Anybus® CompactCom™ 40

CANopen

NETWORK GUIDE

SCM-1202-108 1.2 en-US ENGLISH

Important User Information

Disclaimer

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

HMS Industrial 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 Industrial Networks and is subject to change without notice. HMS Industrial 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 Industrial 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 Industrial 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.

Table of Contents

Page

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 CANopen. The document describes the features that are specific to Anybus CompactCom 40 CANopen. 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 CANopen specific functionality is to be used, in-depth knowledge of CANopen networking internals and/or information from the official CANopen specifications may be required. In such cases, the persons responsible for the implementation of this product should either obtain the CANopen 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 www.anybus.com/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

1.3 Document History

Version	Date	Description
1.0	2019-01-30	First release
1.1	2019-03-06	Added Anybus data type DOUBLE Timing and performance updated
1.2	2021-09-09	Minor corrections

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, p. 3*

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.

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.

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 Trademark Information

2 About the Anybus CompactCom 40 CANopen

2.1 General

The Anybus CompactCom 40 CANopen communication module provides instant CANopen 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.

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

The functionality of the module is described in two categories: Basic and Extended, see *Categorization of Functionality, p. 29*.

2.2 Features

- CiA® 301 version 4.2.0 compliant
- Supports all standard baud rates
- Automatic baud rate detection
- Supports LSS
- Customizable Identity Information
- Up to 64 TPDO’s & 64 RPDO’s (Corresponds to a total of 512 bytes of Process Data in each direction)
- PDO mapping can be customized via network configuration tool or via application (IO assemblies)
- Up to 57343 ADIs can be accessed from the network as Manufacturer Specific Objects and profile specific objects.
- Diagnostic support
- Heartbeat functionality supported (Node Guarding not supported)
- Supports Expedited- and Segmented SDO Transfer (Block Transfer not supported)

3 Basic Operation

3.1 General Information

3.1.1 Software Requirements

Generally, no additional network support code needs to be written to support the Anybus CompactCom 40 CANopen, however due to the nature of the CANopen networking system certain things must be taken into account:

- Only ADIs with instance numbers less than 57343 can be accessed from the network.
- Only ADI elements 0...253 can be accessed from the network.
- To be able to initialize the Read Process Data properly, ADIs mapped as Read Process Data must have both Set and Get access. If not, the startup value for their data will be zero.
- The SDO timeout that is set in the bus configuration tool, must be considered when reading or writing to the objects 1010h or 1011h in the object dictionary. When writing to the object 1010h there is at least 1 second until the module will respond, corresponding to the time it takes to store the parameters in the module. The timeout must also be considered when running the Conformance Test Tool.

For further information about the Anybus CompactCom software interface, consult the general Anybus-CompactCom 40 Software Design Guide.

3.1.2 Electronic Data Sheet (EDS)

Each device on CANopen is associated with an Electronic Data Sheet (an EDS file), which holds a description of the device and its functions. Most importantly, the file describes the object dictionary implementation in the module.

HMS Networks AB supplies a generic EDS file which can serve as a basis for new implementations; however this file must be altered to match the end product (i.e. the ADI and process data configuration, identity settings etc.). All Anybus CompactCom ADIs must be described as specified in the CANopen standard “DS306 Electronic data sheet specification for CANopen” (can be requested from the CiA home page, www.can-cia.org). All application specific objects start from index 2001h, but all ADIs should have a descriptive name in the EDS file, that corresponds to the name in the application.

To verify the EDS-file, download and run the EDS-file checker program from www.can-cia.org.

See also...

- *Fieldbus Conformance Notes, p. 36*

3.1.3 Device Identity

Generic Implementation

In a generic implementation (i.e. no network specific support is implemented) the module will appear as a generic HMS Networks AB device with the following identity information:

Object Entry	Value
Vendor ID	0000 001Bh (HMS Industrial Networks)
Product Code	0000 000Dh (Anybus CompactCom 40)
Manufacturer Device Name	“Anybus CompactCom 40 CANopen”
Manufacturer Hardware Revision	(Hardware FuncID of the Anybus hardware)
Manufacturer Software Revision	(Anybus software revision)

> **!** It is strongly recommended not to use the standard HMS Industrial Networks AB identity.

Vendor Specific Implementation

By implementing support for the CANopen Object (FBh), the module can be customized to appear as a vendor specific implementation rather than a generic Anybus CompactCom device.

> **!** It is strongly recommended not to use the standard HMS Industrial Networks AB implementation.

See also ...

- *CANopen Object (FBh), p. 26*

3.2 Data Exchange

3.2.1 Application Data (ADI)

Application Data Instances (ADIs) can be accessed from the network via dedicated object entries in the object dictionary (2001h-FFFFh).

> **i** *The EDS file must match the actual ADI implementation in the host application.*

3.2.2 Process Data

ADIs mapped as Process Data can be exchanged cyclically as Process Data Objects (PDOs) on the bus. The actual PDO map is based on the Process Data map specified during startup or how the application is implemented. It can be changed from the network during runtime, if the application has implemented the remap commands in the Application Data Object.

The module supports up to 64 TPDOs and up to 64 RPDOs, each capable of carrying up to 8 bytes of data. Each subindex on a PDO corresponds to one process data mapped ADI element (i.e. mapping multiple element ADIs will result in multiple sub-indices on the PDO).

To gain in configurability, the Assembly Mapping Object can be used to remap and replace the Process Data map specified at startup. Each PDO will be represented by an instance in the Assembly Mapping Object. The PDOs will then be remapped when the Anybus CompactCom receives the NMT command Start Node.

> **i** *The EDS file must match the actual Process Data implementation in the host application.*

3.3 Node Address & Data Rate Configuration

3.3.1 General

The CANopen data rate and node address can be set by the host application using Network Configuration Object (04h). Note that in order to ensure network compliance, the recommendations stated for this object in the Anybus CompactCom 40 Software Design Guide must be followed at all times.

The Anybus CompactCom supports automatic data rate detection, i.e. if no valid data rate is set, the Anybus CompactCom will measure the bus traffic at different speeds until the correct data rate has been established. Under normal conditions, i.e. with cyclic bus traffic above 2 Hz, the data rate should be detected within 5 seconds. Note that the automatic data rate detection will not work if there is no traffic on the network.

3.3.2 Layer Setting Services (LSS)

The Anybus CompactCom supports the Layer Setting Service (LSS). This service can be used to set the data rate and node address via the network, and may address the module by its Vendor-ID, Product Code, Revision number and serial number.

It is possible to enable LSS during startup by setting the instance Node Address (01h) to 255 or instance Data Rate (02h) to 10, see *Network Configuration Object (04h), p. 24*

3.4 Network Reset Handling

3.4.1 Reset Node

Upon receiving a Reset Node request from the network, the Anybus CompactCom will issue a reset command to the Application Object (FFh) with CmdExt[1] set to 00h (Power-on reset) and shift to the Anybus state EXCEPTION. The bus interface is shifted into a physically passive state.

3.4.2 Reset Communication

Upon receiving a Reset Communication request from the network, the Anybus CompactCom will reset all Communication object entries to their default values, and shift to the CANopen state Reset Communication. No reset command will be issued to the host application.

3.4.3 Restore Manufacturer Parameters to Default

Upon receiving a Restore Manufacturer Parameters to Default request from the network, the Anybus CompactCom will issue a Reset_Request command to the Application Object (FFh) with CmdExt[1] set to 01h (Factory default reset).

