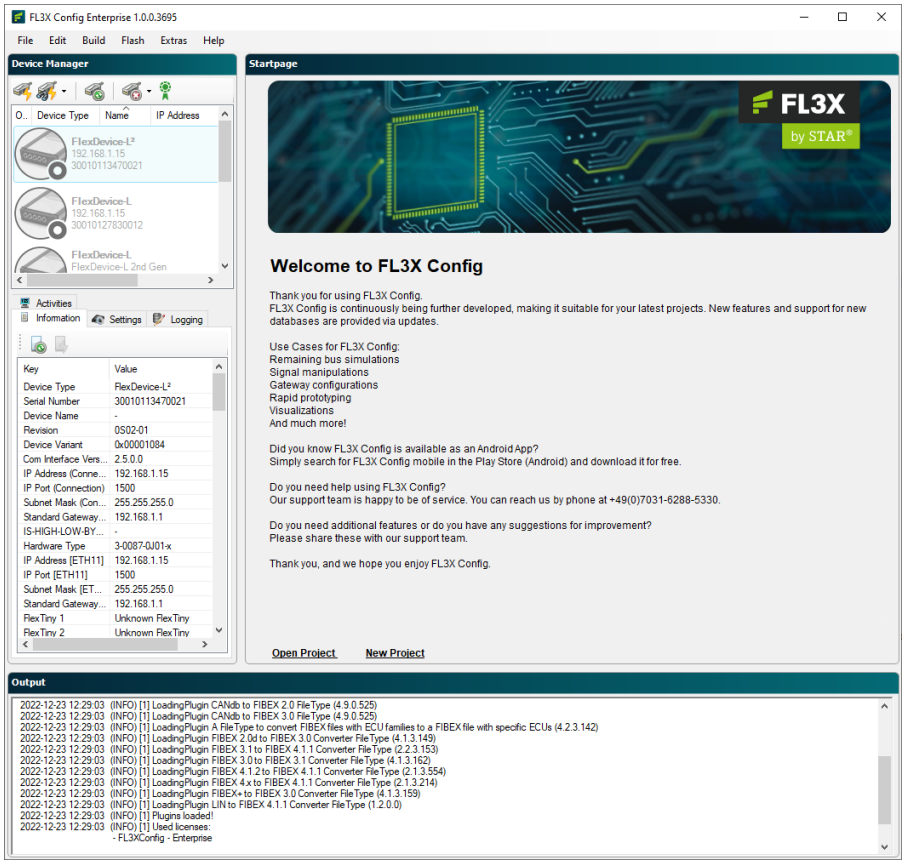
FL3X by STAR®

www.flex-product.com

FL3X Config User Manual



Contact Information

STAR ELECTRONICS GmbH & Co. KG
A Company of the STAR COOPERATION Group
Jahnstrasse 86
73037 Goeppingen
Phone: +49 (0)7031 6288-5656
Phone: +49 (0)7031 6288-5330 (Support)

Sales: sales-ee@star-cooperation.com
Support: support-ee@star-cooperation.com
www.flex-product.com

Company Data

STAR ELECTRONICS GmbH & Co. KG, registered offices: Goeppingen, register court Ulm, HRA 721096
Partner liable to unlimited extent: STAR ELECTRONICS Verwaltungs-GmbH, registered offices: Goeppingen, register court Ulm, HRB 722565
Represented by the executive board: Jörg Gerstlauer, Henning Lange

Copyright Notice

© 2026 STAR ELECTRONICS GmbH & Co. KG. All Rights Reserved.

No part of this document may be reproduced in any form (photocopy, microfilm or another procedure) without prior written consent from *STAR ELECTRONICS GmbH & Co. KG.*

Trademarks

Any trademarks used in this document are the property of their respective owners.

Disclaimer

The information contained in this document does not affect or change General Terms and Conditions of *STAR ELECTRONICS GmbH & Co. KG*. *STAR ELECTRONICS GmbH & Co. KG* does not guarantee the completeness and accuracy of the content of this document and assumes no responsibility for any errors which may appear in this document or due to this document. The content of this document or the associated products are subject to change without notice at any time.

It is currently impossible to develop software that is bug-free in all applications. Therefore, the product is only allowed to be used in the sense of the product use case described herein.

STAR ELECTRONICS GmbH & Co. KG makes no warranty express or implied, as to this document or the information content, materials or products for any particular purpose, nor does *STAR ELECTRONICS GmbH & Co. KG* assume any liability arising out of the application or use of this product, and disclaims all liabilities, including without limitation resulting damages, as permissible by applicable law.

All operating parameters which are provided in this document can vary in different applications or over time. The herein described product, may solely be used as described in chapter 1.2 “Intended use”.

Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written consent of *STAR ELECTRONICS GmbH & Co. KG*.

STAR ELECTRONICS GmbH & Co. KG may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly stated in a written license agreement from *STAR ELECTRONICS GmbH & Co. KG* the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Any semiconductor devices have an inherent chance of failure. You must protect against injury, damage or loss from such failures by incorporating safety design measures into your facility and equipment such as redundancy, fire protection, and prevention of over-current levels and other abnormal operating conditions.
The safety and handling instructions in this document must be followed strictly.

Revision History

Document number: 3-01140A01-D011

Version	Date	Description
D1V0-F	2023-01-20	Release for FL3X Config 1.0
D1V1-F	2023-06-27	Release for FL3X Config 1.1
D1V2-F	2024-03-08	Release for FL3X Config 1.2
D1V3-F	2024-08-23	Release for FL3X Config 1.3
D1V4-F	2024-11-15	Release for FL3X Config 1.4
D1V5-F	2026-03-25	Release for FL3X Config 1.5

Related Hardware / Software Versions

Product	Reference No.	Version	Remarks
PC-HW-Interface	3-9999-0C01	3.9	Interface to Communicate with the *STAR ELECTRONICS* Hardware
FL3X Device-S	3-00860A01-30		Old product name: FlexDevice-S
FL3X Device-L	3-00870A02-01	Rev 2	Old product name: FlexDevice-L
FL3X Device-L	3-00870A03-01	Rev 3	Old product name: FlexDevice-L
FL3X Device-L²	3-00870S02-01	Rev 2	Old product name: FlexDevice-L²
FL3X Device-L²	3-00870S03-01	Rev 3	Old product name: FlexDevice-L²
FL3X Device-PXIe	3-00940A01-50	Rev 1	Old product name: FlexCard PXIe3
FL3X Device-PCIe	3-00950A01-20	Rev 1	Old product name: FlexCard PCIe3
FL3X System-M	3-01050A01		
FL3X System-XS	3-01040A01		

Product	Reference No.	Version	Remarks
FL3X System-XS²	3-01040C01		

Additional Software and Documentation

Additional software components, firmware versions, and documentation packages that are not explicitly listed in this chapter can be obtained from the official FL3X product website:

www.flex-product.com

Contents

1 General 9
1.1 Intended User Group 9
1.2 Intended Use 9
1.3 Meaning of Text Styles 11
1.4 Used Pictographs 11
1.5 Safety and Handling Instructions 11
2 Product Description 13
2.1 FL3X Config 13
2.2 FL3X Config Control 15
2.3 FL3X Config Gateway 16
2.4 FL3X Config Analyzer (Windows) 17
2.5 FL3X Config Tester 17
2.6 FL3X Config API 17
2.7 Supported File Formats 18
2.8 Supported Target Hardware 19
2.8.1 FL3X Device-S 19
2.8.2 FL3X Device-L 20
2.8.3 FL3X Device-L² 21
2.8.4 FL3X Device-PXIe 23
2.8.5 FL3X Device-PCIe 23
2.8.6 FL3X System-M 24
2.8.7 FL3X System-XS 27
2.8.8 FL3X System-XS² 27
2.9 Preconditions and Limitations 27
3 Licensing 31
3.1 Software Licenses (FL3X Config) 31
3.1.1 Differences between the FL3X Config Versions 31
3.1.2 View Available Licenses 32
3.1.3 License Allocation 33
3.2 Runtime Licenses (FL3X Device, FL3X System, Virtual FL3X Device) 33
3.2.1 View Available Runtime Licenses 35
4 Software Installation and Update 36
4.1 Preconditions 36
4.1.1 Minimum System Requirements 36
4.1.2 Recommended System Requirements 36
4.1.3 .NET 8.0 36
4.1.4 CodeMeter Runtime 36
4.2 Installation 37
4.3 Command-Line Options (CLI) 39
4.4 Installing Multiple Versions 40
4.5 Start FL3X Config 40
4.5.1 Software License Expire Dialog 41
4.6 Update FL3X Config 42
4.6.1 Automatic Updates 42
4.7 Uninstall FL3X Config 42
4.8 Update FL3X Config Licenses 42
4.8.1 License Update via CodeMeter License Central WebDepot 42
5 FL3X Config 44
5.1 The Main Window 44

5.2 Device Manager 45
5.2.1 Information Tab 48
5.2.2 Settings Tab 51
5.2.3 Activities Tab 53
5.2.4 Logging Tab 54
5.2.5 Flashing 54
5.2.6 Hardware Licenses 56
5.2.7 Firmware Updater 58
5.3 Output 60
5.4 The Menu Bar 60
5.4.1 File Menu 60
5.4.2 Edit Menu 63
5.4.3 Build Menu 65
5.4.4 Flash Menu 66
5.4.5 Extras Menu 67
5.4.6 Help Menu 67
5.5 Database Manager 68
5.5.1 Database Picker 69
5.6 Device Configuration 70
5.6.1 Selecting the Database of a Bus 72
5.6.2 Base Settings 73
5.6.3 RBS Settings 83
5.6.4 E2E-Protection 96
5.6.5 Standard Signal Handling 102
5.6.6 Offset Settings (FlexRay) 106
5.6.7 Transmission Settings 110
5.7 User Defines 122
5.8 The User Function Editor 124
5.8.1 Functions 125
5.8.2 Source Code Editor 145
5.8.3 Signals and Variables 150
5.8.4 Validation Output 181
5.8.5 Merge User Functions 182
5.8.6 RBS API 185
5.8.7 Differences in Saving User Functions 185
5.8.8 Supported Data Types 185
5.9 Default Values 186
5.9.1 The Concept of Default Values 186
5.9.2 The Default Values Editor 186
5.10 Secure Onboard Communication 191
5.11 Time Synchronization 193
5.11.1 Common Configuration 195
5.11.2 CAN Configuration 195
5.11.3 Ethernet Configuration 197
5.11.4 FlexRay Configuration 199
5.12 Custom E2E-Profiles 201
5.12.1 Templates for Custom E2E-Profiles 202
5.12.2 Configure a Signal Based Custom E2E-Profile 202
5.12.3 Configure a Stack Based Custom E2E-Profile 207
5.12.4 Parameter Description 209
5.13 RBS Based Replay 212
5.14 Database Update 214
5.14.1 Database Update and E2E Profiles / OEM dependent Standard Signal Handlings) 215

5.14.2 Database Update to Change the Used Database 215
5.14.3 Database Update to Reconvert a Database 217
5.15 Change Device Type 219
5.16 Create Partner Project 221
5.17 Support for Visual Studio Code 223
5.17.1 Install the FL3X Config Integration Extension 223
5.17.2 Open a FL3X Config Project 224
5.17.3 Using Code Snippet Suggestions 224
5.17.4 Building the Project inside Visual Studio Code 226
5.18 Handling for ECU Families 226
5.18.1 ECU Family Example 227
5.19 Handling for PDU Update Bits 228
5.19.1 Handling for PDU Update Bits – Case RBS 228
5.19.2 Handling for PDU Update Bits – Case Gateway 228
5.19.3 Handling for Signal/SignalGroup Update Bits – Case Gateway 228
5.20 Handling for PDU-Container 228
5.21 Automotive Ethernet Sleep/Wake-up Management (TC10) 232
5.21.1 Automatic Transmission of Wake-up Request 232
5.21.2 Manual Transmission of Wake-Up Requests 232
5.21.3 Reception of Wake-Up Requests 233
5.22 Settings 235
5.22.1 Application Settings 236
5.22.2 Project Settings 242
5.22.3 Templates for New Projects 244
5.22.4 Target OS 246
5.23 Multi OEM Support 248
5.24 OEM Specific Content 249

6 FL3X Config Gateway 250

6.1 Gateway 250
6.1.1 General 250
6.1.2 Routings 253
6.1.3 Filter 257
6.1.4 Display Items 259
6.1.5 Scaling Signals 262
6.1.6 Auto Routing 265
6.1.7 Split a Routing 267
6.1.8 Delete Routings 268
6.1.9 Export Items 270
6.1.10 Print 271
6.1.11 Import and Export of Routings 272
6.1.12 Limits of a Gateway 273
6.2 Transparent Gateway 275
6.2.1 Transparent Gateway in the Main Window 276
6.2.2 Edit Connectors of a Transparent Gateway 276
6.2.3 Remove a Transparent Gateway Configuration 277
6.2.4 Limits of a Transparent Gateway 277
6.2.5 Create a Transparent Gateway 277
6.2.6 FlexRay Synchronization used in a Transparent Gateway 280
6.3 Differences between RBS and Gateway 281

7 FL3X Config Control 282

7.1 Manipulations and Sequences 282
7.1.1 Manipulations 283

7.1.2 Limits for Signal Values 287
7.1.3 Available Elements for Manipulation 288
7.1.4 Manipulation Methods 289
7.1.5 RAW Value Editor 298
7.1.6 Duration and Delay 298
7.1.7 Trigger 299
7.1.8 Linked Trigger Conditions 302
7.1.9 Trigger Modes 302
7.1.10 Manipulation Sequences 303
7.1.11 Hotkey 306
7.2 Difference between Signal Manipulations within an RBS and a (Transparent) Gateway 306
7.3 Special Case: Using the Same PDU Instance in Multiple Frame Triggerings 306
7.4 Manipulation of Global Variables via FL3X Diagnostics Website 307

8 FL3X Diagnostics Website 309

8.1 Analyzing 309
8.1.1 Configuration for Analyzing as ASAM CMP Probe 309
8.2 Time Synchronization 312
8.3 Trace 312

9 FL3X Config Project Format 314

9.1 Files and folders of a FL3X Config Project 314
9.2 Ignored Files and Folders 315

10 FL3X Config Command Line Application 317

11 Device LEDs, Update Device License 318

11.1 FL3X Device-S, FL3X Device-L & FL3X Device-L² LEDs 318
11.2 FL3X Device-PXIe & FL3X Device-PCIe LEDs 319
11.3 Update Device License 321
11.3.1 Update Device License via FL3X Diagnostics Website 321
11.3.2 Update Devices via FL3X Update 322
11.3.3 Write a License File to the Device 323

12 Troubleshooting 324

12.1 FL3X Config – General 324
12.2 Building and Flashing Issues 329
12.3 FL3X Config Control 332
12.4 User Functions 333
12.5 FL3X Device – General 334
12.6 FL3X Device – Runtime Errors 335
12.7 FL3X Tiny/FPGA Issues 336

13 Ordering Information 338

13.1 FL3X Config 338
13.2 Accessory Parts 338
13.3 Related Documents 338

14 Appendix 340

14.1 Acronyms and Abbreviations 340
14.2 List of Tables 340
14.3 List of Figures 343

1 General

1.1 Intended User Group

This product may only be used by expert technicians and/or engineers who are qualified and familiar with electronic components and systems.

Each person involved with setup or operation of the product must

- be a qualified technician or engineer
- strictly adhere to this manual
- receive a briefing by an authorized person



NOTICE

If you are unsure of how to use the product as intended or have any questions about the use of the product, please discontinue use of the product immediately and contact the STAR ELECTRONICS GmbH & Co. KG Support.



⚠ WARNING

The product may only be used by expert technicians and/or engineers who are qualified and familiar with electronic components and systems!

The use of the product by non-professionals is not permitted and strictly forbidden!

1.2 Intended Use

FL3X Config is a software specially designed for testing purposes in connection with STAR ELECTRONICS GmbH & Co. KG's hardware devices. It was developed to configure and control STAR ELECTRONICS GmbH & Co. KG hardware devices (e.g., the FL3X Device product family) exclusively. It is **NOT** permitted to use the software with or in connection with **any other hardware devices**.
For this intended use, FL3X Config offers the following options:

- Configuration of STAR ELECTRONICS GmbH & Co. KG hardware
- Control the runtime behavior of STAR ELECTRONICS GmbH & Co. KG hardware
- Visualization of data traffic (e.g. Use Case "Analyzing")
- Recording of data traffic (e.g. Use Case "Logging")



⚠ WARNING

Any use of the software on or in connection with a STAR ELECTRONICS GmbH & Co. KG hardware device product outside a fully controlled testing and/or laboratory environment may result in death or serious injury due to unpredictable behavior of a vehicle and/or potentially missing, deactivated or malfunctioning safety devices on a vehicle!



WARNING

The FL3X Config may be used to communicate with networked electronic systems. E.g. FlexRay, CAN or Ethernet.

Any use of the product outside a fully controlled testing and/or laboratory environment may result in death or serious injury due to unpredictable behavior of a vehicle and/or potentially missing, deactivated, or malfunctioning safety devices on a vehicle!

The user is responsible to ensure the safety of the entire system. This includes amongst other things a safety shutdown.



WARNING

The use of FL3X Config in test vehicles requires special care in connection with the verification of effects during test drives, as there may be no experience with the effects of such tests in this context.

Therefore, increased and consistent attention must be paid to the use of FL3X Config in test vehicles to ensure that no damage occurs.

We do not accept any liability in connection with any damage that could occur when using FL3X Config if the appropriate careful monitoring measures are not implemented or observed.



WARNING

FL3X Config may be used to control and/or influence the behaviour of control systems and electronic control units.

Interventions in safety relevant systems (e.g. manipulating the engine management, steering, airbag and/or braking system) or operating the device in public areas (e.g. within a test vehicle in public areas) may result in death or serious injury due to unpredictable behavior of a vehicle and/or potentially missing, deactivated, or malfunctioning safety devices on a vehicle!

Therefore, FL3X Config must be used in a safe manner (including a safety shutdown). Comply with all safety standards and public regulations which are relevant for the operation of the system.
Prior to operating the system in public areas, it should first be tested in a location that is not accessible to the public and has been specifically prepared for test drives, in order to minimize potential risks.



NOTICE

The device is not a calibrated measurement device. STAR ELECTRONICS GmbH & Co. KG accepts no liability whatsoever for the correctness of any measurement results.

	**⚠ WARNING** FL3X Config is **NOT** designed, intended, or authorized and may **NOT** be used for or in connection with the following purposes and/or devices: - use as part of medical systems - life support applications - aviation, space, nuclear, or military applications - any other purposes / devices deviating from the intended use of the product specified by STAR ELECTRONICS GmbH & Co. KG.

	**⚠ WARNING** The product may only be used by expert technicians and/or engineers who are qualified and familiar with electronic components and systems! The use of the product by non-professionals is not permitted and strictly forbidden!

1.3 Meaning of Text Styles

In this document *filenames*, `source code`, specified c-code by FL3X Config, **`functions`**, **`structs`**, "*buttons*", "*controls*" and '*menu entries*' are marked with a different text format.

Any placeholder or variable is maked with '%...%'. E.g. %version%.

1.4 Used Pictographs

The meaning of the pictographs used, is shortly described below.

Follow the specific instructions in the document, where these pictographs are placed.

	**⚠ WARNING** Used to indicate a potentially hazardous situation which, if not avoided, could result in death or serious injury.

	**NOTICE** Used to indicate a situation which may result in an operating failure. Damage of the product may occur, but there is no hazard of injury if not avoided.

	**Information** Used to indicate information provided only for purposes of clarification, illustration, and general information.

	**Reference** References to other documents.

1.5 Safety and Handling Instructions

Please read the instructions for use carefully. To protect the device or the application against damage, or to avoid personal injury the FL3X Config must be handled as described herein.

Changes or modifications of the FL3X Config are not allowed for safety and warranty reasons!

STAR ELECTRONICS GmbH & Co. KG is not liable for any damages arising from non-observance of the product information.

Follow the
a) specific safety and handling instructions placed at dedicated document positions
b) general safety and handling instructions below:

2 Product Description

2.1 FL3X Config

FL3X Config at a Glance

FL3X Config is a program to generate hardware-assisted Remaining Bus Simulations. Numerous bus systems and network description formats are supported. The generated Remaining Bus Simulations run independently on separate hardware.

Main Features

- Support for devices
  - FL3X Device-S
  - FL3X Device-L
  - FL3X Device-L²
  - FL3X Device-PXIe
  - FL3X Device-PCIe
  - FL3X System-M
  - FL3X System-XS
  - FL3X System-XS²
- Virtual FL3X Devices (Requires Runtime license)
  - Virtual FL3X Device for Windows
  - Virtual FL3X Device for ADTF (SiL)
- Supported bus systems: FlexRay, CAN high-speed, CAN low-speed, CAN-FD, LIN, Ethernet (100BASE-TX/1000BASE-T), Automotive Ethernet (10BASE-T1S, 100/1000BASE-T1)
- Embedded OS variants (singlecore or multicore) with FPU support
- Support for DoIP and ISO-TP (on request)
- Creation, design, and administration of several remaining bus simulations
- Support of the network description formats FIBEX, AUTOSAR, CANdb, LDF
- Updating existing projects with new network description files (Database Updater) (Requires FL3X Config Professional License)
- Backward compatibility guaranteed down to FlexConfig RBS 4.3
- Parallel installation of different FL3X Config versions (starting with FlexConfig RBS 4.6)
- Creation and configuration of a CAN RBS is possible without a network description file
- Configurable timing and data behavior for all transmitted elements (including SOME/IP Service Discovery)
- Selection of the ECUs which should be simulated, or a remaining bus simulation should be created for
- Handling of ECU families per selecting one ECU which should be simulated or generating dummy ECUs for the required ECU family
- Self-synchronization of ECUs/clusters which have no cold start ability
- Support of user defined sync/startup frames
- Configurable transmission of Wakeup pattern on simulated FlexRay buses
- Selection of frames which should be simulated
- Supported byte-orders: Big Endian, Little Endian, Mixed Endianess (within a PDU)
- Supports multiplexed PDUs
- Supports Container-PDUs
- Generation and compilation of the remaining bus simulation
- Download directly without any external tool
- Hard real-time support for E2E protection and network management
- Interface for user defined algorithms and network management (ANSI C code)
- Support for configurable OEM specific project templates with presets for
  - standard signal handling
  - target IP address
  - Frame and PDU init values per bus
- Support for multi-OEM projects
- Access RBS run-time parameters via Windows API over Ethernet:
  - Global Parameters (read/write)
  - Signals (read/write)
  - PDUs (read/write)

  - Bus Controller (switch on/off / read state)
  - Simulated ECUs (switch on/off / read state)
- Command line tool to build and download projects
- RBS based replay
  - Support for endless loops
- Visual Studio Code (Requires FL3X Config Enterprise License)
  - Edit User Functions via Visual Studio Code
  - Edit User Files via Visual Studio Code
  - Insert code snippets vie Visual Studio Code
  - Build project via Visual Studio Code
- User Function editor
  - Implementation of frame-based and PDU-based TX and RX functions per drag and drop and user-defined algorithms
  - Implementation of interrupt-based CycleStartInterrupt functions per drag and drop and user-defined algorithms for simulated FlexRay ECUs/clusters
  - Implementation of initialization function and cyclic user function
  - API to change channel behavior at runtime
  - API for data logging
  - User defined .c/.h/.cpp files and folders can be added/included
  - User Libs (*.ulib) can be included
  - Access during runtime to TX or RX main functions via Windows API and Ethernet
  - Integration of I/Os per drag and drop
  - Access to User LEDs
  - Signal and frame-based data handling
  - Rx User Functions: Payload access to last received frame (e.g. for CRC check)
  - Generation of additional Tx events for the simulated Tx frames (e.g. for transmitting CAN frames which don't have cyclic triggered timings)
  - Read/write access to ethernet header data
  - Read/write access to PDU Update Bits
  - Read/Write RAW data on Ethernet and CAN channels
  - Enable/Disable simulated ECUs
  - Enable/Disable bus communication
  - Access to user defines
  - User functions are saved as *.c files (Requires FL3X Config Enterprise License)
  - Supports export of user functions (as *.c/*.h files) (Requires FL3X Config Enterprise License)
  - Supports search for used elements (such as Bus, ECU, Signal, …)
  - Supports import of user functions from other projects (Requires FL3X Config Enterprise License)
- Export FPGA-Images as file for all supported devices
- Device Manager (multi device support)
  - Manage multiple devices
  - Change device name
  - Change IP-settings
  - Change termination
  - Change wake-up/sleep
  - Flash FPGA images to device
- FL3X Diagnostics Website
  - Supports FL3X Diagnostics Website via web browser
  - Supports firmware update via web browser
  - Supports to trace data at runtime via web browser
  - Supports to save the trace data in ASC format
  - Support to configure logging (BETA)
  - Supports to update the runtime licenses
  - Supports to switch between application images
  - Supports password protection for certain pages and functionalities of the FL3X Diagnostics Website
  - Supports ASAM CMP
  - Supports locking the FL3X Config Diagnostics Website
  - Supports to protect the FL3X Config Diagnostics Website via password

- E2E-Profile Manager
  - Support for Custom E2E-Profiles
  - Supports multiple E2E-Profiles per bus
  - Supports individual configuration for each protected element
- Support of automatic application update via internet
- Support of automatic license update via internet
- Time Synchronization according to AUTOSAR 4.3 and PTPv2 (based on IEEE1588 – the standard IEEE1588 is **NOT** fully supported)
- Support for AUTOSAR Secure Onboard Communication (SecOC)
- Supports SOME/IP
  - Service Discovery
  - SOME/IP Events, Fields and Methods
  - SOME/IP TP
  - AUTOSAR PDUs
- Supports FL3X Config mobile (Android App)

Required Components

- FL3X Config software with one of the following versions:
  - FL3X Config Basic: This version needs not software license
  - FL3X Config Professional: This version needs the 'Professional' license
  - FL3X Config Enterprise: This version needs the 'Enterprise' license
- Hardware (at least one of the listed)
  - FL3X Device-L with corresponding FL3X Tinys
  - FL3X Device-L² with corresponding FL3X Tinys
  - FL3X Device-S with corresponding FL3X Tinys
  - FL3X Device-PXIe with correspondig FL3X Tinys
  - FL3X Device-PCIe with correspondig FL3X Tinys
  - FL3X System-M with supported modules
  - FL3X System-XS
  - FL3X System-XS²
- Hardware (Runtime-License)
  - A Runtime license is only required for bus configurations with CAN-FD, FlexRay or Ethernet. For every other supported bus system, no Runtime license is required. E.g., LIN, SENT, CAN-HS do not need any Runtime license
  - FL3X Config Standard Runtime license is required, if a RBS is configured for a CAN-FD, FlexRay or an Ethernet bus.
    FL3X Config Extended Runtime license is required, if a Gateway is configured for a CAN-FD, FlexRay or an Ethernet bus.

2.2 FL3X Config Control

FL3X Config Control at a Glance

FL3X Config Control is used to manipulate a generated RBS or Gateway. It is part of FL3X Config.

Main Features

- Fast and efficient signal manipulation with the provided GUI
- Configuration via drag & drop mechanism
- Various methods to manipulate data
  - Loss
  - Factor
  - Freeze
  - Fix Value
  - Offset
  - Drift
  - Ramp
- The duration and the delay of a manipulation can be adjusted
  - Events
  - Milliseconds
  - Cycles (FlexRay only)
- Various trigger options
  - Trigger operators are '==', '!=', '>', '<', '>=', '<=' and 'Edge'

    - Link up to four triggers
- Enable manipulations via Hotkey
- With the help of a sequence editor various manipulation groups can be stored
- Manipulations can be controlled via a Windows API and Ethernet
- Manipulable elements are:
    - ECU
    - Frames
    - PDUs
    - Switches (of multiplexed PDUs)
    - Signals
    - Signal Update Bits
    - Signal Group Update Bits
    - Global variables
    - SOME/IP Events
    - SOME/IP Fields
    - SOME/IP Methods
- Save manipulations at the target platform
    - Saved manipulations run automatically at power on

2.3 FL3X Config Gateway

FL3X Config Gateway at a Glance

FL3X Config Gateway is used to generate hardware-assisted gateways. It is part of FL3X Config.

Main Features

- Creation, design and administration of several gateways between
    - FlexRay
    - CAN
    - LIN
    - Ethernet (PDU-based)
    - SOME/IP
- Automated generation of transparent gateways
    - FlexRay <-> FlexRay
    - CAN <-> CAN
    - Ethernet (PDU-based) <-> Ethernet (PDU-based)
    - SOME/IP <-> SOME/IP
- Cluster synchronization for FlexRay <-> FlexRay gateways
    - Cycle counter synchronization
    - In-cycle offset
- Gateway routings can be created for Frames, PDUs, signals and SOME/IP elements
- Signal routings from shorter to longer signal size
- Support for:
    - Searching and filtering in source, target and routing lists
    - Highlight matching for source and target elements during routing (Requires FL3X Professional License)
    - Mark/highlight elements and element groups in different colors (Requires FL3X Professional License)
    - Automatic routing of complete busses (Requires FL3X Professional License)
    - Link the SOME/IP SD behavior of routed services
    - Automatic and manual scaling of signals and SOME/IP datatypes (Automatic scaling requires FL3X Professional License)
- Export the configured routings and of the available source and target elements as CSV file or Excel file (Requires FL3X Config Enterprise License)
- Print the configured routings and of the available source and target elements (Requires FL3X Professional License)
- Supports export and import of routings in predefined format (Requires FL3X Config Enterprise License)

2.4 FL3X Config Analyzer (Windows)

	Reference
	See FL3X Config Analyzer (Windows) – User Manual [9].

FL3X Config Analyzer (Windows) at a Glance

FL3X Config Analyzer (Windows) is a tool to visualize bus data. It is part of FL3X Config.

Main Features

- Visualize bus data (only signal values)
- Visualize global variables
- Support for FlexRay and CAN, Ethernet, LIN
- Support Trace view (BETA)
  - Only for busses configured via FL3X Config
  - Support for different filters
  - Support for different view modes (continuous, stacked)
  - Support for RAW and Physical values
- Graph to display signal values
- Support different widgets for visualization
- Label
  - Signal value as text
  - Boolean indicator
  - Bar chart
  - Line chart/graph
  - Gauge
  - Manipulation
- Supports sharing projects with FL3X Config mobile 1.3
- Supports loading projects based on FL3X Config Analyzer (Android) 4.0 or later
- Support loading and analyzing MDF files

2.5 FL3X Config Tester

	Reference
	See FL3X Config Tester User Manual [17].

FL3X Config Tester at a Glance

FL3X Config Tester (*FL3XConfigTesterConsole.exe*) is a Windows command-line application used for testing gateway configurations.

It requires an FL3X Device and a test configuration to send and receive routing packets and test the gateway functionality.

Main Features

- Send different payloads
- Detect missing routings
- Identify routing time violations
- Generate detailed test result reports (*.html*/*.txt*)
- Export test communication as *.asc* trace for easy analysis in FL3X Config Analyzer

2.6 FL3X Config API

	Reference
	See RBS API Manual [11].

FL3X Config API at a Glance

FL3X Config API provides access to additional features via the User Function Editor.

- Support for:
  - Get/set the time per clock group (RTC, PTP, ...)
  - Support for up to 10 stopwatches (start, pause, reset, read)
  - Get the latest bus timestamp (per frame)
  - Get the current power supply voltage
  - Configure, get and set digital and analog inputs/outputs
  - Perform restart of device
- Change channel behavior at runtime
  - Support for FlexRay and CAN
  - Change terminating resistor configuration
  - Change channel configurations at runtime
  - Change sleep/wakeup options
- Data logging
  - Support for FlexRay and CAN
  - Support for ASAM MDF 4.1 file format
  - Support for different time bases (RTC, bus time, internal)
  - Support for pre-logging
- Runtime Log
  - Support to write log messages to the runtime log (visualized on the FL3X Diagnostics Website)
- Replay
  - Support to get and set the replay sequence

Required Components

FL3X Config API is supported by the following devices:

- FL3X Device-S
- FL3X Device-L
- FL3X Device-L²
- FL3X Device-PXIe
- FL3X Device-PCIe
- FL3X System-M
- FL3X System-XS
- FL3X System-XS²

2.7 Supported File Formats

Native File Formats

- FIBEX 4.1.1

Converted File Formats

- FIBEX
  - 2.0.0d
  - 2.0.1
  - 3.0.0
  - 3.1.0
  - 4.0.0
  - 4.1.0
  - 4.1.2
  - FIBEX + 1.4
  - ECU Family Handling
- AUTOSAR
  - 3.0.2
  - 3.1.0
  - 3.1.4.DAI.2
  - 3.1.4.DAI.4
  - 3.2.2
  - 4.0.3
  - 4.1.1
  - 4.2.1
  - 4.2.2

- 4.3.0
- 4.3.1
- 4.4.0
- 4.6.0
- 4.7.0

- CANdbc
- LDF

2.8 Supported Target Hardware

Information
If you cannot find any required FPGA image for your device configuration, please check the chapter Troubleshooting -> “Required device configuration is not available for new project/change device”.

2.8.1 FL3X Device-S

Reference
See FL3X Device-S Instructions for Use [3].

FL3X Device-S										
FPGA Image	**CAN**	**FlexRay**	**ETH**	**LIN**	**SENT**	**SPI**	**UART**	**K-Line**	**TS**	**AES/ SecOC**
0x000000B0	4 (2xHS 2xHS/FD)	2 (1xSelfSync)	1	1	-	-	-	-	X	-
0x000000B1	4 (2xHS 2xHS/FD)	1 (SelfSync)	1	5	2	-	-	-	X	-
0x000000B2	4 (2xHS 2xHS/FD)	1 (SelfSync)	1	1	-	1	1	1	X	-
0x000000B4	8 (2xHS 6xHS/FD)	1 (SelfSync)	1	1	-	-	-	-	X	-

Table 1: Supported FPGA images – FL3X Device-S

TS: Support for hardware-based timestamps

AES/SecOC: Support for hardware based Secure Onboard Communication

2.8.1.1 Supported FL3X Tinys

FL3X Tiny III	Supported
FlexRay	X
CAN-FD	X
CAN-FD (4 transceivers)	X
CAN-FD-SIC (4 transceivers)	X
100BASE-TX Switch	X
100BASE-T1	X
100BASE-T1 Switch	X
LIN/SENT	X
K-Line/UART/SPI	X
1000BASE-T1 Switch MACsec	X
10BASE-T1S TC10	X

Table 2: Supported FL3X Tinys – FL3X Device-S

2.8.2 FL3X Device-L

	Reference
	See FL3X Device-L Instruction for use [1].

2.8.2.1 Supported FPGAs

Rev 2 (3-0087-0A02)	Rev 3 (3-0087-0A03)										
FPGA Image	**FPGA Image**	**CAN**	**FlexRay**	**ETH**	**LIN**	**SENT**	**SPI**	**UART**	**K-Line**	**TS**	**AES/ SecOC**
0x00001060	0x00002060	4 (4x HS)	2 (2xSelfSync)	1	-	-	-	-	-	-	-
0x00001061	0x00002061	4 (2x HS, 2x HS/FD)	2 (1xSelfSync)	1	-	-	-	-	-	X	-
0x00001062	0x00002062	8 (8x HS/FD)	-	1	-	-	-	-	-	X	X
0x00001064	0x00002064	10 (10x HS/FD)	-	-	-	-	-	-	-	X	-
0x00001065	0x00002065	4 (4x HS)	3 (1xSelfSync)	-	-	-	-	-	-	-	-
0x00001068	0x00002068	6 (1x HS, 5x HS/FD)	1 (1xSelfSync)	1	-	-	-	-	-	-	-
0x0000106A	0x0000206A	2 (2x HS/FD)	1 (1xSelfSync)	5	-	-	-	-	-	X	-
0x0000106B	0x0000206B	3 (2x HS, 1x HS/FD)	2 (1xSelfSync)	2	-	-	-	-	-	X	-
0x0000106C	0x0000206C	6 (2x HS, 4x HS/FD)	1 (1xSelfSync)	-	4	-	-	-	-	-	-
0x0000106D	0x0000206D	4 (2xHS, 2xHS/FD)	1 (1x SelfSync)	-	4	2	1	1	1	-	-
0x0000106F	0x0000206F	6 (HS/FD)	-	4 (2x GBit)	-	-	-	-	-	X	-
0x00001161	0x00002161	4 (HS/FD)	1 (1x SelfSync)	5 (4x GBit)	-	-	-	-	-	X	-
0x00001162	0x00002162	4 (HS/FD)	-	5 (3x Gbit)	-	-	1	1	1	X	-
0x00001163	0x00002163	4 (HS/FD)	-	6 (5x Gbit)	-	-	-	-	-	X	-

Table 3: Supported FPGA images – FL3X Device-L

TS: Support for hardware-based timestamps

AES/SecOC: Support for hardware based Secure Onboard Communication

2.8.2.2 Supported FL3X Tinys

FL3X Tiny III	Rev 2 (3-0087-0A02)	Rev 3 (3-0087-0A03)
FlexRay	X	X
CAN-FD	X	X
CAN-FD (4 transceivers)	-	X
CAN-FD-SIC (4 transceivers)	X	X
100BASE-TX Switch	X	X
100BASE-T1	X	X
100BASE-T1 Switch	X	X
1000BASE-T1	X	X

FL3X Tiny III	Rev 2 (3-0087-0A02)	Rev 3 (3-0087-0A03)
LIN/SENT	X	X
KLine/UART/SPI	X	X
1000BASE-T1 Switch MACsec	X	X
10BASE-T1S TC10	X	X

Table 4: Supported FL3X Tinys – FL3X Device-L

2.8.3 FL3X Device-L²

	Reference
	See FL3X Device-L Instruction for use [1]

2.8.3.1 Supported FPGAs

Rev 2 (3-0087-0S02)	Rev3 (3-008-0S03)										
FPGA Image	FPGA-Image	CAN	FlexRay	ETH	LIN	SENT	SPI	UART	K-Line	TS	AES/ SecOC
0x10901080	0x20902080	4 (4x HS/FD)	3 (3xSelfSync: 2xA, 1xAB)	1	-	-	-	-	-	X	X (only Rev 2)
0x10911081	0x20912081	6 (6x HS/FD)	1 (1xSelfSync)	1	-	-	-	-	-	X	-
0x10921080	not available	8 (8x HS/FD)	2 (2xSelfSync, 2xA)	1	4	2	-	-	-	X	X
0x10931082	not available	4 (4x HS/FD)	3 (3x SelfSync)	-	-	-	-	-	-	X	-
0x10941083	not available	8 (8x HS/FD)	2 (1x SelfSync AB, 1xA)	3	-	-	-	-	-	X	-
0x1094118A	0x2094218A	8 (8xHS/FD)	-	2 (2x Gbit)	8	4	-	-	-	X	-
0x10951084	0x20952084	4 (4x HS/FD)	2 (2x SelfSync)	4	-	-	-	-	-	X	-
0x10961085	0x20962085	8 (8x HS/FD)	3 (1x SelfSync, 2xA)	1	-	-	-	-	-	X	-
0x10971086	0x20972086	8 (8x HS/FD)	3 (2x SelfSync)	-	-	-	-	-	-	X	-
0x10941089	0x20942089	8 (8xHS/FD)	-	4 (2x Gbit)	-	-	-	-	-	X	-
0x1094108A	0x2094208A	10 (10x HS/FD)	2 (1xSelfSync, 2xA)	1	-	-	-	-	-	X	-
0x1094108B	0x2094208B	8 (8x HS/FD)	2 (1xSelfSync, 2xA)	4	-	-	-	-	-	X	X
0x1092108D	0x2092208D	8 (8x HS/FD)	2 (1xSelfSync, 2xA)	2	4	2	-	-	-	X	-
0x1094108E	0x2094208E	8 (8x HS/FD)	1 (1x SelfSync)	4 (2x GBit)	-	-	-	-	-	X	-
0x109B1183	0x209B2183	4 (4x HS/FD)	3 (2x SelfSync)	2 (2x GBit)	-	-	-	-	-	X	-
0x109B118E	0x209B218E	4 (4x HS/FD)	2 (2x SelfSync)	4 (2x GBit)	-	-	-	-	-	X	-
0x109D1187	0x209D2187	6 (6x HS)	3 (3xSelfSync, 3xA)	-	4	-	-	-	-	X	-

Table 5: Supported FPGA images – FL3X Device-L²

TS: Support for hardware-based timestamps

AES/SecOC: Support for hardware based Secure Onboard Communication

	Information
	The FL3X Device-L² FPGA images • 0x10951084 • 0x20952084 are no longer supported since FlexConfig RBS 5.3. This is because these images contain issues • ETH on Con 4B is not functional! Projects using these images can no longer be created. Loading projects based on these images is possible. Changing the projects to the 100% compatible FPGA images • 0x109B118E • 0x209B218E is strongly recommended. To convert a project using one of these images, use 'Change Device Type' in FlexConfig RBS 5.3 or later (see chapter 5.14.3 "Change Device Type"). The last version that supports creating projects based on these images is FlexConfig RBS 5.2.2.0.

2.8.3.2 Supported FL3X Tinys

FL3X Tiny III	Rev 2 (3-0087-0S02)	Rev 3 (3-008-0S03)
FlexRay	X	X
CAN-FD	X	X
CAN-FD (4 transceivers)	X	X
CAN-FD-SIC (4 transceivers)	X	X
100BASE-TX Switch	X	X
100BASE-T1	X	X
100BASE-T1 Switch	X	X
1000BASE-T1	X	X
LIN/SENT	X	X
K-Line/UART/SPI	X	X
1000BASE-T1 Switch MACsec	X	X
10BASE-T1S TC10	X	X

Table 6: Supported FL3X Tinys – FL3X Device-L²

2.8.4 FL3X Device-PXIe

NOTICE

Installation of the FL3X Device-PXIe driver package is required.

See “FL3X Device-PXIe/PCIe Instructions for Use” [6].

2.8.4.1 Supported FPGAs

FL3X Device-PXIe										
FPGA Image	**CAN**	**FlexRay**	**ETH**	**LIN**	**SENT**	**SPI**	**UART**	**K-Line**	**TS**	**AES/ SecOC**
0x000000D0	2 (2xHS/FD)	1	2	-	-	-	-	-	X	-
0x000000D3	2 (2xHS/FD)	-	6	-	-	-	-	-	X	-

Table 7: Supported FPGA images – FL3X Device-PXIe

TS: Support for hardware-based timestamps

AES/SecOC: Support for hardware based Secure Onboard Communication

2.8.4.2 Supported FL3X Tinys

FL3X Tiny III	Supported
FlexRay	X
CAN-FD	X
CAN-FD (4 transceivers)	X
CAN-FD-SIC (4 transceivers)	X
100BASE-TX Switch	X
100BASE-T1	X
100BASE-T1 Switch	X
1000BASE-T1	X
LIN/SENT	X
K-Line/UART/SPI	X
1000BASE-T1 Switch MACsec	X
10BASE-T1S TC10	X

Table 8: Supported FL3X Tinys – FL3X Device-PXIe

2.8.5 FL3X Device-PCIe

NOTICE

Installation of the FL3X Device-PCIe driver package is required.

See “FL3X Device-PXIe/PCIe Instructions for Use” [6].

2.8.5.1 Supported FPGAs

FL3X Device-PCIe										
FPGA Image	**CAN**	**FlexRay**	**ETH**	**LIN**	**SENT**	**SPI**	**UART**	**K-Line**	**TS**	**AES/ SecOC**
0x000000E0	2 (2xHS/FD)	1	2	-	-	-	-	-	X	-
0x000000E4	-	-	7	-	-	-	-	-	-	-

Table 9: Supported FPGA images – FL3X Device-PCIe

TS: Support for hardware-based timestamps

AES/SecOC: Support for hardware based Secure Onboard Communication

2.8.5.2 Supported FL3X Tinys

FL3X Tiny III	Supported
FlexRay	X
CAN-FD	X
CAN-FD (4 transceivers)	X
CAN-FD-SIC (4 transceivers)	X
100BASE-TX Switch	X
100BASE-T1	X
100BASE-T1 Switch	X
1000BASE-T1	X
LIN/SENT	X
K-Line/UART/SPI	X
1000BASE-T1 Switch MACsec	X
10BASE-T1S TC10	X

Table 10: Supported FL3X Tinys – FL3X Device-PCIe

2.8.6 FL3X System-M

The FL3X System-M can be configured with 6 Slots (2x Slots for PCIe,2x Slots for Mini-PCIe, 1x CMI/CMI-10G and 1x CMI/Mini-PCIe).

2.8.6.1 Slot for PCIe

For this slot, following devices are supported.

