

Anybus[®] CompactCom 40 EtherCAT

Network Guide

Doc.Id. HMSI-27-220
Rev. 1.30



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

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

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

Important User Information

This document is intended to provide a good understanding of the functionality offered by EtherCAT. The document only describes the features that are specific to the Anybus CompactCom 40 EtherCAT. For general information regarding the Anybus CompactCom 40, consult the Anybus CompactCom 40 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

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

Table of Contents

Preface	About This Document
	Related Documents 5
	Document History 5
	Conventions & Terminology 6
	Support..... 6
Chapter 1	About the Anybus CompactCom 40 EtherCAT
	General..... 7
	Features 7
Chapter 2	Basic Operation
	General Information 8
	<i>Software Requirements</i> 8
	<i>EtherCAT Slave Interface (ESI) File</i> 9
	<i>Device Identity</i> 9
	<i>File Access over EtherCAT (FoE)</i> 10
	EtherCAT Implementation Details 11
	<i>General Information</i> 11
	<i>EtherCAT Synchronization</i> 11
	<i>Sync Managers</i> 11
	<i>FMMUs</i> 11
	<i>Addressing Modes</i> 12
	<i>Watchdog Functionality</i> 13
	CANopen over EtherCAT Implementation Details 14
	<i>General Information</i> 14
	<i>Implemented Services</i> 14
	Data Exchange..... 15
	<i>Application Data (ADI)</i> 15
	<i>Process Data</i> 15
	Network Reset Handling..... 16
	<i>Reset Node</i> 16
	<i>Restore Manufacturer Parameters to Default</i> 16
	Configured Station Alias (Node Address) 16
	Device ID 16
	Modular Device Profile 17
Chapter 3	Object Dictionary (CANopen over EtherCAT)
	Standard Objects 18
	<i>General</i> 18
	<i>Object Entries</i> 19
	Manufacturer and Profile Specific Objects..... 22
	<i>General</i> 22
	<i>Network Data Format</i> 22
	<i>Error Codes</i> 23

<i>Object Entries</i>	24
<i>Modular Device Profile, Object Entries</i>	25
Chapter 4 Anybus Module Objects	
General Information	28
Anybus Object (01h).....	29
Diagnostic Object (02h)	30
Network Object (03h).....	32
Network Configuration Object (04h).....	34
Chapter 5 Host Application Objects	
General Information	36
Assembly Mapping Object (EBh).....	37
Sync Object (EEh)	38
EtherCAT Object (F5h).....	41
Appendix A Categorization of Functionality	
Basic.....	46
Extended.....	46
Appendix B Implementation Details	
SUP-Bit Definition.....	47
Anybus State Machine	47
Application Watchdog Timeout Handling	47
Appendix C Technical Specification	
Front View.....	48
Protective Earth (PE) Requirements	49
Power Supply	50
Environmental Specification	50
EMC Compliance.....	50
Appendix D Timing & Performance	
General Information	51
Internal Timing	51
<i>Startup Delay</i>	51
<i>NW_INIT Handling</i>	51
<i>Event Based WrMsg Busy Time</i>	52
<i>Event Based Process Data Delay</i>	52
Appendix E Copyright Notices	

P. About This Document

For more information, documentation etc., please visit the HMS website, 'www.anybus.com'.

P.1 Related Documents

Document	Author
Anybus CompactCom 40 Software Design Guide	HMS
Anybus CompactCom 40 Hardware Design Guide	HMS
Anybus CompactCom 40 Driver User Manual	HMS
IEC 61158-6	IEC
CiA Draft Standard 301 v4.02	CAN in Automation

P.2 Document History

Summary of Recent Changes (1.21... 1.30)

Change	Page(s)
Corrected lengths in EtherCAT Host Object (F5h), attributes #6, #7, and #8, added attribute #16	42, 43
Updated APPStatus table (ALStatusCode for Synchronization loss changed to 002Ch)	40
Updated information on creating and deleting diagnostic instances	31
Corrected data types for object attributes #3 and #4 in EtherCAT Host Object (F5h)	41
Extended and advanced joined to one category (extended)	

Revision List

Revision	Date	Author(s)	Chapter(s)	Description
1.00	2014-03-21	KeL	All	First official revision
1.01	2014-04-04	KeL		Added EtherCAT logo to trademark acknowledgements
1.10	2014-07-15	KeL	2, 3, 4, C, D, E	Updates and additions
1.20	2015-01-19	KeL	2, 3, 4, 5, B, C, D	Misc. Updates and corrections
1.21	2015-02-24	KaD	3, 4, 5	Minor updates
1.30	2015-10-23	KeL	4, 5	Minor updates and corrections

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.

1. About the Anybus CompactCom 40 EtherCAT

1.1 General

The Anybus CompactCom 40 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)
- Support for Modular Device Profile
- RJ45 connectors
- DS301 compliant
- Galvanically isolated bus electronics
- Network Identity customization
- EMCY support
- Up to 57343 ADIs can be accessed from the network as Manufacturer Specific Objects and Device Profile Specific Objects (generic mode).
- Up to 16383 ADIs can be accessed from the network as Manufacturer Specific Objects and Device Profile Specific Objects (modular device profile enabled)
- Up to 1486 bytes of fast cyclic I/O in each direction
- EtherCAT Slave Interface file provided by HMS
- Support for Sync0 functionality using distributed clocks
- File access over EtherCAT (FoE)
- Support for process data remap from the network
- Network cycle time down to 100 μ s
- Possible to implement DS402 drive profile, Semi device profiles, and other device profiles

2. Basic Operation

2.1 General Information

2.1.1 Software Requirements

No additional network support code needs to be written in order to support the Anybus CompactCom 40 EtherCAT, however due to the nature of the EtherCAT networking system certain restrictions must be taken into account:

- ADIs with instance numbers up to 57343 (DFFFh) can be accessed from the network. If the Modular Device Profile is implemented and running, instance numbers are limited to 16383 (3FFFh).
- When mapping ADIs to process data, there is a limit of 1486 elements or 1486 bytes, whichever comes first, that can be mapped in either direction.
- 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 40 software interface, consult the general Anybus CompactCom 40 Software Design Guide.

2.1.2 EtherCAT Slave Interface (ESI) File

Each device on EtherCAT is associated with a EtherCAT Slave Interface (ESI) file 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, it is strongly recommended to 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.
- The use of an own Vendor ID.
- Change of the product revision.
- The host application supports the Remap_ADI commands.
- Slow application response times. Explicit requests should be handled within 1 ms 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.

