C).

C: Monitor Section

This section visualizes how data is allocated in the Input, Output and General Data areas.

Color	Meaning
White	Not allocated
Yellow	Data allocated by a Response or Consume transaction
Blue	Data allocated by a Query or Produce transaction
Red	Collision; area has been allocated more than once
Grey	Reserved (illustrates memory consumption, area can be allocated if necessary)
Green	Data allocated by Trigger byte, Transmit/Receive Counter, or Control/Status Registers

18. Node Monitor

18.1 General

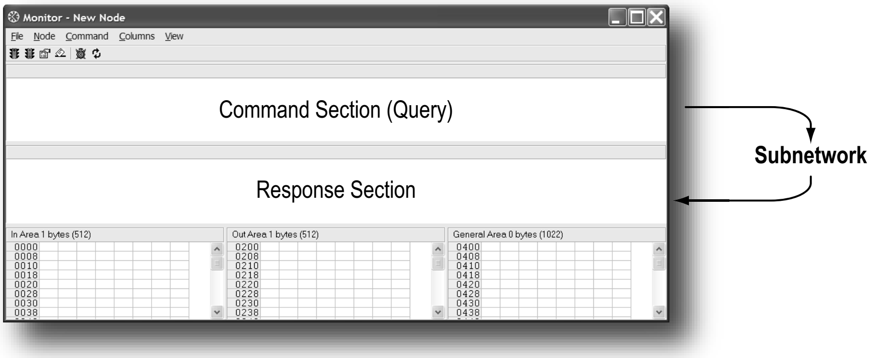
The Node Monitor can provide valuable information when setting up the communication with the sub-network, by allowing individual commands to be issued manually, and monitoring the response (if applicable). It also provides an overview of the memory used by a particular node.

Note: The node monitor has a negative influence on the overall performance of the gateway, i.e. it should be used only when necessary.

The Node Monitor behaves somewhat differently in the three protocol modes:

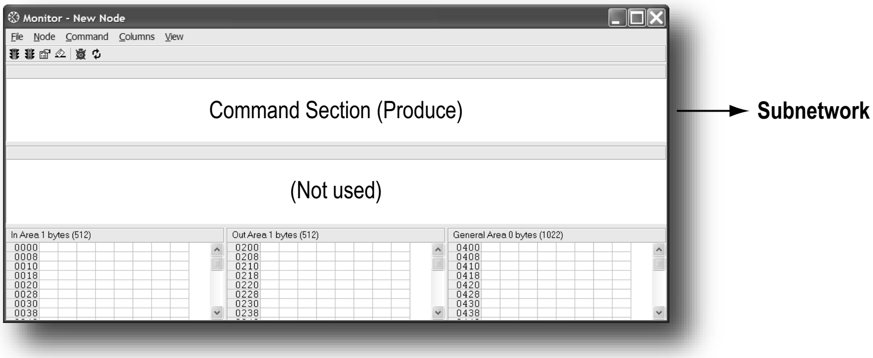
- **Master Mode and DF1 Master Mode**

  The selected Command (Query Transaction) or Service is sent to the sub-network. The response to the Query can be monitored in the Response Section.

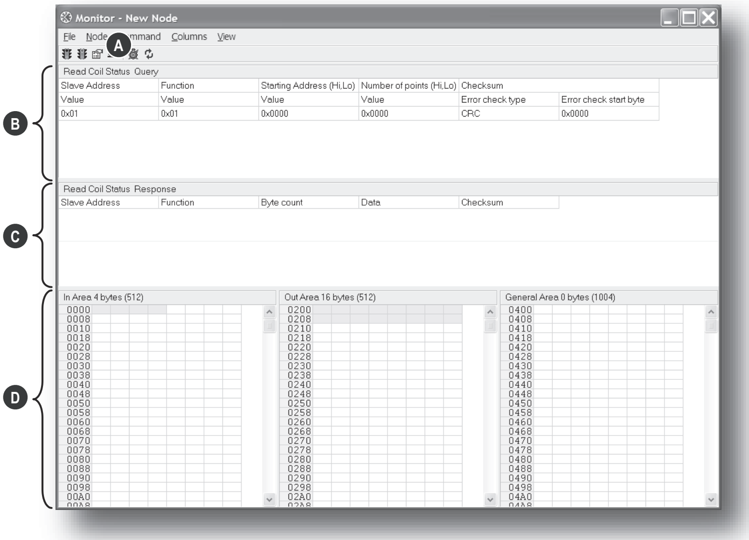


- **Generic Data Mode**

  The selected command (Transaction Produce) is sent to the sub-network. It is not possible to monitor any responses etc. generated by other nodes.



18.2 Navigating the Node Monitor



Read Coil Status Query

Slave Address	Function	Starting Address (Hi,Lo)	Number of points (Hi,Lo)	Checksum	
Value	Value	Value	Value	Error check type	Error check start byte
0x01	0x01	0x0000	0x0000	CRC	0x0000

Read Coil Status Response

Slave Address	Function	Byte count	Data	Checksum

In Area 4 bytes (512)	Out Area 16 bytes (512)	General Area 0 bytes (1004)
0000	0200	0400
0008	0208	0408
0010	0210	0410
0018	0218	0418
0020	0220	0420
0028	0228	0428
0030	0230	0430
0038	0238	0438
0040	0240	0440
0048	0248	0448
0050	0250	0450
0058	0258	0458
0060	0260	0460
0068	0268	0468
0070	0270	0470
0078	0278	0478
0080	0280	0480
0088	0288	0488
0090	0290	0490
0098	0298	0498
00A0	02A0	04A0
00A8	02A8	04A8

A: Drop-down Menu & Toolbar Icons

See "Drop-down Menu" on page 96 and "Toolbar Icons" on page 97.

B: Command Section

This section holds the currently selected command. The individual frame objects in the command can be edited in a similar way as in the Transaction and Command Editors.

C: Response Section (Master Mode and DF1 Master Mode only)

This section holds the response to the selected Command.

D: Monitor Section

This section displays the data associated with the node. Areas in dark grey are reserved for the Status & Control Registers, and areas displayed in light grey represent the data that is used by the node.

The data displayed in this section will be refreshed based on the refresh-icons in the toolbar. For more information, see "Toolbar Icons" on page 97.

18.2.1 Drop-down Menu

File

There is only one entry in this menu:

- **Exit**
  This will close the Node Monitor. Note however that if the node has been disabled using “Stop Node” (see below), it will not resume data exchange until enabled again using “Start node”.

Node

This menu controls the data exchange for the node. This feature can help isolate problems associated with a particular node.

- **Start Node**
  Enable the transactions associated with the node.

- **Stop Node**
  Disable the transactions associated with the node.

Command

This menu is used to specify and issue a command manually.

- **Select Command**
  Select a command to be sent to the sub-network.

- **Send Command**
  Send the specified command to the sub-network.

Columns

This menu specifies the number of columns in the Monitor Section.

- **Free**
  The number of columns depends on the width of the window.

- **8 Multiple**
  The number of columns will be fixed to 8.

View

This menu specifies the data representation in the Monitor Section.

- **Hex**
  Display the data in hexadecimal format.

- **Decimal**
  Display the data in decimal format.

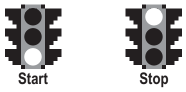
18.2.2 Toolbar Icons

The toolbar features icons for the most commonly used functions.

- **Start Node & Stop Node**

  These icons corresponds to the functions in the “Node” menu.

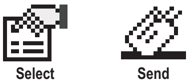
  See also “Node” on page 96.



- **Select Command & Send Command**

  These icons corresponds to the functions in the “Command” menu.

  See also “Command” on page 96.

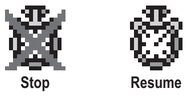


- **Resume Refresh & Stop Refresh**

  The data displayed in the Monitor Section will normally be refreshed automatically (cyclically).

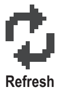
  Click on “Stop” to stop automatic data refresh. Data will now only be refreshed if you click “Refresh” (see below).

  Press “Resume” to resume automatic refreshing of data.



- **Refresh**

  Refreshes the data displayed in the Monitor Section.



19. Data Logger

19.1 General

This feature allows the sub-network traffic to be logged into a buffer for examination. This may provide valuable information when debugging the lowest levels of the sub-network communication.

Note that the logger function is part of the gateway itself and is separate from ACM. This means that logging can be performed even if the gateway is physically disconnected from the PC running ACM.

19.2 Operation

Start & Stop Logging

- **Start logging**
  Select “Start Logging” in the “Tools”-menu. ACM will then prompt for the desired mode of operation, see below.

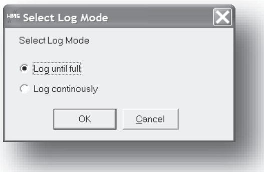
- **Stop logging**
  Select “Stop Logging” in the “Tools”-menu. This will open the log-window, see below.

Modes of Operation

Select the desired mode of operation and click “OK” to start logging data.

- **Log until full**
  Data will be logged until the log-buffer is full.

- **Log continuously**
  Data will be logged continuously until logging is stopped by clicking “Stop Logging”. The log-buffer will contain the most recent data.



Select Log Mode

Select Log Mode

● Log until full

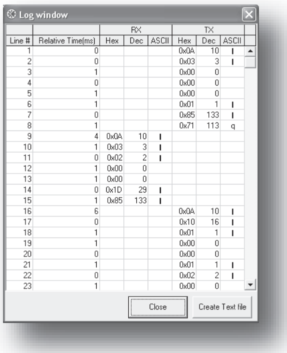
○ Log continously

Log Window

The logged data is displayed in hexadecimal, decimal and ASCII format for both directions. The time between the log-entries is displayed in a separate column.

The data may optionally be saved in ASCII text format by clicking “Create Text file”.

Click “Close” to exit.



