



# User Manual

## Anybus<sup>®</sup> Communicator<sup>™</sup>

### for PROFINET

Doc. Id. HMSI-27-309  
Rev. 3.11

---

# 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. All other trademarks are the property of their respective holders.

|                  |   |
|------------------|---|
| <b>Warning:</b>  | 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.  |
| <b>ESD Note:</b> | 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 PROFINET User Manual  
Copyright© HMS Industrial Networks AB  
Doc: HMSI-27-309

# Table of Contents

|                  |  |    |
|------------------|--|----|
| <b>Preface</b>   | <b>About This Document</b>                                   |    |
|                  | Related Documents .....                                      | 8  |
|                  | Document History .....                                       | 8  |
|                  | Conventions & Terminology .....                              | 9  |
|                  | <i>Glossary</i> .....  | 9  |
|                  | Support .....  | 9  |
| <b>Chapter 1</b> | <b>About the Anybus Communicator for PROFINET</b>            |    |
|                  | General .....  | 10 |
|                  | External View .....  | 11 |
|                  | Status LEDs .....  | 12 |
|                  | Hardware Installation .....                                  | 13 |
|                  | Software Installation .....                                  | 14 |
|                  | <i>Anybus Configuration Manager</i> .....                    | 14 |
| <b>Chapter 2</b> | <b>Basic Operation</b>                                       |    |
|                  | General .....  | 15 |
|                  | Data Exchange Model .....                                    | 16 |
|                  | <i>Memory Map</i> .....                                      | 16 |
|                  | <i>Data Exchange Example</i> .....                           | 17 |
|                  | Subnetwork Protocol .....                                    | 18 |
|                  | <i>Protocol Modes</i> .....                                  | 18 |
|                  | <i>Protocol Building Blocks</i> .....                        | 18 |
|                  | <i>Master Mode</i> .....                                     | 19 |
|                  | <i>Generic Data Mode</i> .....                               | 20 |
|                  | <i>DF1 Master Mode</i> .....                                 | 20 |
|                  | PROFINET IO .....  | 21 |
|                  | <i>General</i> .....   | 21 |
|                  | <i>I/O Configuration</i> .....                               | 21 |
|                  | <i>GSDML File</i> .....                                      | 21 |
|                  | <i>Data Representation (IO Data &amp; Record Data)</i> ..... | 22 |
|                  | Modbus-TCP (Read-Only) .....                                 | 23 |
|                  | <i>General</i> .....   | 23 |
|                  | <i>Data Representation (Modbus-TCP Register Map)</i> .....   | 23 |
|                  | <i>Supported Exception codes</i> .....                       | 23 |
| <b>Chapter 3</b> | <b>File System</b>   |    |
|                  | General .....  | 24 |
|                  | File System Overview .....                                   | 25 |
|                  | System Files .....   | 25 |

|                   |  |    |
|-------------------|--|----|
| <b>Chapter 4</b>  | <b>Basic Network Configuration</b>               |    |
|                   | General Information .....                        | 26 |
|                   | Ethernet Configuration File (“ethcfg.cfg”) ..... | 27 |
|                   | <i>General</i> .....                             | 27 |
|                   | PROFINET Settings .....                          | 28 |
|                   | IP Access Control .....                          | 28 |
|                   | Anybus IPconfig (HICP) .....                     | 29 |
| <b>Chapter 5</b>  | <b>FTP Server</b>                                |    |
|                   | General.....                                     | 30 |
|                   | FTP Connection Example (Windows Explorer).....   | 31 |
| <b>Chapter 6</b>  | <b>Web Server</b>                                |    |
|                   | General.....                                     | 32 |
|                   | Authorization .....                              | 33 |
|                   | Content Types.....                               | 34 |
| <b>Chapter 7</b>  | <b>Server Side Include (SSI)</b>                 |    |
|                   | General.....                                     | 35 |
|                   | Functions .....                                  | 36 |
|                   | Changing SSI output.....                         | 45 |
|                   | <i>SSI Output String File</i> .....              | 45 |
|                   | <i>Temporary SSI Output Change</i> .....         | 46 |
| <b>Chapter 8</b>  | <b>E-mail Client</b>                             |    |
|                   | General.....                                     | 47 |
|                   | E-mail Definitions.....                          | 48 |
| <b>Chapter 9</b>  | <b>Navigating ACM</b>                            |    |
|                   | Main Window.....                                 | 49 |
|                   | <i>Drop-down Menus</i> .....                     | 50 |
|                   | <i>Toolbar Icons</i> .....                       | 53 |
| <b>Chapter 10</b> | <b>Basic Settings</b>                            |    |
|                   | Fieldbus Settings.....                           | 54 |
|                   | Communicator Parameters .....                    | 55 |
|                   | Sub-network Parameters .....                     | 56 |

|                   |  |    |
|-------------------|--|----|
| <b>Chapter 11</b> | <b>Nodes</b>   |    |
|                   | General.....   | 57 |
|                   | Adding & Managing Nodes.....   | 57 |
|                   | Node Parameters.....   | 57 |
|                   | <i>Master Mode and Generic Data Mode</i> .....                       | 57 |
| <b>Chapter 12</b> | <b>Transactions</b>  |    |
|                   | General.....   | 58 |
|                   | Adding & Managing Transactions.....                                  | 59 |
|                   | Transaction Parameters (Master Mode).....                            | 60 |
|                   | <i>Parameters (Query &amp; Broadcast)</i> .....                      | 60 |
|                   | <i>Parameters (Response)</i> .....                                   | 61 |
|                   | Transaction Parameters (Generic Data Mode).....                      | 62 |
|                   | <i>Produce Transactions</i> .....                                    | 62 |
|                   | <i>Consume Transactions</i> .....                                    | 63 |
|                   | Transaction Editor.....  | 64 |
| <b>Chapter 13</b> | <b>Frame Objects</b>   |    |
|                   | General.....   | 65 |
|                   | Adding and Editing Frame Objects.....                                | 65 |
|                   | Constant Objects (Byte, Word, Dword).....                            | 66 |
|                   | Limit Objects (Byte, Word, Dword).....                               | 67 |
|                   | Data Object.....   | 68 |
|                   | Variable Data Object.....  | 68 |
|                   | Checksum Object.....   | 70 |
| <b>Chapter 14</b> | <b>Commands</b>  |    |
|                   | General.....   | 71 |
|                   | Adding & Managing Commands.....                                      | 71 |
|                   | <i>Drop-down Menu</i> .....  | 72 |
|                   | <i>Toolbar Icons</i> .....   | 72 |
|                   | The Command Editor.....  | 73 |
|                   | <i>General</i> .....   | 73 |
|                   | <i>Basic Navigation</i> .....  | 73 |
|                   | <i>Drop-down Menu</i> .....  | 74 |
|                   | <i>Editing a Command</i> .....                                       | 74 |
|                   | <i>Example: Specifying a Modbus-RTU Command in Master Mode</i> ..... | 75 |

|                   |   |    |
|-------------------|---|----|
| <b>Chapter 15</b> | <b>DF1 Protocol Mode</b>                                      |    |
|                   | General.....  | 76 |
|                   | Communicator Parameters .....                                 | 77 |
|                   | Sub-network Parameters .....                                  | 78 |
|                   | Node Parameters .....   | 79 |
|                   | Services.....   | 79 |
|                   | <i>Available Services</i> .....                               | 80 |
|                   | Integrity Check .....   | 81 |
|                   | Read Diagnostics .....  | 81 |
|                   | Read Data .....   | 82 |
|                   | Write Data .....  | 82 |
| <b>Chapter 16</b> | <b>Sub-network Monitor</b>                                    |    |
|                   | General.....  | 83 |
|                   | Operation.....  | 83 |
| <b>Chapter 17</b> | <b>Node Monitor</b>   |    |
|                   | General.....  | 84 |
|                   | Navigating the Node Monitor.....                              | 85 |
|                   | <i>Drop-down Menu</i> .....                                   | 86 |
|                   | <i>Toolbar Icons</i> .....                                    | 87 |
| <b>Chapter 18</b> | <b>Data Logger</b>  |    |
|                   | General.....  | 88 |
|                   | Operation.....  | 88 |
|                   | Configuration .....   | 89 |
| <b>Chapter 19</b> | <b>Configuration Wizards</b>                                  |    |
|                   | General.....  | 90 |
|                   | Selecting a Wizard Profile .....                              | 90 |
|                   | Wizard - Modbus RTU Master .....                              | 91 |
| <b>Chapter 20</b> | <b>Control and Status Registers</b>                           |    |
|                   | General.....  | 92 |
|                   | <i>Handshaking Procedure</i> .....                            | 92 |
|                   | <i>Data Consistency</i> .....                                 | 93 |
|                   | Status Register Contents (Gateway to Control System).....     | 94 |
|                   | <i>General Information</i> .....                              | 94 |
|                   | <i>Status Codes in Master Mode and DF1 Master Mode</i> .....  | 94 |
|                   | <i>Status Code in Generic Data Mode</i> .....                 | 95 |
|                   | Control Register Contents (Control System to Gateway).....    | 96 |
|                   | <i>General Information</i> .....                              | 96 |
|                   | <i>Control Codes in Master Mode and DF1 Master Mode</i> ..... | 96 |
|                   | <i>Control Codes in Generic Data Mode</i> .....               | 96 |

|                   |  |     |
|-------------------|--|-----|
| <b>Chapter 21</b> | <b>Advanced Fieldbus Configuration</b>                     |     |
|                   | General.....   | 97  |
|                   | Mailbox Editor.....  | 97  |
| <b>Appendix A</b> | <b>Connector Pin Assignments</b>                           |     |
|                   | PROFINET Connector (Ethernet) .....                        | 98  |
|                   | Power Connector .....                                      | 98  |
|                   | PC Connector .....   | 99  |
|                   | Subnetwork Interface .....                                 | 100 |
|                   | <i>General Information</i> .....                           | 100 |
|                   | <i>Bias Resistors (RS485 Only)</i> .....                   | 100 |
|                   | <i>Termination (RS485 &amp; RS422 Only)</i> .....          | 100 |
|                   | <i>Connector Pinout (DB9F)</i> .....                       | 100 |
|                   | <i>Typical Connection (RS485)</i> .....                    | 101 |
|                   | <i>Typical Connection (RS422 &amp; 4-Wire RS485)</i> ..... | 101 |
|                   | <i>Typical Connection (RS232)</i> .....                    | 101 |
| <b>Appendix B</b> | <b>Technical Specification</b>                             |     |
|                   | Mechanical Properties.....                                 | 102 |
|                   | Electrical Characteristics .....                           | 102 |
|                   | Environmental Characteristics .....                        | 102 |
|                   | Regulatory Compliance .....                                | 103 |
| <b>Appendix C</b> | <b>Troubleshooting</b>                                     |     |
| <b>Appendix D</b> | <b>ASCII Table</b>   |     |
| <b>Appendix E</b> | <b>Copyright Notices</b>                                   |     |

## P. About This Document

For more information, documentation etc., please visit the HMS website [www.anybus.com](http://www.anybus.com).

### P.1 Related Documents

| Document name  | Author        |
|--|---------------|
| Anybus Communicator - PRT Installation Sheet                               | HMS           |
| DF1 Protocol and Command Set - Reference Manual, 1770-6.5.16, October 1996 | Allen-Bradley |

### P.2 Document History

#### Summary of Recent Changes (3.01... 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                                    | 31, 32   |
| Added description for Consume/Response to "Object Delimiter" parameter            | 39       |
| Changed "Maximum Data Length" limit   | 39       |
| Removed obsolete "Start Bits" parameter   | 48       |
| Removed obsolete "ABCC ExtLink Wizard" entry                                      | 64       |
| Replaced "Sales and Support" info with link to website                            | 8        |
| Added parameters to checksum object description                                   | 40       |
| Added info about Modbus registers   | 23       |
| Minor text edits, typo corrections  | Multiple |

#### Summary of Recent Changes (3.10... 3.11)

| Revision | Change                | Page(s) |
|----------|-----------------------|---------|
| 3.11     | Added compliance info | 79      |

#### Revision List

| Revision | Date          | Author | Chapter | Description   |
|----------|---------------|--------|---------|---|
| 1.00     | 2005-06-07    | PeP    | All     | 1st release   |
| 2.00     | 2006-03-27    | PeP    | All     | 2nd major release   |
| 2.01     | 2006-12-22    | PeP    | -       | Misc. minor corrections & updates                                   |
| 2.02     | 2008-10-14    | HeS    | -       | Misc. minor corrections & updates                                   |
| 2.03     | 2009-04-24    | KeL    | All     | Misc. minor corrections and updates                                 |
| 3.00     | 2011-01-19    | KaD    | All     | Misc. minor corrections, new template and DF1 functionality         |
| 3.01     | 2011-09-30    | KaD    | All     | Misc corrections and updates, new Anybus Configuration Manager name |
| 3.10     | February 2015 | ThN    | All     | Misc. corrections and updates, new Doc ID                           |
| 3.11     | March 2015    | ThN    | B       | Added compliance info   |



## 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 <sup>®</sup> Communicator <sup>™</sup>   |
| PRT                     | PROFINET-IO   |
| ACM                     | Anybus Configuration Manager  |
| Broadcaster             | A protocol specific node in the subnetwork scan- that hold transactions destined to all nodes.                                      |
| Command                 | A protocol specific transaction.  |
| Configuration           | List of configured nodes with transactions on the subnetwork.   |
| Fieldbus                | The network to which the communicator is connected.   |
| Fieldbus Control System | Fieldbus master   |
| Frame                   | Higher level series of bytes forming a complete telegram on the subnetwork  |
| Monitor                 | A tool for debugging the ABC and the network connections.   |
| Node                    | A device in the configuration that defines the communication with a node on the subnetwork  |
| Scan list               | List of configured slaves with transactions on the subnetwork.  |
| Subnetwork              | The network that logically is located on a subsidiary level with respect to the fieldbus and to which the ABC acts as a gateway.    |
| Transaction             | A generic building block that is used in the subnetwork scan-list and defines the data that is sent and received on the subnetwork. |
| IO Controller           | PROFINET device which acts as a client for several IO devices, usually a PLC. (Comparable to a PROFIBUS-DP class 1 master).         |
| IO Supervisor           | PROFINET programming device with commissioning and diagnostic functions (Comparable to a PROFIBUS-DP class 2 master).               |
| IO Device               | Field device assigned to an IO Controller. In this case the ABC.  |
| Module (or I/O module)  | Hardware or logical component of a PROFINET network device.   |
| Higher Level Network    | In this case, PROFINET  |
| Network                 |   |
| Fieldbus                |   |

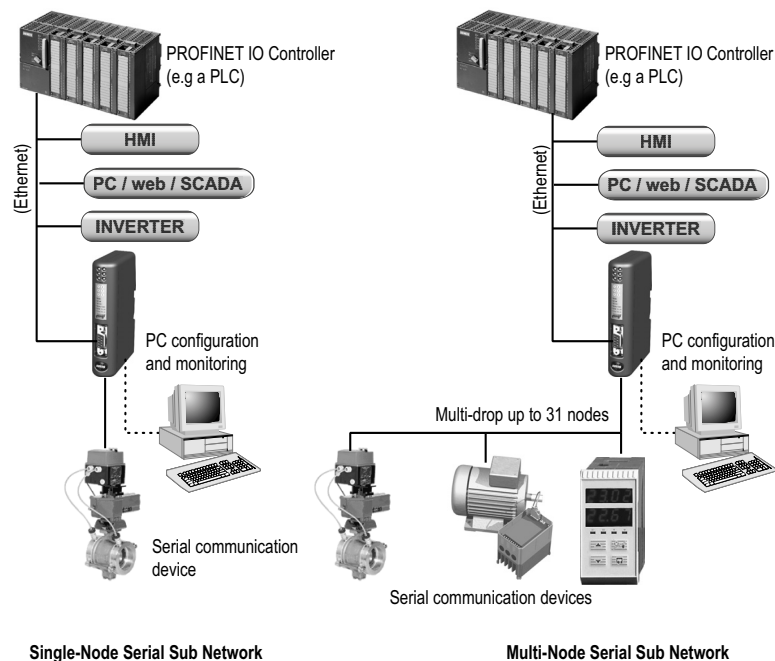
## P.4 Support

For general contact information and support, please refer to the contact and support pages at [www.anybus.com](http://www.anybus.com)

# 1. About the Anybus Communicator for PROFINET

## 1.1 General

The Anybus Communicator module for PROFINET acts as a gateway between virtually any serial application protocol and a PROFINET IO-based network. Integration of industrial devices is enabled without loss of functionality, control and reliability, both when retro-fitting to existing equipment as well as when setting up new installations.



