Fieldbus Appendix

AnyBus®-IC EIP

Rev. 1.12

HMS Industrial Networks AB

	☎	✉
Germany	+49- 721 - 96472 - 0	sales-ge@hms-networks.com
Japan	+81- 45 - 478 -5340	sales-jp@hms-networks.com
Sweden	+46- 35 - 17 29 20	sales@hms-networks.com
U.S.A	+1- 773 - 404 - 3486	sales-us@hms-networks.com

Table of Contents

Chapter 6 FTP Server

Chapter 7 Telnet Server

Chapter 8 SSI (Server Side Include) Script Functionality

Chapter 9 Web Server

Chapter 10 Email Client

Chapter 11 Modbus/TCP

Chapter 12 EtherNet/IP

Chapter 13 Fieldbus Specific Parameters

Chapter 14 Fieldbus Specific HOS Classes

Appendix A Flow Charts

Appendix B Creating an Application Parameter

Appendix C Parameter Data Mapping

Appendix D Firmware Upgrade

Appendix E Electrical Characteristics

Appendix F Environmental Specification

About This Manual

How To Use This Manual

This document is intended to be used in conjunction with the AnyBus-IC Design Guide. The reader of this document is expected to have basic knowledge in the Ethernet network system, and communication systems in general. Please consult the general AnyBus-IC Design Guide for general information about the AnyBus-IC platform.

Note: This document describes the functionality provided by the latest firmware release. Some features may be missing or working somewhat differently in older firmware releases. Please contact HMS to obtain the latest version.

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.

Related Documentation

Document name	Author	Web
Open Modbus/TCP Specification	Schneider Automation	www.modbus.org
RFC 821	Network Working Group	-
RFC 1918	Network Working Group	-
ENIP Specifications	ControlNet International and ODVA	www.odva.org
AnyBus-S Parallel Design Guide	HMS	www.hms-networks.com

Revision List

Revision	Date	Author	Chapter	Description
1.00	2003-04-02	PeP	All	Preliminary version
1.10	2003-05-30	PeP	All	First release
1.11	2003-10-16	PeP/ToT	Appendix D Chapter 14	Corrected power consumption. Removed incorrect text references
1.12	2003-11-10	PeP	All	Minor corrections and adjusmtents

Conventions used in this manual

The following conventions are used throughout this manual:

- Numbered lists provide sequential steps
- Bulleted lists provide information, not procedural steps
- The term ‘module’ is used when referring to the AnyBus-IC EIP.
- The term ‘application’ is used when referring to the hardware that is connected to the Application Connector.
- Hexadecimal values are written in the format NNNNh or 0xNNNN, where NNNN is the hexadecimal value.
- Binary values are written in the format NNNNb, where NNNN is the binary value.
- 16/32 bit values are written in big endian Motorola format
- Floating point values are in the IEEE Standard 754 format

Support

Europe (Sweden)

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

HMS America

E-mail:	us-support@hms-networks.com
Phone:	+1-773-404-2271
Toll Free:	888-8-AnyBus
Fax:	+1-773-404-1797
Online:	www.hms-networks.com

HMS Germany

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

HMS Japan

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

About the AnyBus-IC EIP

The AnyBus-IC EIP integrates all analog and digital functionality required to communicate on an Ethernet network into a single chip. The module features a web server and email client with Server Side Include (SSI) capabilities, allowing commands to be embedded into HTML code and Email messages, providing user friendly access to I/O and parameter data.

Being a member of the AnyBus-IC family, it can be used with intelligent as well as non-intelligent applications, using a serial communication interface, and/or using external shift-registers to form digital inputs and outputs.

The module exists in two versions:

- AnyBus-IC EIP - EtherNet/IP, Modbus/TCP and IT functionality (This product)
- AnyBus-IC EIT - Modbus/TCP and IT functionality

Features

General

- 10 and 100mbit operation, Full and Half Duplex
- Flexible file system providing both volatile and non-volatile storage areas
- Security framework

IT-Functionality

- Integrated FTP server provides easy file management using standard FTP clients.
- Telnet server featuring a command line interface similar to the MS-DOS™ environment.
- Web server with SSI script capability
- Email client capability with SSI script support

Control Protocols

- **Modbus/TCP**
  The module supports the Modbus/TCP protocol and conforms to the Modbus/TCP specification 1.0.

- **Ethernet/IP**
  The module can act as a group 2 and 3 server on an EtherNet/IP based network.

- **Transparent Socket Interface**
  Other protocols can be implemented on top of TCP/IP or UDP/IP using the transparent socket interface.

Compatible Products

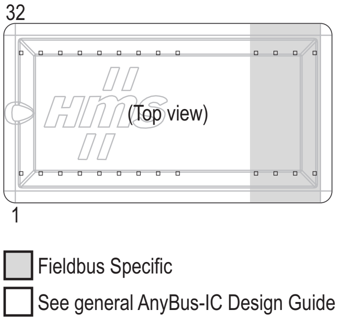
This product is a member of the AnyBus concept of interchangeable fieldbus modules. Standardization of mechanical, electrical and software interfaces ensures that the different AnyBus-IC models are fully interchangeable with only little or no required software and/or hardware adjustments, depending on the application.

Connectors, Switches & Indicators

Application Connector

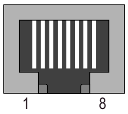
Pin numbers 13-20 on the application connector are used for fieldbus specific signals, see pinout below.

Pin	Pin name	Signal
1-12	(See AnyBus-IC Design Guide)	-
13	FB1	TX+
14	FB2	TX-
15	FB3	RX+
16	FB4	RX-
17	PE	NC
18	Shield	NC
19	FB5	NC
20	FB6	3.3V OUT
21-32	(See AnyBus-IC Design Guide)	-



Ethernet Connector

Pin	Signal
1	TX+
2	TX-
3	RX+
4	-
5	-
6	RX-
7	-
8	-
Housing	Bus Cable Shield (Shielded connector only)



Switches (Fieldbus Specific Input Register)

The module supports limited IP address configuration via the SCC interface on the Fieldbus Specific Input register. This method provides an easy way to configure the module for intranet use. Note that these settings cannot be used on the internet. This is because the used IP address belongs to the private address set, see RFC 1918.

Two kinds of switches are supported: (Consult the AnyBus-IC Design Guide for more information)

- **BCD Switch**

  Two switches are used to specify the last byte of the IP address, one for each decimal digit. Note that this limits the switch range to 1 - 99.

- **Binary Switches**

  8 binary switches are used to specify the last byte of the IP address.

For more information on how the switch is used when configuring the network, see 5-2 “Configuring the IP settings”.

Status Indicators (Fieldbus Specific Output Register)

The Fieldbus Specific Output Register on the SSC interface is used according to the following. The state of these leds can be read using parameter #7 (“LED State”).

- **Link / Activity**

Bit	State	Indication
-	OFF	Device not powered
0 (LSB)	Green	Module connected to an Ethernet network
	Flashing green	RX / TX Activity
1	Red	-

- **Data Rate**

Bit	State	Indication
-	OFF	10Mbit
2	Green	100Mbit
3	Red	-

- **Module Status (MS)**

Bit	State	Indication
-	OFF	Device not powered
4	Green	Device operational
	Flashing green	Device needs commissioning due to missing or incorrect configuration.
5	Red	Major fault, unrecoverable
	Flashing red	Minor fault, recoverable

- **Network Status (NS)**

Bit	State	Indication
-	OFF	No power or no IP address
6	Green	Device has at least one established EIP connection
	Flashing green	Device has no established EIP connections
7 (MSB)	Red	Duplicate IP address detected
	Flashing red	One or more established EIP connections has timed out

- **Power up LED Test Sequence:**

  During power up, all leds are tested according to the EtherNet/IP specification:

Duration	Module Status (MS)	Network Status (NS)	Link / Activity	Data Rate
0.25s	Green	(off)	(off)	(off)
0.25s	Red			
0.25s	Green	Green		
0.25s		Red		
0.25s		(off)	Green	
0.25s			Red	
0.25s			(off)	Green
0.25s				Red
-	(Normal Operation)	(Normal Operation)	(Normal Operation)	(Normal Operation)

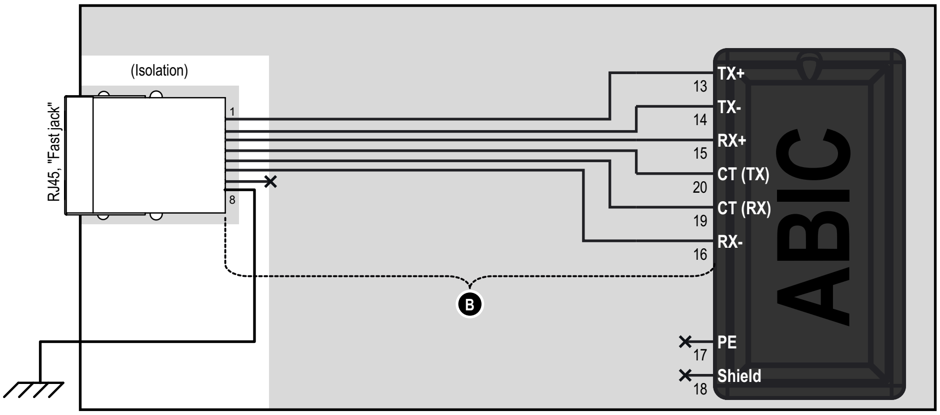
Design Considerations

The PCB is a part of the Physical Layer in that the characteristics of the traces and materials control impedance, capacitance, coupling and voltage withstand. The PCB layout is important in reducing noise ingress and emissions.

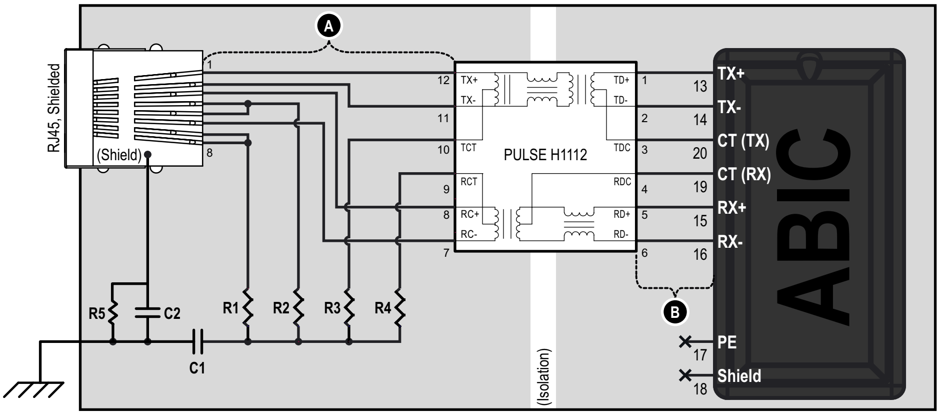
- Earth ground planes and power planes must be properly defined and isolated
- It is important to keep traces short and equal in length:

  A: Connector to transformer (magnetic's)
  B: Transformer to transceiver
- The traces must also match the circuit impedance using micro strip layout techniques. Ethernet TP impedance is 100 Ohms

Example 1: Connector with Integrated Transformer (Fast jack)[1]



Example 2: External Transformer Connection



Ref.	Component	Comments
R1, R2, R3, R4	75 ohm	Network termination
R5	1M ohm	Filter to PE
C1, C2	1nF/2kV	Filter capacitors to PE
PULSE H112	PULSE H112	Other transformers can be used, but may require a different connection/circuit.
RJ45, Shielded	RJ45 connector	-
RJ45, Fast Jack	HFJ11-2450	FastJack (HALO electronics Inc.)

[1] For use with shielded cables only

Filesystem

The filesystem is a fixed-size storage area with a hierarchical directory structure. Any user- or application data can be stored in files within the filesystem. Files can be grouped in directories for increased readability.

The file system provides both non volatile (FLASH) and volatile (RAM) storage. The FLASH disc is a intended for static data such as user HTML files, configuration files etc. The RAM disc area is intended for frequently accessed files such as log files etc. Note that the RAM disc is disabled by default and has to be enabled by the application using parameter #131 ("RAM Disc Path")

The filesystem can be accessed via the network using FTP, Telnet and HTTP. The application can access the filesystem using Modbus Object Messaging via HOS object class 0x86 ("File System Object"). Depending on security level, different users can have access to different files and directories.

