

# Anybus<sup>®</sup> CompactCom<sup>™</sup> 40

EtherCAT<sup>®</sup> Transparent Ethernet

## NETWORK GUIDE

SCM-1202-020 2.5 en-US ENGLISH

---

# Important User Information

## Disclaimer

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

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

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

---

# Table of Contents

Page

<b>1</b>	<b>Preface .....</b>	<b>5</b>
1.1	About this document .....	5
1.2	Related Documents .....	5
1.3	Document History .....	5
1.4	Document Conventions .....	6
1.5	Document Specific Conventions .....	7
1.6	Trademark Information .....	7
<b>2</b>	<b>About the Anybus CompactCom 40 EtherCAT .....</b>	<b>8</b>
2.1	General .....	8
2.2	Features .....	9
2.3	Transparent Ethernet .....	10
<b>3</b>	<b>Basic Operation .....</b>	<b>11</b>
3.1	General Information .....	11
3.2	EtherCAT Implementation Details .....	15
3.3	CANopen over EtherCAT Implementation Details .....	17
3.4	Data exchange .....	18
3.5	Network Reset Handling .....	19
3.6	Configured Station Alias (Node Address) .....	20
3.7	Device ID .....	20
3.8	Modular Device Profile .....	21
<b>4</b>	<b>Object Dictionary (CANopen over EtherCAT) .....</b>	<b>22</b>
4.1	Standard Objects .....	22
4.2	Manufacturer and Profile Specific Objects .....	27
<b>5</b>	<b>Anybus Module Objects .....</b>	<b>33</b>
5.1	General Information .....	33
5.2	Anybus Object (01h) .....	34
5.3	Diagnostic Object (02h) .....	35
5.4	Network Object (03h) .....	37
5.5	Network Configuration Object (04h) .....	39
5.6	Network Ethernet Object (0Ch) .....	46
5.7	Functional Safety Module Object (11h) .....	48

<b>6</b>	<b>Host Application Objects .....</b>	<b>55</b>
6.1	General Information .....	55
6.2	Functional Safety Object (E8h).....	56
6.3	Assembly Mapping Object (EBh) .....	58
6.4	Sync Object (EEh) .....	59
6.5	EtherCAT Object (F5h) .....	61
6.6	Ethernet Host Object (F9h) .....	74
<b>7</b>	<b>Transparent Ethernet.....</b>	<b>79</b>
7.1	General Information .....	79
7.2	Enabling Transparent Ethernet .....	79
7.3	MAC Address Synchronization .....	80
7.4	IP Configuration Synchronization .....	80
7.5	Routing Restrictions .....	80
<b>8</b>	<b>Firmware Upgrade .....</b>	<b>82</b>
<b>A</b>	<b>Categorization of Functionality .....</b>	<b>83</b>
A.1	Basic.....	83
A.2	Extended .....	83
<b>B</b>	<b>Implementation Details .....</b>	<b>84</b>
B.1	SUP-bit Definition .....	84
B.2	Anybus State Machine .....	84
B.3	Application Status Register .....	85
B.4	Application Watchdog Timeout Handling.....	85
<b>C</b>	<b>Technical Specification.....</b>	<b>86</b>
C.1	Front View .....	86
C.2	Functional Earth (FE) Requirements.....	87
C.3	Power Supply .....	87
C.4	Environmental Specification.....	88
C.5	EMC Compliance .....	88
<b>D</b>	<b>Timing &amp; Performance .....</b>	<b>89</b>
D.1	General Information .....	89
D.2	Internal Timing .....	89
<b>E</b>	<b>Secure HICP (Secure Host IP Configuration Protocol) .....</b>	<b>91</b>
E.1	General .....	91
E.2	Operation .....	91

---

<b>F</b>	<b>Backward Compatibility .....</b>	<b>92</b>
F.1	Initial Considerations .....	92
F.2	Hardware Compatibility .....	92
F.3	General Software .....	98
F.4	Network Specific — EtherCAT.....	100
<b>G</b>	<b>Copyright Notices .....</b>	<b>102</b>

**This page intentionally left blank**

# 1 Preface

## 1.1 About this document

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

The reader of this document is expected to be familiar with high level software design and communication systems in general. The information in this network guide should normally be sufficient to implement a design. However if advanced EtherCAT specific functionality is to be used, in-depth knowledge of EtherCAT networking internals and/or information from the official EtherCAT specifications may be required. In such cases, the persons 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.

For additional related documentation and file downloads, please visit the support website at [www.anybus.com/support](http://www.anybus.com/support).

## 1.2 Related Documents

Document	Author	Document ID
Anybus CompactCom 40 Software Design Guide	HMS	HMSI-216-125
Anybus CompactCom M40 Hardware Design Guide	HMS	HMSI-216-126
Anybus CompactCom B40 Design Guide	HMS	HMSI-27-230
Anybus CompactCom Host Application Implementation Guide	HMS	HMSI-27-334
Anybus CompactCom 40 EtherCAT Network Guide	HMS	SCM-1202-034
IEC 61158-6	IEC	
SEMI Device FW Upgrade, V1.0.0	EtherCAT Technology Group	ETG.5003.2 S (R) V1.0.0
EtherCAT Slave Information Specification, V1.0.10	EtherCAT Technology Group	ETG.2000 S (R) V1.0.10
CiA Draft Standard 301 v4.02	CAN in Automation	

## 1.3 Document History

Version	Date	Description
1.0	2016-10-28	First release
1.1	2017-01-19	Minor corrections
1.2	2017-06-15	Added the Get_Object_Description service and attributes 20, 21, 22 to the EtherCAT object Added note to Modular Device Profile, Object Entries section In Watchdog Functionality, PDI watchdog is now supported Added information in the Object Dictionary section Added information about the ESI generator in the ESI section Added a reference to the Watchdog Functionality section in the ERR LED description Minor corrections
1.3	2017-07-11	Added appendix on backward compatibility
1.4	2017-09-29	Added safety objects (11h and E8h) and information about FSoE (Fail Safe over EtherCAT)
1.5	2017-12-04	Updated Network Ethernet Object (0Ch)
1.6	2018-01-11	Updated trademark information
1.7	2018-01-17	Updated trademark information
1.8	2018-05-25	Updated trademark information

Version	Date	Description
		Updated network data types Updated EtherCAT Host Object Minor corrections
1.9	2018-11-01	Added instance attribute #27 to EtherCAT Object (F5h) Updated description of network reset
2.3	2019-05-27	Rebranding Minor updates
2.4	2020-02-27	Added instance attributes #28 - #30 to EtherCAT Object (F5h). Updated Restore Manufacturer Parameters to Default section Updated Device ID section Added note to Mapping ADIs on PDOs section
2.5	2021-02-02	Minor updates Added ESI entries for FoE in the File Access over EtherCAT section Added new commands in the EtherCAT Object: Get_Object_Access, Get_Data_Type and Get_Enum_Data Added General Synchronization Error to Application Status Register Redesigned Reset Node section Corrected the Instance Attribute examples in the Diagnostic Object

## 1.4 Document Conventions

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

1. First do this
2. Then do this

Bulleted lists are used for:

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

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

Program code and script examples

Cross-reference within this document: [Document Conventions, p. 6](#)

External link (URL): [www.hms-networks.com](http://www.hms-networks.com)



### **WARNING**

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



### **Caution**

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



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



*Additional information which may facilitate installation and/or operation.*



## 1.5 Document Specific Conventions

- The terms “Anybus” or “module” refers to the Anybus CompactCom module.
- The terms “host” or “host application” refer to the device that hosts the Anybus.
- Hexadecimal values are written in the format NNNNh or 0xNNNN, where NNNN is the hexadecimal value.
- A byte always consists of 8 bits.
- The terms “basic” and “extended” are used to classify objects, instances and attributes.

## 1.6 Trademark Information

Anybus® is a registered trademark of HMS Industrial Networks.



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



Safety over EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.

All other trademarks are the property of their respective holders.

## 2 About the Anybus CompactCom 40 EtherCAT

### 2.1 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 network guide covers the Transparent Ethernet version of the product. Transparent Ethernet has to be enabled during setup, or the device will appear as an Anybus CompactCom 40 with full IT functionality. The IT functionality is described in the network guide for the standard Anybus CompactCom 40.

IT functionality is not available when Transparent Ethernet is enabled.

This product conforms to all aspects of the host interface for Anybus CompactCom 40 modules defined in the Anybus CompactCom 40 Hardware and Software Design Guides, making it fully interchangeable with any other device following that specification. Generally, no additional network related software support is needed, however in order to be able to take full advantage of advanced network specific functionality, a certain degree of dedicated software support may be necessary.

## 2.2 Features

- CANopen over EtherCAT (CoE)
  - Complete Access support
- Support for Modular Device Profile
- Ethernet 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
- Ethernet over EtherCAT (EoE)
- Transparent Ethernet
- Black channel interface, offering a transparent channel supporting Fail Safe over EtherCAT (FSoE).
- File access over EtherCAT (FoE)
- Support for process data remap from the network
- Network cycle time down to 100  $\mu$ s
- Possible to implement DS402 drive profile, Semi device profiles, and other device profiles



*If the TwinCAT 3, or a later version of 2.11, tool is used, the max amount of process data will be 1473 bytes, due to limitations in the tool.*

---

## 2.3 Transparent Ethernet

Transparent Ethernet offers the possibility for a host application, that includes an IT implementation (web pages, file system, a proprietary protocol etc.), to let the Anybus CompactCom handle an industrial Ethernet protocol (in this case EtherCAT), without the need for extra Ethernet ports.

Ethernet communication is routed straight to the host application system using an RMII interface. The host application must include an Ethernet controller and a TCP/IP stack. EtherCAT protocol messages will be routed to the Anybus CompactCom internal software. Please note that the Transparent Ethernet functionality has to be enabled during startup by setting attribute #16 (instance #1) in the Anybus Object.

16-bit parallel mode can not be used, as specific host application connector pins are reserved for transparent Ethernet. Also TCP/UDP ports may be reserved, and can, in that case, not be used for the transparent Ethernet communication.

See also ...

- [Transparent Ethernet, p. 79](#)
- Anybus CompactCom 40 Hardware Design Guide
- [Anybus Object \(01h\), p. 34](#)

## 3 Basic Operation

### 3.1 General Information

#### 3.1.1 Software Requirements

No additional network support code needs to be written in order to support the Anybus CompactCom 40 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.



*If the TwinCAT 3, or a later version of 2.11, tool is used, the max amount of process data will be 1473 bytes, due to limitations in the tool.*

---

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

### 3.1.2 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.

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. To aid with the ESI file creation, HMS provides a tool called HMS EtherCAT ESI Generator, which is freely downloadable from the Anybus CompactCom 40 EtherCAT product page on [www.anybus.com](http://www.anybus.com).

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.
- The use of Ethernet over EtherCAT (EoE).
- 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.*

### 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 (HMS Industrial Networks Secondary Vendor ID, has to be replaced by the Vendor ID of the 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 \(F5h\), p. 61](#)

### 3.1.4 File Access over EtherCAT (FoE)

The module supports File Access over EtherCAT (FoE) for downloading firmware files from a Client machine to the Server. All FoE requests not concerning files with the extension `.hiff` (HMS firmware files) or the extension `.info`, will be forwarded to the Application File System Interface object. Since FoE offers only very basic FTP functionality, saved files (other than `.hiff` files) will end up in the root folder of the Application File System Interface object.

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

FoE is not supported by Anybus IP.



*.hiff-files will be renamed to `firmware.hiff` when downloaded.*

To indicate or remove support for File Access over EtherCAT (FoE) in the ESI file, see the following:

To support File Access over EtherCAT (FoE), the `<Mailbox>` element should look like this:

```
<Mailbox DataLinkLayer="1">
  <CoE SdoInfo="1" CompleteAccess="1" PdoAssign="0" PdoConfig="0"
    PdoUpload="1"/>
  <FoE/>
</Mailbox>
```

To remove support for File Access over EtherCAT (FoE), the `<Mailbox>` element should look like this:

```
<Mailbox DataLinkLayer="1">
  <CoE SdoInfo="1" CompleteAccess="1" PdoAssign="0" PdoConfig="0"
    PdoUpload="1"/>
</Mailbox>
```

### 3.1.5 Fail Safe over EtherCAT (FSoE)

The Anybus CompactCom 40 EtherCAT supports FSoE. This profile makes it possible for a user to send data on a black channel interface, i.e. a safe channel over EtherCAT, using an add-on safety module. For an application to support FSoE, the Functional Safety Object (E8h, host application object) has to be implemented. The Anybus CompactCom serial channel is used for the functional safety communication. When this channel is used for the host application, a second separate serial channel is implemented for the functional safety communication. See the Anybus CompactCom Hardware Design Guide for more information.

See...

- [Functional Safety Module Object \(11h\), p. 48](#)
- [Functional Safety Object \(E8h\), p. 56](#)

### 3.1.6 Ethernet over EtherCAT (EoE)

The module supports transparent tunneling of non-EtherCAT Ethernet frames to and from an EtherCAT slave, using Ethernet over EtherCAT (EoE).

EoE is not supported for Anybus IP in limited mode.

When Transparent Ethernet is enabled all Ethernet frames received over EoE (except those with reserved port numbers) will be routed to the host application.

Since the Ethernet frames are embedded in mailbox communication, the performance will be reduced compared to normal Ethernet communication. The data throughput will depend on...

- The EtherCAT process data cycle time
- The mailbox size (in bytes)



To be able to use Ethernet over EtherCAT (EoE), the Anybus CompactCom 40 device needs to be assigned a MAC address. Users with devices containing older software (prior to software version 2.00) will need to set the MAC address manually, in the Ethernet Host Object, to be able to use Ethernet over EtherCAT (EoE).

