Anybus® Communicator™

PROFINET® IRT (2.32)

USER MANUAL

Important User Information

Disclaimer

The information in this document is for informational purposes only. Please inform HMS Industrial Networks of any inaccuracies or omissions found in this document. HMS Industrial Networks disclaims any responsibility or liability for any errors that may appear in this document.

HMS Industrial Networks reserves the right to modify its products in line with its policy of continuous product development. The information in this document shall therefore not be construed as a commitment on the part of HMS Industrial Networks and is subject to change without notice. HMS Industrial Networks makes no commitment to update or keep current the information in this document.

The data, examples and illustrations found in this document are included for illustrative purposes and are only intended to help improve understanding of the functionality and handling of the product. In view of the wide range of possible applications of the product, and because of the many variables and requirements associated with any particular implementation, HMS Industrial Networks cannot assume responsibility or liability for actual use based on the data, examples or illustrations included in this document nor for any damages incurred during installation of the product. Those responsible for the use of the product must acquire sufficient knowledge in order to ensure that the product is used correctly in their specific application and that the application meets all performance and safety requirements including any applicable laws, regulations, codes and standards. Further, HMS Industrial Networks will under no circumstances assume liability or responsibility for any problems that may arise as a result from the use of undocumented features or functional side effects found outside the documented scope of the 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.

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

Table of Contents

1   Preface

1.1   About This Document

This document describes how to install and configure the Anybus Communicator PROFINET IRT (2.32) gateway

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

1.2   Document history

Version	Date	Description
1.0	2017-02-07	First release
1.1	2017-11-22	Updated for new firmware
1.2	2019-04-11	Added section about PROFINET Asset Management

1.3 Document Conventions

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

1. First do this

2. Then do this

Unordered (bulleted) lists are used for:

• Itemized information

• Instructions that can be carried out in any order

...and for action-result type instructions:

► This action...

→ leads to this result

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

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

This is a cross-reference within this document: *Document Conventions, p. 4*

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

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

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

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

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

2 Description

2.1 Introduction

Anybus Communicator PROFINET IRT (2.32) is intended for connecting non-networked industrial devices and equipment to a PROFINET network. It performs an intelligent protocol conversion and presents the serial data to the PROFINET IO Controller as easily processed I/O data.

The gateway is capable of converting almost any type of serial protocol such as Modbus RTU, ASCII, DF1 or any other type of proprietary Query/Response or Produce/Consume protocol. It can address up to 31 nodes on the serial subnetwork and supports the RS-232, RS-422 and RS-485 physical standards.
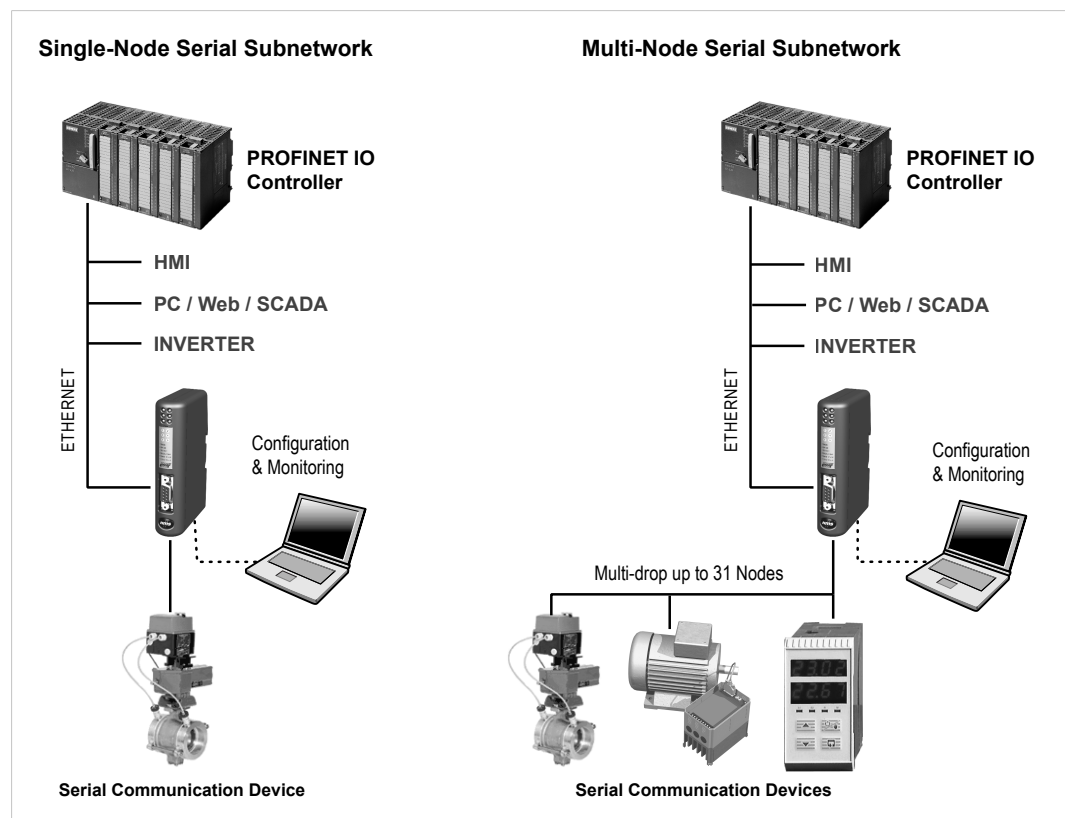

Fig. 1   Network examples

2.2   Basic Operation

Anybus Communicator PROFINET IRT (2.32) is designed to exchange data between a serial subnetwork and a higher level network. It does not have a fixed protocol for the subnetwork 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. It can also monitor communication on the subnetwork and notify the higher level network when data has changed.

An essential part of the Anybus Communicator package is Anybus Configuration Manager, a Windows-based application used to supply the gateway with a description of the subnetwork protocol. The application has a easy to use graphical interface and does not require programming skills.
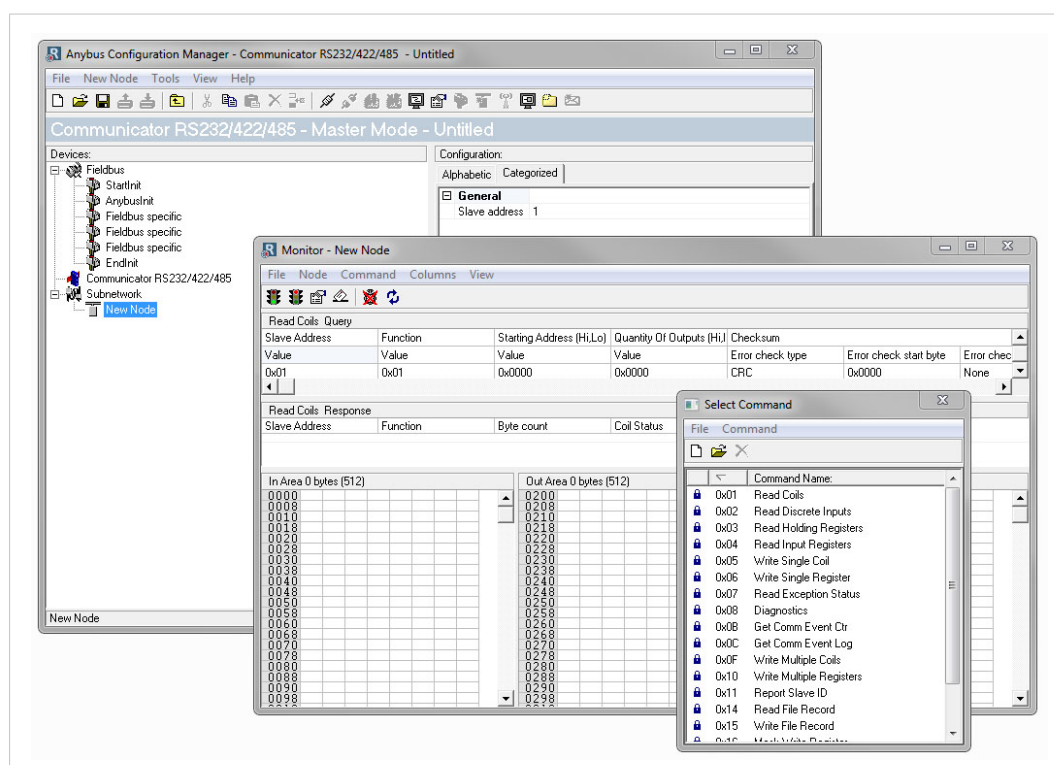

Fig. 2   Anybus Configuration Manager

2.3   Data Exchange Model

2.3.1   Overview

The data exchanged on the subnetwork and the data exchanged on the higher level network reside in the same internal memory in the Anybus Communicator. In order to exchange data with the subnetwork, the higher level network simply reads and writes data to memory locations that have been specified in Anybus Configuration Manager. The same memory locations can then be exchanged on the subnetwork.
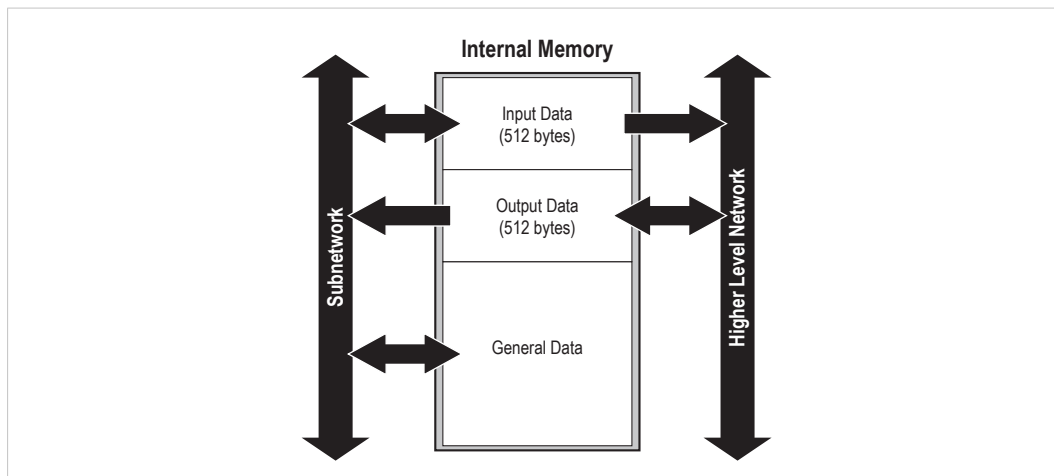
Fig. 3   Memory buffer structure

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

Input Data (512 bytes)	This area can be read from by the higher level network.
Output Data (512 bytes)	This area can be read from and written to by the higher level network.
General Data (max. 1024 bytes)	This area cannot be accessed by 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 exchanged on the subnetwork but can be a maximum of 1024 bytes.

2.3.2   Memory Map

When building the subnetwork configuration in Anybus Configuration Manager the areas in the memory buffer will be mapped to the following memory locations:

Fig. 4   Memory Map

2.3.3   Data Exchange Example

In this example, a temperature regulator on the serial subnetwork exchanges data with a PLC on the higher level network via the internal memory buffers in the Anybus Communicator.
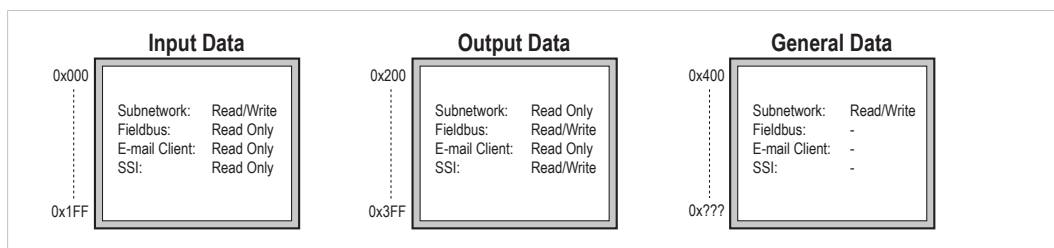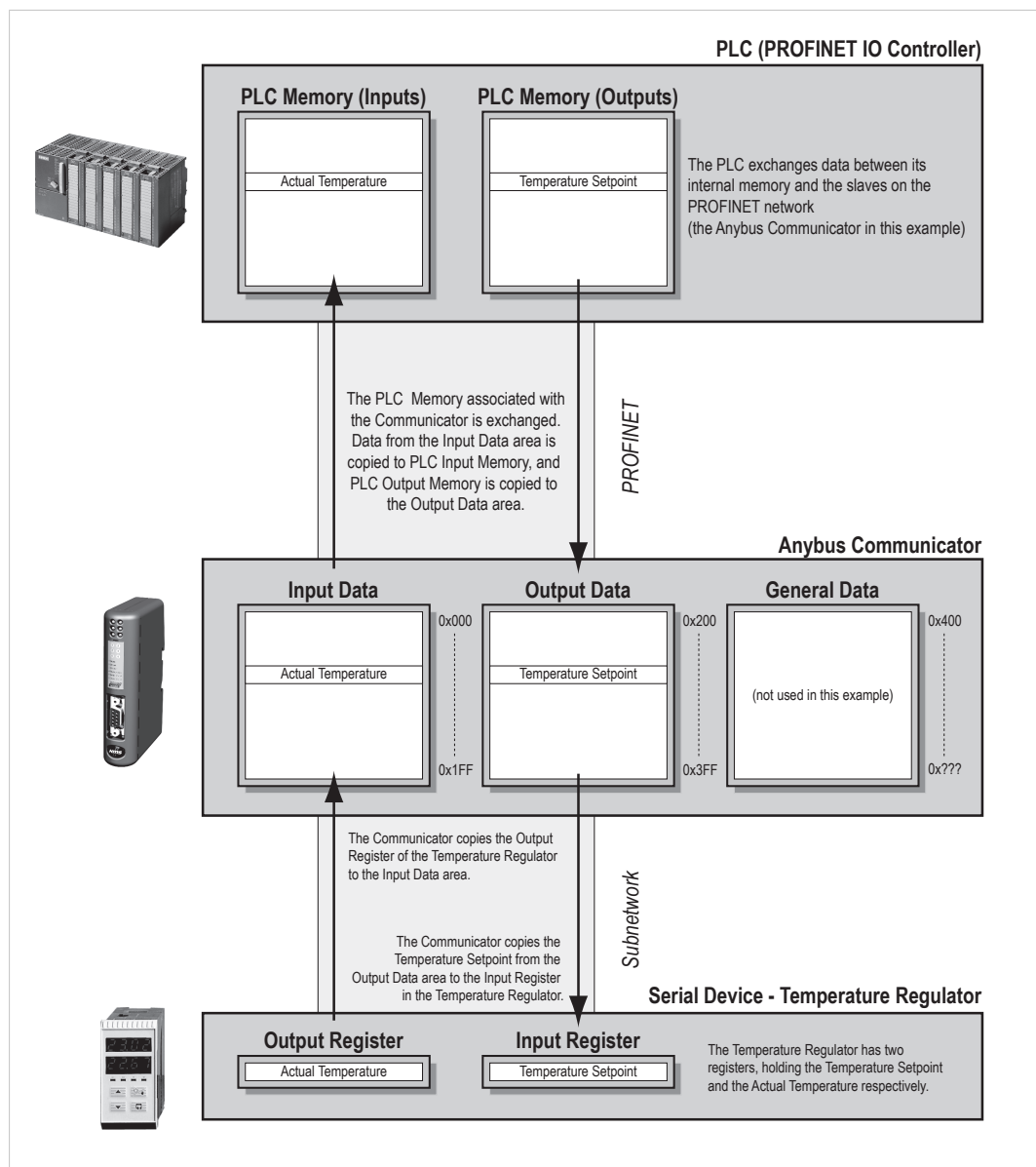

Fig. 5   Example

2.4   Subnetwork Protocol

2.4.1   Protocol Modes

> **ⓘ**   *This section is a general description for Anybus Communicator gateways. Some models may not support all features or allow data flow in both directions.*

The Anybus Communicator features three distinct modes of operation regarding subnetwork communication: *Master Mode*, *Generic Data Mode* and *DF1 Master Mode*.

Master Mode

In this mode, the Anybus Communicator acts as a master on the subnetwork and serial communication takes place in query-response fashion. The nodes on the network are not permitted to issue messages until they are addressed by the gateway.

Broadcasts are an exception: Most protocols offer some way of sending messages simultaneously to all nodes on the network without expecting a response. This is also implemented in the Anybus Communicator, which features a dedicated broadcaster node.
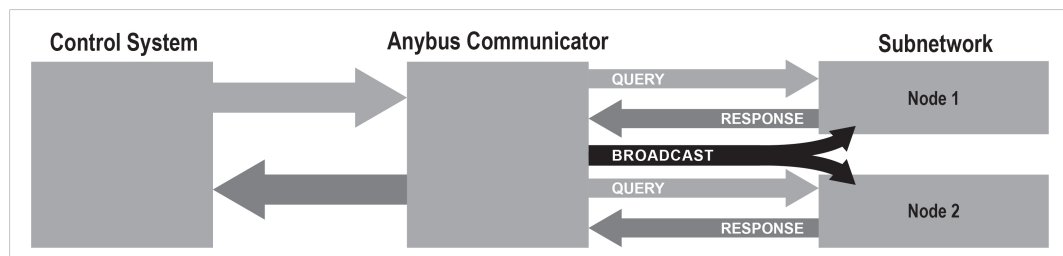
Fig. 6   Master Mode

Anybus Configuration Manager comes preloaded with the most commonly used Modbus RTU commands in Master Mode. Note however that this does not prevent other protocols based on the same query-response message-scheme to be implemented.

Generic Data Mode

In this mode there is no master-slave relationship between the subnetwork nodes and the Anybus Communicator. Any node on the subnetwork, including the Anybus Communicator, may spontaneously *produce* or *consume* messages. Nodes do not have to respond to messages or wait for a query in order to send one.

The consumed data can be accessed from the higher level network, and/or vice versa.

Fig. 7   Generic Data Mode

DF1 Master Mode

In this mode the Anybus Communicator act as a DF1 protocol master on the subnetwork. Serial communication takes place in query-response fashion. The nodes on the network are not permitted to issue messages until they are addressed by the gateway.

