X-Gateway Interface Addendum

EtherNet/IP Scanner

Doc: HMSI-27-249
Rev: 2.00

HMS Connecting Devices™

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

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

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

Table of Contents

Preface **About This Document**

How To Use This Document P-1
Important User Information P-1
Related Documents P-1
Document History P-1
Conventions & Terminology P-3
Sales and Support P-3

Chapter 1 **About the EtherNet/IP Scanner Interface**

General Description 1-1
Features 1-1
External Views 1-1
Interface Status LEDs 1-2

Chapter 2 **File System**

General Information 2-1
Structure 2-2
System Files 2-2

Chapter 3 **Network Configuration**

TCP/IP Settings 3-1
HICP (Anybus IPconfig) 3-2
DHCP 3-2
Speed and Duplex 3-2
IP Access Control 3-3
Gateway Config Interface 3-4

Chapter 4 **Web Interface**

General Information 4-1
Scan List Config 4-2

Chapter 5 **Data Exchange**

General Information 5-1
Control & Status Word Details 5-2
Status Word 5-2
Control Word 5-2
Statistics List/Live List Interpretation 5-3

Chapter 6 CIP Object Implementation

General Information 6-1

Identity Object, Class 01h 6-2
- *General Information* 6-2
- *Class Attributes* 6-2
- *Instance Attributes* 6-2
- *Details: Status Attribute* 6-3

Message Router, Class 02h 6-4
- *General Information* 6-4
- *Class Attributes* 6-4
- *Instance Attributes* 6-4

Assembly Object, Class 04h 6-5
- *General Information* 6-5
- *Class Attributes* 6-6
- *Instances* 6-6
- *Instance Attributes (Instances 101, 768... 831)* 6-6
- *Instance Attributes (Instances 102, 1024... 1087)* 6-6

Connection Manager Object, Class 06h 6-7
- *General Information* 6-7
- *Class Attributes* 6-7
- *Instance Attributes, Instance 01h* 6-7
- *Details: Class 1 Connections* 6-8
- *Details: Class 3 Target Connections* 6-8

Diagnostic Object, Class AAh 6-9
- *General Information* 6-9
- *Class Attributes* 6-9
- *Instance Attributes, Instance 01h* 6-9

Connection Configuration Object, Class F3h 6-10
- *General Information* 6-10
- *Class Attributes* 6-10
- *Instance Attributes* 6-10

Port Object, Class F4h 6-11
- *General Information* 6-12
- *Class Attributes* 6-12
- *Instance Attributes, Instance 02h* 6-12

TCP/IP Interface Object, Class F5h 6-13
- *General Information* 6-13
- *Class Attributes* 6-13
- *Instance Attributes* 6-13

Ethernet Link Object, Class F6h 6-14
- *General Information* 6-14
- *Class Attributes* 6-14
- *Instance Attributes* 6-14

Chapter 7 FTP Server

General Information 7-1

Chapter 8 E-mail Client

General Information 8-1

SMTP Server Settings 8-1

E-mail Definitions 8-2

Appendix A Server Side Include (SSI)

Functions A-2

DisplayMacID A-2
DisplaySerial A-2
DisplayFWVersion A-2
DisplayBLVersion A-2
DisplayIP A-2
DisplaySubnet A-2
DisplayGateway A-3
DisplayDhcpState A-3
StoreIPConfig A-3
GetText A-4
printf A-4
scanf A-8
IncludeFile A-9
SaveToFile A-9
SaveDataToFile A-9
DisplayScannerMode A-10
SetScannerMode A-10

Changing SSI output A-11

SSI Output String File A-11
Temporary SSI Output change A-12

Gateway Control A-12

Refreshing Dynamic Gateway Status Information A-12
Restarting the Gateway A-12

Appendix B Technical Specification

Scanner Interface Details B-1

LAN (Ethernet) Connector Pinout (RJ45) B-1

Important User Information

This document is intended to provide a good understanding of the functionality offered by the Interface described here.

The reader is expected to be familiar with high level software design, and communication systems in general. The use of advanced interface-specific functionality may require in-depth knowledge of networking internals and/or information from the network specifications. In such cases, the persons responsible for the implementation of this product should either obtain the necessary specifications to gain sufficient knowledge, or alternatively limit the implementation in such a way that this is not necessary.

Liability

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

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

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

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

Intellectual Property Rights

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

Trademark Acknowledgements

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

!

WARNING: This is a class A product. in a domestic environment this product may cause radio interference in which case the user may be required to take adequate measures.

ESD Note: This product contains ESD (Electrostatic Discharge) sensitive parts that may be damaged if ESD control procedures are not followed. Static control precautions are required when handling the product. Failure to observe this may cause damage to the product.

Ethernet/IP Scanner X-Gateway Interface Addendum

Doc: HMSI-27-249, Rev: 2.00
May 2014

P. About This Document

P.1 How To Use This Document

This document describes network specific features and procedures needed when operating the EtherNet/IP Scanner Interface for the Anybus X-Gateway. For general information and operating instructions for the Anybus X-Gateway, consult the Anybus-X Generic Gateway User Manual.

The reader of this document is expected to be familiar with local area networks, and communication systems in general.

P.2 Related Documents

Document	Author
Anybus-X Generic Gateway User Manual	HMS
Anybus-S Ethernet 100mbit Fieldbus Appendix	HMS
EtherNet/IP Scanner Interface, Installation Sheet	HMS

P.3 Document History

Revision List

Revision	Date	Author	Chapter	Description
1.00	2007-05-22	PeP	All	1st official release
1.10	2009-09-18	KeL	5	New chapter
1.11	2011-08-22	KaD	4	Updates and minor corrections
1.12	May 2014	SDa	Several	New hardware and Anybus Configuration Manager

P.4 Conventions & Terminology

The following conventions are used throughout this manual:

- Numbered lists provide sequential steps
- Bulleted lists provide information, not procedural steps
- The term 'X-Gateway' refers to the Anybus X-Gateway
- The term 'Scanner interface' refers to the Anybus-X EtherNet/IP Scanner Interface
- The term 'user manual' refers to the Anybus-X Generic Gateway User Manual
- Hexadecimal values are written in the format NNNNh, where NNNN is the hexadecimal value
- 16/32 bit values are generally stored in Motorola (big endian) format unless otherwise stated
- The term "byte" always refers to a string of 8 bits

P.5 Sales and Support

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

1. About the EtherNet/IP Scanner Interface

1.1 General Description

The EtherNet/IP Scanner interface provides EtherNet/IP scanner functionality for the Anybus X-Gateway platform. It exchanges data with up to 64 EtherNet/IP nodes (adapters), and features additional IT functionality such as built-in FTP and web servers, and e-mail client capabilities.

Dynamic content capabilities allow data from the input/output buffers to be monitored on web pages, or included in e-mail messages.

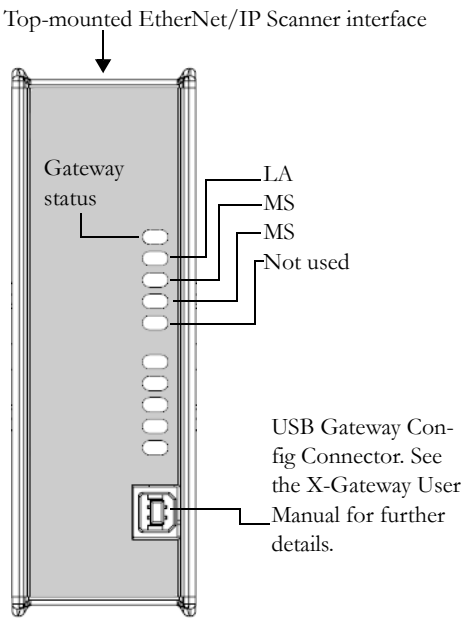
See "Data Exchange" on page 19.

