

Anybus[®] CompactCom[™] 40 - DeviceNet NETWORK GUIDE

HMSI-27-264
Version 2.5
Publication date 2024-12-18

Important User Information

Disclaimer

The information in this document is for informational purposes only. Please inform HMS Networks of any inaccuracies or omissions found in this document. HMS Networks disclaims any responsibility or liability for any errors that may appear in this document.

HMS Networks reserves the right to modify its products in line with its policy of continuous product development. The information in this document shall therefore not be construed as a commitment on the part of HMS Networks and is subject to change without notice. HMS Networks makes no commitment to update or keep current the information in this document.

The data, examples and illustrations found in this document are included for illustrative purposes and are only intended to help improve understanding of the functionality and handling of the product. In view of the wide range of possible applications of the product, and because of the many variables and requirements associated with any particular implementation, HMS Networks cannot assume responsibility or liability for actual use based on the data, examples or illustrations included in this document nor for any damages incurred during installation of the product. Those responsible for the use of the product must acquire sufficient knowledge in order to ensure that the product is used correctly in their specific application and that the application meets all performance and safety requirements including any applicable laws, regulations, codes and standards. Further, HMS Networks will under no circumstances assume liability or responsibility for any problems that may arise as a result from the use of undocumented features or functional side effects found outside the documented scope of the 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.

Copyright © 2024 HMS Networks

Contact Information

Postal address:

Box 4126

300 04 Halmstad, Sweden

E-Mail: info@hms.se

Table of Contents

1. Preface	1
1.1. About this document	1
1.2. Related Documents	1
1.3. Document History	1
1.4. Document Conventions	1
1.5. Document Specific Conventions	2
1.6. Abbreviations	2
1.7. Trademark Information	2
2. About the Anybus CompactCom 40 DeviceNet	4
2.1. General	4
2.2. Features	4
3. Basic Operation	5
3.1. General Information	5
3.1.1. Software Requirements	5
3.1.2. Electronic Data Sheet (EDS)	5
3.2. Device Customization	6
3.2.1. Modular Device Functionality	6
3.2.2. Quick Connect	7
3.3. Communication Settings	7
3.3.1. Setting Baud Rate	8
3.3.2. Setting Node Address	8
3.4. Diagnostics	9
3.5. Data Exchange	10
3.5.1. Application Data (ADIs)	10
3.5.2. Process Data	10
3.5.3. Translation of Data Types	11
4. CIP Objects	12
4.1. General Informaton	12
4.2. Identity Object (01h)	13
4.2.1. Category	13
4.2.2. Object Description	13
4.2.3. Supported Services	13
4.2.4. Class Attributes	13
4.2.5. Instance Attributes	14
4.2.6. Device Status	14
4.2.7. Service Details: Reset	15
4.3. Message Router (02h)	16
4.3.1. Category	16
4.3.2. Object Description	16
4.3.3. Supported Services	16
4.3.4. Class Attributes	16
4.3.5. Instance Attributes	16
4.4. DeviceNet Object (03h)	17
4.4.1. Category	17
4.4.2. Object Description	17
4.4.3. Supported Services	17
4.4.4. Class Attributes	17
4.4.5. Instance Attributes	18
4.5. Assembly Object (04h)	19

4.5.1. Category	19
4.5.2. Object Description	19
4.5.3. Supported Services	19
4.5.4. Class Attributes	19
4.5.5. Instance 64h Attributes (Producing Instance)	19
4.5.6. Instance 96h Attributes (Consuming Instance)	20
4.6. Connection Object (05h)	21
4.6.1. Category	21
4.6.2. Object Description	21
4.6.3. Supported Services	21
4.6.4. Class Attributes	21
4.6.5. Instances #1, #10... #14 Attributes (Explicit messaging)	22
4.6.6. Instance #2 Attributes (Poll or "COS/Cyclic consuming")	22
4.6.7. Instance #3 Attributes (Bit-strobe)	24
4.6.8. Instance #4 Attributes (COS/Cyclic producing)	25
4.7. Parameter Object (0Fh)	26
4.7.1. Category	26
4.7.2. Object Description	26
4.7.3. Supported Services	26
4.7.4. Class Attributes	26
4.7.5. Instance Attributes	27
4.7.6. Default Values	28
4.8. Acknowledge Handler Object (2Bh)	29
4.8.1. Category	29
4.8.2. Object Description	29
4.8.3. Supported Services	29
4.8.4. Class Attributes	29
4.8.5. Instance Attributes (01h)	29
4.9. Base Energy Object (4Eh)	30
4.9.1. Category	30
4.9.2. Object Description	30
4.9.3. Supported Services	30
4.9.4. Class Attributes	30
4.9.5. Instance Attributes (01h)	31
4.10. Power Management Object (53h)	32
4.10.1. Category	32
4.10.2. Object Description	32
4.10.3. Supported Services	32
4.10.4. Class Attributes	32
4.10.5. Instance Attributes	32
4.11. ABCC ADI Object (A2h)	33
4.11.1. Category	33
4.11.2. Object Description	33
4.11.3. Supported Services	33
4.11.4. Class Attributes	33
4.11.5. Instance Attributes	34
5. Anybus Module Objects	35
5.1. General Information	35
5.2. Anybus Object (01h)	36
5.2.1. Category	36
5.2.2. Object Description	36
5.2.3. Supported Commands	36
5.2.4. Object Attributes (Instance #0)	36
5.2.5. Instance Attributes (Instance #1)	37

5.3. Diagnostic Object (02h)	38
5.3.1. Category	38
5.3.2. Object Description	38
5.3.3. Supported Commands	38
5.3.4. Object Attributes (Instance #0)	38
5.3.5. Instance Attributes (Instance #1)	38
5.4. Network Object (03h)	39
5.4.1. Category	39
5.4.2. Object Description	39
5.4.3. Supported Commands	39
5.4.4. Object Attributes (Instance #0)	39
5.4.5. Instance Attributes (Instance #1)	39
5.5. Network Configuration Object (04h)	40
5.5.1. Category	40
5.5.2. Object Description	40
5.5.3. Supported Commands	40
5.5.4. Object Attributes (Instance #0)	40
5.5.5. Instance Attributes (Instance #1, Node Address)	40
5.5.6. Instance Attributes (Instance #2, Baud rate)	41
5.5.7. Instance Attributes (Instance #3, QuickConnect)	41
5.5.8. Multilingual Strings	42
5.6. Anybus File System Interface Object (0Ah)	43
5.6.1. Category	43
5.6.2. Object Description	43
5.6.3. Supported Commands	43
5.6.4. Object Attributes (Instance #0)	43
5.6.5. Instance Attributes (Instance #1)	43
6. Host Application Objects	44
6.1. General Information	44
6.2. CIP Identity Host Object (EDh)	45
6.2.1. Category	45
6.2.2. Object Description	45
6.2.3. Supported Commands	45
6.2.4. Object Attributes (Instance #0)	45
6.2.5. Instance Attributes (Instance #1)	45
6.3. Energy Reporting Object (E7h)	46
6.3.1. Category	46
6.3.2. Object Description	46
6.3.3. Supported Commands	46
6.3.4. Object Attributes (Instance #0)	46
6.3.5. Instance Attributes (Instance #1)	46
6.4. SYNC Object (EEh)	47
6.4.1. Category	47
6.4.2. Object Description	47
6.4.3. Supported Commands	47
6.4.4. Object Attributes (Instance #0)	47
6.4.5. Instance Attributes (Instance #1)	47
6.5. Energy Control Object (F0h)	48
6.5.1. Category	48
6.5.2. Object Description	48
6.5.3. Supported Commands	49
6.5.4. Object Attributes (Instance #0)	49
6.5.5. Instance Attributes (Instance #1 - #8)	50
6.6. DeviceNet Host Object (FCh)	54

6.6.1. Category	54
6.6.2. Object Description	54
6.6.3. Supported Commands	54
6.6.4. Object Attributes (Instance #0)	54
6.6.5. Instance Attributes (Instance #1)	55
6.6.6. Command Details: Process_CIP_Message_Request	58
Appendix A. Categorization of Functionality	59
1. Basic	59
2. Extended	59
Appendix B. Implementation Details	60
1. DeviceNet Implementation	60
1.1. Predefined Connection Set	60
1.2. Unconnected Message Server (UCMM)	60
2. Anybus State Machine	60
3. SUP-Bit Definition	61
Appendix C. CIP Request Forwarding	62
Appendix D. Technical Specification	64
1. Front View	64
1.1. Front View, DeviceNet Connector	64
1.2. Front View, M12 Connectors	64
1.3. Network Status	64
1.4. Module Status	64
1.5. DeviceNet Connector	65
1.6. M12 Connectors, Code A	65
2. Functional Earth (FE) Requirements	65
3. Power Supplies	65
3.1. Supply Voltage	65
3.2. DeviceNet Power Supply	65
4. Power Consumption	66
4.1. Anybus CompactCom M40 DeviceNet	66
4.2. Anybus CompactCom B40-1 DeviceNet	66
5. Environmental Specification	66
6. EMC Compliance	66
6.1. Environmental Specification	66
Appendix E. Backward Compatibility	67
1. Initial Considerations	67
2. Hardware Compatibility	68
2.1. Module	68
2.2. Chip	68
2.3. Brick	69
2.4. Host Application Interface	70
3. General Software	72
3.1. Extended Memory Areas	72
3.2. Faster Ping-Pong Protocol	72
3.3. Requests from Anybus CompactCom to Host Application During Startup	72
3.4. Anybus Object (01h)	72
3.5. Control Register CTRL_AUX-bit	72
3.6. Status Register STAT_AUX-bit	73
3.7. Control Register CTRL_R-bit	73
3.8. Modifications of Status Register, Process Data Read Area, and Message Data Read Area	73

- 4. Network Specific — DeviceNet 73
 - 4.1. DeviceNet Host Object (FCh) 73
 - 4.2. EDS file (Electronic Datasheet file used by configuration tool) 74

This page is intentionally left blank.