Communication in DF1 Master mode is based on *services*. A service represents a set of commands and operations on the subnetwork that have been predefined in the Anybus Communicator. Each service is associated with a set of parameters controlling how and when to use it on the subnetwork.
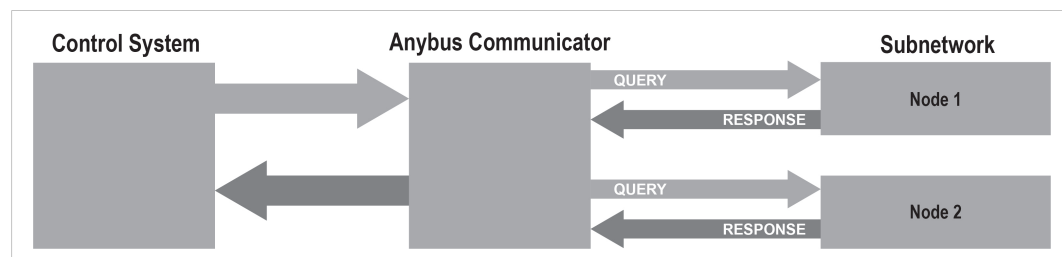
Fig. 8   DF1 Master Mode

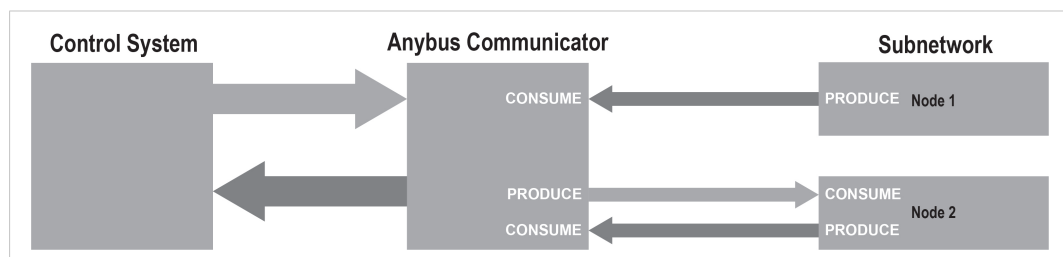Anybus Configuration Manager comes preloaded with a number of services which can be selected by the user. The actual DF1 commands that perform the services during runtime have been predefined in the Anybus Communicator.

2.4.2   Protocol Building Blocks

The following building blocks are used in Anybus Configuration Manager to describe the subnetwork communication.

Node	A *node* represents a single device on the subnetwork. Each node can be associated with a number of *transactions*.
Transaction	A *transaction* represents a complete serial telegram and consists of a number of *frame objects*. Each transaction is associated with a set of parameters controlling how and when to use it on the subnetwork.
Commands	A *command* is a predefined transaction which is stored in a list in Anybus Configuration Manager. This simplifies common operations by allowing transactions to be stored and reused.
Frame Object	A *frame object* is a low level entity that is used to compose a *transaction*. A frame object can represent a fixed value (a constant), a range of values (limit objects), a block of data, or a calculated checksum.

See *Anybus Configuration Manager, p. 36* on how to use protocol building blocks.

3   Installation

!	This product contains parts that can be damaged by electrostatic discharge (ESD). Use ESD prevention measures to avoid damage.

3.1   Installation Overview

These are the basic steps for installing Anybus Communicator gateways.

Depending on the fieldbus network type there may also be configuration switches, etc. on the Anybus Communicator that need setting. See the following sections in this document for more information.

Basic installation steps

1.   Mount the Anybus Communicator on the DIN rail.

2.   Connect the serial and fieldbus network interfaces.

3.   Configure the fieldbus network interface hardware (if applicable).

4.   Connect the configuration cable between the gateway and a PC.

5.   Connect the power cable and apply power.

6.   Download Anybus Configuration Manager from www.anybus.com/support
      and install it on the PC following the instructions in the installer.

      (Anybus Configuration Manager requires Microsoft® Windows XP or later)

7.   Continue to *Network Configuration, p. 17*

3.2   External Parts
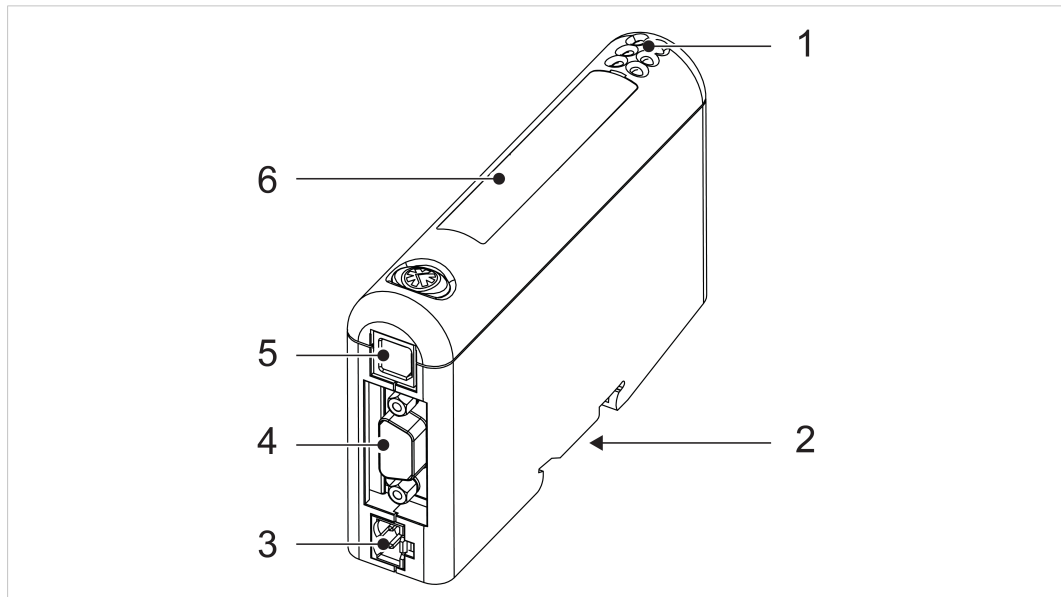
Fig. 9   Overview

1	LED indicators
2	DIN rail mount
3	Power connector
4	Serial subnetwork interface
5	PC connector
6	PROFINET IRT network interface

3.3   DIN Rail Mounting

> **!**   The unit must be electrically grounded through the DIN rail for EMC compliance.

Mount on DIN rail

1.   Hook the unit onto the upper lip of the rail and push gently downwards.

2.   Push the unit towards the rail until it snaps into place.
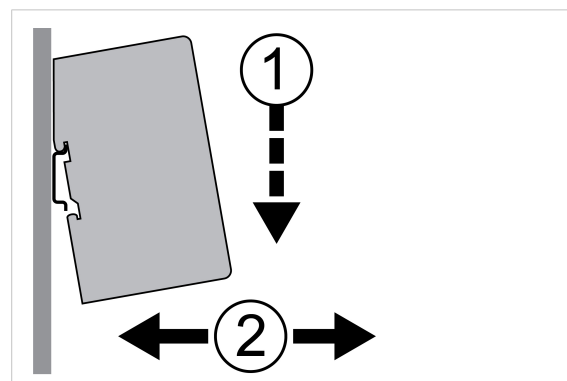
Fig. 10   Push down to mount or remove

Remove from DIN rail

1.   Push the unit gently downwards on the rail.

2.   Pull the bottom end of the unit free of the rail and remove it.

3.4   Serial Subnetwork Interface
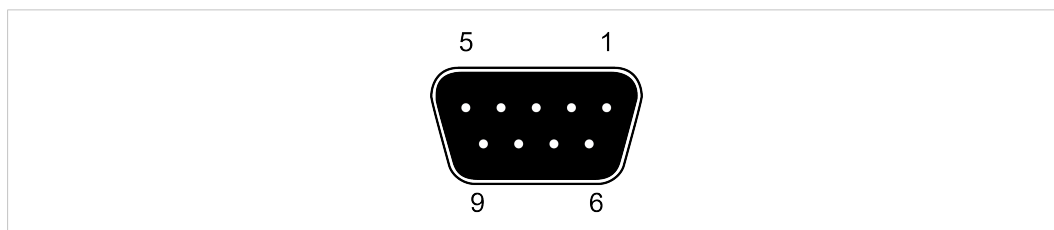
Fig. 11   D-sub connector (DE-9F)

Pin	Description	RS-232	RS-422	RS-485
1	+5 V Output (100 mA max.)	x	x	x
2	RS-232 Rx	x		
3	RS-232 Tx	x		
4	(reserved)			
5	Signal Ground	x	x	x
6	RS-422 Rx +		x	
7	RS-422 Rx -		x	
8	RS-485 + / RS-422 Tx +		x	x
9	RS-485 - / RS-422 Tx -		x	x
Housing	Shield	x	x	x

> **!** Do not connect Signal Ground to Protective Earth (PE) of other nodes on the subnetwork as this may damage the on-board serial transceivers. Connect it only to the Signal Ground on other nodes.

Bias and/or termination resistors may be required depending on the type of serial network. Please refer to the User Manual for more information.

3.5   Termination and Bias Resistors

3.5.1   Termination (RS-485 and RS-422)

The serial subnetwork should be terminated at each end node to prevent reflections on the serial lines. The resistor value should match the characteristic impedance of the cable, typically 100–120 Ω.

3.5.2   Bias Resistors (RS-485 only)

RS-485 enters an indeterminate state when idle, which may cause serial receivers to pick up noise from the serial lines and interpret it as data. To prevent this, the serial lines should be forced into a known state using *bias resistors*.

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

Bias resistors should only be installed on one node. Installing bias resistors on several nodes may compromise signal quality and cause transmission problems.

3.6   PROFINET Interface

The PROFINET IRT interface contains a dual port Ethernet switch with RJ45 type connectors. The two ports are labeled **LAN 1** and **LAN 2**.

Pin	Function
1	TD+
2	TD-
3	RD+
6	RD-
4, 5, 7, 8	(reserved)
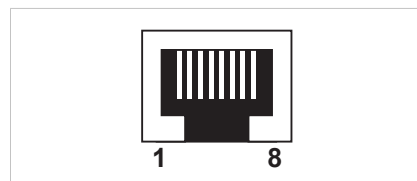
Fig. 12   Ethernet connector (RJ45)

3.7   Power Connector

See also *Technical Data, p. 75* regarding power supply requirements.

Pin	Signal
1	+24 VDC
2	Power Ground
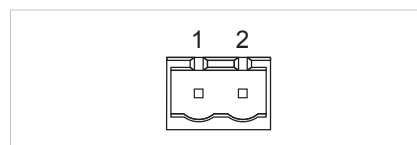
Fig. 13   Power connector

3.8   PC Connector (RJ11)

Pin	Signal
1	GND (signal ground)
2	GND (signal ground)
3	RS-232 Rx (input)
4	RS-232 Tx (output)

Fig. 14   Configuration cable

3.9 LED Indicators

The LED indicators provide model-specific diagnostic information about the communication and status of the network interfaces as well as general device status.
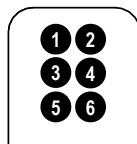
LED 1 to 4	Model-specific information
LED 5	Serial subnetwork status
LED 6	Device status

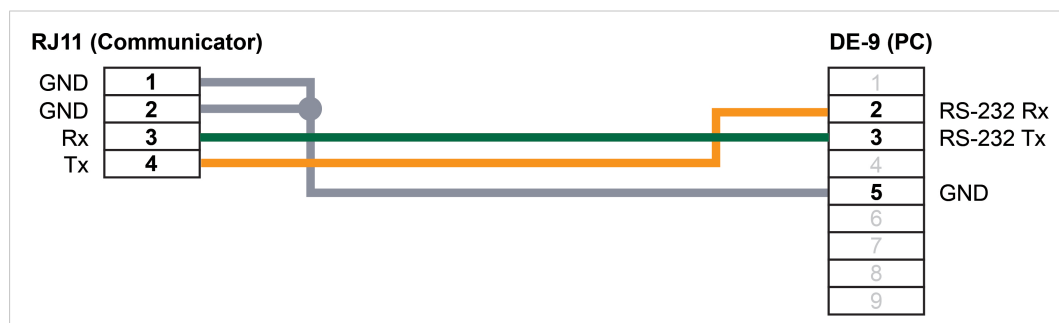| LED	Indication	Meaning
1 - Network Status	Off	Offline – No power – No connection to IO Controller
	Green	Online (RUN) – Connection to IO Controller
	Green, 1 flash	Online (STOP) – Connection to IO Controller – IO Controller in STOP state or IO data bad – RT synchronization not finished
	Red	Fatal error
	Red, 1 flash	Station name error
	Red, 2 flashes	IP address error
	Red, 3 flashes	Configuration error
	Alternating red/green	Firmware update in progress
2 - Module Status	Off	No power or initializing
	Green	Normal operation
	Green, 1 flash	Diagnostic event present
	Red	Fatal error
	Alternating red/green	Firmware update in progress
3 - Link/Activity 1 4 - Link/Activity 2	Off	No power or no link detected
	Green	Link OK
	Green, flickering	Transmitting/receiving data

5 - Subnet Status	Off	No power
	Green	Running
	Green, flashing	Running, one or more transaction errors
	Red	Transaction error/timeout or subnet stopped
6 - Device Status	Off	No power
	Green	Initializing
	Green, flashing	Running
	Red	Bootloader mode
	Alternating red/green	Configuration invalid or missing

LED Indicator Timing Intervals
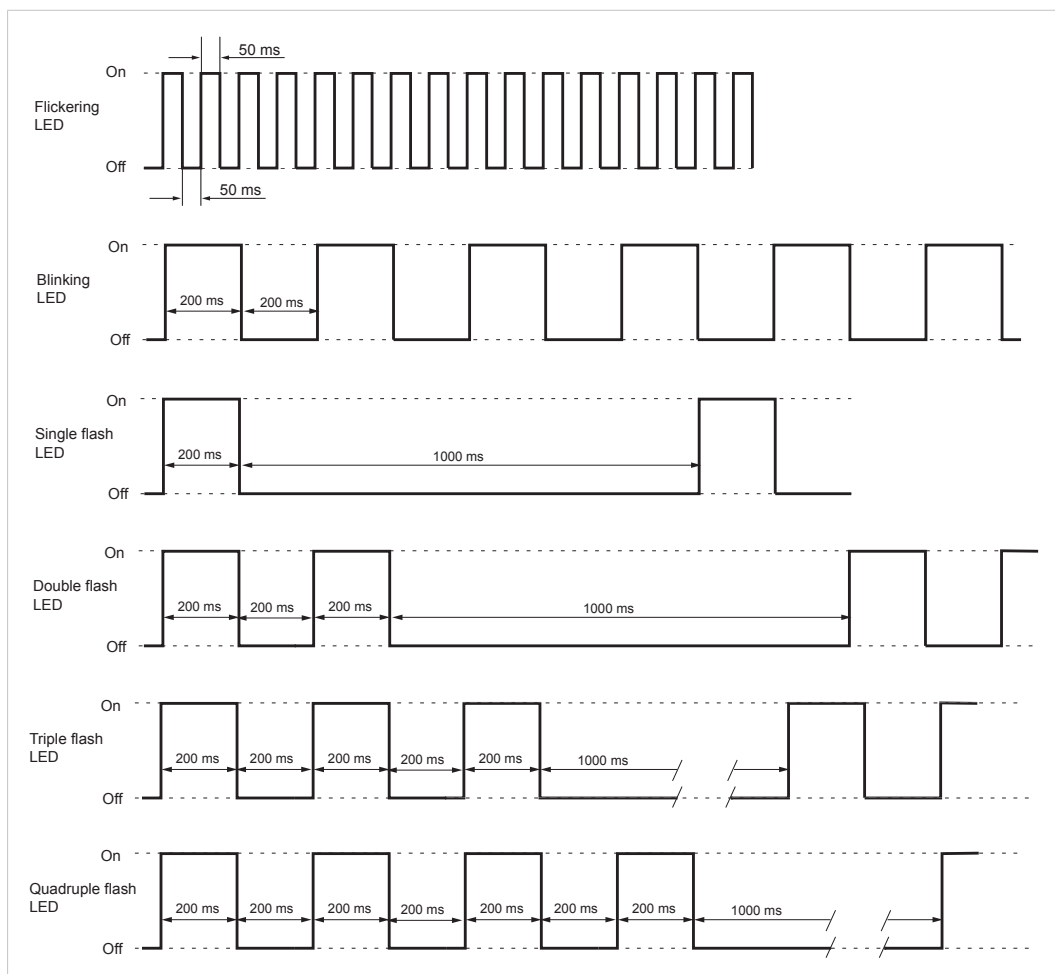
Fig. 15   **LED indicator timing intervals**

4 Network Configuration

4.1 General

To be able to communicate over Ethernet the PROFINET IRT interface needs a valid TCP/IP network configuration. This section explains the basic concepts of TCP/IP networking and describes how to configure the TCP/IP settings for the interface.

TCP/IP Settings

The TCP/IP settings are usually configured from the PROFINET IO Controller but can also be set locally using the IPconfig tool. See *TCP/IP Configuration, p. 19*.

When Ethernet communication has been established, the TCP/IP settings can also be changed from the web interface. See *Web Pages, p. 22*.

DHCP and BootP

The TCP/IP settings can be set automatically from a DHCP or BootP server. If no DHCP server is found, the module will fall back on its current settings. If no current settings are available the module will halt and the status LED will indicate a network configuration error. The network configuration may still be accessed using IPconfig.

DCP (Discovery and Control Protocol)

Anybus Communicator PROFINET IRT (2.32) supports DCP, which allows a PROFINET IO Controller or Supervisor to change the network settings during runtime. The settings applied through DCP will replace the currently stored settings.

4.2 Basic TCP/IP Concepts

IP Address

The IP address is used to identify each node on a TCP/IP network. IP addresses are written as four decimal integers (0–255) separated by dots, where each integer represents the binary value of one byte of the IP address. This is known as *dot-decimal notation*.

Example: `10000000 00001010 00000010 00011110` is written as `128.10.2.30`

The following IP addresses are reserved for special purposes and cannot be used:

0.n.n.n	First byte zero — used for broadcast messages
127.n.n.n	First byte 127 — used for loopback addresses to the local host
n.n.n.0	Last byte zero — identifies a whole network/subnet
n.n.n.255	Last byte 255 — used for broadcast messages

Subnet Mask

The IP address is divided into three parts: *Net ID*, *Subnet ID* and *Host ID*. A subnet mask is a 32-bit binary pattern, where a set bit allocates a bit for Network/Subnet ID, and a cleared bit allocates a bit for the Host ID. The subnet mask is usually written in dot-decimal notation.

Example: To make the IP address `128.10.2.30` belong to subnet `128.10.2`, the subnet mask must be `255.255.255.0`.

```
                                              Host ID
                                            ┌────────┐
Subnet mask:        11111111 11111111 11111111 00000000  (255.255.255.0)
                    └──────────────────────────┘
                          Net ID / Subnet ID
```
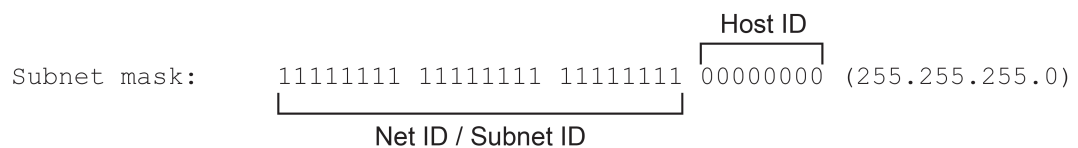
Default Gateway

For devices to be able to communicate over Ethernet they must either belong to the same subnet or communicate via a gateway or router.

A gateway or router routes communication between networks, i.e. it enables the nodes on one network to access the nodes on another. The *default gateway* address in the TCP/IP settings of your product specifies the IP address of the gateway or router on the local network.

4.3   TCP/IP Configuration

4.3.1   Installing the IPconfig Utility

IPconfig is a Windows-based tool for configuration of TCP/IP settings in HMS devices. The tool will detect all compatible and active HMS devices on the local network.