- FL3X Interface-PMC
- FL3X Interface-PCIe
- PCIe SFP Quad (LREC9714HF-4SFP)

2.8.6.1.1 FL3X Interface-PMC

Supported FPGAs for FL3X Interface-PMC:

FL3X Interface-PMC										
FPGA Image	**CAN**	**FlexRay**	**ETH**	**LIN**	**SENT**	**SPI**	**UART**	**K-Line**	**TS**	**AES/ SecOC**
0x00001020	4 (4xHS/FD)	1 (1xSelfSync)	-	-	-	-	-	-	-	-

Table 11: Supported FPGA images – FL3X Interface-PMC (FL3X System-M)

TS: Support for hardware-based timestamps

AES/SecOC: Support for hardware based Secure Onboard Communication

Supported FL3X Tinys for FL3X Interface-PMC:

FL3X Tiny2	Supported
FL3X Tiny2 2*FlexRay	X
FL3X Tiny2 2*CAN-FD	X

Table 12: Supported FL3X Tinys – FL3X Interface-PMC (FL3X System-M)

2.8.6.1.2 FL3X Interface-PCIe

Supported FPGAs for FL3X Interface-PCIe:

FL3X Interface-PCIe										
FPGA Image	**CAN**	**FlexRay**	**ETH**	**LIN**	**SENT**	**SPI**	**UART**	**K-Line**	**TS**	**AES/ SecOC**
0x000001E0	2 (2xHS/FD)	1	1	-	-	-	-	-	X	-

Table 13: Supported FPGA images – FL3X Interface-PCIe (FL3X System-M)

TS: Support for hardware-based timestamps

AES/SecOC: Support for hardware based Secure Onboard Communication

Supported FL3X Tinys for FL3X Interface-PCIe:

FL3X Tiny III	Supported
FlexRay	X
CAN-FD	X
CAN-FD (4 transceivers)	X
CAN-FD-SIC (4 transceivers)	X
100BASE-TX Switch	X
100BASE-T1	X
100BASE-T1 Switch	X
1000BASE-T1	X
LIN/SENT	X
K-Line/UART/SPI	X
1000BASE-T1 Switch MACsec	X
10BASE-T1S TC10	X

Table 14: Supported FL3X Tinys – FL3X Interface-PCIe (FL3X System-M)

2.8.6.1.3 PCIe SFP Quad (LREC9714HF-4SFP)

Supported device variant for PCIe SFP Quad (LREC9714HF-4SFP):

PCIe SFP Quad (LREC9714HF-4SFP)										
Variant	**CAN**	**FlexRay**	**ETH**	**LIN**	**SENT**	**SPI**	**UART**	**K-Line**	**TS**	**AES/ SecOC**
0x00000002	-	-	4 (4x10/100/1000 Mbit/s)	-	-	-	-	-	-	-

Table 15: Supported device variant– 'PCIe SFP Quad (LREC9714HF-4SFP)' module

TS: Support for hardware-based timestamps

AES/SecOC: Support for hardware based Secure Onboard Communication

Supported SFP modules for 'PCIe SFP Quad (LREC9714HF-4SFP)' module:

SFP-Module	Supported
FL3X SFP 1000BASE-T1	X
F24-EX-10070H	X

Table 16: Supported SFP-Modules – PCIe SFP Quad (LREC9714HF-4SFP)

2.8.6.1.4 PCIe SFP+ Dual (LREC9812BF-2SFP+)

Supported device variant for PCIe SFP+ Dual (LREC9812BF-2SFP+):

PCIe SFP+ Dual (LREC9812BF-2SFP+)										
Variant	**CAN**	**FlexRay**	**ETH**	**LIN**	**SENT**	**SPI**	**UART**	**K-Line**	**TS**	**AES/ SecOC**
0x00000002	-	-	2 (2x10Gbit/s)	-	-	-	-	-	-	-

Table 17: Supported device variant– 'PCIe SFP+ Dual (LREC9812BF-2SFP+)' module

TS: Support for hardware-based timestamps

AES/SecOC: Support for hardware based Secure Onboard Communication

Supported SFP modules for 'PCIe SFP+ Dual (LREC9812BF-2SFP+)' module:

SFP-Module+	Supported
LRXP0010-Y3ATR	X

Table 18: Supported SFP-Modules – PCIe SFP+ Dual (LREC9812BF-2SFP+)

2.8.6.2 Slot for Mini-PCIe

For this slot, the only supported device is the 'Mini PCIe SFP Single (DeLOCK 95267)' module.

2.8.6.2.1 Mini PCIe SFP Single (DeLOCK 95267)

Supported device variant for Mini PCIe SFP Single (DeLOCK 95267):

Mini PCIe SFP Single (DeLOCK 95267)										
Variant	**CAN**	**FlexRay**	**ETH**	**LIN**	**SENT**	**SPI**	**UART**	**K-Line**	**TS**	**AES/ SecOC**
0x00000002	-	-	1 (1x10/100/1000 Mbit/s)	-	-	-	-	-	-	-

Table 19: Supported device variant– 'Mini PCIe SFP Single (DeLOCK 95267)' module

TS: Support for hardware-based timestamps

AES/SecOC: Support for hardware based Secure Onboard Communication

Supported SFP modules for 'Mini PCIe SFP Single (DeLOCK 95267)' module:

SFP-Module	Supported
FL3X SFP 1000BASE-T1	X
F24-EX-10070H	X

Table 20: Supported SFP-Modules – Mini PCIe SFP Single (DeLOCK 95267)

2.8.6.3 Slot for CMI/Mini-PCIe

For this slot, following devices are supported:

- Mini-PCI: see chapter 2.8.6.2.
- CMI: The only supported device is the 'CMI 8xDI 8xDO (CMI-DIO02/UB1018)' module.

2.8.6.3.1 CMI 8xDI 8xDO (CMI-DIO02/UB1018)

Supported device variant for CMI 8xDI 8xDO (CMI-DIO02/UB1018):

CMI 8xDI 8xDO (CMI-DIO02/UB1018)			
Variant	**Digital Inputs (DI)**	**Digital Outputs (DO)**	**TS**
0x00000002	8	8	-

Table 21: Supported device variant– 'CMI 8xDI 8xDO (CMI-DIO02/UB1018)' module

TS: Support for hardware-based timestamps

Supported FL3X Tinys for 'CMI 8xDI 8xDO (CMI-DIO02/UB1018)' module:

The 'CMI 8xDI 8xDO (CMI-DIO02/UB1018)' module has a fix layout with digital inputs and digital outputs and does not support any FL3X Tinys.

2.8.6.4 Slot for CMI10G

For this slot, following devices are supported:

- All devices supported by the CMI part of the Slot for CMI/Mini-PCIe (chapter 2.8.6.3)
  - CMI 8xDI 8xDO (CMI-DIO02/UB1018) (see chapter 2.8.6.3.1)
- CMI 10G Dual Port 10GBASE-T (CMI-10GLAN03)

2.8.6.4.1 CMI 10G Dual Port 10GBASE-T (CMI-10GLAN03)

Supported device variant for CMI 10G Dual Port 10GBASE-T (CMI-10GLAN03):

CMI 10G Dual Port 10GBASE-T (CMI-10GLAN03)										
Variant	**CAN**	**FlexRay**	**ETH**	**LIN**	**SENT**	**SPI**	**UART**	**K-Line**	**TS**	**AES/ SecOC**
0x00000002	-	-	2 (x1/2,5/5/10 Gbit/s)	-	-	-	-	-	-	-

Table 22: Supported device variant– 'CMI 10G Dual Port 10GBASE-T (CMI-10GLAN03)' module

TS: Support for hardware-based timestamps

AES/SecOC: Support for hardware based Secure Onboard Communication

Supported FL3X Tinys for 'CMI 10G Dual Port 10GBASE-T (CMI-10GLAN03)' module:

The 'CMI 10G Dual Port 10GBASE-T (CMI-10GLAN03)' module has a fix layout and does not support any FL3X Tinys.

2.8.7 FL3X System-XS

FL3X System-XS										
FPGA Image	**CAN**	**FlexRay**	**ETH**	**LIN**	**SENT**	**SPI**	**UART**	**K-Line**	**TS**	**AES/ SecOC**
0x00000070	1 (1xHS/FD)	-	-	1	1	-	-	-	-	-

Table 23: Supported FPGA images – FL3X System-XS

TS: Support for hardware-based timestamps

AES/SecOC: Support for hardware based Secure Onboard Communication

Supported FL3X Tinys for FL3X System-XS:

The FL3X System-XS has a fix layout and does not support any FL3X Tinys.

2.8.8 FL3X System-XS²

FL3X System-XS²										
FPGA Image	**CAN**	**FlexRay**	**ETH**	**LIN**	**SENT**	**SPI**	**UART**	**K-Line**	**TS**	**AES/ SecOC**
0x00000075	2 (2xHS/FD)	-	-	2	2	-	-	-	-	-

Table 24: Supported FPGA images – FL3X System-XS²

TS: Support for hardware-based timestamps

AES/SecOC: Support for hardware based Secure Onboard Communication

Supported FL3X Tinys for FL3X System-XS²:

FL3X System-XS² has a fix layout and does not support any FL3X Tinys.

2.9 Preconditions and Limitations

There are several preconditions and limitations for FL3X Config. Additional limitations for FL3X Config Gateway can be found in chapter 6.1.11.

Mixed Endianness 28
Byte-order in multiplexed PDUs 28
Frames with length "0" 28
Signals larger than 32-bits 28
Using the minimum value of a 32-bit signed Signal 28
Maximum number of ECUs 28
Considering timings of multiplexed PDUs by an RBS 28
Cluster Bit-order 28
Container/Contained PDUs 28

Secure Onboard Communication 29
Ethernet & SOME/IP 29
Ethernet & IPv6 29
IP Configurations 29
FlexRay Performance issues with some FL3X Device L² FPGA-Images 29
Support for SD-Cards 29
Priority for CAN Frames based on the Frame ID 29
CAN Burst Transmission Behaviour on Bus Startup/Wakeup 30
Link Status Indication for Specific FL3X Tinys with Integrated Switch 30

Mixed Endianness

Mixed Endianness within a PDU is supported under the precondition that segments of signals with the same byte-order are lying within byte-borders. Additionally, single bytes, which contain signals of different byte order, are also supported.

Mixed Endianness is not supported for multiplexed PDUs.

- All elements (switch, static part, dynamic part) of a multiplexed PDU must have the same endianness.
- All signals of the static/dynamic part must have the same endianness as the PDU itself.

Byte-order in multiplexed PDUs

For multiplexed PDUs it is assumed that the byte order of the switch and the corresponding segments is equal. The appropriate Switched PDUs on the other hand can contain signals with a different byte order.

Frames with length “0”

Frames with the length “0” are not supported and cannot be configured in the Bus Config dialog as Tx-Frame.

Signals larger than 32-bits

Standard Signal Handling and signal manipulation is not supported for signals larger than 32-bits.

Signals, which are larger than 32-bits and whose length is not evenly divisible by 8, are not supported.

The bit position of signals, which are larger than 32-bits and whose length is evenly divisible by 8, must be aligned at a byte boundary.

Using the minimum value of a 32-bit signed Signal

Normally a 32-bit signed signal contains values between -2147483648 and 2147483647. If you use the minimum value (-2147483648) in a user function (for example in if-statements) or with a scaling in the FL3X Config Gateway Editor, then the expected results will not always be received, because a C90 compiler is used. This compiler only allows the values between -2147483647 and 2147483647.

Maximum number of ECUs

An RBS at a Connector sends frames. These frames have an original sender ECU (defined in the database). The number of the original sender ECUs, which are used in all connectors together, must not be more than 255.

Considering timings of multiplexed PDUs by an RBS

The following rules are considered by simulating frames with multiplexed PDUs:

- If the multiplexed PDU has its own timing, it will be considered when sending the frame.
- If the multiplexed PDU does not have its own timing, then the timing of the static PDU is used.
- If no static PDU exists or it has no timing, then the timings of the switched PDUs are used.

Cluster Bit-order

Clusters with increasing bit-order (Cluster-parameter “IS-HIGH-LOW-BIT-ORDER =’false’”) are not supported. If you want to work with such clusters, then you must edit the database file and delete the corresponding parameter or set it to ‘true’.

Container/Contained PDUs

- There are several limitations for Container/Contained PDUs: A direct manipulation (PDU Loss) for Container-PDUs is no longer available (this was possible with FlexConfig RBS 4.2). Use the manipulation “Loss” for the corresponding Frame instead.
  If a project which contains manipulations for Container-PDUs is loaded, then these manipulations are not removed automatically. You must remove them manually.
- Limitation for routings from Multiplexed-PDUs to Contained-PDUs:
  The dynamic PDU must not be divided into segments. Any segmentation lead to invalid data on the routing target.
- More information about Container PDUs can be found in chapter 5.19.3.

Secure Onboard Communication

The Secure Onboard Communication needs an OEM-dependent configuration. Please read the OEM related manual.

Ethernet & SOME/IP

- Routings and manipulations are supported for the SOME/IP datatypes with the following preconditions:
  The datatype must be on the lowest level of the data structure.
- SOME/IP is only supported based on AUTOSAR databases (Issue 439123).

Ethernet & IPv6

- The IPv6 ‘Flow Label’ field according to RFC6437 is not supported.
  The ‘Flow Label’ field is set to ‘0’ for all IPv6 packets.

IP Configurations

The maximum number of different IP configurations for a single project is 100.

FlexRay Performance issues with some FL3X Device L² FPGA-Images

Some FPGA-Images for FL3X Device-L² do not support the full performance for FlexRay on all connectors. The following FPGA-Images are affected:

- 0x00900080 & 0x10901080: Connector 3A/3B
- 0x10931082: Connector 3A/3B, Connector 4A/4B
- 0x10951084: Connector 3A/3B (see chapter 2.8.3.1 for more information)
- 0x10971086: Connector 3A/3B (see chapter 2.8.3.1 for more information)

Workaround: Use these connectors only for FlexRay configurations with reduced bus load.

Support for SD-Cards

Supported SD-Cards:

- micro SD form factor
- Up to 400 GB
- FAT32 file system

To enable access to the SD-Cards, specific User Defines must be set in project configuration:

- MOUNT_SDCARD_EXTERN
- MOUNT_SDCARD_INTERN

(See chapter 5.6.7.8)

Additional control of SD-Cards is available via the FL3X Diagnostics Website (see chapter 8) of the used hardware platform.

Priority for CAN Frames based on the Frame ID

For CAN-HS and CAN-FD, a frame with a lower frame ID has a higher priority when it is transmitted (bitwise arbitration of frame IDs).
The internal scheduler of FL3X Config (which puts TX frames into the message buffer of the CAN-CC) only considers the (cyclic) timing of frames and ignores any priority based on the frame-ID. This may lead to a short delay/jitter between two consecutive frames with the same frame-ID.

CAN Burst Transmission Behaviour on Bus Startup/Wakeup

FL3X Config uses internal buffers for each CAN-CC. The buffers are filled based on the RBS/GW behavior. E.g. an RBS with a CAN TX frame with a cyclic timing of 50ms will lead to a new buffer entry every 50ms. The CAN-CC tries to send any frame hold in the buffer. When the CAN-CC tries to send a frame and there is no ACK, it will not clear the buffer and tries to send the frame again and again (until it can finally be sent).

In scenarios, where the CAN bus has no external ACK, the buffers will get filled. Any TX frames exceeding the maximum buffer size are discarded (and increase the “TX Error counter” shown on the FL3X Diagnostics Website (see chapter 8)).
As soon as an external ACK is available on the CAN bus, the CAN-CC starts to empty the buffers and transmits all frames in the buffers as fast as possible. The cyclic timing of any frames stored in the buffers in this case are not considered. So, this results in kind of a transmission burst.

Link Status Indication for Specific FL3X Tinys with Integrated Switch

The following FL3X Tiny variants have an integrated Ethernet switch:

- FL3X Tiny3 2*100BASE-TX Switch
- FL3X Tiny3 3*100BASE-T1 Switch
- FL3X Tiny3 3*1000BASE-T1 Switch MACSec

Due to the internal design of these switch-based FL3X Tinys, a logical link is always established between the FL3X device and the integrated switch, even if no physical link is present on the external switch ports (i.e., the switch arms toward the connector have no active link).

As a result, the “Link Up” status shown on the FL3X Diagnostic Website is not meaningful for these FL3X Tiny types. The displayed link status reflects only the internal connection to the switch and does not indicate the actual physical link state of the external Ethernet interface.

Impact

The link information shown in FL3X Diagnostics cannot be used to validate external connectivity for the above FL3X Tiny models.

Test specifications and any diagnostic procedures relying on the “Link Up” field must be adapted accordingly.

Recommendation

Use alternative diagnostic methods (e.g., external network tools or physical layer measurements) to verify the presence or absence of a real Ethernet link on the connector side.

3 Licensing

3.1 Software Licenses (FL3X Config)

There are three different FL3X Config versions. Two of them require a license. The difference between those versions is described in chapter 3.1.1.

- FL3X Config Basic version: This version requires no license and is free of charge.
- FL3X Config Professional version: This version requires a FL3X Config Professional license.
- FL3X Config Enterprise version: This version requires a FL3X Config Enterprise license.

3.1.1 Differences between the FL3X Config Versions

The FL3X Config Basic version includes all basic features (e.g., RBS, FL3X Config Gateway, FL3X Config Control or FL3X Config Analyzer). There are several usability and collaboration features, which are not included in the FL3X Config Basic version.
The FL3X Config Professional version provides a better usability for some use cases.
The FL3X Config Enterprise version includes the features of the FL3X Config Professional version and provides enhanced collaboration features.

The following table shows the differences:

Feature	Information	FL3X Config Basic	FL3X Config Professional	FL3X Config Enterprise
Drag&Drop for Buses	[Main Window, Change Device] This feature allows to assign a bus to another connector via Drag&Drop.	-	✓	✓
Database Update	[Database Update] A database update is available between two different database files.	-	✓	✓
Change Device - Automatic Bus Assignment	[Change Device] All configured buses can be assigned automatically to the matching connector of the selected target device.	-	✓	✓
Autorouting for Buses	[FL3X Config Gateway] Create all routings between two selected buses.	-	✓	✓
Create Child Routing	[FL3X Config Gateway] Split a frame routing or a PDU routing into its childs (e.g., a PDU routing is split into signal routings).	-	✓	✓
Create Parent Routing	[FL3X Config Gateway] Merge multiple routings to a parent routing (e.g., merge all signal routings for a PDU into a PDU routing).	-	✓	✓
Automatic Scaling for Routings	[FL3X Config Gateway] For signals with compu methods, the necessary scaling functions are automatically added when a signal routing is created.	-	✓	✓
Find Matching Targets for Routings	[FL3X Config Gateway] When an element is selected in the Source area, the matching element in the Target area is selected automatically and vice versa.	-	✓	✓
Highlight Matching Elements for Routings	[FL3X Config Gateway] When an element is selected in the Source area, all elements in the Target area show a flag and vice versa. This flag indicates if a routing between those elements is possible with or without limitations.	-	✓	✓
Mark Gateway Elements with Colors	[FL3X Config Gateway] The elements in the Source/Target/Routings area can be marked with different colors.	-	✓	✓
Gateway CSV Export	[FL3X Config Gateway] The elements of the Source, Target and Routings area can be	-	✓	✓

Feature	Information	FL3X Config Basic	FL3X Config Professional	FL3X Config Enterprise
	printed or exported. Supported formats for the export are Microsoft Excel and csv.			
Gateway Routings Import/Export	[FL3X Config Gateway] Routings can be imported and exported. This feature uses Microsoft Excel files with a predefined format.	-	-	✓
Projects Optimized for Version-Control Software	[Collaboration] UserFunctions are saved in separate .c files for easier handling with version control software.	-	-	✓
Projects Optimized for External IDE	[Collaboration] Create additional files to edit UserFunctions in Visual Studio Code (including Code Snippets).	-	-	✓
UserFunctions Export and Merge	[Collaboration] Export UserFunctions to .c-files. Import UserFunctions from other projects.	-	-	✓
Create Partner Project	[Project Generation] Create partner project with the opposite configuration of the current project for automated FL3X Device tests.	-	-	✓
License Type				
	No license required	✓		
	Local license (Dongle or PC bound license file)		✓	
	Floating license (Dongle or PC bound license file)			✓
Support level				
	No guaranteed response time	✓		
	regular 1 workday reponse time, max. 5 workday reponse time		✓	✓
Communication				
	Only via E-Mail	✓		
	Phone & E-Mail		✓	✓

Table 25: Differences between the FL3X Config versions.

3.1.2 View Available Licenses

The current used licenses of a running FL3X Config instance are shown via [Help] -> [License Information] (see chapter 5.4.6.4 License Information)

The detailed information about the available licenses is shown by the “CodeMeter WebAdmin”.

To open the “CodeMeter Webadmin” use the button “Web Admin” in the CodeMeter Control Center (see Figure 1: CodeMeter Control Center) or use the following link: http://localhost:22352/container.html

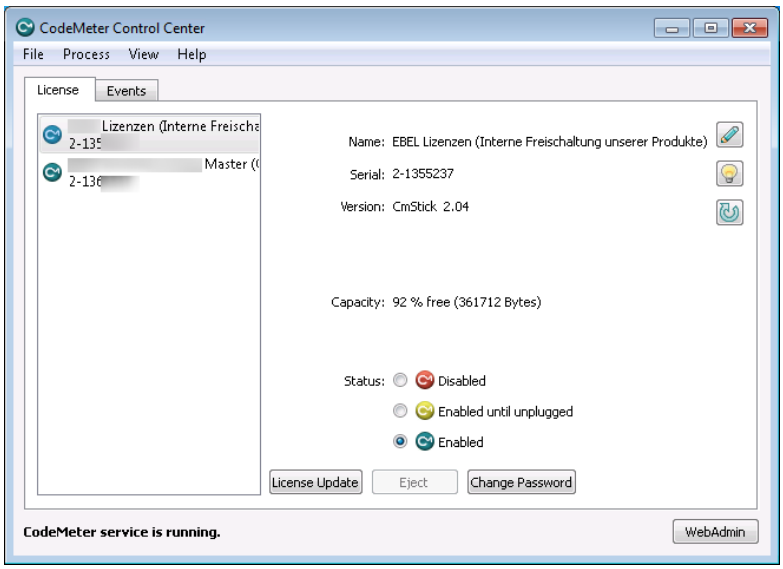


Figure 1: CodeMeter Control Center

3.1.3 License Allocation

FL3X Config tries to allocate a license (local or floating) when the application starts. If no license is found or could be allocated at start of the application, FL3X Config will start as FL3X Config Basic version.

An allocated license is checked every 30 seconds. If the allocated license if no longer available, the software FL3X Config will show that the license is missing and all features which require the license are blocked until the license is available again.

	NOTICE
	FL3X Config can be used on a Windows PC with parallel user logins. If the Windows PC allows parallel user logins, each user login requires a separate FL3X Config license. Example: 1) User A is logged into the PC and has a running FL3X Config instance (this instance has allocated one FL3X Config license). 2) User A locks his session (FL3X Config is still running). 3) User B now logs into the PC and starts FL3X Config. => The FL3X Config session of User B allocates a new FL3X Config license. => So in this example, **two** FL3X Config licenses are required if both users want to use the Professional/Enterprise features (see also Table 25).

	NOTICE
	Using FL3X Config via Remote Desktop: FL3X Config locks the license for the current session. Reconnecting to a Remote Desktop will generate a new session. A running FL3X Config instance locks the license for the previous session. Therefore, a new additional FL3X Config instance will not get a valid license. Please close the previous FL3X Config instance and then open new instances to get valid licenses.

3.2 Runtime Licenses (FL3X Device, FL3X System, Virtual FL3X Device)

Some features require a runtime license. Runtime licenses are set with an expiration date. This date will be compared with the release date of the FL3X Config version the project was compiled with.
This release date must be smaller or equal than the license expiration date to use the licensed features.

There are also ‘lifetime’ licenses available with no expiration date.

The following table and Figure 2 show the required runtime license for each feature.

Feature	Information	No runtime license required	FL3X Config Standard Runtime	FL3X Config Extended Runtime
User Function				
Initialize User Function		✓	(✓)	(✓)
Main User Function		✓	(✓)	(✓)
User Libraries		✓	(✓)	(✓)
Analyzing / Logging				
Bus Analyzing		✓	(✓)	(✓)
Bus Logging		✓	(✓)	(✓)
Bus Systems (Common)				
SENT Configurations (via User Library)		✓	(✓)	(✓)
Ethernet Configurations for PC-Ethernet Connector (e.g. Time Synchronization)		✓	(✓)	(✓)
Bus Systems (RBS)				
LIN Bus Configurations		✓	(✓)	(✓)
CAN-HS Bus Configurations		✓	(✓)	(✓)
CAN-FD Bus Configurations		-	✓	(✓)
FlexRay Bus Configurations		-	✓	(✓)
Ethernet Bus Configurations		-	✓	(✓)
Bus Systems (Gateway)				
LIN Bus Configurations		✓	(✓)	(✓)
CAN-HS Bus Configurations		✓	(✓)	(✓)
CAN-FD Bus Configurations		-	-	✓
FlexRay Bus Configurations		-	-	✓
Ethernet Bus Configurations		-	-	✓
Virtual FL3X Device for Windows				
Virtual FL3X Device for Windows	Without a runtime license the Virtual FL3X Device for Windows will not work. A FL3X Config Standard Runtime or a FL3X Config Extended Runtime is always required (based on the features used).	-	✓	✓

Table 26: List of features that require a Runtime License

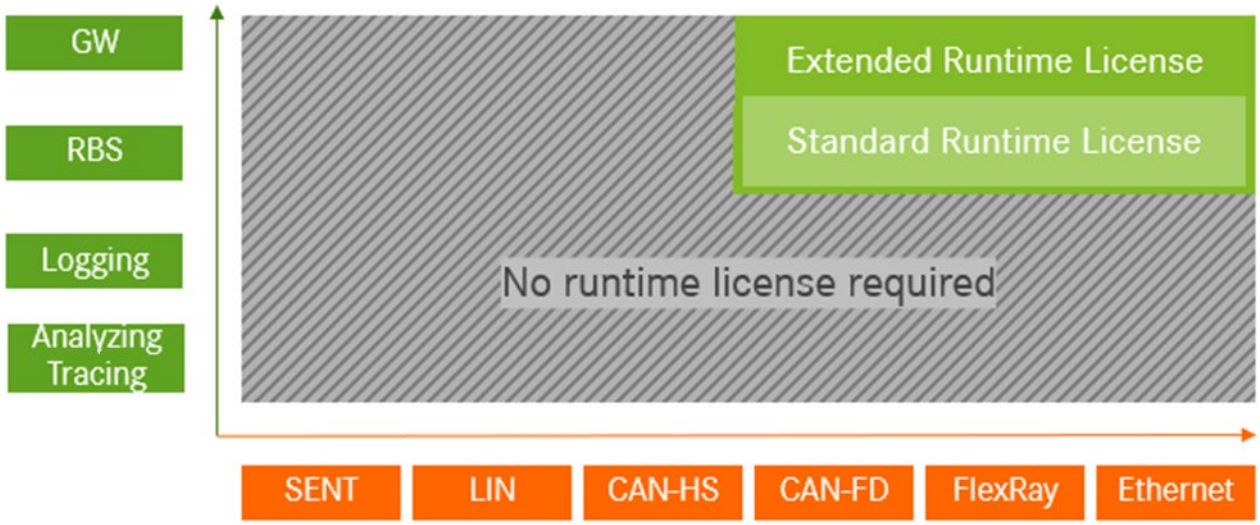


Figure 2: List of features that require a Runtime License

3.2.1 View Available Runtime Licenses

To view the licenses of a device, use the License Information view in the Device Manager (see chapter 5.2 Device Manager).

4 Software Installation and Update

	Reference
	More information about the used FL3X Config version can be found in the following documents: 3-01140A01-D012_Release_Notes_FL3X_Config 3-01140A01-D013_Open_Issues_FL3X_Config

4.1 Preconditions

The installation requires administrative rights on your client computer.

4.1.1 Minimum System Requirements

Operating System: Windows 10 (64-bit) or Windows 11 (64-bit)

CPU: 1 GHz

RAM: 3 GB

Storage: 4,5 GB

Other:

- Windows Installer 4.5 or higher
- Microsoft .NET 8.0 Desktop Runtime (8.0.24)

4.1.2 Recommended System Requirements

Operating System: Windows 10 (64-bit) or Windows 11 (64-bit)

CPU: 2,5GHz (dual core)

RAM:

- 16 GB for projects with FlexRay/CAN/LIN (without Ethernet)
- 32 GB for projects with Ethernet
- 64 GB for projects with Ethernet and RBS Based Replay

Storage: 6,5 GB

Other:

- Windows Installer 4.5 or higher
- Microsoft .NET 8.0 Desktop Runtime (8.0.24)

4.1.3 .NET 8.0

The FL3X Config setup includes the .NET Desktop Runtime 8.0.24 installer. The .NET runtime is automatically installed if required.

	NOTICE
	The Microsoft .NET 8.0 Desktop Runtime is necessary to run the application. Please visit the Microsoft homepage for prerequisites and download of this runtime. Information: https://learn.microsoft.com/de-de/dotnet/core/install/windows Download: https://dotnet.microsoft.com/en-us/download/dotnet/8.0

4.1.4 CodeMeter Runtime

The FL3X Config setup includes the CodeMeter Runtime installer. The CodeMeter runtime is automatically installed if required.

	NOTICE
	The Wibu CodeMeter User Runtime is necessary to run the application. Please visit the Wibu-Systems homepage for prerequisites and download of this runtime. https://www.wibu.com/us/support/user/downloads-user-software.html

4.2 Installation

To install FL3X Config the installation packages *FL3XConfig_Setup_%version%_64Bit.exe* is required. Start the installation by double-clicking on the installation package and following the instructions of the installation wizard.

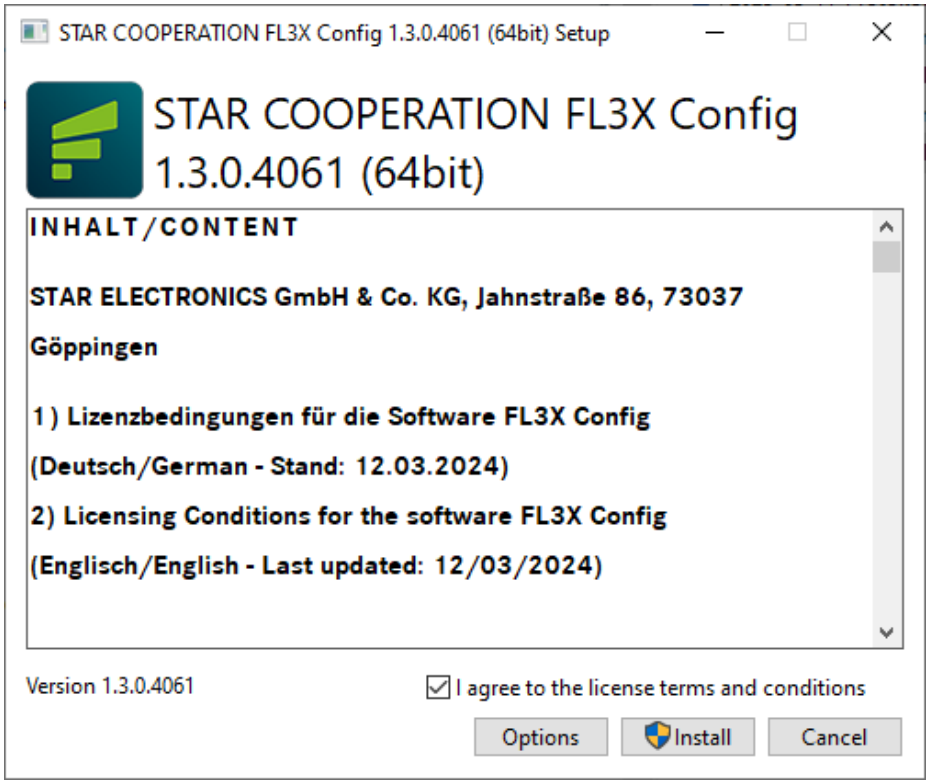


Figure 3: FL3X Config – Installation wizard

Use the [Options] button to change the installation settings:

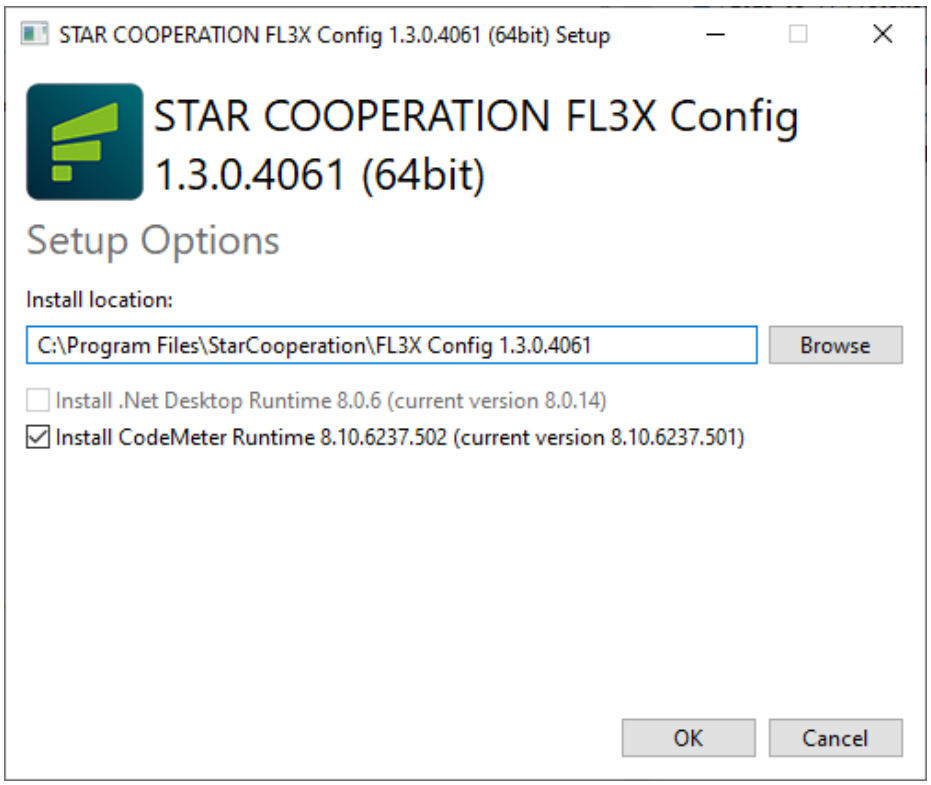


Figure 4: FL3X Config – Installation wizard – Options menu

- Install location
  The path where FL3X Config is installed
- Install .Net Desktop Runtime
  Installs the required .Net Desktop Runtime
  Enabled when a newer version is provided
  Disable this option to avoid installing the.Net Desktop Runtime
- Install CodeMeter Runtime
  Installs the required CodeMeter Runtime
  Enabled when a newer version is provided
  Disable this option to avoid installing/updating the CodeMeter Runtime

	NOTICE
	If the runtime options (.Net Desktop or CodeMeter) are disabled manually, it is your responsibility to ensure that the required runtimes are installed.

	NOTICE
	The following additional installers are available on request: *FL3XConfig_%version%_64Bit.msi* This installer does not include the .NET Desktop Runtime. Included modules: - CodeMeter Runtime (merge modules) - FL3X Config *FL3XConfig_Reduced_%version%_64Bit.msi* This installer includes only FL3X Config Included modules: - FL3X Config *CodeMeterRuntime64.msi* This installer installs the CodeMeter Runtime Included modules: - CodeMeter Runtime Please contact the *STAR ELECTRONICS* support team.

4.3 Command-Line Options (CLI)

MSI packages

Open a command prompt (cmd.exe) and type "*msiexec*". A window will open and show all command-line parameters.

More information can be found on the Microsoft documentation page:

	Reference
	https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/msiexec and https://learn.microsoft.com/en-us/windows/win32/msi/command-line-options

EXE installer

The exe installer provides following command-line options:

`InstallDotNetPackage=0` -> Disable installing the .Net Desktop Runtime

`InstallCodeMeterRuntime=0` -> Disable installing the CodeMeter Runtime

E.g. to runt the installation without the CodeMeter Runtime:

`FL3XConfig_Setup_V1.5_64Bit.exe InstallCodeMeterRuntime=0`

	NOTICE
	If the runtime options (.Net Desktop or CodeMeter) are disabled manually, it is your responsibility to ensure that the required runtimes are installed.

`-q, -quiet, -s, -silent` -> Silent install. The installer runs an installation without displaying a user interface. No prompts, messages, or dialog boxes display to the user.

`-passive` -> The installer displays a progress bar to the user that indicates that an installation is in progress but displays no prompts or error messages to the user.

`-norestart` -> The installer never restarts the computer after the installation.

`-forcerestart` -> The installer restarts the computer after every installation.

`-promptrestart` -> Displays a message that a restart is required to complete the installation and asks the user whether to restart the system now. Cannot be used with the `/quiet` option.

`-layout` -> create a local image of the bootstrapper (i.e. download files so they can be burned to DVD)

`-l, -log` -> Writes logging information into a log file at the specified existing path. The path to the log file location must already exist.

`-uninstall` -> Uninstall product option. Uninstalls the product.

`-repair` -> Repair the installation (or install if not installed).

More information can be found on the Microsoft documentation page:

	Reference
	https://learn.microsoft.com/en-us/windows/win32/msi/standard-installer-command-line-options

4.4 Installing Multiple Versions

Multiple parallel versions of FlexConfig RBS/FL3X Config can be installed. Thus, it is possible to use multiple versions of FL3X Config at the same time. Each project contains information about the last used version. If this version is installed, it is used automatically when the project is opened. If that version is currently not installed, the preferred version can easily be chosen through the "Version Selector" (see Figure 5). To open the "Version Selector" manually, right-click on the project file and select '*Open with version...* '. To use the newest version automatically, activate the "*Always open with newest version*" option.

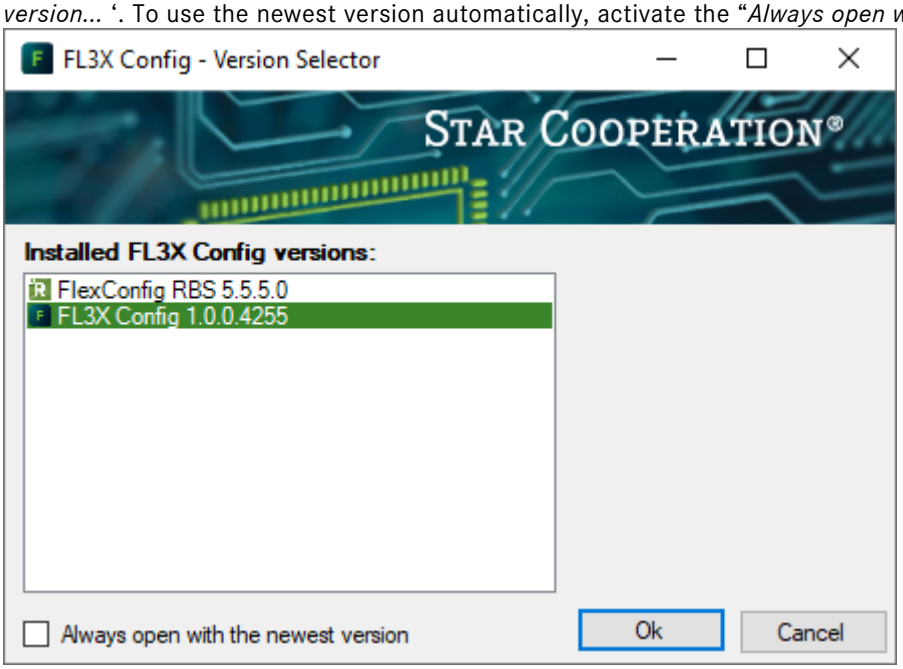


Figure 5: FL3X Config – Version Selector

4.5 Start FL3X Config

To start the application, click *"Start"* button from the Windows taskbar and then select *'FL3X Config %Version%–> FL3X Config %version%.exe'*.

Any license container with a FL3X Config Professional license or FL3X Config Enterprise license must be available before starting the application. If there is no license container (dongle, floating license via network) available, the application will run in FL3X Config Basic mode.

	Information
	For more information about the license check, see chapter 3.1.2 and chapter 3.1.3. Information about a connected dongle can be found in the “License Information” dialog. Click *‘Help -> License Information…’* to open the dialog (see chapter 5.4.6.4)

4.5.1 Software License Expire Dialog

If the following dialog is shown, the license expires soon. For more Information, please see the chapter 5.22.1.1

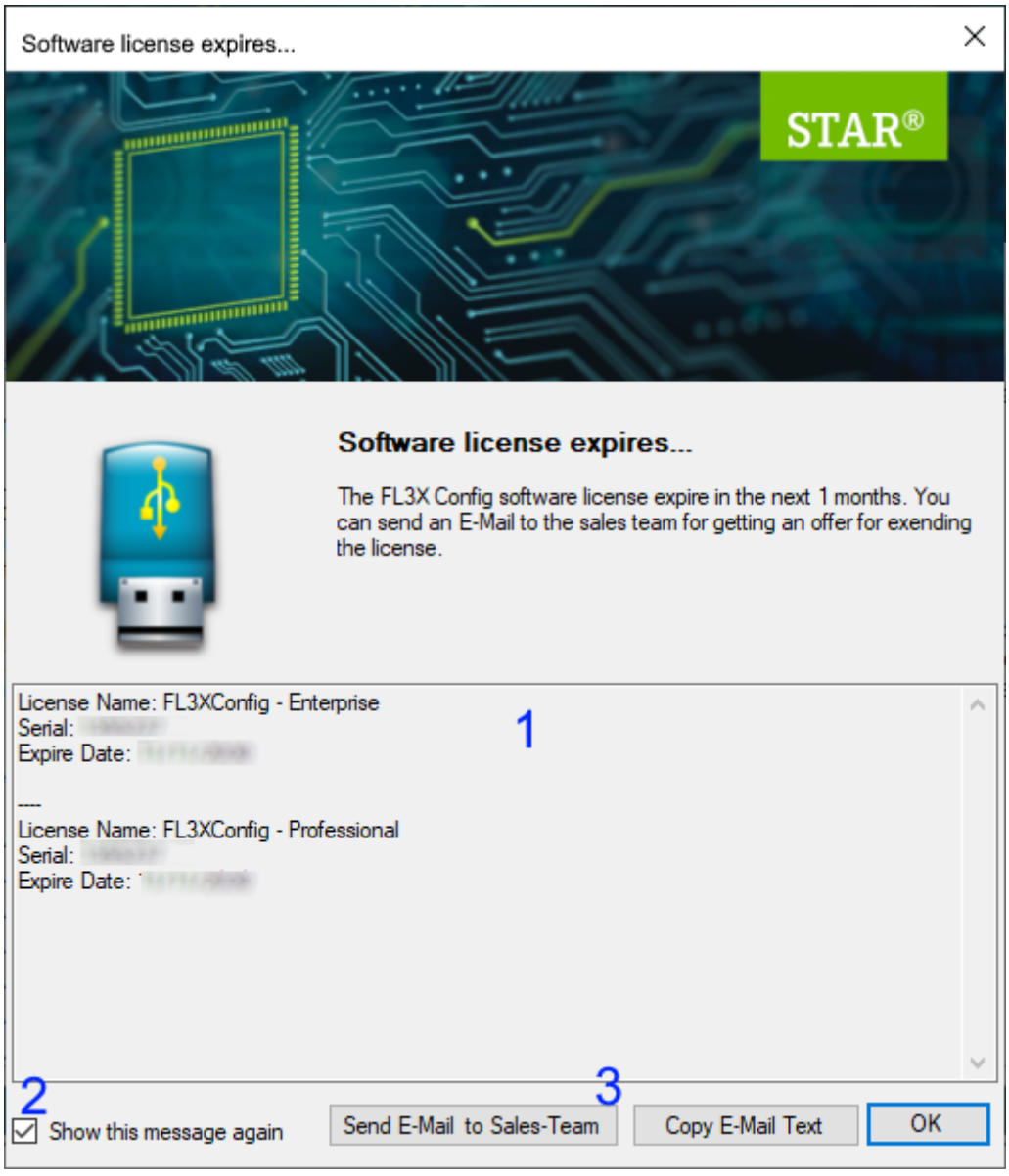


Figure 6: FL3X Config – Software license expire dialog

No	Description
1	Information about expiring software license(s).
2	Deactivate this option if you do not want to see this dialog again. This will also deactivate the corresponding setting in the global settings dialog.
3	*Send E-Mail to Sales-Team*: Opens the default E-Mail application with a predefined text and information about the expiring licenses. *Copy E-Mail Text*: Copies the text to the clipboard.

4.6 Update FL3X Config

To update FL3X Config it is necessary to follow these steps.

Step 1: (optional) Remove FL3X Config — Uninstall FL3X Config as described in chapter 4.7.

Step 2: Install new version — Install the new version by running the setup file as described in the previous chapter.

	Information
	Since FL3X Config supports parallel installation of different versions, removing the older version is not necessary (see chapter 4.3).

4.6.1 Automatic Updates

See also chapter 5.4.6.2. FL3X Config automatically checks for new versions (if enabled). The automatic update will install the latest available FL3X Config version and all installed OEM-Packages.

To check if the automatic update has correctly updated all OEM-Packages, start the new installed version, click “New Project” and check if all expected OEM Packages are available at “Project types”.

For problems with the auto update, see also:

Chapter 12 Troubleshooting

12.1 FL3X Config – General

- 7 Problems with downloading the automatic application update.
- 8 Problems with installing the automatic application update.

4.7 Uninstall FL3X Config

To uninstall FL3X Config please follow these steps.

Step 1 — Open the Control Panel of the Windows OS.

Step 2 — Select ‘Programs and Features’.

Step 3 — Select the “FL3X Config” application and click the *“Uninstall”*-Button.

4.8 Update FL3X Config Licenses

For more information about the licenses, see chapter 3 Licensing

	NOTICE
	Only license containers (Dongles) which are connected to the local PC can be updated. For floating licenses perform the manual update directly at the server or connect the dongle with the local PC.

4.8.1 License Update via CodeMeter License Central WebDepot

Software licenses are updated via CodeMeter License Central WebDepot.

1) Ensure that you received a Ticket number via E-Mail from the STAR ELECTRONICS sales team.
   If you did not receive an E-Mail with a ticket number, please contact the STAR ELECTRONICS sales team: sales-ee@star-cooperation.com
2) Use a web browser and open the following URL:
   https://license.star-cooperation.com
3) Follow the instructions.