2.1.3 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 (HMS Industrial Networks Secondary Vendor ID, has to be replaced by Vendor ID of end product vendor.)
Product Code	0000 0036h (Anybus CompactCom 40 EtherCAT)
Device Name	'Anybus CompactCom 40 EtherCAT'
Serial Number	(Assigned during manufacturing)

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 conformance tests and be certified, a separate Vendor ID has to be requested from ETG.

See also...

- “EtherCAT Object (F5h)” on page 41

2.1.4 File Access over EtherCAT (FoE)

The module supports File Access over EtherCAT (FoE) for downloading a firmware file from a Client machine to the Server. All FoE requests not concerning files with the extension `hiff` (HMS firmware files) or the extension `.nfo`, will be forwarded to the Application File System Interface object, see the Anybus CompactCom 40 Software Design Guide for more information.

If a firmware file, downloaded through FoE, is pending for update, the file with the extension `hiff` will be possible to upload via FoE.

2.2 EtherCAT Implementation Details

2.2.1 General Information

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

- Application Layer: CANopen over EtherCAT
- FMMUs: 4
- Sync Managers: 4
- RAM Size: 16 kByte

See also...

- “CANopen over EtherCAT Implementation Details” on page 14

2.2.2 EtherCAT Synchronization

EtherCAT synchronization and jitter accuracy may depend on different things:

- How often the master sends out sync frames
- Temperature variations in the environment (large impact)
- The implementation of the EtherCAT slave device
- Which Ethernet physical layer is used in the slave devices (RJ45, E-Bus etc.)

The Anybus CompactCom 40 EtherCAT modules all demonstrate less than 1 μ s synchronization accuracy. For RJ45 products the accuracy may be around 50 ns under good conditions, and for E-Bus products around 30 ns.

2.2.3 Sync Managers

The module features four Sync Managers:

- **Sync Manager 0**
Used for mailbox write transfers (Master to Slave).
The module has a configurable write mailbox size with default size of 276 bytes, corresponding to 255 bytes plus relevant protocol headers and padding.
- **Sync Manager 1**
Used for mailbox read transfers (Slave to Master).
The module has a configurable read mailbox size with default size of 276 bytes, corresponding to 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).

2.2.4 FMMUs

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

2.2.5 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.

2.2.6 Watchdog Functionality

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

- **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 enabled by default in the ESI file, with a default time period of 100 ms.

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

See also...

- “SUP-Bit Definition” on page 47

2.3 CANopen over EtherCAT Implementation Details

2.3.1 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 15
- “Object Dictionary (CANopen over EtherCAT)” on page 18

2.3.2 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 subindex
Emergency	Reports unexpected conditions and diagnostic events.

2.4 Data Exchange

2.4.1 Application Data (ADI)

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

See also...

- “Manufacturer and Profile Specific Objects” on page 22

2.4.2 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 or how the application is implemented. It can be changed from the network during runtime, if the application has implemented the remap commands in the Application Data Object.

The module supports up to 6 TPDOs and up to 6 RPDOs, 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). The number of TPDOs and RPDOs can be extended if the Assembly Mapping Object is implemented.

To gain in configurability, the Assembly Mapping Object can be used to remap and replace the Process Data map specified at startup. Each PDO will be represented by an instance in the Assembly Mapping Object. The PDOs will then be remapped when the module enters the Safe-Operational state.

If the Modular Device Object is implemented, i.e. the Modular Device Profile is enabled, the Assembly Mapping Object will be ignored.

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 18
- “Manufacturer and Profile Specific Objects” on page 22
- “Assembly Mapping Object (EBh)” on page 37
- Application Data Object (see Anybus CompactCom 40 Software Design Guide)
- Modular Device Object (see Anybus CompactCom 40 Software Design Guide)

2.5 Network Reset Handling

2.5.1 Reset Node

If a valid firmware has been downloaded via FoE (File access over EtherCAT), the Anybus CompactCom 40 EtherCAT will send a reset type 00h ('Power-on reset') to the application at the transition from BOOT to INIT.

Prior to the reset command a Reset_Request command has to be sent to the host application to make sure that a reset can be performed.

2.5.2 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').

A factory default reset can only be performed in the EtherCAT state PREOPERATIONAL. Performing a reset in another state will generate SDO abort code 08000020h (invalid state).

See also...

- "Standard Objects" on page 18, entry 1011h ('Restore Parameters')

2.6 Configured Station Alias (Node Address)

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

The Anybus CompactCom 40 EtherCAT slave module does not support local configuration of the Configured Station Alias. For most applications it is recommended to leave the Configured Station Alias unchanged, but it is possible to assign each slave an address from the network.

2.7 Device ID

The Device ID is used by the master to explicitly identify a slave. This is e.g. useful when changing a faulty device during runtime¹. 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 1.²

See also...

- "Network Configuration Object (04h)" on page 34

1. A so called HotConnect application.
 2. Please note that in the Anybus CompactCom M30 EtherCAT, the Network Configuration Object, instance 3 was used for the Device ID.

2.8 Modular Device Profile

The Anybus CompactCom 40 EtherCAT supports the Modular Device Profile, that is enabled if the Modular Device Object is implemented in the application. Running this profile, the module supports a maximum of 63 slots, including the coupler in slot 0. The maximum number of ADIs, that can be accessed from the network, is 16383.

The value of the Device Type Object (1000h) is changed to 00005001h.

Enabling the Modular Device Profile will override the settings of the Assembly Mapping Object, if this object is implemented.

See also....

- Modular Device Object (Anybus CompactCom 40 Software Design Guide)
- “Modular Device Profile, Object Entries” on page 25

3. Object Dictionary (CANopen over EtherCAT)

3.1 Standard Objects

3.1.1 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).