1.   Download IPconfig from www.anybus.com/support.

2.   Unpack the contents of the zip archive and run the installer program.

4.3.2   Scanning for Connected Devices

When IPconfig is started it will automatically scan all available local networks for HMS devices. Detected devices will be listed in the main window. To refresh the list, click on **Scan**.
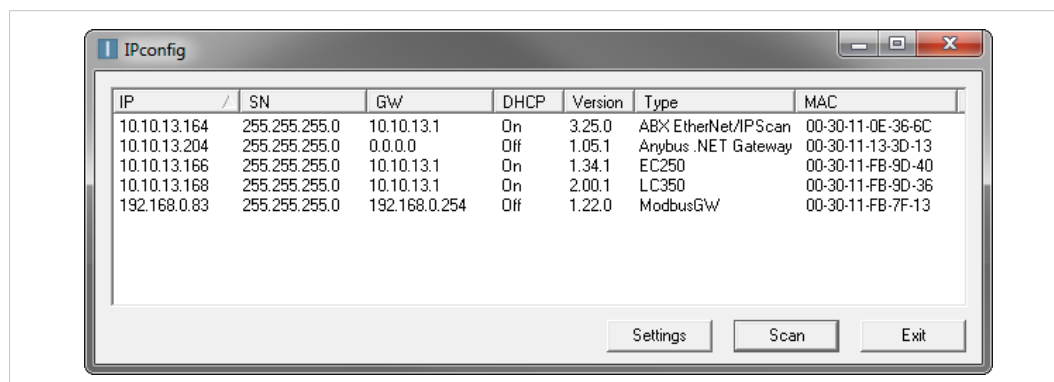
Fig. 16   IPconfig main window

IP	IP address of the device
SN	Subnet mask
GW	Default gateway
DHCP	Automatically managed IP configuration
Version	Firmware version
Type	Product name
MAC	Ethernet MAC address (System ID)

4.3.3   Ethernet Configuration

To change the IP settings for a device, double-click on the entry in the main window or right-click on it and select **Configuration**.
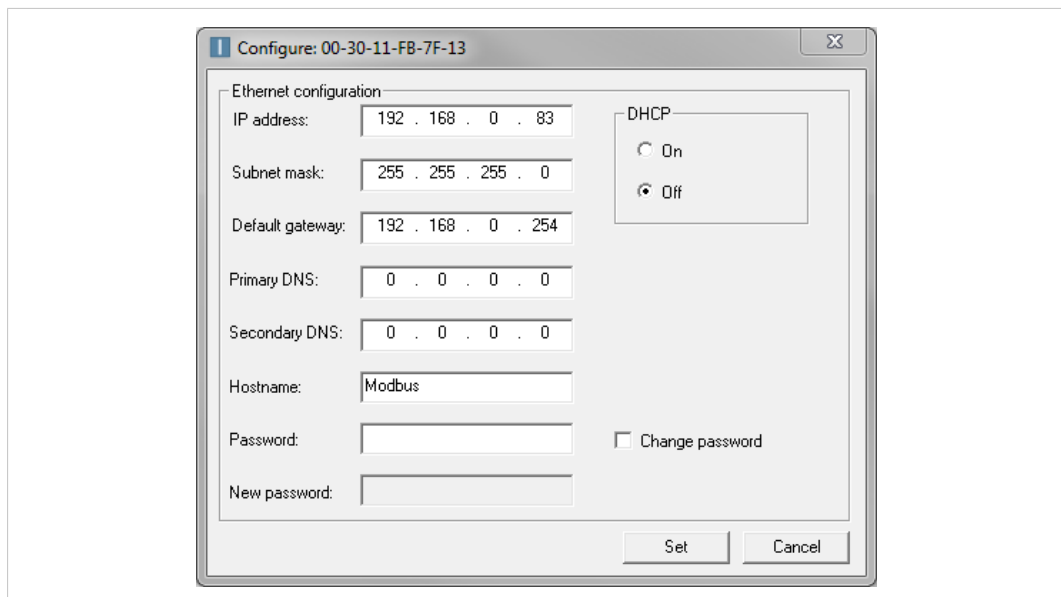
Fig. 17   Ethernet configuration

Enter static IP settings as required, or select DHCP if using dynamic IP addressing.

!	Do not enable DHCP if there is no DHCP server available on the network.

You can add a name for the device in the **Hostname** field. Only characters a–z, A–Z, 0–9 and _ (underscore) are allowed.

The default password for changing IP settings is blank (no password). If a password has been set for the device you must enter it to be able to change the settings.

To set a new password, check the **Change password** box and enter the current password in the **Password** field, then enter the new password in the **New password** field.

!	For security reasons the default password should always be changed.

Click on **Set** to save the new settings. The device will reboot automatically.

4.3.4   IPconfig Settings

Additional settings for IPconfig can be accessed by clicking on **Settings**.
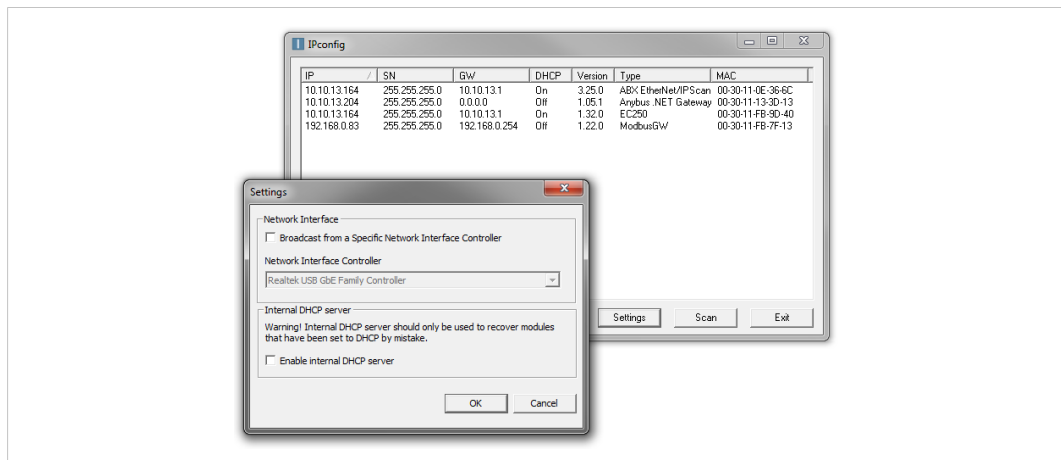
Fig. 18   IPconfig settings

Network Interface

Check this option to select a specific network interface to use when scanning for devices from a computer which has more than one interface. If this option is left unchecked, all available networks will be scanned.

Internal DHCP Server

If a device has been set to use DHCP but there is no DHCP server on the network, the device may not be detected by IPconfig. To recover access to the device an internal DHCP server in IPconfig can be temporarily activated:

1.   Click the checkbox for **Internal DHCP Server**, then click **OK**. IPconfig will automatically refresh the scan and list the missing device in the main window.

2.   Select the device and configure it to use static IP addressing instead of DHCP.

3.   Disable the internal DHCP server.

> ❗ Do not enable the internal DHCP server if there is already an active DHCP server on the network.

4.4   Web Pages

Network configuration settings and status of the PROFINET IRT network interface can be accessed by pointing a web browser to the IP address of the interface.

Module Overview

Fig. 19   Overview tab

Provides basic information about the Anybus Communicator including the serial number and the installed firmware version.
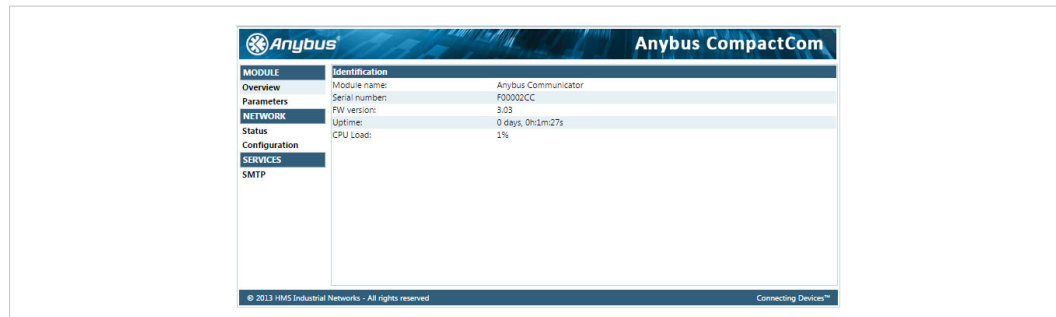
Network Status
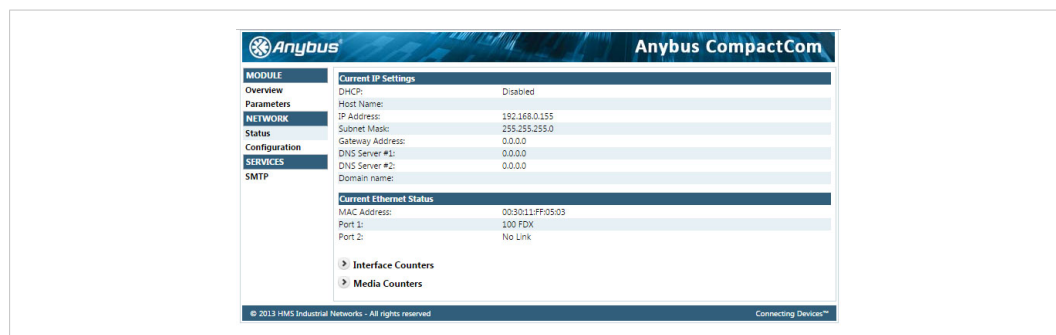

Fig. 20   Status tab

Displays an overview of the current network status.

Network Configuration
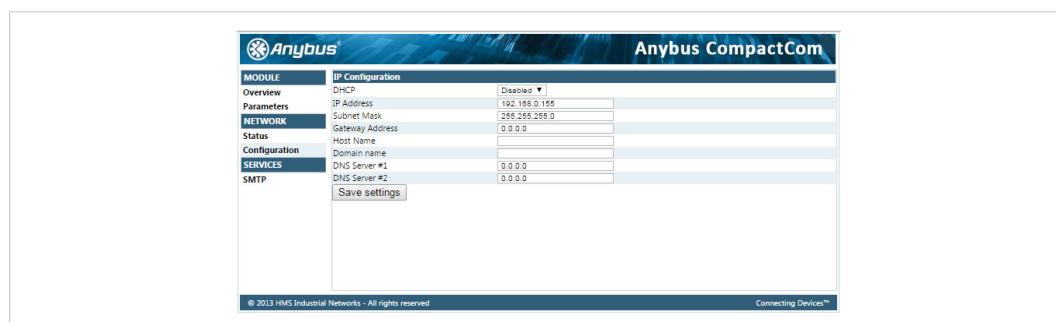
Fig. 21   Configuration tab

Provides access to the TCP/IP network settings. These parameters can also be configured using the *IPconfig* tool.

5 PROFINET Data Exchange

5.1 Overview

PROFINET is the open Industrial Ethernet standard for automation from PROFIBUS and PROFINET International. The PROFINET IRT device provides PROFINET IO Isochronous Real Time Communication.

PROFINET makes a clear distinction between fast cyclical data, *IO Data*, and acyclical data, *Record Data*. PROFINET IO Data corresponds to what is here generally referred to as *I/O Data*. PROFINET Record Data corresponds to what is referred to as *Parameter Data*.

PROFINET IO Data (I/O Data)

PROFINET IO Data is exchanged cyclically and is built up by I/O modules. The actual I/O configuration is determined by the PROFINET IO Controller. The modules are mapped to the Input and Output Buffers in the order of their slot number.

PROFINET Record Data (Parameter Data)

Record Data is exchanged using acyclic Record Data Read/Write requests.

See also *Data Representation (IO Data and Record Data), p. 35*.

5.2 GSD File

All PROFINET devices are associated with an XML-based *GSD* file. This file contains information about the basic capabilities and configuration options of the device.

The latest version of the GSD file for Anybus Communicator PROFINET IRT (2.32) can be downloaded from www.anybus.com/support.

5.3 PROFINET Asset Management

5.3.1 Asset Management Record

With the *asset management record* functionality data about the assets available on a non PROFINET network can be recorded and read out over a PROFINET network.

Together with the *Identification & Maintenance data* functionality an extensive registration of devices and machines is possible, even in facilities where the devices are not installed in the PROFINET environment.

Factory owners and system integrators can collect data about devices installed beyond the *Anybus gateway*.

The recorded data can be used as basis for the design of easier maintenance and operation processes, despite the increasing complexity of processes and associated machines.

5.3.2 Recording and Reading Data

An *asset management* file containing all the *assets* and their corresponding data on the non PROFINET network is created and uploaded via an *FTP* server to the Gateway *file system*.

The *asset management* file can be transferred from a computer connected to a PROFINET network.
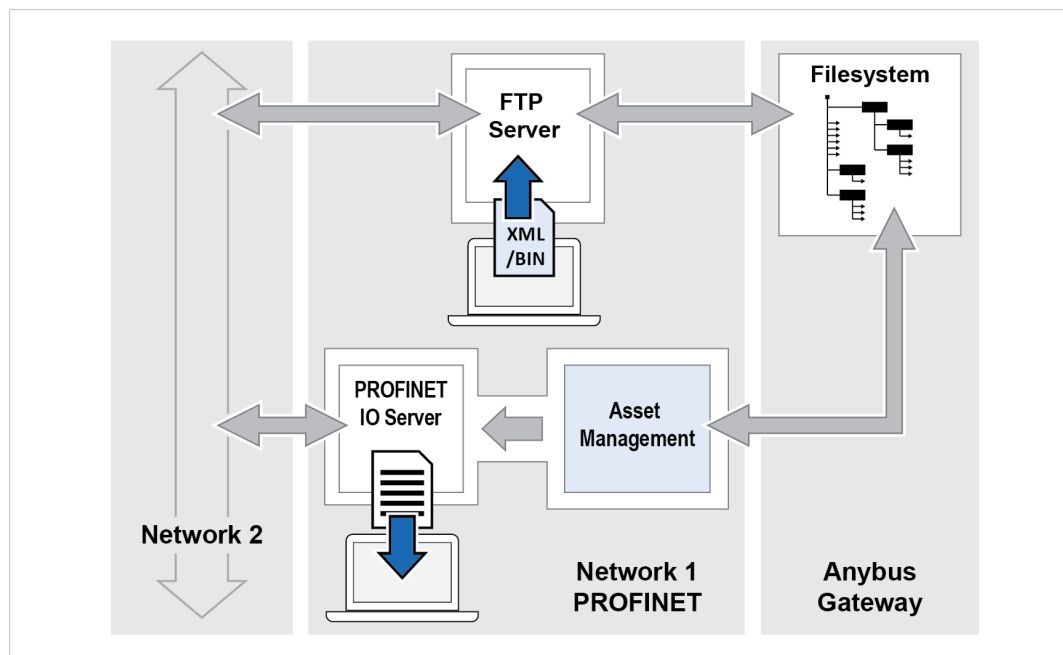
Fig. 22   The Asset Management Default Mode

By using the *superposed parameter channel* mode it is also possible to transfer the *asset management* file from a PLC connected to a non PROFINET network.

For further details about the *superposed parameter channel* mode, please refer to www.anybus.com/support.

Record Data

Data about the *assets* on the non PROFINET network is recorded and stored in an *XML* file or an *binary* file.

Read Data

Each time an *instance* is requested the *asset management* data is read out over the PROFINET network.

The recorded *asset management* data can be downloaded to a computer connected to the PROFINET network.

5.3.3 Supported File Formats

The following file formats are supported for the *asset management* file.

Format	Version
XML	XML Version 1.0
Binary file Little-endian	N/A

5.3.4 Supported Asset Management Records

Supported *asset management records*:

- Unique ID
- Location
- Hardware Revision
- Annotation
- Order ID
- Serial Number
- Software Revision
- Serial Number

5.3.5   XML Based Asset Management

Creating the Asset Management XML File

Creating the *asset management* XML file:

1. List all assets and their corresponding data on the non PROFINET network.

2. Create an XML file that include one *asset management record* for each asset.

    Repeat all the *attributes* after each other.

3. When all *attributes* are listed, close the *element* by using a *closing entry*.

4. Name the XML file *asset_mgmt*.

XML File Size Limitation

ⓘ   *The size of the asset management file may not exceed 95 kb.*

Up to 32 *instances* can be added.

In order to keep the file size small, consider the following:

• Keep strings as short as possible.

• Do not pad with empty spaces for strings.

• Try to use as few spaces as possible for indentation in the file.

• The number of white-space also affects the file size.

• Avoid using *optional name strings*.

XML Attribute Name and Data Format

ⓘ   *The order of the elements is significant for the XML schema to work with the Anybus Gateways.*

   If the XML schema is incorrect, the XML file will not work and no data will be recorded.

When creating the XML file, add the *elements* and their *attributes* in the same order as the *attribute names* are listed in the table below.

Each *element* consists of a series of *attributes* and their various data.

Each *attribute* is described by one *entry*.

The supported *attribute names* are specified in the table.

Example 1: XML *element* including an *attribute* with the *location* record.

```
<AbccAttribute>
<Name Value="Location Type"/>
<Attribute Value="3"/>
<Data Value="1"/>
</AbccAttribute>
```

Attribute Name and Data Format

Attribute Name	Data Format	Description
AM info Type Location Type	Unsigned 8	The value can be set in either of two formats, `0x12` or `18`.
AM Type Identification IM Hardware Revision	Unsigned 16	The value can be set in either of two formats, `0x1234` or `4660`.
IM Annotation	String of length `X`	Maximum number of elements in array: 64.
IM Order ID	String of length `X`	Maximum number of elements in array: 64.
IM Serial Number	String of length `X`	Maximum number of elements in array: 16.
AM Software Revision	String of length `X`	Maximum number of elements in array: 64.
AM Hardware Revision	String of length `X`	Maximum number of elements in array: 64.
IM Software Revision	String	Format of the string shall be `C.X.Y.Z`. `C` is one character. `X`, `Y` and `Z` represent a value between 0 and 255. `X` – Major version `Y` – Minor version `Z` – Internal
IM Unique Identifier	Array of Unsigned 8 Length is 16	Format of the value shall be `0xXX;0xYY…0xZZ`. 16 values in hex-format, where each value is separated by a ";".
Location LT	Array of Unsigned 16 Length is up to 12 elements.	Format of the value shall be `0xXXXX;0xYYYY…0xZZZZ`. Up to 12 values in hex-format, where each value is separated by a ";".
Location SS AM Device Identification	Array of Unsigned 16 Length is 4.	Format of the value shall be `0xXXXX;0xYYYY…0xZZZZ`. 4 values in hex-format, where each value is separated by a ";".