Restrictions

- **Case Sensitivity**
  The file system is case sensitive. This means that the file 'AnyBus.txt' is not identical to the file 'AnyBus.TXT'.

- **Filename / Pathname length**
  Filenames can be a maximum of 48 characters long. Pathnames can be 256 characters in total, filename included.

- **File size**
  The file size is not restricted. Naturally, a file cannot be larger than the available space.

- **Free space**
  Approximately 1.4MB non-volatile / 1.0MB volatile

Important Note:

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

The following operations will erase one or more FLASH segments:

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

Security Framework

The file system features two security levels; Admin and Normal. Security level is set at a per user basis, or globally using parameter #124 (“Admin Mode Cfg”). (See 4-2 “Global Admin Mode”).

- **Admin Mode**

  Admin users has full access to the filesystem through FTP and Telnet. This enables the user to access areas of the filesystem, that is restricted or inaccessible in Normal mode.

  The Admin user accounts are defined in the file ‘ad_pswd.cfg’.

- **Normal Mode**

  This mode is recommended for normal operation, so that web pages and other settings are protected from FTP and Telnet access.

  The accounts for normal users are defined in the file ‘sys_pswd.cfg’.

Files within the file system can be protected from web access through username/password authorization, see 4-5 “Password files” and 4-6 ““web_accs.cfg””. It is also possible to configure which IP addresses and what protocols that are allowed to connect to the module, see 4-4 ““ip_accs.cfg””.

Normal Mode

In this mode, the FTP and Telnet servers are enabled only if there is a subdirectory called “\user”. When a normal user connects via FTP or Telnet, this directory will be their root directory. The user will not be able to access files outside this directory and it’s subdirectories.

If user/password protection for FTP and Telnet is required in normal mode, a file called “sys_pswd.cfg” must be placed in the directory “\user\pswd\”. Files in this directory cannot be accessed from a web browser.

The module will run in this mode if parameter #124 (“Admin Mode Cfg”) is not set, and a valid admin password file (See 4-5 “Password files”) is found.

Note: The application has unrestricted access to the filesystem regardless of security settings.

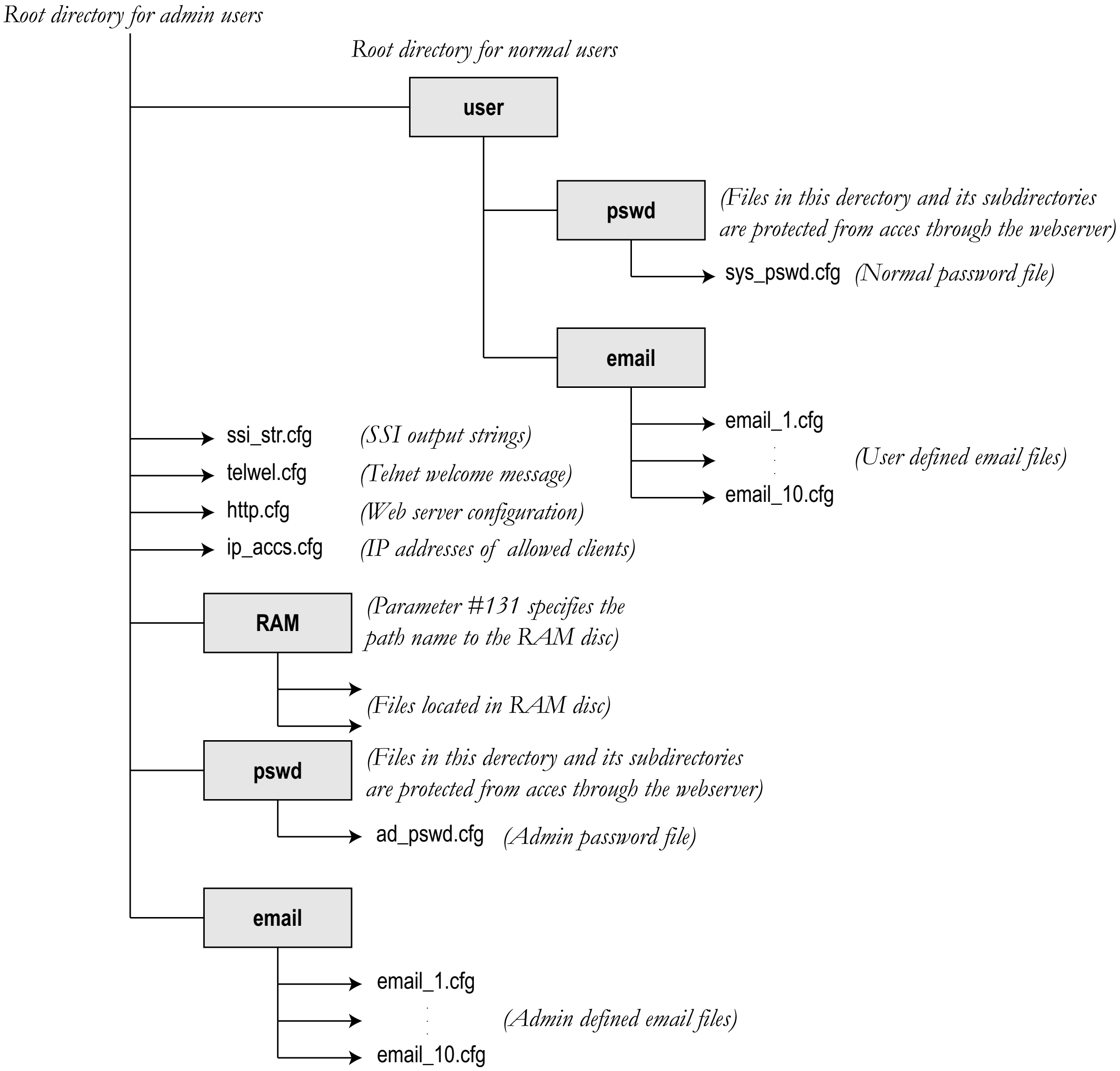
Global Admin Mode

If no admin password file (See 4-5 “Password files”) is found during module startup or if parameter #124 (“Admin Mode Cfg”) has been set, the module will run in Global Admin Mode; i.e. all users will have Admin access rights. No login is needed for Telnet, and the FTP server accepts any username/password combination.

Global Admin Mode is primarily intended for product configuration and development.

Structure

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.



Root directory for admin users
Root directory for normal users
user
pswd
(Files in this directory and its subdirectories are protected from acces through the webserver)
sys_pswd.cfg
(Normal password file)
email
email_1.cfg
(User defined email files)
email_10.cfg
ssi_str.cfg
(SSI output strings)
telwel.cfg
(Telnet welcome message)
http.cfg
(Web server configuration)
ip_accs.cfg
(IP addresses of allowed clients)
RAM
(Parameter #131 specifies the path name to the RAM disc)
(Files located in RAM disc)
pswd
(Files in this directory and its subdirectories are protected from acces through the webserver)
ad_pswd.cfg
(Admin password file)
email
email_1.cfg
(Admin defined email files)
email_10.cfg

Virtual Files

The module also contains a virtual file system containing a set of files used to build the default web configuration web page. The virtual file system can be overwritten or disabled, but not erased; A file with the same name in the file system replaces the file in the virtual file system until it is removed.

The entire virtual file system can be enabled/disabled using parameter #130 (“VFS Enable”). For more information about the virtual files and their contents, see 9-1 “Default Web Pages”

System Files

The module uses these files for configuration purposes. The system files are ASCII files and can be edited with any text editor. Depending on security settings, the files may be inaccessible for normal users. Generally, the module has to be restarted in order for any changes in these files to have effect.

Note: It is very important to follow the exact syntax specifications for each configuration file, otherwise the module might have problems interpreting it, which can result in a faulty or non-expected behaviour.

Configuration files

'ip_accs.cfg'

It is possible to configure which IP addresses and what protocols that are allowed to connect to the module. This information is stored in the file '\ip_accs.cfg'.

The file should contain one or several of the headers below.

```
[Web]
[FTP]
[Telnet]
[Modbus/TCP]
[Ethernet/IP]
[All]
```

Under each header the allowed IP addresses should be listed. The wildcard '*' can be used to allow series of IP addresses. If a protocol header is not given, the system will use the configuration listed under the header 'All'. If the 'All' header is not given, the protocol will not accept any connections.

Example:

```
[Web]
10.10.12.*
10.10.13.*
[FTP]
10.10.12.*
[Telnet]
10.10.12.*
[All]
*.*.*.*
```

The above example will allow any IP address beginning with 10.10.12 to access all protocols in the module. IP addresses beginning with 10.10.13 will be able to access the web server, but not the FTP and Telnet servers. The Modbus/TCP and Ethernet/IP servers will accept connections from any IP address.

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\config\ip_access_rights.cfg
```

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

‘http.cfg’

This file holds web server configuration data. For more information about the contents of this file, see 9-1 “Configuration”.

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\config\http_ocnfiguration.cfg
```

Password files

ad_pswd.cfg & sys_pswd.cfg

User/password information for FTP and Telnet is stored in the files ‘sys_pswd.cfg’ (normal mode users) and ‘ad_pswd.cfg’ (administration mode users). These files should be placed in ‘\user\pswd’ and ‘\pswd\ respectively. These directories are protected from web browser access.

The file format is the following:

```
User1:password1
User2:password2
...
User3:password3
```

Example:

```
Bilbo:Hobbit
```

In this example, the username is ‘Bilbo’, and the password is ‘Hobbit’.

If no ‘:’ is present, the password will be equal to the username.

Example:

```
Username
```

In this example, both username and password will be ‘Username’.

‘web_accs.cfg’

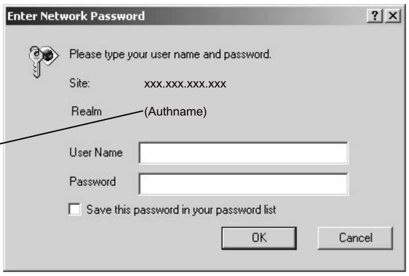
To protect a directory from web access, a file called ‘web_accs.cfg’ must be placed in the directory to protect. This file shall contain a list of users that are allowed to browse the protected directory and its subdirectories. Multiple of these password files may be present in the system, allowing different users to access different files and directories.

The file format is the same as for the ‘ad_pswd.cfg’ and ‘sys_pswd.cfg’ files, except that the optional parameter ‘AuthName’ can be added. The value of this parameter will be presented in the login window. If it is not given, the requested file/pathname will be presented instead.

Example:

```
User:Password

[AuthName]
(Message goes here)
```



Enter Network Password

Please type your user name and password.

Site: xxx.xxx.xxx.xxx

Realm (Authname)

User Name

Password

□ Save this password in your password list

The contents of this file can be redirected by placing the line ‘[File path]’ on the first row, followed by a list of password files.

Example:

```
[File path]
\user\pswd\my_passwords\web_pswd.cfg
\user\pswd\my_passwords\more_pswd.cfg

[AuthName]
(Message goes here)
```

In this example, the accepted user/passwords will be loaded from the files
‘\user\pswd\my_passwords\web_pswd.cfg’ and
‘\user\pswd\my_passwords\more_pswd.cfg’
If any errors in the format of these files is detected the user/password protection will be ignored.

Other

‘telwel.cfg’

The default Telnet welcome message can be changed by putting this file in the root directory. The file should contain the desired welcome message in ASCII format.

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]
\my_settings\telnet_welcome_message.txt
```

‘ssi_str.cfg’

With this file it is possible for the user to reconfigure the SSI output strings. For more information about the content of this file, see 8-11 “Changing SSI Output”.

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\config\ssi_strings.txt
```

Email files (‘email_1.cfg’ – ‘email_10.cfg’)

With these files it is possible to configure predefined email messages that shall be sent on predefined events. It is possible to have 10 admin-defined emails located in the directory “\email\” and 10 user defined emails located in the directory “\user\email\”.

For more information about the format of these files, see 10-1 “Sending a predefined email on data event”.

Network Configuration

Introduction

Before the module can be used on the network, some basic network settings must be configured.

IP address

The IP address is used to identify each node on the TCP/IP network. Therefore, each node on the network must have a unique IP address. IP addresses are written as four decimal integers (0-255) separated by periods, where each integer represents the binary value of one byte in the IP address. This is called dotted-decimal notation.

