Anybus® BY HMS NETWORKS

Anybus® CompactCom™ 30

CANopen

NETWORK GUIDE

HMSI-168-78 4.2 en-US ENGLISH

HMS Connecting Devices™

Important User Information

Liability

Every care has been taken in the preparation of this document. Please inform HMS Industrial Networks of any inaccuracies or omissions. The data and illustrations found in this document are not binding. We, HMS Industrial Networks, reserve the right to modify our products in line with our policy of continuous product development. The information in this document is subject to change without notice and should not be considered as a commitment by HMS Industrial Networks. HMS Industrial Networks assumes no responsibility for any errors that may appear in this document.

There are many applications of this product. Those responsible for the use of this device must ensure that all the necessary steps have been taken to verify that the applications meet all performance and safety requirements including any applicable laws, regulations, codes, and standards.

HMS Industrial Networks will under no circumstances assume liability or responsibility for any problems that may arise as a result from the use of undocumented features, timing, or functional side effects found outside the documented scope of this product. The effects caused by any direct or indirect use of such aspects of the product are undefined, and may include e.g. compatibility issues and stability issues.

The examples and illustrations in this document are included solely for illustrative purposes. Because of the many variables and requirements associated with any particular implementation, HMS Industrial Networks cannot assume responsibility for actual use based on these examples and illustrations.

Intellectual Property Rights

HMS Industrial Networks has intellectual property rights relating to technology embodied in the product described in this document. These intellectual property rights may include patents and pending patent applications in the USA and other countries.

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 30 CANopen. The document describes the features that are specific to Anybus CompactCom 30 CANopen. For general information regarding Anybus CompactCom 30, consult the Anybus CompactCom 30 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 30 Software Design Guide	HMS	HMSI-168-97
Anybus CompactCom 30 Hardware Design Guide	HMS	HMSI-168-31
Anybus CompactCom Host Application Implementation Guide	HMS	HMSI-27-334
IEC 61158-6	IEC	
CiA Draft Standard 301 v4.02	CAN in Automation	

1.3 Document History

Version	Date	Description
2.00	2007-05-30	First official version
2.01	2007-07-09	Minor update
2.02	2008-06-30	Minor update
3.00	2010-04-14	Change of concept, updates
3.01	2011-02-10	Minor updates
3.02	2012-01-26	Updated certification instructions
3.03	2012-09-24	Minor update
4.0	2017-05-11	First version in DOX, updated
4.1	2018-06–29	Corrected default pdo mapping scheme Minor corrections
4,2	2019-02-26	Rebranding

1.4 Document Conventions

Ordered lists are used for instructions that must be carried out in sequence:

1. First do this
2. Then do this

Unordered (bulleted) lists are used for:

- Itemized information
- Instructions that can be carried out in any order

...and for action-result type instructions:

► This action...

→ leads to this result

Bold typeface indicates interactive parts such as connectors and switches on the hardware, or menus and buttons in a graphical user interface.

```
Monospaced text is used to indicate program code and other
kinds of data input/output such as configuration scripts.
```

This is a cross-reference within this document: *Document Conventions, p. 4*

This is an external link (URL): www.hms-networks.com

This is additional information which may facilitate installation and/or operation.

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

Caution
This instruction must be followed to avoid a risk of personal injury.

WARNING
This instruction must be followed to avoid a risk of death or serious injury.

1.5 Terminology

- 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 30 CANopen

2.1 General

The Anybus CompactCom 30 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

- DS301 v4.02 compliant
- Galvanically isolated bus electronics
- Supports all standard baud rates
- Automatic baud rate detection
- Supports LSS
- Customizable Identity Information
- Up to 36 TPDO’s & 36 RPDO’s (Corresponds to a total of 288 bytes of Process Data)
- PDO mapping can be customized via network configuration tool
- Up to 16383 ADIs can be accessed from the network as Manufacturer Specific Objects.
- Diagnostic support
- Supports Expedited- and Segmented SDO Transfer (Block Transfer not supported)

3 Tutorial

3.1 Introduction

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

3.2 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 Industrial Networks 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. 9*.
- HMS 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 Industrial Networks 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 Draft Standard 301 v4.02. 29-bit Identifier Fields are not allowed.