1. Preface

1.1. About this document

This document is intended to provide a good understanding of the functionality offered by the Anybus CompactCom 40 DeviceNet. The document describes the features that are specific to Anybus CompactCom 40 DeviceNet. For general information regarding 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 network guide should normally be sufficient to implement 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 persons 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.

For additional related documentation and file downloads, please visit the support website at <http://www.hms-networks.com/technical-support>.

1.2. Related Documents

Document	Author	Document ID
Anybus CompactCom 40 Software Design Guide	HMS	HMSI-216-125
Anybus CompactCom M40 Hardware Design Guide	HMS	HMSI-216-126
Anybus CompactCom B40 Design Guide	HMS	HMSI-27-230
Anybus CompactCom Host Application Implementation Guide	HMS	HMSI-27-334
DeviceNet Specification	ODVA	
CIP specification, Volumes 1 (CIP Common)	ODVA	

1.3. Document History

Version	Date	Description
1.00	2014-09-15	First official revision
1.10	2014-10-02	Misc. updates
1.11	2015-01-30	Minor update
1.20	2015-10-23	Minor update
2.0	2017-01-23	Moved from FM to DOX M12 connectors added Misc. updates
2.1	2017-07-10	Added appendix on backward compatibility
2.2	2018-08-28	Corrected table for baud rate settings Minor corrections
2.3	2019-02-28	Added description of objects E7h and F0h Rebranding
2.4	2024-06-07	Migrated document from DOX to Paligo Added attribute 23 to DeviceNet Host Object (FCh), instance attribute (instance 1) Minor corrections
2.5	2024-12-18	Added Reset as a supported command

1.4. Document Conventions

Numbered lists indicate tasks that should be carried out in sequence:

1. First do this
2. Then do this

Bulleted lists are used for:

- Tasks that can be carried out in any order
- Itemized information
- An action
 - and a result

User interaction elements (buttons etc.) are indicated with bold text.

Program code and script examples

Cross-reference within this document: [Document Conventions \(page 1\)](#)

External link (URL): www.hms-networks.com

**WARNING**

Instruction that must be followed to avoid a risk of death or serious injury.

**CAUTION**

Instruction that must be followed to avoid a risk of personal injury.

**IMPORTANT**

Instruction that must be followed to avoid a risk of reduced functionality and/or damage to the equipment, or to avoid a network security risk.

**NOTE**

Additional information which may facilitate installation and/or operation.

1.5. Document Specific Conventions

- The terms “Anybus” or “module” refers to the Anybus CompactCom module.
- The terms “host” or “host application” refer to the device that hosts the Anybus.
- 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.

1.6. Abbreviations

Abbreviation	Meaning
API	assigned packet interval
RPI	requested packet interval
T	target (in this case the module)
O	origin (in this case the master)

1.7. Trademark Information

Anybus® is a registered trademark of HMS Industrial Networks .

DeviceNet™ is a trademark of ODVA, Inc.

All other trademarks are the property of their respective holders.

2. About the Anybus CompactCom 40 DeviceNet

2.1. General

The Anybus CompactCom 40 DeviceNet communication module provides instant DeviceNet conformance tested connectivity via the patented Anybus CompactCom host interface. Any device that supports this standard can take advantage of the features provided by the module, allowing seamless network integration regardless of network type.

The modular approach of the Anybus CompactCom 40 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 Anybus CompactCom 40 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.

2.2. Features

- Pluggable 5.08 DeviceNet connector or M12 connectors
- Brick version
- CIP Parameter Object Support
- Explicit messaging
- UCMEM Capable
- Bit-strobed I/O
- Change-of-state / Cyclic I/O
- Polled I/O
- Expansion possibilities via CIP forwarding
- Customizable Identity object
- Max. read process data: 512 bytes
- Max. write process data: 512 bytes
- Max. process data (read + write, in bytes): 1024 bytes
- Automatic Baud Rate Detection
- Modular Device functionality
- Quick Connect supported

3. Basic Operation

3.1. General Information

3.1.1. Software Requirements

No additional network support code needs to be written in order to support the Anybus CompactCom 40 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 \(02h\) \(page 38\)](#)) can be created by the host application during normal conditions. An additional 6th instance may be created in event of a major fault. This limit is set by the module, not by the network.

For in-depth information regarding the Anybus CompactCom software interface, consult the general Anybus CompactCom 40 Software Design Guide.

See also...

- [Diagnostic Object \(02h\) \(page 38\)](#) (Anybus Module Object)
- Anybus CompactCom 40 Software Design Guide, “Application Data Object (FEh)”

For in depth information regarding the Anybus CompactCom software interface, consult the Anybus CompactCom 40 Software Design Guide.

3.1.2. Electronic Data Sheet (EDS)

Since the module implements the Parameter Object, it is possible for configuration tools such as 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 \(page 6\)](#)
- [Parameter Object \(0Fh\) \(page 26\)](#) (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.

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:	002Bh (Generic Device)
Product Code:	003Fh (Anybus CompactCom 40 DeviceNet(TM))
Product Name:	"Anybus 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 tasks:

- 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 \(01h\) \(page 13\)](#) (CIP object)
- [DeviceNet Host Object \(FCh\) \(page 54\)](#) (Host Application Object)
- [CIP Request Forwarding \(page 62\)](#)



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.

3.2.1. 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...

- [Identity Object \(01h\) \(page 13\)](#) (CIP object)
- Anybus CompactCom 40 Software Design Guide, "Modular device Object (ECh)"

3.2.2. 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 Object \(03h\) \(page 17\)](#) (CIP object)
- [DeviceNet Host Object \(FCh\) \(page 54\)](#) (Host Application Object)
- [Network Configuration Object \(04h\) \(page 40\)](#)

3.3. Communication Settings

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

Baud rate	Setting Baud Rate (page 8)
Node Address (MAC ID)	Setting Node Address (page 8)
QuickConnect	Network Configuration Object (04h) (page 40)

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 setup, the parameters cannot be changed from the network, but still be read.

See also...

- [Identity Object \(01h\) \(page 13\)](#)
- [DeviceNet Object \(03h\) \(page 17\)](#) (CIP object)
- [Network Configuration Object \(04h\) \(page 40\)](#) (Anybus Module Object)

3.3.1. 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.
Baud rate 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.

3.3.2. 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.
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.
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 also ...

- [DeviceNet Host Object \(FCh\) \(page 54\)](#) (Host Application Object)
- [Network Configuration Object \(04h\) \(page 40\)](#) (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 \(01h\) \(page 13\)](#) (CIP Object)
- [Diagnostic Object \(02h\) \(page 38\)](#) (Anybus Module Object)

3.5. Data Exchange

3.5.1. 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.

- [Parameter Object \(0Fh\) \(page 26\)](#) (CIP Object)
- [ABCC ADI Object \(A2h\) \(page 33\)](#) (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.

If the Host Assembly Mapping Object (EBh) is implemented, it is possible for the application to define a set of assembly instances that can be used by an IO connection. The IO connection is triggered by setting attribute #9 in instance #2, #3, or #4 in the Connection Object (05h, CIP object). The Anybus CompactCom will send remapping commands to the Application Data Object (FEh). The module will map the ADIs (bound to CIP assembly instances), pointed to by class attributes #14 and #16 in the Connection Object, to instances defined by the application in the Assembly Mapping Object. Attributes #21 and #22 in the DeviceNet Host Object are used to bind each CIP assembly instance to the correct assembly mapping instance.

See also...

- [Assembly Object \(04h\) \(page 19\)](#) (CIP Object)
- [Connection Object \(05h\) \(page 21\)](#) (CIP Object)
- [DeviceNet Host Object \(FCh\) \(page 54\)](#)
- Anybus CompactCom 40 Software Design Guide, "Assembly Mapping Object (EBh)"

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	
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	Bit fields of size 1 - 7
PAD0-8	BYTE	Bit fields of size 0 - 8 used for padding
PAD9-16	BYTE	Bit fields of size 9 - 16 used for padding
BOOL1	BOOL	

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

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 \(01h\) \(page 13\)](#)
- [Message Router \(02h\) \(page 16\)](#)
- [DeviceNet Object \(03h\) \(page 17\)](#)
- [Assembly Object \(04h\) \(page 19\)](#)
- [Connection Object \(05h\) \(page 21\)](#)
- [Parameter Object \(0Fh\) \(page 26\)](#)
- [Acknowledge Handler Object \(2Bh\) \(page 29\)](#)

CIP Energy Objects:

- [Base Energy Object \(4Eh\) \(page 30\)](#)
- [Power Management Object \(53h\) \(page 32\)](#)

Vendor Specific Objects:

- [ABCC ADI Object \(A2h\) \(page 33\)](#)

It is possible to implement additional CIP-objects in the host application using the CIP forwarding functionality, see [DeviceNet Host Object \(FCh\) \(page 54\)](#) and [CIP Request Forwarding \(page 62\)](#).