Example:

```
Address 10000000 00001010 00000010 00011110 is written as 128.10.2.30
```

Subnet Mask

The IP address is divided into three parts - *net ID, subnet ID* and *host ID*. To separate the *net ID* and the *subnet ID* from the *host ID*, a *subnet mask* is used.

The subnet mask is a 32-bit binary pattern, where a set bit allocates a bit for network/subnet ID, and a cleared bit allocates a bit for the host ID. Like the IP address, the subnet mask is commonly written in dotted-decimal notation.

Example:

To make the IP address 128.10.2.30 belong to subnet 128.10.2, the subnet mask shall be set to 255.255.255.0.

```
Subnet Mask:   11111111 11111111 1111111 00000000     (255.255.255.0)
```

Special case IP addresses

The following IP addresses are reserved and should not be used:

0.x.x.x	- IP address where the first byte is zero
127.x.x.x	- IP address where the first byte is 127
x.x.x.0	- IP address where the last byte is zero
x.x.x.255	- IP address where the last byte is 255

Configuring the IP settings

The module offers several ways to set the IP settings (IP address, Subnet mask & Gateway address):

- Stored parameter settings
- DHCP
- HICP
- Switches on the SSC interface (Fieldbus Specific Input register)[1]
- ARP

Note: Some of these configuration methods may be overridden by others. See Appendix A-1 "IP Configuration".

Stored Parameter Settings & DHCP

The module will use the stored IP configuration parameter settings if the NA bit in parameter #8 ("Configuration Bits") is set and/or the switch is set to zero.

The following information is stored in parameters:

- IP address
- Subnet mask
- Gateway address
- SMTP address
- DHCP state (Enabled / Disabled)

If DHCP is enabled or if the switch is set to 0, the module will attempt to retrieve the following information via DHCP:

- IP address
- Subnet mask
- Gateway address
- SMTP address

The module supports DHCP Reboot, i.e. it will ask the DHCP server for the IP address stored in parameter #103 ("IP Address Cfg"). If that address is free to use, it will be assigned to the module. If not, the module will be assigned a new IP address.

HMS IP Configuration Protocol (HICP)

HICP is an acronym for 'HMS IP Configuration Protocol', and will be used by a future Windows-based application that will be able to detect HMS modules on the network and configure their IP settings. Since the protocol is based on broadcast messages, it will be possible to detect and configure modules that are outside of the host's subnet. Please note that the required software is not yet available, and that this feature should be disabled if this functionality is not desired.

[1] Note that these settings cannot be used on the Internet. This is because the IP address series used in this mode belongs to the private address set, see RFC 1918.

Switches on the SSC Interface (Fieldbus Specific Input register)[1]

The configuration switch provides an easy way to configure the module for intranet use. The switch represents the binary value of the last byte in the IP address.

The module will use the switch setting if the NA bit in parameter #8 ("Configuration Bits") is cleared and the switches are set to a value other than zero.

The module will then use the following settings:

```
IP address:       192.168.0.n
Subnet mask:      255.255.255.0
Gateway address:  0.0.0.0 (No gateway set)
```

The last byte of the IP address ('n') represents the binary value of the switches. Subnet mask and Gateway address settings are fixed to the above values when using the configuration switches.

(For more information about how the switches are decoded, consult the general AnyBus-IC Design Guide)

Address Resolution Protocol (ARP)

The IP address can be changed during runtime using the ARP command from a PC. The new IP address will be stored in the IP configuration parameters.

The module will then use the following settings:

```
IP address:    Address provided using ARP
Subnet mask:   255.255.255.0
Gateway:       0.0.0.0 (No gateway)
DHCP:          OFF
```

Below is an example on how to change the IP address from a MS DOS™ window:

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

The 'ARP -s' command will store the IP address and MAC address in the PC's ARP table. When the 'PING' command is executed, the PC will send this message to the module using the specified MAC address. When the module receives this message and detects that it was address with the correct MAC address but not the current IP address, it will adopt the new IP address.

(The 'ARP -d' command is optional, but it removes the static route from the PC ARP table.)

This method can be used to reconfigure modules that already has been configured, or even to reconfigure modules outside the host's subnet.

Note: As the Arp command automatically configures the subnet mask to 255.255.255.0, the first three bytes of the IP address must be the same as for the PC executing the command.

Example:

```
PC:            10.10.12.67
Module:        10.10.12.x (Where x is a value between 1 and 254)
```

FTP Server

It is possible to upload/download files to/from the file system using a standard FTP client. Depending on security settings, different parts of the filesystem can be accessed by the user:

- **Normal users**

  The root directory will be '\user' unless the user has Admin access rights, see below.

- **Admin users**

  The user will have unrestricted access to the file system, i.e. the root directory will be '\'.

- **Global Admin Mode**

  Any username/password combination will be accepted. All users has unrestricted access to the file system, i.e. the root directory will be '\'.

The FTP server can be enabled/disabled with the "FTP src enable" parameter (#122).

For more information about the security framework in the module, see 4-2 "Security Framework".

Telnet Server

Through a Telnet client, the user can access the filesystem using a command line interface similar to MS-DOS™. Depending on security settings, different parts of the filesystem can be accessed by the user:

- **Normal users**

  The root directory will be '\user' unless the user has Admin access rights, see below.

- **Admin users**

  The user will have unrestricted access to the file system, i.e. the root directory will be '\'.

- **Global Admin Mode**

  No login is required in this mode. All users have unrestricted access to the file system, i.e. the root directory will be '\'.

It is possible to configure which IP addresses are allowed to connect to the telnet server, see 4-4 "'ip_accs.cfg'". The telnet server can be enabled/disabled using parameter #123 ("Telnet Enable").

Welcome Message

It is possible to change the default Telnet welcome message by adding the file "\telwel.cfg" containing the Telnet welcome message to use. The file should contain the desired welcome message in ASCII format.

For more information see 4-7 "'telwel.cfg'".

General commands

admin

Syntax:

```
admin
```

Provided that the user can supply a valid admin username/password combination, this command enables admin access in normal mode. Note that this command has no effect in administration mode.

help

Syntax:

```
help [[general][diagnostic][filesystem]]
```

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

```
General commands:

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

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

version

Syntax:

```
version
```

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

exit

Syntax:

```
exit
```

This command closes the Telnet session.

Diagnostic commands

arps

Syntax:

```
arps
```

Display ARP stats and table

iface

Syntax:

```
iface
```

Display net interface stats

sockets

Syntax:

```
sockets
```

Display socket list

routes

Syntax:

```
routes
```

Display IP route table

File System Operations

For commands where filenames, directory names or paths shall be given as an argument the names can be written directly or within quotes. (Filenames that include spaces must be surrounded by quotes)

It is also possible to use relative pathnames using '.', '\' and '..'

dir

Syntax:

```
dir [path]
```

Lists the contents of a directory. If no path is given, the contents of the current directory is listed

md

Syntax:

```
md [[path][directory name]]
```

Creates a directory. If no path is given, the new directory is created in the current directory.

rd

Syntax:

```
rd [[path][directory name]]
```

Removes a directory. The directory can only be removed if it is empty.

cd

Syntax:

```
cd [path]
```

Changes current directory.

format

Syntax:

```
format
```

Formats the filesystem. This is a privileged command i.le. it can only be called in administration mode.

del

Syntax:

```
del [[path][filename]]
```

Deletes a file.

copy

Syntax:

```
copy [[source path][source file]] [[destination path][destination file]]
```

This command creates a copy of the source file at a specified location.

ren

Syntax:

```
ren [[path][old name]] [[path][new name]]
```

Renames a file or directory.

move

Syntax:

```
move [[source path][source file]] [[destination path]]
```

This command moves a file or directory from the source location to a specified destination.

type

Syntax:

```
type [[path][filename]]
```

Display the contents of a file.

mkfile

Syntax:

```
mkfile [[path][filename]]
```

Creates an empty file.

append

Syntax:

```
append [[path][filename]] [The line to append]
```

Appends a line to a file.

df

Syntax:

```
df
```

This command displays information about the filesystem.

SSI (Server Side Include) Script Functionality

It is possible to provide web pages and email messages with dynamic content. This makes it possible to access I/O data and other information in a user friendly manner; Configuration settings and I/O data can be altered using standard html forms, reports can be sent via email messages etc. Due to natural reasons, SSI commands that are used for data input cannot be used in email messages.

To accomplish this, the module uses a simple script system called SSI (Server Side Includes). These are commands that can be embedded into html code and email messages to access functions and data within the module.

Syntax:

```
<?--#exec cmd_argument='SSI COMMAND'-->
```

The example html code below uses the SSI command 'DisplayIP' to include the IP address of the module on a webpage.

Example:

```
<html>
<head><title>SSI Example - IP Address</title></head>
<body>
<center><?--#exec cmd_argument='DisplayIP'--></center>
</body>
</html>
```

Note: It is very important to follow the exact syntax specification for each command, otherwise the module might have problems interpreting it, which can result in a faulty or non-expected behaviour.

Command Set Summary

The following SSI functions are implemented:

Command	Description
DisplayIP	Display the IP address of the module
DisplaySubnet	Display the module's subnet mask
DisplayGateway	Display the currently used Gateway address.
DisplayEmailServer	Display the currently used SMTP server address
DisplayDHCPState	Display the currently used DHCP state
StoreEtnConfig[a]	Store a passed IP configuration in the module
printf	Include a formatted string. The string may contain data from the I/O area or a parameter
scanf[a]	Read a string passed from an object in an html form, interpret the string according a specified format, and store the result in the Output data area or in a parameter.
IncludedFile	Include the contents of a file
SaveToFile[a]	Saves the content of a passed form to a file
GetText[a]	Reads a string from an object in a HTML form and stores in the OUT area.
GetConfigItem	Gets a value from a tag in a file
SetConfig[a]	Writes a configuration from a form to a configuration file
SsiOutput	Redefines the SSI success/failure output

[a] This command is used for data input and cannot be used in email messages.

SSI Commands

Ethernet Address Display Functions

DisplayIP

This SSI function returns the IP address of the module as a string.

Syntax:

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

DisplaySubnet

This SSI function returns the configured subnet mask as a string.

Syntax:

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

DisplayGateway

This SSI function returns the configured gateway address as a string.

Syntax:

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

DisplayDhcpState

This SSI function indicates whether the DHCP functionality is enabled or disabled.

Syntax:

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

DisplayEmailServer

This SSI function returns the currently used SMTP server address as a string.

Syntax:

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

Store Function

StoreEtnConfig

This function stores a configuration from an HTML form in the corresponding AnyBus-IC parameters.

Syntax:

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

Accepted fields in form:

Field Name	Value
SetIp	IP address, e.g. “xx.xx.xx.xx”
SetSubnet	Subnet address, e.g. “xx.xx.xx.xx”
SetGateway	Gateway address, e.g. “xx.xx.xx.xx”
SetSmtpServer	SMTP server address, e.g. “xx.xx.xx.xx”
SetDhcpState	“ON” or “OFF”

Default Output Strings

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

For information about how to change the SSI output strings, see 8-11 “Changing SSI Output”.

Formatted Display

printf

This SSI function returns a formatted string which may contain data from the module I/O area or parameter data. The formatting of the string is similar to the standard C function 'printf()'.

Syntax:

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

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

- Flags (in any order), which modify the specification:

  -, which specifies left adjustment of the converted argument in its field.

  +, which specifies that the number will always be printed with a sign.

  space: if the first character, specifies padding to the field with leading zeros.

  0: for numeric conversion, specifies padding to the field with leading zeros

  #, 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 is 0 if the zero padding flag is present.
- A period, which separates the field width from the precision.
- A number, the precision, that specifies the maximum number of characters to be printed from a string, or the number of digits to be printed after the decimal point for e, E, or f conversions, or the number of significant digits for g or G conversion, or the minimum number of digits to be printed for an integer (leading 0s will be added to make up the necessary width)
- A length modifier h,l (letter ell), or L. "h" indicates that the corresponding argument is to be printed as a short or unsigned short; "l" (ell) indicates that the argument is a long 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; signed decimal notation
o	byte, short; unsigned octal notation (without a leading zero)
x, X	byte, short; unsigned hexadecimal notation (without a leading 0x or 0X), using abcdef for 0x or ABCDEF for 0X
u	byte, short; unsigned 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. The default precision
f	long; 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	long; decimal notation of the form [-]m.dddddd e±xx or [-]m.ddddddE±xx where the number of d's is specified by the precision. The default precision is 6; a precision of 0 suppresses the decimal point.
g, G	long; %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 zeroes 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	Area
InReadSByte(offset)	Reads a signed byte from position offset in the IN area	IN
InReadUByte(offset)	Reads an unsigned byte from position offset in the IN area	
InReadSWord(offset)	Reads a signed word from position offset in the IN area	
InReadUWord(offset)	Reads an unsigned word from position offset in the IN area	
InReadSLong(offset)	Reads a signed longword from position offset in the IN area	
InReadULong(offset)	Reads an unsigned longword from position offset in the IN area	
InReadString(offset)	Reads a string (char*) from position offset in the IN area	
InReadFloat(offset)	Reads float value from position offset in the IN area	
OutReadSByte(offset)	Reads a signed byte from position offset in the OUT area	OUT
OutReadUByte(offset)	Reads an unsigned byte from position offset in the OUT area	
OutReadSWord(offset)	Reads a signed word from position offset in the OUT area	
OutReadUWord(offset)	Reads an unsigned word from position offset in the OUT area	
OutReadSLong(offset)	Reads a signed longword from position offset in the OUT area	
OutReadULong(offset)	Reads an unsigned longword from position offset in the OUT area	
OutReadString(offset)	Reads a string (*char) from position offset in the OUT area	
OutReadFloat(offset)	Reads a float value from position offset in the OUT area	
ParReadSByte(ParameternNo)	Reads parameter (ParameterNo) and returns it as a signed byte.	Parameter
ParReadUByte(ParameterNo)	Reads parameter (ParameterNo) and returns it as an unsigned byte.	
ParReadSWord(ParameterNo)	Reads parameter (ParameterNo) and returns it as a signed word.	
ParReadUWord(ParameterNo)	Reads parameter (ParameterNo) and returns it as an unsigned word.	
ParReadSLong(ParameterNo)	Reads parameter (ParameterNo) and returns it as a signed long.	
ParReadULong(ParameterNo)	Reads parameter (ParameterNo) and returns it as an unsigned long.	
ParReadString(ParameterNo)	Reads parameter (ParameterNo) and returns it as a string (char*).	
ParReadFloat(ParameterNo)	Reads parameter (ParameterNo) and returns it as a float.	
ParReadFormatted(ParameterNo)	Reads parameter (ParameterNo), formats it and returns it as a string.	

Formatted Input

scanf

This SSI function reads a string passed from an object in a HTML form, interprets the string according to the specification in “format”, and stores the result in the OUT area or the AnyBus-IC parameter area according to the passed arguments. The formatting of the string is similar to the standard C function scanf()

An input field is defined as a string of non-white space characters; it extends either to the next white space character or until the field width, if specified, is exhausted. White space characters are bland, tab, new line, carriage return, and form feed.

Syntax:

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

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

Character	Input data, 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*
%	Liteal %; no assignment is made.

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

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

Argument	Description	Area
OutWriteByte(offset)	Writes a byte to position offset in the OUT area	IN
OutWriteWord(offset)	Writes a word to position offset in the OUT area	
OutWriteLong(offset)	Writes a long to position offset in the OUT area	
OutWriteString(offset)	Writes a string to position offset in the OUT area	
OutWriteFloat(offset)	Writes a float to position offset in the OUT area	
ParWriteByte(Parameters)	Writes a byte to parameter (ParameterNr)	Parameter
ParWriteWord(ParameterNr)	Writes a word to parameter (ParameterNr)	
ParWriteLong(ParameterNr)	Writes a long to parameter (ParameterNr)	
ParWriteString(ParameterNr)	Writes a string to parameter (ParameterNr)	
ParWriteFloat(ParameterNr)	Writes a float to parameter (ParameterNr)	
ParWriteFormatted(ParameterNr)	Writes a formatted value to parameter (ParameterNr)	

Default Output:

```
Write succeeded
Write failed
```

For information about how to change the SSI output, please see 8-11 “Changing SSI Output”.

Unformatted Input

GetText

The SSI function gets the text from an html object and stores it to the OUT area or the AnyBus-IC parameter area. GetText can be used instead of scanf(”%s”, ...) when reading an input string that shall not stop at a white space character.

Syntax:

<?--#exec cmd_argument=’GetText("ObjName", Arg1, n)’-->

Arguments:

ObjName	- The name of the object with the passed data string
Arg	- Specifies where to write the data
n	- Optional. Specifies maximum number of characters to read

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

Argument	Description	Area
OutWriteString(offset)	Writes the input string to position offset in the OUT area.	OUT
ParWriteString(ParameteNr)	Writes the input string to a parameter (ParameterNr) Requires parameter to be a string.	Parameter
ParWriteFormatted(ParameterNr)	Writes value to a parameter (ParameterNr). Parses the input string according to the parameter descriptor, e.g., an integer value, and writes it to the parameter.	

Default Output:

Success	- Write succeeded
Failure	- Write failed

For information about how to change the SSI output, please see 8-11 “Changing SSI Output”.

File Operations

IncludeFile

This SSI function returns the contents of a specified file.

Syntax:

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

Arguments:

File Name	- File name to include

Default Output:

Success	- contents of the file
Failure	- Failed to open "File Name"

For information about how to change the SSI output, please see 8-11 "Changing SSI Output".

SaveToFile

This SSI function saves the contents of a passed form to a file.

The passed name/value pair will be written to the file "File name" separated by the "Separator" string. The content can either be Appended to the file or Overwrite the current content of the file.

All fields in the passed form except fields with name starting with underscore '_' will be stored in the file.

Syntax:

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

Default output:

Success – Form saved to file

Failure – Failed to save form

For information about how to change the SSI output, please see 8-11 "Changing SSI Output".

Miscellaneous

GetConfigItem

This SSI function reads the content under the defined tag in the defined configuration file.

The format of the configuration file shall be as follows:

```
[Tag1]
Value1
[Tag2]
Value2
...
[TagN]
ValueN
```

Syntax:

```
<?--#exec cmd_argument='GetConfigItem( "File name", "Tag", "Line Separator"
)'-->
```

Arguments

File name	- Configuration file to read from.
Tag	- Tag in file to read the value from.
Line separator	- This argument is optional. Specifies how to separate new lines in the value. When not given, new lines in the value will be separated with CRLF.

Default output:

Success	– The content under the tag
Failure	– Could not get value for [Tag]

For information about how to change the SSI output, please see 8-11 “Changing SSI Output”.

SetConfig

This SSI function saves a configuration passed from an HTML form to the defined configuration file. The format of the configuration file shall be as described in 8-9 “GetConfigItem”.

All fields in the passed form except fields with name starting with underscore ‘_’ will be stored in the file. The name will be stored as the Tag and the value will be stored as Value.

If the file is not found, it is created (if the path exists). If the tag name is not found in the file, it is created.

Syntax:

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

Default output:

Success – Configuration stored to “File name”
Failure – Could not store configuration to “File name”

For information about how to change the SSI output, please see 8-11 “Changing SSI Output”.

Example:

A form containing the following fields...

- Name = “Speed”, value = “48”
- Name = “Temp”, value = “20”
- Name = “_B1”, value = “submit” (Button used to submit the form, to avoid storage the name starts with ‘_’)

... will generate the following configuration file:

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

Changing SSI Output

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

- Changing SSI output defaults by creating a file called “\ssi_str.cfg” containing the output strings for all SSI functions in the system.
- 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 correct according to the specification below, the SSI functions will use the output strings specified in this file instead of the default strings.

The file has 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"
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]

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

[GetConfigItem]
Failure: "String to use on failure"[1]

[SetConfig]
Failure: "String to use on success"[1]
Failure: "String to use on failure"[1]
```

The content of this file can be redirected to another file by placing the line ‘[File path]’ on the first row, and a file path on the second.

Example:

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

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

[1] To include the filename, insert ‘ %s’ 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 and thereafter the SSI functions will use the Default outputs or the outputs according to the file “\ssi_str.cfg”. Max size of strings is 128 bytes.

Syntax:

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

Example:

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

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

SSI Examples

Displaying I/O Data on a Web Page

The following is an example of an HTML file that when uploaded to the module displays in hex the second byte of data from the IN area and the third byte of data of the OUT area table using the SSI “printf” command.

Example:

```
<html>
<head><title>AnyBus-IC I/O Data Example</title></head>
<body><center>
<?--#exec cmd_argument='printf("IN 2 = 0x%02X",InReadUByte(2))'-->
<?--#exec cmd_argument='printf("OUT 3 = 0x%02X",OutReadUByte(3))'-->
</center></body>
</html>
```

SSI in email messages

If the contents of byte 3 in the IN area is larger than 40h, the message below is sent to the address listed under [RecptOne] in the configuration file ‘MyAddresses.cfg’. This makes it possible to make admin defined email messages to have a recipient configurable through a web page.

Example:

```
[Register]
IN, 0x0003, BYTE
[Register match]
0x40, 0xFF, >
[To]
<?--#exec cmd_argument='GetConfigItem( "MyAddresses.cfg", "RecptOne" )'-->
[From]
homer@powerplant.com
[Subject]
Nuclear meltdown
[Message]
Doh!
```

Web Server

The module features a web server with SSI capabilities. For more information about the SSI functionality, see 8-1 “SSI (Server Side Include) Script Functionality”.

By default the HTTP server is enabled, but it can be enabled/disabled during runtime, see 13-1 “Web Srv Enable (Parameter #121)”.

Default Web Pages

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

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

The virtual file system contains the following files:

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

The virtual file system can be enabled/disabled using parameter #130 (“VFS Enable”)

Configuration

The file ‘\http.cfg’ holds configuration settings for the web server. The file contains two configurable items:

- **Content Types**

  There are a number of file types that by default will return predefined content types when requested through the web server (see 9-2 “Default Content Types”).

  When a file is requested through the web server it will first search for the file types specified in this file. If it’s not found in this file it will search for it in its predefined content types. This means that adding file type in this file will replace it’s predefined type. File types shall be added under the header [FileTypes], see 9-2 “File format”. A maximum 50 file-types can be defined.

- **SSI File Types**

  By default, the web server is configured to scan all “.shtm” files for SSI:s, but it is possible to add other file types that should be scanned. A maximum 50 SSI file-types can be defined.

  File types shall be added under the header [SSIFileTypes], see 9-2 “File format”. A maximum 50 SSI file types can be defined.

Default Content Types

By default, the following content types are detected by their filename 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
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.wmlscript	*.wmlsc
text/xml	*.xml
application/pdf	*.pdf

If a file extension is not recognized, the content type defaults to binary data "/".

File format

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

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

Example

```
[FileTypes]
tif:image/tiff
tiff:image/tiff
doc:application/msword
avi:video/x-msvideo

[SSIFileTypes]
htm
html
xml
```

Security

All files except files in the directories "\user\pswd\", "\pswd\" and files named 'web_accs.cfg' can be viewed by default. Other directories can be protected by placing a file called 'web_accs.cfg'(see 4-5 "Password files") in the directory to protect. The file contains a list of users that are allowed to browse that directory. Also, it is possible to configure which IP addresses are allowed to connect to the web sever, 4-4 "'ip_accs.cfg'".

Email Client

It is possible to send emails from the module. To send an email, the SMTP server address must be configured. Without a valid SMTP address the module will not be able to send any email messages. (Note - the module can currently only connect to mail servers that do not require authentication).

Send email

The application can send email messages through the module using parameter #129 “Send email”.

Sending a predefined email on data event

It is possible to send predefined email messages to predefined receivers, triggered by an event in the Input/Output data area. The area is scanned once every 0.5 second. This means that an event must be present longer than 0.5 seconds to be detected by the module.