To indicate or remove support for Ethernet over EtherCAT (EoE) in the ESI file, see the following:

To support Ethernet over EtherCAT (EoE), the <Mailbox> element should look like this:

```
<Mailbox DataLinkLayer="1">
  <EoE IP="0" MAC="0" TimeStamp="0" />
  <CoE SdoInfo="1" CompleteAccess="1" PdoAssign="0" PdoConfig="0"
    PdoUpload="1"/>
  <FoE/>
</Mailbox>
```

To remove support for Ethernet over EtherCAT (EoE), the <Mailbox> element should look like this:

```
<Mailbox DataLinkLayer="1">
  <CoE SdoInfo="1" CompleteAccess="1" PdoAssign="0" PdoConfig="0"
    PdoUpload="1"/>
  <FoE/>
</Mailbox>
```



## 3.2 EtherCAT Implementation Details

### 3.2.1 General Information

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

<b>Application Layer:</b>	CANopen over EtherCAT
<b>FMMUs:</b>	4
<b>Sync Managers:</b>	4
<b>RAM Size:</b>	16 kByte

See also...

- [CANopen over EtherCAT Implementation Details, p. 17](#)

### 3.2.2 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  $\mu$ s synchronization accuracy. For RJ45 products the accuracy may be around 50 ns under good conditions, and for E-Bus products around 30 ns.

### 3.2.3 Sync Managers

The module features four Sync Managers:

<b>Sync Manager 0</b>	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.
<b>Sync Manager 1</b>	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.
<b>Sync Manager 2</b>	Contains the RxPDOs (in practice, Sync Manager 2 holds the Read Process Data).
<b>Sync Manager 3</b>	Contains the TxPDOs (in practice, Sync Manager 3 holds the Write Process Data).

### 3.2.4 FMMUs

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

### 3.2.5 Addressing Modes

As a full EtherCAT, the module supports the following addressing modes:

- position addressing
- node addressing
- logical addressing

### 3.2.6 Watchdog Functionality

#### 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, p. 84](#)

#### PDI Watchdog

PDI watchdog functionality is supported.

### 3.3 CANopen over EtherCAT 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, p. 18](#)
- [Object Dictionary \(CANopen over EtherCAT\), p. 22](#)

#### 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 indexes
Get Object Description	Reads details of an index
Get Entry Description	Reads details of a subindex
Emergency	Reports unexpected conditions and diagnostic events.

## 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 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, p. 27](#)

### 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 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.



*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, p. 22](#)
- [Manufacturer and Profile Specific Objects, p. 27](#)
- [Assembly Mapping Object \(EBh\), p. 58](#)
- Application Data Object (see Anybus CompactCom 40 Software Design Guide)
- Modular Device Object (see Anybus CompactCom 40 Software Design Guide)

## 3.5 Network Reset Handling

### 3.5.1 Reset Node

If the host application needs to reset the Anybus CompactCom 40 EtherCAT, the reset process must be implemented in the host application as shown in the flowchart below.

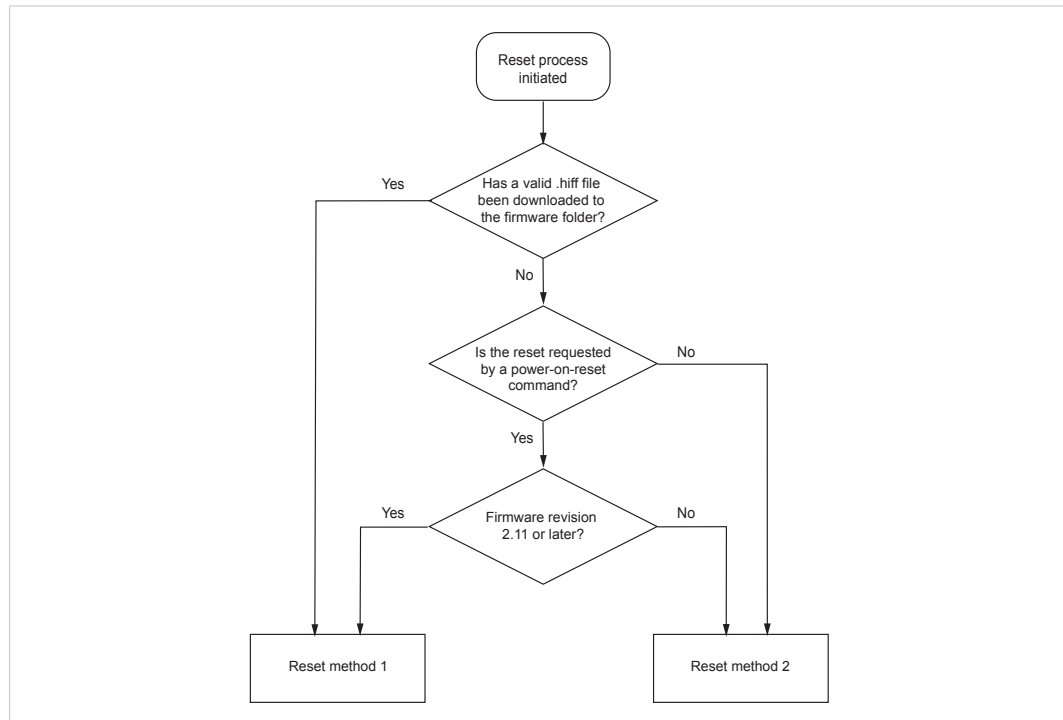


Fig. 1

- Reset method 1:** The reset signal is activated and remains so until the Anybus CompactCom can be initialized.
- Reset method 2:** A short pulse, longer than 10  $\mu$ s, is applied to the reset signal. After at least another 20 ms, the reset signal is activated again and remains so until the Anybus CompactCom can be initialized.
- This reset method should be used with applications where packet loss is not acceptable (for example, with SEMI device applications). This method ensures that other nodes on the network do not falsely detect that the CompactCom is up during reset.

#### Example of Reset Initiated by Firmware Upgrade from the EtherCAT Network

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 the CompactCom will send a Reset\_Request command to the host application, to make sure that a reset can be performed.

In this case, there is no new firmware file in the firmware folder. The firmware file has been downloaded to the CompactCom via FoE from the EtherCAT network. The reset is requested by a power-on-reset command from the CompactCom. The reset method that will be used is dependent on the firmware's revision number.

### 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).

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

The reset is triggered by writing value 0x64616F6C (=“LOAD”) to index 0x1011, Subindex 1 of the object dictionary.

See also...

- [Standard Objects, p. 22](#), entry 1011h (‘Restore Parameters’)

## 3.6 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.

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.

See also...

- [EtherCAT Object \(F5h\), p. 61](#), attribute #19 (Set Device ID as Configured station alias)

## 3.7 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 runtime, a so called HotConnect application. 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.

If the host application sets the value of the Network Configuration Object, instance 1 (Device ID), the CompactCom will indicate support for explicit device identification using register 0x134 in the SII part of the EtherCAT EEPROM. When this is the case, the ESI file must be updated by adding the following line:

```
<IdentificationReg134>true</IdentificationReg134> in the  
<Devices><Device><Info> element.
```

See also...

- [Network Configuration Object \(04h\), p. 39](#)

## 3.8 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 5001 (0x1389).

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, p. 30](#)

## 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	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 <a href="#">EtherCAT Object (F5h)</a> , p. 61. If the host application Modular Device Object is implemented, the default value is 5001 (0x1389).
1001h	Error register	00h	Error register	U8	RO	This information managed through the Diagnostic Object, see <a href="#">Diagnostic Object (02h)</a> , p. 35.
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 <a href="#">EtherCAT Object (F5h)</a> , p. 61.
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 <a href="#">EtherCAT Object (F5h)</a> , p. 61.
		02h	Product Code	U32	RO	
		03h	Revision Number	U32	RO	
		04h	Serial Number	U32	RO	



Index	Object Name	Subindex	Description	Type	Access	Notes
1600h - 1xxxh	Receive PDO mapping	00h	No. of mapped application objects in PDO	U8	RO/RW	No. of mapped objects (0.. 254), see <a href="#">Mapping ADIs on PDOs, p. 25</a> for more information. Receive PDO mapping is writable when dynamic process data is supported by the application (remap commands). <b>Note:</b> only writable in PREOP device state.
		01h	Mapped object #1	U32	RO/RW	-
		02h	Mapped object #2	U32	RO/RW	-
		...	...	...	...	-
		NNh	Mapped object #NN	U32	RO/RW	-
1A00h - 1xxxh	Transmit PDO mapping	00h	No. of mapped application objects in PDO	U8	RO/RW	No. of mapped objects (0.. 254), see <a href="#">Mapping ADIs on PDOs, p. 25</a> for more information. Transmit PDO mapping is writable when dynamic process data is supported by the application (remap commands). <b>Note:</b> only writable in PREOP device state.
		01h	Mapped object #1	U32	RO/RW	-
		02h	Mapped object #2	U32	RO/RW	-
		...	...	...	...	-
		NNh	Mapped object #NN	U32	RO/RW	-
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
1C12h	Sync Manager Rx PDO Assign	00h	No. of assigned PDOs	U8	RO/RW	When using static PDO mapping this subindex is read only. When using dynamic PDO mapping, it is writable (only writable in PREOP device state).
		01h - NNh	Assigned PDO	U16	RO/RW	
1C13h	Sync Manager Tx PDO Assign	00h	No. of assigned PDOs	U8	RO/RW	If more than one sync mode is supported, this entry is writable (only writable in PREOP device state).
		01h - NNh	Assigned PDO	U16	RO/RW	

Index	Object Name	Subindex	Description	Type	Access	Notes
1C32h	Output SyncManager Parameter	00h	Number of entries	U8	RO	12 (0Bh)
		01h	Synchronization Type	U16	RO/RW	00h: Free Run 02h: DC Sync0 See <a href="#">Sync Object (EEh), p. 59</a> .
		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 See <a href="#">Sync Object (EEh), p. 59</a> .
		05h	Minimum Cycle Time	U32	RO	Minimum cycle time in nanoseconds.
		06h	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.
1C33h	Input SyncManager Parameter	0Ch	Cycle Time Too Small	U16	RO	Cycle time too small
		00h	Number of entries	U8	RO	12 (0Bh)
		01h	Synchronization Type	U16	RO/RW	00h: Free Run 02h: DC Sync0 See <a href="#">Sync Object (EEh), p. 59</a> .
		02h	Cycle Time	U32	RO	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 See <a href="#">Sync Object (EEh), p. 59</a> .
		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).

### 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 indexes per object	Notes
Generic, static mapping	RO	1 - 6 Depends on how many ADI mapping items that are mapped by the application during setup. Each PDO can hold 254 ADI mapping items.	1 - 254 Depends on how many ADI mapping items that are mapped by the application during setup. One PDO mapping object at the time will be filled with mapped items.	
Generic, dynamic mapping	RW	1 - 6 Depends on how many ADI mapping items that are mapped by the application during setup. Each PDO can hold 254 ADI mapping items.	254 (except the 6th object, that has 216 sub indexes as the maximum number of entries is 1486)	If theTwinCAT 3, or a later version of 2.11, tool is used, the maximal number of entries will be 1473 bytes, due to limitations in the tool.
Assembly Mapping Object implemented in host	RO/RW	Number of assembly mapping instances in that direction (max 63)	1486/(number of objects) (max 254) or the max value of implemented ADI mappings	See <a href="#">Assembly Mapping Object (EBh)</a> , p. 58 for more information. Access is RO if the corresponding assembly instance is static, RW if it is dynamic The names for the PDO configuration objects can be configured using the Name attribute in the Assembly Mapping Object. If the TwinCAT 3, or a later version of 2.11, tool is used, the maximal number of entries will be 1473 bytes, due to limitations in the tool.
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)	If theTwinCAT 3, or a later version of 2.11, tool is used, the maximal number of entries will be 1473 bytes, due to limitations in the tool.

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.



*Mapped ADI elements of type PAD1 - PAD16 will be translated to Object index 0x0000, Sub-index 0x00 in the respective PDO configuration objects (16xxh/1Axxh).*

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

Mode	Access	Number of sub indexes 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.

## 4.2 Manufacturer and Profile Specific Objects

### 4.2.1 General

Each object entry in the manufacturer specific range (2001h...FFFFh) corresponds to an instance (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.



*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	
	Number of sub elements = 1	Number of sub elements > 1
BOOL	UNSIGNED8	OCTET_STRING
SINT8	INTEGER8	ARRAY_OF_SINT
SINT16	INTEGER16	ARRAY_OF_INT
SINT32	INTEGER32	ARRAY_OF_DINT
UINT8	UNSIGNED8	OCTET_STRING
UINT16	UNSIGNED16	ARRAY_OF_UINT
UINT32	UNSIGNED32	ARRAY_OF_UDINT
CHAR	VISIBLE_STRING	VISIBLE_STRING
ENUM	UNSIGNED8 or ENUM	OCTET_STRING
BITS8	BITARR8	OCTET_STRING
BITS16	BITARR16	ARRAY_OF_UINT
BITS32	BITARR32	ARRAY_OF_UDINT
OCTET	OCTET_STRING	OCTET_STRING
SINT64	INTEGER64	OCTET_STRING
UINT64	UNSIGNED64	OCTET_STRING
FLOAT	REAL32	OCTET_STRING
DOUBLE	REAL64	OCTET_STRING
PAD0-16	NULL	Not supported
BOOL1	BOOL	Not supported
BIT1 - BIT7	BIT1 - BIT7	Not supported

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.

Single element ADIs are represented as a simple variable, with the exception of CHAR which will always be represented as VISIBLE\_STRING, and OCTET which will always be represented as OCTET\_STRING.

### 4.2.3 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	-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:
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.
Protected access	0800 0021h	Data cannot be read or stored because of local control
Object Specific Error	0800 0000h	General error

If no corresponding error code can be defined on CANopen, the default error code will be General error (0800 000h).

#### 4.2.4 Object Entries

The exact representation of an ADI depends on its number of elements. In the following example, ADIs no. 0002h and 0004h only contain 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.

The offset between the ADI no. and the object index is always 2000h.

Index	Object Name	Subindex	Description	Type	Access
2001h	ADI 0001h	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.		
		...	The data type and access rights of the ADI values are determined by the ADI itself.		
		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			

#### 4.2.5 Fail Safe over EtherCAT, Object Entries

The object below shall be implemented if Fail Safe over EtherCAT is enabled.