3.3 Certification

When using the default settings of all parameters, the Anybus CompactCom 30 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 Industrial Networks, 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 30 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 at www.anybus.com for more information.

3.3.1 Reidentifying Your Product

After successful setting of attribute #5 (Setup Complete) in the Anybus Object (01h), the Anybus CompactCom 30 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.
CANopen Object (FBh), Instance 1	#2, Product Code	With this attribute you set the Product Code of the device	0000 000Ah	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, Manufactur-er Device Name	With this attribute you set the Product Name of the device.	Anybus-CC CANopen	“Widget”	This information must match the keyword values of the Device section in the EDS file.

3.3.2 Factory Default Reset

Reset command to Application Object (FFh) must be supported

When Anybus CompactCom 30 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. 13*.

3.3.3 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. 9* .

4 Basic Operation

4.1 General Information

4.1.1 Software Requirements

Generally, no additional network support code needs to be written to support the Anybus CompactCom 30 CANopen, however due to the nature of the CANopen networking system certain things must be taken into account:

- An ADI cannot be mapped as both Write- and Read Process Data simultaneously. Any attempt to do so will result in an error.
- An ADI cannot be mapped more than once. Any attempt to do so will result in an error.
- Only ADIs with instance numbers less than 16384 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 during the NW INIT state, ADIs mapped as Read Process Data must have both Set and Get access. If not, the startup value for their data will be zero.
- Requests towards instances in the Application Data Object shall be handled within 3.5 seconds, or the module will generate an error on the bus.
- 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 1.5 seconds 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.
- When using the default PDO mapping scheme, only the first element of an array will be represented cyclically. There is however no such limitation if creating a custom PDO mapping scheme through the bus configuration tool.
- For data consistency reasons, the module will not accept SDO downloads to ADIs mapped as Read Process Data during the NMT Operational State.

For further information about the Anybus CompactCom software interface, consult the general Anybus-CompactCom 30 Software Design Guide.

4.1.2 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 Industrial Networks 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. 7*

4.1.3 Device Identity

Generic Implementation

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

Object Entry	Value
Vendor ID	0000 001Bh (HMS Industrial Networks)
Product Code	0000 000Ah (Anybus CompactCom)
Manufacturer Device Name	“Anybus-CC CANopen”
Manufacturer Hardware Revision	-
Manufacturer Software Revision	(Anybus software revision)

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 30 device.

See also ...

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

4.2 Data Exchange

4.2.1 Application Data (ADI)

Application Data Instances (ADIs) can be accessed from the network via dedicated object entries in the Manufacturer Specific range (2001h - 5FFFh), see *Manufacturer Specific Objects, p. 16*.

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

4.2.2 Process Data

On CANopen, ADIs mapped as Process Data can be exchanged cyclically as Process Data Objects (PDOs) on the bus. Which ADIs that are actually exchanged this way is in the end determined by the network configuration tool.

For natural reasons, only object entries which correspond to Process Data-mapped ADIs may be mapped as PDOs; any attempt to map an object entry which does not fit this criteria will result in an error.

The module supports up to 36 RPDOs and 36 TPDOs, each capable of carrying up to 8 bytes of data.

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

Default PDO Mapping Scheme

If no PDO-mapping is specified in the network configuration tool, the module will fall back on a simple default mapping scheme with 4 TPDOs and 4 RPDOs, with one ADI in each PDO.