It is possible to have up to 10 user defined, and 10 admin defined emails, triggered on different events. These shall be placed in the directories “\user\email\” for user configurable emails and “\email” for non-user configurable emails. The files must be named ‘email_1.cfg’, ‘email_2.cfg’ ... ‘email_10.cfg’

File format:

```
[Register]
Area, Offset, Type

[Register match]
Match Value, Mask, Match operand

[To]
Recipient

[From]
Sender

[Subject]
Subject line

[Headers]
Extra Headers

[Message]
Message Body
```

Parameter	Description
Area	Source data area. Possible values are ‘IN’ or ‘OUT’
Offset	Source offset data area, shall be written in decimal or hexadecimal
Type	Source data type. Possible values are ‘BYTE’, ‘WORD’ or ‘LONG’
Match Value	Value to compare with the source data. Shall be written in decimal or hexadecimal.
Mask	The module performs a logical AND operation on the source data with this mask before the value is compared with the Match Value. The value shall be written in decimal or hexadecimal.
Match Operand	This parameter specifies how the data shall be compared with the Match Value. Possible values are ‘<’, ‘=’ or ‘>’
Recipient	Destination email address. Multiple recipients can be specified, separated by semicolon.
Sender	Sender email address
Subject line	Email subject (Only one line)
Extra Headers	This parameter is optional. It may be useful for advanced users to, for example, send HTML emails.
Message Body	The actual email message.

The data is read from the I/O area at the location specified by the Area and Offset parameters. The data size to read is specified by the Type parameter. The module performs a logical AND between the read data and the Mask value. The result is compared with the Match Value. The Match Operand specifies how the data shall be compared.

Parameter #127 (“Triggered Emails”) indicates how many event triggered email messages that has been sent successfully. Parameter #128 (“SMTP Errors”) indicates how many email messages that failed to send.

Note: If the [Register] or [Register match] information is changed, a reset is required for changes to have effect. Other changes will have effect immediately.

Example:

```
[Register]
IN, 0x0003, BYTE

[Register match]
0x20, 0x7F, >

[To]
support@hms-networks.com

[From]
AnyBus@hms-networks.com

[Subject]
Status

[Message]
All data correct.
```

- A byte is read from the IN area, at offset 0x0003h
- The module performs a logical <data> AND 7Fh
- If the result is larger than 20h, the email message is sent to support@hms-networks.com

Note: Hexadecimal values shall be written in the format 0xN where N is the hexadecimal value.

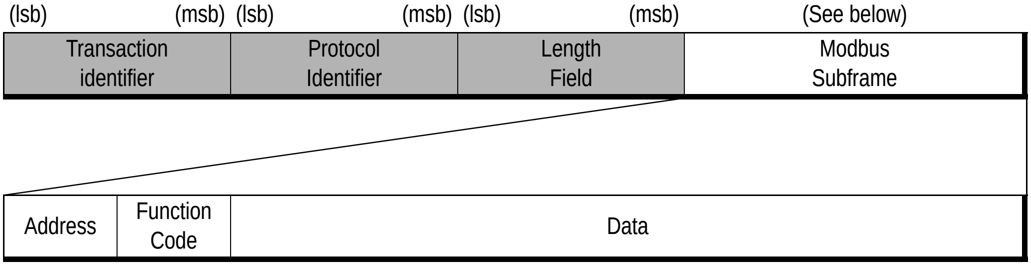
Modbus/TCP

The implementation of the Modbus/TCP server is done according to the Modbus/TCP specification v1.0. All commands according to class 0 and class 1 is implemented, as well as some of the class 2 commands. The module can handle 8 simultaneous connections.

The Modbus/TCP server can be enabled/disabled using parameter #133 (“MB/TCP enable”).

Message frame format

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.



Note that the Modbus/TCP message frame does not include a CRC field as Modbus does, since the TCP/IP frame format already features sophisticated error checking.

For detailed information regarding the Modbus/TCP protocol, consult the Open Modbus Specification v1.0.

Port

All Modbus/TCP messages are received/transmitted on TCP port no. 502.

Supported Functions

Function code	Function name	Class	Affects area	
1	Read coils	1	OUT [Bit area]	(0XXXX registers)
2	Read input discretes	1	IN [Bit area]	(1XXXX registers)
3	Read multiple registers	0	OUT [Register area]	(4XXXX registers)
4	Read input registers	1	IN [Register area]	(3XXXX registers)
5	Write coil	1	OUT [Bit area]	(0XXXX registers)
6	Write single register	1	OUT [Register area]	(4XXXX registers)
15	Force multiple coils	2	OUT [Bit area]	(0XXXX registers)
16	Force multiple registers	0	OUT [Register area]	(4XXXX registers)
22	Mask write register	2	OUT [Register area]	(4XXXX registers)
23	Read/Write registers	2	OUT [Register area]	(4XXXX registers)

Modbus/TCP Addressing

The I/O areas can be configured to a maximum size of 48 bytes each. When accessing these areas using Modbus/TCP, the addressing is done according to the following table:

Modbus/TCP Register Type	Modbus/TCP Area	AnyBus-IC Area
1XXXX	Bit area	In Area
3XXXX	Register area	
0XXXX	Bit area	Out Area
4XXXX	Register area	

The amount of the I/O areas that should be allocated as bit areas is configured using the parameters #134("In bit size") and #135 ("Out bit size").

Supported Exception Codes

Exception code	Name	Description
01h	Illegal function	Module does not support the function code in the query.
02h	Illegal data address	Data address received in the query is outside the initialized memory area.
03h	Illegal data value	The data in the request is illegal.

EtherNet/IP

The module can act as a Group 2 and 3 server on an EtherNet/IP based network. EtherNet/IP is based on the Control and Information protocol (CIP) which is also the application layer for DeviceNet and ControlNet to exchange data between nodes.

CIP makes use of abstract object modelling to describe the communications of a product. Objects are well defined subsets of the functionality of a device. This include functions, called 'Services' and data variables called 'Attributes'. If more than one copy of an object is needed, each copy is called an 'Instance'.

The EtherNet/IP protocol can be enabled/disabled using parameter #160 ("EIP enable cfg").

Implementation Notes

I/O Data

The first four bytes in an I/O data connection contains control bytes sent from the Scanner. The way these bytes should treated can be configured using parameter #162 ("EIP Strip Status")

Objects

Application Parameters can also be accessed from the fieldbus, by mapping them to a Vendor Specific EtherNet/IP Object using the CIP Mapping Object Class, see Appendix C-1 "Parameter Data Mapping".

Implemented Objects

EtherNet/IP requires some mandatory objects; these are implemented as well as some vendor specific objects. The mandatory objects are in the specification from ODVA.

The following vendor specific objects are implemented:

- Identity Object, Class 01h
- Message Router, Class 02h
- Assembly Object, Class 04h
- Connection Manager, Class 06h
- EtherNet/IP Object, Class 64h - C7h (User Objects)
- TCP/IP Interface Object, Class F5h
- Ethernet Link Object, Class F6h

Identity Object, Class 01h

Services

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

Class attributes

#	Access	Name	Type	Default Value	Description
1	Get	Revision	UINT	0x0001	Revision 1

Instance attributes

#	Access	Name	Type	Default Value	Description
1	Get[a]	Vendor ID	UINT	0x005A	HMS Industrial Networks AB
2	Get[a]	Device type	UINT	0x000C	Communication Adapter
3	Get[a]	Product code	UINT	0x0002	ABIC-ETN
4	Get	Revision	Struct of:	-	Revision:
		Major revision	USINT	-	Major version
		Minor revision	USINT	-	Minor version
5	Get	Status	WORD	-	Device Status, see below
6	Get	Serial number	UDINT	ABIC serial number	Serial number
7	Get[a]	Product name	SHORT_STRING	AnyBus-IC EtherNet/IP	Product name

a. Can be changed by parameters in the AnyBus IC if a correct password has been entered. The password is distributed to vendors that want to change the default AnyBus-IC values.

Status Attribute

Bits	Description
0	Owned, shall be set when at least one connection is configured
1	(reserved, set to 0)
2	Configured, is always set to 1
3	(reserved, set to 0)
4-7	See extended device status
8	Is set for minor recoverable faults
9	Is set for minor recoverable faults
10	Is set for major recoverable faults
11	Is set for major unrecoverable faults
12-15	(reserved, set to 0)

Extended Device Status

Value	Description
0010	Faulted I/O connection
0011	No I/O connection established
0100	Non volatile configuration bad
0110	Connection in run mode
0111	Connection in idle mode
0110	Connection in run mode

Reset Service

The Identity object provides a reset service. There are two different types of reset requests:

- **Type 0: 'Power Cycling Reset'**

  This service emulates a power cycling of the module. The module will by default perform a reset of the entire module. However, if the Interrupt Config [RES] bit in parameter #12 ("Interrupt Config") is set, the module will hand over the reset request to the application by issuing an interrupt. Parameter #13 ("Interrupt Cause") will show reset as cause (The RES bit will be set).

- **Type 1: 'Out of box reset'**

  This service performs an "out of box" configuration before reset. The module will by default reset the parameters in the list below to their default values and then perform a reset of the entire module.

  - IP address cfg
  - Subnet mask cfg
  - GW address cfg
  - DHCP enable cfg
  - Data rate cfg
  - Duplex cfg
  - SMTP server cfg

  If the Interrupt Config [DEF] bit in parameter #12 ("Interrupt Config") is set, the module will issue an interrupt. The application is then responsible for resetting any application/AnyBus-IC configuration data. Parameter #13 ("Interrupt Cause") will show default as cause (The DEF bit will be set).

  If the Interrupt Config [RES] bit in parameter #12 ("Interrupt Config") is set, the module will issue an interrupt. The application is then responsible for resetting itself and the module. Parameter #13 ("Interrupt Cause") will show reset as cause (The RES bit will be set).

Message Router, Class 02h

Services

Class services:	-
Instance services:	-

Assembly Object, Class 04h

Services

Class services: Get Attribute Single

Instance services: Get Attribute Single
Set Attribute Single
Get Member

Descriptione

The Assembly object uses static assemblies. The assembly instance IDs used are in the vendor specific range.

Class attributes

#	Access	Name	Type	Value	Description
1	Get	Revision	UINT	0x0002	Revision 2
2	Get	Max Instance	UINT	0x0096	0x96 is the highest instance number

Instance attributes, Instance/Connection Point 64h (IN)

#	Access	Name	Type	Value	Description
3	Get	Data	ARRAY of BYTE	-	Data produced by the ABIC to the master.
4	Get	Size	UINT	From parameter #43 (“FB In Actual”)	Data size in bytes (Maximum 48 Bytes)

Instance attributes, Instance/ Connection Point 96h (OUT)

#	Access	Name	Type	Value	Description
3	Set	Data	ARRAY of BYTE	-	Data consumed by the ABIC from the master.[a]
4	Get	Size	UINT	From parameter #42 (“FB Out Actual”)	Data size in bytes (Maximum 48 Bytes)

[a] Rockwell Automation PLCs have the first four bytes consumed by a device defined as status information. This status information is not defined in the EtherNet/IP specification; it is a Rockwell Automation implementation. Since all known PLCs/masters have this implementation, the module strips off the first four bytes in the consumed data by default. However, this behaviour can be changed using parameter #162 (“EIP Strip Status”), see 13-9 “EIP Strip Status (Parameter #162)”.

The first bit in the first byte tells if the master is in idle mode. In this case the module will take action according to Parameter #11 “Idle Action Config”). The value of the Run/Idle bit is presented in parameter #100 (“FB Status”). This means that the actual connection to the module is always 4 bytes larger than configured.

Connection Manager, Class 06h

Services

Class services:	Forward Open Forward Close
Instance services:	-

Ethernet/IP Object Number 64h - C7h (User objects)

Services

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

Description

HMS Object Specification (HOS) attributes (e.g. AnyBus-IC- or Application Parameters) can be mapped to Ethernet/IP Class number 0x64 - 0xC7, Instance 0x00-0xFF and Attribute 0x00-0xFF.

Up to 32 Attributes can be mapped.

