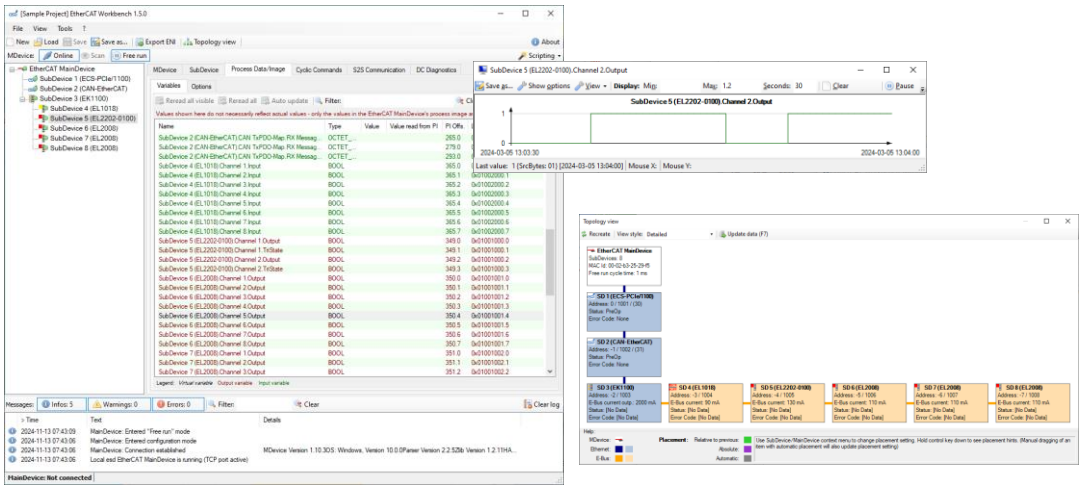
EtherCAT Workbench

EtherCAT Network Configuration Tool



Software Manual

to Product P.4510.01
(Demo Version: P.4512.01)

esd electronics gmbh
Vahrenwalder Str. 207 • 30165 Hannover • Germany
http://www.esd.eu
Phone: +49 (0) 511 3 72 98-0 • Fax: +49 (0) 511 3 72 98-68

Notes

The information in this document has been carefully checked and is believed to be entirely reliable. esd electronics makes no warranty of any kind with regard to the material in this document and assumes no responsibility for any errors that may appear in this document. In particular descriptions and technical data specified in this document may not be constituted to be guaranteed product features in any legal sense.

esd electronics reserves the right to make changes without notice to this, or any of its products, to improve reliability, performance or design.

esd electronics gmbh
Vahrenwalder Str. 207
30165 Hannover
Germany

Tel.: +49-511-37298-0
Fax: +49-511-37298-68
E-Mail: info@esd.eu
Internet: www.esd.eu

This manual contains important information and instructions on safe and efficient handling of the EtherCAT Workbench. Carefully read this manual before commencing any work and follow the instructions.
This manual is an integral part of the product. Please retain it for future reference.

Links

esd electronics gmbh assumes no liability or guarantee for the content of Internet pages to which this document refers directly or indirectly. Visitors follow links to websites at their own risk and use them in accordance with the applicable terms of use of the respective websites.

Trademark Notices

CANopen® and CiA® are registered EU trademarks of CAN in Automation e.V.
EtherCAT® is registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.
Windows® is a registered trademark of Microsoft Corporation in the United States and other countries.

All other trademarks, product names, company names or company logos used in this manual are reserved by their respective owners.

Document Information

Document file:	EtherCAT_Workbench_Manual_en_20.docx
Date of print:	2025-11-26
Document number:	QM.1012.P.4510.01

Software version:	1.5.0

Document History

The changes in the document listed below affect changes in the hardware as well as changes in the description of the facts, only.

Rev.	Chapter	Changes versus previous version	Date
1.9	all	English manual created from previous version 1.8 All chapters revised and new features described.	2024-12-06
2.0	4.4.2.1	Table supplemented	2025-11-26

Technical details are subject to change without further notice.

Classification of Notes

This manual contains noticeable descriptions for a safe use of the CANreal and important or useful information.

NOTICE

Notice statements are used to notify people on hazards that could result in things other than personal injury, like property damage.

NOTICE
This NOTICE statement contains the general mandatory sign and gives information that must be heeded and complied with for a safe use.

INFORMATION

INFORMATION
Notes to point out something important or useful.

Qualified Personnel

This documentation is directed exclusively towards personnel qualified in control and automation engineering.

Intended Use

The intended use of the EtherCAT Workbench is the operation as EtherCAT network configuration and monitoring tool.
The guarantee given by esd does not cover damages which result from improper use, usage not in accordance with regulations or disregard of safety instructions and warnings.

Typographical Conventions

Throughout this manual the following typographical conventions are used to distinguish technical terms.

Convention	Example
File and path names	**`/dev/null`** or **`<stdio.h>`**
Function names	***open()***
Programming constants	`NULL`
Programming data types	`uint32_t`
Variable names	*Count*

Number Representation

All numbers in this document are base 10 unless designated otherwise. Hexadecimal numbers have a prefix of 0x, and binary numbers have a prefix of 0b. For example, 42 is represented as 0x2A in hexadecimal and 0b101010 in binary.

Table of Contents

List of Tables

List of Figures

1 Overview

1.1 Abbreviations and Terms

Abbr.	Term	Description/Comment
APRD	Auto-Increment Physical Read	EtherCAT command to read value from a single SubDevice address by its auto-increment address
BRD	Broadcast Read	EtherCAT command to read value from all SubDevices
CoE	CAN application protocol over EtherCAT	(Former: “CANopen over EtherCAT”)
DC	Distributed Clock	
EBUS		Backplane bus for EtherCAT modules
EEPROM	Electrically Erasable Programmable Read-Only Memory	
ENI	EtherCAT Network Information	Contains configuration information for an EtherCAT MDevice
EoE	Ethernet over EtherCAT	
ESC	EtherCAT SubDevice Controller	
ESI	EtherCAT SubDevice Information	Contains information about EtherCAT SubDevice devices
ETG	EtherCAT Technology Group	Homepage: www.ethercat.org
FMMU	Field bus Memory Management Unit	
GUI	Graphical User Interface	
HDD	Hard Disk Drive	
HexBin	Hexadecimal Binary	Used to display data in hexadecimal notation, e.g. decimal 43707 = 0xAABB = HexBin: “BB AA” (little-endian)
Init	EtherCAT device state “Init”	
IP	“Init” → “PreOp”	State transition from “Init” to “PreOp”
LCID	Locale Identifier	A number that describes a language/culture setting
MAC	Media Access Control	
MBox	Mailbox	EtherCAT SubDevice Mailbox
NIC	Network Interface Card	Also used synonymic to “Network Interface”
NOP	No Operation	EtherCAT command that is ignored by the SubDevices
Op	EtherCAT device state “Operational”	
OS	Operating System	
PDI	Process Data Interface	
PDO	Process Data Object	
PDU	Protocol Data Unit	
PreOp	EtherCAT device state “Pre-Operational”	
SafeOp	EtherCAT device state “Safe-Operational”	
SM	SyncMan, Synchronization Manager	
SoE	Servo drive profile over EtherCAT	
WKC	Working Counter	Data field within an EtherCAT PDU used for error detection, see ETG.1000 documents for details

1.2 Inclusive Language

The term ***MainDevice*** (abbreviated ***MDevice***) replaces the term "Master" and ***SubordinateDevice*** (abbreviated ***SubDevice***) replaces the term "Slave". Despite that, API calls, data structures, macros, etc. which names contain the “old” terms are not renamed, because this would have the technical effect breaking (backward) compatibility. The latter is in no way intended to undermine the replacement of these terms or offend anyone. Some pictures in this documentation still contain the obsolete terms and will be replaced gradually in the future.

1.3 Introduction

The esd EtherCAT Workbench consists of an EtherCAT MDevice and the EtherCAT Workbench application which controls and configures the MDevice. This document describes the EtherCAT Workbench application.
The data exchange between the EtherCAT Workbench and the MDevice instance is based on local inter-process communication (IPC) mechanisms (Figure 1) or TCP/IP based network communication (Figure 2). This flexibility allows to attach the EtherCAT Workbench to esd MDevice instances on remote (embedded) devices. The EtherCAT Workbench for Windows comes with an EtherCAT MDevice instance which is installed on the local PC as a Windows service so the PC can be connected directly to an EtherCAT SubDevice segment for network configuration.

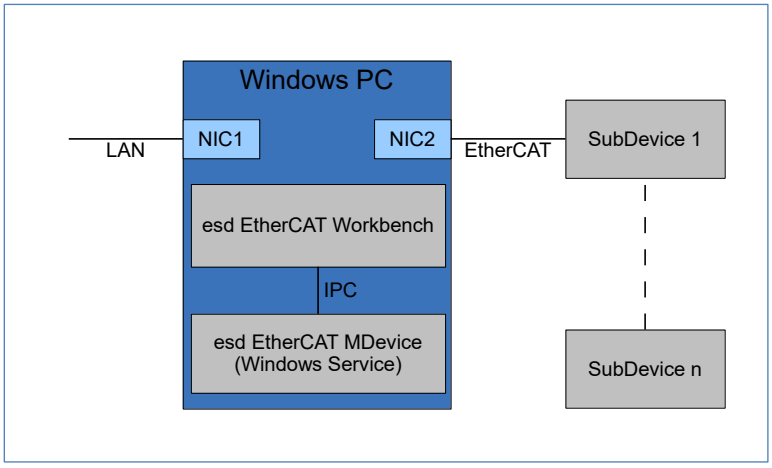


Figure 1: EtherCAT Workbench with local MDevice

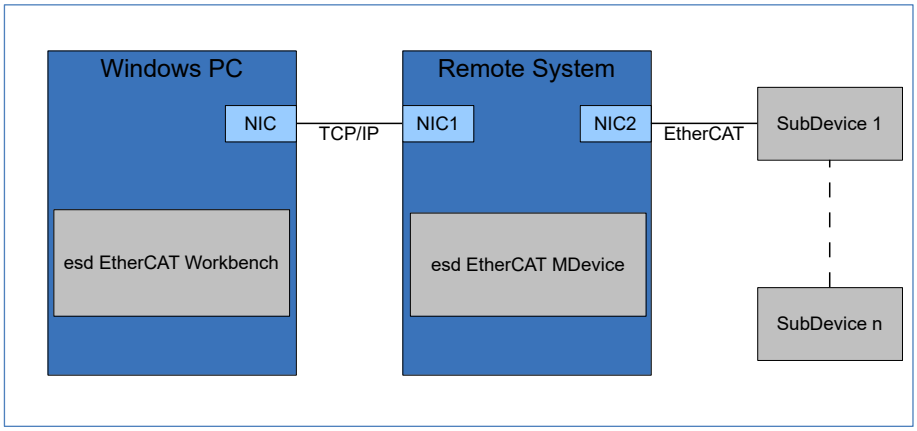


Figure 2: EtherCAT Workbench with remote MDevice

1.4 Features

- Offline configuration based on device vendor ESI files
  - SubDevice configuration
    - Initialization commands
    - Mailbox/FMMU set up
    - EoE / CoE set up
    - PDO mapping
    - Distributed clocks set up
    - Access to ESI EEPROM (read/write/create/edit etc.)
    - Export of SubDevice Configuration Information (SCI) files for preconfigured device configuration
  - Network topology overview
  - Export of ENI file according to ETG.2100 specification (SubDevice configuration, process data layout, cyclic commands, etc.) which can be processed by most of the EtherCAT MDevices on the market.
- Online configuration with bus scan based on device vendor ESI files and/or device ESI EEPROM data
  - Free run mode
    - Based on current configuration.
  - MDevice configuration
    - Network adapter(s) to use, cycle time, etc.
  - SubDevice / network diagnostics
    - SubDevice state, register data, ESI EEPROM data, CoE objects, FoE, basic AoE and VoE commands, etc.
  - Process data visualization
- Scriptable GUI, see chapter 4

- **EtherCAT Workbench Demo Version restrictions**
  - Usage limited to 30 days from installation date
  - Exported ENI files are limited to 3 EtherCAT SubDevices

1.5 Requirements

- esd EtherCAT MDevice (Win32 version for exclusive operation with the Workbench is part of the distribution)
- Microsoft Windows (Windows 10 or later, 32/64 bit) with .NET Framework 4
- Screen resolution at least 1024×768 pixel
- Memory as recommended for operating system
- Disk space approximately 20 MB

1.6 Installation

Installation is done by running the “setup.exe” (requires administrator privileges). Simply follow the on-screen instructions to proceed. Usually, all options can be left at their default values. (When the esd EtherCAT MDevice is or will be installed on another machine, its installation can be avoided on the *Select Components* page by deselecting the item *Local esd EtherCAT MDevice*)

1.7 Application Start

Start the application from the start menu → ESD → EtherCAT Workbench or double click the EtherCAT Workbench desktop icon, if installed.
For command line option see 4.6 Workbench Command Line Arguments.

It is allowed to start multiple application instances of the EtherCAT Workbench, but the following restrictions apply:

- Instances started from the same program folder share the same SubDevice device library (see 2.3.1). To add or remove files to/from the library (see 2.3.2.1) all instances must be closed.
- Only one instance at a time can connect to the local or the same remote MDevice service.

An explanatory dialog is shown when you start multiple instances of the EtherCAT Workbench:

INFORMATION
If you don't want to share the same device library across multiple instances or different Windows users there is a simple solution:
Copy and paste the complete installed program folder (usually C:\Program Files (x86) \esd\EtherCAT\EtherCAT Workbench) to an arbitrary different folder location and start the application `EtherCAT Workbench.exe` from there.

NOTICE
To use online configuration (see 3) for devices connected to the local computer the local MDevice service must have been installed and started before (e.g. automatically at system start). If the local service is installed but not running the EtherCAT Workbench can handle the start, but administrator privileges are required, and a Windows UAC (User Access Control) dialog is displayed.

1.7.1 Demo Unlocking

After installation the esd EtherCAT Workbench runs as a “Demo” version with some limitations. It must be unlocked by a license file to remove these limitations.
That license file will be bound to a specific PC[1] and created by esd after you purchased the esd EtherCAT Workbench. To request the license a special tool is needed, the *License Requester*.

It is installed with the EtherCAT Workbench and prepares an email to request the license for you, see section 1.7.1.2 for details.
You will receive the license file by email and must install it to the EtherCAT Workbench, see section 1.7.1.3 for details.

NOTICE
The software from esd and from third parties used in the EtherCAT Workbench is subject to the license terms of the respective authors or rights holders. EtherCAT Workbench may only be used in accordance with these license terms!

1.7.1.1 3rd Party Software Licenses

The license terms of 3rd parties (3rd Party Licenses) are contained in the document 3rd_party_licensor_notice.pdf. The document is installed on your system during installation and is linked via the Windows start menu. Select -> *ESD -> 3rd Party Licensor Notice*.

1 For other licensing variants please contact esd (licence@esd.eu).

1.7.1.2 License Requester

This tool shows your Hardware Id[2] and allows you to enter your company name. This information is required by esd to create the license file. (The selected company name will be visible in the “About window of the licensed EtherCAT Workbench”)

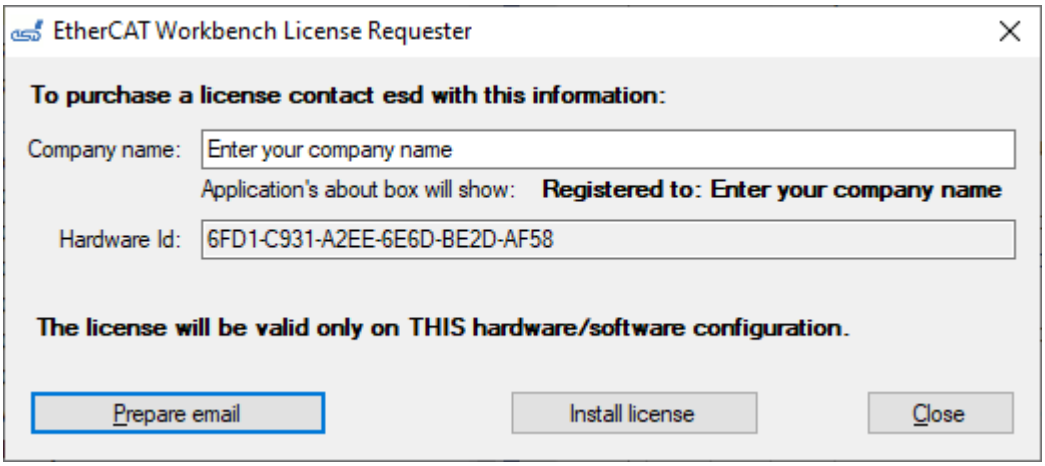


Figure 3: Licenser Requester sample

It can be started by the start menu entry “Programs” → “esd” → “EtherCAT Workbench” → “Request or install License” or within the Workbench by the *Purchase license* button in the upper right corner of the main window.
It also offers to prepare an email with this information. If that fails or is not wanted, just prepare your own email to licence@esd.eu and copy the *Hardware Id* and *Company name* texts manually.

1.7.1.3 License Installation

A purchased license file can be installed manually or by using the *License Requester* described above.
Manually: Copy the file `.license` to the folder where you installed the Workbench, e.g. to:
`C:\Program Files\esd\EtherCAT\EtherCAT Workbench\`

By License Requester (must be started with administrator privileges): Click the *Install license* button and select the file `.license` in the file selection dialog that is shown.
When the Workbench is running while the file is copied, it must be restarted to make use of the license file.

2 This is just a “checksum” about your CPU/HDD/OS, etc. – no personal information is included.

1.8 Application Overview

After starting the EtherCAT Workbench, its main form is shown (Figure 4).

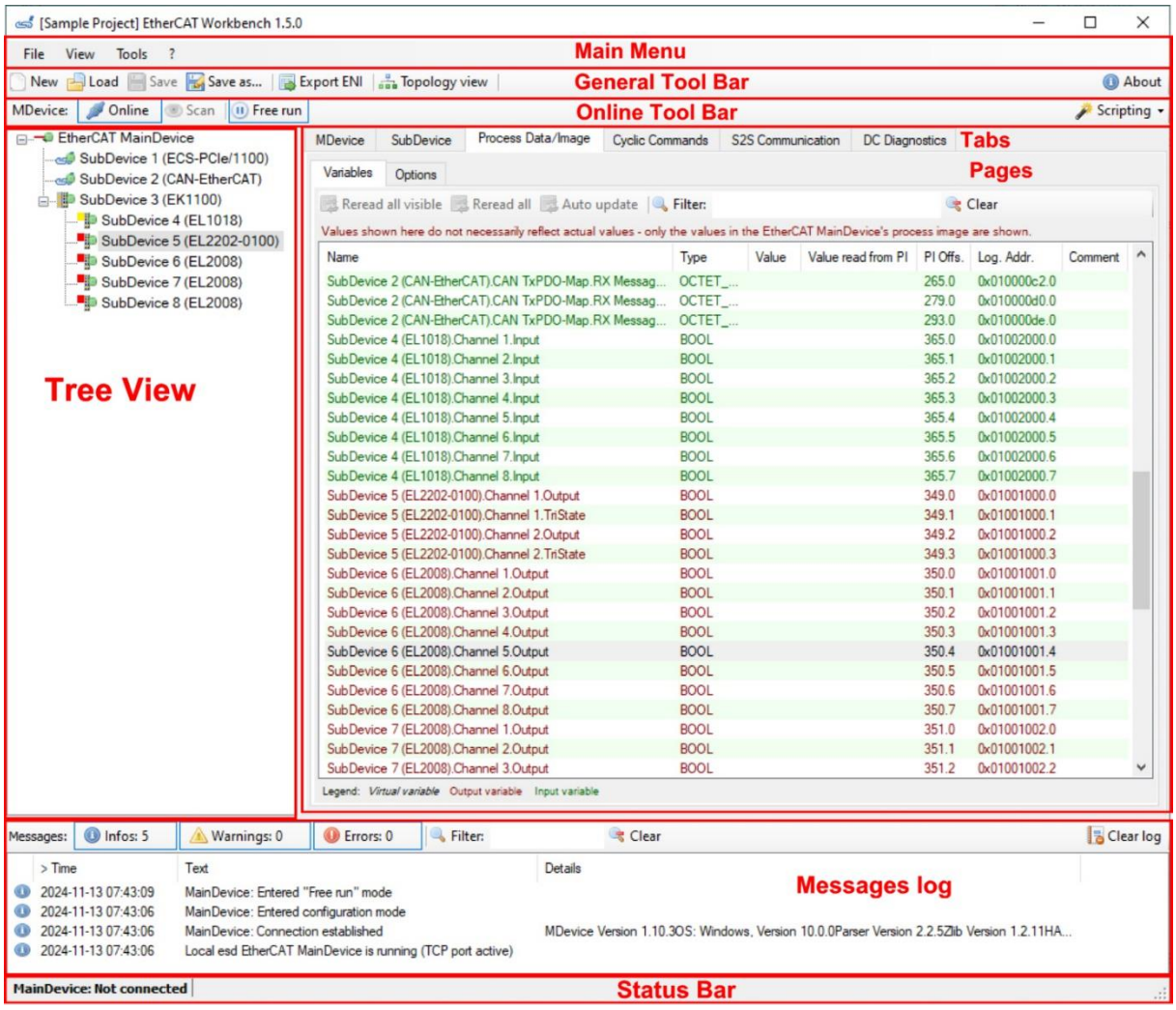


Figure 4: Main form, (Example with tab: *SubDevice*, page: *General*)

1.8.1 Main Menu

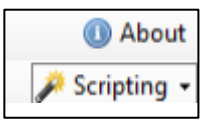
In the main menu bar, you can choose the menu items *File*, *View*, *Tools* and "?"

Main Menu Item	Description	See Section
File	Klick on this menu item if you want to • Manage projects (*New projects*, *Load project...*, *Recent projects,* • *Save project (as...)*, see project files. • Open *MDevice/SubDevice menu* (Depending on the node selected in the tree view). • *Export MDevice configuration file (ENI)* • *Exit EtherCAT Workbench*	1.8.3 1.8.4 1.8.3
View	• Show / hide text right of the toolbar buttons with *Toolbar buttons with captions.* • *Auto minimize variable monitors with main window* • *Minimize all variable monitor windows* (see Monitor)	3.4.3

Main Menu Item	Description	See Section
Tools	• *Copy ESI files to SubDevice library...* • *Copy folder with ESI files to SubDevice library…* • *Copy SCI file(s) to SubDevice library…* • *Open SubDevice library* • *Configure alias addresses for all SubDevices*: Opens the SubDevice *Alias editor,* see	2.3.14
	• *Stop / Restart local EtherCAT MDevice service* When the local esd EtherCAT MDevice needs to be stopped (e.g. because of a misconfigured ENI) it can be stopped and restarted with these items. Usually it is started/stopped automatically with the Workbench start/close. • *Reset all window settings* Can be used to reset some simple settings that are not project specific, such as window positions, table column widths, recent projects, favourite SubDevices, etc. • *Edit global settings,* Can be used to edit some global settings, i.e. settings that are used for all projects or affect the whole application. See for global settings	5.2
? (Help)	• *Manual* - Opens this PDF manual in the system default PDF application • *Product Web page* - Opens the English product website of the esd EtherCAT Workbench at esd.eu • Click *About* or the button *About* in the general toolbar to show the *About EtherCAT Workbench* Dialog with version and registration information. Use the button *Copy Info to* copy some support-relevant information as text to the clipboard.	

1.8.2 Toolbars

The upper general toolbar under the main menu bar controls general tasks, such as loading/saving project files. The second toolbar *MDevice:* is used for online tasks, such as connecting to the MDevice or scanning for SubDevices.

General Toolbar Buttons	Description	
New / *Load* / *Save* / *Save as..*	Start new project or load/save existing project file.	
Export ENI	Save ENI to a .xml file	See section 2.2
Topology view	Shows a graphical overview of the EtherCAT network.	See section 2.6
About	Show the *About EtherCAT Workbench* Dialog	

***Online* Toolbar**

Buttons	**Description**	
Online	Connect/Disconnect to/from the EtherCAT MDevice.	See section 3.1
Scan	Start a SubDevice scan (when online).	See section 3.2
Free run	Switch MDevice to free run mode (to read process data, etc.).	See section 3.3
Dropdown menu *Scripting*		See section 4.3

1.8.3 Project Files

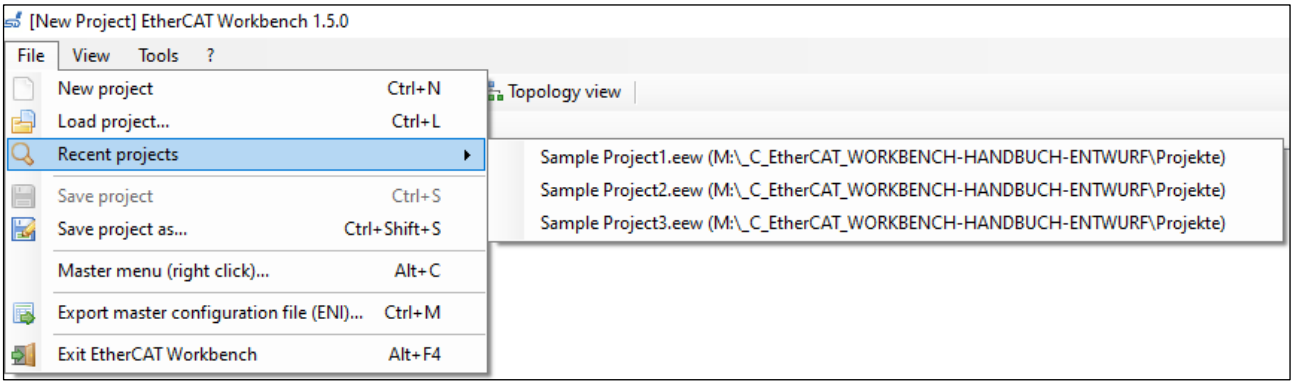


Figure 5: Project Files (Example)

All settings made in the tree view and the tabs, data entered or retrieved online can be saved in project files which have the extension ".eew".

On the very first start the EtherCAT Workbench begins a new empty project.
To start a new project when another project is open, select *New project* from the *File* menu.
Make your settings and changes to the project. The title bar shows the project name and when you have unsaved changes to the project, an asterisk appears:

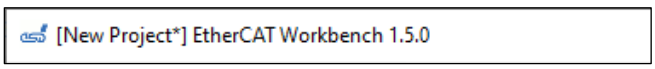


Figure 6: Title bar of the new project with asterisk

Save the project file with *Save project*.
When saving for the first time, you can assign a project name that can be changed later with *Save project as...*

Reload a previously saved project file with *Load project*.
To quickly access recently loaded projects, use the *Recent projects* submenus.
The number of recent projects can be configured in the *Global Settings* as *MaxRecentProjects*,the default is 10. See 5.2
The default in the *Global Settings* is to re-open the last project when the EtherCAT Workbench starts. Toggle the parameter *AutoLoadRecentProject* to “False” to change that behaviour.

1.8.4 SubDevice Tree View

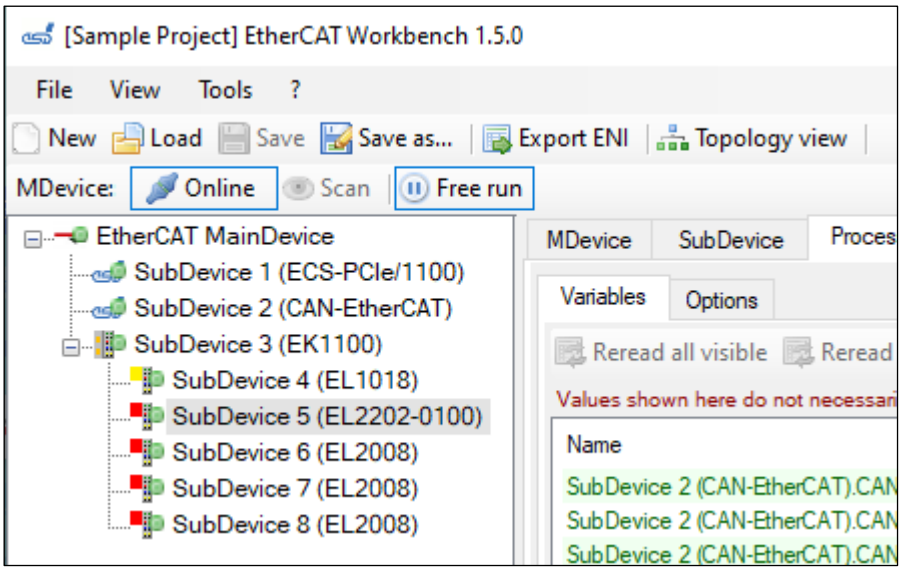


Figure 7: SubDevice tree view

This figure shows the devices in the EtherCAT network. The first device is always the EtherCAT MDevice, below the MDevice the EtherCAT SubDevices are displayed.
The order (top down) represents the physical SubDevice order, i.e. the way the EtherCAT frame will go.
This list is either populated manually See 2.3.1
or by running an online scan of an existing network. See 3.2

In online mode the SubDevice icons will additionally display information about the SubDevice states, too. Details about the used images can be obtained by the context menu entry *Show icon/color legend.* See 3 See 1.8.4.1
The context menu also allows to remove/add SubDevices

SubDevice without ESI can't be added manually. See 2.3.2.2

1.8.4.1 Tree View Context Menus

There are different context menus for MDevice and SubDevice nodes. The menu items displayed in these context menus can vary depending on the application.
To open the context menus of the tree view either:

- Right click the MDevice or a SubDevice node
- With an already selected MDevice or a SubDevice node:
  Select -> *MDevice menu* or -> *SubDevice menu* from the *File* menu,
  or press the shortcut key 'ALT+C'

Context menu items of the EtherCAT MDevice node:

Append new SubDevice (Ins...)

