Network Interface Appendix

Anybus® CompactCom 30 DeviceNet

HMS
Connecting Devices™

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

Important User Information

This document is intended to provide a good understanding of the functionality offered by DeviceNet. The document only describes the features that are specific to the Anybus CompactCom 30 DeviceNet. 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 DeviceNet-specific functionality may require in-depth knowledge in DeviceNet networking internals and/or information from the official DeviceNet specifications. In such cases, the people responsible for the implementation of this product should either obtain the DeviceNet 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 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.

Table of Contents

Appendix A Categorization of Functionality

Appendix B Implementation Details

Appendix C CIP Request Forwarding

Appendix D Technical Specification

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 Software Design Guide	HMS
Anybus CompactCom Hardware Design Guide	HMS
Anybus CompactCom Software Driver User Guide	HMS
DeviceNet Specification	ODVA
Common Industrial Protocol (CIP) specification	ODVA

P.2 Document History

Summary of Recent Changes (2.21... 2.30)

Change	Page(s)
Added attribute #13, Enable Quick Connect to DeviceNet Host Object	44
Added attribute #10, Quick Connect to CIP DeviceNet Object	22
Added Quick Connect to features	8
Added wiring requirements for brick connector	11
Added note on when CIP request forwarding is applicable	51

Revision List

Revision	Date	Author(s)	Chapter(s)	Description
1.00	2005-09-15	PeP	-	First official release
1.01	2005-10-19	PeP	-	Misc. minor corrections
1.05	2006-05-03	PeP	-	Misc. visual and structural updates
1.06	2007-04-26	PeP	-	Minor update
2.00	2010-04-14	KeL	All	Change to new concept
2.01	2010-11-12	KeL	P, 6	Minor updates
2.02	2011-12-05	KeL	2	Minor update
2.03	2012-04-13	KeL	2	Minor update
2.10	2012-10-19	KeL, KaD	1, 6	M12 connectors added and minor update
2.20	2013-02-18	KeL	1, 3	Updates for brick version, minor update
2.21	2013-05-17	KeL	1	MInor update
2.30	2014-04-24	KeL	1, 4, 6, C	Misc. 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 DeviceNet

1.1 General

The Anybus CompactCom 30 DeviceNet communication module provides instant DeviceNet 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 30 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

- DeviceNet connector or M12 connectors
- Brick version
- CIP Parameter Object Support
- Explicit messaging
- UCMM Capable
- Bit-strobed I/O
- Change-of-state / Cyclic I/O
- Polled I/O
- Expansion possibilities via CIP forwarding
- Customizable Identity object
- Automatic Baudrate Detection
- Quick Connect supported

1.3 Front View

#	Item
1	Network Status LED
2	Module status LED
3	DeviceNet Connector
4	M12 Female Connector
5	M12 Male Connector
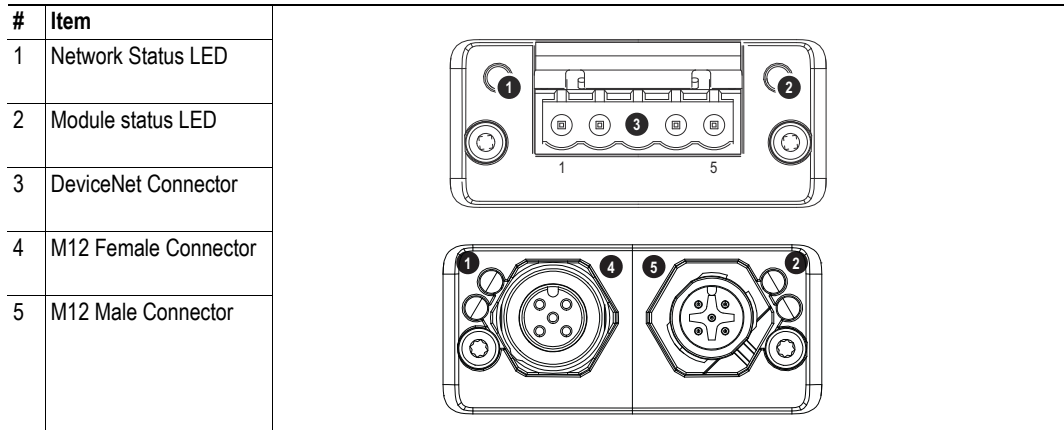
Network Status

State	Indication
Off	Not online / No power
Green	On-line, one or more connections are established
Flashing Green (1 Hz)	On-line, no connections established
Red	Critical link failure
Flashing Red (1 Hz)	One or more connections timed-out
Alternating Red/Green	Self test

Module Status

State	Indication
Off	No power
Green	Operating in normal condition
Flashing Green (1 Hz)	Missing or incomplete configuration, device needs commissioning
Red	Unrecoverable Fault(s)
Flashing Red (1 Hz)	Recoverable Fault(s)
Alternating Red/Green	Self test

DeviceNet Connector

This connector provides DeviceNet connectivity.
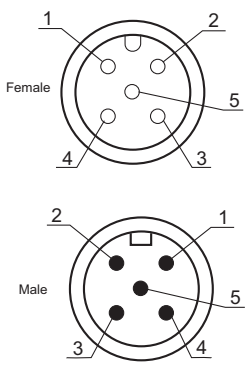
Pin	Signal	Description
1	V-	Negative bus supply voltage[a]
2	CAN_L	CAN low bus line
3	SHIELD	Cable shield
4	CAN_H	CAN high bus line
5	V+	Positive bus supply voltage[a]

a. DeviceNet bus power. For more information, see D-53 "Technical Specification".