For problems with the license update, see also:

Chapter 12 Troubleshooting

12.1 FL3X Config – General

- 4 License transfer could not be confirmed
- 5 How to update the ‘Certified Time’ of a CmContainer (Dongle)

5 FL3X Config

5.1 The Main Window

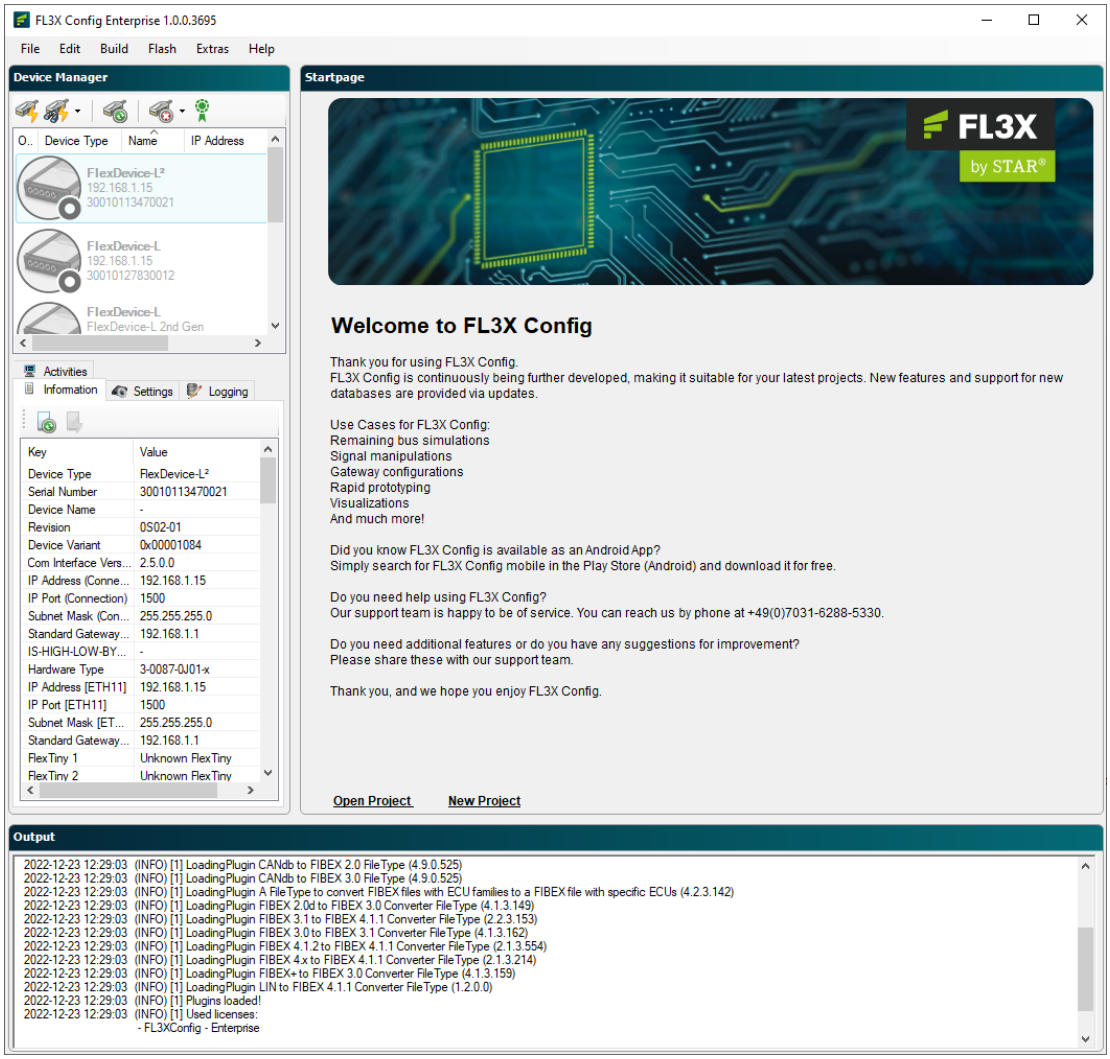


Figure 7: FL3X Config Main Window (after starting FL3X Config)

After starting FL3X Config, the window shown in Figure 7 appears. The “Startpage”-section offers the possibility to directly open projects, which have been edited in the last sessions (“Open existing project”). Additionally, projects can be opened using the “*Open Project*” option or to generate a new project with “*New Project*”.

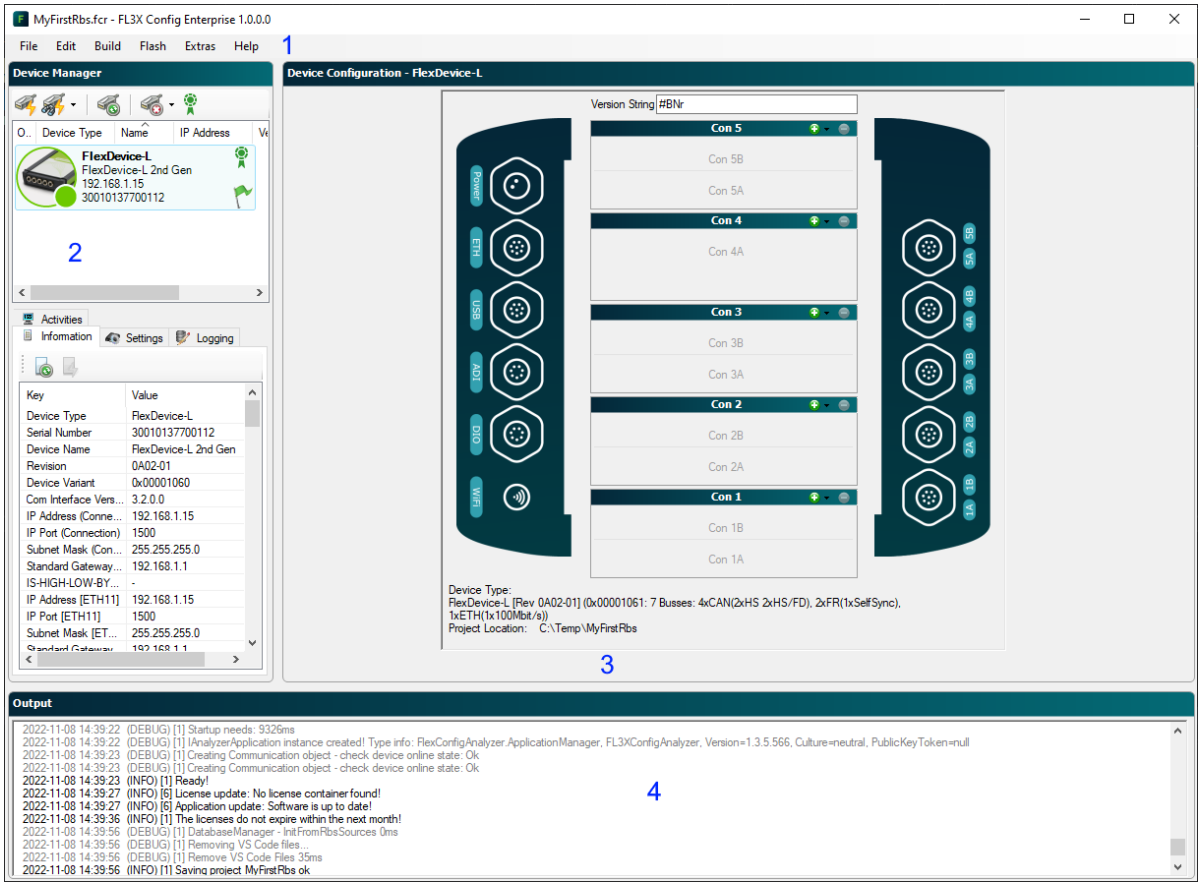


Figure 8: FL3X Config Main window after loading a project (selected device: FL3X Device-L2)

After this has been done the main window has the form as shown in Figure 8. Here an overview of the general project configuration is given.

No	Description
1	Menu bar of FL3X Config. For a detailed description please refer to chapter 5.4.
2	The “Device Manager”-section of the main window. Here one of the available devices can be selected. The list of devices can be changed via the buttons at the top of the “Device Manager”-section. The buttons are *“Connect device”, “Search devices”, “Refresh”* and *“Disconnect”* (selected/all devices). Different properties, settings and activities of a selected device are shown here. For a detailed description please refer to chapter 5.2.
3	The “Device Configuration”-section of the main window. Here the different connectors of the appropriate device for the project can be configured. For a detailed description please refer to chapter 5.5.
4	The “Output”-section of the main window. Here the states of different processes during project generation are displayed. For a detailed description please refer to chapter 5.2.5.

Table 27: Description of the FL3X Config Main Window

5.2 Device Manager

The following figure (Figure 9) shows the “Device Manager”-section of the main window.

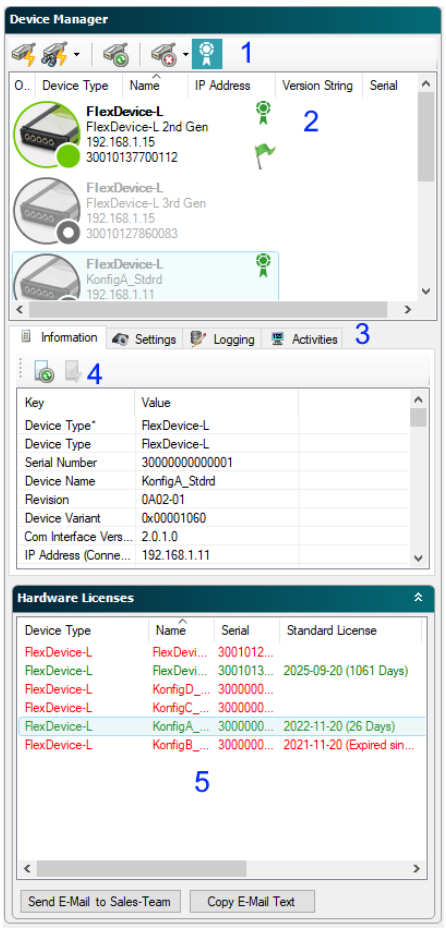


Figure 9: FL3X Config Main Window – “Device Manager” section

The items of this figure are explained in Table 28.

No	Description
1	These buttons provide access to the currently available devices (e.g., FL3X Device-S). The meaning of the buttons in consecutive order is: • *"Connect device..."*: This button adds a new device to the device list. The IP-Address of the new device must be configured. After clicking the *"OK"*- button the device will be connected and added to the device list. • *"Search devices"*: To search for devices, which are connected via Ethernet, this button can be clicked. The used IP-Address is based on the IP settings of the *'Extras->Settings'* menu. **Attention:** Administrator rights are required for *"Search devices"*. Additionally, the arrow on the right side of the button can be clicked. After this has been done, a drop-down menu opens, which provides the following options. o *'Search devices'*: This menu item provides the possibility to search for devices using the IP Address from the '*Extras->Settings*' dialog (s. chapter 5.22.1.6). o *'Advanced->Search devices*': This menu item can be used to search for devices using the IP Address which can be specified here. • *"Refresh"*: This button provides the possibility to refresh the properties of the devices shown in the devices list. • *"Delete"*: This button provides the possibility to delete the device(s) from the devices list. Using the drop-down menu, the following options are available: o *'Selected'*: If this menu item is used, the selected device will be deleted from the devices list. *'All'*: If this menu item is used, all devices of the devices list will be deleted. • *"Show/Hide Hardware Licenses":* This button shows or hides the hardware licenses area, which is marked with the number **5**. For more information, see chapter 5.2.6.
2	In this section a list of devices is given, which have been used in previous sessions, or which are currently connected with FL3X Config using the IP settings (the IP-settings for the connection can be done in the '*Extras->Settings*' menu or via the "*Search devices*"-button). All devices within the list that are greyed out, are not connected with FL3X Config. The row above contains different properties of devices ("Online", "Name", "IP Address", "Version String", "Serial"). Clicking on one of these properties enables one to sort the devices shown in alphabetical order (ascending or descending). On the right side of each listed device, different properties are shown ("Device Name", "Version String", "IP Address", "Serial-Number"). Additionally, the following flags can be shown:  : This flag shows the state of the runtime license with colors: • Green: The runtime license is available and does not expire in the next 180 days • Orange: The runtime license is available and expires in 90-180 days • Red: The runtime license is either expired or expires in 90 days. This flag is not shown if no runtime license is available. The runtime license information is also shown as tooltip by this flag. For more information, see chapter 5.2.6  : This flag signalizes that the current project is compatible with the device. The project and the device are compatible, if: • The device type of the project and the type of the device are the same. • The configured bus types must match the installed FL3X Tinys of the selected device. • The necessary licenses (Standard, Extended) are available on the device.
3	This section contains four tabs ("Information", "Settings", "Activities", "Logging"), which give information to the currently selected device. For more information, see chapters 5.2.1, 5.2.2, 5.2.3, and 5.2.4. If the selected device is a FL3X System M, each slot device will be shown also in separate tabs with their own "Information" and "Setting" pages. The slot device may have some restrictions in the "Settings" tab. The corresponding configuration of such options are disabled in the "Setting" tab.
4	With "*Refresh*", the information can be read from the selected device. With "*Write Properties*", write the current settings to the selected device.
5	This window shows the information about the available hardware licenses of all devices shown in the device list. For more information, see chapter 5.2.6.

Table 28: Description of the "Device Manager" section (Main Window)

5.2.1 Information Tab

The “Information”- tab of the “Device Manager” shows different properties (keys) and their corresponding values (s. Figure 10). These keys are appropriate to the currently selected device and are not editable.

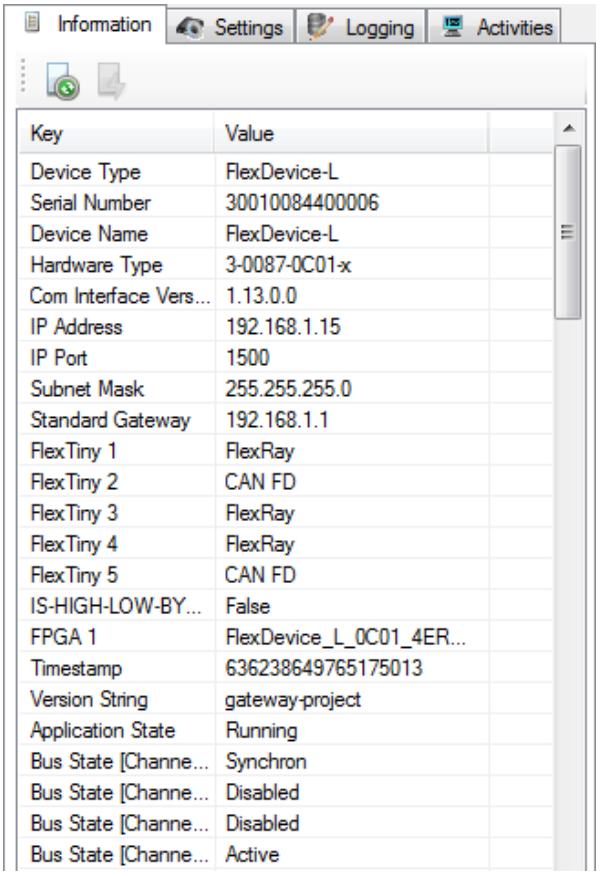


Figure 10: “Device Manager” section – “Information” tab

All properties are shown and described in Table 29.

Key (Property)	Description
Application State	This property returns the state of the application, which is running on the selected device. Different possible states are: • **Initializing**: The application is in the initialization state. Among other things, important communication interfaces and the database are being initialized. • **Running**: After initialization, the application is running. • **Stopped**: The application has been stopped. This, for example, is possible by using the PC-HW-Interface [2]. • **Error**: The application is in error state if an error occurred during the initialization process. This is possible, if, for example, no runtime licenses are available.
Bus License CAN-FD	Only for FlexConfig RBS and for devices which supports bus licenses (e.g. FL3X Device-S) If the **CAN-FD** bus license is activated, it returns *true*; otherwise it returns *false*.
Bus License Ethernet	Only for FlexConfig RBS and devices which supports bus licenses (e.g. FL3X Device-S) If the **Ethernet** bus license is activated, it returns *true*; otherwise it returns *false*.
Bus License FlexRay	Only for FlexConfig RBS and devices which supports bus licenses (e.g. FL3X Device-S) If the **FlexRay** bus license is activated, it returns *true*; otherwise it returns *false*.
Bus State [Channel *x*]	This property returns the state of the different busses. Each number *x* corresponds to an appropriate connector. The assignment depends on the device, FPGA images and FL3X Tinys. The assignment could be: • **CON1**, **CON2**, **CON3**, **etc**. • **CON1A**, **CON1B**, **CON2A**, **CON2B**, **CON3A**, **CON3B**, **CON3C**, **etc**. The possible states are: • **Unknown**: The state of the bus has not yet been set. • **None**: The state of the bus is currently undefined. • **Active**: The bus is active (relevant for CAN). • **Inactive**: The bus is inactive (relevant for CAN). • **Disabled**: The bus is disabled. • **Synchronous**: The bus is in normal active state (FlexRay). • **Asynchronous**: The bus is in normal passive state (FlexRay). • **Startup**: The bus is in start-up state (FlexRay). • **Halt**: The bus is in halt state (FlexRay). • **FIFO error**: A FIFO error occurred (CAN, FlexRay).
Bus Type [Channel *x*]	This property provides the bus type of the FL3X Tiny in the corresponding channel. Possible bus types are: **CAN**, **CAN-FD, LIN**, **Ethernet**, **FlexRay**, etc. See **FL3X Tiny [Channel *x*]**.
Com Interface Version	This property contains the version of the PC-HW-Interface [2] used within the selected device.
Deadline Violations [Channel *bus-type.x*]	This property returns the number of Deadline Violations. It only applies to FlexRay and returns '0' for other bus types. The property shows the number of the frames which could not be sent from the device in the scheduled slot. This is normal FlexRay behavior for the dynamic segment. The RBS/gateway is not necessarily faulty if many deadline violations occur; the dynamic segment might be overloaded.

Key (Property)	Description
Deadline Violation Overflow [Channel *bus-type.x*]	This property returns *true*, if a Deadline Violation overflow occurred; otherwise it returns *false*.
Device Name	This property contains the user-defined name of the device. The name of the device can be set in the “Settings”-tab.
Device Type	The device type could be: **FL3X Device-S**, **FL3X Device-L**, **FL3X Device-L²**, **FL3X Device-PXIe**, **FL3X Device-PCIe**, **FL3X System-M**, **FL3X System XS**, **FL3X System XS²**, **FL3X Interface-S**, **FL3X Interface -L**, **FL3X Interface -L²**, **FL3X Interface -PXIe**, **FL3X Interface -PCIe**
Device Variant	The Device Variant is a hexadecimal number, consisting of all FPGA image types that are used in the project. All FPGA image types are appended after each other, to form the Device Variant. See chapter 5.4.1.1 (New Project Dialog) and **FPGA *x***.
FL3X Tiny *x*	This property provides the type of the FL3X Tiny (e.g. FlexRay).
FL3X Tiny [Channel *x*]	This property provides the type of FL3X Tiny in the corresponding channel.
FPGA *x*	The FPGA property provides the type of the FPGA image, as a hexadecimal value. The first two digits refer to the revision of the FL3X Device (e.g. “0x1089” would refer to Rev. 2 of the FL3X Device, while “0x0089” would refer to Rev. 1). The last two digits correspond to the image version.
FPGA Version *x*	This property contains the version number of the corresponding FPGA image.
Hardware Type	In the current version of FL3X Config the Hardware Type corresponds to the Device Type.
IP Address	This property displays the current IP-Address of the device.
IP Port	This property displays the current IP-Port of the device.
IP Address [*adapter name*]	IP address of the specified adapter is displayed by this property. The IP-Address can be set in the “Settings”-tab or in the *‘Extras->Settings’*-menu.
IP Port [*adapter name*]	IP port of the specified adapter is displayed by this property. The IP-Port can be set in the “Settings”-tab or in the *‘Extras->Settings’*-menu.
IS-HIGH-LOW-BYTE-ORDER	This property returns *true*, if the byte order is big-endian; otherwise it is little-endian.
RBS State	The RBSs State corresponds to the Application State.
Revision	This property displays the device type and revision, separated by a hyphen.
Runtime License Analyzing	If the **Analyzing** license (only for FlexConfig RBS) is activated, it returns *true*; otherwise it returns *false*.
Runtime License FlexConfig Control	If the **FlexConfig Control** license (only for FlexConfig RBS) is activated, it returns *true*; otherwise it returns *false*.
Runtime License FlexConfig Gateway	If the **FlexConfig Gateway** license (only for FlexConfig RBS) is activated, it returns *true*; otherwise it returns *false*.
Runtime License FlexConfig RBS	If the **FlexConfig RBS** license (only for FlexConfig RBS) is activated, it returns *true*; otherwise it returns *false*.
FL3X Config Standard License	Following values are possible: **False**: The device has no FL3X Config Standard license **Lifetime**: The device has a lifetime FL3X Config Standard license **Date** in format dd/mm/yyyy: The device has a FL3X Config Standard license, which expires by given date.
FL3X Config Extended License	Following values are possible: **False**: The device has no FL3X Config Extended license **Lifetime**: The device has a lifetime FL3X Config Extended license **Date** in format dd/mm/yyyy: The device has a FL3X Config Extended license, which expires by given date.
Serial Number	This property contains the serial number of the device.

Key (Property)	Description
Skipped Frames [Channel *bus-type.x*]	This property returns the number of skipped frames. It only applies to FlexRay and returns '0' for other bus types. A frame is skipped, when a new frame overwrites an older frame, before the older frame could be sent. This mainly occurs with dynamic frames.
Sleep [Channel *x*]	This property displays the settings of the Sleep behavior for the corresponding channel. See chapter 5.2.2 (Settings Tab).
Software Sleep	This property returns *true*, if the Software Sleep feature has been activated. See chapter 5.2.2 (Settings Tab).
Standard Gateway	This property displays the current Standard Gateway of the device.
Subnet Mask	This property displays the current Subnet Mask of the device.
Standard Gateway [*adapter name*]	The Standard Gateway of the specified adapter is displayed by this property. The Standard Gateway can be set in the "Settings"-tab or in the *'Extras->Settings'*-menu.
Subnet Mask [*adapter name*]	The Subnet Mask of the specified adapter is displayed by this property. The Subnet Mask can be set in the "Settings"-tab or in the *'Extras->Settings'*-menu.
Termination [Channel *x*]	This property provides the settings of the Termination for channel *x*. See chapter 5.2.2 (Settings Tab).
Timestamp	This property contains the timestamp of the instant when the project was built.
Version String	This property contains a user edited string, appropriate to the project, which has been flashed to the selected device.
Wakeup [Channel *x*]	This property displays the settings of the Wake-Up behavior for the corresponding channel. See chapter 5.2.2 (Settings Tab).

Table 29: Description of the keys in the "Information" tab

5.2.2 Settings Tab

The "Settings"-tab of the "Device Manager" shows different properties of the currently selected device. Especially all values of the properties are shown which are editable. Which properties are editable depends on the selected device.

The "Settings"-tab is shown in Figure 11 and the description is given in Table 30.

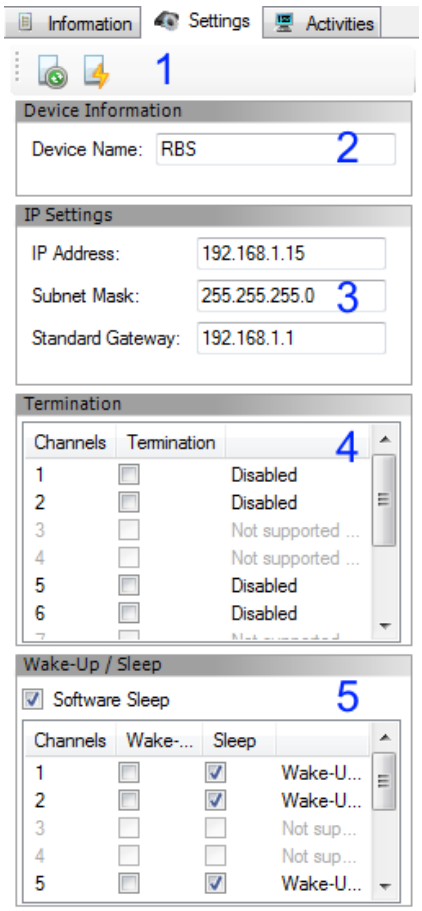


Figure 11: “Device Manager” section – “Settings” tab

No	Description
1	The shown buttons in consecutive order are “*Refresh*” and “*Write Properties*”. • Pressing the “*Refresh*”-button refreshes the information of the selected device. • Pressing the “*Write Properties*”-button writes the edited settings of the selected device to the device.
2	This section contains the editable name of the device.
3	This section shows the editable IP settings.
4	In this section, the termination values for the different busses of the current device are shown. Depending on the device type these values are editable. • This section is editable for the FL3X Device-S and FL3X Device-L/L². Here the termination can be set via software. The following options are available: ○ “Enabled”: If this option is used or shown, the connector for this channel is terminated with the line resistance. ○ “Disabled”: If this option is used or shown, the connector for this channel is high resistance. ○ “Not supported at this channel”: For the appropriate channel, the setting of the termination is not supported.

No	Description
5	Shows the Wake-Up and Sleep settings for the different busses of the current device. Depending on the device type these values are editable. • This section is editable for the FL3X Device-S, FL3X Device-L. Here the Wake-Up and Sleep behavior can be set per software. The checkbox for the “Software Sleep” has the function to enable/disable sleep-function calls within the user functions. Independent of this the Wake-Up and Sleep behavior for each channel can be set with additional checkboxes. The following options are available: ○ “Enabled”: The property (Wake-Up, Sleep) is enabled. ○ “Disabled”: The property is disabled. ○ “Reading state failed”: The reading of the settings (Wake-Up, Sleep) has not been possible. E.g., due to not existent/defect tiny. ○ “The state is not known”: If the state of the settings (Wake-Up, Sleep) has not yet been proved. ○ “Not supported at this channel”: For the appropriate channel the setting of the property is not supported.

Table 30: Description of the “Settings” tab

NOTICE

Activating the Wake-Up and/or Sleep option for unconfigured communication channels or inconsistently configured hardware channels (e.g switch between FPGA image and used FL3X Tiny) can result in an undefined behavior of the device.

5.2.3 Activities Tab

The “Activities”- tab of the “Device Manager” shows different license dependent possibilities of the currently selected device (see Figure 12). The different options can be used to start “*FL3X Config Control*”, to “*Build & Download*” or to download the current project.

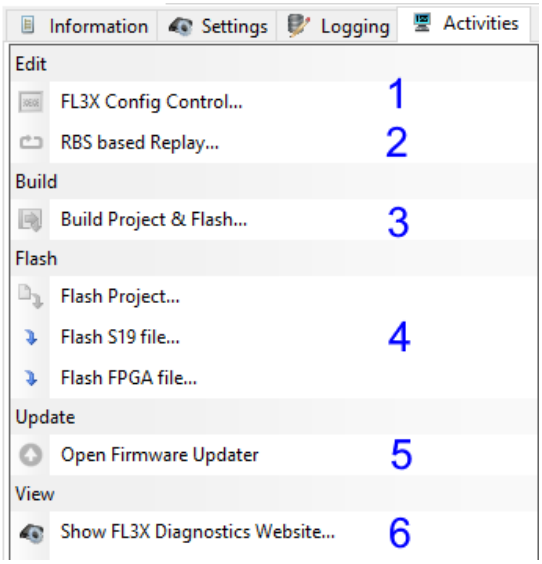


Figure 12: “Device Manager” section – “Activities” tab

The different possible activities are given and described in Table 31.

No	Description
1	This option can be used to start the “FL3X Config Control”-Editor. See chapter 7 FL3X Config Control
2	This option can be used to start the “RBS based Replay”-Editor. See chapter 5.13 RBS Based Replay
3	The ‘*Build Project & Flash…’* option can be used to build the actual project and download it to the selected device. This option is only enabled when a device is connected and selected. See also chapter 5.2.5 Flashing
4	• The ’*Flash Project’* option can be used to download the current project to the currently selected device. See also chapter 5.2.5.1 Flash Project. • The *‘Flash S19 file…’* option can be used to download an S19 file to the selected device. See also chapter 5.2.5.2 Flash S19 File • The *‘Flash FPGA file…’* option can be used to download an FPGA file to the selected device. See also chapter 5.2.5.3 Firmware Update – Flashing FPGA
5	This option can be used to open show the firmware update window. See chapter 5.2.7 Firmware Updater
6	This option opens the FL3X Diagnostics Website (see chapter 8) (via web browser), which shows several information about the selected device. E.g., the version of FL3X Config, which was used to compile the project, is shown.

Table 31: Description of the activities in the “Activities” tab

	NOTICE
	If the ‘Device Manager’ does not contain any device, the actions for the “selected device” are not available.

5.2.4 Logging Tab

The “Logging”-tab of the “Device Manager” shows the logging status of the selected device (see Figure 13).

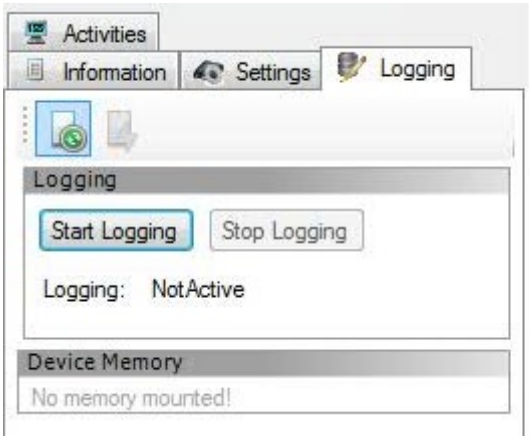


Figure 13: “Device Manager” section – “Logging” tab

If the device has any configured logging instances, they can be started and stopped via the buttons “Start Logging” / “Stop Logging”. The current memory consumption of the device is also shown.

For more information about data logging, see chapter 5.8.6.

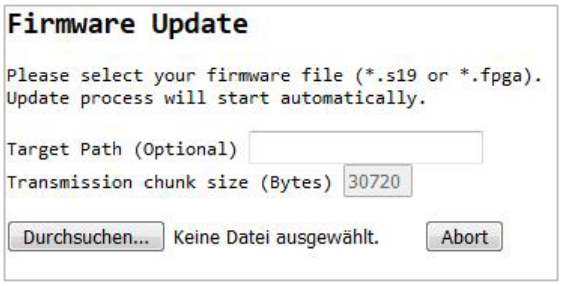
5.2.5 Flashing

Flashing a device performs the following steps:

1. Connect the device (check if it is available)
2. Flash the FL3X Config project (S19 file)
3. Restart the device to apply the flashed project
4. Wait for the device and then reconnect
5. Check the device FPGA firmware

6. Flash the FPGA firmware (if necessary)
7. Restart the device to apply the flashed FPGA firmware
8. Wait for the device and then reconnect
9. Update the device real time clock (RTC) with the current PC time

	NOTICE
	Flashing via Wi-Fi is generally supported. The default Wi-Fi IP-Address of the device is “198.51.100.50” for the FL3X Device Rev3 devices (e.g. 0A03, 0S03) and “172.30.1.1” for the FL3X Device Rev2 devices (e.g. 0A02, 0S02). For some devices (FL3X Device-L (0A02), FL3X Device-L² (0S02)) there is a reset of the Wi-Fi connection after flashing is finished. This results in an error message (when the Wi-Fi connection is lost/interrupted, the project timestamp cannot be verified).

	Information
	Flashing the device is also possible via the FL3X Diagnostics Website (see chapter 8).  Hint: Flashing the *.s19* via FL3X Diagnostics Website (see chapter 8) does not update the FPGA firmware and does not set the real time clock (RTC).

5.2.5.1 Flash Project

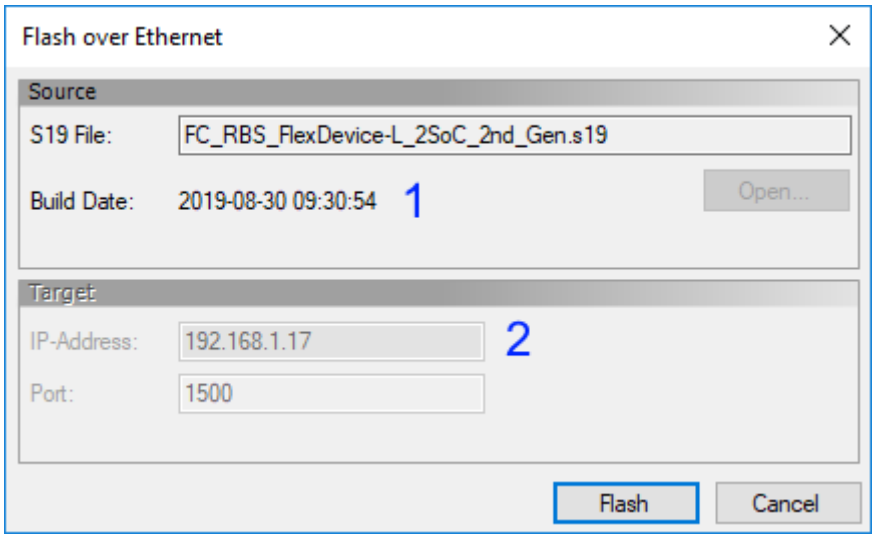


Figure 14: Flashing the current project over Ethernet to the selected device

This option flashes the current compiled project to the selected device (e.g., FL3X Device-S). Please take care about the *‘Build date’* (**1**) (see chapter 5.1) of the selected file to be sure to flash the latest configured project.
The IP-Address of the connected device (e.g., FL3X Device-S) is used (**2**).

5.2.5.2 Flash S19 File

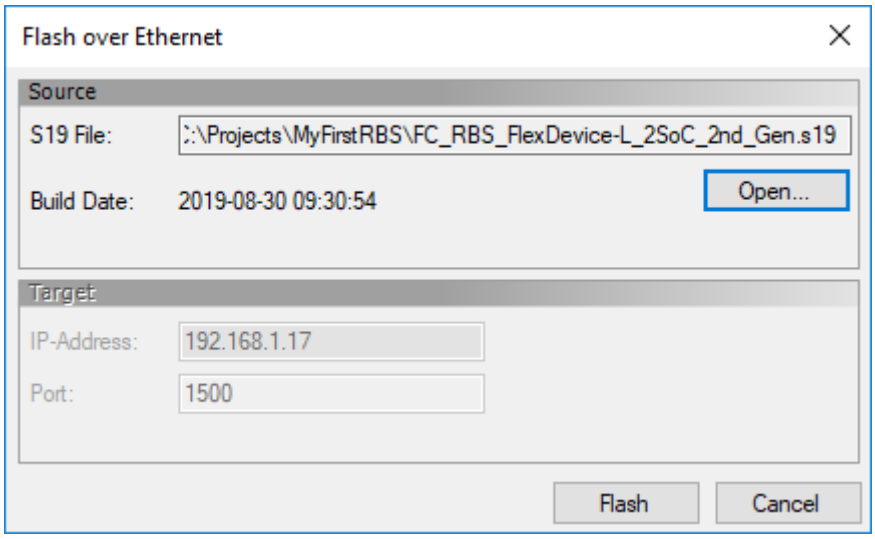


Figure 15: Flashing a .s19 file over Ethernet to the selected device

This option flashes a *.s19* file to the device (e.g., FL3X Device-S). Select the *.s19* file via the [Open…] button.
The IP-Address of the connected device (e.g., FL3X Device-S) is used.

5.2.5.3 Firmware Update – Flashing FPGA

After a project is downloaded to the device, there is an automatic check if an update of the device firmware is required. In this case, the update is performed automatically. The update process takes about 3 minutes for the FL3X Device-S and FL3X Device-L/L².

The update process will check the build-in resources and the **.fpga* file in the project folder as source for the required firmware. The **.fpga* file will be copied into the project folder after a successful project build.

	NOTICE
	Do NOT disconnect the power supply of the device while a firmware update is in progress!

	NOTICE
	FL3X Device-PXIe / FL3X Device-PCIe After flashing the FPGA, the change does not immediately take effect. You can continue to use the FL3X Device-PXIe / FL3X Device-PCIe. The change of the FPGA image takes effect when the FL3X Device is unpowered. To accomplish this, shut down the PC. A restart is not sufficient!

5.2.6 Hardware Licenses

The ‘Hardware Licenses’ section shows the licenses for each device in the device list.

	Information
	For more information about the hardware licenses see chapter 3.2 Runtime Licenses (FL3X Device, FL3X System, Virtual FL3X Device).

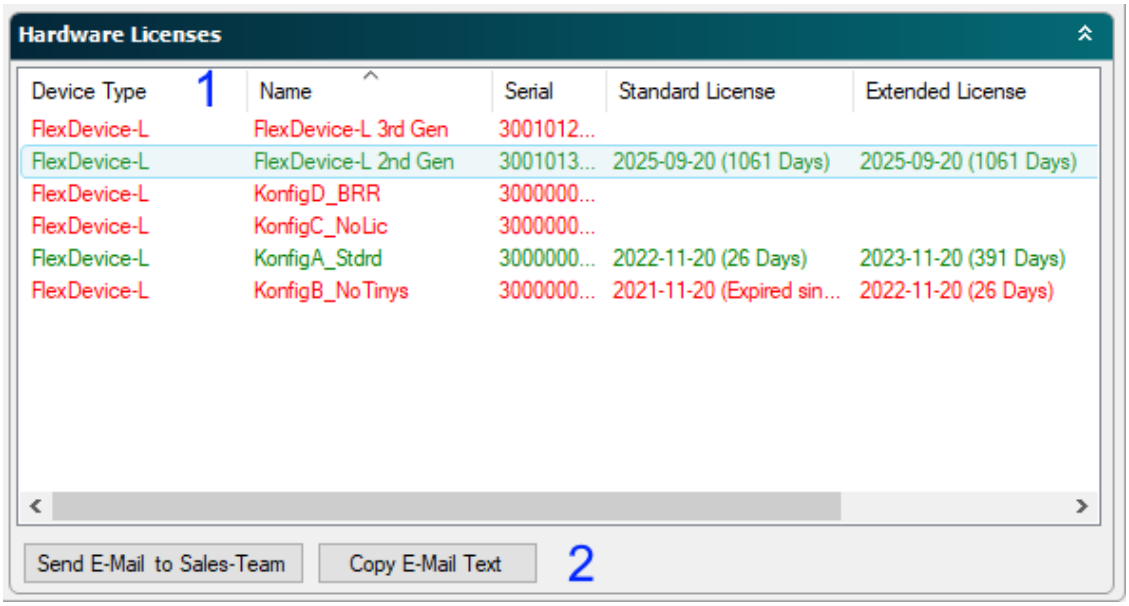


Figure 16: Device Manager – section “Hardware Licenses”

No	Description
1	There are several columns. For more information about each colum, see Table 29: • *Device Type* • *Name* (device name) • *Serial* (serial number) • *Standard License* • *Extended License* • *Version String* • *IP Address* The text color is based on the runtime license expire date: Green: The runtime license is available and does not expire in the next 180 days. Orange: The runtime license is available and expires in 90-180 days. Red: The runtime license does not exist or is either expired or expires in 90 days. If a device has both runtime licenses, then the ‘FL3X Config Extended’ license will be used to set the text color.
2	The buttons are enabled if at least one device is selected. These buttons provide an easy way to collect all information about the licenses. *Send E-Mail to Sales-Team*: Opens the default E-Mail application with a predefined text and information about the expiring licenses. *Copy E-Mail Text*: Copies the text to clipboard.

5.2.6.1 Runtime License Expire Dialog

The following dialog can be shown if the runtime license expires soon. For more Information, please see the chapter 5.22.1.6.

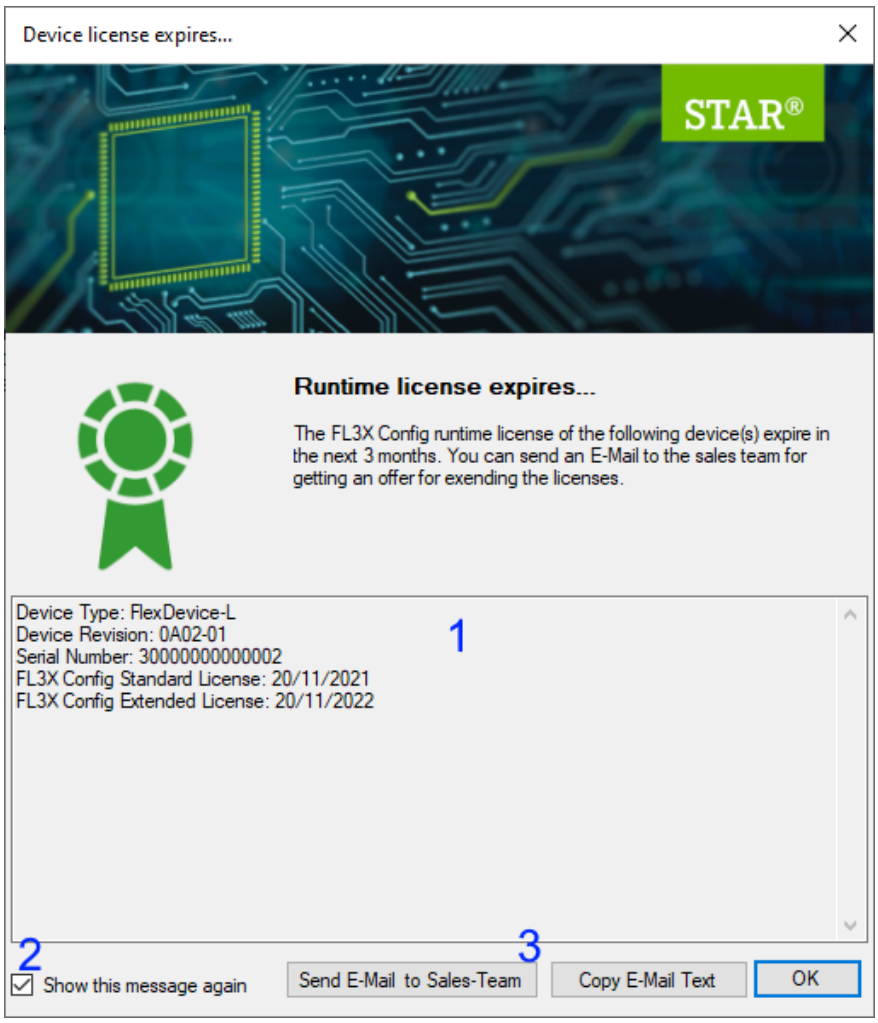


Figure 17: Runtime license expire dialog

No	Description
1	Information about the device(s) whose runtime expire soon.
2	Deactivate this option if you do not want to see this dialog again. This will also deactivate the corresponding setting in the global settings dialog.
3	*Send E-Mail to Sales-Team*: Opens the default E-Mail application with a predefined text and information about the expiring licenses. *Copy E-Mail Text*: Copies the text to the clipboard.

5.2.7 Firmware Updater

If a firmware update is available for a connected device there is a green arrow shown for it in the device manager (see Figure 18). Click "*Open Firmware Updater*" in the context menu for the device to open the "Firmware Updater" dialog. When the "Firmare Updater" dialog is opened via FL3X Config, the IP address and port of the device will be automatically used and the Firmware Updater tries to connect to the device.

	NOTICE
	For FL3X System-M/XS/XS² devices, the Firmware Updater only detects available updates and provides a link to the website where the update can be downloaded. For these devices, users need to manually perform the updates.

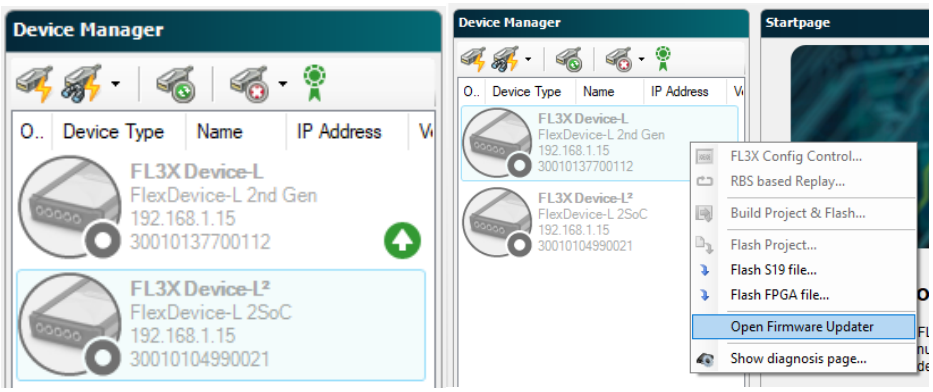


Figure 18: Device Manager - Firmware Update available

5.2.7.1 Firmware Updater Dialog

The Firmware Update dialog (see Figure 19) allows the user to connect to the device. If the connection to a device was successful, the current firmware versions of the device are shown. If there is any update available, the version of each update is shown on the right side of the dialog (1). To start the update, click the [Update Firmware] button (4).

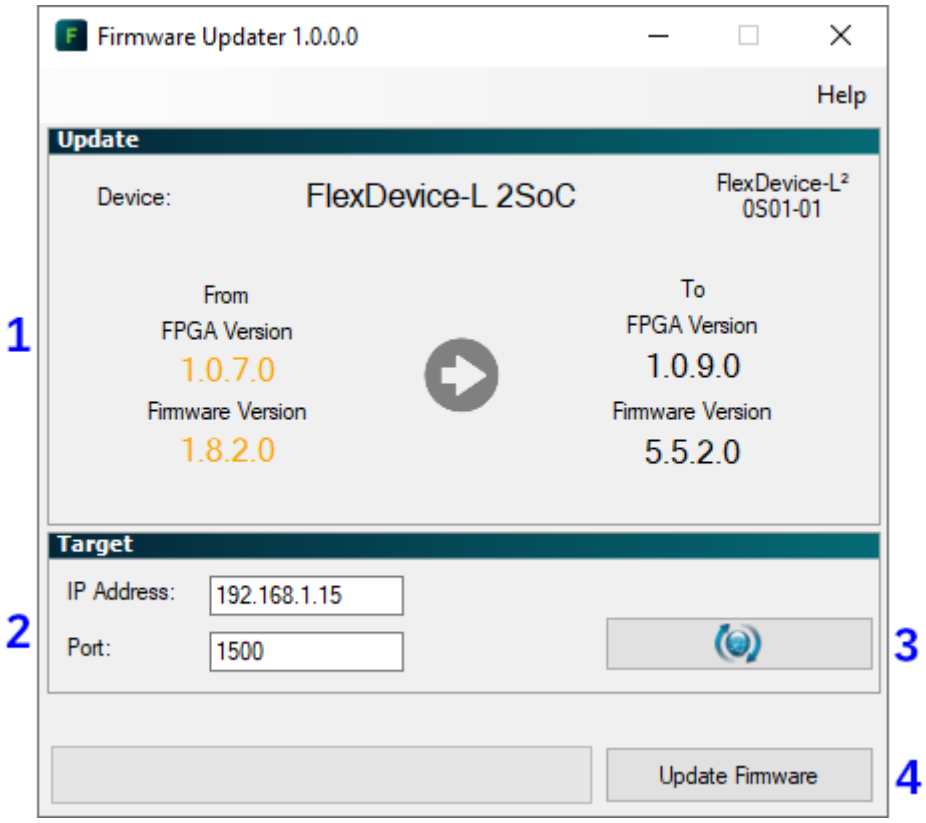


Figure 19: Firmware Update Dialog

No	Description
1	The “Update” section shows - the device name, device type and device revision on the top. - a list of available updates on the bottom (current version on the left, new version on the right).
2	IP address and port of a connected device. If the Firmware Updater was opened via FL3X Config, both fields are not editable.
3	This button is used to connect/reconnect to the device.
4	Starts the update. Any running update is shown by the progress bar on the left.