### Subnetwork

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.

- PROFINET IO
- Modbus-TCP server (read only)
- Security framework with per user access rights and IP access control
- Server Side Include (SSI) functionality
- Web server and Email client capability
- Easy file management via FTP
- 10/100 Mbit/s, twisted pair

## 1.2 External View

For wiring and pin assignments, see “Connector Pin Assignments” on page 98.

### A: PROFINET Connector (Ethernet)

This connector is used to connect the module to the network.

See also...

- “PROFINET Connector (Ethernet)” on page 98

### B: Status LEDs

See also...

- “Status LEDs” on page 12

### C: PC-connector

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

See also...

- “PC Connector” on page 99

### D: Subnetwork Connector

This connector is used to connect the module to the serial subnetwork.

See also...

- “Subnetwork Interface” on page 100

### E: Power Connector

This connector is used to apply power to the module.

See also...

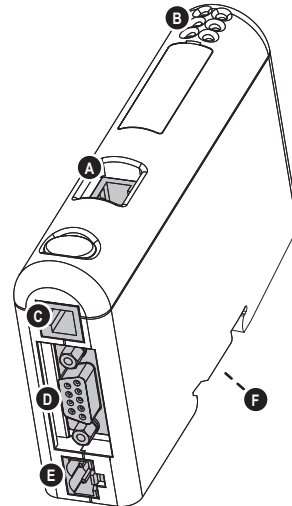
- “Power Connector” on page 98
- “Technical Specification” on page 128

### F: DIN-rail Connector

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

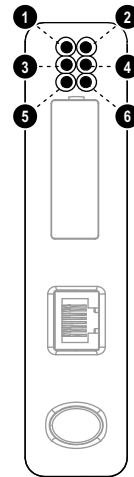
See also...

- “Hardware Installation” on page 13



## 1.3 Status LEDs

| #                              | State                 | Status  |
|--------------------------------|-----------------------|---|
| 1 - Comm. Status               | Off                   | Off line<br>- No connection with IO Controller  |
|                                | Green                 | On line, Run<br>- Connection with IO Controller established<br>- IO Controller is in RUN state  |
|                                | Green, flashing       | On line, STOP<br>- Connection with IO Controller established<br>- IO Controller in STOP state   |
| 2 - Module Status              | Off                   | No power or not initialized   |
|                                | Green                 | Initialized, no error   |
|                                | Green, 1 flash        | Diagnostic data available   |
|                                | Green, 2 flashes      | Blink. Used by engineering tools for identification.  |
|                                | Red, 1 flash          | Configuration Error<br>- Too many modules/submodules<br>- I/O size or Configuration mismatch  |
|                                | Red, 3 flashes        | No Station Name or no IP address assigned   |
|                                | Red, 4 flashes        | Internal error  |
| 3 - Link/Activity              | Off                   | No link or power off  |
|                                | Green                 | Link established  |
|                                | Green, flashing       | Receiving/transmitting data   |
| 4 - (not used)                 | -                     | -   |
| 5 - Subnet Status <sup>a</sup> | Off                   | Power off   |
|                                | 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                   | Power off   |
|                                | Alternating Red/Green | Invalid or missing configuration  |
|                                | Green                 | Initializing  |
|                                | Green, flashing       | Running   |
|                                | Red                   | Bootloader mode <sup>b</sup>  |
|                                | 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 the HMS support department |



a. This LED turns 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 turn red.

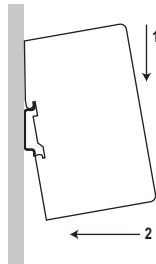
b. The gateway is in bootloader mode, and firmware must be restored in order for it to work properly. Start up the Anybus Configuration Manager and connect to the Anybus Communicator. Choose Tools/Options/Module. Click “Factory Restore” to restore firmware. See “Tools” on page 61.

## 1.4 Hardware Installation

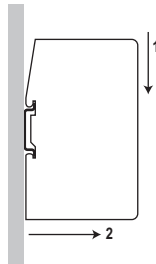
Perform the following steps when physically installing the Anybus Communicator module:

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

The DIN-rail mechanism works as follows:



To snap the module *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 module *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 module to the PROFINET (Ethernet) network
3. Connect the module to the serial subnetwork
4. Connect the module 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 attempts to detect the serial port automatically. If not successful, select the correct port manually in the "Port"-menu)
7. Configure the Anybus Communicator using the Anybus Configuration Manager and download the configuration
8. Set up the PROFINET communication in accordance with the module 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](http://www.anybus.com).
  - Unzip the archive on your computer and double-click on the installation executable.

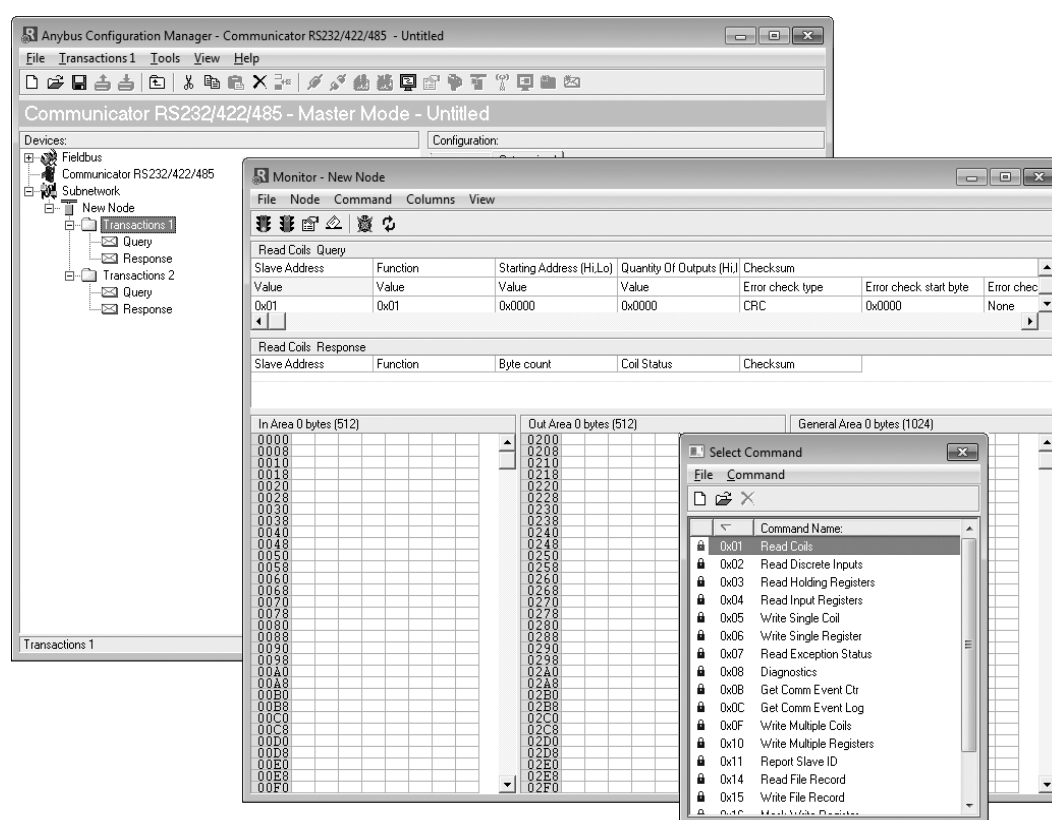
## 2. Basic Operation

### 2.1 General

The Anybus Communicator gateway is designed to exchange data between a serial sub-network and a higher level network. Unlike most other gateway devices of similar kind, it does not have a fixed protocol for the sub-network, and 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 of 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.



## 2.2 Data Exchange Model

Internally, the data exchanged on the subnetwork, and the data exchanged on the higher level network, reside in the same memory.

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

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

- **Input Data (512 bytes)**

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

(How this data is represented on the higher level network will be described later in this chapter).

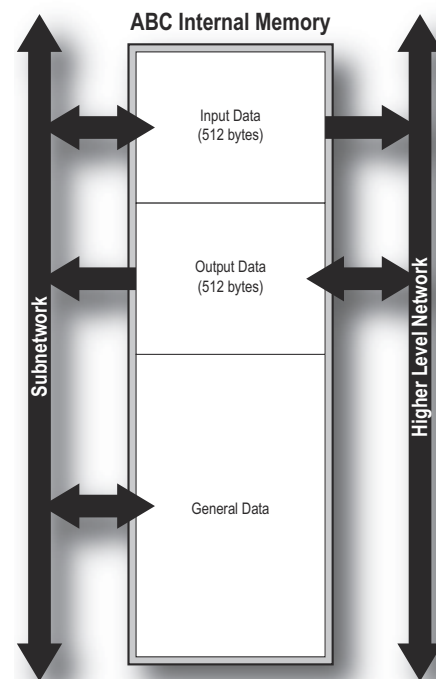
- **Output Data (512 bytes)**

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

(How this data is represented on the higher level network will be described later in this chapter).

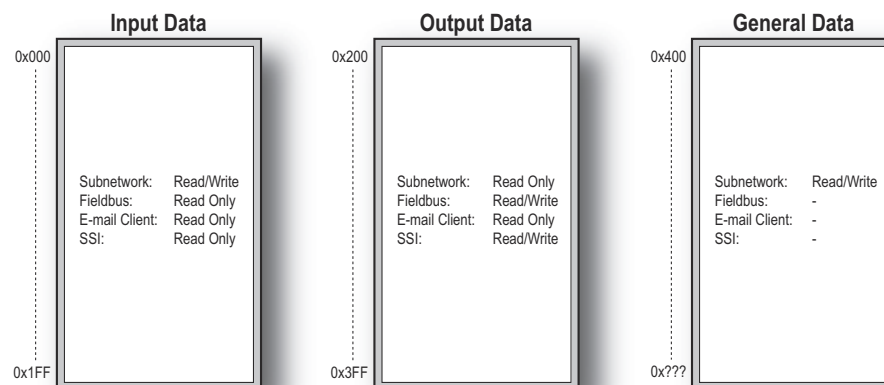
- **General Data (Up to 1024 bytes)**

This area cannot be accessed from the higher level network, but can be used for transfers between individual nodes on the subnetwork, 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 subnetwork. The Anybus Communicator can handle up to 1024 bytes of general data.



### 2.2.1 Memory Map

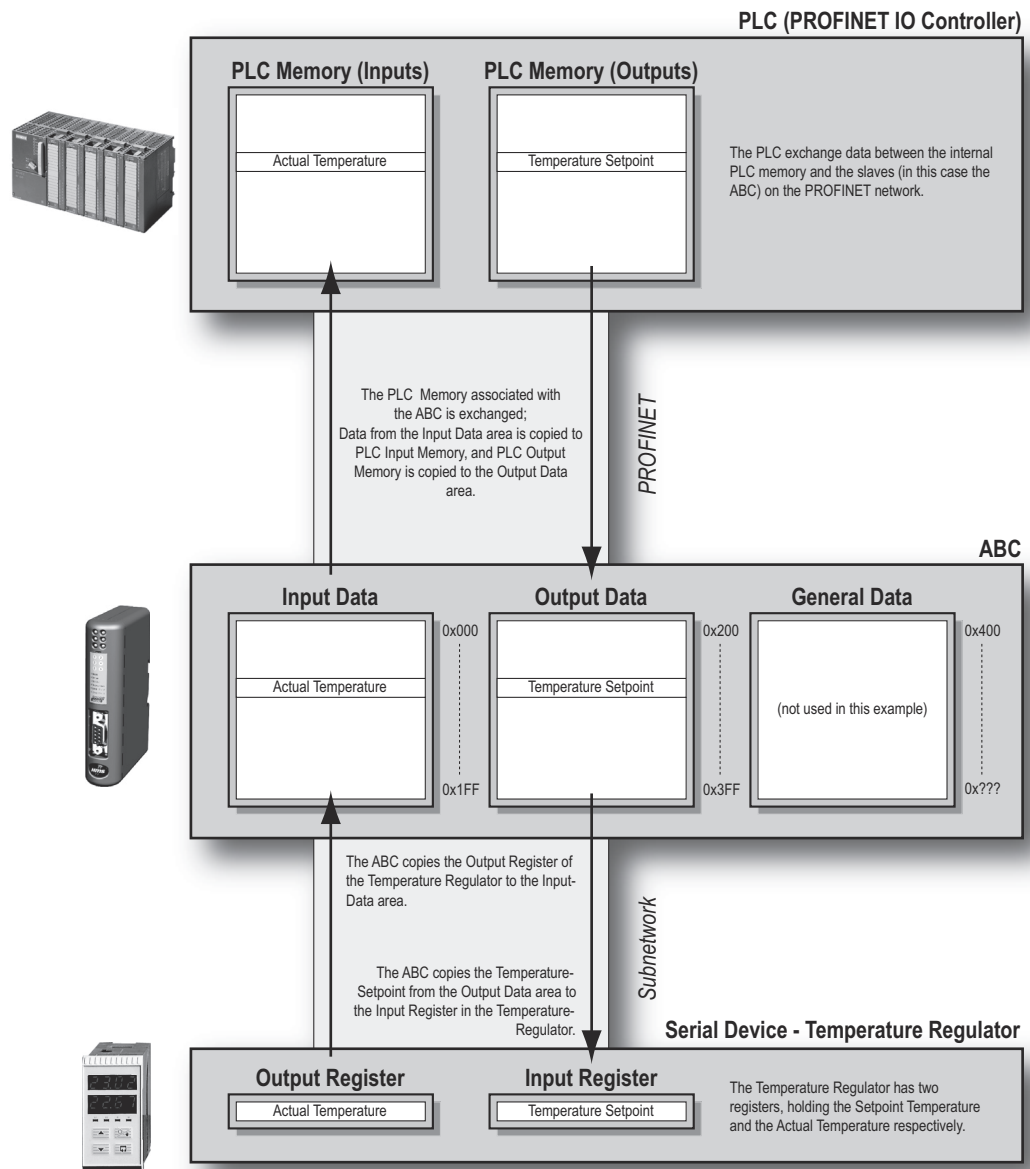
When building the subnetwork configuration using the Anybus Configuration Manager, the different areas described above are mapped to the memory locations (addresses) specified below.





## 2.2.2 Data Exchange Example

In the following example, a temperature regulator on the subnetwork exchanges information with a PLC on the higher level network, via the internal memory buffers in the module.



## 2.3 Subnetwork Protocol

### 2.3.1 Protocol Modes

The Anybus Communicator features three distinct modes of operation regarding the subnetwork communication, called “Master Mode”, “Generic Data Mode” and “DF1 Master Mode”. Note that the protocol mode only specifies the basic communication model, not the actual subnetwork protocol.

- **Master Mode**

In this mode, the module acts as a master on the subnetwork, and the serial communication takes place in a query-response fashion. The nodes on the network are not permitted to issue messages unless they have been addressed by the module first.

See also “Master Mode” on page 19.

- **Generic Data Mode**

In this mode, there is no master-slave relationship between the subnetwork nodes and the module; any node on the subnetwork, including the Anybus Communicator, may spontaneously produce or consume messages.

See also “Generic Data Mode” on page 20.

- **DF1 Master Mode**

In this mode, the module acts as a master on the subnetwork, using the DF1 protocol. The serial communication takes place in a query-response fashion.

See also “DF1 Protocol Mode” on page 76.

### 2.3.2 Protocol Building Blocks

The following building blocks are used in Anybus Configuration Manager to describe the subnetwork communication. How these blocks apply to the three protocol modes will be described later in this document.

