



User Manual

Anybus[®] Communicator[™] for PROFINET

Rev. 2.03

HMS Industrial Networks AB


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


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



Table of Contents

Preface	About This Document
	How To Use This Document P-1
	Important User Information P-1
	Related Documents..... P-2
	Document History P-2
	Conventions & Terminology..... P-3
	Glossary P-3
	Support P-4
 Chapter 1	 About the Anybus Communicator for PROFINET
	External View 1-2
	Status LEDs 1-3
	Hardware Installation 1-4
	Software Installation 1-5
	<i>ABC Config Tool</i> 1-5
 Chapter 2	 Basic Operation
	General..... 2-1
	Data Exchange Model..... 2-2
	<i>Memory Map</i> 2-2
	<i>Data Exchange Example</i> 2-3
	Sub-Network Protocol 2-4
	<i>Protocol Modes</i> 2-4
	<i>Protocol Building Blocks</i> 2-4
	<i>Master Mode</i> 2-5
	<i>Generic Data Mode</i> 2-5
	PROFINET-IO..... 2-6
	<i>General</i> 2-6
	<i>I/O Configuration</i> 2-6
	<i>GSDML-File</i> 2-6
	<i>Data Representation (IO Data ↔ Record Data)</i> 2-7
	Modbus/TCP (Read-Only)..... 2-8
	<i>General</i> 2-8
	<i>Data Representation (Modbus/TCP Register Map)</i> 2-8
	<i>Supported Exception codes</i> 2-8
 Chapter 3	 File System
	General..... 3-1
	Overview 3-2
	System Files..... 3-2

Chapter 4	Basic Network Configuration	
	General Information	4-1
	Ethernet Configuration File ('ethcfg.cfg')	4-2
	<i>General</i>	4-2
	PROFINET Settings	4-3
	IP Access Control	4-3
	Anybus IPconfig (HICP)	4-4
Chapter 5	FTP Server	
	General	5-1
Chapter 6	Web Server	
	General	6-1
	Authorization	6-2
	Content Types	6-3
Chapter 7	Server Side Include (SSI)	
	Functions	7-2
	Changing SSI output	7-10
	<i>SSI Output String File</i>	7-10
	<i>Temporary SSI Output change</i>	7-11
Chapter 8	Email Client	
	General	8-1
	Email Definitions	8-2
Chapter 9	Navigating the ABC Config Tool	
	Main Window	9-1
	<i>Pull-down Menu</i>	9-2
	<i>Toolbar Icons</i>	9-5
Chapter 10	Basic Settings	
	Fieldbus Settings	10-1
	ABC Parameters	10-2
	Sub-Network Parameters	10-3
Chapter 11	Nodes	
	General	11-1
	Adding & Managing Nodes	11-1
	Node Parameters	11-1

Chapter 12	Transactions	
	General.....	12-1
	Adding & Managing Transactions.....	12-1
	Transaction Parameters (Master Mode).....	12-2
	<i>Parameters (Query & Broadcast)</i>	12-2
	<i>Parameters (Response)</i>	12-3
	Transaction Parameters (Generic Data Mode).....	12-3
	<i>Produce-Transactions</i>	12-3
	<i>Consume-Transactions</i>	12-4
	Transaction Editor	12-5
Chapter 13	Frame Objects	
	General.....	13-1
	Adding and Editing Frame Objects	13-1
	Constant Objects (Byte, Word, Dword).....	13-2
	Limit Objects (Byte, Word, Dword)	13-3
	Data Object.....	13-4
	Variable Data Object	13-4
	Checksum Object.....	13-6
Chapter 14	Commands	
	General.....	14-1
	Adding & Managing Commands	14-1
	<i>Pull-Down Menu</i>	14-2
	<i>Toolbar Icons</i>	14-2
	The Command Editor	14-3
	<i>General</i>	14-3
	<i>Basic Navigation</i>	14-3
	<i>Pull-down Menu</i>	14-4
	<i>Editing a Command</i>	14-5
	<i>Example: Specifying a Modbus-RTU Command in Master Mode</i>	14-6
Chapter 15	Sub Network Monitor	
Chapter 16	Node Monitor	
	General.....	16-1
	Navigating the Node Monitor.....	16-2
	<i>Pull-Down Menu</i>	16-3
		16-4
	<i>Toolbar Icons</i>	16-4
Chapter 17	Data Logger	
	General.....	17-1
	Operation.....	17-1
	Configuration	17-2

Chapter 18	Configuration Wizards	
	General.....	18-1
	Selecting a Wizard Profile	18-1
	Wizard - Modbus RTU Master	18-2
Chapter 19	Control and Status Registers	
	General.....	19-1
	<i>Handshaking Procedure</i>	19-1
	<i>Data Consistency</i>	19-2
	Status Register Contents (Gateway to Control System)	19-3
	<i>General Information</i>	19-3
	<i>Status Codes in Master Mode</i>	19-3
	<i>Status Code in Generic Data Mode</i>	19-3
	Control Register Contents (Control System to Gateway).....	19-5
	<i>General Information</i>	19-5
	<i>Control Codes in Master Mode</i>	19-5
	<i>Control Codes in Generic Data Mode</i>	19-5
Chapter 20	Advanced Fieldbus Configuration	
	General.....	20-1
	Mailbox Editor.....	20-1
Appendix A	Connector Pin Assignments	
	PROFINET Connector (Ethernet).....	A-1
	Power Connector	A-1
	PC Connector	A-2
	Sub-network Interface.....	A-3
	<i>General Information</i>	A-3
	<i>Bias Resistors (RS485 Only)</i>	A-3
	<i>Termination (RS485 & RS422 Only)</i>	A-3
	<i>Connector Pinout (DB9F)</i>	A-4
	<i>Typical Connection (RS485)</i>	A-5
	<i>Typical Connection (RS422 & 4-Wire RS485)</i>	A-5
	<i>Typical Connection (RS232)</i>	A-5
		A-5
Appendix B	Technical Specification	
	Mechanical Properties.....	B-1
	Electrical Characteristics	B-1
	Environmental Characteristics	B-1
	Regulatory Compliance	B-2
Appendix C	Troubleshooting	
Appendix D	ASCII Table	

About This Document

How To Use This Document

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

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

Important User Information

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

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

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

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

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

Related Documents

[illegible]

Document History

Summary of Recent Changes (2.02... 2.03)

[illegible]

Revision List

[illegible]

Conventions & Terminology

The following conventions are used throughout this document:

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

Glossary

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

Support

For technical support consult the online FAQ (www.anybus.com), or contact the nearest support centre:

HMS Sweden (Head Office)

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

HMS North America

E-mail: us-support@hms-networks.com
Phone: +1-312-829-0605
Toll Free: 888-8-Anybus
Fax: +1-312-738-5873
Online: www.anybus.com

HMS Germany

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

HMS Japan

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

HMS China

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

HMS Italy

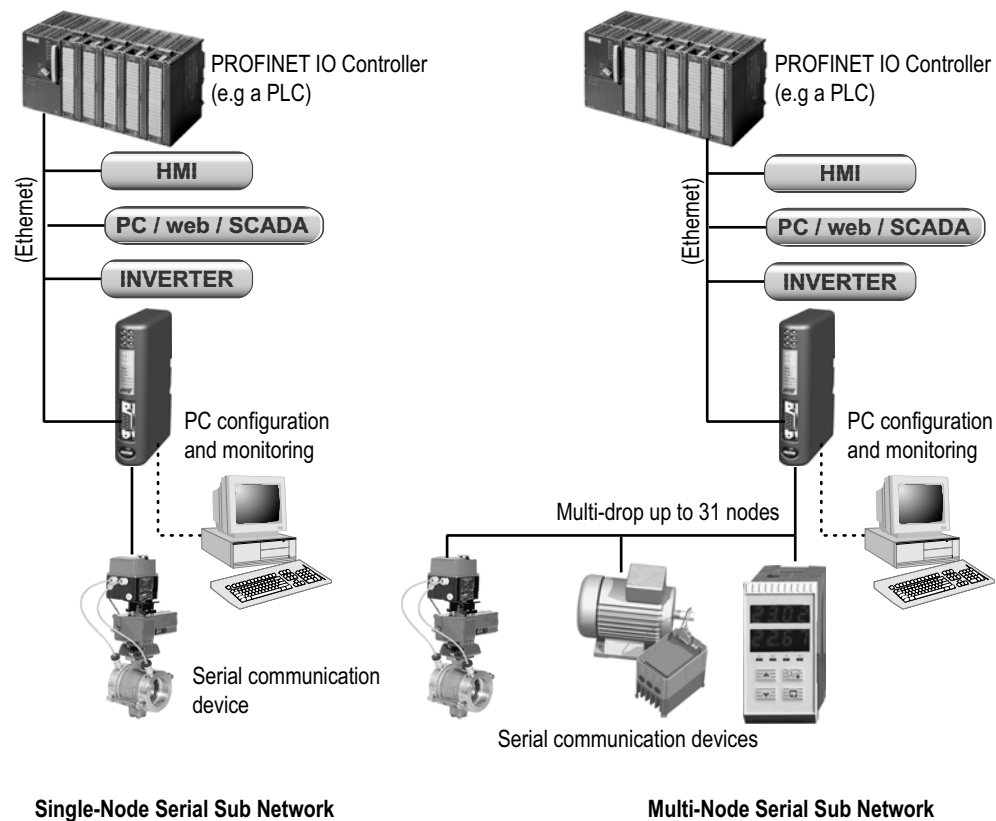
E-mail: it-support@hms-networks.com
Phone: +39 039 59662 27
Fax: +39 039 59662 31
Online: www.anybus.com

HMS France

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

About the Anybus Communicator for PROFINET

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



Sub Network

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

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

Ethernet Interface

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

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

External View

For wiring and pin assignments, see A-1 “Connector Pin Assignments”.

A: PROFINET Connector (Ethernet)

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

See also...

- A-1 “PROFINET Connector (Ethernet)”

B: Status LEDs

See also...

- 1-3 “Status LEDs”

C: PC-connector

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

See also...

- A-2 “PC Connector”

D: Sub-network Connector

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

See also...

- A-3 “Sub-network Interface”

E: Power Connector

This connector is used to apply power to the ABC.

See also...

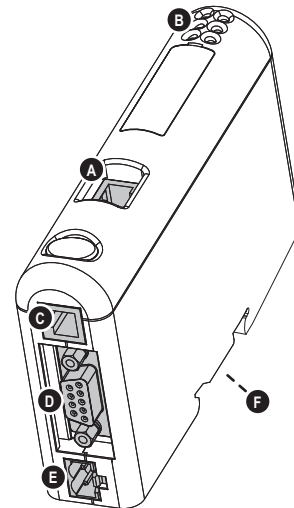
- A-1 “Power Connector”
- B-1 “Technical Specification”

F: DIN-rail Connector

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

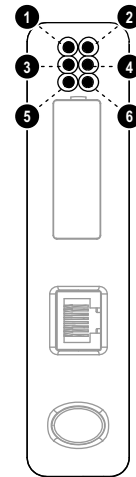
See also...

- 1-4 “Hardware Installation”



Status LEDs

#	State	Status
1 - Comm. Status	Off	Off line - No connection with IO Controller
	Green	On line, Run - Connection with IO Controller established - IO Controller is in RUN state
	Green, flashing	On line, STOP - Connection with IO Controller established - IO Controller in STOP state
2 - Module Status	Off	No power or not initialized
	Green	Initialized, no error
	Green, 1 flash	Diagnostic data available
	Green, 2 flashes	Blink. Used by engineering tools for identification.
	Red, 1 flash	Configuration Error - Too many modules/submodules - I/O size or Configuration mismatch
	Red, 3 flashes	No Station Name or no IP address assigned
	Red, 4 flashes	Internal error
3 - Link/Activity	Off	No link or power off
	Green	Link established
	Green, flashing	Receiving/transmitting data
4 - (not used)	-	-
5 - Subnet Status ^a	Off	Power off
	Green, flashing	Running correctly, but one or more transaction error(s) have occurred
	Green	Running
	Red	Transaction error/timeout or subnet stopped
6 - Device Status	Off	Power off
	Alternating Red/Green	Invalid or missing configuration
	Green	Initializing
	Green, flashing	Running
	Red, flashing	Contact the HMS support department



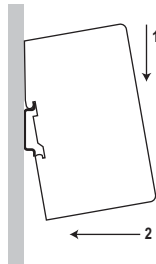
a. This led turns green when all transactions have been active at least once. This includes any transactions using "change of state" or "change of state on trigger". If a timeout occurs on a transaction, this led will turn red.

Hardware Installation

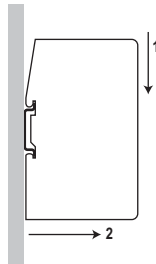
Perform the following steps when physically installing the ABC:

1. Snap the ABC on to the DIN-rail (See 1-2 “DIN-rail Connector”)

The DIN-rail mechanism works as follows:



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



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

2. Connect the ABC to the PROFINET (Ethernet) network
3. Connect the ABC to the serial sub-network
4. Connect the ABC to the PC via the Configuration Cable.
5. Connect the power cable and apply power
6. Start the ABC Config program on the PC
(The ABC Config software attempts to detect the serial port automatically. If not successful, select the correct port manually in the “Port”-menu).
7. Configure the ABC using the ABC Config Tool and download the configuration
8. Set up the PROFINET communication in accordance with the ABC configuration

Software Installation

ABC Config Tool

System requirements

- Pentium 133 MHz or higher
- 10 MB of free space on the hard drive
- 8 MB RAM
- Screen resolution of 800x600 (16 bit colour) or higher
- Microsoft Windows™ NT4 / 2000 / XP
- Internet Explorer 4.01 SP1 or newer

Installation

- **Anybus Communicator resource CD**

Insert the CD and follow the on-screen instructions. If the installation does not start automatically, right-click on the CD-drive icon and select Explore. Execute 'setup.exe' and follow the on-screen instructions.