5.2.7.2 Firmware Update - Available Updates

The “Available Updates” dialog shows all the updates available for the different Devices/Revisions. It can be opened via “*Help*” -> “*Available Updates*”.

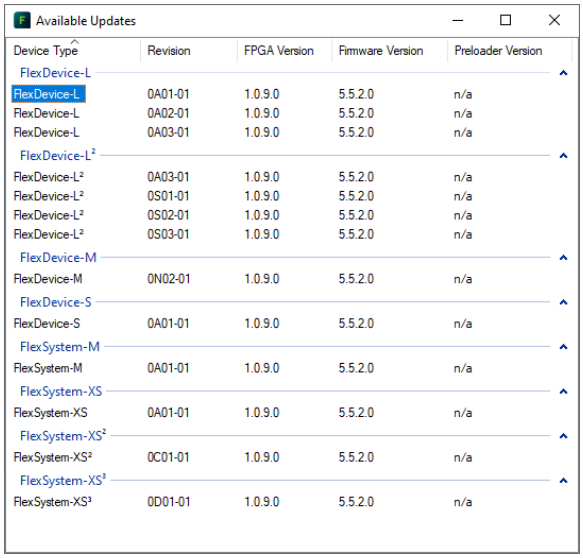


Figure 20: Firmware Update - Available Updates

5.3 Output

An “Output”-section of the main window is exemplarily shown in Figure 21.

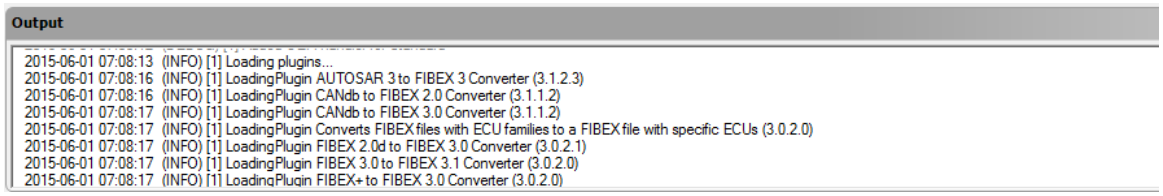


Figure 21: FL3X Config Main Window – “Output”-section

The “Output”-section displays relevant process steps during project configuration/generation. Additionally, in some cases the process state of a process is displayed.

Some of the things which are displayed are

1. Loading/Saving a project.
2. Loading of relevant Converter Plugins (e. g. CANdbc to FIBEX 4, AUTOSAR4 to FIBEX 4, ...).
3. Loading of OEM specific handlers.
4. Loading of files during bus configuration.
5. The build process steps (e.g., pre-processing).
6. Errors and warnings during the compilation process step.

It is possible to mark the generated output of the “Output”-section and copy it to an editor.

5.4 The Menu Bar

The menu bar is located at the top of the FL3X Config main window.



Figure 22: Menu bar

5.4.1 File Menu

The *‘File’* menu handles file operations for the project like *‘New Project...’*, *‘Open Project...’*, or *‘Save Project...’*.

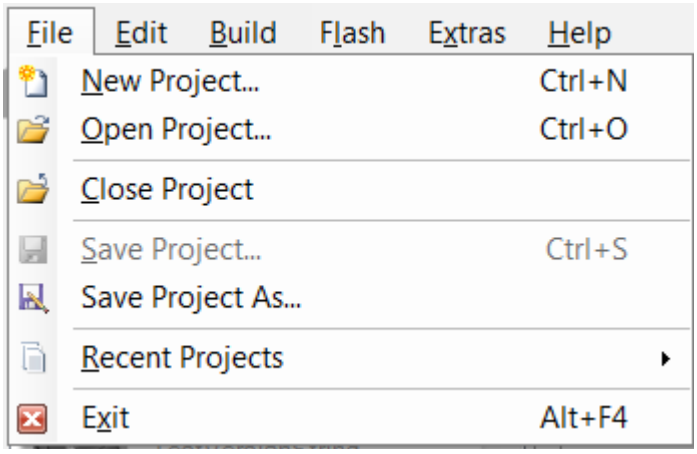


Figure 23: The *'File'* menu

- *'New Project...'*: opens a dialog to create a new empty project (see chapter 5.4.1.1).
- *'Open Project...'*: Opens a dialog to open an existing project.

NOTICE
All projects created with FlexConfig RBS version 4.3.2.0 or later can be loaded. All projects created with FL3X config version 1.0 or later can be loaded. All projects created with FL3X Config RBS version 4.3.2.0 or later or with FL3X Config 1.0 or later are converted to FIBEX 4.1.1 when they are loaded.

Information
The configuration file of FL3X Config Analyzer (Android) has the extension '*.fca*', which can also be opened by FL3X Config.

- *'Close Project'*: Closes the current project.
- *'Save Project...'*: Saves the current project. If there are unsaved changes, the project is marked with an "*" (see Figure 22).
- *'Save Project As...'*: Opens a "Select folder" dialog create a new project folder where the current project should be saved to.
- *'Recent Projects'*: Shows a list of recently opened projects.
- *'Exit'*: Closes the application.

5.4.1.1 New Project Dialog

If the '*New Project...*' menu item has been chosen, the following dialog is shown.

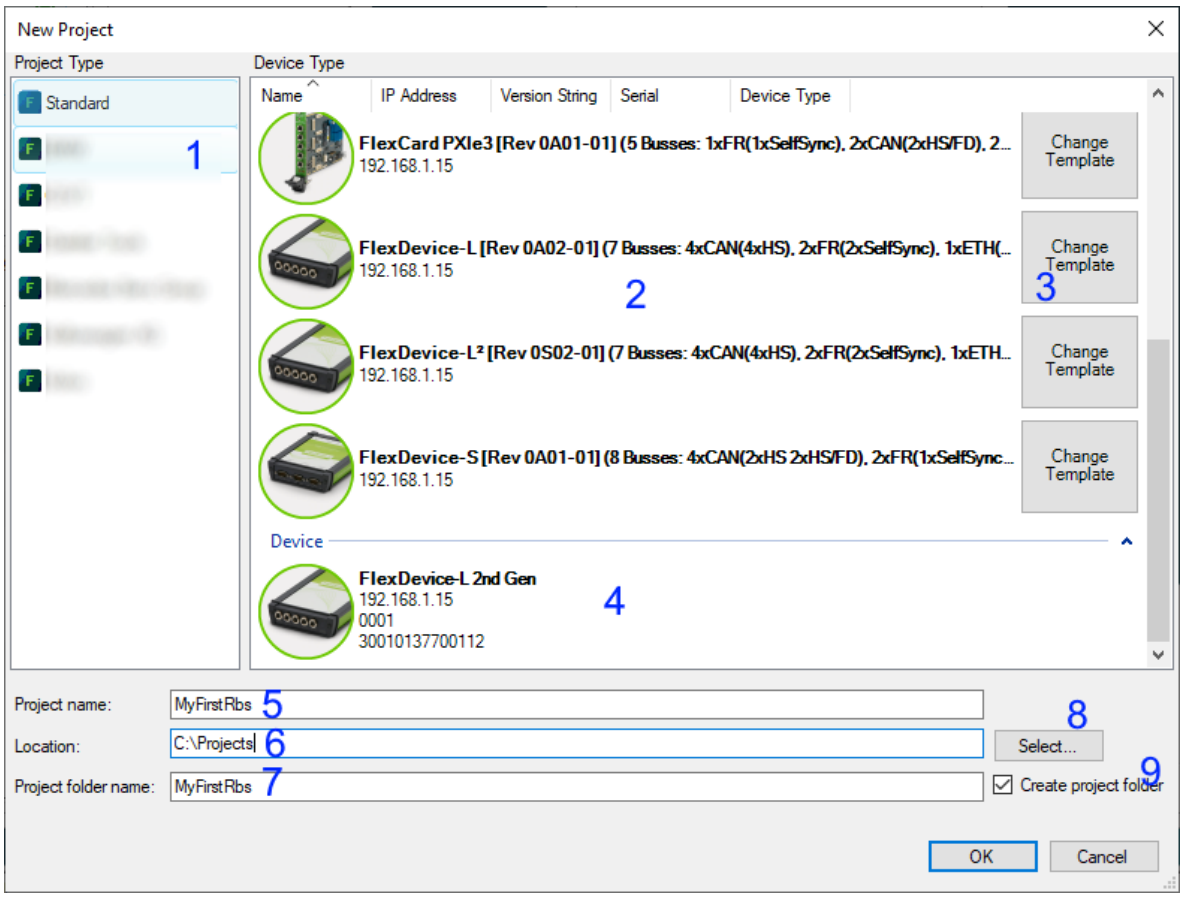


Figure 24: The “New Project”-dialog

The elements, of this dialog, are described in Table 32.

No	Description
1	Here the required project type can be selected. If no OEM package is installed, only the project type ‘Standard’ is available. With each OEM package installed, new project types will appear.
2	In the “Template” section the device type (FL3X Device-S, FL3X Device-L/L² etc.) and an appropriate configuration (see item 3) can be selected for the new project. In this section, all devices and configurations are available, which are supported by Star-Electronics. It is important to keep in mind that the chosen device and configuration must later fit to the real existing hardware! Another way to select the device type and configuration depending on the real existing hardware directly is given in (4).
3	Using the “*Change Template*” button makes it possible to choose an appropriate device configuration for the selected template device. From scratch, all possible configurations are available. It is important to keep in mind that the real, available hardware will eventually restrict the selection of the bus configuration! See chapter 2.8.1 (FL3X Device-S), chapter 2.8.2.1 (FL3X Device-L) or chapter 2.8.3.1 (FL3X Device-L²) for more information about which bus configuration is available.
4	In the “Device” section a real existing device, which is connected via Ethernet to the PC, can be selected for the new Project. If no device is connected this section is empty. If a project is created for such a connected device, the current active FPGA-Image of the device will be used.
5	The name of the project (and the project file **.fcr*) can be edited here.
6	The location or respectively the path of the project can be edited here. Each project must be stored in an exclusive (empty) folder.
7	A separate project folder can be edited here. Each project must be stored in an exclusive (empty) folder. By default, the project folder is named as the project.
8	Clicking the “*Select...*” button opens a folder browser to select the location.

No	Description
9	With this checkbox, the creation of a separate folder (see item 7) within the location folder can be enabled/disabled. If the location folder shall be used directly this check box must be unmarked and the location folder must be empty.

Table 32: Description of the “New Project” dialog

NOTICE

The newly generated project-folder should not be edited manually! Editing within the project-folder can lead to unpredictable behavior!

5.4.2 Edit Menu

The *‘Edit’* menu contains different menu items, to configure the actual project (s. the following subsections).

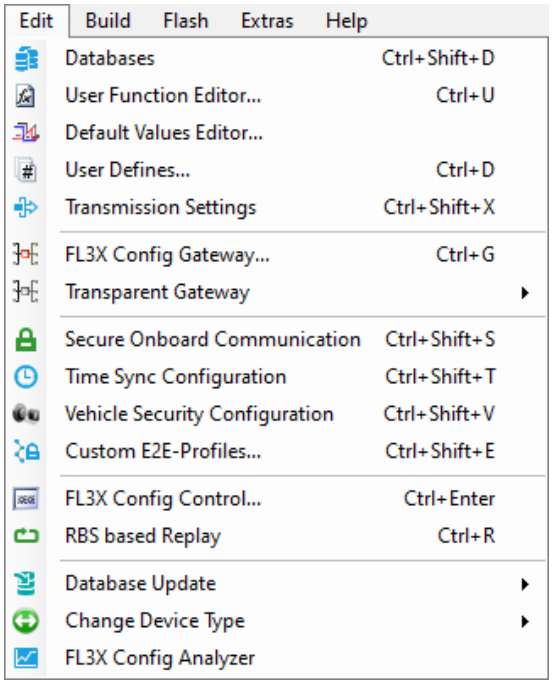


Figure 25: The *‘Edit’* menu

5.4.2.1 Databases

Use this menu entry to open the Database Manager. The Database Manager gives an overview over all databases which are included in the project. Use the Database Manager to add databases to the project or to remove databases from the project. Details about the Database Manager can be found in chapter 5.5.

5.4.2.2 User Functions

A User Function can be used to enter user specific code. User Functions are described in more detail in chapter 5.8.

5.4.2.3 Default Values

Default Values are used to predefine values for transmitted signals. They are described in more detail in chapter 5.9.

5.4.2.4 User Defines

User Defines are C-code defines for the code which is running on the device (FL3X Device-S, FL3X Device-L/L²). The user can add, remove, enable, and disable defines and change the optional define value. These defines will be included to the compilation. Defines can be used to enable special functionality within the hardware. They are described in more detail in chapter 5.6.7.8.

5.4.2.5 Transmission Settings

Use this menu entry to open the Transmission Settings dialog. The Transmission Settings dialog shows the relevant information of transmitted elements for all configured busses. Details about the available settings can be found in chapter 5.6.7.

5.4.2.6 FL3X Config Gateway

Use this menu entry to open the FL3X Config Gateway dialog for adding the gateway routings. Creating a gateway is described in more detail in chapter 6.1.

5.4.2.7 Transparent Gateway

This menu entry has the sub menu entry “Create new…” to create a new Transparent Gateway. More information about a Transparent Gateway can be found in chapter 6.2.
It is possible to create multiple Transparent Gateways. Use the “*Edit …*” menu items to edit the Transparent Gateways.

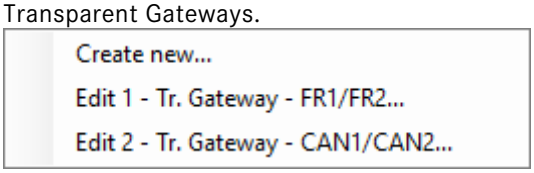


Figure 26: The sub menu entries for Transparent Gateways

5.4.2.8 Secure Onboard Communication

Use this menu entry to open the Secure Onboard Communication dialog. The dialog is described in more detail in chapter 5.10.

5.4.2.9 Time Sync Configuration

Use this menu entry to open the Time Sync Configuration dialog. The dialog is described in more detail in chapter 5.10.

5.4.2.10 Vehicle Security Configuration

Use this menu entry to open the Vehicle Security Configuration dialog. This dialog depends on the used OEM package. Please read the additional information provided for each OEM package.

5.4.2.11 Custom E2E-Profiles

Use this menu entry to open the Custom E2E-Profiles dialog. See also chapter 5.12.

5.4.2.12 FL3X Config Control

Use menu entry *‘FL3X Config Control…’* to configure manipulations and sequences for transmitted entities such as signals, PDUs and frames. Manipulations and sequences are described in more detail in chapter 6.2.4.

5.4.2.13 RBS Based Replay

Use this menu entry to open the RBS Based Replay dialog. RBS Based Replay can load a mf4 log file and replay its content on configured busses. The RBS Based Replay dialog is described in more detail in chapter 5.13 RBS Based Replay.

5.4.2.14 Database Update

Use menu entry *‘Database Update’* to update the current database. See chapter 5.14 for more details.

5.4.2.15 Change Device Type

To change the device of your project, use the menu entry ‘*Change Device Type*’. See chapter 5.14.3 for more details.

5.4.2.16 Create Partner Project (Beta)

Use menu entry ‘Create Partner Project’ to create an opposite project matching to actual one. See chapter 5.16 for more details.

5.4.2.17 FL3X Config Analyzer

Use this menu entry to open the FL3X Config Analyzer (Windows).

	Reference
	More Information about the FL3X Config Analyzer (Windows) can be found in the document “3-01140A01-D014 FL3X Config Analyzer (Windows) – User Manual” [9]. This document can be found in the installation folder of FL3X Config (or via “*Windows*” -> “*Startmenu*” -> “*FL3X Config %VERSIO%*” -> “*FL3X Config Analyzer Windows - User Manual*”)

5.4.3 Build Menu

The *’Build’* menu provides functions to generate and compile the current project.

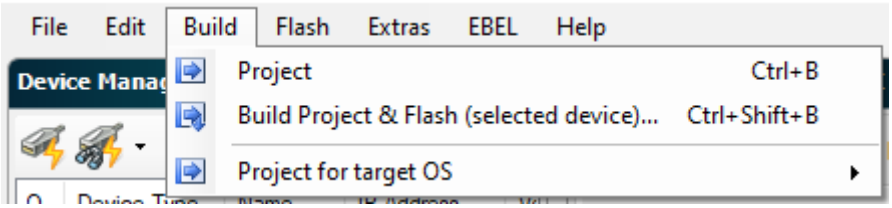


Figure 27: The *‘Build’* menu

	Information
	A connection with the device should be established before using ‘Build Project & Flash (selected device)’ (to get the IP-Address of the target device).

	Information
	Building a project with user functions will take only a short time if the project is already compiled and uncritical changes are made to the user functions. Uncritical changes are e.g. Add/remove/modify user files Change user function code

5.4.3.1 Project

To build the project, the menu item ‘*Project*’ must be used. For code generation and compilation details see the output window of FL3X Config.

	NOTICE
	During the build process, the computer may react with delays to user inputs. This is related to the high processor load.

5.4.3.2 Project & Flash...

Using the menu item ‘*Project & Flash...*’, builds the current project and automatically downloads it to the selected device. This option is only enabled when a device is connected and selected.

See chapter 5.2.5 Flashing for more information.

5.4.3.3 Project for Target OS

Build the project once for the selected target OS.

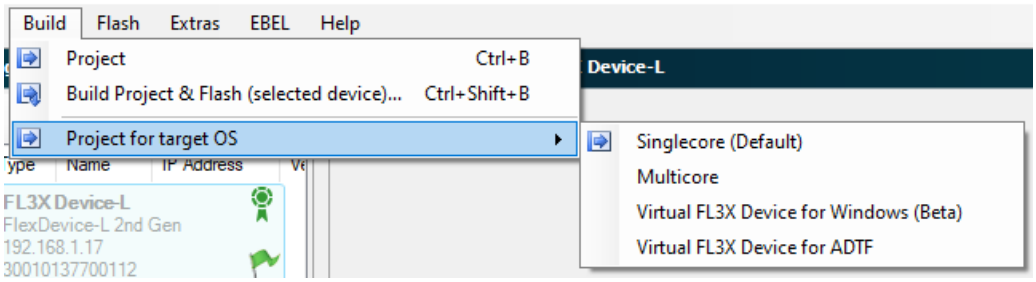


Figure 28: The *'Build – Project for'* menu

The target OS specified for the Project (see chapter 5.22.2.1) is marked with the Build icon .

For more information about the different target OS see chapter 5.22.4.

5.4.4 Flash Menu

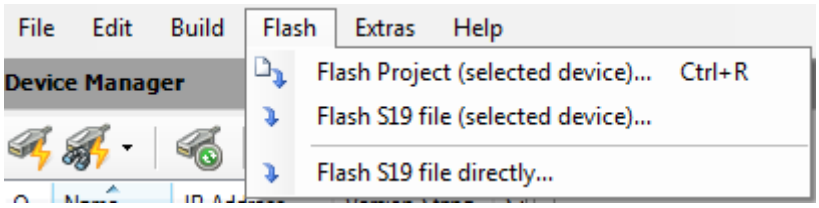


Figure 29: The *'Flash'* menu

The 'Flash' menu is responsible for flashing a generated project (or the *.s19* file of a project) to the device (e.g. FL3X Device-S) via Ethernet.

	NOTICE
	If the 'Device Manager' does not contain any device, the actions for the "selected device" within the menu -> "Flash" are not available.

See chapter 5.2.5 Flashing for more information.

5.4.4.1 Flash Project (Selected Device) ...

The 'Flash Project' option can be used to download the current project to the currently selected device.

See chapter 5.2.5.1 for more information.

5.4.4.2 Flash S19 File (Selected Device) ...

The *'Flash S19 file...'* option can be used to download an S19 file to the selected device.

See chapter 5.2.5.2 for more information.

5.4.4.3 Flash S19 File Directly...

This option flashes any *.s19* file into the device (e.g. FL3X Device-S). Select the file via the "Open..." button, enter the correct IP-Address and click the *"Flash"* button to start the flash operation.

See also chapter 5.2.5.2.

	Information
	The *.s19* is flashed over the TFTP protocol, so it is also possible to use any other TFTP client for downloading binaries.

	NOTICE
	FL3X Config has no chance to check the correctness of the *.s19* file (version of the connected device, firmware with correct Ethernet stack). Ensure to flash always files with correct properties when using this option. This option does not update the device FPGA firmware and does not set the real time clock (RTC) of the device.

5.4.5 Extras Menu

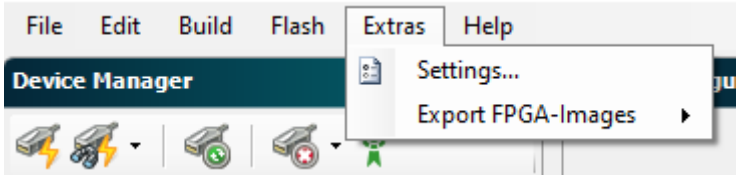


Figure 30: The *'Extras'* menu

5.4.5.1 Settings...

This menu entry opens the settings dialog, where you can configure the settings for the application and the project.

The settings dialog is described in chapter 5.22.

5.4.5.2 Export FPGA-Images

This menu entry allows to export the FPGA-Images for all device configurations (device templates) that are available in the "New Project" dialog. The FPGA-Images are exported as a *.fpga* file. These files can be used to update the FPGA of a device manually. Please contact the *STAR ELECTRONICS* support team for more information.

5.4.6 Help Menu

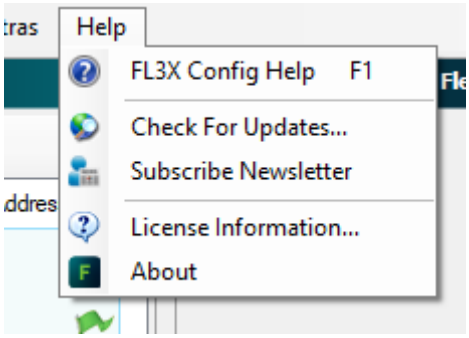


Figure 31: The *'Help'* menu

5.4.6.1 FL3X Config Help

To open the Help-Browser you have the options to select the '*Help -> FL3X Config-Help*' menu item or to press the [F1] key.

5.4.6.2 Check for Updates...

This menu entry starts a search for license and application updates. All results are also shown in the output section of the main window (see chapter 5.2.5).

If no update is found, a message box will be shown with the corresponding information.

If an application update is found, a new dialog will be opened, which shows the update information. You can start the installation of the update via this dialog (see also chapter 4.6.1 Automatic Updates).

The proxy settings (see chapter 5.22.1.4) are used for license and application updates.

5.4.6.3 Subscribe Newsletter

Opens a dialog with information on how to subscribe a newsletter.

5.4.6.4 License Information

This menu entry opens a dialog which provides information about the maintenance and the currently enabled software features of FL3X Config.

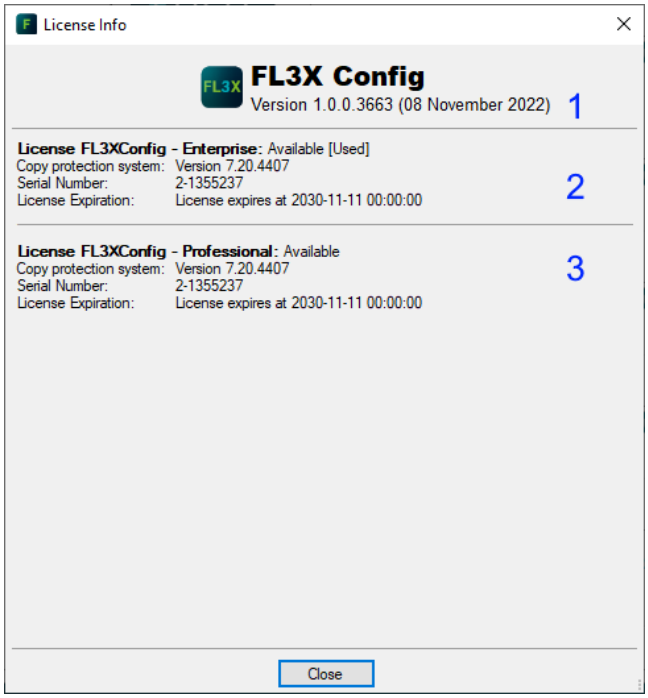


Figure 32: License Information Dialog

No	Description
1	Shows information about the application version and the release date of the application.
2	Shows the FL3X Config Enterprise license when available.
3	Shows the FL3X Config Professional license when available.

Table 33: Description of the “License Information” dialog

For more information about the software licenses, see chapter 3.1.2.

5.4.6.5 About

The shown dialog provides information about the current software version and the contact details of *STAR ELECTRONICS*. It also inherits copyright information and the license agreement.

5.5 Database Manager

The Database Manager gives an overview over all databases which are included in the project. Databases can be added or removed via the Database Manager. To open the Database Manager, click on menu-bar -> *‘Edit -> Databases’*.

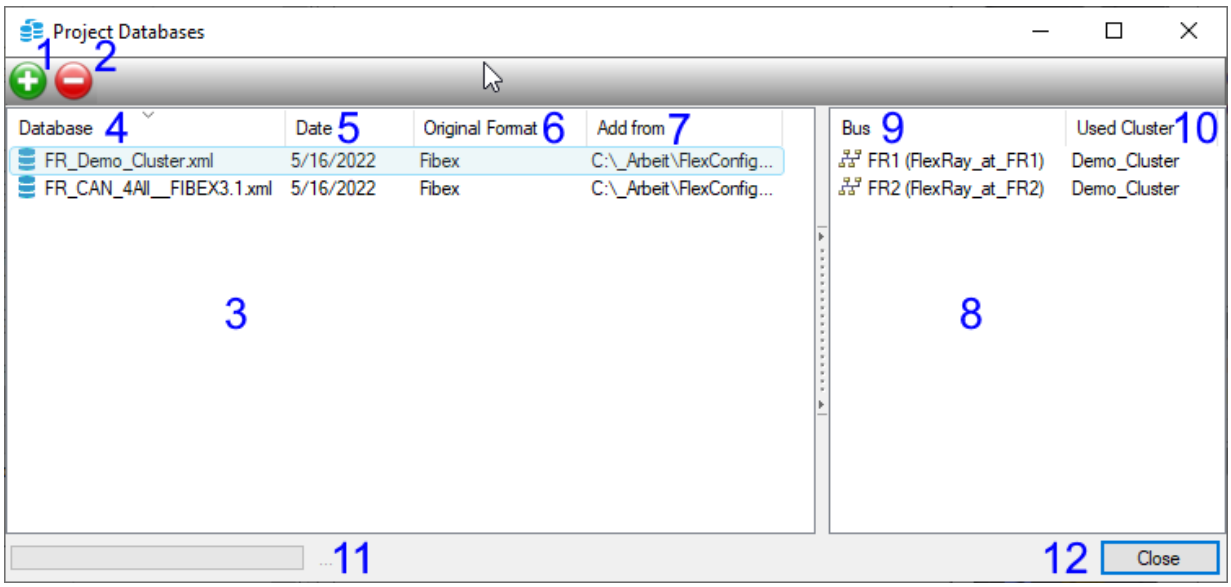


Figure 33: Database Manager dialog

No	Description
1	Add button to add a database to the project. The file open file dialog will appear to select, load and add a database from the file system to the project. See chapter 2.7 for supported file formats.
2	Remove button to remove a database from the project. Hint: Only Databases which are not assigned to any bus can be removed.
3	List of databases added to the project. These Databases can be used for the bus configuration.
4	The file name of the database.
5	The date when the database was added to the project.
6	The original format of the added database (e.g. AUTOSAR, CANdb, FIBEX, LDF).
7	The location (file path) the original database was added from.
8	More information about the selected database.
9	List of RBS busses, which use the selected database.
10	The used cluster of the selected database.
11	Progress bar (used when a database is loaded).
12	Use the **[Close]** button to close the Database Manager **Hint:** All changes in the Database Manager will affect the project directly. It is not possible to cancel an operation (add/remove a database). The only way of reverting such an operation is to **not** save the project (but all other changes of the project will be lost too).

Table 34: Description of the “Database Manager” dialog

Each database can only be added **once** to the project. An added database can be used for multiple RBS/Gateway configurations.

5.5.1 Database Picker

The “Database Picker” dialog is shown every time when a database must be selected. The “Database Picker” dialog provides the same functionality as the “Database Manager” (see chapter 5.5) with the difference, that this dialog can be closed via **[OK]** (and return to the previous window/dialog with the selected database) or **[Cancel]** (and return to the previous window/dialog **without** selecting a database).

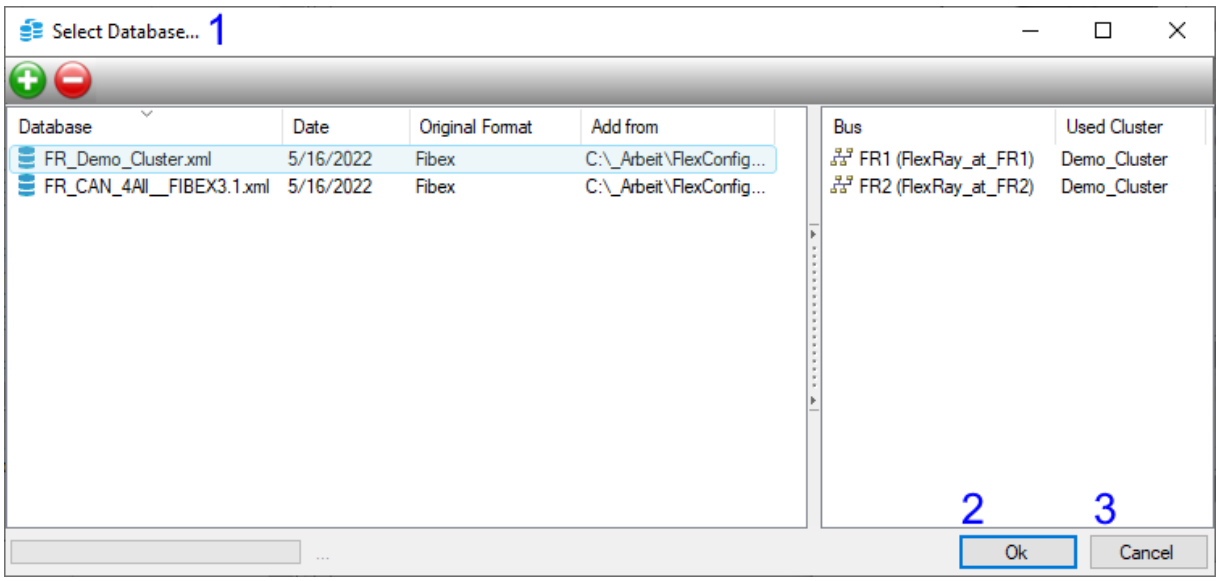


Figure 34: Database Picker dialog

No	Description
1	The Database Picker Dialog to select a database.
2	The **[OK]** button the use/accept the selected database.
3	The **[Cancel]** button if no database should be used/selected. **Hint: [OK]** and **[Cancel]** only affect the selection of a database. Adding and removing a database will affect the project directly. To cancel this dialog will not revert those changes.

Table 35: Description of the “Database Picker” dialog

Use the “Database Picker” dialog to select any database that was already added to the project. Or add a new database to the project and then select it.

NOTICE

Adding or removing a database via the “Database Picker” dialog affect the project directly. Closing this dialog via **[Cancel]** will not revert those changes.

The only way of reverting such an operation is to **not** save the project (but all other changes of the project will be lost too).

5.6 Device Configuration

The “Device Configuration” section of the main window is shown in Figure 35.

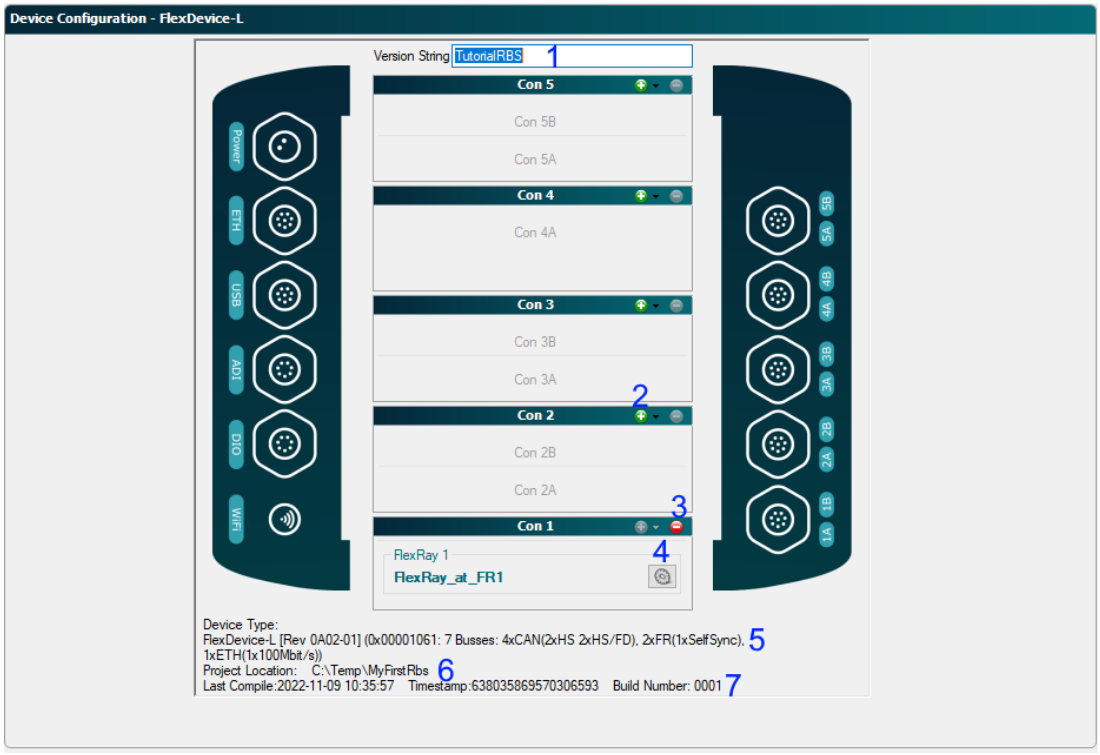


Figure 35: “Device Configuration”-section (here for FL3X Device-L)

The different items are explained in Table 36.

No	Description
1	The **Version String** is a user defined string to identify the actual project-configuration.
2	**Add Button** (green plus) to add a bus to the appropriate connector. Depending on the connector, the bus can be of type CAN, LIN, FlexRay or Ethernet.
3	**Delete Button** (red minus) to remove a bus from the appropriate connector. Deleting a bus usually results in deleting all configurations (e.g. routings, default values) that used the corresponding bus. All active user functions are deleted if they are created for the deleted bus or use parameters of the deleted bus. Inactive user functions will not be deleted. If active ‘user main’ and ‘initialize’ functions use parameters of the deleted bus, they will be deactivated.
4	“Edit bus configuration…” button of the appropriate connector. If this button is pressed, a new dialog opens to configure the bus (s. chapter 5.5).
5	The **Device Type** specifies the type and the busses of the device (e.g., FL3X Device-S or FL3X Device-L²).
6	The **Project Location** specifies the folder of the actual project.
7	The **Last Compile** shows the date and the time of the last compile process. Additionally, a **Timestamp** identifier is shown. The **Build Number** shows the number of times the project has been built.

Table 36: Description of the items of the “Device Configuration” section

Within the “Device Configuration” section it is possible to move a defined bus (e.g., “FlexRay_at_FR1” in Figure 35) from one connector to another connector. This can be done via drag & drop. If the left mouse button is pressed and moved slightly then all alternative connectors are shown with a green frame.

It is possible that already defined connectors are shown with a green frame. In this case the busses can be swapped. Technically this means that the appropriate connectors support both bus types. The drag & drop event in this case will result in a swap of the busses.

The following symbols can appear while moving a bus to another connector:

- Assignment is not possible. Move the mouse pointer over this symbol to see more information.
- Assignment is possible with some limitations. Move the mouse pointer over this symbol to see more information.

If a project is created for a FL3X System-M, the slot configuration must be done before configuring any connectors.

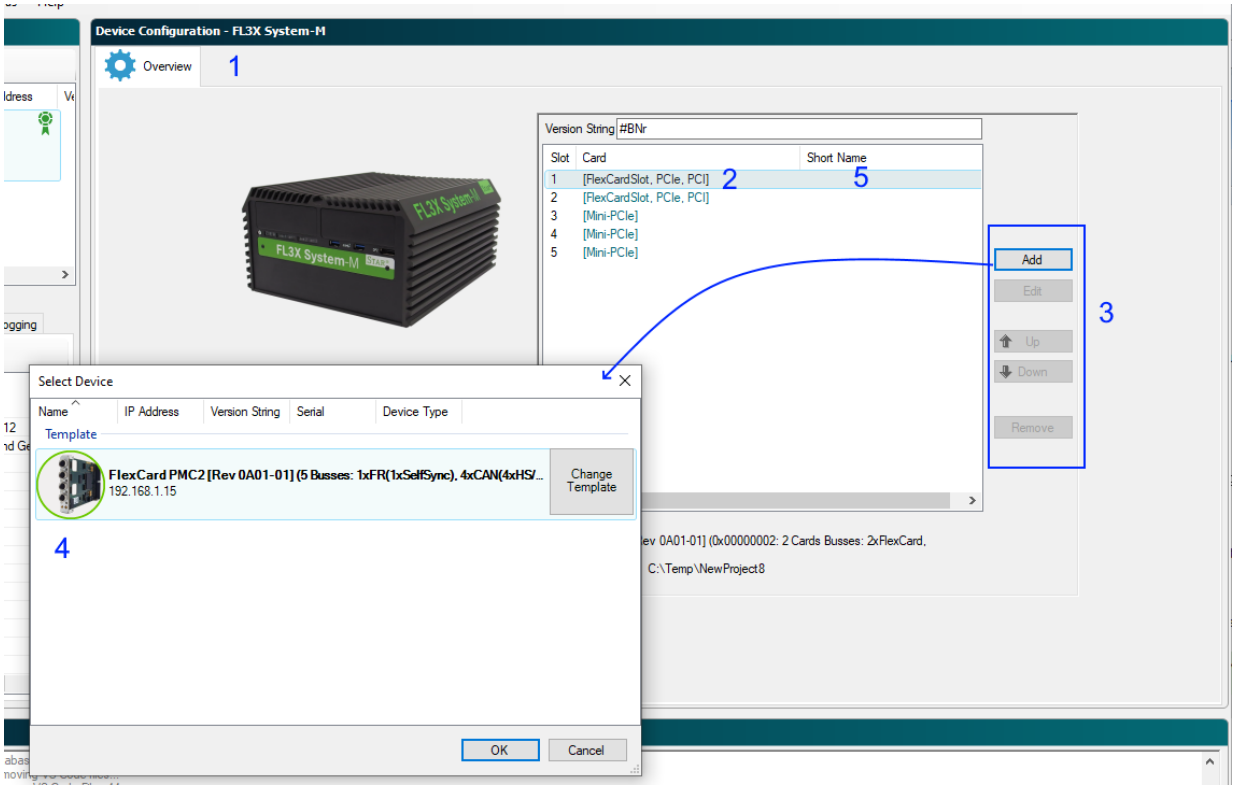


Figure 36: FL3X System-M – Configure slots

No	Description
1	The tab control lists the configuration window (Overview) and each configured slot.
2	Within the ‘Overview’ tab, select a slot to configure it
3	Buttons for configuring slots. **Add** button is activated when an empty slot is selected. Clicking on the **Add** button opens a device selection dialog, where the device can be selected (see 4). **Edit** button opens the tab for the selected slot device. The tab control (1) can be also used for this action. **Up** and **Down** buttons can be used to change the order of the selected slot. **Remove** button deletes the configured device from the slot.
4	The “Select Device” dialog allows to add a device to an empty slot.
5	For configured slots, a short name can be assigned.

Table 37: FL3X System-M – Configure slots

It is also possible to reassign a bus to another slot of FI3X System-M by moving the dragged bus to the slot tab header.

5.6.1 Selecting the Database of a Bus

After the bus type was selected for a connector, the ‘*Edit bus configuration...*’ button (see Figure 35) of the connector can be clicked. If this is done for the first time, different dialogs, depending on the bus type, appear.

The following dialog is shown to select a database (all bus types beside CAN/CAN-FD):

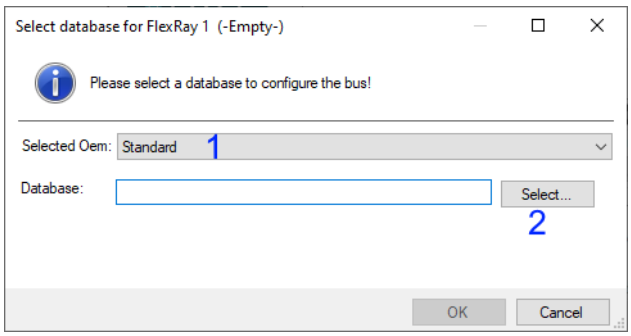


Figure 37: Info dialog for FlexRay/ETH/LIN bus

No	Description
1	This dropdown list is only shown if the Multi OEM requirements are fulfilled. Please see chapter 5.23 for more information.
2	Opens the “Database Picker” (see chapter 5.5.1) to select a database.

If the bus type is CAN, then the dialog shown in Figure 38 is shown.

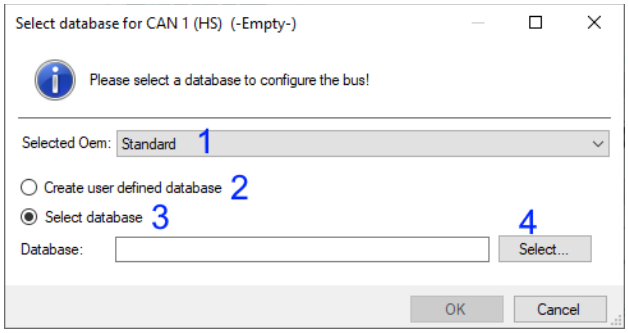


Figure 38: Database selection dialog for CAN bus

No	Description
1	This dropdown list is only shown if the Multi OEM requirements are fulfilled. Please see chapter 5.23 for more information.
2	Select this option to create a custom (user defined) CAN database (see chapter 5.6.3.1.1).
3	Select this option to choose an existing database.
4	Opens the “Database Picker” (see chapter 5.5.1) to select a database.

5.6.2 Base Settings

The “Base Settings” tab of the “Bus Configuration” window is used to create the main settings for the selected bus of the appropriate device. All data is based on the previously selected database file.

This dialog can be opened by clicking the *‘Edit bus configuration…’* button of a connector within the FL3X Config main window. Dependent on the bus type different “Base Settings”-dialogs are available, which are described in more detail in the following chapters.

5.6.2.1 Base Settings CAN/CAN-FD

A “Base Settings” tab for bus type CAN-FD is shown in Figure 39. The dialog for busses of bus type CAN is quite similar. The only difference is that for busses of bus type CAN-FD two bitrates are available and for busses of bus type CAN only one single bitrate is selectable (see **10**).

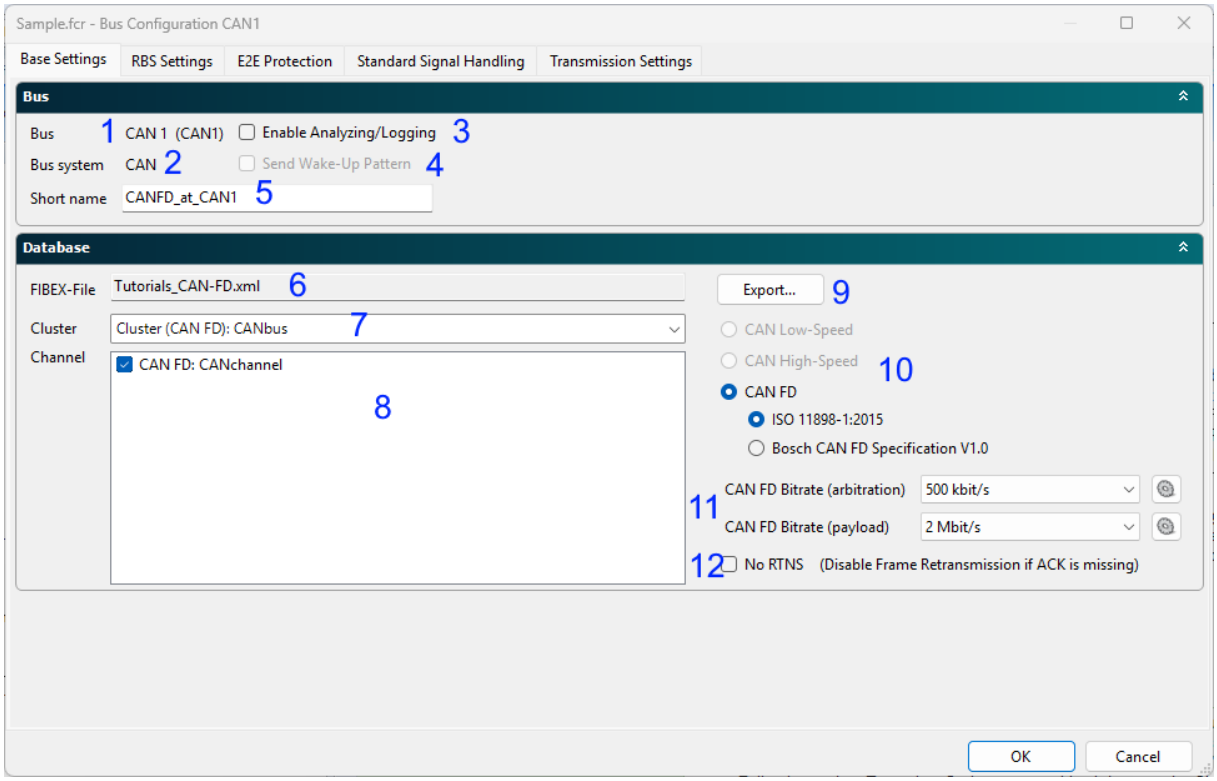


Figure 39: Bus Configuration – “Base Settings” (CAN-FD)

The different elements of this dialog are described in Table 38.