4.2. Identity Object (01h)

4.2.1. Category

Extended

4.2.2. 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 \(FCh\) \(page 54\)](#)
- [CIP Identity Host Object \(EDh\) \(page 45\)](#)

4.2.3. Supported Services

Class: Get_Attribute_Single
Get_Attributes_All

Instance: Get_Attribute_Single
Set_Attribute_Single
Get_Attributes_All
Reset

4.2.4. Class Attributes

#	Name	Access	Type	Value
1	Revision	Get	UINT	0001h (Object revision)
2	Max instance	Get	UINT	Maximum instance number
3	Number of instances	Get	UINT	Number of instances
100	Module ID List	Get	Array of UINT32	If modular device functionality is enabled, a request to this attribute will generate a Get_List request to the Modular Device Object in the host application. See Modular Device Functionality (page 6)

4.2.5. Instance Attributes

Attributes #1–4 and #6–7 can be customized by implementing the DeviceNet Host Object, see [DeviceNet Host Object \(FCh\) \(page 54\)](#)

#	Name	Access	Type	Value/Description
1	Vendor ID	Get	UINT	005Ah (HMS Industrial Networks)
2	Device Type	Get	UINT	002Bh (Generic Device)
3	Product Code	Get	UINT	003Fh (Anybus CompactCom 40 DeviceNet(TM))
4	Revision	Get	Struct of: USINT USINT	Major and minor firmware revision
5	Status	Get	WORD	See Device Status table below
6	Serial Number	Get	UDINT	Unique serial number (assigned by HMS Industrial Networks)
7	Product Name	Get	SHORT_STRING	"Anybus CompactCom 40 DeviceNet(TM)"
11	Active language	Set	Struct of: USINT USINT USINT	Requests sent to this instance are forwarded to the Application Object. If the request is accepted, the module will update the language accordingly.
12	Supported Language List	Get	Array of: Struct of: USINT USINT USINT	List of languages supported by the host application. The list is read from the Application Object and translated to CIP standard.

4.2.6. Device Status

bit(s)	Name		
0	Module Owned This bit is set when at least one of the supported connection types has been established. For more information on connection types, see DeviceNet Implementation (page 60) .		
1	(reserved, set to 0)		
2	Configured 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.		
3	(reserved, set to 0)		
4... 7	Extended Device Status:		
	Value:	Meaning:	Priority (higher number means higher priority):
	0010b	Faulted I/O Connection	3
	0011b	No I/O connection established	0
	0101b	Major fault	4
	0110b	Connection in Run mode	1
	0111b	Connection in Idle mode	2
	(other)	(reserved)	
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) (page 38))
9	Set for minor unrecoverable faults		
10	Set for major recoverable faults		
11	Set for major unrecoverable faults		
12... 15	(reserved, set to 0)		

4.2.7. Service Details: Reset

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

4.3. Message Router (02h)

4.3.1. Category

Extended

4.3.2. 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 module it is used internally to direct object requests.

4.3.3. Supported Services

Class: -

Instance: -

4.3.4. Class Attributes

-

4.3.5. Instance Attributes

-

4.4. DeviceNet Object (03h)

4.4.1. Category

Extended

4.4.2. Object Description

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

4.4.3. 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)

4.4.4. Class Attributes

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

4.4.5. Instance Attributes

#	Name	Access	Type	Comments	
1	MAC ID	Get/Set	USINT	Currently used node address (0 - 63). Set access right is conditional. For more information, see Communication Settings (page 7) .	
2	Baud Rate	Get/Set	USINT	<u>Value</u>	<u>Baud Rate</u>
				0	125 kbps
				1	250 kbps
				2	500 kbps
				Set access right is conditional. For more information, see Communication Settings (page 7) . Setting this attribute will also affect attribute #100 (Disable auto baud).	
3	BOI	Get/Set	BOOL	Defines CAN controller action in case of a Bus-Off interrupt. 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.	
				<u>Value</u>	<u>Meaning</u>
				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
				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.	
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	Get	BOOL	Indicates if the MAC ID (node address) has changed since startup. The attribute is implemented only if the node address (MAC ID) is set from the Network Configuration Object at startup.	
				<u>Value</u>	<u>Meaning</u>
				True	Changed
				False	No change
7	Baud rate Switch changed	Get	BOOL	Indicates if the baud rate has changed since startup. The attribute is implemented only if the baud rate is set from the Network Configuration Object at startup.	
				<u>Value</u>	<u>Meaning</u>
				True	Changed
				False	No change
8	MAC ID Switch value	Get	USINT	Actual value of node address switches. The attribute is implemented only if the node address (MAC ID) is set from the Network Configuration Object at startup.	
9	Baud rate Switch value	Get	USINT	Actual value of baud rate switches. The attribute is implemented only if the baud rate is set from the Network Configuration Object at startup.	
10	Quick Connect	Get/Set	BOOL	Enables/Disables the Quick Connect feature. Disabled by default. Enabled if attribute #13 ("Enable Quick Connect") in the DeviceNet Host Object (FCh) is set to true DeviceNet Host Object (FCh) (page 54)	
				<u>Value</u>	<u>Meaning</u>
				True	Enable
				False	Disable
100	Disable auto baud	Get/Set	BOOL	Enables/Disables auto baud. Stored in non-volatile memory	
				<u>Value</u>	<u>Meaning</u>
				True	Enable
				False	Disable

4.5. Assembly Object (04h)

4.5.1. Category

Extended

4.5.2. 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

- [DeviceNet Host Object \(FCh\) \(page 54\)](#)
- [Process Data \(page 10\)](#)

4.5.3. Supported Services

Class: Get_Attribute_Single

Instance: Get_Attribute_Single
Set_Attribute_Single

4.5.4. Class Attributes

#	Name	Access	Type	Value
1	Revision	Get	UINT	0002h (Object revision)
2	Max instance	Get	UINT	Maximum instance number

4.5.5. Instance 64h Attributes (Producing Instance)

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

#	Name	Access	Type	Value/Description
3	Produced Data	Get	Array of BYTE	Process data, written from the application and sent to the CIP network. This data corresponds to the Write Process Data.
4	Size	Get	UINT	Number of bytes in attribute #3

See also...

[Data Exchange \(page 10\)](#)

[DeviceNet Host Object \(FCh\) \(page 54\)](#)

4.5.6. Instance 96h Attributes (Consuming Instance)

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

#	Name	Access	Type	Value/Description
3	Produced Data	Get	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 \(page 10\)](#)

[DeviceNet Host Object \(FCh\) \(page 54\)](#)

4.6. Connection Object (05h)

4.6.1. Category

Extended

4.6.2. 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 end-point. For more information on connection types, see [DeviceNet Implementation \(page 60\)](#).

4.6.3. Supported Services

Class: Get_Attribute_Single

Instance: Get_Attribute_Single
 Set_Attribute_Single

4.6.4. Class Attributes

#	Name	Access	Type	Value
1	Revision	Get	UINT	0001h (Object revision)

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

#	Name	Access	Type	Comments	
1	State	Get	USINT	<u>Value</u>	<u>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	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:	
				<u>Value</u>	<u>Message Group</u>
				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	2500 ms (timing associated with this connection)	
12	Watchdog timeout action	Get/Set	USINT	<u>Value</u>	<u>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	BOOL	Specifies the multiplier applied to the expected packet rate value to derive the value for the Inactivity/Watchdog Timer.	
				<u>Value</u>	<u>Meaning</u>
				0	x4 (default)
				1	x8
				2	x16
				3	x32
				4	x64
				5	x128
				6	x256
				7	x512
				8-255	Reserved

4.6.6. Instance #2 Attributes (Poll or “COS/Cyclic consuming”)

When attribute #9 (EPR, Expected Packet Rate) is set from the network, an IO connection is triggered. The Anybus will send remap commands to the Anybus Application Data Object to map the CIP assembly instances pointed to by the attributes #14 and #16, to the process data area.

#	Name	Access	Type	Comments	
1	State	Get	USINT	<u>Value</u>	<u>State</u>
				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	<u>Value</u>	<u>Meaning</u>
				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	<u>Value</u>	<u>Meaning</u>
				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	<u>Value</u>	<u>Message Group</u>
				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	<u>Value</u>	<u>Meaning</u>
				FFFFh	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	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	0000h (COS/Cyclic)	
				0007h (Polled)	
14	Produced Connection path	Get (COS/Cyclic)	EPATH	No value (COS/Cyclic)	
		Get/Set (Polled)		20 04 25 nn nn 30 03h (Polled, nn = producing instance number in assembly object)	
				Contents will be stored in non volatile storage	
15	Consumed Connection path length	Get/Set	UINT	0007h	
16	Consumed Connection path	Get/Set	EPATH	20 04 25 nn nn 30 03h (nn = consuming instance number in assembly object) Contents will be stored in non volatile storage	
17	Production Inhibit Time	Get	UINT	0000h	
18	Connection Timeout Multiplier	Get/Set	BOOL	Specifies the multiplier applied to the expected packet rate value to derive the value for the Inactivity/Watchdog Timer.	
				<u>Value</u>	<u>Meaning</u>
				0	x4 (default)
				1	x8
				2	x16
				3	x32
				4	x64
				5	x128

#	Name	Access	Type	Comments	
				6	x256
				7	x512
				8-255	Reserved

4.6.7. Instance #3 Attributes (Bit-strobe)

When attribute #9 (EPR, Expected Packet Rate) is set from the network, an IO connection is triggered. The Anybus will send remap commands to the Anybus Application Data Object to map the CIP assembly instances pointed to by the attributes #14 and #16, to the process data area.

#	Name	Access	Type	Comments	
1	State	Get	USINT	<u>Value</u>	<u>State</u>
				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	Timing associated with this connection	
12	Watchdog timeout action	Get/Set	USINT	0000h (Transition to the timed out state)	
13	Produced Connection path length	Get	UINT	0007h	
14	Produced Connection path	Get/Set	EPATH	20 04 25 nn nn 30 03h (nn = producing instance number in assembly object) Contents will be stored in non volatile storage	
15	Consumed Connection path length	Get	UINT	0007h	
16	Consumed Connection path	Get/Set	EPATH	20 04 25 nn nn 30 03h (nn = consuming instance number in assembly object) Contents will be stored in non volatile storage	
17	Production Inhibit Time	Get	UINT	0000h	
18	Connection Timeout Multiplier	Get/Set	BOOL	Specifies the multiplier applied to the expected packet rate value to derive the value for the Inactivity/Watchdog Timer.	
				<u>Value</u>	<u>Meaning</u>
				0	x4 (default)
				1	x8
				2	x16
				3	x32
				4	x64
				5	x128
				6	x256
				7	x512
				8-255	Reserved

4.6.8. Instance #4 Attributes (COS/Cyclic producing)

When attribute #9 (EPR, Expected Packet Rate) is set from the network, an IO connection is triggered. The Anybus will send remap commands to the Anybus Application Data Object to map the CIP assembly instances pointed to by the attributes #14 and #16, to the process data area.