Index	Object Name	Subindex	Description	Type	Access	Notes
F980h	Device safety address	00h	Device safety address	UN-SIGN-ED8	RO	-

#### 4.2.6 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 <a href="#">ADI to SDO Translation, p. 31</a> .
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 <a href="#">ADI to SDO Translation, p. 31</a> .
9nnnh	Information data	Any	Information objects, one for each module, occupying a slot, except the coupler.	Any	RW	For more information, see <a href="#">Module Identification Objects, p. 32</a> .
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 (Subindexes 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.
		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	-

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.



*This list comparison can be skipped, by setting the attribute Compare identity lists (attribute 22) of the EtherCAT object to FALSE.*

### 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, p. 25](#).

## 5 Anybus Module Objects

### 5.1 General Information

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

Standard Objects:

- [Anybus Object \(01h\), p. 34](#)
- [Diagnostic Object \(02h\), p. 35](#)
- [Network Object \(03h\), p. 37](#)
- [Network Configuration Object \(04h\), p. 39](#)
- File System Interface Object (0Ah), see Anybus CompactCom 40 Software Design Guide
- [Network Ethernet Object \(0Ch\), p. 46](#)
- [Functional Safety Module Object \(11h\), p. 48](#)

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 40 Software Design Guide.

### Supported Commands

<b>Object:</b>	Get_Attribute
	Reset
<b>Instance:</b>	Get_Attribute
	Set_Attribute
	Get_Enum_String

### Object Attributes (Instance #0)

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

### 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)	Value: Color: 01h Green 02h Red
13... 15	-	-	-	Consult the general Anybus CompactCom 40 Software Design Guide for further information.
16	GPIO configuration	Get/Set	UINT16	Configuration of the host interface GPIO pins. To enable Transparent Ethernet, this attribute has to be set to 0002h during SETUP state.
17	Virtual attributes	Get/Set	-	Consult the general Anybus CompactCom 40 Software Design Guide for further information.
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.

## 5.3 Diagnostic Object (02h)

### Category

Extended

### 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

<b>Object:</b>	Get_Attribute
	Create
	Delete
<b>Instance:</b>	Get_Attribute

### Object Attributes (Instance #0)

#	Name	Access	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 (Instance #1)

#### 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) in one of two possible ways:

- If the Event Code is 00h — FEh:

MSB	(UINT32)	LSB
(Not used)	(Not used)	Event Code
		00h

- If the Event Code is FFh (network specific):

MSB	(UINT32)	LSB
(Not used)	(Not used)	Network specific information (high byte)
		Network specific information (low byte)

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

Bit	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
Emergency Error 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.

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.

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.

## 5.4 Network Object (03h)

### Category

Basic

### Object Description

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

### Supported Commands

<b>Object:</b>	Get_Attribute
<b>Instance:</b>	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. Consult the general Anybus CompactCom 40 Software Design Guide for further information.
6	Read process data size	Get	UINT16	Current read process data size (in bytes). Updated on every successful Map_ADI_Read_Area, Map_ADI_Read_Ext_Area and Remap_ADI_Read_Area. Consult the general Anybus CompactCom 40 Software Design Guide for further information.
7	Exception Information	Get	UINT8	Additional information may be provided here when the module has entered the EXCEPTION state, see exception information in table below. Consult the general Anybus CompactCom 40 Software Design Guide for further information.
8... 10	(reserved)	-		

### 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.
09h	The MAC address is missing when running Anybus IP.



## 5.5 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.

As soon as the used combination of IP address, Subnet mask and Gateway is changed, the module informs the application by writing the new set to instance #1, attribute #16 in the Ethernet Host Object (F9h).

See also...

- [Ethernet Host Object \(F9h\), p. 74](#)

### Supported Commands

<b>Object:</b>	Get_Attribute Reset
<b>Instance:</b>	Get_Attribute Set_Attribute

### Object Attributes (Instance #0)

#	Name	Access	Type	Value
1	Name	Get	Array of CHAR	"Network configuration"
2	Revision	Get	UINT8	01h
3	Number of instances	Get	UINT16	13
4	Highest instance no.	Get	UINT16	21

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

Extended

See also [Device ID, p. 20](#).

Changes have immediate effect.

#	Name	Access	Type	Value
1	Name	Get	Array of CHAR	"Device ID" Multilingual, see <a href="#">Multilingual Strings, p. 45</a>
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.

### Instance Attributes (Instance #3, IP Address)

Value is used after module reset.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"IP address" (Multilingual, see <a href="#">Multilingual Strings, p. 45</a> )
2	Data type	Get	UINT8	04h (= UINT8)
3	Number of elements	Get	UINT8	04h (four elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of UINT8	If read, the actual value will be received. If written, the written value is reflected in attribute #6 until a reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)
6	Configured Value	Get	Array of UINT8	Holds the configured value, which will be written to attribute #5 after the module has been reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)



This attribute should not be set by the application at every power on, as this would cause certification problems.

### Instance Attributes (Instance #4, Subnet Mask)

Value is used after module reset.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"Subnet mask" (Multilingual, see <a href="#">Multilingual Strings, p. 45</a> )
2	Data type	Get	UINT8	04h (= UINT8)
3	Number of elements	Get	UINT8	04h (four elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of UINT8	If read, the actual value will be received. If written, the written value is reflected in attribute #6 until a reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)
6	Configured Value	Get	Array of UINT8	Holds the configured value, which will be written to attribute #5 after the module has been reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)



This attribute should not be set by the application at every power on, as this would cause certification problems.

### Instance Attributes (Instance #5, Gateway)

Value is used after module reset.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"Gateway" (Multilingual, see <a href="#">Multilingual Strings, p. 45</a> )
2	Data type	Get	UINT8	04h (= UINT8)
3	Number of elements	Get	UINT8	04h (four elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of UINT8	If read, the actual value will be received. If written, the written value is reflected in attribute #6 until a reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)
6	Configured Value	Get	Array of UINT8	Holds the configured value, which will be written to attribute #5 after the module has been reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)



This attribute should not be set by the application at every power on, as this would cause certification problems.

### Instance Attributes (Instance #6, DHCP)

Value is used after module reset.

#	Name	Access	Data Type	Description									
1	Name	Get	Array of CHAR	“DHCP” (Multilingual, see <a href="#">Multilingual Strings, p. 45</a> )									
2	Data type	Get	UINT8	08h (= ENUM)									
3	Number of elements	Get	UINT8	01h (one element)									
4	Descriptor	Get	UINT8	07h (read/write/shared access)									
5	Value	Get/Set	ENUM	<table><tr><th>Value</th><th>String</th><th>Meaning</th></tr><tr><td>00h</td><td>“Disable”</td><td>DHCP disabled (default)</td></tr><tr><td>01h</td><td>“Enable”</td><td>DHCP enabled (Multilingual, see <a href="#">Multilingual Strings, p. 45</a>)</td></tr></table>	Value	String	Meaning	00h	“Disable”	DHCP disabled (default)	01h	“Enable”	DHCP enabled (Multilingual, see <a href="#">Multilingual Strings, p. 45</a> )
Value	String	Meaning											
00h	“Disable”	DHCP disabled (default)											
01h	“Enable”	DHCP enabled (Multilingual, see <a href="#">Multilingual Strings, p. 45</a> )											
6	Configured Value	Get	ENUM	<p>Holds the configured value, which will be written to attribute #5 after the module has been reset.</p> <table><tr><th>Value</th><th>String</th><th>Meaning</th></tr><tr><td>00h</td><td>“Disable”</td><td>DHCP disabled</td></tr><tr><td>01h</td><td>“Enable”</td><td>DHCP enabled</td></tr></table>	Value	String	Meaning	00h	“Disable”	DHCP disabled	01h	“Enable”	DHCP enabled
Value	String	Meaning											
00h	“Disable”	DHCP disabled											
01h	“Enable”	DHCP enabled											

### Instance Attributes (Instances #7 - #8)

(Reserved)

### Instance Attributes (Instance #9, DNS1)

This instance holds the address to the primary DNS server. Changes are valid after reset.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"DNS1" (Multilingual, see <a href="#">Multilingual Strings, p. 45</a> )
2	Data type	Get	UINT8	04h (= UINT8)
3	Number of elements	Get	UINT8	04h (four elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of UINT8	If read, the actual value will be received. If written, the written value is reflected in attribute #6 until a reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)
6	Configured Value	Get	Array of UINT8	Holds the configured value, which will be written to attribute #5 after the module has been reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)

### Instance Attributes (Instance #10, DNS2)

This instance holds the address to the secondary DNS server. Changes are valid after reset.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"DNS2" (Multilingual, see <a href="#">Multilingual Strings, p. 45</a> )
2	Data type	Get	UINT8	04h (= UINT8)
3	Number of elements	Get	UINT8	04h (four elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of UINT8	If read, the actual value will be received. If written, the written value is reflected in attribute #6 until a reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)
6	Configured Value	Get	Array of UINT8	Holds the configured value, which will be written to attribute #5 after the module has been reset. Valid range: 0.0.0.0 - 255.255.255.255 (Default =0.0.0.0)

### Instance Attributes (Instance #11, Host name)

This instance holds the host name of the module. Changes are valid after reset.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	“Host name” (Multilingual, see <a href="#">Multilingual Strings, p. 45</a> )
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	40h (64 elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of CHAR	If read, the actual value will be received. If written, the written value is reflected in attribute #6 until a reset. Host name, 64 characters
6	Configured Value	Get	Array of CHAR	Holds the configured value, which will be written to attribute #5 after the module has been reset. Host name, 64 characters

### Instance Attributes (Instance #12, Domain name)

This instance holds the domain name. Changes are valid after reset.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	“Host name” (Multilingual, see <a href="#">Multilingual Strings, p. 45</a> )
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	30h (48 elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of CHAR	If read, the actual value will be received. If written, the written value is reflected in attribute #6 until a reset. Domain name, 48 characters
6	Configured Value	Get	Array of CHAR	Holds the configured value, which will be written to attribute #5 after the module has been reset. Domain name, 48 characters

### Instance Attributes (Instance #13, SMTP Server)

This instance holds the SMTP server address. Changes are valid after reset.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	“SMTP server” (Multilingual, see <a href="#">Multilingual Strings, p. 45</a> )
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	40h (64 elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of CHAR	If read, the actual value will be received. If written, the written value is reflected in attribute #6 until a reset. SMTP server address, 64 characters.
6	Configured Value	Get	Array of CHAR	Holds the configured value, which will be written to attribute #5 after the module has been reset. SMTP server address, 64 characters.

### Instance Attributes (Instance #14, SMTP User)

This instance holds the user name for the SMTP account. Changes are valid after reset.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"SMTP user" (Multilingual, see <a href="#">Multilingual Strings, p. 45</a> )
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	40h (64 elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of CHAR	If read, the actual value will be received. If written, the written value is reflected in attribute #6 until a reset. SMTP account user name, 64 characters
6	Configured Value	Get	Array of CHAR	Holds the configured value, which will be written to attribute #5 after the module has been reset. SMTP account user name, 64 characters

### Instance Attributes (Instance #15, SMTP Password)

This instance holds the password for the SMTP account. Changes are valid after reset.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"SMTP Pswd" (Multilingual, see <a href="#">Multilingual Strings, p. 45</a> )
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	40h (64 elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	Array of CHAR	If read, the actual value will be received. If written, the written value is reflected in attribute #6 until a reset. SMTP account password, 64 characters
6	Configured Value	Get	Array of CHAR	Holds the configured value, which will be written to attribute #5 after the module has been reset. SMTP account password, 64 characters

### Instance Attributes (Instances #16 - #20)

(Reserved)

### Instance Attributes (Instance #21, FSoE Address)

This instance holds the FSoE address when running Fail Safe over EtherCAT. Data written to the Value attribute (#5) will be saved in nonvolatile memory.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	FSoE Address (Multilingual, see <a href="#">Multilingual Strings, p. 45</a> )
2	Data type	Get	UINT8	Data type: UINT16
3	Number of elements	Get	UINT8	One data element
4	Descriptor	Get	UINT8	Bit 0: 1 = read access Bit 1: 1 = write access
5	Value	Get/Set	UINT16	FSoE address set by the host application Range: 1–65535 Default: 1 Needs power cycle to be updated
6	Configured Value	Get	UINT16	Configured value for FSoE Address. The value will be written to attribute #5 after the module has been reset.

## 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
21	FSOE Address	FSOE Adresse	Dirección FSOE	Indirizzo FSOE	FSOE Adresse

## Reset

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

## 5.6 Network Ethernet Object (0Ch)

### Category

Extended

### Object Description

This object provides Ethernet-specific information to the application.

### Supported Commands

**Object:** Get\_Attribute

**Instance:** Get\_Attribute

### Object Attributes (Instance #0)

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

### Instance Attributes (Instance #1)

#	Name	Access	Data Type	Description
1	MAC Address	Get	Array of UINT8	Reserved, used for backwards compatibility. (Device MAC address.) (See also <a href="#">Ethernet Host Object (F9h)</a> , p. 74)
2 - 3	(reserved)			
4	MAC Address	Get	Array of UINT8	Device MAC address
5 - 6	(reserved)			

### Instance Attributes (Instance #2)

#	Name	Access	Data Type	Description
1 - 4	(reserved)			
5	Interface Counters	Get	Array of UINT32	See table below for array indexes.
6	Media Counters	Get	Array of UINT32	See table below for array indexes.