- **Node**

A node represents a single device on the subnetwork. 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 (below). Each transaction is associated with a set of parameters controlling how and when to use it on the subnetwork.

- **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.

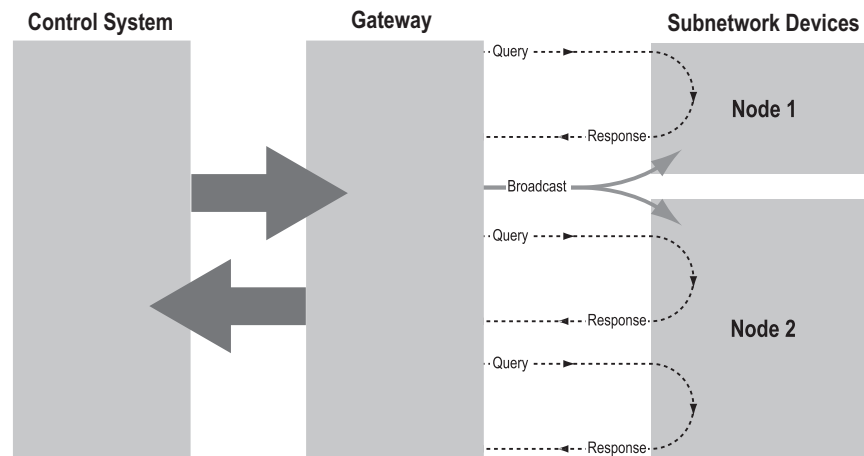
- **Frame Object**

“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 Anybus Communicator issues a query on the subnetwork, the addressed node is expected to issue a response to that query. Nodes are not permitted to issue responses spontaneously, i.e. without first receiving a query.

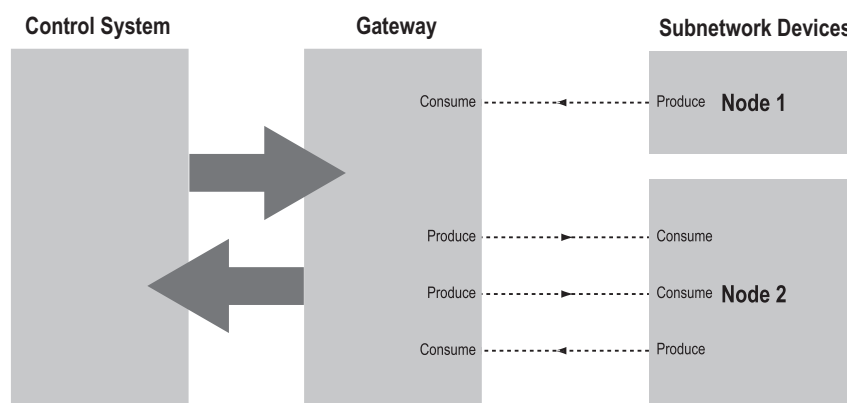
There is one exception to this rule; the broadcaster. Most protocols offer some way of broadcasting messages to all nodes on the network, without expecting them to respond to the broadcasted message. This is also reflected in the Anybus Communicator, which features a dedicated broadcaster node.



In Master Mode, Anybus Configuration Manager comes preloaded with most commonly used Modbus RTU commands, which can conveniently be reached by right-clicking on a node in the Anybus Configuration Manager and selecting “Insert New Command”. Note however that this does not in any way prevent other protocols based on the same query-response message-scheme to be implemented.

### 2.3.4 Generic Data Mode

In this mode, there is no master-slave relationship between the nodes on the subnetwork and the Anybus Communicator. Any node, including the module, may spontaneously produce or consume a message. Nodes do not have to respond to messages, nor do they have to wait for a query in order to send one.



In the figure above, the Anybus Communicator “consumes” data that is “produced” by a node on the subnetwork. 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 subnetwork to be “consumed” by a node.

### 2.3.5 DF1 Master Mode

Please refer to “DF1 Protocol Mode” on page 76.

## 2.4 PROFINET IO

### 2.4.1 General

The PROFINET IO interface provides PROFINET IO Soft Real-Time Communication. PROFINET is the open Industrial Ethernet standard for Automation from PROFIBUS International.

#### Supported Features

- Soft Real-Time (RT) communication
- Cyclic data exchange (10 ms cycle time)
- Acyclic Data exchange (Record Data Requests)
- Up to 64 slots / 1 subslot
- Up to 512 bytes of I/O in each direction
- TCP/IP Configuration via DCP (Discovery and Configuration Protocol)

### 2.4.2 I/O Configuration

PROFINET makes a distinction between fast cyclical data, a.k.a. “IO data”, and acyclical data, called “Record Data”. By default, all data in the input and output data areas are exchanged as IO data. It is however possible to specify how much data to exchange as IO data, and how much data to exchange using acyclic Record Data read/write requests.

On PROFINET, the IO data is built up by I/O modules. In the case of the ABC, the actual I/O module configuration is adopted from the I/O controller/supervisor, provided that the total I/O sizes specified by the IO controller does not exceed the sizes specified in the Anybus Configuration Manager.

For information about how the IO and Record Data relates to the input and output data areas, see “Data Representation (IO Data & Record Data)” on page 22.

### 2.4.3 GSDML File

On PROFINET, all devices are associated with a GSDML file. The GSDML file is the equivalent of the PROFIBUS GSD file, and is based on the EXtensible Markup Language (XML).

This file holds information about the device (in this case the Anybus Communicator), its features, and possible I/O configurations. The latest version of the GSDML file for the Anybus Communicator can be downloaded from the HMS website, “[www.anybus.com](http://www.anybus.com)”.

## 2.4.4 Data Representation (IO Data & Record Data)

As mentioned previously, the actual I/O configuration is determined by the IO Controller. The modules are mapped to the input and output data areas in the order of their slot number.

*Example:*

In this example, the I/O Sizes for the ABC has been set to the following values:

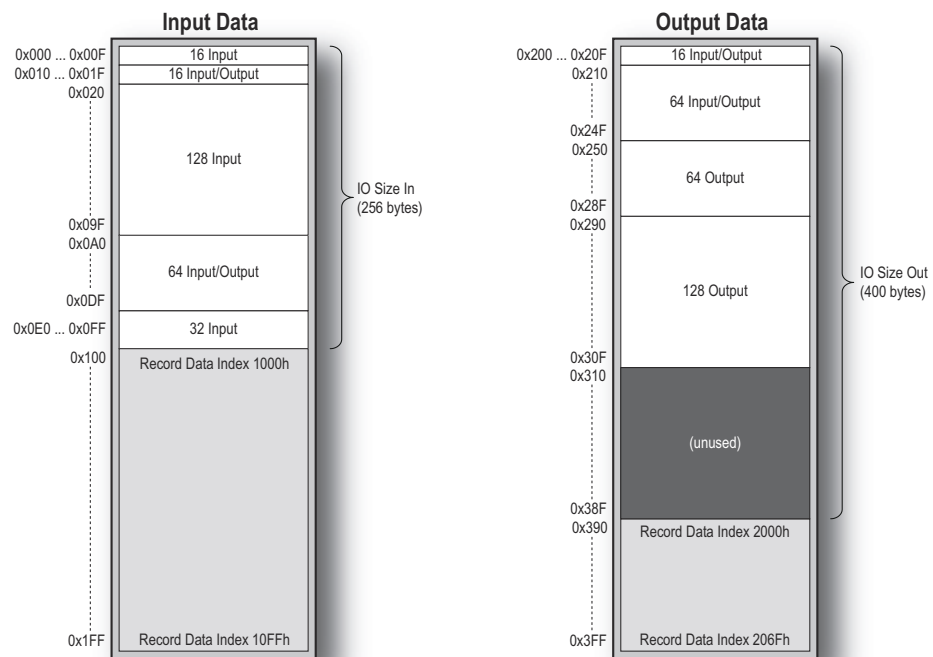
IO Size In= 256 bytes (0x0100)

IO Size Out= 400 bytes (0x0200)

The following modules are specified in the IO Controller:

| Slot | Module Size | Direction    | Notes                      |
|------|-------------|--------------|----------------------------|
| 0    | 0           | -            | (Device Access Point, DAP) |
| 1    | 16 bytes    | Input        | -                          |
| 2    | 16 bytes    | Input/Output | -                          |
| 3    | 128 bytes   | Input        | -                          |
| 4    | 64 bytes    | Input/Output | -                          |
| 5    | 32 bytes    | Input        | -                          |
| 6    | 64 bytes    | Output       | -                          |
| 7    | 128 bytes   | Output       | -                          |

Resulting memory layout:



Note the “unused” part of the output data area. The reason for this is that only 272 bytes (128+64+64+16) are actually used in the I/O module configuration, although IO Size Out is set to 400 bytes.

## 2.5 Modbus-TCP (Read-Only)

### 2.5.1 General

The Modbus-TCP protocol is an implementation of the standard Modbus protocol running on top of TCP/IP. The same function codes and addressing model are used. The built in Modbus-TCP server provides read-only access to the input and output data areas via a subset of the functions defined in the Modbus-TCP specification.

All Modbus-TCP messages are received/transmitted on TCP port no. 502. For detailed information regarding the Modbus-TCP protocol, consult the Open Modbus Specification.

### 2.5.2 Data Representation (Modbus-TCP Register Map)

The following function codes are implemented:

| Modbus Function         | Function Code | Associated with                     | No. of I/O:s or data points per command |
|-------------------------|---------------|-------------------------------------|---|
| Read Input Registers    | 4             | Input Data area<br>(0x000....0x1FF) | 1 - 125 registers                       |
| Read Multiple Registers | 3             | Output Data area<br>(0x200...0x3FF) | 1 - 125 registers                       |

The Input & Output Data areas are mapped to Modbus registers as follows:

| Register Type               | Register # | Memory Location | Area             | Comments           |
|-----------------------------|------------|-----------------|------------------|--------------------|
| Input Registers<br>(3xxxx)  | 0x0000     | 0x000...0x001   | Input Data area  | (Status Register)  |
|                             | 0x0001     | 0x002...0x003   |                  | -                  |
|                             | 0x0002     | 0x004...0x005   |                  | -                  |
|                             | 0x0003     | 0x006...0x007   |                  | -                  |
|                             | ...        | ...             |                  | -                  |
|                             | 0x00FF     | 0x1FE...0x1FF   |                  | -                  |
| Output Registers<br>(4xxxx) | 0x0000     | 0x200...0x201   | Output Data area | (Control Register) |
|                             | 0x0001     | 0x202...0x203   |                  | -                  |
|                             | 0x0002     | 0x204...0x205   |                  | -                  |
|                             | 0x0003     | 0x206...0x207   |                  | -                  |
|                             | ...        | ...             |                  | -                  |
|                             | 0x00FF     | 0x3FE...0x3FF   |                  | -                  |

**Note:** If enabled, the control and status registers occupies input register 0x0000 and output register 0x0000.

### 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  |

## 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. 2 Mb)**  
This section is intended for static files such as web files, configuration 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.

#### Important Note:

The non-volatile storage is located in FLASH memory. Each FLASH segment can only be erased approximately 100000 times due to the nature of this type of memory.

The following operations will erase one or more FLASH segments:

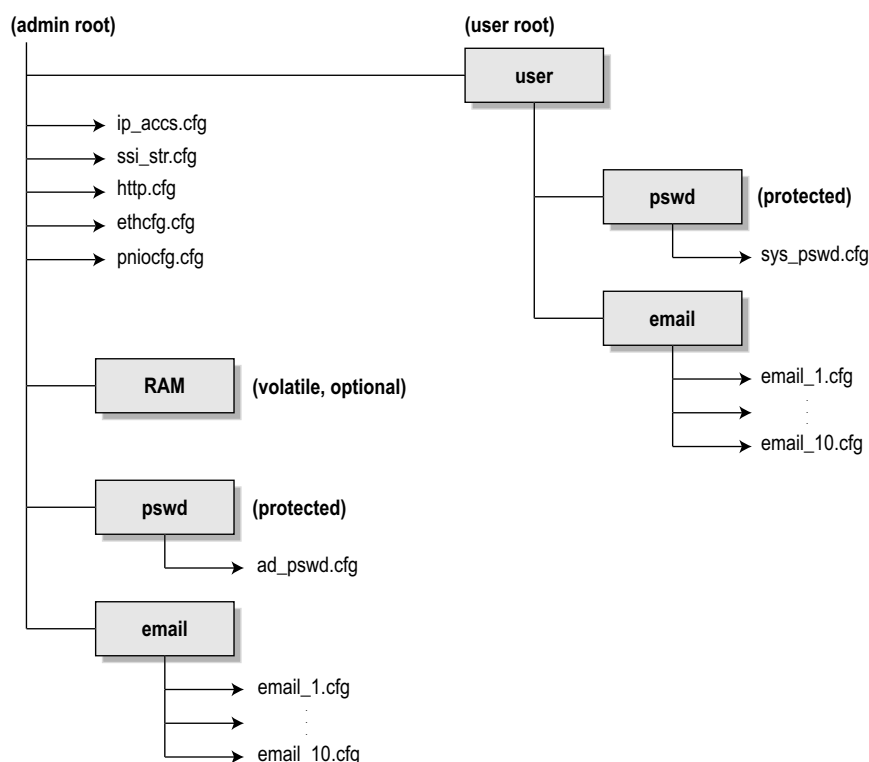
- Deleting, moving or renaming a file or directory
- Writing or appending data to an existing file
- Formatting the file system

#### Conventions

- \ (backslash) is used as a path separator
- File paths originate from the system root, and as such must begin with \
- File paths must not end with \
- Names may contain spaces but must not begin or end with one
- Names must not contain one of the following characters: \ / : \* ? " < > |
- Names cannot be longer than 48 characters (plus null termination)
- File paths 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



## 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 which can be altered using a standard text editor. Note that some of these files may also be altered by the Anybus Communicator itself, e.g. when using SSI (see “Server Side Include (SSI)” on page 35).

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 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 33.

## 4. Basic Network Configuration

### 4.1 General Information

The Anybus Communicator offers two modes of operation regarding the network settings (see below). Which mode to use is determined by the “TCP/IP Settings” parameter in Anybus Configuration Manager, see “Fieldbus Settings” on page 54.

- **TCP/IP Settings: Enabled**

When operating in this mode, 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
- Settings received from the network (i.e. via HICP or DCP) will be lost in the event of a power loss or reset.

- **TCP/IP Settings: Disabled**

When operating in this mode, the Anybus Communicator module will use the settings stored in the system file “ethcfg.cfg”. If this file is missing, the module will attempt to retrieve its settings via DHCP or HICP for 30 seconds. If no configuration has been received within this period, the module will halt and indicate an error on its status LEDs.

#### **DCP (Discovery and Basic Configuration)**

The Anybus Communicator fully supports the DCP protocol, which allows an IO Controller/Supervisor to change the TCP/ IP settings during runtime.

#### **DHCP/BootP**

The Anybus Communicator can retrieve the TCP/IP settings from a DHCP or BootP server. If no DHCP server is found, the module will fall back on its current settings (i.e. the settings currently stored in “\ethcfg.cfg”).

If no current settings are available (i.e. “ethcfg.cfg” is missing, or contains invalid settings), the module will halt and indicate an error on the onboard status LEDs. The network configuration may however still be accessed via HICP, see “Anybus IPconfig (HICP)” on page 29.

## 4.2 Ethernet Configuration File (“ethcfg.cfg”)

### 4.2.1 General

To be able to participate on the network, the Anybus Communicator needs a valid TCP/IP configuration. These settings are stored in the system file “\ethcfg.cfg”.