Log window

		RX			TX		
Line #	Relative Time(ms)	Hex	Dec	ASCII	Hex	Dec	ASCII
1	0				0x0A	10	ǀ
2	0				0x03	3	ǀ
3	1				0x00	0	
4	0				0x00	0	
5	1				0x00	0	
6	1				0x01	1	ǀ
7	0				0x85	133	ǀ
8	1				0x71	113	q
9	4	0x0A	10	ǀ			
10	1	0x03	3	ǀ			
11	0	0x02	2	ǀ			
12	1	0x00	0				
13	1	0x00	0				
14	0	0x1D	29	ǀ			
15	1	0x85	133	ǀ			
16	6				0x0A	10	ǀ
17	0				0x10	16	ǀ
18	1				0x01	1	ǀ
19	1				0x00	0	
20	0				0x00	0	
21	1				0x01	1	ǀ
22	0				0x02	2	ǀ
23	1				0x00	0	

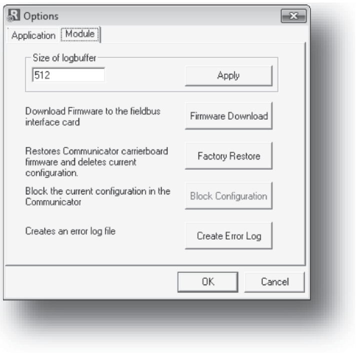
19.3 Configuration

By default, the log-buffer can hold 512 bytes of data in each direction. To specify a different size for the buffer, select “Options” in the “Tools”-menu.

A window with various settings will appear. Select the “Module” tab, and enter the desired number of buffer entries under “Size of logbuffer” (valid settings range from 1–512).

Click “Apply” to validate the new settings.

Click “OK” to exit.



Download Firmware to the fieldbus interface card

Restores Communicator carrierboard firmware and deletes current configuration.

Block the current configuration in the Communicator

Creates an error log file

20. Configuration Wizards

20.1 General

When creating a new sub-network configuration, the Anybus Configuration Manager provides a choice between starting out with a blank configuration, or using a predefined template, a.k.a a wizard.

The wizard automatically creates a sub-network configuration based on information supplied by the user, i.e the user simply has to "fill in the blanks". Note however that this will only work when the sub-network fits the wizard profile; in all other cases the 'Blank Configuration' option must be used.

20.2 Selecting a Wizard Profile

The following window appears each time the Anybus Configuration Manager is started, or upon selecting the 'New' entry in the 'File'-menu (unless it has been disabled in the 'Options'-menu, see "Tools" on page 61).

Currently, the following wizards are available:

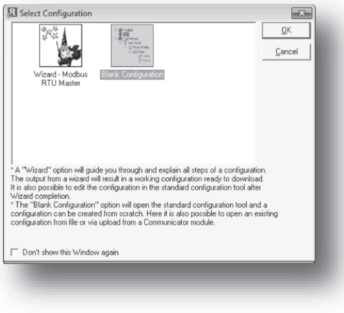
- **Wizard - Modbus RTU Master**

  This option is suitable for Modbus RTU-based networks.

  See also "Wizard - Modbus RTU Master" on page 101.

- **Blank Configuration**

  This option creates an empty configuration.



* A "Wizard" option will guide you through and explain all steps of a configuration. The output from a wizard will result in a working configuration ready to download. It is also possible to edit the configuration in the standard configuration tool after Wizard completion.
* The "Blank Configuration" option will open the standard configuration tool and a configuration can be created from scratch. Here it is also possible to open an existing configuration from file or via upload from a Communicator module.

□ Don't show this Window again

Highlight the desired wizard and click 'OK' to continue.

20.3 Wizard - Modbus RTU Master

This wizard can be used to create a Modbus-RTU-based network configuration based on certain information about the sub-network. The online help system explains each configuration step in detail.

- **Important Notes:**

  Many OEM devices do not fully comply with the Modbus standard. For example, they may implement a variation of this standard or be limited to the use of specific Modbus commands other than the ones used by this wizard. In all cases, the user should consult the documentation of the devices that shall be used on the sub-network for information about their serial communication requirements, and if necessary contact the manufacturer of the device to obtain further information about the serial communication protocol.

  In the event that the wizard doesn't handle a particular Modbus command required by a device, it is possible to specify this command manually as a transaction in the Anybus Configuration Manager.

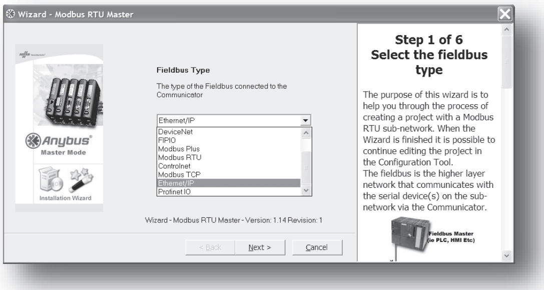
Using this wizard involves the following steps:

Step 1: Communicator Type

Select 'EtherNet/IP'.

Click 'Next' to continue.

Tip: It is possible to return to a previous menu at any time without losing any settings by clicking 'Previous'.



Wizard - Modbus RTU Master

Fieldbus Type

The type of the Fieldbus connected to the Communicator

Ethernet/IP

- DeviceNet
- FIPIO
- Modbus Plus
- Modbus RTU
- Controlnet
- Modbus TCP
- Ethernet/IP
- Profinet IO

Wizard - Modbus RTU Master - Version: 1.14 Revision: 1

Step 1 of 6
Select the fieldbus type

The purpose of this wizard is to help you through the process of creating a project with a Modbus RTU sub-network. When the Wizard is finished it is possible to continue editing the project in the Configuration Tool.
The fieldbus is the higher layer network that communicates with the serial device(s) on the sub-network via the Communicator.

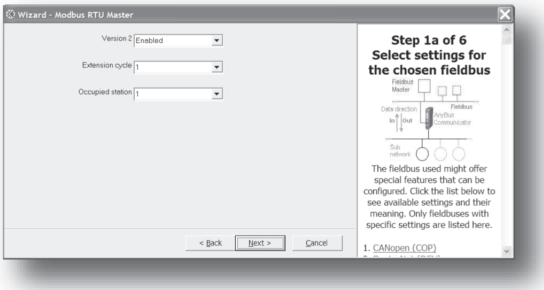
Step 1a: I/O Sizes

Specify the sizes of the input and output data areas.

Click 'Next' to continue.

See also...

- "EtherNet/IP" on page 23
- "I/O Sizes" on page 64



Wizard - Modbus RTU Master

Version 2: Enabled

Extension cycle: 1

Occupied station: 1

Step 1a of 6
Select settings for the chosen fieldbus

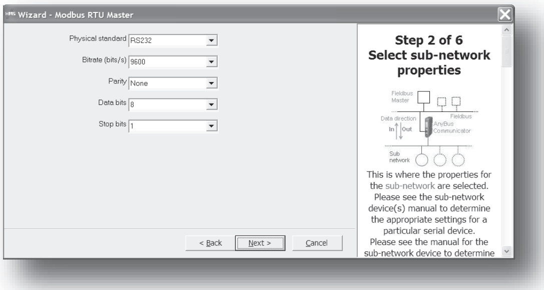
The fieldbus used might offer special features that can be configured. Click the list below to see available settings and their meaning. Only fieldbuses with specific settings are listed here.

1. CANopen (COP)

Step 2: Physical Settings

Select the physical properties of the sub-network.

Click 'Next' to continue.



Wizard - Modbus RTU Master

Physical standard: RS232

Bitrate (bits/s): 9600

Parity: None

Data bits: 8

Stop bits: 1

Step 2 of 6
Select sub-network properties

This is where the properties for the sub-network are selected.
Please see the sub-network device(s) manual to determine the appropriate settings for a particular serial device.
Please see the manual for the sub-network device to determine

Steps 3 - 6

Consult the online help system for further information.

21. Control and Status Registers

21.1 General

The Control and Status Registers are disabled by default, but can be enabled using ACM (see "Control/ Status Word" on page 65). These registers form an interface for exchanging status information between the sub-network and the fieldbus control system.

The main purpose of these registers is to...

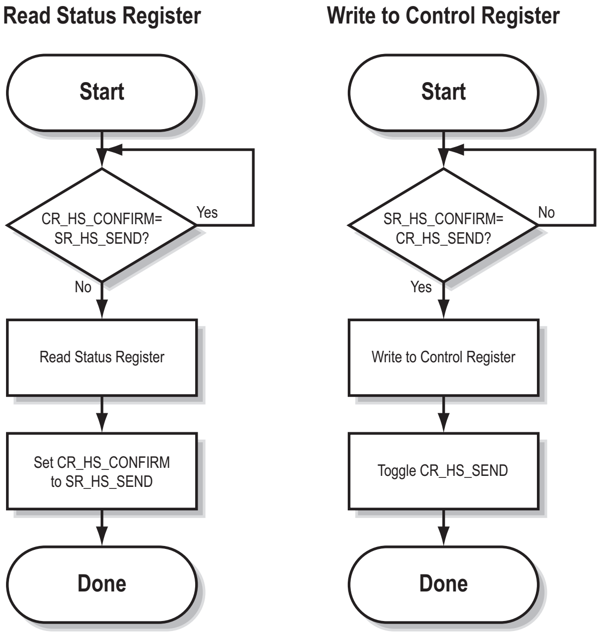
- Report sub-network related problems to the fieldbus control system
- Ensure that only valid data is exchanged in both directions
- Enable the fieldbus control system to start/stop data exchange with selected nodes on the sub-network

If enabled, these registers occupy the first two bytes in the input and output data areas (0x000–0x001 and 0x200–0x201 respectively), which means they can be accessed from the fieldbus just like any other data in these areas.