M12 Connectors, Code A

The female M12 connector is used when modules are used in a daisy-chain topology.

Pin	Name	Description
1	Drain	Shield
2	V+	Positive voltage bus power. 11 - 25 VDC[a]
3	V-	Ground bus power
4	CAN_H	CAN high
5	CAN_L	CAN low

a. DeviceNet bus power. For more information, see D-53 "Technical Specification".

1.4 Network Connector, Brick Version

The Anybus CompactCom 30 DeviceNet can also be acquired in a brick version, without a fieldbus connector, but instead a pin connector to the carrier board (the host device). The concept and assembly are described in the Anybus CompactCom Mounting Kit Appendix (Doc. Id. HMSI-168-30).T

Pin no.	Signal	Port
1	Shield	Input
2	V+	
3	CAN_H	
4	Shield	
5	CAN_L	
6	V-	
7	Drain	
8	Shield	
11	Shield	Output
12	Drain	
13	V-	
14	Shield	
15	CAN_L	
16	CAN_H	
17	V+	
18	Shield	
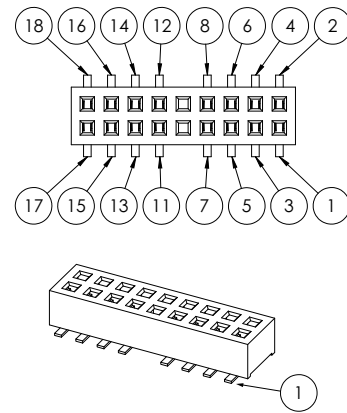
Wiring requirements

- Route CAN_H and CAN_L side by side with similar lengths. Use Shield as a reference plane for this signal pair and for V+ and V-.
- If the device is designed with one fieldbus connecoter e.g. a screw terminal, it is recommended to connect CAN-H and CAN-L only to pins 3 and 5 of the brick header.
- If the device is designed with two fieldbus connectors, e.g. M12 connectors, it is recommended to use brick header pins 3 and 5 for one of the connectors, while pins 15 and 16 are used for the other connector. Which one of the connectors that is used as input or output does not matter. Route V+and V- with thic traces directly between the connectors.

Pin no. 1
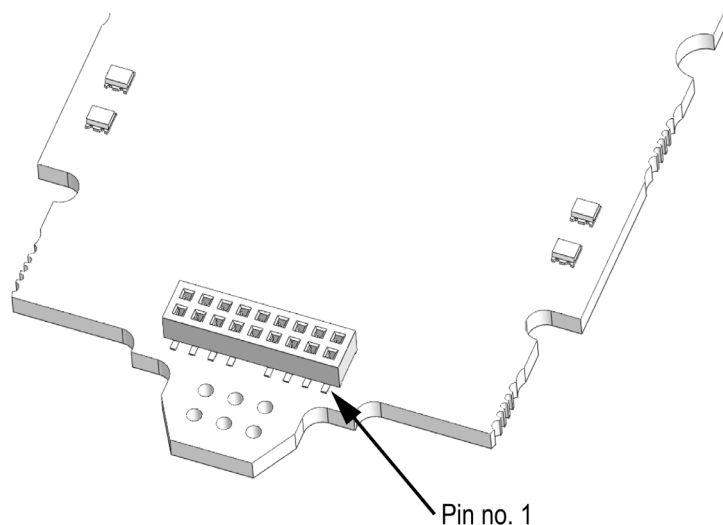
2. Tutorial

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 DeviceNet networks.

2.2 Fieldbus Conformance Notes

- The Anybus CompactCom 30 DeviceNet 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 DeviceNet Host Object must be customized.
- 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 30 DeviceNet 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 30 DeviceNet 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 DeviceNet Host Object (FCh). Therefore, the attributes listed below shall be implemented and proper values returned.

Object/Instance	Attribute	Explanation	Default	Customer sample	Comment
DeviceNet Object (FCh), 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.
DeviceNet Object (FCh), Instance 1	#2, Device Type[a]	With this attribute you set the Device Type of the device.	0000h	002Bh[a] (Generic Device (keyable))	
DeviceNet Object (FCh), Instance 1	#3, Product Code	With this attribute you set the Product Code of the device	0064h	2222h	
DeviceNet Object (FCh), Instance 1	#4, Revision	With this attribute you set the Revision of the device.		1.1	
DeviceNet Object (FCh), 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.
DeviceNet Object (FCh), Instance 1	#6, Product Name	With this attribute you set the Product Name of the device.	Anybus-CC DeviceNet	"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 DeviceNet 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

Generally, no additional network support code needs to be written in order to support the Anybus CompactCom DeviceNet. However, due to the nature of the DeviceNet networking system, certain restrictions must be taken into account:

- Certain functionality in the module requires that the command 'Get_Instance_Number_By_Order' (Application Data Object, FEh) is implemented in the host application.
- Up to 5 diagnostic instances (See "Diagnostic Object (02h)" on page 39) can be created by the host application during normal conditions. An additional 6th instance may be created in event of a major fault.