- **From website**

Download and execute the self-extracting .exe-file from the HMS website (www.anybus.com).

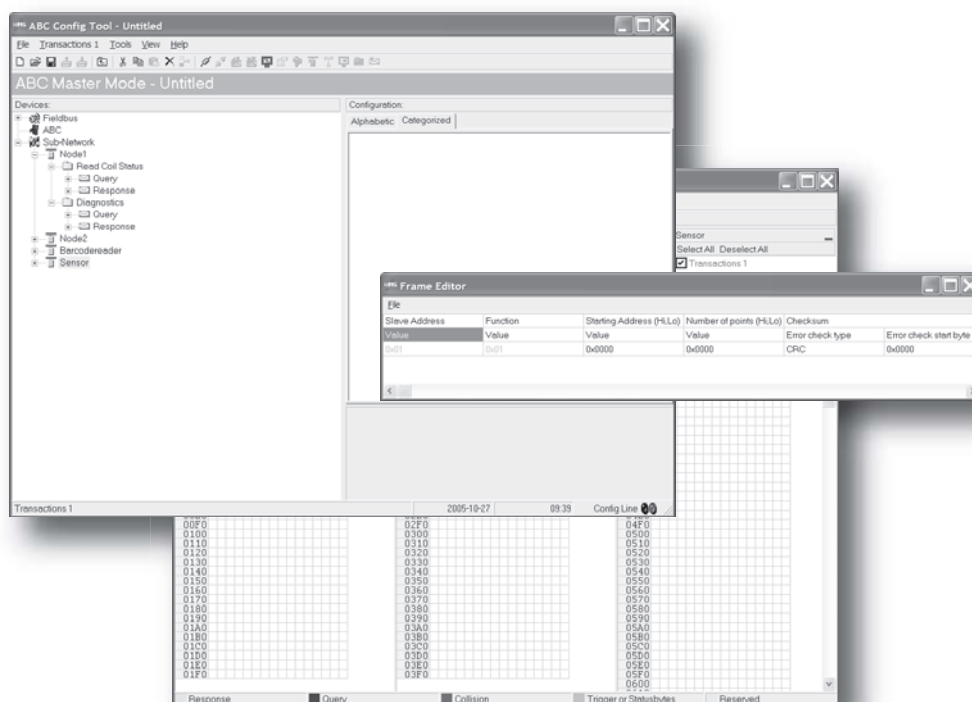
Basic Operation

General

The ABC is designed to exchange data between a serial sub-network and a higher level network. Unlike most other gateway devices of similar kind, it does not have a fixed protocol for the sub-network, and can be configured to handle almost any form of serial communication.

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

An essential part of the Anybus Communicator package is the ABC Config Tool, a Windows™ application which is used to supply the ABC with a description of the sub-network protocol. No programming skills are required; instead, a visual protocol description-system is used to specify the different parts of the serial communication.



Data Exchange Model

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

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

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

- **Input Data**

This area can be read by the higher level network, the Web server and the Email client.

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

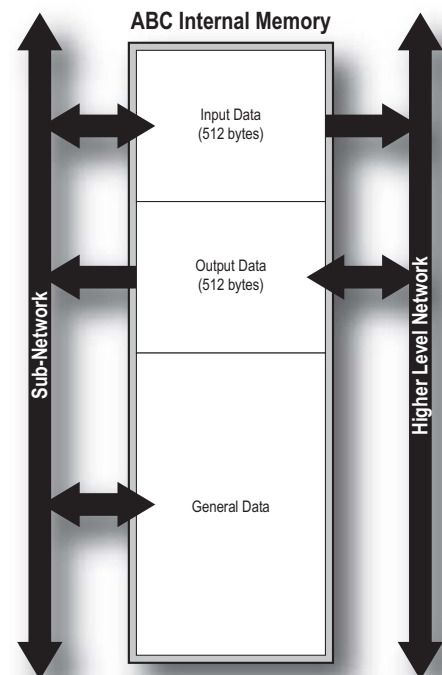
- **Output Data**

This area can be read from/written to by the higher level network, the Web server and the Email client.

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

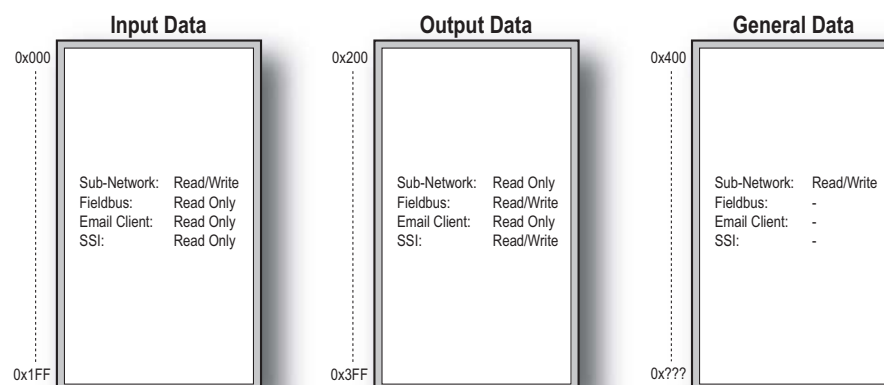
- **General Data**

This area is not exchanged on the higher level network, and can be used for transfers between individual nodes on the sub-network, or as a general “scratch pad” for data. The actual size of this area depends on the amount of data that is exchanged on the sub-network. The ABC can handle up to 1024 bytes of General Data.



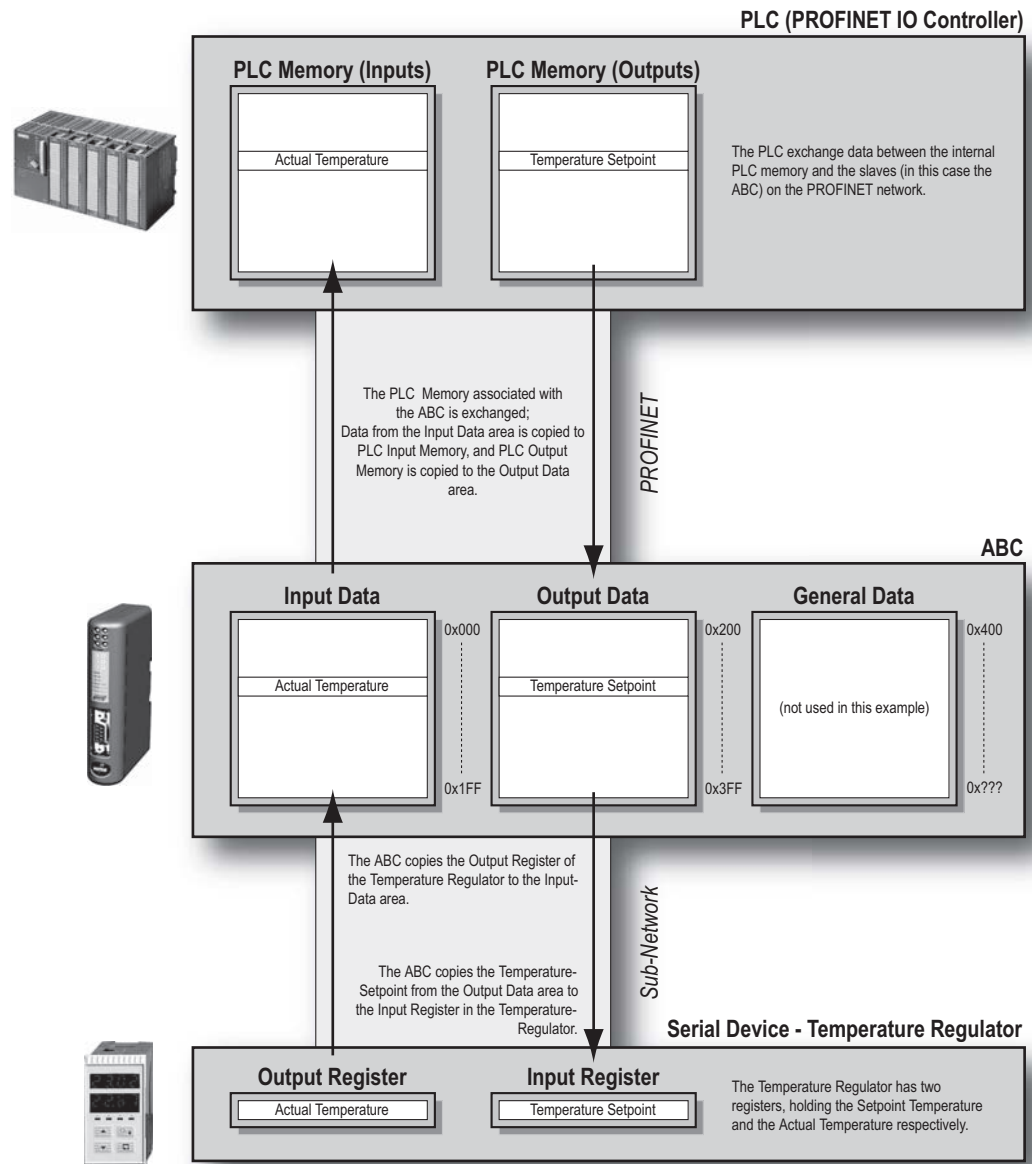
Memory Map

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



Data Exchange Example

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



Sub-Network Protocol

Protocol Modes

The ABC features two distinct modes of operation regarding the sub-network communication, called 'Master Mode' and 'Generic Data Mode'. Note that the protocol mode only specifies the basic communication model, not the actual sub-network protocol.

- **Master Mode**

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

For more information about this mode, see 2-5 "Master Mode".

- **Generic Data Mode**

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

For more information about this mode, see 2-5 "Generic Data Mode".

Protocol Building Blocks

The following building blocks are used in ABC Config Tool to describe the sub-network communication. How these blocks apply to the two protocol modes will be described later in this document.

- **Node**

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

- **Transaction**

A 'Transaction' represents a complete serial telegram, and consists of a number of Frame Objects (below). Each Transaction is associated with a set of parameters controlling how and when to use it on the sub-network.

- **Commands**

A 'Command' is simply a pre-defined Transaction stored in a list in the ABC Config Tool. This simplifies common operations by allowing Transactions to be stored and re-used.

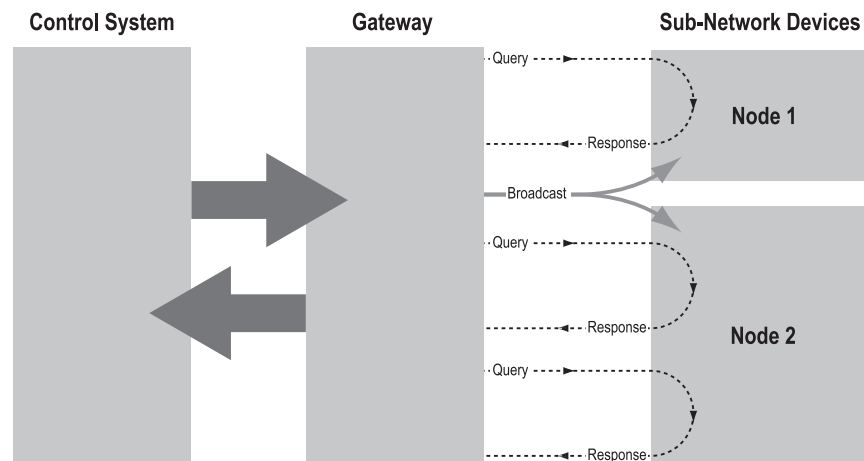
- **Frame Object**

'Frame Objects' are low level entities used to compose a Transaction (see above). A Frame Object can represent a fixed value (a constant), a range of values (limit objects), a block of data or a calculated checksum.

Master Mode

In this mode, the communication is based on a Query/Response scheme; when the ABC issues a Query on the sub-network, the addressed node is expected to issue a Response to that Query. Nodes are not permitted issue Responses spontaneously, i.e. without first receiving a Query.

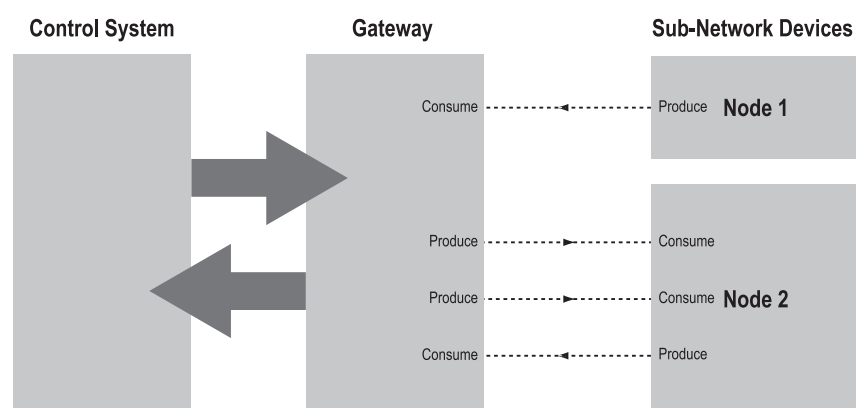
There is one exception to this rule; the Broadcaster. Most protocols offer some way of broadcasting messages to all nodes on the network, without expecting them to respond to the broadcasted message. This is also reflected in the ABC, which features a dedicated Broadcaster node.