- RPDO Default COB IDs

RPDO no.	Default COB ID	Default Transmission Type	Description
1	200h + Node ID	254	Default enabled according to DS301
2	300h + Node ID		
3	400h + Node ID		
4	500h + Node ID		
5...36	8000 0580h		Default Disabled

- TPDO Default COB IDs

TPDO no.	Default COB ID	Default Transmission Type	Description
1	180h + Node ID	254	Default enabled according to DS301
2	280h + Node ID		
3	380h + Node ID		
4	480h + Node ID		
5...36	8000 0500h		Default Disabled

If no ADIs have been mapped to Process Data, the RPDOs will be mapped to a dummy object entry (0005h) and the TPDOs will be mapped to object 1001h (Error Register).

When using the default PDO mapping scheme, only the first element of an array will be represented cyclically. There is however no such limitation if creating a custom PDO mapping scheme through the bus configuration tool.

Alternative/New PDO Mapping Scheme

The difference between this alternative behavior and the default behavior is that the module will try to use all available mapped ADIs or elements of an ADI to prepare all required PDOs. This scheme will try to map each ADI or element of and ADI in the same order as they are mapped to the process data. A maximum of 8 TPDOs and 8 RPDOs will be used for this mapping scheme. If an ADI (simple variable) or element of an ADI (array) is mapped and does not fit completely within a PDO, this ADI or element of an ADI will be moved to the next available PDO, as long as there are PDOs available. This can lead to that the 8 available default PDOs are not fully utilized or not sufficient for the theoretical maximum size of 64 bytes of process data.

This scheme is due to that it is not possible to divide an ADI (simple variable) or an element of an ADI (array) into two different PDOs, causing data inconsistency over the CANopen network and making it impossible to use for any other CANopen node on the network, with the current PDO mapping possibilities available for CANopen.

- RPDO Default COB IDs (Alternative/New)

RPDO No	Default COB ID		Default Transmission Type	Description
	Node ID 1.. 63	Node ID >= 64		
1	200h + Node-ID.	200h + Node-ID.	254	Default enabled according to DS301.
2	300h + Node-ID.	300h + Node-ID.	254	Default enabled according to DS301.
3	400h + Node-ID.	400h + Node-ID.	254	Default enabled according to DS301.
4	500h + Node-ID.	500h + Node-ID.	254	Default enabled according to DS301.
5	0x240 + Node ID	0x80000580	254	Dependant on Node ID the RPDO will be enabled or not. (If the PDO is required for transferring the mapped ADI as READ process data.)
6	0x340 + Node ID	0x80000580	254	
7	0x440 + Node ID	0x80000580	254	
8	0x540 + Node ID	0x80000580	254	
9-36	0x80000580	0x80000580	254	Default disabled

- TPDO Default COB IDs

TPDO No	Default COB ID		Default Transmission Type	Description
	Node ID 1.. 63	Node ID >= 64		
1	180h + Node-ID.	180h + Node-ID.	254	Default enabled according to DS301.
2	280h + Node-ID.	280h + Node-ID.	254	Default enabled according to DS301.
3	380h + Node-ID.	380h + Node-ID.	254	Default enabled according to DS301.
4	480h + Node-ID.	480h + Node-ID.	254	Default enabled according to DS301.
5	0x1C0 + Node ID	0x80000500	254	Dependant on Node ID the TPDO will be enabled or not. (If the PDO is required for transferring the mapped ADI as WRITE process data.)
6	0x2C0 + Node ID	0x80000500	254	
7	0x3C0 + Node ID	0x80000500	254	
8	0x4C0 + Node ID	0x80000500	254	
9-36	0x80000500	0x80000500	254	Default disabled

If there are no ADIs mapped during setup:
the first four RPDOs will be mapped with one Dummy Object (0005h)
the first four TPDO's will be mapped with the object 1001h (Error Register)
the transmission type will be 254 and enabled

PDO Triggering Modes

The module supports two triggering modes:

- Event Driven

  Message transmission is triggered by:

Transmission Type	Description	Notes
254/255	COS	When Process data have been changed. (The performance will be dependent on the number of PDO's using COS)
1…240	Cyclic Synchronous	For synchronous this is the expiration of the specified transmission period, synchronized by the reception of the SYNC object. The data will be synchronized only to the module (current process data in buffer) and not all the way down to the application.
0	Acyclic Synchronous	A transmission type of zero means that the message shall be transmitted synchronously with the SYNC object but not periodically. Only on when COS is fulfilled (SYNC & COS).