#### *File Format:*

|                                      |   |   |
|--------------------------------------|---|---|
| [IP address]<br>xxx.xxx.xxx.xxx      | • | <b>IP address</b>   |
| [Subnet mask]<br>xxx.xxx.xxx.xxx     |   |   |
| [Gateway address]<br>xxx.xxx.xxx.xxx | • | <b>Subnet mask</b>  |
| [DHCP/BOOTP]<br>ON or OFF            | • | <b>Gateway address</b>  |
| [SMTP address]<br>xxx.xxx.xxx.xxx    | • | <b>DHCP/BootP</b>   |
| [SMTP username]<br>username          |   | ON - Enabled<br>OFF - Disabled  |
| [SMTP password]<br>password          |   | <b>SMTP server/login settings</b><br>Username and Password is only necessary if required by the server. |
| [DNS1 address]<br>xxx.xxx.xxx.xxx    | • | <b>Primary and Secondary DNS</b><br>Needed to be able to resolve host names                             |
| [DNS2 address]<br>xxx.xxx.xxx.xxx    |   |   |
| [Domain name]<br>domain              | • | <b>Default domain name for not fully qualified host names</b>   |
| [Host name]<br>anybus                | • | <b>Host name</b>  |
| [HICP password]<br>password          | • | <b>HICP password</b>  |

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

- DCP (See “DCP (Discovery and Basic Configuration)” on page 26).
- HICP (See “Anybus IPconfig (HICP)” on page 29)
- SSI (See “Server Side Include (SSI)” on page 35)
- DHCP/BootP (See “DHCP/BootP” on page 26)
- Mailbox Commands (See “Mailbox Editor” on page 97)

See also...

- “FTP Server” on page 30
- “Fieldbus Settings” on page 54

## 4.3 PROFINET Settings

The file “\pnio.cfg” holds various PROFINET-related settings. The file is read once during startup, i.e. the Anybus Communicator must be restarted in order for any changes to have effect (unless its contents has been changed by an IO Controller/Supervisor via the DCP protocol. In such case, the settings will have effect immediately).

*Example:*

|                               |  |
|-------------------------------|--|
| [Station Name]<br>Nice Device | • <b>Station Name</b><br>Station name as ASCII string, maximum 64 characters.            |
| [Station Type]<br>ABS-PRT     | • <b>Station Type</b><br>Station type as ASCII string, maximum 64 characters.            |
| [Vendor ID]<br>0x010C         | • <b>Vendor ID</b><br>16 bit hexadecimal value, with the prefix 0x. Assigned by the PNO. |
| [Device ID]<br>0x0001         | • <b>Device ID</b><br>16 bit hexadecimal value, with the prefix 0x. Assigned by vendor.  |

## 4.4 IP Access Control

It is possible to specify which IP addresses that are permitted to connect to the Anybus Communicator. This information is stored in the system file “\ip\_accs.cfg”.

*File Format:*

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

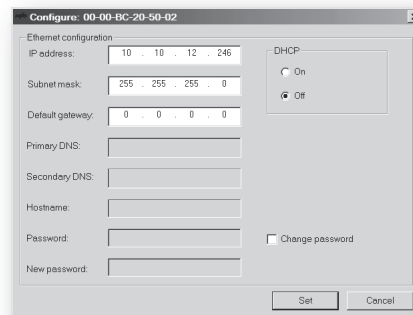
**Note:** \* (asterisk) may be used as a wildcard to select IP series.

## 4.5 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”).

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 Anybus Communicator will appear as “ABC-PRT”. 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.

## 5. FTP Server

### 5.1 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:

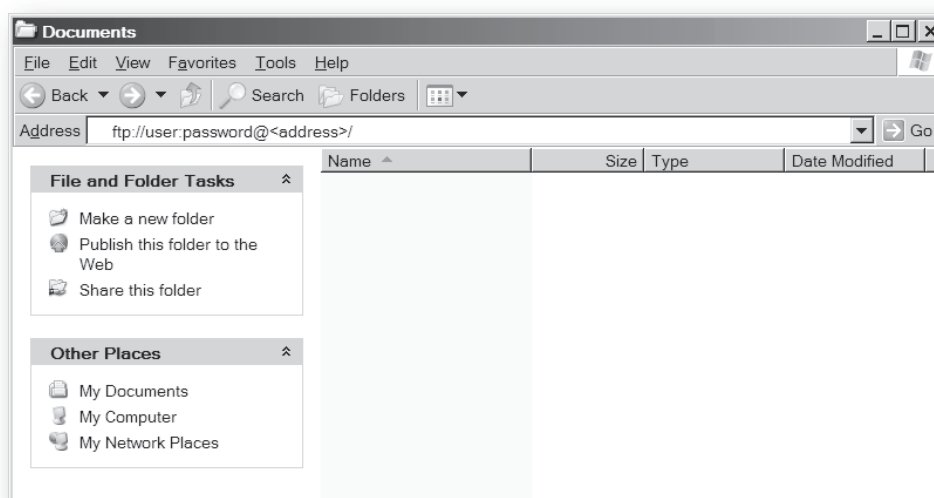
```
Username1:Password1  
Username2:Password2  
Username3:Password3
```

**Note:** If no valid user accounts have been defined, the Anybus Communicator will grant admin-level access to all users. In such case, the FTP accepts any username/password combination, and the root directory will be “\”.

## 5.2 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 module using the specified settings. If successful, the built-in file system is displayed in the Explorer window.



## 6. Web Server

### 6.1 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 35
- “IP Access Control” on page 28

#### 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         | - 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                |



## 6.2 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 the 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 32.

## 6.3 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.

## 7. Server Side Include (SSI)

### 7.1 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 Communicator encounters such a command, it will execute it, and replace it with the result (when applicable).

#### *Syntax*

XXX... below represents a command opcode and parameters associated with the command.

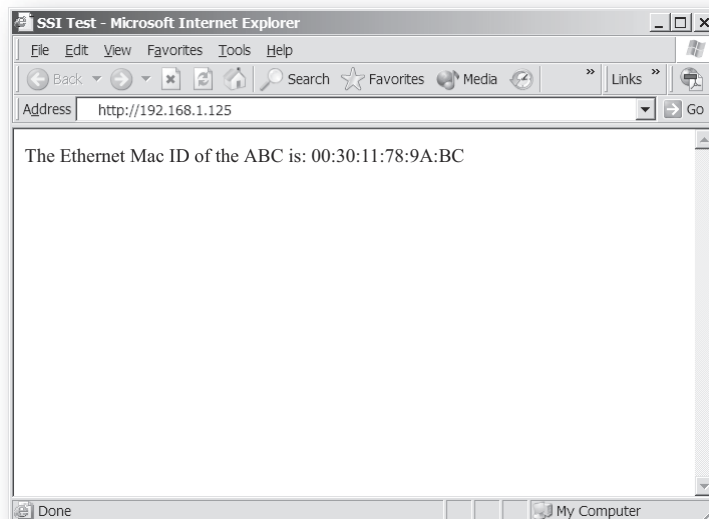
```
<?--#exec cmd_argument="XXXXXXXXXXXXXXXXXXXXX"-->
```

#### *Example*

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

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

Resulting web page:



## 7.2 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 network interface.

*Syntax:*

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

### DisplayFWVersion

This function returns the main firmware revision of the network interface.

*Syntax:*

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

### DisplayBLVersion

This function returns the bootloader firmware revision of the network interface.

*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 DHCP is supported, and "Arg2" if it is not.

*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"-->
```

**DisplaySMTPPwd**

This function returns the password used for SMTP authentication.

*Syntax:*

```
<?--#exec cmd_argument="DisplaySMTPPwd"-->
```

**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"-->
```

## StoreEtnIPConfig

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

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

*Syntax:*

```
<?--#exec cmd_argument="storeEtnIPConfig"-->
```

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
SetSMTPPwd
```

*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 function retrieves a text string from an object and stores it in the Output Data area.

*Syntax:*

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

ObjName- Name of object.

offset - Specifies the destination offset from the beginning of the Output Data area.

n - Specifies maximum number of characters to read (Optional)

*Default output:*

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

## printf

This function includes a formatted string, which may contain data from the input and output data areas, on a web page. The formatting of the string is similar to the C-language function printf().

### Syntax:

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

Like the C-language 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.ddddd e+-xx or [-]m.dddddE+-xx, where the number of d's specified by the precision. The default precision is 6; a precision of 0 suppresses the decimal point. |
| g, G      | float; %e or %E is used if the exponent is less than -4 or greater than or equal to the precision; otherwise %f is used. Trailing zeros and trailing decimal point are not printed.                     |
| %         | no argument is converted; print a %   |



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

| Argument  | Description   |
|---|---|
| InReadSByte( <i>offset</i> )                                    | Read a signed byte from position <i>offset</i> in the Input Data area                   |
| InReadUByte( <i>offset</i> )                                    | Read an unsigned byte from position <i>offset</i> in the Input Data area                |
| InReadSWord( <i>offset</i> )                                    | Read a signed word from position <i>offset</i> in the Input Data area                   |
| InReadUWord( <i>offset</i> )                                    | Read an unsigned word from position <i>offset</i> in the Input Data area                |
| InReadSLong( <i>offset</i> )                                    | Read a signed longword from position <i>offset</i> in the Input Data area               |
| InReadULong( <i>offset</i> )                                    | Read an unsigned longword from position <i>offset</i> in the Input Data area            |
| InReadString( <i>offset</i> )                                   | Read a string (char*) from position <i>offset</i> in the Input Data area                |
| InReadFloat( <i>offset</i> )                                    | Read a floating point (float) value from position <i>offset</i> in the Input Data area  |
| OutReadSByte( <i>offset</i> )                                   | Read a signed byte from position <i>offset</i> in the Output Data area                  |
| OutReadUByte( <i>offset</i> )                                   | Read an unsigned byte from position <i>offset</i> in the Output Data area               |
| OutReadSWord( <i>offset</i> )                                   | Read a signed word (short) from position <i>offset</i> in the Output Data area          |
| OutReadUWord( <i>offset</i> )                                   | Read an unsigned word (short) from position <i>offset</i> in the Output Data area       |
| OutReadSLong( <i>offset</i> )                                   | Read a signed longword (long) from position <i>offset</i> in the Output Data area       |
| OutReadULong( <i>offset</i> )                                   | Read an unsigned longword (long) from position <i>offset</i> in the Output Data area    |
| OutReadString( <i>offset</i> )                                  | Read a null-terminated string from position <i>offset</i> in the Output Data area       |
| OutReadFloat( <i>offset</i> )                                   | Read a floating point (float) value from position <i>offset</i> in the Output Data area |
| CipReadSByte( <i>class</i> ,<br><i>inst</i> , <i>attr</i> )     | Read a signed byte from a CIP-object  |
| CipReadUByte( <i>class</i> ,<br><i>inst</i> , <i>attr</i> )     | Read an unsigned byte from a CIP-object   |
| CipReadSWord( <i>class</i> ,<br><i>inst</i> , <i>attr</i> )     | Read a signed word from a CIP-object  |
| CipReadUWord( <i>class</i> ,<br><i>inst</i> , <i>attr</i> )     | Read an unsigned word from a CIP-object   |
| CipReadSLong( <i>class</i> ,<br><i>inst</i> , <i>attr</i> )     | Read a signed longword from a CIP-object  |
| CipReadULong( <i>class</i> ,<br><i>inst</i> , <i>attr</i> )     | Read an unsigned longword from a CIP-object   |
| CipReadFloat( <i>class</i> ,<br><i>inst</i> , <i>attr</i> )     | Read a floating point value from a CIP-object   |
| CipReadShort-String( <i>class</i> , <i>inst</i> , <i>attr</i> ) | Read a short string from a CIP-object   |
| CipReadString( <i>class</i> ,<br><i>inst</i> , <i>attr</i> )    | Read a null-terminated string from a CIP-object   |
| CipReadUByteArray( <i>class</i> , <i>inst</i> , <i>attr</i> )   | Read an unsigned byte-array from a CIP-object   |
| CipReadUWordArray( <i>class</i> , <i>inst</i> , <i>attr</i> )   | Read an unsigned word-array from a CIP-object   |
| CipReadULongArray( <i>class</i> , <i>inst</i> , <i>attr</i> )   | Read an unsigned longword-array from a CIP-object                                       |

## scanf

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

This function reads a string passed from an object in a HTML form, interprets the string according to the specification in format, and stores the result in the Output Data 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         | Unsigned decimal number; unsigned byte, unsigned short  |
| x         | Hexadecimal number (with or without leading 0x or 0X); byte, short  |
| c         | Characters; char*. The next input characters (default 1) are placed at the indicated spot. The normal skip over white space is suppressed; to read the next non-white space character, use %1s. |
| s         | Character string (not quoted); char*, pointing to an array of characters large enough for the string and a terminating "\0" that will be added.   |
| e, f, g   | Floating-point number with optional sign, optional decimal point and optional exponent; float*  |
| %         | Literal %; no assignment is made.   |

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

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

| Argument                                  | Description  |
|---|--|
| OutWriteByte( <i>offset</i> )             | Write a byte to position <i>offset</i> in the Output Data area                 |
| OutWriteWord( <i>offset</i> )             | Write a word to position <i>offset</i> in the Output Data area                 |
| OutWriteLong( <i>offset</i> )             | Write a long to position <i>offset</i> in the Output Data area                 |
| OutWriteString( <i>offset</i> )           | Write a string to position <i>offset</i> in the Output Data area               |
| OutWriteFloat( <i>offset</i> )            | Write a floating point value to position <i>offset</i> in the Output Data area |
| CipWriteByte( <i>class, inst, attr</i> )  | Write a byte value to a CIP-object   |
| CipWriteWord( <i>class, inst, attr</i> )  | Write a word value to a CIP-object   |
| CipWriteLong( <i>class, inst, attr</i> )  | Write a longword to a CIP-object   |
| CipWriteFloat( <i>class, inst, attr</i> ) | Write a floating point value to a CIP-object                                   |

*Default output:*

```
Write succeeded
Write failed
```

## IncludeFile

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

### Syntax:

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

### Default output:

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

## SaveDataToFile

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

This 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', [Append|Overwrite] )"-->
```

### Default output:

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

### DisplayScannerMode

This function returns the current scanner mode (run or idle state).

*Syntax:*

```
<?--#exec cmd_argument="DisplayScannerMode  
( 'Output when Run', 'Output when Idle' )"-->
```

### SetScannerMode

**Note:** This function cannot be used in email messages.

This function is used to set the EtherNet/IP Scanner to Run or Idle. A variable called “scanner\_state” shall be sent to the page with the value “run” or “idle” (other values will be ignored).

*Syntax:*

```
<?--#exec cmd_argument="setScannerMode"-->
```

*Default output:*

```
Failure - Change scanner mode not possible
```

## 7.3 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. Temporarily changing the SSI output by calling the SSI function “SsiOutput()”.

