

# Network Interface Appendix

# Anybus<sup>®</sup> CompactCom EtherCAT

Doc.Id. HMSI-168-65  
Rev. 2.22

---

# Important User Information

This document is intended to provide a good understanding of the functionality offered by EtherCAT. The document only describes the features that are specific to the Anybus CompactCom 30 EtherCAT. For general information regarding the Anybus CompactCom, consult the Anybus CompactCom design guides.

The reader of this document is expected to be familiar with high level software design, and communication systems in general. The use of advanced EtherCAT-specific functionality may require in-depth knowledge in EtherCAT networking internals and/or information from the official EtherCAT specifications. In such cases, the people responsible for the implementation of this product should either obtain the EtherCAT specification to gain sufficient knowledge or limit their implementation in such a way that this is not necessary.

## Liability

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

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

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

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

## Intellectual Property Rights

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

## Trademark Acknowledgements

Anybus® is a registered trademark of HMS Industrial Networks AB. All other trademarks are the property of their respective holders.



EtherCAT® is registered trademark and patented technology, licensed by  
Beckhoff Automation GmbH, Germany

<b>Warning:</b>	This is a class A product. In a domestic environment this product may cause radio interference in which case the user may be required to take adequate measures.
<b>ESD Note:</b>	This product contains ESD (Electrostatic Discharge) sensitive parts that may be damaged if ESD control procedures are not followed. Static control precautions are required when handling the product. Failure to observe this may cause damage to the product.

## P. About This Document

For more information, documentation etc., please visit the HMS website, “[www.anybus.com](http://www.anybus.com)”.

### P.1 Related Documents

Document	Author
Anybus CompactCom 30 Software Design Guide	HMS
Anybus CompactCom 30 Hardware Design Guide	HMS
Anybus CompactCom Software Driver User Guide	HMS
IEC 61158-6	IEC
CiA Draft Standard 301 v4.02	CAN in Automation

### P.2 Document History

#### Summary of Recent Changes (2.21 ... 2.22)

**Note:** The changes described below, are valid from firmware rev. 1.06. Please refer to Network Interface Appendix rev. 2.06 or earlier for previous functionality.

Change	Page(s)
Moved front view and connector information to technical specification	34
Removed note for recommendations on what attributes to implement in EtherCAT object	29
Added info to RJ-45 connector	35

#### Revision List

Revision	Date	Author(s)	Chapter(s)	Description
1.00	2008-04-09	PeP	-	First official version
1.01	2008-10-24	HeS	B	Minor Update
1.02	2009-08-11	KeL	Preface	Minor update
1.03	2009-09-28	KeL	1, 2, 4, 5	Minor update
2.00	2010-04-14	KeL	All	Change of concept
2.01	2011-02-09	KeL	P, 1, 2, 3, 6	Minor update
2.02	2011-04-08	KeL	TM info	Minor update
2.03	2011-04-18	KeL	2	Minor update
2.04	2011-08-10	KaD	P, 5, A	Minor update and additions
2.05	2012-01-26	KeL	2, 3, 5	Minor update
2.06	2012-02-10	KaD	E	Minor update
2.07	2012-02-24	KeL	3, 5	Minor update
2.08	2012-05-04	KeL	2, 3	Minor corrections
2.10	2012-09-12	KeL	1	Updates
2.20	2013-02-18	KeL	1, 5	Updated with brick version and minor correction
2.21	2013-05-17	KeL	1	Updated brick connector
2.22	2015-03-02	KeL	6, D	Minor update

## P.3 Conventions & Terminology

The following conventions are used throughout this manual:

- Numbered lists provide sequential steps
- Bulleted lists provide information, not procedural steps
- The terms ‘Anybus’ or ‘module’ refers to the Anybus CompactCom module.
- The terms ‘host’ or ‘host application’ refers to the device that hosts the Anybus module.
- Hexadecimal values are written in the format NNNNh or 0xNNNN, where NNNN is the hexadecimal value.
- A byte always consists of 8 bits.

## P.4 Support

For general contact information and support, please refer to the contact and support pages at [www.anybus.com](http://www.anybus.com).

# Table of Contents

<b>Preface</b>	<b>About This Document</b>
	Related Documents ..... 6
	Document History ..... 6
	Conventions & Terminology ..... 7
	Support..... 7
<b>Chapter 1</b>	<b>About the Anybus CompactCom EtherCAT</b>
	General..... 4
	Features ..... 4
<b>Chapter 2</b>	<b>Tutorial</b>
	Introduction ..... 5
	Fieldbus Conformance Notes ..... 5
	Conformance Test Guide..... 5
	<i>Reidentifying Your Product</i> ..... 6
	<i>Factory Default Reset</i> ..... 6
<b>Chapter 3</b>	<b>Basic Operation</b>
	General Information ..... 7
	<i>Software Requirements</i> ..... 7
	<i>EtherCAT Slave Interface File</i> ..... 8
	<i>Device Identity</i> ..... 8
	EtherCAT Implementation Details ..... 9
	<i>General Information</i> ..... 9
	<i>Sync Managers</i> ..... 9
	<i>FMMUs</i> ..... 9
	<i>Addressing Modes</i> ..... 9
	<i>Watchdog Functionality</i> ..... 10
	<i>Implemented Services</i> ..... 11
	CANopen Implementation Details..... 12
	<i>General Information</i> ..... 12
	<i>Implemented Services</i> ..... 12
	Data Exchange..... 13
	<i>Application Data (ADI)</i> ..... 13
	<i>Process Data</i> ..... 13
	Network Reset Handling..... 14
	<i>Reset Node</i> ..... 14
	<i>Restore Manufacturer Parameters to Default</i> ..... 14
	Station Alias (Node Address) ..... 14
	Device ID1 ..... 14