This item is available at empty projects only.

- Add the very first SubDevice to the project, you may also use the key *Ins*.
- This opens the device library for selecting a SubDevice to add. See 2.3.1

Insert new SubDevice before "SubDevice 1 (Name of the device)"

This item is available at projects with assigned SubDevices only.

- Insert a SubDevice node as new SubDevice node 1.

Delete all SubDevices

This item is available at projects with assigned SubDevices only.

- Remove all SubDevice nodes from the MDevice node.

Rename...

Rename the MDevice node or a previously added SubDevice node (See Renaming and Change revision). See 2.3.2

Request state change

Online connection to the MDevice only:

- Request an EtherCAT state change of the MDevice or a SubDevice. (Init, PreOp, SafeOp, Op)

Show icon/color legend

Shows the following dialog explaining the different overlay icons on the tree view items:

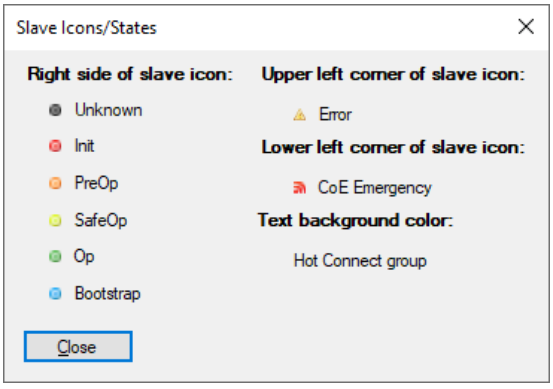


Figure 8: Example of SubDevice Icons/States

Context menu items of the EtherCAT SubDevice nodes:

Append/Insert new SubDevice at Port X (Ethernet)	Append or insert SubDevices at different ports of the selected SubDevices if the SubDevice has more than one outgoing port (EtherCAT socket) like E-Bus couplers or junctions.	
Delete SubDevice	Delete this SubDevice from the project.	
Change Revision...	Change the software revision of the SubDevice.	See 2.3.2
Export SCI...	Export an SCI file for the SubDevice including the SubDevice settings you have made.	See 2.3.16
Rename	Rename the SubDevice.	
Create / Delete / Edit hot connect group...	-	See 2.3.15
Request state change	Online connection to the MDevice only: • Request an EtherCAT state change of the MDevice or a SubDevice. (Init, PreOp, SafeOp, Op).	
Show icon/color legend	See SubDevice Icons/States	See Figure 8
see		
Go to / Find …	Shortcut navigation: Open the corresponding tab page and item in the SubDevice tab view.	

1.8.5 Tab Pages

Tab pages are used to categorize all the data and configuration pages.
The terms “page” or “tab” will be used to describe on what tab page an information is found. The screenshot in Figure 4 for example shows the tab *SubDevice* activated. This page itself has subpages, too: page *General* is selected.
This hierarchical order should help to find the desired information quickly – this manual tries to adopt this order, too.

1.8.6 Messages Log

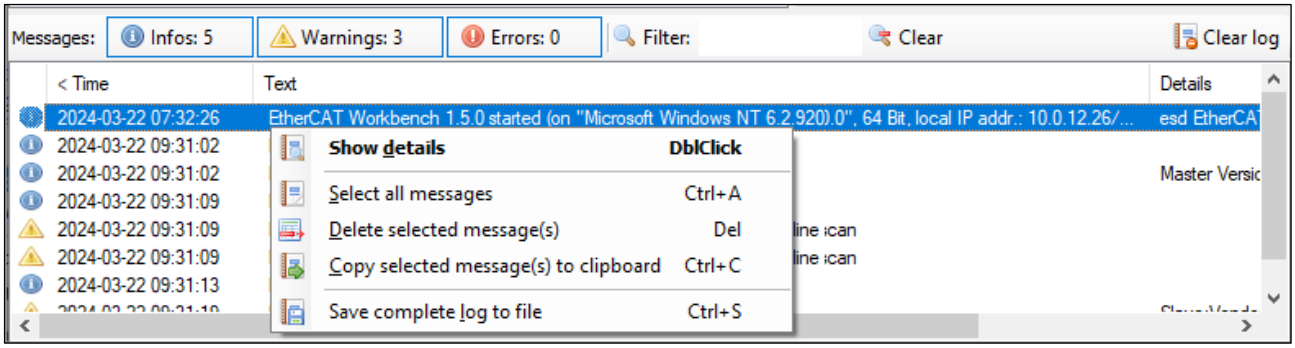


Figure 9: Messages log with toolbar and context menu

Almost any activity will give feedback about its success or failure. All these events are logged in the *Messages log*. Every event is of a severity level/type and also includes the date it occurred.

Info	e.g. basic information about what is happening etc. Usually not important.
Warning	e.g. information about events that should be observed.
Error	e.g. information about something that shouldn't have happened and that prevents normal operation.

To sort the list by a specific column its header can be clicked: a "<" or ">" character in front of the column header text then shows that the list is sorted by that column, ascending or descending.

Toolbar
Allows filtering and clearing the log: every message type has its button showing the number of messages of that type and whether these types should be visible or not. When a button is down/highlighted the corresponding type of message will be visible. In addition, the list of messages can be filtered according to a case insensitive *Filter* text.
The button *Clear* log will delete all messages, including filtered.

Context Menu
Allows some other operations which should be self-explanatory, e.g. removing items from the list or saving the list to a file.
Show details might show some more text/information for some events.

2 EtherCAT Network Configuration

The EtherCAT MDevice basically just needs an ENI file to initialize and control an EtherCAT network. This file contains the complete configuration information about the MDevice, all SubDevices, all frames that need to be sent cyclically or to initialize the SubDevices, etc. In most cases the automatically generated ENI file for a new online or offline configuration can be used without any further changes but the EtherCAT Workbench allows to fine tune nearly every aspect of this configuration.
Nearly everything that is configured in the EtherCAT Workbench application affects this ENI file. This chapter describes all these configuration options.
The ENI is automatically generated and only transferred to the MDevice when “Free run mode” (see 3.3) is entered. This means that most changes will become effective only then. (The online tasks are described in chapter 3)

2.1 MDevice Settings

MDevice settings include the data that affect the “MDevice” section of the ENI as well as the esd EtherCAT MDevice configuration.

2.1.1 General Settings

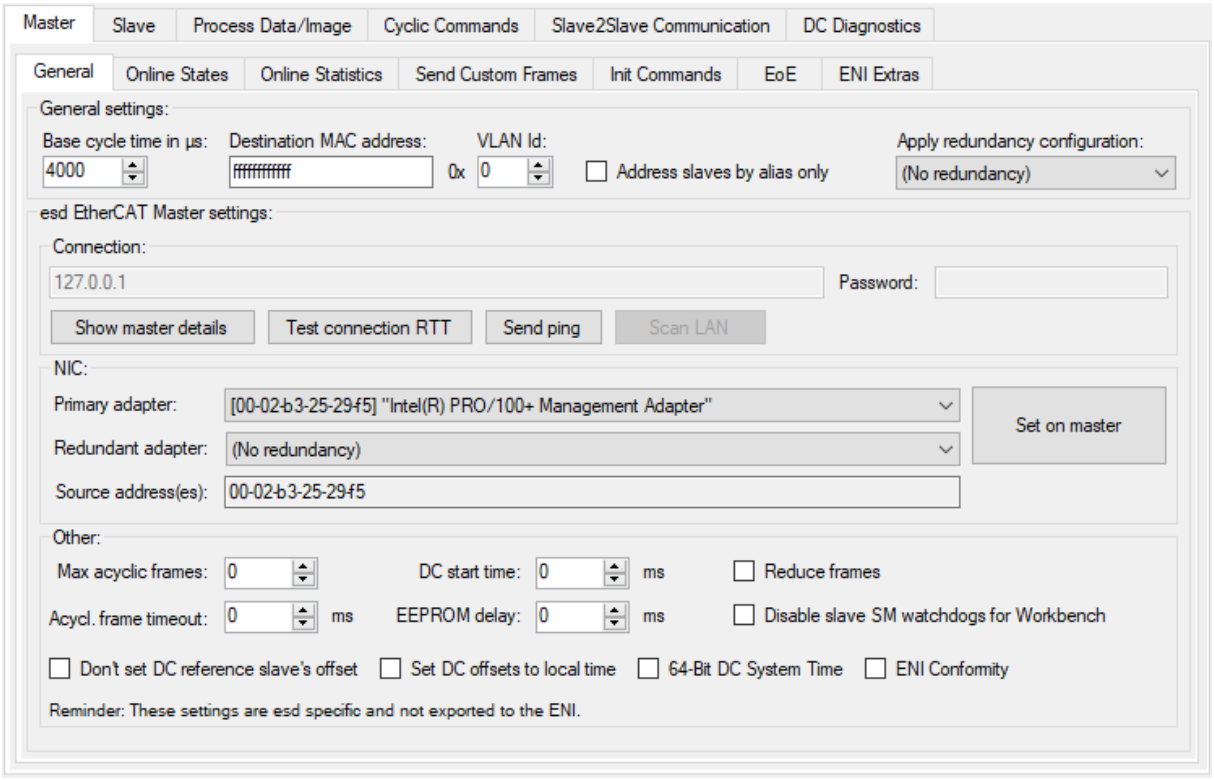


Figure 10: MDevice configuration, page *General*

General settings:

Base cycle time in µs: Used to determine the interval of the EtherCAT commands that handle the SubDevice process data, see also *Page Options*. Actual used value may differ, according to MDevice settings/capabilities. See 2.3.10:

Destination MAC address: The MDevice transmits all EtherCAT frames using this value as Ethernet destination address.
Default is the Ethernet broadcast address “FFFFFFFFFFFF”.
Some MDevices also support using e.g. an Ethernet multicast address.

VLAN Id:

If this value is greater than 0, a VLAN tag with the given Id will be added to all Ethernet frames sent by the MDevice. This value is not part of the ENI files specification ETG.2100 and can only be used in combination with esd MDevices which control a SubDevice segment connected to a VLAN capable switch.

Address SubDevices by alias only

(Checkbox) When this is checked, the initialization commands for all SubDevices are changed to use the alias address (ESC register 0x0012 / EEPROM ESI word 0x0004), to allow changing the order of the SubDevices without the need to create a new ENI.

Use *Configure alias addresses [...]* from the Workbench's *Tools* menu to edit these addresses and ensure each is unique.

INFORMATION
Using this feature requires **every** SubDevice to have a valid alias
- When a new alias is written to a EEPROM of a SubDevice, it usually needs to be rebooted to use that value
- Not all SubDevices support an alias address
- The Workbench still requires the correct order. (Never use the Workbench with a network that differs from the displayed one)

Apply redundancy configuration

This setting becomes important in connection with **EtherCAT cable redundancy.**
If you want to use cable redundancy, you must select either:

- *Optimized frame size* which is **best for usage with the esd MDevice** and results in shorter EtherCAT frames (more detailed: separate commands in frame for each SubDevice).
- *Broadest compatibility* is best for usage with non-esd EtherCAT MDevices but less optimized (more detailed: separate LWR/LRD commands instead of LRW).

Apply redundancy configuration is forced when you select a *Redundant adapter* within the dialog *NIC* group.

INFORMATION
It is in principle permissible to keep the cable redundancy selections without cable redundancy usage, but with the *(no redundancy)* selection frames are shorter and always compatible with non-esd MDevices.

esd EtherCAT MDevice settings

Connection:	Connection string is entered here.	See 3.1
Show MDevice details	Shows some technical details about the esd EtherCAT MDevice instance the EtherCAT Workbench is connected to.	
Test connection RTT	Sends a small data packet to the esd EtherCAT MDevice and measures the time until it is echoed back ("**R**ound **T**rip **T**ime"). The result is added to the Messages log.	
Send ping	Sends an ICMP ping request (5000 ms timeout) to the given host. Result will be shown in messages log.	
NIC:	Used to select the NIC(s) of the EtherCAT MDevice	See 3.1

Other:

Max acyclic frames	Maximum number of acyclic frames per cycle. 0 = no limit
DC start time	Delay for Distributed Clocks setup after changed to Op state.
Reduce frames	Configures the esd MDevice to reduce the total number of cyclic frames by combining as many cyclic EtherCAT commands as possible in each frame ignoring the frame layout defined by the ENI file. This increases the performance of implementations where the network layer has a large processing overhead per frame. Shouldn't be needed, as command layout can be configured by the Workbench itself, *Page Options.* See 3.4
Acycl. frame timeout	Time out value for the acyclic frames, see also esd EtherCAT MDevice manual.
EEPROM delay	Used when writing the EEPROM data of the SubDevice in Free run mode – for SubDevices that require more time until they acknowledge EEPROM write commands.

Disable SubDevice SM watchdogs for Workbench
Write the register 0x420 of each SubDevice to 0 when going to free run. (Affects only Workbench usage, not the exported ENI. Selecting "Set SM watchdog" on an init commands page of a SubDevice overrides this).
Note that some SubDevices might deny that or require additional changes that are not included here.

Don't set DC reference SubDevice's offset
0x0920 of the DC reference SubDevice shall be written to 0 only.

Set DC offsets to local time
The DC times shall start with local system time (no drift compensation).

64-Bit DC System Time
Spread DC Time as 64-Bit value.

ENI Conformity No esd Vendor-Specific element in ENI export (MDevice version > 1.11.3 required).

2.1.2 Online States

EtherCAT

Current state: Reflects the current network state of the EtherCAT MDevice.
(Also displayed as icon on the MDevice in the SubDevice tree view.)

Request state: Request the EtherCAT MDevice to change to a specific state.
Note that “SafeOp” and “Op” require the ENI, therefore these states are only possible in “Free run” mode. See 3.3

MDevice

SubDevices: Number of EtherCAT SubDevices:

- *No.:* Total number of SubDevices in ENI
- *Active*: Number of active SubDevices (determined by reading *AL state* registers in free run mode, see *Options*) See 2.4
- *MBox*: Number of SubDevices with Mailbox support
- *Primary*: Number of SubDevices at MDevice’s primary network interface
- *Redundant*: Number of SubDevices at MDevice’s redundant network interface

DC deviation: Distributed clock deviation between MDevice system time and SubDevice reference clock.

Cyclic frames: Number of frames that are sent cyclically, should match frames listed in *Cyclic Commands* tab page. See 2.4

State flags: State flags from the esd EtherCAT MDevice (see also its manual).

- Bold items are currently set flags.
- *Count* indicates how often the flag was set (reset by context menu or when going offline etc.)
- *Trigger log entry* indicates whether a log entry is created when the flag changes from unset to set.
- Double click the state flags line to toggle the *Trigger log entry* switch.

To *Reset all counters,* click on a free position in the list to open the contest menu and click on it.

Last update Date when all currently displayed information on this page was received from the MDevice.

2.1.3 Online Statistics

Shows a list of several Ethernet/EtherCAT statistics. Grouped by “Primary” and “Redundant” network interface.

MDevice	EtherCAT statistical data of the MDevice instance.
NIC	Ethernet statistical data of the NICs (e.g. *Primary NIC*, *Redundant NIC*).
Device	EtherCAT statistical data of the device instance (e.g. *Primary Device*).

The list is either updated by clicking the *Update* button in the toolbar, or by entering a value in the *Auto update interval:* field.

2.1.4 Send Custom Frames

This page allows sending EtherCAT frames with arbitrary commands. **Used for diagnostic purposes only.**
The context menu allows to add/edit the commands within the EtherCAT frame, Figure 11 shows a sample frame:

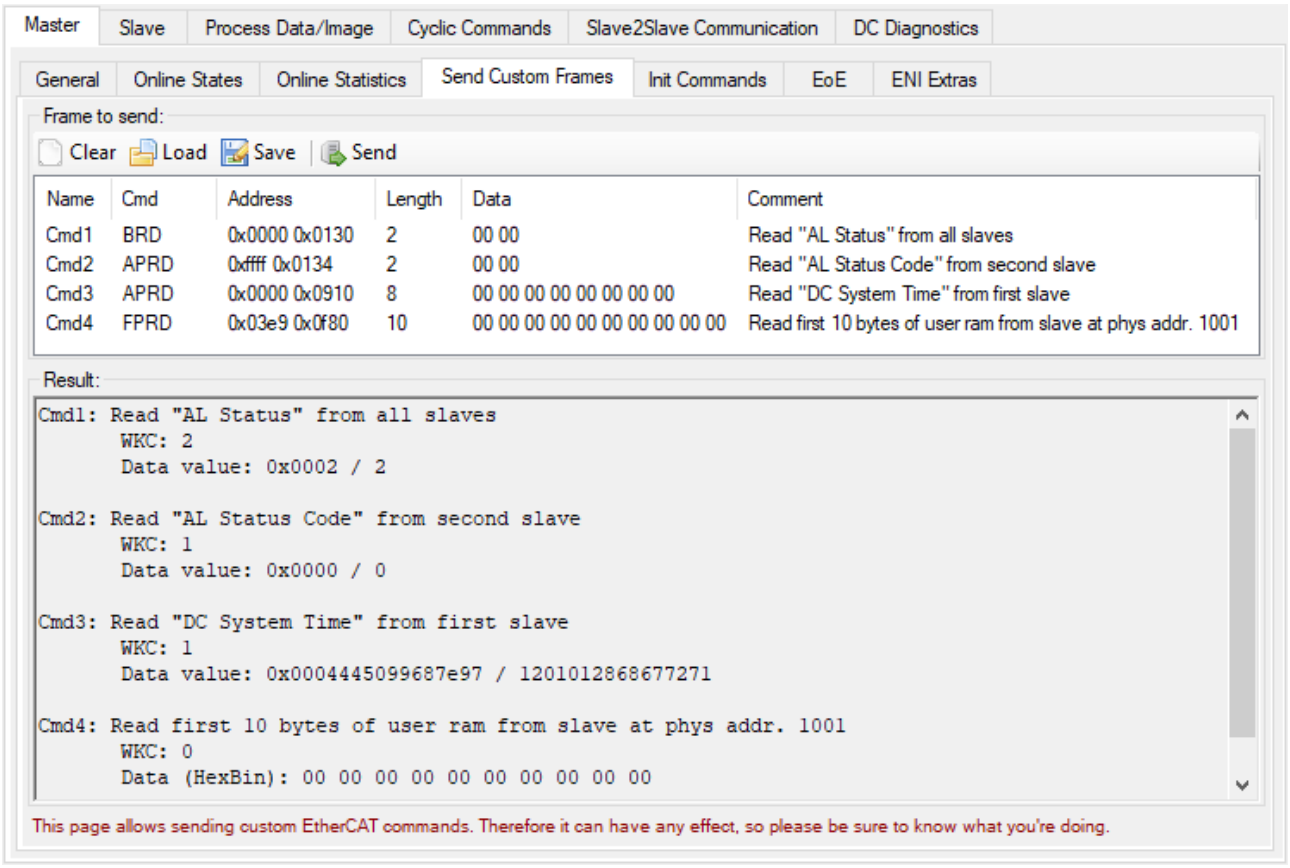


Figure 11: Example for *Send custom frame*

To open the context menu right click on the list or a list item.
The menu items of the context menu are self-explaining:

- *Edit command* - opens a command editor
- *Append command* - opens a command editor
- *Duplicate command*
- *Delete command*
- *Move up*
- *Move down*

2.1.5 Init Commands

Distinguish between different location of initialization commands:

- Configuration of EtherCAT commands for MDevice initialization — See below
- Configuration of EtherCAT commands for SubDevice initialization — See 2.3.8
- Configuration of Mailbox CoE (CAN over EtherCAT, per SDO) initialization commands. — See 2.3.9.1

This page allows configuration of the initialization commands which are sent to all SubDevices during network initialization.
Usually this should be set to *Default*, which means all commands are created automatically based on the corresponding settings.
Click the *View* button to open the *Init Commands (MDevice)* dialog, the list of configuration commands is shown with.
If *Custom* is chosen the currently active automatically created commands are used, but they will never change automatically again when a corresponding setting changes.
The *View* button changes to *View/Edit* then to allow a manual customization.

To *Insert a new item*, *Delete*, or *Move Up / down* an item, right click on an entry in the command list, to open the context menu.

2.1.6 EoE

Automatically enabled when a SubDevice supports the EoE protocol. These parameters configure the required virtual switch within the EtherCAT MDevice to support EoE communication.

Max Ports	Number of ports the virtual switch can handle
Max Frames	Number of frames the virtual switch can queue
Max MACs	Number of MACs the virtual switch can keep in its cache
Save auto configuration settings:	This is used to configure the settings that are assigned to SubDevices that have *Auto config by MDevice presets* checked in their EoE settings tab, — See 2.3.9.2

2.1.6.1 esd MDevice specific

Virtual Switch	When this is enabled, the MDevice transfers all EoE frames between the EoE SubDevices, i.e. the EtherCAT network becomes the backbone of an Ethernet switch whose ports are the EoE SubDevices that are configured as “Switch Port”.
Virtual Port	When this is enabled, all EoE frames are forwarded to/from the virtual network interface that was created by using the esd TAP driver, see also next section. (When multiple esd TAP devices were created, set the index accordingly.)

2.1.6.2 esd TAP Driver

The TAP3 driver offers an additional network interface. It is installed by the start menu entry *EtherCAT Workbench → esdTAP → Add an esd TAP device*.
(On some systems a reboot might be required then.)

Use the start menu entries *List all esd TAP devices*/*Remove all esd TAP devices* accordingly.
It is configured the same way a local interface is configured, i.e. IP address, subnet mask, etc.

2.1.6.3 Samples

EtherCAT as Switch

- Enable *Virtual Switch* [*MDevice -> EoE -> Virtual Switch*]
- Set all participating EoE SubDevices to *Switch Port*
  [*SubDevice -> Mailbox -> EoE -> Switch Port*]

The EtherCAT network now acts as a virtual Ethernet switch – connect your devices to its ports (each EoE SubDevice) as you would do with a normal Ethernet switch.

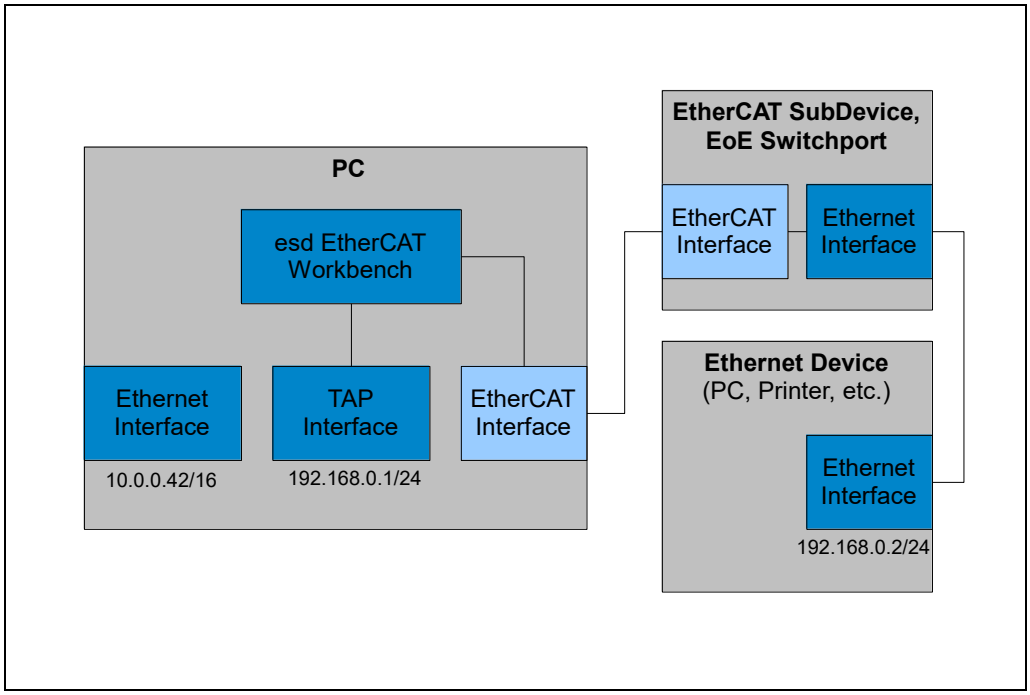


Figure 12: EoE *Virtual Port* sample with IP settings

Connect Device behind EoE SubDevice to PC with Workbench

- Set EoE SubDevice to *Switch Port* [*SubDevice -> Mailbox -> EoE -> Switch Port*]
- Install esd TAP driver on PC with Workbench.
  - Configure the TAP interface's IP settings to match the settings of the device that is behind the EoE SubDevice. Make sure these settings don't interfere with your other interface's settings. (Especially there must not be two or more interfaces on the PC within the same sub net)
- Enable *Virtual Port* [*MDevice -> EoE -> Virtual Port*]

You should now be able to "ping" etc. the device behind the EoE SubDevice from the PC and vice versa.

3 See also http://en.wikipedia.org/wiki/TAP.The esd TAP driver is based on the TAP-Win32/TAP-Win64 driver from OpenVPN™, see "3rd Party License Notices" for details.

Connect Device behind EoE SubDevice to LAN/Internet

- Same settings as with previous sample, but Ethernet Interface and TAP Interface must be bridged.
  For the sample in Figure 12 this would mean: Ethernet and TAP interface on the PC would become a single interface with the IP 10.0.0.42/16 and the device behind the SubDevice must use IP settings accordingly, e.g. 10.0.0.43/16.

The Ethernet device behind the EoE SubDevice is now in the “10.0.” sub net and can use the same Gateway and DNS settings etc. as the PC with the Workbench.

2.1.7 ENI Extras

This can be used to alter the resulting ENI .xml file after the EtherCAT Workbench application has created it (either explicitly when exporting the ENI file or implicitly when going to free run mode). Usually this should not be needed. It allows deleting nodes, adding nodes, and setting nodes content. Use the *Show examples* button to get some examples.

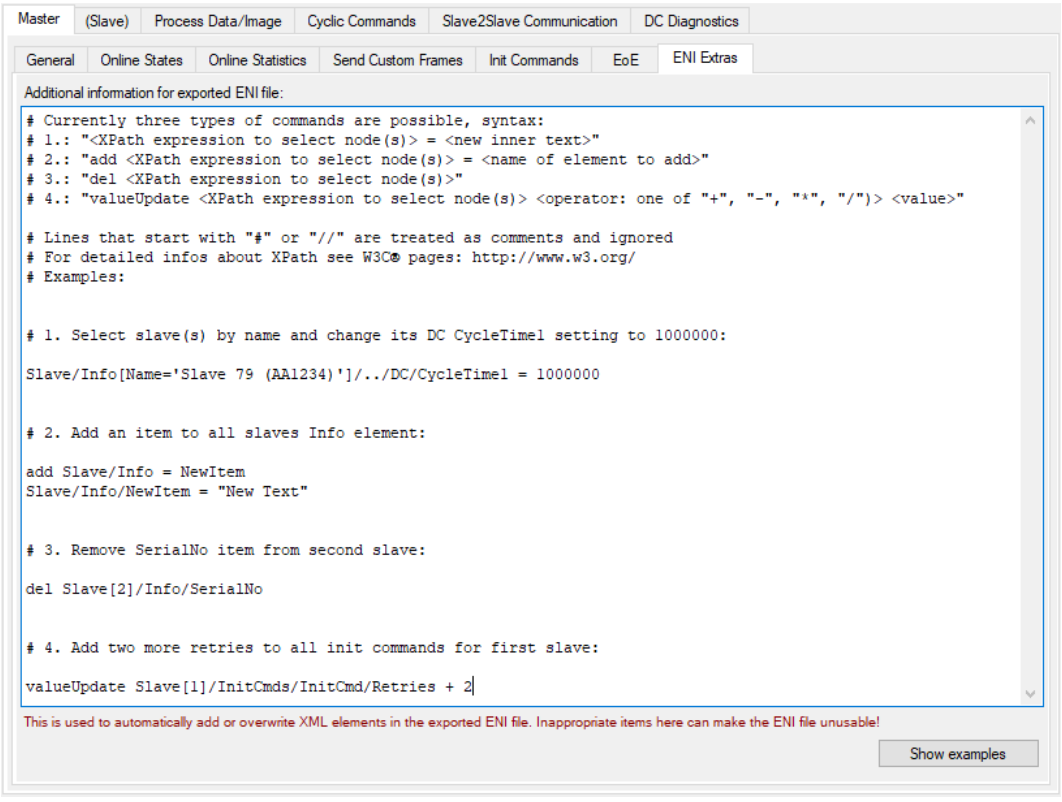


Figure 13: MDevice ENI Extras

2.2 ENI Export

Available via the toolbar button *Export ENI* or menu *File*: This stores the configuration in an ENI file according to ETG.2100 specification. This file can be used as EtherCAT network configuration for the esd EtherCAT MDevice and MDevice implementation of several other vendors.
This configuration is also used by the built-in MDevice of the Workbench when switching to free run mode.

2.3 SubDevice Configuration

2.3.1 Adding SubDevices to the Network

As stated in section 1.8.4 the SubDevice list is either populated manually or by running an online scan of an existing network. This chapter describes how to add the SubDevices manually.

The EtherCAT Workbench manages a repository of EtherCAT SubDevice device descriptions (the *SubDevice Library*) which is based on the ESI files of the device vendors. Later in this chapter it is described how this repository can be extended.