### 7.3.1 SSI Output String File

If the file “\ssi\_str.cfg” is found in the file system and the file is consistent with 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

### 7.3.2 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) )"-->
```

## 8. E-mail Client

### 8.1 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 35

#### 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 27

#### 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 48.

Note that the Input- and Output Data areas are scanned twice per second, i.e. to ensure that an event is detected by the module, it must be present longer than 0.5 seconds.

## 8.2 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 module.

*File Format:*

```
[Register]
Area, Offset, Type

[Register Match]
Value, Mask, Operand

[To]
recipient

[From]
sender

[Subject]
subject line

[Headers]
Optional extra headers

[Message]
message body
```

| 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 email address   |                 |
| Subject | E-mail subject. One line only.   |                 |
| Headers | Optional; may be used to provide additional headers.                             |                 |
| Message | The actual message.  |                 |

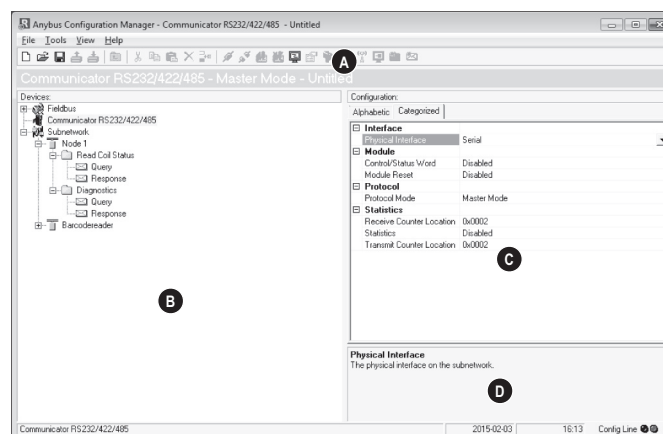
**Note:** Hexadecimal values must be written with the prefix “0x” in order to be recognized by the module.



## 3. Navigating ACM

### 3.1 Main Window

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



- **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.

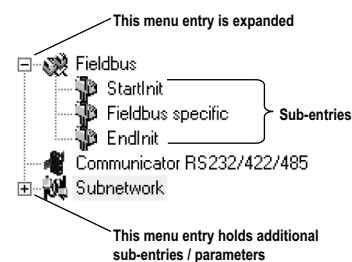
- **B: Navigation Section**

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.

### 3.1.1 Drop-down Menus

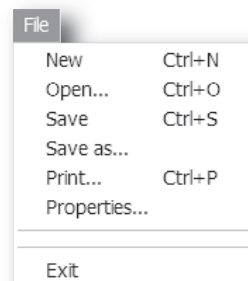
#### File

- **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.

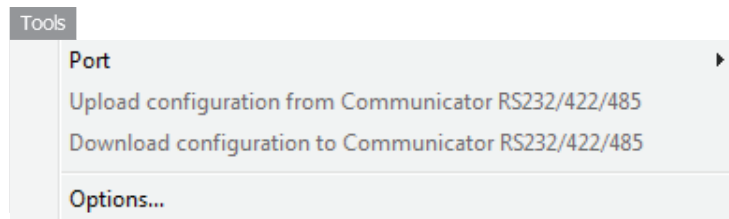
| Item                                | Description   |
|-------------------------------------|---|
| Select a Name for the Configuration | Enter a descriptive name for the new configuration                                |
| Enable Password                     | Enables password protection   |
| Download 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!

- **Exit**  
Close ACM.

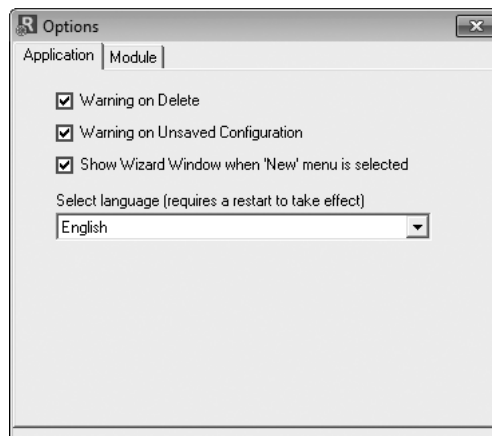


## Tools



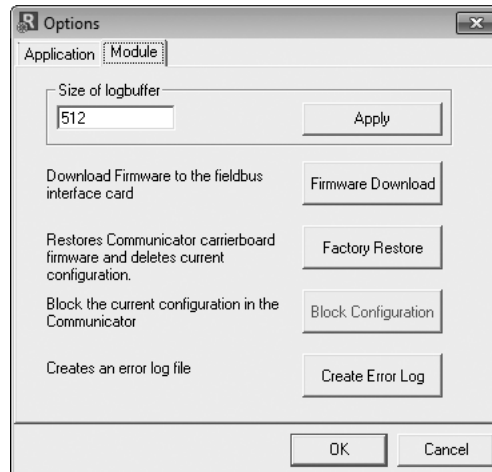
- Port**  
 Select the COM-port used for the configuration of the gateway.
- Upload configuration from Communicator RS232/422/485**  
 Upload the configuration from the gateway to ACM.
- Download configuration to Communicator RS232/422/485**  
 Download the current configuration to the gateway.
- Start Logging**  
 Start the Data Logger (see “Data Logger” on page 58).  
 Note that when the Data Logger is active, this menu entry is changed to “Stop Logging”.
- Options**

This will open the following window:



| 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:



| 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 58. |
| Firmware Download   | Download firmware to the embedded fieldbus interface.<br><b>Warning: Use with caution.</b>   |
| 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.<br><b>Warning: Use with caution.</b>  |
| Create Error log    | Creates an error log file  |

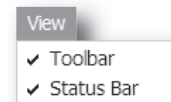
## View

- **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.



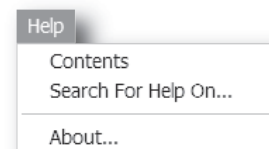
## Help

- **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.





### 3.1.2 Toolbar Icons

The toolbar features icons for the most commonly used functions.


- **New, Open & Save**  
See “File” on page 20.






New
Open
Save
- **Upload from ABC & Download to ABC**  
See “Tools” on page 21.


Upload
Download
- **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.




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




Disconnect
- **Start Logging & Stop Logging**  
See “Tools” on page 21 & “Data Logger” on page 58.


Start Log.
Stop Log.
- **Sub-network Monitor**  
Clicking on this icon will launch the sub-network Monitor (see “Sub-network Monitor” on page 53).





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 67)





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

Node
Broadcaster
- **Node Monitor**  
Clicking on this icon will launch the Node Monitor (see “Node Monitor” on page 54)



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

Add Transactions
Add Transaction

## 10. Basic Settings

### 10.1 Fieldbus Settings

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

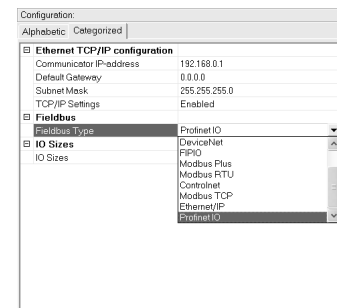


#### General

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 module.

#### Fieldbus Type

The Anybus Configuration Manager supports a wide range of networking systems. Make sure that this parameter is set to “Profinet IO”.

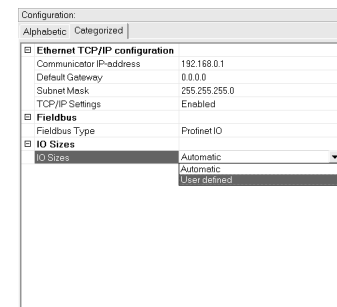


Fieldbus Type

#### TCP/IP Settings

To be able to participate on the network the following settings must be configured:

- **Communicator IP-address**  
(see “Basic Network Configuration” on page 26)
- **Default Gateway**  
(see “Basic Network Configuration” on page 26)
- **Subnet Mask**  
(see “Basic Network Configuration” on page 26)
- **TCP/IP Settings**  
Enabled- Use the settings above.  
Disabled- Use the settings stored in “ethcfg.cfg”



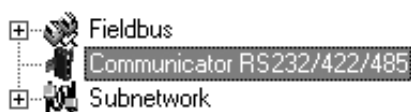
IO Sizes

#### IO Sizes

These parameters specify how data from the internal memory buffer shall be exchanged on PROF-INET. This can either be handled automatically by the module, or specified manually.

- **Automatic**  
All data will be represented as I/O Data.  
(see also “Data Representation (IO Data & Record Data)” on page 22)
- **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 “Data Representation (IO Data & Record Data)” on page 22)

## 4.2 Communicator Parameters



### Interface

Only serial communication is currently supported.

### Control/Status Word

See “Control and Status Registers” on page 62.

| 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.<br><b>Note:</b> 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.

## 4.3 Sub-network Parameters



### 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<br>2400<br>4800<br>9600<br>19200<br>35700<br>38400<br>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.



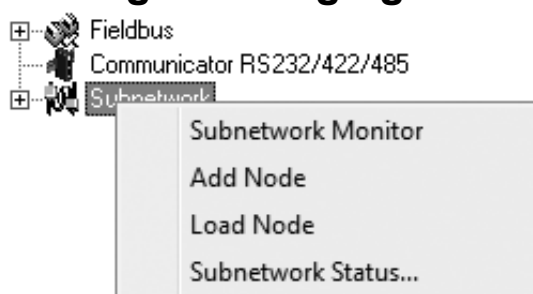
## 5. Nodes

### 5.1 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.

### 5.2 Adding & Managing Nodes



| Function                     | Description  |
|------------------------------|--|
| Paste                        | Paste a node from the clipboard                                  |
| Subnetwork Monitor           | Launch the subnet monitor (see "Sub-network Monitor" on page 53) |
| Add Node                     | Add a node to the configuration                                  |
| Add Broadcaster <sup>a</sup> | 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.

### 5.3 Node Parameters

#### 5.3.1 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.<br>For more information, see "The Command Editor" on page 43. |

## 6. Transactions

### 6.1 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**

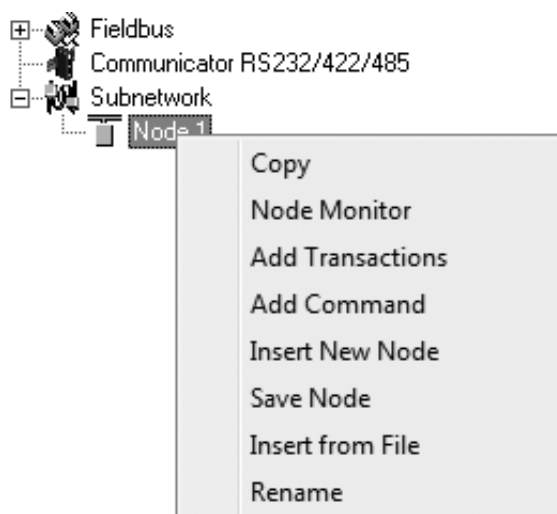
Transactions can be added as desired for both directions. Transactions sent to the sub-network are called “Transaction Produce”, and transactions issued by other nodes are called “Transaction Consume”.

- **DF1 Master Mode**

Please refer to “DF1 Protocol Mode” on page 46.

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

## 6.2 Adding & Managing Transactions



| Function                             | Description   |
|--------------------------------------|---|
| Copy                                 | Copy a node to the clipboard  |
| Delete <sup>a</sup>                  | Delete a node   |
| Node Monitor                         | Launch the node monitor (see "Node Monitor" on page 54)   |
| Add Transaction(s) <sup>b</sup>      | On regular nodes, this adds a Query and a Response. The two transactions will be grouped in order to increase readability.<br>On the Broadcaster, a single transaction will be added. |
| Add Transaction Consume <sup>c</sup> | Add a "Consume"-transaction   |
| Add transaction Produce <sup>c</sup> | 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   |

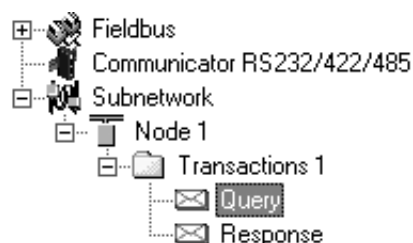
a. Only available if more than one node exists

b. Only available in Master Mode

c. Only available in Generic Data Mode

## 6.3 Transaction Parameters (Master Mode)

### 6.3.1 Parameters (Query & Broadcast)

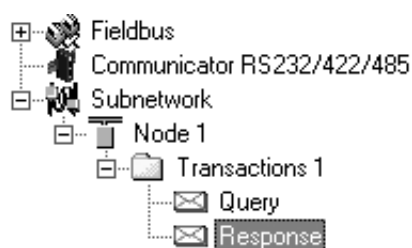


| Alphabetic Categorized                 |            |
|--|------------|
| <b>General</b>                         |            |
| Offline options for fieldbus           | Clear      |
| Offline options for sub-network        | Clear      |
| Update mode                            | Cyclically |
| 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

| Parameter                               | Description  |
|---|--|
| Minimum time between broadcasts (10 ms) | <p>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.</p> <p>The entered value is multiplied by 10. An entered value of 5 will result in 50 ms.</p> <p><b>Note:</b> This setting is only relevant for the Broadcaster node.</p>                    |
| Offline options for fieldbus            | <p>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.</p> <ul style="list-style-type: none"> <li>• <b>Clear</b> - The data destined for the slave-devices is cleared (set to zero)</li> <li>• <b>Freeze</b> - The data destined for the slave-device is frozen</li> <li>• <b>NoScanning</b> - The updating of the sub-network is stopped</li> </ul> |
| Offline options for sub-network         | <p>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.</p> <ul style="list-style-type: none"> <li>• <b>Clear</b> - Data is cleared (0) on the higher level network if the sub-network goes offline</li> <li>• <b>Freeze</b> - Data is frozen on the higher level network if the sub-network goes offline</li> </ul>                                    |
| Reconnect time (10 ms)                  | <p>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.</p> <p>The entered value is multiplied by 10. An entered value of 5 will result in 50 ms.</p> <p><b>Note:</b> This setting is not relevant for the Broadcaster node.</p>   |
| Retries                                 | <p>This parameter specifies how many times a timeout may occur in sequence before the node is disconnected.</p>  |
| Timeout time (10 ms)                    | <p>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.</p> <p>The entered value is multiplied by 10. An entered value of 5 will result in 50 ms.</p>   |
| Trigger byte address                    | <p>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").</p> <p>Valid settings range from 0x200 to 0x3FF and 0x400 to 0xFF</p>   |

| Parameter           | Description  |
|---------------------|--|