---

<b>Chapter 4</b>	<b>Object Dictionary (CANopen over EtherCAT)</b>	
	Standard Objects .....	15
	<i>General</i> .....	15
	<i>Object Entries</i> .....	15
	Manufacturer Specific Objects .....	17
	<i>General</i> .....	17
	<i>Network Data Format</i> .....	17
	<i>Object Entries</i> .....	18
<b>Chapter 5</b>	<b>Anybus Module Objects</b>	
	General Information .....	19
	Anybus Object (01h).....	20
	Diagnostic Object (02h) .....	22
	Network Object (03h).....	24
	Network Configuration Object (04h).....	26
<b>Chapter 6</b>	<b>Host Application Objects</b>	
	General Information .....	28
	EtherCAT Object (F5h).....	29
<b>Appendix A</b>	<b>Miscellaneous</b>	
	Extended LED Functionality .....	31
<b>Appendix B</b>	<b>Categorization of Functionality</b>	
	Basic.....	32
	Extended.....	32
	Advanced .....	32
<b>Appendix C</b>	<b>Implementation Details</b>	
	SUP-Bit Definition .....	33
	Anybus State Machine .....	33
	Application Watchdog Timeout Handling .....	33
<b>Appendix D</b>	<b>Technical Specification</b>	
	Front View.....	34
	Network Connector, Brick Version.....	36
	Protective Earth (PE) Requirements .....	37
	Power Supply .....	37
	Environmental Specification .....	37
	EMC Compliance.....	37

---

# Appendix E Timing & Performance

- General Information ..... 38
- Process Data..... 39
  - Overview ..... 39
  - Anybus Read Process Data Delay (Anybus Delay)..... 39
  - Anybus Write Process Data Delay (Anybus Delay)..... 39
  - Network System Read Process Data Delay (Network System Delay)..... 40
  - Network System Write Process Data Delay (Network System Delay)..... 40

# 1. About the Anybus CompactCom EtherCAT

## 1.1 General

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

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.

## 1.2 Features

- CANopen over EtherCAT (CoE)
- RJ45 connectors
- Brick version
- DS301 compliant
- Galvanically isolated bus electronics
- Network Identity customization
- EMCY support
- Up to 16383 ADIs can be accessed from the network as Manufacturer Specific Objects.
- Up to 256 bytes of fast cyclic I/O in each direction
- EtherCAT Slave Interface file<sup>1</sup> provided by HMS

---

1. In earlier versions of the module, this file is called Device Description File (DDF).

## 2. Tutorial

### 2.1 Introduction

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

### 2.2 Fieldbus Conformance Notes

- This product is pre-certified for network conformance. This is done to ensure that the final product *can* be certified, but it does not mean that the final product will not require recertification.
- The Anybus CompactCom 30 EtherCAT module has a Secondary Vendor ID by default. This ID must be replaced when an indesign with the module is made as the Secondary Vendor ID can not be used for conformance testing.
- The EtherCAT Technology Group (ETG) demands that each vendor of products supporting EtherCAT, use their own EtherCAT Vendor ID for the end product. Vendor IDs are obtained free of charge from the EtherCAT Technology Group (ETG) (membership in the ETG is also free of charge). Vendors of CANopen products using a custom Vendor ID should apply for the same ID from the ETG.

Contact HMS for further information.

### 2.3 Conformance Test Guide

When using the default settings of all parameters, the Anybus CompactCom EtherCAT module is pre-certified 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 ESI file, supplied by HMS, will require a certification. A Vendor ID can be obtained from ETG and is compulsory for certification. This section provides a guide for successful conformance testing your product, containing the Anybus CompactCom EtherCAT module, to comply with the demands for network certification set by the ETG.

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

---

**IMPORTANT:** *This section provides guidelines and examples of what is needed for certification. Depending on the functionality of your application, there may be additional steps to take. Please contact HMS Industrial Networks at [www.anybus.com](http://www.anybus.com) for more information.*

### 2.3.1 Reidentifying Your Product.

After successful setting of the “Setup Complete” attribute in the Anybus Object (01h), the Anybus module asks for identification data from the EtherCAT Host Object (F5h). Therefore, the attributes listed below shall be implemented and proper values returned.

Object/Instance	Attribute	Explanation	Default	Customer sample	Comment
EtherCAT Host Object (F5h), Instance 1	#1, Vendor ID	With this attribute you set the Vendor ID of the device.	Vendor ID: 0000 001Bh	Vendor ID: 1111 1111h	This information must match the keyword values of the “Vendor” section in the ESI file.
EtherCAT Host Object (F5h), Instance 1	#2, Product Code	With this attribute you set the Product Code of the device.	Product Code: 0000 0034h	Product Code: 2222 2222h	This information must match the keyword values of the “Device” section in the ESI file.
EtherCAT Host Object (F5h), Instance 1	#3, Major Revision	With this attribute you set the Major Revision of the device.		1	
EtherCAT Host Object (F5h), Instance 1	#4, Minor Revision	With this attribute you set the Minor Revision of the device.		0	
EtherCAT Host Object (F5h), Instance 1	#5, Serial Number	With this attribute you set the Serial Number of the device.		12345678h	
EtherCAT Host Object (F5h), Instance 1	#6, Manufacturer Device Name	With this attribute you set the Manufacturer Device Name of the device.	“Anybus-CC EtherCAT”	“Widget”	This information must match the keyword values of the “Device” section in the ESI file.
EtherCAT Host Object (F5h), Instance 1	#7, Manufacturer Hardware Version	With this attribute you set the Manufacturer Hardware Version of the device.		“1.00”	

### 2.3.2 Factory Default Reset

#### Reset command to Application Object (FFh) must be supported

When Anybus CompactCom 30 EtherCAT modules are delivered, they are required to be in their “Factory Default” state. When a Factory Default Reset command is received from the network, the Anybus module will erase all non-volatile information and inform the host application that a reset to factory default is required. This is done by sending a Reset command to the Application Object (FFh) of the host (Factory Default). For more details, please consult the Anybus CompactCom 30 Software Design Guide.