### Instance Attributes (Instance #3)

(reserved)



## Interface Counters

Array indexes of Interface Counters attribute (#5)

Index	Name	Description
0	In octets	Octets received on the interface
1	In Unicast Packets	Unicast packets received on the interface
2	In Non-Unicast Packets	Non-unicast packets (multicast/broadcast) packets received on the interface
3	In Discards	Inbound packets received on the interface but discarded
4	In Errors	Inbound packets that contain errors (does not include In Discards)
5	In Unknown Protos	Inbound packets with unknown protocol
6	Out Octets	Octets transmitted on the interface
7	Out Unicast packets	Unicast packets transmitted on the interface
8	Out Non-Unicast Packets	Non-unicast (multicast/broadcast) packets transmitted on the interface
9	Out Discards	Outbound packets discarded
10	Out Errors	Outbound packets that contain errors

## Media Counters

Array indexes of Media Counters attribute (#6)

Index	Name	Description
0	AlignmentErrors;	Frames received that are not an integral number of octets in length
1	FCSErrors;	Frames received that do not pass the FCS check
2	SingleCollisions;	Successfully transmitted frames which experienced exactly one collision
3	MultipleCollisions;	Successfully transmitted frames which experienced more than one collision
4	SQETestErrors;	Number of times SQE test error is generated
5	DeferredTransmissions;	Frames for which first transmission attempt is delayed because the medium is busy
6	LateCollisions;	Number of times collision is detected later than 512 bit-times into the transmission of a packet
7	ExcessiveCollisions;	Frames for which transmission fails due to excessive collisions
8	IMACTransmitErrors;	Frames for which transmission fails due to an internal MAC sublayer transmit error
9	ICarrieSenseErrors;	Times that the carrier sense condition was lost or never asserted when attempting to transmit a frame
10	IFrameTooLong;	Frames received that exceed the maximum permitted frame size
11	IMACRecieveErrors;	Frames for which reception on an interface fails due to an internal MAC sublayer receive error

## 5.7 Functional Safety Module Object (11h)

### Category

Extended

### Object Description

This object contains information provided by the Safety Module connected to the Anybus CompactCom module. Please consult the manual for the Safety Module used, for values of the attributes below.

### Supported Commands

**Object:**

- Get\_Attribute
- Error\_Confirmation
- Set\_IO\_Config\_String
- Get\_Safety\_Output\_PDU
- Get\_Safety\_Input\_PDU

**Instance:** Get\_Attribute

### Object Attributes (Instance #0)

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

**Instance Attributes (Instance #1)**

#	Name	Access	Data Type	Description
1	State	Get	UINT8	Current state of the Safety Module Please consult the manual for the Safety Module used.
2	Vendor ID	Get	UINT16	Identifies vendor of the Safety Module. E.g. 0001h (HMS Industrial Networks) Please consult the manual for the Safety Module used.
3	IO Channel ID	Get	UINT16	Describes the IO Channels that the Safety Module is equipped with. Please consult the manual for the Safety Module used.
4	Firmware version	Get	Struct of UINT8 (Major) UINT8 (Minor) UINT8 (Build)	Safety Module firmware version. Format: version "2.18.3" would be represented as: first byte = 0x02, second byte = 0x12, third byte = 0x03.
5	Serial number	Get	UINT32	32 bit number, assigned to the Safety Module at production. Please consult the manual for the Safety Module used.
6	Output data	Get	Array of UINT8	Current value of the Safety Module output data, i.e. data FROM the network <b>Note:</b> This data is unsafe, since it is provided by the Anybus CompactCom module.
7	Input data	Get	Array of UINT8	Current value of the Safety Module input data, i.e. data sent TO the network. <b>Note:</b> This data is unsafe, since it is provided by the Anybus CompactCom module.
8	Error counters	Get	Struct of UINT16 (ABCC DR) UINT16 (ABCC SE) UINT16 (SM DR) UINT16 (SM SE)	Error counters (each counter stops counting at FFFFh)  ABCC DR: Responses (unexpected) from the Safety Module, discarded by the Anybus CompactCom module.  ABCC SE: Serial reception errors detected by the Anybus CompactCom module.  SM DR: Responses (unexpected) from the Anybus CompactCom module, discarded by the Safety Module.  SM SE: Serial reception errors detected by the Safety Module.
9	Event log	Get	Array of UINT8	Latest Safety Module event information (if any) is logged to this attribute. Any older event information is erased when a new event is logged. For evaluation by HMS support.
10	Exception information	Get	UINT8	If the Exception Code in the Anybus object is set to "Safety communication error" (09h), additional exception information is presented here, see table below.
11	Bootloader version	Get	Struct of UINT8 Major UINT8 Minor	Safety Module bootloader version. Format: version "2.12" would be represented as: first byte = 0x02, second byte = 0x0C
12	Vendor block safe uc1	Get	Array of UINT8	The Safety Module may supply additional vendor-specific data to the Anybus CompactCom. If such data is available it is presented in this attribute.
13	Vendor block safe uc2	Get	Array of UINT8	The Safety Module may supply additional vendor-specific data to the Anybus CompactCom. If such data is available it is presented in this attribute.

**Exception Information**

If Exception Code 09h is set in the Anybus object, there is an error regarding the functional safety module in the application. Exception information is presented in instance attribute #10 according to this table:

Value	Exception Information
00h	No information
01h	Baud rate not supported
02h	No start message
03h	Unexpected message length
04h	Unexpected command in response
05h	Unexpected error code
06h	Safety application not found
07h	Invalid safety application CRC
08h	No flash access
09h	Answer from wrong safety processor during boot loader communication
0Ah	Boot loader timeout
0Bh	Network specific parameter error
0Ch	Invalid IO configuration string
0Dh	Response differed between the safety microprocessors (e.g. different module types)
0Eh	Incompatible module (e.g. supported network)
0Fh	Max number of retransmissions performed (e.g. due to CRC errors)
10h	Firmware file error
11h	The cycle time value in attribute #4 in the Functional Safety Host Object can not be used with the current baud rate
12h	Invalid SPDU input size in start-up telegram
13h	Invalid SPDU output size in start-up telegram
14h	Badly formatted input SPDU
15h	Anybus to safety module initialization failure

## Command Details: Error\_Confirmation

### Category

Extended

### Details

<b>Command Code</b>	10h
<b>Valid for:</b>	Object

### Description

When the Safety Module has entered the Safe State, for any reason, it must receive an error confirmation before it can leave the Safe State. With this command it is possible to reset all safety channels of the safety which, for any reason, are in the Safe State at the same time. The application issues this command to the Anybus CompactCom module, when an error has been cleared by for example an operator. The Anybus CompactCom forwards the command to the Safety Module.

The channel Safe State can also be confirmed by the safety PLC or by the safety module.

With this command

- Command Details  
(no data)
- Response Details  
(no data)

## Command Details: Set\_IO\_Config\_String

### Category

Extended

### Details

<b>Command Code</b>	11h
<b>Valid for:</b>	Object

### Description

This command is sent from the host application when there is a need to change the default configuration of the safety inputs and outputs. This string is used by networks where there are no other means (e.g. PLC or some other tool) to provide the configuration to the safety module. Consult the specification of the safety module for more information. The byte string passed is generated by HMS and need to be passed unmodified using this command.

Information about this string is located in the specification of the safety module to which the string shall be sent.

- Command Details

Field	Contents
CmdExt[0]	(not used)
CmdExt[1]	
Data[0... n]	Data (byte string) The data consists of an IO configuration string, where the data format depends on the safety network.

- Response Details

(no data)

## Command Details: Get\_Safety\_Output\_PDU

### Category

Extended

### Details

<b>Command Code</b>	12h
<b>Valid for:</b>	Object

### Description

This command can be issued by the application to get the complete safety output PDU sent by the PLC. The Anybus CompactCom 40 EtherCAT will respond with the complete safety PDU, that the application then has to interpret.

- Command Details  
(no data)
- Response Details

Field	Contents
CmdExt[0]	(not used)
CmdExt[1]	
Data[0... n]	Safety PDU from PLC

## Command Details: Get\_Safety\_Input\_PDU

### Category

Extended

### Details

<b>Command Code</b>	13h
<b>Valid for:</b>	Object

### Description

This command can be issued by the application to get the complete safety input PDU sent by the safety module. The Anybus CompactCom 40 EtherCAT will respond with the complete safety PDU, that the application then has to interpret.

- Command Details  
(no data)
- Response Details

Field	Contents
CmdExt[0]	(not used)
CmdExt[1]	
Data[0... n]	Safety PDU from safety module

**Object Specific Error Codes**

Error Code	Description	Comments
01h	The safety module rejected a message.	Error code sent by safety module is found in MsgData[2] and MsgData[3].
02h	Message response from the safety module has incorrect format (for example, wrong length).	-



## 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 40 Software Design Guide)
- Application File System Interface Object (see Anybus CompactCom 40 Software Design Guide)
- *Functional Safety Object (E8h), p. 56*
- *Assembly Mapping Object (EBh), p. 58*
- *Sync Object (EEh), p. 59*
- Modular Device Object (see Anybus CompactCom 40 Software Design Guide)
- Application Data Object (see Anybus CompactCom 40 Software Design Guide)
- *Ethernet Host Object (F9h), p. 74*

Network Specific Objects:

- *EtherCAT Object (F5h), p. 61*

## 6.2 Functional Safety Object (E8h)

### Category

Extended

### Object Description



Do not implement this object if a safety module is not used.

This object specifies the safety settings of the application. It is mandatory if Functional Safety is to be supported and a Safety Module is connected to the Anybus CompactCom module.

### Supported Commands

**Object:** Get\_Attribute

**Instance:** Get\_Attribute

### Object Attributes (Instance #0)

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

### Instance Attributes (Instance #1)

#	Name	Access	Data Type	Default Value	Comment
1	Safety enabled	Get	BOOL	-	When TRUE, enables communication with the Safety Module. <b>Note:</b> If functional safety is not supported, this attribute must be set to FALSE.
2	Baud Rate	Get	UINT32	1020 kbit/s	This attribute sets the baud rate of the communication in bits/s between the Anybus CompactCom and the Safety Module. Valid values: <ul style="list-style-type: none"> <li>625 kbit/s</li> <li>1000 kbit/s</li> <li>1020 kbit/s (default)</li> </ul> Any other value set to this attribute, will cause the module to enter the EXCEPTION state. The attribute is optional. If not implemented, the default value will be used. <b>Note:</b> The host application shall never implement this attribute when using the IXXAT Safe T100.
3	(reserved)				

#	Name	Access	Data Type	Default Value	Comment
4	Cycle Time	Get	UINT8	-	<p>Communication cycle time between the Anybus and the Safety module in milliseconds.</p> <p><b>Note:</b> The host application shall never implement this attribute when using the IXXAT Safe T100.</p> <p>Valid values:</p> <ul style="list-style-type: none"> <li>• 2 ms</li> <li>• 4 ms</li> <li>• 8 ms</li> <li>• 16 ms</li> </ul> <p>If another value is set in this attribute the Anybus will enter Exception state.</p> <p>Optional attribute; If not implemented the minimum cycle time for the chosen baud rate will be used:</p> <ul style="list-style-type: none"> <li>• 2 ms for 1020 kbit/s</li> <li>• 2 ms for 1000 kbit/s</li> <li>• 4 ms for 625 kbit/s</li> </ul> <p>The Anybus CompactCom validates the cycle time according to the minimum values above. If e.g. baud rate is 625 kbit/s and the cycle time is set to 2 ms the Anybus CompactCom will enter the EXCEPTION state.</p>
5	FW upgrade in progress	Set	BOOL	False	<p>Indicates if the Anybus CompactCom is upgrading the connected Safety module firmware. This means that the Anybus CompactCom will stay in the NW_INIT state longer than normal.</p>

## 6.3 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.

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, p. 22](#) for assembly to PDO mapping.

## 6.4 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, p. 85](#)

See also ...

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

### Supported Commands

<b>Object:</b>	Get_Attribute
<b>Instance:</b>	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	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) The Anybus latency is added to this value before it is presented on EtherCAT.
5	Input processing	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) The Anybus latency is added to this value before it is presented on EtherCAT.
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)

## 6.5 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 startup; if an attribute is not implemented in the host application, simply respond with an error message (06h, "Invalid CmdExt[0]"). In such case, the module will use its default value.

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



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

<b>Object:</b>	Get_Attribute
	Get_Object_Description
	Get_Object_Access
	Get_Data_Type
	Get_Enum_Data
<b>Instance:</b>	Get_Attribute
	Set_Attribute

### Object Attributes (Instance #0)

#	Name	Access	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	E000 001Bh	These values replace the settings in object entry 1018h. (Identity Object) Note: The default Vendor ID is an HMS secondary Vendor ID, that cannot be used when running the conformance test tool.

### Extended

#	Name	Access	Type	Default Value	Comment
2	Product Code	Get	UINT32	0000 0036h	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)	"CompactCom 40 EtherCAT"	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	0000 0000h	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. <a href="#">See ADI Translation, Example, p. 69</a>
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 subindexes of objects in the profile specific CoE object area (0x1000-0x1FFF). Subindexes already implemented in the module cannot be replaced by ADIs and this attribute can only be used to add subindexes to objects explicitly defined in the module to be extendable. 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. <a href="#">See: Object Subindex Translation, Example, p. 69</a>
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. FoE is not supported for Anybus IP.
17	Enable EoE	Get	BOOL	TRUE (=1)	Enables/Disables functionality related to Ethernet over EtherCAT. If EoE is disabled the module will not accept any mailbox requests of EoE type and no IT functionality in the module will be usable.
18	Change shift reg switch functionality	Get	BOOL	FALSE (=0)	Normally when running shift register operation mode, switch 1 is used for the last octet of the IP address and switch 2 is used for the Device ID. If this attribute is set to TRUE this behavior is changed so switch 1 is used for Device ID and switch 2 is used for the last octet of the IP address.
19	Set Device ID as Configured station alias	Get	BOOL	FALSE (=0)	Normally the Configured station alias value can only be set from the EtherCAT configuration tool. This is the case when this attribute is either false or not implemented. If this attribute is set to True, the value set to instance 1 of the network configuration object (Device ID) will be set to the configured station alias as well (both the register 0x0012 and the EEPROM).
20	EtherCAT state	Set	UINT8	1 (=INIT)	Whenever the EtherCAT state of the CompactCom is changed the module will write the new state to this attribute.  1: INIT 2: PRE-OPERATIONAL 3: BOOT 4: SAFE-OPERATIONAL 8: OPERATIONAL  <b>Note:</b> Since this attribute is set using the acyclic message channel it should not be relied upon for checking e.g. process data validity. Use the CompactCom state for such information. <b>Note:</b> Writes towards this attribute are informational only, it is not possible for the host application to NAK a state transition by returning an error on the Set_Attribute request.