| Update mode         | <p>This parameter is used to specify when the transaction shall be sent to the slave:</p> <ul style="list-style-type: none"> <li>• <b>Cyclically</b><br/>The transaction is issued cyclically at the interval specified in the "Update time" parameter.</li> <li>• <b>On data change</b><br/>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.</li> <li>• <b>Single shot</b><br/>The Query is issued once at start up.</li> <li>• <b>Change of state on trigger</b><br/>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.</li> </ul> |
| Update time (10 ms) | <p>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").</p> <p>The entered value is multiplied by 10. An entered value of 5 will result in 50 ms.</p>  |

### 6.3.2 Parameters (Response)



| Alphabetic Categorized |          |
|------------------------|----------|
| Trigger                |          |
| Trigger byte           | Disabled |
| Trigger byte address   | 0x05FF   |

| Parameter            | Description   |
|----------------------|---|
| Trigger byte         | <p>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.</p> <p>The location of the trigger byte is specified by the "Trigger byte address" parameter below.</p> |
| Trigger byte address | <p>This parameter specifies the location of the trigger byte in the internal memory buffer.</p> <p>Valid settings range from 0x000 to 0x1FF and 0x400 to 0xFFF</p>  |

## 6.4 Transaction Parameters (Generic Data Mode)

### 6.4.1 Produce Transactions



| Alphabetic  | Categorized |
|---|-------------|
| <div>General</div> <div>Offline options for fieldbus</div> <div>Update mode</div> |             |
| Clear   |             |
| Cyclically  |             |
| <div>Timing</div> <div>Update time (10ms)</div>                                   |             |
| 100   |             |
| <div>Trigger</div> <div>Trigger byte address</div>                                |             |
| 0x05FF  |             |

| Parameter                    | Description   |
|------------------------------|---|
| Offline options for fieldbus | <p>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.</p> <ul style="list-style-type: none"> <li>• <b>Clear</b><br/>Data is cleared (0) on the sub-network if the higher level network goes offline</li> <li>• <b>Freeze</b><br/>Data is frozen on the sub-network if the higher level network goes offline</li> <li>• <b>NoScanning</b><br/>Stop subnet scanning for this transaction if the higher level network goes offline</li> </ul>   |
| Update mode                  | <p>The update mode for the transaction:</p> <ul style="list-style-type: none"> <li>• <b>Cyclically</b><br/>The transaction is sent cyclically at the interval specified in "Update Time".</li> <li>• <b>On data change</b><br/>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.</li> <li>• <b>Single shot</b><br/>The transaction is sent once at startup.</li> <li>• <b>Change of state on trigger</b><br/>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.</li> </ul> |
| Update time (10 ms)          | <p>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").</p> <p>The entered value is multiplied by 10. An entered value of 5 will result in 50 ms.</p>  |

| Parameter            | Description   |
|----------------------|---|
| Trigger byte address | <p>This parameter specifies location of the trigger byte in the internal memory buffer.</p> <p>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.</p> <p>This way, the control system can instruct the gateway to produce a specific transaction on the sub-network by updating the corresponding trigger byte.</p> <p>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.</p> <p><b>Note:</b> This parameter has no effect unless the “Update mode” parameter is set to “Change of state on trigger”.</p> <p>Valid settings range from 0x200 to 0x3FF and 0x400 to 0xFF</p> |

## 6.4.2 Consume Transactions

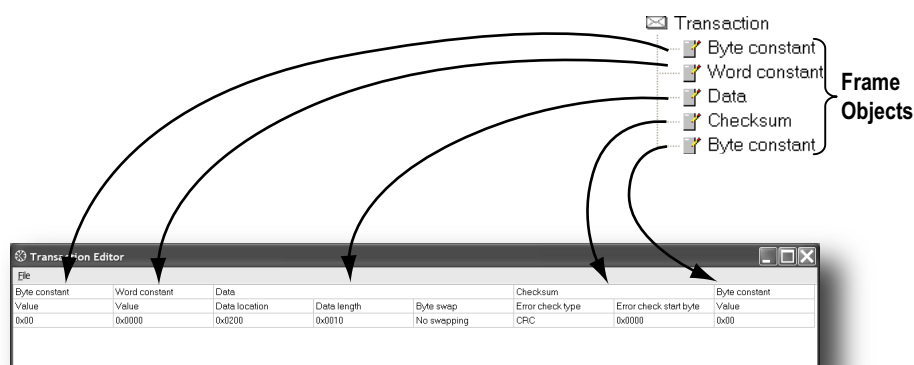


| Alphabetic                          | Categorized              |
|-------------------------------------|--------------------------|
| <input checked="" type="checkbox"/> | <input type="checkbox"/> |
| <b>General</b>                      |                          |
| Offline options for sub-network     | Clear                    |
| <b>Timing</b>                       |                          |
| Offline timeout time (10ms)         | 100                      |
| <b>Trigger</b>                      |                          |
| Trigger byte                        | Disabled                 |
| Trigger byte address                | 0x05FF                   |

| Parameter                       | Description  |
|---------------------------------|--|
| Offline options for sub-network | <p>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.</p> <ul style="list-style-type: none"> <li>• <b>Clear</b><br/>Data is cleared (0) on the higher level network if the sub-network goes offline</li> <li>• <b>Freeze</b><br/>Data is frozen on the higher level network if the sub-network goes offline</li> </ul>  |
| Offline timeout time (10 ms)    | <p>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.</p> <p>The entered value is multiplied by 10. An entered value of 5 will result in 50 ms.</p>   |
| Trigger byte                    | <ul style="list-style-type: none"> <li>• <b>Enable</b><br/>Enables the trigger byte. The location of the trigger byte must be specified in “Trigger byte address”.<br/>The trigger byte value will be increased each time a valid transaction has been consumed by the gateway.<br/>The trigger byte will also be increased if the offline option is set to “Clear” and the offline timeout time value is reached.<br/>This feature enables the control system to be notified each time new data has been consumed on the sub-network.</li> <li>• <b>Disable</b><br/>Disables the trigger byte functionality.</li> </ul> |
| Trigger byte address            | <p>This parameter specifies the location of the trigger byte in the internal memory buffer.</p> <p>Valid settings range from 0x000 to 0x1FF and 0x400 to 0xFF.</p> <p>Please note that the trigger byte address must be unique to each transaction. It can not be shared by two or more transactions.</p>  |

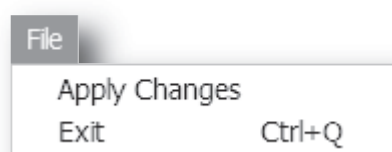
## 6.5 Transaction Editor

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.



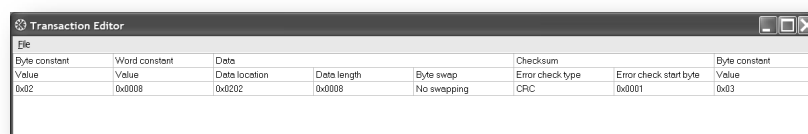
To edit the value of a parameter, click on it and enter a new value using the keyboard. When editing transactions which are based on predefined commands, certain parts of the transaction may not be editable.

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:*



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).



## 7. Frame Objects

### 7.1 General

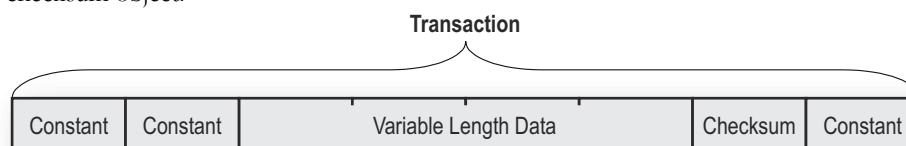
Each transaction consists of Frame Objects which makes up the serial telegram frame. Each Frame Object specifies how the gateway shall interpret or generate a particular part of the telegram.

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.



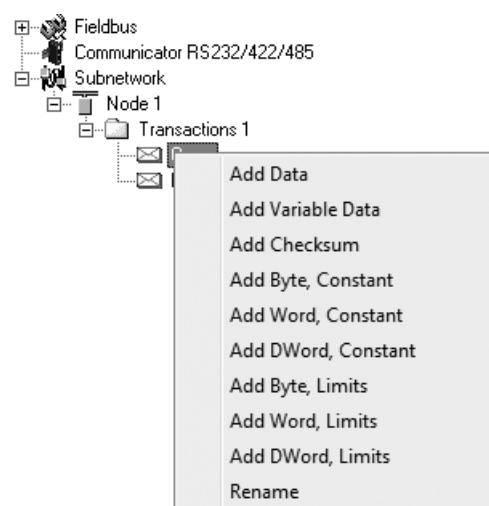
### 7.2 Adding and Editing Frame Objects

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 34.

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 34.



## 7.3 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 |

## 7.4 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 | <p>This is the largest allowed value for the range.</p> <p>Range: 0x00 to 0xFFh(byte)<br/>0x0000 to 0xFFFFh(word)<br/>0x00000000 to 0xFFFFFFFFh(dword)</p> <p><b>Note:</b> The value must be larger than the Minimum Value.</p> |
| Minimum Value | <p>This is the smallest allowed value for the range.</p> <p>Range: 0x00 to 0xFEh(byte)<br/>0x0000 to 0xFFFEh(word)<br/>0x00000000 to 0xFFFFFEEh(dword)</p> <p><b>Note:</b> The value must be less than the Maximum Value.</p>   |

## 7.5 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.

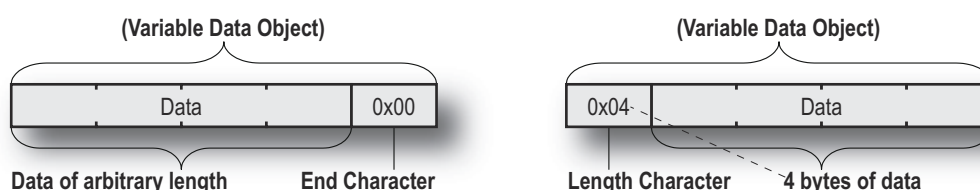
To specify the properties of the object, select it in the Navigation Section and enter the desired settings in the Parameter section as follows:

| Parameter     | Description   |
|---------------|---|
| Byte Swapping | <ul style="list-style-type: none"> <li>• <b>No Swapping</b><br/>No swapping is performed on the data</li> <li>• <b>Swap 2 bytes</b><br/>A, B, C, D becomes B, A, D, C</li> <li>• <b>Swap 4 bytes</b><br/>A, B, C, D becomes D, C, B, A</li> </ul> |
| 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.   |

## 7.6 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                          | <ul style="list-style-type: none"> <li>• <b>No Swapping</b><br/>No swapping will be performed on the data</li> <li>• <b>Swap 2 bytes</b><br/>A, B, C, D becomes B, A, D, C</li> <li>• <b>Swap 4 bytes</b><br/>A, B, C, D becomes D, C, B, A</li> </ul>   |
| Fill unused bytes                      | <ul style="list-style-type: none"> <li>• <b>Enabled<sup>a</sup></b><br/>Fill unused data with the value specified in "Filler byte".</li> <li>• <b>Disabled</b><br/>Don't fill</li> </ul>   |
| 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<br>(Produce/Query)    | <ul style="list-style-type: none"> <li>• <b>Length Character</b><br/>Length character visible in internal memory buffer but <i>not</i> sent out on the sub-network</li> <li>• <b>Length Character Visible</b><br/>Length character visible in internal memory buffer <i>and</i> sent out on the sub-network</li> <li>• <b>End Character</b><br/>End character visible in internal memory buffer but <i>not</i> sent out on the sub-network</li> <li>• <b>End Character Visible</b><br/>End character visible in the internal memory buffer <i>and</i> sent out on the sub-network</li> <li>• <b>No Character</b><br/>No end- or length-character generated in the internal memory buffer</li> </ul>  |
| Object Delimiter<br>(Consume/Response) | <ul style="list-style-type: none"> <li>• <b>Length Character</b><br/>Length character visible in internal memory buffer but <i>not</i> received from the sub-network</li> <li>• <b>Length Character Visible</b><br/>Length character visible in internal memory buffer <i>and</i> received from the sub-network</li> <li>• <b>End Character</b><br/>End character visible in internal memory buffer but <i>not</i> received from the sub-network</li> <li>• <b>End Character Visible</b><br/>End character visible in the internal memory buffer <i>and</i> received from the sub-network</li> <li>• <b>No Character</b><br/>No end or length characters included in the received string or generated in the internal memory buffer</li> </ul> |
| End Character Value                    | End Character value <sup>b</sup>   |
| 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"