3.1.2 Object Entries

Index	Object Name	Subindex	Description	Type	Access	Notes
1000h	Device Type	00h	Device Type	U32	RO	Default 0000 0000h (No profile). Can be managed through the EtherCAT Object, which can optionally be implemented in the host application. See "EtherCAT Object (F5h)" on page 41. Note: If the host application Modular Device Object is implemented, the default value is 0000 5001h.
1001h	Error register	00h	Error register	U8	RO	This information managed through the Diagnostic Object, see "Diagnostic Object (02h)" on page 30.
1003h	Pre-defined error field	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 41.
1009h	Manufacturer hardware version	00h	Manufacturer hardware version	Visible string	RO	
100Ah	Manufacturer software version	00h	Manufacturer Software 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 41.
		02h	Product Code	U32	RO	
		03h	Revision Number	U32	RO	
		04h	Serial Number	U32	RO	
10F1h	Error Settings object	00h	Number of entries	U8	RO	
		02h	Sync error counter limit	U32	RW	
1600h - 1xxh	Receive PDO mapping	00h	No. of mapped application objects in PDO	U8	RO/RW ^a	No. of mapped objects (0.. 254), see "Mapping ADIs on PDOs" on page 21 for more information.
		01h	Mapped object #1	U32	RO/RW ^a	-
		02h	Mapped object #2	U32	RO/RW ^a	-
		-
		NNh	Mapped object #NN	U32	RO/RW ^a	-
1A00h - 1xxh	Transmit PDO mapping	00h	No. of mapped application objects in PDO	U8	RO/RW ^a	No. of mapped objects (0.. 254), see "Mapping ADIs on PDOs" on page 21 for more information.
		01h	Mapped object #1	U32	RO/RW ^a	-
		02h	Mapped object #2	U32	RO/RW ^a	-
		-
		NNh	Mapped object #NN	U32	RO/RW ^a	-
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	Subindex	Description	Type	Access	Notes
1C12h	Sync Manager Rx PDO Assign	00h	No. of assigned PDOs	U8	RO/RW ^b	
		01h - NNh	Assigned PDO	U16	RO/RW ^b	
1C13h	Sync Manager Tx PDO Assign	00h	No. of assigned PDOs	U8	RO/RW ^c	
		01h - NNh	Assigned PDO	U16	RO/RW ^c	
1C32h	SM output parameter	00h	Max subindex supported	U8	RO	12 (0Bh)
		01h	Sync mode	U16	RO/RW	00h: Free Run 02h: DC Sync0 Also see "Sync Object (EEh)" on page 38.
		02h	Cycle time	U32	RW	Cycle time in nanoseconds
		03h	Shift time	U32	RW	Shift time in nanoseconds
		04h	Synchronization Types supported	U16	RO	Bit 0 set: FREE_RUN supported Bit 2 set: DC Sync0 supported. Bit 5 set: Output shift with local timer All other bits are set to 0 Also see "Sync Object (EEh)" on page 38.
		05h	Minimum cycle time	U32	RO	Minimum cycle time in nanoseconds.
		06h	Output Calc and Copy Time	U32	RO	Output Calc and Copy Time in nanoseconds.
		09h	Delay time	U32	RO	Delay time in nanoseconds. Always set to 0.
		0Ch	Cycle Time Too Small	U16	RO	Cycle time too small
1C33h	SM input parameter	00h	Max subindex supported	U8	RO	12 (0Bh)
		01h	Sync mode	U16	RO/RW	00h: Free Run 02h: DC Sync0 Also see "Sync Object (EEh)" on page 38.
		02h	Cycle time	U32	RW	Cycle time in nanoseconds, same value as 1C32h, subindex 2.
		03h	Shift time	U32	RW	Shift time in nanoseconds.
		04h	Synchronization Types supported	U16	RO	Bit 0 set: FREE_RUN supported Bit 2 set: DC Sync0 supported. Bit 5 set: Input shift with local timer All other bits are set to 0 Also see "Sync Object (EEh)" on page 38.
		05h	Minimum cycle time	U32	RO	Minimum cycle time in nanoseconds, same value as 1C32h, subindex 5.
		06h	Input Calc and Copy Time	U32	RO	Input Calc and Copy Time in nanoseconds.
		0Ch	Cycle Time Too Small	U16	RO	Cycle time too small, same value as 1C32h, subindex 12 (0Bh).

a. Writable when dynamic process data is supported by the application (remap commands).

b. When using static PDO mapping this subindex is read only. When using dynamic PDO mapping, it is writable.

c. If more than one sync mode is supported, this entry is writable.

Mapping ADIs on PDOs

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

Mode	Access	Number of objects (in each direction)	Number of sub indices per object
Generic, static mapping	RO	1 - 6 ^a	1 - 254 ^b
Generic, dynamic mapping	RW	1 - 6 ^a	254 (except the 6th object, that has 216 sub indices as the maximum number of entries is 1486)
Assembly Mapping Object implemented in host ^c	RO/RW ^d	Number of assembly mapping instances in that direction (max 63)	1486/(number of objects) (max 254)
Modular device, static mapping	RO	Same as the number of modules that have objects mappable in that direction (max 63)	Same as the number of ADIs mapped in that direction during setup
Modular device, dynamic mapping	RW	Same as the number of modules that have objects mappable in that direction (max 63)	1486/(number of objects) (max 254)

- a. Depends on how many ADI mapping items that are mapped by the application during setup. Each PDO can hold 254 ADI mapping items.
- b. Depends on how many ADI mapping items that are mapped by the application during setup. One PDO mapping object at the time will be filled with mapped items.
- c. See "Assembly Mapping Object (EBh)" on page 37 for more information.
- d. RO if the corresponding assembly instance is static, RW if it is dynamic.

Please note that in Generic mode and in Modular Device Profile mode, the ADI to PDO mapping is performed by the application at startup. Also note that if both the Assembly Mapping Object and the Modular Device Object are implemented in the host, the Modular Device Profile mode will be enabled, overriding the settings of the Assembly Mapping Object.

The PDO assignment objects (1C12h and 1C13h) are configured depending on how the host application is set up:

Mode	Access	Number of sub indices per object	Content
Generic, static mapping	RO	Same as the number of PDO mapping objects in that direction.	All PDO mapping objects in that direction.
Generic, dynamic mapping	RW	Same as the number of PDO mapping objects in that direction.	All PDO mapping objects in that direction.
Assembly Mapping Object implemented in host	RW	Same as the number of PDO mapping objects in that direction.	The first PDO in that direction
Modular device, static mapping	RO	Same as the number of PDO mapping objects in that direction.	All PDO mapping objects in that direction.
Modular device, dynamic mapping	RW	Same as the number of PDO mapping objects in that direction.	All PDO mapping objects in that direction.

3.2 Manufacturer and Profile Specific Objects

3.2.1 General

Each object entry in the manufacturer specific range (2001h...FFFFh) corresponds to an instance (a.k.a. ADI) within the Application Data Object (FEh), i.e. network accesses to these objects result 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.