A SubDevice is added by using the context menu 1.8.4.1 of the SubDevice tree view: depending on the selected SubDevice different options will appear. If no SubDevices exist, the MDevice has to be selected, and the context menu will show an item *Append new SubDevice*. If SubDevices exist the context menu will offer more options depending on the selected SubDevice, e.g. different ports to add the new SubDevice to.
After clicking a menu item to add/insert the new SubDevice a window will show up:

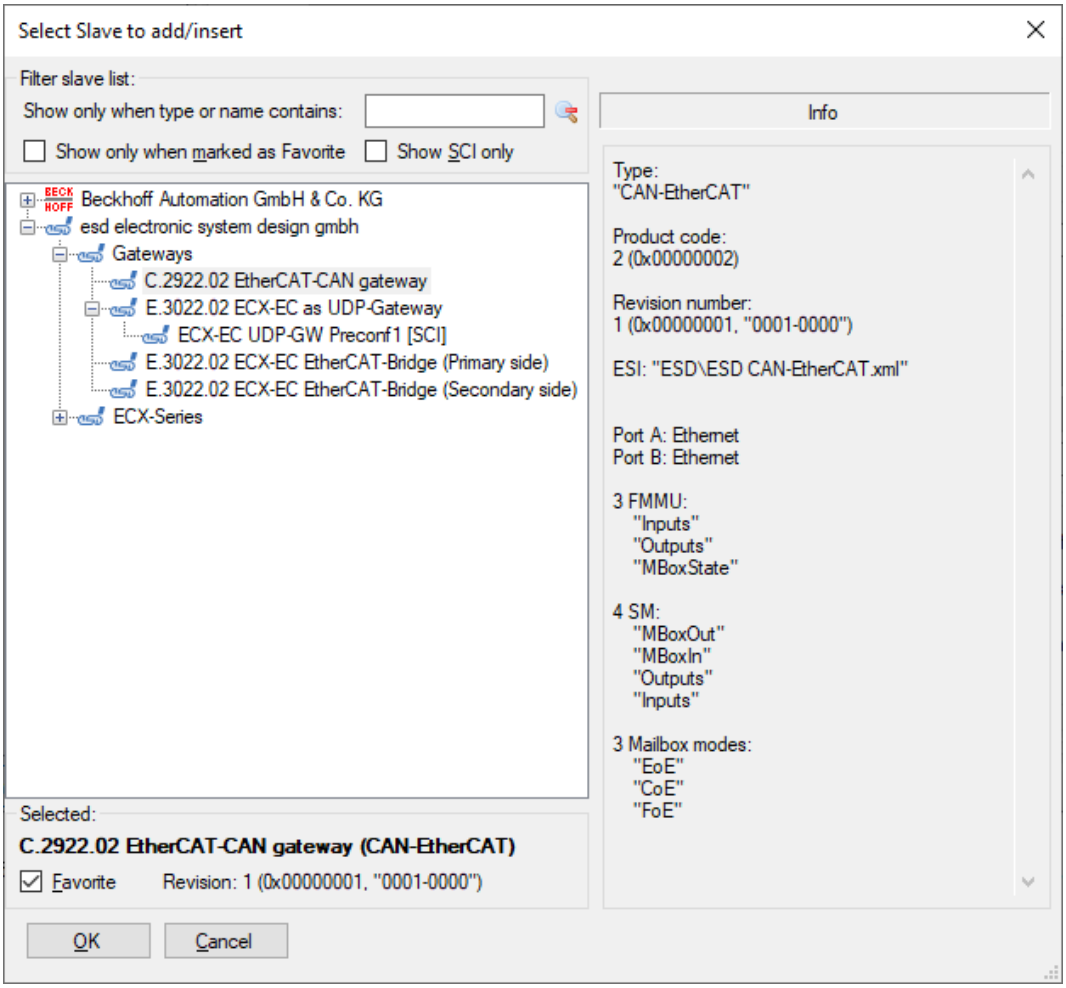


Figure 14: SubDevice device library Adding/Inserting a SubDevice

This window shows a list of all SubDevices that fit the selection (i.e. when it's chosen to add a SubDevice to an Ethernet port, no EBUS SubDevices will be visible) and allows to choose one of them. By clicking "OK" the selected SubDevice is added/inserted.

Within the list, the SubDevices are grouped according to manufacturer and device type given in the ESI file. The right *Info* pane shows some raw information from the ESI file of the SubDevice device.

SCI devices (see Figure 14) are marked with “[SCI]” to the device name appended and are grouped below their ESI device from which they were derived.
Enable the checkbox *Show SCI only* to set a filter to show only SCI devices, to make it easier to find them (but the grouping is still preserved).

Additionally, this form allows to filter the list by text or a *favourite SubDevice* marker.
If *Show only when marked as favourite* is checked, then the list will show only SubDevices that have been marked. To mark/unmark a SubDevice select it and check/uncheck the *Favourite* check box in the lower pane.
When a filter text is entered in the upper pane only SubDevices, whose name or type name contains this text are shown.
The context menu, which opens with a right click on the list or a list item, allows to expand/collapse the tree and to show obsolete items, too.

2.3.2 SubDevice Renaming and Change Revision

Select *Rename* from the MDevice or SubDevice context menu or press the shortcut key F2 on the selected MDevice or SubDevice to open the rename dialog: See 1.8.4.1

Select *Change Revision* from the SubDevice context menu. See 1.8.4.1

Change Revision is useful, for example, if you upgrade the SubDevice to a new software revision and want to continue using the project file without removing, adding and reconfiguring the SubDevice again.

The *Change SubDevice Revision Number* dialog opens. As for the add/insert dialog, only matching SubDevices - here with the same product number - are available for selection. Only the Revision number differs. The device with the current Revision is marked in bold type.

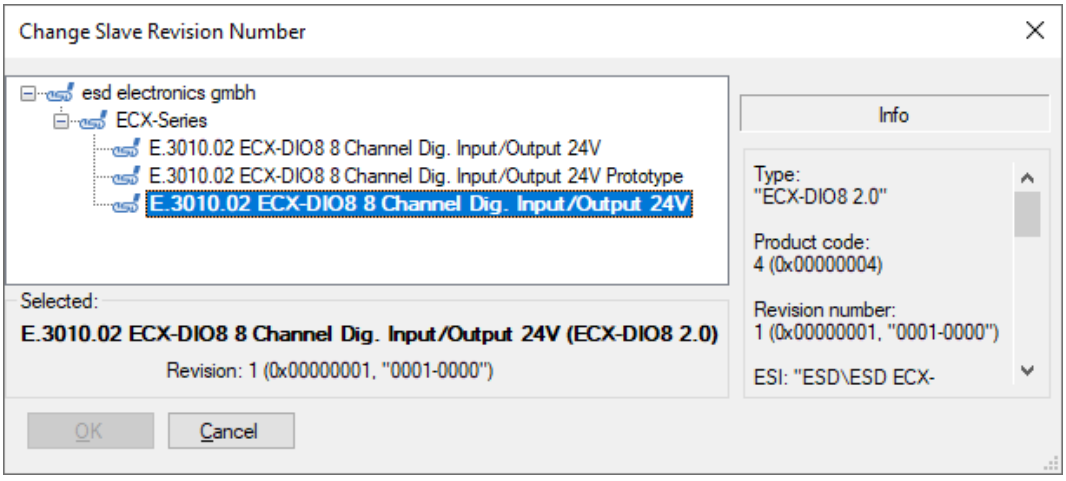


Figure 15: Change SubDevice Revision Number

After changing the revision, it may be necessary to step through the different SubDevice tabs and rebuild the SubDevice properties like dictionary, process data, DC, Modules. Then save and reload the project.

Removing SubDevices
To remove a SubDevice, *Delete SubDevice* from its context menu can be clicked[4]. When a SubDevice is connected to this SubDevice it is connected to its previous SubDevice. If that’s not possible it is deleted, too (same procedure for the deleted SubDevice then).

4 Or by pressing the “Del” key within the slave tree view.

2.3.2.1 Installing new ESI Files

The ESI files (.xml files) for the SubDevice library just need to be copied in the *SubDeviceLibrary* sub directory of the application's installation directory. It's parsed on each application start.
When the application is running, *Copy ESI file(s) to SubDevice library* from the *Tools* menu can be used to simplify this.

Entire directories with subdirectories can be added with *Copy folder with ESI files to SubDevice library*.
To add SCI files, use *Copy SCI file(s) to SubDevice library*.
The application needs to be restarted.

2.3.2.2 SubDevices without ESI File

SubDevices without an ESI file can't be added manually, but they are automatically added when found during a *SubDevice scan* in online mode. See 3.2
In this case the required information about the SubDevice is obtained from the SubDevice's ESI EEPROM data.

As not all data/information exists in the EEPROM, it is recommended to obtain an ESI file for every SubDevice. (Usually provided by its vendor)

SubDevices without ESI are shown in italics in the tree view without icon.

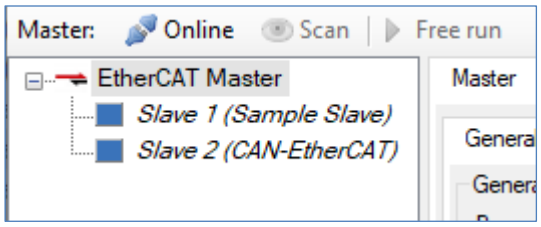


Figure 16: Scanned SubDevices without ESI

Once scanned and added to the project it is possible to save the project with the scanned SubDevice information for later continuing offline configuration.

You can also try to use *Change revision* to find and assign a possibly matching ESI with another revision, if available in the device library (also not recommended). See 2.3.2

2.3.3 General

This page shows some general information about the selected SubDevice: Addresses, port information, vendor Id, product code, etc.

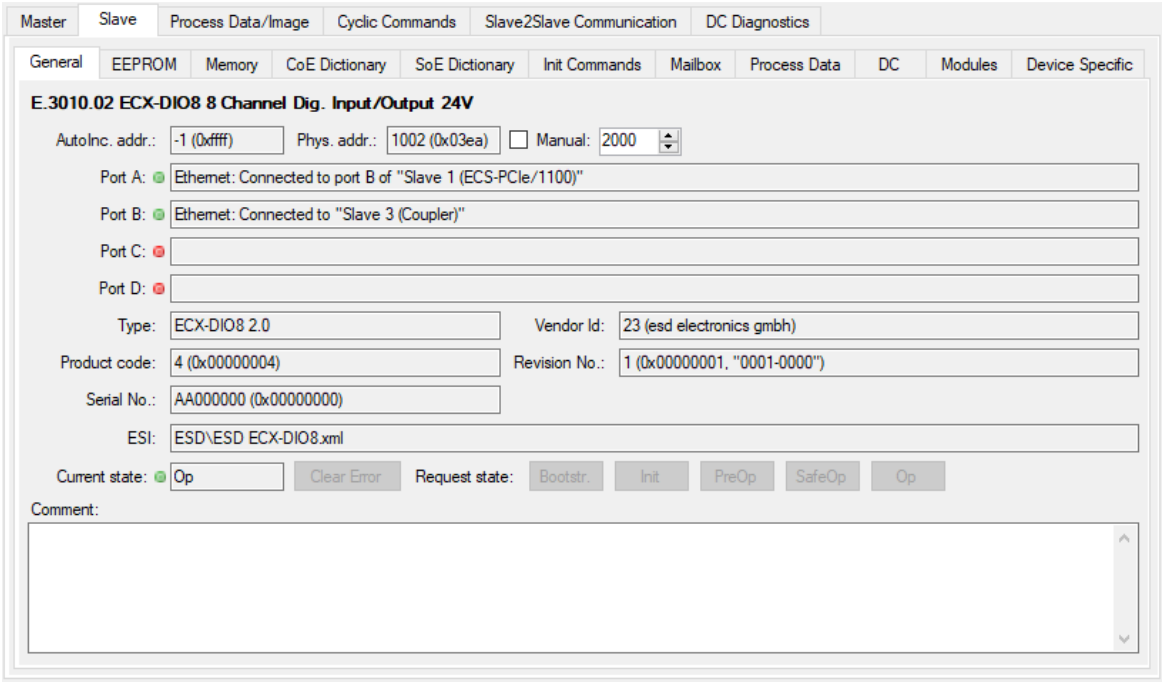


Figure 17: SubDevice configuration, page *General*

Ports: The text field for each port (e.g. “Port A”) shows the type (Ethernet/EBUS) and the SubDevice to which it is connected to. The image shows the – valid in free run mode only – link state of that port (green: “link”, red: “no link”, black: “unknown”).

Configuring the physical EtherCAT address: Use the *Manual* check box next to the *Phys. addr.* text field to override the automatic calculation. When the manually selected address would cause conflicts (i.e. multiple SubDevices would have the same physical address) it is ignored, and the next unused address is used instead.
Using alias addresses click: *Tools* → *Configure alias addresses for all SubDevices* – allows reading/writing of alias address of a single SubDevice or all SubDevices.
(Note: this editor is independent from the EEPROM editor (see 2.3.4), i.e. the data displayed there is not automatically updated when new alias address was written by the *Alias editor*)

Revision No.:	The “0001-0000” interpretation consists of the low and high word of the revision number as decimal.
Serial No.:	If available, e.g. after an online scan can be used for serial number verification; see *SubDevice* → *Init commands* → *Serial number verification.*
ESI:	The relative path of the ESI or SCI file within the SubDevice library from which the SubDevice originates.
Comment: (Text field)	Can be used for storing any user text, it has no effect and is used only within the project and ENI file.

For some operations, like adding an SCI device or a SubDevice without ESI, the EtherCAT Workbench adds comments itself.

2.3.4 EEPROM Data

This is used to read, write, edit, or create a SubDevice's EEPROM data. It contains an EEPROM editor that allows modifying ESI EEPROM data in a convenient property view instead of editing the hexadecimal (raw) data.
Editing EEPROM data is offline within the editor. The SubDevice device itself is involved only when *To device* or *Direct edit* (button Write) is clicked. Then the EEPROM content within the SubDevice will be overwritten or modified.

NOTICE
Writing improper EEPROM content might lead to a variety of errors – inaccurate changes to the device's configuration area might even damage the hardware!

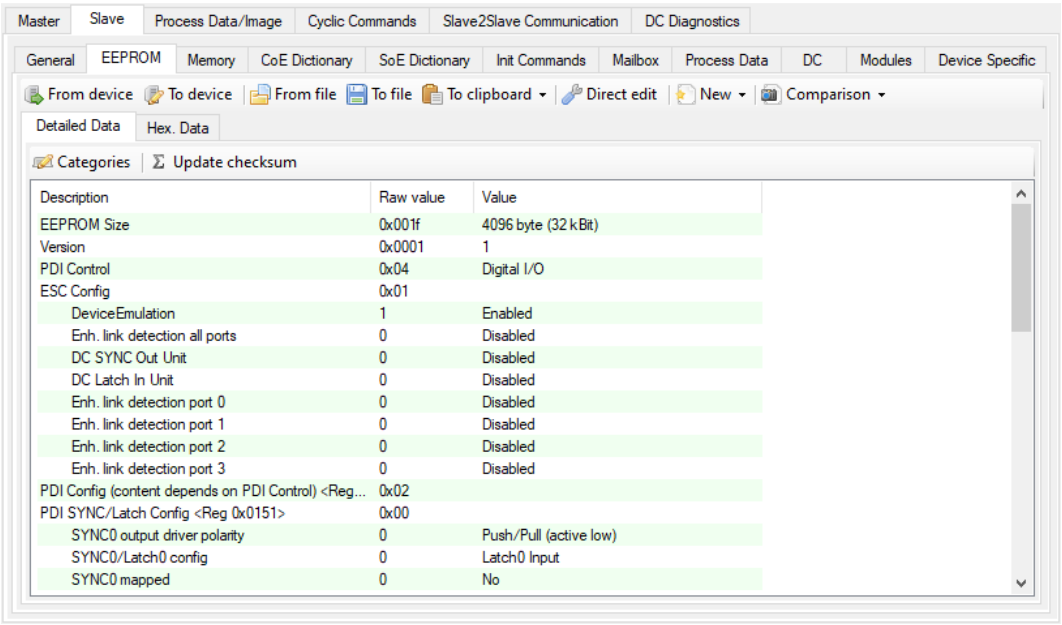


Figure 18: SubDevice configuration, page "EEPROM"

Before using the EEPROM editor, the EEPROM data has to be generated offline or loaded online from the SubDevice device.

Buttons	Description
From device	(Online EEPROM access) Reads the EEPROM data from the device to the EEPROM editor (online mode only). Current EEPROM editor content is overwritten without further notice.
To device	(Online EEPROM modification) Writes current EEPROM editor contents to the device (online mode only, does not ask for confirmation).
From file	(Offline) Loads EEPROM editor content from binary file.
To file	Saves current EEPROM editor content to a binary file.
To clipboard (Menu)	Opens a drop-down menu to select from different options to copy the EEPROM data as ASCII text to the clipboard (Item tooltips contain a short description).

Direct edit (Online EEPROM modification with button *Write!*)
Opens a dialog window that allows writing/reading single words (16 bit) to/from the EEPROM (Note: to write the SubDevice alias address menu *Tools → Configure alias addresses for all SubDevices* can be used)

New (Menu) (Offline)
Opens a drop-down menu to select from different EEPROM data creation methods. Each method just sets the editor content (overwriting existing content without further notice) and does not directly write anything to the device.

NOTICE
Please verify the created data carefully before writing it to a device. The EEPROM usually contains more data than available by .xml file – the serial number as simple example.
Additionally, the Workbench only writes complete EEPROM data (unless the *Direct edit* feature is used)**, therefore the *PreserveOnlineData* flag for certain EEPROM categories that might exist in the .xml ESI is ignored.**

- *From current SubDevice/ ESI settings*
  Creates EEPROM data from current ESI and current SubDevice settings.
- *From other ESI*
  Opens a window to select a SubDevice from the SubDevice library and creates EEPROM data from that SubDevice's ESI.
- *From custom ESI file*
  Opens a window to select a .xml file and then shows a window to select a SubDevice from within that file. Creates EEPROM data based on this selection.
- *From Scanned SubDevice*
  For SubDevices without ESI file: Reset the EEPROM editor to the originally scanned EEPROM data when the SubDevice was added for the first time.
- *Reset EEPROM data*
  Empty the EEPROM editor (this reduces the project file size)

Comparison (Menu) Opens a drop-down menu to select from different options to help you compare the EEPROM data with other data (Item tooltips contain a short description):
- *Show checksum*
- *Compare to file*

EEPROM Editor (page *Detailed Data*)
This list shows the fixed EEPROM content with its description and value. The dynamic content (*Categories*[5]) is edited by an additional editor accessible by the *Categories* button.

Each value is displayed as *Raw value* and *Value*. The raw value is the data as integral number, usually in hexadecimal notation prefixed with "0x", the *Value* column might show this value interpreted or with more information.

To edit a value, use the context menu entry *Edit raw value* or double click the item. A simple input window with the current raw value will show up, only integral numbers (if necessary prefixed with "0x") can be entered. Variables with a size of one bit are just toggled.

5 For more information about the EEPROM categories see ETG document 1000.6

Buttons	Description
Categories	Opens the category editor: The list on the left consists of the categories, the main window shows an editor for the selected category.
Update checksum	Use this to update the configuration area checksum (a wrong checksum is displayed with red text and value *Invalid*).

EEPROM Editor (page *Hex. Data*)
The tab *Hex Data* shows EEPROM address offset on the left, a plain hexadecimal representation of the EEPROM content in the middle columns, 16 bytes each line, and an ASCII text representation on the right.
View only - no editing possible.

2.3.5 Memory (Registers)

This tab page allows to view/edit SubDevice register values. The list of displayed registers is empty until an item selection is made. The *Item selection* drop down list must be used to determine what registers should be shown and how they are interpreted. Currently there's only one choice for detailed register information[6]: *All known registers*. For plain data without details/interpretation *All as Byte* or *All as Word* can be selected.[7]
All known registers will display multiple interpretations for some addresses because they depend on the PDI type for example. Also, not all SubDevice/types support all registers, so **reading all registers will always show warnings about some unread items.**

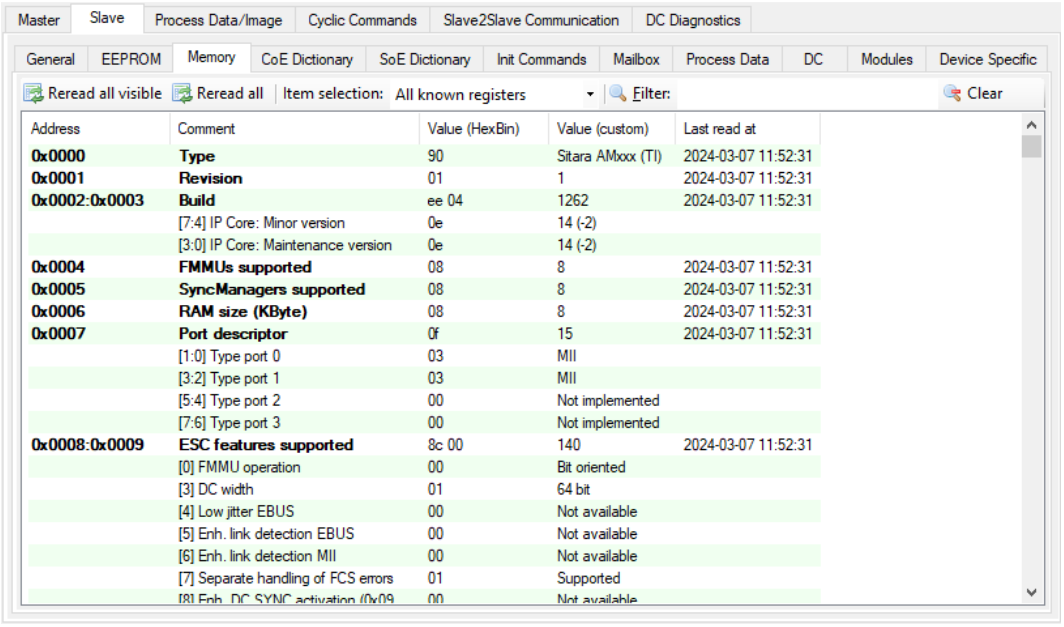


Figure 19: SubDevice configuration, page *Memory*

Other Toolbar Elements

Buttons: *Reread all* / *Reread all visible*	Tries to read all values for all items from current item selection from the device (online mode only). *Reread all visible* updates only the items that are currently displayed when *Filter* is used.
Item selection	Drop down list (*All known registers*)
Filter:	When a text is entered, the list will only show items whose *Comment* or *Address* column contain the entered text (not case sensitive).
Clear	Removes the filter and clears the filter input field, all entries are displayed again.

Context menu

Edit/view value	Value Editor (together with Process Data, CoE, SoE, …) Opens the variable editor for the selected value. (No information about whether a register is writable or not is available)	See 3.4.2
Reread selected item(s)	Tries to read all values for all selected items from the device (online mode only).	

6 The known details/registers won't cover all ESCs and configurations due to their variety.
7 To create a new item selection for a specific SubDevice/PDI type see application's "Templates\ESCMemory" sub folder: Create a copy of an existing file there and adapt it to your needs.

2.3.6 CoE Dictionary

This page is used to view/edit SubDevice's CoE data. Similar to the SubDevice memory items this list is initially empty. A CoE dictionary has to be created first: either by reading it from SubDevice (in online mode) or by creating it from ESI data. Use the toolbar button *Create dictionary* for that.

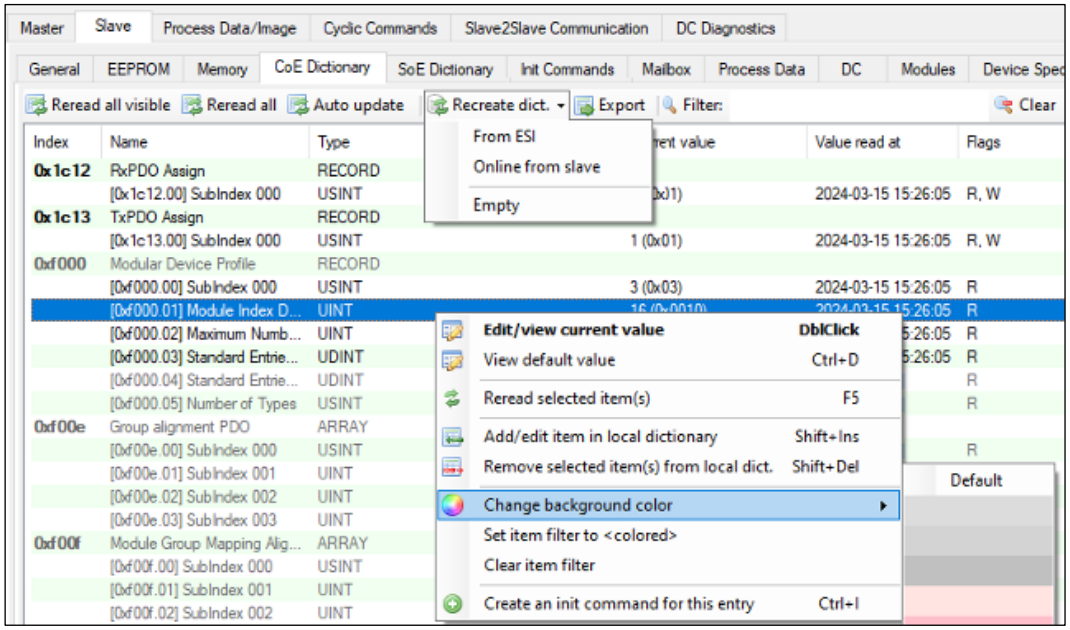


Figure 20: SubDevice configuration, page *CoE Dictionary* with context menus

Recreate dict

- *From ESI*
  Use the (offline) dictionary specified in the ESI.
- Online from SubDevice
  Read the dictionary online from the SubDevice (by SDO info service).
- Empty
  Clears the CoE dictionary viewer (this reduces the project file size)
  (Be careful: You need an online connection to the SubDevice again to restore an online dictionary that was read by *Online from SubDevice*).

Items that consist of multiple items (type "RECORD", "ARRAY", etc.) can't be edited directly, only their individual sub items are editable.

NOTICE
Note that SubDevices will reset most items in certain state changes. Especially changing to Free run mode includes a complete SubDevice initialization that usually resets manual changes here.
Initialization commands are added for SubDevice configuration,
see context menu *Create an init command for this entry* below and 2.3.9.1, Page *CoE*.

Other Toolbar elements

Buttons: *Reread all* / *Reread all visible*
Tries to read all values for all items from current item selection from the device (online mode only).
Reread all visible updates only the items that are currently displayed when *Filter* is used.

Auto update	Start an automatic cyclic update of the visible items like *Reread all visible.*
Export	Exports the CoE dictionary as EtherCAT dictionary file (conforms to EtherCATDict.xsd schema), a file save dialog for file path and name selection is shown before.
Filter:	When a text is entered, the list will only show items whose *Index* or *Name* column contain the entered text (not case sensitive).
Clear	Removes the filter and clears the filter input field, all entries are displayed again.

Context menu

Edit/view current value

Opens the variable editor () for the selected item's value. See 3.4.2

Edit/view default value

Opens the variable editor for the selected item's default value.
Only to view that value, no modification possible.

Reread selected item(s)

Tries to read all values for all selected items from the device (online mode only).

Add/edit item in local dictionary

Opens a dialog window to add or edit items in the local dictionary.
Only intended as "emergency", e.g. when an item needs to be read/written which does not exist in the dictionary.
Does not allow to edit all aspects of an item and does not check for invalid settings, so use with caution!
(Does not alter the SubDevice configuration and recreating the dict. will override all changes done with it).

Remove selected item(s) from local dict.

Allows to hide items that are not supported by the SubDevice or just don't need to be seen.
(It does not alter the SubDevice configuration.)

Change background color

Just a temporary setting for the display of this item within the list view.

Set item filter to <colored>

Enters <*colored*> as filter text, to filter and show only entries that have been marked as colored with *Change background color.*

Create an init command for this entry

Used to simplify creation of custom initialization of CoE init commands. The properties and the data of the CoE init command can be edited in the dialog that subsequently appears. See 2.3.9.1

Use the button *Var.Editor* to edit the data in different formats than hexadecimal. See 3.4.2

2.3.7 SoE Dictionary

This page is similar to the CoE page: For SubDevices supporting SoE an object dictionary is created once and SoE objects may be accessed then.

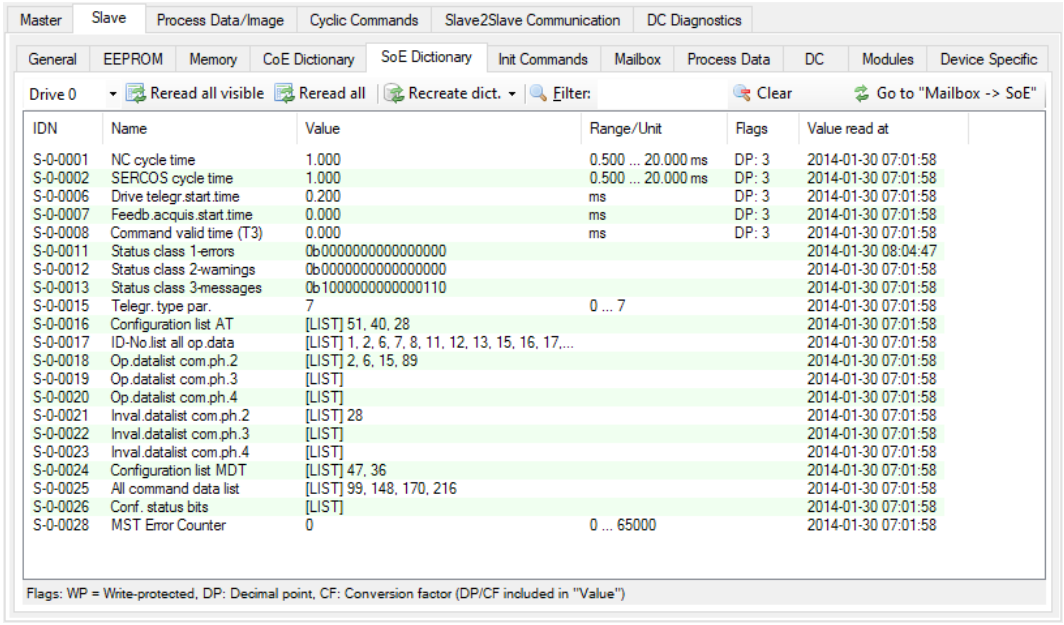


Figure 21: SubDevice configuration, page *SoE Dictionary*

For SubDevices including multiple drives the drop-down List (*Drive 0* in Figure 14) is used to select the drive to work with first. (*Drive* is also referred to as *Channel*, where Channel A equals Drive 0, Channel B equals Drive 1, and so on)

Recreate dict. → *Online from SubDevice* must be used first (and only once per drive) to obtain the list of all objects.
Reread all visible and *Reread all* are used to update the items and *Filter* to limit the displayed items – identical to CoE or other Workbench pages.