The mapping is done through the CIP Mapping Object (HOS class 0xA5) using the Modbus object messaging protocol. (Consult the general AnyBus-IC Design Guide for more information regarding the Object Messaging implementation)

Class/Instance attributes

#	Access	Name	Type	Value	Description
X	Get/Set	X	X	X	X

X = Set by application

TCP/IP Interface Object, Class F5h

Services

Class services: Get Attribute Single
Get Attribute All

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

Class attributes

#	Access	Name	Type	Value	Description
1	Get	Revision	UINT	0x0001	Revision 1
2	Get	Max Instance	UINT	0x0001	1 is the highest instance number
3	Get	Number of Instances	UINT	0x0001	1 instance is implemented

Instance attributes

#	Access	Name	Type	Value	Description
1	Get	Status	DWORD	0x01	1 = The interface configuration attribute contains valid configuration.
2	Get	Configuration capability	DWORD	0x00000005	Capable of containing network configuration via BOOTP.Capable of containing network configuration via DHCP.
3	Get	Configuration control	DWORD	0x00000000	Configuration from non-volatile memory
4	Get	Physical Link Object	Struct of:	-	Physical link -> Ethernet object
		Path size	UINT	0x0002	2 words
		Path	Padded EPATH	20 F6 24 01	Ethernet Class, Instance 1
5	Get	Interface configuration	Struct of:		
		IP Address	UDINT	-	AnyBus-IC Parameter 105 – IP address Act
		Network Mask	UDINT	-	AnyBus-IC Parameter 107– Subnet mask Act
		Gateway Address	UDINT	-	AnyBus-IC Parameter 109 – GW address Act
		Name Server	UDINT	0x00000000	(not used)
		Name Server 2	UDINT	0x00000000	(not used)
		Domain Name	STRING	0x00	(not used)
6	Get	Host Name	STRING	0x00	(not used)

Ethernet Link Object, Class F6h

Services

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

Class attributes

#	Access	Name	Type	Value	Description
1	Get	Revision	UINT	0x0001	Revision 1
2	Get	Max Instance	UINT	0x0001	1 is the highest instance number
3	Get	Number of Instances	UINT	0x0001	1 instance is implemented

Instance attributes

#	Access	Name	Type	Value	Description
1	Get	Interface Speed	UDINT	10 or 100	The actual speed in megabits/s
2	Get	Interface Flags	DWORD		
3	Get	Physical Address	ARRAY of 6 USINTs	MAC address	The ethernet MAC address of the module

Fieldbus Specific Parameters

To be able to use the full functionality of every fieldbus, the Fieldbus Specific Parameters are used. These parameters are specific to the actual fieldbus used and must be configured accordingly.

General

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

Network Configuration

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

Server Settings

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

Email Client

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

File System

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

Modbus/TCP & EtherNet/IP Related Parameters

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

Application Parameters

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

[a] Parameter Dependant

General

Serial Number (Parameter #141)

This function returns the production serial number of the module.

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

FB Status (Parameter #100)

This function returns information about the current network status.

Parameter Name	'FB Status'
Parameter number	100
Modbus Address	7000h
Default value	-
Range	0000h - FFFFh
Size	2 bytes
Stored in NV RAM	No
Access	RW

Bit layout

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

- **BUS**

  1: Bus is on line

  0: Bus is off line

- **IDLE**

  1: Scanner is in idle / program mode

  0: Scanner is in run mode

MAC address (Parameter #116)

This parameter holds the ethernet MAC address of the module.

Parameter Name	'MAC address'
Parameter number	116
Modbus Address	701Bh - 701Dh

Default value	-
Range	-
Size	6 bytes
Stored in NV RAM	No
Access	R

DIP switch SSC (Parameter #104)

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

Parameter Name	‘DIP switch SSC’
Parameter number	104
Modbus Address	7006h
Default value	-
Range	0h - FFh
Size	1 byte
Stored in NV RAM	No
Access	R

Network Configuration

IP address cfg (Parameter #103)

This parameter holds the manually configured IP address. Note that in order for this value to be valid, the NA bit in parameter #8 (“Configuration Bits”) must be set.

Parameter Name	‘IP address cfg’
Parameter number	103
Modbus Address	7004h - 7005h
Default value	0.0.0.0
Range	0.0.0.0 - 255.255.255.255
Size	4 bytes
Stored in NV RAM	No
Access	RW

IP address act (Parameter #105)

This parameter holds the actual IP address.

Parameter Name	‘IP address act’
Parameter number	105
Modbus Address	7007h -7008h
Default value	-
Range	0.0.0.0 - 255.255.255.255
Size	4 bytes
Stored in NV RAM	No
Access	R

Subnet mask cfg (Parameter #106)

With this parameter it is possible to configure the Subnet mask. The module must be restarted in order for changes to have effect.

Parameter Name	‘Subnet mask cfgt’
Parameter number	106
Modbus Address	7009h - 700Ah
Default value	-
Range	0.0.0.0 - 255.255.255.255
Size	4 bytes
Stored in NV RAM	No
Access	RW

Subnet mask act (Parameter #107)

This parameter holds the currently used Subnet mask.

Parameter Name	‘Subnet mask act’
Parameter number	107
Modbus Address	700Bh - 700Ch
Default value	-
Range	0.0.0.0 - 255.255.255.255
Size	4 bytes
Stored in NV RAM	No
Access	R

GW address cfg (Parameter #108)

With this parameter it is possible to configure the Gateway address. The module must be restarted in order for changes to have effect.

Parameter Name	‘GW address cfg’
Parameter number	108
Modbus Address	700Dh - 700Eh
Default value	-
Range	0.0.0.0 - 255.255.255.255
Size	4 bytes
Stored in NV RAM	No
Access	RW

GW address act (Parameter #109)

This parameter holds the currently used Gateway address.

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

Parameter Name	‘GW address act’
Parameter number	109
Modbus Address	700Fh - 7010h
Default value	-
Range	0.0.0.0 - 255.255.255.255
Size	4 bytes
Stored in NV RAM	No
Access	R

DHCP enable cfg (Parameter #114)

With this parameter it is possible to configure the DHCP state (ON or OFF).

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

Parameter Name	'GW address act'
Parameter number	114
Modbus Address	7019h
Default value	1h
Range	0h (Disable) - 01h (Enable)
Size	1 bytes
Stored in NV RAM	No
Access	RW

DHCP enable act (Parameter #115)

This parameter holds the current DHCP state.

Parameter Name	'DHCP enable act'
Parameter number	115
Modbus Address	701Ah
Default value	-
Range	0h (Disabled) - 01h (Enabled)
Size	1 byte
Stored in NV RAM	No
Access	R

Data rate cfg (Parameter #117)

With this parameter it is possible to configure the Ethernet communication speed.

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

Parameter Name	'Data rate cfg'
Parameter number	117
Modbus Address	701Eh
Default value	0h
Range	0h - 2h (0= auto, 1 = 10mbps, 2=100mbps)
Size	1 bytes
Stored in NV RAM	No
Access	RW

Data rate act (Parameter #118)

This parameter holds the currently used Ethernet speed.

Parameter Name	'Data rate cftt'
Parameter number	118
Modbus Address	701Fh
Default value	-
Range	0h - 2h (1 = 10mbps, 2=100mbps)
Size	1 bytes
Stored in NV RAM	No
Access	R

Duplex Cfg (Parameter #119)

With this parameter it is possible to set the Ethernet communication duplex mode. The module must be restarted for changes to have effect

Parameter Name	'Duplex Cfgt'
Parameter number	119
Modbus Address	7020h
Default value	0h (auto)
Range	1h - 2h (0 = auto, 1 = Half Duplex, 2 = Full Duplex)
Size	1 byte
Stored in NV RAM	No
Access	RW

Duplex Act (Parameter #120)

Shows the currently used Ethernet communication duplex mode.

Parameter Name	'Duplex Act'
Parameter number	120
Modbus Address	7021h
Default value	-
Range	1h - 2h (1 = Half Duplex, 2 = Full Duplex)
Size	1 byte
Stored in NV RAM	No
Access	R

HICP Enable (Parameter #136)[1]

HICP protocol enable / disable parameter. (See footnote)

Parameter Name	'HICP Enable'
Parameter number	136
Modbus Address	7039h
Default value	1 (Enabled)
Range	0 - 1 (0 = Disabled, 1 = Enabled)
Size	1 byte
Stored in NV RAM	No
Access	RW

HICP Password (Parameter #137)[1]

This password is used for HICP configuration. This password is sent by the HICP client in order to be able to change the TCP/IP settings. (See footnote)

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

[1] HICP is an acronym for 'HMS IP Configuration Protocol', and will be used by a future Windows-based application that will be able to detect HMS modules on the network and configure their IP settings. Since the protocol is based on broadcast messages, it will be possible to detect and configure modules that are outside of the host's subnet. Please note that the required software is not yet available, and that this feature should be disabled if this functionality is not desired.

Server Settings

Web Srv Enable (Parameter #121)

This parameter enables / disables the web server.

Parameter Name	‘Web Srv Enable’
Parameter number	121
Modbus Address	7022h
Default value	1 (Enabled)
Range	0h - 1h (0h = Disabled, 1h = Enabled)
Size	1 byte
Stored in NV RAM	No
Access	RW

FTP Srv Enable (Parameter #122)

This parameter enables / disables the ftp server.

Parameter Name	‘FTP Srv Enable’
Parameter number	122
Modbus Address	7023h
Default value	1 (Enabled)
Range	0h - 1h (0h = Disabled, 1h = Enabled)
Size	1 byte
Stored in NV RAM	No
Access	RW

Telnet Srv Enable (Parameter #123)

This parameter enables / disables the FTP server.

Parameter Name	‘Telnet Srv Enable’
Parameter number	123
Modbus Address	7024h
Default value	1 (Enabled)
Range	0h - 1h (0h = Disabled, 1h = Enabled)
Size	1 byte
Stored in NV RAM	No
Access	RW

Email Client

SMTP Srv Address (Parameter #126)

With this parameter it is possible to configure the SMTP server address. This parameter must be configured in order to be able to send email messages from the module.

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

Triggered Emails (Parameter #127)

This value indicates how many email trigger events that has been detected.

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

SMTP Errors (Parameter #128)

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

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

Send Email (Parameter #129)

This parameter makes it possible to send the predefined emails manually. The first byte in the word determines if it is a user email or admin email that should be sent, and the LSB sets the email number.

I.e 0004h sends admin email no. 4 and 010Ah sends user email number 10.

Parameter Name	‘Send Email’
Parameter number	129
Modbus Address	702Bh
Default value	0000h
Range	0000h - 010Ah
Size	2 bytes
Stored in NV RAM	No
Access	W

File System

VFS Enable (Parameter #130)

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

Parameter Name	'VFS Enable'
Parameter number	130
Modbus Address	702Ch
Default value	1 (Enabled)
Range	0 - 1 (0 = Disabled, 1 = Enabled)
Size	2 bytes
Stored in NV RAM	No
Access	RW

RAM-Disc Path (Parameter #131)

This parameter sets the name of the directory where the RAM disc shall be mounted. The directory must be empty or non-existing. A value of no name ("") disables the ram disc

Parameter Name	'RAM Disc Path'
Parameter number	131
Modbus Address	703Fh - 7034h
Default value	"\RAM"
Range	String, up to 16 characters long
Size	Up to 16 bytes
Stored in NV RAM	No
Access	RW

Admin Mode Cfg (Parameter #124)

This parameter enables / disables Global Admin Mode. The module must be restarted for changes to take effect. For more information, see 4-2 "Security Framework".

Parameter Name	'Admin Mode Cfg'
Parameter number	124
Modbus Address	7025h
Default value	0 (Normal Mode)
Range	0h - 1h (0h = Normal Mode, 1h = Global Admin Mode)
Size	1 byte
Stored in NV RAM	No
Access	RW

Admin Mode Act (Parameter #125)

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

Parameter Name	‘Admin Mode Act’
Parameter number	125
Modbus Address	7026h
Default value	-
Range	0h - 1h (0h = Normal Mode, 1h = Global Admin Mode)
Size	1 byte
Stored in NV RAM	No
Access	RW

Modbus/TCP

MB/TCP Conn TO (Parameter #132)

This parameter is used to set the Modbus/TCP connection timeout in seconds. If no Modbus/TCP message has been received within the specified time period, the Modbus/TCP connection will be closed.

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

MB/TCP Enable (Parameter #133)

This parameter is used to enable /disable the Modbus / TCP server.

Parameter Name	'MB/TCP Enable'
Parameter number	133
Modbus Address	7036h
Default value	1 (Enabled)
Range	0 - 1 (0 = Disabled, 1 = Enabled)
Size	1 byte
Stored in NV RAM	No
Access	RW

In bit size (Parameter #134)

This parameter sets the number of bytes in the Input data area that should be used for Modbus/TCP bit addressing. The rest of the area uses register addressing.

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

Out bit size (Parameter #135)

This parameter sets the number of bytes in Output data area that should be used for Modbus/TCP bit addressing. The rest of the area uses register addressing.