No reset command will be sent, it is up to the application to remember that a factory reset is requested the next time a reset command is received.

See *Standard Objects, p. 9*, entry 1011h (Restore default parameters)

4 Object Dictionary (CANopen)

4.1 Standard Objects

4.1.1 General

The standard object dictionary is implemented according to the CiA 302 4.2.0 from CiA (CAN in Automation). Note that certain object entries correspond to settings in the CANopen Object (FBh), and the Diagnostic Object (02h).

4.1.2 Object Entries

Index	Object Name	Sub-Index	Description	Type	Stored in NVS	Access	Notes
1000h	Device Type	00h	Device Type	U32		RO	This information is defined by the CANopen Object, which can optionally be implemented in the host application. See *CANopen Object (FBh), p. 26*.
1001h	Error register	00h	Error register	U8		RO	This information is handled through the Diagnostic Object, see *Diagnostic Object (02h), p. 21*. The Anybus diagnostic object allows up to 5 diagnostic events to be reported.
1003h	Pre-defined error field	00h	Number of errors	U8		RW	
		01h...05h	Error field	U32		RO	
1005h	COB-ID Sync	00h	COB-ID Sync	U32	Yes	RW	Default value is 0000 0080h The Anybus CompactCom 40 CANopen module does not have Sync producer support.
1008h	Manufacturer device name	00h	Manufacturer device name	Visible string		RO	This information is determined by the CANopen Object, which can optionally be implemented in the host application. See *CANopen Object (FBh), p. 26*.
1009h	Manufacturer hardware version	00h	Manufacturer hardware version	Visible string		RO	
100Ah	Manufacturer software version	00h	Manufacturer software version	Visible string		RO	

Index	Object Name	Sub-Index	Description	Type	Stored in NVS	Access	Notes
1010h	Store Parameters	00h	Largest sub index supported	U8		RO	02h
		01h	Store all parameters	U32		RW	Baud rate and Node ID cannot be stored using this command. Relevant only for communication parameters. The SDO timeout that is set in the bus configuration tool, must be considered when reading or writing to this object. When writing to the object 1010h there is at least 1 second until the module will respond, corresponding to the time it takes to store the parameters in the module.
		02h	Store Communication parameters	U32		RW	
1011h	Restore parameters	00h	Largest sub index supported	U8		RO	04h The SDO timeout that is set in the bus configuration tool, must be considered when reading or writing to this object.
		01h	Restore all default parameters	U32		RW	-
		02h	Restore communication default parameters	U32		RW	-
		04h	Restore manufacturer parameters to Default.	U32		RW	See *Network Reset Handling, p. 8*
1014h	COB ID EMCY	00h	COB ID EMCY	U32		RW	Default value is 0000 0080h + NodeId
1015h	Inhibit Time EMCY	00h	Inhibit Time EMCY	U16	Yes	RW	Default value is 0000h
1016h	Consumer Heartbeat Time	00h	Number of entries	U8		RO	01h
		01h	Consumer Heartbeat Time	U32	Yes	RW	Node ID + Heartbeat Time. Value must be a multiple of 1 ms.
1017h	Producer Heartbeat Time	00h	Producer Heartbeat Time	U16	Yes	RW	-

Index	Object Name	Sub-Index	Description	Type	Stored in NVS	Access	Notes
1018h	Identity object	00h	Number of entries	U8		RO	04h
		01h	Vendor ID	U32		RO	This information is determined by the CANopen Object, which can optionally be implemented in the host application. See *CANopen Object (FBh), p. 26*.
		02h	Product Code	U32		RO	
		03h	Revision Number	U32		RO	
		04h	Serial Number	U32		RO	
1400h ... 14XXh	RPDO communication parameter	00h	Largest sub-index supported	U8	Yes	RO	02h
		01h	COB ID used by RPDO	U32	Yes	RW	See “Default PDO Mapping Scheme” below for more details.-
		02h	Transmission type.	U8	Yes	RW	See “PDO Transmission Types” below for more details.
1600h ... 16XXh	RPDO mapping parameter	00h	No. of mapped application objects in RPDO	U8	Yes	RO/RW	0-8
		01h	Mapped object #1	U32	Yes	RO/RW	-
		02h	Mapped object #2	U32	Yes	RO/RW	-
		03h	Mapped object #3	U32	Yes	RO/RW	-
		04h	Mapped object #4	U32	Yes	RO/RW	-
		05h	Mapped object #5	U32	Yes	RO/RW	-
		06h	Mapped object #6	U32	Yes	RO/RW	-
		07h	Mapped object #7	U32	Yes	RO/RW	-
		08h	Mapped object #8	U32	Yes	RO/RW	-
1800h ... 18XXh	TPDO communication parameter	00h	Largest sub-index supported	U8	Yes	RO	05h
		01h	COB ID used by TPDO	U32	Yes	RW	Please note that bit 30 (RTR) always must be set, since RTR is never allowed for TxPDOs. See “Default PDO Mapping Scheme” below for more details.-
		02h	Transmission type	U8	Yes	RW	See “PDO Transmission Types” below for more details.
		03h	Inhibit time	U16	Yes	RW	-
		05h	Event Timer (ms)	U16	Yes	RW	-

Index	Object Name	Sub-Index	Description	Type	Stored in NVS	Access	Notes
1A00h ... 1AXXh	TPDO mapping parameter	00h	No. of mapped application objects in TPDO	U8	Yes	RO/RW	0-8
		01h	Mapped object #1	U32	Yes	RO/RW	-
		02h	Mapped object #2	U32	Yes	RO/RW	-
		03h	Mapped object #3	U32	Yes	RO/RW	-
		04h	Mapped object #4	U32	Yes	RO/RW	-
		05h	Mapped object #5	U32	Yes	RO/RW	-
		06h	Mapped object #6	U32	Yes	RO/RW	-
		07h	Mapped object #7	U32	Yes	RO/RW	-
		08h	Mapped object #8	U32	Yes	RO/RW	-

Access to RPDO and TPDO mapping parameters depends on the host application implementation. Access is RO unless remap commands are supported.

Mapping ADIs on PDOs

The Receive PDO mapping objects (1600h - 16XXh) and the Transmit PDO mapping objects (1A00h - 1AXXh) are configured depending on how the host application is set up:

Mode	Access	Number of objects (in each direction)	Number of sub indexes per object	Notes
Generic, static mapping	RO	1 - 64 Depends on how many ADI mapping items that are mapped by the application during setup. Each PDO can hold 8 ADI mapping items.	1 - 8 Depends on how many ADI mapping items that are mapped by the application during setup. One PDO mapping object at the time will be filled with mapped items.	
Generic, dynamic mapping	RW	1 - 64 Depends on how many ADI mapping items that are mapped by the application during setup. Each PDO can hold 8 ADI mapping items.	8	-
Assembly Mapping Object implemented in host	RO/RW	1 - 64 Number of assembly mapping instances in that direction.	8	The Assembly Mapping object is described in the Anybus CompactCom 40 Software design Guide. Access is RO if the corresponding assembly instance is static, RW if it is dynamic

One PDO communication parameter for each PDO mapping parameter object is created by the module. The corresponding PDO configuration parameter object is used for configuration of the PDO.

