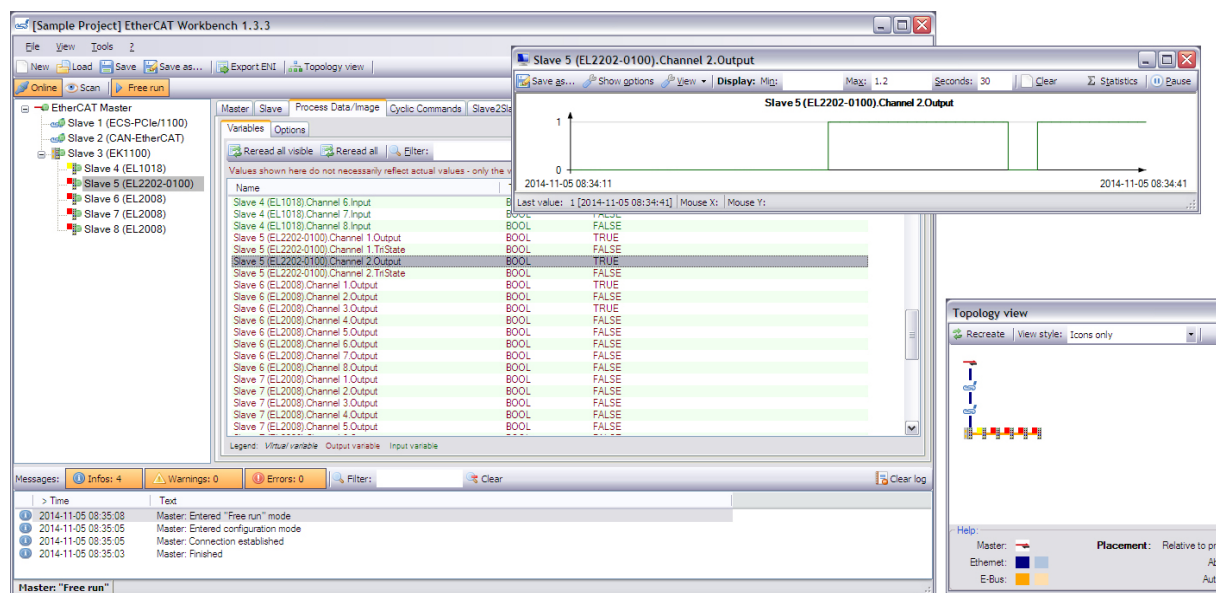




EtherCAT Workbench

EtherCAT Network Configuration Tool



Software Manual

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

NOTE

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.

All rights to this documentation are reserved by **esd electronics**. Distribution to third parties, and reproduction of this document in any form, whole or in part, are subject to **esd electronics's** written approval.

© 2019 esd electronics gmbh, Hannover

esd electronics gmbh
Vahrenwalder Str. 207
30165 Hannover
Germany

Phone: +49-511-372 98-0
Fax: +49-511-372 98-68
E-Mail: info@esd.eu
Internet: www.esd.eu

Trademark Notices

CANopen® and CiA® are registered community trademarks of CAN in Automation e.V.

EtherCAT® is a 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 file:	I:\Texte\Doku\MANUALS\PROGRAM\EtherCAT\Workbench\Englisch\EtherCAT_Workbench_Manual_en_18.odt
Date of print:	2019-11-07

Software version:	from version 1.3.3
--------------------------	--------------------

Document History

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

Revision	Chapter	Changes versus previous version	Date
1.2	1.3	Added "License" section	2012-06-11
	-	Minor changes for new buttons etc. in Workbench V1.1.0	
	4.	Added "Scripting" section	
	2.1.6	Added "esd Master specific", "esd TAP Driver" and "Samples" section	
	2.3	Added "MDP Specific" section	
	5	Added "Order Information"	
1.3	1.2	Added "XP or later" to Requirements/Windows	2013-03-11
	-	Minor changes for Workbench V1.2.0 (some screen shots etc. updated)	
	2.5	Added section "Slave2Slave Communication"	
	3.5	Added section "Send Custom Frames"	
	3.6	Added section "DC Diagnostics"	
	3.4, 2.3.9	Updated descriptions for "Process Data options"	
	2.4	Updated "Cyclic Commands" for new "Custom Commands" page	
	2.3.11	Section "MDP Specific" renamed to "Device Specific" and revised	
1.4	2.3.8.3	Added slave FoE page descriptions	2013-05-21
1.5	2.6	Chapter "Network Topology" updated	2014-02-24
	2.3.6	Chapter "SoE dictionary" added	
	2.3.8.3	Description for "Favorites" button on Page "FoE" added	
	2.3.12	Chapter "Hot Connect" added	
	2.3.8.4	Description for Page "SoE" added	
	3.6	Chapter "Deviation" added	
1.6	2.3.9	Added "LRD/LRW/LWR only in Op" descriptions	2014-08-07
	2.1.1	Added "Address slaves by alias only" description	
	2.3.7	Chapter Slave "Init. commands" updated	
	3.7	Added chapter "Monitor mode"	
	-	Minor changes (editorial changes / screen shots updated)	
1.7	4.	Script methods in 4.4.2.1 added/updated. Chapter 4.6 Workbench command line arguments added	2014-11-05
	-	Minor/editorial changes	
1.8	-	Editorial changes only (change of company name on page 1 and 2)	2019-11-07

Technical details are subject to change without further notice.

Table of contents

Abbreviations and terms.....	6
1. Introduction.....	7
1.1 Features.....	8
1.2 Requirements.....	8
1.3 Installation.....	8
1.3.1 Demo unlocking.....	9
1.3.1.1 “License Requester”.....	9
1.3.1.2 License Installation.....	9
1.4 Application overview.....	10
1.4.1 Toolbars/menu.....	10
1.4.2 Slave tree view.....	13
1.4.3 Tab pages.....	13
1.4.4 Messages log.....	13
2. EtherCAT network configuration.....	14
2.1 Master settings.....	14
2.1.1 General settings.....	14
2.1.2 Online states.....	17
2.1.3 Online statistics.....	18
2.1.4 Send custom frames.....	18
2.1.5 Init commands.....	19
2.1.6 EoE.....	19
2.1.6.1 esd Master specific.....	19
2.1.6.2 esd TAP driver.....	19
2.1.6.3 Samples.....	19
2.1.7 ENI extras.....	20
2.2 ENI export.....	21
2.3 Slave configuration.....	21
2.3.1 Adding slaves to the network.....	21
2.3.1.1 Installing new ESI files.....	22
2.3.1.2 Slaves without ESI file.....	22
2.3.2 General.....	23
2.3.3 EEPROM data.....	23
2.3.4 Memory (Registers).....	26
2.3.5 CoE dictionary.....	27
2.3.6 SoE dictionary.....	29
2.3.7 Init. commands.....	30
2.3.8 Mailbox settings.....	31
2.3.8.1 Page “CoE”.....	31
2.3.8.2 Page “EoE”.....	32
2.3.8.3 Page “FoE”.....	33
2.3.8.4 Page “SoE”.....	34
2.3.9 Process data.....	35
2.3.10 DC.....	36
2.3.11 Device Specific.....	36
2.3.12 Hot Connect.....	37
2.3.12.1 Example.....	37
2.3.12.2 Remarks.....	38
2.4 Cyclic commands.....	39
2.5 Slave2Slave Communication.....	41
2.6 Network Topology.....	42
3. Online configuration.....	44
3.1 Connection to the EtherCAT Master.....	44

3.2	Slave scan	45
3.3	Changing to free run mode	46
3.4	Process data	47
3.4.1	Virtual variables	49
3.4.2	Variable editor	50
3.4.3	Variable monitor	51
3.5	Send Custom Frames	54
3.6	DC Diagnostics	55
3.6.1	Deviation	55
3.6.2	SYNC Generation	56
3.7	Monitor mode	57
4.	EtherCAT Workbench Scripting	58
4.1	Overview	58
4.2	Limitations	58
4.3	Dropdown menu "Scripting"	58
4.4	Available variables etc.	59
4.4.1	Variables	59
4.4.2	Classes	59
4.4.2.1	EtherCATWorkbenchScript	59
4.4.2.2	EtherCATWorkbenchScript.CallbackResult	63
4.4.3	Enums	63
4.5	Tips/Terms	65
4.5.1	User input interference	65
4.5.2	Asynchronous actions	65
4.5.3	Starting scripts automatically / command line	65
4.6	Workbench command line arguments	65
5.	Order Information	66

Abbreviations and terms

Abbr.	Term	Description/Comment
APRD	Slave Auto-Increment Physical Read	EtherCAT slave device EtherCAT command to read value from a single slave address by its auto-increment address
BRD	Broadcast Read	EtherCAT command to read value from all slaves
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 master
EoE	Ethernet over EtherCAT	
ESC	EtherCAT Slave Controller	
ESI	EtherCAT Slave Information	Contains information about EtherCAT slave devices Homepage: www.ethercat.org
ETG	EtherCAT Technology Group	
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 slave device Mailbox
NIC	Network Interface Card	Also used synonymic to "Network Interface"
NOP	No Operation	EtherCAT command that is ignored by the slaves
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. Introduction

The esd EtherCAT Workbench consists of an EtherCAT master and the EtherCAT Workbench application which controls and configures the master. This document describes the EtherCAT Workbench application.

The data exchange between the EtherCAT Workbench and the master instance is based on local interprocess communication (IPC) mechanisms (Fig. 1) or TCP/IP based network communication (Fig. 2). This flexibility allows to attach the EtherCAT workbench to esd master instances on remote (embedded) devices. The EtherCAT Workbench for Windows comes with an EtherCAT master instance which is installed on the local PC as a Windows service so the PC can be connected directly to an EtherCAT slave segment for network configuration.

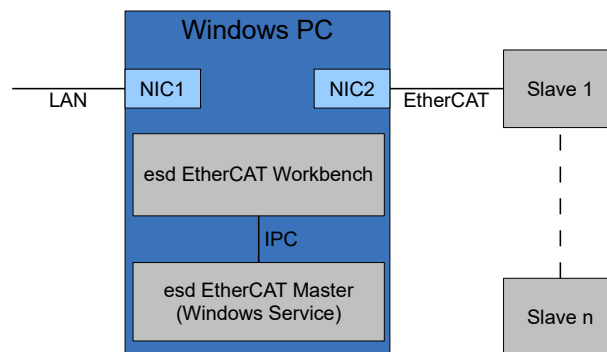


Fig. 1: EtherCAT Workbench with local Master

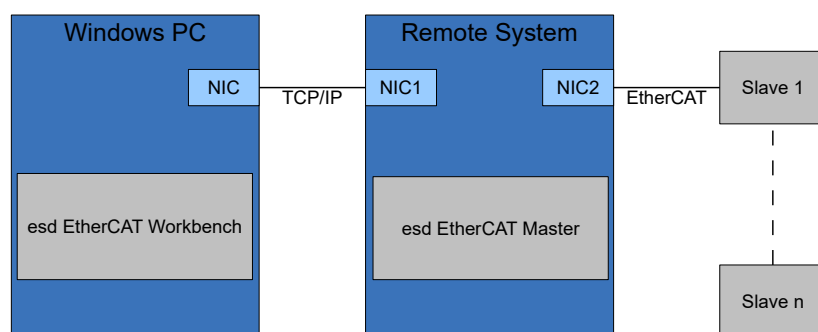


Fig. 2: EtherCAT Workbench with remote master

1.1 Features

- Offline configuration based on device vendor ESI files
 - Slave 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.)
 - Network topology overview
 - Export of ENI file according to ETG.2100 specification (Slave configuration, process data layout, cyclic commands, etc.) which can be processed by most of the EtherCAT masters 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.
 - Master configuration
 - Network adapter(s) to use, cycle time, etc.
 - Slave / network diagnostics
 - Slave state, register data, ESI EEPROM data, CoE objects, FoE, 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 slaves

1.2 Requirements

- esd EtherCAT Master (Win32 version for exclusive operation with the Workbench is part of the distribution)
- Microsoft Windows (XP or later, 32/64 bit) with .NET Framework 2, Service Pack 2
- Screen resolution at least 1024×768 pixel
- Memory as recommended for operating system
- Disk space approximately 20 MB

1.3 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 Master 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 Master”)

1.3.1 Demo unlocking

After installation the esd EtherCAT Workbench runs as a “Demo” version with some limitations. It has to be unlocked by a license file to remove these limitations.

That license file will be bound to a specific PC¹ 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 1.3.1.1 for details.

You will receive the license file by email and have to install it to the EtherCAT Workbench, see section 1.3.1.2 for details.

1.3.1.1 “License Requester”

This tool shows your Hardware Id² and allows you to enter your company name. These 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)

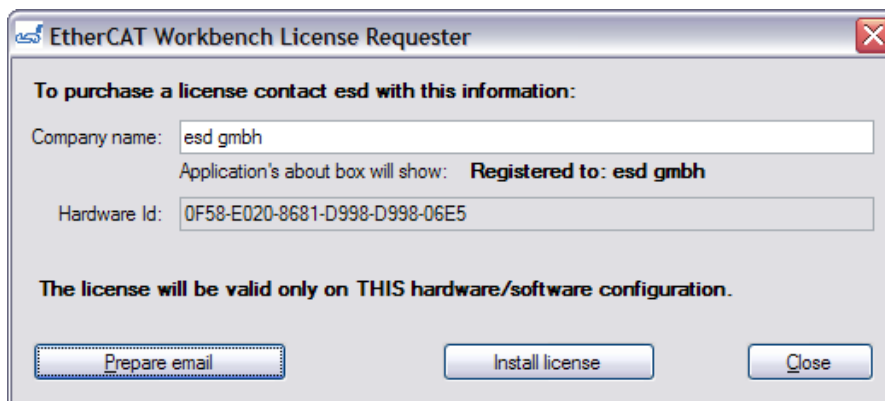


Fig. 3: Licenser Requester sample

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 these information. If that fails or is not wanted just prepare your own email to info@esd.eu and copy the “Hardware Id” and “Company name” texts manually.

1.3.1.2 License Installation

A purchased license file can be installed manually **or** by using the “License Requester” described above. **Manually:** Copy the “.license” file 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 “.license” file in the file selection dialog that is shown.

When the Workbench is running while the file is copied, it has to be restarted to make use of the license file.

¹ For other licensing variants please contact esd.

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

1.4 Application overview

After starting the EtherCAT Workbench its main form is shown. Fig. 4 describes its components. All changes are summarized/stored as “project”.

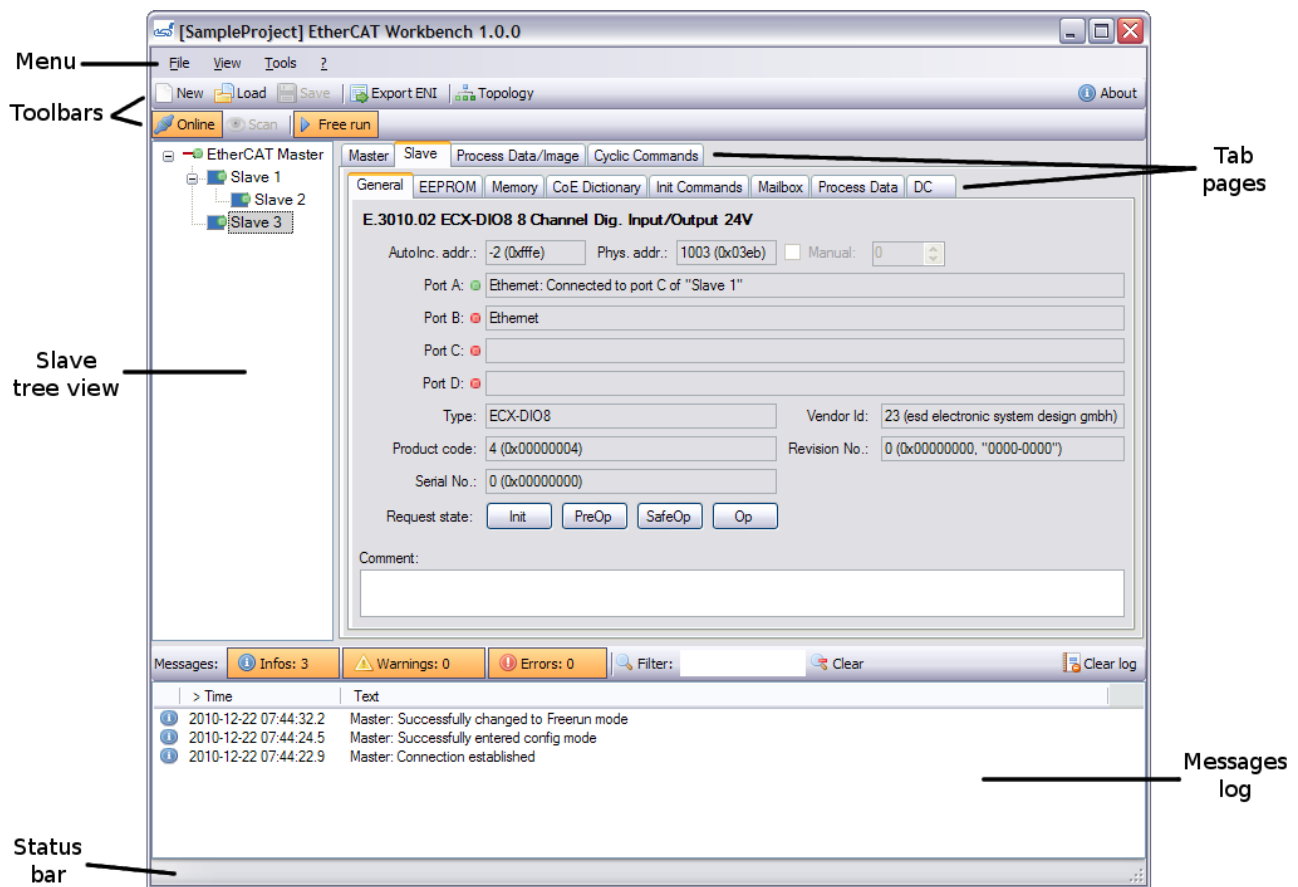


Fig. 4: Main form

1.4.1 Toolbars/menu

The first (upper) toolbar controls general tasks, such as loading/saving project files. The second toolbar is used for online tasks, such as connecting to the master or scanning for slaves.