3.2.2 Network Data Format

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

Anybus Data Type	Network Data Type
BOOL	UNSIGNED8
SINT8	INTEGER8
SINT16	INTEGER16
SINT32	INTEGER32
UINT8	UNSIGNED8
UINT16	UNSIGNED16
UINT32	UNSIGNED32
CHAR	VISIBLE_STRING
ENUM	UNSIGNED8 or ENUM
BITS8	BITARR8
BITS16	BITARR16
BITS32	BITARR32
OCTET	OCTET_STRING
SINT64	INTEGER64
UINT64	UNSIGNED64
FLOAT	REAL32
PAD0-16	NULL
BIT1 - BIT7	BIT1 - BIT7

Note 1: ADIs with multiple elements are represented either as arrays (all elements share the same data type) or as records (the elements may have different data types). Exceptions to this are ‘CHAR’ which will always be represented as VISIBLE_STRING, and ‘OCTET’ which will always be represented as OCTET_STRING.

Note 2: Single element ADIs are represented as a simple variable, with the exception of ‘CHAR’ which will always be represented as VISIBLE_STRING, and ‘OCTET’ which will always be represented as OCTET_STRING.

3.2.3 Error Codes

If an error occurs when an object in the application is requested from the module, the error code returned is translated to an CANopen abort code as follows:

Anybus CompactCom Error Code	CANopen Abort Code	Description (CANopen)
Reserved	N/A	-
Fragmentation error (serial mode)	N/A	-
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	Subindex 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 long.
Not enough data	0607 0013h	Data type does not match, length of service parameter too short.
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
Value too high	0609 0031h	Value of parameter higher than upper limit (only for write access).
Value too low	0609 0032h	Value of parameter lower than lower limit (only for write access).
Write access to a read process data mapped ADI	0601 0006h	Object mapped to RxPDO, SDO download blocked.
Object Specific Error	0800 0000h	General error

Note: If no corresponding error code can be defined on CANopen, the default error code will be General error (0800 0000h).

3.2.4 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 one element each, causing them to be represented as simple variables rather than arrays. The other ADIs have more than 1 element (of the same data type), causing them to be represented as arrays. If an ADI has more than 1 element, of different data types, it will be represented as a record.

Index	Object Name	Subindex	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	ADIs with multiple elements (i.e. arrays) are represented as multiple subindexes.			
		...				
		...				
		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	ADIs with multiple elements (i.e. arrays) are represented as multiple subindexes.			
		...				
		...				
		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	ADIs with multiple elements (i.e. arrays) are represented as multiple subindexes.			
		...				
		...				
		NNh				
...
5FFFh	ADI 3FFFh	00h	Number of entries (NNh)	U8	RO	-
		01h	ADI value(s) (Attribute #5)	-	-	
		02h	ADIs with multiple elements (i.e. arrays) are represented as multiple subindexes.			
		...				
		...				
		NNh				

3.2.5 Modular Device Profile, Object Entries

The objects listed in the table below, shall be implemented if the Modular Device Profile mode is enabled.

Index	Object Name	Subindex	Description	Type	Access	Notes
6000h-6FFFh	Input data	Any	ADIs for all modules, except the coupler, that are write process data mappable will be represented in this range.	Any	R, RW	For more information, see "ADI to SDO Translation" on page 26.
7000h-7FFFh	Output data	Any	ADIs for all modules, except the coupler, that are read process data mappable will be represented in this range.	Any	W, RW	For more information, see "ADI to SDO Translation" on page 26.
9nnnh	Information data	Any	Information objects, one for each module, occupying a slot, except the coupler.	Any	RW	For more information, see "Module Identification Objects" on page 27
F000h	Modular Device Profile	00h	Number of entries (NNh)	U8	R	Value: 5
		01h	Index distance	U16	R	This value decides how many objects are assigned to each slot. The value is the same for all modules, and thus gives the index distance between two slots. Value: "Number of ADIs per slot", attribute #12 in the Modular Device Object. See Anybus CompactCom 40 Software Design Guide for more information.
		02h	Maximum number of modules	U16	R	Value: "Number of slots", Attribute 11 in the Modular Device Object. See Anybus CompactCom 40 Software Design Guide for more information.
		04h	General Information	U32	R	Value: 0000 0700h (Subindices 9, 10, and 11 are supported in the 9nnnh module identification objects)
		05h	Module PDO group of the device	U16	R	Set to 0 to force the coupler process data to be positioned ahead of the process data. This allows for better integration towards the modular device host object.
F030h	Configured Module Ident List	00h	Number of Entries (Number of slots-1)	U8	R	The master writes the configured module list to these objects, so that the slave can compare the expected module configuration to the actual configuration. ^a
		01h	Module identity of the module configured on position 1 (slot 1).	U32	RW	
				
		0nh	Module identity of the module configured on position n (slot n).	U32	RW	

Index	Object Name	Subindex	Description	Type	Access	Notes
F050h	Detected Module Ident List	00h	Number of Entries (Number of slots-1)	U8	R	This object contains information about the modules, in the occupied slots, scanned from the application.
		01h	Module identity of the module configured on position 1 (slot 1).	U32	RW	
				
		0nh	Module identity of the module configured on position n (slot n).	U32	RW	
F600h-F6FFh	Input data area for the coupler	Any	ADIs for the coupler that are write process mappable will be represented in this range.	Any	R, RW	-
F700h-F7FFh	Output data area for the coupler	Any	ADIs for the coupler that are read process mappable will be represented in this range.	Any	W, RW	-

- a. If the Configured Module Ident List (F030h) does not match the Detected Module Ident List (F050h), the module will indicate a mismatch configuration by setting the ALStatusCode register to 0070h. The module will not enter SAFE-OPERATIONAL state.

ADI to SDO Translation

In the Modular Device Profile, all ADIs have to be mapped in numbering order. The number of ADIs mapped per slot is defined in the Modular Device Object, where the same number of objects is assigned to each slot. Depending on whether the ADIs are write or read mappable, they will be mapped to different object ranges. An ADI that is both read and write mappable will be mapped to both ranges. Please note that the SDOs are assigned in number order, but occupy different ranges, depending on type.

The ADIs, that are neither read nor write mappable, will not be mapped to an SDO, resulting in “empty SDOs” as shown in the table below.

