Network Interface Appendix

Anybus® CompactCom 30 ControlNet

Doc.Id. HMSI-168-79
Rev. 2.11

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

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

Important User Information

This document is intended to provide a good understanding of the functionality offered by Anybus CompactCom 30 ControlNet. The document only describes the features that are specific to the Anybus CompactCom 30 ControlNet. For general information regarding the Anybus CompactCom, consult the Anybus CompactCom design guides.

The reader of this document is expected to be familiar with high level software design, and communication systems in general. The use of advanced ControlNet-specific functionality may require in-depth knowledge in ControlNet networking internals and/or information from the official ControlNet specifications. In such cases, the people responsible for the implementation of this product should either obtain the ControlNet specification to gain sufficient knowledge or limit their 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 application 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

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 e.g. include 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.

Mar 2015 Doc Id HMSI-168-79

Table of Contents

Appendix E Timing & Performance

P. About This Document

For more information, documentation etc., please visit the HMS website, 'www.anybus.com'.

P.1 Related Documents

Document	Author
Anybus CompactCom 30 Software Design Guide	HMS
Anybus CompactCom Hardware Design Guide	HMS
Anybus CompactCom Software Driver User Guide	HMS
ControlNet Specification	ODVA
Common Industrial Protocol (CIP) specification	ODVA

P.2 Document History

Summary of Recent Changes (2.10... 2.11)

Change	Page(s)
Changed Get_Instance_By_Order to Get_Instance_Number_By_Order	10
Added CIP Identity Host Object (EDh)	38
Corrected error response description for command Set_Configuration_Data	43
Front view information moved to Technical Specification	51

Revision List

Revision	Date	Author(s)	Chapter(s)	Description
1.00	2009-09-01	KeL	-	First official release
2.00	2010-05-04	KeL	All	Change of concept
2.01	2011-02-10	KeL	P, 6	Minor updates
2.02	2011-05-04	KaD	P, 5, A	Minor updates
2.03	2012-01-26	KeL	P, A, 2	Minor updates
2.04	2012-04-13	KeL	2	Minor update
2.10	2012-10-04	KaD	4, 6, B	Update
2.11	2015-03-12	KeL	3, 6, D	Minor updates

P.3 Conventions & Terminology

The following conventions are used throughout this manual:

- Numbered lists provide sequential steps
- Bulleted lists provide information, not procedural steps
- The terms 'Anybus' or 'module' refers to the Anybus CompactCom module.
- The terms 'host' or 'host application' refers to the device that hosts the Anybus module.
- Hexadecimal values are written in the format NNNNh or 0xNNNN, where NNNN is the hexadecimal value.
- A byte always consists of 8 bits.

P.4 Support

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

1. About the Anybus CompactCom 30 ControlNet

1.1 General

The Anybus CompactCom 30 ControlNet communication module provides instant ControlNet connectivity via the patented Anybus CompactCom host interface. Any device that supports this standard can take advantage of the features offered by the module, allowing seamless network integration regardless of network type.

The modular approach of the Anybus CompactCom platform allows the CIP-object implementation to be extended to fit specific application requirements. Furthermore, the Identity Object can be customized, allowing the end product to appear as a vendor-specific implementation rather than a generic Anybus module.

This product conforms to all aspects of the host interface for Active modules defined in the Anybus CompactCom Hardware- and Software Design Guides, making it fully interchangeable with any other device following that specification. Generally, no additional network related software support is needed, however in order to be able to take full advantage of advanced network specific functionality, a certain degree of dedicated software support may be necessary.

1.2 Features

- Galvanically isolated bus electronics
- CIP Parameter Object Support
- Explicit messaging
- UCMM Capable
- Expansion possibilities via CIP forwarding
- Customizable Identity object
- Redundancy available

2. Tutorial

2.1 Introduction

This chapter is a complement to the Anybus CompactCom Implementation Tutorial. The ABCC tutorial describes and explains a simple example of an implementation with Anybus CompactCom. This chapter includes network specific settings that are needed for a host application to be up and running and possible to certify for use on ControlNet networks.

2.2 Fieldbus Conformance Notes

- The Anybus CompactCom 30 ControlNet has been pre-compliance tested by ODVA's authorized Independent Test Lab and found to comply with the ODVA Conformance Test Software. However, in accordance with ODVA's conformance test policy, the final product must still be compliance tested to ensure fieldbus conformance. In order to be able to do this, the vendor information in the ControlNet Host Object must be customized.
- Any change in the parameters in the EDS file, supplied by HMS, will require a new compliance test for certification.
- It is strongly recommended to customize the information in the Identity Object (CIP), to enable the product to appear as a vendor specific implementation rather than a generic Anybus module. ODVA requires that all manufacturers use their own Vendor ID. A Vendor ID can be applied for from ODVA.

  For further information, please contact HMS or ODVA.

2.3 Conformance Test Guide

When using the default settings of all parameters, the Anybus CompactCom ControlNet module is precertified for network compliance. This precertification is done to ensure that your product *can* be certified, but it does not mean that your product will not require certification.

Any change in the parameters in the EDS file, supplied by HMS, will require a certification. A Vendor ID can be obtained from ODVA and is compulsory for certification. This section provides a guide for successful conformance testing your product, containing the Anybus CompactCom ControlNet module, to comply with the demands for network certification set by the ODVA.

Independent of selected operation mode, the actions described in this section have to be accounted for in the certification process. The identity of the product needs to be changed to match your company and device.

IMPORTANT: *This section provides guidelines and examples of what is needed for certification. Depending on the functionality of your application, there may be additional steps to take. Please contact HMS Industrial Networks at www.anybus.com for more information.*

2.3.1 Reidentifying Your Product.

After successful setting of the "Setup Complete" attribute in the Anybus Object (01h), the Anybus module asks for identification data from the ControlNet Host Object (F3h). Therefore, the attributes listed below shall be implemented and proper values returned.

Object/Instance	Attribute	Explanation	Default	Customer sample	Comment
ControlNet Object (F3h), Instance 1	#1, Vendor ID	With this attribute you set the Vendor ID of the device.	005Ah (HMS)	1111h	This information must match the keyword values of the "Device" section in the EDS file.
ControlNet Object (F3h), Instance 1	#2, Device Type[a]	With this attribute you set the Device Type of the device.	0000h	002Bh[a] (Generic Device (keyable))	
ControlNet Object (F3h), Instance 1	#3, Product Code	With this attribute you set the Product Code of the device	002Ch	2222h	
ControlNet Object (F3h), Instance 1	#4, Revision	With this attribute you set the Revision of the device.		1.1	
ControlNet Object (F3h), Instance 1	#5, Serial Number	With this attribute you set the Serial Number of the device.		12345678h	Unique number for all CIP devices produced with the same 'Vendor ID.
ControlNet Object (F3h), Instance 1	#6, Product Name	With this attribute you set the Product Name of the device.	Anybus-CC ControlNet	"Widget"	This information must match the keyword values of the "Device" section in the EDS file.

a. The Device Type default value 0000h must be changed for the module to pass a conformance test. If no other specific profile is implemented, use the value 002Bh (Generic Device (keyable)).

