Network Guide

Anybus® CompactCom 40 DeviceNet™

Doc.Id. HMSI-27-264
Rev. 1.20

Connecting Devices™

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

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

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

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 40 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 information in this manual should normally be sufficient for implementing a design. However, if advanced DeviceNet specific functionality is to be used, in-depth knowledge of DeviceNet networking internals and/or information from the official DeviceNet specifications, may be required. 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.

Oct 2015 Doc Id HMSI-27-264

Table of Contents

Preface About This Document

Related Documents 5
Document History 5
Conventions & Terminology 6
Support 6

Chapter 1 About the Anybus CompactCom 40 DeviceNet

General 7
Features 7

Chapter 2 Basic Operation

General Information 8
Software Requirements 8
Electronic Data Sheet (EDS) 8
Device Customization 9
Modular Device Functionality 9
Quick Connect 10
Communication Settings 11
Setting Baud Rate 12
Setting Node Address 12
Diagnostics 13
Data Exchange 14
Application Data (ADIs) 14
Process Data 14
Translation of Data Types 15

Chapter 3 CIP Objects

General Information 16
Identity Object (01h) 17
Message Router (02h) 20
DeviceNet Object (03h) 21
Assembly Object (04h) 23
Connection Object (05h) 25
Parameter Object (0Fh) 30
Acknowledge Handler Object (2Bh) 33
Base Energy Object (4Eh) 34
Power Management Object (53h) 36
ABCC ADI Object (A2h) 38

Chapter 4 Anybus Module Objects

General Information 40
Anybus Object (01h) 41
Diagnostic Object (02h) 43
Network Object (03h) 44
Network Configuration Object (04h) 45
Anybus File System Interface Object (0Ah) 48

Chapter 5 Host Application Objects

General Information 49
CIP Identity Host Object (EDh) 50
SYNC Object (EEh) 51
DeviceNet Host Object (FCh) 52

Appendix A Categorization of Functionality

Basic 56
Extended 56

Appendix B Implementation Details

DeviceNet Implementation 57
SUP-Bit Definition 58
Anybus State Machine 58

Appendix C CIP Request Forwarding

Appendix D Technical Specification

Front View 61
Protective Earth (PE) Requirements 62
Power Supplies 62
Power Consumption 62
Environmental Specification 63
EMC Compliance 63

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 40 Software Design Guide	HMS
Anybus CompactCom M40 Hardware Design Guide	HMS
Anybus CompactCom B40 Design Guide	HMS
Anybus CompactCom Driver User Manual	HMS
DeviceNet Specification	ODVA
Common Industrial Protocol (CIP) specification	ODVA

P.2 Document History

Summary of Recent Changes (1.11... 1.20)

Change	Page(s)
Removed section 1.3 (information already in Technical Specification appendix)	
Extended and advanced categories joined to one, extended	
Updated description on how to assign a node address	12
Changed device address to node address	
Added description on how to set baud rate	12

Revision List

Revision	Date	Author(s)	Chapter(s)	Description
1.00	2014-09-15	KeL	All	First official version
1.10	2014-10-02	KeL	1, 4, D	Misc. updates
1.11	2015-01-30	KaD	-	Minor update
1.20	2015-10-23	KeL	2	Minor update

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.
- The terms 'basic' and 'extended' are used to classify objects, instances and attributes. Please refer to "Categorization of Functionality" on page 56 for more information.

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 40 DeviceNet

1.1 General

The Anybus CompactCom 40 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 40 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

- Pluggable 5.08 DeviceNet connector
- 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 Baud Rate Detection
- Modular Device functionality
- Quick Connect supported

2. Basic Operation

2.1 General Information

2.1.1 Software Requirements

Generally, no additional network support code needs to be written in order to support the Anybus CompactCom 40 DeviceNet. However, due to the nature of the DeviceNet networking system, certain requirements and facts 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 43) 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 40 Software Design Guide.

See also...

- "Diagnostic Object (02h)" on page 43 (Anybus Module Object)
- Anybus CompactCom 40 Software Design Guide, 'Application Data Object (FEh)'

2.1.2 Electronic Data Sheet (EDS)

Since the module implements the Parameter Object, it is possible for configuration tools such as RSNetWorx from Rockwell, 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 9
- "Parameter Object (0Fh)" on page 30 (CIP-object)
- Anybus CompactCom 40 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 Vendor ID, contact the ODVA.

2.2 Device Customization

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

- Vendor ID: 005Ah (HMS Industrial Networks AB)
- Device Type: 002Bh (Generic Device)
- Product Code: 003Fh (Anybus CompactCom 40 DeviceNet(TM))
- Product Name: 'CompactCom 40 DeviceNet(TM)'

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 17 (CIP-object)
- "DeviceNet Host Object (FCh)" on page 52 (Host Application Object)
- "CIP Request Forwarding" on page 59

IMPORTANT: *According to the CIP specification, the combination of Vendor ID and serial number must be unique. It is not permitted to use a custom serial number in combination with the HMS Vendor ID (005Ah), nor is it permitted to choose Vendor ID arbitrarily. Failure to comply to this requirement will induce interoperability problems and/or other unwanted side effects. HMS approves use of the HMS Vendor ID (005Ah), in combination with the default serial number, under the condition that the implementation requires no deviations from the standard EDS-file.*

To obtain a Vendor ID, contact the ODVA.