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

See also...

- "Diagnostic Object (02h)" on page 39 (Anybus Module Object)
- Anybus CompactCom Software Design Guide, 'Application Data Object (FEh)'

3.1.2 Electronic Data Sheet (EDS)

Since the module implements the Parameter Object, it is possible for configuration tools such as RSNet-Worx 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 15
- "Parameter Object (0Fh)" on page 31 (CIP-object)
- Anybus CompactCom 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: 0062h (Anybus CompactCom DeviceNet)
- Product Name: 'Anybus-CC DeviceNet'

It is possible to customize the identity of the module by implementing the DeviceNet Host Object. Furthermore, it is possible to re-route requests to unimplemented 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 DeviceNet 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 DeviceNet Host Object.
- Implement the required CIP-objects in the host application.

See also...

- "Identity Object (01h)" on page 19 (CIP-object)
- "DeviceNet Host Object (FCh)" on page 44 (Host Application Object)
- "CIP Request Forwarding" on page 51

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...

- **Baud rate**

  See also...

  - "Instance Attributes (Instance #2, 'Baud rate')" on page 42

- **Mac ID**

  See also...

  - "Instance Attributes (Instance #1, 'Device Address')" on page 42

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 19 (CIP-object)
- "Network Configuration Object (04h)" on page 41 (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 19 (CIP-object)
- "Diagnostic Object (02h)" on page 39 (Anybus Module Object)

3.5 Data Exchange

3.5.1 Application Data (ADIs)

ADIs are represented on DeviceNet 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 31 (CIP-object)
- • "ADI Object (A2h)" on page 34 (CIP-object)

3.5.2 Process Data

Process Data is represented on DeviceNet 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 24 (CIP-object)
- • "Connection Object (05h)" on page 26 (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:

- "Identity Object (01h)" on page 19
- "Message Router (02h)" on page 21
- "DeviceNet Object (03h)" on page 22
- "Assembly Object (04h)" on page 24
- "Connection Object (05h)" on page 26
- "Parameter Object (0Fh)" on page 31
- "Acknowledge Handler Object (2Bh)" on page 33

Vendor Specific Objects:

- "ADI Object (A2h)" on page 34

It is possible to implement additional CIP-objects in the host application using the CIP forwarding functionality, see "DeviceNet Host Object (FCh)" on page 44 and "CIP Request Forwarding" on page 51.

4.2 Identity Object (01h)

Category

Extended

Object Description

-

Supported Services

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

Class Attributes

#	Access	Name	Type	Comments
1	Get	Revision	UINT	0001h

Instance #1 Attributes

Extended

#	Access	Name	Type	Comments
1	Get	Vendor ID	UINT	005Ah (HMS Industrial Networks AB[a])
2	Get	Device Type	UINT	0000h (Generic Device[a])
3	Get	Product Code	UINT	0062h (Anybus CompactCom DeviceNet[a])
4	Get	Revision	Struct of: {USINT, USINT}	Major and minor firmware revision[a]
5	Get	Status	WORD	See "Device Status" on page 20
6	Get	Serial Number	UDINT	Assigned by HMS[a]
7	Get	Product Name	SHORT_STRING	"Anybus-CC DeviceNet" (Name of product[a])
11	Set	Active language	Struct of: {USINT, USINT, USINT}	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}	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.

a. Can be customized by implementing the DeviceNet Host Object, see "DeviceNet Host Object (FCh)" on page 44

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 and/or the module's NV storage is changed from default.

b. See "Diagnostic Object (02h)" on page 39.

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 DeviceNet:

- **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 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 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 DeviceNet Object (03h)

Category

Extended

Object Description

-

Supported Services

Class	Get Attribute Single
Instance:	Get Attribute Single
	Set Attribute Single
	Allocate Master/Slave Connection Set (4Bh)
	Release Group 2 Identifier Set (4Ch)

Class Attributes

#	Name	Access	Type	Comments
1	Revision	Get	UINT	0002h

Instance #1 Attributes

Extended

#	Name	Access	Type	Comments
1	MAC ID[a]	Get/Set	USINT	Currently used MacID
2	Baud Rate[ab]	Get/Set	USINT	Value:Baud rate: 0 125 kbps 1 250 kbps 2 500 kbps
3	BOI	Get/Set	BOOL	False
4	Bus off Counter	Get/Set	USINT	00h
5	Allocation Information	Get	Struct of: BYTE USINT	 Allocation choice byte MAC ID of master
6	MAC ID Switch changed[a]	Get	BOOL	Indicates if the MacID has changed since startup Value:Meaning True Changed False No change
7	Baud rate Switch changed[a]	Get	BOOL	Indicates if the baudrate has changed since startup Value:Meaning True Changed False No change
8	MAC ID Switch value[a]	Get	USINT	Actual value of node address switches
9	Baud rate Switch value[a]	Get	USINT	Actual value of baud rate switches
10	Quick Connect[c]	Get/Set	Bool	Enables/Disables the Quick Connect feature. Disabled by default Value:Meaning True Enable False Disable
100	Disable auto baud	Set	BOOL	Value:Meaning True Disable auto baud False Enable auto baud This setting is stored in NV memory.