#	Name	Access	Type	Comments	
1	State	Get	USINT	<u>Value</u>	<u>State</u>
				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	<u>Value</u>	<u>Meaning</u>
				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	<u>Value</u>	<u>Meaning</u>
				FFFFh	Not acknowledged
				Other	CAN ID for reception (Ack.)
6	Initial Comm Characteristics	Get	BYTE	<u>Value</u>	<u>Message Group</u>
				0Fh	Producing only over message group 1 (No Ack.)
				01hh	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/Set	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) Contents will be stored in non volatile storage	
16	Consumed Connection path	Get/Set	EPATH	No value (No ack.)	
				20 2B 25 01 00h (Acknowledged) Contents will be stored in non volatile storage	
17	Production Inhibit Time	Get/Set	UINT	0000h	
18	Connection Timeout Multiplier	Get/Set	BOOL	Specifies the multiplier applied to the expected packet rate value to derive the value for the Inactivity/Watchdog Timer.	
				<u>Value</u>	<u>Meaning</u>
				0	x4 (default)
				1	x8
				2	x16
				3	x32
				4	x64
				5	x128
				6	x256
				7	x512
				8-255	Reserved

4.7. Parameter Object (0Fh)

4.7.1. Category

Extended

4.7.2. 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 table with default values below.

Configuration tools, such as RSNetworkx, 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 \(A2h\) \(page 33\)](#) (CIP Object)
- [DeviceNet Host Object \(FCh\) \(page 54\)](#) (Host Application Object)

4.7.3. Supported Services

Class: Get_Attribute_Single

Instance: Get_Attribute_Single
Set_Attribute_Single
Get_Attributes_All
Get_Enum_String

4.7.4. Class Attributes

#	Name	Access	Type	Value	
1	Revision	Get	UINT	0001h (Object revision)	
2	Max instance	Get	UINT	Maximum created instance number = class attribute 3 in the Application Data Object, see the general Anybus CompatCom 40 Software Design Guide for further information.	
8	Parameter class descriptor	Get	WORD	Default: 0000 0000 0000 01011b	
				<u>Bit</u>	<u>Contents</u>
				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)	

4.7.5. Instance Attributes

#	Name	Access	Type	Comments	
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	<u>Bit</u>	<u>Contents</u>
				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		

4.7.6. 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)

4.8.1. Category

Extended

4.8.2. Object Description

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

4.8.3. Supported Services

Class: Get_Attribute_Single

Instance: Get_Attribute_Single
Set_Attribute_Single

4.8.4. Class Attributes

#	Name	Access	Type	Value
1	Revision	Get	UINT	0001h (Object revision)

4.8.5. Instance Attributes (01h)

#	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)

4.9. Base Energy Object (4Eh)

4.9.1. Category

Extended

4.9.2. 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 (not implemented in the Anybus CompactCom 40 DeviceNet). 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 [Energy Reporting Object \(E7h\) \(page 46\)](#).

4.9.3. Supported Services

Class: Get_Attribute_Single

Instance: Get_Attribute_Single

4.9.4. Class Attributes

#	Name	Access	Type	Value
1	Revision	Get	UINT	0001h (Object revision)

4.9.5. Instance Attributes (01h)

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

This struct data type ODOMETER (attributes #7 and #8) makes it possible to represent very large values, in the range from 0 to 999 999 999 999. It's defined as a STRUCT of UINTs where each position holds a three-digit value, that, multiplied according to [x10ⁿ, x10ⁿ+3, x10ⁿ+6, x10ⁿ+9, x10ⁿ+12], and added to each other will give 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.

#	Name	Access	Type	Value
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	Consumed Energy Odometer	Get	ODOMETER (Struct of : UINT, UINT, UINT, UINT, UINT)	The value of the consumed energy.
8	Generated Energy Odometer	Get	ODOMETER (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

4.10. Power Management Object (53h)

4.10.1. Category

Extended

4.10.2. 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)

4.10.3. Supported Services

Class: Get_Attribute_Single

Instance: Get_Attribute_Single
Power_Management
Set_Pass_Code
Clear_Pass_Code

4.10.4. Class Attributes

#	Name	Access	Type	Value
1	Revision	Get	UINT	0001h (Object revision)

4.10.5. Instance Attributes

#	Name	Access	Type	Value
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)
6	Sleeping State Support	Get	BOOL	Availability (Specifies whether this mode can be entered given the current device state)
			BOOL	0 (Sleeping state not supported)

4.11. ABCC ADI Object (A2h)

4.11.1. Category

Extended

4.11.2. 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 \(0Fh\) \(page 26\)](#) (CIP Object)
- [DeviceNet Host Object \(FCh\) \(page 54\)](#)

4.11.3. Supported Services

Class: Get_Attribute_Single

Instance: Get_Attribute_Single
Set_Attribute_Single

4.11.4. Class Attributes

#	Name	Access	Type	Value
1	Revision	Get	UINT	0002h (Object revision)
2	Max Instance	Get	UINT	Equals attribute #4 in the Application Data Object See the general Anybus CompactCom 40 Software Design Guide for further information.
3	Number of instances	Get	UINT	Equals attribute #3 in the Application Data Object See the general Anybus CompactCom 40 Software Design Guide for further information.

4.11.5. Instance Attributes

Each instance corresponds to an instance within the Application Data Object (for more information, see the general Anybus CompactCom 40 Software Design Guide)

Attributes #5 - #8 are converted to/from CIP standard by the module

#	Name	Access	Type	Value	
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	Get/Set	Determined by attribute #2	Instance value	
6	Max value	Get		The maximum permitted parameter value.	
7	Min value	Get		The minimum permitted parameter value.	
8	Default value	Get		The default parameter value.	
9	Number of Power Management Modes	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.	

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 40 DeviceNet.

The following Anybus Module Objects are implemented:

[Anybus Object \(01h\) \(page 36\)](#)

[Diagnostic Object \(02h\) \(page 38\)](#)

[Network Object \(03h\) \(page 39\)](#)

[Network Configuration Object \(04h\) \(page 40\)](#)

[Anybus File System Interface Object \(0Ah\) \(page 43\)](#)

5.2. Anybus Object (01h)

5.2.1. Category

Basic

5.2.2. Object Description

This object assembles all common Anybus data, and is described thoroughly in the general Anybus CompactCom 40 Software Design Guide.

5.2.3. Supported Commands

Object:	Get_Attribute
	Reset
Instance:	Get_Attribute
	Set_Attribute
	Get_Enum_String

5.2.4. Object Attributes (Instance #0)

#	Name	Access	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

5.2.5. Instance Attributes (Instance #1)

Basic

#	Name	Access	Type	Value	
1	Module type	Get	UINT16	0403h (Standard Anybus CompactCom 40)	
2	-	-	-	See the general Anybus CompactCom 40 Software Design Guide for further information.	
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		
12	LED colors	Get	struct of:	<u>Value:</u>	<u>Color:</u>
			UINT8 (LED1A)	01h	Green
			UINT8 (LED1B)	02h	Red
			UINT8 (LED2A)	01h	Green
			UINT8 (LED2B)	02h	Red
13	LED status	Get	UINT8	See the general Anybus CompactCom 40 Software Design Guide for further information.	
14-16	(reserved)				
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)	

5.3. Diagnostic Object (02h)

5.3.1. Category

Basic

5.3.2. 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.

5.3.3. Supported Commands

Object:

- Get_Attribute
- Create
- Delete

Instance: Get_Attribute

5.3.4. Object Attributes (Instance #0)

#	Name	Access	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)

5.3.5. Instance Attributes (Instance #1)

Basic

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.

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

See also...

- [Diagnostics \(page 9\)](#)
- [Identity Object \(01h\) \(page 13\)](#) (CIP object)

5.4. Network Object (03h)

5.4.1. Category

Basic

5.4.2. 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.

5.4.3. Supported Commands

Object:	Get_Attribute
Instance:	Get_Attribute
	Set_Attribute
	Get_Enum_String
	Map_ADI_Write_Area
	Map_ADI_Read_Area

5.4.4. Object Attributes (Instance #0)

#	Name	Access	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

5.4.5. Instance Attributes (Instance #1)

Basic

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.

#	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. (Consult the general Anybus CompactCom 40 Software Design Guide for further information.)	
6	Read process data size	Get	UINT16	Current read process data size (in bytes) Updated on every successful Map_ADI_Read_Area. (Consult the general Anybus CompactCom 40 Software Design Guide for further information.)	
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.

5.5. Network Configuration Object (04h)

5.5.1. Category

Extended

5.5.2. 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 \(page 7\)](#)
- [Identity Object \(01h\) \(page 13\)](#)

5.5.3. Supported Commands

Object:	Get_Attribute Reset
Instance:	Get_Attribute Set_Attribute Get_Enum_String

5.5.4. 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	15
4	Highest instance number	Get	UINT16	19

5.5.5. Instance Attributes (Instance #1, Node Address)

A Get command always returns the actual value. If an invalid value is assigned to attribute #5 (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.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"Node address" (Multilingual, see page Multilingual Strings (page 42))
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	Get/Set	Array of UINT8	Any change is valid after reset. Node address range: 0-63 default: 63
6	Configured Value	Get	Array of UINT8	Holds the configured value, which will be written to attribute #5. Node address range: 0-63 default: 63

5.5.6. Instance Attributes (Instance #2, Baud rate)

Value is used after module reset.