Asset Management XML File Structure Example

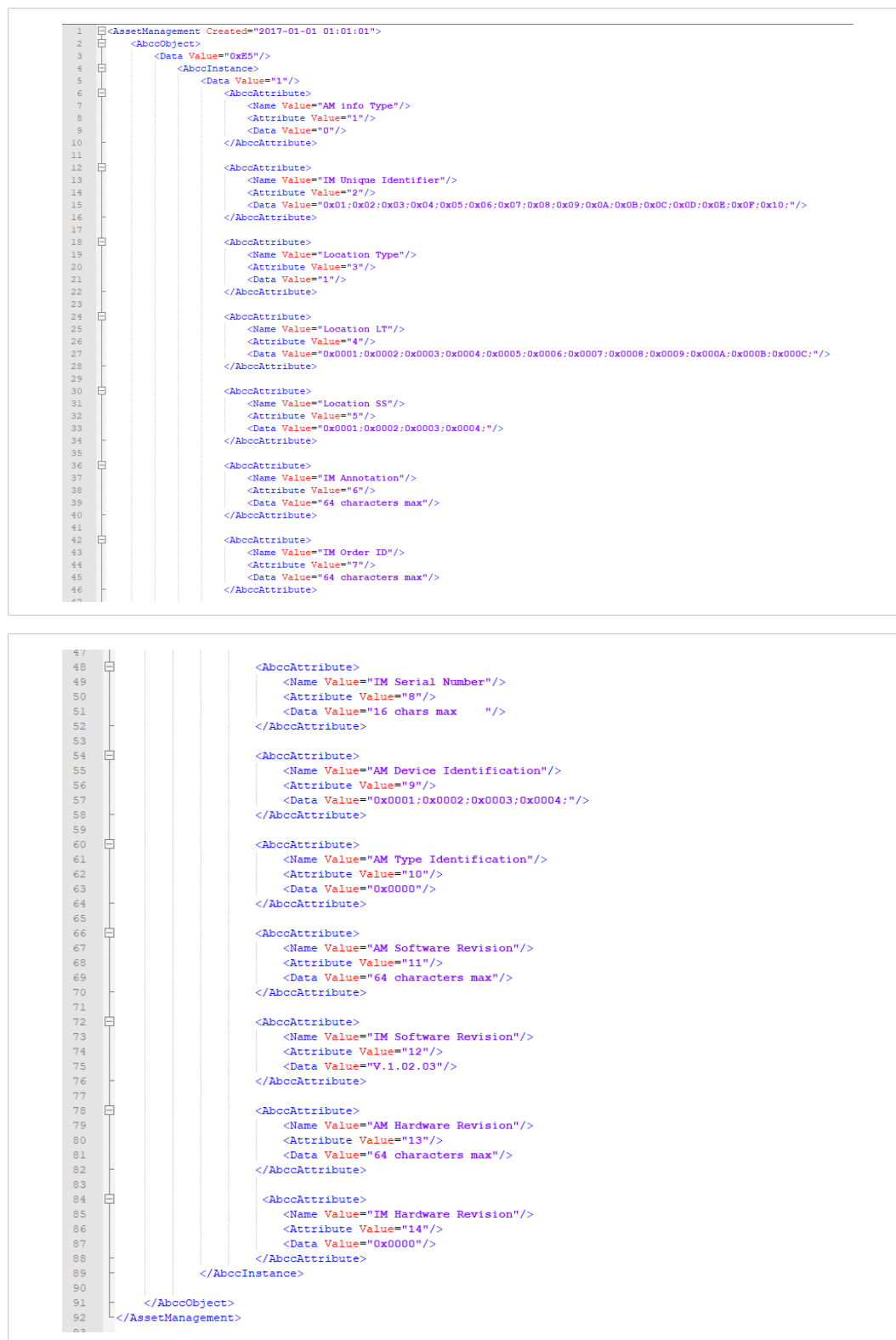The code example presented below can be used as a guide when creating the *asset management* XML file.

Fig. 23   **Asset management XML file structure example**

5.3.6   Binary Based Asset Management

Creating the Asset Management Binary File

Creating the *asset management* binary file:

1.   List all assets and their corresponding data on the non PROFINET network.

2.   Create an Binary file that include a *asset management record* for each asset.

       Repeat all the *attributes* after each other.

3.   When all *attributes* are listed, close the *element* by using a *closing entry*.

4.   Name the bin file *asset_mgmt*.

Binary File Size Limitation

> **(i)**   *The size of the asset management file may not exceed 12 kb.*

> **(i)**   *32 instances can be added, instance 1 to 32.*

In order to keep the file size small, consider the following:

•   Keep strings as short as possible.

•   Do not pad with empty spaces for strings.

Binary File Header

> **(i)**   *Omitted attributes are disabled or set to their default value.*

> **(i)**   *The size of the file header is 70 bytes.*

The supported *file headers* are specified in the table.

Supported File Headers			
File Header	**Byte Number**	**Data Type**	**Comment**
File format version	0-1	UINT16	Version number of the file format. Set to 0.
File checksum	2-5	UINT32	Used for version control of the file. Not used by the gateway. If not used, the field must be set to zero.
Byte offset to Instance 1	6-7	UINT16	Byte offset to the start of the data describing Asset management Instance X. Set to zero if instance is not used.
Byte offset to Instance 2	8-9		
Byte offset to Instance 32	68-69		
Instance data	70-x	N/A	Data for the instance(s), as specified below.

Binary Instance Data

Each *instance* consists of a series of *attributes* and their respective data.

Attribute Description

Each *attribute* is described by one entry.

Attribute Description	Byte number	Data type	Comment
Attribute number	0	UINT8	Attribute number of the data being described.
Data length	1	UINT8	Optional checksum. Shall represent the number of data bytes following. Not used by the gateway.
Attribute data	2–x	Depends on the attribute being described.	Data for the attribute. Format shall be as described for the data-type. Not needed for strings padding or termination.

Attribute Closure Description

Use a *closing entry* to close the instance data.

Attribute Description	Byte number	Data type	Comment
Closure	0–1	UINT16	Data-field which tell that there will not follow any more attributes for this instance. Set to value 0xFFFF.

Attribute Name and Data Format

Supported *attribute names* and *data formats*.

Attribute Name and Data Format		
Attribute Name	**Data Format**	**Description**
AM info Type Location Type	Unsigned 8	The value is set as one byte value.
AM Type Identification IM Hardware Revision	Unsigned 16	The value is set with two bytes, *little-endian* format.
IM Annotation	String of length X	Maximum number of elements in array: 64.
IM Order ID	String of length X	Maximum number of elements in array: 64.
IM Serial Number	String of length X	Maximum number of elements in array: 16.
AM Software Revision	String of length X	Maximum number of elements in array: 64.
AM Hardware Revision	String of length X	Maximum number of elements in array: 64
IM Software Revision	Array of Unsigned 8 Length is 4	First byte is a character. Bytes 2, 3 and 4 represent the version in the format X.Y.Z where X, Y and Z represent a value between 0 and 255. C is one character. X, Y and Z represent a value between 0 and 255. X – Major version Y – Minor version Z – Internal
IM Unique Identifier	Array of Unsigned 8 Length is 16	Format is 16 bytes.
Location LT	Array of Unsigned 16 Length is up to 12 elements.	Each Unsigned 16 comprises two bytes, where each two bytes form an Unsigned 16 in *little-endian* format. The number of Unsigned 16's can be up to 12, placed directly after each other
Location SS AM Device Identification	Array of Unsigned 16 Length is 4.	Each Unsigned 16 comprises two bytes, where each two bytes form an Unsigned 16 in *little-endian* format. The number of Unsigned 16's shall be 4, placed directly after each other.

Asset Management Binary File Example

The binary file structure example presented below can be used as a guide when creating the *asset management* binary file.

Only *instance 1* is supported.

For *instance 1*, only attribute 1 and 2 are defined.

	0	1	2	3	4	5	6	7
0	0x00	0x00	0x01	0x02	0x03	0x04	0x46	0x00
8	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
16	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
24	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
32	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
40	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
48	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
56	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
64	0x00	0x00	0x00	0x00	0x00	0x00	0x01	0x01
72	0x01	0x02	0x10	0x01	0x02	0x03	0x04	0x05
80	0x06	0x07	0x08	0x09	0x0A	0x0B	0x0C	0x0D
88	0x0E	0x0F	0x10	0xFF	0xFF			

Fig. 24   Binary file example

5.3.7 Uploading the Asset Management File to the FTP Server

Use *Windows Explorer* or a standard *FTP client* to transfer the *asset management file* to the *FTP server*.

When the *superposed parameter channel* function is enabled, transfer the *asset management file* via a PLC connected to the network where the gateway is installed.

Transferring the Asset Management File from Windows Explorer

Transfer the *asset management file*, XML or binary file, to the *FTP server* using *Windows Explorer*.

Before You Begin

> (i) *Use only one of the file formats, XML format or binary format.*

> (i) *Only upload one single file on the FTP server.*

- Name the *asset management file*: *asset_mgmt*
- The default port is FTP port 21.
- Make sure that the gateway and your computer are connected to the PROFINET network to be used.
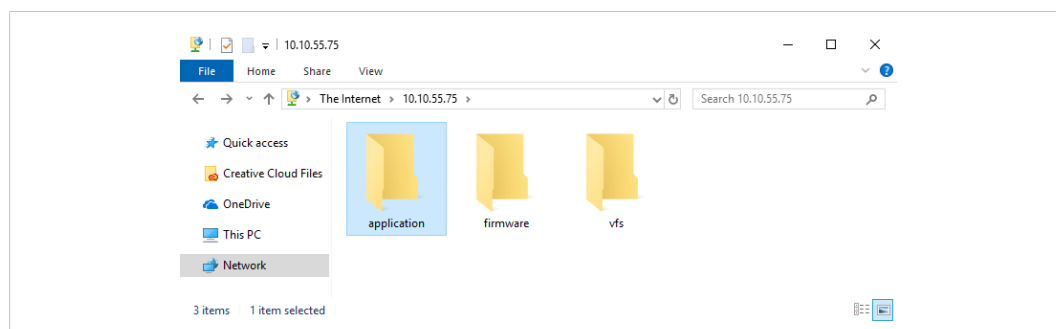
Procedure

Fig. 25   **The FTP Server root folder**

1. Open an **Windows Explorer Window**.

2. Click to select the Address bar.

3. Enter **ftp://Username:Password@IPaddress**.

   – Replace "Username" and "Password" with a valid username and password combination.

   – Replace 'IPaddress' with the IP address of the PROFINET interface.
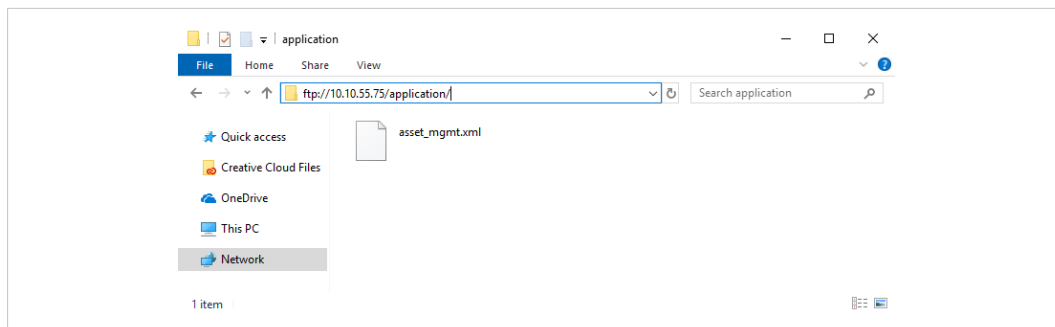
4. Press **Enter**.

Fig. 26   **Application folder with an asset_mgmt.xml file**

5.   Open the *application* folder and save the *asset management file*, XML or Binary file, in the folder.

5.4   Data Representation (IO Data and Record Data)

The actual I/O configuration is determined by the PROFINET IO Controller. The modules are mapped to the Input and Output Buffers in the order of their slot number.

Example:

In this example, the data sizes have been set to the following values:

Input I/O Data Size:	208 bytes
Input Parameter Data Size:	304 bytes
Output I/O Data Size:	176 bytes
Output Parameter Data Size:	336 bytes

The following modules are specified in the IO Controller:

Slot	Module Size	Direction	Comment
0	0	-	Device Access Point (DAP)
1	176 bytes	Output	-
2	208 bytes	Input	-
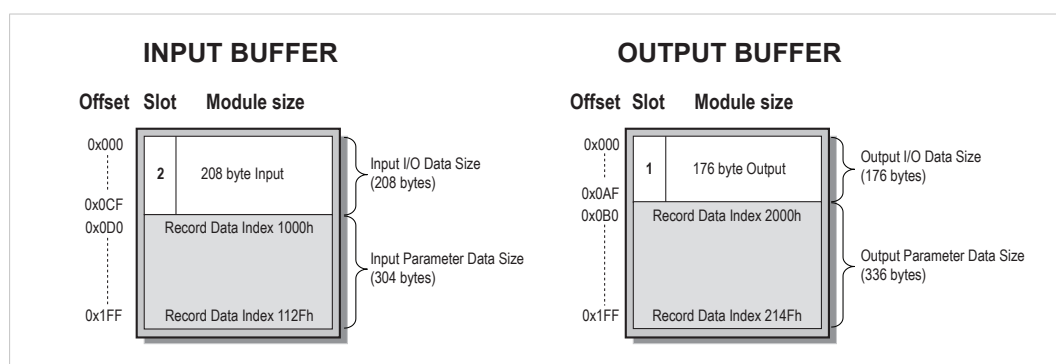
Fig. 27   Input and output buffers

Input Parameter Data

API	Slot	Subslot	Index	Comment
0	0/1/2	1	1000h	First byte of the Input parameter area (address 0x0D0 in the example above)
			1001h	Second byte of the Input parameter area
			…	
			112Fh	Last byte of the Input parameter area

Output Parameter Data

API	Slot	Subslot	Index	Comment
0	0/1/2	1	2000h	First byte of the Output parameter area (address 0x0B0 in the example above)
			2001h	Second byte of the Output parameter area
			…	
			214Fh	Last byte of the Output parameter area

ⓘ   *The Control Word and Status Word and the Live List are not considered in this example.*

6   Anybus Configuration Manager
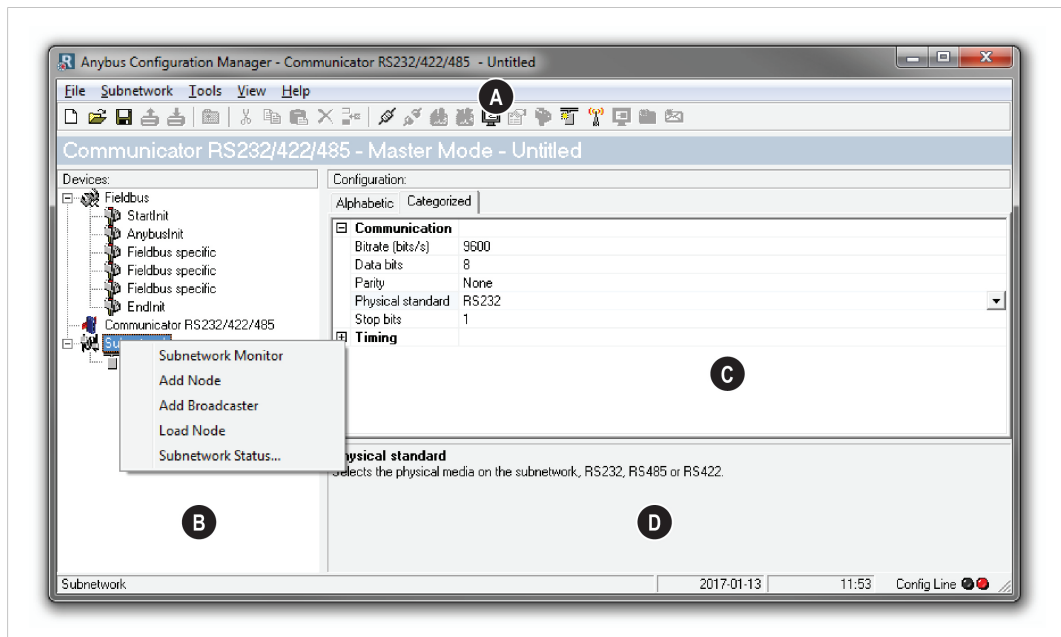
6.1   Overview

Fig. 28   Anybus Configuration Manager

A: Menus and Toolbar

The function of the second menu (after the **File** menu) will change depending on what is currently selected in the navigation tree.

The toolbar contains shortcut buttons to the most common commands. Each button has a tooltip describing its function.

B: Navigation Tree

A hierarchic tree view of the configuration, divided into three main sections:

Fieldbus	Communication settings for the fieldbus network interface
Communicator RS232/422/485	General settings for the Anybus Communicator
Subnetwork	Communication settings for the serial subnetwork

Select an entry to display its available parameters in the Parameter List. Right-click on the entry to show additional options.

C: Parameter List

Lists the parameters or options related to the currently selected entry in the Navigation Tree.

Values can be selected from a dropdown menu or entered manually depending on the parameter. Values can be specified in either decimal or hexadecimal format.

Example: The decimal value `42` can also be entered as `0x2A`.

D: Information Section

Displays information about the currently selected parameter.

6.1.1 File Menu

New	Create a new configuration.
Open...	Open a previously saved configuration.
Save	Save the current configuration.
Save as...	Save the current configuration under a different file name.
Print...	Print the current configuration.
Properties...	Set the name and (optional) passwords for the configuration. **Lost passwords cannot be retrieved!**
Exit	Close Anybus Configuration Manager.

6.1.2 Tools Menu

Port	Select the COM port to use for configuration.
Upload configuration from ...	Fetch the active configuration from the Anybus Communicator .
Download configuration to ...	Send the current configuration from Anybus Configuration Manager to the Anybus Communicator.
Start (Stop) Logging	Start/stop the Data Logger function.
Options	See *Options Dialog, p. 38*.

6.1.3 View Menu

Toolbar	Show/hide the toolbar at the top of the main window.
Status Bar	Show/hide the status bar at the bottom of the main window.

6.1.4 Help Menu

About	Shows general information about Anybus Configuration Manager and the currently connected Anybus Communicator.

A help system is not included in this version of Anybus Configuration Manager.
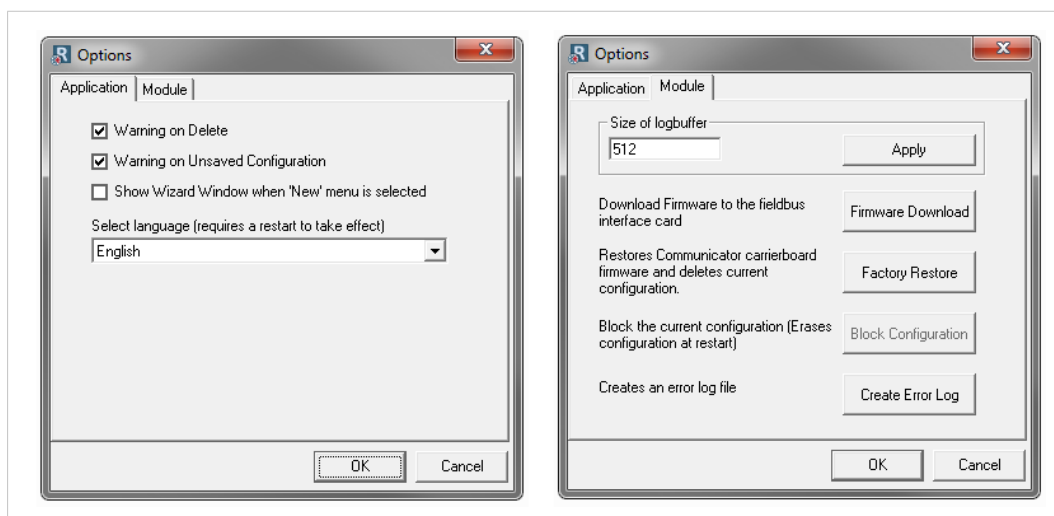
6.1.5   Options Dialog

Fig. 29   Options dialog tabs

Application	
Warning on Delete	A confirmation dialog will be displayed when something is deleted.
Warning on Unsaved Configuration	A confirmation dialog is displayed when closing Anybus Configuration Manager without saving the configuration.
Show Wizard ...	Open a wizard when creating a new configuration
Select Language	The language to use in Anybus Configuration Manager. The program must be restarted for the new language seting to become active.