- Timer Driven

  Message transmission is either triggered by the occurrence of a device-specific event (COS) or if a specified has elapsed without the occurrence of the event.

Transmission Type	Description	Notes
254/255	COS/Timer	Message transmission is either triggered by the occurrence of a device-specific event (COS) or if a specified time has elapsed without occurrence of the event.

4.3 Device Address & Baud Rate Configuration

4.3.1 General

The CANopen baud rate and device 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 30 Software Design Guide must be followed at all times.

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

4.3.2 Layer Setting Services (LSS)

The module supports the Layer Setting Service (LSS). This service can be used to set the Baud Rate and Device Address via the network, and may address the module by its Vendor-ID, Product Code, Revision number and serial number.

It is possible to enforce LSS during startup by setting the instance Device Address (01h) to 255, see *Network Configuration Object (04h), p. 24*

4.4 Network Reset Handling

4.4.1 Reset Node

Upon receiving a Reset Node request from the network, the module 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.

4.4.2 Reset Communication

Upon receiving a Reset Communication request from the network, the module 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.

4.4.3 Restore Manufacturer Parameters to Default

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

See also *Standard Objects, p. 14*, entry 1011h (Restore Parameters)

5 Object Dictionary (CANopen)

5.1 Standard Objects

5.1.1 General

The standard object dictionary is implemented according to the DS302 specification (v4.02) from CiA (CAN in Automation). Note that certain object entries correspond to settings in the CANopen Object (FBh), and the Diagnostic Object (02h).

5.1.2 Object Entries

Index	Object Name	Sub-Index	Description	Type	Access	Notes
0005h	Dummy Object	00h	Dummy Object	U8	WO	-
0006h	Dummy Object	00h	Dummy Object	U16	WO	-
0007h	Dummy Object	00h	Dummy Object	U32	WO	-
1000h	Device Type	00h	Device Type	U32	RO	0000 0000h (No profile)
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. However, an extra event is reserved for internal errors etc.
1003h	Pre-defined error field	00h	Number of errors	U8	RW	
		01h...06h	Error field	U32	RO	
1005h	COB-ID Sync	00h	COB-ID Sync	U32	RW	Default value is 0000 0080h The Anybus CompactCom 30 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*. Object 1009h must be enabled, see *CANopen Object (FBh), p. 26*. (If not enabled, any access to this object will generate an error).
1009h	Manufacturer hardware version	00h	Manufacturer hardware version	Visible string	RO	
100Ah	Manufacturer software version	00h	Manufacturer software version	Visible string	RO	
100Ch	Guard time	00h	Guard time	U16	RW	-
100Dh	Life time factor	00h	Life time factor	U8	RW	-

Index	Object Name	Sub-Index	Description	Type	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 1.5 seconds 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.
		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. 13*
1014h	COB ID EMCY	00h	COB ID EMCY	U32	RO	-
1015h	Inhibit Time EMCY	00h	Inhibit Time EMCY	U16	RW	Default value is 0000h
1016h	Consumer Heartbeat Time	00h	Number of entries	U8	RO	01h
		01h	Consumer Heartbeat Time	U32	RW	Node ID + Heartbeat Time. Value must be a multiple of 1 ms.
1017h	Producer Heartbeat Time	00h	Producer Heartbeat Time	U16	RW	-
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 ... 1423h	Receive PDO parameter	00h	Largest sub-index supported	U8	RO	02h
		01h	COB ID used by PDO	U32	RW	-
		02h	Transmission type.	U8	RW	-