In Master Mode, ABC Config Tool comes pre-loaded with most commonly used Modbus RTU commands, which can conveniently be reached by right-clicking on a node in the ABC Config Tool and selecting 'Insert New Command'. Note however that this does not in any way prevent other protocols based on the same Query-Response message-scheme to be implemented.

Generic Data Mode

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



In the figure above, the ABC 'Consumes' data that is 'Produced' by a node on the sub-network. This 'Consumed' data can then be accessed from the higher level network. This also works the other way around; the data received from the higher level network is used to 'Produce' a message on the sub-network to be 'Consumed' by a node.

PROFINET-IO

General

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

Supported Features

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

I/O Configuration

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

On PROFINET, the IO Data is built up by I/O modules. In the case of the ABC, the actual I/O module configuration is adopted from the I/O Controller/Supervisor, provided that the total I/O sizes specified by the IO Controller does not exceed the sizes specified in the ABC Config Tool.

For information about how the IO- and Record Data relates to the Input- and Output Data areas, see 2-7 "Data Representation (IO Data & Record Data)".

GSDML-File

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

This file holds information about the device (in this case the ABC), it's features, and possible I/O configurations. The latest version of the GSDML-file for the ABC can be downloaded from the HMS website, 'www.anybus.com'.

Data Representation (IO Data & Record Data)

As mentioned previously. The actual I/O configuration is determined by the IO Controller. The modules are mapped to the Input- and Output Data areas in the order of their slot number.

Example:

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

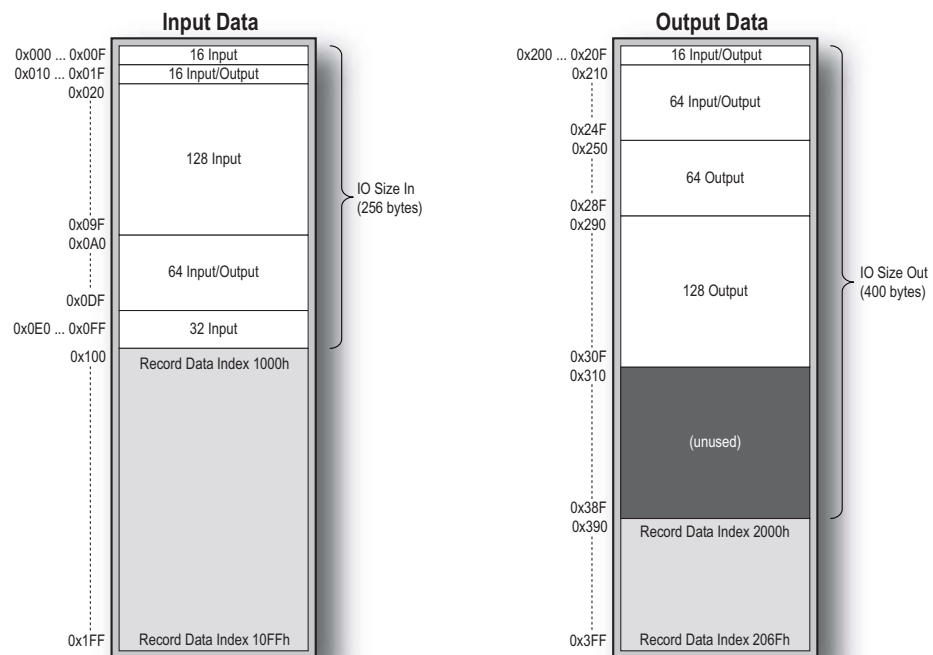
IO Size In = 256 bytes (0x0100)

IO Size Out = 400 bytes (0x0200)

The following modules are specified in the IO Controller:

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

Resulting memory layout:



Note the 'unused'-part of the Output Data area. The reason for this is that although IO Size Out is set to 400, only 272 bytes (128+64+64+16) are actually used in the I/O module configuration, hence it will not be exchanged on PROFINET.

Modbus/TCP (Read-Only)

General

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

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

Data Representation (Modbus/TCP Register Map)

The following function codes are implemented:

Modbus Function	Function Code	Associated with Area
Read Input Registers	4	Input Data area (0x000...0x1FF)
Read Multiple Registers	3	Output Data area (0x200...0x3FF)

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

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

Note: If enabled, the Control- and Status Registers occupies Input Register 0x0000 and Output Register 0x0000.

Supported Exception codes

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

File System

General

General

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

Conventions

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

Important Note:

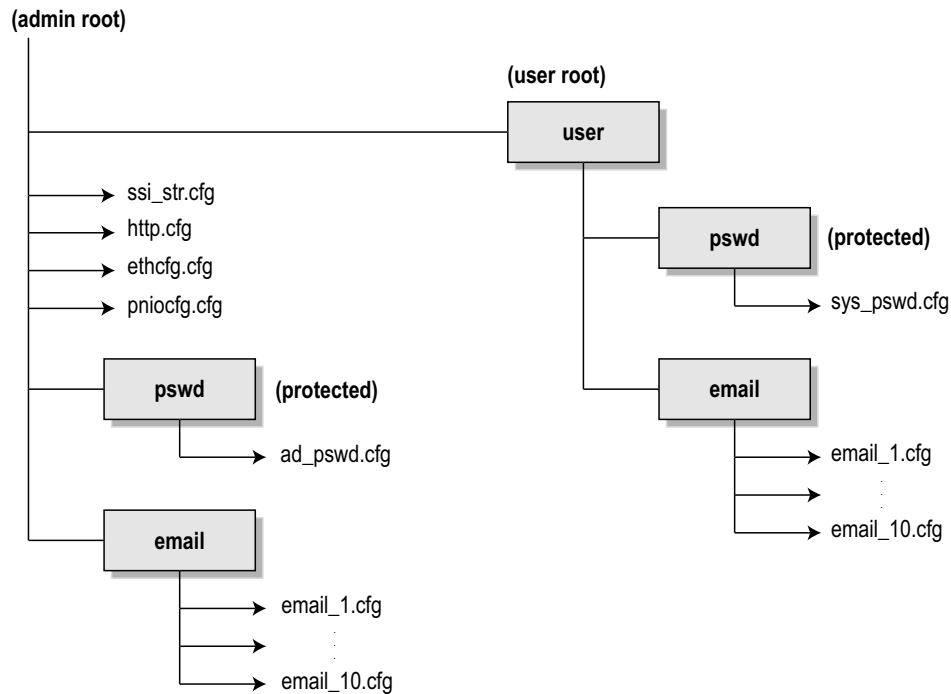
The non-volatile storage is located in FLASH memory. Each FLASH segment can only be erased approximately 1000000 times due to the nature of this type of memory.

The following operations will erase one or more FLASH segments:

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

Overview

The figure below illustrates the structure of the file system, where the system files are located, and which areas that can be accessed by Normal/Admin users.



System Files

The file system contains a set of files used for system configuration. These files, known as “system files”, are regular ASCII files which can be altered using a standard text editor (such as the Notepad in Microsoft Windows™). Note that some of these files may also be altered by the ABC itself, e.g. when using SSI (see 7-1 “Server Side Include (SSI)”).

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

Example:

```
[Key1]
value of key1
```

```
[Key2]
value of key2
```

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

Basic Network Configuration

General Information

The ABC offers two modes of operation regarding the network settings (see below). Which mode to use is determined by the ‘TCP/IP Settings’ parameter in ABC Config Tool, see 10-1 “Fieldbus Settings”.

- **TCP/IP Settings: Enabled**

When operating in this mode, the contents of the system file ‘ethcfg.cfg’ will be ignored completely, causing the following behaviour:

- DNS services will not be available
- Domain and Host name cannot be set
- Email services will not be available
- Settings received from the network (i.e. via HICP or DCP) will be lost in the event of a power loss or reset.

- **TCP/IP Settings: Disabled**

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

DCP (Discovery and Basic Configuration)

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

DHCP/BootP

The ABC can retrieve the TCP/IP settings from a DHCP or BootP server. If no DHCP server is found, the ABC will fall back on it’s current settings (i.e. the settings currently stored in ‘\ethcfg.cfg’).

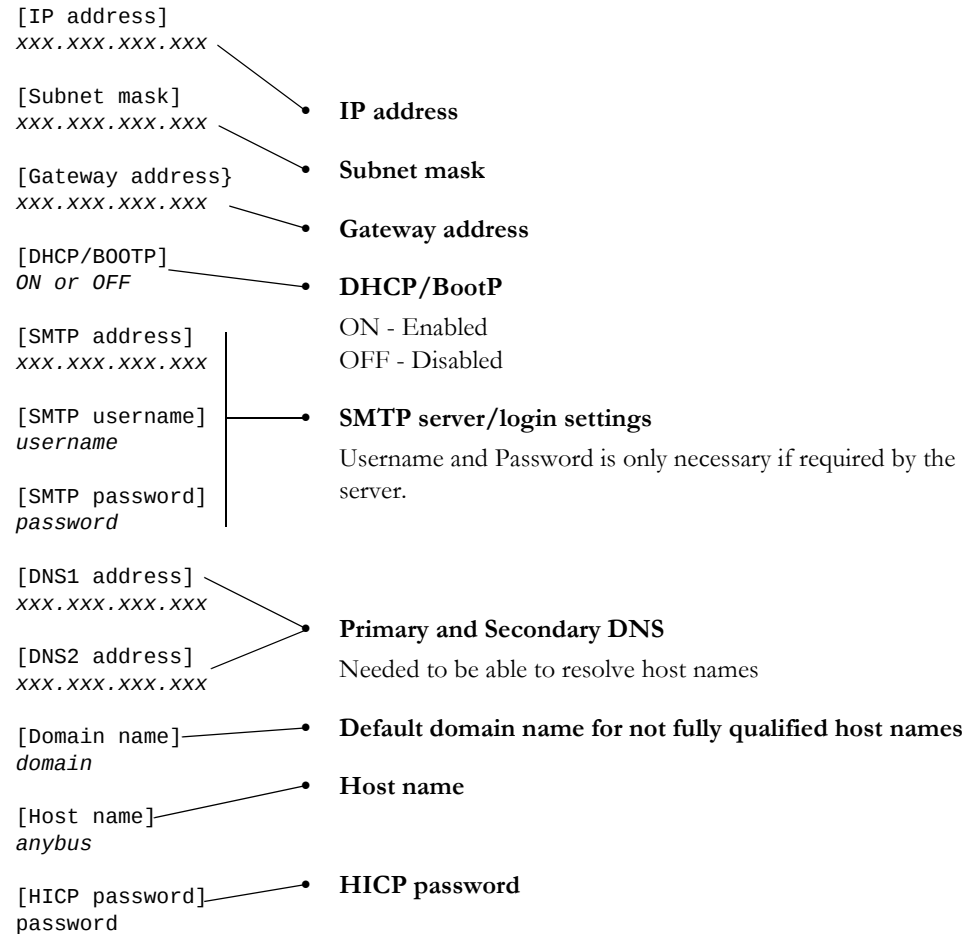
If no current settings are available (i.e. ‘ethcfg.cfg’ is missing, or contains invalid settings), the ABC will halt and indicate an error on the on-board status LEDs (the network configuration may however still be accessed via HICP, see 4-4 “Anybus IPconfig (HICP)”.

Ethernet Configuration File ('ethcfg.cfg')

General

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

File Format:



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

- DCP (See 4-1 "DCP (Discovery and Basic Configuration)").
- HICP (See 4-4 "Anybus IPconfig (HICP)").
- SSI (See 7-1 "Server Side Include (SSI)").

See also...

- 5-1 "FTP Server"
- 10-1 "Fieldbus Settings"

PROFINET Settings

The file ‘\pnio.cfg’ holds various PROFINET-related settings. The file is read once during startup, i.e. the ABC must be restarted on order for any changes to have effect (Unless it’s contents has been changed by an IO Controller/Supervisor via the DCP protocol. In such case, the settings will have effect immediately).

Example:

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

IP Access Control

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

File Format:

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

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

Anybus IPconfig (HICP)

The ABC supports the HICP protocol used by the Anybus IPconfig utility from HMS, which can be downloaded free of charge from the HMS website. This utility may be used to configure the network settings of any Anybus product connected to the network. Note that if successful, this will replace the settings currently stored in the configuration file ('ethcfg.cfg').

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

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



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

When done, click 'Set'. The new IP configuration will now be stored in the configuration file ('ethcfg.cfg').

Note that if 'TCP/IP Settings' has been enabled in the ABC Config Tool, any settings received via HICP will be lost in the event of a power loss or reset.

FTP Server

General

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

The following port numbers are used for FTP communication:

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

Security Levels

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

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

User Accounts

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

- ‘\user\pswd\sys_pswd.cfg’
This file holds the user accounts for normal-level users.
- ‘\pswd\ad_pswd.cfg’
This file holds the user accounts for admin-level users.