#	Name	Access	Type	Default Value	Comment
21	State transition timeouts	Get	Array of UINT32[4]	[1000, 5000, 1000, 200]	<p>This attribute can be implemented to change the EtherCAT state transition timeouts. The attribute is an array of UINT32 with 4 elements where each element means the following:</p> <p>0: PreopTimeout in milliseconds. Timeout for the INIT-&gt;PREOP and INIT-&gt;BOOT transitions. Corresponds to ESI element PreopTimeout.</p> <p>1: SafeopOpTimeout in milliseconds. Timeout for the SAFEOP-&gt;OP and PREOP-&gt;SAFEOP transitions. Corresponds to ESI element SafeopOpTimeout.</p> <p>2: BackToInitTimeout in milliseconds. Timeout for all state transitions back to the INIT state. Corresponds to ESI element BackToInitTimeout.</p> <p>3: BackToSafeopTimeout in milliseconds. Timeout for the OP-&gt;SAFEOP transition. Corresponds to ESI element BackToSafeopTimeout.</p> <p>See section below for more information about state transition timeouts</p>
22	Compare identity lists	Get	BOOL	TRUE (=1)	<p>This attribute is only relevant when using the modular device profile.</p> <p>When set to false the module will not compare the configured and detected module identity lists on the PREOP to SAFEOP transition.</p>
23	FSoE status indicator	Set	ENUM	255 (=Status not indicated)	<p>This attribute is only relevant when safety over EtherCAT is enabled.</p> <p>This attribute is used by the host to update the FSoE status LED. When the CompactCom module gets a LED state change from the safety module this attribute is updated.</p> <p>0 = OFF  1 = Blinking  2 = ON  3 = Single flash  4 = Flickering  5 = Flickering with 1 flash  6 = Flickering with 2 flashes  7 = Flickering with 3 flashes  8 = Flickering with 4 flashes  9 = Flickering with 5 flashes  10 = Flickering with 6 flashes  254 = Fail-safe state  255 = Status not indicated.</p>
24	Clear IdentALSts	Get	BOOL	FALSE (=0)	<p>When this attribute is implemented and set to 1 (=True) the CompactCom will always clear the "IdentALSts" bit in the EEPROM general category. This way the CompactCom will not indicate support for Explicit device identification. If the attribute isn't implemented, or set to 0 (=False), the "IdentALSts" flag in the EEPROM general category will be set to 1 as soon as the value attribute of Network configuration object instance 1 (Device ID) is written. Normally this attribute only needs to be supported by host applications using the shift register operation mode and lacking the Device ID switch.</p>
25	SII Order Number	Get	Array of CHAR(Max 64 Bytes)	Value of object 1008h, Manufacturer Device Name (See attribute #6 above).	<p>This attribute only needs to be implemented if the host application needs a value other than CoE object 0x1008h to be reported in the SII Order Number string.</p>
26	SII Device Name	Get	Array of CHAR(Max 64 Bytes)	Value of object 1008h, Manufacturer Device Name (See attribute #6 above).	<p>This attribute only needs to be implemented if the host application needs a value other than CoE object 0x1008h to be reported in the SII Device Name Information string.</p>

#	Name	Access	Type	Default Value	Comment
27	Last FoE Data ACK delay	Get	UINT16	500	<p>Time in milliseconds that the Anybus CompactCom 40 EtherCAT will delay the ACK to the last FoE_Data request in FoE write file transfers.</p> <p><b>Note:</b> if a file is opened for writing in the CompactCom firmware candidate area before this delay time has passed, the Anybus CompactCom will send the ACK immediately.</p> <p>This delay is intended for applications supporting the SEMI device profile firmware upgrade specification, but could also be used by applications used in a configuration where the EtherCAT master has configured a very short timeout for FoE transfers.</p>
28	Default TxPDO assignment	Get	Array of UINT16 (Max 64 elements)		<p>This attribute gives control of the default content of 0x1C13 to the host application. When the attribute is implemented it will override the default content, making it possible for the host application to assign the required TxPDOs by default.</p> <p>The data of the attribute is an array of UINT16 values where each element is a TxPDO configuration object index, ordered the way that the TxPDOs should be assigned.</p> <p><b>Note:</b> The host application can only assign TxPDOs that actually exist using this attribute, otherwise the module will report AL Status Code 0x0024 (Invalid input mapping) on the PREOP-&gt;SAFEOP transition.</p> <p><b>Note:</b> The module will ignore the attribute if a list longer than 64 UINT16 elements are returned.</p> <p>This attribute is normally used together with the assembly mapping object to assign more than the first TxPDO to object 0x1C13, but can also be used for other PDO mapping modes to adjust the default TxPDO assignment.</p> <p>For an example, see <a href="#">Attribute #28: Default TxPDO Assignment Example, p. 66</a>.</p>
29	Default RxPDO assignment	Get	Array of UINT16 (Max 64 elements)		<p>This attribute gives control of the default content of 0x1C12 to the host application. When the attribute is implemented it will override the default content, making it possible for the host application to assign the required RxPDOs.</p> <p>The data of the attribute is an array of UINT16 values where each element is a RxPDO configuration object index, ordered the way that the RxPDOs should be assigned.</p> <p><b>Note:</b> The host application can only assign RxPDOs that actually exist using this attribute, otherwise the module will report AL Status Code 0x0025 (Invalid output mapping) on the PREOP-&gt;SAFEOP transition.</p> <p><b>Note:</b> The module will ignore the attribute if a list longer than 64 UINT16 elements are returned.</p> <p>This attribute is normally used together with the assembly mapping object to assign more than the first RxPDO to object 0x1C12, but can also be used for other PDO mapping modes to adjust the default RxPDO assignment.</p> <p>For an example, see <a href="#">Attribute #29: Default RxPDO Assignment Example, p. 67</a>.</p>
30	SII General category CoE Details	Get	UINT8		<p>Bit 0: Not used. Set to 1 for future compatibility.</p> <p>Bit 1: Not used. Set to 1 for future compatibility.</p> <p>Bit 2: Enable PDO Assign.</p> <p>Bit 3: Enable PDO Configuration.</p> <p>Bit 4: Enable PDO Upload at start up.</p> <p>Bit 5: Not used. Set to 1 for future compatibility.</p> <p>Bit 6-7: Reserved.</p> <p>For more information, see <a href="#">Attribute #30: SII General category CoE Details, p. 67</a>.</p>

## Attribute #21: State Transition Timeouts

The default state transition timeouts can normally be used. There are however a few situations where a host application needs to change the state transition timeouts:

- If it takes a long time for the application to execute the process data remap commands, the SafeOpTimeout may need to be increased. Remapping from the EtherCAT side is done on transition from PREOP->SAFEOP. This timeout needs to be increased if there is a risk of exceeding the default timeout of 5 seconds.
- If the host application can operate synchronously and needs a long time to e.g. lock on to the sync signal, the SafeOp timeout may need to be increased.
- If the host application supports firmware upgrade over EtherCAT, using the application file system interface object, it may be necessary to increase the BackToInitTimeout. This is needed if e.g. the host application must validate the HIFF file, it has uploaded to the Anybus file system interface object, on the BOOT->INIT transition. For firmware upgrade over EtherCAT, please refer to “SEMI Device FW Upgrade” from the EtherCAT Technology Group.

If these timeouts are changed the ESI file has to be updated with the new timeouts, see “EtherCAT Slave Information Specification” from the EtherCAT Technology Group (table 71).

## Attribute #28: Default TxPDO Assignment Example

Let’s assume the host application has implemented the assembly mapping object and created instance 1-3 and made these instances write assemblies. This will result in 3 TxPDOs (0x1A00, 0x1A01 and 0x1A02) created in the module. However, only TxPDO 0x1A00 is assigned by default to object 0x1C13. The default content of object 0x1C13 will look like this:

Sub-index	Value
0	1
1	0x1A00

If the host application wants to assign all of TxPDOs by default it can implement the “Default TxPDO assignment” attribute. In this case an array with 3 elements shall be created with the following content: { 0x1A00, 0x1A01, 0x1A02 }.

This will result in the following default content in object 0x1C13:

Sub-index	Value
0	3
1	0x1A00
2	0x1A01
3	0x1A02

### Attribute #29: Default RxPDO Assignment Example

Let's assume the host application has implemented the assembly mapping object and created instance 1-3 and made these instances read assemblies. This will result in 3 RxPDOs (0x1600, 0x1601 and 0x1602) created in the module. However, only RxPDO 0x1600 is assigned by default to object 0x1C12. The default content of object 0x1C12 will look like this:

Sub-index	Value
0	1
1	0x1600

If the host application wants to assign all of the RxPDOs by default it can implement the "Default RxPDO assignment" attribute. In this case an array with 3 elements shall be created with the following content: { 0x1600, 0x1601, 0x1602 }.

This will result in the following default content in object 0x1C12:

Sub-index	Value
0	3
1	0x1600
2	0x1601
3	0x1602

### Attribute #30: SII General category CoE Details

This attribute is used by the host application to override the default value of the "CoE Details" field in the SII General category.

The bits in this attribute is handled the following way by the module:

Bit	Description
0	Not used
1	Not used
2	<b>Enable PDO assign</b> Set to 1 if the master should download the PDO assignment (objects 0x1C12/0x1C13) on the PREOP->SAFEOP transition. Set to 0 if the PDO assignment shouldn't be downloaded. <b>Note:</b> This bit shall only be set to 1 if the PDO assignment objects have RW access.
3	<b>Enable PDO configuration</b> Set to 1 if the master should download the PDO configuration (objects 0x16yy/0x1Ayy) on the PREOP->SAFEOP transition. Set to 0 if the PDO configuration shouldn't be downloaded. <b>Note:</b> This bit shall only be set to 1 if the PDO configuration objects have RW access.
4	<b>Enable PDO upload</b> Set to 1 if the device has dynamic process data and make the master upload the PDO assignment and PDO configuration from the device. Set to 0 if the the master should take the PDO assignment and PDO configuration from the ESI file. <b>Note:</b> If this bit is set in addition to bit 2 and/or 3 the master will upload the PDO description on the INIT->PREOP transition and then download it on the PREOP->SAFEOP transition.
5	Not used
6-7	Reserved.

## Command Details: Get\_Object\_Description

### Category

Extended

### Details

<b>Command Code</b>	10h
<b>Valid for:</b>	Object

### Description

This command can optionally be supported by the host application to change the information added by the CompactCom in “Get Object Description” SDO information responses for ADI structs.

Normally an ADI struct is translated into a RECORD object with EtherCAT data type 0x2A (No predefined RECORD).

By implementing support for this command the host application can change both the object code and the object data type returned for an object corresponding to an ADI struct.

This is needed when implementing some EtherCAT profiles, e.g. some of the SEMI device profiles.

If the host application responds with one of the error codes 0x03 (Unsupported object), 0x04 (Unsupported instance) or 0x05 (Unsupported command) on this command, the CompactCom will not use this command for future Get Object Description requests.

- Command Details

Field	Contents	Comments
CmdExt[0]	Object index, low byte	This is the object index that received a Get Object Description SDO information request.
CmdExt[1]	Object index, high byte	

- Response Details

Field	Contents	Comments
CmdExt[0]	Object index, low byte (mirrored from command)	This is the object index that received a Get Object Description SDO information request.
CmdExt[1]	Object index, high byte (mirrored from command)	
Data[0-1]	EtherCAT data type (UINT16)	EtherCAT data type that should be reported in the Get Object Description response for the supplied object index.
Data[2]	EtherCAT object code (UINT8)	EtherCAT object code that should be reported in the Get Object Description response for the supplied object index.

## 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:

### Example 1:

```
[
  {
    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:

### Example 2:

```
[
  {
    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 subindexes, and nothing is forwarded to the host application.

## Command Details: Get\_Object\_Access

### Category

Extended

### Details

<b>Command Code</b>	11h
<b>Valid for:</b>	Object

### Description

This command can optionally be supported by the host application to set the “Object Access” in “SDO Info Service Data” during a “Get Entry Description Request”.

It can be used e.g. to customize the access flags, e.g. making an object writable only in PREOP state, or setting the “Setting” flag that is mandatory for some SEMI device profiles.

If the host application responds with one of the error codes 0x03 (Unsupported object), 0x04 (Unsupported instance) or 0x05 (Unsupported command) on this command the CompactCom will not use this command for future Get Entry Description requests.

- Command Details

Field	Contents	Comments
CmdExt[0]	Object index, low byte	This is the object index that received a Get Entry Description request.
CmdExt[1]	Object index, high byte	
Data[0]	Sub index	This is the entry index that received a Get Entry Description Request.

- Response Details

Field	Contents	Comments
Data[0]	Object Access, low byte	EtherCAT Object Access data that should be reported in the Get Entry Description Response for the supplied object entry.
Data[1]	Object Access, high byte	



## Command Details: Get\_Data\_Type

### Category

Extended

### Details

<b>Command Code</b>	12h
<b>Valid for:</b>	Object

### Description

This command can optionally be supported by the host application to set the “Data Type” in “SDO Info Service Data” during a “Get Entry Description Request”.

This is needed to create e.g. ENUM object entries of size other than 8 bits, which is required when implementing some EtherCAT profiles, e.g. some of the SEMI device profiles.

If the host application responds with one of the error codes 0x03 (Unsupported object), 0x04 (Unsupported instance) or 0x05 (Unsupported command) on this command the CompactCom will not use this command for future Get Entry Description requests.

- Command Details

Field	Contents	Comments
CmdExt[0]	Object index, low byte	This is the object index that received a Get Entry Description request.
CmdExt[1]	Object index, high byte	
Data[0]	Sub index	This is the sub index that received a Get Entry Description Request.