General Toolbar

- Button “New” / “Load” / “Save”
 - Start new project or load/save existing project file
- Button “Export ENI”
 - Save ENI to a .xml file, see section 2.2
- Button “Topology view”
 - Shows a graphical overview of the EtherCAT network, see section 2.6

Online Toolbar

- Button “Online”
 - Connect/Disconnect to/from the EtherCAT master, see section 3.1
- Button “Scan”
 - Start a slave scan (when online), see section 3.2
- Button “Free run”
 - Switch master to free run mode (to read process data, etc.), described in section 3.3
- Dropdown menu “Scripting”
 - See chapter 4

Menu items

Most items are identical to toolbar items or self-explanatory.

- “Tools” → “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, favorite slaves, etc.
- “Tools” → “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.
 - “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
 - Examples³ (“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)

³ Excerpt from <http://msdn.microsoft.com/en-us/library/zdtaw1bw%28v%28vs.80%29.aspx>

- “FlashLoggerButtons”
 - When set to “True” the message type filter button’s title (section 1.4.4) is flashing for some seconds each time a new message of that type is added
- “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
- “XmlEncoding”
 - Encoding of all .xml files created by the EtherCAT workbench
- “HandleLocalMasterService”
 - When set to “True” the local esd EtherCAT Master service is started automatically when the EtherCAT Workbench is started and stopped when the EtherCAT Workbench is closed
- “LocalMasterBindToIP”
 - 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
- “PreferredLCID”
 - Some items in slave’s ESI files (e.g. the slave name) can be available in multiple languages – this settings determines which of them to use. (English texts usually have the LCID 1033, German texts 1031)
- “MasterConnectionCmdTimeout”
 - Timeout for commands sent to the esd EtherCAT Master (in ms)
- “AlwaysUpdateDeviceCache”
 - When set to “True” the slave library created by all slave’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 slave library folder)
- “IgnoreESIIcons”
 - Each slave can have its own icon (displayed in the slave tree view, topology, etc.) – set to “True” to ignore these and to use the same icon (a simple box) for all slaves instead
- “UpdateDeviceCache”
 - When set to “True” the slave library created by all slave’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 slave library folder)
- “Tools” → “Restart local EtherCAT Master service” / “Stop local EtherCAT Master service”
 - When the local esd EtherCAT Master needs to be stopped (e.g. because of a misconfigured ENI) it can be stopped and restarted with this items. Usually it is started/stopped automatically with the Workbench start/close.

1.4.2 Slave tree view

This shows the devices in the EtherCAT network. The first device is always the EtherCAT master, below the master the EtherCAT slaves are displayed. The order (top down) represents the physical slave order, i.e. the way the EtherCAT frame will go.

This list is either populated manually (section 2.3.1, p. 21) or by running an online scan of an existing network (section 3.2, p. 45).

In online mode (chapter 3) the slave icons will additively display information about the slave states, too. Details about the used images can be obtained by the context menu entry “Show icon/color legend”

The context menu also allows to remove/add slaves, see section 2.3.1.

1.4.3 Tab pages

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

1.4.4 Messages log

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 master basically just needs an ENI file to initialize and control an EtherCAT network. This file contains the complete configuration information about the master, all slaves, all frames that need to be sent cyclically or to initialize the slaves, 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 master when “Free run mode” (section 3.3) is entered. This means that most changes will become effective only then. (The online tasks are described in chapter 3)

2.1 Master settings

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

2.1.1 General settings

The screenshot shows the 'General' settings page in the EtherCAT Workbench. The 'Master' tab is selected at the top. Below it, the 'General' sub-tab is active. The 'General settings' section includes fields for 'Base cycle time in µs' (set to 4000), 'Destination MAC address' (set to ffffffff), and 'VLAN Id' (set to 0). There is a checkbox for 'Address slaves by alias only'. The 'esd EtherCAT Master settings' section includes a 'Connection' field (set to 127.0.0.1) and a 'Password' field. Below these are buttons for 'Show master details', 'Test connection RTT', 'Send ping', and 'Scan LAN'. The 'NIC' section includes a 'Primary adapter' dropdown (set to [74-ea-3a-81-d2-67] "Realtek PCIe GBE Family Controller"), a 'Redundant adapter' dropdown (set to [None]), and a 'Source address(es)' field (set to 74-ea-3a-81-d2-67). There is a 'Set on master' button. The 'Other' section includes fields for 'Max acyclic frames' (set to 0), 'DC start time' (set to 0 ms), 'Acycl. frame timeout' (set to 0 ms), and 'EEPROM delay' (set to 0 ms). There are checkboxes for 'Reduce frames' and 'Disable slave SM watchdogs for Workbench'. A reminder at the bottom states: 'Reminder: These settings are esd specific and not exported to the ENI.'

Fig. 5: Master configuration, page “General”

- “Base cycle time”
 - Used to determine the interval of the EtherCAT commands that handle the slave process data, see also section 2.3.9: “Page Options”. Actual used value may differ, according to master settings/capabilities
- “Destination MAC address”
 - The master transmits all EtherCAT frames using this value as Ethernet destination address. Default is the Ethernet broadcast address “FFFFFFFFFFFFFF”. Some master 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 master. This value is not part of the ENI files specification ETG.2100 and can only be used in combination with esd masters which control a slave segment connected to a VLAN capable switch
- “Address slaves by alias only”
 - When this is checked the initialization commands for all slaves are changed to use the alias address (ESC register 0x0012 / EEPROM ESI word 0x0004), to allow changing the order of the slaves 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.



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

esd EtherCAT Master settings

- “Connection:”
 - Connection string is entered here, for details see section 3.1
 - “Show master details”
 - Shows some technical details about the esd EtherCAT Master instance the EtherCAT Workbench is connected to
 - “Test connection RTT”
 - Sends a small data packet to the esd EtherCAT Master and measures the time until it is echoed back (“Round Trip Time”)
 - “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 EtherCAT master’s NIC(s), see section 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 master to reduce the total number of cyclic frames by combining as much cyclic EtherCAT commands as possible in to 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, see 3.4 “Page Options”

- “Acycl. frame timeout”
 - Time out value for the acyclic frames, see also esd EtherCAT Master manual
- “EEPROM delay”
 - Used when writing slave’s EEPROM data in Free run mode – for slaves that require more time until they acknowledge EEPROM write commands
- “Disable slave SM watchdogs for Workbench”
 - Write each slave’s register 0x420 to 0 when going to free run. (Affects only Workbench usage, not the exported ENI. Selecting “Set SM watchdog” on a slave’s init commands page overrides this)

Note that some slaves might deny that or require additional changes that are not included here.

2.1.2 Online states

EtherCAT

- “Current state:”
 - Reflects the current network state of the EtherCAT master. (Also displayed as icon on the master in the slave tree view)
- “Request state:”
 - Request the EtherCAT master to change to a specific state. Note that “SafeOp” and “Op” require the ENI, therefore these states are only possible in “Free run” mode (section 3.3)

Master

- “Slaves:”
 - Number of EtherCAT slaves:
 - “No.”: Total number of slaves in ENI
 - “Active”: Number of active slaves (determined by reading “AL state” registers in free run mode, see section 2.4, “Options”)
 - “MBox”: Number of slaves with Mailbox support
 - “Primary”: Number of slaves at master’s primary network interface
 - “Redundant”: Number of slaves at master’s redundant network interface
- “DC deviation:”
 - Distributed clock deviation between master system time and slave reference clock
- “Cyclic frames:”
 - Number of frames that are sent cyclically, should match frames listed in “Cyclic Commands” tab page described in section 2.4
- “State flags:”
 - State flags from the esd EtherCAT Master (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.

“Last update”

Date when all currently displayed information on this page was received from the master.

2.1.3 Online statistics

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

- “Master”
 - EtherCAT statistical data of the master instance.
- “NIC”
 - Ethernet statistical data of the NICs.
- “Device”
 - EtherCAT statistical data of the device instance.

The list is either updated by clicking “Update” in the toolbar, or by entering a value at “Auto update interval:”.

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, Fig. 6 shows a sample frame:

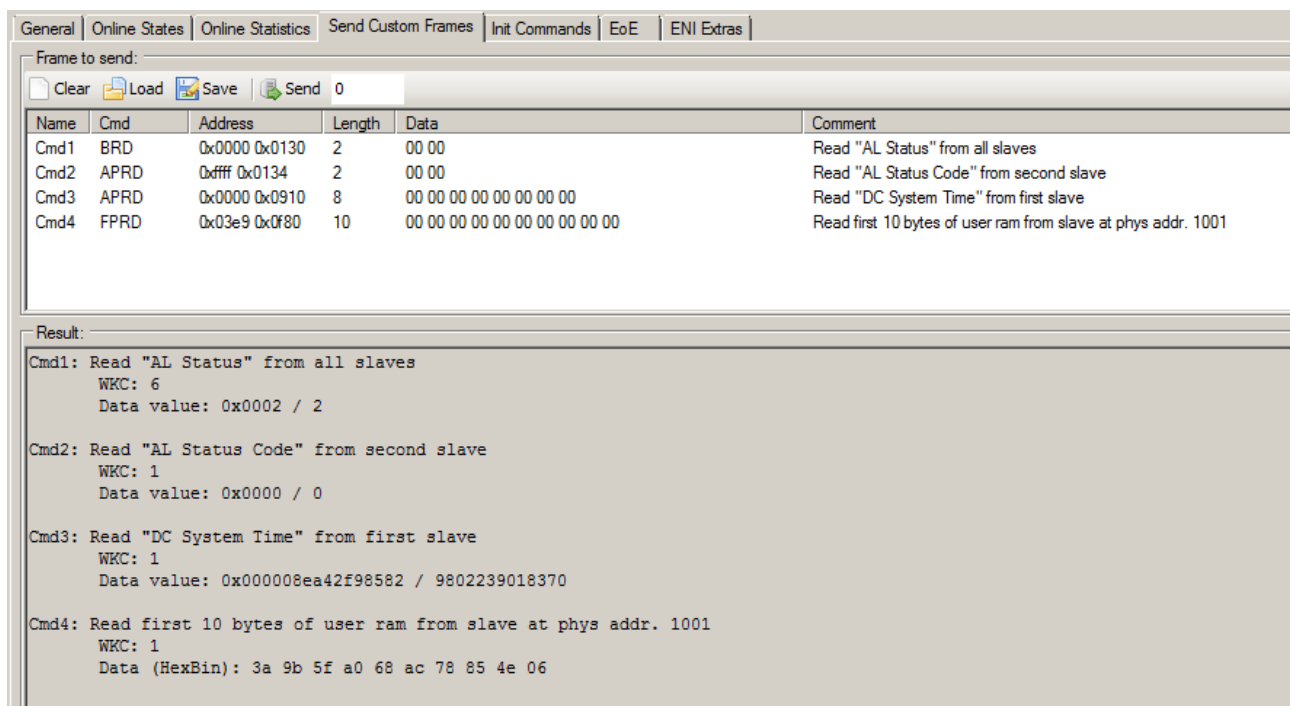


Fig. 6: Example for “Send custom frame”

2.1.5 Init commands

This page allows configuration of the initialization commands which are sent to all slaves during network initialization. Usually this should be set to “Default”, which means all commands are created automatically based on the corresponding settings. The list of configuration commands is shown with the “View” button.

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.

2.1.6 EoE

Automatically enabled when a slave supports the EoE protocol. These parameters configure the required virtual switch within the EtherCAT master 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
- “Slave auto configuration settings:”
 - This is used to configure the settings that are assigned to slaves that have “Auto config by master presets” checked in their EoE settings tab, see also section 2.3.8.2.

2.1.6.1 esd Master specific

- Virtual Switch
 - When this is enabled the master transfers all EoE frames between the EoE slaves, i.e. the EtherCAT network becomes the backbone of an Ethernet switch whose ports are the EoE slaves 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 TAP⁴ 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”
- Set all participating EoE slaves to “Switch Port”

⁴ 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.

EtherCAT network configuration

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

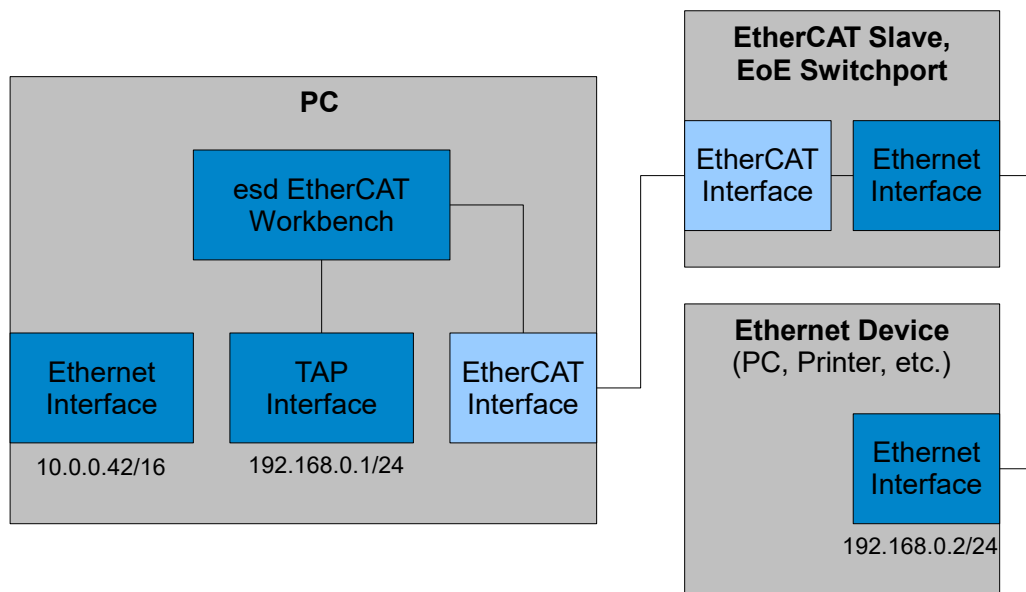


Fig. 7: EoE “Virtual Port” sample with sample IP settings

Connect device behind EoE Slave to PC with Workbench

- Set EoE Slave to “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 slave. 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”

You should now be able to “ping” etc. the device behind the EoE slave from the PC and vice versa.

Connect device behind EoE slave to LAN/Internet

- Same settings as with previous sample, but Ethernet Interface and TAP Interface must be bridged.

For the sample in Fig. 7 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 slave must use IP settings accordingly, e.g. 10.0.0.43/16.

The Ethernet device behind the EoE slave 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.

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 Master and master implementation of several other vendors. This configuration is also used by the built-in master of the workbench when switching to free run mode.

2.3 Slave configuration

2.3.1 Adding slaves to the network

As stated in section 1.4.2 the slave list is either populated manually or by running an online scan of an existing network. This chapter describes how to add the slaves manually.

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

A slave is added by using the context menu⁵ of the slave tree view: depending on the selected slave different options will appear. If no slaves exist, the master has to be selected and the context menu will show an item “Append new slave”. If slaves exist the context menu will offer more options depending on the selected slave, e.g. different ports to add the new slave to.