Default PDO Mapping Scheme

A default PDO mapping scheme is created during startup. This scheme can then be changed by the configuration tool/master. The default mapping scheme is set up according to the predefined connection set as specified in CiA301.

- RPDO Default COB IDs

RPDO no.	Default COB ID	Default Transmission Type	Description
1	200h + Node ID	254	Default enabled according to DS301, if the mapping scheme requires this PDO
2	300h + Node ID		
3	400h + Node ID		
4	500h + Node ID		
5...64	000h		Default disabled Must be configured by the configuration tool to be usable

- TPDO Default COB IDs

TPDO no.	Default COB ID	Default Transmission Type	Description
1	40000180h + Node ID	254	Default enabled according to DS301,if the mapping scheme requires this PDO Please note that the RTR bit is always set
2	40000280h + Node ID		
3	40000380h + Node ID		
4	40000480h + Node ID		
5...64	000h		Default Disabled Must be configured by the configuration tool to be usable

If no ADIs have been mapped to Process Data, one PDO communication parameter object and one empty PDO mapping parameter object in each direction is created.

The Anybus CompactCom shall fill up PDO mapping parameter objects until they contain either 8 bytes or 8 entries when using the static mapping mode. A new PDO mapping parameter object is then created with the object index incremented by 1. This can create PDOs that doesn't contain 8 bytes if e.g. a UINT16 ADI is mapped first, followed by a UINT32 ADI and then a second UINT32 ADI. The first two ADIs will be added to the first PDO and the third to the seconds PDO. In this case the first PDO will only contain 6 bytes.

PDO Mapping Example

In this example 4 ADIs are mapped :

ADI 1:	Read PD Data: 4xUINT8
ADI 2:	Read PD Data: 3xUINT16
ADI 3:	Write PD Data: 2xUINT8
ADI 4:	Write PD Data: 11xUINT16

These ADIs will result in the following mapping:

Object	Sub-index	Data/Mapping information
1600h	00h	06h
	01h	20010108h
	02h	20010208h
	03h	20010308h
	04h	20010408h
	05h	20020110h
	06h	20020210h
1601h	00h	01h
	01h	0x20020310
1A00h	00h	05h
	01h	20030108h
	02h	20030208h
	03h	20040110h
	04h	20040210h
	05h	20040310h
1A01h	00h	04h
	01h	20040410h
	02h	20040510h
	03h	20040710h
	04h	20040710h
1A02h	00h	04h
	01h	20040810
	02h	20040910
	03h	20040A10
	04h	20040B10

PDO Transmission Types

Message transmission is triggered by:

Transmission Type	Description	RxPDO	TxPDO
254/255	Event driven	Data from RxPDO is copied to correct offset in the read process data buffer. The entire read process data buffer is then sent to the host application immediately.	All event driven TxPDOs are sent immediately when the host application sends new write process data to the module (unless inhibited by inhibit timer).
0	Acyclic Synchronous	Data from RxPDO is copied to correct offset in the read process data buffer. The entire read process data buffer is sent to the host application on the next received SYNC.	TxPDOs with transmission type 0 is sent on every received SYNC.
1...240	Cyclic Synchronous	Data from RxPDO is copied to correct offset in the read process data buffer. The entire read process data buffer is sent to the host application on the next received SYNC.	TxPDOs with transmission type 1-240 are sent on every n:th received SYNC, where n is the configured transmission type.
254/255 with event timer configured for the PDO	Timer driven	Event timer is not supported for RxPDO	All timer driven TxPDOs are sent immediately when the host application sends new write process data to the module or when the event timer expires.

RTR-Remote transmission request on PDOs is not supported by the module.

4.2 Manufacturer and Profile Specific Objects

4.2.1 General

Each object entry in the manufacturer specific range (2001h...FFFFh) corresponds to an instance (an ADI) within the Application Data Object (FEh), i.e. network accesses to these objects results in object requests towards the host application. In case of an error, the status (or error) code returned in the response from the host application will be translated into the corresponding CANopen abort code.

Object Range	Description
2001h... 5FFFh	Manufacturer specific objects
6000h... 9FFFh	Profile specific objects
A000h... BFFFh	Standardized variable objects
C000h... FFFFh	Reserved

> **!** As any access to these object entries will result in an object access towards the host application, the time spent communicating on the host interface must be taken into account when calculating the SDO timeout value.

4.2.2 Translation of Status Codes

Status (or error codes) are translated to CANopen abort codes as follows:

ABCC Status Code	ABCC Name	CANopen Abort Code	CANopen Description
00h	Reserved	0800 0000h	General error
01h	Reserved	0800 0000h	General error
02h	Invalid message format	0604 0047h	Internal device incompatibility
03h	Unsupported object	0602 0000h	Object does not exist in the object dictionary
04h	Unsupported instance	0602 0000h	Object does not exist in the object dictionary
05h	Unsupported Command	0604 0043h	General parameter incompatibility reason
06h	Invalid CmdExt[0]	0602 0000h	Object does not exist in the object dictionary
07h	Invalid CmdExt[1]	0609 0011h	Subindex does not exist
08h	Attribute not settable	0601 0002h	Attempt to write a read only object
09h	Attribute not gettable	0601 0001h	Attempt to read a write only object
0Ah	Too much data	0607 0012h	Data type does not match, length of service parameter too high
0Bh	Not enough data	0607 0013h	Data type does not match, length of service parameter too low
0Ch	Out of range	0609 0030h	Invalid value for parameter (only for write access)
0Dh	Invalid state	0800 0022h	Data cannot be transferred or stored to the application because of the present device state.
0Eh	Out of resources	0504 0005h	Out of memory. (Can be generated if a message is outstanding to the application and during this time an abort is done, and then a new Request on an ADI is made)
0Fh	Segmentation failure	0800 0000h	General error
10h	Segmentation buffer overflow	0800 0000h	General error
11h	Value too high	0609 0031h	Value of parameter too high (download only)
12h	Value too low	0609 0032h	Value of parameter too low (download only)
13h	Attribute controlled from other channel	0800 0021h	Data cannot be transferred or stored to the application because of local control
14h	Message channel too small	0800 0000h	General error
15h	General error	0800 0000h	General error
16h	Protected access	0800 0021h	Data cannot be transferred or stored to the application because of local control
17h	No data available	0800 0024h	No data available
18h-FEh	Reserved	0800 0000h	General error
FFh	Object specific error	0800 0000h	General error

If no corresponding error meets the error definition, the default error code is 0800 0000h (General error).

4.2.3 Network Data Format

Data is translated between the native network format and the Anybus data format as follows:

Anybus Data Type	Native CANopen Data Type
BOOL	UNSIGNED8
SINT8	INTEGER8
SINT16	INTEGER16
SINT32	INTEGER32
UINT8	UNSIGNED8
UINT16	UNSIGNED16
UINT32	UNSIGNED32
CHAR	VISIBLE STRING
ENUM	UNSIGNED8
BITS8	UNSIGNED8
BITS16	UNSIGNED16
BITS32	UNSIGNED32
OCTET	OCTET_STRING
SINT64	INTEGER64
UINT64	UNSIGNED64
FLOAT	REAL32
DOUBLE	REAL64
PAD0-PAD8	UNSIGNED8
PAD9-PAD16	UNSIGNED16
BOOL1	BOOLEAN
BIT1-7	UNSIGNED8

- ADIs with multiple elements are represented as arrays (all elements share the same data type) or as records (the elements may have different data types). Exceptions to this are CHAR which will always be represented as VISIBLE STRING, and OCTET which will always be represented as OCTET_STRING.
- Single element ADIs are represented as a simple variable, with the exception of CHAR which will always be represented as VISIBLE STRING, and OCTET which will always be represented as OCTET_STRING.