## 7.7 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. <sup>a</sup>  |
| Error Check Type               | <p>This parameter specifies which type of algorithm to use:</p> <ul style="list-style-type: none"> <li>• <b>CRC (2 bytes)</b><br/>CRC-16 with 0xA001 polynome (Modbus RTU standard)</li> <li>• <b>LRC (1 byte)</b><br/>All bytes are added together as unsigned 8-bit values. The two's complement of the result will be used as a checksum.<br/>(Modbus ASCII standard with Error Check Start Byte = 0x01 and Representation = ASCII)</li> <li>• <b>XOR (1 byte)</b><br/>All bytes are logically XOR:ed together. The resulting byte will be used as a checksum.</li> <li>• <b>ADD (1 byte)</b><br/>All bytes are added together as unsigned 16-bit values. The lowest 8 bits in the result will be used as a checksum.</li> </ul> |
| Error check type combined with | <p>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).</p> <ul style="list-style-type: none"> <li>• <b>None</b><br/>The checksum binary value is transmitted without conversion.</li> <li>• <b>One's complement</b><br/>The checksum value will be converted to its one's complement (inverse code).<br/>Example: 00001100 will be transmitted as 11110011</li> <li>• <b>Two's complement</b><br/>The checksum value will be converted to its two's complement (complement code).<br/>Example: 00001100 will be transmitted as 11110100</li> </ul>   |
| Representation                 | <ul style="list-style-type: none"> <li>• <b>Binary</b><br/>The checksum is transmitted in binary format.</li> <li>• <b>ASCII</b><br/>All characters in the checksum are converted to ASCII values.</li> </ul>   |

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

## 8. Commands

This information is only valid for the Master and Generic Data modes. For DF1 master mode, please refer to “Services” on page 49.

### 8.1 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.

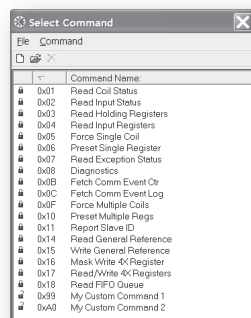
Adding a command to a node actually results in (a) transaction(s) being added according to the directions specified in the command. The frame objects in such a transaction may retrieve their values not only from parameters in the parameter section, but also from other sources such as the “SlaveAddress”-parameter (see “Node Parameters” on page 27). 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 43). For DF1 Master Mode, see “Services” on page 49. In Generic Data Mode, no predefined commands exist, but custom ones may be implemented as desired.

### 8.2 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:



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.

## 8.2.1 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 43.
- **Edit Command**  
Edit the currently selected command using the Command Editor.  
See also “The Command Editor” on page 43.
- **Delete Command**  
Delete the currently selected command from the list. Note that some commands are fixed and cannot be deleted.

## 8.2.2 Toolbar Icons

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





## 8.3 The Command Editor

### 8.3.1 General

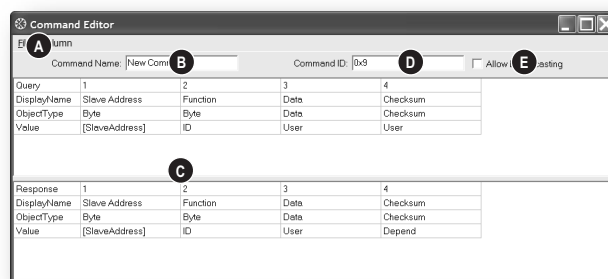
The Command Editor is used to define new commands and edit existing ones. This makes it possible to build a library of commands, which can be stored and reused at a later stage.

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.

### 8.3.2 Basic Navigation

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



#### A: Drop-down Menu

See “Drop-down Menu” on page 44.

#### 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)                             |

### 8.3.3 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.

### 8.3.4 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.<br>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 43).   |
| 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 27).  |
| (other settings) | Other settings are no longer supported.  |

### 8.3.5 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  |
|             | <i>The value of this byte constant will be set using the "SlaveAddress" parameter (see "Node Parameters" on page 27).</i> | <i>The value of this byte constant will be set using the "Command ID"-field.</i> | <i>The size and location of the data associated with this object is determined by the user.</i> | <i>The checksum type etc can be selected by the user. By default, this is set to match the Modbus-RTU standard.</i> |

- **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  |
|             | <i>This value is linked to the "SlaveAddress" parameter in the parameter window.</i> | <i>The value of this byte constant will be set using the "Command ID"-field.</i> | <i>The size and location of the data associated with this object is determined by the user.</i> | <i>This object will retrieve its settings from the corresponding object in the Query.</i> |

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.

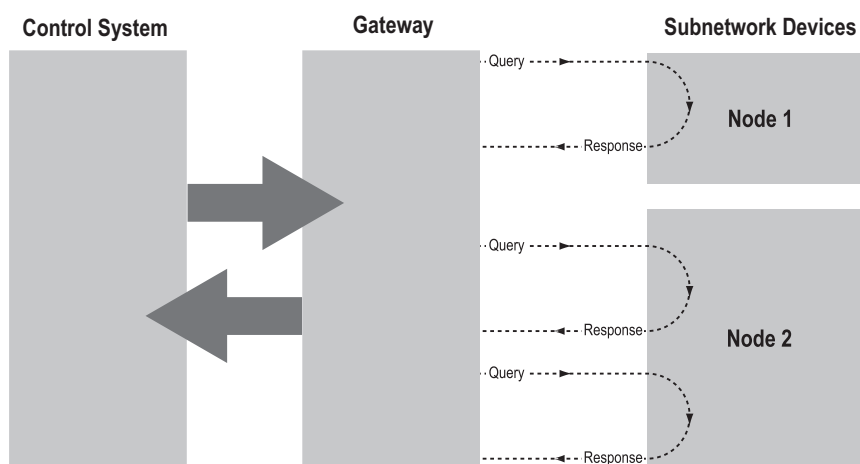
## 9. DF1 Protocol Mode

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

### 9.1 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”.

## 9.2 Communicator Parameters



### Interface

Currently, only serial communication is supported.

### Control/Status Word

(See “Control and Status Registers” on page 62).

| 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.<br><b>Note:</b> 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.

## 9.3 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<br>4800<br>9600<br>19200<br>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

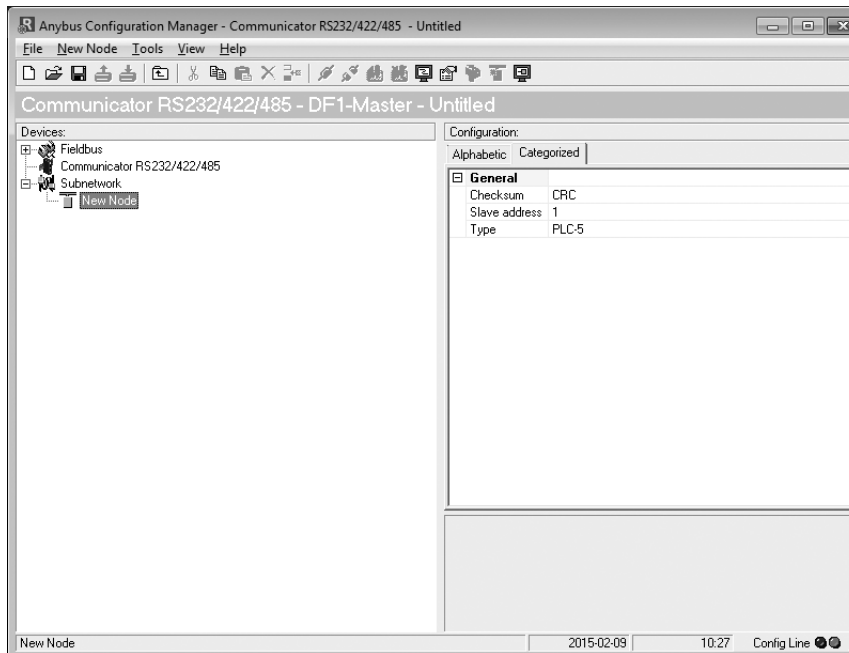
| Parameter                          | Description  | Default              |
|------------------------------------|--|----------------------|
| Master Node Address                | Node address of the master, valid values: 0–254                  | 1                    |
| Poll time, active slaves (10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

- 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.

## 9.4 Node Parameters



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 27.



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

## 9.5 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.

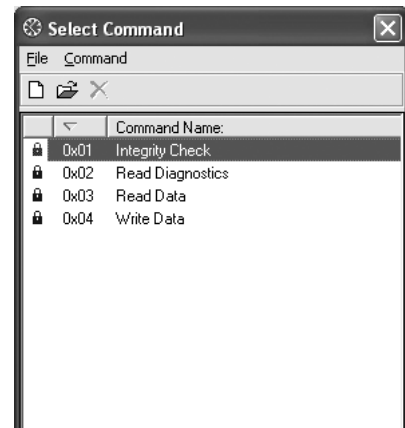
### 9.5.1 Available Services

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.

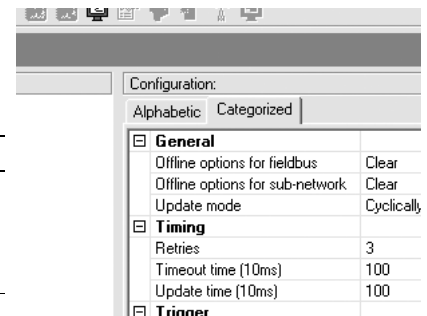


#### General Configuration Parameters

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

##### 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<br>Freeze<br>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<br>Freeze   |
| Update mode                     | The update mode for this service   | Cyclically<br>On data change<br>Single shot<br>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) <sup>a</sup>     | 1000 ms |
| Update time (10 ms)  | The minimum time between two services of this kind (in steps of 10 ms) <sup>a</sup> | 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<br>Valid settings range from 0x200 to 0x3FF and 0x400 to 0xFF | 0x05FF   |

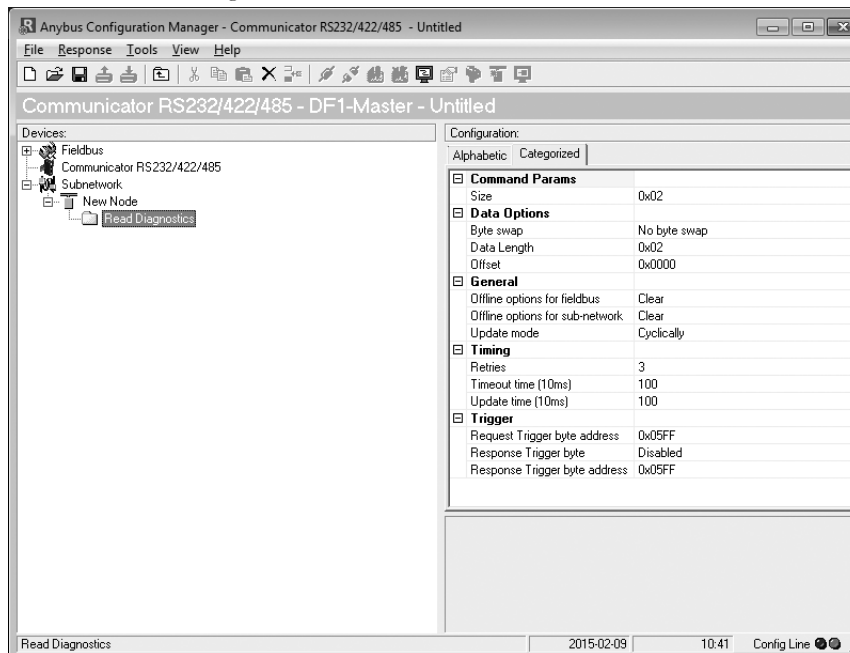


## 9.6 Integrity Check

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.

## 9.7 Read Diagnostics

This service reads diagnostic information from the module.



### 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:

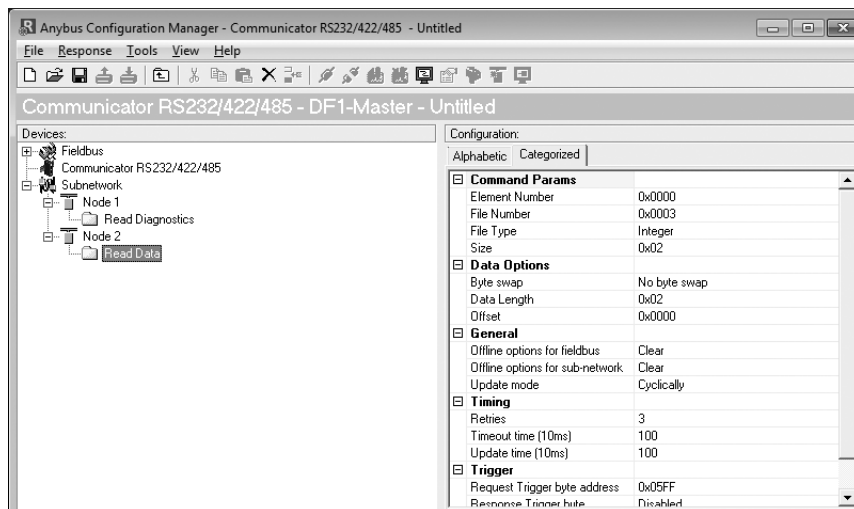
|                              | PLC-5 | SLC500 | MicroLogix |
|------------------------------|-------|--------|------------|
| <b>Size range (in bytes)</b> | 1–26  | 1–28   | 1–26       |

### Data options:

| Parameter   | Description   | Valid settings                                  |
|-------------|---|---|
| Byte swap   | Determines if the data shall be swapped   | No byte swap<br>Swap words<br>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.                   |   |

## 9.8 Read Data

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



### 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<br>SLC500: 0–255<br>MicroLogix: 0–255                               |
| File number    | The file number of the data file to be accessed.  | PLC-5: 3, 7, 8, 10–999<br>SLC500: 3, 7, 8, 10–255<br>MicroLogix: 3, 7, 8, 10–255 |
| File type      | The file type of the data to be accessed.   | Integer<br>Bit<br>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<br>SLC500: 2–236<br>MicroLogix: 2–242                               |

### Data Options

| Parameter   | Description   | Valid settings                                  |
|-------------|---|---|
| Byte swap   | Determines if the data shall be swapped.  | No byte swap<br>Swap words<br>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.<br><b>Note:</b> If the control and status registers are enabled (default), first available data location will be: Input area 0x002, Output area 0x202. | -   |

## 9.9 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.

## 10. Sub-network Monitor

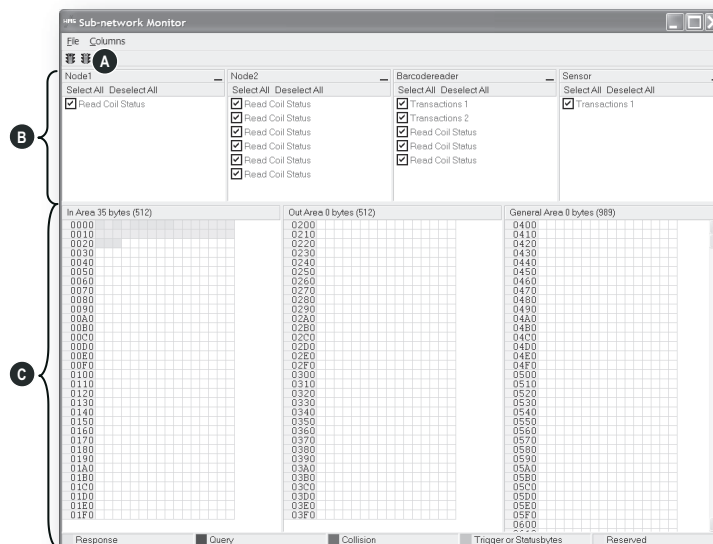
### 10.1 General

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 de-selecting 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.

### 10.2 Operation



#### A: Start Network & Stop Network Icons

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



#### 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 |

# 11. Node Monitor

## 11.1 General

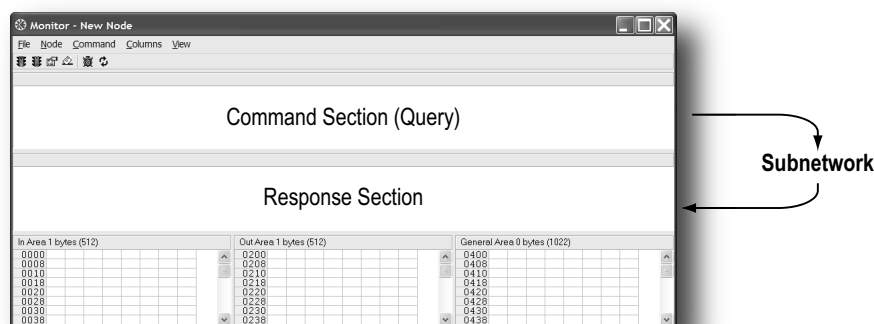
The Node Monitor can provide valuable information when setting up the communication with the sub-network, by allowing individual commands to be issued manually, and monitoring the response (if applicable). It also provides an overview of the memory used by a particular node.

**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:

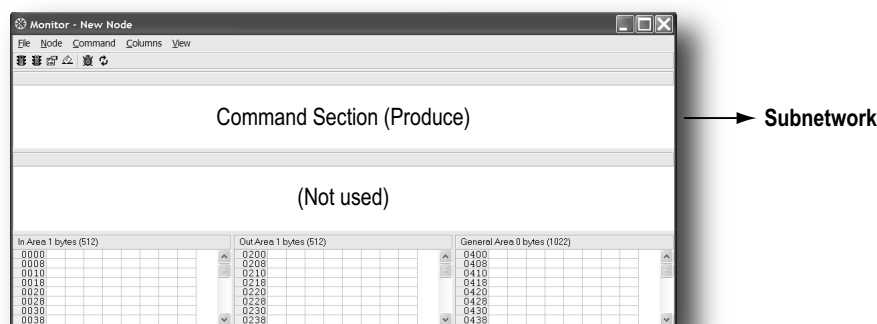
- **Master Mode and DF1 Master Mode**

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.

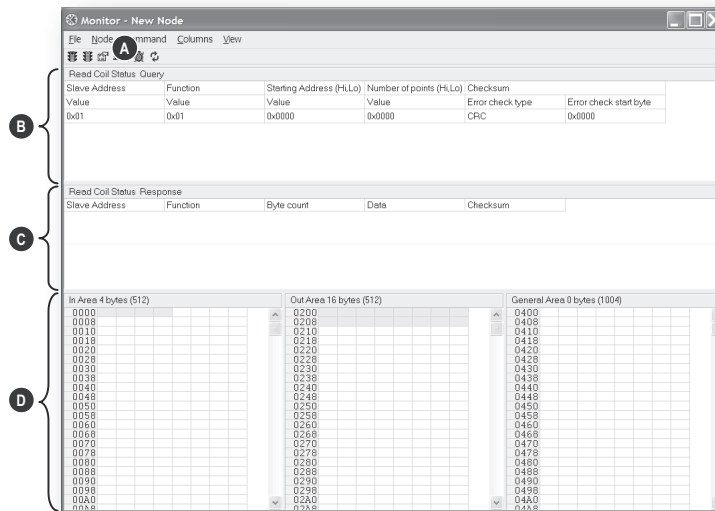


- **Generic Data Mode**

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



## 11.2 Navigating the Node Monitor



### A: Drop-down Menu & Toolbar Icons

See “Drop-down Menu” on page 56 and “Toolbar Icons” on page 57.

### 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 57.

## 11.2.1 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.

## 11.2.2 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 56.



Start



Stop

- **Select Command & Send Command**

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

See also “Command” on page 56.



Select



Send

- **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.



Stop



Resume

- **Refresh**

Refreshes the data displayed in the Monitor Section.



Refresh

## 12. Data Logger

### 12.1 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.

### 12.2 Operation

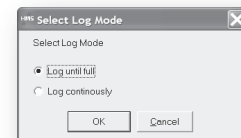
#### Start & Stop Logging

- **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.

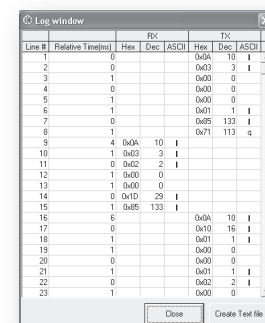


#### Log Window

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.





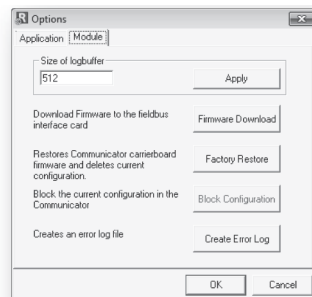
## 12.3 Configuration

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.