After clicking a menu item to add/insert the new slave a window will show up:

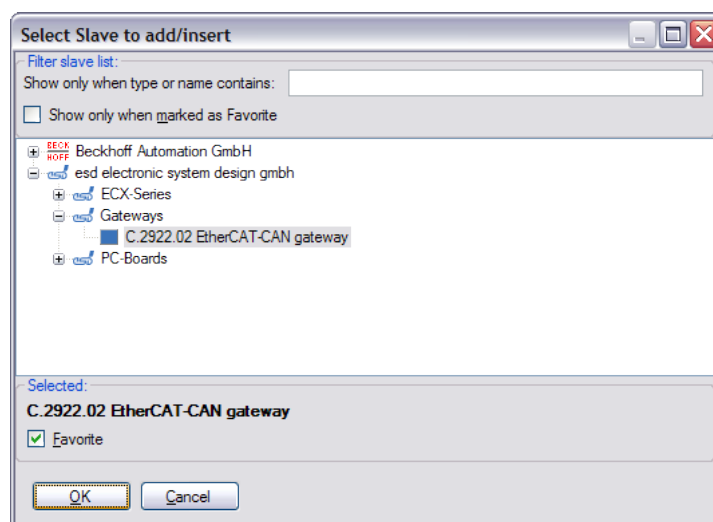


Fig. 8: Adding/inserting a slave

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

Additionally this form allows to filter the list by text or a “favorite slave” marker. If “Show only when marked as favorite” is checked then the list will show only slaves that have been marked. To mark/unmark a slave select it and check/uncheck the “Favorite” check box in the lower pane. When a filter text is entered in the upper pane only slaves whose name or type name contains this text are shown.

The context menu allows to expand/collapse the tree and to show hidden/obsolete items, too.

⁵ Or by pressing the “Insert” key within the slave tree view.

Removing slaves

To remove a slave “Delete slave” from its context menu is clicked⁶. When a slave is connected to this slave it is connected to its previous slave. If that’s not possible it is deleted, too (same procedure for the deleted slave then).

2.3.1.1 Installing new ESI files

The ESI files (.xml files) for the slave library just need to be copied in the “SlaveLibrary” 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 slave library” from the “Tools” menu can be used to simplify this. (Application needs to be restarted anyway)

2.3.1.2 Slaves without ESI file

Slaves without an ESI file can’t be added manually, but they are automatically added when found during a “Slave scan” in online mode (section 3.2). In this case the required information about the slave is obtained from the slave’s ESI EEPROM data.

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

⁶ Or by pressing the “Del” key within the slave tree view.

2.3.2 General

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

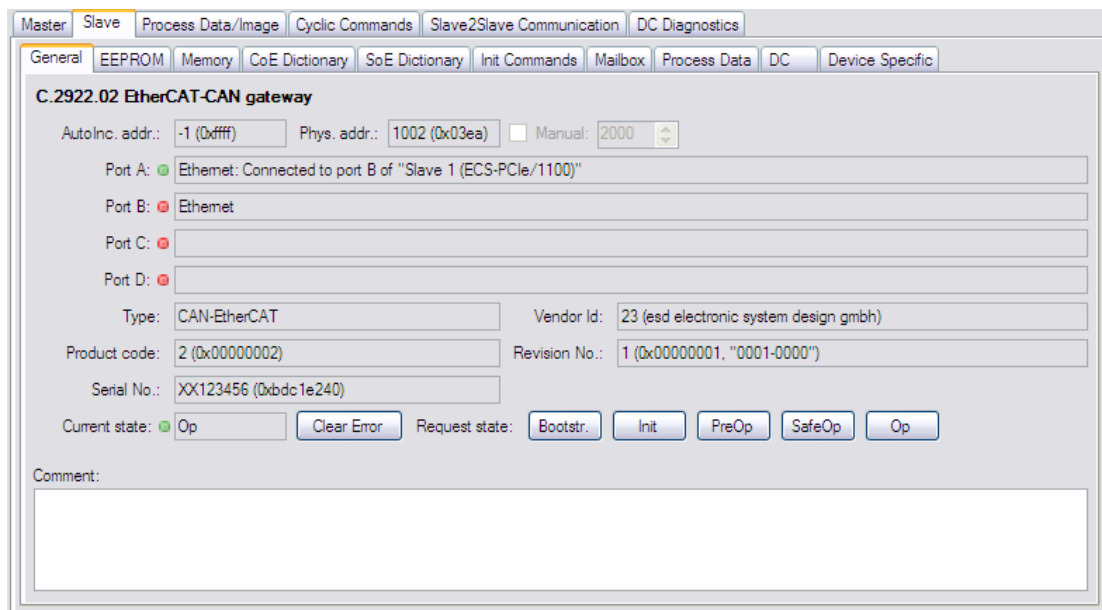


Fig. 9: Slave configuration, page “General”

Ports: The text field for each port (e.g. “Port A”) shows the type (Ethernet/EBUS) and the slave 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 slaves would have the same physical address) it is ignored and the next unused address is used instead.

“Configure aliases”: Opens the slave “Alias editor” that is reachable by menu “Tools” → “Configure alias addresses for all slaves” – allows reading/writing of alias address of a single slave or all slaves. (Note: this editor is independent from the EEPROM editor (see section 2.3.3), i.e. the data displayed there is not automatically updated when new alias address was written by the “Alias editor”)

“Revision No.”: The “0000-0000” interpretation consists of the low and high word of the revision number as decimal.

Text field “Comment”: Can be used for storing any user text, it has no effect and is used only within the project and ENI file.

2.3.3 EEPROM data

This is used to read, write, edit or create a slave’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. All editing and creating is done within that editor, only when “To device” is clicked the EEPROM content within the slave will be modified.

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

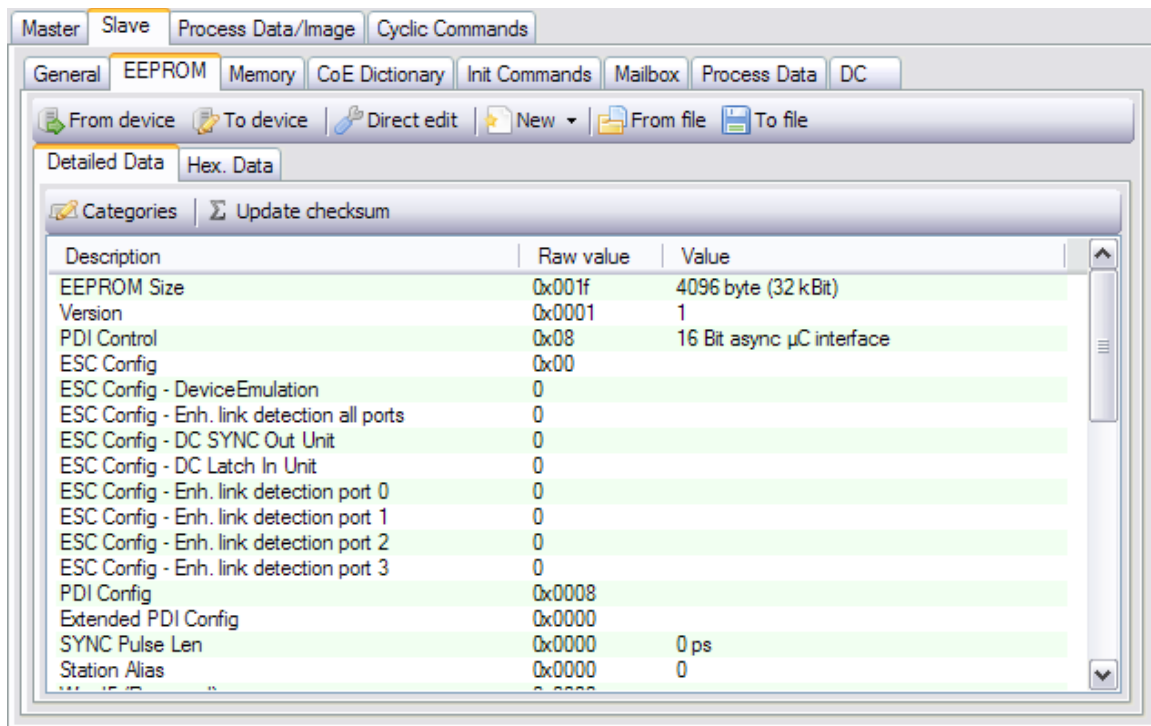


Fig. 10: Slave configuration, page “EEPROM”

- Button “From device”
 - Reads the EEPROM data from the device to the EEPROM editor (online mode only). Current EEPROM editor content is overwritten without further notice
- Button “To device”
 - Writes current EEPROM editor contents to the device (online mode only, does not ask for confirmation)
- Button “From file”
 - Loads EEPROM editor content from binary file
- Button “To file”
 - Saves current EEPROM editor content to a binary file
- Menu Button “To clipboard”
 - 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)
- Button “Direct edit”
 - Opens a dialog window that allows writing/reading single words (16 bit) to/from the EEPROM (Note: to write the slave alias address menu “Tools” → “Configure alias addresses for all slaves” can be used)
- Menu Button “New”
 - 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.

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 slave/ESI settings”
 - Creates EEPROM data from current ESI and current slave settings
- “From other ESI”
 - Opens a window to select a slave from the slave library and creates EEPROM data from that slave’s ESI
- “From custom ESI file”
 - Opens a window to select a .xml file and then shows a window to select a slave from within that file. Creates EEPROM data based on this selection
- Menu Button “Comparison”
 - 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)

EEPROM Editor (page “Detailed Data”)

This list shows the fixed EEPROM content with its description and value. The dynamic content (“Categories”⁷) 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)

- Button “Categories”
 - Opens the category editor: The list on the left consists of the categories, the main window shows an editor for the selected category.

The context menu of the list also allows to open a “Hex editor” for the selected category. (Unknown categories can be edited only that way)
- Button “Update checksum”
 - Use this to update the configuration area checksum (a wrong checksum is displayed with red text and value “Invalid”)

⁷ For more information about the EEPROM categories see ETG document 1000.6

2.3.4 Memory (Registers)

This tab page allows to view/edit slave 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 information⁸: “All known registers”. For plain data without details/interpretation “All as Byte” or “All as Word” can be selected.⁹

“All known registers” will display multiple interpretations for some addresses because they depend on the PDI type for example. Also not all slave/types support all registers, so **reading all registers will always show warnings about some unread items.**

Address	Comment	Value (HexBin)	Value (custom)	Last read at
0x0000	Type	11	ET1100	2013-03-11 06:39:48
0x0001	Revision	00	0	2013-03-11 06:39:48
0x0002:0x0003	Build	00 00	0	2013-03-11 06:39:48
	[7:4] IP Core: Minor version	00	0	
	[3:0] IP Core: Maintenance version	00	0	
0x0004	FMMUs supported	08	8	2013-03-11 06:39:48
0x0005	SyncManagers supported	08	8	2013-03-11 06:39:48
0x0006	RAM size (KByte)	08	8	2013-03-11 06:39:48
0x0007	Port descriptor	3b	59	2013-03-11 06:39:48
	[1:0] Type port 0	03	MII	
	[3:2] Type port 1	02	E-BUS	
	[5:4] Type port 2	03	MII	
	[7:6] Type port 3	00	Not implemented	
0x0008:0x0009	ESC features supported	fc 00	252	2013-03-11 06:39:48
	[0] FMMU operation	00	Bit oriented	
	[3] DC width	01	64 bit	
	[4] Low jitter EBUS	01	Available	
	[5] Enh. link detection EBUS	01	Available	
	[6] Enh. link detection MII	01	Available	
	[7] Separate handling of FCS errors	01	Supported	
	[8] Enh. DC SYNC activation (0x09...	00	Not available	
	[9] EtherCAT LRW command	00	Supported	
	[10] EtherCAT BRW/APRW/FPR...	00	Supported	
	[11] FMMU/SyncManager configur...	00	Variable	
0x0010:0x0011	Configured station address	e9 03	1001	2013-03-11 06:39:48
0x0012:0x0013	Configured station alias	e9 03	1001	2013-03-11 06:39:48
0x0020	Write register enable	00	0	2013-03-11 06:39:48
	[0] Enabled	00	Disabled	

Fig. 11: Slave configuration, page “Memory”

Other Toolbar elements

- Button “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
- “Filter:”
 - When a text is entered, the list will only show items whose “Comment” or “Address” column contain the entered text (not case sensitive)

Context menu

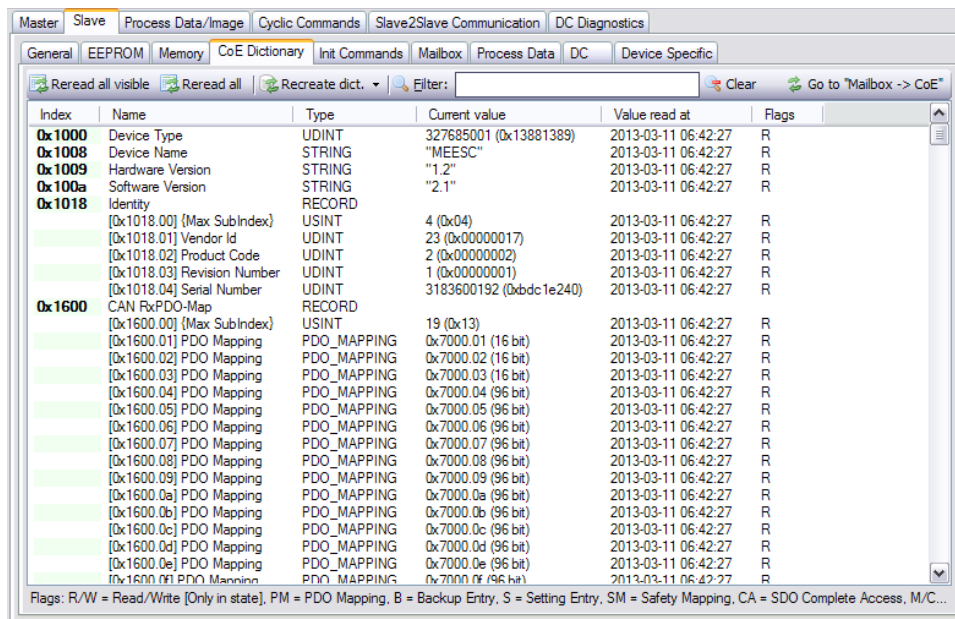
- “Edit/view value”
 - Opens the variable editor (see section 3.4.2) for the selected value. (No information about whether a register is writable or not is available)
- “Reread selected item(s)”
 - Tries to read all values for all selected items from the device (online mode only)

⁸ The known details/registers won’t cover all ESCs and configurations due to their variety

⁹ To create a new item selection for a specific slave/PDI type see application’s “Templates\ESCMemory” sub folder: Create a copy of an existing file there and adapt it to your needs.

2.3.5 CoE dictionary

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



Index	Name	Type	Current value	Value read at	Flags
0x1000	Device Type	UDINT	327685001 (0x13881389)	2013-03-11 06:42:27	R
0x1008	Device Name	STRING	"MEESC"	2013-03-11 06:42:27	R
0x1009	Hardware Version	STRING	"1.2"	2013-03-11 06:42:27	R
0x100a	Software Version	STRING	"2.1"	2013-03-11 06:42:27	R
0x1018	Identity	RECORD			
	[0x1018.00] {Max SubIndex}	USINT	4 (0x04)	2013-03-11 06:42:27	R
	[0x1018.01] Vendor Id	UDINT	23 (0x00000017)	2013-03-11 06:42:27	R
	[0x1018.02] Product Code	UDINT	2 (0x00000002)	2013-03-11 06:42:27	R
	[0x1018.03] Revision Number	UDINT	1 (0x00000001)	2013-03-11 06:42:27	R
	[0x1018.04] Serial Number	UDINT	3183600192 (0xbdc1e240)	2013-03-11 06:42:27	R
0x1600	CAN RxPDO-Map	RECORD			
	[0x1600.00] {Max SubIndex}	USINT	19 (0x13)	2013-03-11 06:42:27	R
	[0x1600.01] PDO Mapping	PDO_MAPPING	0x7000.01 (16 bit)	2013-03-11 06:42:27	R
	[0x1600.02] PDO Mapping	PDO_MAPPING	0x7000.02 (16 bit)	2013-03-11 06:42:27	R
	[0x1600.03] PDO Mapping	PDO_MAPPING	0x7000.03 (16 bit)	2013-03-11 06:42:27	R
	[0x1600.04] PDO Mapping	PDO_MAPPING	0x7000.04 (96 bit)	2013-03-11 06:42:27	R
	[0x1600.05] PDO Mapping	PDO_MAPPING	0x7000.05 (96 bit)	2013-03-11 06:42:27	R
	[0x1600.06] PDO Mapping	PDO_MAPPING	0x7000.06 (96 bit)	2013-03-11 06:42:27	R
	[0x1600.07] PDO Mapping	PDO_MAPPING	0x7000.07 (96 bit)	2013-03-11 06:42:27	R
	[0x1600.08] PDO Mapping	PDO_MAPPING	0x7000.08 (96 bit)	2013-03-11 06:42:27	R
	[0x1600.09] PDO Mapping	PDO_MAPPING	0x7000.09 (96 bit)	2013-03-11 06:42:27	R
	[0x1600.0a] PDO Mapping	PDO_MAPPING	0x7000.0a (96 bit)	2013-03-11 06:42:27	R
	[0x1600.0b] PDO Mapping	PDO_MAPPING	0x7000.0b (96 bit)	2013-03-11 06:42:27	R
	[0x1600.0c] PDO Mapping	PDO_MAPPING	0x7000.0c (96 bit)	2013-03-11 06:42:27	R
	[0x1600.0d] PDO Mapping	PDO_MAPPING	0x7000.0d (96 bit)	2013-03-11 06:42:27	R
	[0x1600.0e] PDO Mapping	PDO_MAPPING	0x7000.0e (96 bit)	2013-03-11 06:42:27	R
	[0x1600.0f] PDO Mapping	PDO_MAPPING	0x7000.0f (96 bit)	2013-03-11 06:42:27	R

Fig. 12: Slave configuration, page "CoE Dictionary"

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



Note that slaves will reset most items in certain state changes. Especially changing to Free run mode includes a complete slave initialization that usually resets manual changes here.