2.3.2 Factory Default Reset

Reset command to Application Object (FFh) must be supported

When Anybus CompactCom 30 ControlNet modules are delivered, they are required to be in their "Factory Default" state. When a Factory Default Reset command is received from the network, the Anybus module will erase all non-volatile information and inform the host application that a reset of the Anybus module is required. This is done by sending a Reset command to the Application Object (FFh) of the host (Power-on + Factory Default). For more details, please consult the Anybus CompactCom 30 Software Design Guide.

3. Basic Operation

3.1 General Information

3.1.1 Software Requirements

No additional network support code needs to be written in order to support the Anybus CompactCom 30 ControlNet.

For in-depth information regarding the Anybus CompactCom 30 software interface, consult the general Anybus CompactCom 30 Software Design Guide.

3.1.2 Electronic Data Sheet (EDS)

Since the module implements the Parameter Object, it is possible for configuration tools such as -RS-NetWorx to automatically generate a suitable EDS-file.

Note that this functionality requires that the command 'Get_Instance_Number_By_Order' (Application Data Object, FEh) has been implemented in the host application.

See also...

- "Device Customization" on page 11
- "Parameter Object (0Fh)" on page 22 (CIP-object)
- Anybus CompactCom 30 Software Design Guide, 'Application Data Object (FEh)'

IMPORTANT: *To comply with CIP-specification requirements, custom EDS-implementations require a new Vendor ID and/or Product Code.*

To obtain a Product Code which complies to the default Vendor ID, please contact HMS.

3.2 Device Customization

By default, the module supports the generic CIP-profile with the following identity settings:

- Vendor ID: 005Ah (HMS Industrial Networks)
- Device Type: 0000h (Generic Device)
- Product Code: 002Ch (Anybus CompactCom 30 ControlNet)
- Product Name: 'Anybus-CC ControlNet'

It is possible to customize the identity of the module by implementing the ControlNet Host Object. Furthermore, it is possible to re-route requests to not implemented CIP-objects to the host application, thus enabling support for other profiles etc.

To support a specific profile, perform the following steps:

- Set up the identity settings in the ControlNet Host Object according to profile requirements.
- Set up the Assembly Instance Numbers according to profile requirements.
- Enable routing of CIP-messages to the host application in the ControlNet Host Object.
- Implement the required CIP-objects in the host application.

See also...

- "Identity Object (01h)" on page 15 (CIP-object)
- "ControlNet Host Object (F3h)" on page 40 (Host Application Object)
- "Command Details: Process_CIP_Message_Request" on page 42

IMPORTANT: *The default identity information is valid only when using the standard EDS-file supplied by HMS. To comply with CIP-specification requirements, custom EDS-implementations require a new Vendor ID and/or Product Code.*

To obtain a Product Code which complies to the default Vendor ID, please contact HMS.

3.3 Communication Settings

As with other Anybus CompactCom products, network related communication settings are grouped in the Network Configuration Object (04h).

In this case, this includes...

- **Mac ID**

  See also...

  - "Instance Attributes (Instance #1, 'Device Address')"' on page 36

The parameters in the Network Configuration Object (04h) are available from the network through the Identity Object (CIP-object).

See also...

- "Identity Object (01h)" on page 15 (CIP-object)
- "Network Configuration Object (04h)" on page 35 (Anybus Module Object)

3.4 Diagnostics

The severity value of all pending events are combined (using logical OR) and copied to the corresponding bits in the 'Status'-attribute of the CIP Identity Object.

See also...

- "Identity Object (01h)" on page 15 (CIP-object)
- "Diagnostic Object (02h)" on page 33 (Anybus Module Object)

3.5 Data Exchange

3.5.1 Application Data (ADIs)

ADIs are represented on ControlNet through the ADI Object (CIP-object). Each instance within this objects corresponds directly to an instance in the Application Data Object on the host application side.

See also...

- "Parameter Object (0Fh)" on page 22 (CIP-object)
- "ADI Object (A2h)" on page 28 (CIP-object)

3.5.2 Process Data

Process Data is represented on ControlNet through dedicated instances in the Assembly Object. Note that each ADI element is mapped on a byte-boundary, i.e. each BOOL occupies one byte.

See also...

- "Assembly Object (04h)" on page 19 (CIP-object)

3.5.3 Translation of Data Types

The Anybus data types are translated to CIP-standard and vice versa according to the table below.

Anybus Data Type	CIP Data Type	Comments
BOOL	BOOL	Each ADI element of this type occupies one byte.
ENUM	USINT	
SINT8	SINT	
UINT8	USINT	
SINT16	INT	Each ADI element of this type occupies two bytes.
UINT16	UINT	
SINT32	DINT	Each ADI element of this type occupies four bytes.
UINT32	UDINT	
FLOAT	REAL	
CHAR	SHORT_STRING	SHORT_STRING consists of a single-byte length field (which in this case represents the number of ADI elements) followed by the actual character data (in this case the actual ADI elements). This means that a 10-character string occupies 11 bytes.
SINT64	LINT	Each ADI element of this type occupies eight bytes.
UINT64	ULINT	

4. CIP Objects

4.1 General Information

This chapter specifies the CIP-objects implementation in the module. The objects described herein can be accessed from the network, but not by the host application.

Mandatory Objects[1]:

- "Identity Object (01h)" on page 15
- "Message Router (02h)" on page 18
- "Assembly Object (04h)" on page 19
- "Connection Manager (06h)" on page 21
- "Parameter Object (0Fh)" on page 22
- "ControlNet Object (F0h)" on page 25

Vendor Specific Objects:

- "ADI Object (A2h)" on page 28

It is possible to implement additional CIP-objects in the host application using the CIP forwarding functionality, see "ControlNet Host Object (F3h)" on page 40 and "Command Details: Process_CIP_Message_Request" on page 42.

1. These objects are mandatory to the module and implemented in the module by default. No action from the application is expected or required.

4.2 Identity Object (01h)

Category

Extended

Object Description

This object provides identification of and general information about the module.

Supported Services

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

Class Attributes

#	Access	Name	Type	Value	Description
1	Get	Revision	UINT	0001h	Revision 1
2	Get	Max Instance	UINT	0001h	Maximum instance number 1

Instance #1 Attributes

Extended

#	Access	Name	Type	Value	Description
1	Get	Vendor ID	UINT	005Ah[a]	HMS Industrial Networks AB
2	Get	Device Type	UINT	0000h[a]	Generic Device
3	Get	Product Code	UINT	002Ch[a]	Anybus CompactCom ControlNet
4	Get	Revision	Struct of: {USINT, USINT}	N/A[a]	Major and minor firmware revision
5	Get	Status	WORD	-	See 4-17 “Device Status”
6	Get	Serial Number	UDINT	Serial number[a]	Assigned by HMS
7	Get	Product Name	SHORT_STRING	“Anybus Compact-Com 30 Control-Net” [a]	Product name
11	Set	Active language	Struct of: {USINT, USINT, USINT}	N/A	Requests sent to this instance are forwarded to the Application Object. The host application is then responsible for updating the language settings accordingly.
12	Get	Supported Language List	Array of struct of: {USINT, USINT, USINT}	N/A	List of languages supported by the host application. This list is read from the Application Object during the NW_INIT state, and translated to CIP standard.[b]

a. Can be customized by implementing the ControlNet Host Object, see “ControlNet Host Object (F3h)” on page 40
b. By default the only supported language is English. To enable more languages the application has to implement the corresponding attributes in the application object.