Index	Object Name	Sub-Index	Description	Type	Access	Notes
1600h ... 1623h	Receive PDO mapping	00h	No. of mapped application objects in PDO	U8	RW	-
		01h	Mapped object #1	U32	RW	-
		02h	Mapped object #2	U32	RW	-
		03h	Mapped object #3	U32	RW	-
		04h	Mapped object #4	U32	RW	-
		05h	Mapped object #5	U32	RW	-
		06h	Mapped object #6	U32	RW	-
		07h	Mapped object #7	U32	RW	-
		08h	Mapped object #8	U32	RW	-
1800h ... 1823h	Transmit PDO parameter	00h	Largest sub-index supported	U8	RO	05h
		01h	COB ID used by PDO	U32	RW	-
		02h	Transmission type	U8	RW	-
		03h	Inhibit time	U16	RW	-
		05h	Event Timer (ms)	U16	RW	-
1A00h ... 1A23h	Transmit PDO mapping	00h	No. of mapped application objects in PDO	U8	RW	-
		01h	Mapped object #1	U32	RW	-
		02h	Mapped object #2	U32	RW	-
		03h	Mapped object #3	U32	RW	-
		04h	Mapped object #4	U32	RW	-
		05h	Mapped object #5	U32	RW	-
		06h	Mapped object #6	U32	RW	-
		07h	Mapped object #7	U32	RW	-
		08h	Mapped object #8	U32	RW	-

5.2 Manufacturer Specific Objects

5.2.1 General

Each object entry in the manufacturer specific range (2001h...5FFFh) 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.

> **!** 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.

5.2.2 Translation of Status Codes

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

ABCC Status Code	CANopen Abort Code #	CANopen Code Description
Reserved	N.A.	-
Fragmentation error (serial mode)	N.A.	-

ABCC Status Code	CANopen Abort Code #	CANopen Code Description
Invalid message format	N.A.	-
Unsupported object	0602 0000h	Object does not exist in the object dictionary.
Unsupported instance	0602 0000h	Object does not exist in the object dictionary.
Unsupported Command	0604 0043h	General parameter incompatibility reason.
Invalid CmdExt[0]	0602 0000h	Object does not exist in the object dictionary. (ADI access).
Invalid CmdExt[1]	0609 0011H	Sub-index does not exist. (ADI access).
Attribute not settable	0601 0002h	Attempt to write a read only object
Attribute not gettable	0601 0001h	Attempt to read a write only object
Too Much Data	0607 0012h	Data type does not match, length of service parameter too high
Not Enough Data	0607 0013h	Data type does not match, length of service parameter too low
Out of range	0609 0030h	Value range of parameter exceeded (only for write access)
Invalid state	0800 0022h	Data cannot be transferred or stored to the application because of the present device state.
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)
Object Specific Error	0800 0000h	General error

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

5.2.3 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
SINT64	INTEGER64
UINT64	UNSIGNED64
FLOAT	REAL32

- ADIs with multiple elements are represented as arrays, with the exception of CHAR, which will always be represented as VISIBLE STRING.
- Single element ADIs are represented as a simple variable, with the exception of CHAR, which will always be represented as VISIBLE STRING.

5.2.4 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 DS302 (v4.02)

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	
...	...	...	...	...	...	...
5FFFh	ADI 3FFFh	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	

6 Anybus Module 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 30 CANopen.

Standard Objects:

6.2 Anybus Object (01h)

Category

Basic

Object Description

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

Supported Commands

Object:	Get_Attribute
Instance:	Get_Attribute
	Set_Attribute
	Get_Enum_String

Object Attributes (Instance #0)

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

Instance Attributes (Instance #1)

Basic

#	Name	Access	Type	Value
1	Module type	Get	UINT16	0401h (Standard Anybus CompactCom 30)
2... 11	-	-	-	See the general Anybus CompactCom 30 Software Design Guide for further information.
12	LED colors	Get	struct of: UINT8 (LED1A) UINT8 (LED1B) UINT8 (LED2A) UINT8 (LED2B)	Value: Color: 01h Green 02h Red 01h Green 02h Red
13...16	LED status	Get	UINT8	See the general Anybus CompactCom 30 Software Design Guide for further information.