No	Description
1	Name of the current configured bus.
2	Type of the bus (CAN here).
3	FL3X Config Analyzer (both Windows [9] and Android) can access and analyze this bus if this option is enabled. Also, this option must be enabled for data logging.
4	Wake-Up option is supported only by FlexRay and ETH with 10Base-T1S.
5	Short name of the configuration. The short name is displayed in all dialogs. Whitespaces are not allowed. The short name of a bus is also displayed in the FL3X Diagnostics Website (see chapter 8): • Ethernet: The title next to the channel-info • Statistics: In the column 'Name'
6	File name of the used database file. Within a project, the used database format is FIBEX 4.1.1. All other supported database formats are automatically converted.
7	Selected cluster for the chosen connector. Existing but not applicable clusters will be shown in grey font. If just one proper cluster is available and the corresponding setting (see chapter 5.22.1.1) is activated, then this cluster will be automatically selected.
8	Within the "Channel" section of this dialog, channel(s) of the appropriate cluster can be selected.
9	The *'Export'* button can be clicked to export the database as a FIBEX 4.1.1 file. This is especially interesting if a CAN database was created within the FL3X Config application
10	To switch between "CAN low-speed", "CAN high-speed" or "CAN-FD (CAN flexible data rate)" the corresponding option must be selected. For the CAN-FD bus additionally the type of the used standard (ISO/Bosch) must be selected. Depending on the selected device and channel, some of the options may be disabled or hidden.
11	For a CAN-FD bus, two bitrates (arbitration bitrate, payload bitrate) can be selected by drop down boxes. If the bus type is CAN-LS or CAN-HS only one bitrate must be selected. The different bitrates are listed in Table 39.
12	Disable automatic frame retransmission (RTNS). According to the CAN Specification, the CAN CC will perform an automatic retransmission for frames which - have lost arbitration - have not been acknowledged by any other CC - have been disturbed by errors during transmission. By default, this option is enabled.

Table 38: Description of the Bus Configurations dialog – "Base Settings" (CAN/CAN-FD)

The devices FL3X Device-S and FL3X Device-L/L² support CAN-FD and CAN high-speed. Be sure that you have selected the proper project template for one of these devices if you want to configure a CAN-FD bus.

See Table 39 for the supported bit rates.

Supported bit rate	CAN high-speed	CAN-FD arbitration	CAN-FD payload
33,3 Kbit/s	X	X	X
50,0 Kbit/s	X	X	X
62,5 Kbit/s	X	X	X
83,3 Kbit/s	X	X	X
100 Kbit/s	X	X	X
125 Kbit/s	X	X	X
250 Kbit/s	X	X	X
500 Kbit/s	X	X	X
1 Mbit/s	X	X	X
2 Mbit/s	-	-	X

Table 39: Base Settings for CAN – Supported bit rates for CAN

The default value of 1Mbit/s is used in the following cases:

- The channel of the loaded database uses a higher bit rate than supported.
- The channel of the loaded database uses an invalid bit rate.

NOTICE

The sampling rates can be configured for CAN / CAN-FD in Expert Mode.
Please contact the STAR Electronics support team for more information.

NOTICE

FL3X System-M does not support CAN or CAN-FD with 33,3kBit/s.

5.6.2.2 Base Settings FlexRay

A “Base Settings” tab for bus type FlexRay is shown in Figure 40.

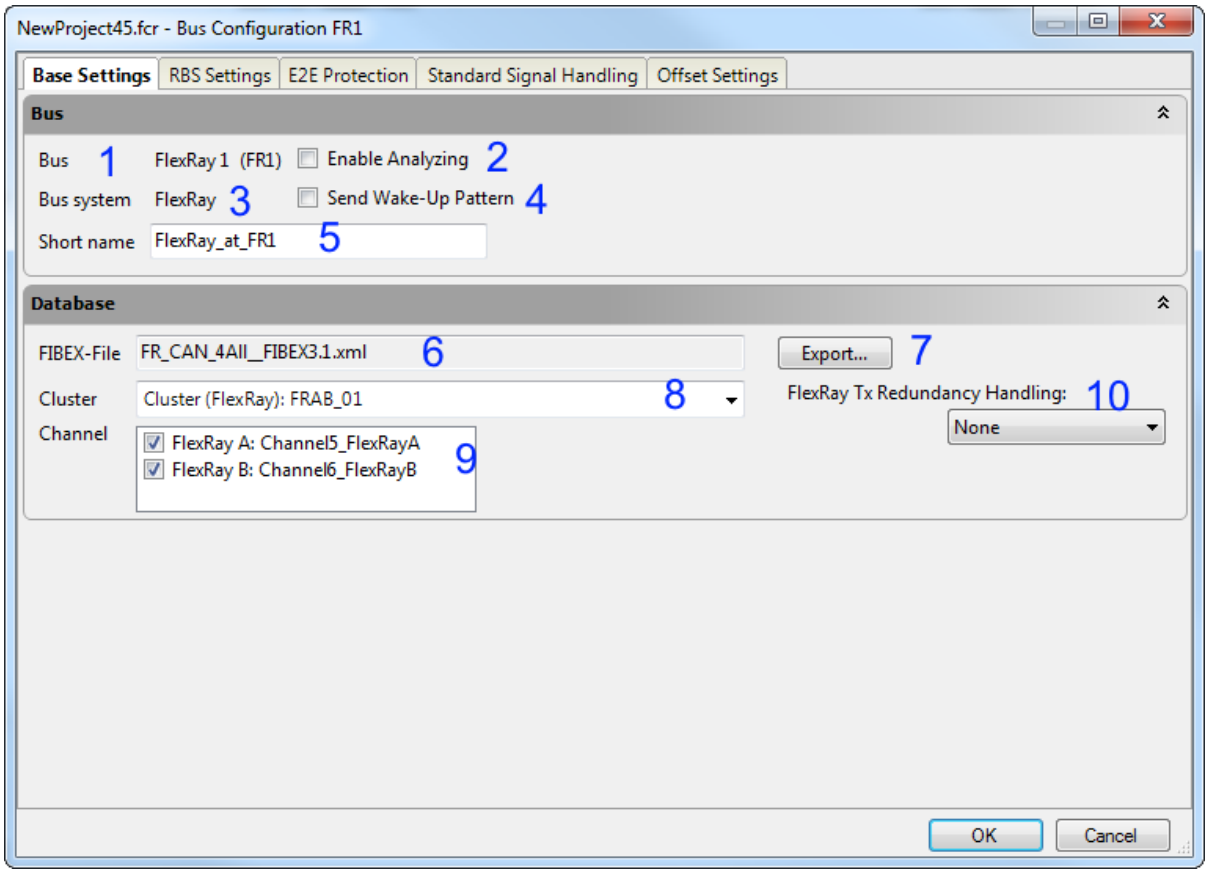


Figure 40: Bus Configuration – “Base Settings” (FlexRay)

The different elements of this dialog are described in Table 40.

No	Description
1	Name of the current configured bus.
2	FL3X Config Analyzer (both Windows [9] and Android) can access and analyze this bus if this option is enabled. Also, this option must be enabled for data logging.
3	Type of the bus (FlexRay here).
4	Send Wakeup Pattern: This option can be enabled to transmit a Wake-Up Pattern on each start of synchronization.
5	Short name of the configuration. The short name is displayed in the Main Window. Whitespaces are not allowed.
6	File name of the used database file. Within a project, the used database format is FIBEX 4.1.1. All other supported database formats are automatically converted.
7	The *‘Export’* button can be clicked to export the database as a FIBEX 4.1.1 file.

No	Description
8	Selected cluster for the chosen connector. Existing but not applicable clusters will be shown in grey font. If just one proper cluster is available and the corresponding setting (see chapter 5.22.1.1) is activated, then this cluster will be automatically selected.
9	Within the “Channel” section of this dialog, channel(s) of the appropriate cluster can be selected. E.g., if there are two FlexRay Channel (A, B) defined, then one of them or both can be selected.
10	FlexRay Tx Redundancy Handling: This feature is available if both channels (A & B) are configured. Static frames, which are sent at both channels at the same slot and with the same timing are defined as redundant frames. The redundancy handling lets the user define how redundant frames should be sent. Following options are possible: - None: There is no explicit handling for redundant frames. They are sent via separate message buffers for channel A and B. - Use Channel A: Redundant frames are sent via one message buffer (per frame) on both channels. If this option is enabled, then the redundant frame on channel B will be shown as ‘disabled’ in the frame list. - Use Channel B: Redundant frames are sent via one message buffer (per frame) on both channels. If this option is enabled, then the redundant frame on channel A will be shown as ‘disabled’ in the frame list. If the option ‘Use Channel A’ or ‘Use Channel B’ is used, it is not possible to send a redundant frame on a single channel.

Table 40: Description of the Bus Configurations dialog – “Base Settings” (FlexRay)

	Information
	The FlexRay parameter “SYNC-NODE-MAX” is always set to ‘15’.

5.6.2.3 Base Settings Ethernet

A “Base Settings” tab for bus type Ethernet is shown in Figure 41.

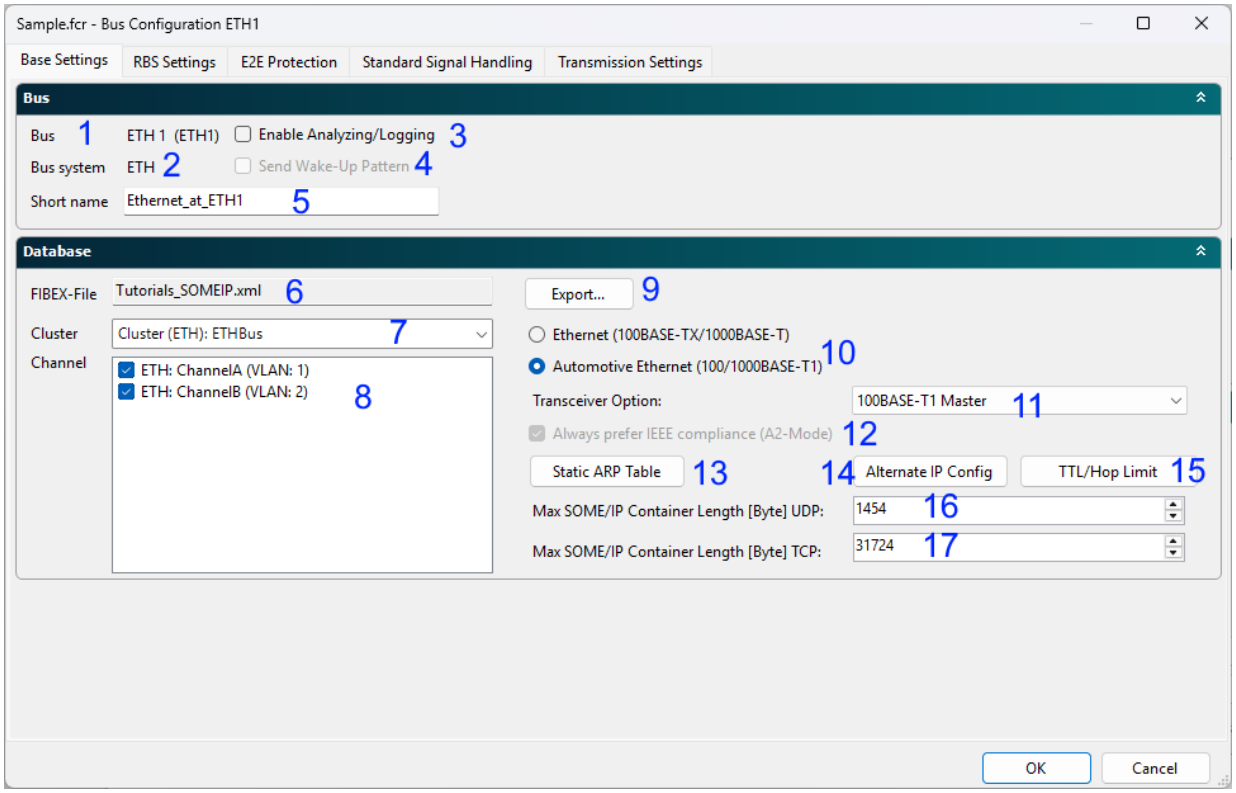


Figure 41: Bus Configuration – “Base Settings” (Ethernet)

The different elements of this dialog are described in Table 41.

No	Description
1	Name and Bus-ID of the current configured bus. In this example, the Bus-ID is '1'.
2	Type of the bus (Ethernet (ETH) here).
3	FL3X Config Analyzer (both Windows [9] and Android) can access and analyze this bus if this option is enabled. Also, this option must be enabled for data logging.
4	Wake-Up option is supported only by FlexRay and ETH with 10Base-T1S.
5	Short name of the configuration. The short name is displayed in the Main Window. Whitespaces are not allowed.
6	File name of the used database file. Within a project, the used database format is FIBEX 4.1.1. All other supported database formats are automatically converted.
7	Selected cluster for the chosen connector. Existing but not applicable clusters will be shown in grey font. If just one proper cluster is available and the corresponding setting (see chapter 5.22.1.1) is activated, then this cluster will be automatically selected.
8	Within the "Channel" section of this dialog, channel(s) of the appropriate cluster can be activated. In case of Ethernet, a channel represents a VLAN.
9	The *'Export'* button can be clicked to export the database as a FIBEX 4.1.1 file.
10	With this radio button, the used physical layer can be selected. The default physical layer is "Automotive Ethernet", which can be configured with the transceiver option "100BASE-T1 Master" or "100BASE-T1 Slave". If "Ethernet" is selected as physical layer, the supported ethernet speeds are shown as transceiver options (E.g., 100BASE-TX-Fixed, 1000BASE-T-Fixed or 10/100/1000-Auto).
11	With this drop-down menu, the "Transceiver Option" can be set for the selected physical layer.
12	To ensure compatibility with IEEE compliant 1000BASE-T1 networks (also known as 'Marvell A2-Mode') this checkbox must be activated. In case network partners using a Marvell PHY, hardware which supports only a none IEEE compliant mode (also known as 'Marvell A0/A1-Mode'), then this checkbox must be disabled. Please note that IEEE compliance is only ensured if it is supported by used device hardware. Otherwise, this configuration will be ignored.
13	The "*Static ARP Table*" button can be clicked to specify additional MAC-Addresses and appropriate IP settings. For more details see chapter 5.6.2.3.2.
14	The "Alternate IP Config" button opens a dialog to specify alternate IP addresses. This configuration is used when there is no IP configuration found in the database for a simulated frame or service. For more details, see chapter 5.6.2.3.3.
15	Opens the dialog for configuring the TTL/Hop Limit values. For more details see the chapter 5.6.2.3.4
16	Multiple SOME/IP messages are transmitted in single Ethernet Frames (an Ethernet Frame acts as container). The maximum size of a container can be configured with this option. This option sets the maximum length for SOME/IP messages using **UDP**. Hint: If a single SOME/IP message exceeds the maximum size of a container, it is fragmented via SOME/IP TP.
17	Multiple SOME/IP messages are transmitted in single Ethernet Frames (an Ethernet Frame acts as container). The maximum size of a container can be configured with this option. This field sets the maximum length for SOME/IP messages using **TCP**. Please do not change this value without contacting the STAR ELECTRONICS support team first as it affects the internal handling of the TCP communication.

Table 41: Description of the Bus Configurations dialog – "Base Settings" (Ethernet)

5.6.2.3.1 10BASE-T1S

Some FPGA images support 10BASE-T1S as physical layer. There are 10BASE-T1S related parameters which are shown below.

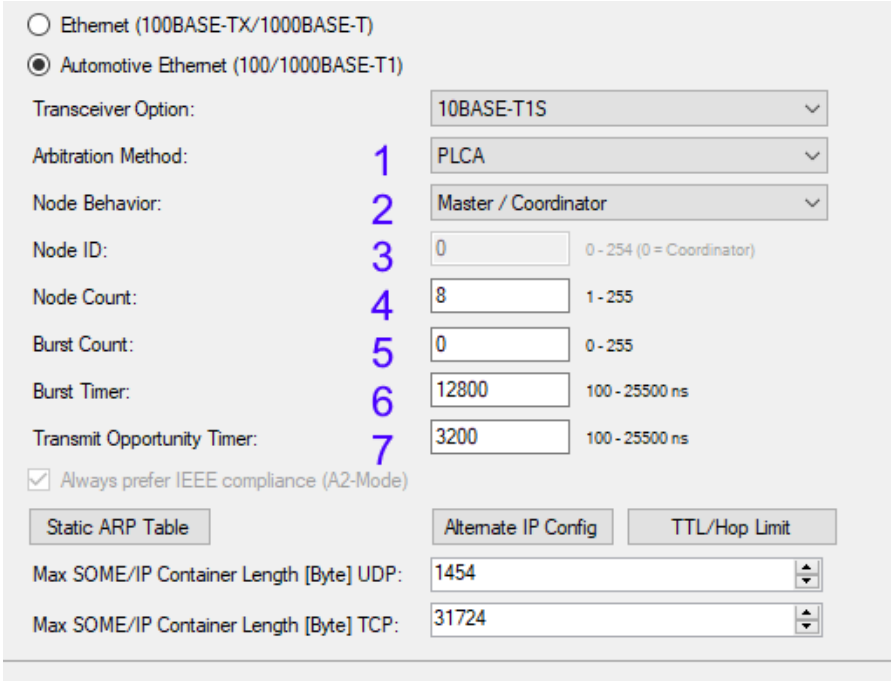


Figure 42: Bus Configuration – "Base Settings" (Ethernet) -10BASE-T1S Parameters

No	Description
1	Arbitration mode can be PLCA (Physical Layer Collision Avoidance) or CSMA/CD. The parameters 2 to 7 are only available if PLCA is used.
2	Node behavior of the bus.
3	Node ID of the bus. 0 is automatically used for Node Behavior = Master.
4	Number of nodes participating in communication.
5	Burst count allows higher bandwidth to the node.
6	Burst timer.
7	Transmission opportunity timer.

Table 42: Description of the Bus Configurations dialog – "Base Settings" (Ethernet) – 10BASE-T1S

	Information
	The 10BASE-T1S feature is only supported by the following Devices/Images: • FL3X Device-PCIe (Image 0xE0) • FL3X Device-PXIe (Image 0xD0) The support for 10BASE-T1S requires special FPGA image versions. Please contact the STAR ELECTRONICS support team.

	NOTICE
	When no PLCA coordinator is present, and therefore no BEACONs are detected within a certain time, the network participants fall back to CSMA/CD. This behavior is specified in Clause 148 of IEEE 802.3cg-2019.

5.6.2.3.2 Static ARP Table Configuration

The "Static ARP Table" dialog is shown in Figure 43.

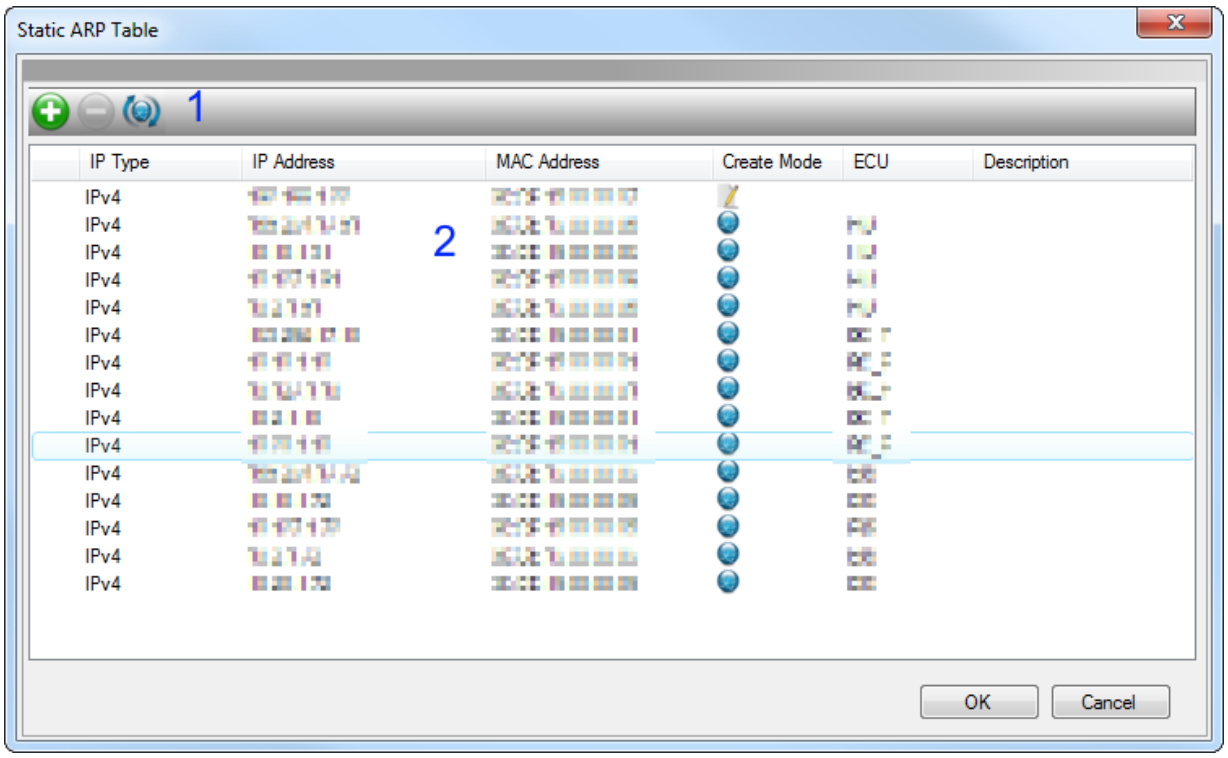


Figure 43: Base Settings for Ethernet – “Static ARP Table” configuration dialog

Generally, the static ARP table is automatically updated after any change in the “RBS Settings” tab. With this dialog, it is possible to additionally specify MAC-Addresses of ECUs which are not contained within the corresponding database. This is especially relevant if IP-Addresses, or more precisely, Application Endpoints are given in the database, where a corresponding ECU is missing. This could happen for an ECU-Extract of an AUTOSAR file (where the external ECUs are not described).

The elements of this dialog are described in Table 43.

No	Description
1	With the +/- buttons a new configuration can be added/removed to the appropriate bus. With the refresh button, all manually deleted elements are restored based on the current configuration.
2	IP Addresses and corresponding MAC Addresses can be edited within this section. The following features can be edited. • IP Type: If this field is clicked a drop-down box opens where the IP-Protocol can be selected (Ipv4/ Ipv6). • IP Address: Within this field an IP Address can be edited. The address must be edited byte wise in decimal syntax separated by a point (e.g., 192.10.10.2). • MAC Address: Within this field a MAC Address can be edited. The address must be edited byte wise in hexadecimal syntax separated by a colon (e.g. FF:FF:FF:FF:FF:FF). • Create Mode: Shows, whether the user has edited/added the item or if it was added automatically from the database (based on the current configuration). • ECU: The ECU, that is assigned to this Application Endpoint within the database. • Description: This field can be used to describe the item.

Table 43: Base Settings for Ethernet – “Static ARP Table” configuration

	NOTICE
	A dynamic resolved MAC in runtime may override the settings from this dialog.

5.6.2.3.3 Alternate IP Configuration

The “Alternate IP Configuration” dialog is shown in Figure 44.

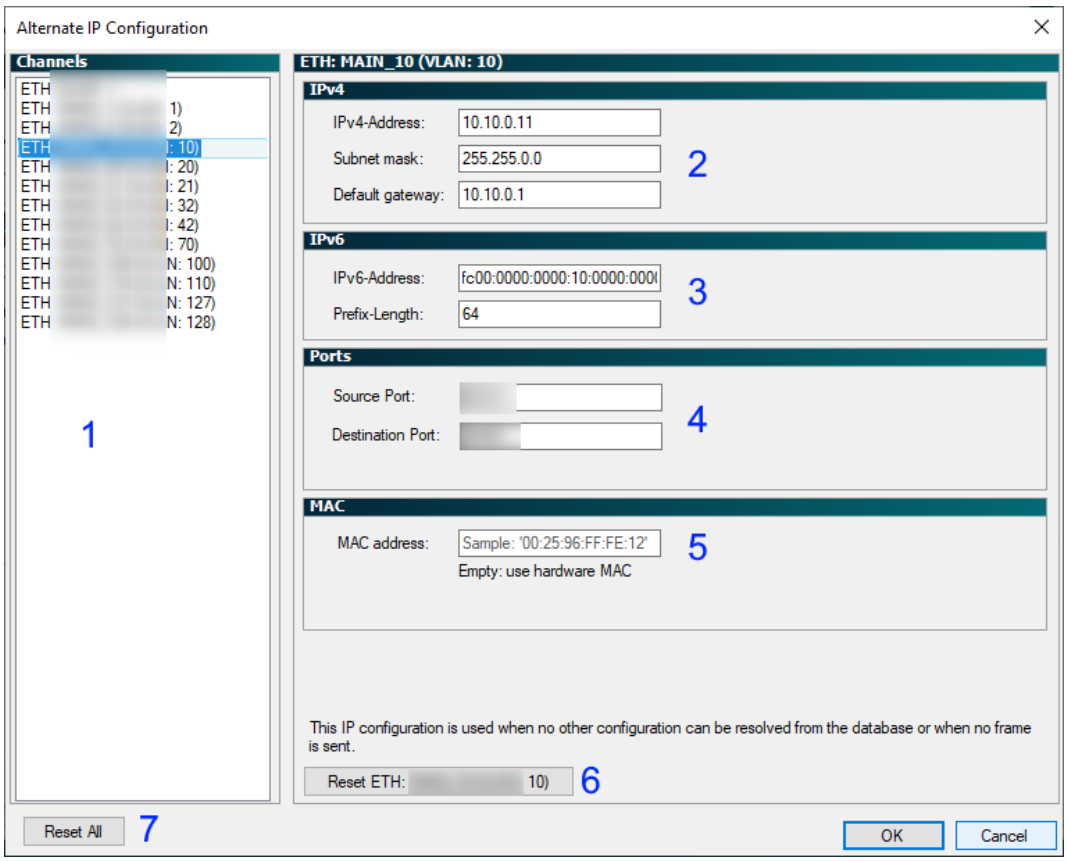


Figure 44: Base Settings for Ethernet – “Alternate IP Configuration” dialog

For each channel of the current bus (see Figure 44: **1**) the alternate IP configuration can be set (see Figure 44: **2 & 3**). If the source and/or destination port number for any endpoint is not defined in the database (e.g., dynamically assigned port numbers), the project will use the values shown in Figure 44: **4**.

A channel without VLAN is always available.

The used default values are:

- IPV4-Address: 10.*%VID%*.0.(10 + *%BUSID%)*
- Subnet mask: 255.255.0.0
- Default Gateway: 10.*%VID%*.0.1
- IPV6-Address: fc00:0000:0000:*%VID%*:0000:0000:*%BUSID%*:0001
- Source Port: Depends on used OEM package
- Destination Port: Depends on used OEM package
- MAC address: Not defined (use hardware MAC)

VID: VLAN-ID of the selected channel.

BUSID: Bus-ID of the configured bus (see Table 41 no. **1**).

The button [Reset All] (**7**) resets all changes for all VLANs. The VLAN specific [Reset] button (**6**) resets only the changes for the selected VLAN.

	Information
	The default values may be different if an OEM package is used. Please read the documentation of the OEM package for more information.

5.6.2.3.4 TTL/Hop Limit Configuration

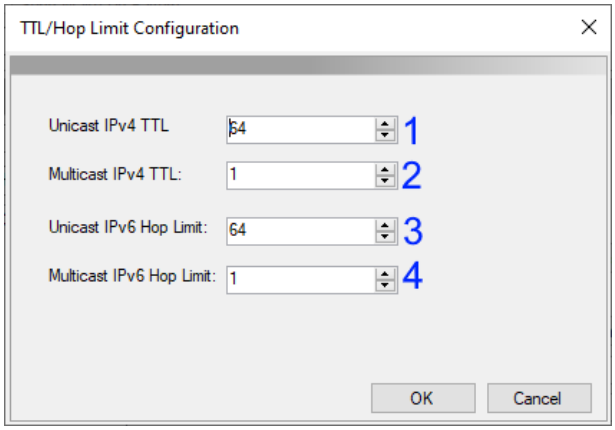


Figure 45: Base Settings for Ethernet – “TTL/Hop Limit Configuration” dialog

The TTL values are used for IPV4 packets and the Hop-Limit values are used for the Ipv6 packets. These values are used to define the maximum count of network devices (e.g., routers) which the packets can pass from the source to the destination.

5.6.2.4 Base Settings LIN

A “Base Settings” tab for bus type LIN is shown in Figure 47.

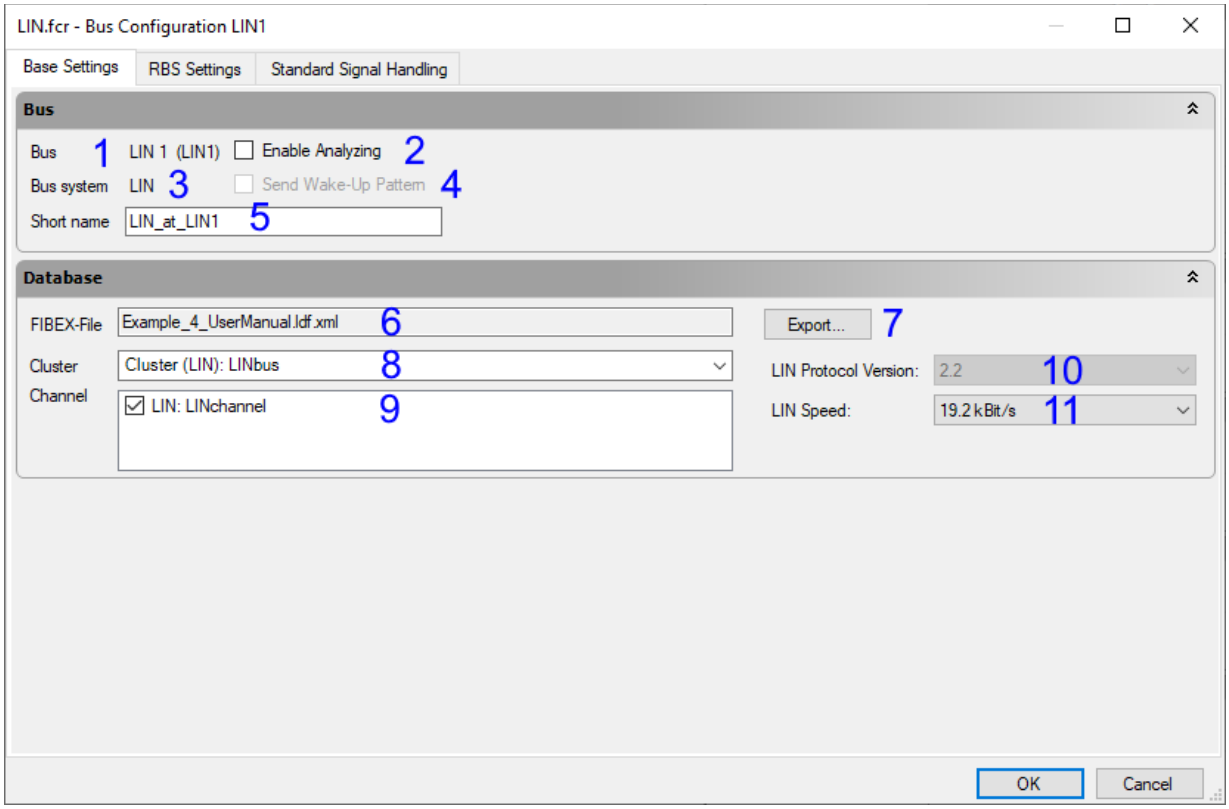


Figure 46: Bus Configuration – “Base Settings” (LIN)

The different elements of this dialog are described in Table 44.

No	Description
1	Name of the current configured bus.
2	FL3X Config Analyzer (both Windows [9] and Android) can access and analyze this bus if this option is enabled. Also, this option must be enabled for data logging.
3	Type of the bus (LIN here).
4	Send Wakeup Pattern: Not supported for LIN
5	Short name of the configuration. The short name is displayed in the Main Window. Whitespaces are not allowed.
6	File name of the used database file. Within a project, the used database format is FIBEX 4.1.1. All other supported database formats are automatically converted.
7	The *'Export'* button can be clicked to export the database as a FIBEX 4.1.1 file.
8	Selected cluster for the chosen connector. Existing but not applicable clusters will be shown in grey font. If just one proper cluster is available and the corresponding setting (see chapter 5.22.1.1) is activated, then this cluster will be automatically selected.
9	Within the "Channel" section of this dialog, channel(s) of the appropriate cluster can be selected.
10	The LIN protocol version defined in the loaded database (e.g **.ldf*)
11	The LIN speed to use (initial the defined speed from the database).

Table 44: Description of the Bus Configurations dialog – "Base Settings" (LIN)

NOTICE

The resistor required for a LIN Master is not set per default.
The resistor can be set via the device FL3X Diagnostics Website (see chapter 8) (see tab "Hardware & Versions" -> "Configurable Termination". The "self check" tab will show if the resistor setting matches the Master/Slave configuration.

5.6.3 RBS Settings

The "RBS Settings" tab of the "Bus Configuration" window is used to configure the RBS for the selected bus. All data is based on the previously selected database file.

This dialog can be opened by clicking the *'Edit bus configuration…'* button of a connector within the FL3X Config main window. A detailed description depending on the bus type is given in the following chapters.

NOTICE

If the "Bus Configuration"-dialog is closed with **[OK]** and frames are ignored (via ignored ECUs, via ignored frames) that are already related in user functions or routings, then separate dialogs pop up for each of these user functions/routings! Here options are available to return to the dialog, or to delete the relevant user function/routing, or to add the relevant frames to the list of frames which are sent by the RBS.

5.6.3.1 RBS Settings CAN/CAN-FD

The "RBS Settings" tab for a bus of type CAN-FD is given in Figure 47.

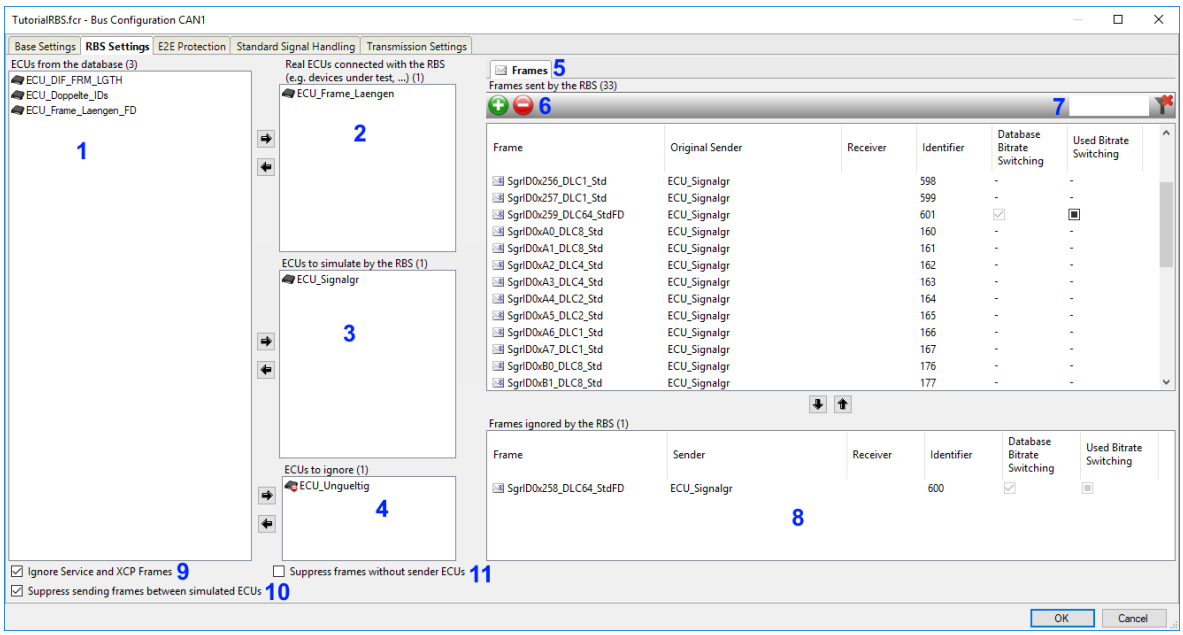


Figure 47: Bus Configuration – “RBS Settings” for CAN-FD

The meaning of the elements used in this dialog is given in Table 45.

No	Description
1	All ECUs defined in the database file and which are currently not in use are listed here.
2	An ECU can be moved/removed to this window by using the arrow buttons. All received frames of the ECUs listed here, will be simulated by the RBS. This includes frames with no dedicated sender ECU within the database.
3	An ECU can be moved/removed to this window by using the arrow buttons. All sent frames of the ECUs listed here, will be simulated by the RBS.
4	An ECU can be moved/removed to this window by using the arrow buttons. All sent frames of the ECUs listed here, will be ignored by the RBS.
5	The “Frames” tab shows the summary of all frames sent by the RBS. The columns of this section are: • Frame: This column contains the short names of all frames sent by the RBS. • Original Sender: This column shows the short names of the original sending ECUs of the corresponding frames. • Identifier: This column shows the CAN Identifiers of the corresponding frames. • Database Bitrate Switching (CAN-FD only): Here the Bitrate Switching property of the corresponding frame is shown. If this property is not available within the corresponding database a “-” sign is shown. Otherwise, a checkbox is shown which is enabled or disabled. • Used Bitrate Switching (CAN-FD only): Here the Bitrate Switching property can be edited by the user. This can be done by clicking on the appropriate field of the corresponding frame. Initially a filled check box is shown here. This means that the value is set to the same value as in the database. The other two possibilities are to check or uncheck the checkbox (enable/disable Bitrate Switching). Within this section, it is possible to create “Transmit Frame User Functions” and/or “Transmit PDU User Functions”. To add a transmit user function, perform a right click on the selected frame to open the context menu. Then click on the menu item *‘Create Transmit Frame UserFunction’* or *‘Create Transmit PDU UserFunction’’*. Attention: User Functions are created but the User Function Editor does not open automatically! To edit the created Transmit User Functions, open the User Function Editor manually (see chapter 5.8).
6	Use the *‘Add Frames’* button to add single frames to the RBS. This is relevant if only single frames of an ECU shall be sent and others not. When a database is used, then only frames that are defined within this database can be selected as additional frames. (See chapter 5.6.3.4) Use the ‘*Remove Frame*’ button to move a selected frame to the “Frames ignored by the RBS” list (**8**).

No	Description
7	Use this filter to show only elements with the configured filter content. E.g. if the text 'ABCD' is entered, only the frames with a short name or an ECU name including 'ABCD' are shown.
8	All frames which shall be ignored (not sent) by the RBS are listed here. Frames can be moved/removed to this window by using the arrow buttons. In addition, all gateway routings and all Tx/Rx UserFunctions of these frames are deleted.
9	When selecting this option, all frames marked as SERVICE, TPL, XCP_PRE_CONFIGURED or XCP_RUNTIME_CONFIGURED in the database will be ignored, which means the device won't send them. This option also prevents these frames as additional frames (6).
10	If this option is selected, the frames sent between simulated ECUs are suppressed. This option should be activated, if possible, to improve runtime performance.
11	If this option is selected, frames without a sender ECU will be suppressed.

Table 45: Bus Configuration – "RBS Settings" for CAN/CAN-FD

5.6.3.1.1 CAN – User Defined Database

If the option "Create user defined database" (s. Figure 38) has been selected, new CAN frames can be defined in the "RBS Settings" tab of the "Bus configuration" window.

Clicking on the 'Add Frames' button will open the following dialog (s. Figure 48) to add a new frame to the user database.

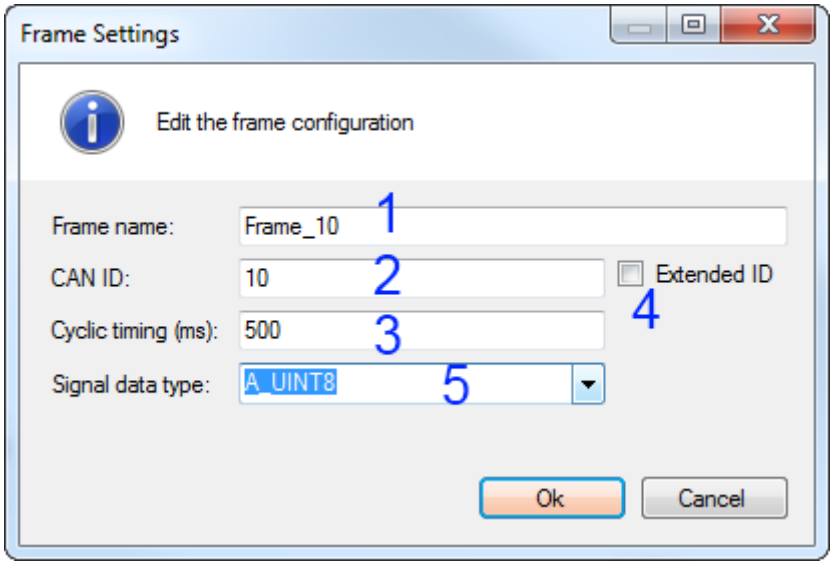


Figure 48: Bus Configuration – "RBS Settings" for CAN – "Frame Settings" Dialog

No	Description
1	The name of the frame.
2	CAN ID.
3	Database timing of the frame.
4	If the "Extended ID" option is not selected, then the ID can be a number between 0 and 2047. If this option is selected, then the ID can be a number between 0 and 536870911.
5	Data type of the signal. The data type cannot be changed after the frame is created.

Table 46: Bus Configuration – "RBS Settings" for CAN – "Frame Settings" Dialog

Clicking on the '*Remove Frames'* button will delete the selected frames from the user database. A custom CAN frame can be edited by double clicking the frame in the "Frames send by the RBS" list.

NOTICE

The option “Create user defined database” on a CAN-FD bus creates a CAN database, which supports CAN high-speed, but **not** CAN-FD.

5.6.3.2 RBS Settings FlexRay

The “RBS Settings” tab for a bus of type FlexRay is given in Figure 49.

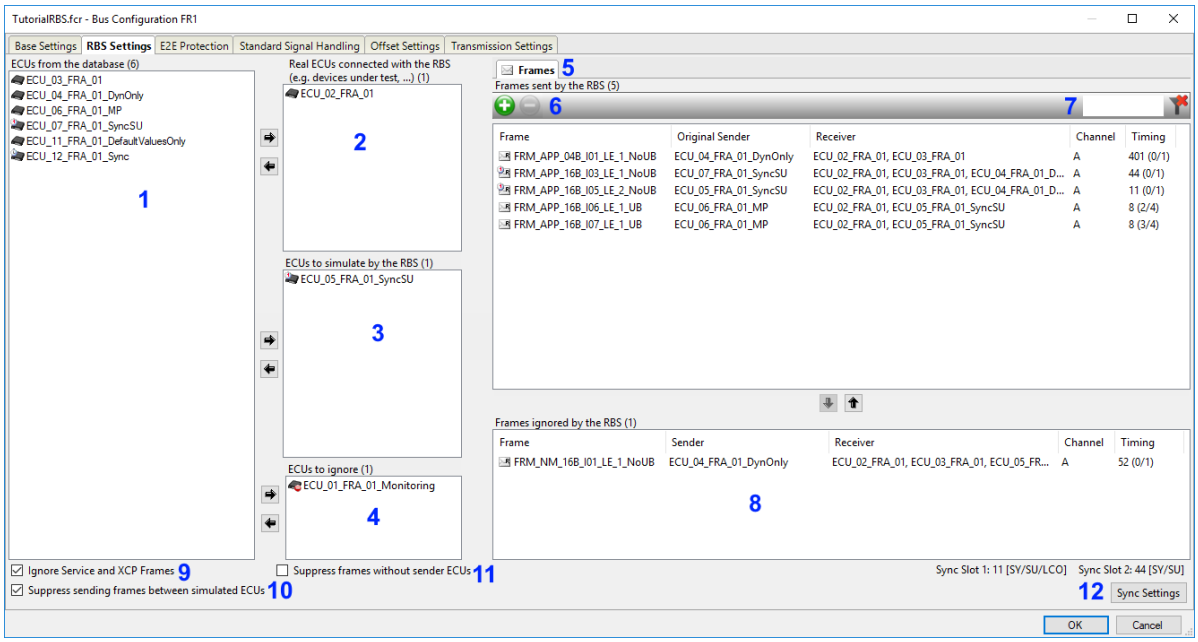


Figure 49: Bus Configuration – “RBS Settings” for FlexRay

The meaning of the elements used in this dialog is given in Table 47.

No	Description
1	All ECUs defined in the database file and which are currently not in use are listed here. The symbol means that this ECU sends Sync Frames. The symbol means that this ECU sends Sync Start-Up Frames. The information if an ECU sends Sync or Start-Up Frames depends on the used database. It is independent of the configuration made in the “Edit sync frames” dialog (see Figure 50).
2	An ECU can be moved/removed to this window by using the arrow buttons. All received frames of the ECUs listed here, will be simulated by the RBS. This includes frames with no dedicated sender ECU within the database. There is one special exception: If the ECU configuration requires to send a frame which has no sending ECU and there is an ECU configured, which sends a frame in the same Slot-ID (but with a different timing). Then the required frame will not be sent by the RBS.
3	An ECU can be moved/removed to this window by using the arrow buttons. All sent frames of the ECUs listed here, will be simulated by the RBS.
4	An ECU can be moved/removed to this window by using the arrow buttons. All sent frames of the ECUs listed here, will be ignored by the RBS.