1.2 Features

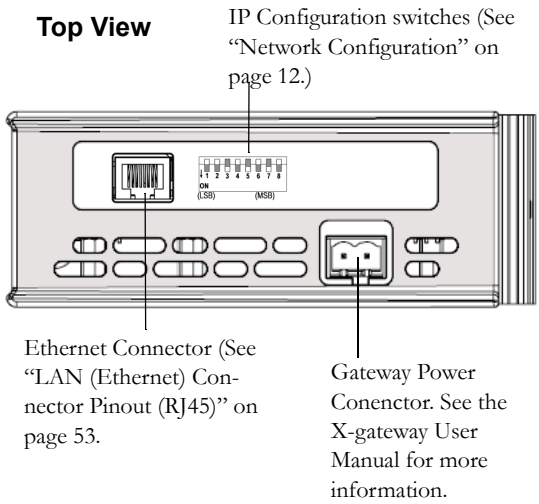
- EtherNet/IP Scanner
- FTP Server
- Web Server
- SMTP Client
- 10/100 Mbit operation, full or half duplex
- On-board IP configuration switches
- Shielded or unshielded cables

1.3 External Views

Front View



Top View



1.3.1 Interface Status LEDs

LED	Colour	Indication
Gateway Status	-	Consult the user manual for further details.
LA	Green	Link established
	Green, flashing	Activity; receiving/transmitting data
	Off	No link or power off
MS	Green	Device operational - Scanner in Run-state
	Green, flashing	Standby - Scanner in Idle-state - Scanner not configured
	Red	Major fault - Major unrecoverable fault
	Red, flashing	Minor fault - Minor recoverable fault (originated or timeout) - Could not open an originated connection
	Alt. Red/Green	Self test - Scanner power-up self test in progress
	Off	No power
NS	Green	Connected - At least one EtherNet/IP connection has been established (target or originated)
	Green, flashing	No connections - No EtherNet/IP connections have been established (Class 1 or Class 3, target or originated)
	Red	Duplicate IP - Configured IP address already in use
	Red, flashing	Connection timeout - One or several EtherNet/IP target connections have timed out - The scanner can only leave this state if all timed-out target connections are re-established, or if the gateway is reset.
	Alt. Red/Green	Self test - Scanner power-up self test in progress
	Off	No power or no IP address

2. File System

2.1 General Information

The scanner interface features a built-in file system that is used to store information such as web files, network communication settings, e-mail messages etc. The file system can be accessed via FTP, the web server, or via the built-in e-mail client (to use SSI).

Conventions & Limitations

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

Storage Areas

The file system features three different storage areas:

- **Area 0 (1151 kByte, Non-volatile)**
  This area is used for static files such as web files etc.
- **Area 1 (128 kByte, Non-volatile)**
  This area is used for configuration files etc.
- **Area 2 (1024 kByte, Volatile)**
  This area is used for temporary storage.

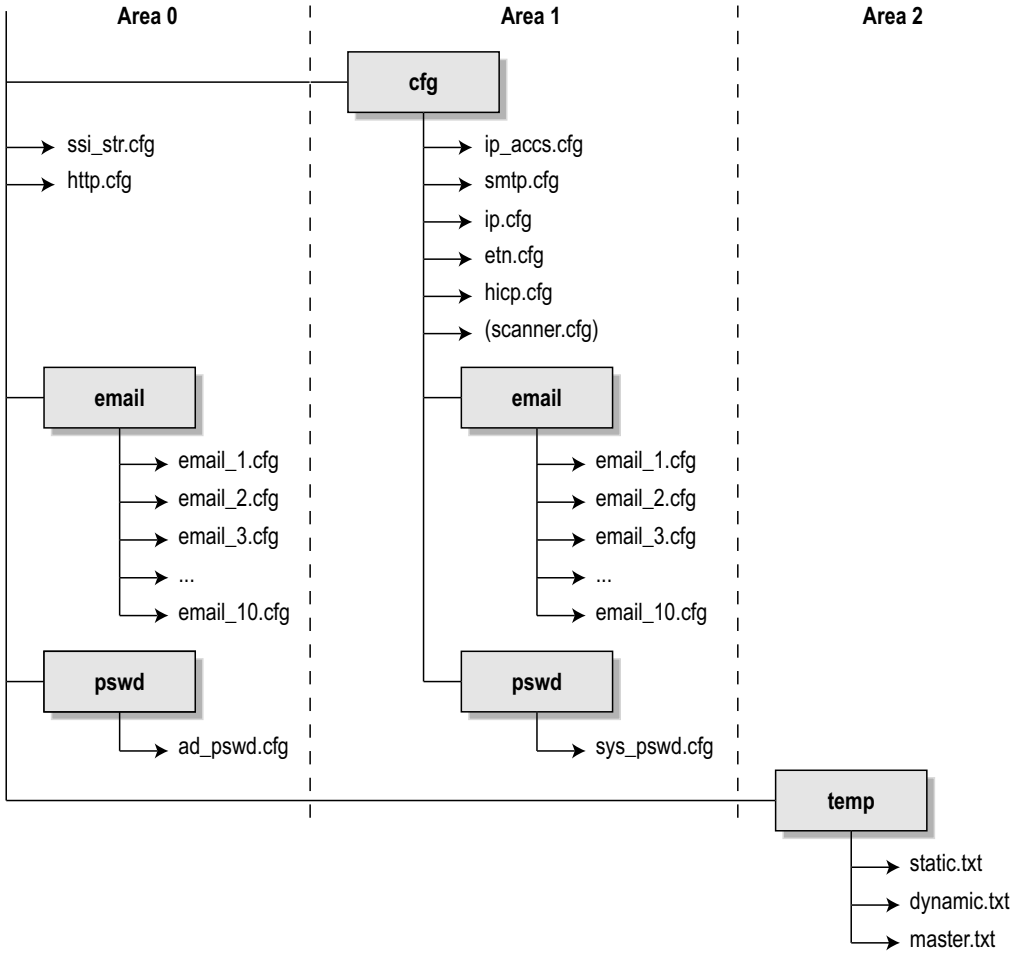
Important:

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

The following operations will erase one or more FLASH segments:

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

2.2 Structure



2.3 System Files

The file system contains a set of files used for system configuration. These files, known as "system files", are regular ASCII files which can be altered using a standard text editor (such as the Notepad in Microsoft Windows™). These files may also be altered indirectly by the built-in web server when using the Server Side Include functionality. Generally, the format of the system files are based on the concept of 'keys', where each 'key' can be assigned a value, see example below.

Example:

```
[Key1]
value of key1

[Key2]
value of key2
```

The exact format specification for each system file is described in detail elsewhere in this document.

Important:

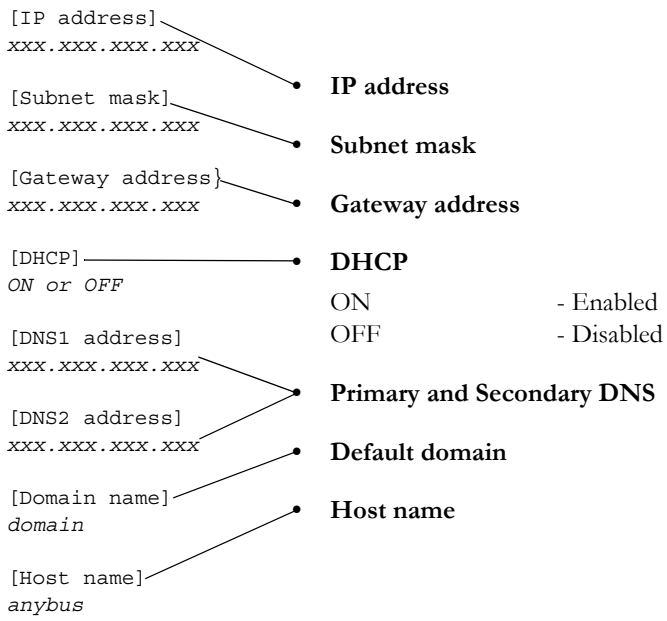
Contrary to that stated above, the file '\cfg\scanner.cfg' holds the scanner configuration in binary format. This file is created automatically by the scanner interface and must not be altered manually.

3. Network Configuration

3.1 TCP/IP Settings

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

File Format:



The network settings may also be affected by...

- EtherNet/IP (See "TCP/IP Interface Object, Class F5h" on page 34.)
- DHCP (See "DHCP" on page 13.)
- HICP (See "HICP (Anybus IPconfig)" on page 13.)
- SSI (See "Server Side Include (SSI)" on page 41.)

If the on-board switches are set to 0 (zero), the scanner interface will use the settings stored in '\cfg\ip.cfg'. If not, the switch setting overrides certain settings in this file, as follows:

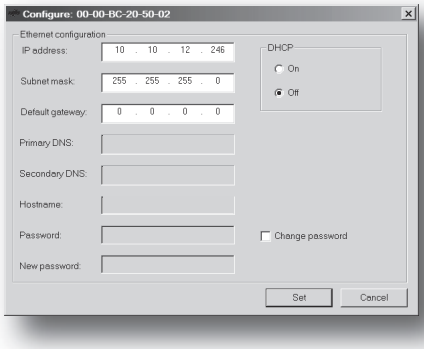
SW 1	SW 2	SW 3	SW 4	SW 5	SW 6	SW 7	SW 8	DHCP	Subnet	Gateway	IP address
OFF	OFF	OFF	OFF	OFF	OFF	OFF	OFF	Determined by '\cfg\ip.cfg'			
OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON	Disabled	255.255.255.0	192.168.0.255	192.168.0.1
OFF	OFF	OFF	OFF	OFF	OFF	ON	OFF	Disabled	255.255.255.0	192.168.0.255	192.168.0.2
...	...	...	...	...	...	...	...	...	...	...	...
ON	ON	ON	ON	ON	ON	OFF	ON	Disabled	255.255.255.0	192.168.0.255	192.168.0.253
ON	ON	ON	ON	ON	ON	ON	OFF	Disabled	255.255.255.0	192.168.0.255	192.168.0.254
ON	ON	ON	ON	ON	ON	ON	ON	(invalid setting)			

Note: These switches are read during startup; any changes require a reset in order to take effect.