File Format:

The format of these files are as follows:

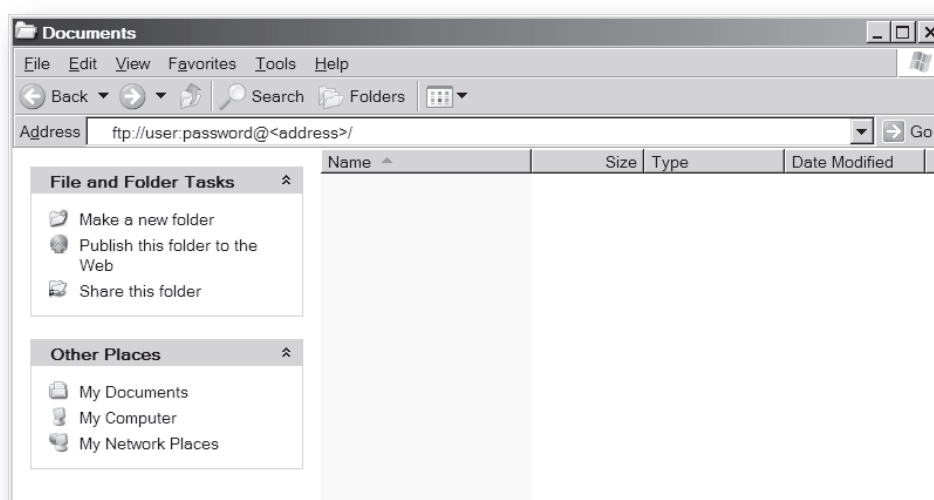
```
Username1:Password1
Username2:Password2
Username3:Password3
```

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

FTP Connection Example (Windows Explorer)

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

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



Web Server

General

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

The web server communicates through port 80.

See also...

- 7-1 “Server Side Include (SSI)”
- 4-3 “IP Access Control”

Protected Files

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

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

Default Web Pages

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

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

These virtual files are:

\index.htm	- Points to the contents of config.htm
\config.htm	- Configuration frame page
\configform.htm	- Configuration form page
\configform2.htm	- Configuration form page
\store.htm	- Configuration store page
\logo.jpg	- HMS logo
\configuration.gif	- Configuration picture
\boarder.bg.gif	- picture
\boarder_m_bg.gif	- picture

Authorization

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

File Format:

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

• List of approved users.


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

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

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

Example:

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

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

[AuthName]
Yeah. Whatsda passwoid?
```

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

Content Types

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

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

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

File Format:

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

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

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

Server Side Include (SSI)

General

Server Side Include (from now on referred to as SSI) functionality enables dynamic content to be used on web pages and in email messages.

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

Syntax

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

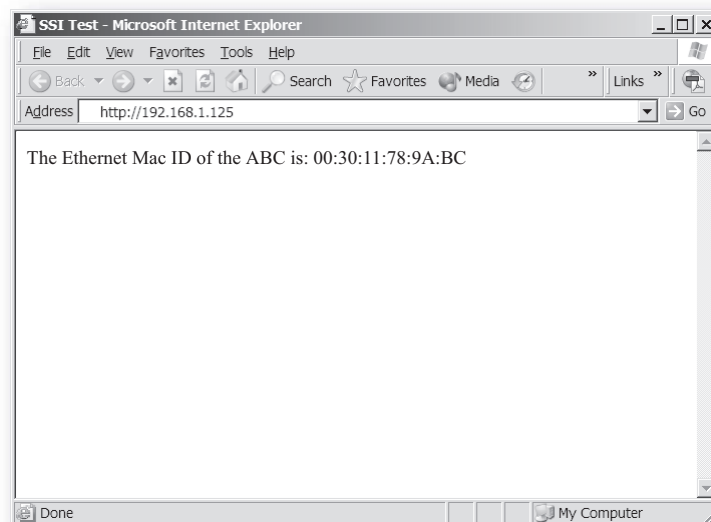
```
<?--#exec cmd_argument='XXXXXXXXXXXXXXXXXXXXX' -->
```

Example

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

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

Resulting webpage:



Functions

DisplayMacID

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

Syntax:

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

DisplaySerial

This function returns the serial number of the network interface.

Syntax:

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

DisplayFWVersion

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

Syntax:

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

DisplayBLVersion

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

Syntax:

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

DisplayIP

This function returns the currently used IP address.

Syntax:

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

DisplaySubnet

This function returns the currently used Subnet mask.

Syntax:

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

DisplayGateway

This function returns the currently used Gateway address.

Syntax:

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

DisplayDNS1

This function returns the address of the primary DNS server.

Syntax:

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

DisplayDNS2

This function returns the address of the secondary DNS server.

Syntax:

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

DisplayHostName

This function returns the hostname.

Syntax:

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

DisplayDomainName

This function returns the default domain name.

Syntax:

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

DisplayDhcpState

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

Syntax:

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

DisplayDhcpSupport

This function returns 'Arg1' if DHCP is supported, and 'Arg2' if it is not.

Syntax:

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

DisplayEmailServer

This function returns the currently used SMTP server address.

Syntax:

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

DisplaySMTPUser

This function returns the username used for SMTP authentication.

Syntax:

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

DisplaySMTPPwd

This function returns the password used for SMTP authentication.

Syntax:

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

DisplayStationName

This function returns the PROFINET Station Name.

Syntax:

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

DisplayStationType

This function returns the PROFINET Station Type.

Syntax:

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

DisplayVendorID

This function returns the PROFINET Vendor ID.

Syntax:

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

DisplayDeviceID

This function returns the PROFINET DeviceID.

Syntax:

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

StoreEtnConfig

Note: This function cannot be used in email messages.

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

Syntax:

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

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

Accepted fields in form:

```
SetIp
SetSubnet
SetGateway
SetEmailServer
SetDhcpState - value "on" or "off"
SetDNS1
SetDNS2
SetHostName
SetDomainName
SetSMTPUser
SetSMTPPwd
```

Default output:

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

GetText

Note: This function cannot be used in email messages.

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

Syntax:

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

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

Default output:

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

printf

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

Syntax:

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

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

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

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

Character	Argument type, Converted to
d, i	byte, short; decimal notation (For signed representation. Use signed argument)
o	byte, short; octal notation (without a leading zero).
x, X	byte, short; hexadecimal notation (without a leading 0x or 0X), using abcdef for 0x or ABCDEF for 0X.
u	byte, short; decimal notation.
c	byte, short; single character, after conversion to unsigned char.
s	char*; characters from the string are printed until a "\0" is reached or until the number of characters indicated by the precision have been printed
f	float; decimal notation of the form [-]mmm.ddd, where the number of d's is specified by the precision. The default precision is 6; a precision of 0 suppresses the decimal point.
e, E	float; decimal notation of the form [-]m.ddddd e+-xx or [-]m.ddddd E+-xx, where the number of d's specified by the precision. The default precision is 6; a precision of 0 suppresses the decimal point.
g, G	float; %e or %E is used if the exponent is less than -4 or greater than or equal to the precision; otherwise %f is used. Trailing zeros and trailing decimal point are not printed.
%	no argument is converted; print a %

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

Argument	Description
InReadSByte(<i>offset</i>)	Read a signed byte from position <i>offset</i> in the Input Data area
InReadUByte(<i>offset</i>)	Read an unsigned byte from position <i>offset</i> in the Input Data area
InReadSWord(<i>offset</i>)	Read a signed word from position <i>offset</i> in the Input Data area
InReadUWord(<i>offset</i>)	Read an unsigned word from position <i>offset</i> in the Input Data area
InReadSLong(<i>offset</i>)	Read a signed longword from position <i>offset</i> in the Input Data area
InReadULong(<i>offset</i>)	Read an unsigned longword from position <i>offset</i> in the Input Data area
InReadString(<i>offset</i>)	Read a string (char*) from position <i>offset</i> in the Input Data area
InReadFloat(<i>offset</i>)	Read a floating point (float) value from position <i>offset</i> in the Input Data area
OutReadSByte(<i>offset</i>)	Read a signed byte from position <i>offset</i> in the Output Data area
OutReadUByte(<i>offset</i>)	Read an unsigned byte from position <i>offset</i> in the Output Data area
OutReadSWord(<i>offset</i>)	Read a signed word (short) from position <i>offset</i> in the Output Data area
OutReadUWord(<i>offset</i>)	Read an unsigned word (short) from position <i>offset</i> in the Output Data area
OutReadSLong(<i>offset</i>)	Read a signed longword (long) from position <i>offset</i> in the Output Data area
OutReadULong(<i>offset</i>)	Read an unsigned longword (long) from position <i>offset</i> in the Output Data area
OutReadString(<i>offset</i>)	Read a null-terminated string from position <i>offset</i> in the Output Data area
OutReadFloat(<i>offset</i>)	Read a floating point (float) value from position <i>offset</i> in the Output Data area

scanf

Note: This function cannot be used in email messages.

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

Syntax:

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

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

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

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

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

Argument	Description
OutWriteByte(offset)	Write a byte to position <i>offset</i> in the Output Data area
OutWriteWord(offset)	Write a word to position <i>offset</i> in the Output Data area
OutWriteLong(offset)	Write a long to position <i>offset</i> in the Output Data area
OutWriteString(offset)	Write a string to position <i>offset</i> in the Output Data area
OutWriteFloat(offset)	Write a floating point value to position <i>offset</i> in the Output Data area

Default output:

```
Write succeeded
Write failed
```

IncludeFile

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

Syntax:

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

Default output:

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

SaveToFile

Note: This function cannot be used in email messages.

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

Syntax:

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

Default output:

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

SaveDataToFile

Note: This function cannot be used in email messages.

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

The [Append|Overwrite] parameter determines if the specified file shall be overwritten, or if the data in the file shall be appended.

Syntax:

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

Default output:

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

Changing SSI output

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

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

SSI Output String File

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

The files shall have the following format:

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

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

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

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

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

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

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

Example:

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

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

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

Temporary SSI Output change

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

Syntax:

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

Example:

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

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

Email Client

General

The built in email client can send predefined email messages based on trigger-events in Input- and Output Data areas. The client supports SSI, however note that some SSI functions cannot be used in email messages (specified separately for each SSI function).

See also...

- 7-1 “Server Side Include (SSI)”

Server Settings

The ABC needs a valid SMTP server configuration in order to be able to send email messages. These settings are stored in the system file ‘\ethcfg.cfg’.

See also...

- 4-2 “Ethernet Configuration File (‘ethcfg.cfg’)”

Event-Triggered Messages

As mentioned previously, the email client can send predefined messages based on events in the Input- and Output Data areas. In operation, this works as follows:

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

Which events that shall cause a particular message to be sent, is specified separately for each message. For more information, see 8-2 “Email Definitions”.

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

Email Definitions

The email definitions are stored in the following two directories:

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

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

File Format:

[Register]
Area, Offset, Type

[Register Match]
Value, Mask, Operand

[To]
recipient

[From]
sender

[Subject]
subject line

[Headers]
Optional extra headers

[Message]
message body

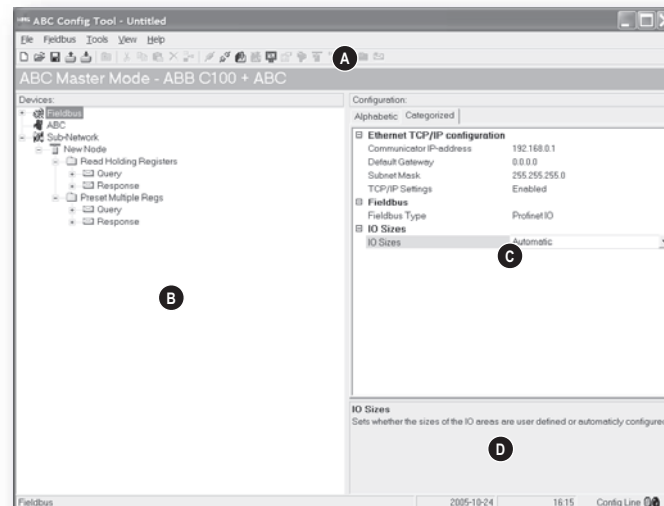
Key	Value	Scanned for SSI
Area	Source area. Possible values: 'IN' (Input Data area) or 'OUT' (Output Data area)	No
Offset	Source offset, written in decimal or hexadecimal.	
Type	Source data type. Possible values are 'byte', 'word', and 'long'	
Value	Used as a reference value for comparison.	
Mask	Mask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

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

Navigating the ABC Config Tool

Main Window

The main window in the ABC Config Tool can be divided in 4 sections as follows:



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

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

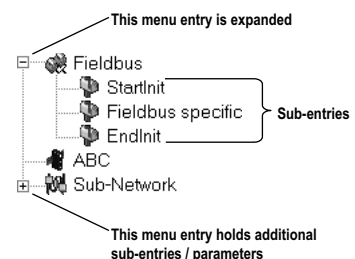
- **B: Navigation Section**

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

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

There are three main levels in the navigation window, namely Fieldbus, ABC and Sub-network.

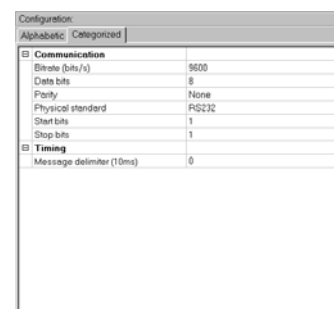
Right-clicking on entries in this section brings out additional selections related to that particular entry.



- **C: Parameter Section**

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

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



Parameter Section

- **D: Information Section**

This section holds information related to the currently selected parameter.



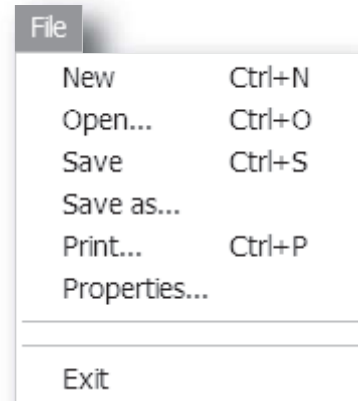
Information Section

Pull-down Menu

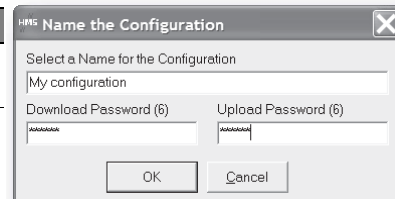
File

This menu features the following entries:

- **New**
Create a new configuration.
See also 12-1 “Configuration Wizards”.
- **Open...**
Open a previously created configuration.
- **Save**
Save the current configuration.
- **Save As...**
Save the current configuration under a new name.
- **Print...**
Send details about the current configuration to a printer.
- **Properties...**
This brings out the following window:



Item	Description
Select a Name for the Configuration	A name for the configuration may be entered here
Download Password(6)	These fields can be used to password-protect the configuration in the gateway.
Upload Password(6)	



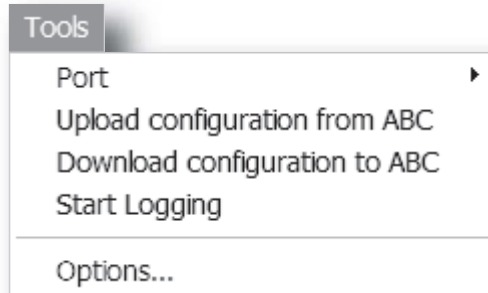
CAUTION: Always keep a copy of the password in a safe place. A lost password cannot be retrieved!

- **Exit**
Close the ABC Config Tool.

Tools

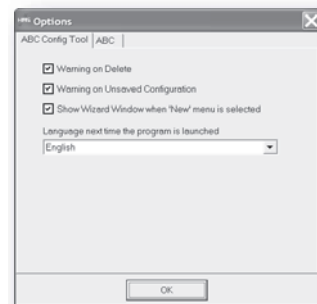
This menu features the following entries:

- Port**
 This entry selects the COM-port used for the configuration of the gateway.
- Upload configuration from ABC**
 Upload the configuration from the gateway to the ABC Config Tool.
- Download configuration to ABC**
 Download the current configuration into the gateway.
- Start Logging**
 Start the Data Logger (see 17-1 “Data Logger”).
 Note that when the Data Logger is active, this menu-entry is changed to ‘Stop Logging’.
- Options**



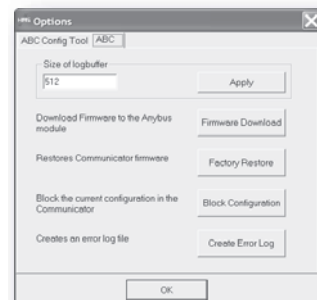
This will bring out the following window:

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



Selecting the ‘ABC’-tab will reveal additional properties:

Item	Description
Size of logbuffer	By default, the Data Logger can log up to 512 entries in each direction. If necessary, it is possible to specify a different number of entries (valid settings range from 1...512). Click ‘Apply’ to validate the new settings. See also 17-1 “Data Logger”.
Firmware Download	Download firmware to the embedded fieldbus interface. Warning: Use with caution.
Factory Restore	Restores the gateway firmware to it's original state (does not affect the embedded fieldbus interface).
Block Configuration	When selected, the downloaded configuration will not be executed by the gateway. Warning: Use with caution.
Create Error log	Creates an error log file



View

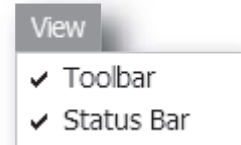
This menu features the following entries:

- **Toolbar**

This entry enables/disables the toolbar icons at the top of the main window.

- **Status Bar**

This entry enables/disables the status bar at the bottom of the main window.



Help

This menu features the following entries:

- **Contents**

Display the table of contents of the on-line help system.

Note: At the time of writing, no on-line help system exists.

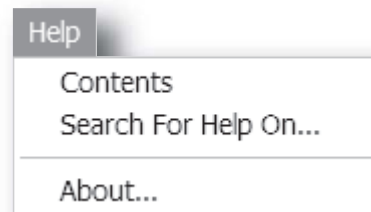
- **Search For Help On...**

Search for a particular topic in the on-line help system.

Note: At the time of writing, no on-line help system exists.

- **About...**

Display general information about the gateway and the current build of ABC Config Tool.



Toolbar Icons

The toolbar features icons for the most commonly used functions.

- **New, Open & Save**

See 9-2 “File”.



- **Upload from ABC & Download to ABC**

See 9-3 “Tools”.



- **Up one Level**

Clicking on this icon will move the selection in the navigation section.



- **Cut, Copy, Paste, Delete, Insert**

These icons are used for common editing functions in the navigation section.



- **Connect**

Clicking on this icon will cause the ABC Config Tool to attempt to connect to the gateway.



- **Disconnect**

Clicking on this icon will cause the ABC Config Tool to disconnect from the gateway.



- **Start Logging & Stop Logging**

See 9-3 “Tools” & 17-1 “Data Logger”.



- **Sub-Network Monitor**

Clicking on this icon will launch the Sub-network Monitor (see 15-1 “Sub Network Monitor”).



- **Add Command**

This icon is used to add commands to the currently selected node.



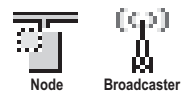
- **Add Mailbox**

(Advanced functionality, see 20-1 “Mailbox Editor”)



- **Add Node & Add Broadcaster**

These icons are used to add nodes to the configuration.



- **Node Monitor**

Clicking on this icon will launch the Node Monitor (see 16-1 “Node Monitor”)



- **Add Transaction(s)**

These icons are used to add transactions to the currently selected node.



Basic Settings

Fieldbus Settings

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

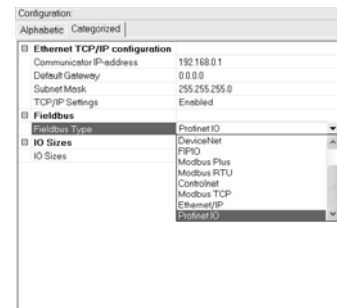


General

During start-up the fieldbus interface of the ABC is initialized to fit the configuration created in the ABC Config Tool. Optionally, some initialisation parameters can be set manually to provide better control over how the data shall be treated by the ABC.

Fieldbus Type

The ABC Config Tool supports a wide range of networking systems. Make sure that this parameter is set to 'Profinet IO'.

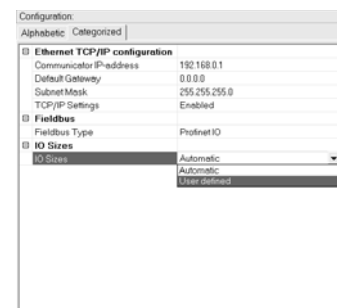


Fieldbus Type

Fieldbus Settings

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

- **Communicator IP-address**
(see 4-1 "Basic Network Configuration")
- **Default Gateway**
(see 4-1 "Basic Network Configuration")
- **Subnet Mask**
(see 4-1 "Basic Network Configuration")
- **TCP/IP Settings**
Enabled - Use the settings above.
Disabled - Use the settings stored in 'ethcfg.cfg'



IO Sizes

IO Sizes

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

- **Automatic**
All data will be represented as I/O Data.
(see also 2-7 "Data Representation (IO Data & Record Data)")
- **User defined**
Additional parameter properties appear; 'IO Size In' and 'IO Size Out'. The specified amount, starting at address 0x0000 of the respective memory buffers, will be reserved for and represented as I/O Data. The remainder will be reserved for Parameter Data.
(see also 2-7 "Data Representation (IO Data & Record Data)")

ABC Parameters

(Select 'ABC' in the Navigation Section to gain access to the parameters described in this section).



Interface

Currently, only serial communications is supported.

Status / Control Word

(See 19-1 “Control and Status Registers”).

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

Module Reset

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

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

Protocol Mode

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

Value	Description
Generic Data Mode	This mode is primarily intended for Produce & Consume-based protocols, where there are no Master-Slave relationship between the gateway and the nodes on the sub-network.
Master Mode	This mode is intended for 'Query & Response'-based protocols, where a single Master exchanges data with a number of Slaves.

See also 2-4 “Protocol Modes”.

Statistics

The Transmit- and Receive Counters indicate how many transactions that have successfully been exchanged on the sub-network. This feature 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.

Both counters are enabled by setting 'Statistics' to 'Enabled'.