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

On / Off line trg (Parameter #138)

This parameter sets trigger source for module on / off line events.

Parameter Name	'On / Off line trg'
Parameter number	138
Modbus Address	7049h
Default value	1 (Link)
Range	1 - 3 0 - None 1 - Link 2 - Modbus/TCP 3 - EtherNet/IP
Size	1 bytes
Stored in NV RAM	No
Access	RW

On / Off line time (Parameter #139)

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

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

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

On / Off line Cmds (Parameter #140)

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

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

- **Value**

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

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

FCxx = Modbus/TCP function code xx

EtherNet/IP

EIP Enb Cfg (Parameter #160)

With this parameter it is possible to enable / disable the EtherNet/IP server. The module must be re-started for changes to have effect.

Parameter Name	'EIP Enb Cfg'
Parameter number	160
Modbus Address	7100h
Default value	1 (Enabled)
Range	0 - 1 (0 = Disabled, 1 = Enabled)
Size	1 byte
Stored in NV RAM	No
Access	RW

EIP Enb Act (Parameter #161)

This parameter indicates whether the EtherNet/IP server is enabled or disabled.

Parameter Name	'EIP Enb Act'
Parameter number	161
Modbus Address	7101h
Default value	-
Range	0 - 1 (0 = Disabled, 1 = Enabled)
Size	1 byte
Stored in NV RAM	No
Access	R

EIP Strip Status (Parameter #162)

This parameter determines if the first four bytes of a consumed EIP connection shall be considered as status bytes or not. If enabled, the FB status parameter shows the master's run / idle mode.

For more information see 12-4 "Assembly Object, Class 04h".

Parameter Name	'EIP Strip Status'
Parameter number	162
Modbus Address	7102h
Default value	1 (ON)
Range	0 - 1 (0 = OFF, 1 = ON)
Size	1 byte
Stored in NV RAM	No
Access	RW

FB Password (Parameter #102)

This parameter is used to unlock the following parameters:

- #152 ‘EtherNet/IP Vendor ID’
- #153 ‘EtherNet/IP Device Type’
- #154 ‘EtherNet/IP Product Code’
- #155 ‘EtherNet/IP Revision’
- #156 ‘EtherNet/IP Product Name’

(The password can be obtained by contacting HMS)

Parameter Name	‘FB Password’
Parameter number	102
Modbus Address	7003
Default value	-
Range	0000h - FFFFh
Size	2 bytes
Stored in NV RAM	No
Access	W

EIP Vendor ID (Parameter #163)

This parameter holds the EtherNet/IP Vendor ID. Before the application can gain access (including a ‘set default’) to this parameter, the module has to be unlocked using parameter #102 (“FB Password”).

Parameter Name	‘EIP Vendor ID’
Parameter number	163
Modbus Address	7103h
Default value	005Ah (HMS Industrial Networks AB)
Range	0000h - FFFFh
Size	2 byte
Stored in NV RAM	No
Access	R(W)*

EIP Device Type (Parameter #164)

This parameter holds the EtherNet/IP Device Type. Before the application can gain access (including a ‘set default’) to this parameter, the module has to be unlocked using parameter #102 (“FB Password”).

Parameter Name	‘EIP Device Type’
Parameter number	164
Modbus Address	7104h
Default value	000Ch (Communication Adapter)
Range	0000h - FFFFh
Size	2 byte
Stored in NV RAM	No
Access	R(W)*

EIP Product Code (Parameter #165)

This parameter holds the EtherNet/IP Product Code. Before the application can gain access (including a 'set default') to this parameter, the module has to be unlocked using parameter #102 ("FB Password").

Parameter Name	'EIP Product Code'
Parameter number	165
Modbus Address	7105h
Default value	0002h ("AnyBus-IC")
Range	0000h - FFFFh
Size	2 byte
Stored in NV RAM	No
Access	R(W)*

EIP Revision (Parameter #166)

This parameter holds the EtherNet/IP Product Code. Before the application can gain access (including a 'set default') to this parameter, the module has to be unlocked using parameter #102 ("FB Password").

By default, this value is set to the ABIC-ETN product revision, e.g version 1.22 = 0116h

Parameter Name	'EIP Product Code'
Parameter number	166
Modbus Address	7106h
Default value	AnyBus-IC Product Revision
Range	0000h - FFFFh
Size	2 byte
Stored in NV RAM	No
Access	R(W)*

EIP Product Name (Parameter #167)

This parameter holds the EtherNet/IP Product Name. Before the application can gain access (including a 'set default') to this parameter, the module has to be unlocked using parameter #102 ("FB Password").

Parameter Name	'EIP Product Name'
Parameter number	167
Modbus Address	7107h - 7116h
Default value	'AnyBus-IC EtherNet/IP'
Range	String, max. 32 characters
Size	32 byte
Stored in NV RAM	No
Access	R(W)*

Application Parameters (Parameters #200 - #299)

An Application Parameter is a user specific AnyBus-IC parameter created by the application during startup (See Appendix B-1 “Creating an Application Parameter”). Just like other parameters, Application Parameters can be accessed by the application via the SCI interface using Modbus messages, or by the user via the MIF interface.

For each Application Parameter it is possible to specify a number of properties, such as datatype, range, name etc. This enables the application to utilize the MIF user interface for internal functions.

Application Parameters can also be accessed from the fieldbus, by mapping them to a Vendor Specific EtherNet/IP Object using the CIP Mapping Object Class, see Appendix C-1 “Parameter Data Mapping”.

Fieldbus Specific HOS Classes

The HMS Object Software (HOS) provides access to parameters and AnyBus-IC functions in an object oriented way using the Modbus Object Messaging protocol.

The following classes are implemented in the module:

Class code	Class name	Documentation
0x80	Device Object	-
0x81	I/O Data Object	-
0x82	Parameter Object	-
0x83	Router Object	-
0x85	Application Parameter Object	-
0x86	File System Object	-
0xA0	Fieldbus Object	See 14-2 “Fieldbus Object (0xA0)”
0xA1	SCI Object Class	-
0xA2	SSC Interface Object	-
0xA3	Monitor Interface Object	-
0xA4	Modbus RTU Object	-
0xA5	CIP Mapping Object	-
0xA6	Socket Object Class	See 14-4 “Socket Object Class (0xA6)”
0xC0	AnyBus-IC Object	-

Fieldbus Object (0xA0)

This object contains the Ethernet specific configuration parameters.

Class Attributes

#	Type	Access	Name	Parameter	Description
1	Byte	R	Class revision	N/A	Revision number of the Class
2	Word	R	Number of Instances	N/A	Number of instances in the Class
3	Word	R	FB status	100	Current bus status
4	Byte[6]	R	MAC address	116	Module MAC address
5	Byte[4]	RW	IP address cfg	103	Configured IP address
6	-	-	(reserved)	-	-
7	Byte[4]	R	IP address act	105	Currently used IP address
8	-	-	(reserved)	-	-
9	Word	RW	Byte order	40	The byte order in data area
10	Word	R	In data size	43	Size of in data
11	Word	RW	Out data size cfg	41	Configured out data size
12	Word	R	Out data size act	42	Actual used out data size
13	Word	RW	In IO Object instance	N/A	Instance number for in data in the IO object
14	Word	RW	Out IO Object instance	N/A	Instance number for out data in the IO object
15	Word	W	FB password	102	Protected parameter password
16	Byte	RW	Data rate cfg	117	Ethernet communication speed
17	Byte	R	Data rate act	118	Current Ethernet communication speed
18	Byte	RW	Duplex cfg	119	Ethernet communication duplex mode
19	Byte	R	Duplex act	120	Current Ethernet communication duplex mode
20	Byte[4]	RW	Subnet mask cfg	106	Configured Subnet mask
21	Byte[4]	R	Subnet mask act	107	Currently used Subnet mask
22	Byte[4]	RW	GW address cfg	108	Configured Gateway address
23	Byte[4]	R	GW address act	109	Currently used Gateway address
24	Byte	RW	DHCP enable cfg	114	Configured DHCP enable status
25	Byte	RW	DHCP enable act	115	Currently used DHCP enable status
26	Byte	RW	WEB srv enable	121	Web Server enable state
27	Byte	RW	FTP srv enable	122	FTP Server enable state
28	Byte	RW	Telnet enable	123	Telnet Server enable state
29	Byte	RW	Admin mode cfg	124	Configured Global admin mode state
30	Byte	R	Admin mode act	125	Actual Global admin mode state
31	Byte[4]	RW	SMTP srv address	126	Email server address state
32	Word	R	Trigged emails	127	Number of triggered emails
33	Word	R	SMTP errors	128	Number of SMTP errors
34	Word	W	Send email	129	Send a predefined email
35	Byte	RW	VFS enable	130	VFS Enable state
36	Byte[16]	RW	RAM disc path	131	Were to mount the RAM disc
37	Word	RW	MB/TCP conn TO	132	Modbus/TCP connection timeout
38	Byte	RW	MB/TCP enable	133	Modbus/TCP enable state
39	Word	RW	In bit size	134	Size of input area that is modbus bit area
40	Word	RW	Out bit size	135	Size of output area that is modbus bit area
41	Byte	RW	HICP enable	136	HICP protocol enabled
42	Byte[30]	RW	HICP password	137	HICP Password
43	Byte	RW	On/Off Line trg	138	On/Off line trigger source
44	Byte	RW	On/Off Line time	139	On/Off line trigger time

#	Type	Access	Name	Parameter	Description
45	DWord	RW	On/Off line cmds	140	On/Off line trigger commands
60	Byte	RW	EIP enable cfg	160	Configured EtherNet/IP enable status
61	Byte	R	EIP enable act	161	Actual EtherNet/IP enable status
62	Word	R(W)[a]	EIP vendor ID	163	Ethernet/IP vendor identification
63	Byte	RW	EIP strip status	162	Strip status information of a connection
64	Word	R(W))*1	EIP device type	164	Ethernet/IP device type
65	Word	R(W)*1	EIP product code	165	EtherNet/IP product code
66	Word	R(W)*1	EIP revision	166	EtherNet/IP revision number
67	Word	R(W)*1	EIP product name	167	EtherNet/IP product name

[a] These parameters can only be written if a correct password is written to parameter #102 (“FB Password”)

Socket Object Class (0xA6)

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

Class Attributes

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

Instance Attributes

The Socket object does not have any instance attributes.

Common Services

Service Code	Need in Implementation		Service Name	Description
	Class	Instance		
0x01	Required	N/A	Get Attribute	Requests an attribute

Object Specific Services

Service Code	Need in Implementation		Service Name	Description
	Class	Instance		
0x80	Required	N/A	Create Socket	Creates a socket instance
0x82	Required	N/A	Close Socket	Closes a socket (and connection)
0x84	N/A	Required	Bind	Binds a socket to a port
0x86	N/A	Required	Listen	Sets a socket to listening state
0x88	N/A	Required	Accept	Accepts connections on a socket in listening state. Creates a new socket-instance for each accepted connection.
0x8A	N/A	Required	Connect	Establishes a connection
0x8C	N/A	Required	Receive	Reads data from a connected socket
0x8E	N/A	Required	Receive From	Reads data from an unconnected socket
0x90	N/A	Required	Send	Write data to a connected socket
0x92	N/A	Required	Send To	Write data to an unconnected socket

Socket Error

All socket response services will return a word called “Socket Error”. In case of no error this will be zero, but if the socket function returned an error the error will internally be read from the socket and be returned in the “Socket Error” byte.

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

Create Socket (0x80) Service (Class Service)

Creates a socket instance.

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

Create Socket Response (0x81)

#	Type	Name	Description
1	Word	Socket Error	Socket error (See 14-5 “Socket Error”)
2	Word	Instance Number	The number of the created socket instance.

Close Socket (0x82) Service (Class Service)

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

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

Close Socket Response (0x83)

#	Type	Name	Description
1	Word	Socket Error	Socket error (See 14-5 “Socket Error”)

Bind (0x84) Service (Instance Service)

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

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

Bind Response (0x85)

#	Type	Name	Description
1	Word	Socket Error	Socket error (See 14-5 “Socket Error”)

Listen (0x86) Service (Instance Service)

Sets a socket to listening state.