The context menu additionally allows to create an init. command for a specific object. See 2.3.9.4

2.3.8 Init. Commands

NOTICE
Distinguish between different location of initialization commands:

- Configuration of EtherCAT commands for MDevice initialization See 2.1.5
- Configuration of EtherCAT commands for SubDevice initialization See below
- Configuration of Mailbox CoE (CAN over EtherCAT, per SDO). initialization commands See 2.3.9.1

This page allows configuration of the initialization commands for the selected SubDevice. Usually this should be set to *Use default init. commands, configured by options below*, which means all commands are created automatically based on the corresponding settings. Simple configuration can be done by options below. Complete commands can be seen with the *View* button then.

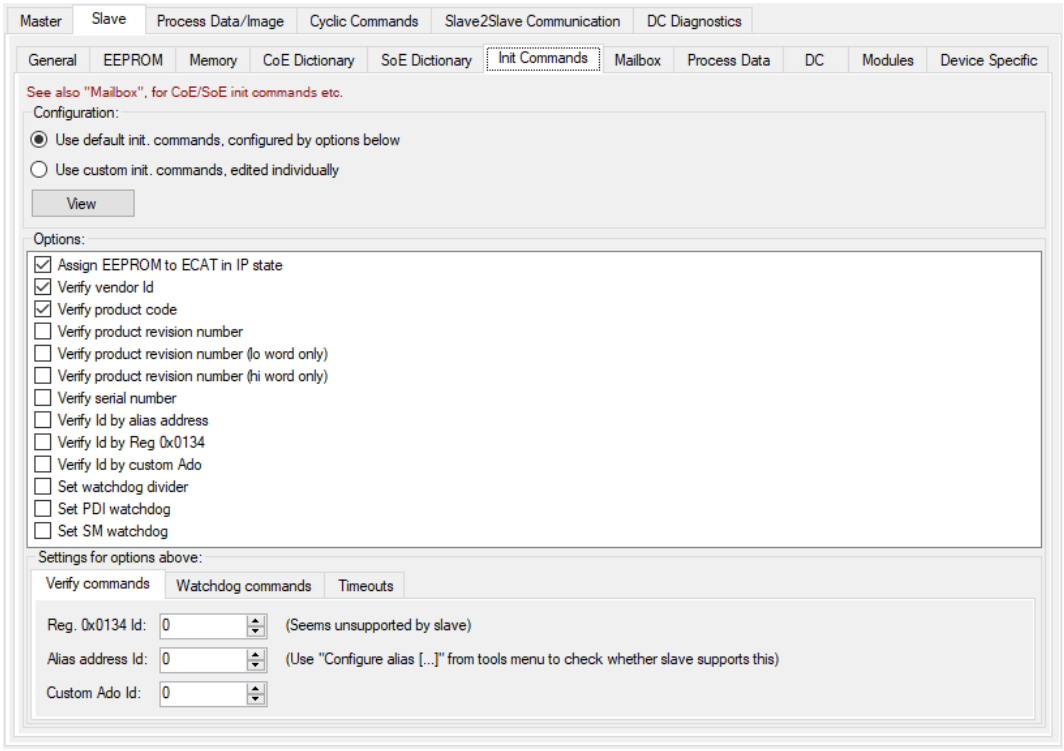


Figure 22: SubDevice configuration, page *Init. Commands*

If *Use custom init. commands, edited individually* is selected, the currently active automatically created commands are used, but they will never change automatically again when a corresponding setting changes. The *View* button changes to *View/Edit* then to allow a manual customization.
Be sure to understand the side effects of selecting custom init commands: e.g. most commands contain the SubDevice address; when another SubDevice is inserted, and the address becomes invalid this is not automatically updated, and all these commands are likely to fail then.

Options:

Assign EEPROM to ECAT in IP state
Adds a cyclic command to assign the SubDevice's EEPROM to the EtherCAT MDevice (SubDevice reg. 0x0500) to the IP state transition.
Required by most of the *Verify …* options.

Verify …
Adds commands to read and verify certain SubDevice values, e.g. its vendor id: When the id differs from the expected one, the SubDevice initialization will fail.

Set watchdog divider,
Set PDI watchdog,
Set SM watchdog Writes the configured values (tab *Watchdog commands* below) to the corresponding SubDevice registers (0x0400 etc.).

Settings for options above pages

Verify commands *Reg. 0x0134 Id:* / *Alias address Id:* / *Custom Ado Id:*

- Explicit Device Id feature: Value that shall be read by the according *Verify Id* command. Reg134/CustomAdo usually configured by switch at the SubDevice,
  Alias usually configured by the Workbench (*Configure alias [...]* in *Tools* menu).

Watchdog command

Used with *Set watchdog divider*, etc. in *Options* above. For details see ESC documentation of watchdog registers

Timeouts *PreOp Retries*:

- *Retries* value in ENI for Init commands to change state to PreOp

Other values correspond to *InfoType* → *State Machine* → *TimeOut* values (e.g. *BackToInitTimeOut*) as defined for ENI

View/Edit opens the *Init Commands* Editor, see MDevice Init commands and context menu.

2.3.9 Mailbox Settings

Page *General*
- *Start* and *Length* for the addresses taken from ESI, usually no modification should be needed (And SubDevices might deny changing this settings)
- *Poll cycle* / *Poll state change*
  Determines how / how often the mailbox states are read

Page *Bootstrap* All values taken from ESI, usually no modification is required

Page *CoE* See 2.3.9.1

Page *EoE* See 2.3.9.2

Page F*oE* See 2.3.9.3

Page S*oE* See 2.3.9.4

Page A*oE* See 2.3.9.5

Page V*oE* See 2.3.9.6

2.3.9.1 Page *CoE*

NOTICE
Distinguish between different location of initialization commands:
- Configuration of EtherCAT commands for MDevice initialization See 2.1.5
- Configuration of EtherCAT commands for SubDevice initialization See 2.3.8
- Configuration of Mailbox CoE (CAN over EtherCAT, per SDO) initialization commands See below

Used to configure CoE initialization commands. Beside the default commands custom commands can be added by the context menu. Custom commands are displayed italic. The default commands are fixed and cannot be deleted or modified. They are added according to ESI specifications and depend on the options listed below.

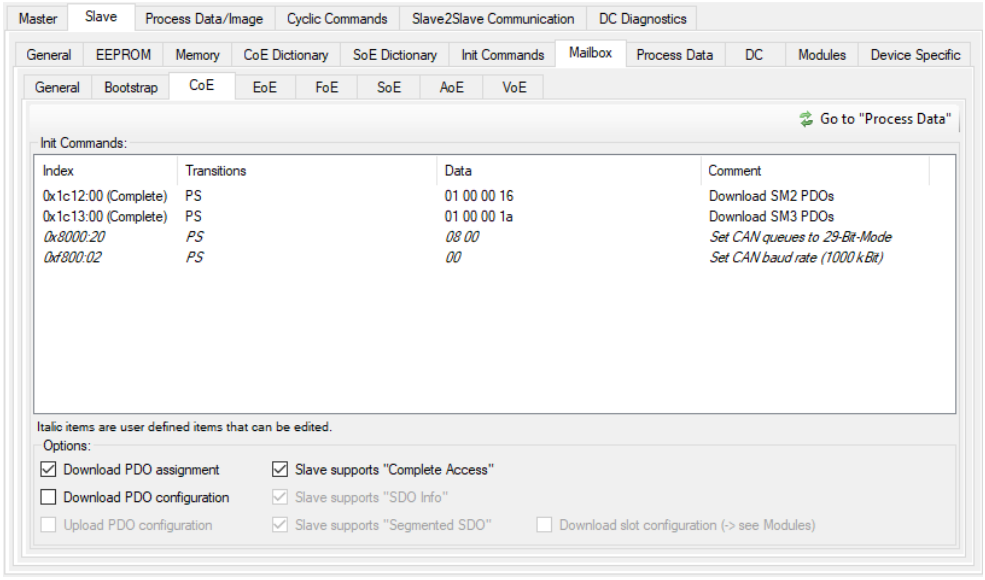


Figure 23: SubDevice configuration, page *Mailbox*, *CoE*

Options:

Download PDO assignment

Determines whether CoE init. commands to assign the PDOs should be created. Initial state set by ESI.

Download PDO configuration

Determines whether CoE init. commands to download the PDO contents should be created. Initial state set by ESI

SubDevice supports Complete Access

Determines whether these init. commands *combine* their data. Initial state set by ESI.
Does not affect custom commands when changed.

Upload PDO configuration

- Reflects information from the ESI file: *PdoUpload* attribute
- If PDOs are not uploaded before MDevice *Free run*, you receive a warning message in the Message log:
  <SubDevice name> has flag *UploadPDOConfig* set, but PDOs were never uploaded...
- **You are prompted to load the PDOs online from the SubDevice: Go to *SubDevice → Process data* and select *Recreate PDO list → Online by SDO Info service*.**

There is a shortcurt *Got to Process Data* in the toolbar of the *CoE* tab for this.

SubDevice supports: *SDO Info*

Reflects information from the ESI file: *SdoInfo* attribute.

SubDevice supports: *Segmented SDO*

Reflects information from the ESI file: *SegmentedSdo* attribute.

Download slot configuration

- Reflects information from the ESI file: *DownloadModuleIdentList* attribute
- The actual setting is made in the *Modules* tab (→ see *Modules/Slots).*

See 2.3.12

Context menu

Edit selected — Opens a dialog window to edit the selected init. command. Only allowed for custom commands (displayed italic).

Append new item — Opens a dialog window to edit and append a new custom init. Command, See also *CoE/Tab → create an init command for this entry.*

Copy selected item to other SubDevice(s)

Opens a dialog to select other SubDevices and copies the selected init. command to their init. command list. When a fixed command is copied, it's turned into a custom command for the other SubDevices.
When a command already exists at a SubDevice's init. command list a question dialog will ask whether to append it anyway or to alter the existing one. (A command already exists when it's for the same object index and sub index and shares at least one transition.)

Delete selected item — Deletes the selected custom command

2.3.9.2 Page *EoE*

This page is used to configure the SubDevice's Ethernet settings when it supports EoE.

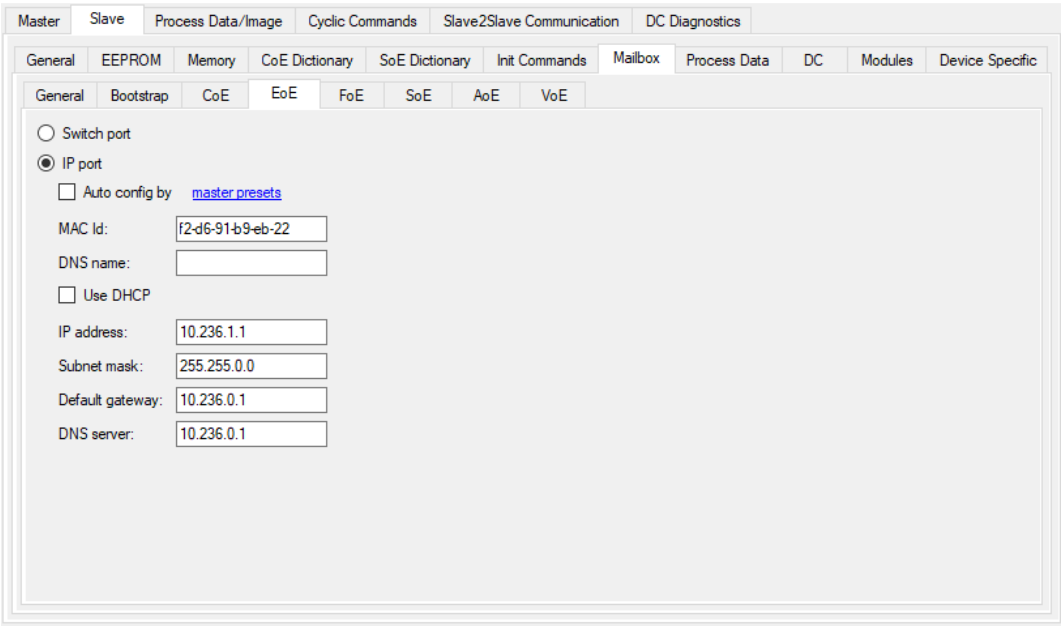


Figure 24: SubDevice configuration, page *Mailbox*, *EoE*

Switch port SubDevice shall act as a switch port. No IP address is assigned, and no further configuration is necessary.

IP port SubDevice shall have its own IP address, etc. MAC Id, DNS name and other TCP/IP settings can be configured either manually or automatically by MDevice's EoE presets. See 2.1.6
The IP port settings are only visible if you select the radio button. The IP address settings are only visible if you de-select the checkbox *Use DHCP*.

For usage and instructions see the examples. See 2.1.6

2.3.9.3 Page *FoE*

This page is used for FoE uploads and downloads.

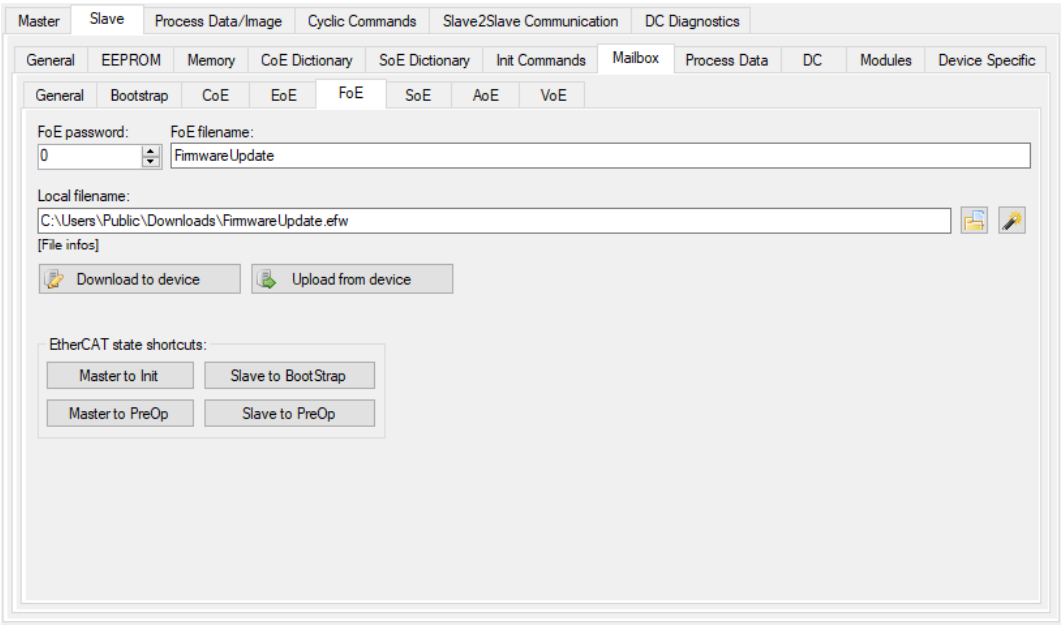


Figure 25: SubDevice configuration, page *Mailbox*, *FoE*

FoE password	Password for the FoE access – a 32-bit unsigned integer
FoE filename	Name of the file to be up- or downloaded (The ETG specification allows only ASCII characters – a warning is shown when this string contains other characters)
Local filename	The local filename for the FoE transfer, i.e. the file that is downloaded to the device, or the destination of the uploaded file
[File info]	When clicked, this shows whether the file exists and if so some additional information like size, etc.
Download to device / Upload from device	Starts the download/upload
EtherCAT state shortcuts	Many SubDevices require e.g. the BootStrap state for downloads. These buttons are just duplicates of buttons found in other tabs to simplify EtherCAT state changes
-Button:	Opens the standard Windows file dialog to select a file
-Button	Opens a context menu for user defined inputs / *Favorites*. Initially it will only contain an item *Edit favorites*: by this an .xml file that contains these entries is shown. If the file does not exist a sample file is created that shows how own entries can be added.

2.3.9.4 Page *SoE*

Used to configure SoE initialization commands. Beside the default commands custom commands can be added by the context menu. Custom commands are displayed italic. The default commands are fixed and cannot be deleted or modified.
Very similar to Page *CoE*, see 2.3.9.1

.

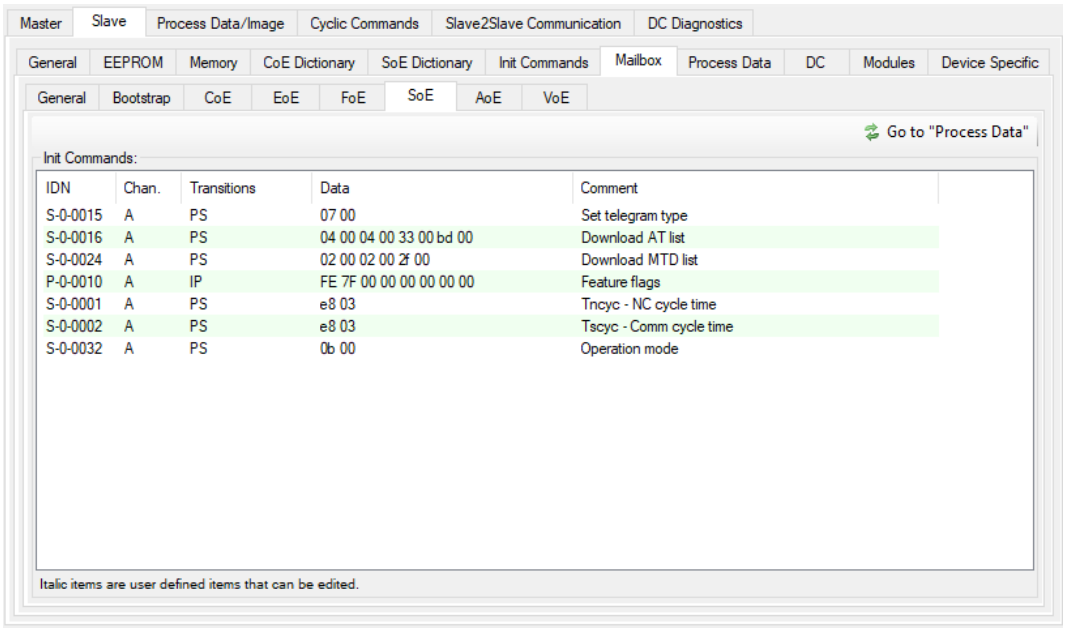


Figure 26: SubDevice configuration, page *Mailbox*, *SoE*

For a description of the context menu -> See 2.3.9.1

2.3.9.5 Page AoE

See 3.6.3.2

2.3.9.6 Page VoE

See 3.6.3.3

2.3.10 Process Data

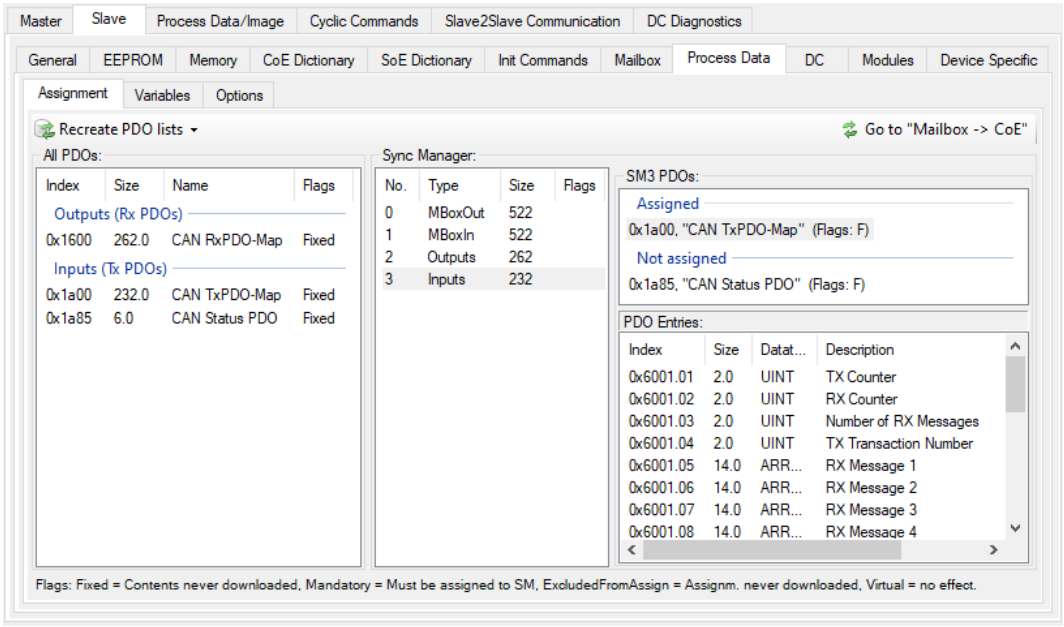


Figure 27: SubDevice configuration, page *Process Data*

This page shows an overview of all PDOs (*All PDOs:*) and allows to assign the PDOs to the sync manager and to configure the PDO content. When an item from *Sync Manager:* is selected the available PDOs are displayed next to it (*SMX PDO selection:*). Assigned PDOs are listed under *Assigned*, unassigned PDOs under *Not assigned*. Double click on an item under *Assigned* or *Not assigned* or use the context menu (see below) to assign / unassign PDOs.

If the PDO lists are empty use the *Recreate PDO lists* toolbar menu described below to fill the lists with predefined entries.

When items can't be assigned or unassigned this is caused by the items being marked to exclude each other or by being marked as mandatory. (Information: In the ESI or online data when obtained online).

Click on an item under *Assigned* or *Not assigned* to view the PDO content in *PDO Entries* below.

Page *Assignment*

Recreate PDO lists (Drop down menu)

- *From ESI*
  Recreates the PDO lists from ESI (default).
- *Online by SDO Info service*
  Reads the PDO information from the device.
  Done by reading CoE objects 0x1600/0x1a00 and following
  (see also ETG document "Modular Device Profile Framework").
- *From EEPROM*
  Build the PDO lists from EEPROM information.
  You have to create an EEPROM content first. See 2.3.4
- *Set Empty*
  Remove all items. Be careful: If you remove PDOs that you retrieved by "Online by SDO Info service" you need an online connection to restore the lists!

You can also use the context menus of the items in the lists *All PDOs* and *SM2 PDO* to edit/delete the items, append Rx/Tx PDOs, toggle assignment, etc..

Editing PDOs

- Select *Edit selected item, Append Rx PDO, Append Tx PDO or Edit PDO* to open the PDO Editor window:
  In the **PDO Editor** window click on the entry you want to edit.
  Via the context menu you can select:
  - *Edit Append new entry* – to open *Edit PDO Entry* dialog
  - *Append from Dictionary*
    - You have to create the CoE dictionary first If none exists (in that case go to *SubDevice → CoE dictionary).*
    - Select a dictionary index and subindex from the combo box dialog that is displayed afterwards.
  - *Clone Item*
  - *Remove item*
  - *Move up/down*

Page *Variables*

- Lists all the variables within the activated PDOs for that SubDevice (**a double click on an entry** will switch to that entry on the *Process Data/Image* tab).

Page *Options*

On this page the following process data commands can be selected:

- *SubDevice Supports LRW command*
  Determines whether the selected SubDevice supports the *LRW* command for reading/writing the process data.
  If that is not supported, the commands *LRD* and *LWR* are used instead. Initial state set by ESI data.
- *Always separate commands for this SubDevice*
  When selected this SubDevice's process data handling is not combined with other SubDevices process data, i.e. separate EtherCAT commands to read/write the SubDevice's process data are always generated, regardless of global *Combine PD commands* setting See 3.4
- *LRD/LRW only in Op* / *LWR only in Op*
  When selected EtherCAT commands to read/write process data from this SubDevice are sent only in Operational and not in SafeOp too.
  Usually this shouldn't be needed, but it might be useful to avoid WKC errors if the SubDevice handles a command only in Op (Should be used with *separate commands* only, else other SubDevices might be affected accidentally, too)
- *Sync unit*
  Used to define groups of which SubDevice's process data access is combined, i.e. accessed with the same EtherCAT command.
- *Cycle multiplier*
  Used to determine the interval for the EtherCAT commands to handle the SubDevice's process data. Multiplier base is the MDevice base cycle time, See 2.1.1

2.3.11 DC

Operation mode Used to select the DC mode.
Modes are read from the ESI and can't be edited here.

Details for the selected operation mode
Some details concerning SYNC/Latch for the selected operation mode can be selected here.

Options: In this field you can select:

- *Force this SubDevice being used as reference*
  By default, the first SubDevice with DC support is used as reference. This check box is used to override this.
- *Add custom value to "Shift/Start Time"*
  During SubDevice's DC SYNC unit setup this value is added to the start time register (0x0990).

Toolbar of tab *DC - Recreate DC Options*
When you add a SubDevice, DC options are created from the ESI.
You can change the settings via this menu:

From EEPROM Choose this item to create DC options from the EEPROM instead.

From ESI Choose this item to switch back to ESI options.

No DC Removes DC support completely. This can only be used for test purposes. If the SubDevice needs a DC mode, you can't remove it with this selection!

2.3.12 Modules

Modules are an approach to variable SubDevice configurations. The typical use is drives and gateways.
Modules can represent a real hardware plug-in but also a modular software extension. A module may extend the CoE dictionary and/or the PDO list of the SubDevice.
The SubDevice (ESI) has to support Modules for this configuration option else *SubDevice has no modules(slots)* is shown in the tab!

The modules tab is divided into two areas. The *Slots* list on the left and *Modules* selection on the right. When you click a slot item (or the already assigned module item below the slot) the Modules selection is filled with matching modules than can be assigned to the slot. The slots are numbered "flat" starting from "1:" up to the maximum possible module assignments ("2:", "3" , …, "max" then a slot name follows.

Basically, there are two patterns of modules / slots:

- A group of different slots, each of which can be assigned a different selection of modules.
- An array of identical slots each of which can be assigned a module from a module class selection. The slot name is postfixed with an index in square brackets (*Terminals[n]* as example, or a hybrid of both e.g.: *Axis2[n]*)

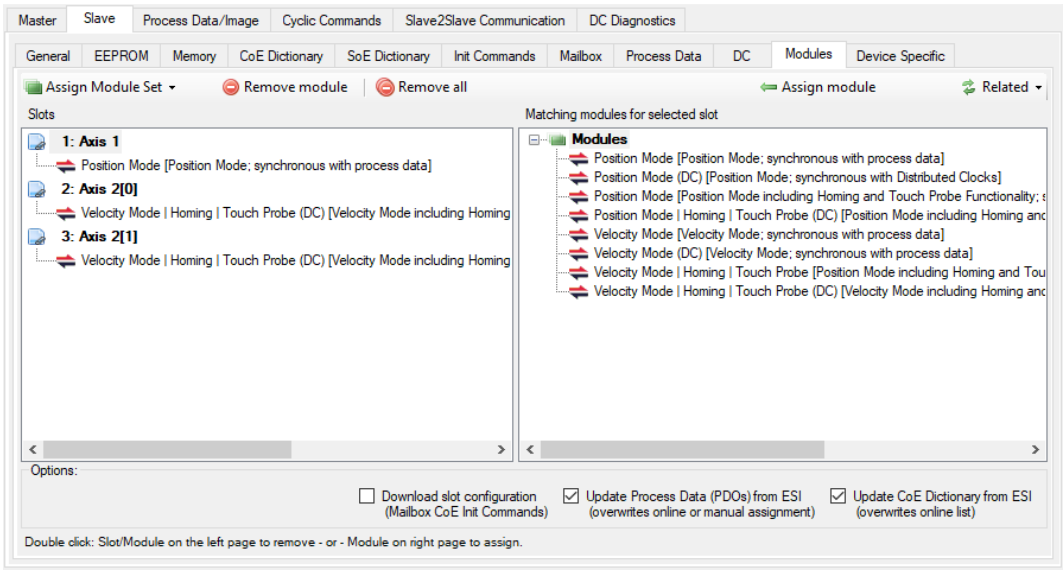


Figure 28: Assignment of modules to slot groups

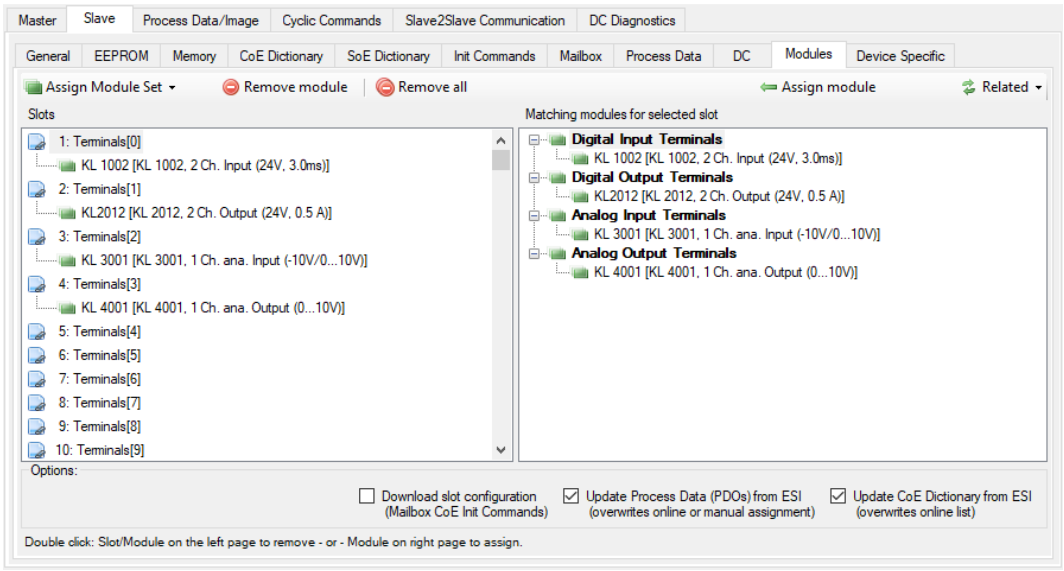


Figure 29: Assignment of module classes to a slot array

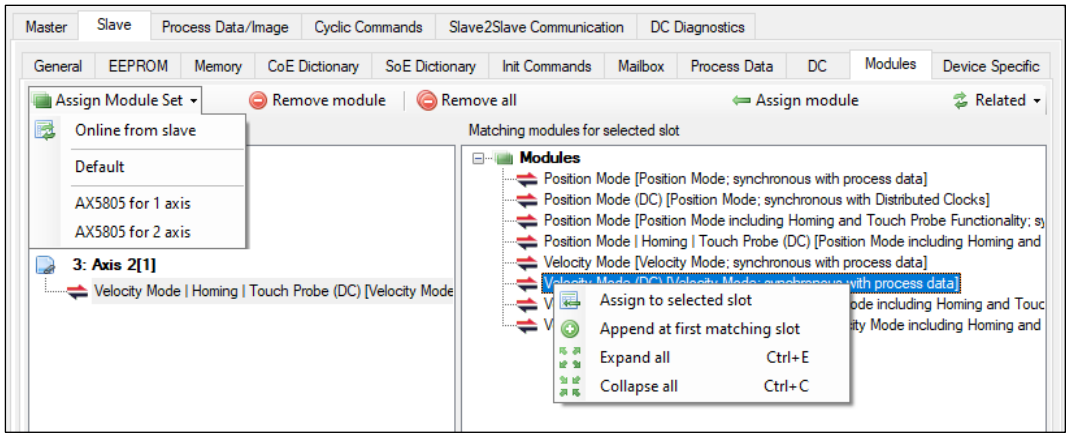


Figure 30: Modules context menu

To assign modules manually for both modules / slots patterns:

- Select a slot on the left *Slots* list in the *Modules* tab
- On the right *Matching modules for selected slot* list, expand all modules or a module class with a click on "+" (or expand all modules and classes from the context menu, see below)
- Select a single module on the right
- Click *Assign module* from the toolbar (alternatively: double click on the selected module, alternatively: select *Assign to selected slot* from the context menu)

The selected module is assigned to the selected slot.