Initialization commands are added for slave configuration, see context menu "Create an init command for this entry" below and 2.3.8.1, Page "CoE".

Other Toolbar elements

- Button "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
- "Filter:"
 - When a text is entered, the list will only show items whose "Index" or "Name" column contain the entered text (not case sensitive)

Context menu

- "Edit/view current value"
 - Opens the variable editor (see section 3.4.2) for the selected item's value
- "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 slave 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 slave or just don't need to be seen. (It does not alter the slave configuration)
- “Change background color”
 - Just a temporary setting for the display of this item within the list view
- “Create an init command for this entry”
 - Used to simplify creation of custom initialization commands, see 2.3.8.1

2.3.6 SoE dictionary

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

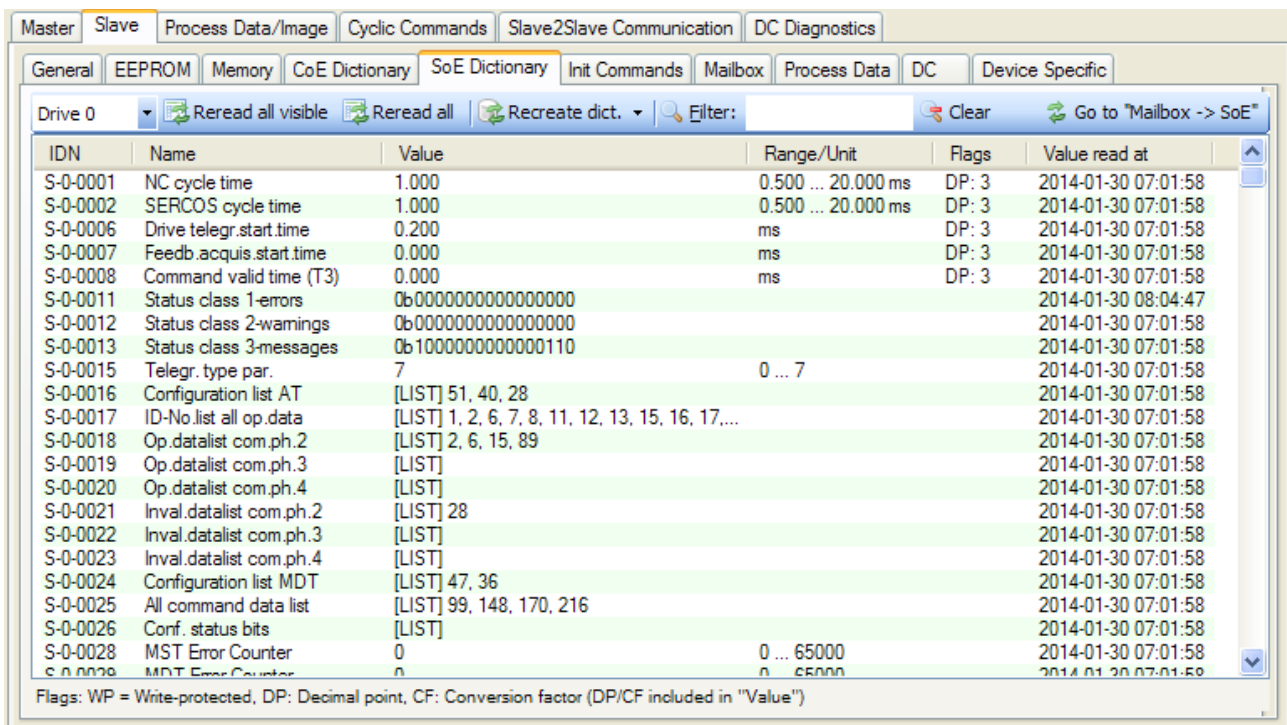


Fig. 13: Slave configuration, page "SoE Dictionary"

For Slaves including multiple drives the drop down List ("Drive 0" in Fig. 13) 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 slave" 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 (see 2.3.8.4) for a specific object.

2.3.7 Init. commands

This page allows configuration of the initialization commands for the selected slave. 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.

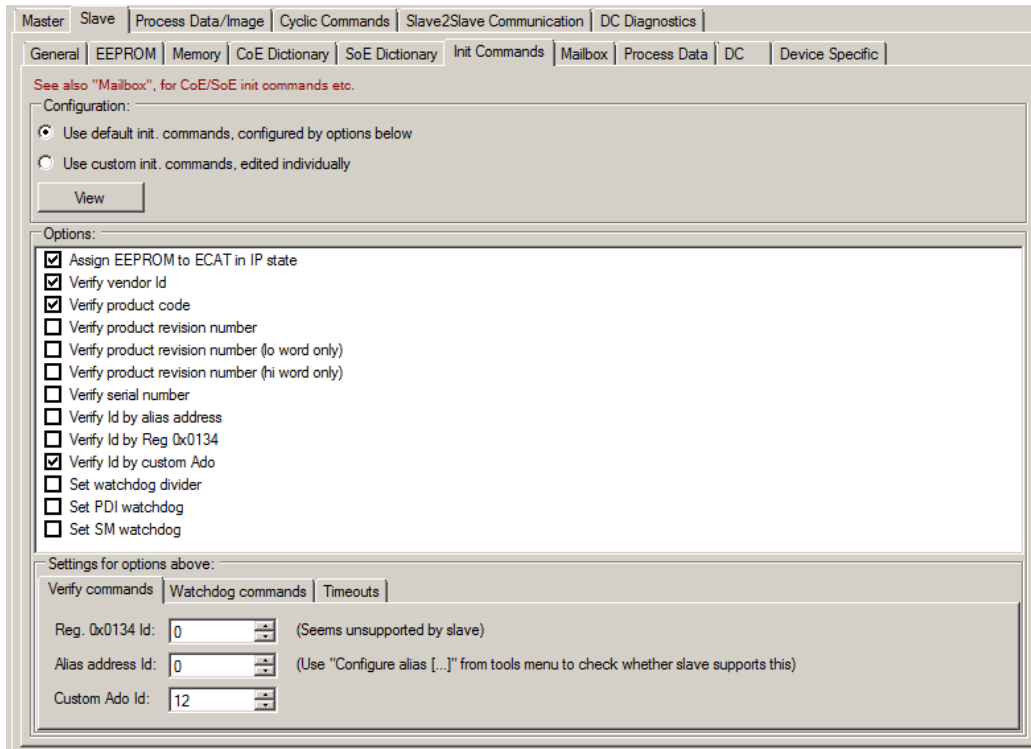


Fig. 14: Slave configuration, page “Init Commands”

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 slave address; when another slave 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 slave’s EEPROM to the EtherCAT master (slave reg. 0x0500) to the IP state transition. Required by most of the “Verify ...” options
- “Verify ...”
 - Adds commands to read and verify certain slave values, e.g. its vendor id: When the id differs from the expected one, the slave initialization will fail
- “Set watchdog divider”, “Set PDI watchdog” and “Set SM watchdog”
 - Writes the configured values (tab “Watchdog commands” below) to the corresponding slave registers (0x0400 etc.)

“Settings for options above”

- “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 slave, Alias usually configured by the Workbench (“Configure alias [...]” in “Tools” menu)
- “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
- “Watchdog commands”
 - Used with “Set watchdog divider”, etc. in “Options” above. For details see ESC documentation of watchdog registers

2.3.8 Mailbox settings

- Page “General”
 - “Start” and “Length” for the addresses taken from ESI, usually no modification should be needed (And slaves 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

2.3.8.1 Page “CoE”

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.

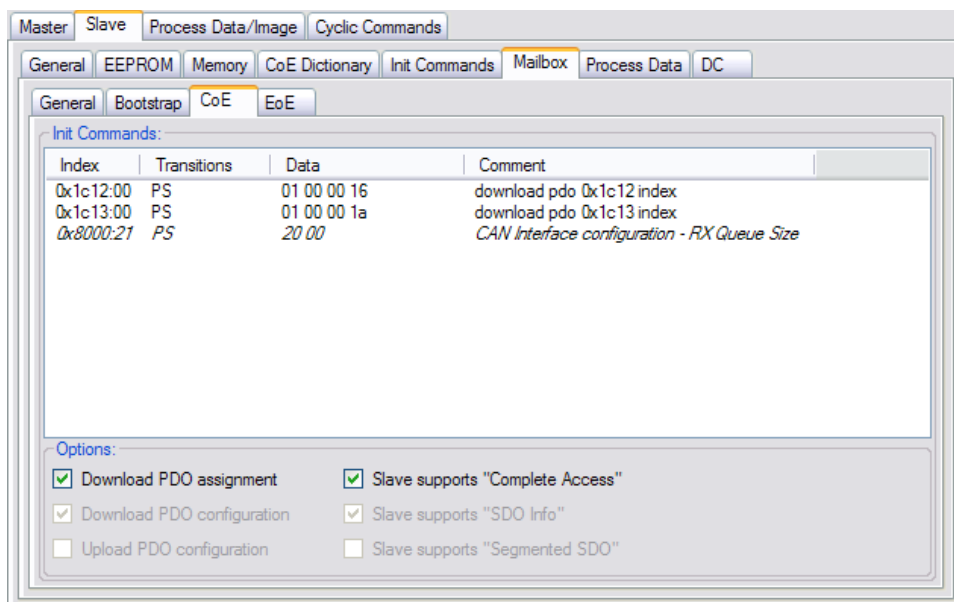


Fig. 15: Slave configuration, page “Mailbox”, “CoE”

“Options:”

- “Download PDO assignment”

EtherCAT network configuration

- 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
- “Slave supports Complete Access”
 - Determines whether these init. commands “combine” their data. Initial state set by ESI.
Does not affect custom commands when changed

Context menu

- “Edit selected item”
 - 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
- “Copy selected item to other slave(s)”
 - Opens a dialog to select other slaves 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 slaves.

When a command already exists at a slave’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

2.3.8.2 Page “EoE”

This page is used to configure the slave’s Ethernet settings when it supports EoE.

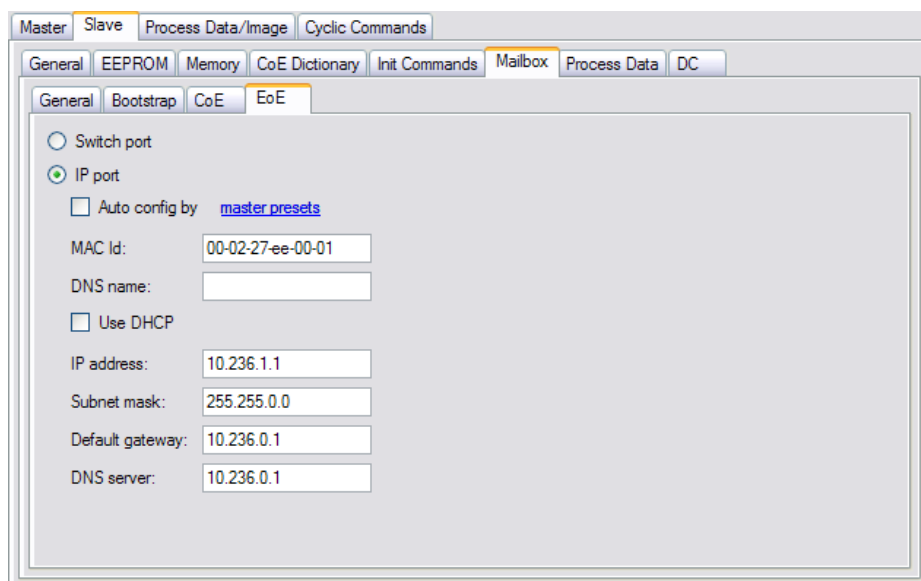


Fig. 16: Slave configuration, page “Mailbox”, “EoE”

- “Switch port”
 - Slave shall act as a switch port. No IP address is assigned and no further configuration is necessary

- “IP port”
 - Slave shall have its own IP address, etc. MAC Id, DNS name and other TCP/IP settings can be configured either manually or automatically by master’s EoE (section 2.1.6) presets.

2.3.8.3 Page “FoE”

This page is used for FoE up- and downloads.

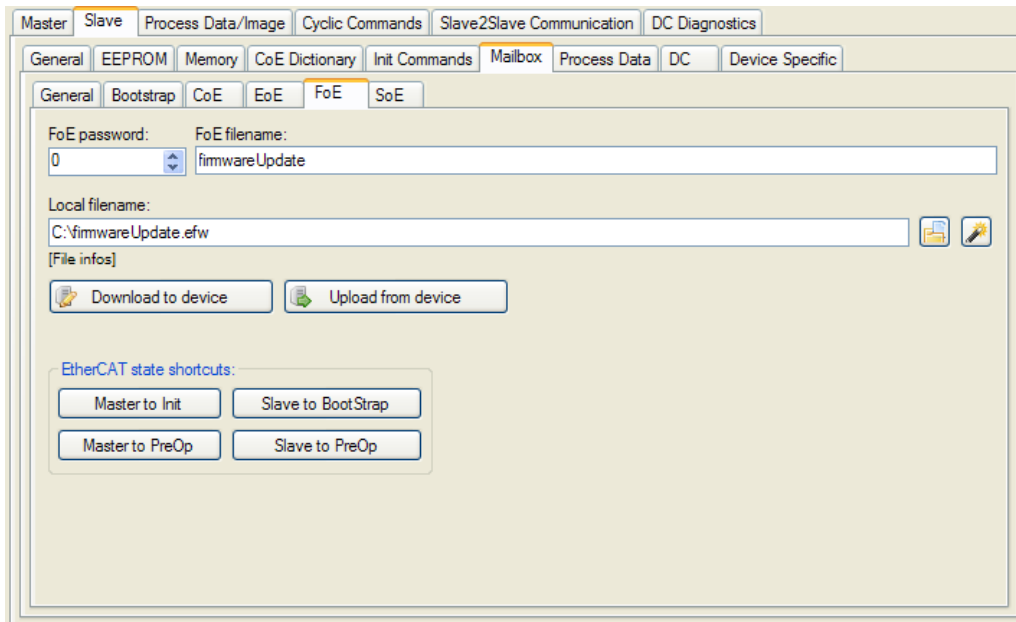




Fig. 17: Slave configuration, page “Mailbox”, “FoE”

- “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 infos]”
 - 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 slaves 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”

EtherCAT network configuration

- 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.

2.3.8.4 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 2.3.8.1.