2.2.1 Modular Device Functionality

Modular devices consist of a backplane with a certain number of "slots". The first slot is occupied by the "coupler" which contains the Anybus CompactCom module. All other slots may be empty or occupied by modules. The Anybus CompactCom 40 DeviceNet module is configurable as a modular slave.

When mapping ADIs to process data, the application shall map the process data of each module in slot order.

A list of modules in a Modular Device is available to the DeviceNet network master by a request to the CIP Identity object.

See also ...

- "Modular device Object (ECh)" (see Anybus CompactCom 40 Software Design Guide)
- "Identity Object (01h)" on page 17 (CIP object)

2.2.2 Quick Connect

The module supports the Quick Connect functionality. The functionality is disabled by default and can be enabled in the DeviceNet Host Object. Enabling the functionality in the DeviceNet Host Object, will make it possible to enable/disable the use of it in the DeviceNet object (CIP) or the Network Configuration Object.

The module itself has a connection time down to 100 ms when Quick Connect is enabled. The actual connection time to the network will depend on the performance of the application, response time, and amount of process data to be mapped.

See also ...

- "DeviceNet Host Object (FCh)" on page 52
- "DeviceNet Object (03h)" on page 21 (CIP object)
- "Network Configuration Object (04h)" on page 45

2.3 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...

  - "Setting Baud Rate" on page 12

- **Node Address (MAC ID)**

  See also...

  - "Setting Node Address" on page 12

- **QuickConnect**

  See also...

  - "Instance Attributes (Instance #3, 'QuickConnect')" on page 47

The parameters in the Network Configuration Object (04h) are available from the network through the Identity Object (CIP-object). If the parameters are set through switches from the application during set-up, the parameters cannot be changed from the network, but still be read.

See also...

- "Identity Object (01h)" on page 17 (CIP-object)
- "DeviceNet Object (03h)" on page 21 (CIP-object)
- "Network Configuration Object (04h)" on page 45 (Anybus Module Object)

2.3.1 Setting Baud Rate

If automatic baud rate detection is disabled, the baud rate can be set according to the methods in the table below.

Method	Actions Required to be Performed by Host Application	Comments
Baud rate set only from network	• Set attribute #5 in the Network Configuration Object (04h), Instance #2 to 04h or larger. • Set attribute #10 in the DeviceNet Host Object (FCh) to TRUE.	An invalid value (04h or larger) is set by the host application. The module will go online using the latest configured baud rate. When a value is set from the network, the value will be set in attribute #5 in the Network Configuration Object (04h), Instance #2. The new baud rate will valid after the next reset.
Baud rate set only from application	• Set attribute #10 in the DeviceNet Host Object (FCh) to FALSE. • Set attribute #5 in the Network Configuration Object (04h), Instance #2 to any value between 00h - 03h • Each time the host application changes the value, the new value shall be written to attribute #5 in the Network Configuration Object (04h), Instance #2.	If an invalid value is set by the host application (04h or larger), the module will enter the "Communication faulted state" at network initialization. If, after network initialization, the configured and the last value set by the host application differ, a minor fault will be indicated. Attributes #7 and #9 in the CIP DeviceNet Object (03h) will be updated, see page 21.
Node address set from network or from application	• Set attribute #5 in the Network Configuration Object (04h), Instance #2. • Each time the host application changes the value, the new value shall be written to attribute #5 in the Network Configuration Object (04h), Instance #2. • Set attribute #10 in the DeviceNet Object (FCh) to TRUE.	If an invalid value is set by the host application, the module will return to the latest used. When a value is set from the network, the value will be set in attribute #5 in the Network Configuration Object (04h), Instance #2. The new baud rate will valid after the next reset. If, after network initialization, the configured and the last value set by the host application differ, a minor fault will be indicated. Attributes #7 and #9 in the CIP DeviceNet Object (03h) will be updated, see page 21.

2.3.2 Setting Node Address

There are three different methods to set the node address (the MAC ID) of the module.

Method	Actions Required to be Performed by Host Application	Comments
Node address set only from network	• Set attribute #5 in the Network Configuration Object (04h), Instance #1 to 64 or larger. • Set attribute #9 in the DeviceNet Host Object (FCh) to TRUE.	An invalid node address (64 - 255) is set by the host application. The module will go online using the latest configured address. When a node address is set from the network, the address will be set in attribute #5 in the Network Configuration Object (04h), Instance #1. The module deletes all Connection objects and restarts the network access process.

Method	Actions Required to be Performed by Host Application	Comments
Node address set only from application	• Set attribute #9 in the DeviceNet Host Object (FCh) to FALSE. • Set attribute #5 in the Network Configuration Object (04h), Instance #1 to any value between 0 - 63. • Each time the host application changes the value, the new value shall be written to attribute #5 in the Network Configuration Object (04h), Instance #1.	If an invalid value is set by the host application (64 - 255), the module will enter the "Communication faulted state" at network initialization. If, after network initialization, the configured and the last value set by the host application differ, a minor fault will be indicated. Attributes #6 and #8 in the CIP DeviceNet Object (03h) will be updated, see page 21.
Node address set from network or from application	• Set attribute #5 in the Network Configuration Object (04h), Instance #1. • Each time the host application changes the value, the new value shall be written to attribute #5 in the Network Configuration Object (04h), Instance #1. • Set attribute #9 in the DeviceNet Object (FCh) to TRUE.	If an invalid value is set by the host application, the module will return to the latest used. When a node address is set from the network, the address will be set in attribute #5 in the Network Configuration Object (04h), Instance #1. The module deletes all Connection objects and restarts the network access process. If, after network initialization, the configured and the last value set by the host application differ, a minor fault will be indicated. Attributes #6 and #8 in the CIP DeviceNet Object (03h) will be updated, see page 21.