Module	
Size of logbuffer	The number of entries logged in each direction by the Data Logger. Can be set between 0 (no logging) and 512 (default).
Firmware Download	Used for updating the firmware in the PROFINET IRT interface. **Use with caution!**
Factory Restore	Resets the Anybus Communicator to the factory default settings. The settings in the PROFINET IRT interface are **not** affected.
Block Configuration	Prevents the downloaded configuration from being used by the Anybus Communicator. **Use with caution!**
Create Error Log	Create an error log file for troubleshooting.

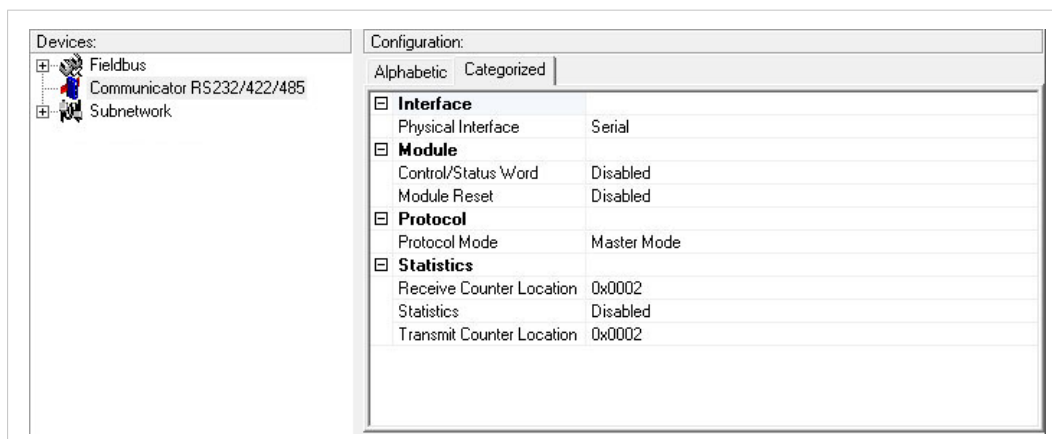## 6.2   Communicator Settings

Fig. 30   Communicator parameters

Interface

The physical interface type of the subnetwork. Cannot be changed from **Serial**.

Control/Status Word

Enabled	Enables the Control and Status Registers. The *Data Valid* bit in the Control Register must be set to start the subnetwork communication.
Enabled but no startup lock	Same as Enabled, but the *Data Valid* bit does not have to be set.
Disabled	Control and Status Registers will not be used.

Module Reset

Specifies how the gateway behaves in the event of a fatal error.

Enabled	The gateway will be restarted, no error indication.
Disabled	The gateway will halt and indicate an error.

Protocol Mode

Select the protocol mode for the subnetwork. See also *Subnetwork Protocol, p. 9*.

Master Mode	Intended for Query/Response based protocols, where a single Master exchanges data with a number of Slaves.
Generic Data Mode	Intended for Producer/Consumer based protocols, where there is no master-slave relationship between the subnetwork nodes and the gateway.
DF1 Master	Intended for the DF1 protocol. The gateway can only be configured as a Master with half-duplex communication.

Statistics

The Transmit and Receive Counters indicate the number of successful transactions on the network. This parameter is primarily intended for debugging purposes.

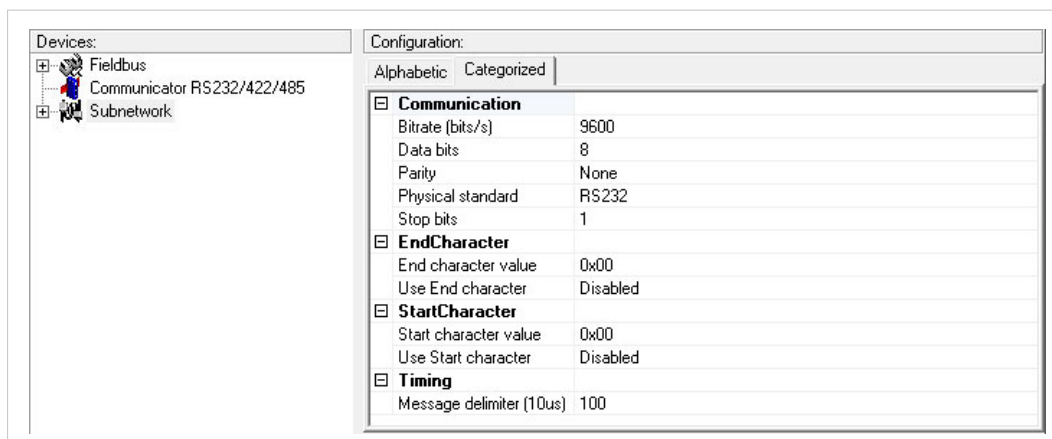|	
Receive Counter Location	Specifies the location of the Receive Counter in the internal memory buffer.
Transmit Counter Location	Specifies the location of the Transmit Counter in the internal memory buffer.
Statistics	Enable/disable the counters.

6.3   Subnetwork Settings

Fig. 31   Subnetwork parameters

Communication

The values that can be selected depend on the selected protocol mode.

Parameter	Description	Possible values [a]
Bitrate (bits/s)	Selects the bit rate	1200 to 57600
Data bits	Selects the number of data bits	7, 8 (8)
Parity	Selects the parity mode	None, Odd, Even
Physical standard	Selects the interface type	RS232, RS422, RS485
Stop bits	Selects the number of stop bits	0, 1 (1)

a.   Possible values in DF1 Master mode are given in parenthesis.

EndCharacter/StartCharacter

This parameter group is only available in Generic Data Mode.

Start and End Characters are used to indicate the beginning and end of a serial message.

Parameter	Description	Possible values
End character value	End character ASCII code	0x00 to 0xFF
Use End character	Enable or disable use of the End character	Enabled/disabled
Start character value	Start character ASCII code	0x00 to 0xFF
Use Start character	Enable or disable use of the Start character	Enabled/disabled

Example: A message should be initiated with `<ESC>` and terminated with `<LF>`. The Start Character should then be `0x1B` (ASCII code for `<ESC>`) and the End Character should be `0x0A` (ASCII code for `<LF>`)

Timing – Message delimiter

This parameter is available in Master Mode and Generic Data Mode.

- **Master Mode**

  Message delimiter specifies the time that separates two messages in steps of **10 ms**.

  If set to 0 (zero), the standard Modbus delimiter of 3.5 characters will be used. The actual time in ms will then be calculated automatically based on the communication settings.

- **Generic Data Mode**

  Message delimiter specifies the time separating two messages in steps of **10 µs**.

DF1 Settings

Fig. 32   Subnetwork parameters (DF1 Master mode)

This parameter group is only available in DF1 Master mode.

Parameter	Description	Default value
Master Node Address	Node address of the master. Valid values: 0 to 254	1
Poll time, active slaves (10 ms)	How often active slaves should be polled. Entered in steps of 10 ms	10 (= 100 ms)
Poll time, inactive slaves (10 ms)	How often inactive slaves should be polled. Entered in steps of 10 ms	100 (= 1000 ms)

In the parameter window the poll time value is displayed in steps of 10 ms, which means that the displayed value is a tenth of the actual time in ms.

Incrementing the value by 1 will thus result in a change of 10 ms.

6.4 Nodes

6.4.1 General

A node in Anybus Configuration Manager 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 have been defined in the configuration.

A maximum of 31 nodes can be created in Anybus Configuration Manager.
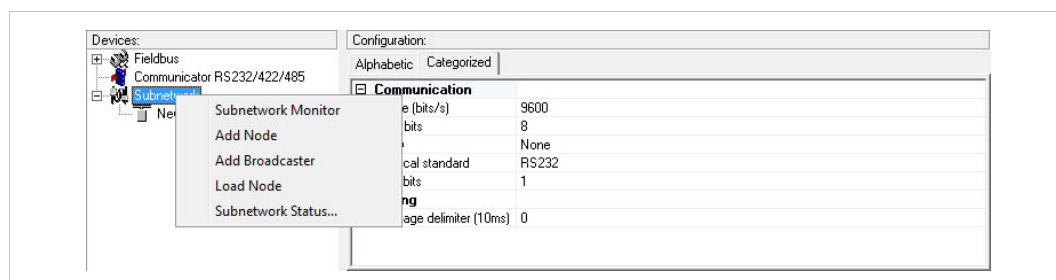
6.4.2 Adding and Managing Nodes

Fig. 33   Adding nodes

| Subnetwork Context Menu (right click)	
Menu Command	**Description**
Paste	Paste a node from the clipboard
Subnetwork Monitor	Open the Subnetwork Monitor. See *Subnetwork Monitor, p. 64*.
Add Node	Add a node to the configuration
Add Broadcaster	Add a broadcaster node to the configuration (Master mode only)
Load Node	Add a previously saved node
Subnetwork Status...	View diagnostic information about the subnetwork

6.4.3 Node Parameters
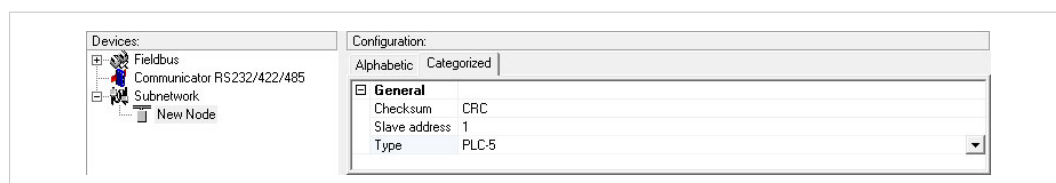
Fig. 34   Node parameters

Master Mode and Generic Data Mode	
Parameter	**Description**
Slave address	This value may be used to set the node address in certain commands. See *Commands (Master Mode & Generic Data Mode), p. 56*.

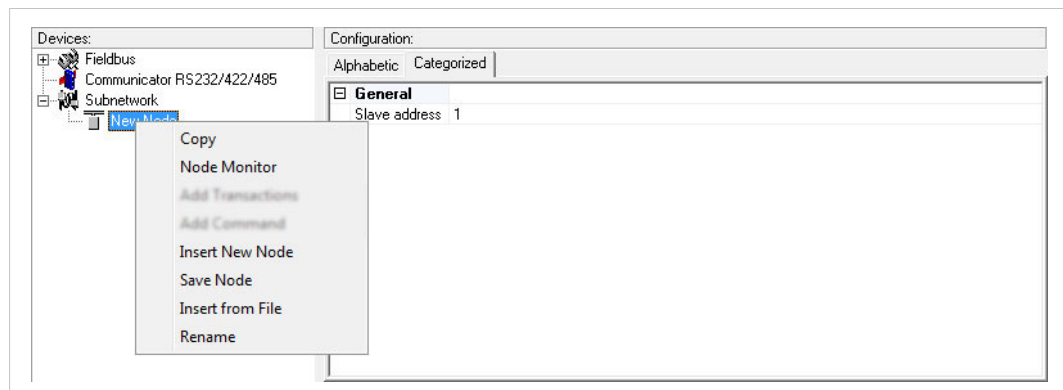| DF1 Master Mode		
Parameter	**Description**	**Possible values**
Checksum	The type of checksum used	BCC, CRC (default)
Slave address	Sets the node address	0 to 254
Type	The PLC type of the slave	PLC-5, SLC-500, Micrologix

6.4.4   Node Context Menu

Fig. 35   Node context menu

Right-click on a node to open a context menu with additional commands for managing or adding nodes.

Node Context Menu (right click)

Menu Command	Description
Copy	Copy the selected node to the clipboard
Delete	Delete the selected node (only available when there is more than one node)
Node Monitor	Open the Node Monitor. See *Node Monitor, p. 65*.
Insert New Node	Insert a new node above the selected node
Save Node	Save the selected node
Insert from File	Insert a previously saved node above the selected node
Rename	Rename the selected node

6.5 Transactions (Master Mode and Generic Data Mode)

6.5.1 General

Transactions in Anybus Configuration Manager are representations of the actual serial telegrams exchanged on the serial subnetwork. 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 have been defined in the configuration.

Transactions are only available in Master Mode and Generic Data Mode.

- **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 subnetwork. The Broadcaster can only send transactions.

- **Generic Data Mode**

  Transactions can be added as desired for both directions. Transactions sent to the subnetwork are called *Transaction Produce*, and transactions issued by other nodes are called *Transaction Consume*.

The gateway can in theory support up to 150 transactions. The actual number of transactions that can be defined depends on their respective memory requirements, and may therefore be significantly less.
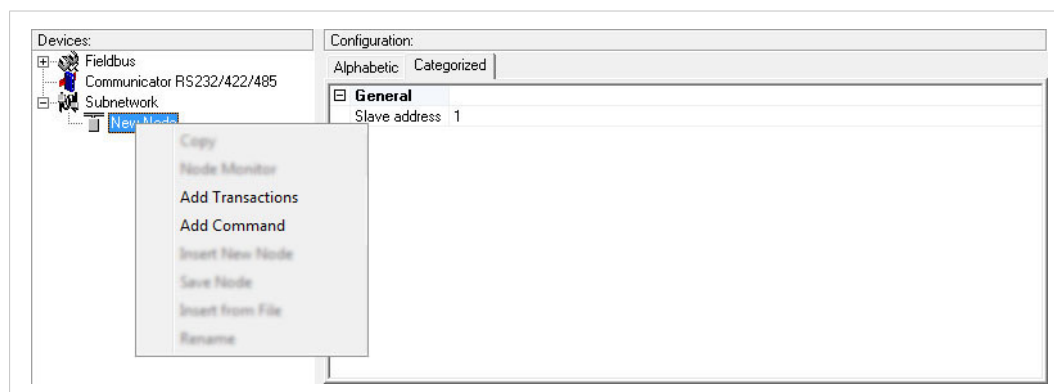
6.5.2 Adding and Managing Transactions

Fig. 36 Node context menu

Right-click on a node to open a context menu with additional commands for adding transactions and commands.

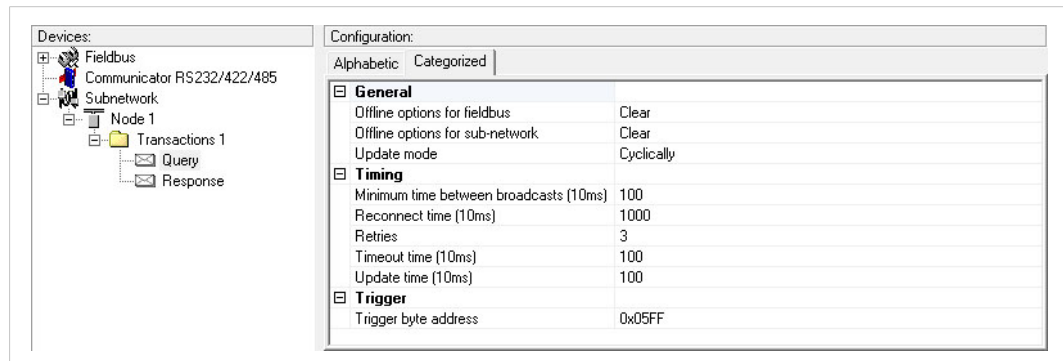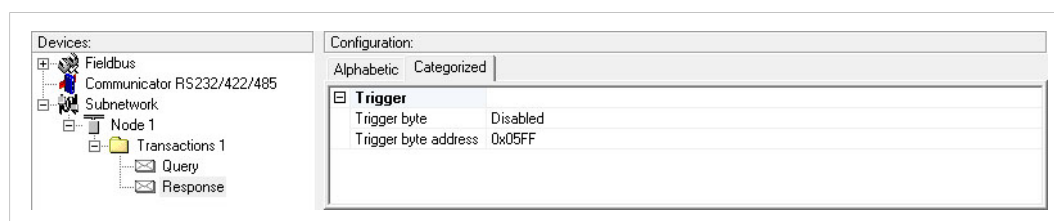| Node Context Menu (right click)	
Menu Command	**Description**
Add Transactions (Master Mode)	On regular nodes: add a Query and a Response (grouped) On the Broadcaster: add a single transaction
Add Transaction Consume (Generic Data Mode)	Add a Consume type transaction
Add Transaction Produce (Generic Data Mode)	Add a Produce type transaction
Add Command	Add a predefined transaction

6.5.3 Query/Broadcast Transactions (Master Mode)

Fig. 37   Transaction parameters

Query/Broadcast Transaction Parameters

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

Query/Broadcast Transaction Parameters (continued)

Parameter	Description
Update mode	Specifies when the transaction shall be sent to the slave.
	Cyclically = The transaction is issued cyclically at the interval specified in the *Update time* parameter.
	On data change = The data area is polled for changes at the time interval defined by the *Update time* parameter. A transaction is issued when a change in data is detected.
	Single shot = The transaction is issued once at start up.
	Change of state on trigger = The transaction 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 1.
	The trigger byte is checked at the interval specified in *Update time*.
	The location of the trigger byte is specified in *Trigger byte address*.
Update time (10 ms)	How often the transaction will be issued in steps of 10 ms.
	Relevant only when *Update mode* is set to *Cyclically*, *On data change* or *Change of state on trigger*.
	The entered value is multiplied by 10. A value of 5 means 50 ms.