Device Status

Bit(s)	Name
0	Module Owned
1	(reserved)
2	Configured[a]
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 (other) (reserved)
8	Set for minor recoverable faults[b]
9	Set for minor unrecoverable faults[b]
10	Set for major recoverable faults[b]
11	Set for major unrecoverable faults[b]
12... 15	(reserved)

a. This bit shows if the product has other settings than "out-of-box". The value is set to true if the configured attribute in the Application Object is set.

b. See "Diagnostic Object (02h)" on page 33.

Service Details: Reset Service

The module forwards reset requests from the network to the host application. For more information about network reset handling, consult the general Anybus CompactCom Design Guide.

There are two types of network reset requests on ControlNet:

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

  This service emulates a power cycling of the module, and corresponds to Anybus reset type 0 (Power cycling). For further information, consult the general Anybus CompactCom 30 Software Design Guide.

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

  This service sets a "out of box" configuration and performs a reset, and corresponds to Anybus reset type 2 (Power cycling + factory default). For further information, consult the general Anybus CompactCom 30 Software Design Guide.

4.3 Message Router (02h)

Category

Extended

Object Description

This object provides access to CIP addressable objects within the device.

Supported Services

Class	-
Instance:	-

Class Attributes

-

Instance Attributes

-

4.4 Assembly Object (04h)

Category

Extended

Object Description

The Assembly object uses static assemblies and holds the Process Data sent/received by the host application. The default assembly instance IDs used are in the vendor specific range.

See also...

- "Process Data" on page 13
- "ControlNet Host Object (F3h)" on page 40

Supported Services

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

Class Attributes

#	Name	Access	Type	Value	Comments
1	Revision	Get	UINT	0002h	Revision 2

Instance 05h Attributes (Configuration Data)

The instance number for this instance can be changed by implementing the corresponding attribute in the ControlNet Host Object.

Configuration data that is sent through the Forward_Open service will be written to this instance.

#	Name	Access	Type	Comments
3	Data	Get/Set	N/A	Configuration data written to the application when the Forward_Open service has configuration data included

See also...

- "ControlNet Host Object (F3h)" on page 40
- "Command Details: Set_Configuration_Data" on page 43

Instance 64h Attributes (Producing Instance)

The instance number for this instance can be changed by implementing the corresponding attribute in the ControlNet Host Object.

#	Name	Access	Type	Comments
3	Produced Data	Get	Array of BYTE	This data corresponds to the Write Process Data

See also...

- "Data Exchange" on page 13
- "ControlNet Host Object (F3h)" on page 40

Instance 96h Attributes (Consuming Instance)

The instance number for this instance can be changed by implementing the corresponding attribute in the ControlNet Host Object.

#	Name	Access	Type	Comments
3	Consumed Data	Set	Array of BYTE	This data corresponds to the Read Process Data

See also...

- "Data Exchange" on page 13
- "ControlNet Host Object (F3h)" on page 40

4.5 Connection Manager (06h)

Category

Extended

Object Description

This object is used for connection and connectionless communications.

Supported Services

Class	-
Instance:	Forward Open Forward Close

Class Attributes

-

Instances Attributes

-

4.6 Parameter Object (0Fh)

Category

Extended

Object Description

This object allows configuration tools such as RSNetworx to extract information about the Application Data Instances (ADIs) and present them with their actual name and range to the user.

Since this process may be somewhat time consuming, especially when using the serial host interface, it is possible to disable support for this functionality in the ControlNet Host Object.

Due to limitations imposed by the CIP standard, ADIs containing multiple elements (i.e. arrays etc.) cannot be represented through this object. In such cases, default values will be returned, see 4-23 "Default Values".

See also...

- "Default Values" on page 23
- "ADI Object (A2h)" on page 28 (CIP Object)
- "ControlNet Host Object (F3h)" on page 40 (Host Application Object)

Supported Services

Class	Get Attribute Single
Instance:	Get Attribute Single Set Attribute Single Get Attributes All Get Enum String

Class Attributes

#	Name	Access	Type	Value
1	Revision	Get	UINT	0001h (Revision of the object)
2	Max instance	Get	UINT	Maximum created instance number = class attribute 3 in the Application Data Object[a]
8	Parameter class descriptor	Get	WORD	Default: 0000 0000 0000 01011b Bit:Contents: 0 Supports parameter instances 1 Supports full attributes 2 Must do non-volatile storage save command 3 Parameters are stored in non-volatile storage
9	Configuration Assembly instance	Get	UINT	0000h (Configuration assembly not supported)

a. Consult the general Anybus CompactCom 30 Software Design Guide for further information.

Instance Attributes

Extended

#	Name	Access	Type	Value
1	Parameter Value	Get/Set	Specified in attributes 4, 5 & 6.	Actual value of parameter This attribute is read-only if bit 4 of Attribute #4 is true
2	Link Path Size	Get	USINT	0007h
3	Link Path	Get	Packed EPATH	20h A2h 25h nn nn 30h 05h (Path to the object from where this parameter's value is retrieved, in this case the ADI Object)
4	Descriptor	Get	WORD	Bit:Contents: 0 Supports Settable Path (N/A) 1 Supports Enumerated Strings 2 Supports Scaling (N/A) 3 Supports Scaling Links (N/A) 4 Read only Parameter 5 Monitor Parameter (N/A) 6 Supports Extended Precision Scaling (N/A) 14 Write only Parameter
5	Data type	Get	EPATH	Data type code
6	Data size	Get	USINT	Number of bytes in parameter value
7	Parameter Name String	Get	SHORT_STRING	Name of the parameter, truncated to 16 chars
8	Units String	Get	SHORT_STRING	(not supported)
9	Help String	Get	SHORT_STRING	
10	Minimum value	Get	(Data Type)	Minimum value of parameter Minimum length of strings
11	Maximum value	Get	(Data Type)	Maximum value of parameter Maximum length of strings
12	Default value	Get	(Data Type)	Default value of parameter
13	Scaling Multiplier	Get	UINT	0001h (not supported)
14	Scaling Divisor	Get	UINT	
15	Scaling Base	Get	UINT	
16	Scaling Offset	Get	INT	0000h (not supported)
17	Multiplier link	Get	UINT	
18	Divisor Link	Get	UINT	
19	Base Link	Get	UINT	
20	Offset Link	Get	UINT	
21	Decimal precision	Get	USINT	

Default Values

#	Name	Value	Description
1	Parameter Value	0	-
2	Link Path Size	0	Size of link path in bytes.
3	Link Path	-	NULL Path
4	Descriptor	0010h	Read only Parameter
5	Data type	C6h	USINT
6	Data size	1	-
7	Parameter Name String	(reserved)	-
8	Units String	“”	-
9	Help String	“”	-
10	Minimum value	N/A	0
11	Maximum value	N/A	0

#	Name	Value	Description
12	Default value	N/A	0

4.7 ControlNet Object (F0h)

Category

Extended

Object Description

This object provides a consistent Station Management interface to the Physical and Data Link Layers. The object makes diagnostic information from these layers available to client applications. Each node shall support one ControlNet object per link.