A Get command always returns the actual value. If an invalid value is assigned to attribute #5 (i.e. using a Set command), the module will accept the 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 the switches, the latest valid value will be used when going online.

#	Name	Access	Data Type	Description	
1	Name	Get	Array of CHAR	“Baud rate” (Multilingual, see page Multilingual Strings (page 42))	
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	Get/Set	ENUM	<u>Value:</u>	<u>Enum string :</u>
				00h	125 kbps
				01h	250 kbps
				02h	500 kbps
				03h	Autobaud (default)
6	Configured Value	Get	ENUM	Holds the configured value, which will be written to attribute #5 after the module has been reset.	
				<u>Value:</u>	<u>Enum string :</u>
				00h	125 kbps
				01h	250 kbps
				02h	500 kbps
				03h	Autobaud (default)
				Default: 04h (invalid value)	

5.5.7. 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 \(FCh\) \(page 54\)](#)
- [Anybus State Machine \(page 60\)](#)

#	Name	Access	Data 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	<u>Value:</u>	<u>Meaning:</u>
				00h	Disable (Default)
				01h	Enable
6	Configured Value	Get/Set	BOOL	Holds the configured value, which will be written to attribute #5 after the module has been reset.	
				<u>Value:</u>	<u>Meaning:</u>
				00h	Disable (Default)
				01h	Enable

5.5.8. Multilingual Strings

The instance names and enumeration strings in this object are multilingual, 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

5.6. Anybus File System Interface Object (0Ah)

5.6.1. Category

Extended

5.6.2. 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.

5.6.3. Supported Commands

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

5.6.4. Object Attributes (Instance #0)

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

5.6.5. Instance Attributes (Instance #1)

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

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:

- Assembly Mapping Object (EBh) - (see Anybus CompactCom 40 Software Design Guide)
- Modular Device Object (ECh, see Anybus CompactCom 40 Software Design Guide)
- [Energy Reporting Object \(E7h\) \(page 46\)](#)
- [SYNC Object \(EEh\) \(page 47\)](#)
- [Energy Control Object \(F0h\) \(page 48\)](#)
- Application Data Object (FEh, see Anybus CompactCom 40 Software Design Guide)
- Application Object (FFh, see Anybus CompactCom 40 Software Design Guide)

Network Specific Objects:

- [CIP Identity Host Object \(EDh\) \(page 45\)](#)
- [DeviceNet Host Object \(FCh\) \(page 54\)](#)

6.2. CIP Identity Host Object (EDh)

6.2.1. Category

Extended

6.2.2. 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 #1 in the CIP identity host object will be mapped to instance #2 in the CIP identity object, and so on.

6.2.3. Supported Commands

Object: Get_Attribute

Instance: Get_Attribute
Get_Attribute_All

6.2.4. 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

6.2.5. Instance Attributes (Instance #1)

Extended

The instance attributes values replace the default values for the CIP Identity object.

#	Name	Access	Type	Value
1	Vendor ID	Get	UINT16	-
2	Device Type	Get	UINT16	-
3	Product Code	Get	UINT16	-
4	Revision	Get	Struct of UINT 8, UINT 8	(Major Revision, Minor Revision)
5	Status	Get	UINT16	-
6	Serial Number	Get	UINT32	-
7	Product Name	Get	Array of CHAR	-

6.3. Energy Reporting Object (E7h)

6.3.1. Category

Extended

6.3.2. Object Description

Using this object, the host application has a standardized way of reporting its energy consumed or produced. The reporting capabilities of this object are limited. On networks providing more elaborate reporting functionality, the reporting functionality will have to be implemented in a transparent manner by the application.

6.3.3. Supported Commands

Object: Get_Attribute

Instance: Get_Attribute

6.3.4. Object Attributes (Instance #0)

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

6.3.5. Instance Attributes (Instance #1)

Extended

#	Name	Access	Type	Description	
1	Energy Reading	Get	Struct of: UINT32 UINT32	Amount of energy (Wh) consumed or produced by the application. Stored in nonvolatile memory. The first UINT32 represents the lower part of the Energy Reading, the second UINT32 represents the higher part of the Energy Reading	
2	Direction	Get	BOOL	Indicates if the host is consuming or producing energy.	
				Value:	Meaning:
				0:	Producing
				1:	Consuming
3	Accuracy	Get	UINT16	Accuracy in 0.01% of reading 0: Unknown	
4	Current Power Consumption	Get	UINT16	The current power consumption in 0.01% of the Nominal Power consumption	
5	Nominal Current Consumption	Get	UINT32	The nominal power consumption in mW	

6.4. SYNC Object (EEh)

6.4.1. Category

Extended

6.4.2. 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 (EEh)”

6.4.3. Supported Commands

Object: Get_Attribute

Instance: Get_Attribute
Set_Attribute

6.4.4. Object Attributes (Instance #0)

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

6.4.5. 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) (page 21) .
2-8	(not implemented)				

6.5. Energy Control Object (F0h)

6.5.1. Category

Extended

6.5.2. Object Description

This object implements energy control functionality, i.e. energy specific settings, in the host application. The implementation of this object is optional. All instance attributes shall be seen as required and must be implemented in the application. If the Anybus module detects that an attribute is missing during run time an appropriate network error is sent and the Discard Responses counter is increased in the Anybus Object instance attribute Error Counter.

Each enabled instance in the object corresponds to an Energy saving mode. The number of available modes is device specific, and must be defined by the application. The higher the instance number, the more energy is saved. The instance with the highest number always corresponds to the “Power off” mode, i.e. the state where the device is essentially shut down. Instance 1 of the object represents “Ready to operate”, i.e. the mode where the device is fully functional and does not save energy at all. Consequently a meaningful implementation always contains at least two instances, one for energy saving and one for operating. If this object is implemented for PROFINET, at least three instances are needed: “Ready to operate”, “Energy saving mode 1”, and “Power off”.

Highest number of instances is 8. Please note that these modes are always present – they are not dynamically created or deleted. It is not allowed to leave holes in the list of instances.

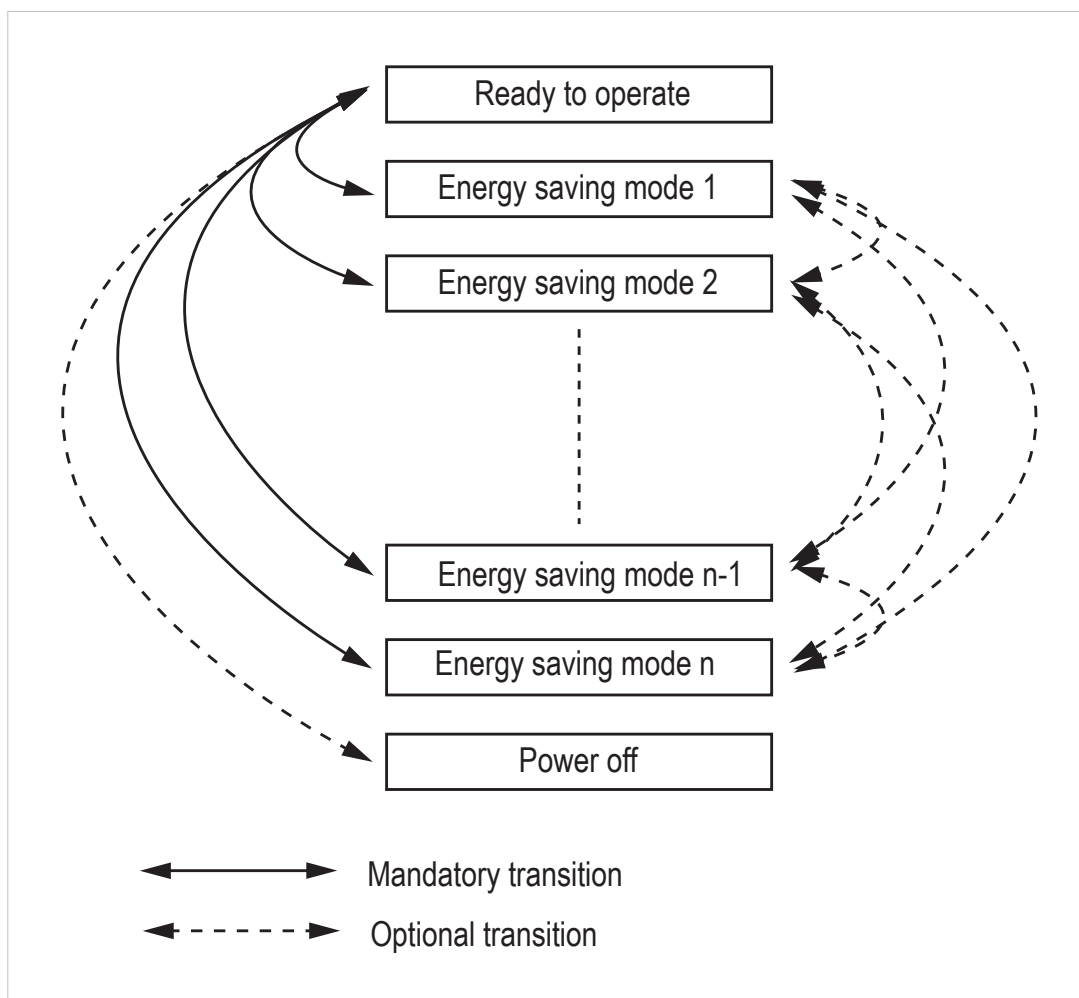


Figure 1.