6.5.4   Response Transactions (Master Mode)

Fig. 38   Parameters

Response Transaction Parameters

Parameter	Description
Trigger byte	Enables/disables the trigger byte function for the response.
	If enabled, the gateway will increment the trigger byte by 1 when it receives new data from the subnetwork. This can be used to notify the control system of updated data.
Trigger byte address	The location of the trigger byte in the internal memory buffer.
	Valid settings range from 0x000 to 0x1FF and 0x400 to 0xFFF

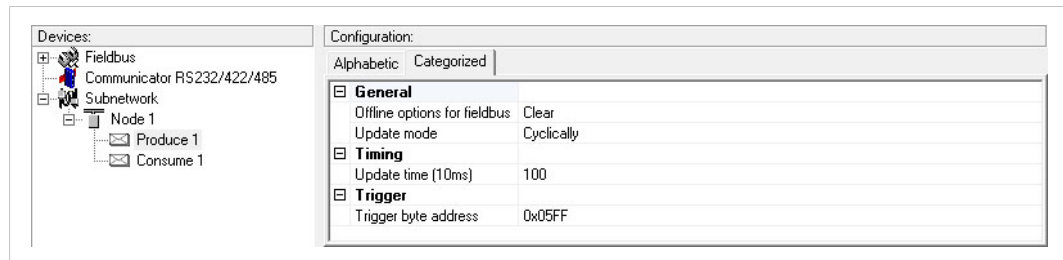## 6.5.5   Produce Transactions (Generic Data Mode)

Fig. 39   Parameters

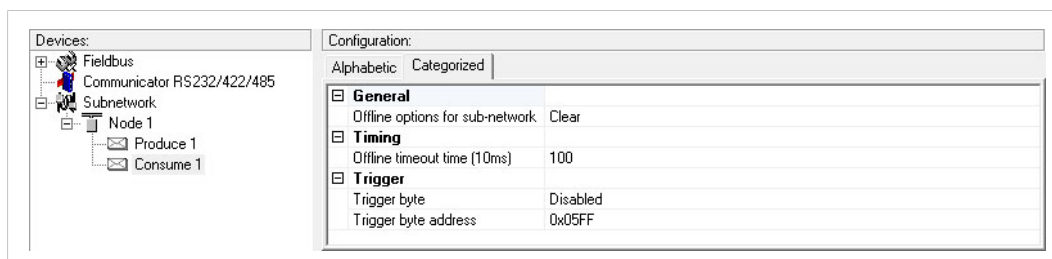| Parameter	Description
Offline options for fieldbus	The action to take for this transaction if the higher level network goes offline. This affects the data that is sent to the subnetwork. **Clear** = The data destined for the slave devices is cleared (set to zero) **Freeze** = The data destined for the slave device is frozen **NoScanning** = The updating of the subnetwork is stopped
Update mode	Specifies when the transaction shall be sent to the slave. **Cyclically** = The transaction is issued cyclically at the interval specified in the *Update time* parameter. **On data change** = The data area is polled for changes at the time interval defined by the *Update time* parameter. A transaction is issued when a change in data is detected. **Single shot** = The transaction is issued once at start up. **Change of state on trigger** = The transaction 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 1. The trigger byte is checked at the interval specified in *Update time*. The location of the trigger byte is specified in *Trigger byte address*.
Update time (10 ms)	How often the transaction will be issued in steps of 10 ms. Relevant only when *Update mode* is set to *Cyclically*, *On data change* or *Change of state on trigger*. The entered value is multiplied by 10. A value of 5 means 50 ms.
Trigger byte address	The location of the trigger byte in the internal memory buffer. Only relevant when *Update mode* is set to *Change of state on trigger*. The specified memory location is monitored by the gateway. Whenever the trigger byte is updated, the gateway will produce the transaction on the subnetwork. This way, the control system can instruct the gateway to produce a specific transaction on the subnetwork by updating the corresponding trigger byte. The trigger byte address must be unique to each transaction. It can not be shared by multiple transactions. Valid settings range from 0x200 to 0x3FF and 0x400 to 0xFFF

6.5.6 Consume Transactions (Generic Data Mode)

Fig. 40   **Parameters**

Parameter	Description
Offline options for subnetwork	The action to take for this transaction if the subnetwork goes offline. This affects the data that is sent to the higher level network. **Clear** = Data is cleared (0) on the higher level network **Freeze** = Data is frozen on the higher level network
Offline timeout time (10 ms)	The maximum allowed time in steps of 10 ms between two messages before the subnetwork is considered to be offline. Zero (0) disables the timeout feature. The entered value is multiplied by 10. A value of 5 means 50 ms.
Trigger byte	Enables/disables the trigger byte function. This function can be used to notify the control system of updated data. If enabled, the trigger byte will be incremented each time a valid transaction is consumed by the gateway. The trigger byte will also be incremented if the offline option is set to **Clear** and the offline timeout limit is reached.
Trigger byte address	The location of the trigger byte in the internal memory buffer. The trigger byte address must be unique to each transaction. It can not be shared by multiple transactions. Valid settings range from 0x000 to 0x1FF and 0x400 to 0xFFF

6.5.7 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.
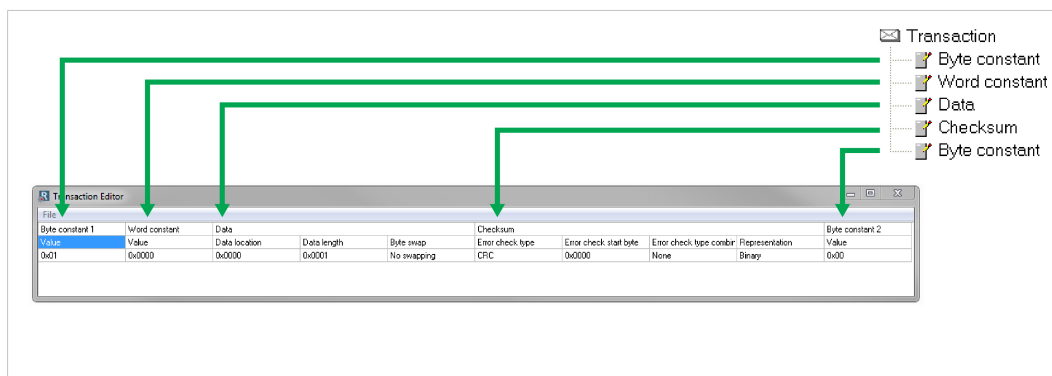

Fig. 41   **Transaction Editor**

To edit the value of a parameter, click on it and enter a new value, then select **Apply Changes** from the **File** menu in the Transaction Editor. To exit the editor without saving, select **Quit**.

When editing transactions that are based on predefined commands, certain parts of the transaction may not be editable.

6.6 Frame Objects (Master Mode & Generic Data Mode)

6.6.1 General

Each transaction consists of *Frame Objects* which make 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:

- Constant Objects

- Limit Objects

- Data Objects

- Variable Data Objects
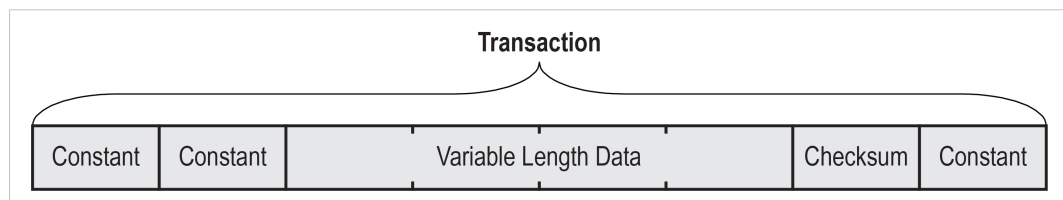
- Checksum Objects

Example:

Fig. 42 Frame object example

6.6.2 Adding and Editing Frame Objects

To add a frame object to a transaction, right-click on the transaction and select one of the entries in the context menu.
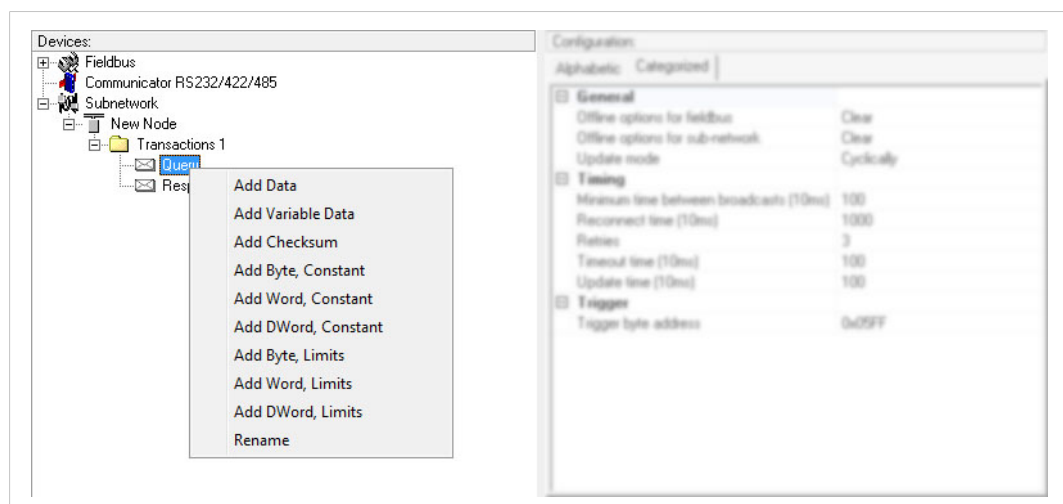

Fig. 43 Adding frame objects

The *Transaction Editor* can also be used to edit transactions and frame objects in a more visual manner. See .

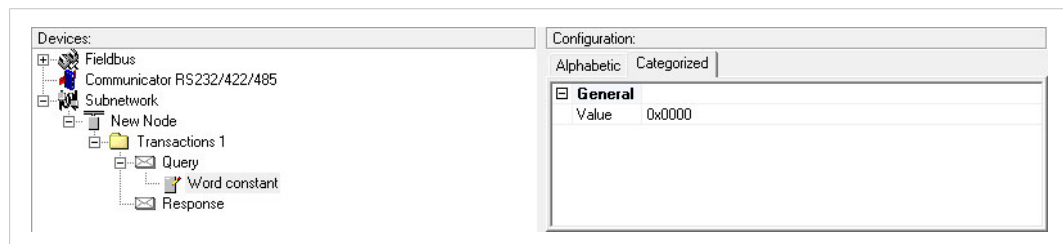### 6.6.3   Constant Objects (Byte, Word, Dword)

Fig. 44   Constant object

Constant Objects contain a fixed value of a **Byte** (8 bits), **Word** (16 bits) or **DWord** (32 bits).

- **Produce and Query Transactions**

    The gateway will send the value as-is without processing.

- **Consume and Response Transactions**

    The gateway will check if the received value matches the specified value. If not, the message will be discarded.

Constant Object Parameters

Parameter	Allowed values
Value	Byte: `0x00` to `0xFFh` Word: `0x0000` to `0xFFFFh` DWord: `0x00000000` to `0xFFFFFFFFh`

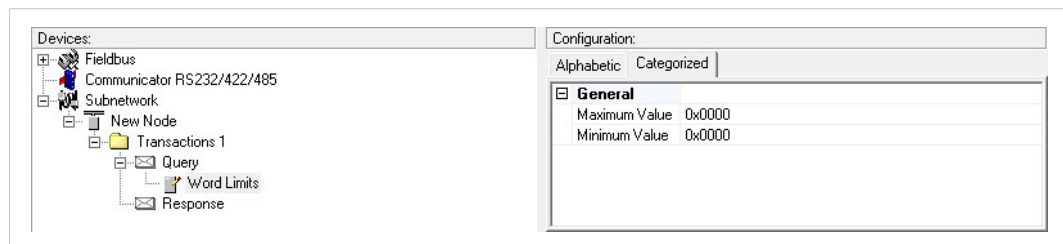### 6.6.4 Limit Objects (Byte, Word, Dword)

Fig. 45   Limit object

Limit Objects define a value range between a maximum and a minimum value. The two values can be given as a **Byte** (8 bits), **Word** (16 bits) or **DWord** (32 bits).

- **Produce and Query Transactions**

  Not used (the object will be ignored)

- **Consume and Response Transactions**

  The gateway will check if the received value fits inside the specified value interval. If not, the message will be discarded.

Limit Object Parameters

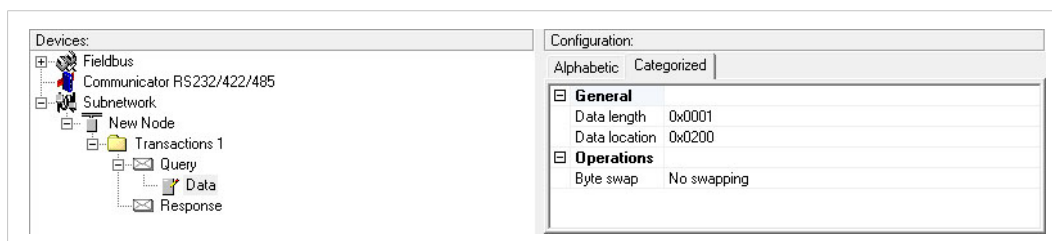| Parameter	Allowed values
Maximum Value	Byte: `0x00` to `0xFFh` Word: `0x0000` to `0xFFFFh` DWord: `0x00000000` to `0xFFFFFFFFh` **Note:** The value must be larger than the Minimum Value.
Minimum Value	Byte: `0x00` to `0xFEh` Word: `0x0000` to `0xFFFEh` DWord: `0x00000000` to `0xFFFFFFFEh` **Note:** The value must be less than the Maximum Value.

6.6.5   Data Object

Fig. 46   Data object

Data Objects are used to present raw data.

- **Produce and Query Transactions**

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

- **Consume and Response Transactions**

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

Data Object Parameters	
Parameter	**Description**
Byte Swap	**No swapping** No swapping is performed on the data **Swap 2 bytes** A, B, C, D becomes B, A, D, C **Swap 4 bytes** A, B, C, D becomes D, C, B, A
Data Length	The length of the data block in bytes. In Response/Consume transactions, incoming messages where the data size differs from this value will be discarded. The maximum data length allowed for one frame is 300 bytes.
Data Location	The location of the data block in the internal memory buffer.

6.6.6   Variable Data Object
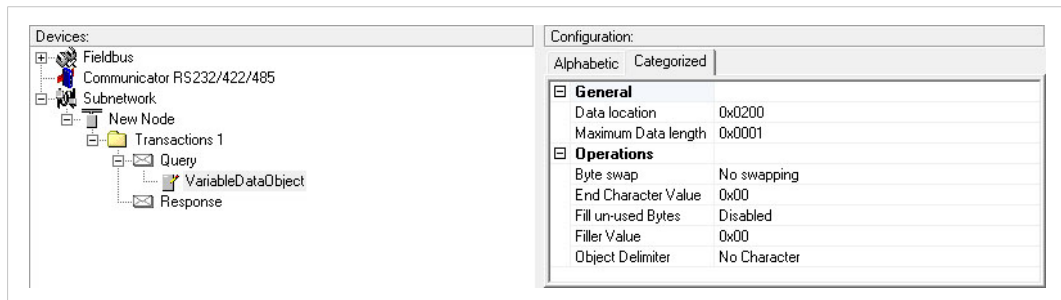

Fig. 47   Variable data object

> **ⓘ**   *Each transaction can only contain one variable data object.*

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:
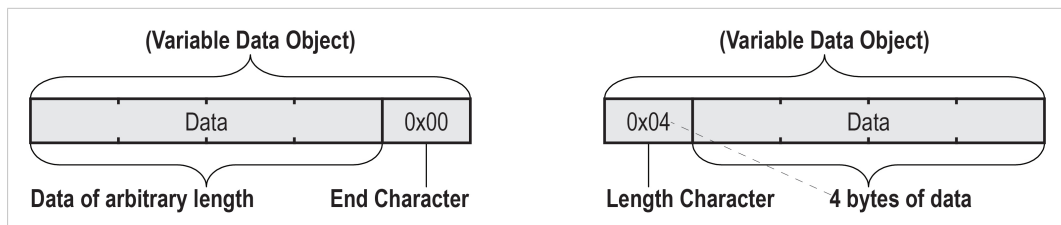

Fig. 48   Example of variable data

•   **Produce and Query Transactions**

    The specified data block will be forwarded from the higher level network to the subnetwork. 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 subnetwork or discarded.

•   **Consume and Response Transactions**

    The specified data block is forwarded from the subnetwork 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 subnetwork or discarded.

Variable Data Object Parameters

| Parameter	Description
Byte Swap	**No swapping** No swapping will be performed on the data **Swap 2 bytes** A, B, C, D becomes B, A, D, C **Swap 4 bytes** A, B, C, D becomes D, C, B, A
Fill Unused Bytes	When enabled, unused bytes in Consume/Response transactions will be filled with the value specified in *Filler Value*. This parameter is ignored in Produce/Query transactions.
Filler Value	Filler byte value.
Data Location	The offset in the internal memory buffer that the data shall be read from or written to
Object Delimiter (Produce/Query)	**Length Character** Length character visible in internal memory buffer but *not* sent to the subnetwork **Length Character Visible** Length character visible in internal memory buffer *and* sent to the subnetwork **End Character** End character visible in internal memory buffer but *not* sent to the subnetwork **End Character Visible** End character visible in the internal memory buffer *and* sent to the subnetwork **No Character** No End or Length character generated in the internal memory buffer
Object Delimiter (Consume/Response)	**Length Character** Length character visible in internal memory buffer but *not* received from the subnetwork **Length Character Visible** Length character visible in internal memory buffer *and* received from the subnetwork **End Character** End character visible in internal memory buffer but *not* received from the subnetwork **End Character Visible** End character visible in the internal memory buffer *and* received from the subnetwork **No Character** No End or Length characters included in the received string or generated in the internal memory buffer
End Character Value	End character value
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.

6.6.7 Checksum Object
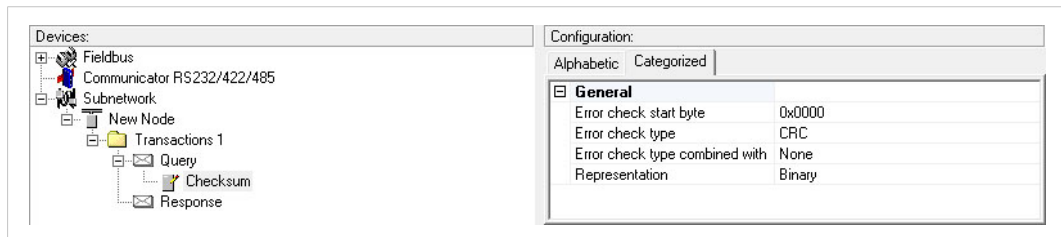
Fig. 49   Checksum object

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.

Checksum Object Parameters

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

6.7   Commands (Master Mode & Generic Data Mode)

6.7.1   General

Commands are 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 subnetwork.

Adding a command to a node results in a transaction or transactions being added according to 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. These parameters cannot be edited directly and will be greyed out in the parameter section of Anybus Configuration Manager.

In Master Mode, commands for the most common Modbus RTU functions have been predefined. Additional commands can easily be added using the Command Editor. In Generic Data Mode there are no predefined commands.

6.7.2   Adding and Managing Commands

To add a command to a node, right-click on the node and select **Add Command**.
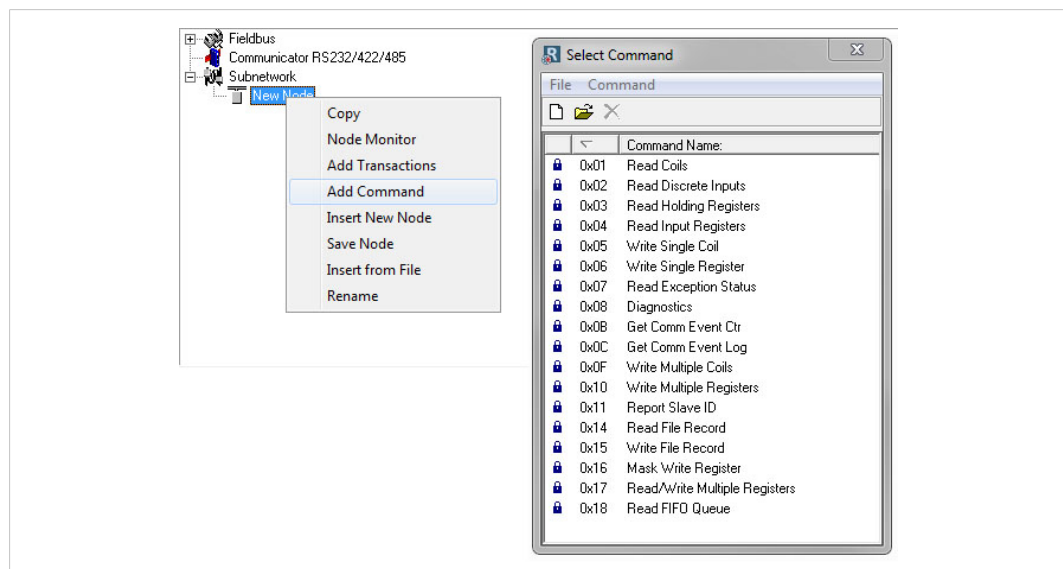

Fig. 50   Adding commands

You can use the **File** and **Command** menus in this window to add, edit and delete custom commands. The predefined commands in Master Mode cannot be edited or deleted.