See also ...

- "DeviceNet Host Object (FCh)" on page 52 (Host Application Object)
- "Network Configuration Object (04h)" on page 45 (Anybus Module Object)

2.4 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 17 (CIP-object)
- "Diagnostic Object (02h)" on page 43 (Anybus Module Object)

2.5 Data Exchange

2.5.1 Application Data (ADIs)

ADIs are represented on DeviceNet through the ABCC 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 30 (CIP-object)
- "ABCC ADI Object (A2h)" on page 38 (CIP-object)

2.5.2 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 23 (CIP-object)
- "Connection Object (05h)" on page 25 (CIP-object)

2.5.3 Translation of Data Types

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

Anybus Data Type[a]	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	
BITS8	BYTE	Each ADI element of this type occupies one byte.
BITS16	WORD	Each ADI element of this type occupies two bytes.
BITS32	DWORD	Each ADI element of this type occupies four bytes.
OCTET	USINT	Each ADI element of this type occupies one byte.
BITS1-7	BYTE	Blt fields of size 1 - 7
PAD0-8	BYTE	Blt fields of size 0 - 8 used for padding
PAD9-16	BYTE	Blt fields of size 9 - 16 used for padding

a. For more information about the Anybus data types, please consult the Anybus CompactCom 40 Software Design Guide.

3. CIP Objects

3.1 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 17
- "Message Router (02h)" on page 20
- "DeviceNet Object (03h)" on page 21
- "Assembly Object (04h)" on page 23
- "Connection Object (05h)" on page 25
- "Parameter Object (0Fh)" on page 30
- "Acknowledge Handler Object (2Bh)" on page 33

CIP Energy Objects:

- "Base Energy Object (4Eh)" on page 34
- "Power Management Object (53h)" on page 36

Vendor Specific Objects:

- "ABCC ADI Object (A2h)" on page 38

It is possible to implement additional CIP-objects in the host application using the CIP forwarding functionality, see "DeviceNet Host Object (FCh)" on page 52 and "CIP Request Forwarding" on page 59.

3.2 Identity Object (01h)

Category

Extended

Object Description

The Identity Object provides identification of and general information about the module.

The object supports multiple instances. Instance 1, which is the only mandatory instance, describes the whole product. It is used by applications to determine what nodes are on the network and to match an EDS file with a product on the network. The other (optional) instances describe different parts of the product e.g. the software.

If modular device functionality is enabled, a list of the modules in the slots can be retrieved and made available to the network master by sending a get request to class attribute 100.

Instance attributes 1 - 4 and 6 - 7 can be customized by implementing the DeviceNet Host Object.

Additional identity instances can be registered by implementing the CIP Identity Host Object (host application object).

See also ....

- "DeviceNet Host Object (FCh)" on page 52
- "CIP Identity Host Object (EDh)" on page 50

Supported Services

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

Class Attributes

#	Access	Name	Type	Comments
1	Get	Revision	UINT	0001h
2	Get	Max instance	UINT	Maximum instance number
3	Get	Number of instances	UINT	Number of instances
100	Get	Module ID List	Array of UINT32	List of Module IDs supported in the application if Modular Device functionality is enabled. See "Modular Device Functionality" on page 9 for more information,

Instance #1 Attributes

Extended

#	Access	Name	Type	Comments
1	Get	Vendor ID	UINT	005Ah (HMS Industrial Networks AB[a])
2	Get	Device Type	UINT	002Bh (Generic Device[a])
3	Get	Product Code	UINT	003Fh (Anybus CompactCom 40 DeviceNet(TM)[a])
4	Get	Revision	Struct of: {USINT, USINT}	Major and minor firmware revision[a]
5	Get	Status	WORD	See "Device Status" on page 19
6	Get	Serial Number	UDINT	Assigned by HMS[a]
7	Get	Product Name	SHORT_STRING	"CompactCom 40 DeviceNet(TM)" (Name of product[a])
11	Get/Set	Active language	Struct of: {USINT, USINT, USINT}	Set requests sent to this instance are forwarded to the application using the Set_Lang_Request service of the Anybus Application Object. The host application shall acknowledge or not acknowledge the set request. The module is then responsible for updating the language settings in the Anybus Object 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 and translated to CIP standard.

a. Can be customized by implementing the DeviceNet Host Object, see "DeviceNet Host Object (FCh)" on page 52

Device Status

bit(s)	Name	
0	Module Owned	
1	(reserved, set to 0)	
2	Configured[a]	
3	(reserved, set to 0)	
4... 7	Extended Device Status: Value: Meaning: 0010b Faulted I/O Connection 0011b No I/O connection established 0101b Major fault 0110b Connection in Run mode 0111b Connection in Idle mode (other) (reserved)	Priority (higher number means higher priority): 3 0 4 1 2
8	Set for minor recoverable faults	These bits represent a combination of network specific faults (see CIP specifications) and faults generated by the module (see "Diagnostic Object (02h)" on page 43)
9	Set for minor unrecoverable faults	
10	Set for major recoverable faults	
11	Set for major unrecoverable faults	
12... 15	(reserved, set to 0)	

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 (FFh) is set.