Supported Services

Class	Get Attribute Single
Instance:	Get Attribute Single Set Attribute Single[1] Get And Clear Reset Enter Listen Only

1. Set once, depending on how MAC ID have been set.

Class Attributes

#	Name	Access	Type	Value	Description
1	Revision	Get	UINT	0001h	Revision 1
2	Max instance	Get	UDINT	00000001h	Maximum instance number

Instance #1 Attributes

Extended

#	Name	Access	Type	Description
81h	Current_link_config	Get	Struct of:	
	Link_config		Struct of:	
	NUT_length		UINT	NUT length in steps of 10 µs
	smax		USINT	0... 99
	umax		USINT	1... 99
	slotTime		USINT	In steps of 1 µs
	blanking		USINT	In steps of 1.6 µs
	gb_start		USINT	In steps of 10 µs
	gb_center		USINT	In steps of 10 µs
	reserved		UINT	Reserved
	modulus		USINT	Value: 127 (required)
	gb_prestart		USINT	In steps of 10 µs
	TUI		Struct of:	
	unique_ID		UDINT	Keeper CRC
	status_flag		UINT	TUI flag
	reserved		USINT[16]	Reserved

#	Name	Access	Type	Description
82h	diagnostic_counters	Get Get And Clear	Struct of:	
	buffer_errors		UINT	Buffer event counter
	error_log		BYTE[8]	Bad MAC frame log
	event_counters		Struct of:	
	good_frames_transmitted		BYTE[3]	Good MAC frames transmitted (LSB first)
	good_frames_received		BYTE[3]	Good MAC frames received (LSB first)
	selected_channel_-frame_errors		USINT	Framing errors detected on active receive channel
	channel_A_frame_errors		USINT	Framing errors detected on channel A
	channel_B_frame_errors		USINT	Framing errors detected on channel B
	aborted_frames_transmit-ted		USINT	MAC frames aborted during transmission (transmit underflows)
	highwaters		USINT	LLC transmit underflow and LLC receive overflow
	NUT_overloads		USINT	No unscheduled time NUT (All time used for sched-uled transmissions)
	slot_overloads		USINT	More scheduled data queued for one NUT than allowed by sched_max_frame parameter
	blockages		USINT	Single Lpacket size exceeds sched_max_frame parameter
	non_concurrence		USINT	Two or more nodes could not agree whose turn it is to transmit
	aborted_frames_received		USINT	Incomplete MAC frames received
	lonely_counter		USINT	Number of times nothing detected on network for 8 or more NUTs
	duplicate_node		USINT	MAC frame received from node with local node's MAC ID
	noise_hits		USINT	Noise detected that locked the modem rx PLL
	collisions		USINT	Rx data detected just before start of transmission
	Mod_MAC_ID		USINT	MAC ID of current moderator node
	non_lowman_modes		USINT	Moderator frames detected from non-lowman nodes
	rouge_count		USINT	Rouge events detected
	unheard_moderator		USINT	MAC frames being detected but no moderators
	vendor_specific		USINT	-
	reserved		BYTE[4]	(reserved)
	vendor_specific		USINT	00h. Not used
	vendor_specific		USINT	00h. Not used
	reserved		BYTE	(reserved)
83h	station_status	Get	Struct of:	
	smac_ver		USINT	MAC implementation
	vendor_specific		BYTE(4)	Vendor specific
	channel_state		BYTE	Channel state LEDs, redundancy warning, and active channel bits
84h	MAC_ID	Get/Set[a]	Struct of:	
	MAC_ID_current		USINT	Current MAC ID
	MAC_ID_switches		USINT	MAC ID switch settings
	MAC_ID_changes		BOOL	MAC ID switches changed since reset
	reserved		USINT	(reserved)
86h	error_log	Get	Struct of:	
	buffer_errors		USINT	Buffer event counter
	error_log		BYTE[8]	Bad MAC frame log

a. Set once, depending on how MAC ID have been set

4.8 ADI Object (A2h)

Category

Extended

Object Description

This object maps instances in the Application Data Object to ControlNet. All requests to this object will be translated into explicit object requests towards the Application Data Object in the host application; the response is then translated back to CIP-format and sent to the originator of the request.

Class attributes have to be converted to follow the ControlNet specifications

See also...

- Application Data Object (see Anybus CompactCom 30 Software Design Guide)
- "Parameter Object (0Fh)" on page 22 (CIP Object)

Supported Services

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

Class Attributes

#	Name	Access	Type	Value
1	Revision	Get	UINT	Object revision (Current value = 0001h)
2	Max Instance	Get	UINT	Equals attribute #4 in the Application Data Object[a]
3	Number of instances	Get	UINT	Equals attribute #3 in the Application Data Object[a]

a. Consult the general Anybus CompactCom 30 Software Design Guide for further information.

Instances Attributes

Each instance corresponds to an instance within the Application Data Object (for more information, consult the general Anybus CompactCom 30 Software Design Guide)

Extended.

#	Name	Access	Type	Description
1	Name	Get	SHORT_STRING	Parameter name (Including length)
2	ABCC Data type	Get	USINT	Data type of instance value
3	No. of elements	Get	USINT	Number of elements of the specified data type
4	Descriptor	Get	USINT	Bit field describing the access rights for this instance Bit:Meaning: 0 Set = Get Access 1 Set = Set Access

#	Name	Access	Type	Description
5	Value	Get/Set	Determined by attribute #2	Instance value
6	Max value	Get		The maximum permitted parameter value.
7	Min value	Get		The minimum permitted parameter value.
8	Default value	Get		The default parameter value.

5. Anybus Module Objects

5.1 General Information

This chapter specifies the Anybus Module Object implementation and how they correspond to the functionality in the Anybus CompactCom 30 ControlNet.

The following Anybus Module Objects are implemented:

5.2 Anybus Object (01h)

Category

Basic, extended

Object Description

This object assembles all common Anybus data, and is described thoroughly in the general Anybus CompactCom 30 Software Design Guide.

Supported Commands

Object:	Get_Attribute
Instance:	Get_Attribute Set_Attribute Get_Enum_String

Object Attributes (Instance #0)

(Consult the general Anybus CompactCom 30 Software Design Guide for further information.)

Instance Attributes (Instance #1)

Basic

#	Name	Access	Type	Value
1	Module type	Get	UINT16	0401h (Standard Anybus CompactCom)
2... 11	-	-	-	Consult the general Anybus CompactCom 30 Software Design Guide for further information.
12	LED colors	Get	struct of: UINT8(LED1A) UINT8(LED1B) UINT8(LED2A) UINT8(LED2B)	Value:Color: 01h Green 02h Red 01h Green 02h Red
13... 15	-	-	-	Consult the general Anybus CompactCom 30 Software Design Guide for further information.

Extended

#	Name	Access	Type	Value
16	GPIO configuration	Get/Set[a]	UINT16	Configuration of the host interface GPIO pins. See the table below.

a. Set access of attribute GPIO configuration is only valid in state SETUP.