a. Implementation of attributes 6 to 9 are conditional as well as access right for attributes 1 and 2. For further information, see "Communication Settings" on page 16.

b. Setting this attribute will also affect attribute #100 (Disable auto baud).

c. Enabled if attribute #13 ('Enable Quick Connect') in the DeviceNet Host Object (FCh) is set to true, see "DeviceNet Host Object (FCh)" on page 44.

4.5 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 17
- "DeviceNet Host Object (FCh)" on page 44

Supported Services

Class   -

Instance: Get Attribute Single
                     Set Attribute Single

Class Attributes

-

Instance 64h Attributes (Producing Instance)

Extended

The instance number for this instance can be changed by implementing the corresponding attribute in the DeviceNet 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 17
- "DeviceNet Host Object (FCh)" on page 44

Instance 96h Attributes (Consuming Instance)

Extended

The instance number for this instance can be changed by implementing the corresponding attribute in the DeviceNet 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 17
- "DeviceNet Host Object (FCh)" on page 44

4.6 Connection Object (05h)

Category

Extended

Object Description

-

Supported Services

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

Class Attributes

#	Name	Access	Type	Comments
1	Revision	Get	UINT	0001h

Instances #1, #10... #14 (Explicit messaging)

Extended

#	Name	Access	Type	Comments
1	State	Get	USINT	<u>Value:State:</u> 0 Non existent 1 Configuring 2 Waiting for connection ID 3 Established 4 Time out 5 Deferred Delete
2	Instance type	Get	USINT	0000h (Explicit messaging connection)
3	Transport Class trigger	Get	BYTE	83h (Server, Transport class 3)
4	Produced connection ID	Get	UINT	-
5	Consumed connection ID	Get	UINT	-
6	Initial Comm Characteristics	Get	BYTE	The message group over which the communication occurs: <u>Value:Message Group</u> 21 Instance #1 33 Instances #10... #14
7	Produced Connection Size	Get	UINT	262 bytes
8	Consumed Connection Size	Get	UINT	262 bytes
9	Expected Packet Rate	Get/Set	UINT	2500ms
12	Watchdog timeout action	Get/Set	USINT	<u>Value:Action:</u> 0001h Auto delete (default) 0003h Deferred delete
13	Produced Connection path length	Get	UINT	0000h (No connection path)
14	Produced Connection path	Get	EPATH	-
15	Consumed Connection path length	Get	UINT	0000h (No connection path)
16	Consumed Connection path	Get	EPATH	-
17	Production Inhibit Time	Get	UINT	0000h
18	Connection Timeout Multiplier	Get/Set	UINT	0000h

Instance #2 (Poll or "COS/Cyclic consuming")

Extended

#	Name	Access	Type	Comments
1	State	Get	USINT	Value:State: 0 Non existent 1 Configuring 2 Waiting for connection ID 3 Established 4 Time out
2	Instance type	Get	USINT	0001h (I/O Connection)
3	Transport Class trigger	Get	BYTE	Value:Meaning: 82h Server, Polled, Class 2 80h Server, COS/Cyclic, Class 0, No Ack. 82h Server, COS/Cyclic, Class 2, Ack.
4	Produced connection ID	Get	UINT	Value:Meaning: FFFFh Not consuming (COS/Cyclic) Other CAN ID for transmission
5	Consumed connection ID	Get	UINT	-
6	Initial Comm Characteristics	Get	BYTE	Value:Meaning: 01h Polled - Produces over message group 1 - Consumes over message group 2 F1h COS/Cyclic, No Ack - Consumes only over message group 2 01h COS/Cyclic, Ack - Produces over message group 1 (Ack) - Consumes over message group 2
7	Produced Connection Size	Get	UINT	Value:Meaning: 0000h COS/Cyclic Other Size of Write Process Data (Polled)
8	Consumed Connection Size	Get	UINT	Size of Read Process Data
9	Expected Packet Rate	Get/Set	UINT	-
12	Watchdog timeout action	Get	USINT	0000h (Transition to the timed out state)
13	Produced Connection path length	Get	UINT	0000h (COS/Cyclic) 0007h (Polled)
14	Produced Connection path	Get	EPATH	No value (COS/Cyclic) 20 04 25 nn nn 30 03h (Polled)
15	Consumed Connection path length	Get	UINT	0007h
16	Consumed Connection path	Get	EPATH	20 04 25 nn nn 30 03h
17	Production Inhibit Time	Get	UINT	0000h
18	Connection Timeout Multiplier	Get/Set	UINT	0000h