Note: Internally, these registers are stored in Motorola-format (i.e. MSB first). If the higher level network uses a different byte order, the upper and lower bytes will appear swapped.

21.1.1 Handshaking Procedure

A special handshaking procedure, which is illustrated in the two flowcharts below, must be followed when accessing these registers to ensure that both parts receive proper information.



21.1.2 Data Consistency

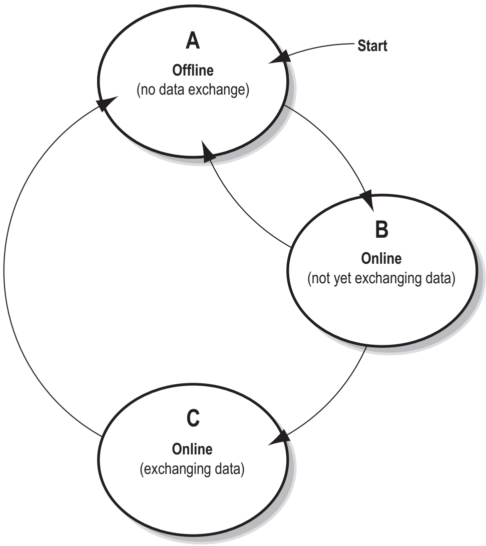
The “Data Valid”-bits in the Control and Status Registers are used to ensure data consistency during start-up and fieldbus offline/online transitions.

If the “Control/Status Word”-parameter in ACM is set to “Enabled”, the gateway will wait for the fieldbus control system to set the “Data Valid”-bit in the Control Register before it starts exchanging data on the sub-network.

If the same parameter is set to “Disabled” or “Enabled but no startup lock”, communication will start as soon as the fieldbus goes online.

State Machine

The fieldbus network participation can be described using a state machine as described below.



A: Offline (No data exchange)

1. Clear the “Data Valid”-bit in the Control Register.
2. Write initial data to the Output Area according to the sub-network configuration.
3. Wait until the fieldbus control system and the gateway are online on the fieldbus network, and shift to state B.

B: Online (Not yet exchanging data)

4. Wait until the “Data Valid”-bit in the Status Register is cleared by the gateway.
5. Set the “Data Valid”-bit in the Control Register.
6. When the “Data Valid”-bit in the Status Register is set by the gateway, shift to state C.
7. If the gateway goes offline on the fieldbus, shift to state A.

C: Online (Exchanging data)

Exchanging valid data in both directions.
If the gateway goes offline on the fieldbus, shift to state A.

Note: The gateway cannot spontaneously clear the “Data Valid”-bit in the Status Register.

Latency

The “Data Valid”-bit in the Status Register may in some cases be delayed. This latency can be caused by a missing node or a bad connection to a node with a long timeout value assigned to it.

Therefore, the fieldbus control system should not wait for this bit to be set before communicating with the sub-network devices; it should be considered as an aid for the fieldbus control system to know when all data has been updated.

21.2 Status Register Contents (Gateway to Control System)

21.2.1 General Information

The Status Register is (if enabled) located at 0x000–0x001 and constitutes a bit-field as follows:

bit(s)	Name	Description
15	Send (SR_HS_SEND)	These bits control the handshaking towards the fieldbus control system. See also... - “Handshaking Procedure” on page 102 - “Control Register Contents (Control System to Gateway)” on page 106
14	Confirm (SR_HS_CONFIRM)	
13	Data Valid (Master Mode and DF1 Master Mode Only)	This bit is set when all transactions have been executed successfully at least once. Once set, it will not change. 1:Data Valid 0:Data not Valid **Note:** This bit is not used in Generic Data Mode.
12... 8	Status Code	This field holds the last status report from the gateway. See also... - “Status Codes in Master Mode and DF1 Master Mode” on page 104 - “Status Code in Generic Data Mode” on page 105
7... 0	Data	

Note: Internally, this is treated as a Motorola-format word (i.e. MSB first). If the higher level network uses a different byte order, the upper and lower bytes will appear swapped.

21.2.2 Status Codes in Master Mode and DF1 Master Mode

(This table is valid only in Master Mode and DF1 Master Mode).

Code	Condition	Type	Data	Description
0x00	Retransmission Counter Updated	Warning	Counter	The number of retransmissions on the sub-network has increased. If this problem persists, this may eventually trigger a Single- or Multiple Node(s) Missing condition.
0x01	Single Node Missing	Error	Slave address	A single node is missing.
0x02	Multiple Nodes Missing	Error	Number of nodes	Multiple nodes are missing.
0x03	Buffer Overrun	Warning	Slave address	A node returned more data than expected.
0x04	Other Error	Error	Slave address	Undefined error
0x1F	No Error	Warning	-	No errors

Note: Conditions of type “Error” will eventually be followed by a “No Error” condition when the cause has been resolved. Conditions of type “Warning” are however considered informational and may not necessarily be followed by a “No Error” condition later on.

21.2.3 Status Code in Generic Data Mode

(This table is valid only in Generic Data Mode).

Code	Condition	Type	Data	Description
0x00	Invalid Transaction Counter Updated	Error	Counter	The number of invalid transactions (i.e. received transactions which does not match any of the consume-transactions defined in the sub-network configuration) has increased.
0x01	Frame Error	Warning	-	End character is enabled, but a message delimiter timeout occurs prior to receiving it.
0x02	Offline Timeout Counter Updated	Error	Counter	The of number of timed out consume-transactions has increased. See also... - "Consume Transactions" on page 73 (Offline timeout time)
0x03	Buffer Overrun	Warning	-	A node returned more data than expected - or - the gateway was unable to finish processing a message prior to receiving a new one.
0x04	Other Error	Error	-	Undefined error
0x1F	No Error	Warning	-	No errors

Note: Conditions of type "Error" will eventually be followed by a "No Error" condition when the cause no longer is detected. Conditions of type "Warning" are however considered informational and may not necessarily be followed by a "No Error" condition later on.

21.3 Control Register Contents (Control System to Gateway)

21.3.1 General Information

The Control Register is (if enabled) located at 0x200–0x201 and constitutes a bit-field as follows:

bit(s)	Name	Description
15	Confirm (CR_HS_CONFIRM)	These bits control the handshaking towards the gateway. See also... - “Handshaking Procedure” on page 102 - “Status Register Contents (Gateway to Control System)” on page 104
14	Send (CR_HS_SEND)	
13	Data Valid	This bit controls data consistency (see “Data Consistency” on page 103). 1:Output Area valid; exchange data on the sub-network 0:Output Area not valid; do not exchange data on the sub-network **Note:** This bit is only relevant if the Control/Status Registers are set as “Enabled”
12	Execute Command	If set, the specified command will be executed by the gateway (see below).
11... 8	Control Code	This field holds commands which can be executed by the gateway (see below). See also... - “Control Codes in Master Mode and DF1 Master Mode” on page 106 - “Control Codes in Generic Data Mode” on page 106
7... 0	Data	

Note: Internally, this is treated as a Motorola-format word (i.e. MSB first). If the higher level network uses a different byte order, the upper and lower bytes will appear to be swapped.

21.3.2 Control Codes in Master Mode and DF1 Master Mode

(This table is valid only in Master Mode and DF1 Master Mode).

Code	Instruction	Data	Description
0x00	Disable Node	Actual node address	Disables the specified node.
0x01	Enable Node	Actual node address	Enables a previously disabled node.
0x02	Enable Nodes	Actual number of nodes to enable	Enables the specified number of nodes, starting from the first node in the configuration. Remaining nodes will be disabled.

21.3.3 Control Codes in Generic Data Mode

(No Control Codes are currently supported in this mode).

22. CIP Object Implementation

22.1 General

The following CIP-objects are implemented in this product:

Mandatory Objects

Object	Page
Identity Object, Class 01h	108
Message Router, Class 02h	109
Assembly Object, Class 04h	110
Port Object, Class F4h	114
TCP/IP Interface Object, Class F5h	115
Ethernet Link Object, Class F6h	116

Vendor Specific Objects

Object	Page
Diagnostic Object, Class AAh	111
Parameter Data Input Mapping Object, Class B0h	112
Parameter Data Output Mapping Object, Class B1h	113

22.2 Identity Object, Class 01h

22.2.1 General Information

Object Description

-

Supported Services

Class services:	Get Attribute All Get Attribute Single
Instance services:	Get Attribute All Get Attribute Single Reset

22.2.2 Class Attributes

#	Access	Name	Type	Value	Description
1	Get	Revision	UINT	0001h	Revision 1

22.2.3 Instance Attributes

#	Access	Name	Type	Value	Description
1	Get	Vendor ID	UINT	Default: 005Ah	HMS Industrial Networks AB
2	Get	Device Type	UINT	Default: 000Ch	Communication Adapter
3	Get	Product Code	UINT	Default: 0054h	Anybus Communicator
4	Get	Revision	Struct of:		-
			USINT		Major fieldbus version
			USINT		Minor fieldbus version
5	Get	Status	WORD	-	Device status, see table below
6	Get	Serial Number	UDINT	Serial number	(set at production)
7	Get	Product Name	SHORT_STRING	Anybus Communicator	Name of product

Device Status

bit(s)	Name
0	Module Owned
1	(reserved)
2	Configured
3	(reserved)
4... 7	Extended Device Status:
	Value:Meaning: 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)

22.3 Message Router, Class 02h

22.3.1 General Information

Object Description

-

Supported Services

Class services: -
Instance services: -

22.3.2 Class Attributes

-

22.3.3 Instance Attributes

-

22.4 Assembly Object, Class 04h

22.4.1 General Information

Object Description

This object provides access to the I/O Data in the input and output data areas in the Anybus Communicator.

See also...