GPIO configuration Settings

Value	Functionality	Description
0x0000	Standard	GIP[0..1] and $\overline{\text{GOP}}$[0..1] are used as general input/output pins LED1[A..B] is used for network status LED A LED2[A..B] is used for module status LED
0x0001	Extended LED functionality	GIP0 (red) and GIP1 (green) are used for network status LED A GOP0 (red) and GOP1 (green) are used for network status LED B LED1[A..B] is disabled LED2[A..B] is used for module status LED

For more information, see

- "Extended LED Functionality" in "Appendix A" on page 45.

5.3 Diagnostic Object (02h)

Category

Basic

Object Description

This object provides a standardised way of handling host application events & diagnostics, and is thoroughly described in the general Anybus CompactCom 30 Software Design Guide.

Supported Commands

Object:	Get_Attribute Create Delete
Instance:	Get_Attribute

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1	Name	Get	Array of CHAR	'Diagnostic'
2	Revision	Get	UINT8	01h
3	Number of instances	Get	UINT16	See general Anybus CompactCom 30 Software Design Guide
4	Highest instance no.	Get	UINT16	
11	Max no. of instances	Get	UINT16	5+1

Instance Attributes (Instance #1)

Basic

#	Name	Access	Type	Value
1	Severity	Get	UINT8	See general Anybus CompactCom 30 Software Design Guide
2	Event Code	Get	UINT8	

In the Anybus CompactCom 30 ControlNet, the severity level of all instances are logically OR:ed together and represented on the network through the CIP Identity Object. The Event Code cannot be represented on the network and is thus ignored by the module.

See also...

- "Diagnostics" on page 12
- "Identity Object (01h)" on page 15 (CIP-object)

5.4 Network Object (03h)

Category

Basic

Object Description

For more information regarding this object, consult the general Anybus CompactCom 30 Software Design Guide.

Supported Commands

Object: Get_Attribute

Instance: Get_Attribute
Set_Attribute
Get_Enum_String
Map_ADI_Write_Area
Map_ADI_Read_Area

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1	Name	Get	Array of CHAR	“Network”
2	Revision	Get	UINT8	01h
3	Number of instances	Get	UINT16	0001h
4	Highest instance no.	Get	UINT16	0001h

Instance Attributes (Instance #1)

Basic

#	Name	Access	Type	Value
1	Network type	Get	UINT16	0065h
2	Network type string	Get	Array of CHAR	‘ControlNet’
3	Data format	Get	ENUM	00h (LSB first)
4	Parameter data support	Get	BOOL	True
5	Write process data size	Get	UINT16	Current write process data size (in bytes) Updated on every successful Map_ADI_Write_Area[a]
6	Read process data size	Get	UINT16	Current read process data size (in bytes) Updated on every successful Map_ADI_Read_Area[a]
7	Exception information	Get	UINT8	Additional ControlNet exception information presented if the Anybus module has entered state EXCEPTION. 00h = no information

a. Consult the general Anybus CompactCom 30 Software Design Guide for further information.

5.5 Network Configuration Object (04h)

Category

Basic

Object Description

This object holds network specific configuration parameters that may be set by the end user. A reset command (factory default) issued towards this object will result in all instances being set to their default values.

See also...

- "Communication Settings" on page 12
- "Identity Object (01h)" on page 15 (CIP-object)

Supported Commands

Object:	Get_Attribute Reset
Instance:	Get_Attribute Set_Attribute Get_Enum_String

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1	Name	Get	Array of CHAR	'Network configuration'
2	Revision	Get	UINT8	01h
3	Number of instances	Get	UINT16	0001h
4	Highest instance no.	Get	UINT16	0001h

5.5.1 Instance Attributes (Instance #1, 'Device Address')'

Basic

#	Name	Access	Type	Description
1	Name[a]	Get	Array of CHAR	'Address' (Actual ControlNet node address)
2	Data type	Get	UINT8	04h (= UINT8)
3	Number of elements	Get	UINT8	01h (one element)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value[b]	Get/Set	UINT8	Node address Valid values: 1-99 0, 100-255: "Incorrect node configuration" Default: 255

a. Multilingual, see "Multilingual Strings" on page 36.
b. A 'Get' command always returns the actual value.

5.5.2 Multilingual Strings

The instance name in this object is multilingual, and is translated based on the current language settings as follows:

Instance	English	German	Spanish	Italian	French
1	Address	Adresse	Dirección	Indirizzo	Adresse

6. Host Application Objects

6.1 General Information

This chapter specifies the host application object implementation in the module. The objects listed here may optionally be implemented within the host application firmware to expand the ControlNet implementation.

Standard Objects:

- Application Object (see Anybus CompactCom 30 Software Design Guide)
- Application Data Object (see Anybus CompactCom 30 Software Design Guide)

Network Specific Objects:

- "CIP Identity Host Object (EDh)" on page 38
- "ControlNet Host Object (F3h)" on page 40

6.2 CIP Identity Host Object (EDh)

Category

Advanced

Object Description

This object allows for applications to support additional CIP identity instances. It is used to provide additional product identity information.

The first instance in the CIP identity object will not change its behavior. When implementing instances in the CIP identity host object, they will be mapped to the CIP identity object starting at instance 2. Instance no. 1 in the CIP identity host object will be mapped to instance no. 2 in the CIP identity object and so on.

Supported Commands

Instance: Get_Attribute_All

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value	Description
1	Name	Get	STRING	“CIP Identity”	Object name
2	Revision	Get	UINT8	01h	Object revision
3	Number of instances	Get	UINT16	Depends on application	Supported number of instances
4	Highest instance no.	Get	UINT16	Depends on application	Highest implemented instance

Instance Attributes (Instance #1)

Extended

#	Name	Access	Type	Value	Comment
1	Vendor ID	Get	UINT16	-	These values replace the default values for the CIP Identity object.
2	Device type	Get	UINT16	-	
3	Product code	Get	UINT16	-	
4	Revision	Get	Struct of UINT8	-	
	Major revision		UINT8	-	
	Minor revision		UINT8	-	
5	Status	Get	UINT16	-	
6	Serial number	Get	UINT32	-	
7	Product name	Get	Array of CHAR	-	

Command Details: Get_Attribute_All

Category

Advanced

Details

Command Code.: 10h

Valid for: Object Instance

Description

This service must be implemented by the application for all instances that exist in the CIP identity host object. If identity data is requested from the network the Anybus module will issue this command to the application. The application will then respond with a message containing a struct of all attributes in the requested instance.

- **Command Details**

Byte	Parameter	Data type	Description
0	Source ID	UINT8	Selected by the Anybus module
1	Destination object	UINT8	CIP identity host object
2, 3	Instance	UINT16	Instance number
4	Command	UINT8	50h = Get_Attribute_All (command bit set)
5	Data field size (in bytes)	UINT8	0
6	CmdExt[0]	UINT8	0
7	CmdExt[1]	UINT8	0

- **Response Details**

Byte	Parameter	Data type	Description
0	Source ID	UNIT8	Selected by the Anybus module
1	Destination object	UNIT8	CIP identity host object
2, 3	Instance	UINT16	Instance number
4	Command	UNIT8	10h = Get_Attribute_All response
5	Data field size (in bytes)	UNIT8	Number of bytes in the data field
6	CmdExt[0]	UNIT8	0
7	CmdExt[1]	UNIT8	0
0, 1	Vendor ID	UINT16	CIP identity host data
2, 3	Device type	UINT16	
4, 5	Product code	UINT16	
6	Major revision	UNIT8	
7	Minor revision	UNIT8	
8, 9	Status	UINT16	
10 - 13	Serial number	UINT32	
14 - n	Product name	Array of CHAR	

6.3 ControlNet Host Object (F3h)

Category

Basic, extended, advanced

Object Description

This object implements ControlNet specific settings in the host application. It is also used when implementing ControlNet classes in the host application, e.g. when creating profile implementations etc.