4.2.4 Object Entries

The exact representation of an ADI depends on its number of elements. In the following example, ADIs no. 0002h and 0004h only contain 1 element each, causing them to be represented as simple variables rather than arrays. The other ADIs have more than 1 element, causing them to be represented as arrays. The ADI data type is defined according to CiA 302 version 2.4.0

Index	Object Name	Sub-Index	Description	Type	Access	Notes
2001h	ADI 0001h	00h	Number of entries (NNh)	U8	RO	(Sub-Index FFh excluded)
		01h 02h ... ... NNh	ADI value(s) (Attribute #5) ADIs with multiple elements (i.e. arrays) are represented as multiple sub-indexes.	-	-	The data type and access rights of the ADI values are determined by the ADI itself.
		FFh	ADI data type	U32	RO	
2002h	ADI 0002h	00h	ADI value (Attribute #5)	-	-	Data type and Access rights depends on the ADI itself.
		FFh	ADI data type	U32	RO	
2003h	ADI 0003h	00h	Number of entries (NNh)	U8	RO	(Sub-Index FFh excluded)
		01h 02h ... ... NNh	ADI value(s) (Attribute #5) ADIs with multiple elements (i.e. arrays) are represented as multiple sub-indexes.	-	-	Data type and Access rights depends on the ADI itself.
		FFh	ADI data type	U32	RO	
2004h	ADI 0004h	00h	ADI value (Attribute #5)	-	-	Data type and Access rights depends on the ADI itself.
		FFh	ADI data type	U32	RO	
2005h	ADI 0005h	00h	Number of entries (NNh)	U8	RO	(Sub-Index FFh excluded)
		01h 02h ... ... NNh	ADI value(s) (Attribute #5) ADIs with multiple elements (i.e. arrays) are represented as multiple sub-indexes.	-	-	Data type and Access rights depends on the ADI itself.
		FFh	ADI data type	U32	RO	
...	...	...	...	...	...	...
FFFFh	ADI DFFFh	00h	Number of entries (NNh)	U8	RO	(Sub-Index FFh excluded)
		01h 02h ... ... NNh	ADI value(s) (Attribute #5) ADIs with multiple elements (i.e. arrays) are represented as multiple sub-indexes.	-	-	Data type and Access rights depends on the ADI itself.
		FFh	ADI data type	U32	RO	

5 Anybus Module Objects

5.1 General Information

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

Standard Objects:

5.2 Anybus Object (01h)

Object Description

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

Supported Commands

Object:	Get_Attribute
Instance:	Get_Attribute
	Set_Attribute
	Get_Enum_String

Object Attributes (Instance #0)

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

Instance Attributes (Instance #1)

Basic

#	Name	Access	Type	Value
1	Module type	Get	UINT16	0403h (Standard Anybus CompactCom 40)
2... 11	-	-	-	See the general Anybus CompactCom 40 Software Design Guide for further information.
12	LED colors	Get	struct of: UINT8 (LED1A) UINT8 (LED1B) UINT8 (LED2A) UINT8 (LED2B)	Value: Color: 01h Green 02h Red 01h Green 02h Red
13...16	-	-	-	See the general Anybus CompactCom 40 Software Design Guide for further information.
17	Virtual attributes	Get/Set		
18	Black list/White list	Get/Set		

To be able to access ADI values set in virtual attributes via SDO read requests, the ADI can only have 1 element or be of OCTET or CHAR type. This is because virtual attributes does not support the Get_Indexed_Attribute command.

5.3 Diagnostic Object (02h)

Object Description

This object provides a standardised way of handling host application events and diagnostics, and is thoroughly described in the general Anybus CompactCom 40 Software Design Guide.

Supported Commands

Object:	Get_Attribute
	Create
	Delete
Instance:	Get_Attribute

Object Attributes (Instance #0)

#	Name	Access	Type	Value	Description
1... 4	-	-	-		See the general Anybus CompactCom 40 Software Design Guide for further information.
11	Max no. of instances	Get	UINT16	0006h	Of the maximum number of instances there should always be one instance reserved for an event of severity level “Major, unrecoverable”, to force the module into the state EXCEPTION.
12	Supported functionality	Get	BITS32	Bit 0: “0” Bit 1 - 31: reserved	Latching events are not supported Set to “0”

Instance Attributes (Instance #1...n)

#	Name	Access	Type	Value
1	Severity	Get	UINT8	See the Anybus CompactCom 40 Software Design Guide for further information. **Note**: Latching events are not supported.
2	Event Code	Get	UINT8	Standard event code. See the Anybus CompactCom 40 Software Design Guide.
3	NW specific extension	Get	Array of UINT8	CANopen specific EMCY code (2 bytes)

When an instance is created (i.e. a diagnostic event is entered), the following actions are performed:

- A new entry will be created in object entry 1003h (Pre-defined error field) as follows:

High byte	(UINT32)	Low byte
(Not used)	Event Code	00h

- The Error Register (object entry 1001h) is set with the corresponding bit information
- The EMCY Object is sent to the network with the following information:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
00h	Event Code	Error Register (1001h)	Manufacturer Specific Field (Not used)				

When creating a Major-severity, this will not end up as an EMCY-message on the bus, since this effectively forces the Anybus module to enter the EXCEPTION state.

EMCY Error Code	Description
8110h	CAN controller signalled a lost message
8120h	CAN controller reached the warning limit due to error frames.
8210h	A received PDO was smaller than specified by the valid mapping table
8220h	The DLC of a received PDO exceeded the length specified by the valid mapping table.
8130h	An error control event has occurred (either a life guarding event or a heartbeat event).
8140h	CAN controller has recovered from a BUS OFF state.
8150h	COB-ID collision detected.
FF01h	Process data remap was NAKed by the host application. No valid process data map available.

5.4 Network Object (03h)

Object Description

For more information, consult the general Anybus CompactCom 40 Software Design Guide.

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

Instance Attributes (Instance #1)

Basic

#	Name	Access	Type	Value
1	Network type	Get	UINT16	0020h
2	Network type string	Get	Array of CHAR	“CANopen”
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 or Map_ ADI_Write_Ext_Area command. 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 or Map_ ADI_Read_Ext_Area command- Consult the general Anybus CompactCom 40 Software Design Guide for further information.
7	Exception Information	Get	UINT8	Additional CANopen specific exception information is presented here in case the Anybus module has shifted to the EXCEPTION state. Value: — Meaning: 00h — (no additional information available) 01h — Invalid data type reported by the application 06h — The implementation of the Assembly Mapping Host Object is incorrect, e.g. the attribute 11 or 12 is not supported. 07h — The application supports the Remap ADI commands, but returned an error response when requesting object attributes 11 or 12 of the Application Data Object (“No. of read process data mappable instances” or “No of write process data mappable instances”) or when issuing the Get_Instance_ Numbers command towards the Application Data Object.
8...10	(reserved)	-	-	(not used)
11... 255	Network specific	-	-	

5.5 Network Configuration Object (04h)

Object Description

This object contains network specific configuration parameters that may be configured by the end user.