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 40 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-on reset). For further information, consult the general Anybus CompactCom 40 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-on reset + factory default). For further information, consult the general Anybus CompactCom 40 Software Design Guide.

3.3 Message Router (02h)

Category

Extended

Object Description

The Message Router Object provides a messaging connection point through which a client may address a service to any object class or instance residing in the physical module.

In the Anybus CompactCom 40 module it is issued internally to direct object requests.

Supported Services

Class	-
Instance:	-

Class Attributes

-

Instance Attributes

-

3.4 DeviceNet Object (03h)

Category

Extended

Object Description

This object provides means for configuring the DeviceNet interface of the module.

Supported Services

Class	Get Attribute Single
Instance:	Get Attribute Single Set Attribute Single Allocate Master/Slave Connection Set (4Bh) Release Master/Slave Connection 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 node address (0 - 63)
2	Baud Rate[a b]	Get/Set	USINT	Value:Baud rate: 0 125 kbps 1 250 kbps 2 500 kbps
3	BOI[c]	Get/Set	BOOL	Defines CAN controller action in case of a Bus-Off interrupt Value: Meaning False The CAN controller is reset, but will not try to restart the communication on the bus True The CAN controller is reset and will try to restart communication on the bus
4	Bus-Off Counter	Get/Set	USINT	00h
5	Allocation Information	Get	Struct of: BYTE USINT	 Allocation choice byte MAC ID (node address) of master
6	MAC ID Switch changed[d]	Get	BOOL	Indicates if the MAC ID (node address) has changed since startup Value: Meaning True Changed False No change
7	Baud rate Switch changed[e]	Get	BOOL	Indicates if the baud rate has changed since startup Value:Meaning True Changed False No change
8	MAC ID Switch value[d]	Get	USINT	Actual value of node address switches
9	Baud rate Switch value[e]	Get	USINT	Actual value of baud rate switches
10	Quick Connect[f]	Get/Set	Bool	Enables/Disables the Quick Connect feature. Disabled by default Value: Meaning True Enable False Disable
100	Disable auto baud	Get/Set	BOOL	Value: Meaning True Disable auto baud False Enable auto baud This setting is stored in NV memory.

a. Set access right for attributes 1 and 2 are conditional. For further information, see “Communication Settings” on page 11.
b. Setting this attribute will also affect attribute #100 (Disable auto baud).
c. A Bus-Off Interrupt is generated from the underlying CAN layer. It indicates that no communication is possible on the bus, e.g. due to a short circuit between lines.
d. Implementation of attributes 6 and 8 is conditional. The attributes are implemented if the node address (MAC ID) is set from the Network Configuration Object at startup.
e. Implementation of attributes 7 and 9 is conditional. The attributes are implemnented if the baud rate is set from the Network Configuration Object at startup.
f. Enabled if attribute #13 (‘Enable Quick Connect”) in the DeviceNet Host Object (FCh) is set to true, see “DeviceNet Host Object (FCh)” on page 52.

3.5 Assembly Object (04h)

Category

Extended

Object Description

The Assembly object uses static assemblies and holds the Process Data sent/received by the host application. It allows data to and from each object to be sent or received over a single connection. The default assembly instance IDs used are in the vendor specific range.

The terms "input" and "output" are defined from the network's point of view. An input will produce data on the network and an output will consume data from the network.

See also...

- "Process Data" on page 14
- "DeviceNet Host Object (FCh)" on page 52

Supported Services

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

Class Attributes

#	Name	Access	Type	Comments
1	Revision	Get	UINT	0002h
2	Max Instance	Get	UINT	Highest instance number

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	Process data, written from the application and sent to the CIP network. Corresponds to the Write Process data.
4	Size	Get	UINT	Number of bytes in attribute #3

See also...

- "Data Exchange" on page 14
- "DeviceNet Host Object (FCh)" on page 52

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	Get/Set	Array of BYTE	Process data, received from the CIP network master and read by the application. Corresponds to the Read Process data.
4	Size	Get	UINT	Number of bytes in attribute #3

See also...

- "Data Exchange" on page 14
- "DeviceNet Host Object (FCh)" on page 52

3.6 Connection Object (05h)

Category

Extended

Object Description

This object allocates and manages the internal resources associated with both I/O and Explicit Messaging Connections. It is used to model the communication specific characteristics of an application-to-application(s) relationship.

A specific Connection Object Instance manages the communication specific aspects related to an endpoint.