- Response Details

Field	Contents	Comments
Data[0]	Data Type, low byte	EtherCAT Data Type data that should be reported in the Get Entry Description Response for the supplied object entry.
Data[1]	Data Type, high byte	

## Command Details: Get\_Enum\_Data

### Category

Extended

### Details

<b>Command Code</b>	13h
<b>Valid for:</b>	Object

### Description

This command can optionally be implemented to define EtherCAT Enumeration definitions in a transparent way by the host application. Using this command combined with the Get\_Data\_Type command it is possible to define object entries of an ENUM data type within a RECORD object. It is also possible to create ENUMs with bit lengths other than 8 bits.

If the host application responds with one of the error codes 0x03 (Unsupported object), 0x04 (Unsupported instance) or 0x05 (Unsupported command) on this command the CompactCom will stop using this command.

### Usage of the Get\_Enum\_Data command

Due to limitations in the Get\_Enum\_String command for the application data object, it is not possible to create an object entry of ENUM data type within a RECORD object. This is however required by some EtherCAT profiles. Some EtherCAT profiles also require ENUM object entries of other lengths than 8 bits. Follow the steps below to create a RECORD object that contains an ENUM object entry with a bit length other than 8.

1. Create a structured ADI containing the desired elements using a data type where the bit length matches the ENUM data type. In this example we will create a RECORD object with object index 0x2001 with an UINT16 object entry followed by an ENUM32 object entry. The ADI created shall have instance number 1, two elements and the data type element shall contain [UINT16, UINT32]. Normally this would yield a RECORD object with object index 0x2001 where the object entries have data types UINT16 and UINT32.
2. The next step is to override the data type for sub-index 2 (the sub-index that shall be an ENUM32 entry) on the object. This is done using the Get\_Data\_Type command. For details, see [Command Details: Get\\_Data\\_Type, p. 71](#). When a Get\_Data\_Type command addressing object index 0x2001 and sub-index 2 is received, the data type added in the response shall be within the EtherCAT ENUM data type area. This area goes from index 0x0800 up to and including 0xFFFF. Each index defines an ENUM data type. In this example we return data type 0x0800 in the response to the Get\_Data\_Type command.
3. Finally the Get\_Enum\_Data command needs to be implemented. The Get\_Enum\_Data command is used by the CompactCom to fetch information about the enumeration from the host application when this information is requested by the EtherCAT master. The structure for the ENUM data type objects (0x0800-0xFFFF) looks like this:

#### Enumeration Data Type Object Definition

Sub-index	Description	Data Type	Value
0	Number of ENUM values	UINT8	N (Number of values for the enumeration)
1	Allowed value 1	OCTET_STRING	Byte 0-3: Value 1 Byte 4 and onwards: Textual description of value 1
2	Allowed value 2	OCTET_STRING	Byte 0-3: Value 2 Byte 4 and onwards: Textual description of value 2
...	...	...	...
N	Allowed value N	OCTET_STRING	Byte 0-3: Value N Byte 4 and onwards: Textual description of value N

The data returned in the response for the Get\_Enum\_Data command will be sent unchanged to the EtherCAT master. The host application must therefore fill in the response with the correct data according to the table above depending on requested sub-index, or return an appropriate error code.

- Command Details

Field	Contents	Comments
CmdExt[0]	Object index, low byte	This is the data type index requested by the EtherCAT master.
CmdExt[1]	Object index, high byte	
Data[0]	Sub index	This is the data type sub index requested by the EtherCAT master.

- Response Details

Field	Contents	Comments
Data[...]	Enumeration data of the requested index and sub index	For details on the data to return, see the Enumeration Data Type Object Definition table above.

## 6.6 Ethernet Host Object (F9h)

### Object Description

This object implements Ethernet features in the host application.

### Supported Commands

**Object:** Get\_Attribute

**Instance:** Get\_Attribute

Set\_Attribute

### Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1	Name	Get	Array of CHAR	"Ethernet"
2	Revision	Get	UINT8	02h
3	Number of instances	Get	UINT16	0001h
4	Highest instance no.	Get	UINT16	0001h

### Instance Attributes (Instance #1)

- If an attribute is not implemented, the default value will be used.
- The module is preprogrammed with a valid MAC address. To use that address, do not implement attribute #1.
- Do not implement attributes #9 and #10, only used for PROFINET devices, if the module shall use the preprogrammed MAC addresses.
- If new MAC addresses are assigned to a PROFINET device, these addresses (in attributes #1, #9, and #10) have to be consecutive, e.g. (xx:yy:zz:aa:bb:01), (xx:yy:zz:aa:bb:02), and (xx:yy:zz:aa:bb:03) with the first five octets not changing.

#	Name	Access	Data Type	Default Value	Comment
1	MAC address	Get	Array of UINT8	-	6 byte physical address value; overrides the preprogrammed Mac address. Note that the new Mac address value must be obtained from the IEEE. Do not implement this attribute if the preprogrammed Mac address is to be used.
2	Enable HICP	Get	BOOL	True (Enabled)	Enable/Disable HICP
3	Enable Web Server	Get	BOOL	True (Enabled)	Enable/Disable Web Server (Not used if Transparent Ethernet is enabled.)
4	(reserved)				Reserved for Anybus CompactCom 30 applications.
5	Enable Web ADI access	Get	BOOL	True (Enabled)	Enable/Disable Web ADI access (Not used if Transparent Ethernet is enabled.)
6	Enable FTP server	Get	BOOL	True (Enabled)	Enable/Disable FTP server (Not used if Transparent Ethernet is enabled or if device supports IIoT secure functionality.)
7	Enable admin mode	Get	BOOL	False (Disabled)	Enable/Disable admin mode (Not used if Transparent Ethernet is enabled.)
8	Network Status	Set	UINT16	-	See below.

#	Name	Access	Data Type	Default Value	Comment
9	Port 1 MAC address	Get	Array of UINT8	-	<b>Note:</b> This attribute is only valid for PROFINET devices. 6 byte MAC address for port 1 (mandatory for the LLDP protocol). This setting overrides any Port MAC address in the host PROFINET IO Object. Do not implement this attribute if the preprogrammed Mac address is to be used.
10	Port 2 MAC address	Get	Array of UINT8	-	<b>Note:</b> This attribute is only valid for PROFINET devices. 6 byte MAC address for port 2 (mandatory for the LLDP protocol). This setting overrides any Port MAC address in the host PROFINET IO Object. Do not implement this attribute if the preprogrammed Mac address is to be used.
11	Enable ACD	Get	BOOL	True (Enabled)	Enable/Disable ACD protocol. If ACD functionality is disabled using this attribute, the ACD attributes in the CIP TCP/IP object (F5h) are not available.
12	Port 1 State	Get	ENUM	0 (Enabled)	The state of Ethernet port 1. <ul style="list-style-type: none"> <li>This attribute is not read by EtherCAT and Ethernet POWERLINK devices, where Port 1 is always enabled.</li> </ul> 00h: Enabled 01h: Disabled. The port is treated as existing. References to the port can exist, e.g. in network protocol or on website.
13	Port 2 State	Get	ENUM	0 (Enabled)	The state of Ethernet port 2. <ul style="list-style-type: none"> <li>This attribute is not read by EtherCAT and Ethernet POWERLINK devices, where Port 2 is always enabled.</li> </ul> 00h: Enabled 01h: Disabled. The port is treated as existing. References to the port can exist, e.g. in network protocol or on website. 02h: Inactive. The attribute is set to this value for a device that only has one physical port. All two-port functionality is disabled. No references can be made to this port. <b>Note:</b> This functionality is available for PROFINET, Ethernet/IP and Modbus-TCP devices.
14	(reserved)				
15	Enable reset from HICP	Get	BOOL	0 = False	Enables the option to reset the module from HICP.
16	IP configuration	Set	Struct of: UINT32 (IP address) UINT32 (Subnet mask) UINT32 (Gateway)	N/A	Whenever the configuration is assigned or changed, the Anybus CompactCom module will update this attribute.

#	Name	Access	Data Type	Default Value	Comment
17	IP address byte 0–2	Get	Array of UINT8 [3]	[0] = 192 [1] = 168 [2] = 0	First three bytes in IP address. Used in standalone shift register mode if the configuration switch value is set to 1-245. In that case the IP address will be set to: Y[0].Y[1].Y[2].X Where Y0-2 is configured by this attribute and the last byte X by the configuration switch.
18	Ethernet PHY Configuration	Get	Array of BITS16	0x0000 for each port	Ethernet PHY configuration bit field. The length of the array shall equal the number of Ethernet ports of the product. Each element represents the configuration of one Ethernet port (element #0 maps to Ethernet port #1, element #1 maps to Ethernet port #2 and so on). <b>Note:</b> Only valid for EtherNet/IP and Modbus-TCP devices.  Bit 0:        Auto negotiation fallback duplex 0 = Half duplex 1 = Full duplex  Bit 1–15:    Reserved
20	SNMP read-only community string	Get	Array of CHAR	“public”	<b>Note:</b> This attribute is only valid for PROFINET devices. Sets the SNMP read-only community string. Max length is 32.
21	SNMP read-write community string	Get	Array of CHAR	“private”	<b>Note:</b> This attribute is only valid for PROFINET devices. Sets the SNMP read-write community string. Max length is 32.
22	DHCP Option 61 source	Get	ENUM	0 (Disabled)	<b>Note:</b> This attribute is currently only valid for Ethernet/IP devices. See below (DHCP Option 61, Client Identifier)
23	DHCP Option 61 generic string	Get	Array of UINT8	N/A	<b>Note:</b> This attribute is currently only valid for Ethernet/IP devices. See below (DHCP Option 61, Client Identifier)
24	Enable DHCP Client	Get	BOOL	1 = True	<b>Note:</b> This attribute is currently valid for Ethernet/IP and PROFINET devices. Enable/disable DHCP Client functionality 0:            DHCP Client functionality is disabled 1:            DHCP Client functionality is enabled
25	Enable WebDAV Server	Get	BOOL	1 = True	<b>Note:</b> This attribute is currently valid for devices with IIoT Secure functionality. Enable/disable WebDAV server 0:            WebDAV functionality is disabled 1:            WebDAV functionality is enabled

## Network Status

This attribute holds a bit field which indicates the overall network status as follows:

Bit	Contents	Description	Comment
0	Link	Current global link status 1= Link sensed 0= No link	EtherCAT only: This link status indicates whether the Anybus CompactCom is able to communicate using Ethernet over EtherCAT (EoE) or not. That is, it indicates the status of the logical EoE port link and is not related to the link status on the physical EtherCAT ports.
1	IP established	1 = IP address established 0 = IP address not established	
2	(reserved)	(mask off and ignore)	
3	Link port 1	Current link status for port 1 1 = Link sensed 0 = No link	EtherCAT only: This link status indicates whether the Anybus CompactCom is able to communicate using Ethernet over EtherCAT (EoE) or not. That is, it indicates the status of the logical EoE port link and is not related to the link status on the physical EtherCAT ports.

Bit	Contents	Description	Comment
4	Link port 2	Current link status for port 2 1 = Link sensed 0 = No link	Not used for EtherCAT
5... 15	(reserved)	(mask off and ignore)	

## DHCP Option 61 (Client Identifier)



*Only valid for EtherNet/IP devices*

The DHCP Option 61 (Client Identifier) allow the end-user to specify a unique identifier, which has to be unique within the DHCP domain.

Attribute #22 (DHCP Option 61 source) is used to configure the source of the Client Identifier. The table below shows the definition for the Client identifier for different sources and their description.

Value	Source	Description
0	Disable	The DHCP Option 61 is disabled. This is the default value if the attribute is not implemented in the application.
1	MACID	The MACID will be used as the Client Identifier
2	Host Name	The configured Host Name will be used as the Client Identifier
3	Generic String	Attribute #23 will be used as the Client Identifier

Attribute #23 (DHCP Option 61 generic string) is used to set the Client Identifier when Attribute #22 has been set to 3 (Generic String). Attribute #23 contains the Type field and Client Identifier and shall comply with the definitions in RFC 2132. The allowed max length that can be passed to the module via attribute #23 is 64 octets.

Example:

If Attribute #22 has been set to 3 (Generic String) and Attribute #23 contains 0x01, 0x00, 0x30, 0x11, 0x33, 0x44, 0x55, the Client Identifier will be represented as an Ethernet Media Type with MACID 00:30:11:33:44:55.

Example 2:

If Attribute #22 has been set to 2 (Host Name) Attribute #23 will be ignored and the Client Identifier will be the same as the configured Host Name.



## 7 Transparent Ethernet

### 7.1 General Information

Transparent Ethernet offers the possibility for a host application, that includes an IT implementation, to let the Anybus CompactCom handle an industrial Ethernet protocol (in this case EtherCAT), without the need for extra Ethernet ports.

Ethernet communication that is not related to EtherCAT is internally routed via the RMII interface to the Ethernet port and the TCP/IP stack of the host application. The IP configurations and the MAC addresses of the host application and the Anybus CompactCom must be the same.

The RMII interface is accessed through the host application connector. Please note that the 16 bit parallel interface is not available when transparent Ethernet is enabled. See the *Anybus CompactCom M40 Hardware Design Guide* for more information.