Supported Commands

Object:	Get_Attribute (01h)
	Reset (Factory Default) (05h)
Instance:	Get_Attribute (01h)
	Set_Attribute (02h)

Object Attributes (Instance #0)

#	Name	Access	Type	Value
1	Name	Get	Array of CHAR	“Network Configuration”
2	Revision	Get	UINT8	01h
3	Number of instances	Get	UINT16	0002h
4	Highest instance no.	Get	UINT16	0002h

Instance Attributes (Instance #1, Node Address)

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	“Node Address” (Multilingual, see page *25*)
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	UINT8	Value: — Meaning: 1... 127: — CANopen device address value 255 — Enable LSS at startup. Start with Node Address 255 if no address previously set by LSS. (Note that this value may be updated by the LSS service, see *Layer Setting Services (LSS), p. 8*)
6	Configured value	Get	UINT8	Configured value

Instance Attributes (Instance #2, Data Rate)

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"Data Rate" (Multilingual, see page 25)
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	Value: ENUM string: 0: "10 kbps" 1: "20 kbps" 2: "50 kbps" 3: "reserved" 4: "125 kbps" 5: "250 kbps" 6: "500 kbps" 7: "800 kbps" 8: "1 Mbps" 9: "Auto" 10: "LSS" (default) Enable LSS at startup. Start with Data Rate Auto if no data rate previously set by LSS. (Note that this value may be updated by the LSS service, see *Layer Setting Services (LSS), p. 8*)
6	Configured value	Get	UINT8	Configured value

Command Details: Reset

Details

Command Code:	05h
Valid for:	Object

Description

Resets Baud Rate and Device Address values to default

- Command Details

Field	Comments
CmdExt[0]	(reserved)
CmdExt[1]	01h = Factory Default Reset

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	Data rate	Datenrate	Veloc transf	Velocità dati	Vitesse

6 Host Application Objects

6.1 General Information

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

Standard Objects:

- Assembly Mapping Object (EBh), see Anybus CompactCom 40 Software Design Guide
- Application Data Object (FEh) see Anybus CompactCom 40 Software Design Guide
- Application Object (FFh) see Anybus CompactCom 40 Software Design Guide

Network Specific Objects:

- *CANopen Object (FBh), p. 26*

6.2 CANopen Object (FBh)

Object Description

This object implements CANopen specific settings in the host application.

The implementation of this object is optional; the host application can support none, some, or all of the attributes specified below. The Anybus CompactCom 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 the module attempts to retrieve a value of an attribute not listed below, respond with an error message (06h, Invalid CmdExt[0]).

Support for this object is optional, but in order to certify the product a new Vendor ID should be assigned. The CAN in Automation group recommends requesting a Vendor ID. It is also highly recommended to support all attributes listed below, if the object is implemented, since this has a very high impact on CANopen-specific functionality.

Supported Commands

Object:	Get_Attribute (01h)
Instance:	Get_Attribute (01h)

Object Attributes (Instance #0)

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

Instance Attributes (Instance #1)

#	Name	Access	Data Type	Default Value	Comment
1	Vendor ID	Get	UINT32	0000 001Bh	These values replace the settings in object entry 1018h. (Identity Object) A unique Vendor ID has to be requested and assigned if the product is to be certified and/or if LSS services are used in the network.
2	Product Code	Get	UINT32	0000 000Dh	
3	Major revision	Get	UINT16	Major revision	
4	Minor revision	Get	UINT16	Minor revision	
5	Serial Number	Get	UINT32	Unique number	
6	Manufacturer Device Name	Get	Array of CHAR (Max. 64 bytes)	“Anybus CompactCom 40 CANopen”	Replaces object entry 1008h (Manufacturer Device Name)
7	Manufacturer Hardware Version	Get	Array of CHAR (Max. 64 bytes)	n where n is the hardware funcID of the Anybus Hardware	Specifies the value of object entry 1009h (Manufacturer Hardware Version) “n” is the hardware funcID of the Anybus Hardware
8	Manufacturer Software Version	Get	Array of CHAR (Max. 64 bytes)	x.yy.zz	Replaces object entry 100Ah (Manufacturer Software Version)
9	-	-	-	-	(reserved)
10	Device Type	Get	UINT32	0000 0000h	Replaces object entry 1000h Device type
11 - 13	-	-	-	-	(reserved)
14	-	-	-	-	(not used)
15 - 16	-	-	-	-	(reserved)
17	Read PD buffer initial value	Get	Array of UINT8		If this attribute is implemented, the host application returns the initial value for the entire read process data buffer on a single command. See below for more information

Read PD Buffer Initial Value

The Anybus CompactCom uses this attribute as an optimized way of fetching the initial value of the read process data buffer. The length of this attribute is equal to the length of the read process data.

If the attribute isn’t implemented, the module will read the value of each read process data mapped ADI element when receiving the NMT command Start Node to have valid initial values in the read process data buffer. This generates one Get_Attribute_Indexed command for each mapped ADI entry, so the more ADI entries that are mapped to read process data, the longer this process will take. This can cause issues depending on the behavior of the CANopen master.

The host application can speed up this process by implementing this attribute, returning the initial value for the entire read process data buffer on a single command.

If less data than the read process data length is returned, the Anybus CompactCom will set the remaining process data buffer to 0.

If more data than the read process data length is returned, the additional bytes are ignored.

If more than 255 bytes of read process data is used the host application must use the large message area of the Anybus CompactCom, otherwise the read process data exceeding 255 bytes will be set to 0.

This page intentionally left blank

A Categorization of Functionality

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

A.1 Basic

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

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

A.2 Extended

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

Some of the functionality offered may be specialized and/or seldom used. As most of the available network functionality is enabled and accessible, access to the specification of the industrial network may be required.

B Implementation Details

B.1 SUP-Bit Definition

The supervised bit (SUP) indicates that the network participation is supervised by another network device. CANopen specific interpretation:

SUP-bit	Interpretation
0	[LSS active] - *or* - [No error control mechanism is enabled]
1	[Heartbeat consumer - *and* - Heartbeat producer is enabled & error free] - *and* - [LSS not active]

B.2 Anybus State Machine

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

Anybus State	CANopen NMT State	Implementation	Comment
NW_INIT	NMT state Initialization	The Anybus is performing a network initialization. The Anybus scans the application for a possible CANopen Host Object. If there is one, the data is read from the object instances attributes. If there is Read Process data mapped; the Anybus CompactCom starts reading out the start-up values from the ADIs to use these as initial Read PD. The CANopen network process data channel is not active.	Network specific application objects may receive commands from the Anybus. The application shall regard process data from network as not valid
WAIT_PROCESS	PRE-OPERATIONAL Send Bootup Event message on the bus the first-time the state is entered. If a Heart beat error occurs this state is entered. If LSS is enabled this state is entered.	If there is no valid Node address (set in the Network Configuration Object) the module enters the LSS init. state. The Anybus will remain in LSS init. state until a valid Node ID is set from the network. If the Baud rate instance in Network configuration object is set to Auto, the module will start to scan for the used baud rate. It will stay in this state until the node has detected a baud rate or has restarted.	The application shall regard process data from the network as not valid
IDLE	STOPPED When NMT state Stopped is entered the communication stops altogether (except heartbeat, if active).	The network device is in idle mode	The application may act upon received data or go to an idle state The process data is not valid.
PROCESS_ACTIVE	OPERATIONAL Services on SDO, PDO SYNC EMCY-objects can be executed on the node.	The network process data channel is active and error free	Normal data handling can be performed.