## 3. Basic Operation

### 3.1 General Information

#### 3.1.1 Software Requirements

No additional network support code needs to be written in order to support the Anybus CompactCom EtherCAT, however due to the nature of the EtherCAT networking system certain restrictions must be taken into account:

- Only ADIs with instance numbers less than 16384 can be accessed from the network.
- When mapping to ADIs, there's a limit of 254 elements or 256 bytes, whichever comes first, that can be mapped in either direction.
- There is no equivalent to reset type 00h ('Power-on reset') on EtherCAT.
- The flexible nature of the Anybus concept allows the application to modify the behavior on EtherCAT in ways which contradict the generic EtherCAT Slave Information file or in other ways voids network certification. Those responsible for the implementation of the final product should ensure that their level of implementation matches their own requirements and policies regarding network certification and interoperability.
- The use of advanced EtherCAT-specific functionality may require in-depth knowledge in EtherCAT networking internals and/or information from the official EtherCAT specifications. In such cases, those responsible for the implementation of the product should either obtain the EtherCAT specification to gain sufficient knowledge or limit their implementation in such a way that this is not necessary.

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

### 3.1.2 EtherCAT Slave Interface File

Each device on EtherCAT is associated with a EtherCAT Slave Interface (ESI) file<sup>1</sup> in XML format, which holds a description of the device and its functions.

HMS supplies a generic ESI file which can serve as a basis for new implementations. However, due to the flexible nature of the Anybus CompactCom concept, it is possible to alter the functionality of the module in ways which contradicts the information in this file. This may cause trouble if the master expects the configuration stated in the file. In some cases, these problems can be rectified by the end user by manually changing I/O parameters etc. To ensure interoperability and to reduce the complexity for the end user, create a custom ESI file to match the final implementation of the product.

The EtherCAT Technology Group (ETG) requires that the Vendor ID is changed to reflect the vendor of the end product. The following scenarios, among others, may require additional changes to the EtherCAT Slave Interface file.

- The use of a custom Product Code
- Slow application response times. Explicit requests should be handled within 1ms in order to comply with the generic ESI file supplied by HMS. This may not be sufficient for a slow serial link with a substantial amount of I/O (in such case, the mailbox timeout value in the file needs to be increased accordingly).

Note that deviations from the generic ESI file requires the use of custom Product Codes apart from the required custom Vendor ID.

See also...

- “EtherCAT Slave Interface file provided by HMS” on page 4

### 3.1.3 Device Identity

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

Object Entry	Value
Vendor ID	E000 001Bh <sup>a</sup> (HMS Industrial Networks Secondary Vendor ID, has to be replaced by Vendor ID of end product vendor.)
Product Code	0000 0034h (Anybus CompactCom EtherCAT)
Device Name	'Anybus-CC EtherCAT'
Serial Number	(Assigned during manufacturing)

a. For firmware revision 1.02 and later.

By implementing support for the EtherCAT Object (F5h), the module can be customized to appear as a vendor specific implementation rather than a generic Anybus device. For the end product to pass the ETG performance tests and be certified, a separate Vendor ID has to be requested from ETG.

See also...

- “Fieldbus Conformance Notes” on page 5
- “EtherCAT Object (F5h)” on page 29

---

1. In earlier versions of the module, this file is called Device Description File (DDF).

## 3.2 EtherCAT Implementation Details

### 3.2.1 General Information

The module implements a full EtherCAT slave with the following basic properties:

- Application Layer: CANopen
- FMMU Channels: 8
- SM Channels: 4
- RAM Size: 8kByte
- Bit-oriented FMMU operation

See also...

- “CANopen Implementation Details” on page 12

### 3.2.2 Sync Managers

The module features four Sync Managers:

- **Sync Manager 0**  
Used for mailbox write transfers (Master to Slave).  
The module has a fixed mailbox size of 276 bytes, corresponding to the maximum ADI size of 255 bytes plus relevant protocol headers and padding.
- **Sync Manager 1**  
Used for mailbox read transfers (Slave to Master).  
The module has a fixed mailbox size of 276 bytes, corresponding to the maximum ADI size of 255 bytes plus relevant protocol headers and padding.
- **Sync Manager 2**  
Contains the RxPDOs (in practice, Sync Manager 2 holds the Read Process Data).
- **Sync Manager 3**  
Contains the TxPDOs (in practice, Sync Manager 3 holds the Write Process Data).

### 3.2.3 FMMUs

There are eight FMMUs. The EtherCAT master can use the FMMUs freely for any purpose.

### 3.2.4 Addressing Modes

There are a number of different addressing modes which can be applied when communicating with EtherCAT slaves. As a full EtherCAT slave device, the module supports position addressing, node addressing and logical addressing.

### 3.2.5 Watchdog Functionality

Apart from the standard watchdog functionality, the following additional watchdogs are implemented:

- **PDI Watchdog**

This watchdog monitors the CPU in the module. Each access from the CPU to the ESC resets this watchdog.

**Note:** This watchdog is configured and enabled by the EtherCAT master.

- **Output I/O Sync Manager Watchdog**

If enabled, this watchdog monitors the PDO communication towards the Anybus module. If the master doesn't update the Read Process Data within the specified time period, this will trigger a timeout condition in the module, causing it to shift from OPERATIONAL to SAFE-OPERATIONAL. The supervision-bit (SUP) is also affected by this.

The sync manager watchdog is disabled by default in the module. This means that the module will not leave the state PROCESS\_ACTIVE if the communication with the master breaks down. To enable the sync manager watchdog by default, change ESI file. Make sure that the parameter Controlbyte has the value "#x64" and that Reg0420, that gives the timeout in ms, has a value > 0:

**Example:**

```
<Sm StartAddress="#x1000" ControlByte="#x64" Enable="1">Outputs</Sm>
...
<Reg0420>100</Reg0420>
```

The sync manager watchdog can always be disabled/enabled manually in the configuration tool for the master.

See also...

- "SUP-Bit Definition" on page 33

### 3.2.6 Implemented Services

The module implements the following EtherCAT services:

Service	Description
Auto increment physical read (APRD)	-
Auto increment physical write (APWR)	-
Auto increment Read Write (APRW)	-
Configured address read (FPRD)	-
Configured address write (FPWR)	-
Configured address Read Write (FPRW)	-
Broadcast Read (BRD)	-
Broadcast Write (BWR)	-
Logical Read (LRD)	-
Logical Write (LWR)	-
Logical Read Write (LRW)	-
Auto increment physical read multiple write (ARMW)	-
Configured read multiple write (FRMW)	-

## 3.3 CANopen Implementation Details

### 3.3.1 General Information

As mentioned previously, the module implements CANopen over EtherCAT. The object implementation is based on the DS301 communication profile.