6.3 Diagnostic Object (02h)

Category

Basic, Extended

Object Description

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

Supported Commands

Object:	Get_Attribute
	Create
	Delete
Instance:	Get_Attribute

Object Attributes (Instance #0)

#	Name	Access	Type	Value	Description
1... 4	-	-	-		See the general Anybus CompactCom 30 Software Design Guide for further information.
11	Max no. of instances	Get	UINT16	5 (6)	5 instances, i.e. diagnostic events, can be created. An internal 6th instance is reserved for internal CANopen diagnostics.

Instance Attributes (Instance #1...n)

Basic

#	Name	Access	Type	Value
1	Severity	Get	UINT8	See below
2	Event Code	Get	UINT8	

Extended

#	Name	Access	Type	Value
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.

An internal 6th instance is reserved for internal CANopen diagnostics. This includes the following EMCY error codes:

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	The saved PDO configuration has no corresponding process data map.

6.4 Network Object (03h)

Category

Basic

Object Description

For more information, consult the general Anybus CompactCom 30 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 Consult the general Anybus CompactCom 30 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 30 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
8...10	(reserved)	-	-	(not used)
11... 255	Network specific	-	-	

6.5 Network Configuration Object (04h)

Category

Basic

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, Device Address)

Basic

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	“Device Address”
2	Data type	Get	UINT8	04h (= UINT8)
3	Number of elements	Get	UINT8	01h (one element)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	UINT8	Value: Meaning: 1... 127: CANopen device address value 255 Enforce LSS at startup (Note that this value may be updated by the LSS service, see *Layer Setting Services (LSS), p. 13*)

Instance Attributes (Instance #2, Baud Rate)

Basic

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	“Baud Rate”
2	Data type	Get	UINT8	08h (= ENUM)
3	Number of elements	Get	UINT8	01h (one element)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	ENUM	Value: ENUM string: 0: “10 kbps” 1: “20 kbps” 2: “50 kbps” 3: “reserved” (Firmware versions 3.xxx) “100 kbps” (Firmware versions previous to 3.0) 4: “125 kbps” 5: “250 kbps” 6: “500 kbps” 7: “800 kbps” 8: “1 Mbps” 9: “Auto” 10: “LSS” (default) (Note that this value may be updated by the LSS service, see *Layer Setting Services (LSS), p. 13*)

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

7 Host Application Objects

7.1 CANopen Object (FBh)

Category

Basic, extended

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 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 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)

Basic

#	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 000Ah	
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. 24 bytes)	“Anybus-CC CANopen”	Replaces object entry 1008h (Manufacturer Device Name)

Extended

#	Name	Access	Data Type	Default Value	Comment
7	Manufacturer Hardware Version	Get	Array of CHAR (Max. 24 bytes)	Object entry 1009h not implemented	Specifies the value of object entry 1009h (Manufacturer Hardware Version)
8	Manufacturer Software Version	Get	Array of CHAR (Max. 24 bytes)	x.yy	Replaces object entry 100Ah (Manufacturer Software Version)
9 - 13	-	-	-	-	(reserved)
14	Default PDO map configuration	Get	UINT8	00h	00h: Standard default PDO mapping (minimalistic mapping) 01h: Use all available ADIs for default mapping (no consistency checking of the mapping is done). This must be maintained by the application by ADI mapping order and ADI data typed mapped. 02h - FFh (reserved)

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] - *or* - [Node guarding 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 starts reading out the startup values from the ADIs to use these as initial Read Process Data. 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 Guarding or Heart beat error occur 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 node guarding and 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.