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	Value:State: 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	CAN ID for transmission
5	Consumed connection ID	Get	UINT	CAN ID for reception
6	Initial Comm Characteristics	Get	BYTE	The message group over which the communication occurs: Value:Message Group 21 Instance #1 33 Instances #10... #14
7	Produced Connection Size	Get	UINT	512 bytes
8	Consumed Connection Size	Get	UINT	512 bytes
9	Expected Packet Rate	Get/Set	UINT	2500ms (timing associated with this connection)
12	Watchdog timeout action	Get/Set	USINT	Value:Action: 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	Specifies the multiplier applied to the expected packet rate value to derive the value for the Inactivity/Watchdog Timer. Value:Meaning: 0: x4 (default) 1: x8 2: x16 3: x32 4: x64 5 x128 6: x256 7: x512 8-255: (reserved)

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	CAN ID for reception (Polled)
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, nn = producing instance number in assembly object)
15	Consumed Connection path length	Get	UINT	0007h
16	Consumed Connection path	Get	EPATH	20 04 25 nn nn 30 03h (nn = consuming instance number in assembly object)
17	Production Inhibit Time	Get	UINT	0000h
18	Connection Timeout Multiplier	Get/Set	UINT	Specifies the multiplier applied to the expected packet rate value to derive the value for the Inactivity/ Watchdog Timer. Value:Meaning: 0: x4 (default) 1: x8 2: x16 3: x32 4: x64 5 x128 6: x256 7: x512 8-255: (reserved)

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	CAN ID for transmission
5	Consumed connection ID	Get	UINT	CAN ID for reception
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 (nn = producing instance number in assembly object)
15	Consumed Connection path length	Get	UINT	0007h
16	Consumed Connection path	Get	EPATH	20 04 25 nn nn 30 03h (nn = consuming instance number in assembly object)
17	Production Inhibit Time	Get	UINT	0000h
18	Connection Timeout Multiplier	Get/Set	UINT	Specifies the multiplier applied to the expected packet rate value to derive the value for the Inactivity/Watchdog Timer. Value:Meaning: 0: x4 (default) 1: x8 2: x16 3: x32 4: x64 5 x128 6: x256 7: x512 8-255: (reserved)

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 (nn = producing instance number in assembly object)
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	Specifies the multiplier applied to the expected packet rate value to derive the value for the Inactivity/Watchdog Timer. Value:Meaning: 0: x4 (default) 1: x8 2: x16 3: x32 4: x64 5 x128 6: x256 7: x512 8-255: (reserved)

3.7 Parameter Object (0Fh)

Category

Extended

Object Description

The Parameter Object provides an interface to the Application Data Instances (ADIs) of the module. It can provide a full description of each parameter, including minimum and maximum values and a text string describing the parameter.

Each parameter is represented by one instance. Instance numbers start at 1, and are incremented by one, with no gaps in the list. Due to limitations imposed by the CIP standard, ADIs containing multiple elements (i.e. arrays and structures) cannot be represented through this object. In such cases, default values will be returned, see "Default Values" on page 32.

Configuration tools, such as RSNetworx, can extract information about the 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.

See also...

- "ABCC ADI Object (A2h)" on page 38 (CIP Object)
- "DeviceNet Host Object (FCh)" on page 52 (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 40 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	21 mm mm 25 nn nn 30 05h (Path to the object from where this parameter's value is retrieved, in this case the ADI Object. "mm mm" is A2 00h by default, but can be customized using the Anybus DeviceNet Host Object to change the ABCC ADI Class Object number)
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

3.8 Acknowledge Handler Object (2Bh)

Category

Extended

Object Description

This object notifies the producing application of acknowledge reception, acknowledge timeouts, and production retry limit.

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	16 ms (Time to wait for acknowledge, in ms, before resending)
2	Retry Limit	Get/Set	USINT	01h (number of ack timeouts before retry limit reached event)
3	Producing Connection Instance	Get	UINT	04h (Connection instance, which contains the path of the producing I/O application object, which will be notified of Ack Handler events)

3.9 Base Energy Object (4Eh)

Category

Extended

Object Description

The Base Energy Object acts as an "Energy Supervisor" for CIP Energy implementations. It is responsible for providing a time base for energy values, provides energy mode services, and can provide aggregation services for aggregating energy values up through the various levels of an industrial facility. It also provides a standard format for reporting energy metering results. The object is energy type independent and allows energy type specific data and functionality to be integrated into an energy system in a standard way. The Anybus CompactCom 40 DeviceNet module supports one instance of the Base Energy Object. For instance, an electric power monitor may count metering pulse output transitions of a separate metering device. The count of such transitions, represented by a Base Energy Object instance, would reflect the energy consumption measured by the separate metering device. An instance of the Base Energy Object may exist as a stand-alone instance, or it may exist in conjunction with an Electrical and/or Non-Electrical Energy Object instance[1]. If an instance of any of these objects is implemented in a device, it must be associated with a Base Energy Object instance in the device.

For this object to be able to access the network, the Energy Reporting Object (E7h) must be implemented in the host application, see the Anybus CompactCom 40 Software Design Guide for more information.