Module	ADI	Type	SDO
0 (Coupler)	1	Write mappable	F600h
	2	Read mappable	F701h
	3	Write mappable	F602h
	4	Read mappable	F703h
	5	Not mappable	-
1	6	Read mappable	7000h
	7	Write mappable	6001h
	8	Writable	-
	9	Read only	-
	10	Read mappable	7004h
2	11	Writeable	-
	12	Read only	-
	-	-	-
	14	Write mappable	6008h
	15	Write and Read mappable	6009h and 7009h

Module Identification Objects

The first SDO in the 9nnnh range for each module, shall be predefined according to the table below:

Subindex	Type	Access	Name and Description
00h (0)	U8	R	Highest sub-index supported. Value: 11 (0Bh)
09h (9)	U16	R	Module PDO group. Value: 1. (The PDO group is set to 1 for all modules except the coupler to allow coupler data to be put before module data.)
0Ah (10)	U32	R	Module Identity (Module identity for the module according to the host application.)
0Bh (11)	U16	r	Slot (Module number)

PDO Mapping

The Receive PDO mapping objects and the Transmit PDO mapping objects are configured depending on how the host application is set up. One object in the 16xxh series is created for each module, that holds at least one read mappable ADI. The object numbers will be 1600h + slot number -1. One object in the 1Axxh series is created for each module, that holds at least one write mappable ADI. The object numbers will be 1A00h + slot number -1.

If the coupler holds any write or read mappable ADIs, objects will be created for these. Any objects for the coupler are created after all other mapping objects have been created.

For more information, see “Mapping ADIs on PDOs” on page 21.

4. Anybus Module Objects

4.1 General Information

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

Standard Objects:

- “Anybus Object (01h)” on page 29
- “Diagnostic Object (02h)” on page 30
- “Network Object (03h)” on page 32
- “Network Configuration Object (04h)” on page 34
- “File System Interface Object (0Ah), see Anybus CompactCom 40 Software Design Guide

Network Specific Objects:

(none)

4.2 Anybus Object (01h)

Category

Basic

Object Description

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

Supported Commands

Object: Get_Attribute
 Instance: Get_Attribute
 Set_Attribute
 Get_Enum_String

Object Attributes (Instance #0)

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

Instance Attributes (Instance #1)

Basic

#	Name	Access	Type	Value
1	Module type	Get	UINT16	0403h (Anybus CompactCom 40)
2... 11	-	-	-	Consult the general Anybus CompactCom 40 Software Design Guide for further information.
12	LED colors	Get	struct of: UINT8 (LED1A) UINT8 (LED1B) UINT8 (LED2A) UINT8 (LED2B)	<u>Value:Color:</u> 01h Green 02h Red 01h Green 02h Red
13... 16	-	-	-	Consult the general Anybus CompactCom 40 Software Design Guide for further information.
17	Virtual attributes	Get/Set		
18	Black list/White list	Get/Set		
19	Network time	Get	UINT64	64-bit value expressed in nanoseconds. Base: 12:00 AM, January 1, 2000. The Network time attribute contains the value of the DC system time register of the EtherCAT slave controller.

4.3 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 40 Software Design Guide.

An EMCY Object (Emergency Object) is sent on the network each time a diagnostic instance is created or deleted.

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 40 Software Design Guide for further information.
11	Max no. of instances	Get	UINT16	5 + 1 (one instance is reserved for a major unrecoverable event)
12	Supported functionality	Get	BITS32	Bit 0: 0 (The module does not support latching events) Bits 1 - 31: 0

Instance Attributes

Basic

#	Name	Access	Type	Value
1	Severity	Get	UINT8	See Anybus CompactCom 40 Software Design Guide
2	Event Code	Get	UINT8	
3	NW specific extension	Get	Array of UINT8	CANopen specific EMCY code (2 bytes)
4 -7	(not used)			

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

Bit no.	Description	Condition for setting bit
0	Generic error	Always set when another error bit in this object is set.
1	Current	Event code is 20h - 23h OR Event code is FFh AND the high byte in NW specific information is 20h - 23h.
2	Voltage	Event code is 30h - 33h OR Event code is FFh AND the high byte in NW specific information is 30h - 33h.
3	Temperature	Event code is 40h - 42h OR Event code is FFh AND the high byte in NW specific information is 40h - 42h.
4	Communication error	Event code is 80h - 82h OR Event code is FFh AND the high byte in NW specific information is 80h - 82h OR Anybus state equals ERROR.
5	Device profile specific	Always 0
6	Reserved	Always 0
7	Manufacturer specific	Event code is FFh AND the high byte in NW specific information is FFh.

3. If the diagnostic instance is created in the state WAIT_PROCESS or higher, an 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)				

No EMCY object is sent if the instance is created in either of the states SETUP or NW_INIT.

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.

An EMCY object with error code 0000h ("error reset") is sent when a diagnostic instance is deleted.

4.4 Network Object (03h)

Category

Basic

Object Description

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

Supported Commands

Object:	Get_Attribute
Instance:	Get_Attribute
	Set_Attribute
	Get_Enum_String
	Map_ADI_Write_Area
	Map_ADI_Read_Area
	Map_ADI_Write_Ext_Area
	Map_ADI_Read_Ext_Area

Object Attributes (Instance #0)

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

Instance Attributes (Instance #1)

Basic

#	Name	Access	Type	Value
1	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, Map_ADI_Write_Ext_Area and Remap_ADI_Write_Area. ^a
6	Read process data size	Get	UINT16	Current read process data size (in bytes). Updated on every successful Map_ADI_Read_Area, Map_ADI_Read_Ext_Area and Remap_ADI_Read_Area. ^a
7	Exception Information	Get	UINT8	Additional information may be provided here when the module has entered the EXCEPTION state, see "Exception information" on page 33.
8... 10	(reserved)	-	-	Consult the general Anybus CompactCom 40 Software Design Guide for further information.

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

Exception information

Value	Description
00h	No additional information available.
01h	(reserved)
02h	
03h	
04h	
05h	
06h	The implementation of the Assembly Mapping Host Object is incorrect, e.g. the attribute 11 or 12 is not supported.
07h	The application supports the Remap ADI commands, but returned an error response when requesting object attributes 11 or 12 of the Application Data Object ("No. of read process data mappable instances" or "No of write process data mappable instances") or when issuing the Get_Instance_Numbers command towards the Application Data Object.
08h	The implementation of the Modular Device Object in the host application is not correct, e.g. an error response is received on the Get_List command.