3.2 HICP (Anybus IPconfig)

The scanner interface supports the HICP protocol used by the Anybus IPconfig utility, which can be downloaded free of charge from the HMS website. This utility may be used to access the network settings of any Anybus product connected to the network via UDP port 3250.

Upon starting the program, the network is scanned for Anybus products. The network can be rescanned at any time by clicking 'Scan'. In the list of detected devices, the scanner interface will appear as 'ABM-EIP'.



To alter the settings of the interface, double-click on its entry in the list. A window will appear, containing the settings associated with the scanner interface.

Validate the new settings by clicking **Set**. The new IP configuration will be stored in '\cfg\ip.cfg'.

Optionally, the configuration can be protected from unauthorized access by a password. To enter a password, click on the 'Change password' checkbox, and enter the password under 'New password'.

The password is stored in the system file '\cfg\hicp.cfg'.

File Format:

```
[Password]
<password>
```

3.3 DHCP

The scanner interface can retrieve the TCP/IP settings from a DHCP server. If unsuccessful, it will fall back on the current settings (i.e. the settings currently stored in '\cfg\ip.cfg').

If no current settings are available (i.e. set to 0), the scanner interface will halt and indicate an error for the on-board status LEDs (the network configuration can, however, still be accessed via HICP, see See "HICP (Anybus IPconfig)" on page 13..

3.4 Speed and Duplex

The scanner interface supports 10 or 100 Mbit operation in full or half duplex. These settings are stored in the system file 'cfg\etn.cfg'. The settings can also be altered via the Ethernet Link Object. See "Ethernet Link Object, Class F6h" on page 35..

File Format:

```
[AutoNeg]
xxx
```
Autonegotiation; valid settings: 'ON' or 'OFF'

```
[Speed]
xxx
```
Speed; valid settings: '100' or '10'

```
[Duplex]
xxxx
```
Duplex; valid settings: 'FULL' or 'HALF'

3.5 IP Access Control

It is possible to specify which IP addresses permitted to connect to the scanner interface. This information is stored in the system file '\cfg\ip_accs.cfg'.

File Format:

```
[Web]
xxx.xxx.xxx.xxx
```
Nodes listed here may access the web server.

```
[FTP]
xxx.xxx.xxx.xxx
```
Nodes listed here may access the FTP server.

```
[Ethernet/IP]
xxx.xxx.xxx.xxx
```
Nodes listed here may connect to the interface via EtherNet/IP.

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

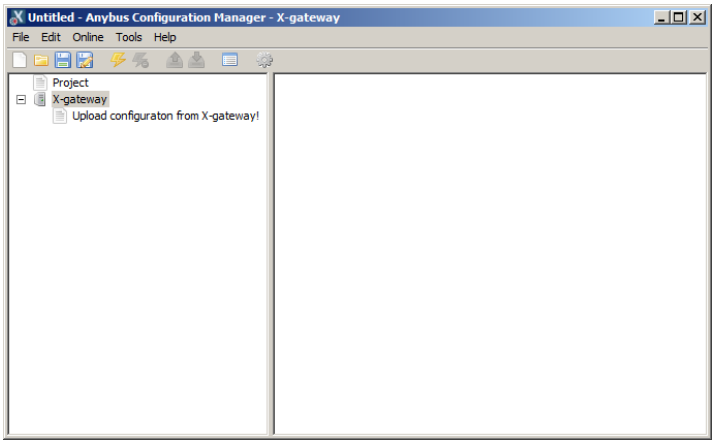
Note: '*' may be used as a wildcard to select IP series.

IMPORTANT:

Under no circumstances should the address '0.0.0.0' be used for [All], as this will in effect prevent all external access to the scanner interface. Failure to observe this will render the product unusable and require service from HMS support.

3.6 Gateway Config Interface

The X-Gateway and certain settings pertaining to the Ethernet Scanner interface may be configured by using the software tool **Anybus Configuration Manager** (ACM), which is available from www.anybus.com/support.



See also...

- The Anybus X-gateway User Manual, for full details on using ACM.
- The online help in ACM, for further help on the available settings.

Note that it is also possible to use the Hyper Terminal application if preferred.

4. Web Interface

4.1 General Information

The Ethernet interface features a fast full featured web server with SSI capabilities. A default web interface provides access to most common options via any standard web browser. The web interface is however customizable and can be fully designed to fit a particular product.

The default web interface differs slightly depending on the network type on the other side of the gateway (i.e. slave, master, fieldbus type etc.), however the basic functions are essentially the same. The scan list config option will be described on the next page.

- **General Status**

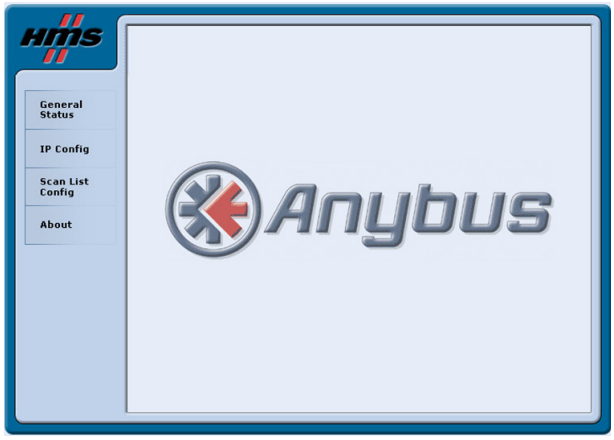
  This page provides an overview of the gateway initialization parameters and general gateway diagnostics (these values correspond to the values set using the gateway config interface).

- **IP Config**

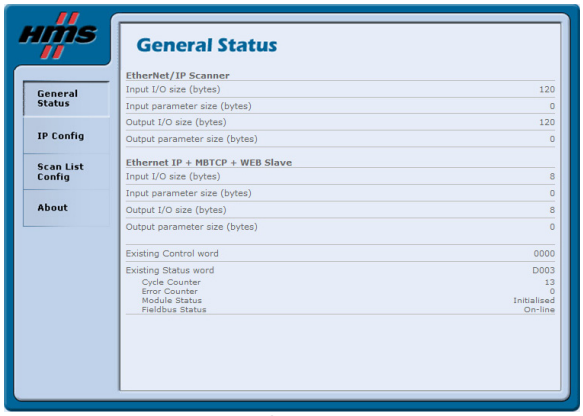
  This page holds the current TCP/IP settings, DNS configuration, and SMTP server settings.

- **About**

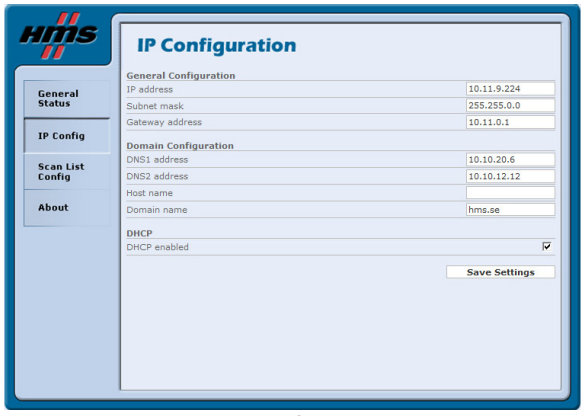
  This page holds the software version numbers and serial numbers of the different components of the gateway. This page also holds the MAC-ID of the Ethernet interface.