The implementation of this object is optional; the host application can support none, some, or all of the attributes specified below. The module will attempt to retrieve the values of these attributes during start-up; if an attribute is not implemented in the host application, simply respond with an error message (06h, "Invalid CmdExt[0]"). In such case, the module will use its default value.

If the module attempts to retrieve a value of an attribute not listed below, respond with an error message (06h, "Invalid CmdExt[0]").

See also...

- "Identity Object (01h)" on page 15
- "Assembly Object (04h)" on page 19
- Anybus CompactCom 30 Software Design Guide, "Error Codes"

IMPORTANT: *To comply with CIP-specification requirements, the combination of Vendor ID (instance attribute #1) and serial number (instance attribute #5) must be unique. The default Vendor ID, serial number, and Product Code combination is valid only if using the standard ESD-file supplied by HMS.*

Supported Commands

Object: Process_CIP_Message_Request (See "Command Details: Process_CIP_Message_Request" on page 42)
Set_Configuration_Data (See "Command Details: Set_Configuration_Data" on page 43)
Get_Configuration_Data (See "Command Details: Get_Configuration_Data" on page 44)

Instance: -

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1	Name	Get	Array of CHAR	'ControlNet'
2	Revision	Get	UINT8	01h
3	Number of instances	Get	UINT16	0001h
4	Highest instance no.	Get	UINT16	0001h

Instance Attributes (Instance #1)

Basic

#	Name	Access	Type	Default Value	Comment
1	Vendor ID	Get	UINT16	005Ah	These values are forwarded to the ControlNet Identity Object (CIP).
2	Device Type	Get	UINT16	0000h	
3	Product Code	Get	UINT16	002Ch	
4	Revision	Get	struct of: UINT8 Major UINT8 Minor	(software revision)	
5	Serial Number	Get	UINT32	(set at production)	
6	Product Name	Get	Array of CHAR	'Anybus-CC ControlNet'	

Extended

#	Name	Access	Type	Default Value	Comment
7	Producing Instance No.	Get	UINT16	0064h	See also... - "Instance 64h Attributes (Producing Instance)" on page 20 (CIP-instance)
8	Consuming Instance No.	Get	UINT16	0096h	See also... - "Instance 96h Attributes (Consuming Instance)" on page 20 (CIP-instance)
12	Enable Parameter Object	Get	BOOL	True	Value:Meaning: True: Enable CIP Parameter Object False: Disable CIP Parameter Object If the parameter object is disabled and CIP routing is enabled, any requests will be routed to the application See also... - "Parameter Object (0Fh)" on page 22 (CIP-object)
15	Assembly object configuration instance number	Get	UINT16	0005h	See also... - "Instance 05h Attributes (Configuration Data)" on page 19 (CIP-instance)

Advanced

#	Name	Access	Type	Default Value	Comment
11	Enable CIP request forwarding	Get	BOOL	False	Value:Meaning: True: Requests to unknown ControlNet objects or unknown assembly object instances are routed to the application - "Command Details: Process_CIP_Message_Request" on page 42

6.3.1 Command Details: Process_CIP_Message_Request

Category

Advanced

Details

Command Code: 10h

Valid for: Object Instance

Description

By setting the 'Enable CIP Request Forwarding'-attribute (#11), all requests to unimplemented CIP-objects or unimplemented assembly object instances will be forwarded to the host application through this command. The application then has to evaluate the request and return a proper response. The module supports up to 6 pending CIP-requests; additional requests will be rejected by the module.

Note that since the telegram length on the host interface is limited, the request data size must not exceed 255 bytes. If it does, a the module will send a 'resource unavailable' response to the originator of the request and the message will not be forwarded to the host application.

Note: This command is identical to the 'Process_CIP_Request'-command in the Anybus CompactCom Ethernet, similar - but not identical - to the 'Process_CIP_Request'-command in the Anybus CompactCom DeviceNet.

- **Command Details**

Field	Contents	Notes
CmdExt[0]	CIP Service Code	CIP service code from original CIP request
CmdExt[1]	Request Path Size	Number of 16-bit words in the Request Path field
MsgData[0... m]	Request Path	CIP Padded EPATH (Class, Instance, Attr. etc.)
MsgData[m... n]	Request Data	Service-specific data

- **Response Details**

Field	Contents	Notes
CmdExt[0]	CIP Service Code	(Reply bit set)
CmdExt[1]	00h	(reserved, set to zero)
MsgData[0]	General Status	CIP General Status Code
MsgData[1]	Size of Additional Status	Number of 16-bit words in Additional Status array
MsgData[2... m]	Additional Status	Additional Status, if applicable
MsgData[m... n]	Response data	Actual response data, if applicable

IMPORTANT: *When using this functionality, make sure to implement the common CIP Class Attribute (attribute #1, 'Revision') for all objects in the host application firmware. Failure to observe this will prevent the module from successfully passing conformance tests.*

6.3.2 Command Details: Set_Configuration_Data

Category

Advanced

Details

Command Code: 11h

Valid for: Object Instance

Description

If the data segment in the CIP Forward_Open service contains configuration data, this will be forwarded to the host application with this command. If implemented, the host application should evaluate the request and return a proper response.

Since the telegram length on the host interface is limited, segmentation is needed for data sizes larger than 255 bytes. The maximum total amount of configuration data that will be accepted by the module is 458 bytes.

Note: This command must be implemented in order to support configuration data. If not implemented, the CIP Forward_Open request will be rejected by the module if it contains configuration data.

- **Command Details**

Field	Contents
CmdExt[0]	Reserved
CmdExt[1]	Segmentation control bits (see "Message Segmentation" on page 47)
MsgData[0... n]	Actual configuration data

- **Response Details (success)**

Field	Contents
CmdExt[0]	(reserved, set to zero)
CmdExt[1]	(reserved, set to zero)
MsgData[0]	00h: accepted

- **Response Details (error)**

Field	Contents
CmdExt[0]	(reserved, set to zero)
CmdExt[1]	(reserved, set to zero)
MsgData[0]	Anybus protocol error code
MsgData[1]	If MsgData[0] equals 0Ch (value out of range), the content of this field points to the erroneous attribute.

6.3.3 Command Details: Get_Configuration_Data

Category

Advanced

Details

Command Code: 13h

Valid for: Object Instance

Description

If the configuration data is requested from the network, the Anybus module will issue this command to the application. The application shall send the stored configuration data in the response message.

Since the telegram length on the host interface is limited, segmentation is needed for data sizes larger than 255 bytes. The maximum total amount of configuration data that will be accepted by the module is 458 bytes.

Note: This command must be implemented in order to support configuration data. If not implemented, the request will be rejected by the Anybus module.