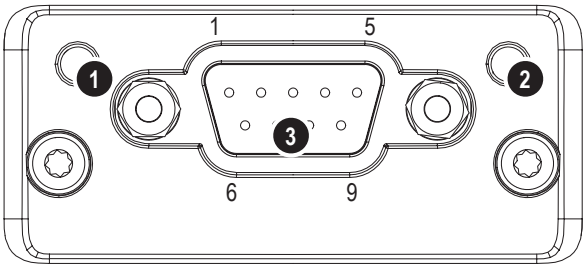
Anybus State	CANopen NMT State	Implementation	Comment
ERROR	-	The Anybus CAN controller has entered the BUS-off state. If the BUS-off occurs in the OPERATIONAL state the module will return to PRE-OPERATIONAL state if the CAN controller is changed to ERROR-ACTIVE again.	The application should go to a safe state.
EXCEPTION	-	An illegal configuration or a NMT service RESET NODE requests have been received. The Anybus will reset the CAN controller and stop communication over the network. This state could also be entered due to an application error (invalid network configuration parameter, timeout etc.) or because the Anybus is waiting for a reset to be executed	Details can be read from the Network Object (03h), Instance #1, Attribute # 7 (Exception information) Correct any errors. Reset the Anybus.

C Technical Specification

C.1 Front View

C.1.1 Front View (CANopen Connector)

#	Item
1	RUN LED
2	ERROR LED
3	CANopen Interface



The flash sequences for LEDs 1 & 2 are defined in CiA 303–3.

C.1.2 RUN LED

LED State	Description	Comments
Off	-	No power.
Green	OPERATIONAL	The module is in the state OPERATIONAL.
Green, blinking	PRE-OPERATIONAL	The module is in the state PRE-OPERATIONAL.
Green, 1 flash	STOPPED	The module is in the state STOPPED.
Green, flickering	Autobaud	Baud rate detection in progress or LSS in progress (alternately flickering with ERROR LED)
Red	EXCEPTION state (Fatal Event)	The module has shifted into the state EXCEPTION.

If both LEDs turns red, this indicates a fatal event; the bus interface is shifted into a physically passive state.

C.1.3 ERROR LED

LED State	Description	Comments
Off	-	No power or the device is in working condition.
Red, single flash	Warning limit reached	A bus error counter reached or exceeded its warning level.
Red, flickering	LSS	LSS services in progress (alternately flickering with RUN LED).
Red, double flash	Error Control Event	A heartbeat event (Heartbeat consumer) has occurred.
Red	Bus off (Fatal Event)	Bus off

If both LEDs turns red, this indicates a fatal event; the bus interface is shifted into a physically passive state.

C.1.4 CANopen Interface

Pin	Signal
1	-
2	CAN_L
3	CAN_GND
4	-
5	CAN_SHLD
6	-
7	CAN_H
8	-
9	-
Housing	CAN_SHIELD

C.2 Functional Earth (FE) Requirements

In order to ensure proper EMC behavior, the module must be properly connected to functional earth via the FE pad/FE mechanism described in the *Anybus CompactCom 40 Hardware Design Guide*. Proper EMC behavior is not guaranteed unless these FE requirements are fulfilled.

C.3 Power Supply

C.3.1 Supply Voltage

The Anybus CompactCom 40 CANopen requires a regulated 3.3 V power source as specified in the general *Anybus CompactCom 40 Hardware Design Guide*.

C.3.2 Power Consumption

TheAnybus CompactCom 40 CANopen is designed to fulfil the requirements of a Class A module. For more information about the power consumption classification used on the Anybus-CompactCom platform, consult the general Anybus-CompactCom 40 Hardware Design Guide.

The current hardware design consumes up to 180 mA

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.

In line with HMS Networks AB policy of continuous product development, we reserve the right to change the exact power requirements of this product without prior notification. However, in any case, the Anybus CompactCom 40 CANopen will remain as a Class A module.

C.4 Environmental Specification

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

C.5 EMC Compliance

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

D Timing & Performance

D.1 General Information

This chapter specifies timing and performance parameters that are verified and documented for the Anybus CompactCom 40 CANopen.

The following timing aspects are measured:

Category	Parameters	Page
Startup Delay	T1, T2	*34*
NW_INIT Handling	T100	*34*
Event Based WrMsg Busy Time	T103	*34*
Event Based Process Data Delay	T101, T102	*35*

For further information, please consult the Anybus CompactCom 40 Software Design Guide.

D.2 Internal Timing

D.2.1 Startup Delay

The following parameters are defined as the time measured from the point where /RESET is released to the point where the specified event occurs.

Parameter	Description	Max.	Unit.
T1	The Anybus CompactCom 40 CANopen module generates the first application interrupt (parallel mode)	12.4	ms
T2	The Anybus CompactCom 40 CANopen module is able to receive and handle the first application telegram (serial mode)	12.4	ms

D.2.2 NW_INIT Handling

This test measures the time required by the Anybus CompactCom 40 CANopen module to perform the necessary actions in the NW_INIT-state.

Parameter	Conditions
No. of network specific commands	Max.
No. of ADIs (single UINT8) mapped to Process Data in each direction. (If the network specific maximum is less than the value given here, the network specific value will be used.)	32
Event based application message response time	> 1 ms
No. of simultaneously outstanding Anybus commands that the application can handle	1

Parameter	Description	Communication	Max.	Unit.
T100	NW_INIT handling	Event based modes	46.6	ms

D.2.3 Event Based WrMsg Busy Time

The Event based WrMsg busy time is defined as the time it takes for the module to return the H_WRMSG area to the application after the application has posted a message.

Parameter	Description	Min.	Max.	Unit.
T103	H_WRMSG area busy time	22.0	23.4	µs

D.2.4 Event Based Process Data Delay

“Read process data delay” is defined as the time from the last rising edge of the CAN transceiver’s RxD signal frame has been received by the network interface, to when the RDPDI interrupt is asserted to the application.

Due to the nature of CANopen, the read process data buffer is posted each time a single RxPDO is received. One RxPDO can contain a maximum amount of 8 bytes of process data. Verification has therefore been done up to 8 bytes of process data.

“Write process data delay” is defined as he time from the WRPD interrupt until the first rising bit on the CAN transceiver TXD signal.

Eight different IO sizes (1, 8, 16, 32, 64, 128, 256, and 512 bytes) were used in the tests, all giving the similar test results.

Parameter	Description	Delay (min.)	Delay (typ.)	Delay (max.)	Unit
T101	Read process data delay, 1 byte	15.6	-	16.8	µs
	Read process data delay, 8 bytes	16.1	-	18.0	µs
T102	Write process data delay, 1 byte	10.8	-	20.6	µs
	Write process data delay, 8 bytes	12.8	-	23.6	µs
	Write process data delay, 16 bytes	14.0	-	25.0	µs
	Write process data delay, 32 bytes	16.0	-	28.0	µs
	Write process data delay, 64 bytes	21.0	-	34.0	µs
	Write process data delay, 128 bytes	31.0	-	44.0	µs
	Write process data delay, 256 bytes	52.0	-	68.0	µs
	Write process data delay, 512 bytes	92.0	-	112	µs

E Conformance Test Guide

E.1 Introduction

This chapter includes network specific settings that are needed for a host application to be up and running and possible to certify for use on CANopen networks.