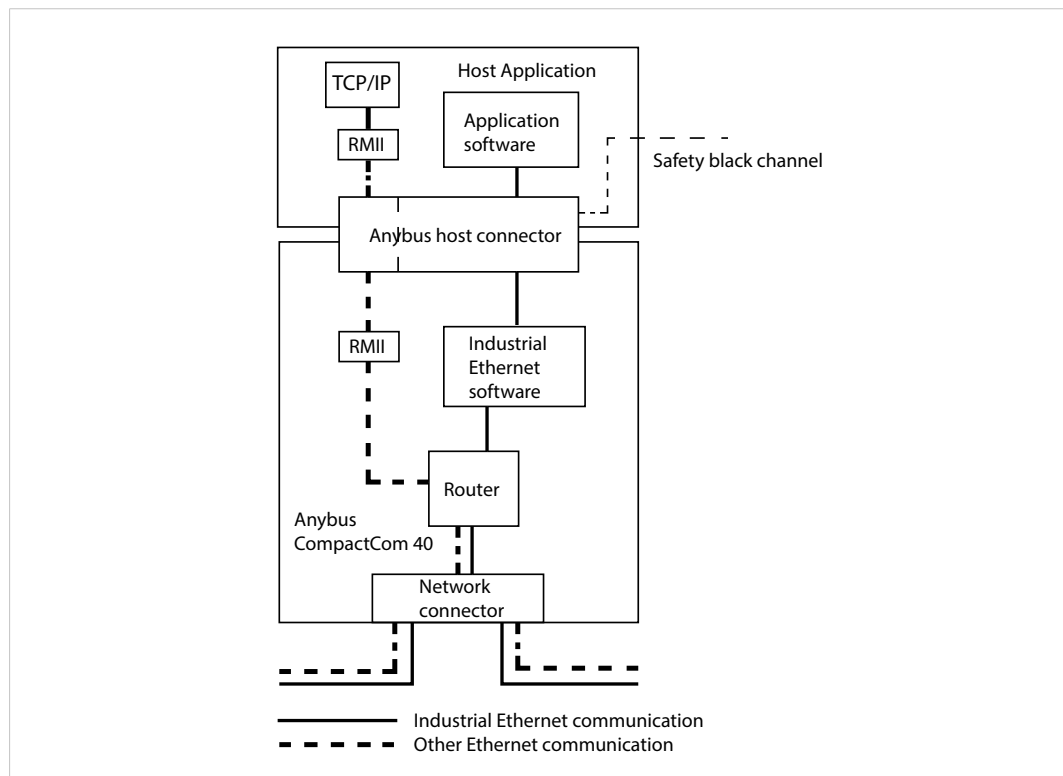


Fig. 2



Transparent Ethernet has to be enabled by setting instance attribute #16 in the Anybus Object (01h) during setup, see below.

16-bit parallel mode cannot be used when using transparent Ethernet.

MAC addresses and IP configurations have to be synchronized, see below.

Some EtherTypes, TCP/UDP ports and multicast MAC addresses may be reserved for use by the industrial Ethernet network. These must not be used for transparent Ethernet communication. See below for more information.

The Transparent Ethernet interface only supports 100 Mbit, full duplex operation.

### 7.2 Enabling Transparent Ethernet

Transparent Ethernet is not enabled at delivery. Attribute #16 (instance #1) in the Anybus Object (01h) has to be set to 0002h during setup. If this attribute is not changed, the Anybus

CompactCom 40 EtherCAT will start up with full IT functionality instead of transparent Ethernet functionality. Transparent Ethernet cannot be enabled after setup is finished. Once Transparent Ethernet is enabled, no IT functionality is enabled.

### 7.3 MAC Address Synchronization

The host application and the Anybus CompactCom must use the same MAC address when communicating on Ethernet. The host application must make sure that this is the case. This can be accomplished in either of the two ways described below:

- The pre-programmed MAC address in attribute #1 (instance #1) in the Network Ethernet Object (0Ch) is read and used by the host application when communicating on Ethernet.
- The Ethernet Host Object (F9h, instance #1, attribute #1) is implemented in the application, set with a MAC address provided and used by the application. At initialization, the Anybus CompactCom will read and then use the application provided MAC address from this object.

### 7.4 IP Configuration Synchronization

The host application TCP/IP stack and the Anybus CompactCom 40 EtherCAT TCP/IP stack must use the same IP configuration when communicating on Ethernet. The Anybus CompactCom 40 EtherCAT will write its currently used IP configuration to instance attribute #16 in the Ethernet Host Object (F9h) whenever the configuration is assigned or changed. The host application must use this configuration. DNS server and domain names can be read from the Network Configuration Object (04h) after an IP configuration update.

### 7.5 Routing Restrictions

The internal router receives all frames from the network. The frames that are intended for the industrial Ethernet network internal software, are recognized and routed to the Anybus CompactCom. The remaining Ethernet frames will be routed to the host application. Some restrictions apply to the use of e.g. UDP and TCP ports, sometimes also depending on industrial Ethernet network. If the host application is intended only for use with EtherCAT, the restrictions for the other networks can be ignored.



The host application is responsible for taking the following restrictions into consideration. If they are not followed, the Ethernet communication will not work correctly.

### 7.5.1 EtherTypes

The Anybus CompactCom internally uses bit 12 and bit 13 (mask 3000h) in the EtherType. Thus the host application cannot implement protocols using EtherType, where these bits are used..

The following EtherTypes are used by PROFINET and should not be used by the host application:

- 8892h (PNIO)
- 88CCh (LLDP)
- 88E3h (MRP)

### 7.5.2 Multicast MAC Addresses

The host application must not transmit or receive any data from and to the following MAC addresses as they may be used by the industrial Ethernet network (PROFINET):

- 01-0E-CF-XX-XX-XX
- 01-80-C2-00-00-0E
- 01-00-5E-40-F8-00 ... 01-00-5E-40-FB-FF
- X3-XX-00-00-00-00

(X: any number 0-F)

### 7.5.3 UDP/TCP Ports

The following ports may be used by the Anybus CompactCom, and must not be used by the host application:

- UDP 67 & 68 (DHCP)
- UDP 161 (SNMP)
- UDP 3250 (HICP)

The following ports are reserved for use by PROFINET:

- UDP 34962 (PROFINET RT Unicast)
- UDP 34963 (PROFINET RT Multicast)
- UDP 34969 (PROFINET RPC Context Manager)
- UDP 53247 (PROFINET RPC Client/Server)

The following ports are reserved for use by EtherNet/IP:

- UDP 2222 (Implicit messaging)
- UDP & TCP 44818 (Explicit messaging)

The following port is reserved for use by Modbus TCP:

- TCP 502 (Modbus messaging)

## 8 Firmware Upgrade

The Anybus CompactCom 40 EtherCAT firmware can be updated either by running the Firmware Manager II tool (FMII), available at [www.anybus.com/support](http://www.anybus.com/support), or by downloading the firmware upgrade file directly to the host application file system. For any of these methods to work the following needs to be implemented and/or performed:

- HICP needs to be enabled (FMII only).
- An FTP server needs to be implemented in the host application.
- A directory named “firmware” in the host application FTP root.
- The file module.nfo in the “firmware” directory in the Anybus CompactCom file system has to be copied to the “firmware” directory in the host application file system (FMII only). The Anybus CompactCom file system is accessed via the Anybus File System Object (0Ah).

Once a firmware file has been downloaded, the host application must be able to:

- detect a new file in the “firmware” directory
- download this file to the “firmware” directory in the Anybus CompactCom (The Anybus CompactCom file system is accessed via the Anybus File System Object (0Ah).)

The firmware will be updated upon the next reset of the Anybus CompactCom 40 EtherCAT.

## **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.



*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 Status Register

The application status register is primarily used in SYNC applications. It is used in applications where the network in question supports the ability to indicate critical process data errors to the master. If this is supported, the Anybus CompactCom module will accept and handle the below listed status codes written by the application.

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

Value	Error	ALStatusCode: ALSTATUSCODE_XXX (#)	Comment
0000h	No error	-	Application can operate in PROCESS_ACTIVE
0001h	Not yet synchronized	NOSYNCEERROR (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 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 prevents the application from going to PROCESS_ACTIVE.
0005h	Synchronization loss	FATALSYNCEERROR (002Ch)	Application has lost the lock to the synchronization. If the module is in PROCESS_ACTIVE, it will change 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 PROCESS_ACTIVE, it will change to ERROR.
0007h	Output error	DCSYNCEIOERROR (0033h)	A problem in the application prevents it from acting on outputs. If the module is in PROCESS_ACTIVE, it will change to ERROR.
0008h	General synchronization error	SYNCEERROR (0x001A)	There is a general synchronization error. If in PROCESS_ACTIVE the module will go to ERROR state.

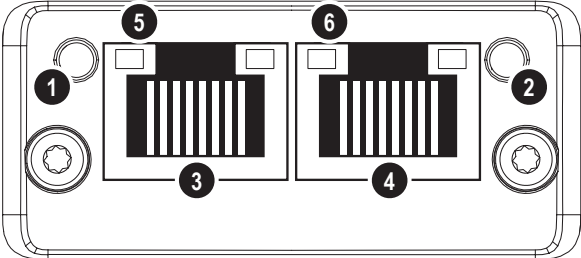
### B.4 Application Watchdog Timeout Handling

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

## C Technical Specification

### C.1 Front View

#### C.1.1 Front View (RJ45 Connectors)

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

The flash sequences for the RUN LED and the ERROR LED are defined in ETG1300\_S\_R\_V1i1i0\_IndicatorLabelingSpecification.pdf (ETG).

#### C.1.2 RUN LED

This LED reflects the status of the EtherCAT device.

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



### C.1.3 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	Sync Manager watchdog timeout	See <a href="#">Watchdog Functionality, p. 16</a> for more information.
Red	Application controller failure	Anybus module in EXCEPTION. If RUN and ERR turn red, this indicates a fatal event, forcing the bus interface to a physically passive state. Contact HMS technical support.
Flickering	Booting error detected	E.g. due to firmware download failure.

### C.1.4 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

### C.1.5 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 Functional Earth (FE) Requirements

In order to ensure proper EMC behavior, the module must be properly connected to functional earth via the FE pad / FE mechanism described in the general Anybus CompactCom M40 Hardware Design Guide.

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

## C.3 Power Supply

### C.3.1 Supply Voltage

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

### C.3.2 Power Consumption

The Anybus CompactCom 40 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 mA.



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

*In line with HMS policy of continuous product development, we reserve the right to change the exact power requirements of this product without prior notification. Note however that in any case, the Anybus CompactCom 40 EtherCAT will remain as a Class B module.*

---

## C.4 Environmental Specification

Consult the Anybus CompactCom Hardware M40 Design Guide for further information.

## C.5 EMC Compliance

Consult the Anybus CompactCom Hardware M40 Design Guide for further information.

## D Timing & Performance

### D.1 General Information

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

The following timing aspects are measured:

Category	Parameters	Page
Startup Delay	T1, T2	<a href="#">89</a>
NW_INIT Handling	T100	<a href="#">89</a>
Event Based WrMsg Busy Time	T103	<a href="#">89</a>
Event Based Process Data Delay	T101, T102	<a href="#">90</a>

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. (If the network specific maximum is less than the value given here, the network specific value will be used.)		32		
Event based application message response time		> 1 ms		
Ping-pong application response time		> 10 ms		
No. of simultaneously outstanding Anybus commands that the application can handle		1		

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

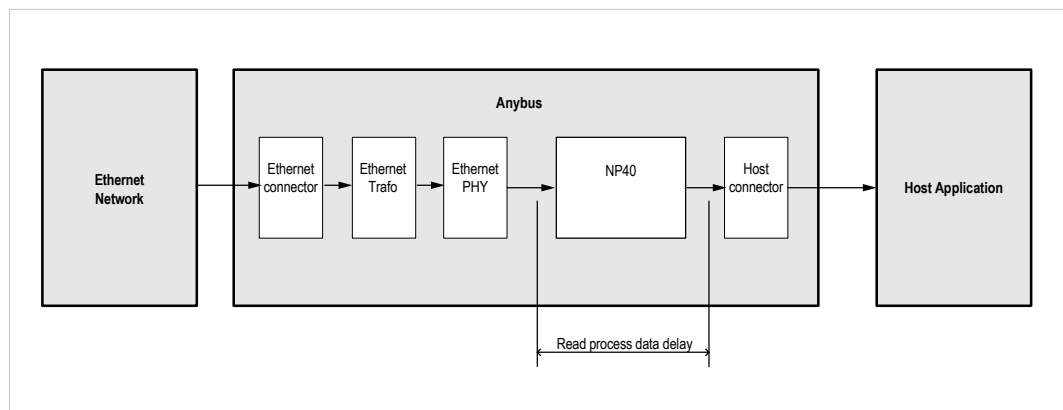


Fig. 3

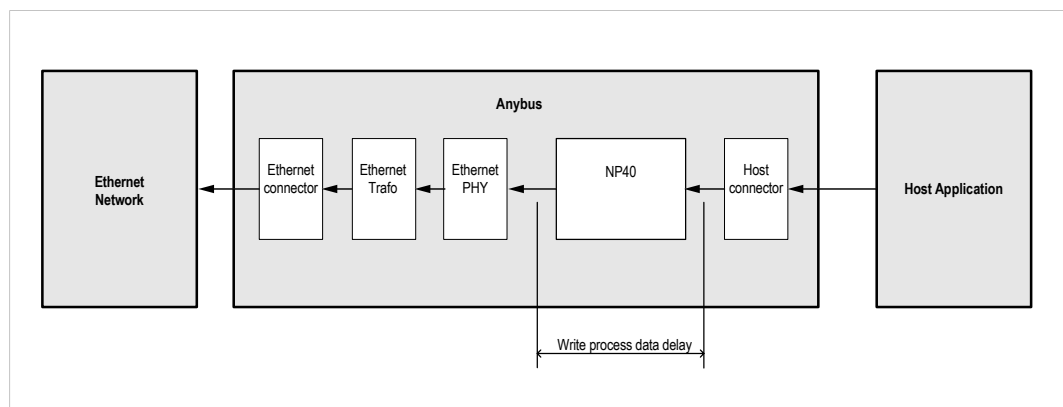


Fig. 4

## E Secure HICP (Secure Host IP Configuration Protocol)

### E.1 General

The Anybus CompactCom 40 EtherCAT supports the Secure HICP protocol used by the Anybus IPconfig utility for changing settings, e.g. IP address, Subnet mask, and enable/disable DHCP. The protocol offers secure authentication. Anybus IPconfig can be downloaded free of charge from the HMS website, [www.anybus.com](http://www.anybus.com). This utility may be used to access the network settings of any Anybus product connected to the network via UDP port 3250.

### E.2 Operation

When the application is started, the network is automatically scanned for Anybus products. The network can be rescanned at any time by clicking **Scan**.

To alter the network settings of a module, click on its entry in the list. The settings for the module will appear to the right.