6.5.3. Supported Commands

Object:

- Get_Attribute
- StartPause
- EndPause
- Preview_Pause_Time (not PROFINET)

Instance: Get_Attribute

6.5.4. Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1	Name	Get	Array of CHAR	"Energy Control"
2	Revision	Get	UINT8	02h
3	Number of instances	Get	UINT16	-
4	Highest instance no.	Get	UINT16	Highest created instance number. Maximum value is 8.
11	Current Energy Saving Mode	Get	UINT16	Instance number of the currently used Energy saving mode. During a mode transition the new Energy saving mode shall be presented. "Ready to operate" will equal instance #1, and "Power off" mode will equal Highest instance number.
12	Remaining time to destination	Get	UINT32	When changing mode this parameter will reflect the actual time (in milliseconds) remaining until the shift is completed. If a dynamic value cannot be generated the static value for the transition from the source to destination mode shall be used. If the value is infinite, or unknown, the maximum value 0xFFFFFFFF shall be used.
13	Energy consumption to destination	Get	FLOAT	When changing mode this parameter will reflect the actual energy (in kWh) which will be consumed until the shift is completed. If a dynamic value cannot be generated the static value for the transition from the source to destination mode shall be used. If the value is undefined the value 0.0 shall be used.
14	Transition to "Power off" mode supported	Get	BOOL	Indicates whether transition to "Power off" mode is supported or not.
				0: Not supported
				1: Supported

6.5.5. Instance Attributes (Instance #1 - #8)

#	Name	Access	Data Type	Description	
1	ModeAttributes	Get	BITS16	Bit 0:	Meaning:
				0:	Only static time and energy values are available (Value of bit 0 attribute is not implemented)
				1:	Dynamic time and energy values are available
				Bit 1-15:	Reserved
2	TimeMinPause	Get	UINT32	Minimum pause time in milliseconds. (t_{pause}) If the value is infinite, or unknown, the maximum value 0xFFFFFFFF shall be used.	
3	TimeToPause	Get	UINT32	Maximum time to go to this Energy saving mode.(ms, t_{off}) If the value is infinite, or unknown, the maximum value 0xFFFFFFFF shall be used.	
4	TimeToOperate	Get	UINT32	Maximum time needed to go to the “Ready to operate” mode. (ms, t_{on}) If the value is infinite, or unknown, the maximum value 0xFFFFFFFF shall be used.	
5	TimeMinLengthOfStay	Get	UINT32	The minimum time that the device must stay in this mode. In milliseconds.(ms, $t_{\text{off_min}}$) If the value is infinite, or unknown, the maximum value 0xFFFFFFFF shall be used.	
6	TimeMaxLengthOfStay	Get	UINT32	Maximum time that is allowed to stay in this mode. In milliseconds. If no maximum value is available or if not implemented, the maximum value FFFFFFFFh shall be used.	
7	ModePowerConsumption	Get	FLOAT	Amount of power consumed in this mode. (kW) If the value is undefined the value 0.0 shall be used.	
8	EnergyConsumptionToPause	Get	FLOAT	Amount of energy required to go to this mode. (kWh) If the value is undefined the value 0.0 shall be used.	
9	EnergyConsumptionToOperate	Get	FLOAT	Amount of energy required to go to the “Ready to operate” mode from this mode. (kWh) If the value is undefined the value 0.0 shall be used.	
10	Availability	Get	BOOL	Indicates if this energy saving mode is available given the current device state. Not used for PROFINET.	
				False	Not available
				True	Available (Value if attribute not implemented)
11	Power Consumption	Get	UINT32	Indicates the power consumption of the device when in this state. Not used for PROFINET.	

Command Details: Start_Pause

Details

Command Code: 10h

Valid for: Object

Description

This command is sent to the host application when the system wants to initialize a pause of the system. The length of the pause is specified in milliseconds. The response of the message contains the destination mode (i.e. the instance number of the selected energy saving mode).

- Command Details

Field	Contents	Comments
Data[0]	Pause time (low word, low byte)	Pause time (ms)
Data[1]	Pause time (low word, high byte)	
Data[2]	Pause time (high word, low byte)	
Data[3]	Pause time (high word, high byte)	

- Response Details

Field	Contents	Comments
Data[0]	Instance number (low byte)	Instance number of selected Energy mode
Data[1]	Instance number (low byte)	

If the application is unable to select a state, given the requested pause time, it shall return one of the error codes in the table below.

#	Error code	Description
0x0D	Invalid state	Given the state of the device and the requested pause time it is currently not possible to enter any energy saving mode
0x12	Value too low	The requested pause time is too short

Command Details: End_Pause

Details

Command Code: 11h

Valid for: Object

Description

This command is sent to the host application when the system wants to return the system from a pause mode back to "Ready to operate" mode. In the response message the number of milliseconds to actualize the switch is returned.

- Command Details
(none)
- Response Details

Field	Contents	Comments
Data[0]	Time To Operate (low word, low byte)	Time needed to switch to "Ready to operate"
Data[1]	Time To Operate (low word, high byte)	
Data[2]	Time To Operate (high word, low byte)	
Data[3]	Time To Operate (high word, high byte)	

If the application is unable to end the pause it shall return the error code in the table below.

#	Error code	Description
0x0D	Invalid state	Given the state of the device, it is currently not possible to end the pause

Command Details: Preview_Pause_Time**Details**

Command Code: 12h

Valid for: Object

Description

Not used for PROFINET devices.

This command is sent to the host application when the system wants to preview the application's choice of Energy saving mode. The length of the pause is specified in milliseconds. The response shall contain the destination mode the application would have chosen if the StartPause service was sent (that is, the instance number of the selected energy saving mode). No transition to an Energy saving mode occurs.

- Command Details

Field	Contents	Comments
Data[0]	Pause time (low word, low byte)	Pause time (ms)
Data[1]	Pause time (low word, high byte)	
Data[2]	Pause time (high word, low byte)	
Data[3]	Pause time (high word, high byte)	

- Response Details

Field	Contents	Comments
Data[0]	Instance number (low byte)	Instance number of selected Energy mode
Data[1]	Instance number (low byte)	

If the application is unable to select a state, given the requested pause time, it shall return one of the error codes in the table below.

#	Error code	Description
0x0D	Invalid state	Given the state of the device and the requested pause time it is currently not possible to enter any energy saving mode
0x12	Value too low	The requested pause time is too short

6.6. DeviceNet Host Object (FCh)

6.6.1. Category

Basic, extended

6.6.2. 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 startup; 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 an attribute is not implemented in the host application, at all times respond with an error message (06h, “Invalid CmdExt[0]”)

See also...

- [Identity Object \(01h\) \(page 13\)](#)
- 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 EDS-file supplied by HMS.

6.6.3. Supported Commands

Object: Get_Attribute
 Process_CIP_Message_Request, see [CIP Request Forwarding \(page 62\)](#)

Instance: Get_Attribute

6.6.4. Object Attributes (Instance #0)

#	Name	Access	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

6.6.5. 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 Instance 64h Attributes (Producing Instance) in Assembly Object (04h) (page 19) (CIP)
8	Consuming Instance No.	Get	UINT16	0096h	See Instance 96h Attributes (Consuming Instance) in Assembly Object (04h) (page 19) (CIP)
9	Enable Address from Net	Get	BOOL	True	Value:
					Meaning:
					True
					Can be set from network
10	Enable Baud Rate from Net	Get	BOOL	True	False
					Cannot be set from network
					See Identity Object (01h) (page 13) (CIP)
11	Enable CIP forwarding	Get	BOOL	False	Value:
					Meaning:
					True
					Enable CIP forwarding
12	Enable Parameter Object	Get	BOOL	True	False
					Disable CIP forwarding
					See CIP Request Forwarding (page 62)
13	Enable Quick Connect	Get	BOOL	False	Value:
					Meaning:
					True
					Enable Quick Connect
20	Anybus CompactCom ADI Object Number	Get	UINT16	00A2h	False
					Disable Quick Connect
					See DeviceNet Object (03h) (page 17) (CIP)
21	Producing instance number list	Get	Array of UINT16		Value:
					Meaning:
					True
					Enable CIP Parameter Object
22	Consuming instance number list	Get	Array of UINT16		False
					Disable CIP Parameter Object
					See Parameter Object (0Fh) (page 26) (CIP)
23	Enable Group 2 Only mode	Get	BOOL	False	Value:
					Meaning:
					True
					Enable Quick Connect

Multiple Assembly Instances

The Anybus Assembly Mapping object has two arrays on class level (object attributes #11 Write PD Instance List and #12 Read PD Instance List) listing present instances defined by the application. The arrays of the DeviceNet Host Object instance attributes #21 and #22 (Producing instance number list and Consuming instance number list) are bound to the instance lists in the Anybus Assembly Mapping object. They list the corresponding CIP instance number to each assembly instance defined by the application. If the arrays do not match, the Anybus will enter the state EXCEPTION.

The example below shows how the mapping of assembly instances defined by the application maps to the CIP instance numbers listed in attribute 21 and 22. The first table lists a set of instances defined by the application in the Anybus Assembly Mapping object. The instances are listed in the class attributes #11 or #12 of the ABCC Assembly Mapping object. The next table shows how the assembly instances in the Anybus Assembly Mapping object are mapped to a CIP instance number listed in either attribute #21 or #22.

Assembly Mapping Object (EBh) Instances	
1	Read PD
2	Read PD
10	Write PD
11	Write PD
100	Read PD
101	Write PD

Assembly Mapping Object Attribute	Value		Value	ABCC EtherNet/IP object instance attributes
11 – Write PD Instance List	1	<->	70	21 – Producing Instance Number List
	2	<->	71	
	100	<->	150	
12 – Read PD Instance List	10	<->	20	22 – Consuming Instance Number List
	11	<->	21	
	101	<->	100	

6.6.6. Command Details: Process_CIP_Message_Request

Category

Extended

Details

Command Code: 10h

Valid for: Object Instance

Description

By setting the attribute Enable CIP Request Forwarding (#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 command Process_CIP_Message_Request in the Anybus CompactCom 40 EtherNet/IP.