Supported Services

Class:	Get_Attribute_Single
Instance:	Get_Attribute_Single

Class Attributes

#	Name	Access	Type	Value
1	Revision	Get	UINT	0002h (Object revision)

1. These objects are not implemented in the Anybus CompactCom 40 DeviceNet

Instance Attributes

Extended

#	Name	Access	Type	Value/Description
1	Energy/ Resource Type	Get	UINT	Type of energy managed by this instance Always 0 (Generic)
2	Base Energy Object Capabilities	Get	UINT	Always 0 (Energy measured)
3	Energy Accuracy	Get	UINT	Specifies the accuracy of power and energy metering results, either in 0.01 percent of reading (default) or 0.01 of other units specified in attribute #4. If 0, unknown.
4	Energy Accuracy Basis	Get	UINT	Always 0 (Percent of reading)
7[a]	Consumed Energy Odometer	Get	ODOMETER[b] (Struct of : UINT, UINT, UINT, UINT, UINT)	The value of the consumed energy.
8[a]	Generated Energy Odometer	Get	ODOMETER[b] (Struct of : UINT, UINT, UINT, UINT, UINT)	The value of the generated energy.
12	Energy Type Specific Object Path	Get	Struct of: UINT (Path size) padded EPATH (Path)	NULL path

a. Depending on whether the instance reports consumed or generated energy, either attribute #7 or attribute #8 is required.

b. This struct data type makes it possible to represent very large values, in the range from 0 to 999 999 999 999 999. It's defined as a STRUCT of UINTs where each position holds a three-digit value, that, multiplied according to [$x10^{n}$, $x10^{n+3}$, $x10^{n+6}$, $x10^{n+9}$, $x10^{n+12}$], and added to each other will givet the total result.
E.g. if n=0 and the contents in the struct are [123, 234, 345, 456, 567] the resulting value will be 567 456 345 456 567.
The data type is not translated to any Anybus data type, but the value can be read and interpreted from the attributes above.

3.10 Power Management Object (53h)

Category

Extended

Object Description

The Power Management Object provides standardized attributes and services to support the control of devices into and out of paused or sleep states. The Energy Control Object (F0h) has to be implemented for this object to gain access to the network.

See also ..

- Energy Control Object (F0h) (Anybus CompactCom 40 Software Design Guide)

Supported Services

Class:	Get_Attribute_Single
Instance:	Get_Attribute_Single Power_Management Set_Pass_Code Clear_Pass_Code

Class Attributes

#	Name	Access	Type	Value
1	Revision	Get	UINT	0002h (Object revision)

Instance Attributes

Extended

#	Name	Access	Type	Value/Description
1	Power Management Command	Get	DWORD	Collection of bit fields comprising the most recent power management request.
2	Power Management Status	Get	DWORD	Collection of bit fields providing Power Management status information.
3	Client Path	Get	Struct of:	Specifies the EPATH from this instance (server) to its current owner (client).
			UINT (Path Size)	Size of path (in words
			Padded EPATH (Path)	
4	Number of Power Management Modes	Get	UINT	Number of Power Management Mode array entries in attribute 5.
5	Power Management Nodes	Get	Array of:	Array of low power modes
			Struct of:	Modes (Array of mode structures)
			USINT	Minimum Pause Units (Specifies the unit of Minimum Pause Time)
			UINT	Minimum Pause Time
			USINT	Resume Units (Specifies the unit of Resume Time)
			UINT	Resume Time (Required time to transition from the paused stated to the owned state.
			REAL	Power Level (Power in kW for this mode)
			BOOL	Availability (Specifies whether this mode can be entered given the current device state)
6	Sleeping State Support	Get	BOOL	0 (Sleeping state not supported)

3.11 ABCC 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.

The object number can be customized using the DeviceNet Host Object (FCh)

See also...

- Application Data Object (see Anybus CompactCom 40 Software Design Guide)
- "Parameter Object (0Fh)" on page 30 (CIP Object)
- "DeviceNet Host Object (FCh)" on page 52

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 = 0002h)
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 40 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 40 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 = Read access 1: Set = Write access 2: Not set (reserved) 3: Set = Write process data mapping possible 4: Set = Read process data mapping possible
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.
9	Number of subelements	Array of UINT	N/A	Each element defines the number of subelements of the corresponding element of the instance value for structures and variables. The number of subelements may only differ from 1 if the corresponding element is of ABCC data type CHAR or OCTET:

a. Converted to/from CIP standard by the module

4. Anybus Module Objects

4.1 General Information

This chapter specifies the Anybus Module Object implementation and how they correspond to the functionality in the Anybus CompactCom 40 DeviceNet.

The following Anybus Module Objects are implemented:

- "Anybus Object (01h)" on page 41
- "Diagnostic Object (02h)" on page 43
- "Network Object (03h)" on page 44
- "Network Configuration Object (04h)" on page 45
- "Anybus File System Interface Object (0Ah)" on page 48

4.2 Anybus Object (01h)

Category

Basic

Object Description

This object groups common Anybus information, and is described thoroughly in the general Anybus CompactCom 40 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	0403h (Anybus CompactCom 40)
2	Firmware version	Get	struct of: UINT8 Major UINT8 Minor UINT8 Build	(see Anybus CompactCom 40 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	
7	(reserved)			
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 40 Software Design Guide)
14	(reserved)			
15				
16				
17	Virtual attributes	Get/Set	Array of UINT	
18	Black list/White list	Get/Set	struct of UINT8 Infobits UINT8 ListLen UINT16 Prot#1 UINT16 Prot#2 ... UINT16 Prot#n	
19	Network Time	Get	UINT64	0 (The network does not support network time)

4.3 Diagnostic Object (02h)

Category

Basic

Object Description

This object provides a standardized way of handling host application events & diagnostics, and is thoroughly described in the general Anybus CompactCom 40 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 40 Software Design Guide
4	Highest instance no.	Get	UINT16	
11	Max no. of instances	Get	UINT16	5+1 (One instance is reserved for major events)
12	Supported functionality	Get	BITS32	0 (Latching events are not supported)

Instance Attributes (Instance #1)

Basic

#	Name	Access	Type	Value
1	Severity	Get	UINT8	See general Anybus CompactCom 40 Software Design Guide
2	Event Code	Get	UINT8	

In the Anybus CompactCom 40 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 13
- "Identity Object (01h)" on page 17 (CIP-object)

4.4 Network Object (03h)

Category

Basic

Object Description

For more information regarding this object, consult the general Anybus CompactCom 40 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	02h
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(TM)’
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	Holds exception information: Value: Meaning: 00h: No information 01h: Invalid assembly instance mapping, i.e. an assembly instance number is equal to 0, or the assembly instance numbers (consume/produce) are equal.

a. Consult the general Anybus CompactCom 40 Software Design Guide for further information.