## 19. Configuration Wizards

### 19.1 General

When creating a new subnetwork 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 subnetwork 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 subnetwork fits the wizard profile; in all other cases the “Blank Configuration” option must be used.

### 19.2 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 51).

Currently, the following wizards are available:

- **Wizard - Modbus RTU Master**

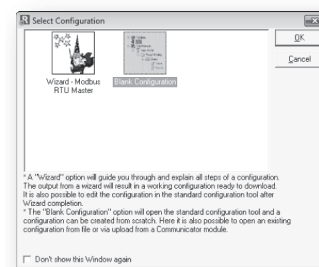
This option is suitable for Modbus RTU-based networks.

See also “Wizard - Modbus RTU Master” on page 91.

- **Blank Configuration**

This option creates an empty configuration.

Highlight the desired wizard and click “OK” to continue.



## 19.3 Wizard - Modbus RTU Master

This wizard can be used to create a Modbus-RTU-based network configuration based on certain information about the subnetwork. 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 subnetwork 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.

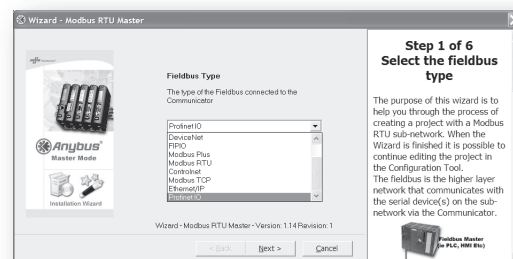
Using this wizard involves the following steps:

### Step 1: Communicator Type

Select "Profinet IO".

Click "Next" to continue.

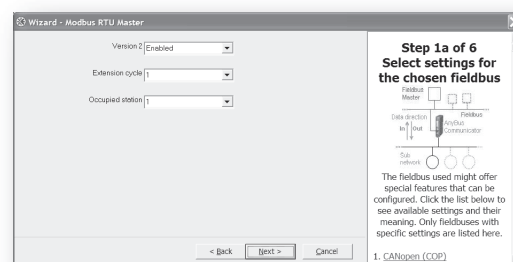
**Tip:** It is possible to return to a previous menu at any time without losing any settings by clicking "Previous".



### Step 1a: I/O Sizes

Specify the sizes of the input and output data areas. For more information, see "IO Sizes" on page 54.

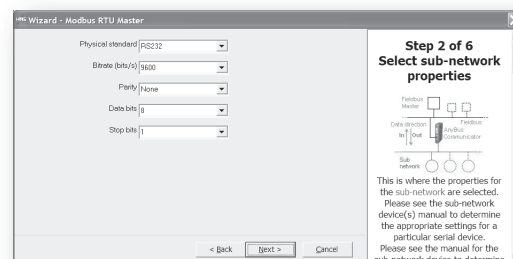
Click "Next" to continue.



### Step 2: Physical Settings

Select the physical properties of the subnetwork.

Click "Next" to continue.



### Steps 3 - 6

Consult the online help system for further information.

## 14. Control and Status Registers

### 14.1 General

The Control and Status Registers are disabled by default, but can be enabled using ACM (see “Control/Status Word” on page 25). 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...

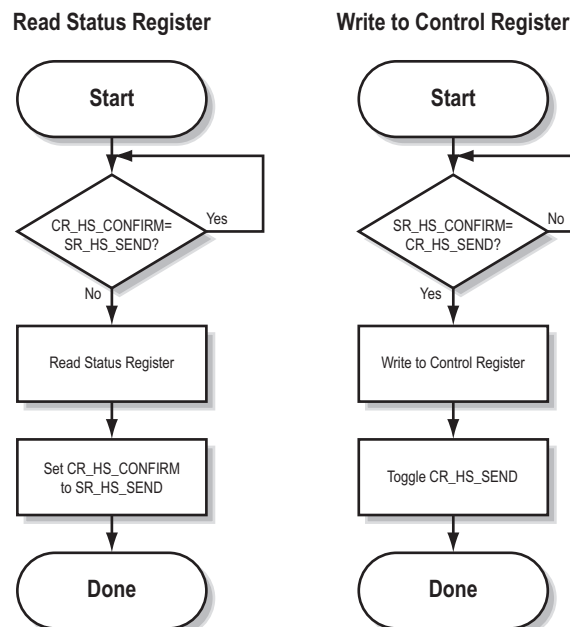
- Report sub-network related problems to the fieldbus control system
- Ensure that only valid data is exchanged in both directions
- Enable the fieldbus control system to start/stop data exchange with selected nodes on the sub-network

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.

#### 14.1.1 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.



### 14.1.2 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.

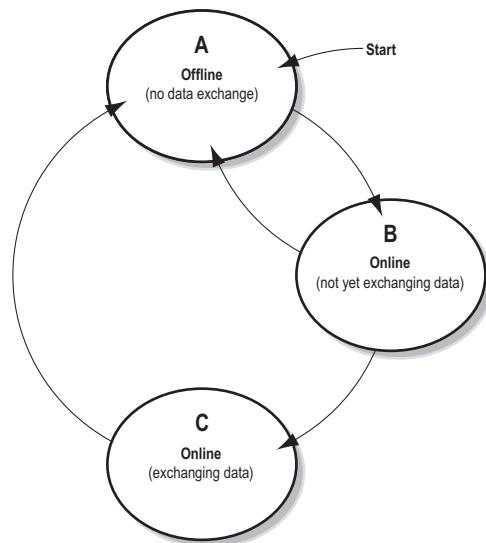
##### B: Online (Not yet exchanging data)

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.

## 14.2 Status Register Contents (Gateway to Control System)

### 14.2.1 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.   |
| 14      | Confirm (SR_HS_CONFIRM)                           | See also...<br>- “Handshaking Procedure” on page 62<br>- “Control Register Contents (Control System to Gateway)” on page 66   |
| 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.<br>1: Data Valid<br>0: Data not Valid<br><b>Note:</b> This bit is not used in Generic Data Mode. |
| 12... 8 | Status Code                                       | This field holds the last status report from the gateway.   |
| 7... 0  | Data  | See also...<br>- “Status Codes in Master Mode and DF1 Master Mode” on page 64<br>- “Status Code in Generic Data Mode” on page 65  |

**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.

### 14.2.2 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.

### 14.2.3 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.<br><br>See also...<br>- "Consume Transactions" on page 33 (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.

## 14.3 Control Register Contents (Control System to Gateway)

### 14.3.1 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<br>(CR_HS_CONFIRM) | These bits control the handshaking towards the gateway.   |
| 14      | Send<br>(CR_HS_SEND)       | See also...<br>- "Handshaking Procedure" on page 62<br>- "Status Register Contents (Gateway to Control System)" on page 64  |
| 13      | Data Valid                 | This bit controls data consistency (see "Data Consistency" on page 63).<br>1: Output Area valid; exchange data on the sub-network<br>0: Output Area not valid; do not exchange data on the sub-network<br><b>Note:</b> 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).   |
| 7... 0  | Data                       | See also...<br>- "Control Codes in Master Mode and DF1 Master Mode" on page 66<br>- "Control Codes in Generic Data Mode" on page 66   |

**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.

### 14.3.2 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. |

### 14.3.3 Control Codes in Generic Data Mode

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



## 15. Advanced Fieldbus Configuration

### 15.1 General

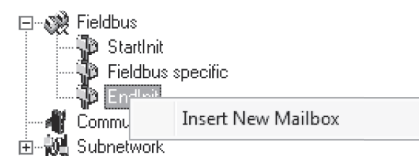
The fieldbus interface of the gateway consists of an embedded Anybus-S communication interface. Normally, the Anybus-S configuration settings are set up automatically by the gateway. However, advanced users can configure the Anybus-S card for specific features. This chapter assumes that the reader is familiar with the Anybus-S and its 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.)

### 15.2 Mailbox Editor

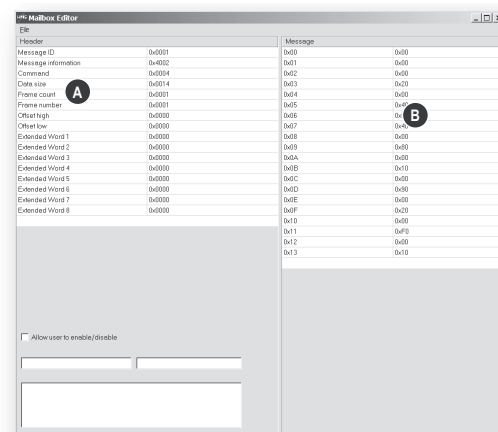
To add a mailbox message to the configuration, right-click on “EndInit” and select “Insert New Mailbox”.



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).

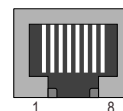


For more information about fieldbus specific mailbox messages, consult the separate Anybus-S Fieldbus Appendix for the fieldbus you are using. For general information about the Anybus-S platform, consult the Anybus-S Design Guide.

## A. Connector Pin Assignments

### A.1 PROFINET Connector (Ethernet)

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



### A.2 Power Connector

| Pin | Description |
|-----|-------------|
| 1   | +24 VDC     |
| 2   | GND         |

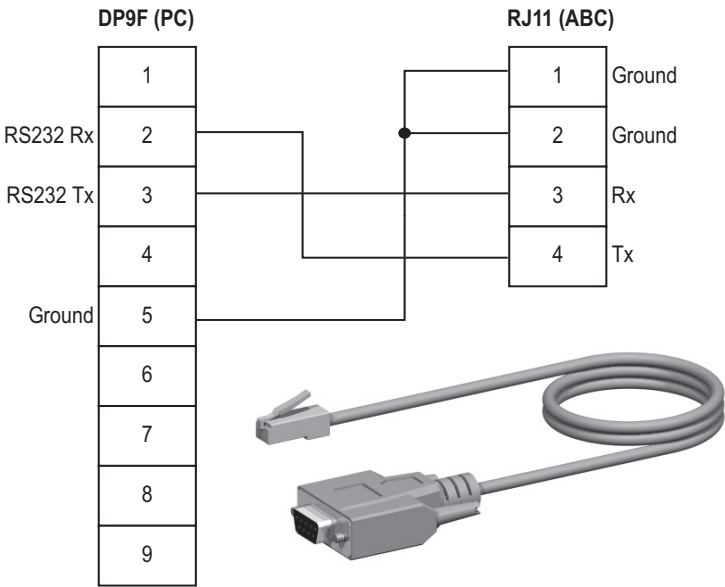


**Notes:**

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

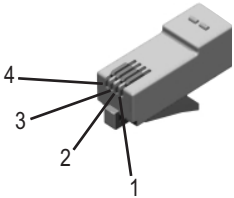
## A.3 PC Connector

### Configuration Cable Wiring



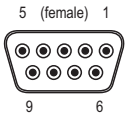
### RJ11 (4P4C modular)<sup>1</sup> : ABC

| 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.

## A.4 Subnetwork Interface

### A.4.1 General Information

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

### A.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.

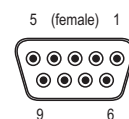
### A.4.3 Termination (RS485 & RS422 Only)

To avoid reflections on the serial lines, it is important to properly terminate the subnetwork 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.

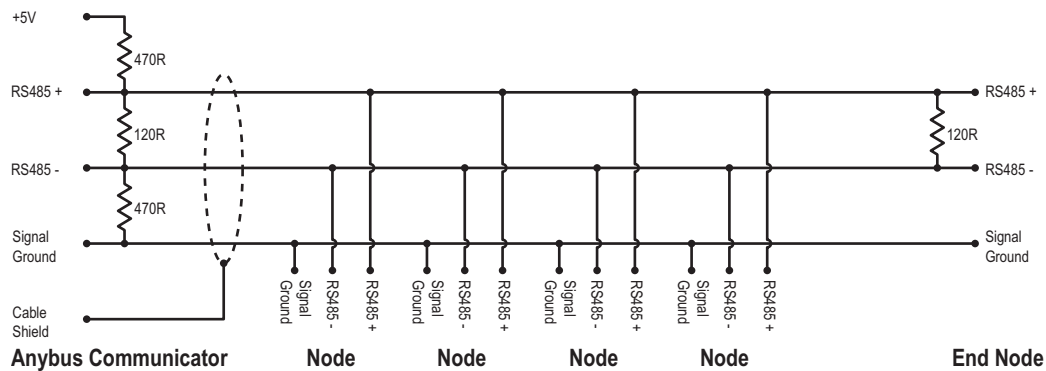
### A.4.4 Connector Pinout (DB9F)

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

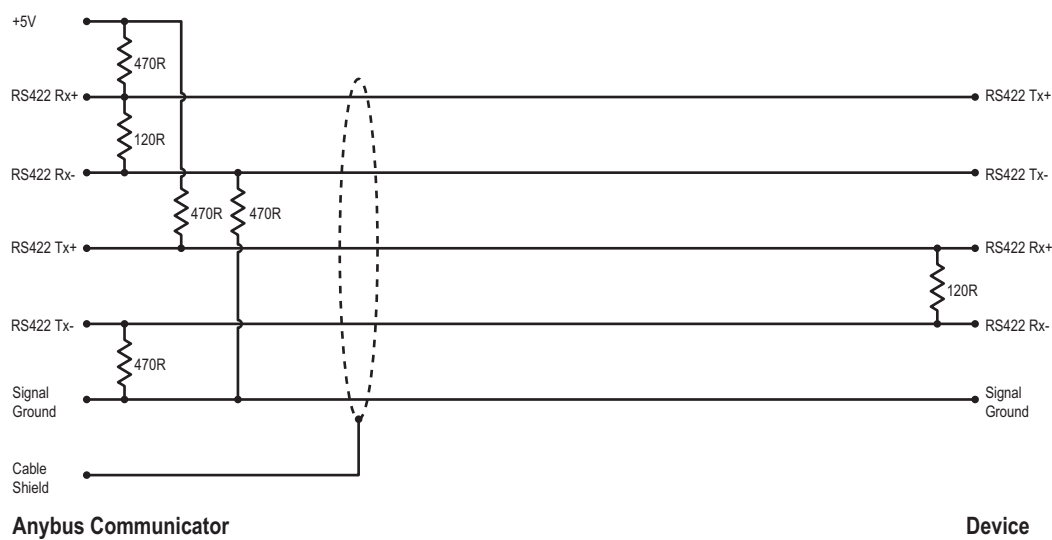


a. Connecting this signal directly to Protective Earth (PE) of other nodes may, in case of grounding loops etc., cause damage to the on-board serial transceivers. It is therefore generally recommended to connect it only to Signal Ground (if available) of other nodes.

A.4.5 Typical Connection (RS485)

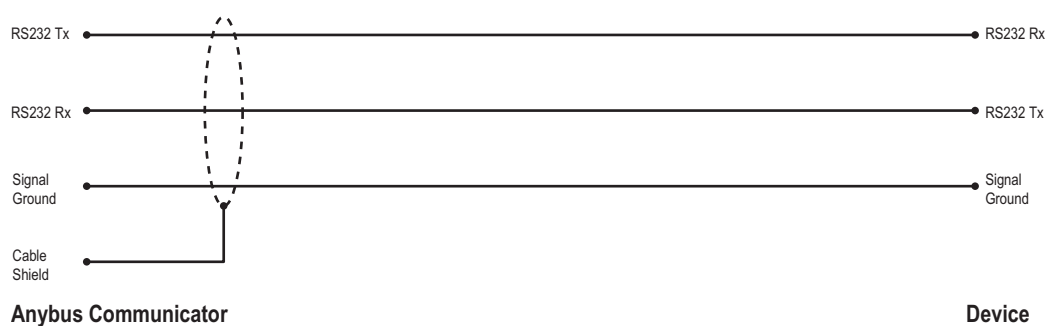


A.4.6 Typical Connection (RS422 & 4-Wire RS485)



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

A.4.7 Typical Connection (RS232)



## **B. Technical Specification**

### **B.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")

### **B.2 Electrical Characteristics**

#### **Power Supply**

Power: 24 VDC  $\pm$  10%

#### **Power Consumption**

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

### **B.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 |

## B.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)  $\pm 8$  kV air discharge,  $\pm 4$  kV contact discharge  
EN 61000-4-4 (1995)  $\pm 2$  kV power port,  $\pm 1$  kV other ports  
EN 61000-4-5 (1995)  $\pm 0.5$  kV power ports (DM/CM),  $\pm 1$  kV signal ports

### UL/c-UL Compliance



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                       | mm <sup>2</sup> | 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 IIb
  - 250 V<sub>RMS</sub> or 250 VDC working voltage
  - 500 V secondary circuit transient rating



## C. Troubleshooting

| Problem  | Solution   |
|--|--|
| Problem during configuration Upload / Download.<br>The Config Line "LED" turns red in ACM. | <ul style="list-style-type: none"> <li>Serial communication failed. Try again</li> </ul>   |
| The serial port seems to be available, but it is not possible to connect to the gateway    | <ul style="list-style-type: none"> <li>The serial port may be in use by another application. Exit ACM and close all other applications including the ones in the system tray.<br/>Try again</li> <li>Select another serial port<br/>Try again</li> </ul>   |
| Poor performance   | <ul style="list-style-type: none"> <li>Right click "sub-network" in the Navigation window and select "sub-network Status" to see status / diagnostic information about the sub-network.<br/>If the gateway reports very many retransmissions, check your cabling and/or try a lower baud rate setting for the sub-network (if possible).</li> <li>Is the Subnet Monitor in ACM active?<br/>The sub-network monitor has a negative influence on the overall performance of the gateway, and should only be used when necessary.</li> <li>Is the Node Monitor in ACM active?<br/>The node monitor has a negative influence on the overall performance of the gateway, and should only be used when necessary.</li> </ul> |
| No sub-network functionality   | <ul style="list-style-type: none"> <li>Use the "Data logger"-functionality to record the serial data communication on the sub-network.</li> <li>If no data is being transmitted, check the configuration in ACM.</li> <li>If no data is received, check the sub-network cables. Also verify that the transmitted data is correct.</li> </ul>   |

## D. ASCII Table

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

\*\*\*\*\*

Copyright 1986 by Carnegie Mellon.

\*\*\*\*\*

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

\*\*\*\*\*

Copyright (c) 1988 Stephen Deering.

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

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

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

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

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

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

\*\*\*\*\*

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

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

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

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

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

\*\*\*\*\*

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

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

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

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

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