E.2 Fieldbus Conformance Notes

- This product is pre-certified for network compliance. While this is done to ensure that the final product can be certified, it does not necessarily mean that the final product will not require recertification. Contact HMS Networks AB for further information.
- The .EDS file associated with this product must be altered to match the final implementation. See also *Electronic Data Sheet (EDS), p. 6*.
- HMS Networks AB recommends that the device identity information is customized to ensure interoperability. CiA (CAN in Automation) members should apply for a unique Vendor ID; non-members may contact HMS Networks AB to obtain a custom Product ID. Note however that a unique Vendor ID is required when certifying the final product.
- The module supports CAN Standard Frames with 11-bit Identifier Field, see CiA 301 v4.2.0. 29-bit Identifier Fields are not allowed.

E.3 Certification

When using the default settings of all parameters, the Anybus CompactCom 40 CANopen is precertified for network compliance. This precertification is done to ensure that your product can be certified, but it does not mean that your product will not require certification.

Any change in the parameters in the EDS file, supplied by HMS Networks AB, will require a certification. A Vendor ID can be obtained from CiA (CAN in Automation) and is compulsory for certification. This section provides a guide for a successful conformance testing of your product, containing the Anybus CompactCom 40 CANopen, to comply with the demands for network certification set by CiA (CAN in Automation).

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

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

E.3.1 Reidentifying Your Product

After successful setting of attribute #5 (Setup Complete) in the Anybus Object (01h), the Anybus CompactCom asks for identification data from the CANopen Object (FBh). Therefore, the attributes listed below shall be implemented and proper values returned.

Object/Instance	Attribute	Explanation	Default	Customer sample	Comment
CANopen Object (FBh), Instance 1	#1, Vendor ID	With this attribute you set the Vendor ID of the device.	Vendor ID: 001Bh	Vendor ID: 1111h	This information must match the keyword values of the "Device" section in the EDS file. See CANopen Object 1018h, *Standard Objects, p. 9*.
CANopen Object (FBh), Instance 1	#2, Product Code	With this attribute you set the Product Code of the device	0000 000Dh	0000 2222h	
CANopen Object (FBh), Instance 1	#3, Major Revision	With this attribute you set the Major Revision of the device.		0001	
CANopen Object (FBh), Instance 1	#4, Minor Revision	With this attribute you set the Minor Revision of the device.		0001	
CANopen Object (FBh), Instance 1	#6, Manufacturer Device Name	With this attribute you set the Product Name of the device.	Anybus CompactCom 40 CANopen	"Widget"	This information must match the keyword values of the Device section in the EDS file. See CANopen object 1008h, *Standard Objects, p. 9*.

E.3.2 Factory Default Reset

Reset command to Application Object (FFh) must be supported

When Anybus CompactCom 40 CANopen products are delivered, they are required to be in their Factory Default state. The application has to support reset with power cycle and factory default reset, see *Network Reset Handling, p. 8*.

E.3.3 Modify the EDS File

Modify the Anybus CompactCom CANopen EDS file so that it corresponds to the vendor product (e.g. Vendor ID, Product Name and Product Number along with ADI object names, that correspond to descriptive names in the application. Also the ADI information must correspond.). The EDS file has to contain all ADIs created by the application. Run the EDS file checker program from www.can-cia.org.

See also *Electronic Data Sheet (EDS), p. 6* .

F Backward Compatibility

F.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.

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

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

F.2 Hardware Compatibility

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

F.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.

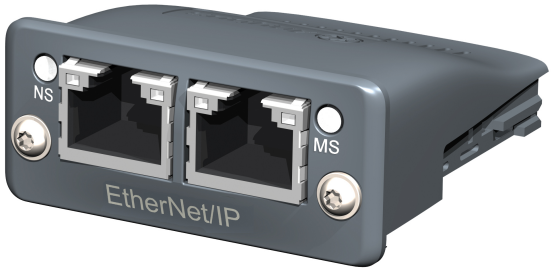


Fig. 1 Anybus CompactCom M30/M40

F.2.2 Chip

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

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

F.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 Networks AB 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 Networks AB for more information on the Anybus CompactCom B40-2.

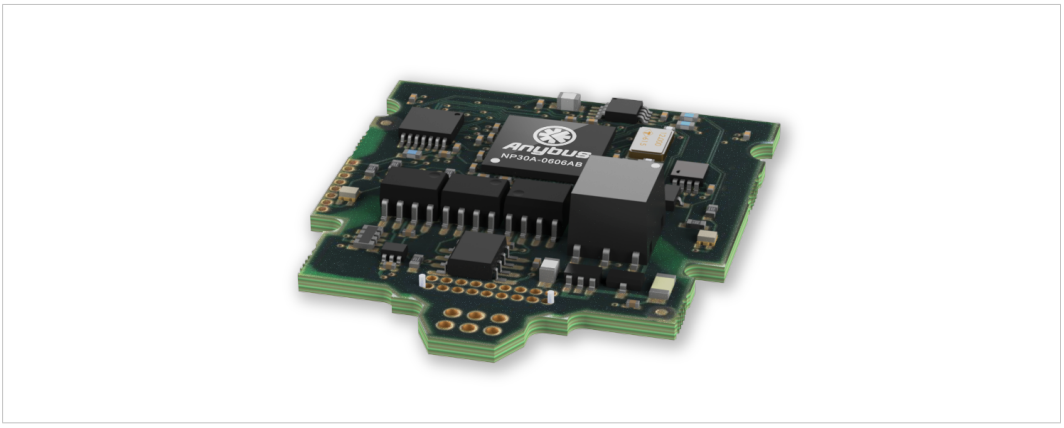


Fig. 2 Anybus CompactCom B30

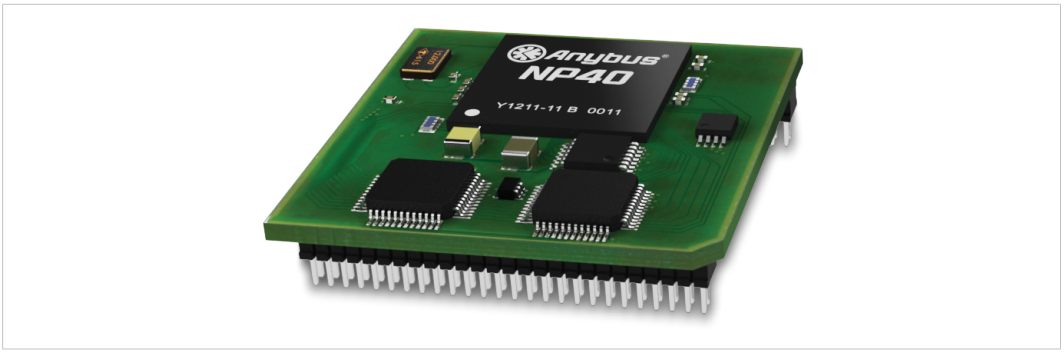


Fig. 3 Anybus CompactCom B40–1 (not for migration)