No	Description
5	The “Frames” tab shows the summary of all frames sent by the RBS. The columns of this section are: • Frame: This column contains the short names of all frames sent by the RBS. If a frame is shown with the symbol  , then this frame is a Sync-Frame. If a frame is shown with the symbol  , then this frame is a Sync Start-Up Frame. The information if a frame is a Sync or Start-Up Frame depends on the used database. It is independent of the configuration made in the Bus Configuration – “RBS Settings” for FlexRay – “Edit Sync Frames” . If a frame is shown with the symbol  then this frame will be sent, because a real ECU needs to receive it. If a frame is shown with the symbol  then this frame will be sent, because a simulated ECU sends it. • Original Sender: This column shows the short name of the original sending ECU of the corresponding frame. • Channel: This column shows the channel (A or/and B) of the corresponding frame. • Timing: The FlexRay timing (Identifier, Base-Cycle, Cycle-Repetition) of the appropriate frame is given within this column. The information shown in this section is given by the corresponding database and is not editable. Within this section, it is possible to create “Transmit Frame User Functions” and/or “Transmit PDU User Functions”. To add a transmit user function, perform a right click on the selected frame to open the context menu. Then click on the menu item *‘Create Transmit Frame UserFunction’* or *‘Create Transmit PDU UserFunction’’*. Attention: User Functions are created but the User Function Editor does not open automatically! To edit the created Transmit User Functions, open the User Function Editor manually (see chapter 5.8).
6	Use the *‘Add Frames’* button to add single frames to the RBS. This is relevant if only single frames of an ECU shall be sent and others not. (See chapter 5.6.3.4) Use the ‘*Remove Frame*’ button to move a selected frame to the “Frames ignored by the RBS” list (**8**).
7	This point is like the filter of the CAN dialog (s. Table 45 for more information).
8	All frames which shall be ignored (not sent) by the RBS are listed here. Frames can be moved/removed to this window by using the arrow buttons. In addition, all gateway routings and all Tx/Rx UserFunctions of these frames are deleted.
9	When selecting this option, all frames marked as SERVICE, TPL, XCP_PRE_CONFIGURED or XCP_RUNTIME_CONFIGURED in the database will be ignored, which means the device will not send them. This option also prevents these frames as additional frames (**10**).
10	If this option is selected, the frames sent between simulated ECUs are suppressed. This option should be activated, if possible, to improve runtime performance.
11	If this option is selected, frames without a sender ECU will be suppressed.
12	This button opens the “Edit Sync Frames” dialog (see chapter 5.6.3.2.1).

Table 47: Bus Configuration – “RBS Settings” for FlexRay

5.6.3.2.1 Edit FlexRay Sync Settings

The “Edit Sync Frames” dialog is shown in Figure 50. In this dialog the Sync-, Sync/Start-Up- and Coldstart feature can be configured for a maximum number of two frames. The settings can be done automatically or by hand. Table 48 below gives a detailed description.)

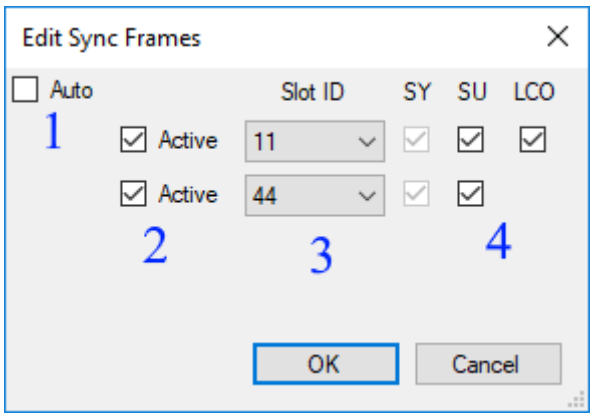


Figure 50: Bus Configuration – “RBS Settings” for FlexRay – “Edit Sync Frames” dialog

No	Description
1	If the “Auto” check box is activated (which is the case by default), the settings for Sync-, Sync/Start-Up- or Coldstart-Frames are done automatically. If the “Auto” check box is deactivated the settings can be done manually.
2	Use the “Active” check box to activate the corresponding settings.
3	With these drop-down lists slot IDs can be selected to configure frames of type Sync, Sync/Start-Up or Coldstart.
4	Use the check boxes to configure whether the considered frame is a Sync-Frame (SY is activated), or a Sync/Start-Up-Frame (SY and SU are activated). The first configured Sync/Start-Up-Frame can also be configured as leading coldstarter (SY, SU and LCO are activated). This means, that the device tries to initiate the FlexRay synchronization for this connector.

Table 48: Bus Configuration – “RBS Settings” for FlexRay – “Edit Sync Frames” Dialog

NOTICE

If possible, only one message buffer is used for each slot ID (FlexRay). There is a maximum of 249 message buffers available (each with 127 bytes, see chapter 12).

NOTICE

The “Sync Settings” configuration is used for all configured channels of the current bus.

If the bus is configured for channel A only, the Sync/Start-Up-Frames are only sent on channel A (and vice versa for channel B only).

If a bus is configured for channel A+B, then the Sync/Start-Up-Frames are always sent on both channels, no matter what ECU configuration is used.

5.6.3.3 RBS Settings Ethernet

Depending on the ECU configuration FL3X Config configures SOME/IP Service Discovery automatically. The user can any time activate/deactivate the configured Services, change the direction of them or add any service, which is defined in the database.

The “RBS Settings” tab for a bus of type Ethernet is given in Figure 51.

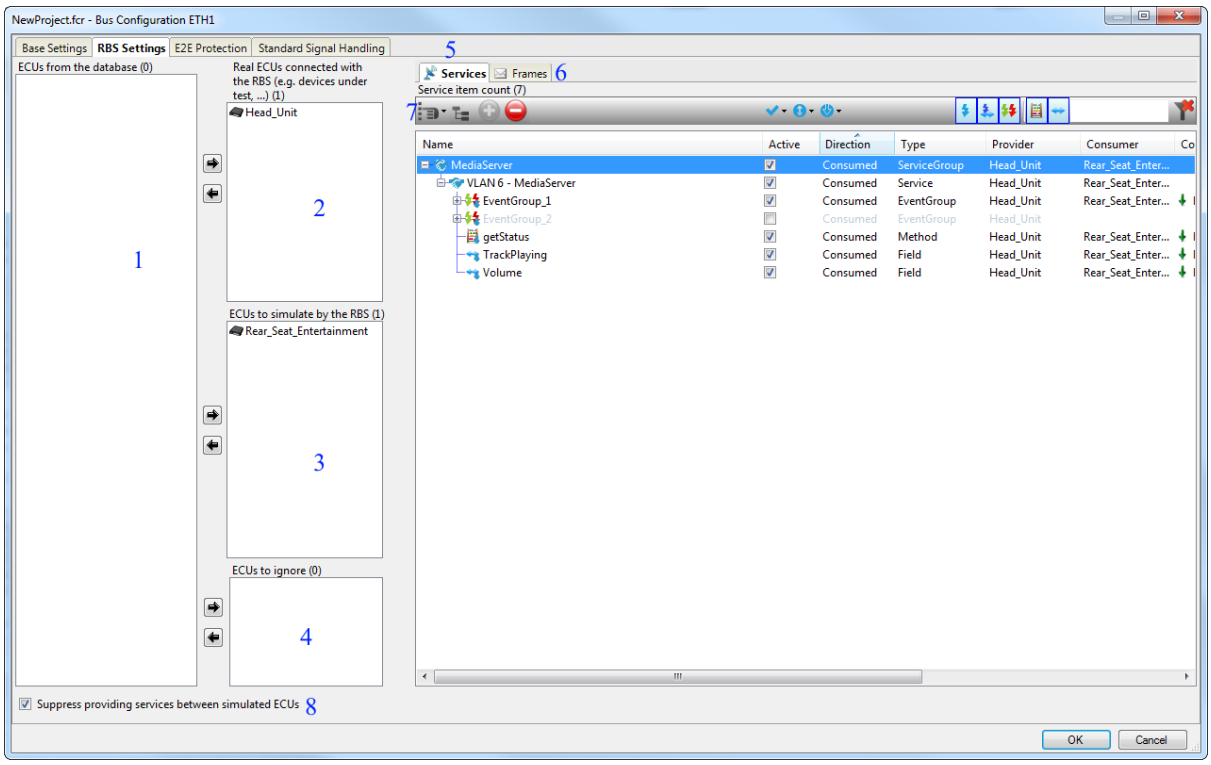
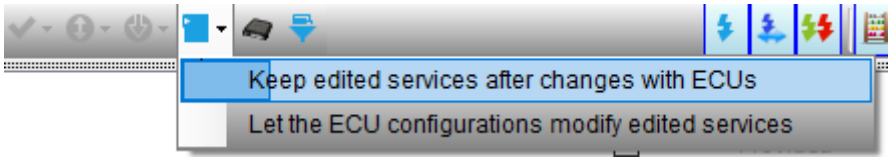
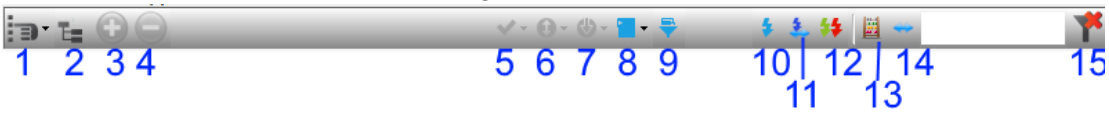
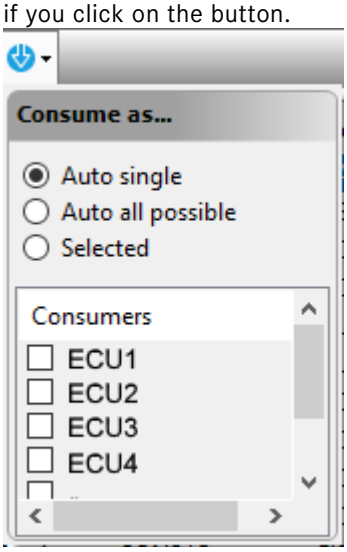


Figure 51: Bus Configuration – “RBS Settings” for Ethernet (tab “Services” in front)

The description of the elements in this tab is given in Table 49.

No	Description
1,2,3,4	These elements are like the CAN dialog (s. Table 45 for more information). With the difference that the configuration of the ECUs also affects the automatically provided and consumed services. The rules for automatically configured services are: • Real ECUs: - All Services, that are provided by these ECUs, will be consumed by the RBS. - All Services, that are consumed by these ECUs (and are not provided by other real ECUs), will be provided by the RBS. • Simulated ECUs: - All Services, that are provided by these ECUs, will be provided by the RBS (see no. 8). - All Services, that are consumed by these ECUs, will be consumed by the RBS. Attention: Real ECUs have always a higher priority than simulated ECUs. E.g. if a service is consumed both by a real ECU and a simulate ECU, the RBS will provide it. • Ignored ECUs: These ECUs are not considered for the automatic service configuration.
5	The “Services” tab shows the summary of all services provided and consumed by the RBS. But it contains only the Service Discovery configuration via SOME/IP. The columns of this section are: • **Name**: Contains the short names of the service elements. For each service there is a tree view with the used VLANs and used event groups (per VLAN). • **Validation**: The column shows warnings for the current configuration. • **Active**: If set, the service or its event group is actively provided/consumed by the RBS. It is possible to activate only a subset of event groups for a single service. • **Direction**: Shows the direction of the service from the point of view of the RBS. An RBS can either provide or consume a service. • **Type**: Shows the type for each service element in the list (e.g. “EventGroup”). • **Edited**: This column shows if the service is modified. All services configured automatically by the ECU configuration are marked as ‘auto’ (checkbox is not checked). If you modify a service (e.g. change direction or deactive some service elements), then this service is marked as “edited” (checkbox is checked). The checkbox can alse be manually edited. Custom services, which are added via “Add additional services” are also marked as ‘edited’.  Depending on the configuration (see figure above), a change of the ECU configuration may override or keep the edited services. If the checkbox of an ‘edited’ service is manually unchecked, it is tried to adapt the service to the ECU configuration. • **Consumer**: Shows all ECUs that are defined as Consumer for this service element in the database. • **Consumed As**: This column is only valid for “Direction = Consumed”. The parameters of this ECU are used by the RBS to consume a service at runtime, e.g. the local IP address. • **Instance ID**: The Instance ID of the service as defined in the database. • **Service ID**: The Service ID of the service as defined in the database. • **Usage**: All elements, that are automatically configured, have the value “RBSAuto”. All elements, that are edited by the user have the value “RBSCustom”. • **VLAN**: If the element is part of a VLAN, then this column contains the VLAN ID. • **Transport Protocol**: The ethernet transport protocol (TCP/UDP) • **Size**: The size (in bytes) of the service element.

No	Description
6	The “Frames” tab shows the summary of all frames sent by the RBS. This includes frames without any service relation (e.g. Network Management). Since FL3X Config version 1.1, frames associated with services are no longer shown in the frame list when the bus is created (see Figure 52 for an example).
7	The menu bar for the services has following buttons:  1. Use this filter to show different levels of the services. 2. Collapse/Expand the list of services. 3. The *‘Add Services’* button makes it possible to add a service to the RBS. Any service in the database, that is not configured yet, can be added either as consumed service or as provided service. 4. Use the ‘*Remove/Disable Service*’ button to disable the selected service. Manually added services can be deleted again. The services, which are added automatically (ECU configuration) can be just disabled. 5. Activates/Deactivates the selected item (also available in the context menu). Inactive services (or event groups) will not be subscribed or consumed. No user function can be created for the inactive service items (e.g. fields). 6. Changes the direction (consumed/provided) of the (selected) services. 7. Changes the consumer behavior. If a service is configured as “consumed” and it has multiple consumer ECUs, FL3X Config tries consuming it with the properties (Application/Network Endpoint) of the first found suitable ECU (E.g. the first found consumer ECU, which is in the list of simulated ECUs). The following window will appear if you click on the button.  **Auto single**: Default configuration. The first found suitable ECU will be used as consumer **Auto all possible**: If this option is selected, the service will be consumed for each consumer ECU **Selected**: This option activates the list of all possible consumers, where you can choose the ECU(s), which should be used as consumer 8. See 5 for more information. 9. Shows/hides the services, which cannot be activated. Some services are always inactive (E.g. Consumed services, which do not have any consumer ECUs in the configuration). The tooltip of a service shows the reason why the service cannot be activated. 10. Shows/hides events 11. Shows/hides field notifier 12. Shows/hides event groups (hiding event groups will also hide events and field notifier) 13. Shows/hides methods 14. Shows/hides fields 15. This point is like the filter of the CAN dialog (s. Table 45 for more information).

No	Description
8	This option suppresses the automatic configuration of provided services between simulated ECUs. Example: A service S1 is provided by ECU_1 and is consumed only by ECU_2. If ECU_1 and ECU_2 are both simulated, then the service S1 will not be provided automatically by the RBS if this option is activated.

Table 49: Bus Configuration – “RBS Settings” for Ethernet

The changes, which the user makes on services are retained, even though a new ECU configuration needs an opposite configuration.

Example 1:
A real ECU is configured, and it consumes one event group from a service, which has two event groups. This service will be configured as “provided” and just the needed event group will be activated in FL3X Config. The user activates now manually the second event group. The second event group will be retained ‘activated’ even though an additional ECU configuration causes the corresponding service to be consumed.

Example 2:
The user changes the direction of a service as ‘consumed’, which was configured from FL3X Config as “provided”. This service will remain as “provided”, even though a new ECU configuration needs the corresponding service to be consumed.

The “Frames” Tab is always shown.
- For any projects created with FL3X Config 1.0 or older, frames that are related to SOME/IP events are shown in this tab (an example can been seen in Figure 52).
- For any projects created with FL3X Config 1.1 or later, frames that are related to SOME/IP events are **not** shown in this tab.
- Frames, which are not related to SOME/IP (E.g. UDP Frames for Network Management), are always shown in this tab.

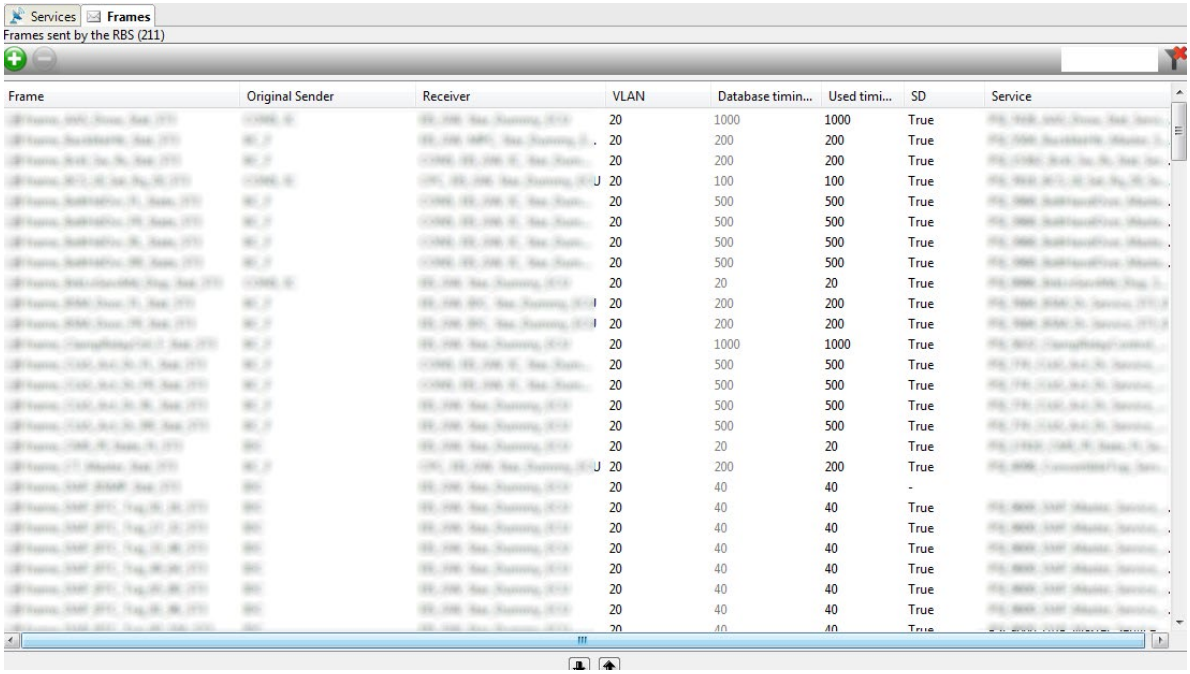


Figure 52: Bus Configuration – “RBS Settings” for Ethernet (tab “Frames” in front)

The column “Service” shows the service of the frame, which has the event connected with the frame. If the frame has no service, it is probably an UDP-Frame.

The column “SD” determines whether “Service Discovery” is used for this frame. If the frame is related with a service (service column not empty), then the “SD” column can be true or false. If it has the value “true”, the related SOME/IP event is activated and will be used to send the frame. If the “SD” column has the value “false”, then the related “SOME/IP” event is not active. The frame will be sent over UDP. If the “SD” column has the value “-”, then the frame is not related with an SOME/IP event and it will be sent over UDP.

The frames, which are related with SOME/IP events cannot be moved to the “ignore list”. If the user disables an event, the related frame will be removed from the list of frames. The user can use the “Add” button, to add a frame for sending it without Service Discovery.

	NOTICE
	The services which use SOME/IP TP can be activated just for Service Discovery. The service elements (e.g. service events) are not supported and cannot be activated.

	NOTICE
	Service items (event, method, field), which are larger than 300MB, are not supported by FL3X Config. Such items are shown as disabled in Bus Config Dialog. Other dialogs do not show those items.

	Information
	If a service item cannot be activated, try to read its tooltip, which should show the reason why the item is disabled.

5.6.3.4 RBS Settings LIN

The “RBS Settings” tab for a bus of type LIN is given in Figure 53.

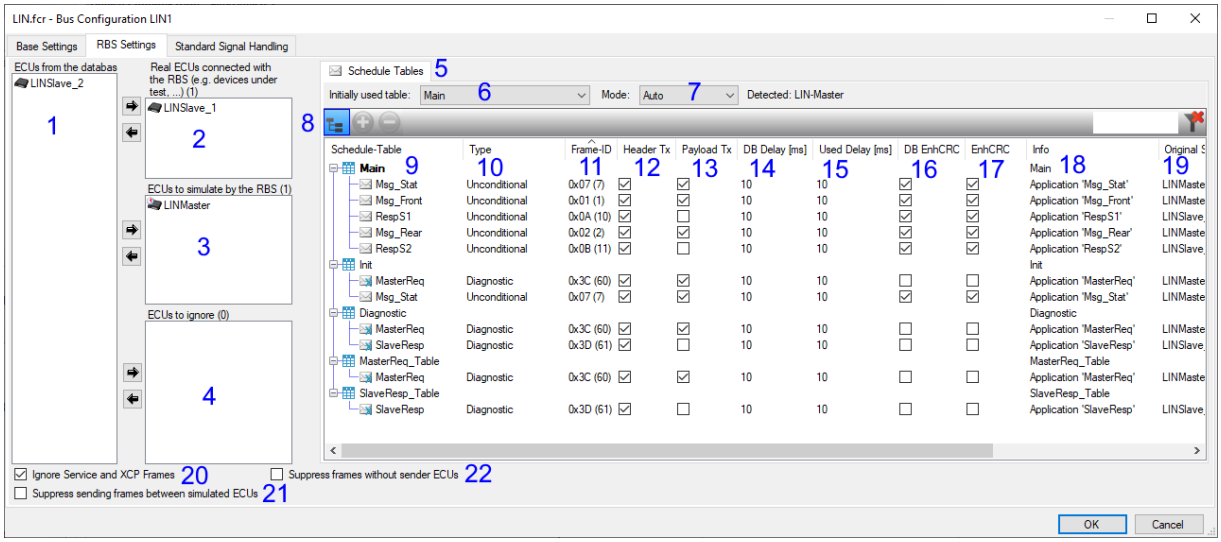
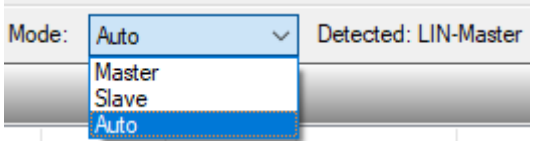


Figure 53: Bus Configuration – “RBS Settings” for LIN

The elements are described Table 52.

No	Description
1	All ECUs defined in the database file and which are currently not in use are listed here. The symbol means that this ECU is the LIN Master. The information if an ECU is the LIN Master depends on the used database.
2	An ECU can be moved/removed to this window by using the arrow buttons. All received frames of the ECUs listed here, will be simulated by the RBS. This includes frames with no dedicated sender ECU within the database.
3	An ECU can be moved/removed to this window by using the arrow buttons. All sent frames of the ECUs listed here, will be simulated by the RBS.
4	An ECU can be moved/removed to this window by using the arrow buttons. All sent frames of the ECUs listed here, will be ignored by the RBS.
5	The “Schedule Tables” tab shows all schedule tables and schedule table entries defined in the database.

No	Description
6	The “Initially used table” defines the schedule table which will be active when the compiled FL3X Config software starts. The DropDown lists all available schedule tables. The selected initial table is highlighted in **bold**.
7	The “Mode” specifies if the RBS should act as LIN master or LIN slave  • Master: The compiled RBS should act as LIN master. LIN Header and RBS related payload is provided by the RBS. • Slave: The compiled RBS should act as LIN slave. Only the RBS related payload is provided by the RBS. • Auto: The mode (master/slave) depends on the current configuration. The detected mode is shown behind the Drop-Down box. **Auto selection:** The RBS always tries to provide a functioning bus. As long the LIN master ECU is NOT specified as real ECU, the RBS will act as LIN master. If the LIN Master ECU is specified as simulated ECU, the RBS will act as LIN master. If the LIN master ECU is configured as real, the RBS will act as LIN slave. If the selected mode does not match the ECU configuration, a warning will be shown. But the configuration is not prevented.
8	The search and filter bar to filter the schedule table view
9	Schedule Table column: Shows the short names and structure of the schedule tables and the schedule table entries. The “Initially used table” is highlight in **bold**. • Schedule Table • Unconditional Frame • Event Triggered Frame (currently not supported) • Sporadic Frame (Currently not supported) • Diagnostic and Protocol Frame (e.g. MasterReq, SlaveResp, Assign Frame-ID, …) Protocol frames are only sent with the initial payload. A protocol stack must be implemented via user functions. • Protocol Frame “Free Format”
10	Type column: Shows the type of a schedule table entry. Possible options: Unconditional, Event-Triggered, Sporadic Diagnostic (MasterReq/SlaveResp) Assign Frame-ID, Unassign Frame-ID, Assign NAD, Conditional Change NAD, Data Dump, Assign PID, FreeFormat, Save Configuration
11	Frame-ID column: Shows the unique LIN identifier of the frame (schedule table entry)
12	Header Tx column: Shows if the LIN (frame) header is transmitted by the RBS • Unchecked: The LIN header is NOT transmitted by the RBS. Normally this is when the RBS act as LIN slave. The LIN headers are transmitted by a real LIN master ECU • Checked: The LIN header is transmitted by the RBS Normally this is when the RBS act as LIN master. The LIN header must not be transmitted by a real (master) ECU. When the RBS acts as LIN master (auto detected or explicitly) then the header Tx flag can be changed by clicking on the check box.

No	Description
13	Payload Tx column: Shows if the frame payload is transmitted by the RBS. • Unchecked: The payload is NOT transmitted by the RBS. This is the default value if the frame (payload) is not expected by real ECUs and not transmitted by any simulated ECU. • Checked: The payload is transmitted by the RBS This is the default value if the frame (payload) is expected by real ECUs or transmitted by a simulated ECU. This option can be changed for each frame. This can be done independent of the selected mode. Hint: All instances of a Frame (over all schedule tables) will be set at once. The instances of one frame cannot be set independent. This is because a slave ECU neither knows which schedule table is active nor at which position within the schedule table the communication is or which entries a schedule table has.
14	DB Delay [ms] column: Shows the frame time (duration of the frame slot) as specified in the database
15	Used Delay [ms] column: Shows the frame time (duration of the frame slot) as used by the RBS. Normally this is the same as specified in 14. This value can be changed by editing the value in the column.
16	DB Enh CRC column: This checkbox shows which checksum model should be used according to the database. • Unchecked: The classic checksum model should be used. • Checked: The enhanced checksum model should be used.
17	Used Enh CRC column: This checkbox shows which checksum model is used by the RBS. Normally this is the same as specified in 16. • Unchecked: The classic checksum model is used. • Checked: The enhanced checksum model is used. This value can be changed by clicking on the check box. Hint: All instances of a Frame (over all schedule tables) will be set at once. The instances of one frame cannot be set independent. This is because a slave ECU neither knows which schedule table is active nor at which position within the schedule table the communication is or which entries a schedule table has.
18	Info column: Shows additional information about the schedule table or schedule table entry.
19	Original Sender column: Shows the short name of the original sending ECU of the corresponding frame (as defined in the loaded database).
20	When selecting this option, all frames marked as SERVICE, TPL, XCP_PRE_CONFIGURED or XCP_RUNTIME_CONFIGURED in the database will be ignored, which means the device will not send them. This option also prevents these frames as additional frames (10).
21	If this option is selected, the frames sent between simulated ECUs are suppressed. This option should be activated, if possible, to improve runtime performance.
22	If this option is selected, frames without a sender ECU will be suppressed.

Table 50: Bus Configuration – “RBS Settings” for LIN

5.6.3.5 Additional Frames

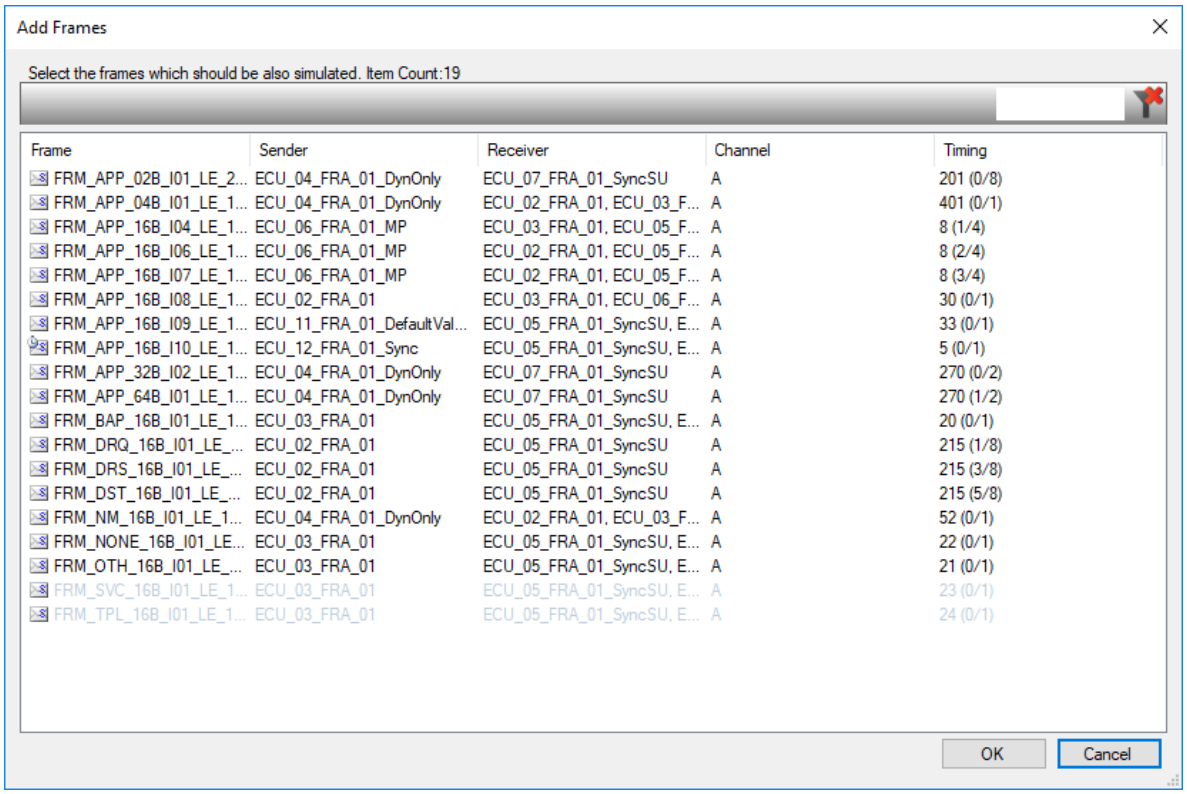


Figure 54: Bus Configuration – Add Frames Dialog

Normally the ECU configuration (real and simulated ECUs) defines the frames to be sent. The “Add Frames” dialog can be used to simulate any frame which is not already shown in the list “Frames sent by the RBS” and “Frames ignored by the RBS”.

Multiple frames can be selected. An ECU configuration is not necessary. Some frames may be shown disabled in this dialog. These are the frames, which cannot be simulated because of the activated settings like “Ignore Service and XCP Frames”.

5.6.4 E2E-Protection

The E2E-Protection configures handling of the end-to-end protection (E2E). The E2E Protection supports both OEM specific implementations of the AUTOSAR E2E-Profiles and other, OEM-related E2E-Protections. All functions required for correct E2E calculation are **automatically** installed with a “FL3X Config – OEM” package. Please see also chapter 5.12 for custom E2E profiles.

	NOTICE
	The E2E-Profiles are handled for each OEM in a special way because of many differences in the implementation. For the support of the E2E Profiles for an OEM, the corresponding OEM Package must be installed.

	Reference
	For more details see the E2E Protection documentation in the OEM Package.

For projects based on an installed OEM package (project type “%OEM%”), all E2E profiles should be configured automatically when a database is loaded for a bus. Open the “Bus Configuration” dialog and switch to the tab “E2E Protection” to see all (automatically) configured E2E profiles (an example is shown in Figure 55)

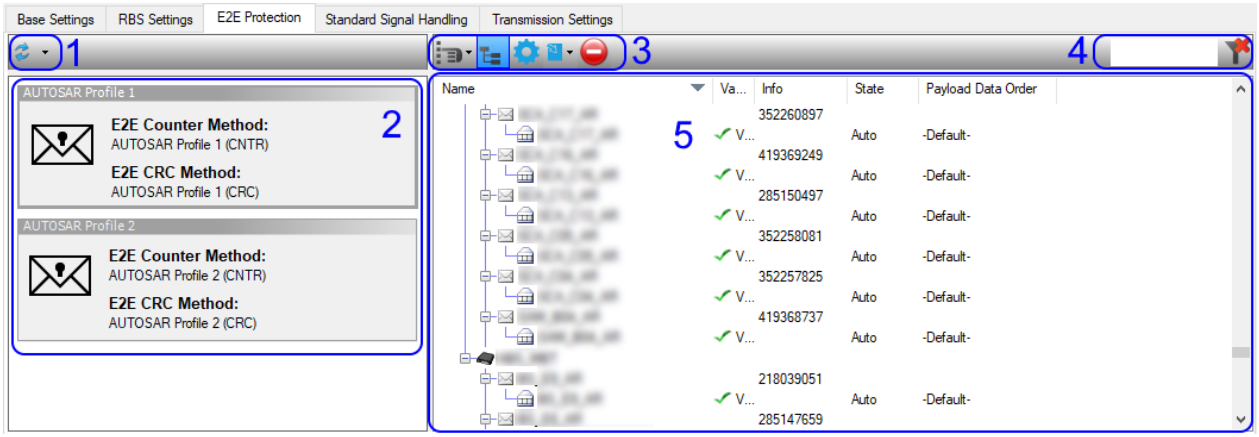


Figure 55: Bus Configuration – Tab “E2E Protection”

The description of the elements in this tab is shown in Table 51.

No	Description
1	Research the E2E Protected elements. This functionality will discard the current E2E configuration and perform a research in the same way as new bus configurations. If there are multiple OEM packages installed, the OEM implementation can be selected.
2	The available OEM dependent E2E-Profiles.
3	The Toolbar to - select the element types to show in the tree view - edit and change the E2E-configuration of a protected element (See chapters 5.6.4.1 and 5.6.4.2)
4	Text filter to search for items in the tree view.
5	Tree view with all PDUs/signal groups of the current bus configuration and the mapping to used/detected E2E profiles. Open the “Edit Protection Dialog” by double click at the protected element. Change/Assign a E2E profile by Drag&Drop the element (PDU/signal group) to one of the E2E profiles (section 2).

Table 51: Bus Configuration – tab “E2E Protection”

5.6.4.1 Edit the E2E Protection

For the most use cases, all information is available from the database and editing is not required. For some reasons it may be helpful to edit the E2E protection of an element.

Edit a protected Element

Some databases have additional PDUs or signal groups which have wrong or no E2E settings so that no DATA-ID is set. Sometimes the databases contain wrong or outdated information. In other cases, the DATA-IDs must be changed for some reason. And sometimes it is required to change the found CRC or SQC signal or to disable one of calculations (SQC/CRC).

All this thigs can be done by editing the E2E protected element (PDU/signal group).

Select the element to edit and use the “Edit” menu or double click at the protected element to open the “Edit protected Element” dialog.

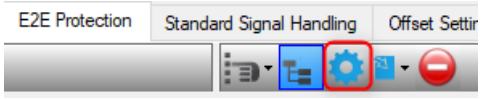


Figure 56: Tab “E2E Protection” – Edit a protected element

Assign an element to a specific E2E profile

In some cases, it is required to assign an element (PDU/signal group) to an E2E profile (e.g. the element is protected but the protection was not found for some reason.

To assign an element to an E2E Profile Drag&Drop the element to the profile. The “Edit protected Element” dialog will appear. After closing the dialog via [OK] button, the element is assigned to the profile.

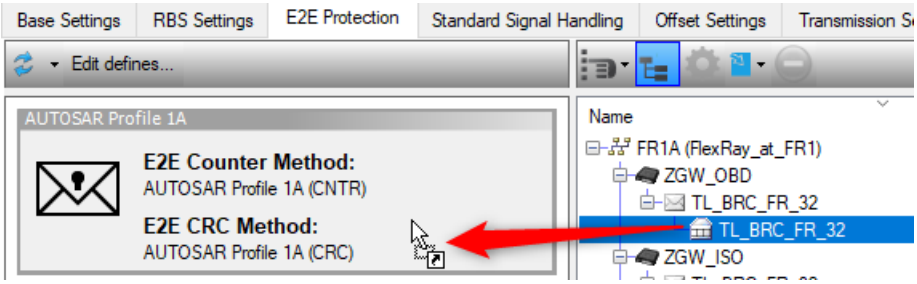


Figure 57: Tab “E2E Protection” – Assign an element to a E2E profile

Remove a protected element

To remove a protected element from a E2E Profile (disable the calculation) there are two different ways.

1. Disable the SQC and CRC calculation via “Edit E2E Protection” or by using the context menu for a protected element.

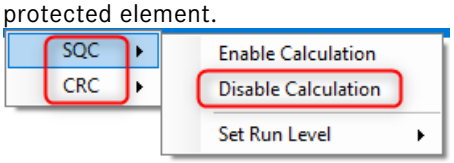


Figure 58: Tab “E2E Protection” – Context menu of an E2E protected Element

2. Remove the assignment to the E2E profile by using the “Delete” button.



Figure 59: Tab “E2E Protection” – Delete E2E profile assignment from protected element

The “Edit protected Element” Dialog

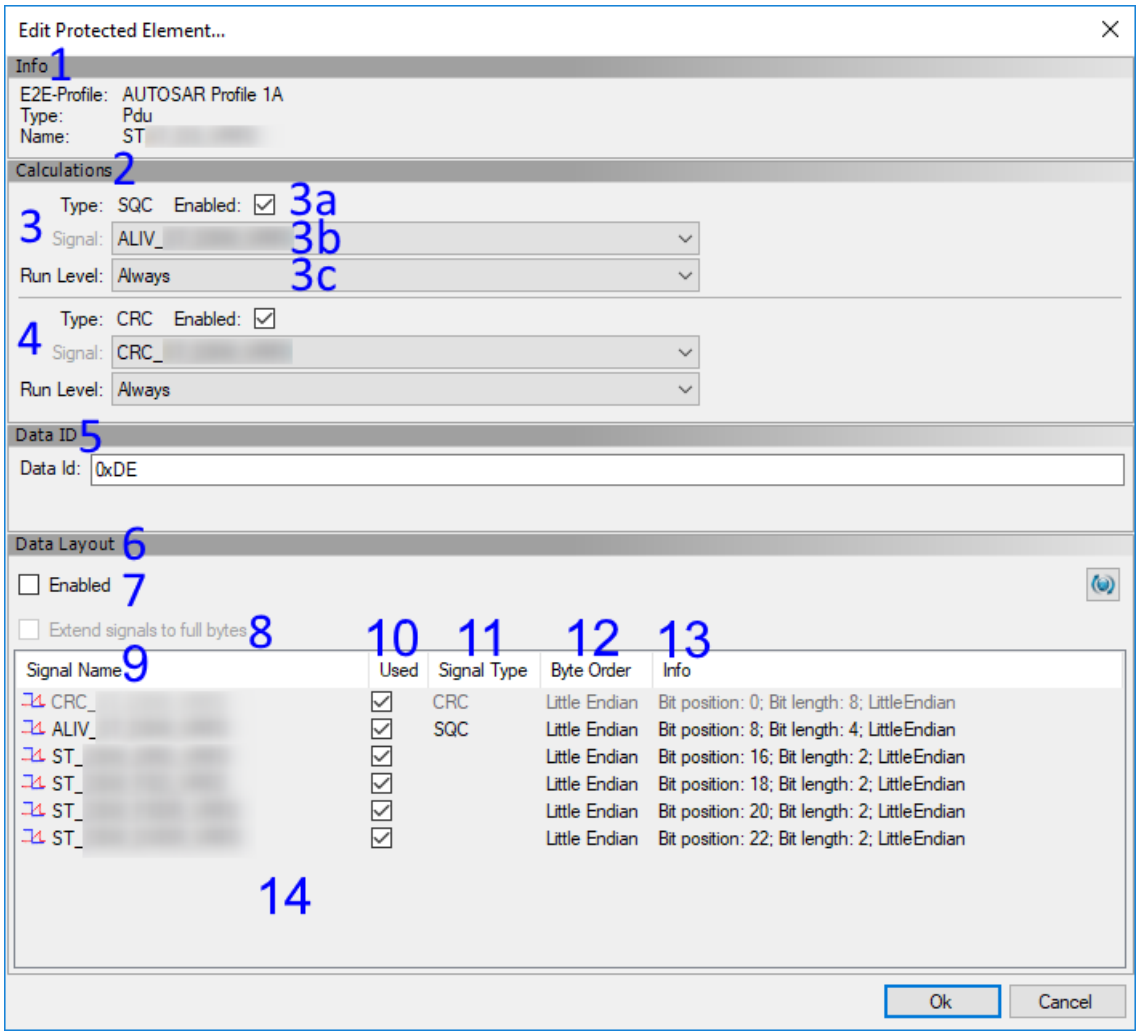


Figure 60: Tab “E2E Protection” – “Edit Protected Element” dialog

No	Description
1	Information about the protected element • Used E2E Profile • Element type (PDU, signal group, …) • Element short name
2	List of the performed calculations and their settings
3	Settings of the SQC calculation
3a	Enable/Disable the calculation A disabled calculation means, that the selected signal will not be affected by the E2E-feature and can be set via routing, User Function, etc.
3b	The target signal of the calculation. This signal value is set to the calculated value. When the signal is set by other features (routing, User Function, etc.) this value will be overwritten with the value of this calculation.
3c	Using the parameter “Run Level”, conditions can be set to trigger the E2E-Profile calculation (see chapter 5.6.5.4). If a PDU Update Bit and/or a Signal Group Update Bit are present, these are also considered for the E2E calculation depending on the payload used. E2E calculation will be triggered according to the set “Run Level” if the following conditions are satisfied: If the E2E calculation uses the **PDU payload** - PDU has no Update Bit - PDU has an Update Bit with the value “1” If the E2E calculation uses the **Signal Group payload**: - The signal group has no Update Bit, the PDU has no Update Bit - Thes signal group has an Update Bit with the value “1”, the PDU has no Update Bit - The signal group has no Update Bit, PDU has an Update Bit with the value “1” - Thes signal group has an Update Bit with the value “1, PDU has an Update Bit with the value “1” If a gateway routing is existent for the E2E signal itself or the appropriate PDU/Frame, then this routing will only be applied for the “Run Level” conditions “Never”, “Manipulation active” or “Manipulation active & Calculation correct”. In the other two cases, the routing will not be applied.
4	Settings of the CRC calculation (same behavior as 3; see 3a – 3c).
5	The used DATA-ID for the E2E (CRC) calculation.
6	The settings for the data layout used for the CRC-calculation. The byte array for the CRC calculation will be filled with the specified signal order (data layout).
7	Enables he special data layout. If the data layout is disabled, the CRC will be calculated about the payload of the protected element (PDU/signal group). The CRC signal itself is excluded from the payload of the CRC calculation. This is because the CRC itself must not be included to the CRC calculation since the CRC signal is the target of the calculation.
8	Extends each signal which is added to the payload of the CRC calculation to full bytes. When disabled, the Signals are attached directly to each other with no bits between.
9	The column signal name shows the short name of the signals of the protected element
10	Includes/excludes a signal from the payload of the CRC calculation. Only included (checked) signals will be assigned to the payload for the CRC calculation. Excluded signals will not lead to not set spaces in the payload. The signals after an excluded signal will follow directly after the signal before the excluded one. CRC signals are implicitly excluded (see point 7)
11	Shows the signal type (SQC/CRC) as defined in the calculation above.
12	Option to select the used byte order of the signal in the CRC payload.

13	Shows the “start bit” and length or the “Sequence number” of the signal in the PDU or signal group according to the setting of point 9.
14	This list shows all signals contained in the protected element (PDU/signal group) and the used order of the signals in the payload for the CRC calculation. To change the order, Drag & Drop the signals to the correct position.

Table 52: Tab “E2E Protection” – “Edit Protected Element” dialog

5.6.4.2 Import Functionality

The E2E Protection feature provides different import functionalities. Import means that the E2E configuration will be updated with the settings from the selected file.

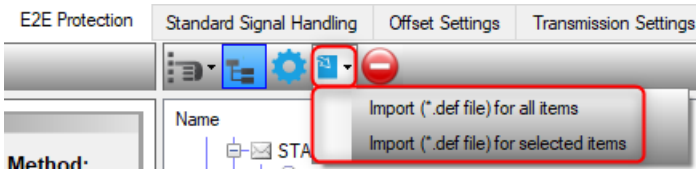


Figure 61: Tab “E2E Protection“ – Import menu

Import (*.def file)

Imports the DATA-ID and Data Layout (order of the signals to calculate the CRC) from the specified define file. See chapter 5.6.4.3 for more information about define files

5.6.4.3 Define Files (*.def)

The define files (file extension *.def*) used with some OEM Packages in the previous versions (until version 5.1) for E2E via Standard Signal Handlings are no longer considered and therefore have no impact on the generated RBS.

This define files where available to specify e.g. the DATA-IDs (CRC-IDs) per signal group

The define file can be imported into the E2E configuration which means that the E2E configuration will be updated with the settings from the define file (see chapter 5.6.4.2)

Define file format

The define file is a normal text file with the file extension *.def. The define file can contain following information.

1. DATA-ID

`#define %ElementName% %DATA-IDs%`

2. A special data layout
   (order of signals for the CRC calculation)

`#define %ElementName%_DataElements %OrderedSignalNames%`

`%ElementName%`

- The short name I protected Elements (PDU/signal grp, ...)
- The defined element name can also start with an optional **`pdu_`**`%ElementName%`
- The short name of the element can also start with **`pdu_`** and the Element will be found during import

`%DATA-IDs%`

- One or more DATA-IDs (comma separated, hex-Format). E.g.

  `0x1F`
  or
  `0x01, 0xA7, 0xFF`

  The count of the DATA-IDs must match the expected DATA-ID count of the E2E profile of the protected element.

`%OrderedSignalNames%`

- The signal short names in the order to set in the Data Layout (comma separated).

- Each ‚signal name' can also be a unique part of the real signal short name.
  (unique according to all signals of the protected element (PDU/signal group)

5.6.4.4 E2E Protection via Standard Signal Calculation

The feature in previous versions (up to version 5.1) to calculate E2E (CRC and Counter) via Standard Signal Handlings is no longer supported. Integrate a user defined CRC calculation via Standard Signal handlings is not possible. User defined CRC calculation must be implemented via User Functions Editor.

All previous made E2E configurations (via Standard Signal Handlings) will be automatically changed to the current E2E feature.

5.6.5 Standard Signal Handling

NOTICE
With release 5.2 the Standard Signal Handling is no longer supported for CRC-calculation. The CRC calculation is now configured via E2E-Profile manager (see chapter 5.6.4). User defined CRC calculation must be implemented via User Functions Editor.

The calculation of standard signals can be configured within the “Standard Signal Handling” tab of the “Bus Configuration” window. To support these calculations predefined C-functions can be used. A C library (*.c* / *.h* file) can be imported to assign the C-functions to specific signals.

The dialog shown in Figure 62 can be opened by clicking the *”Edit bus configuration...”* button of a connector within the FL3X Config main window.