See also...

- “Data Exchange” on page 13
- “Object Dictionary (CANopen over EtherCAT)” on page 15

### 3.3.2 Implemented Services

The module implements the following CANopen services:

Service	Description
SDO Download Expedited	Writes up to four octets to the slave
SDO Download Normal	Writes up to a negotiated number of octets to the slave
Download SDO Segment	Writes additional data if the object size exceeds the negotiated no. of octets.
SDO Upload Expedited	Reads up to four octets from the slave
SDO Upload Normal	Reads up to a negotiated number of octets from the slave
Upload SDO Segment	Reads additional data if the object size exceeds the negotiated no. of octets
Abort SDO Transfer	Server abort of service in case of an erroneous condition
Get OD List	Reads a list of available indices
Get Object Description	Reads details of an index
Get Entry Description	Reads details of a sub-index
Emergency	Reports unexpected conditions.

## 3.4 Data Exchange

### 3.4.1 Application Data (ADI)

Application Data Instances (ADIs) can be accessed from the network via dedicated object entries in the Manufacturer Specific range (2001h - 5FFFh). The SDO information protocol allows nodes to retrieve the name and data type of the ADI.

See also...

- “Manufacturer Specific Objects” on page 17

### 3.4.2 Process Data

ADIs mapped as Process Data will 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 and cannot be changed from the network during runtime.

The module supports one TPDO and one RPDO, each supporting up to 254 SDO mappings. Each SDO equals one Process Data mapped ADI element (i.e. mapping multiple element ADIs will result in multiple SDO mappings).

**Note:** Preferably, the EtherCAT Slave Information file should be altered to match the actual Process Data implementation. This is not a general requirement, but it has a positive impact on compatibility with 3rd party masters.

See also...

- “Standard Objects” on page 15
- “Manufacturer Specific Objects” on page 17

## 3.5 Network Reset Handling

### 3.5.1 Reset Node

There is no equivalent to reset type 00h ('Power-on reset') on EtherCAT.

### 3.5.2 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" on page 15, entry 1011h ('Restore Parameters')

## 3.6 Station Alias (Node Address)<sup>1</sup>

The Station Alias (node address) range is 1... 65535. Address 0 indicates that the device has yet to be configured. The Station Alias is stored in the slave EEPROM and may be used by some masters as a node address.

The Anybus CompactCom 30 EtherCAT slave module does not support local configuration of the Station Alias. For most applications it is recommended to leave the Station Alias unchanged, but it is possible to assign each slave an address from the network.

## 3.7 Device ID<sup>1</sup>

The Device ID is used by the master to explicitly identify a slave. This is e.g. useful when changing a faulty device during runtime<sup>2</sup>. A preconfigured device can be entered into the network, and its Device ID can be set to the same Device ID as the faulty device was appointed.

It is also useful to prevent cable swapping when there are two or more identical devices on the network.

The Device ID range is 1... 65535. Address 0 indicates that the device has yet to be configured. The value can be set using the Network Configuration Object, instance 3.

See also...

- "Network Configuration Object (04h)" on page 26

---

1. This functionality, as described here, is valid from firmware rev. 1.06. Please refer to Network Interface Appendix rev. 2.06 or earlier for previous functionality.  
2. A so called HotConnect application.

## 4. Object Dictionary (CANopen over EtherCAT)

### 4.1 Standard Objects

#### 4.1.1 General

The standard object dictionary is implemented according to the DS301 communication profile. Note that certain object entries correspond to settings in the EtherCAT Object (F5h), and the Diagnostic Object (02h).

#### 4.1.2 Object Entries

Index	Object Name	Sub-Index	Description	Type	Access	Notes
1000h	Device Type	00h	Device Type	U32	RO	0000 0000h (No profile)
1001h	Error register	00h	Error register	U8	RO	This information managed through the Diagnostic Object, see "Diagnostic Object (02h)" on page 22.
1003h	Pre-defined error field	00h	Number of errors	U8	RW	
		01h...05h	Error field	U32	RO	
1008h	Manufacturer device name	00h	Manufacturer device name	Visible string	RO	These entries are managed through the EtherCAT Object, which can optionally be implemented in the host application. See "EtherCAT Object (F5h)" on page 29.
1009h <sup>a</sup>	Manufacturer hardware version	00h	Manufacturer hardware version	Visible string	RO	
1011h	Restore parameters	00h	Largest sub index supported	U8	RO	01h
		01h	Restore all default parameters	U32	RW	-
1018h	Identity object	00h	Number of entries	U8	RO	Number of entries
		01h	Vendor ID	U32	RO	These entries are managed through the EtherCAT Object, which can optionally be implemented in the host application. See "EtherCAT Object (F5h)" on page 29.
		02h	Product Code	U32	RO	
		03h	Revision Number	U32	RO	
		04h	Serial Number	U32	RO	
1600h	Receive PDO mapping	00h	No. of mapped application objects in PDO	U8	RO	No. of mapped objects (0.. 254)
		01h	Mapped object #1	U32	RO	-
		02h	Mapped object #2	U32	RO	-
		...	...	...	...	-
		NNh	Mapped object #NN	U32	RO	-
1A00h	Transmit PDO mapping	00h	No. of mapped application objects in PDO	U8	RO	No. of mapped objects (0.. 254)
		01h	Mapped object #1	U32	RO	-
		02h	Mapped object #2	U32	RO	-
		...	...	...	...	-
		NNh	Mapped object #NN	U32	RO	-
1C00h	Sync Manager Communication Type	00h	Number of entries	U8	RO	4
		01h	Mailbox wr	U8	RO	1
		02h	Mailbox rd	U8	RO	2
		03h	Process Data out	U8	RO	3
		04h	Process Data in	U8	RO	4

Index	Object Name	Sub-Index	Description	Type	Access	Notes
1C12h	Sync Manager Rx PDO Assign	00h	No. of assigned PDOs	U8	RO	1
		01h	Assigned PDO	U16	RO	1600h
1C13h	Sync Manager Tx PDO Assign	00h	No. of assigned PDOs	U8	RO	1
		01h	Assigned PDO	U16	RO	1A00h
1C32h	SM output parameter	00h	Number of entries	U8	RO	1
		01h	Sync mode	U16	RO	0 (FREE_RUN)
1C33h	SM input parameter	00h	Number of entries	U8	RO	1
		01h	Assigned PDO	U16	RO	0 (FREE_RUN)

- a. This object must be enabled, see "EtherCAT Object (F5h)" on page 29 (If not enabled, any access to this object will generate an error).