Start Page



General Status Page

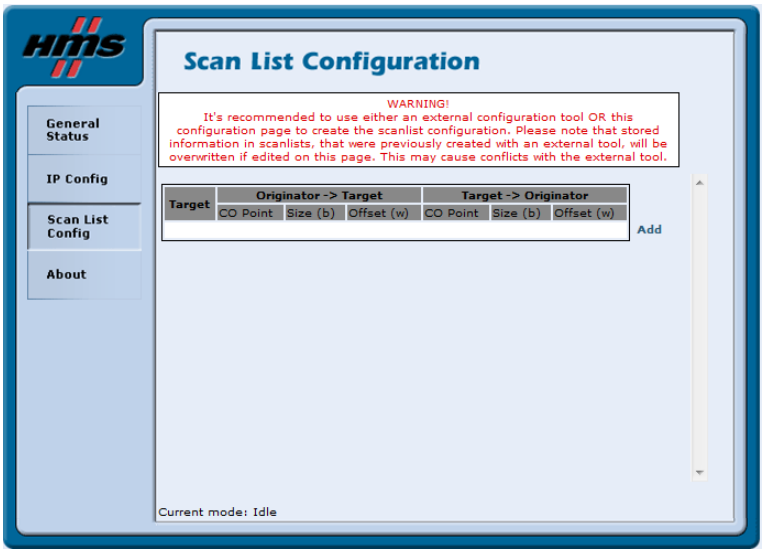


IP Config Page

4.2 Scan List Config

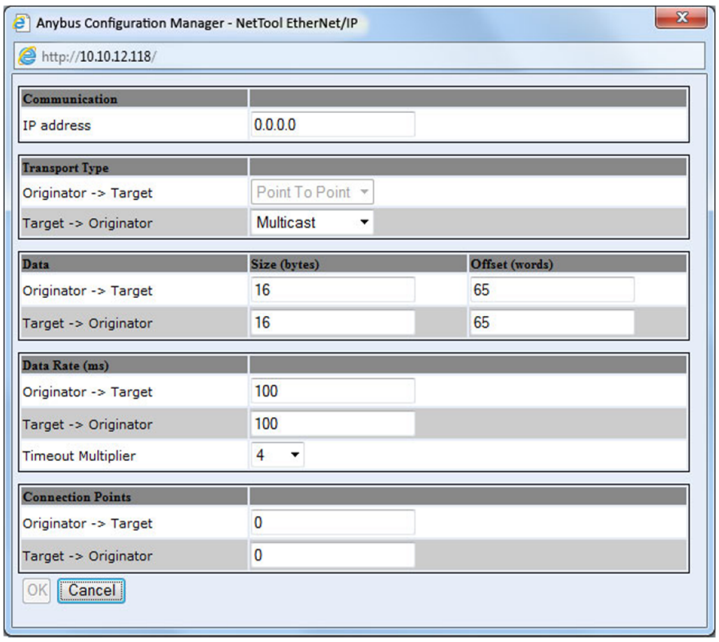
The Ethernet web interface provides the option to configure a scan list for the EtherNet/IP network. In order to save the scan list, the scanner must be in idle mode. To put the scanner in idle mode, change the operation mode in the gateway configuration interface using ACM or the HyperTerminal.

Choose the menu item Scan List Config to see the current empty scan list:



Check that the scanner is in idle mode by checking the current mode in the bottom left corner.

To add a network connection, press the add button (to the right of the list). This will present the window below:



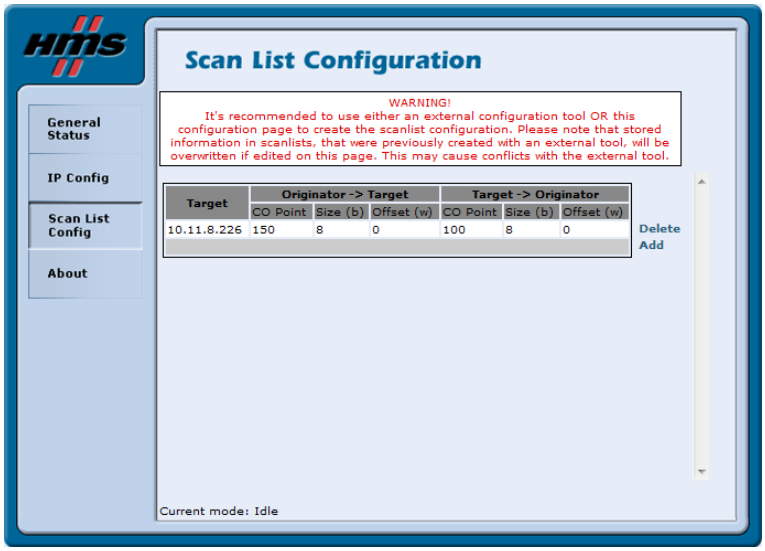
Note: Originator equals the scanner and target equals an adapter.

See the next page for a description of the items in the web interface.

Editable settings:

Item	#	Description
Communication	IP Address	The IP address of the adapter
Transport Type	Originator -> Target	The only option is “Point to Point” (unicast)
	Target -> Originator	Choose from “Point to Point” (unicast) or “Multicast”
Data	Originator -> Target	The size (in bytes) of the data to be sent to the adapter
		The offset from the start of the output area memory of the scanner
	Target -> Originator	The size (in bytes) of the data to be received from the adapter
		The offset from the start of the input area memory of the scanner
Data Rate (ms)	Originator -> Target	Number of milliseconds between each transfer of data to the adapter
	Target -> Originator	Number of milliseconds between each transfer of data to the scanner
	Timeout Multiplier	This number, applied to the data rate (the Requested Package Interval), states the amount of time allowed to pass before the connection is broken
Connection Points	Originator -> Target	Connection Points equals assembly instances, that can be obtained from the manufacturer of the product
	Target -> Originator	Connection Points equals assembly instances, that can be obtained from the manufacturer of the product

When finished configuring an item in the scan list, press the “Ok” button. The newly configured item will show up in the list:



6. CIP Object Implementation

6.1 General Information

EtherNet/IP is based on the Common Industrial Protocol (CIP) which is also the application layer used by DeviceNet and ControlNet to exchange data between nodes.

The following CIP-objects are implemented:

- Identity Object, Class 01h
- Message Router, Class 02h
- Assembly Object, Class 04h
- Connection Manager Object, Class 06h
- Diagnostic Object, Class AAh
- Connection Configuration Object, Class F3h
- Port Object, Class F4h
- TCP/IP Interface Object, Class F5h
- Ethernet Link Object, Class F6h

6.2 Identity Object, Class 01h

6.2.1 General Information

Object Description

-

Supported Services

Class services: Get Attribute All
Get Attribute Single

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

6.2.2 Class Attributes

#	Access	Name	Type	Value
1	Get	Revision	UINT	0001h

6.2.3 Instance Attributes

#	Access	Name	Type	Value
1	Get	Vendor ID	UINT	005Ah (HMS Industrial Networks AB)
2	Get	Device Type	UINT	000Ch (Communications Adapter)
3	Get	Product Code	UINT	0046h (Anybus-X EtherNet/IP Scanner)
4	Get	Revision	Struct of:	
			USINT	Major fieldbus version
			USINT	Minor fieldbus version
5	Get	Status	WORD	See 6-24 “Details: Status Attribute”
6	Get	Serial Number	UDINT	(Serial number)
7	Get	Product Name	SHORT_STRING	“Anybus-X EtherNet/IP Scanner”
103	Set	Scanner Mode	USINT	0: Idle mode 1: Run mode

6.2.4 Details: Status Attribute

bit(s)	Name	Comments
0	Module Owned	-
1	(reserved)	-
2	Configured	-
3	(reserved)	-
4... 7	Extended Device Status	Value: Meaning: 0000b Unknown 0010b Faulted I/O Connection 0011b No I/O connection established 0100b Non-volatile configuration bad 0110b Connection in Run mode 0111b Connection in Idle mode
8	Minor recoverable fault	-
9	Minor recoverable fault	-
10	Major recoverable fault	-
11	Major unrecoverable fault	-
12... 15	(reserved)	-

6.3 Message Router, Class 02h

6.3.1 General Information

Object Description

-

Supported Services

Class services: -

Instance services: -

6.3.2 Class Attributes

-

6.3.3 Instance Attributes

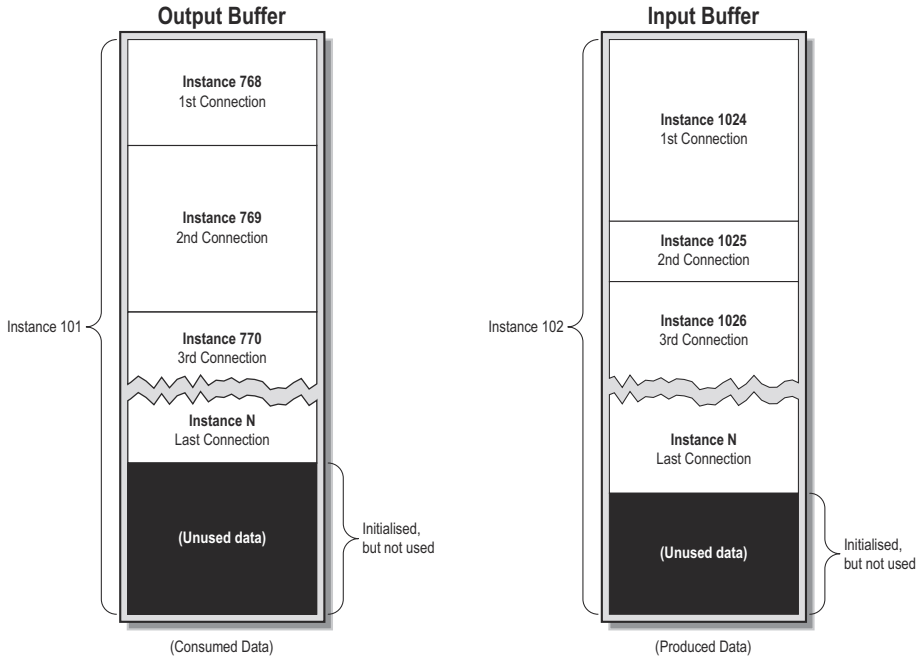
-

6.4 Assembly Object, Class 04h

6.4.1 General Information

Object Description

The Input and Output buffers are represented through instances 101 and 102. In addition, the data associated with each EtherNet/IP connection is represented as dedicated instances (768... 831 and 1024... 1087).