4.5 Network Configuration Object (04h)

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

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	0001h

Instance Attributes (Instance #1, 'Device ID')

See also "Device ID" on page 16.

Extended

Changes have immediate effect.

#	Name	Access	Type	Description
1	Name ^a	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	03h (read/write access)
5	Value	Get/Set	UINT16	1...65535:Valid network address 0:Device not configured (Default)
6	Configured Value	Get	UINT16	Configured value for Device ID. The value always equals the value of attribute #5.

a. Multilingual, see "Multilingual Strings" on page 35.

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
1	Device ID	Geräteadresse	ID Dispos.	ID Dispos.	ID appareil

Reset

When a factory default (reset) command is issued to this object, the configured Device ID will be set to 0 (default value).

5. Host Application Objects

5.1 General Information

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

Standard Objects:

- Application Object (see Anybus CompactCom 40 Software Design Guide)
- Application Data Object (see Anybus CompactCom 40 Software Design Guide)
- Application File System Interface Object (see Anybus CompactCom 40 Software Design Guide)
- “Assembly Mapping Object (EBh)” on page 37
- “Sync Object (EEh)” on page 38
- Modular Device Object (see Anybus CompactCom 40 Software Design Guide)

Network Specific Objects:

- “EtherCAT Object (F5h)” on page 41

5.2 Assembly Mapping Object (EBh)

Category

Extended

Object Description

If the application has implemented this object, the object will replace the PDO mapping created when the application is started. The original mapping will be replaced during the transition from PRE-OPERATIONAL state to SAFE-OPERATIONAL state. The application must support the Remap_ADI commands, if this object is to be implemented.

Each instance in the Assembly Mapping Object corresponds to one PDO. The first read assembly is mapped to object 1600h in the object dictionary, the second to 1601h and so on. Similarly, the first write assembly is mapped to object 1A00h, the second to 1A01h and so on. Up to 64 each of read and write assembly instances are supported.

The table below illustrates an example on how PDO mapping object numbers are assigned for different assembly mapping object instances.

Instance no.	Direction	PDO mapping object number
1	Write	1A00h
2	Read	1600h
3	Write	1A01h
4	Read	1601h
5	Read	1602h

Each assembly mapping instances supports up to 254 ADI elements, corresponding to one full PDO on EtherCAT.

Note: If the Modular Device Object is implemented in the host application, i.e. modular device profile is enabled, the settings of this objects will be ignored.

See also ..

- Anybus CompactCom 40 Software Design Guide, “Assembly Mapping Object”
- “Standard Objects” on page 18 for assembly to PDO mapping.

5.3 Sync Object (EEh)

Category

Extended

Object Description

This object implements the host application SYNC settings.

The implementation of this object is optional; if it is not implemented, the module only supports the EtherCAT Free Run mode.

If there is any problem with the configuration of the sync functionality as a whole, the application must indicate this in the application status register. The module will then change EtherCAT states to SafeOp and indicate the problem in the `ALStatusCode` register, see “Application Status Register” on page 40.

See also...

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

Supported Commands

Object:	Get_Attribute
Instance:	Get_Attribute Set_Attribute

Object Attributes (Instance #0)

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

Instance Attributes (Instance #1)

Extended

The attributes are represented on EtherCAT as follows:

#	Name	Access	Type	Default Value	Comment
1	Cycle time	Get/Set	UINT32		Application cycle time in nanoseconds. Replaces the setting in object entry 1C32h, subindex 2. (SM Output Parameter, Cycle time)
2	Output valid	Get/Set	UINT32	0	Output valid point relative to SYNC events, in nanoseconds. Replaces the setting in object entry 1C32h, subindex 3. (SM Output Parameter, Shift time)
3	Input capture	Get/Set	UINT32	0	Input capture point relative to SYNC events, in nanoseconds. Replaces the setting in object entry 1C33, subindex 3. (SM Input Parameter, Shift time)
4	Output processing ^a	Get	UINT32		Minimum required time, in nanoseconds, between RDPDI interrupt and valid output. Specifies the value of object entry 1C32h, subindex 6. (SM Output Parameter, Output Calc and Copy Time)
5	Input processing ^a	Get	UINT32		Maximum required time, in nanoseconds, from "Input capture" until write process data has been completely written to the Anybus CompactCom module. Specifies the value of object entry 1C33h, subindex 6. (SM Input Parameter, Input Calc and Copy Time)
6	Min cycle time	Get	UINT32		Minimum cycle time supported by the application. Specifies the values of object entries 1C32h and 1C33h, subindex 5. (SM Output and SM Input Parameters, Minimum cycle time)
7	Sync mode	Get/Set	UINT16		Selection of synchronization mode. The attribute enumerates the bits in attribute 8. 0: Free Run (no sync) 1: Synced with DC Specifies the values of object entries 1C32h and 1C33h, subindex 1. (SM Output and SM Input Parameters, Synchronization type).
8	Supported sync modes	Get	UINT16		A list of the synchronization modes the application supports. Each bit corresponds to a mode in attribute 7. Bit 0: 1 = Free run supported Bit 1: 1 = DC supported Specifies the values of object entries 1C32h and 1C33h, subindex 4. (SM Output and Input Parameters, Synchronization types supported)

a. The Anybus latency is added to this value before it is presented on EtherCAT.

Application Status Register

If the application sets an error status to the application status register (H_APPSTATUS), the module sets the EtherCAT state to SafeOp. The H_APPSTATUS is translated to the ALStatusCode register as shown in the table below.

H_APPSTATUS	Error	ALStatusCode: ALSTATUSCODE_XXX (#)	Comment
0000h	No error	-	Application can operate in state PROCESS_ACTIVE
0001h	Not yet synchronized	NOSYNCERROR (002Dh)	Application is not synchronized to the SYNC signal and not ready to go to PROCESS_ACTIVE.
0002h	Sync config error	INVALIDSYNCCFG (0030h)	A problem with the configuration of the Sync host object prevents the application from going to PROCESS_ACTIVE.
0003h	Read process data configuration error	INVALIDOUTPUTMAPPING (0025h)	A problem with the current read process data mapping is prevents the application from going to PROCESS_ACTIVE.
0004h	Write process data configuration error	INVALIDINPUTMAPPING (0024h)	A problem with the current write process data mapping is prevents the application from going to PROCESS_ACTIVE.
0005h	Synchronization loss	FATALSYNCERROR (002Ch)	Application has lost the lock to the synchronization. If the module is in state PROCESS_ACTIVE, it will go to ERROR.
0006h	Excessive data loss	NOVALIDINPUTSANDOUTPUTS (002Bh)	The application has detected a significant loss of process data frames from the network. If the module is in state PROCESS_ACTIVE, it will go to ERROR.
0007h	Output error	DCSYNCIOERROR (0033h)	A problem in the application prevents it from acting on outputs. If the module is in state PROCESS_ACTIVE, it will go to ERROR.