4.5 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 11
- "Identity Object (01h)" on page 17 (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, ‘Node Address’)

Basic

#	Name	Access	Type	Description
1	Name[a]	Get	Array of CHAR	‘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 range: 0-63 default: 63
6	Configured Value	Get	UINT8	Holds the configured value, which will be written to attribute #5. Node address range: 0-63 default: 64 (invalid value)

a. Multilingual, see “Multilingual Strings” on page 47.

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 node address configuration via the network (unless disabled in the DeviceNet Host Object - in such case, the module will enter communication fault state at start up). If an invalid value is set from switches, the latest valid value will be used when going online.

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)
6	Configured Value	Get	ENUM	Holds the configured value, which will be written to attribute #5. Value:Enum. String:Meaning: 00h ‘125kbps’ 125kbps 01h ‘250kbps’ 250kbps 02h ‘500kbps’ 500kbps 03h ‘Autobaud’ Autobaud default: 04h (invalid value)

a. Multilingual, see “Multilingual Strings” on page 47.

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 baud rate configuration via the network (unless disabled in the DeviceNet Host Object - in such case, the module will enter communication fault state at start up). If an invalid value is set from switches, the latest valid value will be used when going online.

Instance Attributes (Instance #3, ‘QuickConnect’)

Please note that this instance will only be implemented if Quick Connect functionality has been enabled in the DeviceNet Host object during startup. It will be activated the first time the module enters Anybus state WAIT_PROCESS.

See also ...

- “DeviceNet Host Object (FCh)” on page 52
- “Anybus State Machine” on page 58

Basic

#	Name	Access	Type	Description
1	Name	Get	Array of CHAR	‘QuickConnect’
2	Data type	Get	UINT8	00h (BOOL)
3	Number of elements	Get	UINT8	01h (one element)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	BOOL	Value:Meaning: 00h ‘Disable’ (Default) 01h ‘Enable’
6	Configured Value	Get	BOOL	Holds the configured value, which will be written to attribute #5 when the module is reset. Value:Meaning: 00h ‘Disable’ 01h ‘Enable’

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	Baud rate	Datenrate	Veloc transf	Velocità dati	Vitesse

4.6 Anybus File System Interface Object (0Ah)

Category

Extended

Object Description

This object provides an interface to the built-in file system. In an Anybus CompactCom 40 Devicenet module, the file system consist of one folder, called "Firmware". This folder is used to save a firmware file to upgrade the module. After a reset, the firmware in the module will be upgraded and the file erased.

Please consult the Anybus CompactCom 40 Software Design Guide for more information.

Supported Commands

(Consult the general Anybus CompactCom 40 Software Design Guide for further information)

Object Attributes (Instance #0)

(Consult the general Anybus CompactCom 40 Software Design Guide for further information)

Instance Attributes

(Consult the general Anybus CompactCom 40 Software Design Guide for further information)

5. Host Application Objects

5.1 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 (FFh, see Anybus CompactCom 40 Software Design Guide)
- Application Data Object (FEh, see Anybus CompactCom 40 Software Design Guide)
- Modular Device Object (ECh, see Anybus CompactCom 40 Software Design Guide)
- Energy Control Object (F0h, see Anybus CompactCom 40 Software Design Guide)
- Energy Reporting Object (E7h, see Anybus CompactCom 40 Software Design Guide)
- "SYNC Object (EEh)" on page 51

Network Specific Objects:

- "CIP Identity Host Object (EDh)" on page 50
- "DeviceNet Host Object (FCh)" on page 52

5.2 CIP Identity Host Object (EDh)

Category

Extended

Object Description

This object allows for applications to support additional CIP identity instances. It is used to provide additional product identity information, e.g. concerning the software installed.

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

Object:	Get_Attribute
Instance:	Get_Attribute Get_Attribute_All

Object Attributes (Instance #0)

#	Name	Access	Type	Value	Comment
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, UINT8	(Major Revision, Minor Revision)	
5	Status	Get	UINT16	-	
6	Serial Number	Get	UINT32	-	
7	Product Name	Get	Array of CHAR	-	

5.3 SYNC Object (EEh)

Category

Extended

Object Description

This object implements the host application SYNC settings.

See also...