Supported Services

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

6.4.2 Class Attributes

#	Access	Name	Type	Value
1	Get	Revision	UINT	0002h
2	Get	Max Instance	UINT	-

6.4.3 Instances

#	Contents	Description
3	Heartbeat instance	Used as a heartbeat instance for connections only accessing one "real" connection point. No attributes are implemented for this instance.
101	Consumed Data (all connections)	Mapped to the Output buffer. Equals the contents of instances 768...831
102	Produced Data (all connections)	Mapped to the Input buffer. Equals the contents of instances 1024...1087
768	Consumed Data (1st connection)	Consumed data associated with I/O connection no. #1 (Configuration Object instance 1)
769	Consumed Data (2nd connection)	Consumed data associated with I/O connection no. #2 (Configuration Object instance 2)
...	...	...
831	Consumed Data (64th connection)	Consumed data associated with I/O connection no. #64 (Configuration Object instance 64)
1024	Produced Data (1st connection)	Produced data associated with I/O connection no. #1 (Configuration Object, instance 1)
1025	Produced Data (2nd connection)	Produced data associated with I/O connection no. #2 (Configuration Object, instance 2)
...	...	...
1087	Produced Data (64th connection)	Produced data associated with I/O connection no. #64 (Configuration Object, instance 2)

6.4.4 Instance Attributes (Instances 101, 768... 831)

#	Access	Name	Type	Value
3	Get/Set	Consumed data	Array of BYTE	Mapped to the Output buffer.

6.4.5 Instance Attributes (Instances 102, 1024... 1087)

#	Access	Name	Type	Value
3	Get	Produced data	Array of BYTE	Mapped to the Input buffer.

6.5 Connection Manager Object, Class 06h

6.5.1 General Information

Object Description

-

Supported Services

Class services: Get Attributes All
Forward Open
Forward Close
Unconnected Send

Instance services: Get Attributes All

6.5.2 Class Attributes

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

6.5.3 Instance Attributes, Instance 01h

#	Access	Name	Type	Description
1	Get	Open Requests	UINT	No. of received 'Forward Open'-requests
2	Get	Open Format Rejects	UINT	No. of 'Forward Open'-requests that have been rejected due to bad format.
3	Get	Open Resource Rejects	UINT	No. of 'Forward Open'-requests that have been rejected due to lack of resources
4	Get	Open Other Rejects	UINT	No. of 'Forward Open'-requests that have been rejected for reasons other than bad format or lack of resources.
5	Get	Close Requests	UINT	No. of received 'Forward Close'-requests.
6	Get	Close Format Rejects	UINT	No. of 'Forward Close'-requests that have been rejected due to bad format.
7	Get	Close Other Rejects	UINT	No. of 'Forward Close'-requests that have been rejected for reasons other than bad format.
8	Get	Connection Timeouts	UINT	No. of connection timeouts.

6.5.4 Details: Class 1 Connections

Class 1 connections are used to transfer I/O data. Each class 1 connection establishes two data transports; one consuming, and one producing.

No. of Supported Originated Class 1 Connections:	64
Max. Input Connection size:	509 bytes
Min. Output Connection size:	504 bytes
Supported Packet Rate (API):	2...3200ms
Supported Trigger Types:	Cyclic

- **Producing Assembly Instances 100, 101, 102**

No. of Supported Target Class 1 Connections:	20 per instance (sharing same transport)
No. of Supported Transports	1
Supported Transport Types	Point-to-Point, Multicast
Supported Packet Rate (API):	2...3200ms
Supported Trigger Types:	Cyclic

Once a class 1 connection has been established, a transport is received. This transport may be of Point-to-Point or Multicast type. If Point-to-Point, the data is transferred using UDP unicast messages, and no other connections can access the data. If Multicast, the data is transferred with UDP multicast messages, and other connections may use the same transport accessing the data.

Producing instances can only be assigned one transport. Therefore, if using Point-to-Point connections, only 1 (one) Class 1 connection can be established. However, 20 connections can be linked to each Multicast transport, allowing 20 Class 1 connections to be established if they all use the same transport.

In order for a connection to use an existing transport, the connection data size must match the data size of the existing transport, or an error response will be returned. If the connection RPI (Requested Packet Interval) does not match the existing connections's API (Actual Packet Interval), the connection will still be established using the API of the existing transport. This API will be returned in the response to the 'Forward Open'-request.

- **Consuming Assembly Instance 150**

No. of Supported Target Class 1 Connections:	1 per instance
No. of Supported Transports	1
Supported Transport Types	Point-to-Point only
Supported Packet Rate (API):	Unlimited

Since consuming instances are used to control the outputs, only one connection is allowed to each consuming instance.The transport used for the connection must be Point-to-Point.

6.5.5 Details: Class 3 Target Connections

Class 3 connections are used to establish connections to the message router. Thereafter the connection is used for explicit messaging. Class 3 connections use TCP connections.

Up to 16 simultaneous class 3 connections towards the message router is supported.

6.6 Diagnostic Object, Class AAh

6.6.1 General Information

Object Description

This vendor specific object provides diagnostic information from the scanner interface.

Supported Services

Class services: Get Attribute All

Instance services: Get Attribute Single

6.6.2 Class Attributes

#	Access	Name	Type	Comments
1	Get	Revision	UINT	0001h

6.6.3 Instance Attributes, Instance 01h

#	Access	Name	Type	Comments
01h	Get	Serial number	UDINT	Serial number
02h	Get	Vendor ID	UINT	-
03h	Get	Fieldbus Type	UINT	0083h = Ethernet
04h	Get	Software version	UINT	-
0Ah	Get	Interface type	UINT	0201h = Master
0Fh	Get	Input Buffer size	UINT	Size of Input buffer (in bytes)
12h	Get	Output Buffer size	UINT	Size of Output buffer (in bytes)
18h	Get	MAC ID	Array of USINT	Ethernet MAC ID of the interface (6 bytes)
19h	Get	IP Address	UDINT	Current IP address
1Ah	Get	Subnet mask	UDINT	Current subnet mask
1Bh	Get	Gateway address	UDINT	Current gateway address
1Ch	Get	SMTP server address	UDINT	SMTP server address
1Dh	Get	DHCP state	UDINT	0=DHCP enabled, 1=DHCP disabled
1Eh	Get	Bootloader version	UDINT	Interface bootloader version
1Fh	Get	Application interface version	UINT	Application interface software version
20h	Get	Fieldbus software version	UINT	Fieldbus interface software version

6.7 Connection Configuration Object, Class F3h

6.7.1 General Information

Object Description

-

Supported Services

Class services:
Create
Delete
Restore
Get Attribute All
Get Attribute Single
Set Attribute Single
Kick Timer (4Bh)
Change Start (4Fh)
Get Status (50h)
Change Complete (51h)
Audit Changes (52h)

Instance services:
Delete
Restore
Get Attribute All
Set Attribute All
Get Attribute Single

6.7.2 Class Attributes

#	Access	Name	Type	Value
1	Get	Revision	UINT	0002h
2	Get	Max Instance	UDINT	0000 0040h (Maximum instance no. = 64)
3	Get	No. of instances	UDINT	-
8	Get	Format number	UINT	0065h (format number for instance attribute 9)
9	Set	Edit Signature	UDINT	Value written by configuration tool to detect modifications in instance attribute values.

6.7.3 Instance Attributes

One instance is created for each connection. All instance attributes except attribute #1 is stored in non-volatile memory.