As with other transactions, the frame objects of each 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.

6.7.3 Command Editor

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.
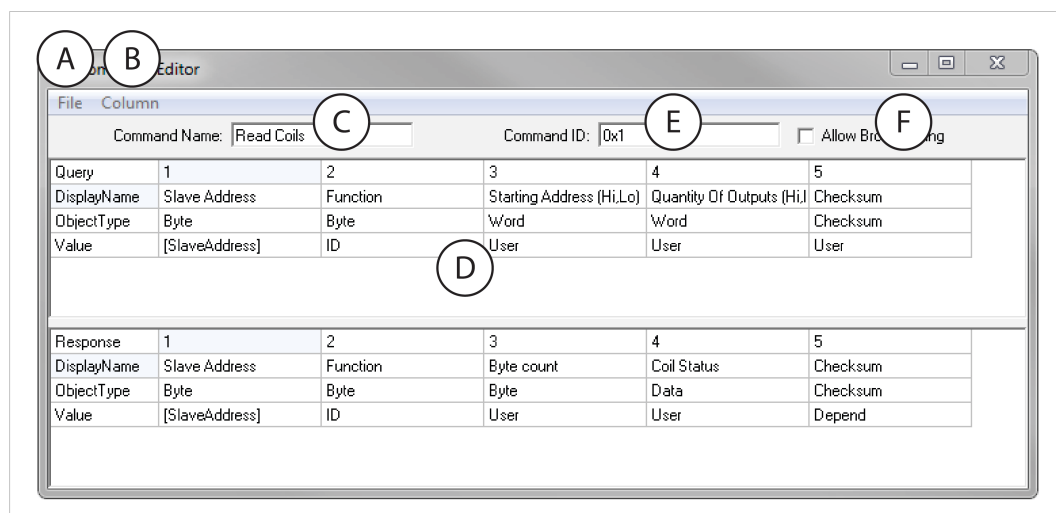

Fig. 51   Command Editor

A. **File menu**

Contains commands for applying changes or exiting the editor.

B. **Column menu**

Contains commands for adding and editing columns.

Append Column	Add a new column to the command
Insert Column	Insert a column at the selected position
Delete Column	Delete the column at the selected position

C. **Command Transactions**

The transactions associated with the command. Depending on the protocol mode this can be a query-response pair or a single transaction.

D. **Command ID**

A unique identifier for the command.

E. **Additional settings**

Allow Broadcasting (Master Mode)	Specifies if the command can be broadcasted.
Produce/Consume (Generic Data Mode)	Selects if the command is producing or consuming data.

Editing a Command

The transaction section in the Command Editor represents the transactions associated with the command. Each column represents a frame object within the transaction, and contains the following parameters:

- **Query/Response/Produce/Consume**

  Indicates the direction of the transaction.

- **DisplayName**

  Each column can be named so that the parts of the command are more easily identifiable in the Transaction Editor or in the parameter section of the main window.

- **ObjectType**

  The type of frame object to be used for the column.

- **Value**

  Specifies where the frame object shall retrieve its value.

Value	Description
User	Settings associated with the object can be edited by the user.
ID	Value will be retrieved from the *Command ID* setting in the Command Editor.
[SlaveAddress]	Value will be retrieved from the *SlaveAddress* parameter. See *Nodes, p. 42*.
Depend	Only relevant for Response transactions in Master Mode. The value will be retrieved from the corresponding part of the Query transaction.

Example: Modbus RTU Command in Master Mode

A Modbus RTU transaction always contains the following parts:

- Slave Address (1 byte)

- Function Code (1 byte)

- Data

- Checksum (CRC-16)

The command consists of a Query and a Response. The required Modbus RTU specific frame objects are automatically added, with a data object inserted between the function code and the checksum. These objects cannot be moved or deleted, but additional objects can be added between the function code and the checksum object.

Enter a descriptive name for the command in the *Command Name* field and a suitable function code in the *Command ID* field. If the command is allowed to be broadcasted, check the *Allow Broadcasting* checkbox.
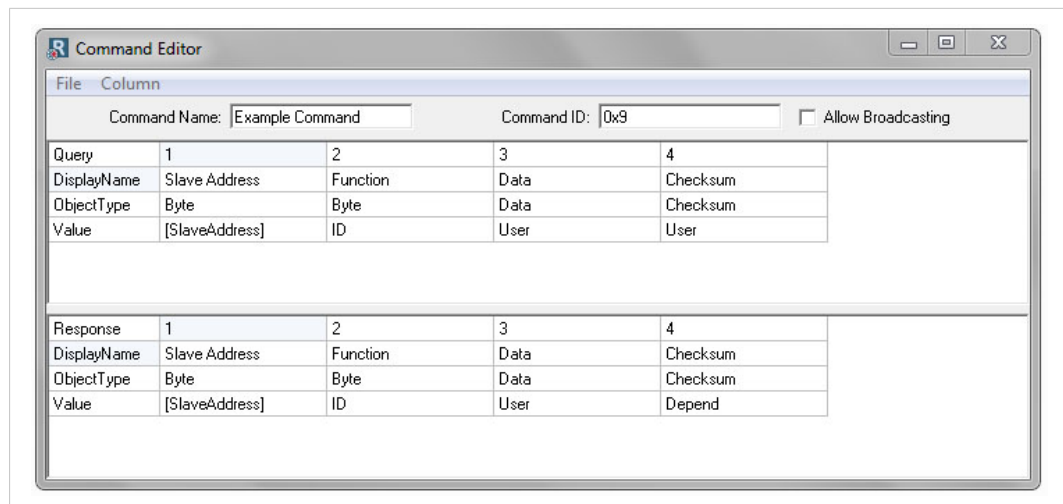
Fig. 52   Example - Modbus RTU Command

Query	1	2	3	4
DisplayName	Slave Address	Function	Data	Checksum
ObjectType	Byte	Byte	Data	Checksum
Value	[SlaveAddress]	ID	User	User
	Linked to the SlaveAddress parameter.	Retrieved from the CommandID field in the Command Editor window.	Size and location of this object is set by the user.	Checksum type and parameters can be selected by the user. Default = CRC-16 (Modbus RTU standard).

Response	1	2	3	4
DisplayName	Slave Address	Function	Data	Checksum
ObjectType	Byte	Byte	Data	Checksum
Value	[SlaveAddress]	ID	User	User
	Linked to the SlaveAddress parameter.	Retrieved from the CommandID field in the Command Editor window.	Size and location of this object is set by the user.	Retrieved from the corresponding object in the Query transaction.

6.8   Services (DF1 Master Mode)

6.8.1   General

Services are commands that can be stored and reused. The user configures each slave with services that can be issued from the master. When the Anybus Communicator is going to execute a service, it automatically chooses the appropriate DF1 command(s) used to perform the service on the selected DF1 node type.

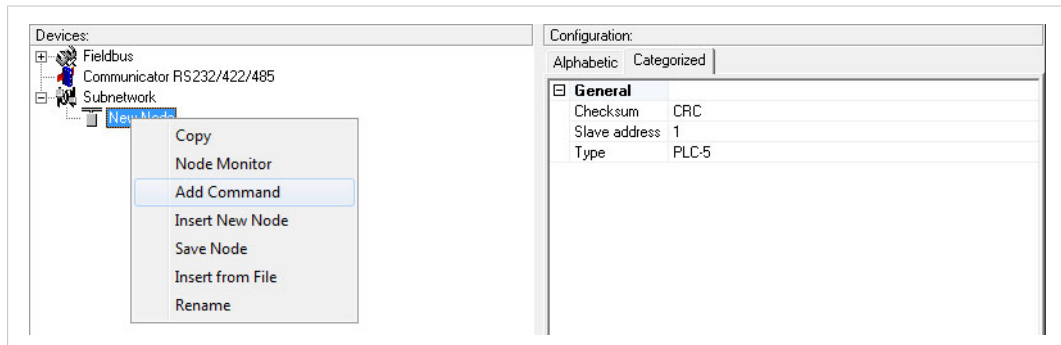A maximum of 50 services are allowed.

Predefined Services

Fig. 53   Adding a service

Right-click on the node and choose **Add Command**. Four different services will be available:

•   Integrity Check

•   Read Diagnostics

•   Read Data

•   Write Data

The parameters of the predefined services can be configured to suit the application.
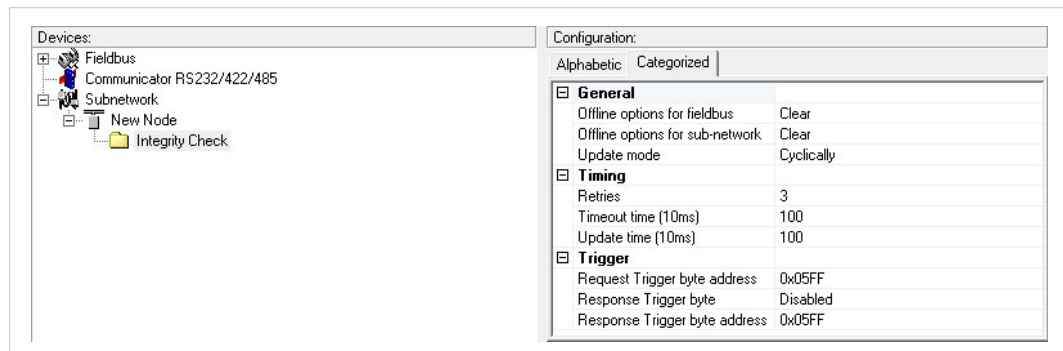
Common Parameters

Fig. 54   **Common 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 subnetwork.	Clear Freeze Noscanning
Offline options for subnetwork	The action to take for this service if the subnetwork goes offline. This option affects the data that is reported to the fieldbus master.	Clear Freeze
Update mode	The update mode for this service	Cyclically On data change Single shot Change of state on trigger

Timing

Parameter	Description	Default
Retries	The number of times to resend this service before the node is disconnected	3
Timeout time (10 ms)	The time to wait before resending this service (in steps of 10 ms)	100 (= 1000 ms)
Update time (10 ms)	The minimum time between two services of this kind (in steps of 10 ms)	100 (= 1000 ms)

Trigger

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

6.8.2 Integrity Check

This service checks that a node is up and running correctly. A telegram is sent to the node, and the node mirrors and returns the telegram.

This service contains no configuration apart from the common parameters.
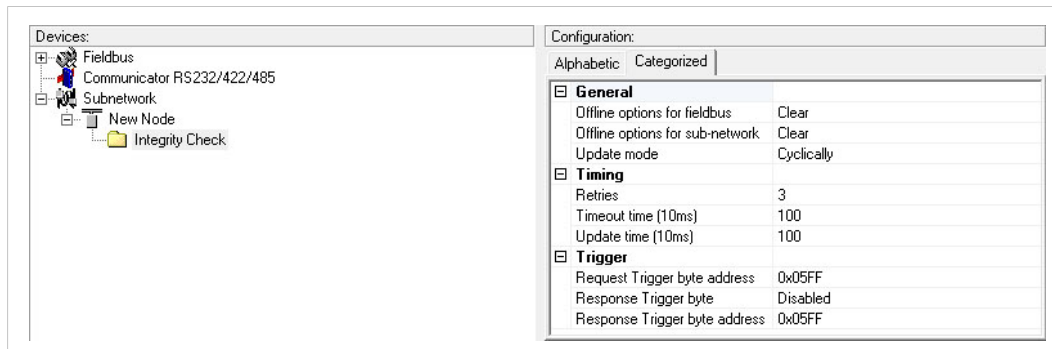

Fig. 55   Integrity Check service

6.8.3 Read Diagnostics

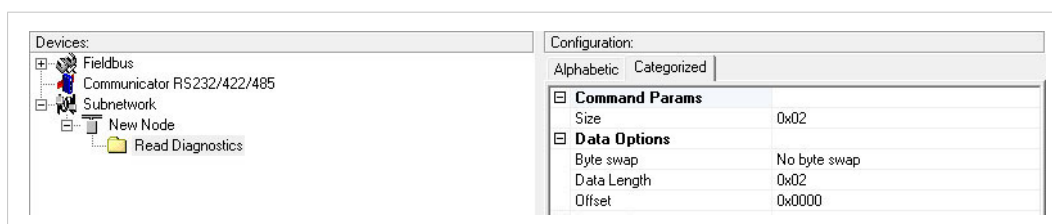This service reads diagnostic information from the gateway.

Fig. 56   Read Diagnostics service

Command Parameters		
Parameter	**Description**	**Valid settings**
Size	The number of bytes that can be 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.	PLC-5: 1–26 SLC-500: 1–28 MicroLogix: 1–26

Data Options		
Parameter	**Description**	**Valid settings**
Byte swap	Determines if the data shall be swapped	No byte swap Swap words Swap double words
Data length	The number of bytes, read from the DF1 network, to write to the area determined in *Offset*.	Less or equal to *Size*
Offset	The offset in the internal memory buffer that the data shall be read from.	-

6.8.4 Read Data

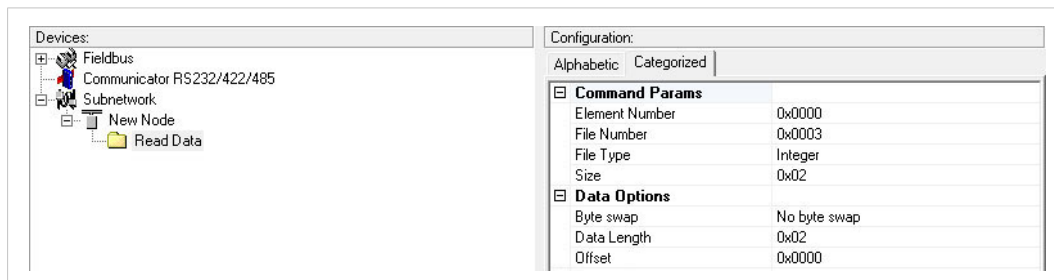This service is used to read data from the nodes in the subnetwork.

Fig. 57   Read Data service

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 SLC-500: 0–255 MicroLogix: 0–255
File Number	The file number of the data file to be accessed.	PLC-5: 3, 7, 8, 10–999 SLC-500: 3, 7, 8, 10–255 MicroLogix: 3, 7, 8, 10–255
File Type	The file type of the data to be accessed.	Integer Bit Float
Size	The number of bytes to read from the slave. One bit/integer element is 2 bytes and one float element is 4 bytes. The parameter must have an even value as only whole elements can be read.	PLC-5: 2–240 SLC-500: 2–236 MicroLogix: 2–242

Data Options

Parameter	Description	Valid settings
Byte Swap	Determines if the data shall be swapped	No byte swap Swap words Swap double words
Data Length	The number of bytes, read from the DF1 network, to write to the area determined in *Offset*.	Less or equal to *Size*
Offset	The offset in the internal memory buffer that the data shall be read from. **Note:** If the Control and Status registers are enabled (default) the first available data address will be 0x002 in the Input area, and 0x202 in the Output area.	-

6.8.5 Write Data

This service is used to write data to the nodes in the subnetwork. The parameters are the same as for the service *Read Data*. The only difference is that data is read from the internal memory buffer in the gateway and written to the subnetwork bus, instead of vice versa.

6.9   Subnetwork Monitor

6.9.1   General

The Subnetwork Monitor is intended to simplify configuration and troubleshooting of the subnetwork. Its main function is to display the data allocated for subnetwork communication and detect if any area has been allocated twice (address collision).

> **!** The Subnetwork Monitor increases the load on the gateway and may therefore reduce performance when used.
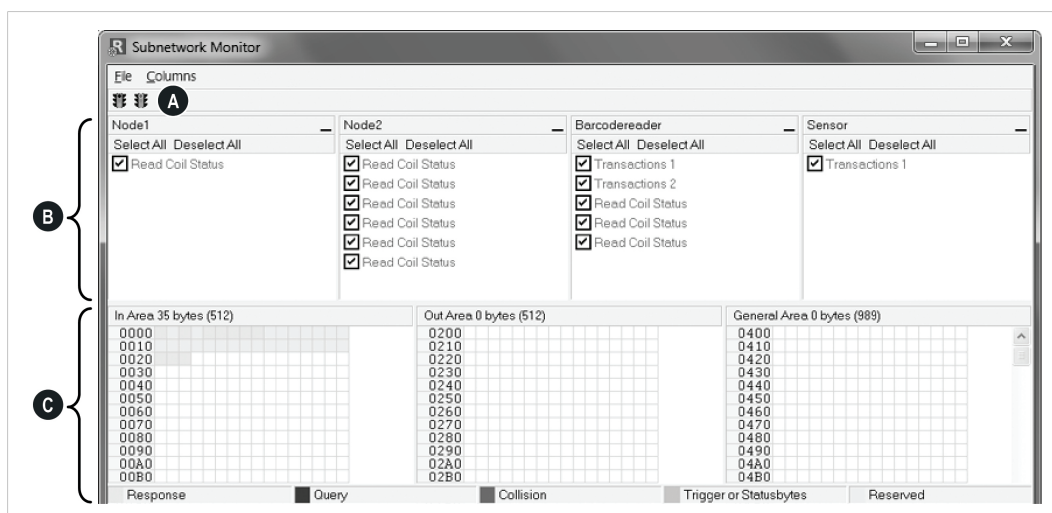
6.9.2   Operation

Fig. 58   Subnetwork Monitor

A: Menus and Toolbar Buttons

File	Start Network	Start subnetwork activity	🚦
	Stop Network	Stop all subnetwork activity	🚦
	Exit	Close the Subnetwork Monitor	
Columns	Free	Number of columns depends on window width	
	8 Multiple	Number of columns will be fixed to 8	

B: Nodes/Transactions Section

Lists all the configured nodes and their 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: Monitor 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

6.10 Node Monitor

6.10.1 General

The Node Monitor can provide valuable information when setting up the communication with the subnetwork, 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. The behavior of the Node Monitor differs slightly depending on the selected protocol mode.