- "EtherNet/IP" on page 23
- "Fieldbus Settings" on page 64

Supported Services

Class services:	Get Attribute Single
Instance services:	Get Attribute Single Set Attribute Single

22.4.2 Class Attributes

#	Access	Name	Type	Value	Description
1	Get	Revision	UINT	0002h	Revision 2
2	Get	Max Instance	UINT	-	The highest initiated instance no.

22.4.3 Instance 64h (100) Attributes

This instance corresponds to I/O data (input) in the gateway.

Note: If the I/O input data size is set to 0 this instance will NOT be initialized.

#	Access	Name	Type	Value	Description
3	Get	Data	Array of BYTE	-	Data produced by the gateway

22.4.4 Instance 96h (150) Attributes

Note: If the I/O output data size is set to 0 this instance will NOT be initialized.

#	Access	Name	Type	Value	Description
3	Set	Data	Array of BYTE	-	Data consumed by the gateway[a]

[a] Rockwell Automation PLCs have the first four bytes consumed by a device defined as status information. This behavior is specific to devices from Rockwell Automation and is not defined in the EtherNet/IP specification. However, since all known PLCs are implemented this way, the Anybus Communicator adopts this behavior and strips off the corresponding four bytes from the consumed data.

22.4.5 Instance C6h (198) Attributes (Heartbeat Input-Only)

This instance is used as heartbeat for input-only connections, and does not carry any data.

22.4.6 Instance C7h (199) Attributes (Heartbeat, Listen-Only)

This instance is used as heartbeat for listen-only connections, and does not carry any data.

22.5 Diagnostic Object, Class AAh

22.5.1 General Information

Object Description

This object groups diagnostic information for the fieldbus interface.

Supported Services

Class services: Get Attribute All

Instance services: Get Attribute Single

22.5.2 Class Attributes

#	Access	Name	Type	Value	Description
1	Get	Revision	UINT	0001h	Revision 1

22.5.3 Instance Attributes, Instance 01h

#	Access	Name	Type	Description
01h	Get	Module serial number	UDINT	Serial number
02h	Get	Vendor ID	UINT	Manufacturer Vendor ID
03h	Get	Fieldbus Type	UINT	Fieldbus Type
04h	Get	Module Software version	UINT	Module software version
0Ah	Get	Module Type	UINT	Module Type
0Fh	Get	IN cyclic I/O length	UINT	Size of I/O Input area (in bytes)
11h	Get	IN total length	UINT	Total number of IN bytes supported
12h	Get	OUT cyclic I/O length	UINT	Size of I/O Output area (in bytes)
14h	Get	OUT total length	UINT	Total number of OUT bytes supported

22.6 Parameter Data Input Mapping Object, Class B0h

22.6.1 General Information

Object Description

This object can be used to access input data acyclically, and is set up dynamically based on the Parameter Data Mailbox initialization (see “Parameter Data Initialization (Explicit Data)” on page 119).

See also...

- “EtherNet/IP” on page 23
- “Fieldbus Settings” on page 64
- “Parameter Data Output Mapping Object, Class B1h” on page 113
- “Parameter Data Initialization (Explicit Data)” on page 119

Supported Services

Class services:	Get Attribute All
Instance services:	Get Attribute Single

22.6.2 Class Attributes

#	Access	Name	Type	Value	Description
1	Get	Revision	UINT	0001h	Revision 1

22.6.3 Instance Attributes, Instance 01h

Each attribute corresponds to a block of Input Data. Note that the size and location of each block must be specified using the Anybus Configuration Manager.

For more information, see A-119 “Parameter Data Initialization (Explicit Data)”.

#	Access	Name	Type	Description
01h	Get	Data	Array of USINT	Mapped block if Input Data
02h	Get	Data	Array of USINT	Mapped block if Input Data
02h	Get	Data	Array of USINT	Mapped block if Input Data
02h	Get	Data	Array of USINT	Mapped block if Input Data
02h	Get	Data	Array of USINT	Mapped block if Input Data
02h	Get	Data	Array of USINT	Mapped block if Input Data
...	...	...	...	...
32h	Get	Data	Array of USINT	Mapped block if Input Data

22.7 Parameter Data Output Mapping Object, Class B1h

22.7.1 General Information

Object Description

This object can be used to access output data acyclically, and is set up dynamically based on the Parameter Data Mailbox initialization (see "Parameter Data Initialization (Explicit Data)" on page 119).

See also...

- "EtherNet/IP" on page 23
- "Fieldbus Settings" on page 64
- "Parameter Data Input Mapping Object, Class B0h" on page 112
- "Parameter Data Initialization (Explicit Data)" on page 119

Supported Services

Class services:	Get Attribute All
Instance services:	Get Attribute Single Set Attribute Single

22.7.2 Class Attributes

#	Access	Name	Type	Value	Description
1	Get	Revision	UINT	0001h	Revision 1

22.7.3 Instance Attributes, Instance 01h

Each attribute corresponds to a block of output data. Note that the size and location of each block must be specified using the Anybus Configuration Manager.

For more information, see "Parameter Data Initialization (Explicit Data)" on page 119

#	Access	Name	Type	Description
01h	Get/Set	Data	Array of USINT	Mapped block of Output Data
02h	Get/Set	Data	Array of USINT	Mapped block of Output Data
01h	Get/Set	Data	Array of USINT	Mapped block of Output Data
02h	Get/Set	Data	Array of USINT	Mapped block of Output Data
01h	Get/Set	Data	Array of USINT	Mapped block of Output Data
02h	Get/Set	Data	Array of USINT	Mapped block of Output Data
...	...	...	...	...
32h	Get/Set	Data	Array of USINT	Mapped block of Output Data

22.8 Port Object, Class F4h

22.8.1 General Information

Object Description

-

Supported Services

Class services:	Get Attribute All Get Attribute Single
Instance services:	Get Attribute All Get Attribute Single

22.8.2 Class Attributes

#	Access	Name	Type	Value	Description
1	Get	Revision	UINT	0001h	Revision 1
2	Get	Max Instance	UINT	0002h	2 is the highest instance number
3	Get	No. of instances	UINT	0001h	1 instance is implemented
8	Get	Entry Port	UINT	0002h	Returns the instance of the Port object that describes the port.
9	Get	All Ports	Array of STRUCT {UINT; UINT;}	0000h 0000h 0000h 0000h 0004h 0002h	Array of structure containing attributes 1 and 2 from each instance. Instance 1 is at byte offset 4. Instance 2 is at byte offset 8, etc. The 4 bytes at offset 0 shall be 0. (Default)

22.8.3 Instance Attributes, Instance 02h

#	Access	Name	Type	Value	Comments
1	Get	Port Type	UINT	0004h	TCP/IP
2	Get	Port Number	UINT	0002h	Port 2
3	Get	Port Object	Struct of:		
		Path Size	UINT	0002h	-
		Path	Padded EPATH	20 F5 24 01h	TCP class, Instance 1
4	Get	Port Name	SHORT_STIRNG	‘TCP/IP’	Name of port
8	Get	Node Address	Padded EPATH	-	-

22.9 TCP/IP Interface Object, Class F5h

22.9.1 General Information

Object Description

This object groups TCP/IP-related settings.

See also...

- "Basic Network Configuration" on page 37
- "Fieldbus Settings" on page 64

Supported Services

Class services:	Get Attribute All Get Attribute Single
Instance services:	Get Attribute All Get Attribute Single Set Attribute Single

22.9.2 Class Attributes

#	Access	Name	Type	Value	Description
1	Get	Revision	UINT	0001h	Revision 3
2	Get	Max Instance	UINT	0001h	1 is the highest instance number
3	Get	No. of instances	UINT	0001h	1 instance is implemented

22.9.3 Instance Attributes

#	Access	Name	Type	Value	Comments
1	Get	Status	DWORD	00000001h	Attribute #5 contains valid information.
2	Get	Configuration Capability	DWORD	00000014h	Attribute #5 is settable Capable of obtaining network configuration via DHCP.
3	Get/Set	Configuration Control	DWORD	-	Value:Meaning: 0 Configuration from non-volatile memory 2 Configuration from DHCP
4	Get	Port Object	Struct of:		
		Path Size	UINT	0002h	2 words
		Path	Padded EPATH	20 F6 24 01h	Path to Ethernet Class, Instance 1
5	Get/Set	Interface Configuration	Struct of:		
		IP Address	UDINT	-	IP address
		Subnet Mask	UDINT	-	Subnet mask
		Gateway Address	UDINT	-	Gateway Address
		Name Server 1	UDINT	-	Primary DNS
		Name Server 2	UDINT	-	Secondary DNS
		Domain Name	STRING	-	Default domain name
6	Get/Set	Host Name	STRING	-	Host name

22.10 Ethernet Link Object, Class F6h

22.10.1 General Information

Object Description

This object groups diagnostic information for the Ethernet interface.

See also...