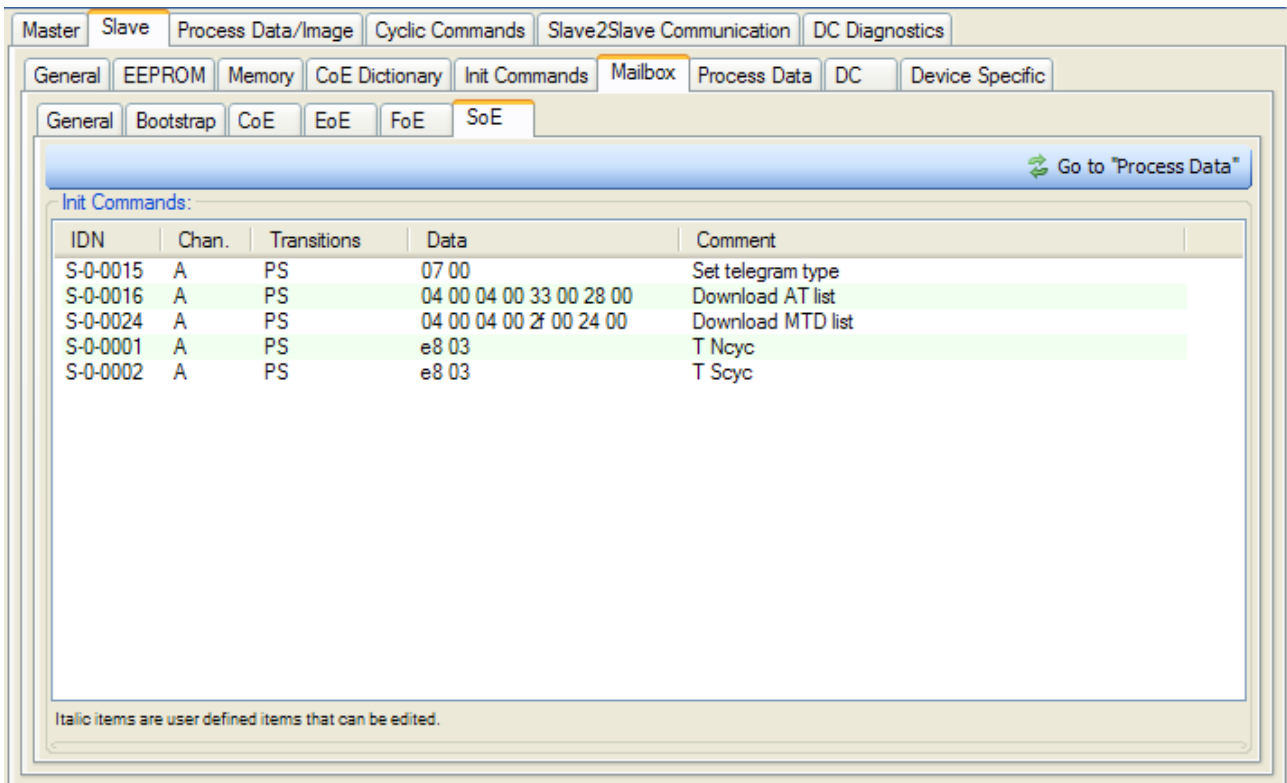


Fig. 18: Slave configuration, page “Mailbox”, “SoE”

2.3.9 Process data

This page is used to assign the PDOs to the sync manager and to configure the PDOs. When an item from “Sync Manager:” is selected the available PDOs are displayed next to it (“SMX PDO selection:”). Assigned PDOs are checked.

When items can’t be checked or unchecked this is caused by the items being marked to exclude each other or by being marked as mandatory. (In the ESI or online data when obtained online)

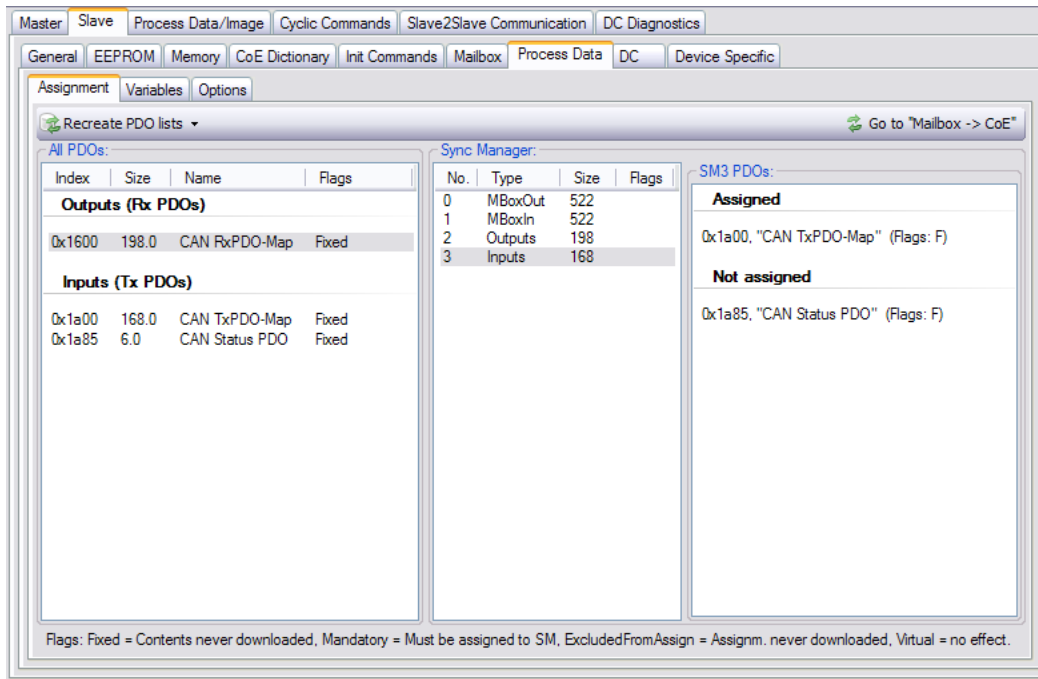


Fig. 19: Slave configuration, page “Process Data”

- “Recreate PDO lists”
 - “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”)
- Editing PDOs
 - The context menu of “All PDOs” allows to add/edit PDOs there
- Page “Variables”
 - Lists all the variables within the activated PDOs for that slave (a double click on an entry will switch to that entry on the “Process Data/Image” tab)
- Page “Options”
 - “Slave Supports LRW command”
 - Determines whether the selected slave 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 slave”

- When selected this slave's process data handling is not combined with other slaves process data, i.e. separate EtherCAT commands to read/write the slave's process data are always generated, regardless of global "Combine PD commands" setting (section 3.4)
- "LRD/LRW only in Op" / "LWR only in Op"
 - When selected EtherCAT commands to read/write process data from this slave 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 slave handles a command only in Op (Should be used with "separate commands" only, else other slaves might be affected accidentally, too)
- "Sync unit"
 - Used to define groups of which slave'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 slave's process data. Multiplier base is the master base cycle time, see 2.1.1

2.3.10 DC

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

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

2.3.11 Device Specific

The content of this tab page depends on the slave. It is determined by the slave MDP number in CoE object 0x1000 or by an internal list of known slaves. If the slave 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 baud 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.

Current available pages:

- "CAN Interface"
 - Selected for slaves supporting the MDP No. 5000
 - Allows setting the CAN baud rate, sending/receiving CAN frames (in "PreOp") and creating CoE init. commands for the baud rate
- "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.

2.3.12 Hot Connect

Hot Connect allows to declare slaves or groups of slaves as being optional, i.e. the EtherCAT network will work with or without these slaves and allow to add/remove them while running.

The Workbench will create separate commands to access the process data of these slaves to allow the master to achieve this. Additionally the master 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.

2.3.12.1 Example

Select the first slave of the group and “Create Hot connect group” in the context menu:

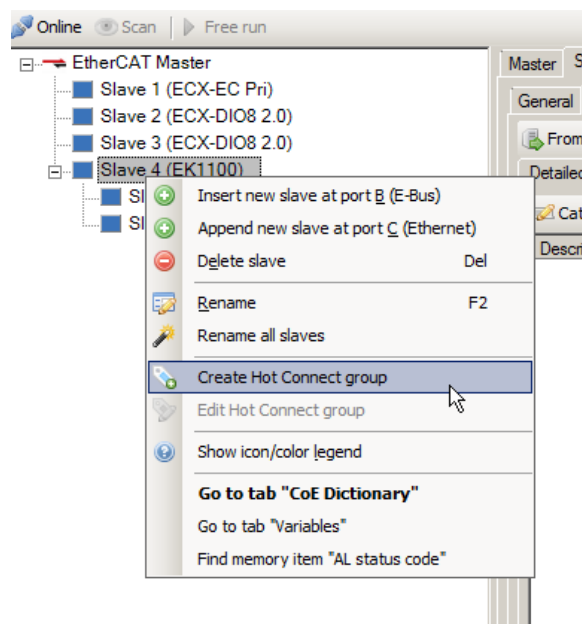


Fig. 20: Creating new Hot Connect group

The Hot Connect group editor window will show up:

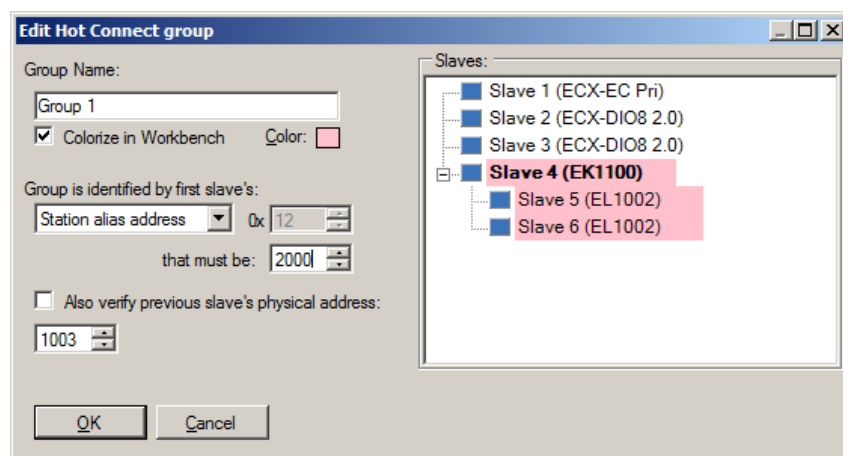


Fig. 21: Hot Connect group editor

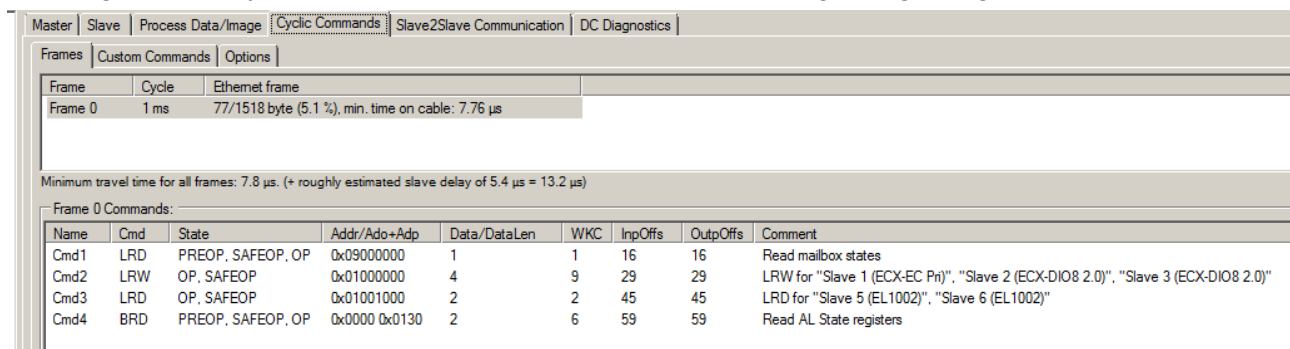
EtherCAT network configuration

- “Group Name” / “Colorize in Workbench”
 - Only used for display and not exported to ENI
- “Group is identified by first slave’s:”
 - A register and a value (“that must be:”) is selected here. (“Station alias address” = ESC Register 0x0012)
 - Additionally the previous slave physical address might be verified, too, to allow the Hot Connect group to be connected at a certain slave only
- “Slaves:”
 - Last slave 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 slave’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:



The screenshot shows the 'Frames' tab in the EtherCAT Workbench. It displays a table of cyclic frames for a group. The table has columns: Frame, Cycle, Ethernet frame, and a detailed description. Below the table, there is a section for 'Frame 0 Commands' with a table listing commands (Cmd1 to Cmd4) with their states, addresses, data lengths, WKC, InpOffs, OutpOffs, and comments.

Frame	Cycle	Ethernet frame	
Frame 0	1 ms	77/1518 byte (5.1 %), min. time on cable: 7.76 µs	

Minimum travel time for all frames: 7.8 µs. (+ roughly estimated slave delay of 5.4 µs = 13.2 µs)

Name	Cmd	State	Addr/Ado+Adp	Data/DataLen	WKC	InpOffs	OutpOffs	Comment
Cmd1	LRD	PREOP, SAFEOP, OP	0x09000000	1	1	16	16	Read mailbox states
Cmd2	LRW	OP, SAFEOP	0x01000000	4	9	29	29	LRW for "Slave 1 (ECX-EC Pri)", "Slave 2 (ECX-DIO8 2.0)", "Slave 3 (ECX-DIO8 2.0)"
Cmd3	LRD	OP, SAFEOP	0x01001000	2	2	45	45	LRD for "Slave 5 (EL1002)", "Slave 6 (EL1002)"
Cmd4	BRD	PREOP, SAFEOP, OP	0x0000 0x0130	2	6	59	59	Read AL State registers

Fig. 22: Tab “Frames” showing separate commands for Hot connect group

Process data for the Hot Connect group’s slaves are access by “Cmd3”

2.3.12.2 Remarks

- Hot Connect is primarily an EtherCAT Master 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 slave can’t be part of multiple groups, removing a group must not interrupt other slave’s connections

2.4 Cyclic commands

The “Frames” page shows an overview of all frames/commands that will be sent cyclically by the master.

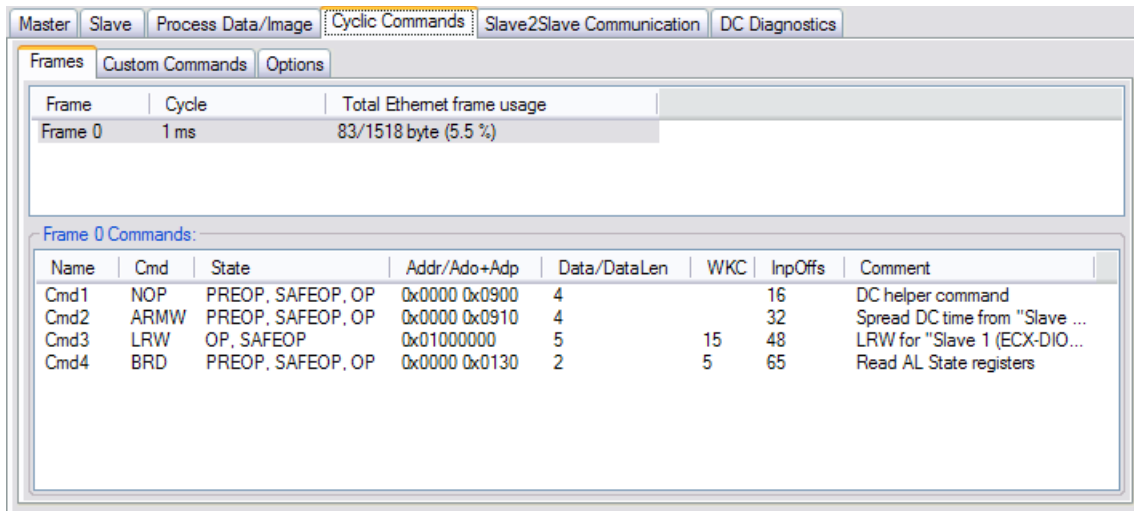


Fig. 23: Tab page “Cyclic Commands”

Page “Custom Commands”

This page allows adding custom cyclic command.