Instance #3 (Bit-strobe)

Extended

#	Name	Access	Type	Comments
1	State	Get	USINT	Value:State: 0 Non existent 1 Configuring 2 Waiting for connection ID 3 Established 4 Time out
2	Instance type	Get	USINT	0001h (I/O Connection)
3	Transport Class trigger	Get	BYTE	82h (Transport class & Trigger Server, Cyclic, Class 2)
4	Produced connection ID	Get	UINT	-
5	Consumed connection ID	Get	UINT	-
6	Initial Comm Characteristics	Get	BYTE	Produces over message group 1 Consumes over message group 2
7	Produced Connection Size	Get	UINT	Size of produced data on this connection. **Max of: 8 bytes, Mapped Process data**
8	Consumed Connection Size	Get	UINT	0008h
9	Expected Packet Rate	Get/Set	UINT	-
12	Watchdog timeout action	Get	USINT	0000h (Transition to the timed out state)
13	Produced Connection path length	Get	UINT	0007h
14	Produced Connection path	Get	EPATH	20 04 25 nn nn 30 03h
15	Consumed Connection path length	Get	UINT	0007h
16	Consumed Connection path	Get	EPATH	20 04 25 nn nn 30 03h
17	Production Inhibit Time	Get	UINT	0000h
18	Connection Timeout Multiplier	Get/Set	UINT	0000h

Instance #4 (COS/Cyclic producing)

Extended

#	Name	Access	Type	Value
1	State	Get	USINT	Value:State: 0 Non existent 1 Configuring 2 Waiting for connection ID 3 Established 4 Time out
2	Instance type	Get	USINT	0001h (I/O Connection)
3	Transport Class trigger	Get	BYTE	Value:Meaning: 00h Client, Cyclic, Class 0 (No Ack.) 10h Client, COS, Class 0 (No Ack.) 02h Client, Cyclic, Class 2 (Ack.) 12h Client, COS, Class 2 (Ack.)
4	Produced connection ID	Get	UINT	CAN ID for transmission
5	Consumed connection ID	Get	UINT	Value:Meaning: FFFFh Not acknowledged Other CAN ID for reception (Ack.)
6	Initial Comm Characteristics	Get	BYTE	Value:Meaning: 0Fh Producing only over message group 1 (No Ack.) 01h Produces over message group 1 Consumes over message group 2 (Ack.)
7	Produced Connection Size	Get	UINT	Size of produced data on this connection.
8	Consumed Connection Size	Get	UINT	0000h (Consumes 0 bytes on this connection)
9	Expected Packet Rate	Get/Set	UINT	Timing associated with this connection.
12	Watchdog timeout action	Get	USINT	0000h (Transition to the timed out state)
13	Produced Connection path length	Get	UINT	0007h
14	Produced Connection path	Get	EPATH	20 04 25 nn nn 30 03h
15	Consumed Connection path length	Get	UINT	0000h (No ack.) 0005h (Acknowledged)
16	Consumed Connection path	Get	EPATH	No value (No ack.) 20 2B 25 01 00h (Acknowledged)
17	Production Inhibit Time	Get/Set	UINT	0000h
18	Connection Timeout Multiplier	Get/Set	UINT	0000h

4.7 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 DeviceNet 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 4-32 "Default Values".

See also...

- "Default Values" on page 32
- "ADI Object (A2h)" on page 34 (CIP Object)
- "DeviceNet Host Object (FCh)" on page 44 (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 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	20 A2 25 nn nn 30 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)
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
11	Maximum value	Get	(Data Type)	Maximum value of parameter
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
12	Default value	N/A	0

4.8 Acknowledge Handler Object (2Bh)

Category

Extended

Object Description

-

Supported Services

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

Class Attributes

#	Name	Access	Type	Value
1	Revision	Get	UINT	0001h

Instances Attributes (01h)

Extended

#	Name	Access	Type	Value
1	Acknowledge Timer	Get/Set	UINT	16ms
2	Retry Limit	Get/Set	USINT	01h
3	Producing Connection Instance	Get	UINT	04h

4.9 ADI Object (A2h)

Category

Extended

Object Description

This object maps instances in the Application Data Object to DeviceNet. 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.

See also...

- Application Data Object (see Anybus CompactCom Software Design Guide)
- "Parameter Object (0Fh)" on page 31 (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 Software Design Guide for further information.

Instances Attributes

Extended

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

#	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
5	Value[a]	Get/Set	Determined by attribute #2	Instance value
6	Max value[a]	Get		The maximum permitted parameter value.
7	Min value[a]	Get		The minimum permitted parameter value.
8	Default value[a]	Get		The default parameter value.

a. Converted to/from CIP standard by the module

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 DeviceNet.

The following Anybus Module Objects are implemented:

- • "Anybus Object (01h)" on page 37
- • "Diagnostic Object (02h)" on page 39
- • "Network Object (03h)" on page 40
- • "Network Configuration Object (04h)" on page 41

5.2 Anybus Object (01h)

Category

Basic

Object Description

This object groups common Anybus information, and is described thoroughly in the general Anybus CompactCom Software Design Guide.