Toolbar of tab *Modules*

Assign Module Set menu

Online from SubDevice — Retrieve the assigned modules online from the SubDevice

Default — An ESI defined default assignment

Other Named default module assignment (example here: "AX5805 for n axis")
ESI defined default assignments

Buttons

Remove module — Remove module assignment from the slot. The ESI can impose restrictions on which and how many modules may be removed.

Remove all — Remove all modules from all slots unconditionally. This can violate ESI restrictions and should only be used for test purposes. Select a default assignment (see above) to fix assignment violations.

Assign module — Assign the selected module to the selected slot

Context menu

Assign to selected slot — Assign the selected module to the selected slot (the same as *Assign module* from the toolbar)

Append at first matching slot — Assign the selected module, without selecting a slot on the left, to the next unassigned (free) matching slot on the left

Expand all / Collapse all — Expand / Collapse the tree view

Check boxes

Download slot configuration (Mailbox CoE init command)
Adds CoE init commands (see *SubDevice → Mailbox → CoE*) for CoE object F030 hex. (*Configured Module Ident List*). This can be used to set or check the module configuration within the SubDevice. The default is taken from the ESI file of the SubDevice and should not be changed (only for test purposes).

Update Process Data (PDOs) from ESI (overwrites online or manual assignment)
Update the complete PDO configuration and assignment (see *SubDevice → Process Data → Assignment*) each time after the module assignment is changed, online read-in or manual changes are lost. Uncheck to prevent automatic PDO updates. Uncheck temporarily to improve application performance, when you have finished assigning modules check again.

Update CoE Dictionary from ESI (overwrites online list)

Update the complete CoE dictionary (see *SubDevice → CoE Dictionary*) each time after the module assignment is changed. Online read-in or manual changes are lost. Uncheck to prevent automatic dictionary updates. Uncheck temporarily to improve application performance, when you have finished assigning modules check again.

INFORMATION

PDO names that originate from modules are prefixed with slot name and a dot.

Example: Module at Slot “Axis1” →
PDO name: “Axis 1.PDO module name”

Example: Module at Slot “Terminals[5]” →
PDO name: “Terminals[5].PDO module name”

The *Process Data/Image → Variables* tab also reflects these PDO names as variable names.

INFORMATION

CoE object names in the CoE dictionary that originate from modules are given the postfixes “[MODULE]” and if the object index is slot dependent also the slot number in square brackets “[Slot-Number]”.

Tipp: Enter the key word “[MODULE]” as filter string in the CoE tab to find all module associated objects.

2.3.13 Device Specific

The content of this tab page depends on the SubDevice type and features. It is determined by the SubDevice MDP number in CoE object 0x1000 or by an internal list of known SubDevices. If the SubDevice is not recognized the context menu allows selecting the device specific page manually.

Basically, these pages only simplify tasks that are possible by other means, too:
When setting a bit rate for a device supporting the *CAN Interface* (MDP 5000) for example, it will just write the according CoE object as if the user used the *CoE Dictionary* page.

Context menu on the *Device Specific* page
Right click on the empty form (*No specific tools available*) - or - on a free space in the currently displayed form:

Automatic	Switch to the best matching device specific tool.	
CoE Tool	Switch to “CoE tool”	See 3.6.3.1
	Hint: For SubDevices supporting *Mailbox → CoE* you can always switch between *Automatic* and *CoE Tool*.	
CAN Interface	-	See 2.3.13.2
EtherCAT Bridge, primary side / EtherCAT Bridge, secondary side		See 2.3.13.3

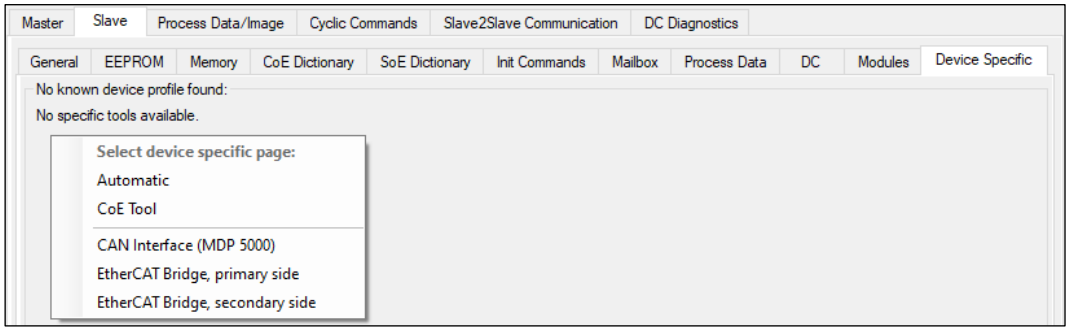


Figure 31: S Context menu on the *Device Specific* page

2.3.13.1 CoE Tool

See “Online” See 3.6.3.1

2.3.13.2 CAN Interface

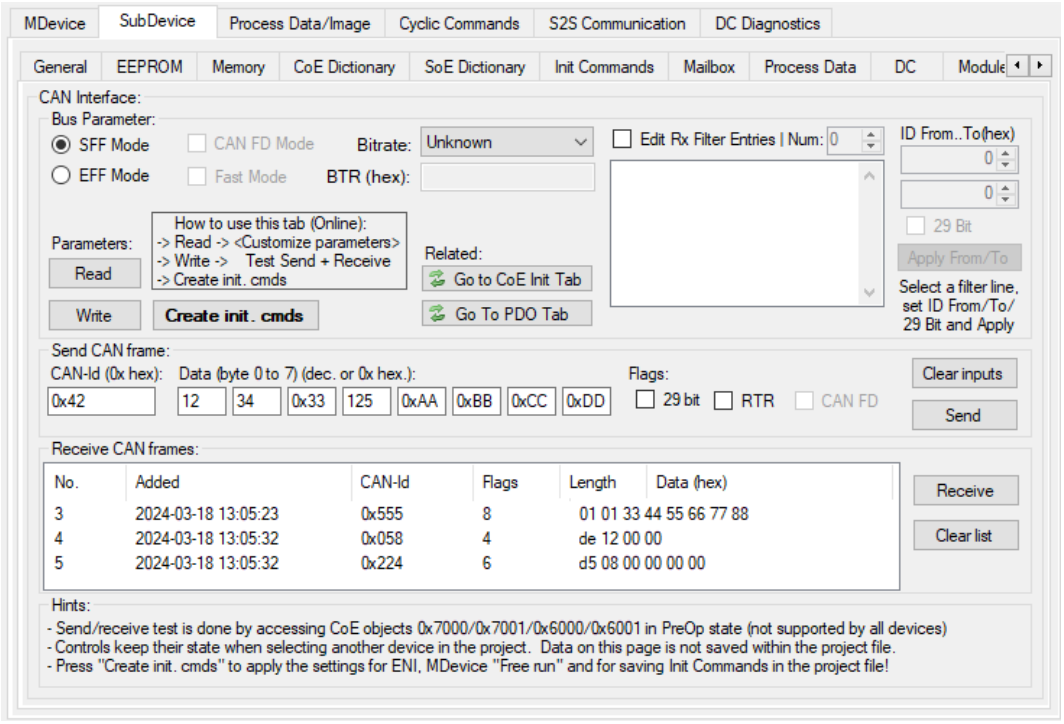


Figure 32: SubDevice / Device Specific CAN Interface

- Selected for SubDevices supporting the MDP No. 5000 (esd CAN-EtherCAT).
- Allows setting the CAN bit rate, sending/receiving CAN frames (in “PreOp”) and creating CoE init. commands for the bit rate.

Select from *Bitrate* and click *Create init. Cmd* to append CoE init commands for the CAN bus bitrate, see Figure 32.
Online only! Click *EEF Mode* or *SSF Mode* to switch the “esd CAN-EtherCAT” device to CAN bus 29-bit (*EFF Mode*) or 11-bit (*SFF Mode*) CAN Id support respectively. (29-bit support includes 11-bit CAN Ids but has larger message queues and process data). *SubDevice → Process Data* (PDOs) and *Process Data/Image* are updated immediately online and init commands are added.

Online only! Click on *Receive* to read and display received CAN frames. Enter a CAN ID (dec. or hex. with prefix “0x” and CAN data bytes dec. or hex. with prefix “0x”), select “29-bit” for 29-bit ID or “RTR” for Remote transmission (no data!) and click *Send* to send CAN bus data.
See the *Hints* at the bottom of the form: The form cannot be used in *Free run* mode!
See esd CAN-EtherCAT manual for further information.

2.3.13.3 EtherCAT Bridge

EtherCAT Bridge (primary side) / EtherCAT Bridge (secondary side)
- Selected for known bridge devices (e.g. esd ECX-EC or Beckhoff EL6692).
- Allows creating bridge variables (items in PDOs 0x1600/0x1a00) and loading/saving them from/to file and device (download to object 0x10ff on primary side).

See the page’s tool tips for more details.

Refer to “ECX-EC EtherCAT Bridge” manual for further information. The esd “UDDC” tool is a better option for configuring the esd “ECX-EC EtherCAT Bridge”.

2.3.14 Alias Addresses

Select *Tools* in the main menu and *Configure alias addresses for all SubDevices*: to open the SubDevice *Alias editor.*
The *Edit SubDevice alias addresses* dialog is available in online mode and allows to assign alias addresses to SubDevices. Alias addresses are written to the EEPROM at the *Configured Station Alias* position as a second fixed address. They can be useful in connection with Hot Connect. see 2.3.15
Each line of the list shows the current alias address in decimal and (hexadecimal) and the SubDevice name and icon.
To automatically assign unique addresses to all SubDevices use the *Write all ascending/descending* buttons with start address *Address*.

Buttons/Menu Items

Reread all Reads the addresses from the SubDevices' EEPROM cells. This is also done before the dialog opens.

Address The address to be written with *Write to selected*, or the base address for *Write all ascending/descending*

Write to selected Writes the entered *Address* to the selected SubDevice's EEPROM.

Write all ascending / Write all descending
Write addresses in sequence ascending or descending respectively to all SubDevice EEPROMS, starting with *Address* as base address and incrementing/decrementing the written address. The result is shown in the list.

Verify memory values
Reads the currently effective addresses from the “Configured station alias” memory address of the SubDevices. Since the address written to the EEPROM only becomes effective after a restart of the SubDevice you are likely to receive a warning message when you execute the verify immediately after writing.

2.3.15 Hot Connect

Hot Connect allows to declare SubDevices or groups of SubDevices as being optional, i.e. the EtherCAT network will work with or without these SubDevices and allow to add/remove them while running.
The Workbench will create separate commands to access the process data of these SubDevices to allow the MDevice to achieve this. Additionally, the MDevice needs to know how to identify such a group – this is configured by the Workbench as well.
To create/edit new Hot Connect groups the context menu of the tree view is used.

2.3.15.1 Example

Select the first SubDevice of the group and *Create Hot connect group* in the context menu:

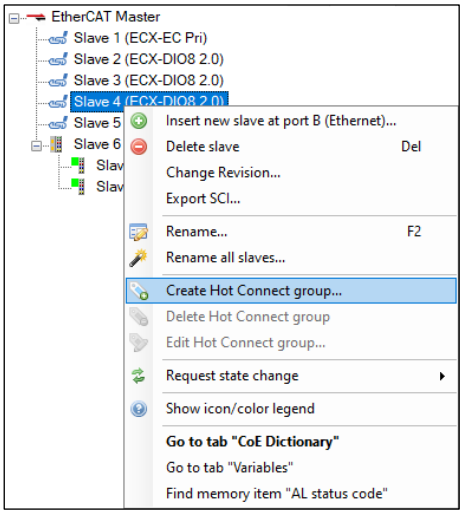


Figure 33: Creating new Hot Connect group.

The Hot Connect group editor window will show up:

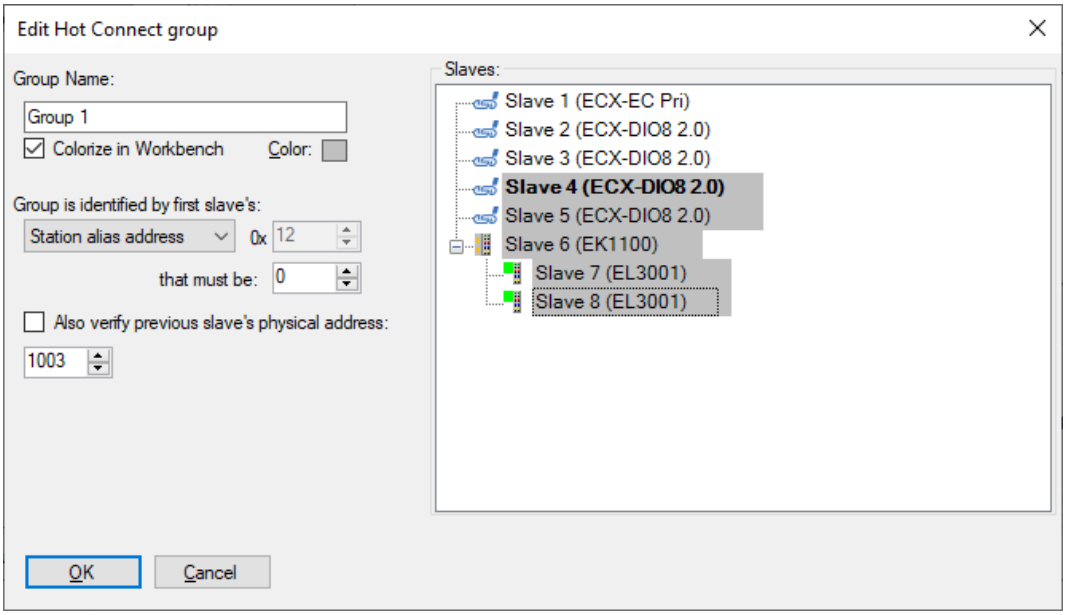


Figure 34: Hot Connect group editor

Group Name / Colorize in Workbench Only used for display and not exported to ENI.

Group is identified by first SubDevice's:
A register and a value (*that must be:*) is selected here.
(*Station alias address* = ESC Register 0x0012)
Additionally, the previous SubDevice physical address might be verified, too, to allow the Hot Connect group to be connected at a certain SubDevice only.

SubDevices: Last SubDevice of the group is selected here.

In this example the group shall be identified by the EK1100's Station Alias that shall be 2000 and includes all its boxes. (Make sure the Station Alias Address is actually used: it's stored in the

SubDevice's EEPROM – check *Configure alias addresses [...]* in the Workbench's *Tools* menu for an editor)
Now the ENI can be exported – nothing more to do in the Workbench.

You might check cyclic frames for the EtherCAT commands regarding this group:

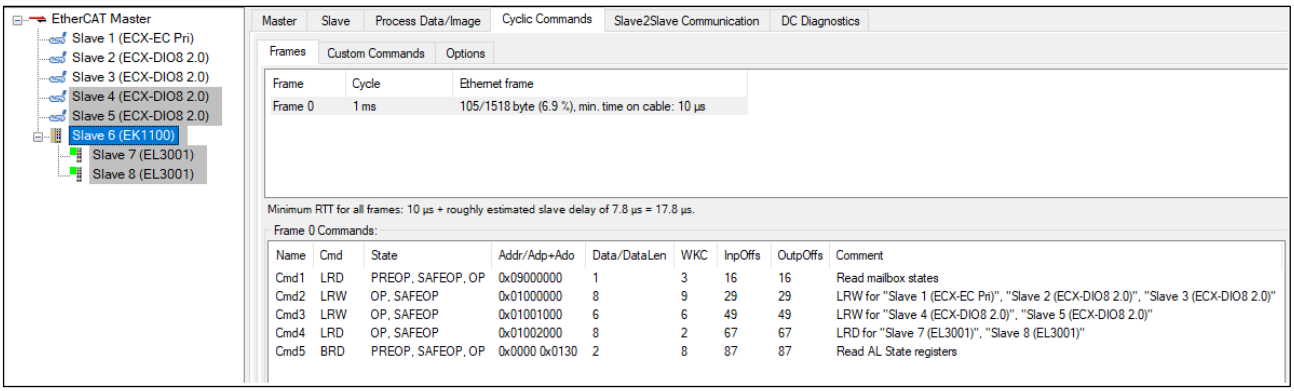


Figure 35: Tab *Frames* showing separate commands for Hot connect group

Process data for the Hot Connect group's SubDevices are access by "Cmd3"

2.3.15.2 Remarks

- Hot Connect is primarily an EtherCAT MDevice feature – the Workbench just exports these settings as described by ETG documents.
- Make sure created groups are allowed: the group editor will show warnings/hints, but not all errors can be checked, especially: No group must overlap another / a SubDevice can't be part of multiple groups, removing a group must not interrupt other SubDevice's connections.

2.3.16 SCI Export

SCI is an abbreviation for SubDevice Configuration Information and a means to create pe-configured device descriptions derived from an ESI device description.
SubDevices without ESI file (scanned SubDevices, italic name in the SubDevice tree) can't be SCI exported!

- Carry out the custom SubDevice configuration, like PDOs, DC mode selection, CoE init commands,
  CoE dictionary creation, assigning modules.

- From the MDevice/SubDevice tree view context menu on a SubDevice select *Export SCI...*:

- First confirm in a dialog window to create or omit the CoE dictionary.

- Then enter a name for the SCI device in the pop-up window

- In the following save file selection system dialog *Export SCI,* select a path and enter a file name (This can be different from the SCI device name, the extension must be .SCI).
  If you want to use the SCI within the EtherCAT Workbench it must be saved in the device library location (typically `C:\Program Files (x86)\esd\EtherCAT\EtherCAT Workbench\SubDeviceLibrary\SCI\`).

- In the SubDevice library device tree of the EtherCAT Workbench (opened for adding devices) the SCI device name is displayed instead of the original device name and the SCI device is grouped below the original device it was derived from.

2.4 Cyclic Commands

The *Frames* page shows an overview of all frames/commands that will be sent cyclically by the MDevice.

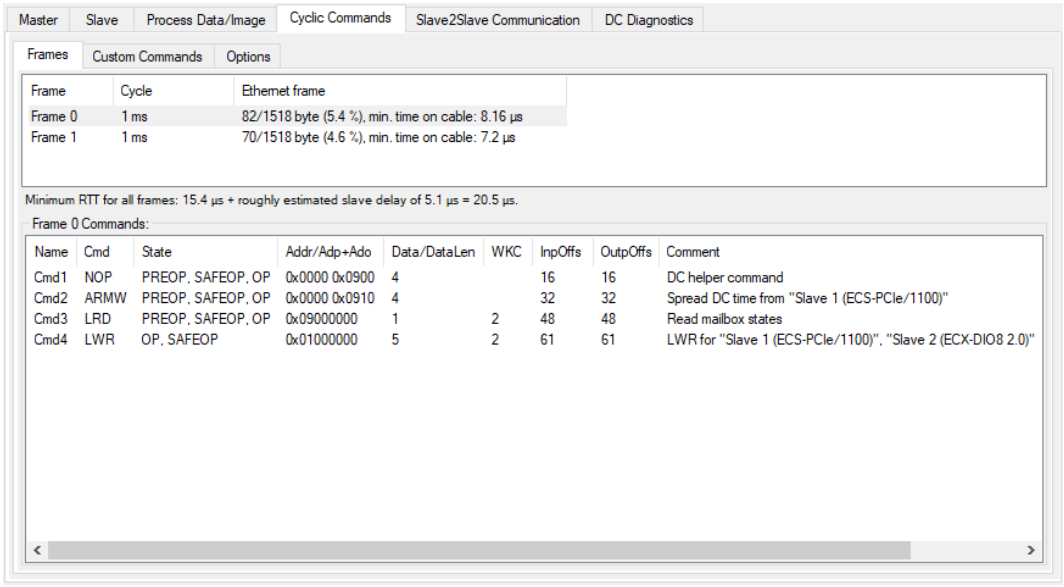


Figure 36: Tab Page *Cyclic Commands*

Page *Custom Commands*
This page allows adding custom cyclic command.

Automatic: Adds commands simply by checking these options.
Checkboxes:

- *Broadcast read AL state registers*
  Adds a “BRD” command to read all SubDevice’s 0x0130 registers.
  (The resulting WKC might be used to determine the number of active SubDevices)
- *Commands for distributed clocks*
  Adds several commands for distributed clocks handling.
  (Automatically checked/unchecked, can be used only to temporary override that automatism)
- *Broadcast read DC System Time Difference*
  Adds a “BRD” command to read all SubDevice’s 0x092c register. (Result can be used to determine whether all SubDevices are still synchronized)

User defined commands:
Allows adding custom commands.
Use the context menu to edit the list. Via the context menu you can edit, duplicate, append, delete or move commands in the list
A command editor will be shown when a command is added or edited:

- Variables within a custom command:
  The command editor also allows defining process variables within the command. These variables are expected to reside *back to back* within the command’s data – no further offset, gaps etc. can be configured.

- Example: Adding a process variable *DC System Time* via a custom command
  1. Select *Append command* from the *User defined commands* context menu in the command editor window that is shown:
  2. Change the command type via *Command:* from “NOP” to “APRD”.
  3. Set the addresses: SubDevice: 0x0000 (the first SubDevice), Physical: 0x0910 (the *DC System Time* register)
  4. Set data length to ‘8’ (assuming the SubDevice supports 64-bit DC time stamps).
  5. Use *Process variables* list’s context menu *Append item* to add a new variable. A variable editor window will be shown, there:
  6. Set “BitLen” to 64, “DataType” to ULINT and name to “DC Time” for example.
  7. Close the dialog windows with “OK”
  8. The *Process Data/Image* page should now contain the new variable, when in SafeOp/Op the command is sent, and the variable can be read.

Page *Options*

Max. commands per frame

Determines the maximum number of EtherCAT commands per frame. Default is 15, which should not need to be modified. A value higher than 15 will make some of the virtual variables unusable. see 3.4.1

2.5 SubDevice2SubDevice Communication

This tab page allows to map input variables/PDOs to output variables/PDOs.

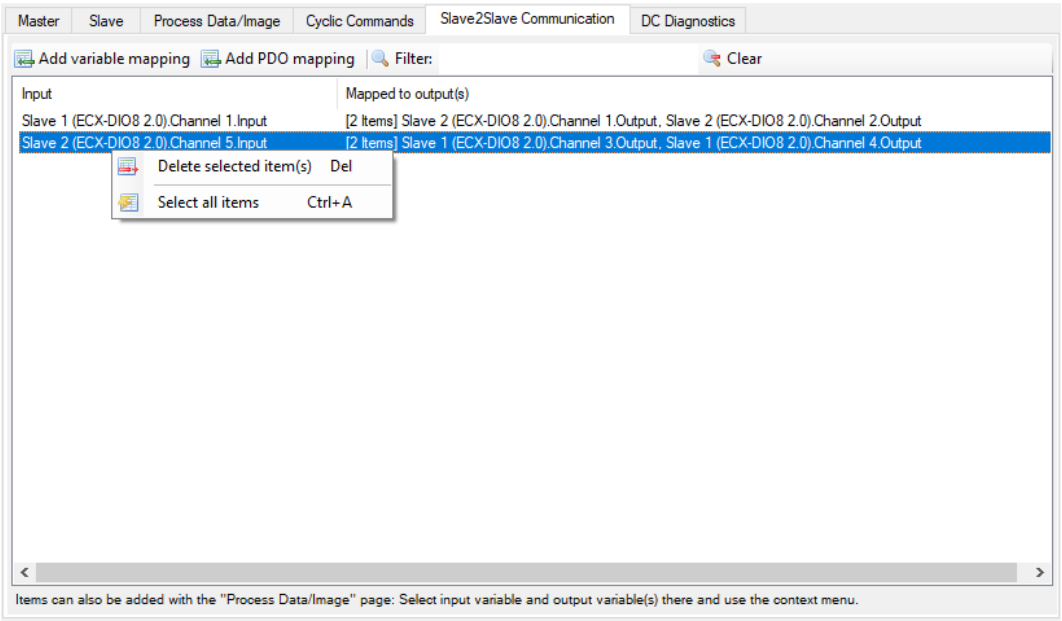


Figure 37: SubDevice2SubDevice Communication: Current mappings

The copy operation is performed by the MDevice after the EtherCAT command to handle the inputs successfully returned – according to “<CopyInfos>” described in ETG.2100 document.
An input can be mapped to multiple outputs, an output can be mapped to a single input only (as only the last copy operation would be “visible”).

Additionally, it’s only possible to map variables to variables or complete PDOs to complete PDOs, both of same size – so it’s not possible, for example, to map an 8-bit input variable to a 16-bit output variable, etc.
To add a mapping use the buttons “Add variable mapping” and “Add PDO mapping”, a dialog similar to Figure 38 will show up:

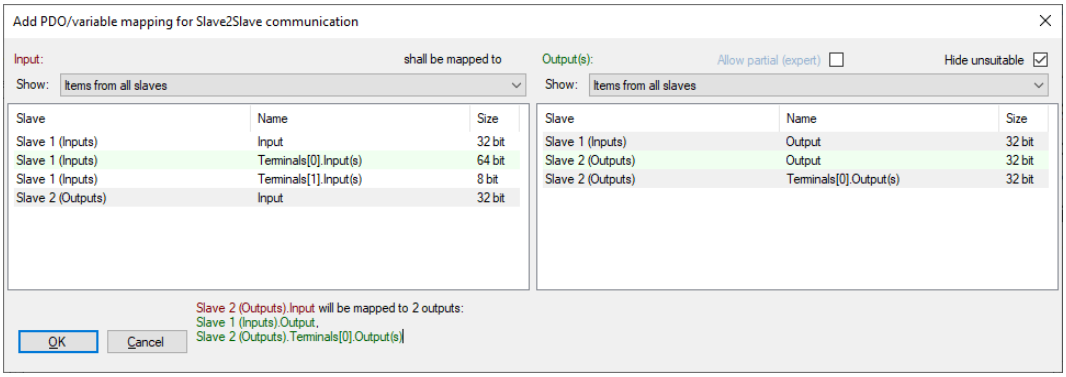


Figure 38: Adding a variable mapping

On the left side the input is selected, on the right side the output(s). The drop-down list *Show:* on each side allows to show only items from a certain SubDevice. Unless *Hide unsuitable* is unchecked only outputs that are not yet mapped and fit the selected input are shown there.
(To select multiple outputs, hold Shift/Control while selecting.)
The lower pane lists the selections and shows what is added when “OK” is clicked.

Checkbox

Allow partial (expert) A source Tx-PDO is copied in sections with offset to the selected multiple smaller destination Rx-PDOs.