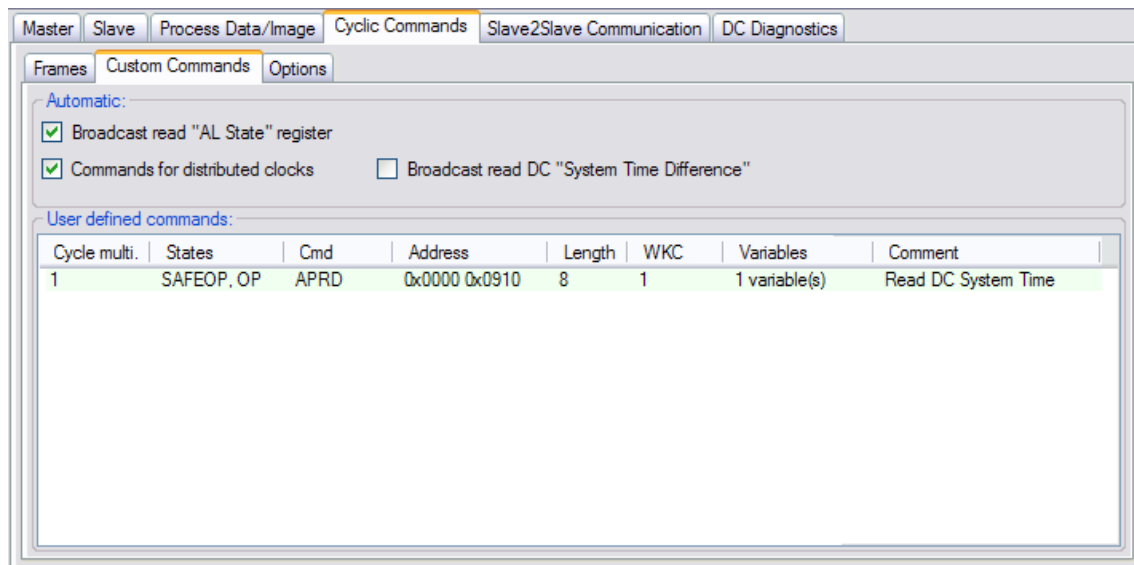


Fig. 24: Cyclic commands, tab page "Custom Commands"

- “Automatic:” Adds commands simply by checking these options:
 - “Broadcast read AL state registers”
 - Adds a “BRD” command to read all slave’s 0x0130 registers. (The resulting WKC might be used to determine the number of active slaves)
 - “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 slave’s 0x092c register. (Result can be used to determine whether all slaves are still synchronized)
- “User defined commands:” Allows adding custom commands. Use the context menu to edit 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 from “NOP” to “APRD”
 3. Set the addresses: Slave: 0x0000 (the first slave), Physical: 0x0910 (the “DC System Time” register)
 4. Set data length to 8 (assuming the slave 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 (section 3.4.1) unusable

2.5 Slave2Slave Communication

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

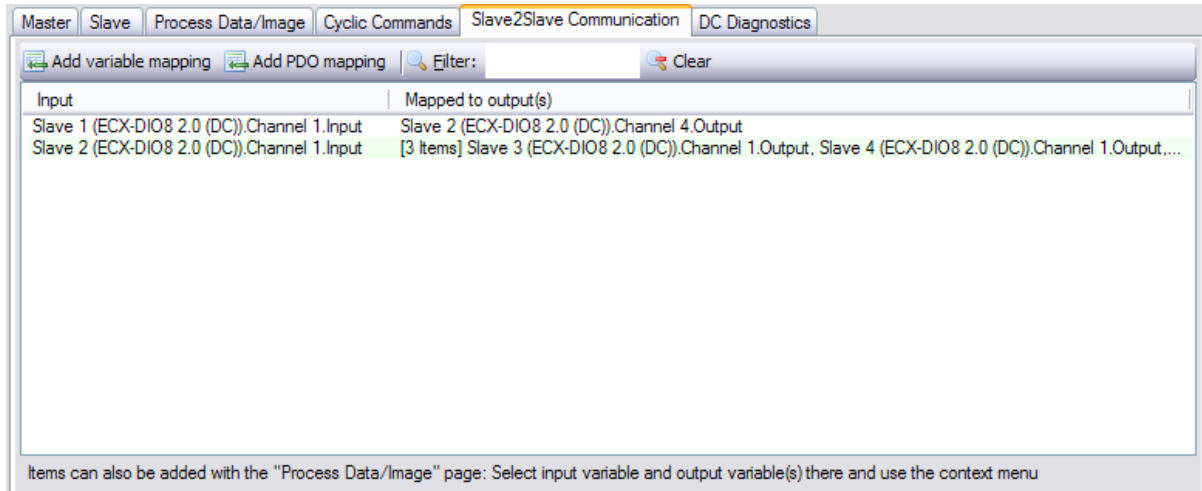


Fig. 25: Slave2Slave Communication: Current mappings

The copy operation is performed by the master 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 Fig. 26 will show up:

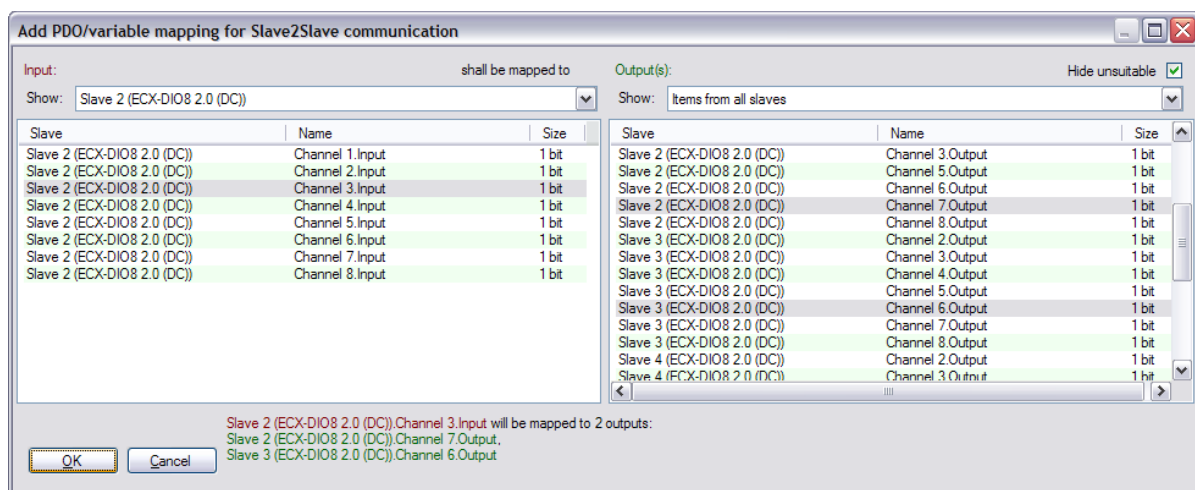


Fig. 26: Adding a variable mapping

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 slave. 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.

2.6 Network Topology

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

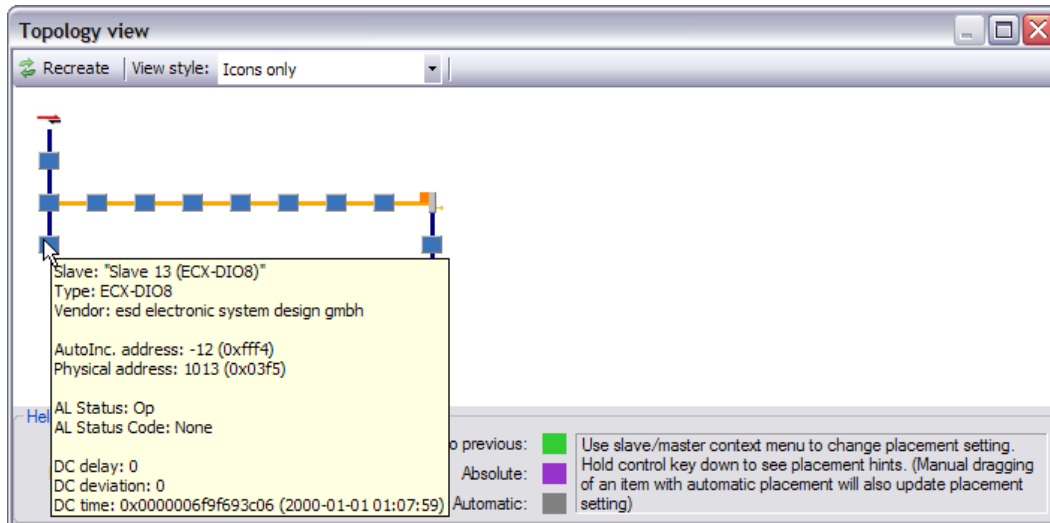


Fig. 27: Topology, view style "Icons"

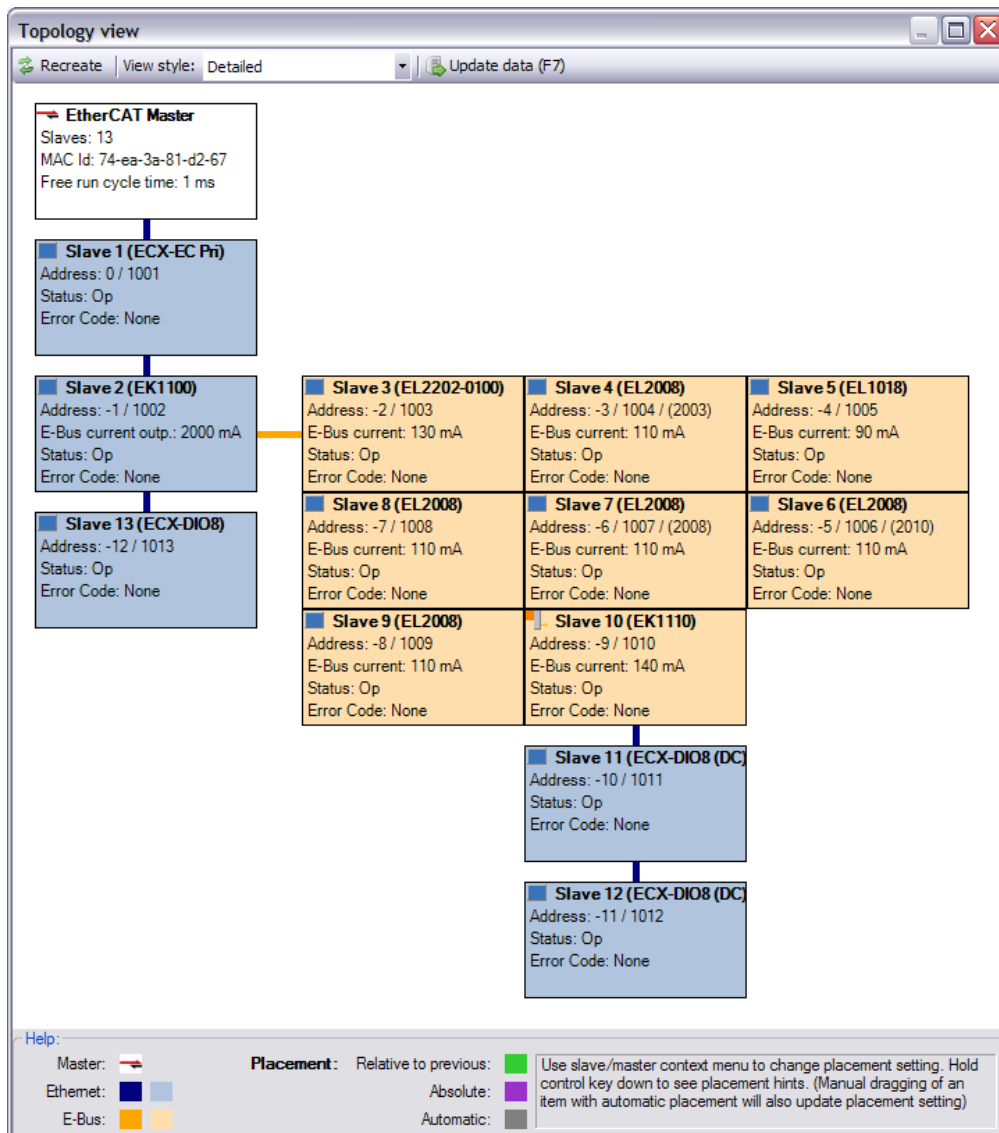


Fig. 28: Topology, view style "Detailed"

Rules for slave placement

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

Manual options

To customize the automatic layout each item (master or slave) has a placement setting (accessible by its context menu) and can be moved manually with the mouse. (This was done for Fig. 28)

- “Auto” (Default)
 - Placed either below or next to previous slave as described above
- “Manual/Relative”
 - Placed relative to the previous slave
- “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 slave’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 slave tree view, see Fig. 13
- “Detailed”: Large text boxes with some general information within, see Fig. 28
 - “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 settings should be restarted.

“Update Data (F7)”

- Used to update online values (i.e. re reading register values in “Detailed” view styles etc.)



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

3. Online configuration

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

3.1 Connection to the EtherCAT Master

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

The screenshot shows the 'General' tab of the EtherCAT Master configuration window. It includes sections for 'General settings' (Base cycle time, Destination MAC address, VLAN Id), 'esd EtherCAT Master settings' (Connection: 127.0.0.1, Show master details, Test connection RTT, Send ping), 'NIC' (Primary adapter, Redundant adapter, Source address(es), Set on master button), and 'Other' (Max acyclic frames, DC start time, Reduce frames, Acycl. frame timeout, EEPROM delay, Disable slave SM watchdogs for workbench). A reminder at the bottom states: 'Reminder: These settings are esd specific and not exported to the ENI.'

Fig. 29: Master configuration, page “General”

The “Connection:” text field determines the protocol, IP address and port of the master. Usually it is running locally 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 here¹⁰.

Connection is started by the “Online” button on the main form then. When the connection is established the button will be “down” or highlighted to signal this. Clicking the button again then will disconnect the application from the EtherCAT master.

When the connection is established a network interface on the master has to be selected. This is done in the “NIC:” pane on the “Settings” page of the master. (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)

After selecting the NIC the network is ready and the master is in “configuration mode” which means:

- All slaves in state “PreOp”
- Process data access, “SafeOp”/“PreOp”, etc. **not** possible in this mode
- A slave scan can be performed (section 3.2)

For process data access, etc. the master has to be set to “free run mode”, see section 3.3.

¹⁰ Please refer to the EtherCAT Master manual/READMEs for details about starting the Master on that remote target.

3.2 Slave scan

Used to populate/update the slave 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 is shown:

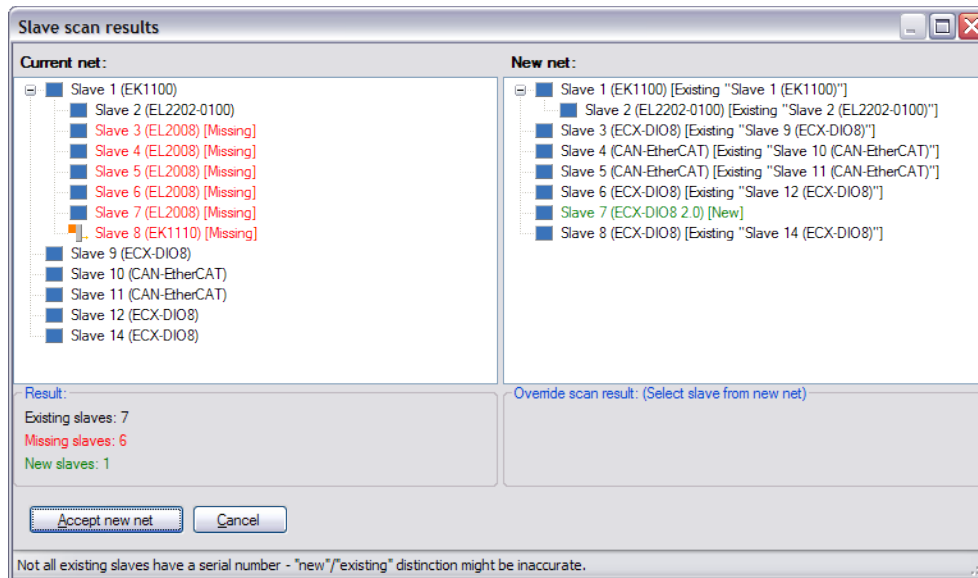


Fig. 30: Dialog window after an online scan

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” slave’s settings (Excluding their names) are kept.

A slave is identified by its vendor id, product code, revision number and serial number. But many slaves have its serial number set to “0”, also a slave 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 slaves. Select the new slave in the tree view above and then decide which of the current slaves it is (or “[New slave]” if it shall not replace an existing one) by this drop down list.

It’s strongly recommended to do a slave scan every time the master connection is established. Most commands require the slave 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 slave’s EEPROM in this situation can lead to writing it to a different slave!

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 master and the master sends all configured initialization commands, cyclic frames, etc. Therefore all slaves should be in operational state and reading/writing process data is possible, too.

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

- “Slave error event”

> Time	Text
2011-02-03 07:03:48.3	Master: Slave error event [ECM_EVENT_SLV_INIT_ERROR] for "Slave 10 (CAN-EtherCAT)"

Fig. 31: “Slave error event” example

- ECM_EVENT_SLV_ID_ERROR
 - Slave Id verification failed, usually caused by a slave exchange
- ECM_EVENT_SLV_INIT_ERROR
 - General initialization error. When no other messages occur, checking slave error registers (See 2.3.4) might help to find the cause
- ECM_EVENT_SLV_NOT_PRESENT
 - Slave not found, determined by reading slave’s status registers
- SM error

> Time	Text
2011-02-04 09:03:53.7	Master: CoE emergency from slave "Slave 10 (CAN-EtherCAT)" received: [SM settings error]

Fig. 32: Slave “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 (Slave’s “Process data” → “Assignment” Tab) and make sure this will fit into the SM buffers (See Slave’s ESI, <Sm> node, “StartAddress” attribute, and make sure SMs don’t overlap etc. – the Workbench will warn about that, too)



Note that changing to Free run involves a complete slave 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 slave's process variables and some additional virtual variables.