5.4 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 1: Support for this object is optional. If implemented, it is highly recommended to support all attributes in the range 1... 6.

Note 2: To pass conformance tests, the end product has to have a Vendor ID valid for the end product vendor.

See also...

- Anybus CompactCom 40 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	01h
4	Highest instance no.	Get	UINT16	01h

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, assigned at production	
6	Manufacturer Device Name	Get	Array of CHAR (Max. 64 bytes)	Product specific	Replaces object entry 1008h (Manufacturer Device Name)
7	Manufacturer Hardware Version	Get	Array of CHAR (Max. 64 bytes)	X.YY (major version.minor version)	Specifies the value of object entry 1009h (Manufacturer Hardware Version)
8	Manufacturer Software Version	Get	Array of CHAR (Max. 64 bytes)	X.YY.ZZ (major version.minor version. build)	Specifies the value of object entry 100Ah (Manufacturer Software 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 this attribute is implemented, 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.
10	Device Type	Get	UINT32	Product specific	If implemented, this value replaces the default value for object entry 1000h (Device type).
11	Write PD assembly instance translation	Get	Array of UINT16	Empty	This attribute can be used by the application to change the default TxPDO mapping object of Write PD instances in the Assembly Mapping Object. It corresponds to attribute 11 in the Assembly Mapping Object, "Write PD Instance List". Each index in the array contains the TxPDO mapping object number that is used for the instance on the same index in the "Write PD Instance List" attribute. Valid values: 1A00h - 1BFFh.
12	Read PD assembly instance translation	Get	Array of UINT16	Empty	This attribute can be used by the application to change the default RxPDO mapping object of Read PD instances in the Assembly Mapping Object. It corresponds to attribute 12 in the Assembly Mapping Object, "Read PD Instance List". Each index in the array contains the RxPDO mapping object number that is used for the instance on the same index in the "Read PD Instance List" attribute. Valid values: 1600h - 17FFh.

#	Name	Access	Type	Default Value	Comment
13	ADI translation	Get	Array of Struct { UINT16 UINT16 }	Empty	This attribute can be used by the application to implement objects in the communication profile specific CoE object area (1000h - 1FFFh). Objects already implemented in the module cannot be replaced by ADIs. The attribute is implemented as an array of packed structs of two UINT16. The first UINT16 contains the ADI instance number, the second contains the object index that the ADI shall correspond to. See example: "ADI Translation, Example"
14	(Reserved)	-	-	-	(Reserved for future use)
15	Object subindex translation	Get	Array of Struct { UINT16 UINT16 UINT8 }	Empty	This attribute can be used by the application to implement subindices of objects in the profile specific CoE object area (0x1000-0x1FFF). Subindices already implemented in the module cannot be replaced by ADIs and this attribute can only be used to add subindices to objects explicitly defined in the module to be extendable. This attribute is currently only supported to add subindices to objects 0x1C32 ("Output Sync Manager Parameter") and 0x1C33 ("Input Sync Manager Parameter"). The attribute is implemented as an array of packed Structs of two UINT16 and one UINT8. The first UINT16 contains the ADI instance number, the second contains the object index that the ADI shall correspond to. The UINT8 contains the subindex of the latter object that the ADI shall correspond to. An object dictionary index/subindex entry may only be translated to an ADI of type VAR. Translating the entry to an ADI of type ARRAY or RECORD is not supported. See: "Object Subindex Translation, Example"
16	Enable FoE	Get	BOOL	TRUE (=1)	This attribute enables/disables functionality related to File access over EtherCAT. If FoE is disabled it is not possible to upgrade firmware via EtherCAT or access the Application File System Interface Object (EAh) via EtherCAT.

ADI Translation, Example

The host application wants to implement the diagnostic object (10F3h) and the timestamp object (10F8h). To do this it needs to create two ADIs that match the CoE implementation of these objects, e.g. ADI F0F3h for the diagnostic object and F0F8 for the timestamp object. It then needs to implement the following data for the ADI translation attribute:

```
[
  {
    F0F3h
    10F3h
  }
  {
    F0F8h
```

```
        10F8h  
    }  
]
```

SDO requests towards these CoE objects will then be forwarded to the corresponding ADI. If a CoE object present in this attribute is implemented by the module, the module will handle all requests to that object by itself, and nothing is forwarded to the host application.

Object Subindex Translation, Example

The host application wants to implement the Sync Error subindex (subindex 32) of the 0x1C32 and 0x1C33 objects. To do this it needs to create two ADIs that match the CoE implementation of these entries. Let's say it creates ADI 0xF0FD for entry 0x1C32:32 and ADI 0xF0FE for entry 0x1C33:32. It then needs to implement the following data for the "Object subindex translation" attribute:

```
[
  {
    0xF0FD
    0x1C32
    32
  }
  {
    0xF0FE
    0x1C33
    32
  }
]
```

SDO requests towards these CoE object/subindex entries will then be forwarded to the corresponding ADI.

If a CoE entry present in this attribute is implemented by the module, the module will handle all requests to that entry by itself, as it will if the object does not support being extended with more subindices, and nothing is forwarded to the host application.

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. 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.

B.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, BOOTSTRAP 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)

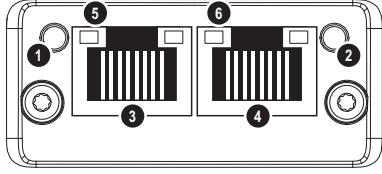
B.3 Application Watchdog Timeout Handling

The Anybus CompactCom 40 EtherCAT module will enter the EXCEPTION state if the application watchdog times out.

C. Technical Specification

C.1 Front View

Ethernet Connector

#	Item	
1	RUN LED ^a	
2	ERROR LED ^a	
3	EtherCAT (IN port)	
4	EtherCAT (OUT port)	
5	Link/Activity (IN port)	
6	Link/Activity (OUT port)	

a. The flash sequences for these LEDs are defined in ETG1300_S_R_V1i1i0_IndicatorLabelingSpecification.pdf (ETG).