#	Access	Name	Type	Comments
1	Get	Connection Status	Struct of:	When a connection is not open, this attribute indicates why
		Gen_status	USINT	
		(reserved)	USINT	
		Ext_status	UINT	
2	Set	Connection Flags	WORD	Connection flags

#	Access	Name	Type	Comments
3	Set	Target Device ID	Struct of:	Device identification used by configuration software to identify target devices associated with this instance.
		Vendor_id	UINT	
		Product_code	UINT	
		Major_rev	USINT	
		Minor_rev	USINT	
5	Set	Net connection parameters	Struct of:	-
		Conn_timeout	USINT	Connection Timeout Multiplier
		Xport_class_and_trigger	BYTE	Transport Class and Trigger
		Rpi_OT	UDINT	Originator to target requested packet interval
		Net_OT	UINT	Originator to target network connection parameters (This attribute specifies the OT connection size)
		Rpi_TO	UDINT	Target to Originator requested packet interval
		Net_TO	UINT	Target to Originator network target connection parameters (This attribute specifies the TO connection size)
6	Set	Connection Path	Struct of:	-
		Open_path_size	USINT	Path size in words (16-bit)
		(reserved)	USINT	Reserved, ignore
		Open connection path	Padded EPATH	Path used in the 'Forward Open'-service of the Connection Manager
7	Set	Config #1 Data	Struct of:	-
		Config_data_size	UINT	Length of Config_data in bytes
		Config_data	Array of Octet	Config #1 data
8	Set	Connection name	Struct of:	-
		Name_size	USINT	Number of characters in the name
		(reserved)	USINT	Reserved, ignore
		Connection_name	STRING2	User assigned connection name encoded in UNICODE
9	Set	Implementation Defined Attribute	Struct of:	-
		Format_number	UINT	0101h
		Impl_defined_data_size	UINT	000Ah
		Impl_defined_data	Array of: UINT, UINT, UINT, UINT, UINT,	 Reserved:(ignore) Offset OT:Offset of data in memory map Reserved:(ignore) Offset TO:Offset of data in memory map Reserved:(ignore)
10	Set	Config #2 Data	Struct of:	-
		Config_data_size	UINT	Length of Config_data in bytes
		Config_data	Array of Octet	Config #2 data
11	Set	Proxy Device ID	Struct of:	-
		Vendor_id	UINT	Vendor ID
		Product_type	UINT	Device Type
		Product_code	UINT	Product Code
		Major_rev	USINT	Major Revision
		Minor_rev	USINT	Minor Revision

6.8 Port Object, Class F4h

6.8.1 General Information

Object Description

-

Supported Services

Class services: Get Attribute All
Get Attribute Single

Instance services: Get Attribute All
Get Attribute Single

6.8.2 Class Attributes

#	Access	Name	Type	Value	Description
1	Get	Revision	UINT	0001h	-
2	Get	Max Instance	UINT	0002h	-
3	Get	No. of instances	UINT	0001h	-
8	Get	Entry Port	UINT	0002h	Instance no. #2 (below)
9	Get	All Ports	Array of STRUCT	0000h 0000h 0000h 0000h 0004h 0002h	Array of structure containing attributes 1 and 2 from each instance. Instance 1 is at byte offset 4. Instance 2 is at byte offset 8, etc. The 4 bytes at offset 0 shall be 0.

6.8.3 Instance Attributes, Instance 02h

#	Access	Name	Type	Value	Description
1	Get	Port Type	UINT	0004h	TCP/IP
2	Get	Port Number	UINT	0002h	Port 2
3	Get	Port Object	Struct of:		-
		Path Size	UINT	0002h	Path Size
		Path	Padded EPATH	20 F5 24 01h	TCP Interface Object (Class F5h,instance 1)
4	Get	Port Name	SHORT_STIRNG	'TCP/IP'	Name of port
8	Get	Node Address	Padded EPATH	-	EPATH describing our TCP/IP address

6.9 TCP/IP Interface Object, Class F5h

6.9.1 General Information

Object Description

This object provides the a mechanism to configure the TCP/IP settings via EtherNet/IP. Note that writing to this object will affect the settings stored in the configuration file 'ip.cfg'.

Supported Services

Class services: Get Attribute All
Get Attribute Single

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

6.9.2 Class Attributes

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

6.9.3 Instance Attributes

#	Access	Name	Type	Value	Description
1	Get	Status	DWORD	0000 0001h	1 = The interface configuration attribute contains a valid configuration
2	Get	Configuration Capability	DWORD	0000 0014h	Interface configuration attribute is settable. Capable of obtaining network configuration via DHCP.
3	Get/Set	Configuration Control	DWORD	-	0 - Configuration from non-volatile memory 2 - Configuration from DHCP
4	Get	Port Object	Struct of:		Physical link -> Ethernet object
		Path Size	UINT	0002h	2 words
		Path	Padded EPATH	20 F6 24 01h	Ethernet Class, Instance 1
5	Get/Set	Interface Configuration	Struct of:		-
		IP Address	UDINT	-	Currently used IP address
		Subnet Mask	UDINT	-	Currently used Subnet mask
		Gateway Address	UDINT	-	Currently used Gateway Address
		Name Server 1	UDINT	-	Primary DNS server
		Name Server 2	UDINT	-	Secondary DNS server
		Domain Name	STRING	-	Default domain name
6	Get/Set	Host Name	STRING	-	Host name

6.10 Ethernet Link Object, Class F6h

6.10.1 General Information

Object Description

This object maintains link specific counters and status information for the Ethernet communications interface.

Supported Services

Class services: Get Attribute All
Get Attribute Single

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

6.10.2 Class Attributes

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

6.10.3 Instance Attributes

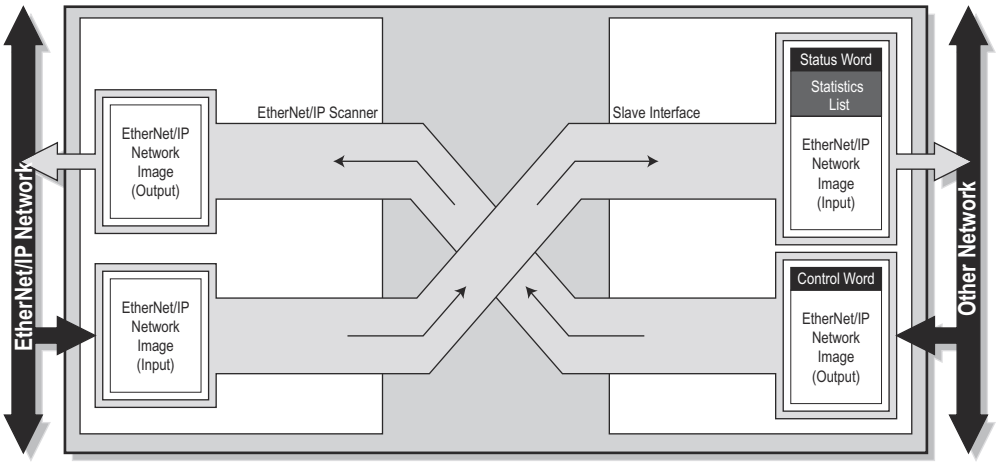
#	Access	Name	Type	Value	Description
1	Get	Interface Speed	UDINT	10 or 100	Actual speed (MBPS)
2	Get	Interface Flags	DWORD	-	Interface flags
3	Get	Physical Address	Array of 6 USINTS	MAC address	Ethernet MAC address
4	Get	Interface Counters	Struct of:		-
		In Octets	UDINT	-	Octets received on the interface
		In Ucast Packets	UDINT	-	Unicast packets received on the interface
		In NUcast Packets	UDINT	-	Non-unicast packets received on the interface
		In Discards	UDINT	-	Inbound packets with unknown protocol
		In Errors	UDINT	-	Inbound packets that contain errors (In Discards not included)
		In Unknown Protos	UDINT	-	Inbound packets with unknown protocol
		Out Octets	UDINT	-	Octets sent on the interface
		Out Ucast Packets	UDINT	-	Non-unicast packets sent on the interface
		Out Discards	UDINT	-	Outbound packets with unknown protocol
		Out Errors	UDINT	-	Outbound packets that contain errors (Out Discards not included)

#	Access	Name	Type	Value	Description
5	Get	Media Counters	Struct of:		-
		Alignment Errors	UDINT	-	Frames received that are not an integral number of octets in length.
		FCS Errors	UDINT	-	Frames received that do not pass the FCS check
		Single Collisions	UDINT	-	Successfully transmitted frames which experienced exactly one collision
		Multiple Collisions	UDINT	-	Successfully transmitted frames which experienced more than one collision
		SQE Test Errors	UDINT	-	Number of times SQRE test error message is generated
		Deferred Transmissions	UDINT	-	Frames for which the first transmission attempt is delayed because the medium is busy
		Late Collisions	UDINT	-	Number of times a collision is detected later than 512 bit-times into the transmission of a packet
		Excessive Collisions	UDINT	-	Frames for which a transmission fails due to excessive collisions
		MAC Transmit Errors	UDINT	-	Frames for which transmission fails due to an internal MAC sublayer receive error
		Carrier Sense Errors	UDINT	-	Times that the carrier sense condition was lost or never asserted when attempted to transmit a frame
		Frame Too Long	UDINT	-	Frames received that exceed the maximum permitted frame size
		MAC Receive Errors	UDINT	-	Frames for which reception on an interface fails due to an internal MAC sublayer receive error
6	Get/Set	Interface Control	Struct of:		-
		Control Bits	UDINT	-	Interface Control Bits
		Force Interface Speed	UDINT	-	Speed at which the interface shall be forced to operate. Returns 'Object state Conflict' if autonegotiation is enabled.