Normal copy example	
Dialog selection left → right	SubDevice2SubDevice copy commands
TXPDO1(32 bit) RXPDO1(32 bit), RXPDO2 (32 bit)	TXPDO1(bit 0-31) → RXPDO1 (bit 0-31)
	TXPDO1(bit 0-31) → RXPDO2 (bit 0-31)

Allow partial (expert)” example	
Dialog selection left → right	SubDevice2SubDevice copy commands
TXPDO1(32 bit) → RXPDO1(16 bit), RXPDO2(16 bit)	TXPDO1(bit 0-16) → RXPDO1 (bit 0-15)
	TXPDO1(bit 17-31 bit) → RXPDO2 (bit 0-15)

2.6 Network Topology

Opens a window that shows a graphical overview of the EtherCAT network topology.

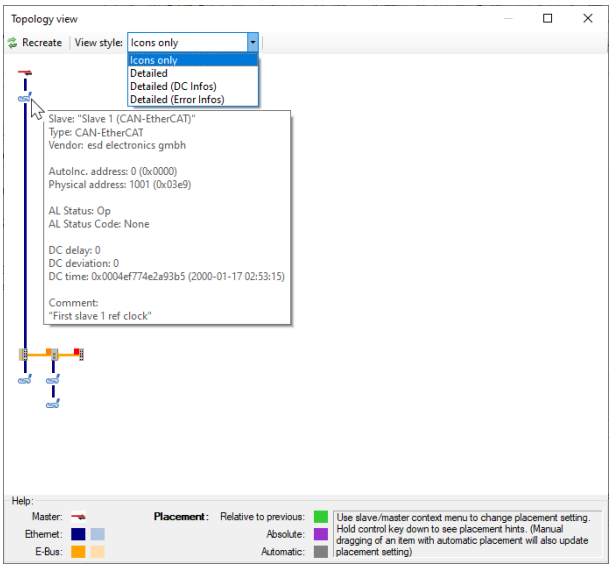


Figure 39: Topology, view style *Icons*

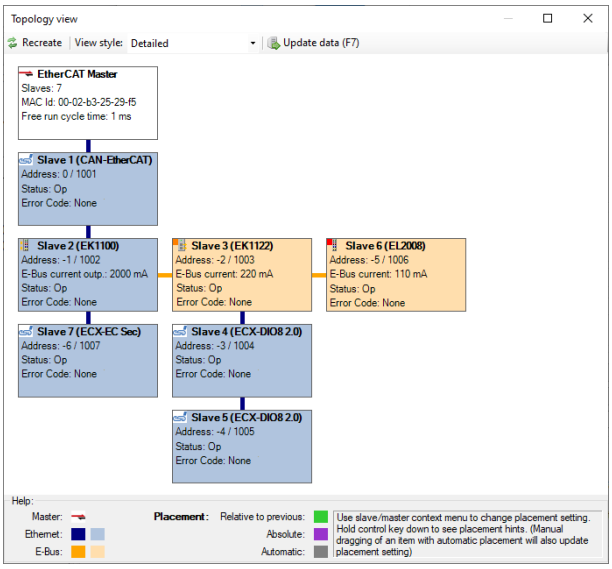


Figure 40: Topology, view style *Detailed*

Rules for SubDevice placement
The SubDevice placement follows a simple rule: Each Ethernet SubDevice is aligned below the previous SubDevice. Each EBUS SubDevice is aligned right to the previous SubDevice.

Context Menus
Via the context menu (no SubDevice selected), you can select:

- *Reset all placements to "Auto"*
- *Recreate topology*

Via the context menu (SubDevice selected, offline), you can select:

- *Placement (on recreate)*
  and in the submenu:
  - *Auto*
  - *Manual/ Relative*
  - *Manual/Absolute*
- *Reset all placements to "Auto"*
- *Recreate topology*
- *Select in main window and close topology*

Manual options
To customize the automatic layout each item (MDevice or SubDevice) has a placement setting (accessible by its context menu) and can be moved manually with the mouse. (This was done for Figure 40)

Auto (Default)	Placed either below or next to previous SubDevice as described above.
Manual/Relative	Placed relative to the previous SubDevice.
Manual/Absolute	Keeps its current absolute position when display is recreated.

To reset all items to “Auto” use the context menu entry *Reset all placements to “Auto”*. When an item whose placement setting is “Auto” is moved with the mouse, its setting is automatically changed to *Manual/Relative*. To see each SubDevice’s placement setting hold down the Control key.
To update/recreate the display according to the placement settings the button *Recreate* or the “F5” key can be used.

View Style

Icons Only: Only small images as in the SubDevice tree view. See Figure 39 above

Detailed: Large text boxes with some general information within see Figure 40

- *Detailed (DC Infos)*:
  with ESC DC register values (0x0910/0x0928/0x092c)
- *Detailed (Error Infos)*:
  with ESC Error register values (0x0300 ... 0x0313)

When changing between *Icons Only* view and another any manual placement setting should be restarted.

Update Data (F7)
Used to update online values (i.e. re reading register values in *Detailed* view styles etc.)

INFORMATION
Not all SubDevices support all registers – usually the read access will fail, but some SubDevices might e.g. always return “0” in the error registers.

If a *Redundant adapter* in the *MDevice → General* tab page is assigned, a second MDevice icon is displayed that *symbolizes* the redundant cable connection (no second MDevice instance is involved).

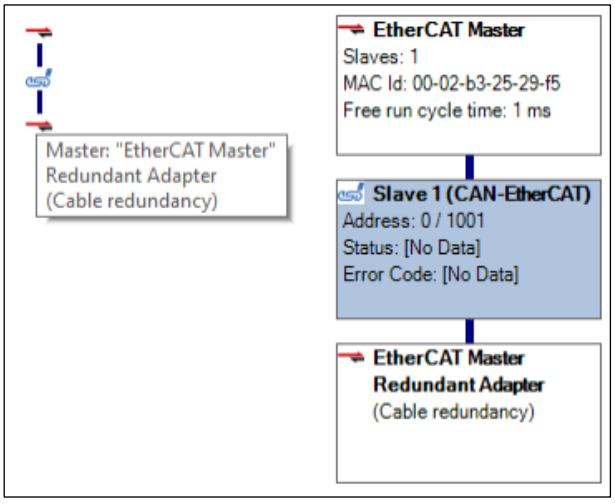


Figure 41: Topology, view style *Icons*

3 Online Configuration

This chapter describes the online configuration of an EtherCAT SubDevice network either connected to a local network interface of the host PC the EtherCAT Workbench runs on or to an esd EtherCAT MDevice which supports a remote connection.

3.1 Connection to the EtherCAT MDevice

Required settings to establish the connection are configured by the *General* tab page.

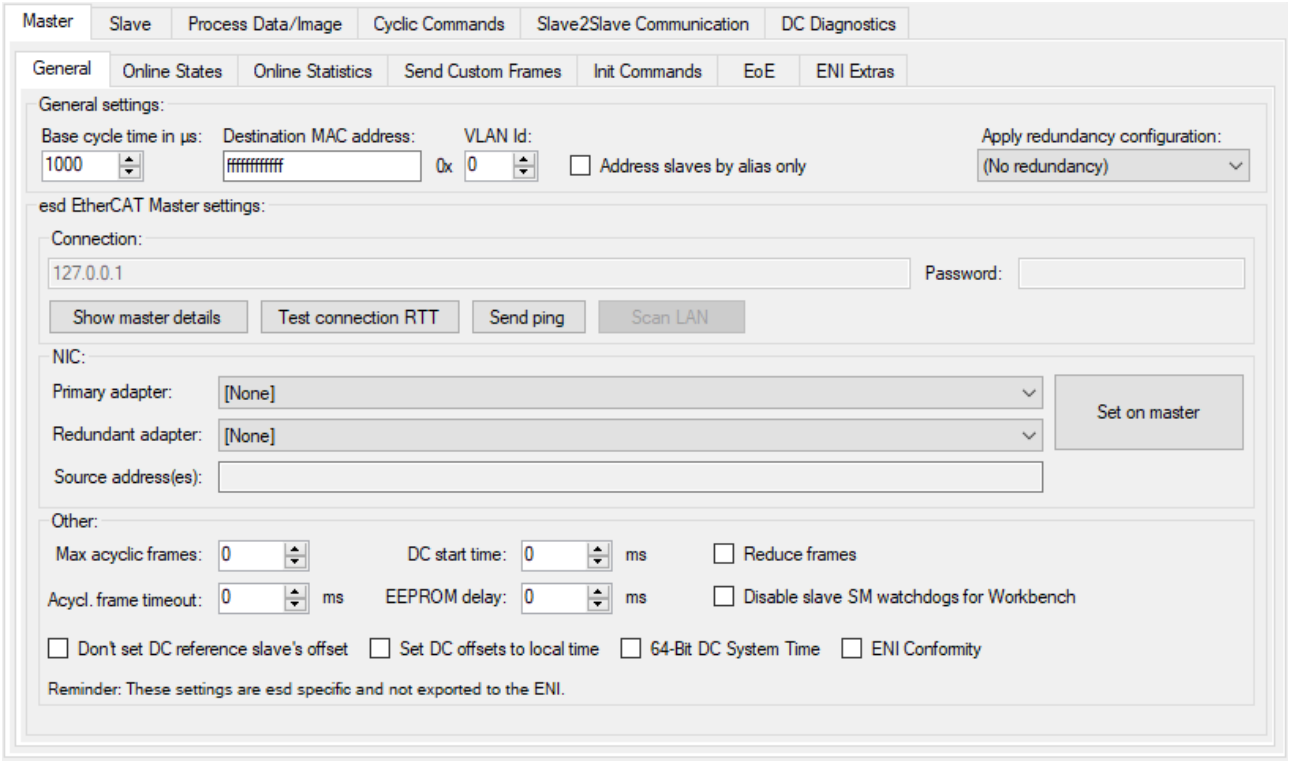


Figure 42: MDevice configuration, page *General*

The *Connection:* text field determines the protocol, IP address and port of the MDevice. Usually, it is on the local computer, and the default does not need to be changed. When it's running on a remote machine with a different IP address it should be entered here[8]. Connection is started by the *Online* button on the main form.

The various states of the *Online* button and the other buttons on the online toolbar are described in detail below:

Online Toolbar

Online toolbar: offline after program start

"uac shield overlay icon" - This symbol indicates that a Windows UAC request may follow.

8 Please refer to the EtherCAT MDevice manual/READMEs for details about starting the MDevice on that remote target.

However, this is only the case if the local MDevice service has to be started in order to provide the online functions. After pressing the button, the symbol is no longer displayed. If you establish a connection to a remote MDevice, no UAC is requested, although the icon is displayed.

In the global settings this behavior can be changed with the setting “HandleLocalMDeviceService”.
See 5.2

When the connection is established, the button will be “down” or highlighted to signal this. Clicking the button again will disconnect the application from the EtherCAT MDevice.

Online toobar: Online, before adapter *Set on MDevice*

When the connection is established a network interface on the MDevice has to be selected. This is done in the *NIC:* pane on the *Settings* page of the MDevice. (As this choice will be saved within the project file this is usually done only once, and the user is automatically asked to do this if it’s not yet done). Select a primary adapter (no need for Redundant adapter for now) and confirm with *Set on MDevice*.

After *Set on MDevice* the NIC the network is ready, and the MDevice is in *configuration mode* which means:

- All SubDevices in state “PreOp”.
- Process data access, “SafeOp”/“PreOp”, etc. **not** possible in this mode.
- A SubDevice scan can be performed.
See 3.2

For process data access, etc. the MDevice has to be set to *Free run mode.*
See 3.3

If no data are available, you get the message: *Set on MDevice failed: No data available.*
This means that the MDevice receives no data from any EtherCAT SubDevice. This is usually caused by selecting the wrong network interface. Also check the (remote) computer's security system settings.
Continue with SubDevice *Scan* (by default automatically performed) and *Free run*. This will be described in more detail below.

Buttons

- *Online*
  Connect/Disconnect to/from the EtherCAT MDevice.
- *Scan*
  Start a SubDevice scan (when online).
see 3.2
- Free run
  Switch MDevice to free run mode (to read process data, etc.).
see 3.3
- The label *Monitor Mode* might be displayed right from the buttons.

If you want to test EtherCAT cable redundancy you may select a *Redundant adapter*. You will then also be prompted to select a redundancy configuration in the *Apply redundancy configuration* drop down menu.

INFORMATION
Please note: The *Redundant adapter* is not part of the EtherCAT Network Information (ENI) Specification. This means, that when you run the MDevice standalone (outside the Workbench online configuration access described here), you must specify the *Redundant adapter* on the MDevice startup command line.

3.2 SubDevice Scan

Used to populate/update the SubDevice tree view. Started by clicking *Scan* in the online toolbar of the main form. Only possible in configuration mode. When started, the net will be scanned and a dialog window that shows the current and the newly scanned network opens:

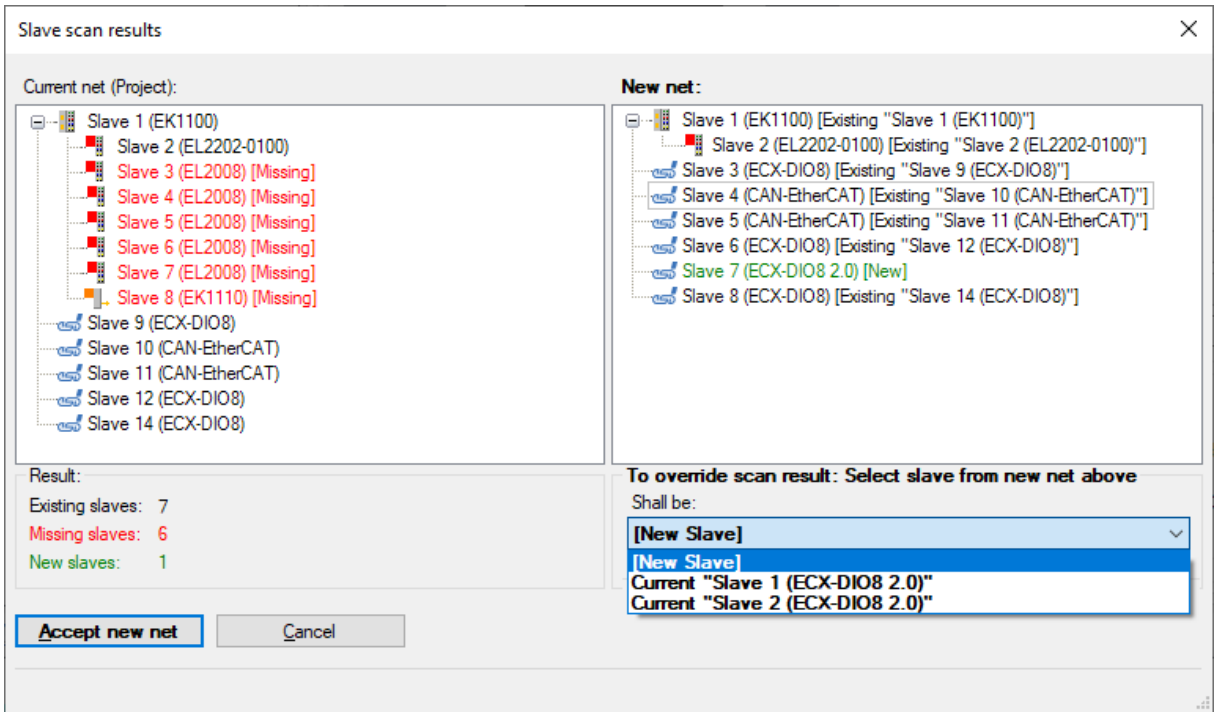


Figure 43: SubDevice scan results

Selecting *Cancel* will close the window without making any changes to the current network. *Accept new net* will replace the current net with the new one, while all “existing” SubDevice’s settings (Excluding their names) are kept.

A SubDevice is identified by its vendor id, product code, revision number and serial number. But many SubDevices have its serial number set to “0”, also a SubDevice might have been replaced by another one with another serial number or even with a different revision number. Therefore the “new”/“existing” distinction might be inaccurate, and a manual selection is possible, too:
In the lower pane of the *New net* a drop down list will show all current SubDevices.
Select the new SubDevice in the tree view above and then decide which of the current SubDevices it is (or “[New SubDevice]” if it shall not replace an existing one) by this drop-down list.

It’s strongly recommended to do a SubDevice scan every time the MDevice connection is established. Most commands require the SubDevice address: when the network has changed and the application does not know about it, the address might have changed, too, therefore e.g. writing a SubDevice’s EEPROM in this situation can lead to writing it to a different SubDevice!

Use the global settings (see 5.2) setting *ScanOnline* to adapt the scan behaviour to *always* or *ask* (default) when you go *Online*.

3.3 Changing to Free Run Mode

The free run mode is entered and left with the *Free run* button in the online toolbar. In free run mode the ENI has been transferred to the MDevice and the MDevice sends all configured initialization commands, cyclic frames, etc. Therefore, all SubDevices should be in operational state and reading/writing process data is possible, too.

Common problems that might occur when changing to free run mode:

- *SubDevice error event*

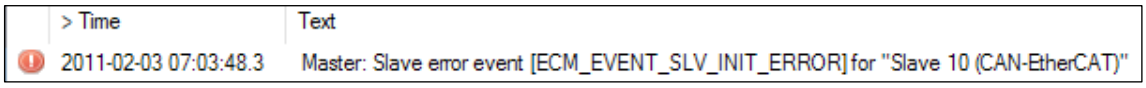


Figure 44: *SubDevice error event* (Example)

- ECM_EVENT_SLV_ID_ERROR
  SubDevice Id verification failed, usually caused by a SubDevice exchange.
- ECM_EVENT_SLV_INIT_ERROR
  General initialization error. When no other messages occur,
  checking SubDevice error registers might help to find the cause. See 2.3.5
- ECM_EVENT_SLV_NOT_PRESENT
  SubDevice not found, determined by reading SubDevice's status registers.
- SM error

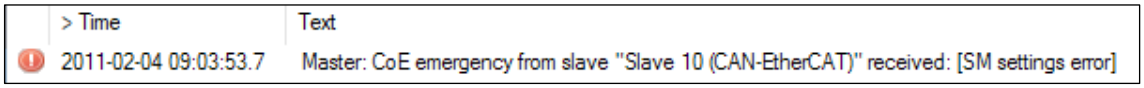


Figure 45: *SubDevice SM settings error* (Example)

- SM errors usually occur when the PDO mapping is invalid. This again often happens when the available PDOs are created dynamically. The device might have changed its PDOs (e.g. just by a reset) and the Workbench wants to configure another PDO assignment: *SM settings error* will be the result.
- SM address error / SM length error might occur when the PDOs are too large, check the configured size (SubDevice's *Process data* → *Assignment* Tab) and make sure this will fit into the SM buffers (See SubDevice's ESI, <Sm> node, *StartAddress* attribute, and make sure SMs don't overlap etc. – the Workbench will warn about that, too).

NOTICE
Note that changing to *Free run* involves a complete SubDevice initialization phase starting at EtherCAT Init state. Therefore, many changes, e.g. in the CoE dictionary, will be reset too. (Unless init cmds were added so these changing are stored in the ENI and automatically retransmitted)

3.4 Process Data

This tab shows the process data image, which consists of all SubDevice's process variables and some additional virtual variables.

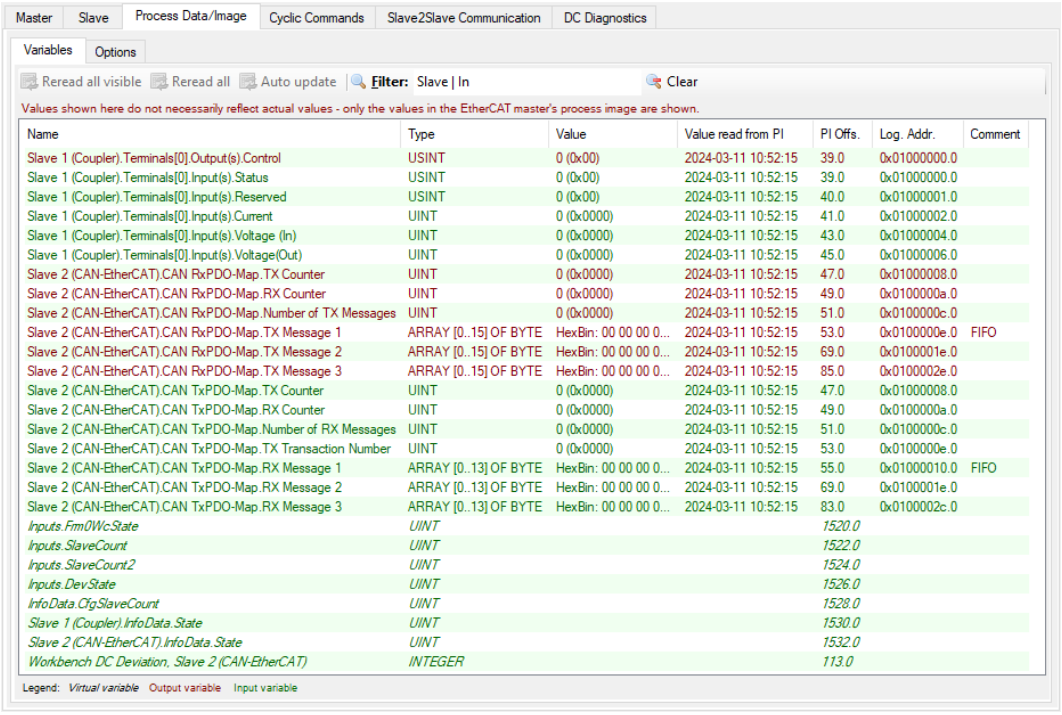


Figure 46: Tab page *Process Data/Image* page *Variables*

It allows reading/writing process variables as well as monitoring variables. All variables can be read by the *Reread* buttons or by context menu. Output variables can be written/edited by double clicking an item with the mouse or by context menu. The variable editor is described in the following section 3.4.2.

INFORMATION
All variables are only written to / read from the MDevice's process image. This means any input read was not necessarily updated by SubDevice and an output written is not necessarily successfully transferred to SubDevice, too.
Additionally, the outputs are only transferred to MDevice after editing, i.e. when the MDevice resets/overwrites the values the Workbench does not re-transfer the value.

Context menu on *Process Data/Image* page *Variables*

- *Edit/view value*
  Opens the Variable editor See 3.4.2
- *Reread selected items*
  Reads the selected items/variables from the EtherCAT MDevice's process data image, *Free run* mode is required.
- *Monitor selected variable(s)*
  To monitor a variable, select *Monitor selected variable(s)* from the context menu. (The number of simultaneously active variable monitors is currently limited to 16.)
- *Copy selected name(s) to clipboard*
  Copies the names of the selected items to the clipboard.
  This can be used for MDevice code programming or as filter text, for example.
- *Create mapping entry for SubDevice2SubDevice communication*
  Additionally mapping entries for SubDevice2SubDevice Communication can be created here: select one input variable and one or more output variables and use

the context menu entry: *Create mapping entry for SubDevice2SubDevice Communication*. (Variables must be of same size) See 2.5

Toolbar/Buttons

Reread all visible	Read = Update all visible process variable items in the list from the process image → see *Filter.*
Reread all	Read = Update all process variable items in the list from the process image.
Auto update	Cyclic automatic update of the visible items.
Filter	Enter a filter text to filter visible items by name. The filter may contain the operators "|" for OR – or - "&" for AND, for example *Inputs & Count*" *SubDevice 1*|*SubDevice 2.*
Clear	Clears the filter text, all items are restored.

Page *Options*

Combine PD commands

Determines whether the EtherCAT commands to handle SubDevice process data shall be combined when possible. When enabled, multiple SubDevice's process data is handled by a single EtherCAT command – achieved by choosing the logical addresses accordingly.
To exclude only single SubDevices from this, see SubDevice process data options. See 2.3.10

Process images contain command header/WKC

Whether space for the EtherCAT command header and WKC shall be reserved in the process images, i.e. for a command to handle process variables that resides at offset `x` this means:

- When checked: Variables start at offset `x + 10` (10 byte for EtherCAT command header)
- When unchecked: Variables start at directly at offset `x`

Make sure this setting matches the offsets that are expected by the EtherCAT MDevice you are using! (Although current ETG.2100 document (V1.0.0) seems to describe this as the "checked" case, some MDevice vendors handle this differently[9].)

First command's alignment / First command's offset

The process image offset of the first EtherCAT command in a frame will be a multiple of this value plus offset.

Logical start address

First logical address for first PDO.

Mailbox states	Logical address mapping for mailbox states.
Log. address align	Alignment for logical addresses for PDOs.

Buttons

Set values to Frame layout	Sets the process data options to default values for process images that contain whole Ethernet frames.
Set values to packed layout	Sets the process data options to default values for process images that contain only process data.

9 The Workbench adds an <PILayout> element with attribute "IncludeCmdHeader" to the <VendorSpecific> element of the ENI's <MDevice> element to distinguish this.

3.4.1 Virtual Variables

- “Inputs.Frm*X*WcState”:
  - Bit 0 set: First EtherCAT command in cyclic frame *X* with wrong WKC,
    Bit 1 set: Second EtherCAT command… Bit 15 set: Complete frame failed.
- “Inputs.SubDeviceCount”
  - Number of active SubDevices, determined by reading “AL state” registers.
    (Enabled in “Options” page of “Cyclic Commands” tab), See 2.4
- “Inputs.SubDeviceCount2”
  - Number of active SubDevices at redundant network interface
- “Inputs.DevState”:
  - Current MDevice state flags (Also shown with more details in MDevice tab “Online states”, See 2.1.2
    - Bit 0 set: Link lost on primary NIC
    - Bit 2 set: Link lost on redundant NIC
    - Bit 3 set: Cyclic frame lost
    - Bit 4 set: Out of send resources
    - Bit 5 set: Watchdog triggered
    - Bit 6 set: Error initializing network adapter
    - Bit 8 set: At least one SubDevice in state “Init”
    - Bit 9 set: At least one SubDevice in state “PreOp”
    - Bit 10 set: At least one SubDevice in state “SafeOp”
    - Bit 11 set: At least one SubDevice indicates an error
    - Bit 12 set: DC not in sync
- “InfoData.CfgSubDeviceCount”
  - Total number of SubDevices in ENI
- “*SubDevice*.InfoData.State”:
  - Bit 0 to 3: SubDevice state
    - 0: Unknown, 1: “Init”, 2: “PreOp”, 3: “Bootstrap”, 4: “SafeOp”, 8: “Op”
  - Bit 5: Id verification error
  - Bit 6: Initialization error
  - Bit 8: SubDevice not present
  - Bit 9: Link error
  - Bit 10: Missing link
  - Bit 11: Unexpected link
  - Bit 12/13/14/15: Communication at Port A/B/C/D

3.4.2 Variable Editor

This dialog is used to edit most variables, e.g. process variables, CoE items or SubDevice register values.

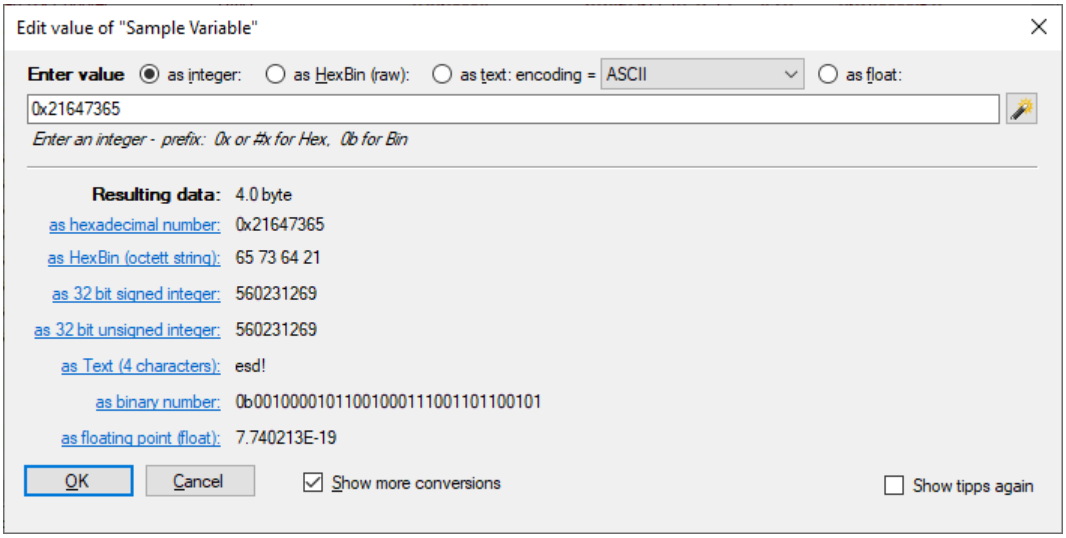


Figure 47: *Variable editor* window with *Show more conversion*

The output of the editor is always a fixed number of data bits depending on the edited variable's size (size is displayed e.g. as "1.2 byte", which means one byte plus two bits). The editor allows editing these bits as different data types while the result can be display in different interpretations, too. The resulting bits are always displayed at *as HexBin*. The check box *Show more conversions* selects whether more interpretations of the resulting bits should be shown or not.