See also...

- [Device Customization \(page 6\)](#)
- [CIP Request Forwarding \(page 62\)](#)

Appendix 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.

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.

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.

Appendix B. Implementation Details

1. DeviceNet Implementation

1.1. 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 Anybus State Machine (page 60).

1.2. 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.
--------------------------------	--

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

3. SUP-Bit Definition

The supervised bit (SUP) indicates that the network participation is supervised by another network device. The SUP bit reflects the Module Owned bit in the Device Status instance attribute of the CIP Identity Object (01h). For more information, see [Identity Object \(01h\) \(page 13\)](#) (CIP Object).

Appendix 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, 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 Instance
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)

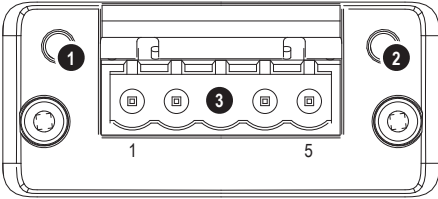
If the host application did not recognize the CIP request, i.e. the CIP object in question is not implemented in the host application, the following response shall be sent to the module.

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 Instance
Dest. Instance (msb)	00h								
(command/error)	0								This message is not an Anybus CompactCom error message. (If this bit is set (1), an Anybus CompactCom error has occurred and an Anybus CompactCom error code is returned.)
(command/response)		0							
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)

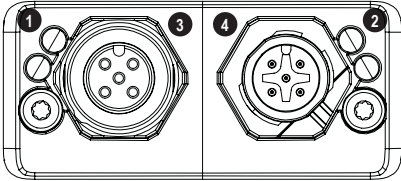
Appendix D. Technical Specification

1. Front View

1.1. Front View, DeviceNet Connector

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

1.2. Front View, M12 Connectors

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

1.3. Network Status

LED 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

1.4. Module Status

LED 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

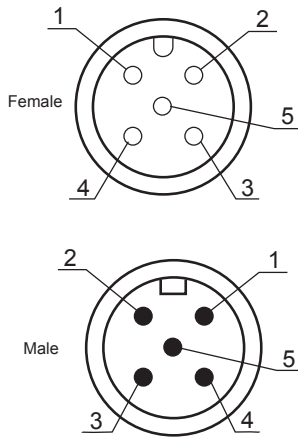
1.5. DeviceNet Connector

This connector provides DeviceNet connectivity.

Pin	Signal	Description
1	V-	Negative bus supply voltage (DeviceNet bus power)
2	CAN_L	CAN low bus line
3	SHIELD	Cable shield
4	CAN_H	CAN high bus line
5	V+	Positive bus supply voltage (DeviceNet bus power)

1.6. 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, DeviceNet bus power. 11–25 VDC	
3	V-	Ground, DeviceNet bus power	
4	CAN_H	CAN high	
5	CAN_L	CAN low	

2. Functional Earth (FE) Requirements

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

tHMS Industrial Networks does not guarantee proper EMC behavior unless these FE requirements are fulfilled.

3. Power Supplies

3.1. Supply Voltage

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

3.2. 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 ($12 \times 750 \text{ mA} = 9 \text{ A}$).

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

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.

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.

4.1. 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

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

4.2. 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.

5. Environmental Specification

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

6. EMC Compliance

6.1. Environmental Specification

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

Appendix E. Backward Compatibility

The Anybus CompactCom M40 series of industrial network modules have significantly better performance and include more functionality than the modules in the Anybus CompactCom 30 series. The 40 series is backward compatible with the 30 series in that an application developed for the 30 series should be possible to use with the 40 series, without any major changes. Also it is possible to mix 30 and 40 series modules in the same application.

This appendix presents the backwards compatibility issues that have to be considered for Anybus CompactCom 40 DeviceNet, when designing with both series in one application, or when adapting a 30 series application for the 40 series.

1. Initial Considerations

There are two options to consider when starting the work to modify a host application developed for Anybus CompactCom 30-series modules to also be compatible with the 40-series modules:

- Add support with as little work as possible i.e. reuse as much as possible of the current design.
 - This is the fastest and easiest solution but with the drawback that many of the new features available in the 40-series will not be enabled (e.g. enhanced and faster communication interfaces, larger memory areas, and faster communication protocols).
 - You have to check the hardware and software differences below to make sure the host application is compatible with the 40-series modules. Small modifications to your current design may be needed.
- Make a redesign and take advantage of all new features presented in the 40-series.
 - A new driver and host application example code are available at www.anybus.com/starterkit40 to support the new communication protocol. This driver supports both 30-series and 40-series modules.
 - You have to check the hardware differences below and make sure the host application is compatible with the 40-series modules.

**NOTE**

This information only deals with differences between the 30-series and the 40-series.

Link to support page: www.anybus.com/support.

2. Hardware Compatibility

Anybus CompactCom is available in three hardware formats; Module, Chip, and Brick.

2.1. Module

The modules in the 30-series and the 40-series share physical characteristics, like dimensions, outline, connectors, LED indicators, mounting parts etc. They are also available as modules without housing.



Figure E.1. Anybus CompactCom M30/M40

2.2. Chip

The chip (C30/C40) versions of the Anybus CompactCom differ completely when it comes to physical dimensions.



IMPORTANT

There is no way to migrate a chip solution from the 30-series to the 40-series without a major hardware update.

2.3. Brick

The Anybus CompactCom B40-1 does not share dimensions with the Anybus CompactCom B30. The B40-1 is thus not suitable for migration. However HMS Industrial Networks has developed a separate brick version in the 40-series, that can be used for migration. This product, B40-2, shares dimensions etc. with the B30. Please contact HMS Industrial Networks for more information on the Anybus CompactCom B40-2.

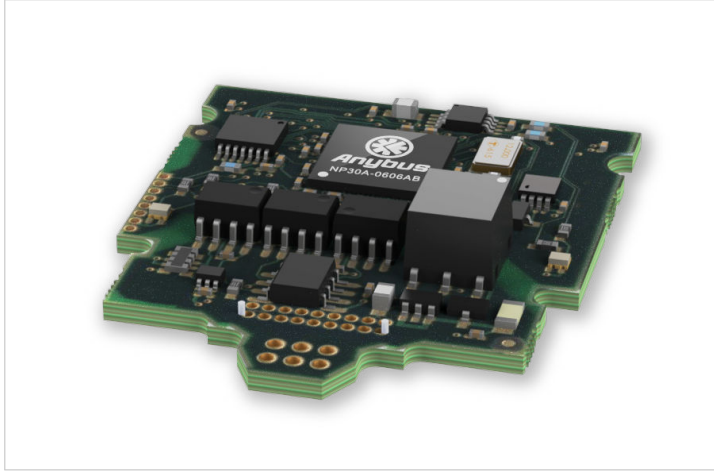


Figure E.2. Anybus CompactCom B30

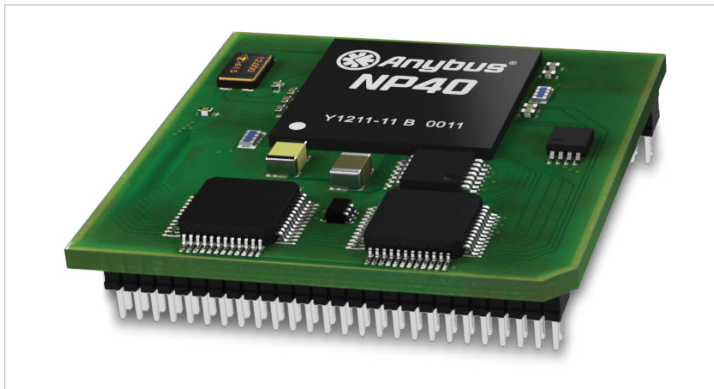


Figure E.3. Anybus CompactCom B40-1 (not for migration)

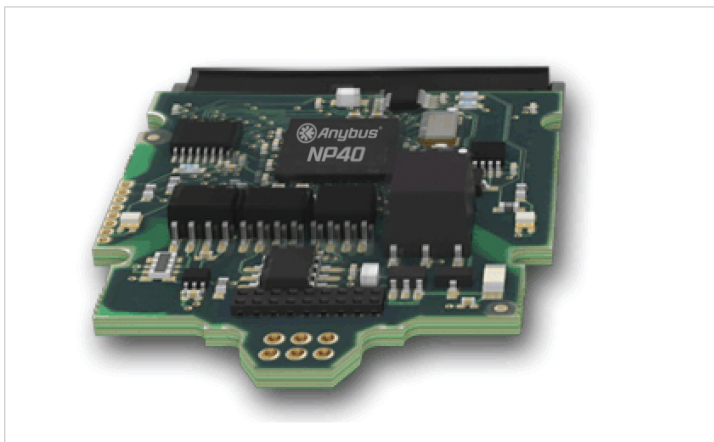


Figure E.4. Anybus CompactCom B40-2

2.4. Host Application Interface

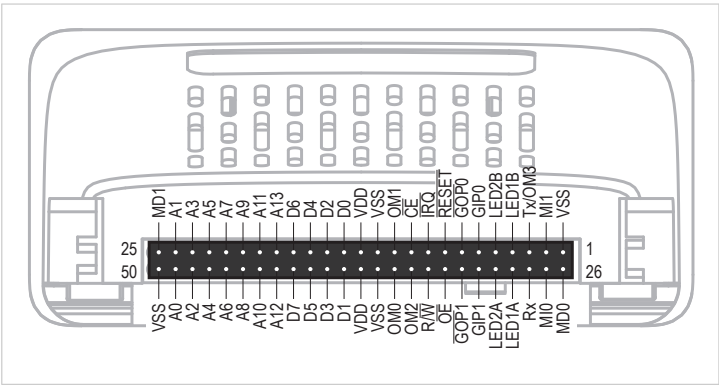


Figure E.5.