B.3 Application Watchdog Timeout Handling

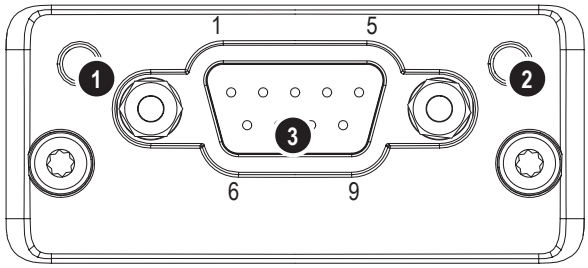
At the time of writing, no Application Watchdog functionality is supported by the module.

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 DR303-3 (CiA).

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.
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.
Red, double flash	Error Control Event	A guard- (NMT-Slave or NMT-master) or 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 30 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 30 CANopen requires a regulated 3.3 V power source as specified in the general *Anybus CompactCom 30 Hardware Design Guide*.

C.3.2 Power Consumption

TheAnybus CompactCom 30 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 30 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 30 Hardware Design Guide, and not on the exact power requirements of a single product.

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. However, in any case, the Anybus CompactCom 30 CANopen will remain as a Class A module.

C.4 Environmental Specification

Consult the *Anybus CompactCom 30 Hardware Design Guide* for further information.

C.5 EMC Compliance

Consult the *Anybus CompactCom 30 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 30 CANopen.

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

For further information, please consult the Anybus CompactCom 30 Software Design Guide.

D.2 Process Data

D.2.1 Overview

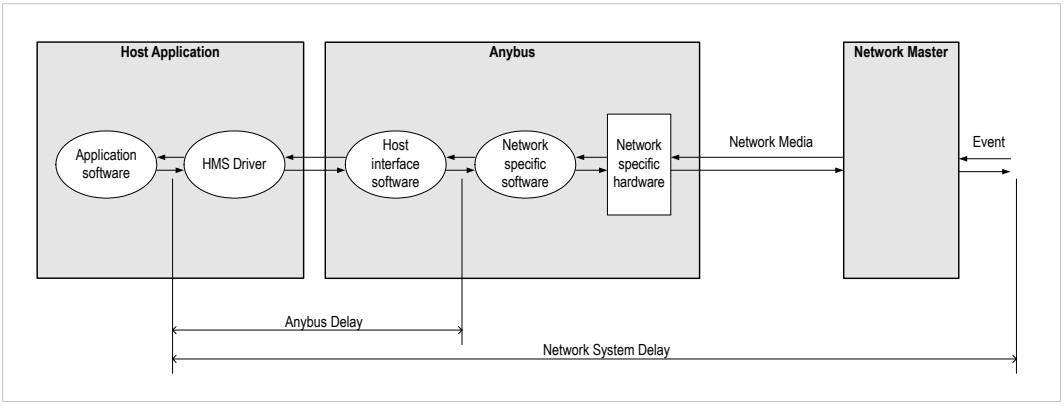


Fig. 1

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

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

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

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

The Write Process Data Delay (labelled “Anybus delay” in the figure) is defined as the time measured from the point the data is available from the host application (just before the data is

written from the host application to the driver), to the point where the new data has been forwarded to the network buffer by the Anybus host interface software.

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

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

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

Parameter	Description	Typ.	Max.	Unit.
T9	Network System Read Process Data delay, 8 ADIs (single UINT8)	1.3	1.9	ms
T10	Network System Read Process Data delay, 16 ADIs (single UINT8)	1.1	1.8	ms
T11	Network System Read Process Data delay, 32 ADIs (single UINT8)	1.3	1.8	ms

Conditions:

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

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

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

Parameter	Description	Min.	Max.	Unit.
T15	Network System Write Process Data delay, 8 ADIs (single UINT8)	0.4	1.5	ms
T16	Network System Write Process Data delay, 16 ADIs (single UINT8)	0.4	1.4	ms
T17	Network System Write Process Data delay, 32 ADIs (single UINT8)	0.5	1.7	ms

Conditions:

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

Box 4126
300 04 Halmstad, Sweden

info@hms.se