Input types

- "as HexBin" (raw)
  - Input is interpreted as byte array entered in hexadecimal number, e.g.: "aa bb cc" will be interpreted as three bytes with the decimal values 170, 187 and 204.
- "as integer"
  - Input is interpreted as an integral number, positive or negative.
  - Input can be prefixed with "0x" for hexadecimal numbers, e.g. "0x10" for decimal "16" or "0b" for binary numbers, e.g. "0b100" for decimal "4".
- "as text"
  - Input is interpreted as string according to selected encoding, e.g. "esd" will be interpreted as 3-byte array "65 73 64" (HexBin) with ASCII encoding.
  - (For Unicode no special code page handling is available, it just uses the Windows/.NET defaults).
- "as float"
  - Input is interpreted as floating point number. e.g. "1.234".
  - As the application sets its locale to "US English" the decimal separator should be a dot ("."), regardless of system settings.
  - For conversion to the resulting data bits only 32- and 64-bit IEEE variables will lead to correct results.
- "from enum"
  - The radio button becomes only visible on a variable with enum type.
  - Input is interpreted as integral number, but it is selected from a drop-down list which displays a name/description for certain numbers.
  - Only available when the edited variable offers such information.
- "  -Button" (magic wand)
  - Opens a context menu for user defined inputs / *Favorites*. Initially it will only contain an item *Edit favorites*: by this an .xml file that contains these entries is

shown. If the file does not exist a sample file is created that shows how own entries can be added.

Conversions (below)

- All interpretations work with the resulting “HexBin” bytes and assume a “little-endian” system.
- The signed integer display always assumes the most significant bit as the sign bit.

To easily switch back and forth between different input modes according to different conversions and interpretations, click on the underlined blue conversion type name on the left.

3.4.3 Variable Monitor

A plotter-like window used to monitor a variable’s value: y axis represents the value, x axis the time. Range, scale etc. is widely configurable. Input is received as plain data whose interpretation can be configured, too.

All visible monitor windows and their settings are saved with the project. A monitor window is linked to the monitored variable’s name, i.e. changing the variable name (e.g. by changing the SubDevice name) will lead to a monitor window that is never updated again. In that case the window has to be closed, and a new monitor must be created. (Changing the monitor window title does not affect the variable name)

INFORMATION
Variable Monitors are updated only twice a second. The process data transfer from MDevice to the Workbench is also limited – so don't expect a thorough variable timeline.

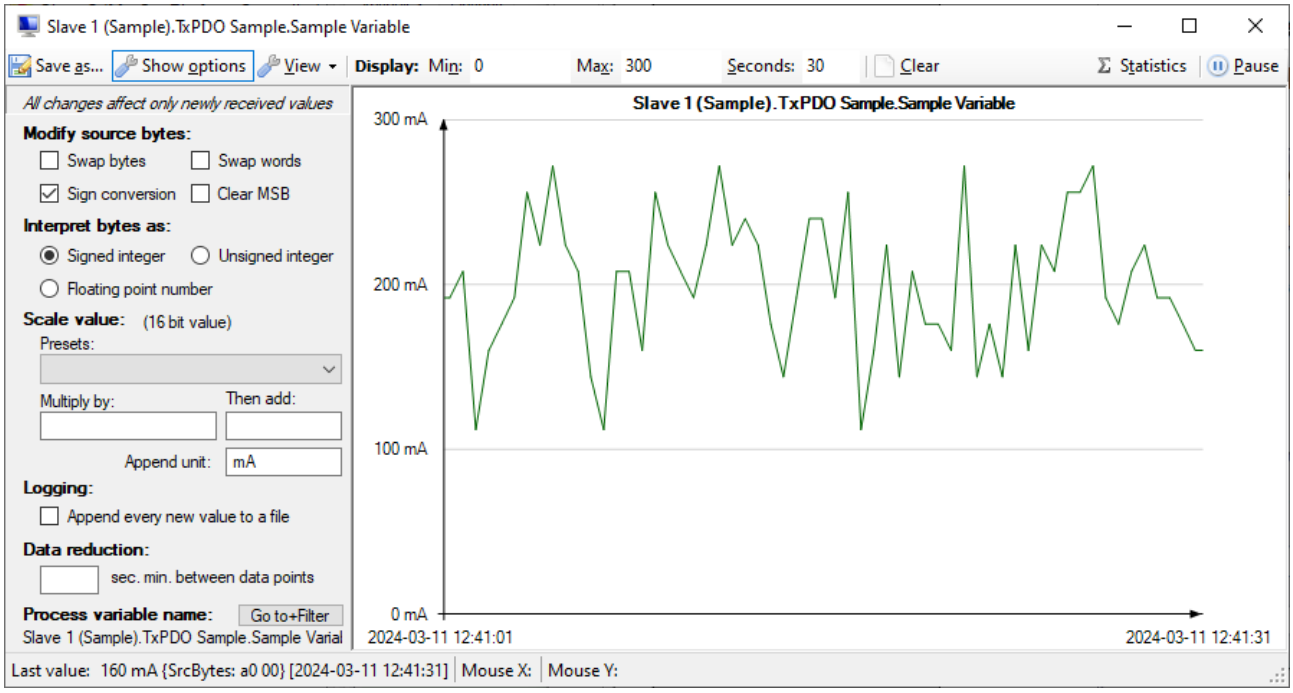


Figure 48: variable monitor window, options visible

Toolbar/Buttons

Save as… Shows a file dialog to save the currently displayed image to a bitmap file (image from the moment the button is clicked is used, regardless of updates while selecting the file).

Show options	Shows or hides the options pane, see description below.
View	Drop down list • *Change title* Shows a text input dialog (*Enter new window title*) to enter a new text for the image and window title. • *Draw data points* Determines whether a small cross is drawn at each data point or not. • *Stepped lines* Determines whether the lines connect the data points directly (when unchecked) or only by horizontal and vertical lines (when checked). *Stepped lines* is especially suited for “BOOL”, “BIT1” and value enumerations. For BOOL and BIT1 this setting is automatically selected. • *Disable anti-aliasing* Determines whether anti-aliasing for all lines should be disabled or not. With *anti-aliasing* the plot line is wider and smother.
Display:	• *Min / Max* Selects the lower and upper limit of the display. Leave empty for automatic selection. **Hint:** Click left on the label *Min* or *Max* label to reset the upper/lower limit to empty = automatic **Hint:** Click right on a plot point to open a context menu on this point and use its Y-value as *Min* or *Max*: • *Seconds* Determines the display width, minimum is “1.0” (one second), maximum is “2592000.0” (30 days). (Should be used only for short periods, as drawing is not optimized for thousands of data points – log to file should be used instead then. Left/right mouse button on the label to increment/decrement by 60 seconds.)
Clear	To remove all acquired data points. (Each data point is automatically removed after it has left the display.)
Statistics	Button to show some information about current data points. (Min./Max./Average etc.)
Pause	To temporarily freeze the display. (Automatically unchecked if window is changed)

Options pane

The options pane is only visible on the left side of the program window when button *Show options* is down. This allows logging and modifying the data before it is displayed.

Modify source bytes	Options to modify the source bytes before interpreting them.
Interpret bytes as	Determines how the source bytes should be interpreted. (Usually, a proper default is set by source PDO.)
Scale value	Options to scale the value (done after source modification and interpretation): • *Presets:* Allows to select predefined values for the following three items:

- *Multiply by:*
  Value is multiplied by the entered value before it's displayed
- *Then add:*
  Entered value is added to the variable's value before it's displayed (done after any multiplication)
- *Append unit:*
  This text is added to the y axis values in the display, no other effect

Logging

Appends each new data point as <*Date*><*Tab*><*Value*> line to a text file. File selection dialog is shown when this is checked, and the selected file is not cleared

- <Date>
  will be the date formatted as defined in the global date format setting ("Menu items" for information about how to change it) See 1.8.1
- <Tab>
  is a tab character (decimal 9)
- <Value>
  is the value that is displayed (not the source bytes etc.)
  - Example line:
    `2010-12-16 12:22:551.2345`

Data reduction

- Allows reducing the number of data points by just ignoring some *sec. min. between data points* (seconds minimum between data points)
- When a value is entered, a new data point is only created when the last data point is older than this. (Data points are just omitted, no averaging etc.)

Process variable name

Shows the name of the process variable from tab *Process Data/Image* where the monitor was opened from. If you do not change the monitor title (see above) the name is the same as the title of the monitor window.

Go to+Filter

Go to tab *Process Data/Image* and set a filter text for the variable the monitor window was opened for.

Status bar

Displays the last received date/value pair and the date/value that the point under the mouse pointer belongs to.

3.5 Send Custom Frames

This tab allows sending arbitrary EtherCAT commands while online.

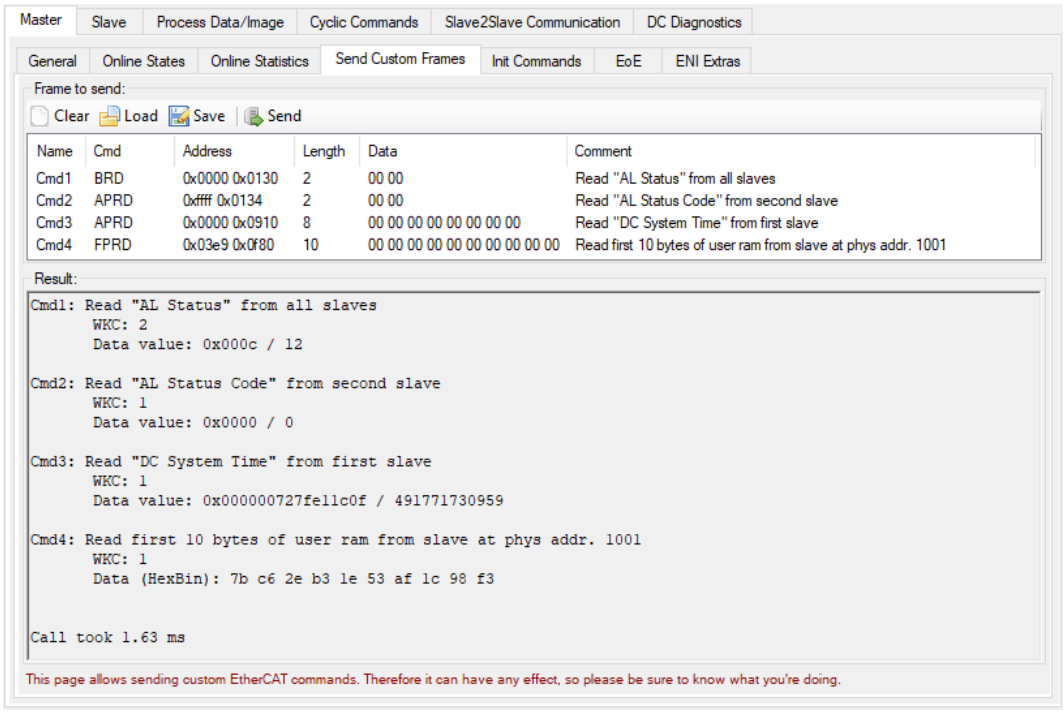


Figure 49: Sample custom frame with result

These commands are not checked in any way, so make sure you understand the side effects of the commands you send.

Use the context menu of the command list to add/remove commands etc.
With the toolbar buttons the complete command list can be saved/read from a file or cleared, the *Send* button sends this frame.
After sending the frame the *Result* text field shows each command's working counter and data.
If the commands do not fit into a single frame, they're automatically split into multiple frames.

3.6 DC Diagnostics

3.6.1 Deviation

This page helps to monitor the DC deviation (ESC register 0x092c: “System Time Difference”).

Figure 50:*DC Diagnostics*, tab page *Deviation*

Buttons

Add cmd to globally read DC deviation
: Adds EtherCAT command “BRD” to cyclic commands and creates a process variable for its data (keep in mind that BRD *OR*s all SubDevice values into a single value).

Add cmd to read deviation from specific SubDevices
: Shows a dialog to select SubDevice(s), then adds EtherCAT command APRD to cyclic commands and creates process variable(s) for them.

Open variable monitor for global deviation /
Open variable monitor for SubDevice deviations
: Opens a variable monitor window for the according process variable – just a shortcut for doing this in the tab.

Remove commands/monitor windows
: Closes the variable monitor windows and removes all hereby added EtherCAT commands.

The EtherCAT commands added by this tab will also be exported to ENI, so it’s advisable to remove them before exporting the ENI, to avoid increasing the frame size unnecessarily.

3.6.2 SYNC Generation

This page allows to setup certain SubDevice DC SYNC units while online. It requires the SubDevice SYNC units assigned to EtherCAT and their clocks should be synchronized.

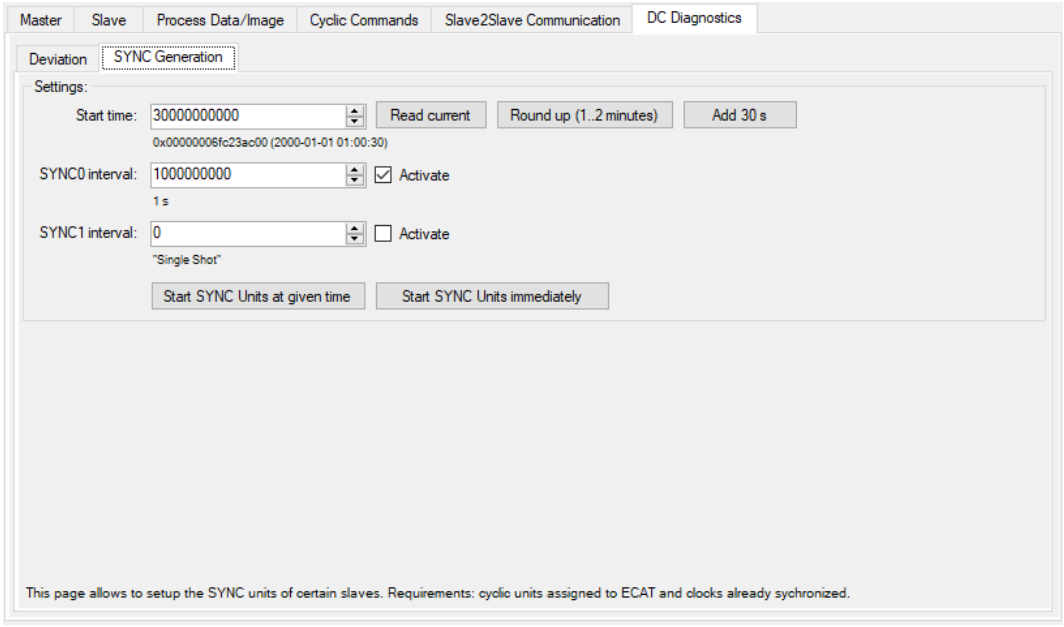


Figure 51:DC Diagnostics, tab page *SYNC Generation*

Affected registers

- *Start time:* Written to reg. 0x0990
- *SYNC0 interval*: Written to reg. 0x09a0
- *SYNC1 interval*: Written to reg. 0x09a4
- *SYNC0/SYNC1 Activate*: Written to reg. 0x0981
  - Set to 0x00 before other values are written

Buttons

Start SYNC Units at given time

First a dialog to select the SubDevice(s) whose SYNC units shall be edited will be shown. Then the values entered here will be written to the according SubDevice registers so they will generate SYNC signals at the desired interval. (This includes automatic generation of frames to deactivate the SYNC units before writing the values, etc.)
As the start time must not be in the past it can be read from the SubDevices (button *Read current*) and certain values can be added automatically by *Round up ...* and *Add 30 s* buttons. (This is intended only for SubDevices with 64-bit DC timestamps.)

Start SYNC Units immediately

Similar to *Start SYNC Units at given time*, but here the entered start time value is ignored. Instead, it's read from the SubDevices, and a small amount is automatically added so the SYNC units start almost immediately.
(This works with 32-bit DC timestamps as well.)

3.6.3 Device Specifc Online Functions

3.6.3.1 CoE Tool

For SubDevice devices supporting CoE the tab page Device Specific offers a CoE Tool which allows read and write access for CoE objects. If the CoE tool is not displayed on the Device Specific tab.

See 2.3.13

.

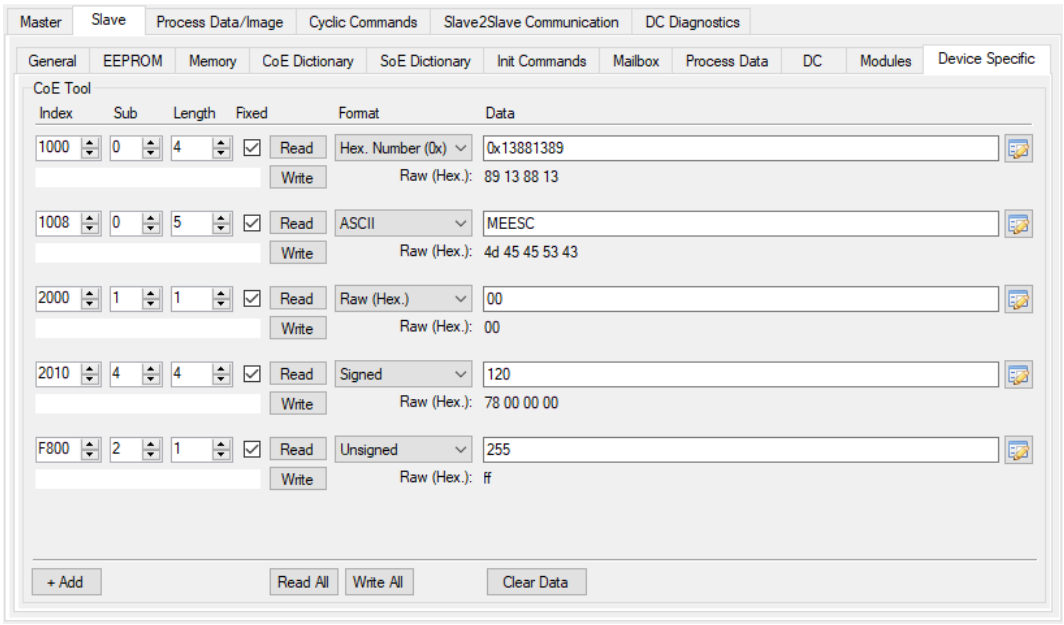


Figure 52:Device Specific CoE Tool

- Enter the *Index* and *Sub* index.
- The *Length* is automatically adapted while you enter data, unless you check the *Fixed* checkbox, which fixes the length while entering data.
- For write access enter the write *Length* or click the up and down buttons. For read access the length is automatically set on read.
- Under *Format* select a basic input and display format for the *Data*.
- For write access enter the write *Data* or use the edit button on the right which opens the value editor.
- With read access the data read is transferred to the data field and formatted as selected under *Format*.

See 2.3.2

The *Raw (Hex.):* field always shows a hexadecimal representation of the read/write data byte-by-byte.

The CoE tool object settings keep unchanged when you click on another SubDevice. So you can read the same objects from the other SubDevice.
The CoE tool object settings are temporary only and not saved in the project file or elsewhere and are lost when you close the Workbench!

Buttons

+Add.	Add more entries for objects with this button.
Read All	Read all objects from top to bottom.
Write All	Write all objects from top to bottom.
Clear Data	Clear all *Data* fields.

3.6.3.2 AoE Commands

AoE (“Automation Device Specification over EtherCAT) is an EtherCAT mailbox protocol which allows to tunnel access to underlying devices.
In the tab page *SubDevice → Mailbox → AoE* commands can be executed online, and *Free run* mode is also required.

- Enter the SubDevice and MDevice *NetId* in the fields *SubDevice/MDevice AoEConfig* each 6 bytes decimal separated with a dot.
- Enter the command data in the hexadecimal input / spin boxes as required.
- Select a command next to *AoE Command*.
  You can select *AoE Commands*: *Read, Write, Read Write, Read Device Info, Read State, Write Control.*
- Enter *Write-Data* for *Write* or *Read Write* commands. The *Write-Length* is automatically adapted while you enter hexadecimal data.
- Press button *Send command*.
- For read access *Read data* and *Read-String* is filled in with the data read from the device. For write access the *Write Data* is sent to the device.

3.6.3.3 VoE Commands

VoE (Vendor specific protocol over EtherCAT) is a vendor specific EtherCAT mailbox protocol without a specific header or protocol. There are write and read services.
In the tab page *SubDevice → Mailbox → VoE* commands can be executed online.

- Fill in the hexadecimal input fields *Vendor ID* and *Vendor Type* as needed.
  Write Length will be adjusted while editing the hexadecimal *Write-Data*.
  Press button *Write* or *Read*.

3.7 Monitor Mode

This mode is determined by the MDevice the Workbench connects to. If the MDevice is running in monitor mode it is driven by its own application – the Workbench is used for monitoring only. Connecting to a MDevice running in monitor mode requires the correct Workbench project file to be loaded, i.e. the MDevice's ENI must have been created by the Workbench and exactly that project file has to be used.

Figure 53: *Monitor mode label*

Monitor mode is recognizable by the [Monitor mode] label next to the *Free run* button, see Figure 53. In monitor mode the *Free run* mode is always active, as the MDevice already runs by its ENI and the Workbench is not used to create one anymore. Therefore, any change to the project that affects the ENI even renders the project incompatible to that running MDevice. (To avoid using different process image layouts for example)

INFORMATION
Next to each exported ENI the according Workbench project file "*_autosave.eew" is also saved – this project can be used as fallback if the original project was accidentally modified to be incompatible.

Generally, the MDevice application is in control in monitor mode, i.e. any change by the Workbench might be overruled by the MDevice or trigger any other action depending on the application. (Especially writing process variables that are also updated cyclically by the MDevice application might have no effect etc.)

4 EtherCAT Workbench Scripting

4.1 Overview

The EtherCAT Workbench includes IronPython [10] (Version 2.7.12, based on Python 2) to allow you to automate some Workbench tasks or to customize the Workbench.

Any ".py" file in the Workbench's *Scripts* sub folder[11] will be shown and can be started/stopped by the Dropdown menu *Scripting*. Any number of scripts can run simultaneously.
For a quick start just have a look at the *Scripts/Samples* folder.

4.2 Limitations

Currently only a limited number of Workbench elements are available to the scripts: it was designed for accessing process variables, creating the SubDevice tree and accessing complete EEPROMs and CoE items – when you miss something, feel free to send any suggestions to esd. The IronPython support is focused on using the classes offered by the Workbench, see *Available variables etc.*. below. Using other python libraries might need a separate python installation and/or adding library paths to your scripts. See also section 4.5.

4.3 Dropdown Menu *Scripting*

The dropdown menu of the *Scripting* button shows the installed scripts and allows to start/stop them. When it is not available the EtherCAT Workbench was installed without scripting support (or some required files are missing).

When a stopped script is clicked, it is started, its symbol is changed, the entry is displayed bold, and its font is changed.

When a running script is clicked it is stopped. Stopping sets the variable "eewScriptRun" inside the script to "false" (see also section 4.4.1). When the script does not end on this within 3 seconds it's terminated, i.e. its thread (each script is started in its own thread) is aborted. (Under some circumstances it's not possible to abort the thread, so make sure your scripts can be ended)

10 IronPython is an implementation of the Python programming language targeting the .NET Framework – http://en.wikipedia.org/wiki/IronPython. Python is a general-purpose, high-level programming language whose design philosophy emphasizes code readability – http://en.wikipedia.org/wiki/Python

11 With the exception of files inside the "lib" folder there.

4.4 Available Variables etc.

4.4.1 Variables

The EtherCAT Workbench **injects** the following variables into the python scripts:

Type	Variable	Comment
EtherCATWorkbenchScript	eew	The main object that offers almost all functionality, see 4.4.2
bool	eewScriptRun	Initially set to “true”, set to “false” when script is stopped (i.e. shall stop itself).
bool	eewVersionIsDebug	“true” for Debug versions of the EtherCAT Workbench.
string	eewVersion	EtherCAT Workbench Version number, e.g. “1.3.2”.
string	eewScriptsArgs	Custom value given by command line, see 4.6

4.4.2 Classes

4.4.2.1 EtherCATWorkbenchScript

Created by Workbench itself, offered to scripts as “eew” variable. CallbackResult’s *Data* member is only as described when its *ErrorCode* member is *Success*, else it’s “null”/“None” or a string with some more information about the error.

Result	Method	Arguments	Description
void	ScriptAbort	(string errText)	Ends the script immediately. (Aborts the thread) Example: div
void	ScriptSleep	(int ms)	Waits inside the script. (ms = milliseconds) Example: div
void	SetStatusText	(string txt)	Each script may have its own status text, that is a text box shown in the Workbench’s online toolbar. This method sets this text and creates the text box the first time it is called. Example: Running LED.py
void	SetWarningText	(string txt)	Each script may have its own warning text, that is a blinking label shown in the Workbench’s online toolbar. This method sets this text and creates the label the first time it is called. Label is released when called with “None” as “txt”. Example: EEPROM write with verify.py
void	AddLogInfo	(string txt)	Adds an “Info” entry to the Workbench’s messages log. Example: div
void	AddLogWarning	(string txt)	Adds a “Warning” entry to the Workbench’s messages log. Example: div
void	AddLogError	(string txt)	Adds an “Error” entry to the Workbench’s messages log. Example: div
System.Windows.Forms.DialogResult	ShowModalMessageBox	(string text, string caption, System.Windows.Forms.MessageBoxButtons buttons, System.Windows.Forms.MessageBoxIcon icon, System.Windows.Forms.MessageBoxDefaultButton defaultButton)	Calls .NET framework's System.Windows.Forms.MessageBox.Show() with the arguments passed and the EtherCAT Workbench window as window parent. Example: eewHelper.py

bool	IsOnlineBusy	()	Returns “True” when the Workbench’s online buttons are disabled, i.e. an asynchronous action is still active. Example: lib\eewHelper.py
MasterMode	GetMasterMode	()	Returns the current MDevice state – does **not** necessarily reflect the actual EtherCAT state. (PreOp, Op, etc.) Example: Connect to MDevice, go to free run.py
CallbackResult	VarRequestUpdate (*1)	(string varName)	Triggers a reread of an input variable. (Equals rereading variable in the Workbench’s “Process Data/Image” tab) Result’s *Data* is: null. Example: GetVariableList.py
CallbackResult	VarRequestUpdateAll (*1)	(bool visibleOnly)	Triggers a reread of all (visible) input variables. (Equals “Reread all visible” , with visibleOnly = true, or “Reread all” in the Workbench’s “Process Data/Image” tab) Result’s *Data* is: null. Example: GetVariableList.py
CallbackResult	VarGetDataType	(string varName)	Looks up a variable’s data type. (In the Workbench’s “Process Data/Image” tab) Result’s *Data* is: string, e.g. “BOOL”, “UINT”, etc. Example: RunningLed.py
CallbackResult	VarGetBitSize	(string varName)	Looks up a variable’s size in bit. (In the Workbench’s “Process Data/Image” tab) Result’s *Data* is: int. Example: -
CallbackResult	VarRequestWrite (*1)	(string varName, byte[] dataBytes)	Triggers writing new data to an output variable. See line below for more convenient alternative methods. Result’s *Data* is: null. Example: -
CallbackResult	VarRequestWrite*XX* (*1)	(string varName, *yy* data)	Alternative to VarRequestWrite *XX* can be: "UI8", "UI16", "UI32", "UI64", "I8", "I16", "I32", "I64", "Float", "Double", "String" *yy* is the according variable type Result’s *Data* is: null. Example: RunningLed.py
CallbackResult	VarGetValue	(string varName)	Reads variable’s current data (From the Workbench’s “Process Data/Image” tab) – see line below for more convenient alternative methods. (Use VarRequestUpdate() to update it first) Result’s *Data* is: byte[]. Example: GetVariableList.py
CallbackResult	VarGetValue*XX*	(string varName)	Alternative to VarGetValue *XX* can be: "UI8", "UI16", "UI32", "UI64", "I8", "I16", "I32", "I64", "Float", "Double", "String" Result’s *Data* is: According to *XX*. Example: -

CallbackResult	VarGetList	(int SlaveAddr,
		DirectionSel ,
		bool bIncludeValues)
			SubDeviceAddr is the auto inc. address of the SubDevice
			eew.DirectionSel .All | .Input | .Output
			bIncludeValues: determines if the current value of the variable is included
			
			Result’s *Data* is: array [] of VarInfo:
			--- 	
			string Name
			int SubDeviceAddr
			EtherCATWorkbenchScript.DirectionSel
			string DataType = piEntry.DataType;
			int BitSize = piEntry.BitLen;
			string valueString
			string valueHexBytes
			--- 	
			Example: GetVariableList.py

CallbackResult	LoadProject (*1)	(string fileName)	Loads a Workbench project from file. Omits the “are you sure” dialog when current project is unsaved. Result’s *Data* is: null. Example: -

CallbackResult	NewProject (*1)	()	Starts a new Workbench project from file. Omits the “are you sure” dialog when current project is unsaved. Result’s *Data* is: null. Example: Connect to MDevice, go to free run.py
CallbackResult	RequestOnlineMode (*1)	(bool online)	Requests to go online or offline. (Equals button “Online”) Result’s *Data* is: null. Example: Connect to MDevice, go to free run.py
CallbackResult	SlaveSelectByAddr	(int autoIncAddr)	Selects a SubDevice in the current SubDevice list by its auto increment address. Result’s *Data* is: null. Example: GetVariableList.py
CallbackResult	SlaveSelectByName	(string name)	Selects a SubDevice in the current SubDevice list by its name. Result’s *Data* is: null. Example: GetVariableList.py
CallbackResult	GetSelectedSlaveAddr	()	Returns the auto increment address of the currently selected SubDevice.
CallbackResult	SlaveEEPROMEditorLoadFromFile	(string fileName)	Loads an EEPROM from file to editor. (Affects the SubDevice that is currently selected. Equals button “From file” in the SubDevice EEPROM editor tab) Result’s *Data* is: null Example: EEPROM write with verify.py
CallbackResult	SlaveEEPROMEditorSaveToFile	(string fileName)	Saves EEPROM from editor to file. (For the SubDevice that is currently selected. Equals button “To file” in the SubDevice EEPROM editor tab) Result’s *Data* is: null. Example: EEPROM write with verify.py
CallbackResult	SlaveEEPROMEditorRequestWrite (*1)	()	Requests writing the current EEPROM to SubDevice. (For the SubDevice that is currently selected. Equals button “To device” in the SubDevice EEPROM editor tab) Result’s *Data* is: null. Example: EEPROM write with verify.py
CallbackResult	SlaveEEPROMEditorRequestRead (*1)	()	Requests reading the current EEPROM from SubDevice. (For the SubDevice that is currently selected. Equals button “From device” in the SubDevice EEPROM editor tab) Result’s *Data* is: null. Example: EEPROM write with verify.py
CallbackResult	SlaveEEPROMEditorGetMD5	()	Gets the MD5 checksum from the current EEPROM editor data. (For the SubDevice that is currently selected) Result’s *Data* is: string (md5 sum). Example: SubDeviceEEPROMEditorGetMD5
CallbackResult	SlaveEEPROMEditorClear	()	Clears current EEPROM editor data. (For the SubDevice that is currently selected) Result’s *Data* is: null. Example: SubDeviceEEPROMEditorClear
CallbackResult	CloseWorkbench	(bool force)	Triggers closing the Workbench. When current project is unsaved an “are you sure” dialog will occur first – unless “force” is set to “true” Result’s *Data* is: null. Example: CreateSubDeviceTreeAndExportENI.py
CallbackResult	RequestFreerunMode (*1)	(bool freeRun)	Triggers free run mode. (Equals button “Free run”) Result’s *Data* is: null. Example: lib\eewHelper.py
CallbackResult	SetMasterConnectionString	(string data)	Sets the text in the “Connection:” text box within the “MDevice/General” tab. Result’s *Data* is: null. Example: Connect to MDevice, go to free run.py