Some signals in the host application interface have modified functionality and/or functions which must be checked for compatibility. See the following sections.

Tx/OM3

In the 30-series, this pin is only used for Tx. It is tri-stated during power up, and driven by the Anybus CompactCom UART after initialization. In the 40-series this pin is used as a fourth operating mode setting pin (OM3). During startup after releasing the reset, this pin is read to determine the operating mode to use. The pin is then changed to a Tx output.

In the 40-series, this pin has a built-in weak pull-up. If this pin, on a 30-series module or brick is unconnected, pulled high, or connected to a high-Z digital input on the host processor, it will be compatible with the 40-series. An external pull-up is recommended, but not required.

**IMPORTANT**

If this pin is pulled low by the host during startup in a 30-series application, any 40-series module or brick, substituted in the application, will not enter the expected operating mode.

Related Information: Anybus CompactCom M40 Hardware Design Guide (HMSI-216-126), Section “Application Connector Pin Overview”

Module Identification (MI[0..1])

These pins are used by the host application (i.e. your product) to identify what type of Anybus CompactCom that is mounted. The identification differs between the 30-series and the 40-series.

**NOTE**

If your software use this identification you need to handle the new identification value.

MI1	MI0	Module Type
LOW	LOW	Active Anybus CompactCom 30
HIGH	LOW	Active Anybus CompactCom 40

MI[0..1] shall only be sampled by the application during the time period from power up to the end of SETUP state. The pins are low at power up and before reset release.

Related Information: Anybus CompactCom M40 Hardware Design Guide (HMSI-216-126), Section “Settings/ Sync”.

GIP[0..1]/LED3[A..B]

These pins are tri-stated inputs by default in the 30-series. In the 40-series, these pins are tri-stated until the state NW_INIT. After that they become open-drain, active low LED outputs (LED3A/LED3B).

No modification of the hardware is needed, if your current design has

- tied these pins to GND
- pulled up the pins
- pulled down the pins
- left the pins unconnected

However, if the application drive the pins high, a short circuit will occur.

If you connect the pins to LEDs, a pull-up is required.

In the 40-series, there is a possibility to set the GIP[0..1] and GOP[0..1] in high impedance state (tri-state) by using attribute #16 (GPIO configuration) in the Anybus object (01h). I.e. if it is not possible to change the host application hardware, this attribute can be configured for high impedance state of GIP and GOP before leaving NW_INIT state.

Related Information: Anybus CompactCom M40 Hardware Design Guide (HMSI-216-126), Section “LED Interface/D8-D15 (Data Bus)”.

GOP[0..1]/LED4[A..B]

These pins are outputs (high state) by default in the 30-series. In the 40-series, these pins are tri-stated until the state NW_INIT, and after that they become push-pull, active low LED outputs (LED4A/LED4B).

This change should not affect your product.

Related Information: Anybus CompactCom M40 Hardware Design Guide (HMSI-216-126), Section 3.2.3, “LED Interface/D8-D15 (Data Bus)”.

Address Pins A[11..13]

The address pins 11, 12, and 13 are ignored by the 30-series. These pins must be high when accessing the 40-series module in backwards compatible 8-bit parallel mode. If you have left these pins unconnected or connected to GND, you need to make a hardware modification to tie them high.

Max Input Signal Level (V_{IH})

The max input signal level for the 30-series is specified as $V_{IH}=V_{DD}+0,2\text{ V}$, and for the 40-series as $V_{IH}=3.45\text{ V}$. Make sure that you do not exceed 3.45 V for a logic high level.

RMII Compatibility

If the RMII mode is being used on an Anybus CompactCom 40 module and it is desired to remain compatible with the 30 series, it is important to disable this connection when switching to an Anybus CompactCom 30 module due to pin conflicts. The RMII port of the host processor should be set to tristate by default, and only be enabled if an RMII capable Anybus CompactCom 40 is detected. In case the RMII connection cannot be disabled through an internal hardware control on the host processor, it will be necessary to design in external hardware (i.e. a FET bus switch) to prevent short circuits

Related Information: Anybus CompactCom M40 Hardware Design Guide (HMSI-216-126), Section 3.2.5, “RMII — Reduced Media-Independent Interface”.

3. General Software

3.1. Extended Memory Areas

The memory areas have been extended in the 40-series, and it is now possible to access larger sizes of process data (up to 4096 bytes instead of former maximum 256 bytes) and message data (up to 1524 bytes instead of former maximum 255 bytes). The 30-series has reserved memory ranges that the application should not use. The 40-series implements new functionality in some of these memory areas.



NOTE

To use the extended memory areas you need to implement a new communication protocol which is not part of this document.

Memory areas not supported by the specific network cannot be used. Make sure you do not access these areas, e.g. for doing read/write memory tests.

Related Information: Anybus CompactCom 40 Software Design Guide (HMSI-216-125), Section “Memory Map”

3.2. Faster Ping-Pong Protocol

The ping-pong protocol (the protocol used in the 30-series) is faster in the 40-series. A 30-series module typically responds to a so called ping within 10-100 µs. The 40-series typically responds to a ping within 2 µs.

Interrupt-driven applications (parallel operating mode) may see increased CPU load due to the increased speed.

3.3. Requests from Anybus CompactCom to Host Application During Startup

All requests to software objects in the host application must be handled and responded to (even if the object does not exist). This applies for both the 30-series and the 40-series. The 40-series introduces additional objects for new functionality.

There may also be additional commands in existing objects added to the 40-series that must be responded to (even if it is not supported).

If your implementation already responds to all commands it cannot process, which is the expected behavior, you do not need to change anything.

3.4. Anybus Object (01h)

Attribute	30-series	40-series	Change/Action/Comment
#1, Module Type	0401h	0403h	Make sure the host application accepts the new module type value for the 40-series.
#15, Auxiliary Bit	Available	Removed	It is not possible to turn off the “Changed Data Indication” in the 40-series. Also see “Control Register CTRL_AUX-bit” and “Status Register STAT_AUX-bit” below.
#16, GPIO Configuration	Default: General input and output pins	Default: LED3 and LED4 outputs	See also .. • GIP[0..1]/LED3[A..B] (page 71) • GOP[0..1]/LED4[A..B] (page 71)

3.5. Control Register CTRL_AUX-bit

30-series The CTRL_AUX bit in the control register indicates to the Anybus CompactCom if the process data in the current telegram has changed compared to the previous one.

40-series The value of the CTRL_AUX bit is always ignored. Process data is always accepted.

All released Anybus CompactCom 30 example drivers from Anybus CompactCom comply with this difference.

Related Information: Anybus CompactCom 40 Software Design Guide (HMSI-216-125), section “Control Register”.

3.6. Status Register STAT_AUX-bit

30-series	The STAT_AUX bit in the status register indicates if the output process data in the current telegram has changed compared to the previous one. This functionality must be enabled in the Anybus object (01h), Attribute #15. By default, the STAT_AUX bit functionality is disabled.
40-series	The STAT_AUX bit indicates updated output process data (not necessarily changed data) from the network compared to the previous telegram. The functionality is always enabled.

All released Anybus CompactCom 30 example drivers from HMS Industrial Networks comply with this difference.

Related Information: Anybus CompactCom 40 Software Design Guide (HMSI-216-125), section “Status Register”.

3.7. Control Register CTRL_R-bit

30-series	The application may change this bit at any time.
40-series	For the 8-bit parallel operating mode, the bit is only allowed to transition from 1 to 0 when the STAT_M-bit is set in the status register. When using the serial operating modes, it is also allowed to transition from 1 to 0 in the telegram immediately after the finalizing empty fragment.

All released Anybus CompactCom 30 example drivers from HMS Industrial Networks comply with this difference.

Related Information: Anybus CompactCom 40 Software Design Guide (HMSI-216-125), section “Control Register”.

3.8. Modifications of Status Register, Process Data Read Area, and Message Data Read Area

In the 40-series, the Status Register, the Process Data Read Area, and the Message Data Read Area are write protected in hardware (parallel interface). If the software for some reason writes to any of those areas, a change is needed.

All released Anybus CompactCom 30 example drivers from HMS Industrial Networks comply with this difference.

4. Network Specific — DeviceNet

4.1. DeviceNet Host Object (FCh)

Attribute	30-series	40-series	Change/Action/Comment
#2, Device Type	Default: 0000h	Default: 002Bh	If the attribute is implemented in the host application, it overrides the default value and there is no difference between the 30-series and the 40-series. If the attribute is not implemented, the default value is used.
#3, Product Code	Default: 0062h	Default: 003Fh	If the attribute is implemented in the host application, it overrides the default value and there is no difference between the 30-series and the 40-series. If the attribute is not implemented, the default value is used.
#6, Product Name	Default: “Anybus-CC DeviceNet”	Default: “CompactCom 40 DeviceNet(TM)”	If the attribute is implemented in the host application, it overrides the default value and there is no difference between the 30-series and the 40-series. If the attribute is not implemented, the default value is used.

4.2. EDS file (Electronic Datasheet file used by configuration tool)

Keywords

The following keywords must be updated when migrating.

Keyword	Comments
ProdType	Must match attribute #2 (Device Type) in the DeviceNet Host Object (FCh).
ProdCode	Must match attribute #3 (Product Code) in the DeviceNet Host Object (FCh).
ProdName	Must match attribute #6 (Product Name) in the DeviceNet Host Object (FCh).
MajRev	Must match the major revision of the product.