## 4.2 Manufacturer Specific Objects

### 4.2.1 General

Each object entry in the manufacturer specific range (2001h... 5FFFh) corresponds to an instance (a.k.a. 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 error code returned in the response from the host application will be translated into the corresponding CANopen abort code.

**Important:** Since 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 Network Data Format

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

Anybus Data Type	Network Data Type	Anybus Data Type	Network Data Type
BOOL	UNSIGNED8	UINT32	UNSIGNED32
SINT8	INTEGER8	CHAR	VISIBLE STRING
SINT16	INTEGER16	ENUM	UNSIGNED8 or ENUM
SINT32	INTEGER32	SINT64	INTEGER64
UINT8	UNSIGNED8	UINT64	UNSIGNED64
UINT16	UNSIGNED16	FLOAT	REAL32

**Note 1:** ADIs with multiple elements are represented as arrays, with the exception of 'CHAR', which will always be represented as VISIBLE STRING.

**Note 2:** Single element ADIs are represented as a simple variable, with the exception of 'CHAR', which will always be represented as VISIBLE STRING.

### 4.2.3 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 one 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.

Index	Object Name	Sub-Index	Description	Type	Access	Notes
2001h	ADI 0001h	00h	Number of entries (NNh)	U8	RO	-
		01h	ADI value(s) (Attribute #5)	-	-	The data type and access rights of the ADI values are determined by the ADI itself.
		02h	<i>ADIs with multiple elements (i.e. arrays) are represented as multiple sub-indexes.</i>			
		...				
		...				
		NNh				
2002h	ADI 0002h	00h	ADI value (Attribute #5)	-	-	-
2003h	ADI 0003h	00h	Number of entries (NNh)	U8	RO	-
		01h	ADI value(s) (Attribute #5)	-	-	
		02h	<i>ADIs with multiple elements (i.e. arrays) are represented as multiple sub-indexes.</i>			
		...				
		...				
		NNh				
2004h	ADI 0004h	00h	ADI value (Attribute #5)	-	-	-
2005h	ADI 0005h	00h	Number of entries (NNh)	U8	RO	-
		01h	ADI value(s) (Attribute #5)	-	-	
		02h	<i>ADIs with multiple elements (i.e. arrays) are represented as multiple sub-indexes.</i>			
		...				
		...				
		NNh				
...	...	...	...	...	...	...
5FFFh	ADI 3FFFh	00h	Number of entries (NNh)	U8	RO	-
		01h	ADI value(s) (Attribute #5)	-	-	
		02h	<i>ADIs with multiple elements (i.e. arrays) are represented as multiple sub-indexes.</i>			
		...				
		...				
		NNh				

## **5. Anybus Module Objects**

### **5.1 General Information**

This chapter specifies the Anybus Module Object implementation in the module.

Standard Objects:

- “Anybus Object (01h)” on page 20
- “Diagnostic Object (02h)” on page 22
- “Network Object (03h)” on page 24
- “Network Configuration Object (04h)” on page 26

Network Specific Objects:

(none)

## 5.2 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)
2... 11	-	-	-	Consult 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... 15	-	-	-	Consult the general Anybus CompactCom 30 Software Design Guide for further information.

### Extended

#	Name	Access	Type	Value
16	GPIO configuration	Get/Set <sup>a</sup>	UINT16	Configuration of the host interface GPIO pins. See the table below.

a. Set access of attribute GPIO configuration is only valid in state SETUP.

### GPIO Configuration Settings

Value	Functionality	Description
0x0000	Standard	GIP[0..1] and GOP[0..1] are used as general input/output pins LED1[A..B] is used for RUN LED LED2[A..B] is used for ERR LED
0x0001	Extended LED functionality	GIP0 (red) and GIP1 (green) are used for EtherCAT LED port 1 GOP0 (red) and GOP1 (green) are used for EtherCAT LED port 2 LED1[A..B] is used for RUN LED LED2[A..B] is used for ERR LED

For more information, see

- “Extended LED Functionality” in “Appendix A” on page 31.

## 5.3 Diagnostic Object (02h)

### Category

Basic

### Object Description

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

### Supported Commands

Object:           Get\_Attribute  
                  Create  
                  Delete

Instance:         Get\_Attribute

### Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1... 4	-	-	-	Consult the general Anybus CompactCom 30 Software Design Guide for further information.
11	Max no. of instances	Get	UINT16	5 + 1

## Instance Attributes

### Basic

#	Name	Access	Type	Value
1	Severity	Get	UINT8	(below)
2	Event Code	Get	UINT8	
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:

1. 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

2. The Error Register (object entry 1001h) is set with the corresponding bit information
3. 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)				

**Note 1:** When creating a Major unrecoverable event, this will not end up as an EMCY-message on the bus, since this effectively forces the Anybus module to enter the EXCEPTION state.

**Note 2:** Bytes 0 and 1 (00h + Event Code) will be replaced by the value of attribute 3 if implemented.

## 5.4 Network Object (03h)

### Category

Basic

### Object Description

For more information regarding this object, consult the general Anybus CompactCom 30 Software Design Guide.