CallbackResult	SetMasterNICs	(string primary, string redundant)	Sets the NIC(s) the MDevice shall use for EtherCAT. When only one NIC exists this is not necessary – else it's needed to avoid user input becoming necessary. Selection is done by MAC-Id. Result's *Data* is: null. Example: Connect to MDevice, go to free run.py
CallbackResult	ApplyRedundancyConfiguration	(RedundancyConfiguration)	Sets the redundancy configuration (see MDevice → General → Apply redundancy configuration) None, Optimized, Compatibility Result's *Data* is: null. Example: "Connect to MDevice, go to free run.py"
CallbackResult	AddCustomButton	(string caption, string tooltip)	Adds a custom button the Workbench's online toolbar. Result's *Data* is: int (button handle – required for the other button methods). Example: Custom buttons.py
CallbackResult	UpdateCustomButton	(int btnHandle, string caption, string tooltip)	Updates caption and tooltip of an existing custom button. Result's *Data* is: null. Example: -
CallbackResult	GetCustomButtonClicked	(int btnHandle)	Checks if a custom button was clicked. **Note**: When a button is clicked it's disabled until the script calls this method. Result's *Data* is: bool (button was clicked). Example: Custom buttons.py
CallbackResult	RemoveCustomButton	(int btnHandle)	Removes a custom button from the Workbench's online toolbar. (When a script ends/is terminated all its buttons are removed automatically) Result's *Data* is: null. Example: -
CallbackResult	AppendRxPDO/AppendTxPDO	(int SlaveAddr, int index, string name, bool flagMandatory = false, bool flagFixed = false, bool flagVirtual = false, bool flagExclude = false)	Adds an Rx or Tx PDO to the respective PDO lists (see SubDevice → Process Data) SubDeviceAddr is the auto inc. address of the SubDevice. flag* are the respective PDO flags Set the PDO content with SetSubDeviceProp(PDOContent) afterwards. Result's *Data* is: null. Example: Append PDO.py
CallbackResult	CoEPDORecreateFromESI	()	For the SubDevice that is currently selected: Recreates the PDO lists from ESI (see SubDevice → Process Data → Recreate PDO lists → From ESI) Result's *Data* is: null. Example: -
CallbackResult	CoEPDORecreateFromEEPROM	()	For the SubDevice that is currently selected: Recreates the PDO lists from EEPROM data (see SubDevice → Process Data → Recreate PDO lists → From EEPROM) Result's *Data* is: null. Example: -
CallbackResult	CoEDictionaryRecreateFromESI	()	For the SubDevice that is currently selected: Recreates the CoE dictionary from ESI (see SubDevice → CoE Dictionary → Recreate dict. → From ESI) Result's *Data* is: null. Example: GetVariableList.py

CallbackResult	DCRecreateFromESI	()	For the SubDevice that is currently selected: Recreates the DC settings from ESI (see SubDevice → DC → Recreate DC options → From ESI) Result's *Data* is: null. Example: ModulesSlotsDC.py
CallbackResult	DCRecreateFromEEPROM	()	For the SubDevice that is currently selected: Recreates the DC settings from EEPROM (see SubDevice → DC → Recreate DC options → From EEPROM) Result's *Data* is: null. Example: ModulesSlotsDC.py
CallbackResult	DCRecreateEmpty	()	For the SubDevice that is currently selected: Removes the DC settings (see SubDevice → DC → Recreate DC options → No DC) Result's *Data* is: null. Example: ModulesSlotsDC.py
CallbackResult	MslGetSlots	(int SlaveAutoIncrAddr)	Gets an array with info on all available slots. SubDeviceAutoIncrAddr is the auto inc. address of the SubDevice. Result's *Data* is: array [] of SlotInfo: --- uint SlotIndex uint TypeSlotGroup uint InstanceIndex uint IndexWithinSlotGroup string TypeName string SlotTitle uint TypeMinInstances uint TypeInstances uint[] AcceptedModuleIdents string[] AcceptedModuleClasses --- Example: ModulesSlotsDC.py
CallbackResult	MslListAllModules	(int SlaveAutoIncrAddr)	Gets an array with info on all available modules. SubDeviceAutoIncrAddr is the auto inc. address of the SubDevice. Result's *Data* is: array [] of ModuleInfo: --- uint ModuleIdent string ModuleTitle string ModuleName string ModuleDescription string ModuleClass string ModuleClassDisplayName uint SubDeviceVendorId --- Example: ModulesSlotsDC.py
CallbackResult	MslListModules	(int SlaveAutoIncrAddr, uint[] ModuleIdentList, string[] ModuleClasses)	Gets an array with info on the selected modules. Module selection is by module ident or module class. SubDeviceAutoIncrAddr is the auto inc. address of the SubDevice. Result's *Data* is: array [] of ModuleInfo: see MslListAllModules Example: ModulesSlotsDC.py

CallbackResult	MslGetSlotAssignment	(int SlaveAutoIncrAddr)	Gets an array with module idents. The array position corresponds to the slot position. The value at an array position is the assigned module ident, or zero for empty/unassigned slots. SubDeviceAutoIncrAddr is the auto inc. address of the SubDevice. Result's *Data* is: array [] of UInt32 Example: ModulesSlotsDC.py
CallbackResult	MslSetSlotAssignment	(int SlaveAutoIncrAddr, uint[] moduleIdentList, bool updateEsiPDOs = true, bool updateEsiCoEDict = true)	Sets the slot assignment of modules by a array of module idents. The array position corresponds to the slot position. The value at an array position is the module to be assigned or zero for an unassigned slot. SubDeviceAutoIncrAddr is the auto inc. address of the SubDevice. UpdateEsiPDOs / updateEsiCoEDict: update PDO lists and/or CoEDictionary Result's *Data* is: null. Example: ModulesSlotsDC.py
CallbackResult	MslClearSlotAssignment	(int SlaveAutoIncrAddr, bool updateEsiPDOs = true, bool updateEsiCoEDict = true)	Resets the slot assignment to all slots unassigned. SubDeviceAutoIncrAddr is the auto inc. address of the SubDevice. UpdateEsiPDOs / updateEsiCoEDict: update PDO lists and/or CoEDictionary Result's *Data* is: null. Example: ModulesSlotsDC.py
CallbackResult	MslAssignModule	(int SlaveAutoIncrAddr, uint? moduleIdent, int slotIndex, bool updateEsiPDOs = true, bool updateEsiCoEDict = true)	Assigns a single module to a single slot by module ident and slot position = slot index (zero based). SubDeviceAutoIncrAddr is the auto inc. address of the SubDevice. UpdateEsiPDOs / updateEsiCoEDict: update PDO lists and/or CoEDictionary Result's *Data* is: null. Example: ModulesSlotsDC.py
CallbackResult	MslUnAssignModule	(int SlaveAutoIncrAddr, int slotIndex, bool updateEsiPDOs = true, bool updateEsiCoEDict = true)	Removes a module from a slot selected by slot position = slot index (zero based). SubDeviceAutoIncrAddr is the auto inc. address of the SubDevice. UpdateEsiPDOs / updateEsiCoEDict: update PDO lists and/or CoEDictionary Result's *Data* is: null. Example: ModulesSlotsDC.py
CallbackResult	MslGetModuleSetNames	(int SlaveAutoIncrAddr)	Gets an array of module set names. (Module sets are predefined in the ESI.) SubDeviceAutoIncrAddr is the auto inc. address of the SubDevice. UpdateEsiPDOs / updateEsiCoEDict: update PDO lists and/or CoEDictionary Result's *Data* is: array [] of string Example: ModulesSlotsDC.py

CallbackResult	MslGetModuleSet	(int SlaveAutoIncrAddr, int nSetNo)	Get the modules of a module set. (First get the names and the number of module sets by "MslGetModuleSetNames") SubDeviceAutoIncrAddr is the auto inc. address of the SubDevice. UpdateEsiPDOs / updateEsiCoEDict: update PDO lists and/or CoEDictionary nsetNo is the zero based index of the module set. Result's *Data* is: array [] of UInt32 Example: ModulesSlotsDC.py
CallbackResult	MslRequestSlotAssignment	(int SlaveAutoIncrAddr)	Requests the module assignment online from the SubDevice (see SubDevice → Modules → Assign Module Set → Online from SubDevice) SubDeviceAutoIncrAddr is the auto inc. address of the SubDevice. UpdateEsiPDOs / updateEsiCoEDict: update PDO lists and/or CoEDictionary Result's *Data* is: null. Example: ModulesSlotsDC.py
CallbackResult	RequestSlaveScan (*1)	()	Triggers a SubDevice scan. (Following "SubDevice scan results" window is closed automatically when no SubDevices existed yet) Result's *Data* is: null. Example: Connect to MDevice, go to free run.py
CallbackResult	CoERequestPDORead (*1)	()	Triggers updating the list of available PDOs. (For the SubDevice that is currently selected. Equals "Recreate list of avail. PDOs → Online..." in the SubDevice Process Data tab) Result's *Data* is: null. Example: -
CallbackResult	CoERequestDictionaryRead (*1)	()	Triggers recreating the CoE dictionary from online data. (For the SubDevice that is currently selected. Equals "Recreate dict. → Online..." in the SubDevice CoE Dictionary tab) Result's *Data* is: null. Example: CoE Access.py
CallbackResult	CoERequestObjectRead (*1)	(ushort objIndex, byte subIdx)	Triggers rereading a CoE object. (For the SubDevice that is currently selected. Equals "Reread..." in the SubDevice CoE Dictionary tab) Result's *Data* is: null. Example: lib\eewHelper.py
CallbackResult	CoERequestObjectWrite (*1)	(ushort objIndex, byte subIdx, byte[] data)	Triggers writing a CoE object. (data in little endian order) (For the SubDevice that is currently selected. Equals editing the object in the SubDevice CoE Dictionary tab) Result's *Data* is: null. Example: lib\eewHelper.py
CallbackResult	CoEGetObjectValue	(ushort objIndex, byte subIdx)	Reads the value that is currently available in the selected SubDevice's CoE Dictionary tab. (Use CoERequestObjectRead() to update it first) Result's *Data* is: byte[]. Example: CoEGetObjectValue
CallbackResult	CoESetFilterText	(string text)	Sets the filter text in the CoE Dictionary tab page. Result's *Data* is: null. Example: -
CallbackResult	CoEGetObjectName	(ushort objIndex)	For the SubDevice that is currently selected: Get the name of the CoE dictionary object with Index "objIndex".
CallbackResult	CoEGetObjectName	(ushort objIndex, byte subIdx)	For the SubDevice that is currently selected: Get the name of the CoE dictionary object with Index "objIndex" and "subIdx", the name of the subindex entry.

CallbackResult	ENIExport	(string fileName)	Exports the ENI file. Result's *Data* is: null. Example: CreateSubDeviceTreeAndExportENI.py
CallbackResult	AddSlave	(int? prevSlave, int? prevPort, uint vendorId, uint prodCode, uint? revNo)	Adds a SubDevice to the tree view. prevSubDevice is the auto inc. address of the prev. SubDevice where it shall be added, if null it's added as last SubDevice. prevPort is the EtherCAT port of preSubDevice to connect to: [0..3]. vendorId, prodCode and optionally revNo identify the SubDevice that shall be added. If both prevSubDevice and prevPort are null the last suitable SubDevice is searched and used. Result's *Data* is: null. Example: CreateSubDeviceTreeAndExportENI.py
CallbackResult	AddBeckhoffSlave	(int? prevSlave, int? prevPort, string beckhoffString)	Just a wrapper for AddSubDevice() with fixed vendorId (2) that determines prodCode and revNo from beckhoffString. That is e.g. "EK1100-0000-0018". (Or only "EK1100" to use latest/best revNo) Example: CreateSubDeviceTreeAndExportENI.py
CallbackResult	ShowWorkbench	()	Shows the Workbench window when it is hidden. Result's *Data* is: null. Example: -
CallbackResult	GetSlaveCount	()	Returns the current SubDevice count as seen in the tree view. Result's *Data* is: int. Example: CreateSubDeviceTreeAndExportENI.py
CallbackResult	GetSlaveProperty	(int SlaveAddr, string propName, object propArgs)	Retrieves a SubDevice's property, see below for details. Example: CreateSubDeviceTreeAndExportENI.py
CallbackResult	SetSlaveProperty	(int SlaveAddr, string propName, object propArgs, object propData)	Sets a SubDevice's property, see below for details. Example: CreateSubDeviceTreeAndExportENI.py
CallbackResult	SlaveCoEInitCmdAdd	(int SlaveAddr, ushort objIndex, byte subIdx, byte[] dataBytes, string comment, string[] transitions, bool? complAcc, int? timeOut)	Adds a CoE init command to the SubDevice. SubDeviceAddr is the auto inc. address of the SubDevice. If transitions is null/None, default ("PS") is used. complAcc defaults to SubDevice's support for complete access, timeOut (in ms) defaults to 0. Result's *Data* is: null. Example: CreateSubDeviceTreeAndExportENI.py
CallbackResult	SlaveCoEInitCmdRemoveAll	(int SlaveAutoIncrAddr)	Removes all "user-" CoE init commands. CoE init commands oroginating from the ESI are not removed. SubDeviceAddr is the auto inc. address of the SubDevice. Result's *Data* is: null. Example: CreateSubDeviceTreeAndExportENI.py
CallbackResult	SetWorkbenchExitCode	(int value)	Sets the Workbench process exit code (e.g. *%ERRORLEVEL%* if started via batch file). The last set value is used, regardless of script / script end. Note: Values in the range [100..119] may be used by Workbench itself – but only for very rare critical errors. Example: -
CallbackResult	SetWorkbenchCloseOnScriptEnd	(bool close)	Determines whether the Workbench shall be closed when the script ends. Not affected by the way the script is ended, i.e. even in case of script exceptions the value remains valid. Note: Any other running scripts will also be terminated when a script ends that set close to True. Example: -

(*1) After this method it should be waited until *IsOnlineBusy()* returns "false", or *eewHelper.WaitForOnlineReady()*, see also 4.5.2

GetSlaveProperty

propName	propArgs	Result on success	Description / Result content
Name	unused	string	SubDevice name as displayed in tree view.
PDOs	unused	int[]	All SubDevice PDOs (Outputs: 0x1600..0x17ff, Inputs: 0x1a00..0x1bff)
AssignedPDOs	unused	int[]	SubDevice PDOs that are assigned to an SM.

SetSlaveProperty

propName	propArgs	propData	Get/Set	Description / Data content
Name	unused	string	Get/Set	New SubDevice name.
AssignedPDOs	unused	int[]	Get/Set	List of all PDOs that shall be assigned. (So PDOs not in this list are unassigned if currently assigned.) SM to assign to is chosen automatically. Example: Append PDO.py
PDOs	unused	int[]	Get	List of all PDOs Example: -
PDOContent	int	tuple[]	Set	propArgs is the PDO index, e.g. 0x1600. A tuple consists of (int idx, int subIdx, string dataType, int bitSize, string name, string comment) and describes an entry in the pdo, e.g. (0x7000, 0x01, “INT”, 16, “Variable 1”, “Example Variable”) The complete PDO content is replaced by the tuples. Example: Append PDO.py
SupportsComplete Access	unused	bool	Get/Set	Equals the “SubDevice supports Complete Access” checkbox. Example: CreateSubDeviceTreeAndExportENI.py
EnumDCOperation Modes	unused		Get	Get an array of strings with names of available DC operation modes. Example: ModulesSlotsDC.py
DCOperationMode	unused		Get	Get an operation mode info object with the fileds (see SubDevice → DC → Force … and Add custom ...) --- int DcMode (zero based Index of the DC mode) bool ForceReference UInt64 ShiftTimeNano --- Example: ModulesSlotsDC.py

DCOperationMode	unused		Set	Set a DC operation mode. Argument is a dictionary {
				'DcMode': int,
				'ForceReference':bool,
				'ShiftTimeNano': UInt64(L)
				}
				Example: ModulesSlotsDC.py

4.4.2.2 EtherCATWorkbenchScript.CallbackResult

Property	Type	Description
ErrorCode	ErrorCode	Result of the method call, *Success* or an error. (see *Enums* below)
Data	Object	See method descriptions.

4.4.3 Enums

EtherCATWorkbenchScript.ErrorCode

Value	Description
Success	The call succeeded.
Unspecified	No other error occurred. CallbackResult's *Data* member might be a string with some more details.
Ambiguous	Action is not possible because it is ambiguous (e.g. when trying to add SubDevice by Beckhoff String and different matching SubDevices exist).
CallbackFailed	Should happen only when the Workench is closing and cannot complete the script calls any longer.
CallbackUnknown	Should not happen. (Used internally during development)
Busy	Call is not possible at the moment, usually because the *online buttons* are disabled.
InvalidArgument	Every method argument is invalid/not allowed.
Offline	Action is not possible when offline.
WrongECATState	Action is not possible in the current EtherCAT state (e.g. process variable access when not in Op).
VarNotFound	Variable not found.
VarOfWrongType	Variable is of wrong type. (e.g. when trying to write an input variable).
NotAvailable	Some needed data/info/etc. is missing (e.g. a variable's data, a SubDevice itself, and so on).
ConversionFailed	Any data conversion failed. (Usually when accessing process variable's data)
NoSlave	No SubDevice is selected or was found (e.g. when accessing SubDevice's EEPROM).
AlreadyExists	Some data/info/etc. already exists (e.g. when saving EEPROM data to a file and the file already exists this error is returned).
OfflineOnly	Action is only possible when offline.

EtherCATWorkbenchScript.MasterMode

Name	Comment
Unknown	Should not happen.
Offline	Not (yet) connected to the MDevice.
Idle	This mode should only occur when the user has to select a NIC. (Else it is automatically changed to "Config" mode from this mode.)
Config	SubDevices should be in PreOp. (*Free run* button not down).

4.4.3.1 Helper Module eewHelper

The eewHelper python module (Path ..\Scripts\lib\eewHelper.py) is a wrapper functions for the eew object and adds some tiny functions.

Usage within EtherCAT Workbench scripts (see examples):

- import eewHelper
- eewHelper.Init(eew)

List of functions:

- eewHelper.Init(eew)
  Must be called before first use of any function.
- eewHelper.WaitForOnlineReady()
  Loops until eew.IsOnlineBusy() returns False.
- eewHelper.WaitForMasterMode(mode)
  Loops until eew.GetMasterMode() returns the given mode.
- eewHelper.RetryFreeRunRequest()
  Calls eew.RequestFreerunMode() but tries again if this returns ErrorCode.Busy.
- eewHelper.AbortScriptWhenFailed(res, msg)
  Calls eew.ScriptAbort() if res is not ErrorCode.Success.
- eewHelper.MBoxConfirm(txt)
  Shows a MessageBox and calls eew.ScriptAbort() if Cancel is clicked.
- eewHelper.MBox(txt)
  Shows a Messagebox of Type Information with OK button.
- eewHelper.WbMBoxConfirm(txt)
  Shows a Workbench-window-modal MessageBox and calls eew.ScriptAbort() if Cancel is clicked.
- eewHelper.WbMBox(txt)
  Shows a Workbench-window-modal Messagebox of Type Information with OK button.
- eewHelper.WorkbenchMessageBox(txt,caption,buttons, icon, defaultButton)
  Shows a full featured Workbench-window-modal Messagebox (text, caption, MessageBoxButtons.*, MessageBoxIcon.* MessageBoxDefaultButton.*)
- eewHelper.ByteArrayToHexBin(ba)
  Returns a HexBin string from the given byte[].
- eewHelper.ByteArrayToInt8(ba)
- eewHelper.ByteArrayToInt16(ba)
- eewHelper.ByteArrayToInt32(ba)
- eewHelper.Int8ToByteArray(i)
- eewHelper.Int16ToByteArray(i)
- eewHelper.Int32ToByteArray(i)
  Data conversion
- eewHelper.CoEWrite(objIdx, subIdx, bytes)
  Writes the given byte[] to the CoE object. (Aborts the script on error.)
- eewHelper.CoEWriteInt8(objIdx, subIdx, i)
- eewHelper.CoEWriteInt16(objIdx, subIdx, i)
- eewHelper.CoEWriteInt32(objIdx, subIdx, i)
  Wrappers for CoEWrite().
- eewHelper.CoERead(objIdx, subIdx)
  Reads the CoE object and returns the data as byte[]. (Aborts the script on error.)
- eewHelper.CoEReadInt8(objIdx, subIdx)
- eewHelper.CoEReadInt16(objIdx, subIdx)
- eewHelper.CoEReadInt32(objIdx, subIdx)
  Wrappers for CoERead().

4.5 Tips/Terms

4.5.1 User Input Interference

When the scripts control the Workbench the same mechanisms as if the user performed that action are used – i.e. scripted actions and user actions might interfere; for example, when a script selects a SubDevice and the user selects another one then, the script might perform its following actions with the wrong SubDevice.

4.5.2 Asynchronous Actions

Almost all online actions are performed asynchronously, i.e. when e.g. a button is clicked it is disabled, the action is triggered, and you can continue working with the Workbench.
Some (any!) time later that event completes. You might not notice this behaviour as the delay is usually minimal, but while scripting the Workbench you must be aware of it.
For example, when you try to update multiple process variables and call "eew.VarRequestUpdate()" consecutively: The second call is likely to return with "eew.ErrorCode.Busy" because the previous request has not yet completed.

To avoid such problems, use e.g. "eewHelper.WaitForOnlineReady()" after asynchronous actions. All methods that are definitively asynchronous contain "request" in their names. Other methods might be asynchronous, too, depending on other states – see "*" marks at the methods in the *Classes* section.

4.5.3 Starting Scripts automatically / Command Line

Any ".py" file given to the Workbench's command line arguments will be started automatically. (And made visible in the Dropdown menu *Scripting* even when outside the *Scripts* sub folder)

4.6 Workbench Command Line Arguments

During Workbench scripting certain command line arguments can be used:

Text	Description
/scriptsautorun	Tells the Workbench to monitor its Scripts folder and automatically start new script files.
/logfile	The argument afterwards determines a filename that all messages log output is written to. (The file is never cleared/deleted by the Workbench; the lines are just appended.)
/hidden	Hides the Workbench window, useful of course only if a script file was also given as argument or /scriptsautorun is used.
/nosplash	Prevents the splash window to popup at startup.
/scriptsargs	The argument afterwards determines a string that is made available to all scripts via the *eewScriptsArgs* variable. May not start with */* and cannot be used multiple times. Example: */scriptsargs "-myArg1 -myArg2='1 2 3'"* The quotes *"* are requried to pass the string that is separated by spaces as single argument - they're removed in *eewScriptsArgs*

5 Appendix

5.1 EEPROM Categories Editor

The Categories Editor is opened from the *SubDevice → EEPROM* tab page with a click on the *Categories* button.

Click on a category in the list the left under *Type* to open a form on the right. Some of the forms contain lists with context menu and further value dialogs can be opened.

Type	
Strings	Edit Strings category with context menu on the string list and *Enter new text* dialog.
General	Edit General category *(Group, Image, OrderNo., Name, CoE:, etc.).*
FMMU	Edit FMMU category.
SyncManager	SyncManager category with context menu and *Edit SyncManager* dialog.
TxPDO/ RxPDO	Edit Tx/Rx category TxPDO category with context menu and *Edit PDO* dialog. The RxPDO category is identical.
DC	Edit DC category DC category with context menu and *Edit DC* dialog.
65 (vedor specific)	Edit EEPROM categories Raw data category editor The context menu of the categories list allows to open a *Hex editor* for the selected category. (Unknown categories can be edited only that way.)

5.2 Global Settings

Select *Tools → Edit global settings*
The Global Settings Window contains the following items:
Application

- *AutoLoadRecentProject*
  When set to “True” the last used project is automatically loaded when application is started, else application is started with a new/empty project.

- *DateFormatString*
  Determines how dates are formatted. Used in several places, e.g. messages log or variable monitors.
  Examples[12] (“Format string”: “Resulting date string”)
  - Default:
    “yyyy-MM-dd HH:mm:ss”: “2000-08-17 16:32:32”
  - Default with milliseconds: (resolution usually > 15 ms)
    “yyyy-MM-dd HH:mm:ss.fff”: “2000-08-17 16:32:32.862”
  - “t”: “16:32”, “T”: “16:32:32”
  - “g”: “08/17/2000 16:32”, “G”: “08/17/2000 16:32:32”
  - “F”: “Thursday, August 17, 2000 16:32:32”
  - “u”: “2000-08-17 23:32:32Z”, “U”: “Thursday, August 17, 2000 23:32:32”
  - “r”: “Thu, 17 Aug 2000 23:32:32 GMT”, “s”: “2000-08-17T16:32:32”
  - “dddd, MMMM dd yyyy”: “Thursday, August 17 2000”

- *ENIExportComments*
  When set to *True* the ENI will contain some comments that might help troubleshooting, e.g. number conversions, etc..

- *ENIExportProjectFile*
  When true a Workbench project file is written next to each exported ENI file. (To always have the exactly matching project for an ENI file.)

- *FlashLoggerButtons*
  When set to *True* the message type filter button’s title is flashing for some seconds each time a new message of that type is added. see 1.8.6

- *MaxExceptions*
  An “Exception” is a serious application error that should not occur. This number describes the limit of such errors at which the application is forced to be closed.

- *MaxLoggerItems*
  Maximum number of items in the *Messages log*. Oldest messages are overwritten when this limit is reached.

- *MaxRecentProjects*
  Maximum number of recent projects (main menu *File → Recent projects*) to remember.

- *ScanOnline*
  SubDevice scan behavior after entering *Online* mode, selection:
  - *Question:* Ask whether a SubDevice scan should be carried out.
  - *Manual:* Don't scan, SubDevice scan only by pressing the “Scan” button.
  - *Always:* Scan automatically without asking.

- *XmlEncoding*
  Encoding of all .xml files created by the EtherCAT Workbench.

12 Excerpt from http://msdn.microsoft.com/en-us/library/zdtaw1bw%28v=vs.80%29.aspx

Local MDevice

- *HandleLocalMDeviceService*
  - When set to “True” the local esd EtherCAT MDevice service is started automatically when entering “Online mode”. If the EtherCAT Workbench is not running with administration rights, a UAC prompt appears when the MDevice service is started for the first time, which must be confirmed.
  - Only if the EtherCAT Workbench is running with administration rights the MDevice service is started without UAC prompt and stopped again when the EtherCAT Workbench is closed. Otherwise, the MDevice service will continue running.
- *LocalMDeviceBindToIP*
  Default is 127.0.0.1, so remote access is not possible. Change to a specific address to bind to or 0.0.0.0 if access from another machine shall be possible.

Localization

- *PreferredLCID*
  Some items in SubDevice’s ESI files (e.g. the SubDevice name) can be available in multiple languages – these settings determine which of them to use. (English texts usually have the LCID 1033, German texts 1031.)

MDevice Connection

- *MDeviceConnectionCmdTimeout*
  Timeout for commands sent to the esd EtherCAT MDevice (in ms).

SubDevice Library

- *AlwaysUpdateDeviceCache*
  When set to “True” the SubDevice library created by all SubDevice’s ESI files is recreated every time the EtherCAT Workbench is started. (Instead of being recreated only when the Workbench detects a change to the SubDevice library folder.)
- *IgnoreESIIcons*
  Each SubDevice can have its own icon (displayed in the SubDevice tree view, topology, etc.) – set to “True” to ignore these and to use the same icon (a simple box) for all SubDevices instead.
- *UpdateDeviceCache*
  When set to “True” the SubDevice library created by all SubDevice’s ESI files is recreated next time the EtherCAT Workbench is started. (Instead of being recreated only when the Workbench detects a change to the SubDevice library folder.)

6 Order Information

Type	Properties	Order No.
EtherCAT Workbench – Licence Key	Single license	P.4510.01
EtherCAT Workbench - Project	Project license	P.4511.01
EtherCAT Workbench - Demo	Demo version	P.4512.01

Table 1: Order information hardware

PDF Manuals
For the availability of the manuals see table below.
Please download the manuals as PDF documents from our esd website https://www.esd.eu for free.

Manuals		Order No.
EtherCAT Workbench-ME	Workbench manual in English	P.4510.21

Table 2: Available Manuals

Printed Manuals
If you need a printout of the manual additionally, please contact our sales team (sales@esd.eu) for a quotation. Printed manuals may be ordered for a fee.