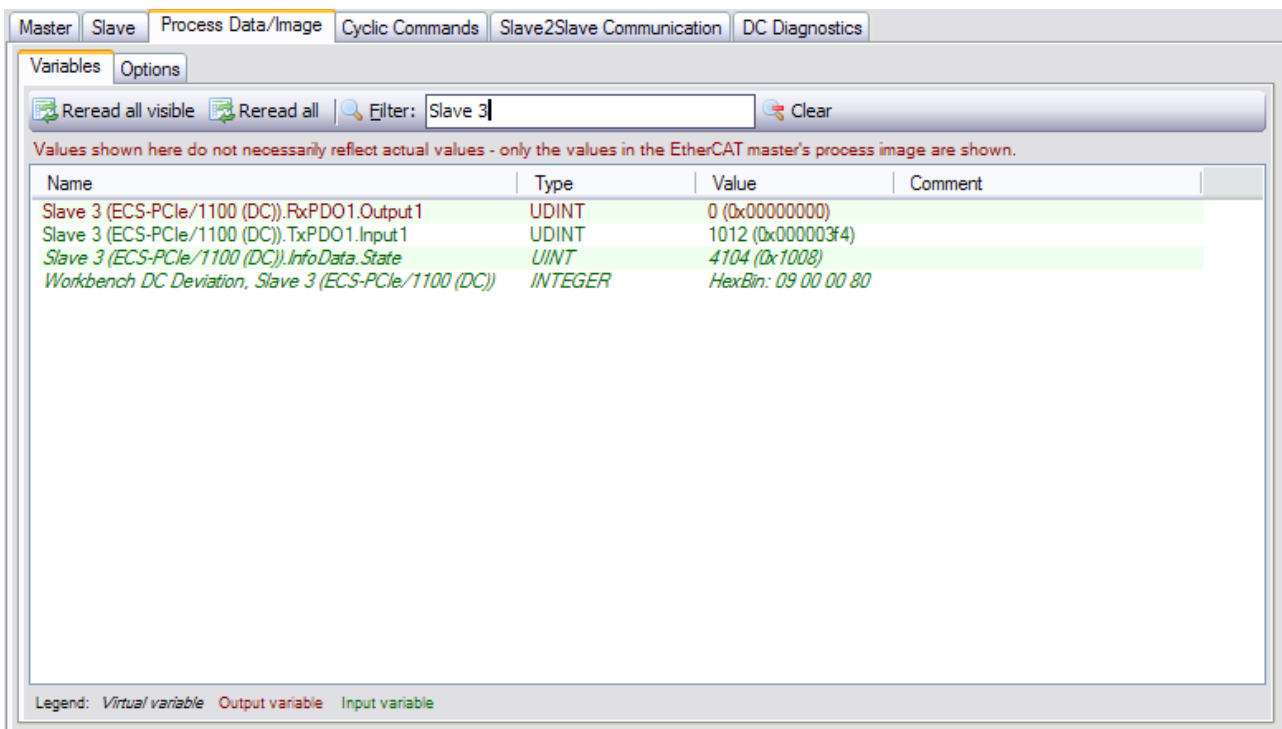


Fig. 33: Tab page “Process Data/Image”

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.

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)



All variables are only written to / read from the master's process image. This means any input read was not necessarily updated by slave and an output written is not necessarily successfully transferred to slave, too.

Additionally the outputs are only transferred to master after editing, i.e. when the master resets/overwrites the values the Workbench does not re-transfer the value.

Additionally mapping entries for “Slave2Slave 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 Slave2Slave Communication”. (Variables must be of same size, see 2.5)

Page “Options”

- “Combine PD commands”
 - Determines whether the EtherCAT commands to handle slave process data shall be combined when possible. When enabled, multiple slave’s process data is handled by a single EtherCAT command – achieved by choosing the logical addresses accordingly.
To exclude only single slaves from this, see slave process data options (section 2.3.9)
- “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 master you are using! (Although current ETG.2100 document (V1.0.0) seems to describe this as the “checked” case, some master vendors handle this differently¹¹)
- “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
- Button “Set values to Frame layout”
 - Sets the process data options to default values for process images that contain whole Ethernet frames
- Button “Set values to packed layout”
 - Sets the process data options to default values for process images that contain only process data

¹¹ The Workbench adds an <PILayout> element with attribute “IncludeCmdHeader” to the <VendorSpecific> element of the ENI’s <Master> element to distinguish this.

3.4.1 Virtual variables

- “Inputs.FrmXWcState”:
 - 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.SlaveCount”
 - Number of active slaves, determined by reading “AL state” registers. (Enabled in “Options” page of “Cyclic Commands” tab, see 2.4)
- “Inputs.SlaveCount2”
 - Number of active slaves at redundant network interface
- “Inputs.DevState”:
 - Current master state flags (Also shown with more details in master 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 slave in state “Init”
 - Bit 9 set: At least one slave in state “PreOp”
 - Bit 10 set: At least one slave in state “SafeOp”
 - Bit 11 set: At least one slave indicates an error
 - Bit 12 set: DC not in sync
- “InfoData.CfgSlaveCount”
 - Total number of slaves in ENI
- “Slave.InfoData.State”:
 - Bit 0 to 3: Slave state
 - 0: Unknown, 1: “Init”, 2: “PreOp”, 3: “Bootstrap”, 4: “SafeOp”, 8: “Op”
 - Bit 5: Id verification error
 - Bit 6: Initialization error
 - Bit 8: Slave 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 slave register values.

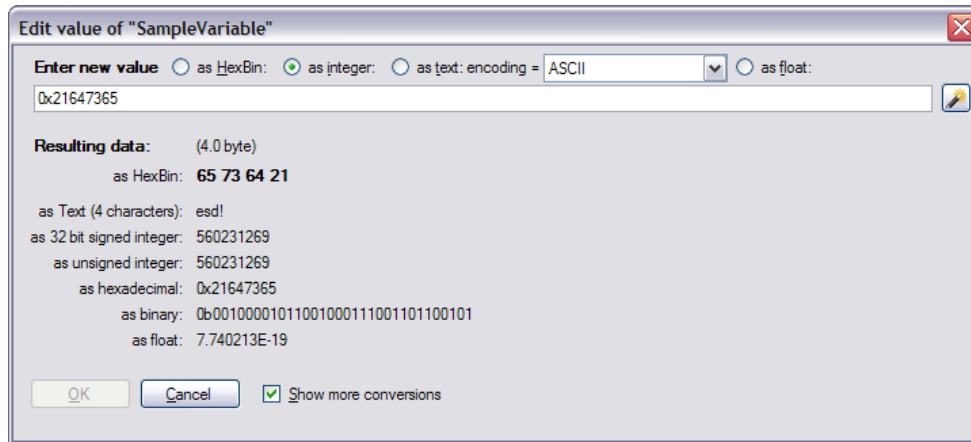


Fig. 34: Variable editor window

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 displayed 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"
 - 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"
 - 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"

- 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

- 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

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 slave 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)



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

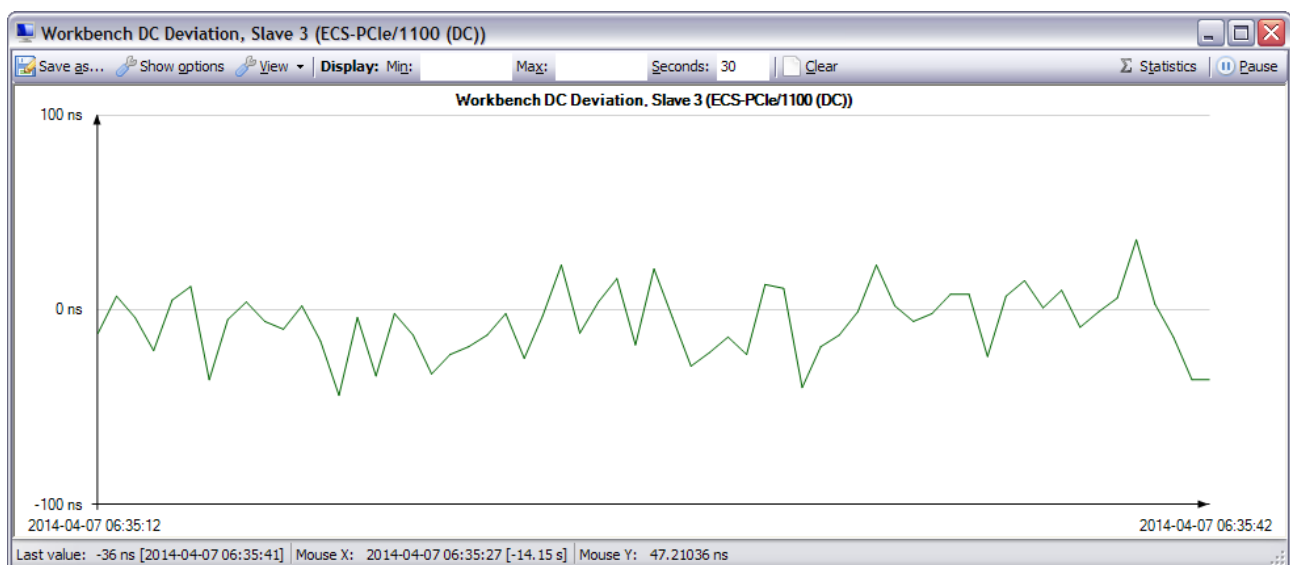


Fig. 35: Variable monitor window

Toolbar

- Button “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)
- Button “Show options”
 - Shows or hides the options pane, see description below
- Drop down list “View”
 - “Change title”
 - Shows a text input dialog to enter a new text for the image and window title
 - “Draw data points”

Online configuration

- 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)
- “Disable anti aliasing”
 - Determines whether anti aliasing for all lines should be disabled or not
- “Display:”
 - “Min” / “Max” selects the lower and upper limit of the display. Leave empty for automatic selection (Left mouse button on the label to reset, right to select value from current data points)
 - “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” 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

Visible when button “Show options” is down. Allows logging and modifying the data before it is displayed.

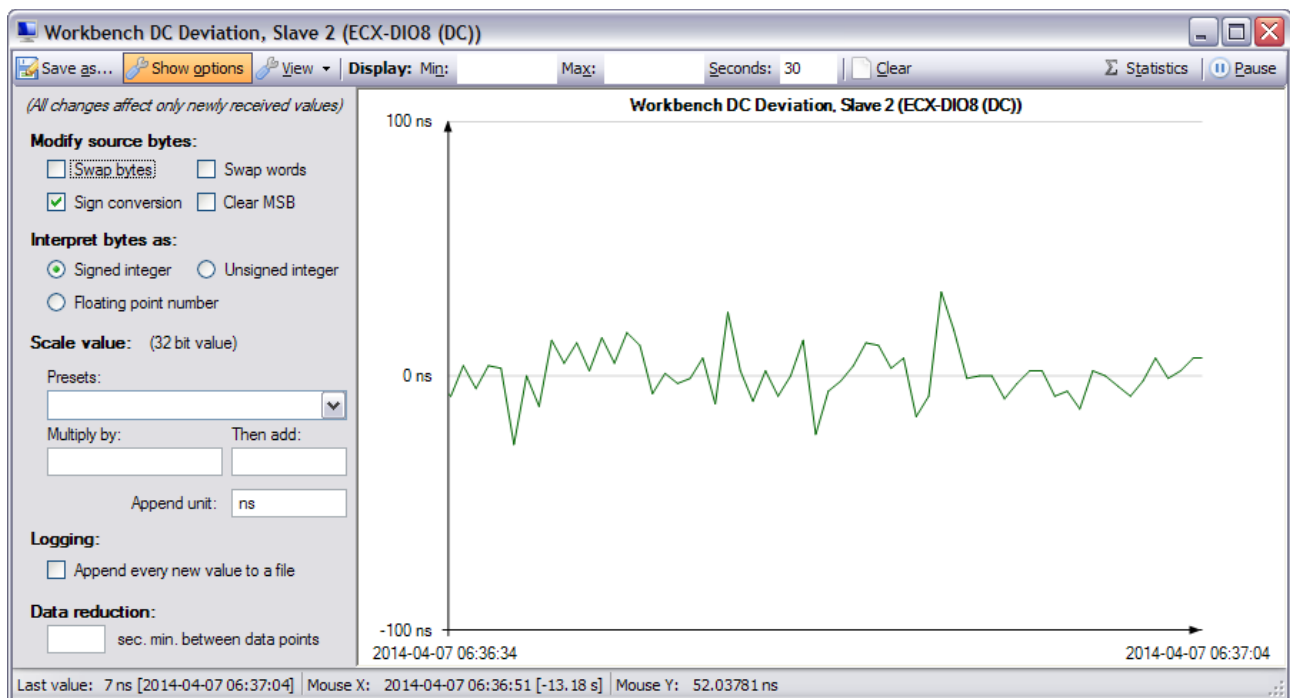


Fig. 36: Variable monitor window, options visible

- “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 (see 1.4.1 “Menu items” for information about how to change it)
 - <Tab> is a tab character (decimal 9)
 - <Value> is the value that is displayed (not the source bytes etc.)
 - Example line:


```
2010-12-16 12:22:55 1.2345
```
- “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.)

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.

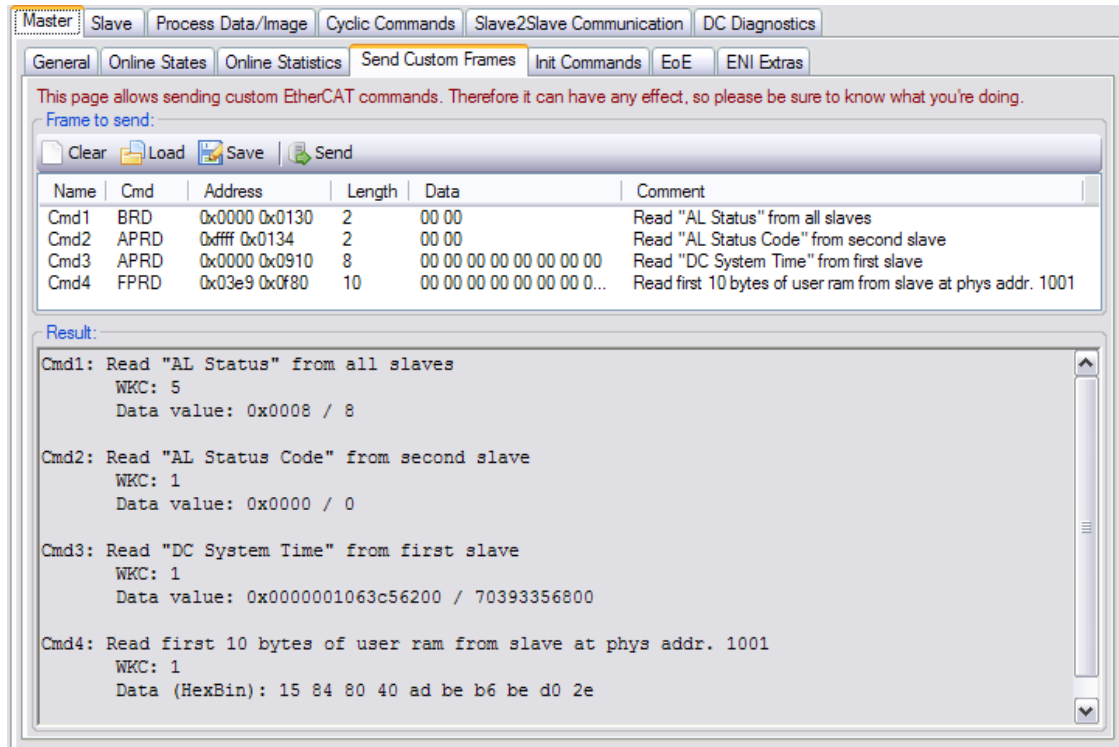


Fig. 37: Sample custom frame with result

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").