5. Data Exchange

5.1 General Information

The Scanner Interface exchanges data with another network as depicted below.



The structure of the EtherNet/IP Network I/O is determined by the configuration created in the configuration tool.

Note: The input/output data sizes of the Scanner Interface are determined by the size of the actual configuration created using the configuration tool and cannot be set via the Gateway Config interface.

5.2 Control & Status Word Details

5.2.1 Status Word

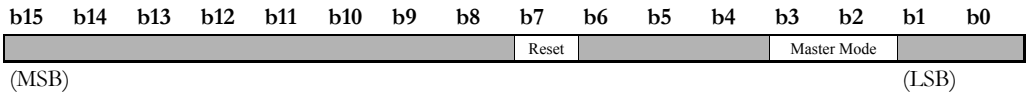
The Status Word holds general status information, as described in the X-gateway user manual. When enabled, it is mapped to the two first bytes of the input area memory buffer.

See the X-gateway user manual for further information.

5.2.2 Control Word

The Control Word, if enabled, controls the communication towards the other nodes on the EtherNet/IP network and is mapped to the two first bytes of the output area memory buffer. The current state can be configured from the fieldbus side in the Control/Status word or it can be configured from the terminal interface.

Control Word Contents:



Mode (b3, b2)	Meaning	Comments
00b	Idle	These settings can be configured from the fieldbus side, from the terminal interface or from the Gateway Config interface.
01b		
10b	Run	
11b		

Reset (b7)	Meaning	Comments
0b	Normal operation	-
1b	Reset gateway	Setting this bit causes the gateway to perform a self-reset.

IMPORTANT: *The Control- and Status Words can be disabled through the Gateway Configuration Interface. In such a case, the master interface will enter 'Run' mode automatically after having completed the start up initialization sequence. If enabled, the scanner will start up in 'Idle' mode, and will not exchange data until the mode has been changed to 'Run'.*

5.3 Statistics List/Live List Interpretation

Connections that are not open cannot be added to the live list for the EtherNet/IP Network Master, as they can not be detected on the network. The status of open connections can be detected, though, and this information will be read and added to the Address Live List. This list holds status information on the number of open connections on the EtherNet/IP network that are either configured, active or faulted. This statistics list can be read from address 640h in the fieldbus specific area.

DPRAM address	**Address Live List (bytes)**	**Description**
640h - 641h	0 - 1	No. of configured connections
642h - 643h	2 - 3	No. of active connections
644h - 645h	4 - 5	No. of faulted connections
646h - 647h	6 - 7	(reserved)

For more information regarding the Live List, consult the X-gateway user manual.

7. FTP Server

7.1 General Information

The built-in FTP server can be used to upload/download files to the file system using a standard FTP client. The server uses the following port numbers:

- **TCP, port 20**
  FTP data port.
- **TCP, port 21**
  FTP command port.

Security Levels

The server features two security levels (below). Security level is set at a per-user basis.

- **Normal Users**
  The root directory will be '\cfg'.
  Note: if no valid user accounts can be found, all users will be treated as Admin Users (below).
- **Admin Users**
  The root directory will be '\', i.e. the user has unrestricted access to the file system.
  Note: The factory default admin login is 'ABX' (username) and 'FTPAccess' (password).

User Accounts

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

- **'\cfg\pswd\sys_pswd.cfg'**
  This file holds account details for normal users.
- **'\pswd\ad_pswd.cfg'**
  This file holds account details for admin users.

File Format:

The user accounts are stored in the following format:

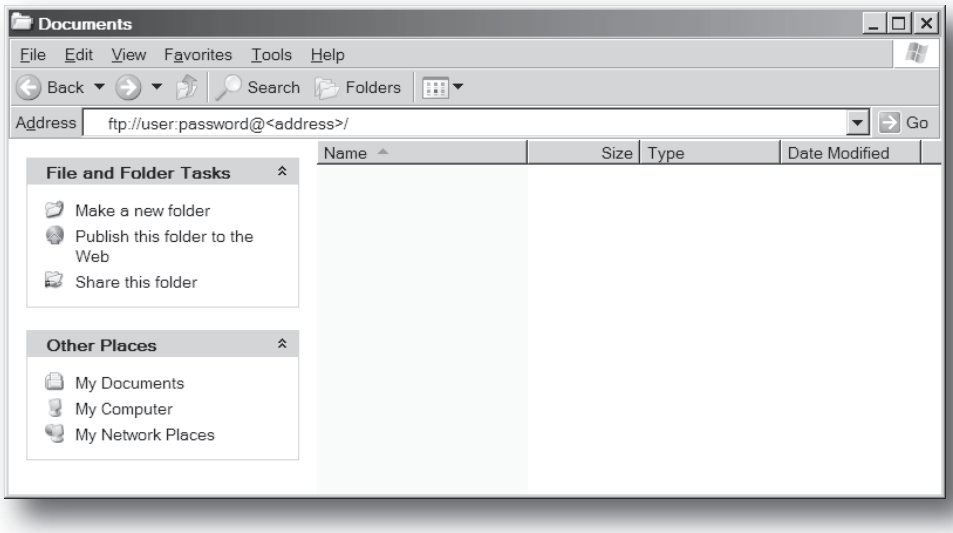
```
Username1:Password1
Username2:Password2
Username3:Password3
```

Accessing the FTP Server Using Windows Explorer™

The FTP client in Windows Explorer provides an easy way to access the file system as follows:

1. Open the Windows Explorer by right-clicking on the 'Start' button and selecting 'Explore'.
2. In the address field, type FTP://<user>:<password>@<address>
   - Substitute <address> with the IP address of the scanner interface
   - Substitute <user> with the username
   - Substitute <password> with the password
3. Press enter.

   The Explorer will now attempt to connect to the scanner interface using the specified settings. If successful, the file system will be displayed in the Explorer window.



8. E-mail Client

8.1 General Information

The built in e-mail client can send predefined e-mail messages based on trigger-events in the Input- and Output buffers.

In operation, this works as follows:

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

Note that the scanner interface process e-mail trigger events once every 0.5 seconds, which means that a trigger-event must be present longer than 0.5 seconds to be detected properly.

Which events that shall cause a particular message to be sent, is specified separately for each message. For more information, see 8-40 "E-mail Definitions". The client supports SSI, however note that some SSI functions cannot be used in e-mail messages (specified separately for each SSI function).

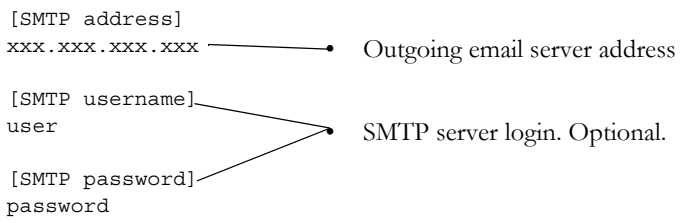
See also...

- 8-39 "SMTP Server Settings"
- 8-40 "E-mail Definitions"
- A-41 "Server Side Include (SSI)"

8.2 SMTP Server Settings

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

File Format:



8.3 E-mail Definitions

E-mail definitions are stored in the following directories:

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

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

File Format:

```
[Register]
Area, Offset, Type

[Register Match]
Value, Mask, Operand

[To]
recipient

[From]
sender

[Subject]
subject line

[Headers]
Optional extra headers

[Message]
message body
```

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

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

A. Server Side Include (SSI)

General

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

SSIs are special commands embedded in the source document. When the scanner interface encounters such a command, it will be executed, and if applicable, replaced by an output string.

Syntax

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

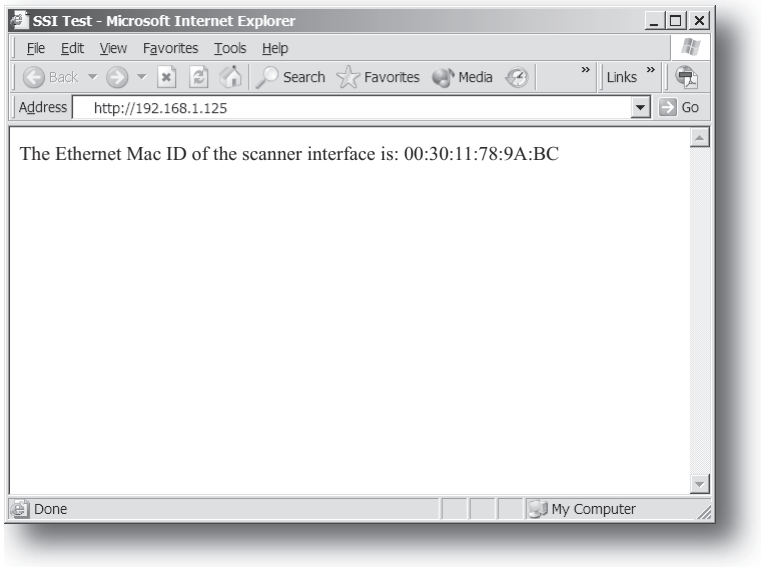
```
<?--#exec cmd_argument='XXXXXXXXXXXXXXXXXXXXX'-->
```

Example

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

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

Resulting web page



A.1 Functions

A.1.1 DisplayMacID

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

Syntax:

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

A.1.2 DisplaySerial

This function returns the serial number of the scanner interface.