Sub-Network Parameters

(To gain access to the parameters described in this section, select 'Sub Network' in the Navigation Section).



Communication

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

Parameter	Description	Valid Settings
Bit rate	Selects the bit rate	1200...57600
Data bits	Selects the number of data bits	7, 8
Parity	Selects the parity mode	None, Odd, Even
Physical standard	Selects the physical interface type	RS232, RS422, RS485
Start bits	Number of start bits.	1
Stop bits	Number of stop bits.	1, 2

Start- and End Character

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

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

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

Timing (Message Delimiter)

The parameters in this category differs slightly between the different Protocol Modes.

- **Master Mode**

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

- **Generic Data Mode**

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

Nodes

General

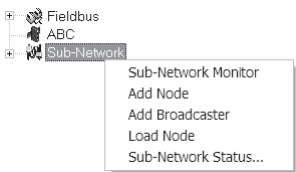
In ABC Config Tool, a node represents a single device on the network. While the gateway doesn't feature a scanlist in the traditional sense, all nodes, and their transactions, will be processed in the order they have been defined in the ABC Config Tool.

The maximum number of nodes that can be created in the ABC Config Tool is 31.

Adding & Managing Nodes

(Right-click on 'Sub Network' in the Navigation Section to gain access to these functions)

Function	Description
Paste	Paste a node from the clipboard
Sub Network Monitor	Launch the subnet monitor (15-1 "Sub Network Monitor")
Add Node	Add a node to the configuration
Add Broadcaster ^a	Add a broadcaster node to the configuration
Load Node	Add a previously saved node
Sub-Network Status...	View diagnostic information about the sub-network



a. This function is only available in Master Mode.

Node Parameters

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

Parameter	Description
Slave Address	The value entered here may be used to set the node address in certain commands. For more information, see 14-3 "The Command Editor".



Transactions

General

As mentioned previously, transactions are representations of the actual serial telegrams exchanged on the serial sub-network. While the gateway doesn't feature a scanlist in the traditional sense, all nodes, and their transactions, will be processed in the order they have been defined in the ABC Config Tool.

Transactions are handled slightly differently in the two protocol modes:

- **Master Mode**

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

- **Generic Data Mode**

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

Theoretically, the gateway supports up to 100 transactions. The actual number may however be less depending on the memory requirements of the defined transactions.

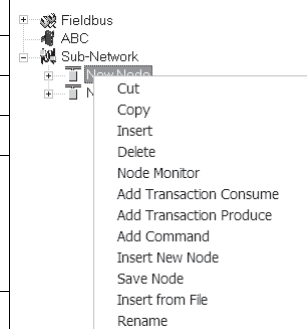
Adding & Managing Transactions

(Right-click on a node in the Navigation Section to gain access to these functions)

Function	Description
Cut	Cut a node to the clipboard
Copy	Copy a node to the clipboard
Insert	Insert a node from the clipboard
Delete	Delete a node
Node Monitor	Launch the node monitor (16-1 "Node Monitor")
Add Transaction(s) ^a	On regular nodes, this adds a Query and a Response. The two transactions will be grouped in order to increase readability. On the Broadcaster, a single transaction will be added.
Add Transaction Consume ^b	Add a 'Consume'-transaction
Add transaction Produce ^b	Add a 'Produce'-transaction
Add Command	Add pre-defined transactions to the node
Insert New Node	Insert a new node above the currently selected one
Save Node	Save the selected node
Insert from File	Insert a previously saved node above the currently selected node
Rename	To increase readability, each node can be given a unique name using this function

a. Only available in Master Mode

b. Only available in Generic Data Mode



Transaction Parameters (Master Mode)

Parameters (Query & Broadcast)

(To gain access to these parameters, select a Query- or Broadcast- transaction in the Navigation Section)

Parameter	Description
Minimum time between broadcasts (10ms)	This parameter specifies how long the gateway shall wait after transmitting a broadcast transaction before processing the next entry in the scanlist. The value should be set high enough to allow the slave devices time to finish the handling of the broadcast. The unit is milliseconds (ms) and the entered value is multiplied by 10, which means that the shortest time is 10 ms. Note: This setting is only relevant for the Broadcaster node.
Offline options for field-bus	This parameter specifies the action to take for this transaction if the higher level network goes off-line. This affects the data that is sent to the sub-network. • Clear - The data destined for the slave-devices is cleared (set to zero) • Freeze - The data destined for the slave-device is frozen • NoScanning -The updating of the sub-network is stopped
Offline options for sub-network	This parameter specifies the action to take for this transaction if the sub-network goes off-line. This affects the data that is reported to the control system. • Clear - Data is cleared (0) on the higher level network if the sub-network goes offline • Freeze - Data is frozen on the higher level network if the sub-network goes offline
Reconnect time (10ms)	This parameter specifies how long the gateway shall wait before attempting to re-connect a disconnected node. A node will be disconnected in case the maximum number of retries (below) has been reached. The unit is milliseconds (ms) and the entered value is multiplied by 10, which means that the shortest time is 10 ms. Note: This setting is not relevant for the Broadcaster node.
Retries	This parameter specifies how many times a timeout may occur in sequence before the node is disconnected.
Timeout time (10ms)	This parameter specifies how long the gateway will wait for a response from a node. If this time is exceeded, the gateway will re-transmit the Query until the maximum number of retries (see above) has been reached. The unit is milliseconds (ms) and the entered value is multiplied by 10, which means that the shortest time is 10 ms.
Trigger byte address	This parameter specifies the location of the trigger byte in internal memory (only relevant when 'Update mode' is set to 'Change of state on trigger').
Update mode	This parameter is used to specify when the transaction shall be sent to the slave: • Cyclically The transaction is issued cyclically at the interval specified in the 'Update time' parameter. • On data change The data area is polled for changes at the time interval defined by Update time. A transaction is issued when a change in data is detected. • Single shot The Query is issued once at start up. • Change of state on trigger The Query is issued when the trigger byte value has changed. This feature enables the control system to notify the gateway when to issue a particular Query. To use this feature correctly, the control system must first update the data area associated with the Query/ transaction, then increase the trigger byte by one. The location of the trigger byte is specified by the 'Trigger byte address' parameter.
Update time (10ms)	This parameter specifies how often the transaction will be issued in steps of 10ms (only relevant when 'Update mode' is set to 'Cyclically').

Parameters (Response)

(To gain access to these parameters, select a Response-transaction in the Navigation Section)

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

Transaction Parameters (Generic Data Mode)

Produce-Transactions

(To gain access to these parameters, select a Produce Transaction in the Navigation Section)

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

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

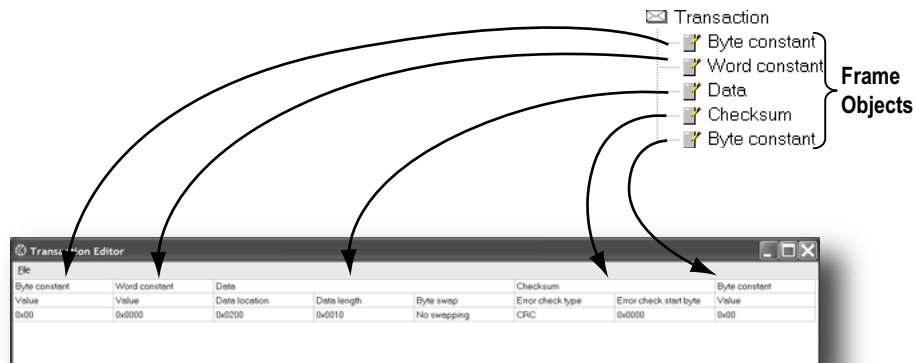
Consume-Transactions

(To gain access to these parameters, select a Consume Transaction in the Navigation Section)

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

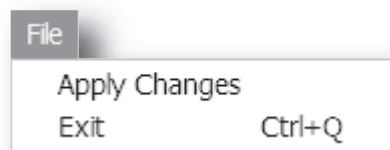
Transaction Editor

The Transaction Editor can be used to edit the individual Frame Objects of a Transaction. The same settings are also available in the Parameter Section of the Main Window, however the Transaction Editor presents the Frame Objects in a more visual manner.



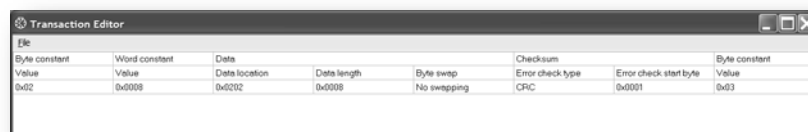
To edit the value of a parameter, click on it and enter a new value using the keyboard. When editing transactions which are based on pre-defined commands, certain parts of the transaction may not be editable.

The File-menu features the following entries:



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

Example:



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

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

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

Frame Objects

General

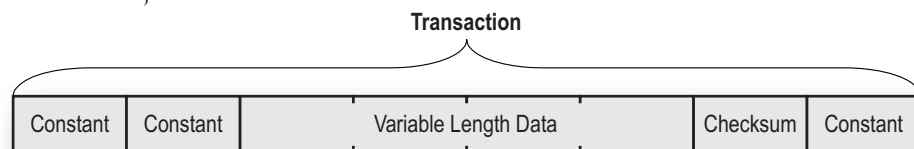
Each transaction consists of Frame Objects which makes up the serial telegram frame. Each Frame Object specifies how the gateway shall interpret or generate a particular part of the telegram.

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

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

Example:

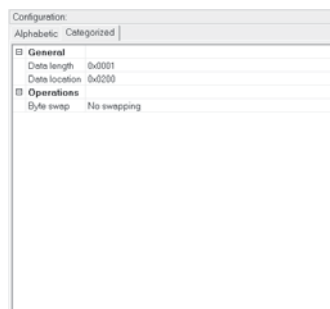
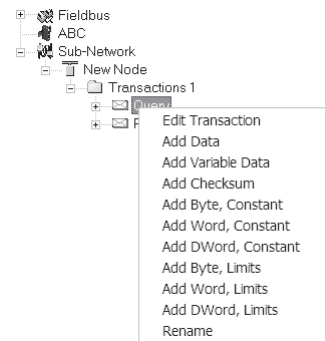
The following Transaction consists of several frame objects; three constants, a data object, and a checksum object.



Adding and Editing Frame Objects

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

The entry called 'Transaction Editor' will launch the Transaction Editor, which is used to edit transactions and frame objects in a more visual manner. For more information, see 12-5 "Transaction Editor".



Data Object, Parameters

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

It is also possible to edit the frame objects in a transaction in a more visual manner using the Transaction Editor, see 12-5 "Transaction Editor"

Constant Objects (Byte, Word, Dword)

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

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

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

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

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

Parameter	Description
Value	Constant value

Limit Objects (Byte, Word, Dword)

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

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

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

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

There are 3 types of interval objects:

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

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

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

Data Object

Data Objects are used to represent raw data as follows:

- **Produce/Query Transactions**
The specified data block is forwarded from the higher level network to the sub-network.
- **Consume/Response Transactions**
The specified data block is forwarded from the sub-network to the higher level network.

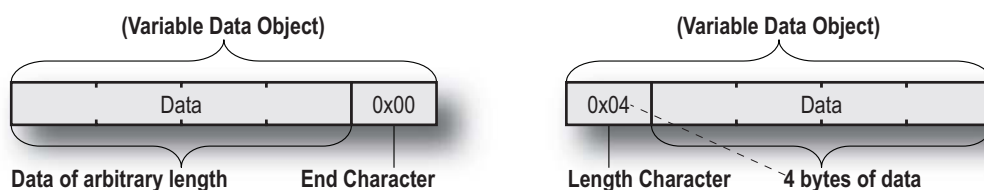
To specify the properties of the object, select it in the Navigation Section enter the desired settings in the Parameter section as follows:

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

Variable Data Object

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

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



- **Produce/Query Transactions**
The specified data block will be forwarded from the higher level network to the sub-network. The control system must supply an End or Length-character in order for the gateway to know the size of the data block.