- **Command Details**

Field	Contents
CmdExt[0]	0
CmdExt[1]	0

- **Response Details (success)**

Field	Contents
CmdExt[0]	0
CmdExt[1]	Segmentation control bits (see "Message Segmentation" on page 47)
MsgData[0-n]	Configuration data from the application

- **Response Details (error)**

Field	Contents
CmdExt[0]	0
CmdExt[1]	Segmentation control bits (see "Message Segmentation" on page 47)
MsgData[0]	Anybus protocol error code

A. Implementation Details

A.1 Extended LED Functionality

On the Anybus CompactCom ControlNet module, only one of the two network status LEDs is available through the application interface connector (LED1[A..B]). If needed, there is the possibility to use both network status LEDs by enabling the extended LED functionality. Doing so will disable LED1[A..B] and instead use GIP[0..1] and $\overline{\text{GOP}}$[0..1] for the two network LEDs.

To enable the extended LED functionality, the application needs to set the Anybus Object Instance 1 attribute 16 (GPIO configuration) to 0x0001 during state SETUP.

See the Anybus CompactCom Hardware Design Guide for Host Interface Signals.

GPIO mode description

		Signal			
		GIP[0..1]	**$\overline{\text{GOP}}$[0..1]**	**LED1[A..B]**	**LED2[A..B]**
GPIO Configuration	**Value: 0x0000** (Default)	General purpose input	General purpose output	Network Status LED A	Module Status LED
	Value: 0x0001 (Extended LED functionality)	Network Status LED A GIP0 (red) GIP1 (green)	Network Status LED B GOP0 (red) GOP1 (green)	Disabled (set to low)	Module Status LED

Note 1: Enabling the extended LED functionality will cause both GIP[0..1] and $\overline{\text{GOP}}$[0..1] to function as outputs.

Note 2: Enabling the extended LED functionality will define both GIP[0..1] and $\overline{\text{GOP}}$[0..1] as active low. This means that LEDs will be lit when the corresponding pin is low.

Note 3: LED behavior is described in chapter 1. See "Network Status" on page 51.

A.2 SUP-Bit Definition

The supervised bit (SUP) indicates that the network participation is supervised by another network device. For ControlNet this bit is set when one or more CIP (Class 1 or Class 3) connection is opened towards the module.

A.3 Anybus State Machine

The table below describes how the Anybus State Machine relates to the ControlNet network.

State	ControlNet Specific Meaning	Notes
WAIT_PROCESS	The module will stay in this state until a Class 1 connection is opened.	-
ERROR	Class 1 connection error, invalid link configuration or dup-MAC-fail	If the error is fatal, such as dup-MAC-fail, the module will stay in this state until it's restarted.
PROCESS_ACTIVE	Error free Class 1 connection active	-
IDLE	Class 1 connection idle	Can only be set for connections consuming data.
EXCEPTION	Some kind of unexpected behavior, e.g. watchdog timeout.	The Module Status LED will turn red to indicate a major fault, and turn the Network Status LED off.

Note: The state does not change for a Class 3 connection.

B. Message Segmentation

B.1 General

Category: Advanced

The maximum message size supported by the Anybus CompactCom is 255 bytes. To provide support for longer messages (needed when using the socket interface), a segmentation protocol is used.

The segmentation protocol is implemented in the message layer and must not be confused with the fragmentation used on the serial host interface. Consult the general Anybus CompactCom 30 Software Design Guide for further information.

The module supports 1 (one) simultaneous segmented message per instance.

B.2 Command Segmentation

When a command message is segmented, the command initiator sends the same command header multiple times. For each message, the data field is exchanged with the next data segment.

Please note that some commands can not be used concurrently on the same instance, since they both need access to the segmentation buffer for that instance.

Command segmentation is used for the following commands:

- Set_Configuration_Data (see "Command Details: Set_Configuration_Data" on page 43)

Segmentation Control bits (Command)

Bit	Contents	Meaning
0	FS	Set if the current segment is the first segment
1	LS	Set if the current segment is the last segment
2	AB	Set if the segmentation shall be aborted
3...7	(reserved)	Set to 0 (zero).

Segmentation Control bits (Response)

Bit	Contents	Meaning
0...7	(reserved)	Ignore.

When issuing a segmented command, the following rules apply:

- When issuing the first segment, FS must be set.
- When issuing subsequent segments, both FS and LS must be cleared.
- When issuing the last segment, the LF-bit must be set.
- For single segment commands (i.e. size less or equal to 255 bytes), both FS and LS must be set.
- The last response message contains the actual result of the operation.
- The command initiator may at any time abort the operation by issuing a message with AB set.
- If a segmentation error is detected during transmission, an error message is returned, and the current segmentation message is discarded. Note however that this only applies to the current segment; previously transmitted segments are still valid.

B.3 Response Segmentation

When a response is segmented, the command initiator requests the next segment by sending the same command multiple times. For each response, the data field is exchanged with the next data segment.

Response segmentation is used for responses to the following commands:

- Get_Configuration_Data (see "Command Details: Get_Configuration_Data" on page 44)

Segmentation Control bits (Command)

Bit	Contents	Meaning
0	(reserved)	(set to zero)
1		
2	AB	Set if the segmentation shall be aborted
3...7	(reserved)	(set to zero)

Segmentation Control bits (Response)

Bit	Contents	Meaning
0	FS	Set if the current segment is the first segment
1	LS	Set if the current segment is the last segment
2...7	(reserved)	(set to zero)

When receiving a segmented response, the following rules apply:

- In the first segment, FS is set
- In all subsequent segment, both FS and LS are cleared
- In the last segment, LS is set
- For single segment responses (i.e. size less or equal to 255 bytes), both FS and LS are set.
- The command initiator may at any time abort the operation by issuing a message with AB set.

C. Categorization of Functionality

The objects, including attributes and services, of the Anybus CompactCom and the application are divided into three categories: basic, advanced and extended.

C.1 Basic

This category includes objects, attributes and services that are mandatory to implement or to use. They will be enough for starting up the Anybus CompactCom and sending/receiving data with the chosen network protocol. The basic functions of the industrial network are used.

Additional objects etc, that will make it possible to certify the product also belong to this category.

C.2 Extended

Use of the objects in this category extends the functionality of the application. Access is given to the more specific characteristics of the industrial network, not only the basic moving of data to and from the network. Extra value is given to the application.

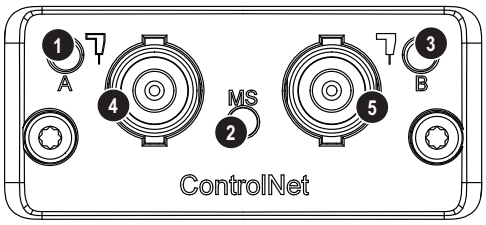
C.3 Advanced

The objects, attributes and services that belong to this group offer specialized and/or seldom used functionality. Most of the available network functionality is enabled and accessible. Access to the specification of the industrial network is normally required.

D. Technical Specification

D.1 Front View

#	Item
1	Network Status LED A
2	Module Status LED
3	Network Status LED B
4	ControlNet Connector A
5	ControlNet Connector B



Note 1: If redundancy is wanted, both connectors should be used, if not use either connector A or connector B.