Syntax:

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

A.1.3 DisplayFWVersion

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

Syntax:

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

A.1.4 DisplayBLVersion

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

Syntax:

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

A.1.5 DisplayIP

This function returns the current IP address.

Syntax:

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

A.1.6 DisplaySubnet

This function returns the current subnet mask.

Syntax:

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

A.1.7 DisplayGateway

This function returns the current gateway address.

Syntax:

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

A.1.8 DisplayDhcpState

This function returns whether DHCP is enabled or disabled.

Syntax:

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

A.1.9 StoreIPConfig

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

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

Syntax:

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

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

Accepted fields in form:

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

Default output:

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

A.1.10 GetText

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

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

Syntax:

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

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

Default output:

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

A.1.11 printf

This function includes a formatted string, which may contain data from the Anybus IN/OUT area, on a web page. The formatting of the string is equal to the standard C function printf().

Syntax:

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

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

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

-	which specifies left adjustment of the converted argument in its field.
+	which specifies that the number will always be printed with a sign
(space)	if the first character is not a sign, a space will be prefixed.
0	for numeric conversions, specifies padding to the field with leading zeroes.
#	which specifies an alternate output form. For o, the first digit will be zero. For x or X, 0x or 0X will be prefixed to a non-zero result. For e, E,f, g and G, the output will always have a decimal point; for g and G, trailing zeros will not be removed.

- A number specifying a minimum field width. The converted argument will be printed in a field at least this wide, and wider if necessary. If the converted argument has fewer characters than the field width it will be padded on the left (or right, if left adjustment has been requested) to make up the field width. The padding character is normally space, but can be 0 if the zero padding flag is present.
- A period, which separates the field width from the precision.
- A number, the precision, that specifies the maximum number of characters to be printed from a string, or the number of digits to be printed after the decimal point for e, E, or F conversions, or the number of significant digits for g or G conversion, or the minimum number of digits to be printed for an integer (leading 0s will be added to make up the necessary width)

- A length modifier h, l (letter ell), or L. "h" Indicates that the corresponding argument is to be printed as a short or unsigned short; "l" indicates that the argument is 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 behavior is undefined.

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

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

Argument	**Description**
InReadSByte(*offset*)	Reads a signed byte from position *offset* in the input buffer
InReadUByte(*offset*)	Reads an unsigned byte from position *offset* in the input buffer
InReadSWord(*offset*)	Reads a signed word (short) from position *offset* in the input buffer
InReadUWord(*offset*)	Reads an unsigned word (short) from position *offset* in the input buffer
InReadSLong(*offset*)	Reads a signed longword (long) from position *offset* in the input buffer
InReadULong(*offset*)	Reads an unsigned longword (long) from position *offset* in the input buffer
InReadString(*offset*)	Reads a string (char*) from position *offset* in the input buffer
InReadFloat(*offset*)	Reads a floating point (float) value from position *offset* in the input buffer
OutReadSByte(*offset*)	Reads a signed byte from position *offset* in the output buffer
OutReadUByte(*offset*)	Reads an unsigned byte from position *offset* in the output buffer
OutReadSWord(*offset*)	Reads a signed word (short) from position *offset* in the output buffer
OutReadUWord(*offset*)	Reads an unsigned word (short) from position *offset* in the output buffer
OutReadSLong(*offset*)	Reads a signed longword (long) from position *offset* in the output buffer
OutReadULong(*offset*)	Reads an unsigned longword (long) from position *offset* in the output buffer
OutReadString(*offset*)	Reads a NULL terminated string (char*) from position *offset* in the output buffer
OutReadFloat(*offset*)	Reads a floating point (float) value from position *offset* in the output buffer
MbReadSWord(*id*)	Used to control the gateway, see A-52 “Gateway Control”.
CipReadSByte(*class*, *inst*, *attr*)	Read a signed byte from a CIP-object
CipReadUByte(*class*, *inst*, *attr*)	Read an unsigned byte from a CIP-object
CipReadSWord(*class*, *inst*, *attr*)	Read a signed word from a CIP-object
CipReadUWord(*class*, *inst*, *attr*)	Read an unsigned word from a CIP-object
CipReadSLong(*class*, *inst*, *attr*)	Read a signed longword from a CIP-object
CipReadULong(*class*, *inst*, *attr*)	Read an unsigned longword from a CIP-object
CipReadFloat(*class*, *inst*, *attr*)	Read a floating point value from a CIP-object
CipReadShortString(*class*, *inst*, *attr*)	Read a short string from a CIP-object
CipReadString(*class*, *inst*, *attr*)	Read a null-terminated string from a CIP-object
CipReadUByteArray(*class*, *inst*, *attr*)	Read an unsigned byte-array from a CIP-object
CipReadUWordArray(*class*, *inst*, *attr*)	Read an unsigned word-array from a CIP-object
CipReadULongArray(*class*, *inst*, *attr*)	Read an unsigned longword-array from a CIP-object

A.1.12 scanf

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

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

Syntax:

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

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

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

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

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

Argument	Description
OutWriteByte(*offset*)	Writes a byte to position *offset* in the output buffer
OutWriteWord(*offset*)	Writes a word (short) to position *offset* in the output buffer
OutWriteLong(*offset*)	Writes a long to position *offset* in the output buffer
OutWriteString(*offset*)	Writes a string to position *offset* in the output buffer
OutWriteFloat(offset)	Writes a floating point (float) value to position *offset* in the output buffer
CipWriteByte(*class, inst, attr*)	Write a byte value to a CIP-object
CipWriteWord(*class, inst, attr*)	Write a word value to a CIP-object
CipWriteLong(*class, inst, attr*)	Write a longword to a CIP-object
CipWriteFloat(*class, inst, attr*)	Write a floating point value to a CIP-object

Default output:

```
Write succeeded
Write failed
```

A.1.13 IncludeFile

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

Syntax:

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

Default output:

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

A.1.14 SaveToFile

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

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

Syntax:

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

Default output:

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

A.1.15 SaveDataToFile

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

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

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

Syntax:

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

Default output:

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

A.1.16 DisplayScannerMode

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

Syntax:

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

A.1.17 SetScannerMode

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

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

Syntax:

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

Default output:

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

See also...

- 6-23 "Identity Object, Class 01h"

A.2 Changing SSI output

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

1. Changing SSI output defaults by creating a file called "\ssi_str.cfg" containing the output strings for all SSI functions in the system

2. Temporary changing the SSI output by calling the SSI function "SsiOutput()".

A.2.1 SSI Output String File

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

The files shall have the following format:

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

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

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

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

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

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

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

Example:

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

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

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

A.2.2 Temporary SSI Output change

The SSI output for the next SSI function to be called can be temporarily altered using the SSI function "SsiOutput()". Note that this will only affect the output string for the next SSI function, after that, the output strings will be reverted to their defaults, or to the strings defined in the file '\ssi_str.cfg'.

The maximum size of a string is 128 bytes.

Syntax:

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

Example:

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

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

A.3 Gateway Control

A.3.1 Refreshing Dynamic Gateway Status Information

The system files 'dynamic.txt' holds dynamic status information from the gateway and the onboard network interfaces. To provide up-to-date information, this file needs to be refreshed before use.

The following SSI command sequence will instruct the gateway to refresh the file:

Syntax:

```
<?--#exec cmd_argument='printf( "Data: %u", MbReadSWord( 21 ) )'-->
```

A.3.2 Restarting the Gateway

It is possible to reset the gateway using the following SSI command sequence:

Syntax:

```
<?--#exec cmd_argument='printf( "Data: %u", MbReadSWord( 1 ) )'-->
```

B. Technical Specification

B.1 Scanner Interface Details

- EtherNet/IP Scanner
- FTP Server
- Web Server
- SMTP Client
- 10/100 Mbit operation, full or half duplex
- On-board IP configuration switches
- Shielded or unshielded cables
- DHCP capable
- HICP capable (supports the Anybus IPconfig utility from HMS)
- DNS support

B.2 LAN (Ethernet) Connector Pinout (RJ45)

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