NOTICE
Standard Signal Handling is not supported for signals >32-bit (see chapter 2.8.3). If Standard Signal handling is configured for signals >32-bit anyway, then the build process will abort with an error.

NOTICE
The name of standard signal functions should **not** begin with “set_”!

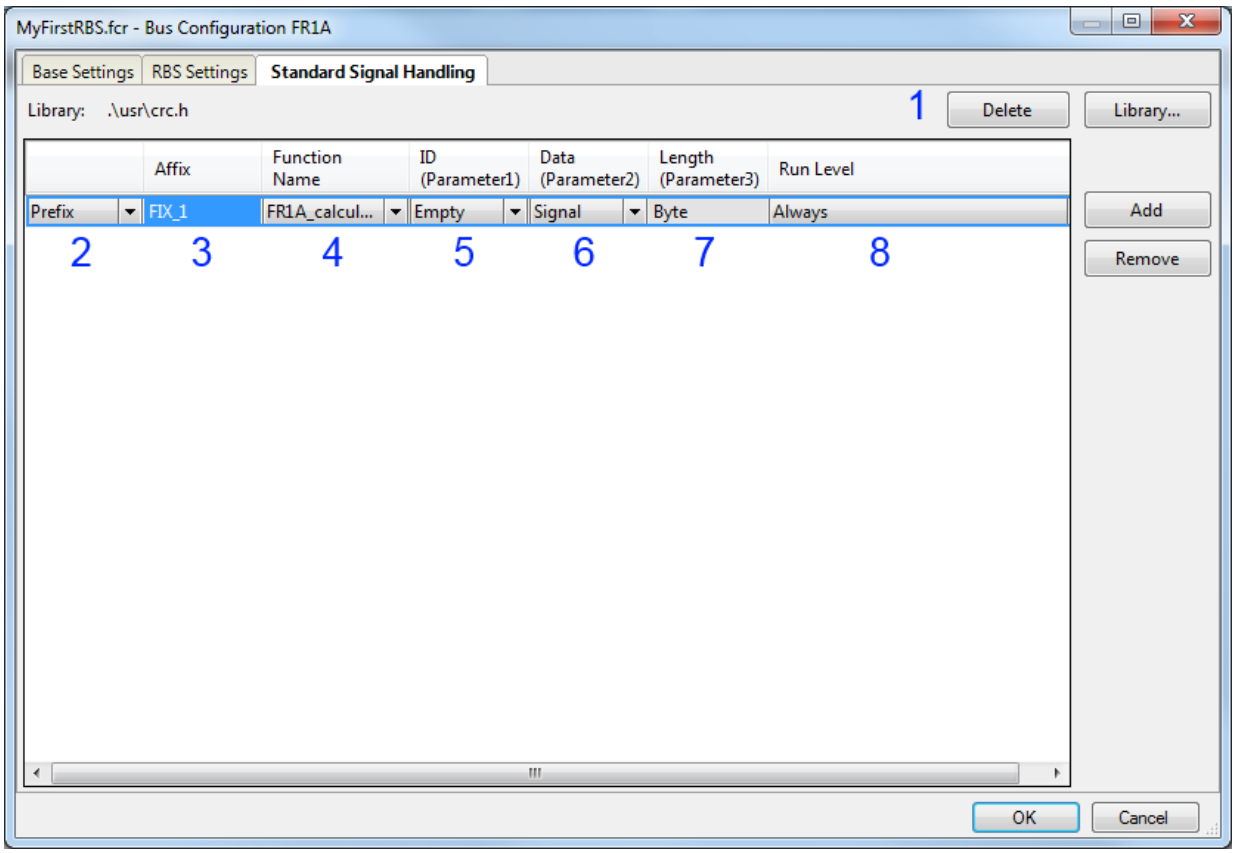


Figure 62: Bus Configuration – “Standard Signal Handling”

No	Description
1	The selected user library is shown here. The functions of this library will be available for Standard Signal Handling.
2	Here the type of affix (prefix or postfix) can be selected. The affix type specifies if the short name of the appropriate standard signal starts (prefix) or ends (postfix) with the affix.
3	The affix of the standard signal can be entered here. A standard signal calculation will only be applied if the short name of a signal contains the affix at the beginning (prefix) or end (postfix).
4	Here a standard signal function can be selected which is contained in the user library, which has previously been added.
5	The kind of information which is used as first parameter in the standard signal function can be specified here. There are three different options available: • Frame ID: CAN ID, LIN frame ID or FlexRay Slot ID • Empty: always zero
6	Here the content of the data array (see the function description in chapter 5.6.5.1), which is provided as second parameter to the standard signal function, can be selected. The data array is always the complete type (e.g. the complete PDU) including the affected signal. There are four different options available: • Frame • PDU • Signal • Signalgroup
7	This Parameter cannot be set. It represents the third parameter of the Standard Signal Function and is the length of the appropriate data array in byte.

No	Description
8	Using the parameter “Run Level”, conditions can be set to trigger the E2E-Profile calculation (see chapter 5.6.5.4). If a PDU Update Bit and/or a Signal Group Update Bit exists, they will also be considered for the E2E calculation. Only when all the existing PDU and Signal Group Update Bits are set to 1 the E2E calculation will be triggered according to the set “Run Level”. If a gateway routing is existent for the E2E signal itself or the appropriate PDU/Frame, then this routing will only be applied for the “Run Level” conditions “Never”, “Manipulation active” or “Manipulation active & Calculation correct”. In the other two cases, the routing will not be applied.

Table 53: Bus Configuration – “Standard Signal Handling”

5.6.5.1 Library Files

Adding a library means adding two files, a header (*.h) and a source code-file (*.c). By clicking on the *‘Library…’* button the dialog showing the included functions is shown. To select a library, the button *‘Select File…’* must be clicked. The RBS will copy the files to the project and no link to the original folder will be kept.

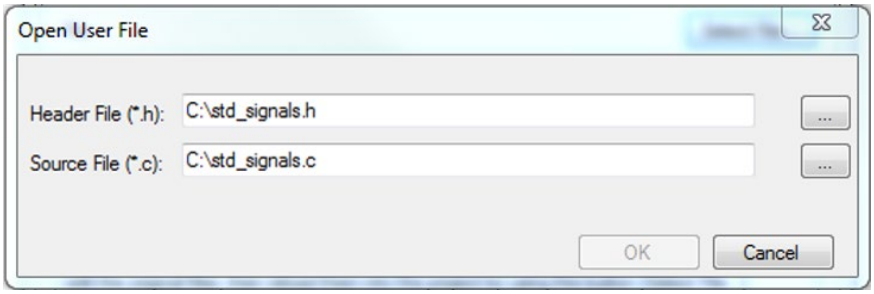


Figure 63: Bus Configuration – “Standard Signal Handling” – “Open User File” dialog

In this dialog user files containing functions for the standard signal calculation can be added.

	NOTICE
	It is not allowed to add a library file with the same name as a “User File” in the User Function Editor (see chapter 5.8.3.4.4). A warning message will be displayed in this case and the file will not be added.

The algorithms of these functions are implemented as ANSI-C functions. FL3X Config expects these functions in a certain definition of parameters and return values. Please refer to the provided example file *std_signals.h* and *std_signals.c*.

Definition of functions for the standard signal calculation	
Function declaration	This is the prototype of the function: `u32 function(u32 value, u08 *data, u32 length);` Example: `u32 FR_calculateValue(u32 value, u08 *data, u32 length);`

Table 54: Bus Configuration – “Standard Signal Handling” – function prototype

Key explanations:

1. function: The name of the user defined function for the standard signal calculation.
2. “u32 value”: This value depends on the configured Standard Signal Handling function, see Table 53 for more details.
3. “u08 *data”: Pointer to the data array on which the signal calculation takes place.
4. “u32 length”: Length in bytes as decimal value representing the number of bytes on which the signal calculation takes place.

NOTICE
The used functions must not contain the ‘`inline`’ keyword. Using this keyword will result in compilation errors.

Information
There are example files included in the delivered software package.

5.6.5.2 Affix Assignment

Defined affixes (prefixes or postfixes) can be used to assign a function to signals. If the affix of the signal short name matches with the assignment the function is called before the signal is transmitted.

So, for each signal short name beginning/ending with this prefix/postfix the user defined function is executed before the frame including the signal is sent. E.g.: select “Prefix” as the affix type and type in “abc_” into the text field *“Affix”* to perform a calculation for all signal short names beginning with the prefix “abc_”.

5.6.5.3 Execution order

A maximum of eight (8) functions can be assigned for standard signal calculation at the same time. The execution order of these functions is defined by their order in the “Standard Signal Handling” table. This means that the entry at the top of the “Standard Signal Handling” table will be executed first and the entry at the bottom at last. Each function assigned to the table can use the current values of standard signals calculated by preceding functions.

5.6.5.4 Run Level

The “Run Level” is used to set the trigger conditions for Standard Signal Handling calculation or E2E-Profile calculation.

Definition of the Run Levels	
Never	The calculation is never executed.
Manipulation active	The calculation is executed if another signal of the same PDU has been manipulated before.
Manipulation active & Calculation correct	The calculation is executed, if two conditions are fulfilled at the same time (conjunction). First condition: Another signal of the same PDU has been manipulated before. Second condition: All standard signals (or E2E signals) with the same affix (same pattern) of the appropriate received PDUs must be correct. If there are no matching signals within the received PDU, this condition is always fulfilled. Received PDUs: If the affected PDU is target of multiple routings (Gateway), then all source PDUs are considered. E.g. PDU_A (Sig_A1) is routed to PDU_C (Sig_C1) and PDU_B (SigB2) is routed to PDU_C (C2). And these PDUs are secured via E2E-CRC. Then the E2E-CRC of PDU_A AND the E2E-CRC of PDU_B must be correct to fulfil the second condition.
Calculation correct	The calculation is executed, if all standard signals (or E2E signals) with the same affix (same pattern) of the appropriate received PDUs must be correct. If there are no matching signals within the received PDU, this condition is always fulfilled. Received PDUs: See the description at Run Level “Manipulation active & Calculation correct”.

Definition of the Run Levels	
Always	The calculation is always executed if the signal itself was not manipulated before. This is the default configuration.

Table 55: Bus Configuration – “Run Levels”

NOTICE

All Run Levels with the condition “Calculation correct” affect always the complete PDU. If a PDU contains several signal groups, an invalid CRC in one of the signal groups will mark all signal groups (and the PDU itself) as “Calculation **not** correct”.

This may lead to wrong CRCs for **all** signal groups of the PDU at the TX side of a gateway.

5.6.6 Offset Settings (FlexRay)

The “Offset Settings” tab of the “Bus Configuration” dialog is shown in Figure 67 . Here a time/cycle-based offset between the actual configured FlexRay Bus (Slave) and another configured FlexRay Bus (Master) can be defined.

If an offset is already defined in the “Bus Configuration” dialog of the Master, then it is not possible to define an additional offset for the corresponding Slave.

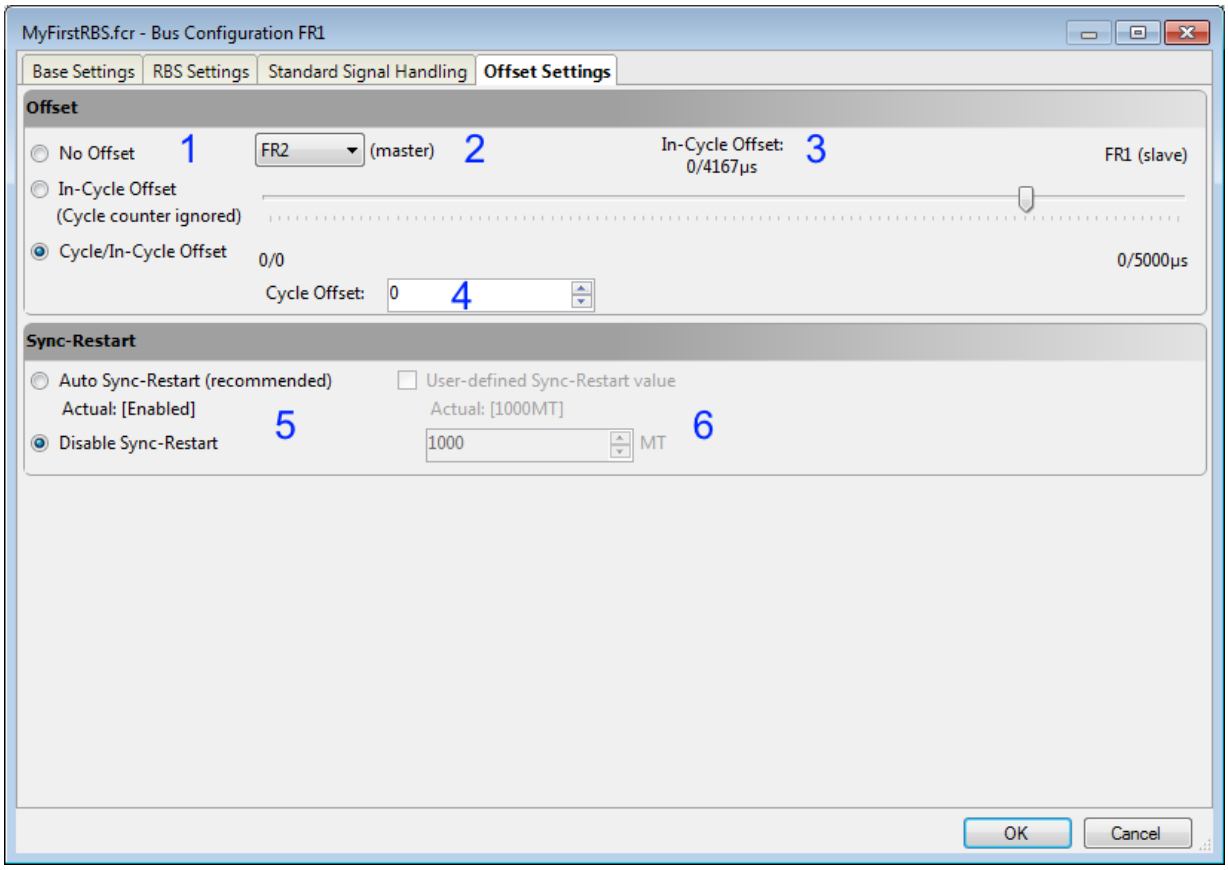


Figure 64: Bus Configuration – “Offset Settings”

The elements, of this dialog, are described in Table 56.

No	Description
1	To define an offset there are three options available: • No Offset: There is no offset defined for the corresponding FlexRay Bus. Nevertheless, it is possible that an offset is defined by the corresponding Master. If this offset is also not defined, then there is no synchronization between the two busses and In-Cycle Offset and Cycle Offset are not used. • In-Cycle Offset: Only the In-Cycle Offset is used. The Cycle Offset is not used. • Cycle/In-Cycle Offset: Both In-Cycle Offset, and Cycle Offset are used. More information about this option can be found in chapters 5.6.6.1 and 5.6.6.2.
2	In this drop-down box the corresponding Master can be set.
3	Here the slider can be set to define the time-based offset between the FlexRay Master- and the Slave bus. E.g. if the Offset is set to 0/4167 µs then the Slave has a delay of 4167 µs to the Master. Be careful with this option – in case of a gateway and if the configured offset is too short, the hardware may not be able to route the configured frames.
4	An additional Cycle Offset can be defined here. If the Cycle Offset is set to a certain value *n*, then the Slave bus has a delay of *n* cycles to the Master Bus. E.g. if *n* is 3 and the current cycle of the Master is 8 then the resulting cycle of the Slave is 5. A Cycle Offset of '0' (default) means, that the two FlexRay Busses are regulated to the same cycle.
5	Enables/Disables the 'SyncRestart' feature of the cluster synchronization. This means, that the corresponding bus can be re-synchronized to achieve the expected offset. Sync-Restart cannot be enabled if the number of external sync-frames is equal or more than 2.
6	If FlexRay cluster synchronization is active and the absolute difference between the configured offset and the measured offset is equal or greater than this value and the Sync-Restart option is enabled, then the synchronization will be restarted.

Table 56: Bus Configuration – "Offset Settings"

The Offset configuration is checked for correctness. If the current configuration is not valid or may cause problems, then one or more warnings are shown. E.g. a circle configuration between three FlexRay busses (FR1 is master of FR2, FR2 is master of FR3, FR3 is master of FR1) is not valid. Such configurations prevent to close the dialog via "*OK*".



NOTICE

The FlexRay Offset configuration does **not** work if the FlexRay parameter 'MacroPerCycle' is different on the selected busses.

5.6.6.1 Effects on FlexRay Bus

The FlexRay offset feature tries to start and hold the configured offset between the two FlexRay clusters. The FlexRay cluster specific properties and the number of sync/startup nodes on each cluster have both influence on the result.

During the "startup phase" of the offset feature, the slave bus is kept in hold-state (e.g. by sending multiple FlexRay CAS) until the master bus has reached the sync-state. Only when the maser bus is in sync-state, sending FlexRay CAS is stopped and it is tried to synchronize the slave bus.

5.6.6.2 Offset Control

5.6.6.2.1 Definition of Offset Control

The 'Offset Control' feature regulates two FlexRay buses. The 'Offset Control' configuration is based on a Master-Slave-Relationship. One FlexRay bus is the master bus, the second FlexRay bus is the slave bus. The regulation tries to maintain the defined offset between the cycle start of the FlexRay bus 1 (Master) and the cycle start of the FlexRay bus 2 (Slave).

A cycle (0-63) as well as the in-cycle duration (usually 4ms to 5ms) can be configured as a defined offset (see also Figure 64).

5.6.6.2.2 For what is Offset Control needed

With offset control, a deterministic behavior can be "forced" between two FlexRay buses.

5.6.6.2.3 Use Cases for Offset Control

Projects that contain two (or more) FlexRay buses with routings between those busses are a use case in which offset control is required to have deterministic routing behavior.

A typical example for this is the transparent gateway (see also chapter 6.2).

Projects that are configured only as remaining bus simulations (without any routings between FlexRay busses) do not require offset control.

5.6.6.2.4 Where can Offset Control be configured?

In FL3X Config there are two dialogs where offset control can be configured:

1) Transparent Gateway Dialog -> Offset Settings

2) Bus Configuration Dialog -> Offset Settings

5.6.6.2.5 Recommended Offset Control Parameters

In about 95% of all use cases, the following (default) configuration is appropriate:
InCycle Offset: 0/4167
Cycle Offset: 0

For all other use cases, project-specific adjustments are necessary (e.g. there are routing errors at runtime).

5.6.6.2.6 Additional Features related to Offset Control

The BUSLOCK feature ensures that a FlexRay bus is kept asynchronous by permanently sending CAS (Collision Avoiding Symbols) on the bus.

This prevents external SYNC startup ECUs with the leading cold starter option from starting the FlexRay bus at a time that would be unfavorable for the configured offset control.

5.6.6.2.7 Required Conditions for Offset Control

The following conditions are required for Offset Control:

The FL3X Device needs to control the FlexRay synchronization state of the configured slave bus. This leads to the following preconditions:

a. A maximum of one real Sync/Startup ECU should be connected to the slave bus.

b. FL3X Device must be allowed to determine the time at which the bus starts up/synchronizes. The FL3X Device needs 2x FlexRay Communication Controllers (CC, e.g.: Bosch E-Ray) on the slave bus for this.

Notes: In some cases when the FlexRay bus on the master side does not regulate much, it can work with only a single Communication Controller on the slave bus.

5.6.6.2.8 Check the Offset Control status at runtime

There are two ways to check the offset control status at runtime:

1) The status of Offset Control is displayed on the FL3X Diagnostics Website (see chapter 8) (tab ‘Self Check’).

2) Projects that have configured Offset Control visualize the status via the green user LED of the FL3X Device (green LED on = Offset Control in range; green LED off = Offset Control is **not** in range)

5.6.6.2.9 A typical scenario for Offset Control

1.) The FL3X Device starts and initializes itself.

2.) The FlexRay bus (master) is configured and tries to start/synchronize.
- If the master bus can be started independently (Self-Sync), it is immediately synchronized and put into the “synchronous” state (green bus LED).
- If the bus cannot be started up automatically, it is switched to the “Start Up” state (green and red bus LED).
- The FlexRay bus (slave) is configured and set to “Hold” (red bus LED).
- The BUSLOCK feature keeps the asynchronous and blocks external start-up attempts.

3.) The FlexRay bus (master) notifies the FlexRay bus (slave) as soon as the time has come at which the FlexRay bus (slave) must start up to maintain the configured offset (offset control).

The FlexRay bus (slave) is switched from the “Hold” state to the “Synchronous” state and runs with a configured offset to the FlexRay master bus.

4.) At runtime, the FlexRay (slave) bus regulates itself to the FlexRay (master) bus.

5.6.6.2.10 Recommended offset control configuration related to FPGA images

Note: The following table lists only images with two or more FlexRay busses.

FPGA Image	Limitation	Recommendation
Type 0x60	-	Con1 -> Offset Control Master Con3 -> Offset Control Slave
Type 0x61	Con1 has only 1xCC	Con1 -> Offset Control Master Con3 -> Offset Control Slave
Type 0x6B	Con1 has only 1xCC	Con1 -> Offset Control Master Con3 -> Offset Control Slave
Type 0x65	Con1 has only 1xCC	Con1 -> Offset Control Master Con3 -> Offset Control Slave
Type 0x90/0x80	The internal connection for Offset Control is **not** available between Con2 and Con3	Con1A -> Offset Control Master Con1B -> Offset Control Slave
Type 0x92/0x80	-	Con2A -> Offset Control Master Con2B -> Offset Control Slave
Type 0x94/0x8A	Con1 has only 1xCC	Con2A -> Offset Control Master Con2B -> Offset Control Slave
Type 0x96/0x85	Con2A has only 1xCC Con2B has only 1xCC	Con2A or Con2B -> Offset Control Master Con1 -> Offset Control Slave
Type 0x92/0x8D	Con1B has only 1xCC	Con2B -> Offset Control Master Con2A -> Offset Control Slave
Type 0x9B/0x183	- The internal connection for Offset Control is **not** available between Con4 and Con3 and for Con4 and Con1. - For Con1 and Con3, both CCs must be configured	Con1 -> Offset Control Master* Con3 -> Offset Control Slave* *see Limitation
Type 0x94/0x83	Con2A has only 1xCC	Con2A -> Offset Control Master Con2B -> Offset Control Slave
Type 0x94/0x89	Con1A has only 1xCC	Con1A -> Offset Control Master Con1B -> Offset Control Slave
Type 0x94/0x8B	Con1B has only 1xCC	Con1B -> Offset Control Master Con1A -> Offset Control Slave
Type 0x9B/0x18E	For Con1 and Con3, both CCs must be configured	Con1 -> Offset Control Master* Con3 -> Offset Control Slave* *see Limitation
Type 0x93/0x82	The internal connection for Offset Control is **not** available between Con2 and Con3 and for Con2 and Con4.	Con3 -> Offset Control Master Con4 -> Offset Control Slave
Type 0x97/0x86	Con3 has only 1xCC Con4 has only 1xCC	Con3 or Con4 -> Offset Control Master Con2 -> Offset Control Slave
Type 0x9C/0x184	-	Con3A -> Offset Control Master Con3B -> Offset Control Slave
Type 0x9D/0x186	Con1A has only 1xCC	Con1A -> Offset Control Master Con1B -> Offset Control Slave
Type 0x9D/0x187	The internal connection for Offset Control is **not** available between Con1 and Con3	Con1A -> Offset Control Master Con1B -> Offset Control Slave

FPGA Image	Limitation	Recommendation
Type 0xB0	Con2 has only 1xCC	Con2 -> Offset Control Master Con1 -> Offset Control Slave

Table 57: Recommended Offset Control configuration for FPGA-Images

5.6.7 Transmission Settings

The “Transmission Settings” display all the relevant information of transmitted elements and allow the user to change them globally. Theses settings take precedence over User Functions. There are two ways to access the “Transmission Settings”, either through ‘*Edit -> Transmission Settings*’ (for all busses) or through the “*Transmission Settings*” – Tab in the Bus Configuration (only for the configured bus).

NOTICE

Elements that do not require a specific timing configuration, such as LIN or SOME/IP provided methods, are not shown in the “Transmission Settings”.

NOTICE

Transmission settings cannot be changed for the items, which are used in byte-based Frame/Container routings or in a time sync configuration.

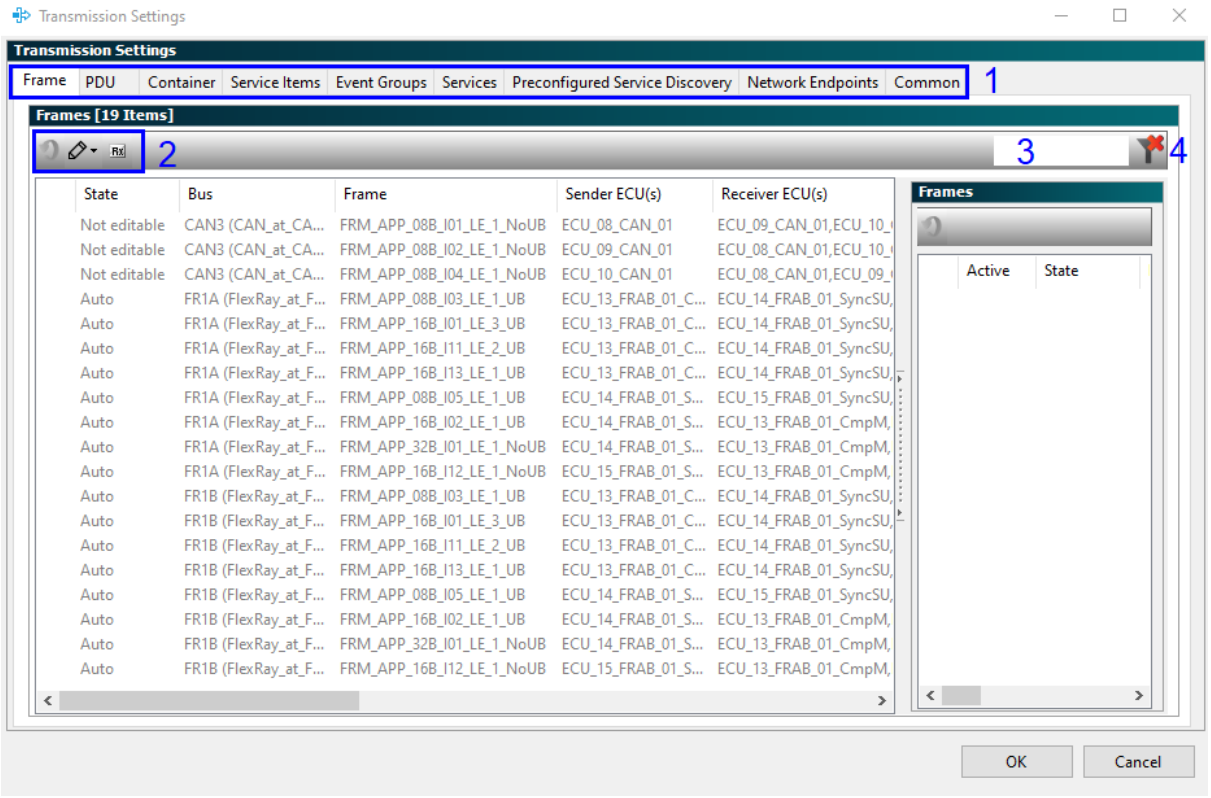


Figure 65: Transmission Settings – “Frame” Tab

No	Description
1	The **Frame**, **PDU**, **Container**, **Service Items**, **Event Groups**, **Services**, **Preconfigured Service Discovery**, **Network Endpoints and Common** tabs allow the user to select the category.
2	The toolbar of each tab may have different buttons, which can be used to configure the transmission settings.
3	This search bar is a text filter. Any matching text will be highlighted and all elements that do not have the matching text will be filtered out.
4	The *“Clear Text Filter”* – button clears the text filter. All elements are shown.

Table 58: Transmission Settings

The following tables describe the parts of the “Transmission Settings” dialog.

Common Settings		
Toolbar		
	The *"Reset Selected Items"* – button sets the selected items to the default value from the database. If the database has no value for the corresponding property, then a default value may be used which is defined in other dialogs of FL3X Config. E.g. the default multicast IP-Addresses can be set via Bus Config dialog.	
	The *"Edit Selected Items"* – button displays all the editable fields of a transmission in a drop-down menu. An item can be edited via this menu or via context menu. Editable fields have a black font, while non-editable fields are greyed out.	
Columns		
State	Info	Indicates if a transmission has been edited. Possible states are: **Auto**, **Edited**, and **Not editable**.
Bus	Info	Bus type and name
Sender ECU(s)	Info	Displays the name(s) of the Sender ECU(s)
Receiver ECU(s)	Info	Displays the name(s) of the Receiver ECU(s)
Frame	Info	Displays the name of the Frame
Frame Info	Info	Displays the Frame Id for CAN Frames, Slot, Base Cycle and the Repetition for the FlexRay Frames
Routing Type	Info	Indicates whether any routing to the corresponding item exists. **Not Routed:** No routing exists **Child Routing:** At least one child item is used by a routing **Parent Routing:** The parent item is used by a routing If the item itself is used by a routing, the **routing type** will be shown (E.g. 'Byte Based PDU Routing')

Table 59: Transmission Settings – Common Settings

5.6.7.1 Frame Transmission Settings

Frame	
Toolbar	
	The *"Edit Selected Items"* – button have following sub elements: **Frame Tx Behavior:** This button is enabled if FlexRay frames are selected. In this case 2 options (PDUTriggered & Absolute) are provided **Frame Timing**: This button is enabled if the selected frames do not contain any PDU. The timing of the selected frames can be set. **Used IP**: This button is enabled for ethernet frames. **Used Port**: This button is enabled for ethernet frames.
	The buttons "*Show Rx Frames"* show receive frames in the list of frames.

Columns		
PDU Count	Info	Number of PDUs in frame
PDU Trigger Mode	Info	Displays the Trigger Mode of the PDUs. Possible trigger modes are **Cyclic**, **Routing**, **CyclicAndRouting**, **Never**, and **Mixed**.
Frame Tx Behavior	Editable	This column is editable, if a FlexRay frame is selected. In this case 2 options (PDUTriggered & Absolute) are provided **PDUTriggered**: The PDU transmission settings determine, how the frame is sent. This is the default setting for all frames. **Absolute**: The FlexRay frame will be set with its absolute timing (E.g. FlexRay frame with base cycle 0 and repetition 4 will be sent all 20 ms as data frame with a cycle time of 5ms).
Frame Timing	Editable	This column is editable if the selected frame does not contain any PDU. In this case the timing of the selected frame can be set. In other cases, this column shows the greatest common divider of the PDU timings and is not editable. Please consider the absolute timing of a FlexRay frame, which does not contain a PDU (E.g. if the absolute timing of the frame is 5ms, then the frame timing can be set as 5ms, 10ms, 15ms etc.). The maximum supported timing for such FlexRay frames is 64x FlexRay cycles. E.g. if the FlexRay cycle length is 5 ms, then maximum supported timing is 320 ms. If a larger timing is needed, User Functions can be used. E.g. The required frame timing is 1.000ms -> set the frame timing to 100 ms and disable the frame transmission 9 of 10 times in the corresponding Frame Transmit User Function.
Database Source IP	Info	Displays the database IP Address of an ethernet frame.
Used Source IP	Editable	Displays the used IP Address of an ethernet frame.
Database Source Port	Info	Displays the database port of an ethernet frame.
Used Source Port	Editable	Displays the used port of an ethernet frame.
Direction	Info	Displays "*Tx*" for frames which are sent and "*Rx*" for frames which are received.
Receiver Count	Info	Displays the number of receive points of this frame. The first number is the active receive points count and the second number is the overall receive point count of this frame in the following format: *ACTIVE / COUNT*

Table 60: Transmission Settings – "Frame" Settings

The frame list has on the right side another list for the ethernet destination configuration, which shows the corresponding information, if any ethernet frame is selected. For *tx* frames this configuration defines to which destinations the frame should be sent and for *rx* frames it defines the receive points over which the frame is received. The destination configuration list contains following columns:

Frame Destination		
Columns		
Active	Editable	Enables or disables sending the frame to this destination
Receiver ECU	Info	Name of the receiver ECU.
Database Destination IP	Info	Displays the database destination IP Address of the destination receiver ECU.
Used Destination IP	Editable	Displays the used destination IP Address of an ethernet frame.
Database Multicast IP	Info	Displays the database destination Multicast IP Address of the destination receiver ECU.
Used Multicast IP	Editable	Displays the used destination Multicast IP Address of an ethernet frame.
Database Destination Port	Info	Displays the database destination port of an ethernet frame.
Used Destination Port	Editable	Displays the used destination port of an ethernet frame.

Table 61: Transmission Settings – "Frame Destination" Settings

NOTICE
Activating multiple receive points for a received frame, may receive a single frame instance multiple times. This occurs for example when both receive points are using the same multicast IP address to which the frame is sent.

5.6.7.1.1 Ethernet Frame IP Configuration Overview

	Where to configure?	Tx Frame	Rx Frame
Source MAC Address	Network Endpoints	Set value of the “Used MAC Address” column of the corresponding endpoint entry of source type “*Tx Frame*” to configure the MAC address over which the frame is sent.	Not configurable
Destination MAC Address	Network Endpoints	Not configurable	Set value of the “Used MAC Address” column of the corresponding endpoint entry of source type “*Rx Frame*” to configure the MAC address over which the frame is received.
Source IP Address	Frame Transmission Settings	Set value of „Used Source IP Address“ column to configure the IP address over which the frame is sent.	Not configurable
Destination IP Address	Frame Transmission Settings	Select the tx frame and use the “Receiver” view on the right side to configure the receivers to which the frame should be sent.	Select the rx frame and use the “Receiver” view on the right side to configure the receiver points over which the frame is received.

Information
The frame view shows the conflict of source and destination IPs. All views where an IP address is configured are considered.

5.6.7.2 PDU Transmission Settings

PDU	
Toolbar	
	The *“Edit Selected Items”* – button have following sub elements: • Frame Trigger Mode • PDU Update Behavior • Used Cyclic Timing • Used Alternative Timing • Offset • Used Debounce • Initial Timing Behavior • Used Collection Timeout You can find more information by the corresponding column definitions below.

	The *"Calculate Contained Offset"* – button calculates the offset of contained PDUs, so that they fit into the container with the appropriate cycle time. Following options are provided: **Override database offsets:** Calculates the offsets of the contained PDUs without considering the database offsets. **Keep database offsets**: The offsets, which are already defined in the database will retain. The offsets will be calculated for the contained PDUs, which do not have any defined offset. **Reset to database offsets**: The calculated offsets will reset. Database offsets will be used.
	The *"Quick Setup"* – button can be used to easily change the settings of the selected transmissions. Currently, it only has one function, which sets the triggering mode of routed items to "Routing".

Columns		
Frame Tx Behavior	Info	Shows the information, how the parent frame is sent
PDU	Info	Name of the PDU
PDU Type	Info	Displays the PDU type. Following PDU types can be shown: • Standard • Update Bit (for standard and contained PDUs) • Secured • TimeSync • Contained Last-Is-Best • Contained Queued • Static • Switched • No Header (for contained PDUs)
Frame Trigger Mode	Editable	The Trigger Mode of the frame can be edited here. Possible trigger modes are: **Cyclic**: The frame should be sent cyclic with the timing of the PDU **Routing**: The frame should be sent just by routings **CyclicAndRouting**: The frame should be sent cyclic with the timing of the PDU and when the routing takes place **Never**: The frame should not be sent
PDU Update Behavior	Editable	This column is editable just for the PDUs with Update Bits or contained PDUs. It has the same options as "Frame Trigger Mode" and determines, when the Update Bit of the PDUs should be set or when the contained PDUs should be added to the container.
Database Cyclic Timing	Info	This column shows the smallest cyclic timing of the PDU, which is saved in the database.
Used Cyclic Timing	Editable	The used cyclic timing for the PDU in milliseconds. This overrides the default setting from the database. The 'Initial Timing Behavior' property sets whether the 'Used Cyclic Timing' or the 'Used Alternative Timing' is initially used for the PDU.
Database Alternative Timing	Info	This column shows the fastest cyclic timing of the PDU, which is saved in the database.
Used Alternative Timing	Editable	The alternative cyclic timing for the PDU in milliseconds. This overrides the default setting from the database. The 'Initial Timing Behavior' property sets whether the 'Used Cyclic Timing' or the 'Used Alternative Timing' is initially used as for the PDU.
Initial Timing Behavior	Editable	This property sets whether the 'Used Cyclic Timing' or the 'Used Alternative Timing' is initially used as cyclic timing of the PDU. The cyclic timing of a PDU can be switched via user Functions 'Set Default Timing' and 'Set Alternative Timing' (see Table 75)

Base Timing Factor	Editable	This property is used for the calculation of the real timing of the PDU, if the “Set Custom Timing” function is used for the PDU in User Function Editor. The value of the custom timing is rounded down to a n-time value of ‘Base Timing Factor’. Example: If a custom timing of 37ms is used in User Function Editor with the function “Set custom Timing” and the ‘Base Timing Factor’ is configured as 10ms, then the PDU timing will be 30ms. If the ‘Base Timing Factor’ is configured as 5ms, then the PDU timing will be 35ms.
Database Offset	Info	The default initial delay, which applies once at start-up. This can be used to offset one of two signals with the same period.
Used Offset	Editable	The used initial delay. This overrides the default setting from the database.
Database Debounce	Info	The default minimum period between messages. The message will be discarded if the defined time is not elapsed.
Used Debounce	Editable	The used minimum period between messages. This overrides the default setting from the database. This column is currently not editable for FlexRay PDUs and contained PDUs. The “Selfcheck” page of the FL3X Diagnostics Website (see chapter 8) shows the number of items which are not lost because of debounce times. Please consider that the device internal clock is used for the time validation and not the PTP time Synchronization or GTS (Global Time Manager).
Container Name	Info	Displays the name of the container.
PDU Header	Info	Displays the header ID of the contained PDUs
Database Collection Timeout	Info	Displays the time to wait before an ethernet container is sent (database value).
Used Collection Timeout	Editable	The used collection timeout. This overrides the setting from the database. This value can be edited only for Ethernet PDUs. All Ethernet PDUs (except PDUs that are transmitted over Port 30500) are collected in containers with this configured timeout.

Table 62: Transmission Settings – “PDU” Settings

5.6.7.3 Container Transmission Settings

Container		
Toolbar		
	The *“Edit Selected Items”* – button have following sub elements: • Used Threshold • Used Timeout You can find more information by the corresponding column definitions below.	
Columns		
Container Name	Info	Displays the name of the container
Database Threshold	Info	Display the threshold value, which is defined in the database. The container PDU is sent automatically if the threshold byte size is reached. (E.g. The max size of a contained is 64 bytes, if the threshold value is 60 bytes, then it will be sent as soon as the total size of actual contained PDUs reach or exceed 60 bytes)
Used Threshold	Editable	The used threshold value. This overrides the default setting from the database.
Database Timeout	Info	Displays the timeout value, which is defined in the database. The timeout value determines the max time to wait after the first contained PDU is added.
Used Timeout	Editable	The used timeout value. This overrides the default setting from the database

Contained Type	Info	Displays the type of the contained PDUS. Following values are possible: **Queued**: All contained PDUs have type “Queued”. A ‘Queued’ contained PDU will be added to the container one more time if its payload is updated. **Last-Is-Best**: All contained PDUs have type “Last-Is-Best”. A ‘Last-Is-Best’ contained PDU cannot exist multiple times in a contained. Each payload update is done on the added instance. **Mixed**: At least one contained PDU has the type “Queued” and at least one contained PDU has the type “Last-Is-Best”. Tooltips can be used to see the type of each contained PDU.
PDU Update Behavior	Info	Shows the information, how the contained PDUS are triggered. Possible trigger modes are **Cyclic**, **Routing**, **CyclicAndRouting**, **Never**, and **Mixed**.
Frame Tx Behavior	Info	Shows the information, how the parent frame is sent.

Table 63: Transmission Settings – “Container” Settings

5.6.7.4 Service Items Transmission Settings

	NOTICE
	Transmission settings are supported for provided events, provided notifier, and consumed methods

Service Items		
Toolbar		
	The *“Edit Selected Items”* – button have following sub elements: • Item Trigger Mode • Used Cyclic Timing • Used Offset • Used Debounce • Used Collection Timeout • Initial Tx • Initial Timing Behavior • Used Collection Timeout You can find more information by the corresponding column definitions below.	
	The *“Quick Setup”* – button can be used to easily change the settings of the selected transmissions. Currently, it only has one function, which sets the triggering mode of routed items to “Routing”.	
Columns		
VLAN	Info	VLAN ID
Provider ECU	Info	Displays the provider ECU of the service instance
Consumer ECU(s)	Info	Displays all consumer ECUs of the service instance, which are defined in the database
Consume As	Info	Displays which ECU is configured to consume the service item.
Direction	Info	Indicates if a service item is **Consumed** or **Provided**.
Service	Info	Name of Service
Item Name	Info	Displays the name of the item.
Item Type	Info	Displays the relevant item type. This can either be an **Event**, **FieldNotifier**, or a **ConsumedMethod**. All other item types are not shown, as they are not relevant for the “Transmission Settings” (see Notice).
Routing Type	Info	Shows if the item is used in configured routing
Item Trigger Mode	Editable	The Trigger Mode of the item can be edited here. Possible trigger modes are **Cyclic**, **Routing**, **CyclicAndRouting**, and **Never**.

Database Cyclic Timing	Info	The default period between messages in milliseconds.
Used Cyclic Timing	Editable	The used period between messages in milliseconds. This overrides the default setting from the database.
Database Alternative Timing	Info	Alternative cyclic timing from database
Initial Timing Behavior	Editable	Selected used timing behavior from startup (Default or Alternative)
Used Alternative Timing	Editable	Cyclic period between messages in milliseconds which are used if alternative timing is enabled (via User Function Editor).
Database Debounce	Info	The default minimum period between messages. The message will be discarded if the allotted time has not elapsed.
Used Debounce	Editable	The used minimum period between messages. This overrides the default setting from the database.
Database Offset	Info	The default initial delay, which applies once at start-up. This can be used to offset one of two signals with the same period.
Used Offset	Editable	The used initial delay. This overrides the default setting from the database.
Database Collection Timeout	Info	Displays the time to wait, before an ethernet container packet is sent.
Used Collection Timeout	Editable	The used collection timeout. This overrides the default setting from the database.
Initial Tx	Editable	Determines, whether the events should be automatically sent after the event group is subscribed.

Table 64: Transmission Settings – “Service Items” Settings

5.6.7.5 Event Group Transmission Settings

Event Groups		
Toolbar		
	The *“Edit Selected Items”* – button have following sub elements: • Used IP Address • Used Port • Used Multicast Threshold You can find more information by the corresponding column definitions below.	
Columns		
Bus	Info	Bus Name
VLAN	Info	VLAN ID
Sender ECU	Info	Displays the provider ECU of the service instance
Consumer ECU(s)	Info	Displays all consumer ECUs of the service instance, which are defined in the database
Consume As	Info	Displays which ECU is configured to consume the event group
Direction	Info	Indicates if the event group item is **Consumed** or **Provided**.
Service	Info	Name of the service
Service Instance	Info	Displays the service instance ID of the provided service instance.
Database Source IP Address	Info	The source IP-Address of the event group which is defined in the database.
Used Source IP Address	Editable	The used IP-Address of the event group. It can be a unicast or a multicast IP-Address for a consumed event group and only unicast for a provided event group. The allowed IP-Address type (IPV4/IPV6) depends on the used IP type in the database. See also the (*) Runtime behavior paragraph.
Database Source Port	Info	The source port of the event group which is defined in the database.
Used Source Port	Editable	The used port of the event group. See also the (*) Runtime behavior paragraph.

Database Destination IP Address	Info	The destination IP-Address of the event group which is defined in the database.
Used Destination IP Address	Editable	The used destination IP-Address for a provided event group. This field is not editable for consumed event groups, and it can be only a multicast IP-Address for the provided event groups.
Database Destination Port	Info	The destination port of the event group which is defined in the database.
Used Destination Port	Editable	The used destination port of the event group. It is editable only for provided event groups.
Database Threshold	Info	Database threshold value
Used Threshold	Editable	The used threshold value (editable for just provided event groups)

(*) Runtime behavior for the “Used IP Address” and “Used Port” columns:

For consumed services:
This address is always used as the destination address for Service Discovery (e.g. Subscribe). Unicast address and multicast address have the same behavior.

For provided services:
- If this address is a unicast address, then it is used as source address when sending the event group (events).
- If this address is a multicast address, then it will be used as the destination address when the "Used Threshold" value is reached. In this case, the unicast address of the service is always used as the source address.

5.6.7.6 Services Transmission Settings

Services		
Toolbar		
	The *“Edit Selected Items”* – button have following sub elements: • SOME/IP SD Mode • Used IP Address • Used Port • Used SD Unicast IP • Used SD Multicast IP • Used SD Unicast Port • Used SD Multicast Port • Used SD Min. Initial Delay • Used SD Max. Initial Delay • Used SD Repetition Delay • Used SD Repetition Count • Used SD Cyclic Delay • Used SD TTL • Used Service Interface Version You can find more information by the corresponding column definitions below.	
Columns		
VLAN	Info	VLAN ID
Sender ECU	Info	Displays the provider ECU of the service instance
Consumer ECU(s)	Info	Displays all consumer ECUs of the service instance, which are defined in the database
Consume As	Info	Displays which ECU is configured to consume the service item.
Direction	Info	Indicates if the service item is **Consumed** or **Provided**.
Service	Info	Name of the service
Service Instance	Info	Displays the service instance ID of the provided service instance.