- "Basic Network Configuration" on page 37

Supported Services

Class services:	Get Attribute All Get Attribute Single
Instance services:	Get Attribute All Get Attribute Single

22.10.2 Class Attributes

#	Access	Name	Type	Value	Description
1	Get	Revision	UINT	0001h	Revision 1
2	Get	Max Instance	UINT	0001h	2 is the highest instance number
3	Get	No. of instances	UINT	0001h	2 instance is implemented

22.10.3 Instance Attributes

#	Access	Name	Type	Value	Comments
1	Get	Interface Speed	UDINT	10 or 100	Actual ethernet interface speed
2	Get	Interface Flags	DWORD	-	-
3	Get	Physical Address	Array of 6 USINTS	(MAC ID)	Physical network address
4	Get	Interface Counters	Struct:		
		In Octets	UDINT	-	Octets received on the interface
		In Ucast Packets	UDINT	-	Unicast packets received on the interface
		In NUcast Packets	UDINT	-	Non-unicast packets received on the interface
		In Discards	UDINT	-	Inbound packets with unknown protocol
		In Errors	UDINT	-	Inbound packets that contain errors (does not include discards)
		In Unknown Protos	UDINT	-	Inbound packets with unknown protocol
		Out Octets	UDINT	-	Octets sent on the interface
		Out Ucast Packets	UDINT	-	Unicast packets sent on the interface
		Out NUcast Packets	UDINT	-	Non-unicast packets sent on the interface
		Out Discards	UDINT	-	Outbound packets with unknown protocol
		Out Errors	UDINT	-	Outbound packets that contain errors (does not include discards)
5	Get	Media Counters	Struct:		
		Alignment Errors	UDINT	-	Frames received that are not an integral number of octets in length
		FCS Errors	UDINT	-	Frames received that do not pass the FCS check
		Single Collisions	UDINT	-	Successfully transmitted frames which experienced exactly one collision
		Multiple Collisions	USINT	-	Successfully transmitted frames which experienced more than one collision
		SQE Test Errors	UDINT	0	-
		Deferred Transmissions	UDINT	-	Frames for which first transmission attempt is delayed because the medium is busy
		Late Collisions	UDINT	-	Number of times a collision is detected later than 512 bit-times into the transmission of a packet
		Excessive Collisions	UDINT	-	Frames for which a transmission fails due to excessive collisions
		MAC Transmit Errors	UDINT	-	Frames for which transmission fails due to an internal MAC sublayer receive error
		Carrier Sense Errors	UDINT	-	Times that the carrier sense condition was lost or never asserted when attempted to transmit a frame
		Frame Too Long	UDINT	-	Frames received that exceed the maximum permitted frame size
		MAC Receive Errors	UDINT	-	Frames for which reception on an interface fails due to an internal MAC sublayer receive error

23. Advanced Fieldbus Configuration

23.1 General

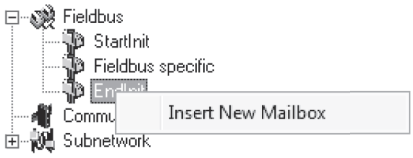
The fieldbus interface of the gateway consists of an embedded Anybus-S communication interface. Normally, the Anybus-S configuration settings are set up automatically by the gateway. However, advanced users can configure the Anybus-S card for specific features. This chapter assumes that the reader is familiar with the Anybus-S and it's application interface. For more information about the Anybus-S platform, consult the Anybus-S Parallel Design Guide.

The standard initialization parameters are determined by the sub-network configuration. Information about the amount of input and output data used for sub-network communication is used by ACM to create the configuration message that sets the sizes of the input and output data areas in the Dual Port RAM of the embedded Anybus-S interface. It is possible to add fieldbus specific mailbox messages to customize the initialization. This is done in the Mailbox Editor, see below.

(A mailbox message is a HMS specific command structure used for low-level communication with an Anybus-S interface. Consult the Anybus-S Parallel Design Guide and the fieldbus appendix for the desired fieldbus for further information.)

23.2 Mailbox Editor

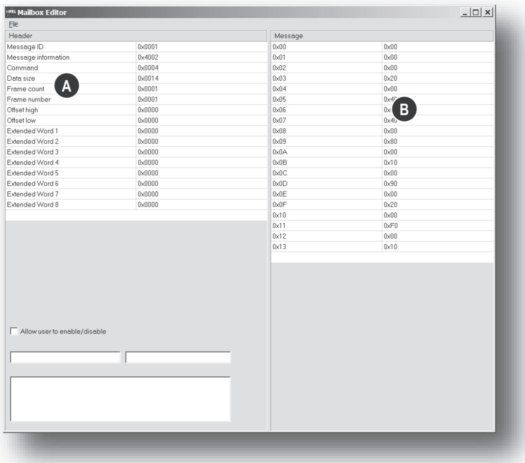
To add a mailbox message to the configuration, right-click on "EndInit" and select "Insert New Mailbox".



A mailbox message consists of a Header section and a data section where the Header consists of 16 words (32 bytes) and the data section consists of up to 128 words (256 bytes). All fields are editable except the Message information field that is fixed to 0x4002, which means that only fieldbus specific mailbox messages can be entered here.

The mailbox message is presented as two columns; one contains header information (A), the other one contains the message data (B).

To add message data, simply change the Data size parameter in the header column (A), and the corresponding number of bytes will appear in the message data column (B).



Header	
Message ID	0x0001
Message information	0x4002
Command	0x0004
Data size	0x0014
Frame count	0x0001
Frame number	0x0001
Offset high	0x0000
Offset low	0x0000
Extended Word 1	0x0000
Extended Word 2	0x0000
Extended Word 3	0x0000
Extended Word 4	0x0000
Extended Word 5	0x0000
Extended Word 6	0x0000
Extended Word 7	0x0000
Extended Word 8	0x0000

Message	
0x00	0x00
0x01	0x00
0x02	0x00
0x03	0x20
0x04	0x00
0x05	0x4[illegible]
0x06	0x[illegible]
0x07	0x4[illegible]
0x08	0x00
0x09	0x80
0x0A	0x00
0x0B	0x10
0x0C	0x00
0x0D	0x90
0x0E	0x00
0x0F	0x20
0x10	0x00
0x11	0xF0
0x12	0x00
0x13	0x10

☐ Allow user to enable/disable

For more information about fieldbus specific mailbox messages, consult the separate Anybus-S Fieldbus Appendix for the fieldbus you are using. For general information about the Anybus-S platform, consult the Anybus-S Design Guide.

A. Parameter Data Initialization (Explicit Data)

A.1 General

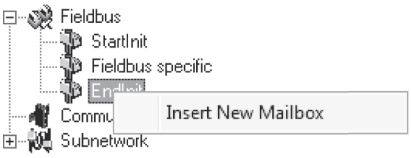
The portion of the input and output data that is declared as parameter data cannot be accessed from the network unless it has been properly initialized.

The purpose of this procedure is to specify which data blocks in the input and output data areas to associate with the instance attributes in the Parameter Data Input Mapping Object and the Parameter Data Output Mapping Object.

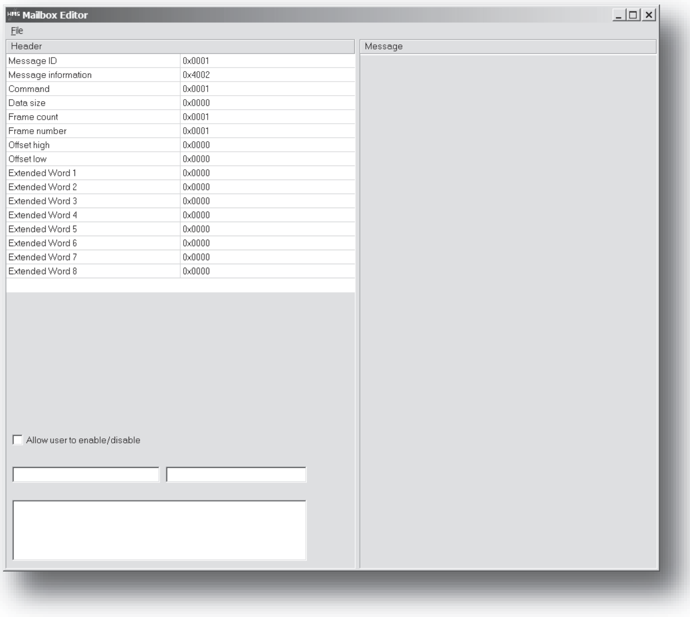
To achieve this, it is required to set up two mailbox messages in the Mailbox Editor of the Anybus Configuration Manager. For more information about the Mailbox Editor, see "Mailbox Editor" on page 118.

A.2 Add a Mailbox Message

To add a mailbox message to the configuration, right-click on 'EndInit' and select 'Insert New Mailbox'.



This causes the following window to appear:



Header	
Message ID	0x0001
Message information	0x4002
Command	0x0001
Data size	0x0000
Frame count	0x0001
Frame number	0x0001
Offset high	0x0000
Offset low	0x0000
Extended Word 1	0x0000
Extended Word 2	0x0000
Extended Word 3	0x0000
Extended Word 4	0x0000
Extended Word 5	0x0000
Extended Word 6	0x0000
Extended Word 7	0x0000
Extended Word 8	0x0000

□ Allow user to enable/disable

This window, a.k.a. the Mailbox Editor, will be used in the examples later in this chapter.

See also "Mailbox Editor" on page 118.

A.3 Mapping Input Parameter Data to EtherNet/IP

Example

In the following example, a total of 160 bytes of data will be mapped to the Parameter Data Input Mapping Object. The data is made up of 5 separate data blocks, each associated with a particular instance attribute.

To achieve this, perform the following steps:

1. Add a new mailbox message to the configuration (see "Add a Mailbox Message" on page 119).

2. Change the 'Command'-value in the mailbox header to 0084h.

3. Adjust the 'Data Size'-value in the mailbox header (left column). In this example, the size shall be set to 20 (0014h), since each mapped attribute occupies 4 bytes of mailbox data.

4. Specify the mapping locations for the attributes in the mailbox data section. As mentioned above, each mapping entry needs 4 bytes; two bytes specifying the offset[1] of the data block, followed by two bytes which specify the length of the data block. Note that these values must be entered in big endian (Motorola) format.