- Anybus CompactCom 40 Software Design Guide, “Sync”
- Anybus CompactCom 40 Software Design Guide, “Sync Object”

Supported Commands

Object:	Get_Attribute
Instance:	Get_Attribute Set_Attribute

Object Attributes (Instance #0)

(Consult the general Anybus CompactCom 40 Software Design Guide for further information)

Instance Attributes (Instance #1)

Extended

#	Name	Access	Type	Default Value	Comment
1	Cycle time	Get/Set	UINT32		When a connection is active, the value of attribute #9 (Expected Package Rate) of the Connection Object, instance #2 (CIP object) should be written to this attribute. See “Connection Object (05h)” on page 25.
2-8	(not imple-mented)				

5.4 DeviceNet Host Object (FCh)

Category

Basic, extended

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 17
- Anybus CompactCom 40 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 59)
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	002Bh	
3	Product Code	Get	UINT16	003Fh	
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	'CompactCom 40 DeviceNet(TM)'	

Extended

#	Name	Access	Type	Default Value	Comment
7	Producing Instance No.	Get	UINT16	0064h	See also... - "Instance 64h Attributes (Producing Instance)" on page 23 (CIP-instance)
8	Consuming Instance No.	Get	UINT16	0096h	See also... - "Instance 96h Attributes (Consuming Instance)" on page 24 (CIP-instance)
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 17 (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 17 (CIP-object)
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 55 - "CIP Request Forwarding" on page 59.

#	Name	Access	Type	Default Value	Comment
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 30 (CIP-object)
13	Enable Quick Connect	Get	BOOL	False	Value:Meaning: True Enable Quick Connect False Disable Quick Connect See also... - "DeviceNet Object (03h)" on page 21
20	Anybus CompactCom ADI Object Number	Get	UINT16	00A2h	This attribute is used either to change the object number of the Anybus CompactCom ADI Object or to disable the object. Valid ranges: 0064h-00C7h and 0300h-04FFh (within the Vendor Specific ranges). Any value outside these ranges will disable the ADI Object.

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 not identical - to the 'Process_CIP_Message_Request'-command in the Anybus CompactCom 40 EtherNet/IP.

See also...

- "Device Customization" on page 9
- "CIP Request Forwarding" on page 59

A. Categorization of Functionality

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

Some of the functionality offered may be specialized and/or seldom used. As most of the available network functionality is enabled and accessible, access to the specification of the industrial network may be 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 all Process Data at a change of state. An inhibit time can be set to prevent the module from sending too often.

  The module supports up to 512 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 512 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, and produces up to 512 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 512 bytes in each direction for this type of connection.

- **Explicit Connection**

  The predefined explicit connection has a buffer of 512 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-58 “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[a] 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 a HW reset is done.
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 behavior, e.g. watchdog timeout.	The Module Status LED will turn red to indicate a major fault, and turn the Network Status LED off.

a. A Bus-Off occurs when it is impossible to communicate on the underlying CAN layer, e.g. if the lines are short circuited.

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.

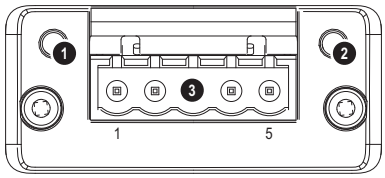
Field	Contents								Notes
	b7	b6	b5	b4	b3	b2	b1	b0	
Source ID	(Source ID)								(Selected by the module)
Dest. Object	FCh								Object = DeviceNet Host Object
Dest. Instance (lsb)	00h								Instance = Object
Dest. Instance (msb)	00h								
(command/error)	0								This message is not an ABCC error 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 Front View

#	Item
1	Network Status LED
2	Module status LED
3	DeviceNet Connector



Network Status

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

Module Status

State	Indication
Off	Not operating
Green	Operating in normal condition
Flashing Green (1 Hz)	Missing, incorrect or incomplete configuration, device needs commissioning.
Red	Unrecoverable Fault(s)
Flashing Red (1 Hz)	Recoverable Fault(s)
Alternating Red/Green	Executing 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-61 “Technical Specification”.

D.2 Protective Earth (PE) Requirements

In order to ensure proper EMC behavior, the module must be properly connected to protective earth via the PE pad / PE mechanism described in the general Anybus CompactCom 40 Hardware Design Guide.

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

D.3 Power Supplies

Supply Voltage

The module/brick requires a regulated 3.3 V power source as specified in the general Anybus CompactCom M40 Hardware Design Guide.

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 40 DeviceNet module accepts 11 - 25 V on the industrial network side of the module.

D.4 Power Consumption

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 40 Hardware Design Guide, and not on the exact power requirements of a single product.

Anybus CompactCom M40 DeviceNet

The Anybus CompactCom M40 DeviceNet is designed to fulfil the requirements of a Class B module. For more information about the power consumption classification used on the Anybus CompactCom platform, consult the general Anybus CompactCom M40 Hardware Design Guide.

The current hardware design consumes up to 280 mA[1].

Maximum current consumption on the network side at 11 - 25 V is 16 mA/module.

Anybus CompactCom B40-1 DeviceNet

The brick alone consumes up to 115 mA. The connector board will add up to 3.5 mA to the power consumption. A complete solution, including a brick, a connector board and LEDs with maximum allowed current consumption, will consume up to 147 mA.

Maximum current consumption on the network side at 11 - 25 V is 39 mA/brick.

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 40 DeviceNet will remain as a Class B module.

D.5 Environmental Specification

Consult the Anybus CompactCom 40 Hardware Design Guide for further information.

D.6 EMC Compliance

Consult the Anybus CompactCom 40 Hardware Design Guide for further information.