Note 2: Network Status LED A and Module Status LED correspond to LED 1 and LED 2 in the instance attributes of the Anybus Object. They are available in the application interface, but the LED placement on the front does not conform to the standard Anybus CompactCom placement of LED 1 and LED 2.

See also:

- "Anybus Object (01h)" on page 31
- Anybus CompactCom HW Design Guide

Network Status

LED	State	Indication
Led A and B	Off	Not online / No power
	Flashing Red (1 Hz)	Incorrect node configuration, duplicate MAC ID etc.
	Alternating Red/Green	Self test of bus controller
	Red	Fatal event or faulty unit
Led A or B	Off	Channel is disabled
	Alternating Red/Green	Invalid link configuration
	Flashing Green (1 Hz)	Temporary errors (node will self correct) or node is not configured to go on-line
	Green	Normal operation
	Flashing Red (1 Hz)	Media fault or no other nodes on the network

Module Status

State	Indication
Off	No power
Green	Operating in normal condition, controlled by a Scanner in Run state
Flashing Green (1 Hz)	The module has not been configured or the Scanner is in Idle state
Red	Unrecoverable fault(s), EXCEPTION, fatal event
Flashing Red (1 Hz)	Recoverable fault(s), MAC ID has been changed after initialization etc.

ControlNet Connectors

These connectors provides ControlNet connectivity. If reduncancy is wanted, both connectors should be used. Otherwise either connector can be used.

D.2 Protective Earth (PE) Requirements

In order to ensure proper EMC behaviour, the module must be properly connected to protective earth via the PE pad / PE mechanism described in the general Anybus CompactCom Hardware Design Guide.

HMS Industrial Networks does not guarantee proper EMC behaviour unless these PE requirements are fulfilled.

D.3 Power Supply

Supply Voltage

The module requires a regulated 3.3V power source as specified in the general Anybus CompactCom Hardware Design Guide.

Power Consumption

The Anybus CompactCom 30 ControlNet is designed to fulfil the requirements of a Class C module. For more information about the power consumption classification used on the Anybus CompactCom platform, consult the general Anybus CompactCom Hardware Design Guide.

The current hardware design consumes up to 660 mA[1].

Note: It is strongly advised to design the power supply in the host application based on the power consumption classifications described in the general Anybus CompactCom Hardware Design Guide, and not on the exact power requirements of a single product.

D.4 Environmental Specification

Consult the Anybus CompactCom Hardware Design Guide for further information.

D.5 EMC Compliance

Consult the Anybus CompactCom Hardware Design Guide for further information.

1. Note that in line with HMS policy of continuous product development, we reserve the right to change the exact power requirements of this product without prior notification. Note however that in any case, the Anybus CompactCom 30 ControlNet will remain as a Class C module.

E. Timing & Performance

E.1 General Information

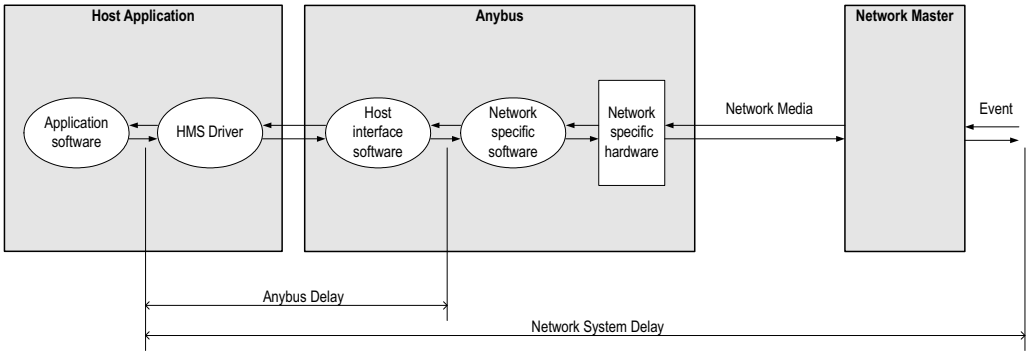
This chapter specifies timing and performance parameters that are verified and documented for the Anybus CompactCom 30 ControlNet.

The following timing aspects are measured:

Category	Parameters	Page
Startup Delay	T1, T2	Please consult the Anybus CompactCom 30 Software Design Guide, App. B.
NW_INIT Delay	T3	
Telegram Delay	T4	
Command Delay	T5	
Anybus Read Process Data Delay (Anybus Delay)	T6, T7, T8	
Anybus Write Process Data Delay (Anybus Delay)	T12, T13, T14	
Network System Read Process Data Delay (Network System Delay)	T9, T10, T11	55
Network System Write Process Data Delay (Network System Delay)	T15, T16, T17	55

E.2 Process Data

E.2.1 Overview



E.2.2 Anybus Read Process Data Delay (Anybus Delay)

The Read Process Data Delay (labelled 'Anybus delay' in the figure above) is defined as the time measured from just before new data is buffered and available to the Anybus host interface software, to when the data is available to the host application (just after the new data has been read from the driver).

Please consult the Anybus CompactCom 30 Software Design Guide, Appendix B, for more information.

E.2.3 Anybus Write Process Data Delay (Anybus Delay)

The Write Process Data Delay (labelled 'Anybus delay' in the figure) is defined as the time measured from the point the data is available from the host application (just before the data is written from the host application to the driver), to the point where the new data has been forwarded to the network buffer by the Anybus host interface software.

Please consult the Anybus CompactCom 30 Software Design Guide, Appendix B, for more information.

E.2.4 Network System Read Process Data Delay (Network System Delay)

The Network System Read Process Data Delay (labelled 'Network System Delay in the figure), is defined as the time measured from the point where an event is generated at the network master to when the corresponding data is available to the host application (just after the corresponding data has been read from the driver).

Parameter	Description	Min.	Max.	Unit.
T9	Network System Read Process Data delay, 8 ADIs (single UINT8)	2.1	4.3	ms
T10	Network System Read Process Data delay, 16 ADIs (single UINT8)	2.1	4.2	ms
T11	Network System Read Process Data delay, 32 ADIs (single UINT8)	2.1	4.3	ms

Conditions:

Parameter	Conditions
Application CPU	-
Timer system call interval	1 ms
Driver call interval	0.2... 0.3 ms
No.of ADIs (single UINT8) mapped to Process Data in each direction.	8, 16 and 32
Communication	Parallel
Telegram types during measurement period	Process Data only
Bus load, no. of nodes, baud rate etc.	Normal

E.2.5 Network System Write Process Data Delay (Network System Delay)

The Network System Write Process Data Delay (labelled 'Network System Delay in the figure), is defined as the time measured from the time after the new data is available from the host application (just before the data is written to the driver) to when this data generates a corresponding event at the network master.

Parameter	Description	Min.	Max.	Unit.
T15	Network System Write Process Data delay, 8 ADIs (single UINT8)	2.1	4.3	ms
T16	Network System Write Process Data delay, 16 ADIs (single UINT8)	2.1	4.2	ms
T17	Network System Write Process Data delay, 32 ADIs (single UINT8)	2.1	4.3	ms

Conditions: as in "Network System Read Process Data Delay (Network System Delay)", p. 55.