Supported Commands

Object:	Get_Attribute
Instance:	Get_Attribute
	Set_Attribute
	Get_Enum_String

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1	Name	Get	Array of CHAR	"Anybus"
2	Revision	Get	UINT8	04h
3	Number of instances	Get	UINT16	0001h
4	Highest instance no.	Get	UINT16	0001h

Instance Attributes (Instance #1)

Basic

#	Name	Access	Type	Value
1	Module type	Get	UINT16	0401h (Standard Anybus CompactCom)

2	Firmware version	Get	struct of:
			  UINT8 Major
			  UINT8 Minor
			  UINT8 Build
				(see Anybus CompactCom Software Design Guide)

3	Serial number	Get	UINT32	
4	Application watchdog timeout	Get/Set	UINT16	
5	Setup complete	Get/Set	BOOL	
6	Exception Code	Get	ENUM	

8	Error counters	Get	struct of:
			  UINT16 DC
			  UINT16 DR
			  UINT16 SE	

9	Language	Get/Set	ENUM	
10	Provider ID	Get	UINT16	
11	Provide specific info	Get/Set	UINT16	

#	Name	Access	Type	Value
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	LED status	Get	UINT8	(see Anybus CompactCom Software Design Guide)
14	(reserved)	Get	UINT8	
15	Auxiliary Bit	Get/Set	UINT8	

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 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 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 Software Design Guide
2	Event Code	Get	UINT8	

In the Anybus CompactCom DeviceNet, 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 16
- "Identity Object (01h)" on page 19 (CIP-object)

5.4 Network Object (03h)

Category

Basic

Object Description

For more information regarding this object, consult the general Anybus CompactCom 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	0025h
2	Network type string	Get	Array of CHAR	'DeviceNet'
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]

a. Consult the general Anybus CompactCom 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 16
- "Identity Object (01h)" on page 19 (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	0002h
4	Highest instance no.	Get	UINT16	0002h

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

Basic

#	Name	Access	Type	Description
1	Name[a]	Get	Array of CHAR	'Device 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	MacID, stored in non volatile memory

a. Multilingual, see "Multilingual Strings" on page 42.

b. A 'Get' command always returns the actual value. If an invalid value is assigned to this attribute (i.e. using a 'Set' command), the module will accept MacID configuration via the network (unless disabled in the DeviceNet Host Object - in such case, the module will enter communication fault state at start up).

Instance Attributes (Instance #2, 'Baud rate')

Basic

#	Name	Access	Type	Description
1	Name[a]	Get	Array of CHAR	'Baud rate'
2	Data type	Get	UINT8	08h (ENUM)
3	Number of elements	Get	UINT8	01h (one element)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value[b]	Get/Set	ENUM	Value:Enum. String:Meaning: 00h '125kbps' 125kbps 01h '250kbps' 250kbps 02h '500kbps' 500kbps 03h 'Autobaud' Autobaud (default)

a. Multilingual, see "Multilingual Strings" on page 42.

b. A 'Get' command always returns the actual value. If an invalid value is assigned to this attribute (i.e. using a 'Set' command), the module will accept baudrate configuration via the network (unless disabled in the DeviceNet Host Object - in such case, the module will enter communication fault state at start up).

Multilingual Strings

The instance names in this object are multi-lingual, and are translated based on the current language settings as follows:

Instance	English	German	Spanish	Italian	French
1	Node address	Geräteadresse	Direcc nodo	Indirizzo	Adresse
2	Data rate	Datenrate	Veloc transf	velocità dati	Vitesse

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 DeviceNet implementation.

Standard Objects:

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

Network Specific Objects:

-

6.2 DeviceNet Host Object (FCh)

Category

Basic, extended, advanced

Object Description

This object implements DeviceNet specific settings in the host application. It is also used when implementing DeviceNet 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 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 "CIP Request Forwarding" on page 51)

Instance: -

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1	Name	Get	Array of CHAR	'DeviceNet'
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 DeviceNet Identity Object (CIP).
2	Device Type	Get	UINT16	0000h	
3	Product Code	Get	UINT16	0062h	
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 DeviceNet'	

Extended

#	Name	Access	Type	Default Value	Comment
7	Producing Instance No.	Get	UINT16	0064h	See also... - "Instance 64h Attributes (Producing Instance)" on page 24 (CIP-instance)
8	Consuming Instance No.	Get	UINT16	0096h	See also... - "Instance 96h Attributes (Consuming Instance)" on page 25 (CIP-instance)
11	Enable CIP forwarding	Get	BOOL	False	Value:Meaning: True Enable CIP forwarding False Disable CIP forwarding See also... - "Command Details: Process_CIP_Message_Request" on page 47 - "CIP Request Forwarding" on page 51.
13	Enable Quick Connect	Get	BOOL	False	Value:Meaning: True Enable Quick Connect False Disable Quick Connect See also... - "DeviceNet Object (03h)" on page 22