In this example, this gives us the following mailbox data:

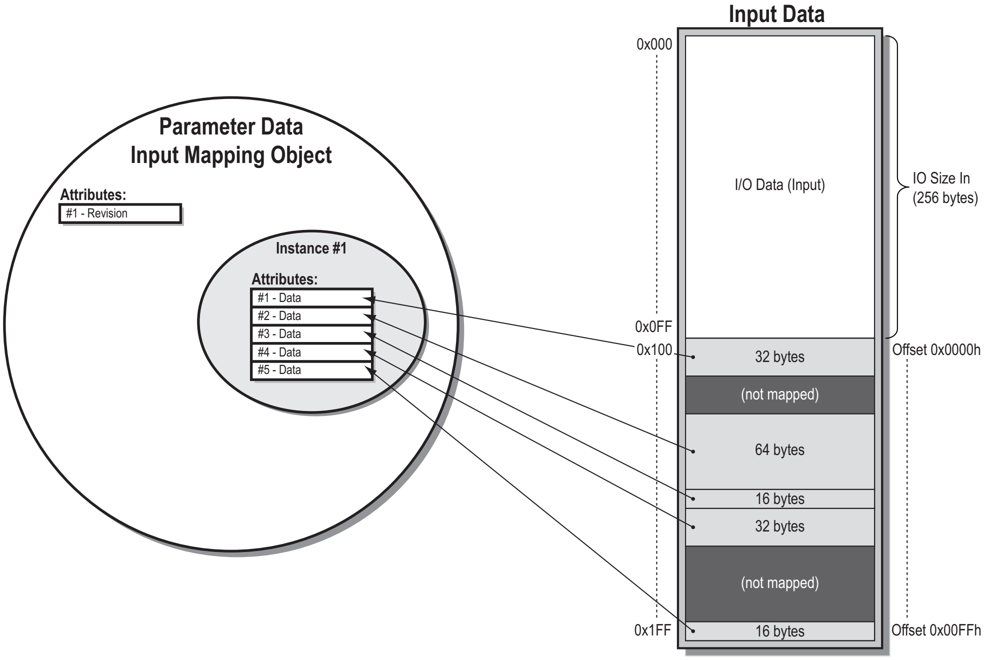
Mailbox Data		Attribute no.	Comments
Location	Data		
0x00	0x00	1	Offset = 0000h
0x01	0x00		
0x02	0x00		Size = 32 bytes
0x03	0x20		
0x04	0x00	2	Offset = 0040h
0x05	0x40		
0x06	0x00		Size = 64 bytes
0x07	0x40		
0x08	0x00	3	Offset = 0080h
0x09	0x80		
0x0A	0x00		Size = 16 bytes
0x0B	0x10		
0x0C	0x00	4	Offset = 0090h
0x0D	0x90		
0x0E	0x00		Size = 32 bytes
0x0F	0x20		
0x10	0x00	5	Offset = 00F0h
0x11	0xF0		
0x12	0x00		Size = 16 bytes
0x13	0x10		

As shown in the table above, the attributes are numbered in the order they are mapped, i.e. it is possible to rearrange the attribute numbering by physically changing the mapping order in the mailbox data.

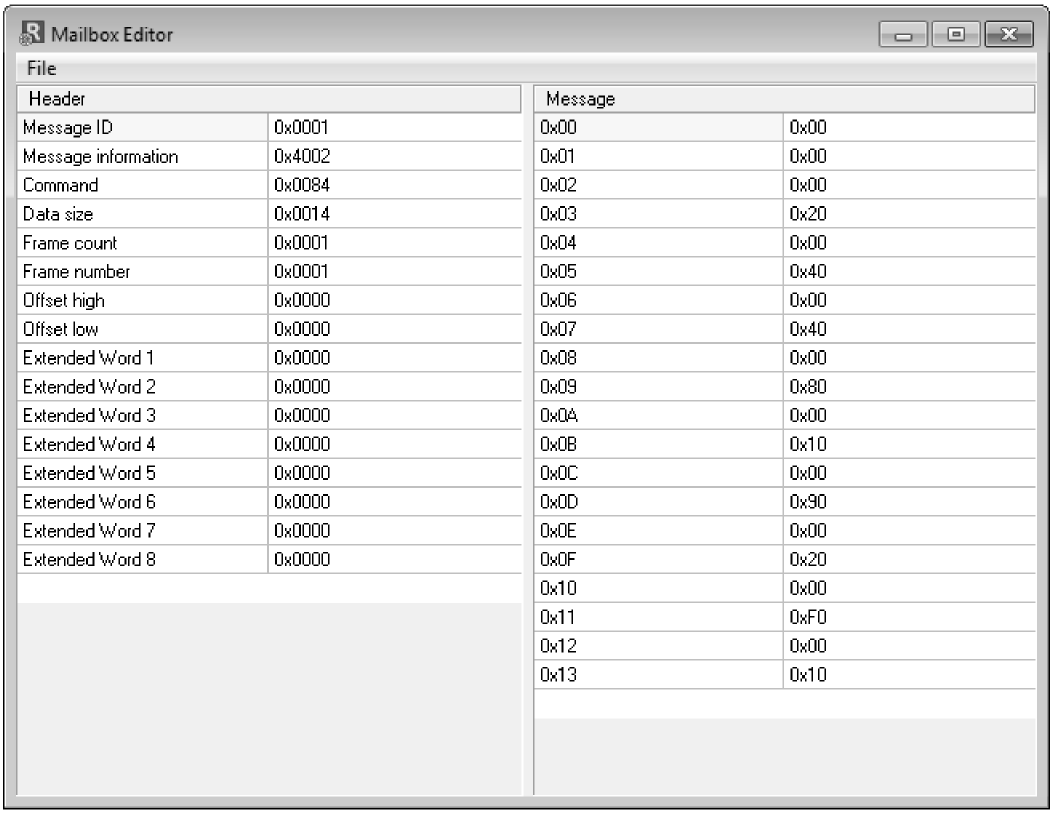
5. To save the new mailbox, select 'Apply changes' in the 'File'-menu.

[1] The offset is specified from the start of the parameter data, not from the physical memory location in the Anybus Communicator.

Resulting Attribute Mapping



Mailbox Editor Screenshot



Header	
Message ID	0x0001
Message information	0x4002
Command	0x0084
Data size	0x0014
Frame count	0x0001
Frame number	0x0001
Offset high	0x0000
Offset low	0x0000
Extended Word 1	0x0000
Extended Word 2	0x0000
Extended Word 3	0x0000
Extended Word 4	0x0000
Extended Word 5	0x0000
Extended Word 6	0x0000
Extended Word 7	0x0000
Extended Word 8	0x0000

Message	
0x00	0x00
0x01	0x00
0x02	0x00
0x03	0x20
0x04	0x00
0x05	0x40
0x06	0x00
0x07	0x40
0x08	0x00
0x09	0x80
0x0A	0x00
0x0B	0x10
0x0C	0x00
0x0D	0x90
0x0E	0x00
0x0F	0x20
0x10	0x00
0x11	0xF0
0x12	0x00
0x13	0x10

A.4 Mapping Output Parameter Data to EtherNet/IP

Example

Mapping output data is similar to mapping input data; in the following example, a total of 144 bytes of data will be mapped to the Parameter Data Output Mapping Object. The data is made up of 4 separate blocks, each associated with a a particular instance attribute.

To achieve this, perform the following steps:

1. Add a new mailbox message to the configuration (see "Add a Mailbox Message" on page 119).

2. Change the 'Command'-value in the mailbox header to 0085h.

3. Adjust the 'Data Size'-value in the mailbox header (left column). In this example, the size shall be set to 16 (0010h), since each mapped attribute occupies 4 bytes of mailbox data.

4. Specify the mapping locations for the attributes in the mailbox data section. As mentioned above, each mapping entry needs 4 bytes; two bytes specifying the offset[1] of the data block, followed by two bytes which specify the length of the data block. Note that these values must be entered in big endian (Motorola) format.

In this example, this gives us the following mailbox data:

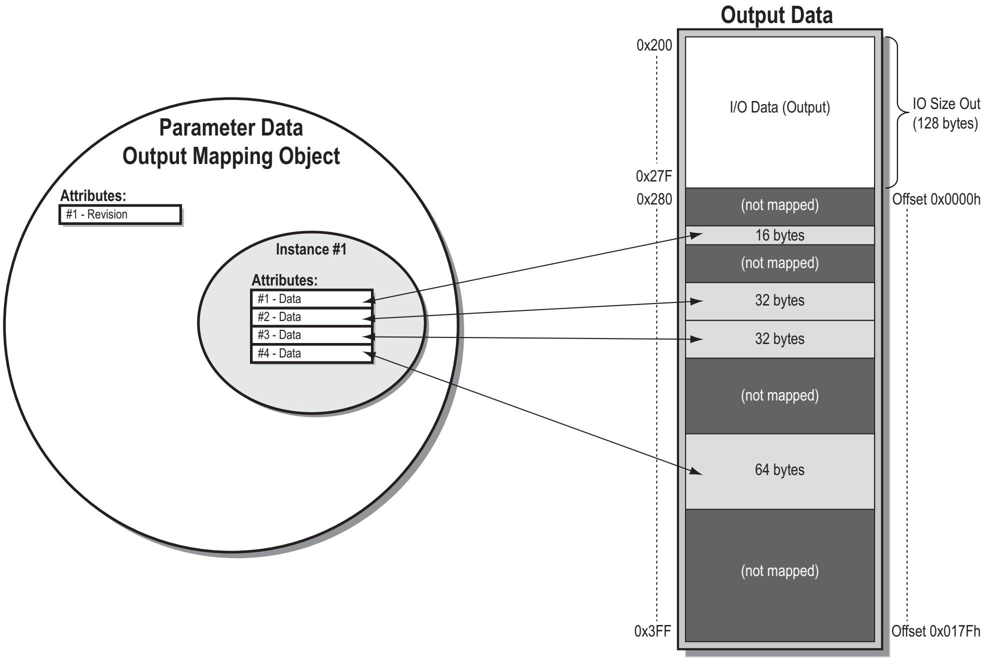
Mailbox Data		Attribute no.	Comments
Location	Data		
0x00	0x00	1	Offset = 0020h
0x01	0x20		
0x02	0x00		Size = 16 bytes
0x03	0x10		
0x04	0x00	2	Offset = 0050h
0x05	0x50		
0x06	0x00		Size = 32 bytes
0x07	0x20		
0x08	0x00	3	Offset = 0070h
0x09	0x70		
0x0A	0x00		Size = 32 bytes
0x0B	0x20		
0x0C	0x00	4	Offset = 00D0h
0x0D	0xD0		
0x0E	0x00		Size = 64 bytes
0x0F	0x40		

As shown in the table above, the attributes are numbered in the order they are mapped, i.e. it is possible to rearrange the attribute numbering by physically changing the mapping order in the mailbox data.