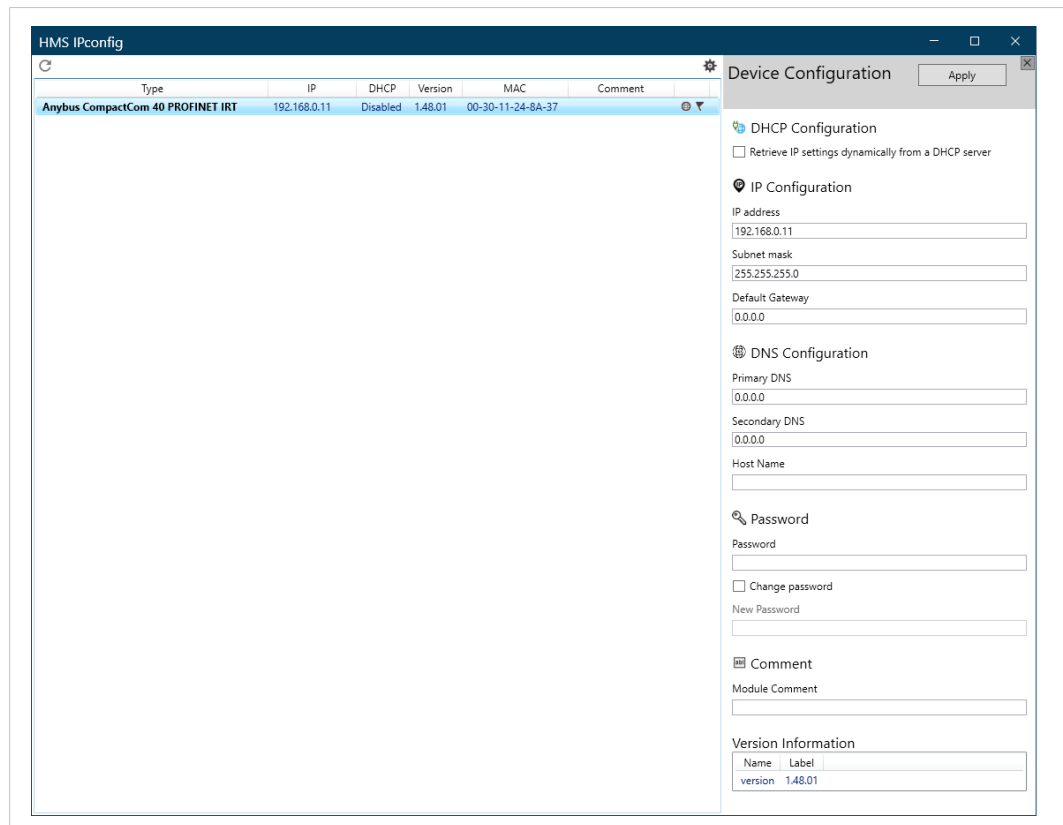


Fig. 5

Click **Apply** to send and apply the settings. Settings are saved in non-volatile memory in the device. Optionally, the configuration can be protected from unauthorized access by a password.

## F Backward Compatibility

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

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

### F.1 Initial Considerations

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

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



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

---

Link to support page: [www.anybus.com/support](http://www.anybus.com/support).

### F.2 Hardware Compatibility

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

#### F.2.1 Module

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



**Fig. 6**      **Anybus CompactCom M30/M40**

## F.2.2      **Chip**

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



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

### F.2.3 Brick

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

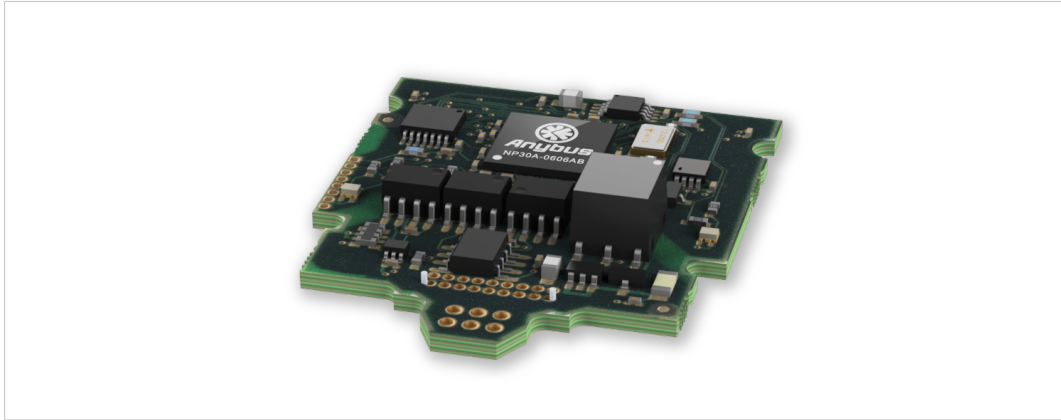


Fig. 7 Anybus CompactCom B30

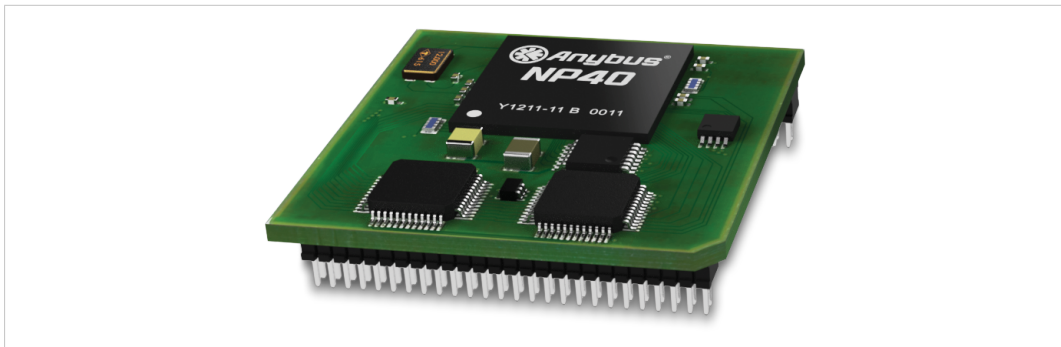


Fig. 8 Anybus CompactCom B40-1 (not for migration)

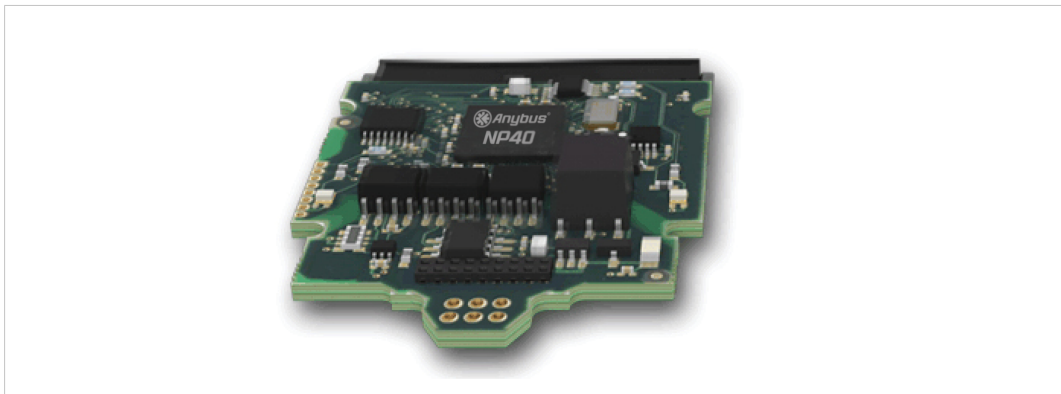


Fig. 9 Anybus CompactCom B40-2





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

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

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

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

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

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

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

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

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

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

This change should not affect your product.

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

**Address Pins A[11..13]**

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

**Max Input Signal Level ( $V_{IH}$ )**

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

**RMII Compatibility**

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

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

## F.3 General Software

### F.3.1 Extended Memory Areas

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



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

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

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

### F.3.2 Faster Ping-Pong Protocol

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

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

### F.3.3 Requests from Anybus CompactCom to Host Application During Startup

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

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

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

### F.3.4 Anybus Object (01h)

Attribute	30-series	40-series	Change/Action/Comment
#1, Module Type	0401h	0403h	Make sure the host application accepts the new module type value for the 40-series.
#15, Auxiliary Bit	Available	Removed	It is not possible to turn off the “Changed Data Indication” in the 40-series. Also see “Control Register CTRL_AUX-bit” and “Status Register STAT_AUX-bit” below.
#16, GPIO Configuration	Default: General input and output pins	Default: LED3 and LED4 outputs	See also .. <ul style="list-style-type: none"> <li><a href="#">GIP[0..1]/LED3[A..B], p. 96</a></li> <li><a href="#">GOP[0..1]/LED4[A..B], p. 96</a></li> </ul>

### F.3.5 Control Register CTRL\_AUX-bit

<b>30-series</b>	The CTRL_AUX bit in the control register indicates to the Anybus CompactCom if the process data in the current telegram has changed compared to the previous one.
<b>40-series</b>	The value of the CTRL_AUX bit is always ignored. Process data is always accepted.

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

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

### F.3.6 Status Register STAT\_AUX-bit

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

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

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

### F.3.7 Control Register CTRL\_R-bit

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

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

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

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

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

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

## F.4 Network Specific — EtherCAT

### F.4.1 Network Configuration Object (04h)

The instance number for the Device ID instance has changed from number 3 (30-series) to number 1 (40-series).

### F.4.2 EtherCAT Object (F5h)

Attribute	30-series	40-series	Change/Action/Comment
#2, Product Code	Default: 0000 0034h	Default: 0000 0036h	If the attribute is implemented in the host application, it overrides the default value and there is no difference between the 30-series and the 40-series. If the attribute is not implemented, the default value is used.
#6, Manufacturer Device Name	Default: "Anybus-CC EtherCAT"	Default: "CompactCom 40 EtherCAT"	If the attribute is implemented in the host application, it overrides the default value and there is no difference between the 30-series and the 40-series. If the attribute is not implemented, the default value is used.

### F.4.3 ESI-file (Configuration file used by engineering tool)

When migrating from the 30-series to the 40-series, a new, updated ESI-file is needed. To help you, there is an ESI-file Generator available from HMS Industrial Networks, see below.

#### ESI-file Generator

An ESI-file generator is available on the HMS Industrial Networks website. The generator will create an up to date ESI file fitted for the specific design. The ESI generator works for both the 30-series and the 40-series.

The generator can be downloaded from [www.anybus.com/starterkit40](http://www.anybus.com/starterkit40).

## Keywords

The ESI-file generator is up to date with the following differences between the 30-series and the 40-series.

The Product Code, Revision Number and Product Name must be updated to reflect the current module. Note: These values can be changed via the EtherCAT object (F5h) and the ESI-file values must match the EtherCAT object values.

```
<Type ProductCode="#x00000036" RevisionNo="#x00020001">
  CompactCom 40 EtherCAT</Type>
```

The EtherCAT state transition timeouts must be present in the ESI-file per the latest specification. Note: These timeout values can be change via the EtherCAT object (F5h) and the ESI-file values must match the EtherCAT object values.

```
<StateMachine>
  <Timeout>
    <PreopTimeout>1000</PreopTimeout>
    <SafeopOpTimeout>5000</SafeopOpTimeout>
    <BackToInitTimeout>1000</BackToInitTimeout>
    <BackToSafeopTimeout>200</BackToSafeopTimeout>
  </Timeout>
</StateMachine>
```

The sync manager start addresses have been changed in the 40-series, and the sync manager sizes are now configurable in the EtherCAT configuration tool.

```
<Sm MinSize="34" MaxSize="1486" DefaultSize="276" StartAddress="#x4000"
ControlByte="#x26" Enable="1">MBoxOut</Sm>
<Sm MinSize="34" MaxSize="1486" DefaultSize="276" StartAddress="#x4800"
ControlByte="#x22" Enable="1">MBoxIn</Sm>
<Sm StartAddress="#x2800" ControlByte="#x20" Enable="1">Inputs</Sm>
```

The 40-series supports File over EtherCAT (FoE) and this must be reflected in the ESI-file. If FoE is disabled in the EtherCAT host object, this keyword must be removed from the ESI-file.

```
<FoE/>
```

Since the 40-series is using the HMS slave controller, the EEPROM byte size and the SII configuration data must be changed according to the following settings.

```
<ByteSize>384</ByteSize>
<ConfigData>80360046F4010000000000000000</ConfigData>
```

The 40-series supports the boot strap state, and requires the following keyword.

```
<BootStrap>0040000400480004</BootStrap>
```

## **G Copyright Notices**

Print formatting routines

Copyright (C) 2002 Michael Ringgaard. All rights reserved.

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

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. 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.
3. Neither the name of the project 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.



Copyright (c) 2002 Florian Schulze.  
All rights reserved.

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

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. 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.
3. Neither the name of the authors nor the names of the contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHORS 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 AUTHORS 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.

ftpd.c - This file is part of the FTP daemon for lwIP

FatFs - FAT file system module R0.09b (C)ChaN, 2013

FatFs module is a generic FAT file system module for small embedded systems. This is a free software that opened for education, research and commercial developments under license policy of following terms.

Copyright (C) 2013, ChaN, all right reserved.

The FatFs module is a free software and there is NO WARRANTY. No restriction on use. You can use, modify and redistribute it for personal, non-profit or commercial products UNDER YOUR RESPONSIBILITY. Redistributions of source code must retain the above copyright notice.

lwIP is licenced under the BSD licence:

Copyright (c) 2001-2004 Swedish Institute of Computer Science.  
All rights reserved.

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

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. 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.
3. The name of the author may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``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 AUTHOR 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.

Copyright 2013 jQuery Foundation and other contributors  
<http://jquery.com/>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

rsvp.js

Copyright (c) 2013 Yehuda Katz, Tom Dale, and contributors

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

libb (big.js)

The MIT Expat Licence.

Copyright (c) 2012 Michael McLaughlin

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the 'Software'), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

The "inih" library is distributed under the New BSD license:

Copyright (c) 2009, Ben Hoyt  
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 Ben Hoyt 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 BEN HOYT "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 BEN HOYT 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.

## MD5 routines

Copyright (C) 1999, 2000, 2002 Aladdin Enterprises.  
All rights reserved.

This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software. Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.

L. Peter Deutsch  
ghost@aladdin.com



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.