Advanced

#	Name	Access	Type	Default Value	Comment
9	Enable Address from Net	Get	BOOL	True	Value:Meaning: True Can be set from network False Cannot be set from network See also... - "Identity Object (01h)" on page 19 (CIP-object)
10	Enable Baud rate from Net	Get	BOOL	True	Value:Meaning: True Can be set from network False Cannot be set from network See also... - "Identity Object (01h)" on page 19 (CIP-object)
12	Enable Parameter Object	Get	BOOL	True	Value:Meaning: True Enable CIP Parameter Object False Disable CIP Parameter Object See also... - "Parameter Object (0Fh)" on page 31 (CIP-object)

6.2.1 Command Details: Process_CIP_Message_Request

Category

Extended

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. 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: This command is similar - but <u>not</u> identical - to the 'Process_CIP_Message_Request'-command in the Anybus CompactCom EtherNet/IP.

See also...

- "Device Customization" on page 15
- "CIP Request Forwarding" on page 51

A. 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.

A.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.

A.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.

A.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.

B. Implementation Details

B.1 DeviceNet Implementation

Predefined Connection Set

The module acts as a Group 2 server and supports the Predefined Master/Slave Connection Set.

- **COS Connection**

  When the master allocates this connection type, the module transmits the Process Data it changes. An inhibit time can be set to prevent the module from sending too often.

  The module supports up to 256 bytes in each direction for this type of connection. The size of the connection is checked against the number of bytes mapped as Process Data.

- **Cyclic Connection**

  When the master allocates this connection type, the module cyclically transmits the Process Data at the configured interval.

  The module supports up to 256 bytes in each direction for this type of connection.

- **Bit Strobe Connection**

  When the master allocates this connection type, the module transmits data when the bit strobe message is received. The module only uses the input bit if no other I/O connections have been configured, and produces up to 8 bytes.

- **Polled Connection**

  When the master allocates this connection type, the module transmits the Process Data data when a poll command is received.

  The module supports up to 256 bytes in each direction for this type of connection.

- **Explicit Connection**

  The predefined explicit connection has a buffer of 262 bytes.

- **Idle/Running**

  The module is considered to be in Idle mode when not receiving any DeviceNet telegrams, or when receiving DeviceNet telegrams with no data. In other cases, the module is considered to be in Run mode.

  This affects the Anybus State machine as describe in B-50 "Anybus State Machine".

Unconnected Message Server (UCMM)

The module is a UCMM capable device, and supports the Unconnected Explicit Message Request port, Group3, Message ID=6.

- **Explicit Message Server**

  The module supports up to 5 simultaneous explicit message connections.

B.2 SUP-Bit Definition

The supervised bit (SUP) indicates that the network participation is supervised by another network device. For DeviceNet this bit is set when the connection object has a connection.

B.3 Anybus State Machine

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

State	DeviceNet Specific Meaning	Notes
WAIT_PROCESS	The module will stay in this state until a Class 0 connection is opened.	(Not set for explicit connections.)
ERROR	Class 0 connection error, bus off event detected or dup-MAC-fail	If the error is fatal, such, such as dup-MAC-fail or bus off, the module will stay in this state until it's restarted.
PROCESS_ACTIVE	Error free Class 0 connection active	-
IDLE	Class 0 connection idle	Can only be set for connections consuming data.
EXCEPTION	Some kind of unexpected behaviour, e.g. watchdog timeout.	The Module Status LED will turn red to indicate a major fault, and turn the Network Status LED off.

C. CIP Request Forwarding

If CIP request forwarding is enabled (DeviceNet Host Object, Instance 1, Attribute 11), all network requests to unknown CIP objects or unknown assembly object instances will be forwarded to the host application. For this purpose, the DeviceNet Host Object implements a command called Process_CIP_Message_Request (Command code 10h), which is used to tunnel CIP requests to the host application.

Note: CIP request forwarding is only relevant for explicit messages. It is not applicable to the messages that carry the cyclic/process data.

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.

- **Command Message Layout**

  This message will be sent by the module to the host application upon receiving an unknown CIP request from the network.

Field	Contents								Notes
	b7	b6	b5	b4	b3	b2	b1	b0	
Source ID	(Source ID)								Selected by the module
Dest. Object	FCh								Destination Object = DeviceNet Host Object
Dest. Instance (lsb)	00h								Destination Instance = Object Instance
Dest. Instance (msb)	00h								
(command/error)	0								This message is not an error message
(command/response)		1							This message is a command
Command number			10h						Process_CIP_Object_Request
Message Data Size	Length of CIP request								-
CmdExt[0]	CIP Service Code								CIP service code from original CIP request
CmdExt[1]									(reserved, ignore)
MsgData[0]	Requested CIP Class no.								(Low byte)
MsgData[1]									(High byte)
MsgData[2]	Requested CIP Instance no.								(Low byte)
MsgData[3]									(High byte)
MsgData[4...n]	CIP Data								Data associated with the CIP request

- **Host Application Response Message Layout (Successful)**

  If the host application recognized the CIP request, i.e. if the CIP object in question is implemented in the host application, the following response shall be sent to the module.