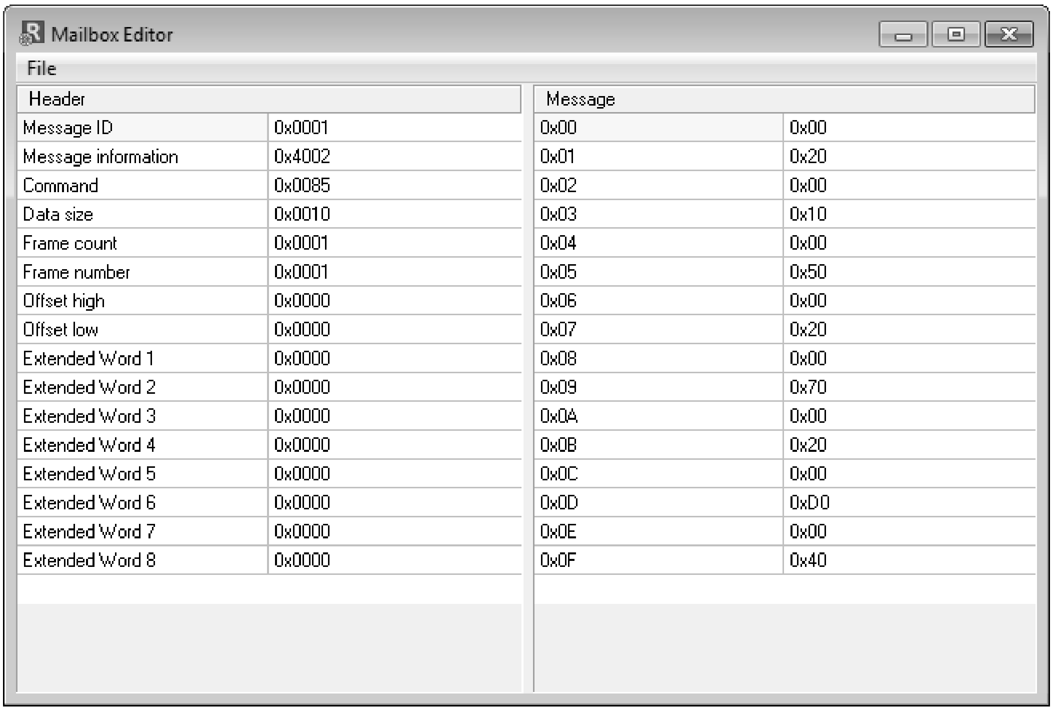
5. To save the new mailbox, select 'Apply changes' in the 'File'-menu.

[1] The offset is specified from the start of the parameter data, not from the physical memory location in the Anybus Communicator.

Resulting Attribute Mapping



Mailbox Editor Screenshot



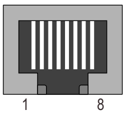
Header	
Message ID	0x0001
Message information	0x4002
Command	0x0085
Data size	0x0010
Frame count	0x0001
Frame number	0x0001
Offset high	0x0000
Offset low	0x0000
Extended Word 1	0x0000
Extended Word 2	0x0000
Extended Word 3	0x0000
Extended Word 4	0x0000
Extended Word 5	0x0000
Extended Word 6	0x0000
Extended Word 7	0x0000
Extended Word 8	0x0000

Message	
0x00	0x00
0x01	0x20
0x02	0x00
0x03	0x10
0x04	0x00
0x05	0x50
0x06	0x00
0x07	0x20
0x08	0x00
0x09	0x70
0x0A	0x00
0x0B	0x20
0x0C	0x00
0x0D	0xD0
0x0E	0x00
0x0F	0x40

B. Connector Pin Assignments

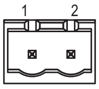
B.1 Ethernet Connector

Pin	Signal
Housing	Cable Shield
1	TD+
2	TD-
3	RD+
4	Termination
5	Termination
6	RD-
7	Termination
8	Termination



B.2 Power Connector

Pin	Description
1	+24 VDC
2	GND

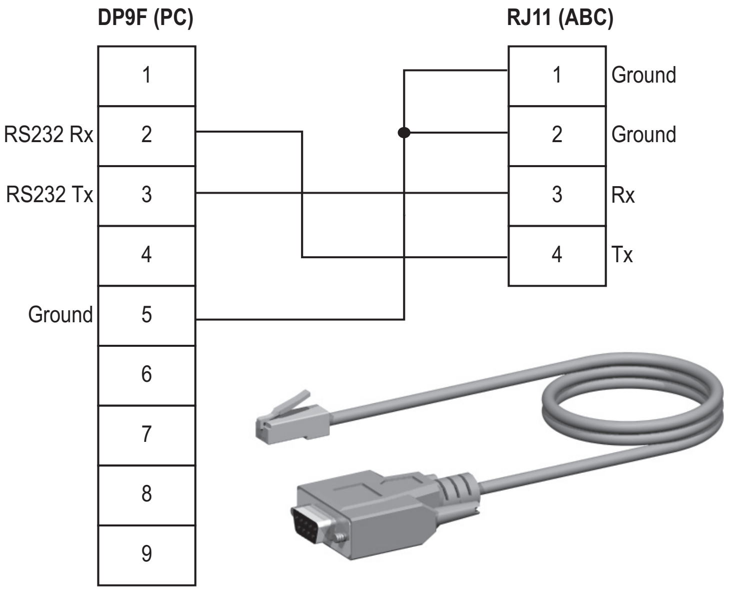


Notes:

- Use 60/75 or 75 °C copper (Cu) wire only.
- Minimum terminal tightening torque: 5–7 lb-in (0.5–0.8 Nm).

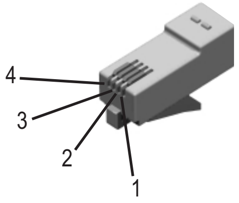
B.3 PC Connector

Configuration Cable Wiring



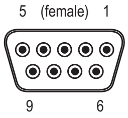
RJ11 (4P4C modular)[1] : ABC

Pin	Description
1	Signal ground
2	
3	RS232 Rx (Input)
4	RS232 Tx (Output)



DB9F : PC

Pin	Description
1	-
2	RS232 Rx (Input)
3	RS232 Tx (Output)
4	-
5	Signal Ground
6 - 9	-



[1] The RJ11 (4P4C modular) is sometimes referred to as an RJ9.

B.4 Sub-network Interface

B.4.1 General Information

The sub-network interface provides for RS232, RS422 and RS485 communications. Depending on the configuration specified in the Anybus Configuration Manager, different signals are activated in the sub-network connector.

B.4.2 Bias Resistors (RS485 Only)

When idle, RS485 enters an indeterminate state, which may cause the serial receivers to pick up noise from the serial lines and interpret this as data. To prevent this, the serial lines should be forced into a known state using pull-up and pull-down resistors, commonly known as bias resistors.

The bias resistors form a voltage divider, forcing the voltage between the differential pair to be higher than the threshold for the serial receivers, typically >200 mV.

Note that bias resistors shall only be installed on one node; installing bias resistors on several nodes may compromise the signal quality on the network and cause transmission problems.

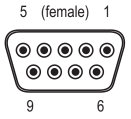
B.4.3 Termination (RS485 & RS422 Only)

To avoid reflections on the serial lines, it is important to properly terminate the sub-network by placing termination resistors between the serial receivers near the end nodes.

The resistor value should ideally match the characteristic impedance of the cable, typically 100–120 Ω.

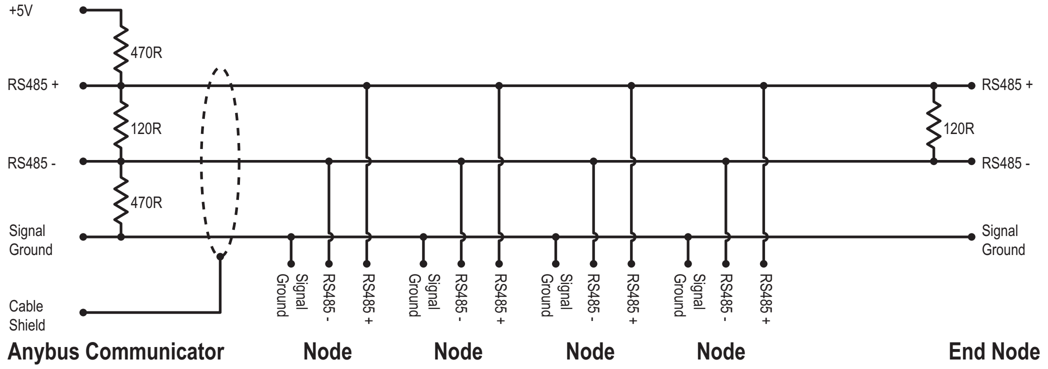
B.4.4 Connector Pinout (DB9F)

Pin	Description	RS232	RS422	RS485
1	+5 V Output(100 mA max)	✓	✓	✓
2	RS232 Rx	✓		
3	RS232 Tx	✓		
4	(reserved)			
5	Signal Ground[a]	✓	✓	✓
6	RS422 Rx +		✓	
7	RS422 Rx -		✓	
8	RS485 + / RS422 Tx+		✓	✓
9	RS485 - / RS422 Tx-		✓	✓
(housing)	Cable Shield	✓	✓	✓