<table>
<tr><th>#</th><th>Type</th><th>Name</th><th colspan="2">Description</th></tr>
<tr><td rowspan="8">1</td><td rowspan="8">Byte</td><td rowspan="8">Backlog</td><td colspan="2">Backlog for incoming connections:</td></tr>
<tr><th>Backlog</th><th>Queue length</th></tr>
<tr><td>0</td><td>1</td></tr>
<tr><td>1</td><td>2</td></tr>
<tr><td>2</td><td>4</td></tr>
<tr><td>3</td><td>5</td></tr>
<tr><td>4</td><td>7</td></tr>
<tr><td>5</td><td>6</td></tr>
</table>

Listen Response (0x87)

#	Type	Name	Description
1	Byte	Socket Error	Socket error (See 14-5 “Socket Error”)

Accept (0x88) Service (Instance Service)

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

#	Type	Name	Description
-	-	-	-

Accept Response (0x89)

#	Type	Name	Description
1	Word	Socket Error	Socket error (See 14-5 “Socket Error”)
2	Word	Instance Number	The number of the new instance for the connected socket.
3	DWord	IP address	Host IP address
4	Word	Port	Host port number

Connect (0x8A) Service (Instance Service)

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

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

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

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

Connect Response (0x8B)

#	Type	Name	Description
1	Word	Socket Error	Socket error (See 14-5 “Socket Error”)

Receive (0x8C) Service (Instance Service)

This service receives data from a connected socket.

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

Receive Response (0x8D)

#	Type	Name	Description
1	Word	Socket Error	Socket error (See 14-5 “Socket Error”)
2	Word	Length	Bytes received
3	Byte[x]	Data	Received data

Receive From (0x8E) (Instance Service)

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

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

Receive From Response (0x8F)

#	Type	Name	Description
1	Word	Socket Error	Socket error (See 14-5 “Socket Error”)
2	Dword	IP address	Host IP address
3	Word	Port	Host port number
4	Word	Length	Bytes received
5	Byte[x]	Data	Received data

Send (0x90) (Instance Service)

This service sends data on a connected socket.

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

Send Response (0x91)

#	Type	Name	Description
1	Word	Socket Error	Socket error (See 14-5 “Socket Error”)
2	Word	Length	Number of sent bytes.

Send To (0x92) (Instance Service)

This service sends data on a connected socket.

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

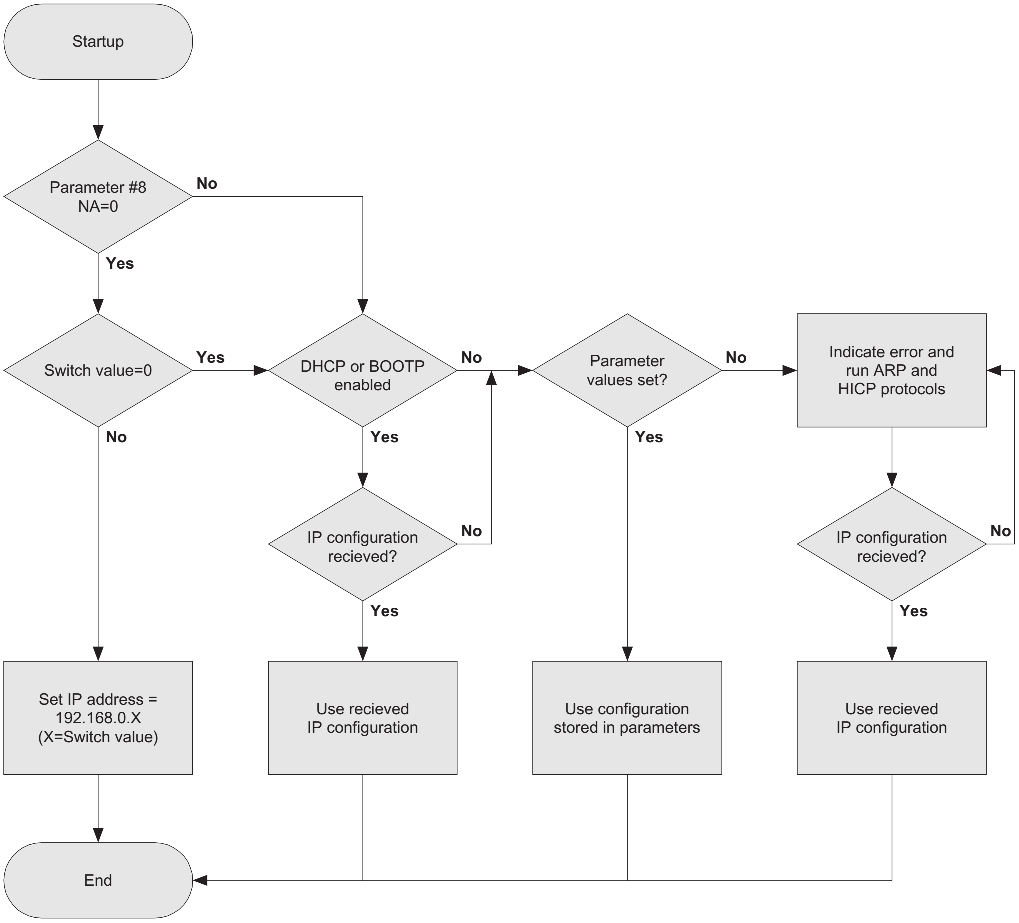
Send To Response (0x93)

#	Type	Name	Description
1	Word	Socket Error	Socket error (See 14-5 “Socket Error”)
2	Word	Length	Number of sent bytes.

Flow Charts

IP Configuration

The flowchart below illustrates the ID configuration process.



Method	Description	Notes
Parameter settings	The IP settings are configured using parameters #103, #106 and #108	A reset is required for changes to have effect.
DIP switch	Module will use 192.168.0.x where X is set with the DIP switch.	This method requires that a DIP switch is mounted on the SSC interface. The switch value is only read once during start-up, i.e. a reset is required for changes to have effect. This configuration can only be used in an intranet and not on the internet, see RFC 1918.
DHCP/BootP	Automatically receive the configuration from a DHCP or BootP server	Requires a DHCP or BootP server on the network.
ARP	Use ARP to configure the IP address. With this method, the IP address can be changed even if it is currently outside the host's subnet.	To use this method, the MAC address of the module must be known. The MAC address can be found on a label on the module.
HICP	Use HICP (HMS IP Configuration Protocol) to configure the IP address.	Modules outside the host's subnet can be configured, since this configureation protocol uses UDP broadcast.

Creating an Application Parameter

Query - “Application Parameter Object”

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

Object Message Sub Field.

Fragment byte count	Fragment protocol	Class ID	Instance ID	Service Code	Attribute	Data Field
(size)	02h	0085h	0000h	0005h	0000h	(See below)

Parameter Size (WORD)	Descriptor (DWORD)	Parameter Info (Size varies)	Extension Word (Optional) (WORD)

Parameter Size

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

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

Descriptor[1]

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
(reserved)	(reserved)	(reserved)	(reserved)	(reserved)	Data Format bit 1	Data Format bit 0	Data Type bit 3	Data Type bit 2	Data Type bit 1	Data Type bit 0	(reserved)	(reserved)	(reserved)	Write	Read

Write Value	Meaning
0	Write access not allowed
1	Write access allowed

Read Value	Meaning
0	Read access not allowed
1	Read access allowed

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

Data Format	Meaning
0 (00)	Dec.
1 (01)	Hex
2 (10)	Bin
3 (11)	Dotted decimal (e.g. IP address etc)

[1] Note that the upper 16 bits of the Descriptor is reserved and should always be set to 0000h.

Parameter Info

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

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

Min Value (size varies)	Max Value (size varies)	Init Value (size varies)	Name[a] (String, 16 bytes)	Unit[a] (String, 16 bytes)

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

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

- **Data type STRING**

Init Value (STRING, 16 bytes)	Name[a] (STRING, 16 bytes)	Unit[a] (STRING, 16 bytes)

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

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

- **Data type BYTE_ARRAY**

Min Value (BYTE)	Max Value (BYTE)	Init Value (BYTE)	Name[a] (String, 16 bytes)	Unit[a] (String, 16 bytes)

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

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

Extension Word (Optional)

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

Value	Description
0000h	-
0001h	Request Modbus Address
Other values	(Reserved for future use)

Response - “Application Parameter Object”

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

Object Message Sub Field

Fragment byte count	Fragment protocol	Class ID	Instance ID	Service Code	Error Code	Data Field
(size)	02h	0085h	0000h	0006h	0000h	(See below)

HOS Instance (WORD)	Parameter Number (WORD)	Modbus Address[1] (WORD)

HOS Instance

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

Parameter Number

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

Modbus Address[1]

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

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

Example

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

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

Query

	01h	5Bh	
	35h	02h	
Class	0085h		Application Parameter Class
Instance	0000h		
Service Code	0005h		Create
Attribute	0000h		
Parameter Size	0002h		Parameter Size = 2 bytes
Descriptor MSB	0000h		
Descriptor LSB	0003h		UINT, DEC, R/W
Min value	0000h		Minimum allowed value: 0
Max value	FFFFh		Maximum allowed value: 65535
Init value	8000h		Initial value: 8000h
Name	53h (‘S’)	70h (‘p’)	“Speed”
	65h (‘e’)	65h (‘e’)	
	64h (‘d’)	00h	
	-	-	
	-	-	
	-	-	
	-	-	
	-	-	
Unit	72h (‘r’)	70h (‘p’)	“rpm”
	6Dh (‘m’)	00h	
	-	-	
	-	-	
	-	-	
	-	-	
	-	-	
	-	-	
Extension Word	0001h		Request Modbus Address
	CRC		

Response

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

Parameter Data Mapping

The mapping procedure consists of two steps:

- **Creating the Application Parameter**
  (See Appendix B-1 “Creating an Application Parameter”)

- **Mapping the created Application Parameter to a Vendor Specific EtherNet/IP Object**
  This is done by creating a new instance in the AnyBus-IC CIP Mapping Object Class (A5h). This class is used to map a vendor specific EtherNet/IP Object Attribute onto an AnyBus-IC Object Attribute.

Query - “CIP Mapping Object”

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

Object Message Sub Field

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

CIP Class (WORD)	CIP Instance (WORD)	CIP Attribute (WORD)	HOS Class (WORD)	HOS Instance (WORD)	HOS Attribute (WORD)	Attribute Size (WORD)

CIP Class

CIP Class to map
(In this case, use a Vendor Specific EtherNet/IP Object. (64h - C7h))

CIP Instance

CIP Instance to map

CIP Attribute

CIP Attribute to map

Attribute Size

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

HOS Class

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

HOS Instance

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

HOS Attribute

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

Response - “CIP Mapping Object”

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

Object Message Sub Field

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

Example

This example will map an Application Parameter to EtherNet/IP Class 144, Instance 1, Attribute 1.

Query

	01h	5Bh	
	17h	02h	
Class	00A5h		CIP Mapping Object
Instance	0000h		
Service Code	0005h		Create
Attribute	0000h		
CIP Class	0090h		CIP Class 144
CIP Instance	0001h		CIP Instance 1
CIP Attribute	0001h		CIP attribute 1
HOS Class	0085h		Application Parameter
HOS Instance	0001h		HOS Instance 1
HOS Attribute	0001h		0001h = Parameter value
Attribute Size	0002h		Size = Word
	CRC		

Response

	01h	5Bh	
	09h	02h	
Class	00A5h		CIP Mapping Object
Instance	0000h		
Service Code	0006h		Create Response
Error Code	0000h		Success
	CRC		

Firmware Upgrade

The module supports the "standard" AnyBus-IC firmware update procedures:

- **Standard Firmware Upgrade**
  (Consult the general AnyBus-IC Design Guide for more information)

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

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

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

Electrical Characteristics

Power Supply

The module requires a regulated +5V DC ±5% power supply.

Power Consumption

The maximum current consumption is 300mA on the 5V power connection.

PE & Shielding

(See 3-1 “Design Considerations”)

Environmental Specification

Temperature

Operating

-10 to +70 degrees Celsius

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

Non Operating

-25 to +85 degrees Celsius

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

Relative Humidity

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

Test performed according to IEC 68-2-30.

EMC compliance

Emission

According to EN 50 081-2:1993

Tested per 55011:1998, Class A, Radiated

Immunity

According to EN 61000-6-2:1999

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