> **!** The Node Monitor increases the load on the gateway and may therefore reduce performance when used.
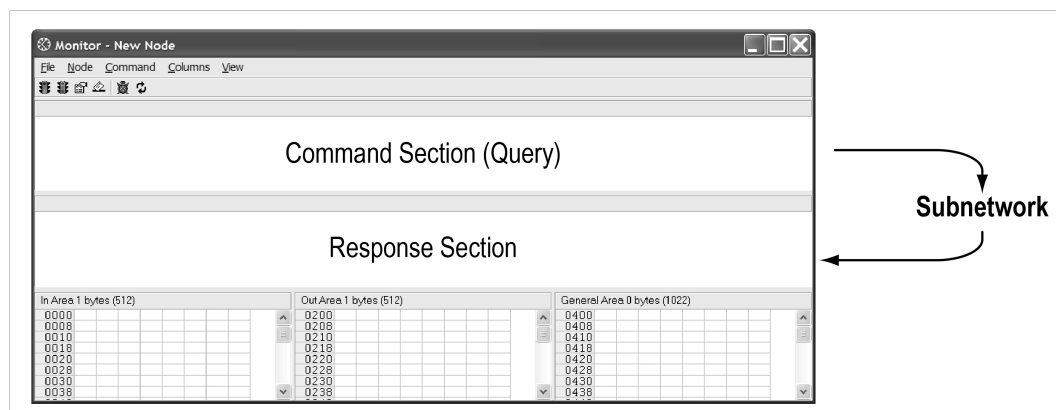
Master Mode and DF1 Master Mode

Fig. 59   Node Monitor – Master Mode

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

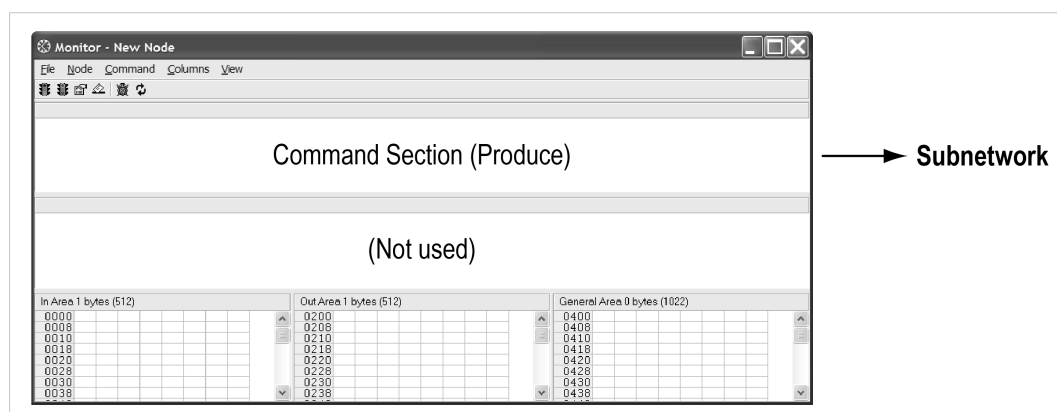**Generic Data Mode**

Fig. 60   Node Monitor – Generic Data Mode

The selected command (Produce Transaction) is sent to the subnetwork. It is not possible to monitor responses or other activity generated by other nodes.
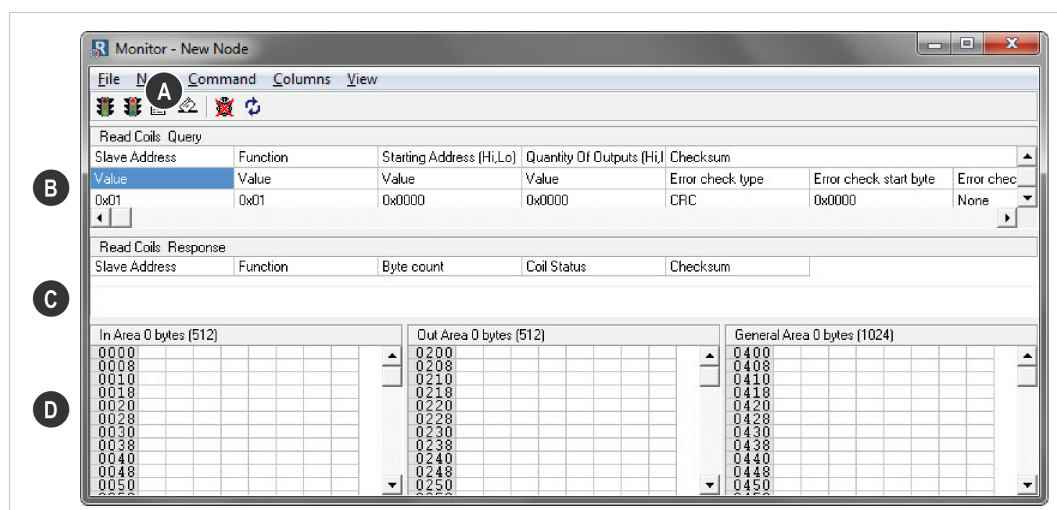
6.10.2   Operation

Fig. 61   Node Monitor

A: Menus and Toolbar Buttons

File	Exit	Closes the Node Monitor	
Node	Start Node	Enable all transactions associated with the node	![icon]
	Stop Node	Disable all transactions associated with the node	![icon]
Command	Select	Select a command to be sent to the subnetwork	![icon]
	Send	Send the selected command to the subnetwork	![icon]
Columns	Free	Number of columns depends on window width	
	8 Multiple	Number of columns will be fixed to 8	
View	Hex	Display the data in hexadecimal format	
	Decimal	Display the data in decimal format	
-	-	Stop/Resume refreshing	![icon]
	-	Refresh the displayed data	![icon]

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)

This section holds the response to the selected Command.

D: Monitor Section

Displays the data associated with the node. Areas in dark grey are reserved for the Status and Control Registers, and areas displayed in light grey represent data used by the node.

The data displayed in this section can be updated by clicking on the Refresh icon in the toolbar.

6.11   Data Logger

6.11.1   General

The subnetwork traffic can be logged into a buffer. This may provide valuable information when debugging the lowest levels of the subnetwork communication.

The logger is built into the gateway and is separate from Anybus Configuration Manager. This means that logging can be performed even if the gateway is disconnected from the computer.

The log buffer will hold 512 bytes of data in each direction by default. The size of the log buffer can be changed from the **Options** dialog. See also *Options Dialog, p. 38*.

> ⓘ   *The Data Logger is not available until a communication port has been selected.*

6.11.2   Operation

Select **Start logging** from the **Tools** menu, then choose the operation mode:

- **Log until full**

  Data will only be logged until the log buffer is full.

- **Log continuously**

  Data will be logged continuously until you select **Stop Logging**. The log buffer will contain the most recent data.

In both operation modes, selecting **Stop Logging** will stop the logging and open the log window.
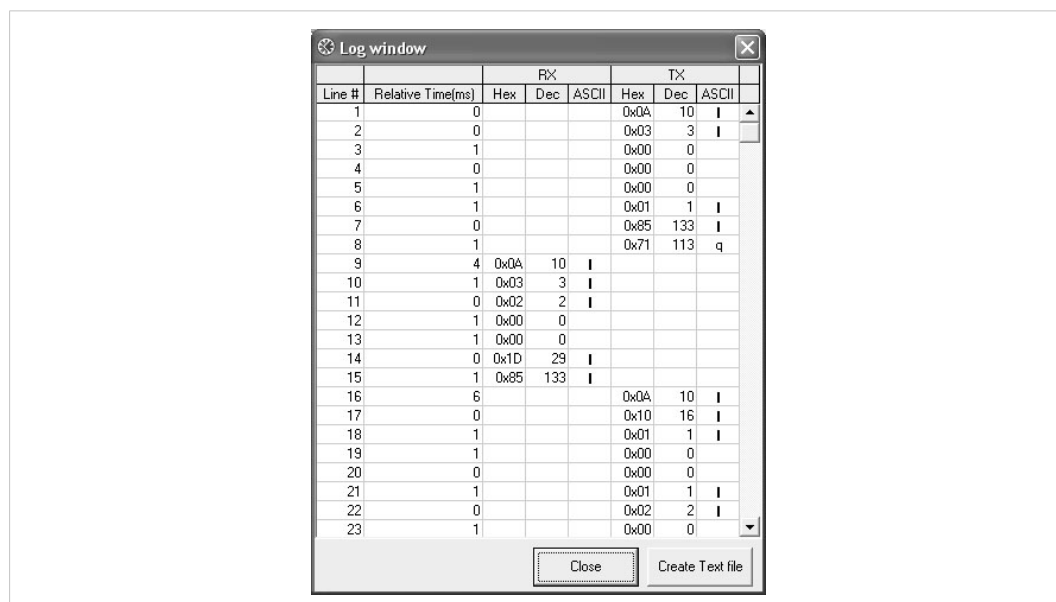
Log Window

Fig. 62   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.

Click **Create Text file** to save the data in ASCII text format.

Click **Close** to exit the log window.

6.12   Configuration Wizard

The configuration wizard will automatically create a subnetwork configuration from a predefined template based on input provided by the user.

The wizard will be displayed each time you create a new configuration unless it has been disabled on the **Options** page.
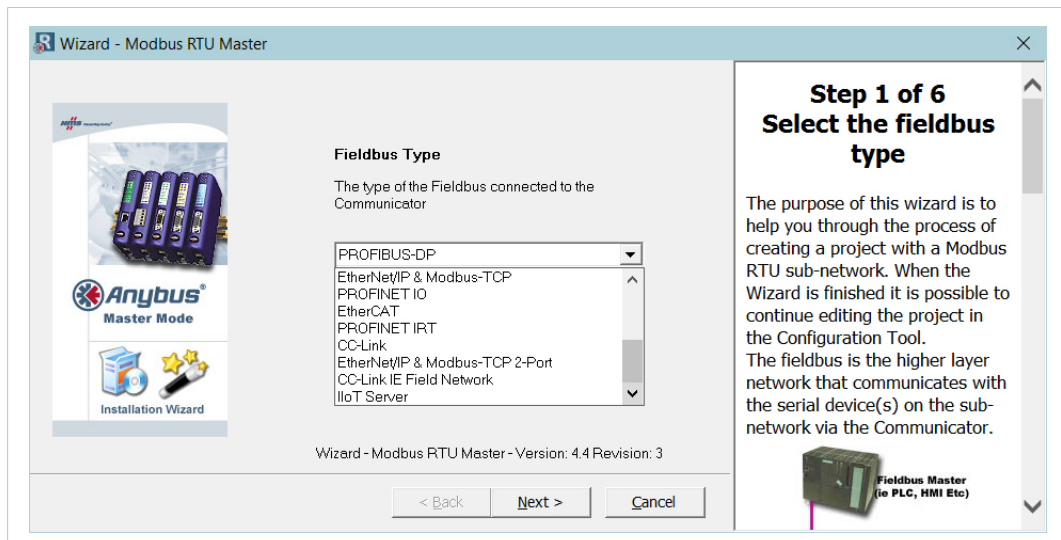

Fig. 63   Configuration wizard

The configuration wizard will help you through each step of creating and saving a basic configuration. You can then continue to edit the configuration in Anybus Configuration Manager before downloading it to the Anybus Communicator.

6.13   Control and Status Registers

6.13.1   General

The Control and Status registers are disabled by default but can be enabled in Anybus Configuration Manager. These registers form an interface for exchanging status information between the subnetwork and the fieldbus control system.

The main purpose of these registers is to:

• Report subnetwork-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 subnetwork

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

> ⓘ *The registers are stored in Motorola format (MSB first). If the higher level network uses a different byte order, the upper and lower bytes will appear swapped.*

6.13.2   Handshaking Procedure

The following handshaking procedure ensures that both parts receive proper information when accessing the Control and Status registers.
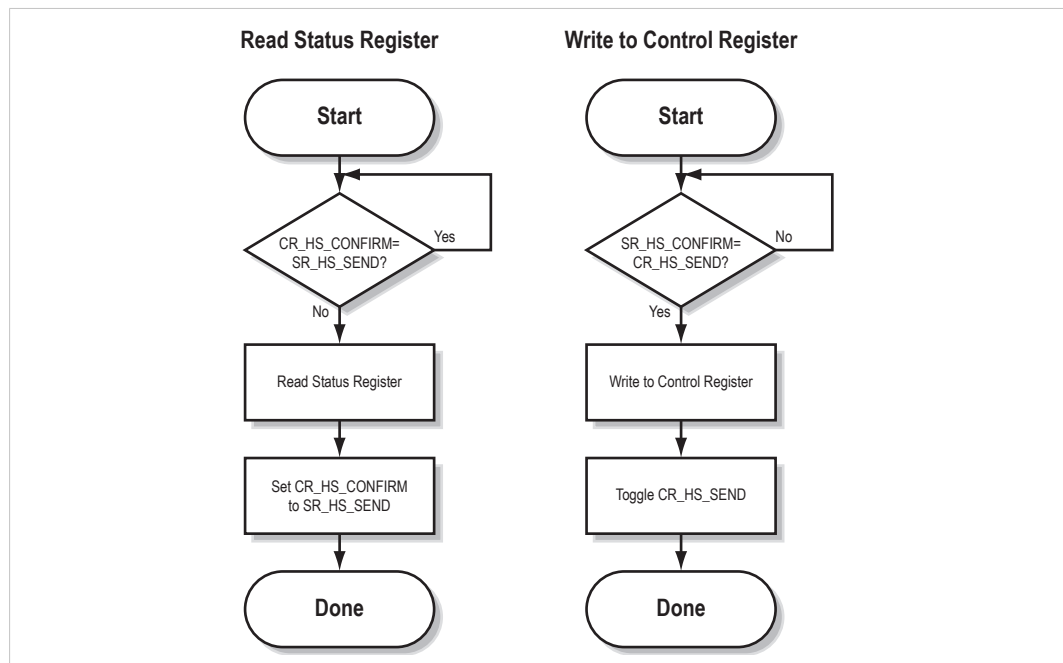

Fig. 64   Control/Status register handshaking procedure

6.13.3 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 Anybus Configuration Manager 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

Fieldbus network participation can be described using a state machine:

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 subnetwork configuration.

3. Wait until the fieldbus control system and the gateway are online on the fieldbus network, then 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 fieldbus goes offline, shift to state A.

C: Online (exchanging data)

• Exchanging valid data in both directions.

   If the fieldbus goes offline, shift to state A.
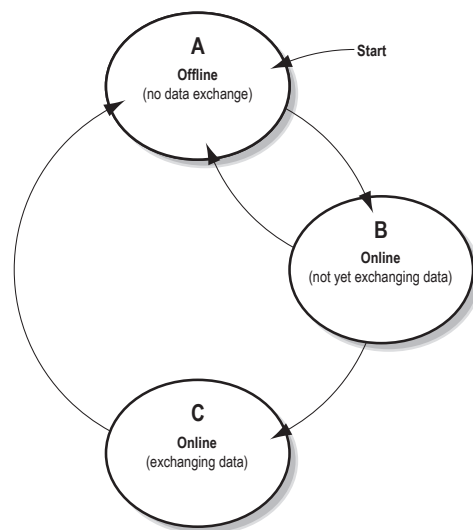
ⓘ *The gateway cannot spontaneously clear the Data Valid bit in the Status Register.*

Note on 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 by a bad connection to a node with a long timeout value assigned to it. The fieldbus control system should therefore never wait for this bit to be set before communicating with the subnetwork devices. The *Data Valid* bit should only be considered as an aid for the fieldbus control system to know when all data has been updated.

6.13.4 Status Register Contents (0x000 to 0x001)

Bit	Name	Description
15	Send (SR_HS_SEND)	Control the handshaking towards the fieldbus control system.
14	Confirm (SR_HS_CONFIRM)	

13	Data Valid	(Master Mode and DF1 Master Mode only) Set when all transactions have been executed successfully at least once. Once set, it will not change.
		  1: Data Valid
		  0: Data Not Valid

12... 8	Status Code	This field holds the last status report from the gateway.
7... 0	Data	

ℹ️ *The registers are stored in Motorola format (MSB first). If the higher level network uses a different byte order, the upper and lower bytes will appear swapped.*

Status Codes

Master Mode and DF1 Master Mode

Code	Condition	Type	Data	Description
0x00	Retransmission Counter Updated	Warning	Counter	The number of retransmissions on the subnetwork has increased. If this problem persists, it may trigger a Single Node MIssing or Multiple Nodes 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

Generic Data Mode

Code	Condition	Type	Data	Description
0x00	Invalid Transaction Counter Updated	Error	Counter	The number of invalid transactions (received transactions thar do not match any of the *Consume* transactions defined in the subnetwork 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.
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 be followed by a *No Error* condition when the cause has been resolved. Conditions of type *Warning* are considered informational and may not necessarily be followed by a *No Error* condition.

6.13.5   Control Register Contents (0x200 to 0x201)

Bit	Name	Description
15	Confirm (CR_HS_CONFIRM)	Control the handshaking towards the gateway.
14	Send (CR_HS_SEND)	
13	Data Valid	Controls data consistency. 1: Output Area valid – Exchange data on the subnetwork 0: Output Area not valid – Do not exchange data on the subnetwork **Note:** This bit is only relevant if the Control/Status Registers are set as *Enabled*
12	Execute Command	If set, the specified command will be executed by the gateway.
11... 8	Control Code	This field holds commands which can be executed by the gateway.
7... 0	Data	

> (i) *The registers are stored in Motorola format (MSB first). If the higher level network uses a different byte order, the upper and lower bytes will appear swapped.*

Control Codes

Master Mode and DF1 Master Mode			
Code	Condition	Data	Description
0x00	Disable Node	Node address	Disables the specified node.
0x01	Enable Node	Node address	Enables a previously disabled node.
0x02	Enable Nodes	Number of nodes	Enables the specified number of nodes, starting from the first node in the configuration. The remaining nodes will be disabled.

Control Codes are not supported in Generic Data Mode.

This page intentionally left blank

A Technical Data

A.1 General Specifications

Model name	Anybus Communicator PROFINET IRT (2.32)
Order code	AB7078
Dimensions (L x W x H)	120 x 75 x 27 mm
Weight	150 g
Operating temperature	0 to +55 °C (IEC 60068-2-1 and IEC 60068-2-2)
Storage temperature	-40 to +85 °C (IEC 60068-2-1 and IEC 60068-2-2)
Humidity range	5–95 % RH, non-condensing (IEC 60068-2-30)
Power supply	24 V ±10 % DC regulated power source
Current consumption	Typical: 100 mA @ 24 VDC Maximum: 200 mA @ 24 VDC
Galvanic isolation	Yes, on both network sides
Mechanical rating	IP20, NEMA rating 1
Mounting	DIN rail (EN 50022) Network shield conductance via DIN rail
Certifications	CE

A.2 Serial Interface

Serial application Interface	Selectable RS-232, RS-422, RS-485
Maximum number of stations	31 nodes via RS-422 or RS-485
Protocol: Modbus RTU	Modbus RTU Master - Query/Response
Protocol: ASCII/Vendor Specific	Request/Response or Produce/Consume
Protocol: Rockwell DF1	DF1 Master

A.3 PROFINET IRT Interface

PROFINET specification	2.32
PROFINET functionality	Isochronous Real-Time (IRT) communication Conformance supporting Class A, B and C Media Redundancy Protocol (MRP) support Discovery and Configuration Protocol (DCP) support Acyclic Data exchange (Record Data Requests) Asset Management
Isochronous cycle times	0.250 ms to 16 ms
Maximum I/O data	Up to 512 byte in each direction
Ethernet	100 Mbit/s, full duplex (fixed) Dual port cut-through switch, RJ45 connectors Ethernet Transport Provider support