### Supported Commands

Object:                   Get\_Attribute

Instance:                Get\_Attribute  
                             Set\_Attribute  
                             Get\_Enum\_String  
                             Map\_ADI\_Write\_Area  
                             Map\_ADI\_Read\_Area

### Object Attributes (Instance #0)

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

### Instance Attributes (Instance #1)

#### Basic

#	Name	Access	Type	Value
1	Network type	Get	UINT16	0087h
2	Network type string	Get	Array of CHAR	'EtherCAT'
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 <sup>a</sup>
6	Read process data size	Get	UINT16	Current read process data size (in bytes) Updated on every successful Map_ADI_Read_Area <sup>a</sup>

#	Name	Access	Type	Value
7	Exception Information	Get	UINT8	<p>Additional information may be provided here when the module has entered the EXCEPTION state.</p> <p><u>Value:Meaning:</u></p> <p>00h (no additional information available)</p> <p>01h Invalid data type reported by application</p> <p>02h Error response to Get_Instance_Number_By_Order (Application Data Object)</p> <p>03h Error response to Get_Attribute: "Highest instance no." (Application Data Object)</p> <p>04h Error response to Get_Attribute: "Number of instances" (Application Data Object)</p> <p>05h Implementation error; "Highest instance no." is lower than "Number of instances" (Application Data Object)</p>
8... 10	(reserved)	-	-	Consult the general Anybus CompactCom 30 Software Design Guide for further information.

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

## 5.5 Network Configuration Object (04h)

This description is valid from firmware rev. 1.06. Please refer to Network Interface Appendix rev. 2.06 or earlier for previous functionality.

### Category

Extended

### Object Description

This object holds network specific configuration parameters that may be set by the end user. A reset command (factory default) issued towards this object will result in all instances being set to their default values.

Supported Commands

Object:	Get_Attribute Reset
Instance:	Get_Attribute Set_Attribute Get_Enum_String

### Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1	Name	Get	Array of CHAR	'Network configuration'
2	Revision	Get	UINT8	01h
3	Number of instances	Get	UINT16	0001h
4	Highest instance no.	Get	UINT16	0003h

## Instance Attributes (Instance #3, 'Device ID')

See also "Device ID1" on page 14.

### Extended

#	Name	Access	Type	Description
1	Name <sup>a</sup>	Get	Array of CHAR	'Device ID'
2	Data type	Get	UINT8	05h (= UINT16)
3	Number of elements	Get	UINT8	01h (one element)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	UINT16	1...65535:Valid network address 0:Device not configured (Default)

a. Multilingual, see "Multilingual Strings" on page 27.

## Multilingual Strings

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

Instance	English	German	Spanish	Italian	French
3	Device ID	Geräteadresse	ID Dispos.	ID Dispos.	ID appareil

## 6. Host Application Objects

### 6.1 General Information

This chapter specifies the host application object implementation in the module. The objects listed here may optionally be implemented within the host application firmware to expand the EtherCAT implementation.

Standard Objects:

- Application Object (see Anybus CompactCom 30 Software Design Guide)
- Application Data Object (see Anybus CompactCom 30 Software Design Guide)

Network Specific Objects:

- “EtherCAT Object (F5h)” on page 29

## 6.2 EtherCAT Object (F5h)

### Category

Basic, extended

### Object Description

This object implements EtherCAT 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 start-up; if an attribute is not implemented in the host application, simply respond with an error message (06h, “Invalid CmdExt[0]”). In such case, the module will use its default value.

If the module attempts to retrieve a value of an attribute not listed below, respond with an error message (06h, “Invalid CmdExt[0]”).

**Note:** To pass conformance tests, the end product has to have a Vendor ID valid for the end product vendor, see “EtherCAT Slave Interface file provided by HMS” on page 4.

See also...

- Anybus CompactCom 30 Software Design Guide, “Error Codes”

### Supported Commands

Object:                   Get Attribute

Instance:                Get Attribute

## Object Attributes (Instance #0)

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

## Instance Attributes (Instance #1)

### Basic

#	Name	Access	Type	Default Value	Comment
1	Vendor ID	Get	UINT32	0000 001Bh	These values replace the settings in object entry 1018h. (Identity Object)

### Extended

#	Name	Access	Type	Default Value	Comment
2	Product Code	Get	UINT32	0000 0034h	These values replace the settings in object entry 1018h. (Identity Object)
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 EtherCAT'	Replaces object entry 1008h (Manufacturer Device Name)
7	Manufacturer Hardware Version	Get	Array of CHAR (Max. 24 bytes)	x.yy	Specifies the value of object entry 1009h (Manufacturer Hardware Version)
9	ENUM ADIs	Get	Array of UINT16	-	By default, ENUMs will be translated to UNSIGNED8 on EtherCAT. By implementing this attribute, ENUMs will be translated to ENUMs on the bus as well. The attributes shall contain a sorted list of ADI instance numbers which are of type ENUM. If implementing this attribute, it is strongly recommended to also implement the optional Application Data Instance attribute #6 ('Max. Value') of all ENUM ADIs, since this improves performance and functionality of ENUMs on the bus significantly.

## A. Miscellaneous

### A.1 Extended LED Functionality

On the Anybus CompactCom EtherCAT module, only the RUN LED and the ERR LED are available through the application interface connectors (LED1[A..B] and LED2[A..B]). If needed, the two EtherCAT port LEDs can also be made available by enabling the extended LED functionality. Doing so will use GIP[0..1] for the EtherCAT port 1 LED and  $\overline{GOP}[0..1]$  for the EtherCAT port 2 LED.

To enable the extended LED functionality, the application needs to set the Anybus Object Instance 1 attribute 16 (GPIO configuration) to 0x0001 during state SETUP.

See the Anybus CompactCom Hardware Design Guide for Host Interface Signals.

#### GPIO mode description

		Signal			
		GIP[0..1]	$\overline{GOP}[0..1]$	LED1[A..B]	LED2[A..B]
GPIO Configuration	Value: 0x0000 (Default)	General purpose input	General purpose output	RUN LED	ERR LED
	Value: 0x0001 (Extended LED functionality)	EtherCAT LED Port 1 GIP0 (red) GIP1 (green)	EtherCAT LED Port 2 $\overline{GOP}$ 0 (red) $\overline{GOP}$ 1 (green)	RUN LED	ERR LED

**Note 1:** Enabling the extended LED functionality will cause both GIP[0..1] and  $\overline{GOP}[0..1]$  to function as outputs.