Field	Contents								Notes
	b7	b6	b5	b4	b3	b2	b1	b0	
Source ID	(Source ID)								(Selected by the module)
Dest. Object	FCh								Object = DeviceNet Host Object
Dest. Instance (lsb)	00h								Instance = Object
Dest. Instance (msb)	00h								
(command/error)	0								This message is not an error message
(command/response)		0							This message is a response
Command number			10h						Process_CIP_Object_Request
Message Data Size	Length of response data								-
CmdExt[0]	CIP Service Code (with reply bit set)								-
CmdExt[1]	00h								(not used, set to zero)
MsgData[0...n]	Response data								-

- **Host Application Response Message Layout (Unsuccessful)**

If the host application did not recognize the CIP request, i.e. the CIP object in question is not implemented in the host application, the following response shall be sent to the module.

Field	Contents								Notes
	b7	b6	b5	b4	b3	b2	b1	b0	
Source ID	(Source ID)								(Selected by the module)
Dest. Object	FCh								Object = DeviceNet Host Object
Dest. Instance (lsb)	00h								Instance = Object
Dest. Instance (msb)	00h								
(command/error)	0								This message is not an ABCC error message[a]
(command/response)		0							This message is a response
Command number			10h						Process_CIP_Object_Request
Message Data Size	02h								2 bytes of message data
CmdExt[0]	94h								CIP error service code with reply bit set
CmdExt[1]	00h								(not used, set to zero)
MsgData[0]	CIP General status code								-
MsgData[1]	Optional additional status								(FFh if no additional status)

a. If this bit is set (1), an Anybus CompactCom error has occurred and an ABCC error code is returned.

D. Technical Specification

D.1 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.2 Power Supply

Supply Voltage

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

Power Consumption

The Anybus CompactCom DeviceNet is designed to fulfil the requirements of a Class A 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 65 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.3 DeviceNet Power Supply

The total number of units that can be connected to the DeviceNet bus is limited by the maximum current that the power supply can deliver to the bus. Maximum current consumption per unit is specified in the DeviceNet specification to 750 mA. If e.g. the supply can deliver 9 A and all units consume maximum current, the maximum numbers of units allowed on the bus are 12 (12x750 mA = 9A).

The Anybus CompactCom 30 DeviceNet module accepts 11 - 25 V on the industrial network side of the module. Maximum current consumption at 11 - 25 V is 36 - 38 mA/module.

Note: If the M12 version of the module is used, the current on the DeviceNet bus has to be limited to a maximum of 3 A, which is the max. current allowed for the M12 connectors.

D.4 Environmental Specification

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 DeviceNet will remain as a Class A module.

D.5 EMC Compliance

Consult the Anybus CompactCom Hardware Design Guide for further information.

# E. Timing & Performance

E.1 General Information

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

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	57
Network System Write Process Data Delay (Network System Delay)	T15, T16, T17	57

E.2 Process Data

E.2.1 Overview
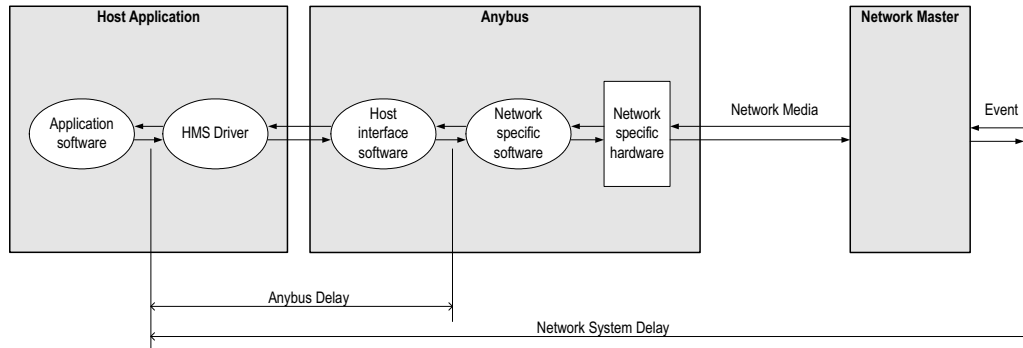
E.2.2 Anybus Read Process Data Delay (Anybus Delay)

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	Avg.	Max.	Unit.
T9	Network System Read Process Data delay, 8 ADIs (single UINT8)	8.2	14	ms
T10	Network System Read Process Data delay, 16 ADIs (single UINT8)	9.4	16	ms
T11	Network System Read Process Data delay, 32 ADIs (single UINT8)	10	16.2	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	Avg.	Max.	Unit.
T15	Network System Write Process Data delay, 8 ADIs (single UINT8)	8.2	14	ms
T16	Network System Write Process Data delay, 16 ADIs (single UINT8)	9.4	16	ms
T17	Network System Write Process Data delay, 32 ADIs (single UINT8)	10	16.2	ms

Conditions: as in "Network System Read Process Data Delay (Network System Delay)" on page 57.