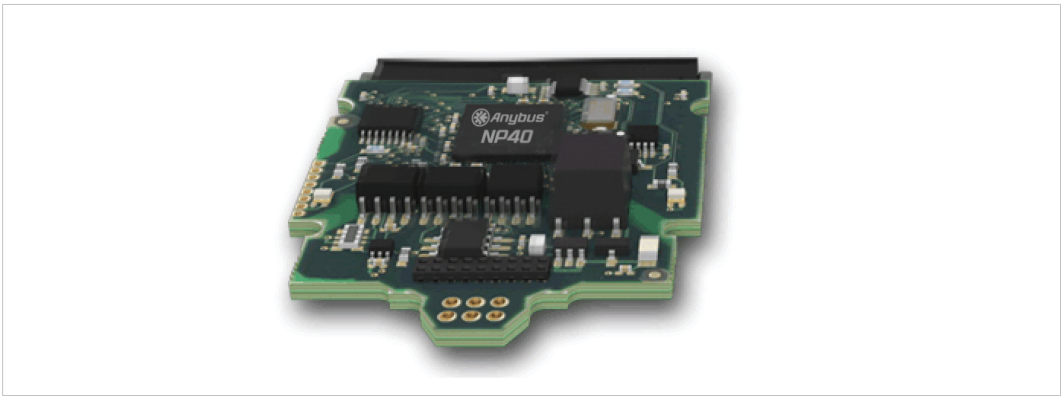


Fig. 4 Anybus CompactCom B40–2

F.2.4 Host Application Interface

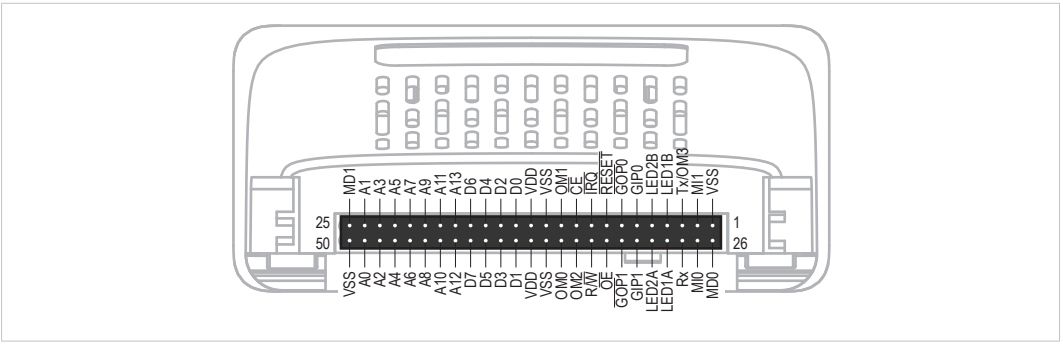


Fig. 5

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.

> **!** 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.

> *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 $V_{IH}=V_{DD}+0,2$ V, and for the 40-series as $V_{IH}=3.45$ 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”.

F.3 General Software

F.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.

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”

F.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.

F.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.

F.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], p. 42* • *GOP[0..1]/LED4[A..B], p. 42*

F.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”.

F.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 Networks AB comply with this difference.

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

F.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 Networks AB comply with this difference.

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

F.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 releasedAnybus CompactCom 30 example drivers from HMS Networks AB comply with this difference.

F.4 Network Specific — CANopen

F.4.1 CANopen Object (FBh)

Attribute	30-series	40-series	Change/Action/Comment
#2, Product Code	Default: 0000 000Ah	Default: 0000 000Dh	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, Manufacturer Device Name	Default: “Anybus-CC CANopen”	Default: “CompactCom 40 CANopen”	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.
#7, Manufacturer Hardware Version	Max 24 bytes	Max 64 bytes	
#8, Manufacturer Software Version	Max 24 bytes	Max 64 bytes	
#10, Device Type	Not implemented	0000 0000h	Object entry 1000h
#14, Default PDO map configuration	Implemented	Not implemented	
#17, Read PD Buffer Initial Value	Not implemented	Implemented	

F.4.2 Object Dictionary

Object Index	30-series	40-series	Change/Action/Comment
0005h	Dummy object	Not implemented	These objects were used for default PDO mapping, which is no longer supported
0006h	Dummy object	Not implemented	
0007h	Dummy object	Not implemented	
1000h (Device Type	0000 0000h (no profile)	CANopen object, attribute #10	
1003h (Predefined error field)	5 diagnostic events, +1 extra	5 diagnostic events, no extra	
1009h	Index not supported if attribute #7 in CANopen is not implemented	Default value: Anybus CompactCom hardware identifier if atttribute #7 in CANopen object is not implemented	
100Ch (Guard time)	Implemented	Not implemented	Node Guarding and Life Guarding are not supported in the 40-series. Legacy functionality.
100Dh (Life time factor)	Implemented	Not implemented	
Receive PDO parameter	1400h - 1423h	1400h - 1440h	
Receive PDO mapping	1600h - 1623h	1600h - 1640h	
Transmit PDO parameter	1800h - 1823h	1800h - 1840h	
Transmit PDO mapping	1A00h - 1A23h	1A00h - 1A40h	

F.4.3 ADI Handling

ADI Handling	30-series	40-series	Change/Action/Comment
Mapping an ADI to both Read and Write process data	Not possible. Error indication	Possible	
Mapping an ADI more than once	Not possible. Error indication	Possible but not recommended	

ADI Handling	30-series	40-series	Change/Action/Comment
ADIs accessible from the network	Up to 16383 Mapped to objects 2001h - 5FFFh	Up to 57343 Mapped to objects 2001h - FFFFh	
ADI requests	Handled within 3.5 s	Blockes until response received	

F.4.4 Process Data

The PDO functionality has been completely re-designed in the Anybus CompactCom 40 CANopen to have a transparent relationship between the CANopen PDO mapping and the Anybus CompactCom process data mapping.

In the Anybus CompactCom 30, the PDO mapping can be changed even though remap commands are not supported.

To change the PDO mapping in the Anybus CompactCom 40, the remap commands must be supported in the host application.

	30-series	40-series	Change/Action/Comment
SDO downloads to ADIs mapped as read process data	Not accepted during operational state	Accepted	
Mapping arrays to PDO	Only first element used	Whole ADI mapped	Only applies to the default mapping.
Number of RPDOs/ TPDOs	36 RPDOs and 36 TPDOs	64 RPDOs and 64 TPDOs	

- There are three mapping modes in the 40-series:
  - Static Mapping Mode – Is used if remap commands are not implemented.
  - Dynamic Mapping Mode – Remap commands implemented. No assembly object.
  - Assembly Mapping Mode – Remap commands and assembly object implemented.
- Initialization of Read Process data
  - **30-series**: Read the process data values in NW_INIT state.
  - **40-series**: Read the values when going from pre-operational to operational state. Implement attribute #17 in the CANopen object to speed up the startup

F.4.5 Miscellaneous

- Translation of SDO access Status Codes are updated.
- Network Data Format translation updated. Support for RECORD objects has been added to support structured ADIs.

F.4.6 LED Indications

LED Indictaion	30-series	40-series	Change/Action/Comment
RUN LED — Flickering green	Baud rate detection in progress	Baud rate detection in progress or LSS in progress	

F.4.7 Firmware Upgrade

Anybus CompactCom 40 CANopen supports firmware upgrade over the application interface using the Anybus File System Interface object.

Box 4126
300 04 Halmstad, Sweden

info@hms.se