**Note 2:** Enabling the extended LED functionality will define both GIP[0..1] and  $\overline{GOP}[0..1]$  as active low. This means that LEDs will be lit when the corresponding pin is low.

**Note 3:** LED behavior is described in chapter 1. See “Front View” on page 34.

## **B. Categorization of Functionality**

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

### **B.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.

### **B.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.

### **B.3 Advanced**

The objects, attributes and services that belong to this group offer specialized and/or seldom used functionality. Most of the available network functionality is enabled and accessible. Access to the specification of the industrial network is normally required.

## C. Implementation Details

### C.1 SUP-Bit Definition

The supervised bit (SUP) indicates that the network participation is supervised by another network device. In the case of EtherCAT, this functionality is mapped to the SyncManager watchdog, which can be used to detect loss of communication with the master. The SyncManager watchdog is enabled by the master.

EtherCAT-specific interpretation:

SUP-bit	Interpretation
0	SyncManager Watchdog is disabled or not running.
1	SyncManager Watchdog is enabled and running.

**Note:** The watchdog and supervised bit (SUP) will not be available if the Read Process Data size is zero.

### C.2 Anybus State Machine

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

Anybus State	Corresponding EtherCAT State
WAIT_PROCESS	INIT or PRE-OPERATIONAL
ERROR	(‘Error Ind’-bit in ‘AL-Status’ is set)
PROCESS_ACTIVE	OPERATIONAL
IDLE	SAFE-OPERATIONAL
EXCEPTION	(EtherCAT interface is forced to INIT state, and the master is informed that a power cycle is required to resume communication)

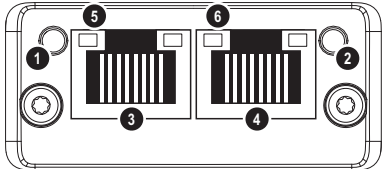
### C.3 Application Watchdog Timeout Handling

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

## D. Technical Specification

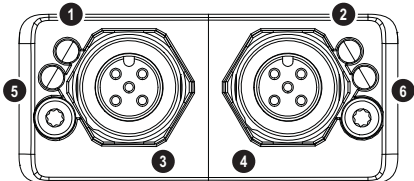
### D.1 Front View

#### Ethernet Connector

#	Item	
1	RUN LED <sup>a</sup>	
2	ERROR LED <sup>a</sup>	
3	EtherCAT (port 1)	
4	EtherCAT (port 2)	
5	Link/Activity (port 1)	
6	Link/Activity (port 2)	

a. The flash sequences for these LEDs are defined in DR303-3 (CiA).

#### M12 Connector

#	Item	
1	RUN LED <sup>a</sup>	
2	ERROR LED <sup>a</sup>	
3	EtherCAT (port 1)	
4	EtherCAT (port 2)	
5	Link/Activity (port 1)	
6	Link/Activity (port 2)	

a. The flash sequences for these LEDs are defined in DR303-3 (CiA).

#### RUN LED

This LED reflects the status of the CoE (CANopen over EtherCAT) communication.

LED State	Indication	Description
Off	INIT	CoE device in 'INIT'-state (or no power)
Green	OPERATIONAL	CoE device in 'OPERATIONAL'-state
Green, blinking	PRE-OPERATIONAL	CoE device in 'PRE-OPERATIONAL'-state
Green, single flash	SAFE-OPERATIONAL	CoE device in 'SAFE-OPERATIONAL'-state
Red <sup>a</sup>	(Fatal Event)	-

a. If RUN and ERR turns red, this indicates a fatal event, forcing the bus interface to a physically passive state. Contact HMS technical support.

#### ERR LED

This LED indicates EtherCAT communication errors etc.

LED State	Indication	Description
Off	No error	No error (or no power)
Red, blinking	Invalid configuration	State change received from master is not possible due to invalid register or object settings.

LED State	Indication	Description
Red, double flash	Application watchdog timeout	Sync manager watchdog timeout
Red <sup>a</sup>	Application controller failure	Anybus module in EXCEPTION

a. If RUN and ERR turns red, this indicates a fatal event, forcing the bus interface to a physically passive state. Contact HMS technical support.

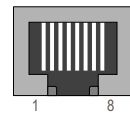
## Link/Activity

These LEDs indicate the EtherCAT link status and activity.

LED State	Indication	Description
Off	No link	Link not sensed (or no power)
Green	Link sensed, no activity	Link sensed, no traffic detected
Green, flickering	Link sensed, activity detected	Link sensed, traffic detected

## Ethernet Connector (RJ45)

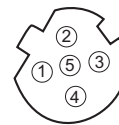
Pin	Signal	Notes
1	Tx+	-
2	Tx-	-
3	Rx+	-
4	-	Normally left unused; to ensure signal integrity, these pins are tied together and terminated to PE via a filter circuit in the module.
5	-	
6	Rx-	-
7	-	Normally left unused; to ensure signal integrity, these pins are tied together and terminated to PE via a filter circuit in the module.
8	-	



Anybus CompactCom 30 EtherCAT uses 100BASE-TX for communication.