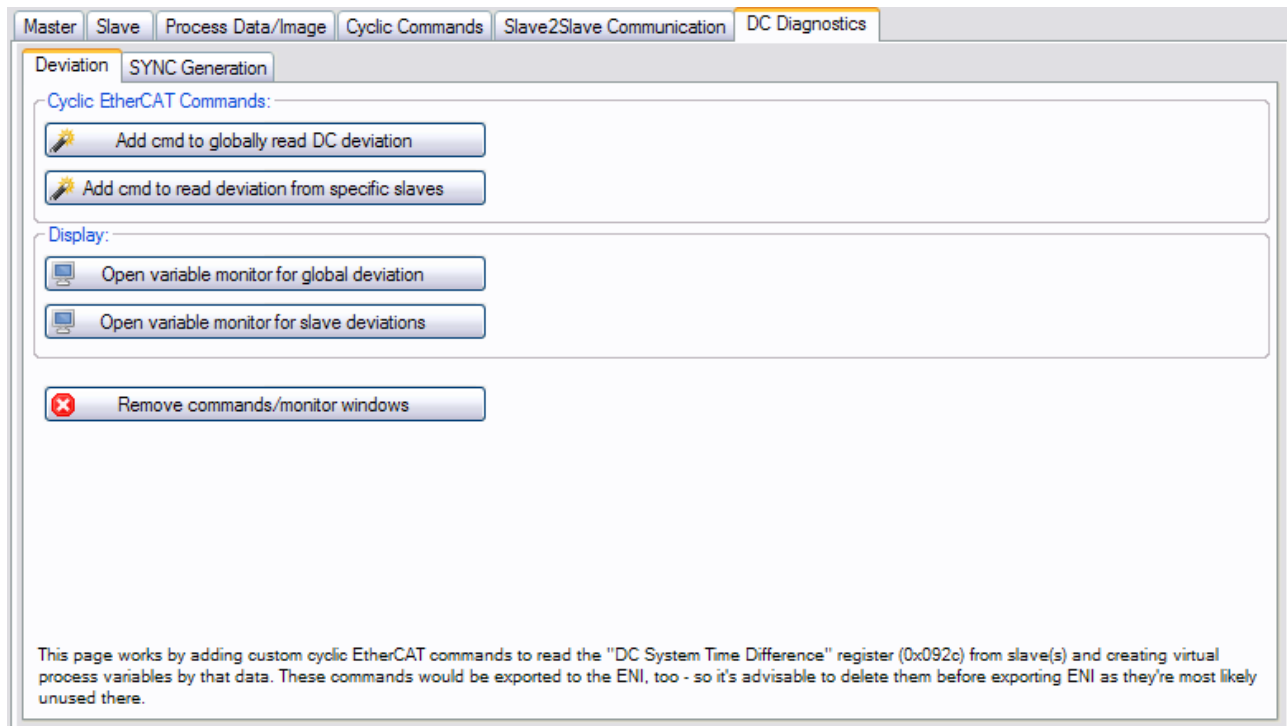


Fig. 38: DC Diagnostics, tab page "Deviation"

- **"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 ORs all slave's values into a single value)
- **"Add cmd to read deviation from specific slaves"**
 - Shows a dialog to select slave(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 slave deviations"**
 - Opens a variable monitor window for the according process variable – just a shortcut for doing this in the Process data 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 slave's DC SYNC units while online. It requires the slave's SYNC units assigned to EtherCAT and their clocks should be synchronized.

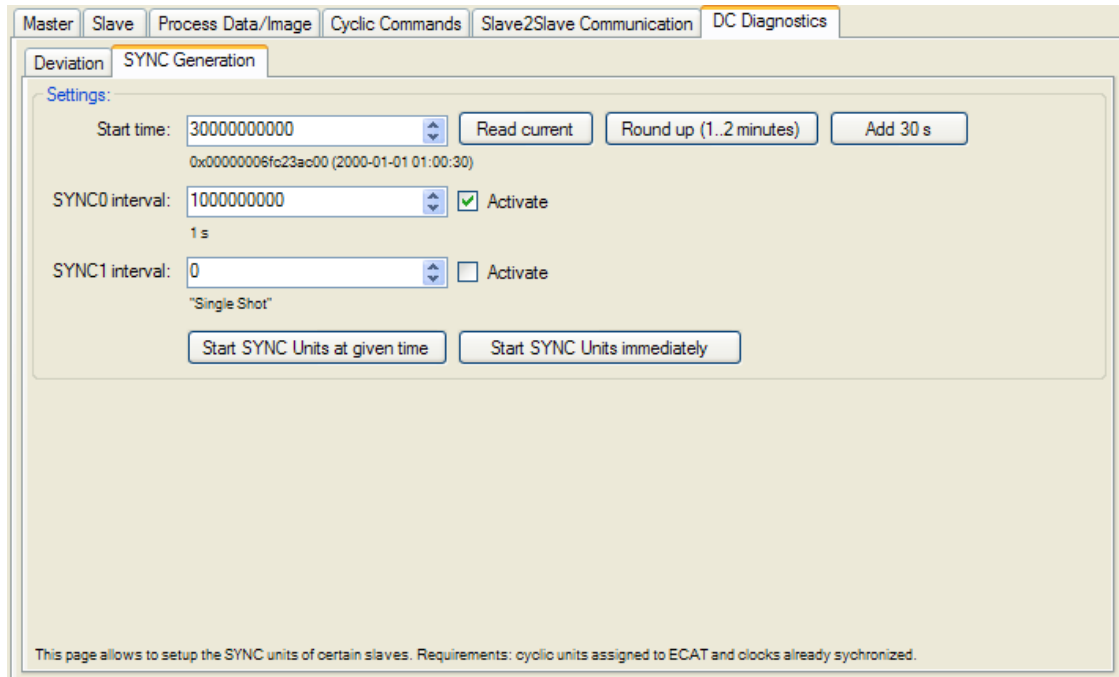


Fig. 39: DC Diagnostics, tab page "SYNC Generation"

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

"Start SYNC Units at given time"

First a dialog to select the slave(s) whose SYNC units shall be edited will be shown. Then the values entered here will be written to the according slave 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 slaves (button "Read current") and certain values can be added automatically by "Round up ..." and "Add 30 s" buttons. (So this is intended only for slaves 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 slaves and a small amount is automatically added so the SYNC units start almost immediately. (So this works with 32 bit DC timestamps as well)

3.7 Monitor mode

This mode is determined by the master the Workbench connects to. If the master is running in monitor mode it is driven by its own application – the Workbench is used for monitoring only.

Connecting to a master running in monitor mode requires the correct Workbench project file to be loaded, i.e. the master's ENI must have been created by the Workbench and exactly that project file has to be used.

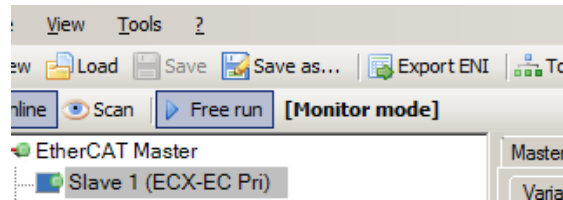


Fig. 40: Monitor mode label

Monitor mode is recognizable by the “[Monitor mode]” label next to the “Free run” button, see Fig. 40.

In monitor mode the “Free run” mode is always active, as the master already runs by its ENI and the Workbench is not used to create one any more. Therefore any change to the project that affects the ENI even renders the project incompatible to that running master. (To avoid using different process image layouts for example)



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

4. EtherCAT Workbench Scripting

4.1 Overview

The EtherCAT Workbench includes [IronPython](#)¹² (Version 2.6.2) to allow you to automate some Workbench tasks or to customize the Workbench.

Any “.py” file in the Workbench’s “Scripts” sub folder¹³ 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 slave 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 4.4 “Available variables etc.”. 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”

This menu 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).

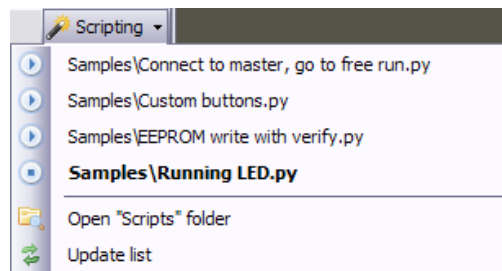


Fig. 41: Scripting menu

When a stopped script is clicked, it’s started and its symbol and font changes as Fig. 41 shows for “Samples\Running LED.py”.

When a running script is clicked it’s 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)

¹² 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>

¹³ 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 section 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 infos about the error.

Result	Method	Arguments	Description
void	ScriptAbort	(string errText)	Ends the script immediately. (Aborts the thread)
void	ScriptSleep	(int ms)	Waits inside the script. (ms = milliseconds)
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.
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”.
void	AddLogInfo	(string txt)	Adds an “Info” entry to the Workbench’s messages log.
void	AddLogWarning	(string txt)	Adds a “Warning” entry to the Workbench’s messages log.
void	AddLogError	(string txt)	Adds an “Error” entry to the Workbench’s messages log.
bool	IsOnlineBusy	()	Returns “True” when the Workbench’s online buttons are disabled, i.e. an asynchronous action is still active.
MasterMode	GetMasterMode	()	Returns the current master state – does not necessarily reflect the actual EtherCAT state. (PreOp, Op, etc.)
CallbackResult	VarRequestUpdate*	(string varName)	Triggers a reread of an input variable. (Equals rereading variable in the Workbench’s “Process Data/Image” tab)

EtherCAT Workbench Scripting

			Result's "Data" is: null.
CallbackResult	VarRequestUpdateAll*	(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.
CallbackResult	VarGetData Type	(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.
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.
CallbackResult	VarRequestWrite*	(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.
CallbackResult	VarRequestWriteXX*	(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.
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[].
CallbackResult	VarGetValueXX	(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.
CallbackResult	LoadProject*	(string fileName)	Loads a Workbench project from file. Omits the "are you sure" dialog when current project is unsaved. Result's "Data" is: null.
CallbackResult	NewProject*	()	Starts a new Workbench project from file. Omits the "are you sure" dialog when current project is unsaved. Result's "Data" is: null.
CallbackResult	RequestOnlineMode*	(bool online)	Requests to go online or offline. (Equals button "Online") Result's "Data" is: null.
CallbackResult	SlaveSelectByAddr	(int autoIncAddr)	Selects a slave in the current slave list by its auto increment address. Result's "Data" is: null.
CallbackResult	SlaveSelectByName	(string name)	Selects a slave in the current slave list by its name. Result's "Data" is: null.
CallbackResult	SlaveEEPROMEditorLoadFromFile	(string fileName)	Loads an EEPROM from file to editor. (Affects the slave that is currently selected. Equals button "From file" in the slave EEPROM editor tab) Result's "Data" is: null
CallbackResult	SlaveEEPROMEditorSaveToFile	(string fileName)	Saves EEPROM from editor to file. (For the slave that is currently selected. Equals button "To file" in the slave EEPROM editor tab) Result's "Data" is: null.
CallbackResult	SlaveEEPROMEditorRequestWrite*	()	Requests writing the current EEPROM to slave. (For the slave that is currently selected. Equals button "To device" in the

			slave EEPROM editor tab) Result's "Data" is: null.
CallbackResult	SlaveEEPROMEditorRequestRead*	()	Requests reading the current EEPROM from slave. (For the slave that is currently selected. Equals button "From device" in the slave EEPROM editor tab) Result's "Data" is: null.
CallbackResult	SlaveEEPROMEditorGetMD5	()	Gets the MD5 checksum from the current EEPROM editor data. (For the slave that is currently selected) Result's "Data" is: string (md5 sum).
CallbackResult	SlaveEEPROMEditorClear	()	Clears current EEPROM editor data. (For the slave that is currently selected) Result's "Data" is: null.
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.
CallbackResult	RequestFreerunMode*	(bool freeRun)	Triggers free run mode. (Equals button "Free run") Result's "Data" is: null.
CallbackResult	SetMasterConnectionString	(string data)	Sets the text in the "Connection:" text box within the "Master/General" tab. Result's "Data" is: null.
CallbackResult	SetMasterNICs	(string primary, string redundant)	Sets the NIC(s) the master 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.
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).
CallbackResult	UpdateCustomButton	(int btnHandle, string caption, string tooltip)	Updates caption and tooltip of an existing custom button. Result's "Data" is: null.
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).
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.
CallbackResult	RequestSlaveScan*	()	Triggers a slave scan. (Following "Slave scan results" window is closed automatically when no slaves existed yet) Result's "Data" is: null.
CallbackResult	CoERequestPDOResult*	()	Triggers updating the list of available PDOs. (For the slave that is currently selected. Equals "Recreate list of avail. PDOs → Online..." in the slave Process Data tab) Result's "Data" is: null.
CallbackResult	CoERequestDictionaryRead*	()	Triggers recreating the CoE dictionary from online data. (For the slave that is currently selected. Equals "Recreate dict. → Online..." in the

EtherCAT Workbench Scripting

			slave CoE Dictionary tab) Result's "Data" is: null.
CallbackResult	CoERequestObjectRead*	(ushort objIndex, byte subIdx)	Triggers rereading a CoE object. (For the slave that is currently selected. Equals "Reread..." in the slave CoE Dictionary tab) Result's "Data" is: null.
CallbackResult	CoERequestObjectWrite*	(ushort objIndex, byte subIdx, byte[] data)	Triggers writing a CoE object. (data in little endian order) (For the slave that is currently selected. Equals editing the object in the slave CoE Dictionary tab) Result's "Data" is: null.
CallbackResult	CoEGetObjectValue	(ushort objIndex, byte subIdx)	Reads the value that is currently available in the selected slave's CoE Dictionary tab. (Use CoERequestObjectRead() to update it first) Result's "Data" is: byte[].
CallbackResult	CoESetFilterText	(string text)	Sets the filter text in the CoE Dictionary tab page. Result's "Data" is: null.
CallbackResult	ENIExport	(string fileName)	Exports the ENI file. Result's "Data" is: null.
CallbackResult	AddSlave	(int? prevSlave, int? prevPort, uint vendorId, uint prodCode, uint? revNo)	Adds a slave to the tree view. prevSlave is the auto inc. address of the prev. slave where it shall be added, if null it's added as last slave. prevPort is the EtherCAT port of prevSlave to connect to: [0..3]. vendorId, prodCode and optionally revNo identify the slave that shall be added. If both prevSlave and prevPort are null the last suitable slave is searched and used. Result's "Data" is: null.
CallbackResult	AddBeckhoffSlave	(int? prevSlave, int? prevPort, string bechhoffString)	Just a wrapper for AddSlave() with fixed vendorId (2) that determines prodCode and revNo from bechhoffString. That is e.g. "EK1100-0000-0018". (Or only "EK1100" to use latest/best revNo)
CallbackResult	ShowWorkbench	()	Shows the Workbench window when it is hidden. Result's "Data" is: null.
CallbackResult	GetSlaveCount	()	Returns the current slave count as seen in the tree view. Result's "Data" is: int.
CallbackResult	GetSlaveProperty	(int slaveAddr, string propName, object propArgs)	Retrieves a slave's property, see below for details.
CallbackResult	SetSlaveProperty	(int slaveAddr, string propName, object propArgs, object propData)	Sets a slave's property, see below for details.
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 slave. slaveAddr is the auto inc. address of the slave. If transitions is null/None, default ("PS") is used. complAcc defaults to slave's support for complete access, timeOut (in ms) defaults to 0.
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.

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.
----------------	------------------------------	--------------	--

* After this method it should be waited until "IsOnlineBusy()" returns "false", see also 4.5.2

GetSlaveProperty

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

SetSlaveProperty

propName	propArgs	propData	Description / Data content
Name	unused	string	New slave name.
AssignedPDOs	unused	int[]	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.
PDOContent	int	tuple[]	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.
SupportsCompleteAccess	unused	bool	Equals the "Slave supports Complete Access" checkbox described in 2.3.8.1.

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	Any other error occurred. CallbackResult's "Data" member might be a string with some more details.
Ambiguous	Action not possible because ambiguous. (e.g. when trying to add Slave by Beckhoff String and different matching slaves exist)
CallbackFailed	Should happen only when the Workbench is closing and cannot complete the script calls any longer.

EtherCAT Workbench Scripting

CallbackUnknown	Should not happen. (Used internally during development)
Busy	Call is not possible at the moment, usually because the “online buttons” are disabled.
InvalidArgument	Any method argument is invalid/not allowed.
Offline	Action not possible when offline.
WrongECATState	Action not possible in current EtherCAT state. (e.g. process variable access when not in Op)
VarNotFound	Variable was 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 slave itself, and so on)
ConversionFailed	Any data conversion failed. (Usually when accessing process variable’s data)
NoSlave	No slave is selected or was found. (e.g. when accessing slave’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 master.
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	Slaves should be in PreOp. (“Free run” button not down)
Freerun	“Free run” button down, slaves should be in “Op”.

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 slave and the user selects another one then, the script might perform its following actions with the wrong slave.

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 behavior 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 “`eeew.VarRequestUpdate()`” consecutively: The second call is likely to return with “`eeew.ErrorCode.Busy`” because the previous request has not yet completed.

To avoid such problems use e.g. “`eeewHelper.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
<code>/scriptsautorun</code>	Tells the Workbench to monitor its Scripts folder and automatically start new script files
<code>/logfile</code>	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)
<code>/hidden</code>	Hides the Workbench window, useful of course only if a script file was also given as argument or <code>/scriptsautorun</code> is used
<code>/scriptsargs</code>	<p>The argument afterwards determines a string that is made available to all scripts via the <code>eeewScriptsArgs</code> variable. May not start with <code>/</code> and can not be used multiple times. Example: <code>/scriptsargs "-myArg1 -myArg2='1 2 3'"</code></p> <p>The quotes <code>"</code> are required to pass the string that is separated by spaces as single argument - they’re removed in <code>eeewScriptsArgs</code></p>

5. Order Information

Type	Properties	Order No.
EtherCAT Workbench - License 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

PDF Manuals

Manuals are available in English and usually in German as well. For availability of English manuals see table below.

Please download the manuals as PDF documents from our esd website www.esd.eu for free.

Manuals		Order No.
EtherCAT Workbench-ME	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.