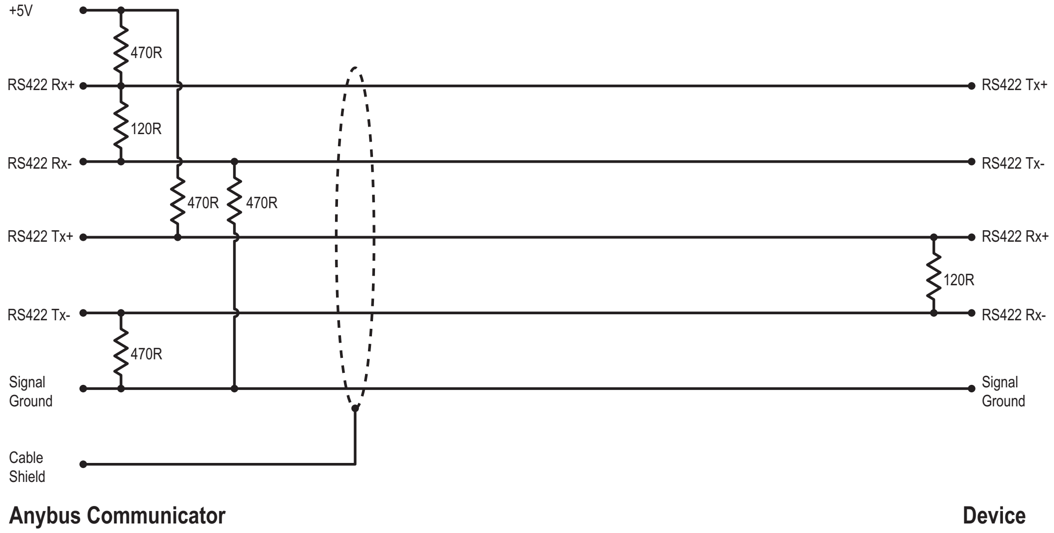


[a] Connecting this signal directly to Protective Earth (PE) of other nodes may, in case of grounding loops etc., cause damage to the on-board serial transceivers. It is therefore generally recommended to connect it only to Signal Ground (if available) of other nodes.

B.4.5 Typical Connection (RS485)

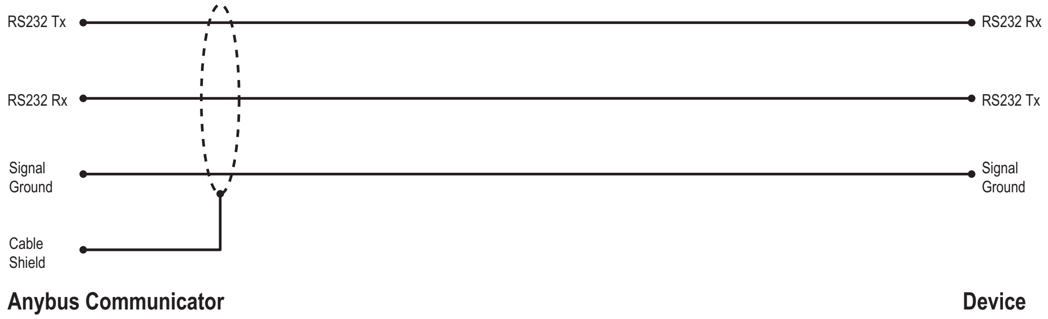


B.4.6 Typical Connection (RS422 & 4-Wire RS485)



Note: Bias resistors are normally not needed on RS422, but may be required when using 4-wire RS485.

B.4.7 Typical Connection (RS232)



C. Technical Specification

C.1 Mechanical Properties

Housing

Plastic housing with snap-on connection to DIN-rail, protection class IP20.

Dimensions (L x W x H)

120 mm x 75 mm x 27 mm (4.72” x 2.95” x 1.06”)

C.2 Electrical Characteristics

Power Supply

Power: 24 VDC ± 10%

Power Consumption

Maximum power consumption is 280 mA on 24 VDC. Typically around 100 mA.

C.3 Environmental Characteristics

Relative Humidity

The product is designed for a relative humidity of 0 to 95 % non-condensing.

Temperature

Operating:	0 °C to +55 °C
Non-operating:	-25 °C to +85 °C

C.4 Regulatory Compliance

EMC Compliance (CE)

This product is in accordance with the EMC directive 89/336/EEC, with amendments 92/31/EEC and 93/68/EEC through conformance with the following standards:

- **EN 50082-2 (1993)**

  EN 55011 (1990) Class A

- **EN 61000-6-2 (1999)**

  EN 61000-4-3 (1996) 10 V/m

  EN 61000-4-6 (1996) 10 V/m (all ports)

  EN 61000-4-2 (1995) ±8 kV air discharge, ±4 kV contact discharge

  EN 61000-4-4 (1995) ±2 kV power port, ±1 kV other ports

  EN 61000-4-5 (1995) ±0.5 kV power ports (DM/CM), ±1 kV signal ports

UL/c-UL Compliance

C UL US LISTED

IND: CONT. EQ.
FOR HAZ LOC.
CL I, DIV 2
GP A,B,C,D
TEMP
CODE
E203225

WARNING - EXPLOSION HAZARD - SUBSTITUTION OF ANY COMPONENTS MAY IMPAIR SUITABILITY FOR CLASS I, DIVISION 2.

WARNING - EXPLOSION HAZARD - WHEN IN HAZARDOUS LOCATIONS, TURN OFF POWER BEFORE REPLACING OR WIRING MODULES.

WARNING - EXPLOSION HAZARD - DO NOT DISCONNECT EQUIPMENT UNLESS POWER HAS BEEN SWITCHED OFF OR THE AREA IS KNOWN TO BE NON-HAZARDOUS.

ATTENTION – RISQUE D’EXPLOSION – LE REMPLACEMENT DE TOUT COMPOSANTS INVALIDE LA CERTIFICATION CLASS I, DIVISION 2.

ATTENTION – RISQUE D’EXPLOSION – EN ZONE EXPLOSIVE, VEUILLEZ COUPER L’ALIMENTATION ÉLECTRIQUE AVANT LE REMPLACEMENT OU LE RACCORDEMENT DES MODULES.

ATTENTION – RISQUE D’EXPLOSION – NE PAS DÉCONNECTER L’ÉQUIPEMENT TANT QUE L’ALIMENTATION EST TOUJOURS PRÉSENTE OU QUE LE PRODUIT EST TOUJOURS EN ZONE EXPLOSIVE ACTIVE.

Additional installation and operating instructions

- Max Ambient Temperature: 55 °C (for Hazloc environments)
- Field wiring terminal markings (wire type (Cu only, 14–30 AWG)).
- Use 60/75 or 75 °C copper (Cu) wire only.
- Terminal tightening torque must be 5–7 lb-in (0.5–0.8 Nm).
- Use in overvoltage category 1 pollution degree 2 environment.
- Installed in an enclosure considered representative of the intended use.
- Secondary circuit intended to be supplied from an isolating source and protected by overcurrent protective devices installed in the field sized per the following:

Control circuit wire size		Maximum protective device rating
AWG	**mm²**	**Amperes**
22	0.32	3
20	0.52	5
18	0.82	7
16	1.3	10
14	2.1	20
12	3.3	25

Galvanic isolation on sub-network interface

- **EN 60950-1 (2001)**

  Pollution Degree 2

  Material Group IIIb

  250 V_{RMS} or 250 VDC working voltage

  500 V secondary circuit transient rating

CIP Product Compliance

ODVA™
CONFORMANT

D. Troubleshooting

Problem	Solution
Problem during configuration Upload / Download. The Config Line "LED" turns red in ACM.	• Serial communication failed. Try again
The serial port seems to be available, but it is not possible to connect to the gateway	• The serial port may be in use by another application. Exit ACM and close all other applications including the ones in the system tray. Try again • Select another serial port Try again
Poor performance	• Right click "sub-network" in the Navigation window and select "sub-network Status" to see status / diagnostic information about the sub-network. If the gateway reports very many retransmissions, check your cabling and/or try a lower baud rate setting for the sub-network (if possible). • Is the Subnet Monitor in ACM active? The sub-network monitor has a negative influence on the overall performance of the gateway, and should only be used when necessary. • Is the Node Monitor in ACM active? The node monitor has a negative influence on the overall performance of the gateway, and should only be used when necessary.
No sub-network functionality	• Use the "Data logger"-functionality to record the serial data communication on the sub-network. • If no data is being transmitted, check the configuration in ACM. • If no data is received, check the sub-network cables. Also verify that the transmitted data is correct.

E. ASCII Table

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	NUL 0	SOH 1	STX 2	ETX 3	EOT 4	ENQ 5	ACK 6	BEL 7	BS 8	HT 9	LF 10	VT 11	FF 12	CR 13	SO 14	SI 15
1x	DLE 16	DC1 17	DC2 18	DC3 19	DC4 20	NAK 21	SYN 22	ETB 23	CAN 24	EM 25	SUB 26	ESC 27	FS 28	GS 29	RS 30	US 31
2x	(sp) 32	! 33	" 34	# 35	$ 36	% 37	& 38	' 39	(40	) 41	* 42	+ 43	, 44	- 45	. 46	/ 47
3x	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55	8 56	9 57	: 58	; 59	< 60	= 61	> 62	? 63
4x	@ 64	A 65	B 66	C 67	D 68	E 69	F 70	G 71	H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79
5x	P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87	X 88	Y 89	Z 90	[91	\ 92	] 93	^ 94	_ 95
6x	` 96	a 97	b 98	c 99	d 100	e 101	f 102	g 103	h 104	i 105	j 106	k 107	l 108	m 109	n 110	o 111
7x	p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119	x 120	y 121	z 122	{ 123	| 124	} 125	~ 126	DEL 127

F. Copyright Notices

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