RUN LED

This LED reflects the status of the EtherCAT communication.

LED State	Indication	Description
Off	INIT	EtherCAT device in 'INIT'-state (or no power)
Green	OPERATIONAL	EtherCAT device in 'OPERATIONAL'-state
Green, blinking	PRE-OPERATIONAL	EtherCAT device in 'PRE-OPERATIONAL'-state
Green, single flash	SAFE-OPERATIONAL	EtherCAT device in 'SAFE-OPERATIONAL'-state
Flickering	BOOT	The EtherCAT device is in 'BOOT' state
Red ^a	(Fatal Event)	-

a. If RUN and ERR turn 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.
Red, single flash	Unsolicited state change	Slave device application has changed the EtherCAT state autonomously.
Red, double flash	Application watchdog timeout	Sync manager watchdog timeout.
Red ^a	Application controller failure	Anybus module in EXCEPTION.
Flickering	Booting error detected	E.g. due to firmware download failure.

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

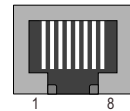
Link/Activity

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	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	-	



C.2 Protective Earth (PE) Requirements

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

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

C.3 Power Supply

Supply Voltage

The module requires a regulated 3.3V power source as specified in the general Anybus CompactCom 40 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 430 mA¹.

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.

C.4 Environmental Specification

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

C.5 EMC Compliance

Consult the Anybus CompactCom Hardware 40 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 40 EtherCAT will remain as a Class B module.

D. Timing & Performance

D.1 General Information

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

The following timing aspects are measured:

Category	Parameters	Page
Startup Delay	T1, T2	51
NW_INIT Handling	T100	51
Event Based WrMsg Busy Time	T103	52
Event Based Process Data Delay	T101, T102	52

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

D.2 Internal Timing

D.2.1 Startup Delay

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

Parameter	Description	Max.	Unit.
T1	The Anybus CompactCom 40 EtherCAT module generates the first application interrupt (parallel mode)	11	ms
T2	The Anybus CompactCom 40 EtherCAT module is able to receive and handle the first application telegram (serial mode)	11	ms

D.2.2 NW_INIT Handling

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

Parameter	Conditions
No. of network specific commands	Max.
No. of ADIs (single UINT8) mapped to Process Data in each direction	32 ^a
Event based application message response time	> 1 ms
Ping-pong application response time	> 10 ms
No. of simultaneously outstanding Anybus commands that the application can handle	1

a. Or maximum amount in case the network specific maximum is less.

Parameter	Description	Communication	Max.	Unit.
T100	NW_INIT handling	Event based modes	3.6	ms

D.2.3 Event Based WrMsg Busy Time

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

Parameter	Description	Min.	Max.	Unit.
T103	H_WRMSG area busy time	2.8	7.2	µs

D.2.4 Event Based Process Data Delay

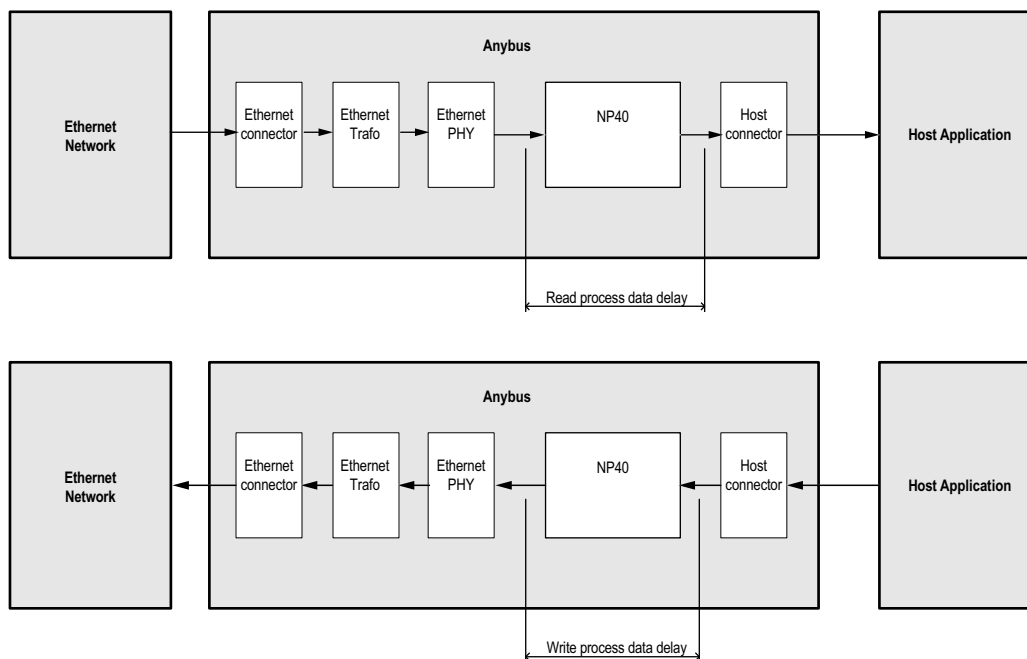
“Read process data delay” is defined as the time from when the last bit of the network frame has been received by the network interface, to when the RDPDI interrupt is asserted to the application.

“Write process data delay” is defined as the time from when the application exchanges write process data buffers, to when the first bit of the new process data frame is sent out on the network.

The tests were run in 16-bit parallel event mode, with interrupts triggered only for new process data events. Eight different IO sizes (2, 16, 32, 64, 128, 256, 512 and 1024 bytes) were used in the tests, all giving the same test results.

The delay added by the PHY circuit has not been included, as this delay is insignificant compared to the total process data delay.

Parameter	Description	Delay (min.)	Delay (typ.)	Delay (max.)	Unit
T101	Read process data delay	-	-	228	ns
T102	Write process data delay	-	-	170	ns



E. Copyright Notices

Format - lightweight string formatting library.

Copyright (C) 2010-2013, Neil Johnson

All rights reserved.

Redistribution and use in source and binary forms,
with or without modification,
are permitted provided that the following conditions are met:

- * Redistributions of source code must retain the above copyright notice,
this list of conditions and the following disclaimer.
- * Redistributions in binary form must reproduce the above copyright notice,
this list of conditions and the following disclaimer in the
documentation and/or other materials provided with the distribution.
- * Neither the name of nor the names of its contributors
may be used to endorse or promote products derived from this software
without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.