## M12 Connector, Code D

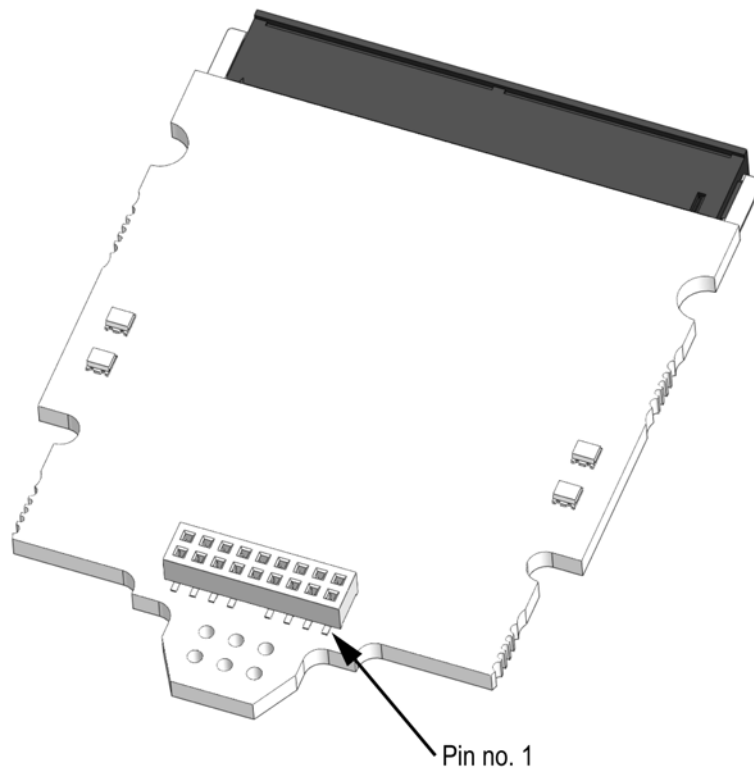
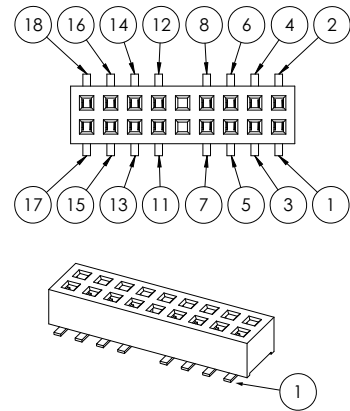
Pin	Signal	Notes
1	Tx+	Transmit positive
2	Rx+	Receive positive
3	Tx-	Transmit negative
4	Rx-	Receive negative
5 (Thread)	Shield	Shield



## D.2 Network Connector, Brick Version

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

Pin no.	Signal	Port
1	Shield	Output
2	TXD+	
3	TXD-	
4	Shield	
5	Shield	
6	RXD-	
7	RXD+	
8	Shield	
11	Shield	Input
12	TXD+	
13	TXD-	
14	Shield	
15	Shield	
16	RXD-	
17	RXD+	
18	Shield	



## D.3 Protective Earth (PE) Requirements

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

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

## D.4 Power Supply

### Supply Voltage

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

### Power Consumption

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

The current hardware design consumes up to 370 mA<sup>1</sup>.

**Note:** It is strongly advised to design the power supply in the host application based on the power consumption classifications described in the general Anybus CompactCom Hardware Design Guide, and not on the exact power requirements of a single product.

## D.5 Environmental Specification

Consult the Anybus CompactCom Hardware Design Guide for further information.

## D.6 EMC Compliance

Consult the Anybus CompactCom Hardware Design Guide for further information.

- 
1. Note that in line with HMS policy of continuous product development, we reserve the right to change the exact power requirements of this product without prior notification. Note however that in any case, the Anybus CompactCom EtherCAT will remain as a Class B module.

## E. Timing & Performance

### E.1 General Information

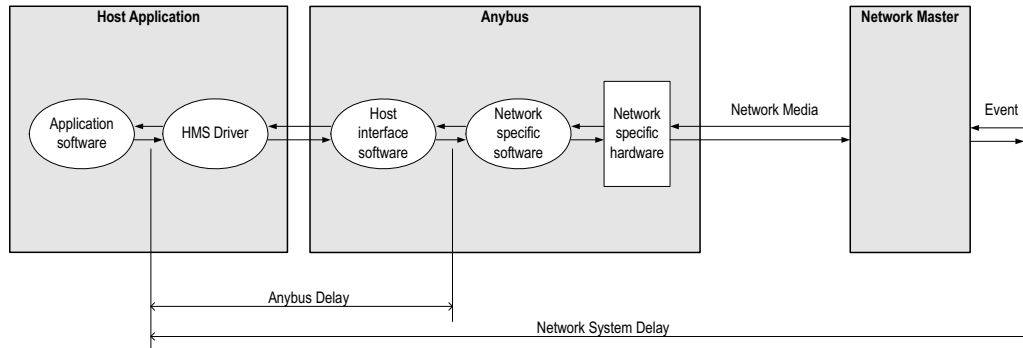
This chapter specifies timing and performance parameters that are verified and documented for the Anybus CompactCom 30 EtherCAT.

The following timing aspects are measured:

Category	Parameters	Page
Startup Delay	T1, T2	Please consult the Anybus CompactCom 30 Software Design Guide, App. B.
NW_INIT Delay	T3	
Telegram Delay	T4	
Command Delay	T5	
Anybus Read Process Data Delay (Anybus Delay)	T6, T7, T8	
Anybus Write Process Data Delay (Anybus Delay)	T12, T13, T14	
Network System Read Process Data Delay (Network System Delay)	T9, T10, T11	40
Network System Write Process Data Delay (Network System Delay)	T15, T16, T17	40

## E.2 Process Data

### E.2.1 Overview



### E.2.2 Anybus Read Process Data Delay (Anybus Delay)

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

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

### E.2.3 Anybus Write Process Data Delay (Anybus Delay)

The Write Process Data Delay (labelled ‘Anybus delay’ in the figure) is defined as the time measured from the point the data is available from the host application (just before the data is written from the host application to the driver), to the point where the new data has been forwarded to the network buffer by the Anybus host interface software.

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

## E.2.4 Network System Read Process Data Delay (Network System Delay)

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

Parameter	Description	Min.	Max.	Unit.
T9	Network System Read Process Data delay, 8 ADIs (single UINT8)	1.2	2.4	ms
T10	Network System Read Process Data delay, 16 ADIs (single UINT8)	1.2	2.4	ms
T11	Network System Read Process Data delay, 32 ADIs (single UINT8)	1.2	2.4	ms

### Conditions:

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

## E.2.5 Network System Write Process Data Delay (Network System Delay)

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

Parameter	Description	Min.	Max.	Unit.
T15	Network System Write Process Data delay, 8 ADIs (single UINT8)	1.2	2.4	ms
T16	Network System Write Process Data delay, 16 ADIs (single UINT8)	1.2	2.4	ms
T17	Network System Write Process Data delay, 32 ADIs (single UINT8)	1.2	2.4	ms

Conditions: as in "Network System Read Process Data Delay (Network System Delay)" on page 40.