The End- or Length-character itself may either be forwarded to the sub-network or discarded.
- **Consume/Response Transactions**
The specified data block is forwarded from the sub-network to the higher level network. The End- or Length-character will be generated by the gateway automatically (if applicable)
The End- or Length-character itself may either be forwarded to the higher level network or discarded.

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

Parameter	Description
Byte Swapping	• No Swapping No swapping will be performed on the data • Swap 2 bytes A, B, C, D becomes B, A, D, C • Swap 4 bytes A, B, C, D becomes D, C, B, A
Fill unused bytes	• Enabled^a Fill unused data with the value specified in 'Filler byte'. • Disabled Don't fill
Filler byte	Filler byte value. Only used if 'Fill unused bytes' has been enabled.
Data Location	The offset in the internal memory buffer where the data shall be read from / written to
Object Delimiter	• Length Character Length character is visible in the internal memory buffer but <i>not</i> on the sub-network • Length Character Visible The length character is visible both in the internal memory buffer <i>and</i> on the sub-network. • End Character The end character is visible in the internal memory buffer but <i>not</i> on the sub-network. • End Character Visible The end character is visible both in the internal memory buffer <i>and</i> on the sub-network • No Character^a No End- or Length-character is generated in the internal memory buffer.
End Character Value	End Character value ^b
Maximum Data Length	The maximum allowed length (in bytes) of the variable data object. If the actual length of the data exceeds this value, the message will be discarded. The value must not exceed 300 bytes, which is the maximum data length allowed for one frame.

a. Only relevant for Consume/Response transactions

b. Only used if 'Object Delimiter' is set to 'End Character' or 'End Character Visible'

Checksum Object

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

Parameter	Description
Error Check Start byte	This parameter 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 0xFFFF polynome (Modbus RTU standard) • LRC (1 byte) All bytes are added together as unsigned 8-bit values. The 2's complement of the result will be used as a checksum. • 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. • AddInvASCII (2 bytes) All bytes are added together as unsigned 8-bit values. The lowest 8 bits in the result are inversed and used as a checksum, represented as hexadecimal ASCII (2 bytes).

Commands

General

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

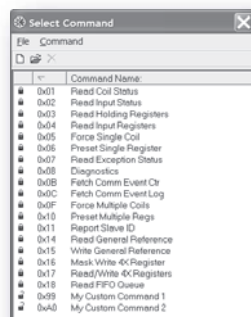
Adding a command to a node actually results in (a) transaction(s) being added according to the directions specified in the command. The Frame Objects in such a transaction may retrieve their values not only from parameters in the parameter section, but also from other sources such as the ‘SlaveAddress’-parameter (see 11-1 “Node Parameters”). In such case, the parameters in the parameter section will be greyed out and cannot be edited directly.

In Master Mode, ABC Config Tool comes pre-loaded with commands for most common Modbus RTU functions. Additional commands can easily be added using the Command Editor (see 14-3 “The Command Editor”). In Generic Data Mode, no pre-defined commands exist, but custom ones may be implemented as desired.

Adding & Managing Commands

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

A list of commands will appear:



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

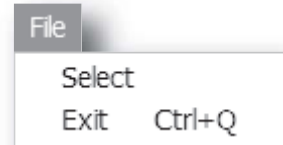
Just like other transactions, the frame objects of added command may be edited in the Navigation/Parameter Section or using the Transaction Editor. Note however that certain frame objects may be locked for editing.

Pull-Down Menu

File

This menu features the following entries:

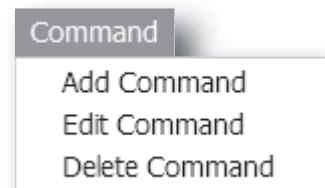
- **Select**
Add the currently selected Command to the node.
- **Exit**
Exit without adding a command to the node.



Command

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

- **Add Command**
Add a custom command to the list, and open the new command in the Command Editor.
See also 14-3 “The Command Editor”.
- **Edit Command**
Edit the currently selected command using the Command Editor.
See also 14-3 “The Command Editor”.
- **Delete Command**
Delete the currently selected command from the list. Note that some commands are fixed and cannot be deleted.



Toolbar Icons

The toolbar features icons for the most commonly used functions.

- **Add Command**
(Same as ‘Add Command’ in the ‘Command’-menu).
- **Edit Command**
(Same as ‘Edit Command’ in the ‘Command’-menu).
- **Delete Command**
(Same as ‘Delete Command’ in the ‘Command’-menu).



Add Command



Edit Command



Delete Command

The Command Editor

General

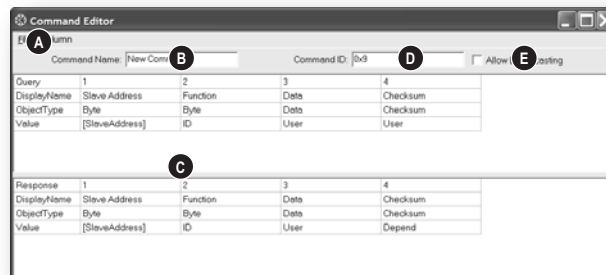
The Command Editor is used to define new commands and edit existing ones. This makes it possible to build a library of commands, which can be stored and re-used at a later stage.

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 uses Master Mode. The procedures involved are similar in General Data Mode, but without the limitations imposed by the Modbus RTU protocol.

Basic Navigation

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



A: Pull-down Menu

See 14-4 "Pull-down Menu".

B: Name of Command

Actual name of the command, in text form.

C: Command Transactions

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

D: Command ID

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

E: Other Settings

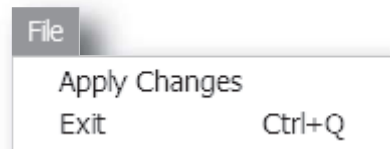
Setting	Description
Allow Broadcasting	Specifies if it is allowed to broadcast the command (only relevant in Master Mode)
Produce	The command is producing data (Generic Data Mode only)
Consume	The command is consuming data (Generic Data Mode only)

Pull-down Menu

File

This menu features the following entries:

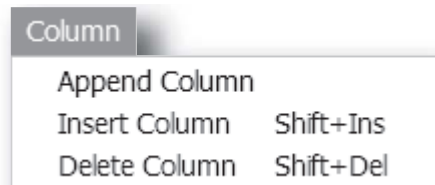
- **Apply Changes**
Save changes and exit to the main window.
- **Exit**
Exit without saving.



Column

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

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



Editing a Command

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

Each column features four rows with the following parameters:

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

Value	Description
Depend	This setting is only relevant for Responses in Master Mode. The value will be retrieved from the corresponding part of the 'Query'-transaction.
Id	The value will be retrieved from the 'Command ID'-setting (see 14-3 "Basic Navigation").
User	The settings associated with the object can be edited by the user.
[SlaveAddress]	The value will be retrieved from the 'SlaveAddress'-parameter (see 11-1 "Node Parameters").
(other settings)	Other settings are no longer supported.

Example: Specifying a Modbus-RTU Command in Master Mode

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

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

Furthermore, each command always consists of a Query and a Response.

- **Example Query**

Query	1	2	3	4
DisplayName	Slave Address	Function	Data	Checksum
Object Type	Byte Object	Byte Object	Data Object	Checksum Object
Value	[SlaveAddress]	ID	User	User
	The value of this byte constant will be set using the 'SlaveAddress' parameter (see 11-1 "Node Parameters").	The value of this byte constant will be set using the 'Command ID'-field.	The size and location of the data associated with this object is determined by the user.	The checksum type etc can be selected by the user. By default, this is set to match the Modbus-RTU standard.

- **Example Response**

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

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

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

Sub Network Monitor

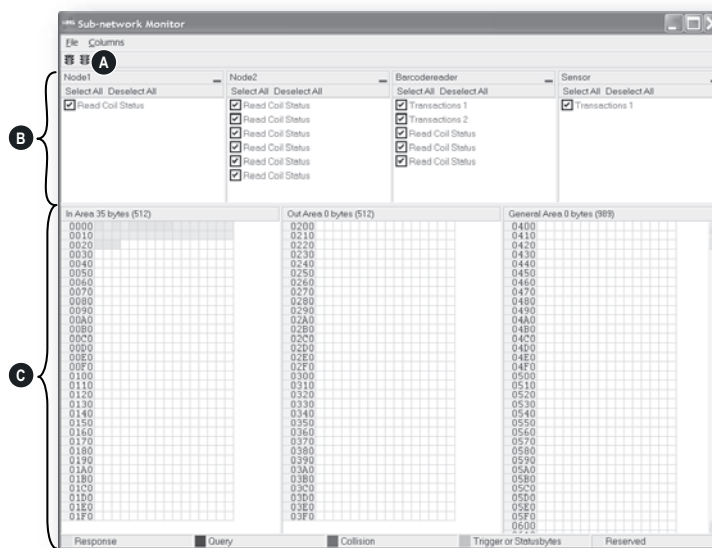
General

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

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

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

Operation



A: Start Network & Stop Network Icons

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



B: Nodes / Transactions

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

C: Monitor Section

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

Colour	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.

Node Monitor

General

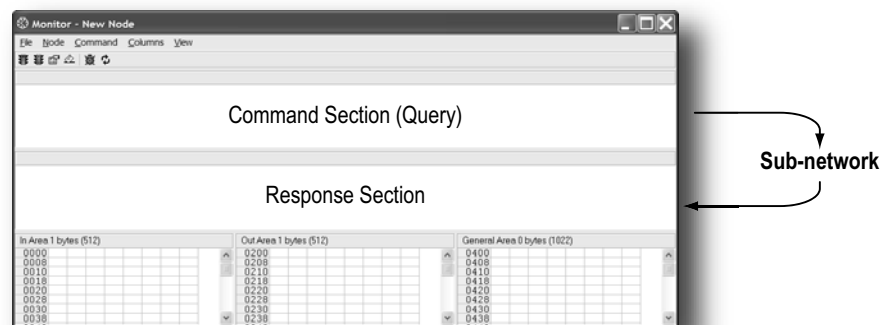
The Node Monitor can provide valuable information when setting up the communication with the sub-network, by allowing individual commands to be issued manually, and monitoring the response (if applicable). It also provides an overview of the memory used by a particular node.

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

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

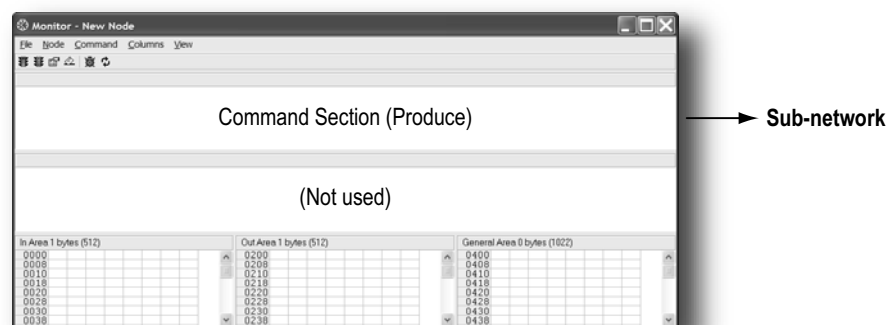
- **Master Mode**

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

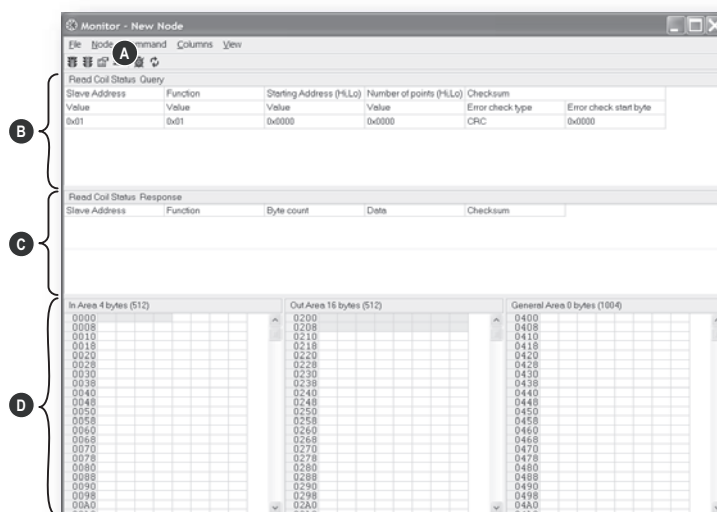


- **Generic Data Mode**

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



Navigating the Node Monitor



A: Pull-down Menu & Toolbar Icons

See 16-3 “Pull-Down Menu” and 16-4 “Toolbar Icons”

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 only)

This section holds the response to the selected Command.

D: Monitor Section

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

The data displayed in this section will be refreshed based on the refresh-icons in the toolbar. For more information, see 16-4 “Toolbar Icons”

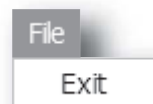
Pull-Down Menu

File

There is only one entry in this menu:

- **Exit**

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



Node

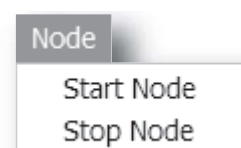
This menu controls the data exchange for the node. This feature can help isolate problems associated with a particular node.

- **Start Node**

Enable the transactions associated with the node.

- **Stop Node**

Disable the transactions associated with the node.



Command

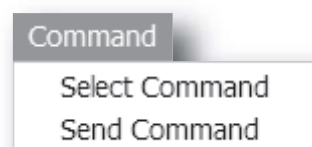
This menu is used to specify and issue a command manually.

- **Select Command**

Select a command to be sent on the sub-network.

- **Send Command**

Send the specified command to the sub-network.



Columns

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

- **Free**

The number of columns depends on the width of the window.

- **8 Multiple**

The number of columns will be fixed to 8.



View

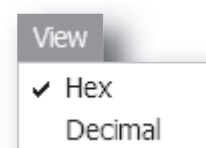
This menu specifies the data representation in the Monitor Section.

- **Hex**

Display the data in hexadecimal format.

- **Decimal**

Display the data in decimal format.



Toolbar Icons

The toolbar features icons for the most commonly used functions.

- **Start Node & Stop Node**

These icons corresponds to the functions in the 'Node'-menu.
See also 16-3 "Node".



Start



Stop

- **Select Command & Send Command**

These icons corresponds to the functions in the 'Command'-menu.
See also 16-3 "Command".



Select



Send

- **Resume Refresh & Stop Refresh**

When enabled, the data displayed in the Monitor Section will be re-freshed cyclically. When disabled, i.e. stopped, the data will have to be refreshed manually using the 'Refresh'-icon (see below).



Stop



Resume

- **Refresh**

When clicking on this icon, the data displayed in the Monitor Section will be re-freshed.



Refresh

Data Logger

General

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

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

Operation

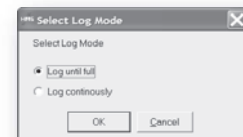
Start & Stop Logging

- **Start logging**
Select 'Start Logging' in the 'Tools'-menu. ABC Config Tool will then prompt for the desired mode of operation, see below.
- **Stop logging**
Select 'Stop Logging' in the 'Tools'-menu. This will open the log-window, see below.

Modes of Operation

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

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

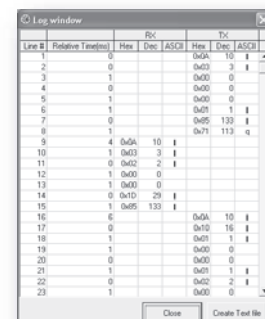


Log Window

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

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

Click 'Close' to exit.



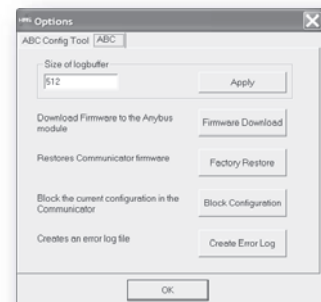
Configuration

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

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

Click 'Apply' to validate the new settings.

Click 'OK' to exit.



Configuration Wizards

General

When creating a new sub network configuration, the ABC Config Tool provides a choice between starting out with a blank configuration, or using a predefined template, a.k.a a wizard.

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

Selecting a Wizard Profile

The following window appears each time the ABC Config Tool is started, or upon selecting the ‘New’ entry in the ‘File’-menu (unless it has been disabled in the ‘Options’-menu, see 9-3 “Tools”).

Currently, the following wizards are available:

- **ABCC ExtLink Wizard**

This wizard is intended for use with the Anybus-CompactCom Modbus-RTU fieldbus communication adapter.

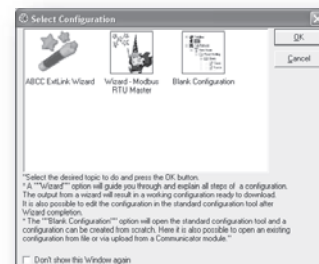
- **Wizard - Modbus RTU Master**

This option is suitable for Modbus RTU-based networks.

See also 18-2 “Wizard - Modbus RTU Master”.

- **Blank Configuration**

This option creates an empty configuration.



Highlight the desired wizard and click ‘OK’ to continue.

Wizard - Modbus RTU Master

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

- **Important Notes:**

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

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

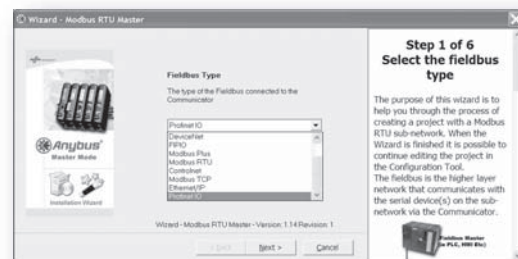
Using this wizard involves the following steps:

Step 1: Communicator Type

Select 'Profinet IO'.

Click 'Next' to continue.

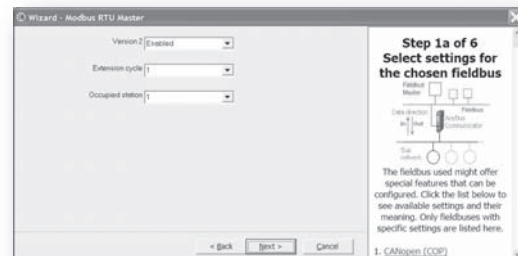
Tip: It is possible to return to a previous menu at any time without losing any settings by clicking 'Previous'.



Step 1a: I/O Sizes

Specify the sizes of the input and output data areas. For more information, see 10-1 "IO Sizes".

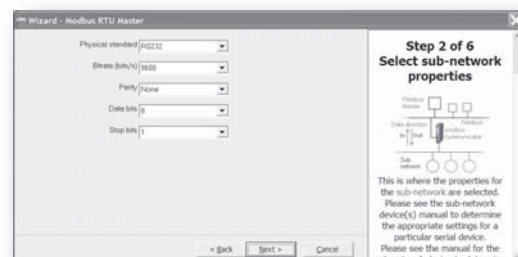
Click 'Next' to continue.



Step 2: Physical Settings

Select the physical properties of the sub network.

Click 'Next' to continue.



Steps 3 - 6

Consult the on line help system for further information.

Control and Status Registers

General

The Control- and Status Registers are disabled by default, but can be enabled using the ABC Config Tool (see 10-2 “Status / Control Word”). These registers form an interface for exchanging status information between the sub-network and the fieldbus control system.

The main purpose of these registers is to...

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

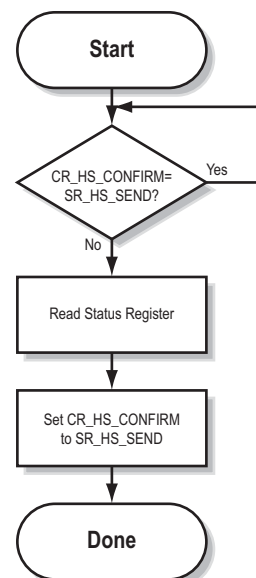
If enabled, these registers occupy the first two bytes in the Input- and Output Data areas (0x000-0x001 and 0x200-0x201 respectively), which means they can be accessed from the fieldbus just like any other data in these areas.

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

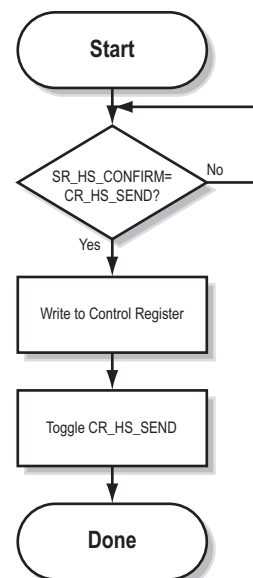
Handshaking Procedure

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

Read Status Register



Write to Control Register



Data Consistency

The 'Data Valid'-bits in the Control- and Status Registers are used to ensure data consistency during start-up and fieldbus off-line/on-line transitions.

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

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

State Machine

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

A: Offline (No data exchange)

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

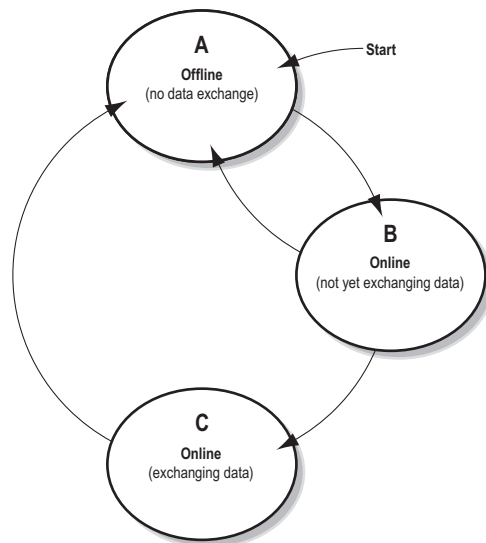
B: Online (Not yet exchanging data)

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

C: Online (Exchanging data)

Exchanging valid data in both directions.

If the gateway goes offline on the fieldbus, shift to state A.



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

Latency

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

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

Status Register Contents (Gateway to Control System)

General Information

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

bit(s)	Name	Description
15	Send (SR_HS_SEND)	These bits control the handshaking towards the fieldbus control system.
14	Confirm (SR_HS_CONFIRM)	See also... - 19-1 "Handshaking Procedure" - 19-5 "Control Register Contents (Control System to Gateway)"
13	Data Valid (Master Mode Only)	This bit is set when all transactions have been executed successfully at least once. Once set, it will not change. 1: Data Valid 0: Data not Valid Note: This bit is not used in Generic Data Mode.
12... 8	Status Code	This field holds the last status report from the gateway.
7... 0	Data	See also... - 19-3 "Status Codes in Master Mode" - 19-3 "Status Code in Generic Data Mode"

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

Status Codes in Master Mode

(This table is valid only in Master Mode).

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

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

Status Code in Generic Data Mode

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

Code	Condition	Type	Data	Description
0x00	Invalid Transaction Counter Updated	Error	Counter	The number of invalid transactions (i.e. received transactions which doesn't match any of the Consume-transactions defined in the sub-network configuration) has increased.

Code	Condition	Type	Data	Description
0x01	Frame Error	Warning	-	End character is enabled, but a message delimiter timeout occurs prior to receiving it.
0x02	Offline Timeout Counter Updated	Error	Counter	The of number of timed out Consume-transactions has increased. See also... - 12-4 "Consume-Transactions" (Offline timeout time)
0x03	Buffer Overrun	Warning	-	A node returned more data than expected - or - the gate-way was unable to finish processing a message prior to receiving a new one.
0x04	Other Error	Error	-	Undefined error
0x1F	No Error	Warning	-	No errors

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

Control Register Contents (Control System to Gateway)

General Information

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

bit(s)	Name	Description
15	Confirm (CR_HS_CONFIRM)	These bits control the handshaking towards the gateway.
14	Send (CR_HS_SEND)	See also... - 19-1 "Handshaking Procedure" - 19-3 "Status Register Contents (Gateway to Control System)"
13	Data Valid	This bit controls data consistency (see 19-2 "Data Consistency"). 1: Output Area valid; exchange data on the sub-network 0: Output Area not valid; do not exchange data on the sub-network Note: This bit is only relevant if the Control/Status Registers are set as 'Enabled'
12	Execute Command	If set, the specified command will be executed by the gateway (see below).
11... 8	Control Code	This field holds commands which can be executed by the gateway (see below).
7... 0	Data	See also... - 19-5 "Control Codes in Master Mode" - 19-5 "Control Codes in Generic Data Mode"

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

Control Codes in Master Mode

(This table is valid only in Master Mode).

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

Control Codes in Generic Data Mode

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

Advanced Fieldbus Configuration

General

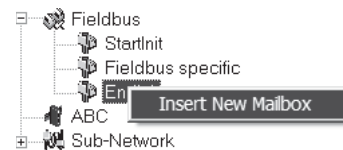
The fieldbus interface of the gateway consists of an embedded Anybus-S communication interface. Normally, the Anybus-S configuration settings are set up automatically by the gateway. However, advanced users can configure the Anybus-S card for specific features. This chapter assumes that the reader is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

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

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

Mailbox Editor

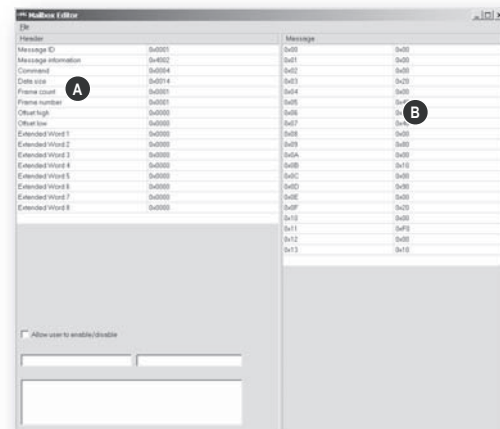
To add a mailbox message to the configuration, right-click on 'EndInit' and select 'Insert New Mailbox'.



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

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

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

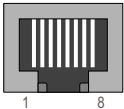


For more information about fieldbus specific mailbox messages, consult the separate Anybus-S Fieldbus Appendix for the fieldbus you are using. For general information about the Anybus-S platform, consult the Anybus-S Design Guide.

Connector Pin Assignments

PROFINET Connector (Ethernet)

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



Power Connector

Pin	Description
1	+24V DC
2	GND

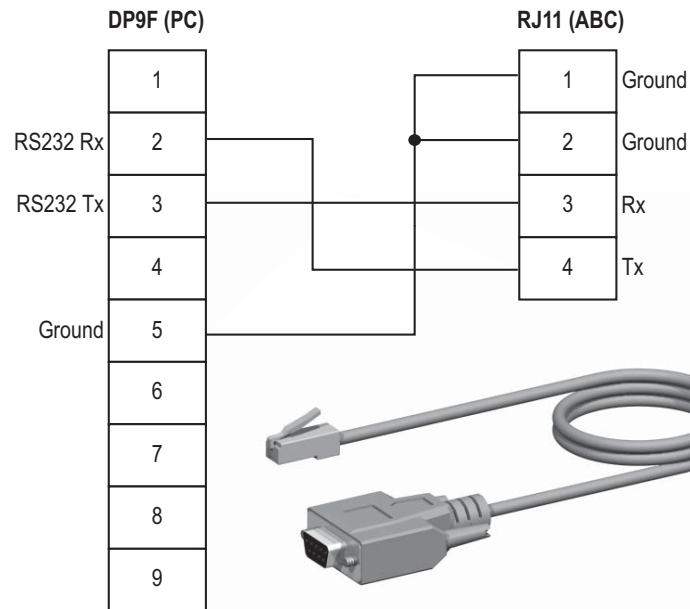


Notes:

- Use 60/75 or 75×C copper (CU) wire only.
- The terminal tightening torque must be between 5... 7 lbs-in (0.5... 0.8 Nm)

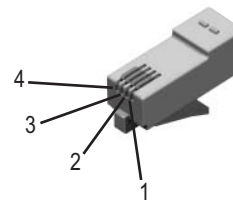
PC Connector

Configuration Cable Wiring



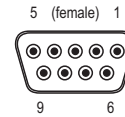
RJ9 (ABC)

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



DB9F (PC)

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



Sub-network Interface

General Information

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

Bias Resistors (RS485 Only)

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

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

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

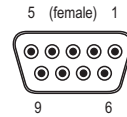
Termination (RS485 & RS422 Only)

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

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

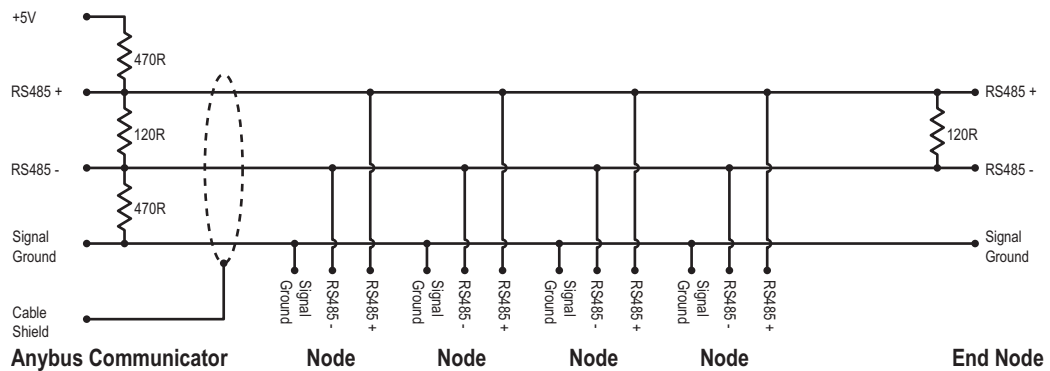
Connector Pinout (DB9F)

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

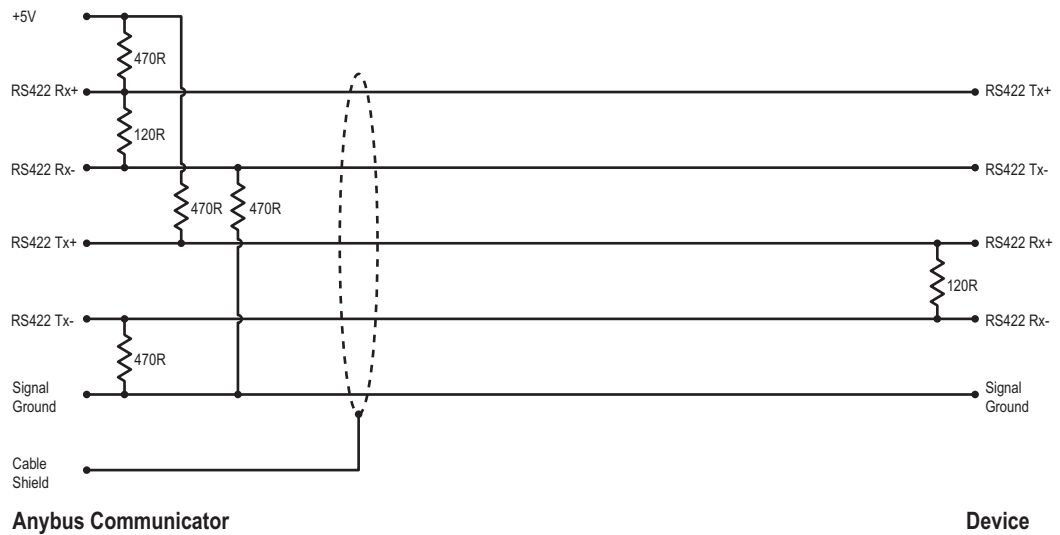


- a. Connecting this signal directly Protective Earth (PE) of other nodes may, in case of grounding loops etc., cause damage to the on-board serial transceivers. It is therefore generally recommended to connect it only to Signal Ground (if available) of other nodes.

Typical Connection (RS485)

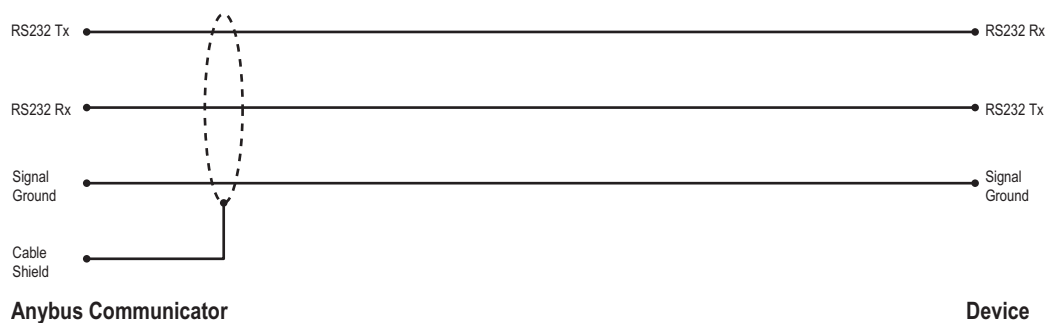


Typical Connection (RS422 & 4-Wire RS485)



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

Typical Connection (RS232)



Technical Specification

Mechanical Properties

Housing

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

Dimensions

120 mm x 75 mm x 27 mm, L x W x H (inches: 4.72" x 2.95" x 1.06"; L x W x H)

Electrical Characteristics

Power Supply

Power: 24V $\pm$ 10%

Power Consumption

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

Environmental Characteristics

Relative Humidity

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

Temperature

Operating:	$\pm 0^{\circ}\text{C}$ to $+55^{\circ}\text{C}$
Non Operating:	-25°C to $+85^{\circ}\text{C}$

Regulatory Compliance

EMC Compliance (CE)

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

- **EN 50082-2 (1993)**
EN 55011 (1990) Class A

- **EN 61000-6-2 (1999)**
EN 61000-4-3 (1996) 10V/m
EN 61000-4-6 (1996) 10V/m (all ports)
EN 61000-4-2 (1995) ±8kV Air Discharge
 ±4kV Contact discharge
EN 61000-4-4 (1995) ±2kV Power port
 ±1kV Other ports
EN 61000-4-5 (1995) ±0.5kV Power ports (DM/CM)
 ±1kV Signal ports

UL/c-UL compliance

The certification has been documented by UL in file E214107.

Galvanic isolation on sub-network interface

- **EN 60950-1 (2001)**
Pollution Degree 2
Material Group IIb
250 V_{RMS} or 250 VDC Working voltage
500 V Secondary circuit transient rating

Troubleshooting

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

ASCII Table

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