SOME/IP SD Mode	Editable	The following values are available for provided services: • **Routing**: Before provided services are offered, they must be actively consumed on the other Gateway side (at least on event group must be subscribed or for services without event groups, at least one offer must be received). • **StandAlone**: SOME/IP Service Discovery is processed independent according to its configuration. For consumed services, the value is always "-" and cannot be changed.
Database IP Address	Info	The IP-Address of the service which is defined in the database.
Used IP Address	Editable	The used IP-Address of the service.
Database Port	Info	The port of the service which is defined in the database.
Used Port	Editable	The used port of the service.
Database SD Unicast IP	Info	The unicast IP-Address for Service Discovery which is defined in the database.
Used SD Unicast IP	Editable	The used unicast IP-Address for Service Discovery
Database SD Unicast Port	Info	The unicast port for Service Discovery which is defined in the database.
Used SD Unicast Port	Editable	The used unicast port for Service Discovery. A warning will be shown if the expected default SD port 30490 is not used.
Database SD Multicast IP	Info	The multicast IP-Address for Service Discovery which is defined in the database.
Used SD Multicast IP	Editable	The used multicast IP-Address for Service Discovery
Database SD Multicast Port	Info	The multicast port for Service Discovery which is defined in the database.
Used SD Multicast Port	Editable	The used multicast port for Service Discovery. A warning will be shown if expected default SD port 30490 is not used.
Database SD Min. Initial Delay	Info	Database value of minimum initial delay in milliseconds before SOME/IP SD will start to offer or find services
Used SD Min. Initial Delay	Editable	Used value of minimum initial delay in milliseconds before SOME/IP SD will start to offer or find services
Database SD Max. Initial Delay	Info	Database value of maximum initial delay in milliseconds before SOME/IP SD will start to offer or find services
Used SD Max. Initial Delay	Editable	Used value of maximum initial delay in milliseconds before SOME/IP SD will start to offer or find services
Database SD Repetition Delay	Info	Database value of start delay between repetitions in milliseconds
Used SD Repetition Delay	Editable	Used value of start delay between repetitions in milliseconds
Database SD Repetition Count	Info	Database value of number of repetitions
Used SD Repetition Count	Editable	Used value of number of repetitions
Database SD Cyclic Delay	Info	Database value of delay between cyclic offers in milliseconds
Used SD Cyclic Delay	Editable	Used value of delay between cyclic offers in milliseconds
Database SD TTL	Info	Database value of lifetime (Time to live) of SOME/IP SD offer or subscription
Used SD TTL	Editable	Used value of lifetime of SOME/IP SD offer or subscription
Database Service Interface Version	Info	Database value for interface version used inside SOME/IP SD messages

Used Service Interface Version	Editable	Used value for interface version used inside SOME/IP SD messages

Table 65: Transmission Settings – “Services” Settings

5.6.7.7 Preconfigured Service Discovery

The "SOME/IP Preconfigured Service Discovery" feature allows to simulate SOME/IP consumers for provided services. There are some limitations for this feature:

1) For each configured “Preconfigured Service Discovery” element, the provided service starts transmitting its events and field notifications directly after the project started (normally directly after PowerOn). The SOME/IP Service Discovery handling, that is normally required, is ignored for the configured elements.

2) The transmission of events/field notifications requires that the MAC address of the target IP-Address is known.
For a unicast IP-Address, use the Static ARP Table Configuration feature described in chapter 5.6.2.3.2.
For a multicast IP-Address, no extra configuration is necessary.

3) The TTL parameter is not used and has no expiration date. The TTL parameter **cannot** be overwritten at runtime. In particular, any SOME/IP Service Discovery message received at runtime by a real ECU (with the same IP-Address and Port) will **not** change any parameter of the simulated SOME/IP consumer.

4) The simulated consumer **cannot** be stopped at runtime (even if a real ECU with the same IP-Address and Port sends a StopSubscribe command).

5) The transmission of events/field notifications will continue after every link change (link down -> linkup).

Preconfigured Service Discovery		
Toolbar		
	Opens a dialog to select a provided service and add a “Preconfigured Service Discovery” configuration for it.	
	Deletes the selected items	
	The *“Edit Selected Items”* – button have following sub elements: • IP Address • Port You can find more information for each column below.	
Columns		
Bus	Info	Bus Name
VLAN	Info	VLAN ID
Service	Info	Name of Service
Service Instance	Info	Service Instance value
IP Address	Editable	The IP-Adress of the simulated consumer.
Port	Editable	The port of the simulated consumer. Default is 30491
Transport Protocol	Info	Used transport protocol (UDP/TCP)
Sender ECU	Info	Displays the provider ECU of the service instance
Consumer ECU(s)	Info	Displays all consumer ECUs of the service instance, which are defined in the database

Table 66: Transmission Settings – “Preconfigured Service Discovery” Settings



NOTICE

The IP-Adress of the simulated consumer must be different to the IP-Adress of the Provider.



Information

It is possible to configure multiple simulated consumers for the same provided service if each simulated consumer uses a different IP Address/Port.

5.6.7.8 Network Endpoints

FL3X Config configures automatically self-defined network endpoints for services or frames if the corresponding information is missing in the database. In FL3X Config 1.1 or older, those self-defined network endpoints were not transparent to the user.

Since FL3X Config 1.2, the “Network Endpoints” feature allows to add or modify the local network endpoints including the self-defined endpoints.

The network endpoints list is filled with all endpoints based on the configured services, event groups and frames. Any change in these items lead to changes of the network endpoint configuration. Such network endpoints are hereinafter referred to as “auto network endpoints”.

Network Endpoints		
Toolbar		
	Adds a new “custom network endpoint” whose IP and MAC configuration can be set. Custom network endpoints may be needed by using some User Libraries or if the database is incomplete.	
	Deletes the selected custom network endpoints.	
	The *“Edit Selected Items”* – button have following sub elements: • IP Address • MAC Address • Net Mask • Prefix Length • Gateway • ECU Name You can find more information for each column below.	
Columns		
Bus	Info/Editable	Bus Name is editable for custom network endpoints.
VLAN	Info/Editable	VLAN ID is editable for custom network endpoints.
IP Type	Info/Editable	IP Type is editable for custom network endpoints. It is limited to the IP Type of the loaded database of the bus. E.g., if the database uses “IP Type =IPv4”, then an added custom network endpoint can also only be configured as “IP Type = IPv4”.
IP Address	Info/Editable	IP Address is editable for custom network endpoints. Auto network endpoints show the IP addresses from the lists of services, event groups or frames.
Database MAC Address	Info	Shows the database MAC Address for the auto network endpoints.
Used MAC Address	Editable	The used MAC Address.
Database Net Mask/Prefix Length	Info	Shows the database net mask for the IPV4 items and the prefix length for the IPV6 items.
Used Net Mask/Prefix Length	Editable	Shows the used net mask for the IPV4 items and the prefix length for the IPV6 items.
Database Gateway	Info	Shows the database gateway address.
Used Gateway	Editable	Shows the used gateway address and is editable for IPv4 endpoints
ECU Name	Info/Editable	ECU Name is editable for custom network endpoints. Auto end points show the name of first found ECU.
Source	Info	Shows the source of the end point or ‘Custom’ for the custom network endpoints.

Table 67: Transmission Settings – “Network Endpoints” Settings

It is also possible to change the network endpoint configuration of the PC-Ethernet connector. This bus name will be available if a custom endpoint item is added. Do not use the configured IP Address of a device for any custom network endpoint configuration with the PC-Ethernet Connector. In case of conflicts (device IP is the same as the IP used for a custom endpoint configuration), the custom network endpoint

configuration for the PC-Ethernet Connector is ignored and only the device settings are used. The device IP configuration can be changed via device manager in main window of FL3X Config (see chapter 5.2.2).

5.6.7.9 Common Transmission Settings

Common settings which are relevant to all transmission settings:

- See chapter 5.6.7.9.1

5.6.7.9.1 Performance Optimization

Transmission settings can be optimized automatically during configuration time to improve performance. The optimization can only be applied to all items of all busses. The selected optimization is applied by clicking the “Optimize” button.

Performance Optimization Options		
Optimizations	**Description**	**Note**
PDU - Offset	The used offset values of cyclic PDU items are evenly distributed to reduce bursts.	This will only affect cyclic CAN and Ethernet PDUs.
PDU – Collection Timeout	Sets or increases the used collection timeout value to reduce ethernet frame count of cyclic items.	This will only affect cyclic Ethernet PDUs.
Service Item - Offset	The used offset values of cyclic Service Items are evenly distributed to reduce bursts.	This will only affect cyclic SOME/IP Service Elements.
Service Item - Collection Timeout	Sets or increases the used collection timeout value to reduce ethernet frame count of cyclic items.	This will only affect cyclic SOME/IP Service Elements.
Optimization Target Items	**Description**	
Optimize default values	Only items that have the default value for the optimized property are optimized. Already edited values will not be affected.	
Optimize all – Overwrite edited values	All items will be optimized. Already edited values are overwritten.	

Table 68: Description of performance optimization options in Transmission Settings

5.7 User Defines

To edit the User Defines, open the dialog via the menu (see chapter 5.4.2.4).

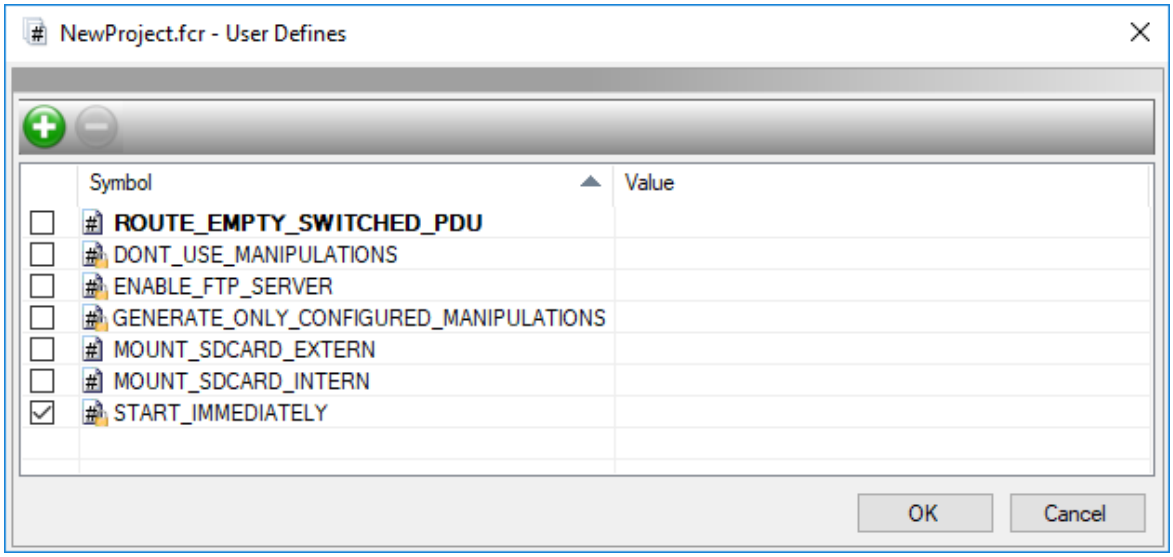


Figure 66: User Defines

User Defines are C-code Defines for the code which is running on the appropriate device (FL3X Device-S, FL3X Device-L). The user can add, remove, enable, or disable defines and change the optional define values. These defines will be included to the compilation. Defines can be used to enable special functionality within the hardware.

Defines can also be used in User Functions (see chapter 5.8.2.3).

Click on  to add a new define to the list. The checkbox next to define decides, if the define is activated in the current project or not.

The following table describes all defines which are created automatically.

User Define Name	Description	Values	Effect
ROUTE_EMPTY_SWITCHED_PDU	Enables/Disables routing of empty switched PDUs	0 1	Disabled Enabled
DONT_USE_MANIPULATIONS	If this define is set all manipulations are disabled. The RBS/Gateway Code does not contain any code parts for manipulations. This define can be used to reduce the code size of a project.	-	Disables all manipulations
ENABLE_FTP_SERVER	Enables/Disables a FTP Server on the hardware platform to access content of the mounted drives (e.g. SD-Card).	-	
GENERATE_ONLY_CONFIGURED_MANIPULATIONS	If this define is set only the configured manipulations are considered for code-generation. With this define set the code size of a project can be reduced if it is too large. When using FL3X Config Control or the PC-Hardware Interface only these manipulations can be applied to the current RBS/Gateway	-	Only configured manipulations are considered
MOUNT_SDCARD_EXTERN	Mounts the external SD-Card (must be formatted as FAT32). Additional control of SD-Cards is available via the provided FL3X Diagnostics Website (see chapter 8) of the used hardware platform.	-	
MOUNT_SDCARD_INTERN	Mounts the internal SD-Card (must be formatted as FAT32). Additional control of SD-Cards is available via the provided FL3X Diagnostics Website (see chapter 8) of the used hardware platform.	-	
START_IMMEDIATELY	This is a default define. It is used to start the RBS/Gateway immediately after a reset.	There is no value for this User Define. If the define is enabled, the mechanism is active. If the define is disabled, the mechanism is not active.	See the columns „Description” and “Values”

Table 69: Automatically created User Defines

NOTICE

Only enabled user defines can be evaluated via the user function editor.

NOTICE

User Defines are case sensitive. To prevent any problems, do not use two or more User Defines with the same name and different case. The behavior of the application cannot be predicted for such a configuration.

5.8 The User Function Editor

To write user functions, which can be integrated into an RBS or gateway, FL3X Config contains a “User Function Editor”. The editor can be opened by clicking on

‘Edit -> User Function Editor’ in the menu bar of FL3X Config.

The appropriate window of the “User Function Editor” is shown in Figure 67.

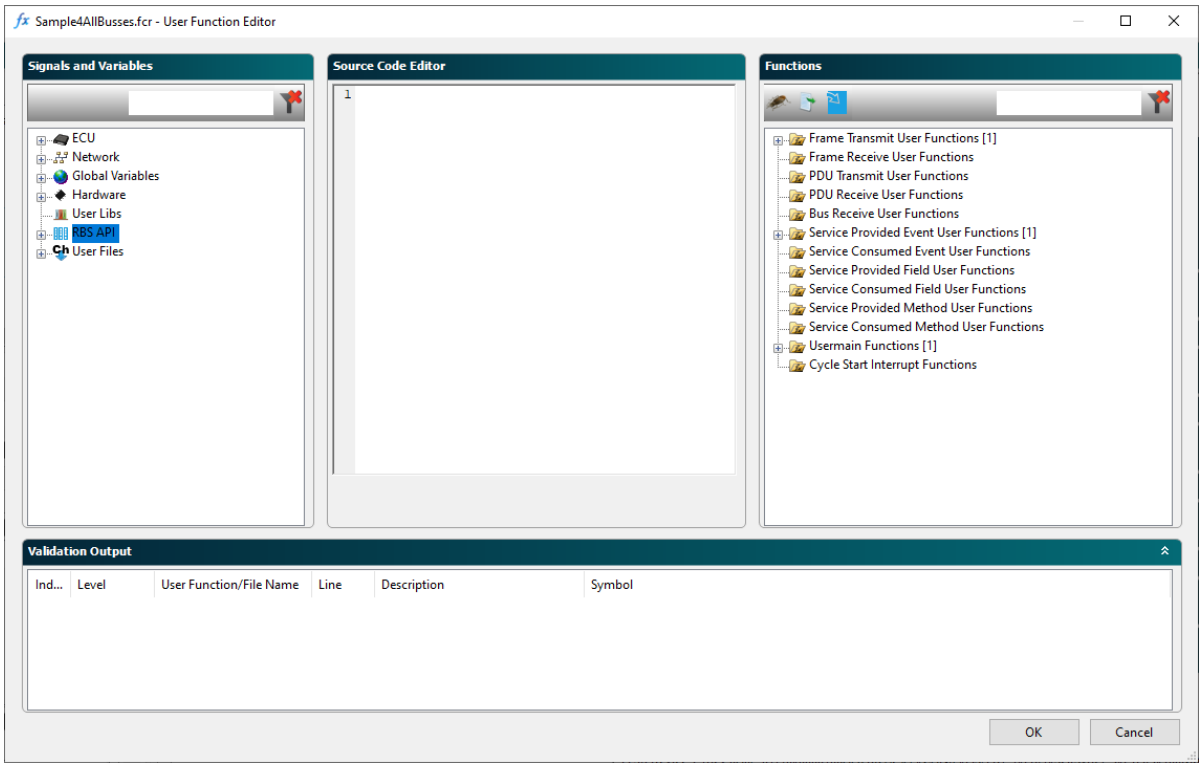


Figure 67: User Function Editor

The “User Function Editor”-window contains four different sections, which are described in more detail in the following chapters.

NOTICE

Exiting the “User Function Editor” by clicking the [OK] button implies that all user functions and added functions are validated!

If an error occurs a dialog opens, which gives the advice to fix all errors!

Otherwise, it is not possible to exit the editor via [OK].

	NOTICE
	The user functions are normally not validated after opening an existing project. In some cases, the functions are changed automatically during loading the project. A message box will be shown in this case to inform the user and make it possible to open the user function editor for validating the functions.

5.8.1 Functions

The “Functions”-section is the section, where all user functions from the RBS/gateway can be selected, created, removed, or validated.

To support different types of events, which trigger the execution of a user function, the “User Function Editor” contains four different types of user functions. These different types of user functions are described in more detail in the following chapters.

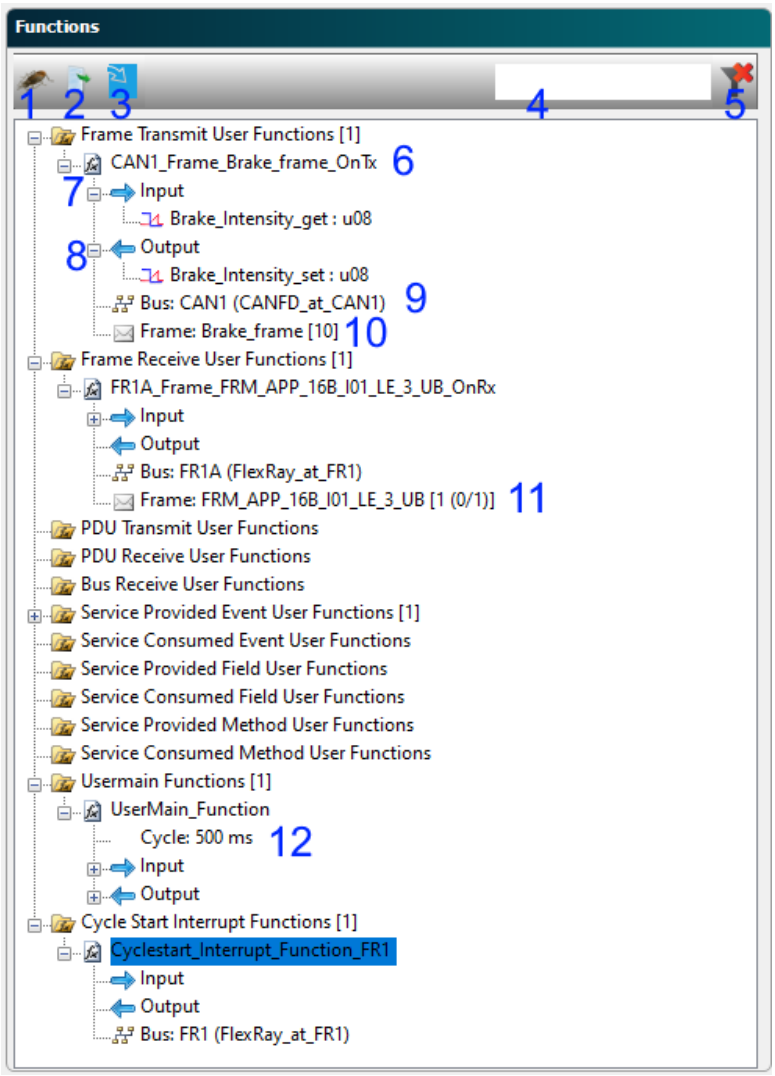


Figure 68: User Function Editor – Section “Functions”

As can be seen in Figure 68 the “Functions”-section contains a tree view where different folders for each type of user function exist. By clicking on the “+”,”-” symbols in front of each folder the appropriate folder can be expanded/collapsed. If the “+” symbol is clicked all child-items, which are appropriate to the type of user function, are shown. These child items again contain child items and so on.

The child items of the different types of user functions and the different buttons within the “Functions”-section are explained in Table 70.

No	Description
1	The *"Validate All"*-button. (s. chapter 5.8.1.16)
2	The *"Export Functions"*-button. (s. chapter 5.8.1.17)
3	The *"Merge User Functions"*-button. (s. chapter 5.8.5)
4	The search field can be used to filter the nodes shown in the function tree with the given text. The function codes are not considered by filtering.
5	Clears the filtering.
6	***f***$_x$: Name of the user function. The default name depending on the user function type is • *bn*_Frame_*sn*_OnTx, type "Frame Transmit User Function" • *bn*_Frame_*sn*_OnRx, type "Frame Receive User Function" • *bn*_PDU_*sn*_OnTx, type "PDU Transmit User Function" • *bn*_PDU_*sn*_OnRx, type "PDU Receive User Function" • UserMain_Function, type "Usermain Function" • Initialize_Function, type "Initialize User Function" • Cyclestart_Interrupt_Function_*bn*, type "Cyclestart Interrupt Function" • *bn*_VLAN*vlan*_Event_*sn*_On_*srvid_srvins* Tx, type "Service Provided Event User Function" • *bn*_VLAN*vlan*_Event_On_*sn_srvid_srvins* Rx, type "Service Consumed Event User Function" • *bn*_VLAN*vlan*_Get_Field_*sn*_On_*srvid_srvins* Rx, type "Service Provided Field User Function" or "Service Consumed Field User Function" • *bn*_VLAN*vlan*_Set_Field_*sn*_On_*srvid_srvins* Rx, type "Service Provided Field User Function" • *bn*_VLAN*vlan*_Notify_Field_*sn*_On_*srvid_srvins* Tx, type "Service Provided Field User Function" • *bn*_VLAN*vlan*_Notify_Field_*sn*_On_*srvid_srvins* Rx, type "Service Consumed Field User Function" • *bn*_VLAN*vlan*_Set_Field_*sn*_On_*srvid_srvins* Tx, type "Service Consumed Field User Function" • *bn*_VLAN*vlan*_Method_On_*sn_srvid_srvins* Rx, type "Service Provided Method User Function" or "Service consumed Method User Function" *bn*_VLAN*vlan*_Method_On_*sn_srvid_srvins* Tx, type "Service Consumed Method User Function" where *bn* is the name of the bus, *sn* the short name of the appropriate frame/PDU/service element, *vlan* the Vlan number, *srvid* the service id and *srvins* the service instance id
7	The **Input** item of a user function can contain further child-items as signals, update bits, frames, PDUs, global variables, service elements, low level or complex (service) data types. These child items deliver information to the user function by using appropriate get-functions.
8	The **Output** item of a user function can contain further child-items as signals, update bits, frames, PDUs, global variables, service elements, low level or complex (service) data types. For these child items information is set by using appropriate set-functions.
9	The **Bus** item shows the name of the appropriate Bus.
10	The **Tx Frame** item shows an identifier which is appropriate to the selected Tx frame. The identifier is the concatenation of the short name and the identifier (CAN, LIN) or absolutely-scheduled timing information (FlexRay).
11	The **Rx Frame** item shows an identifier which is appropriate to the selected Rx frame. The identifier is the concatenation of the short name and the identifier (CAN, LIN) or absolutely-scheduled timing information (FlexRay).
12	The **Cycle** item contains the Cycle Time in milliseconds.

Table 70: User Function Editor – Section "Functions"

There are limitations for PDUs within the User Function Editor (see table below).

PDU Type \ Feature	RX UF	TX UF	Set TX Event	Get PDU	Set PDU
Standard PDU	X	X	X	X	X
Multiplexed PDU	X	X	X	X	X
Dynamic PDU (MUX PDU)	-	-	-	X	X

PDU Type \ Feature	RX UF	TX UF	Set TX Event	Get PDU	Set PDU
Static PDU (MUX PDU)	-	-	-	X	X
Container PDU	-	-	X “Set Container Tx Raw”	X “Get Container PDU Raw” ⇨only for Tx PDUs	X “Set Container PDU Raw” ⇨only for Tx PDUs
Contained PDU	X	X	-	X	X

Table 71: User Function Editor – Limitations for PDUs

5.8.1.1 Frame Transmit User Functions

A “Frame Transmit User Function” can be written for each frame which is sent by the RBS or gateway. The “Frame Transmit User Function” of a frame is called immediately before the frame is sent.

A new “Frame Transmit User Function” can be added by clicking on *‘Add Frame Transmit User Function’* within the context menu (s. Figure 69). The context menu is available by right clicking on the appropriate “Frame Transmit User Functions”-folder.

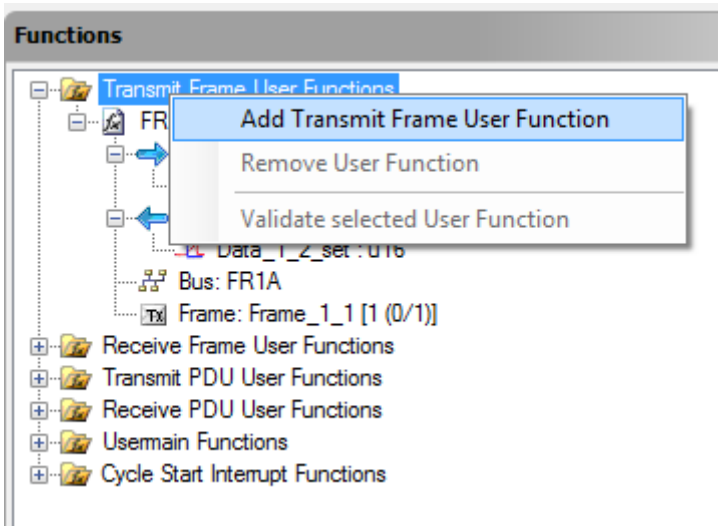


Figure 69: User Function Editor – Add “Frame Transmit User Function”

After this has been done a window appears (s. Figure 70), where the tab of the relevant bus must be selected first. In the next step the frame must be selected for which the function has to be written. Finally, the ‘*Create Frame Transmit User Function’*-button has to be clicked.

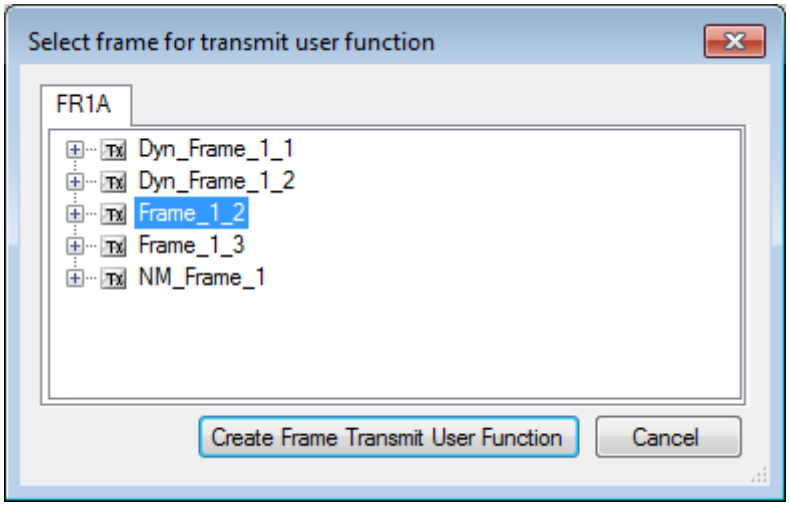


Figure 70: User Function Editor – Frame selection to create a “Frame Transmit User Function”

After that, an empty user function appears in the “Source Code Editor”-section (see chapter 5.8.2.1.1 for more details).

Additionally, the “Frame Transmit User Functions”-folder contains a new user function (s. Figure 68).

A “Frame Transmit User Function” can be removed by clicking on the option ‘*Remove User Function*’ within the context menu. The context menu is available by right clicking on the appropriate “Frame Transmit User Function”.

To validate a function, the option ‘*Validate Selected User Function*’ can be selected within the context menu.

	NOTICE
	Each Frame Transmit User Function (CAN, FlexRay and Ethernet) has an input parameter for every configured E2E profile. By dereferencing and setting the parameter to 0, the corresponding E2E method is disabled.

5.8.1.2 Frame Receive User Functions

A “Frame Receive User Function” can be written for each frame which is received by the appropriate RBS or Gateway. The “Frame Receive User Function” of a frame is called immediately after the frame is received.

A new “Frame Receive User Function” can be added by clicking on *‘Add Frame Receive User Function’* within the context menu (s. Figure 71). The context menu is available by right clicking on the “Frame Receive User Function” folder.

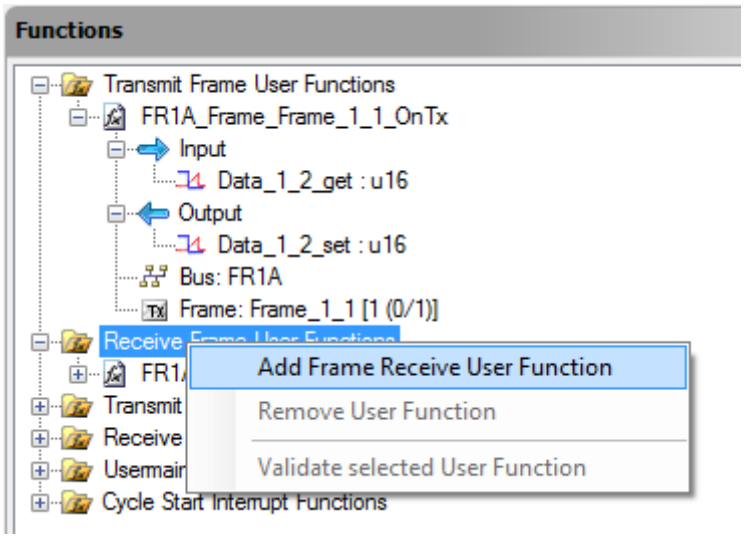


Figure 71: User Function Editor – Add “Frame Receive User Function”

After this has been done a window appears (s. Figure 72), where first the tab of the relevant bus must be selected. In the next step the frame must be selected for which the function has to be written. Finally, the “*Create Frame receive user function”*-button has to be clicked.

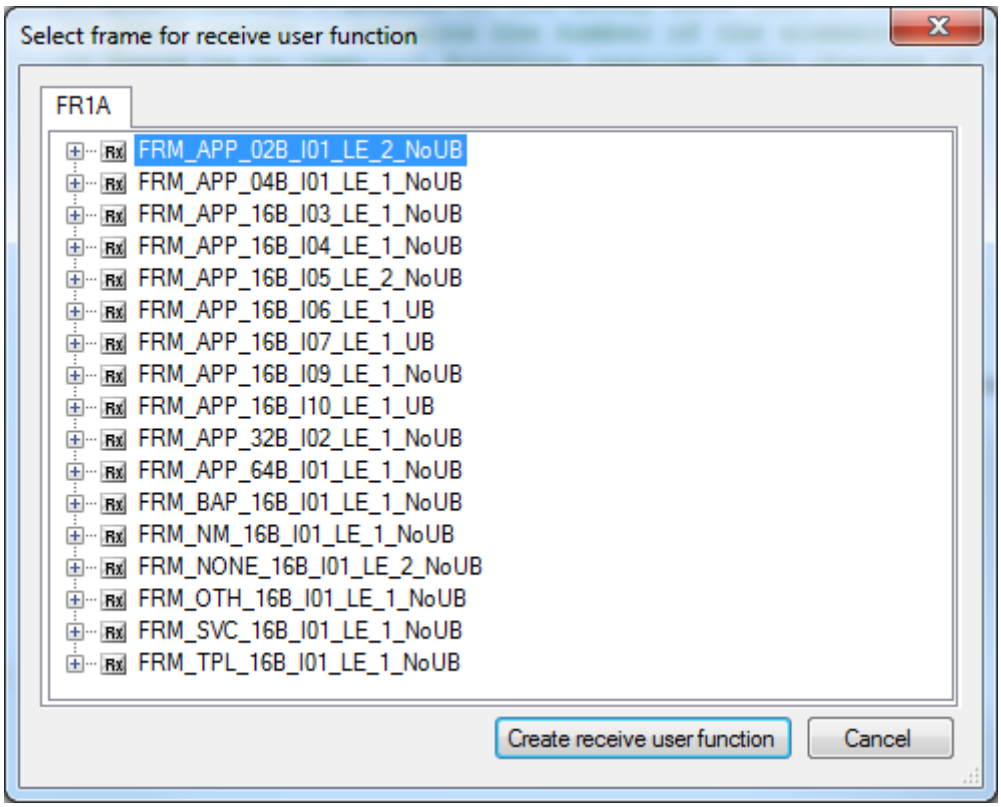


Figure 72: User Function Editor – Frame selection to create a “Frame Receive User Function”

After that, an empty user function appears in the “Source Code Editor”-section (see chapter 5.8.2.1.2 for more details).

Additionally, the “Frame Receive User Functions”-folder contains a new user function (s. Figure 68).

A “Frame Receive User Function” can be removed by clicking on the option ‘*Remove User Function*’ within the context menu. The context menu is available by right clicking on the appropriate “Frame Receive User Function”.

To validate a function, the option ‘*Validate Selected User Function*’ can be selected within the context menu.

5.8.1.3 PDU Transmit User Functions

A “PDU Transmit User Function” can be written for each PDU which is sent by the RBS or gateway. The “PDU Transmit User Function” of a PDU is called immediately before the PDU is sent.

A new “PDU Transmit User Function” can be added by clicking on *‘Add PDU Transmit User Function’* within the context menu (see Figure 73). The context menu is available by right clicking on the appropriate “PDU Transmit User Functions”-folder.

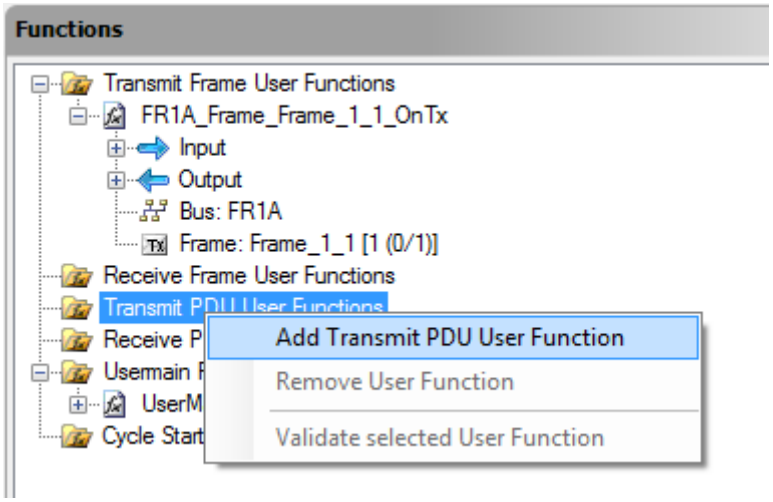


Figure 73: User Function Editor – Add “PDU Transmit User Function”

After this has been done a window appears (s.), where the tab of the relevant bus must be selected first. In the next step the PDU must be selected for which the function has to be written. Finally, the ‘Create PDU transmit user function’-button must be clicked.

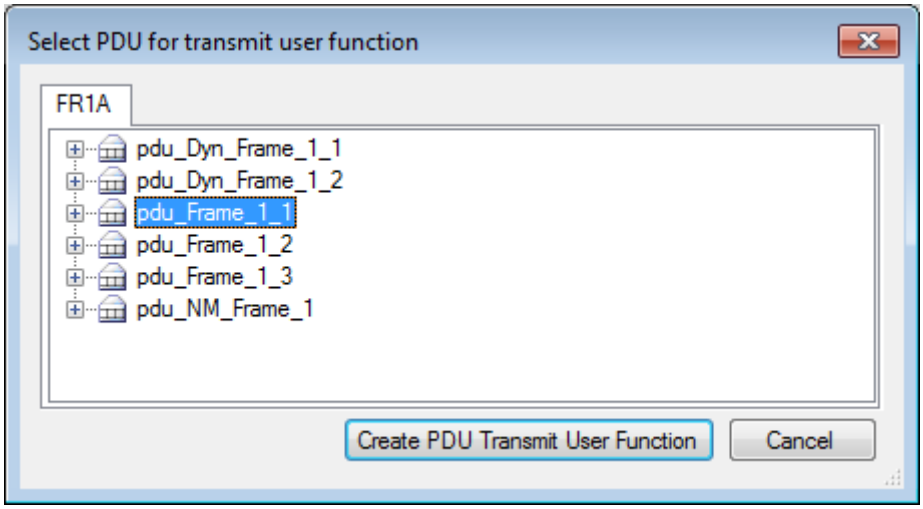


Figure 74: User Function Editor – PDU selection to create a “PDU Transmit User Function”

After that an empty user function appears in the “Source Code Editor”-section (see chapter 5.8.2.1.2 for more details).

Additionally, the “PDU Transmit User Functions”-folder contains a new user function (s. Figure 68).

A “PDU Transmit User Function” can be removed by clicking on the option ‘*Remove User Function*’ within the context menu. The context menu is available by right clicking on the appropriate “PDU Receive User Function”.

To validate a function, the option ‘*Validate Selected User Function*’ can be selected within the context menu.

	NOTICE
	Each PDU Transmit User Function (CAN, FlexRay and Ethernet) has an input parameter for every configured E2E profile. By dereferencing and setting the parameter to 0, the corresponding E2E method is disabled.

5.8.1.4 PDU Receive User Functions

A “PDU Receive User Function” can be written for each PDU which is received by the RBS or gateway. The “PDU Receive User Function” of a PDU is called after the PDU is received.

A new “PDU Receive User Function” can be added by clicking on *‘Add PDU Receive User Function’* within the context menu (s. Figure 75). The context menu is available by right clicking on the appropriate “PDU Receive User Functions”-folder.

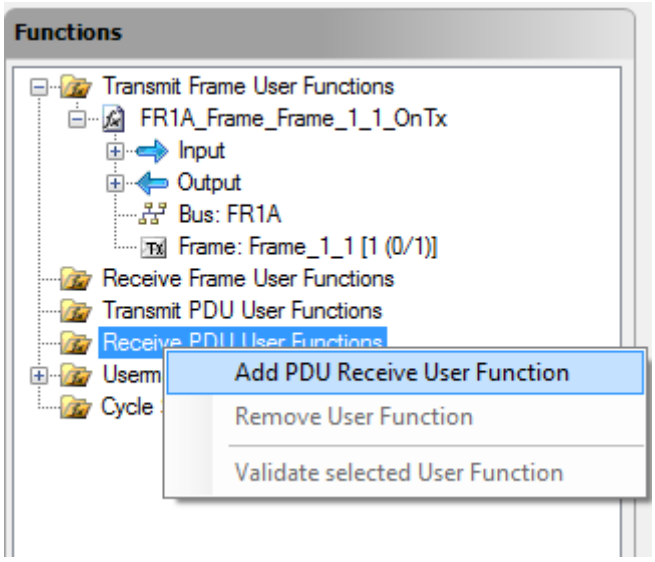


Figure 75: User Function Editor – Add “PDU Receive User Function”

After this has been done a window appears (s. Figure 76), where the tab of the relevant bus must be selected first. In the next step the PDU must be selected for which the function must be written. Finally, the ‘Create PDU Receive user function’-button must be clicked.

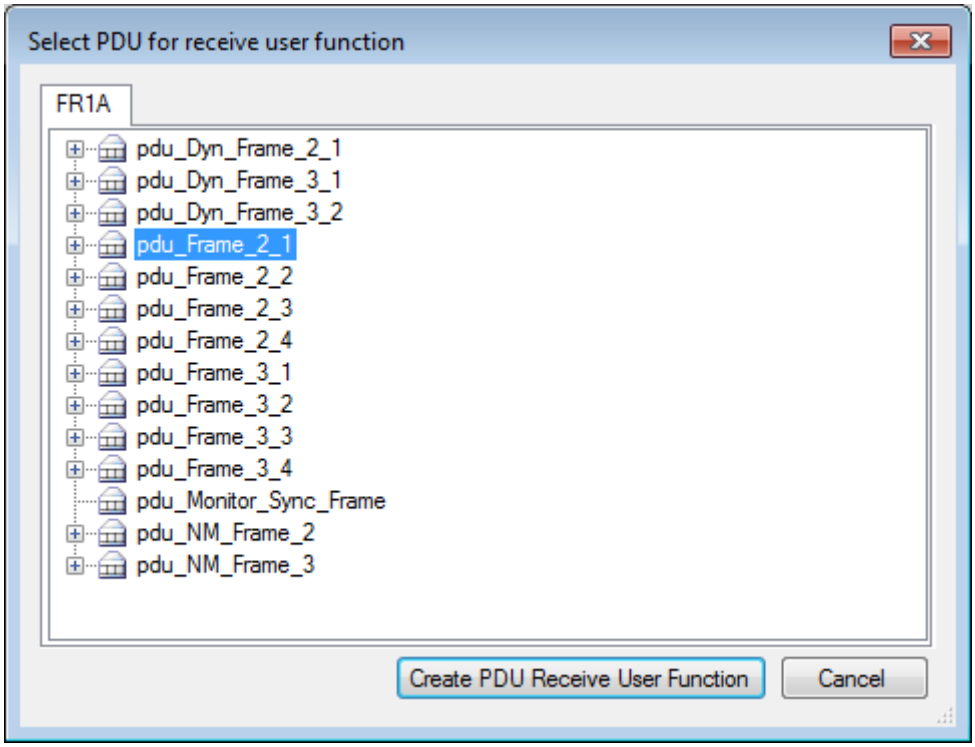


Figure 76: User Function Editor – PDU selection to create a “PDU receive User Function”

After that an empty user function appears in the “Source Code Editor”-section (see chapter 5.8.2.1.2 for more details).

Additionally, the “PDU Receive User Functions”-folder contains a new user function (s. Figure 68).

A “PDU Receive User Function” can be removed by clicking on the option ‘Remove User Function’ within the context menu. The context menu is available by right clicking on the appropriate “PDU Receive User Function”.

To validate a function, the option ‘Validate Selected User Function’ can be selected within the context menu.

5.8.1.5 Bus Receive User Functions

A “Bus Receive User Function” can be added for receiving raw Ethernet or CAN/CAN-FD frames.

It can be added by clicking on *‘Add Ethernet/CAN Bus Receive User Function’* within the context menu (s. Figure 77). The context menu is available by right clicking on the appropriate “Bus Receive User Functions”-folder.

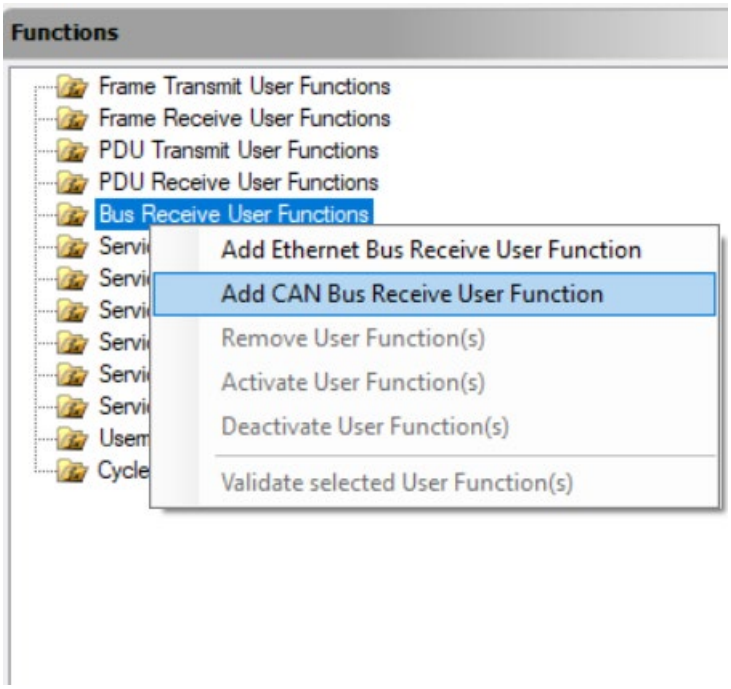


Figure 77: User Function Editor – Creating a “Bus Receive User Function”

After this has been done a window appears (s. Figure 78), where some filters can be configured. Click **[OK]** to create the User Function.

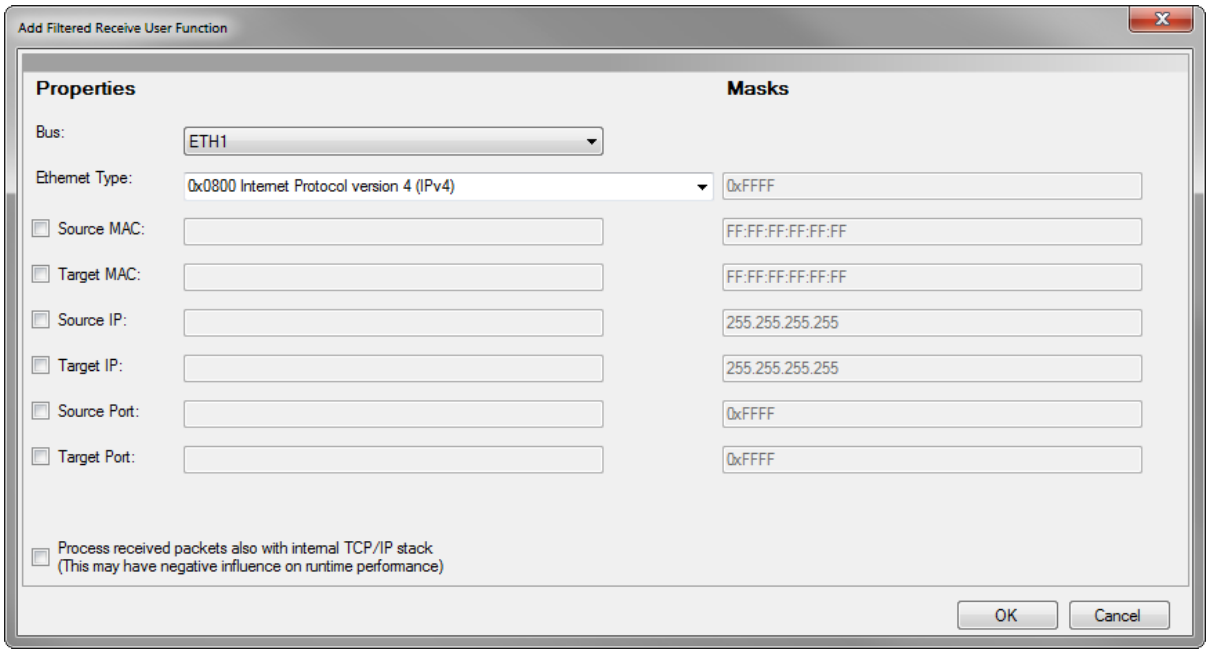


Figure 78: User Function Editor – Dialog “Add Filtered Receive User Function” for an Ethernet Bus

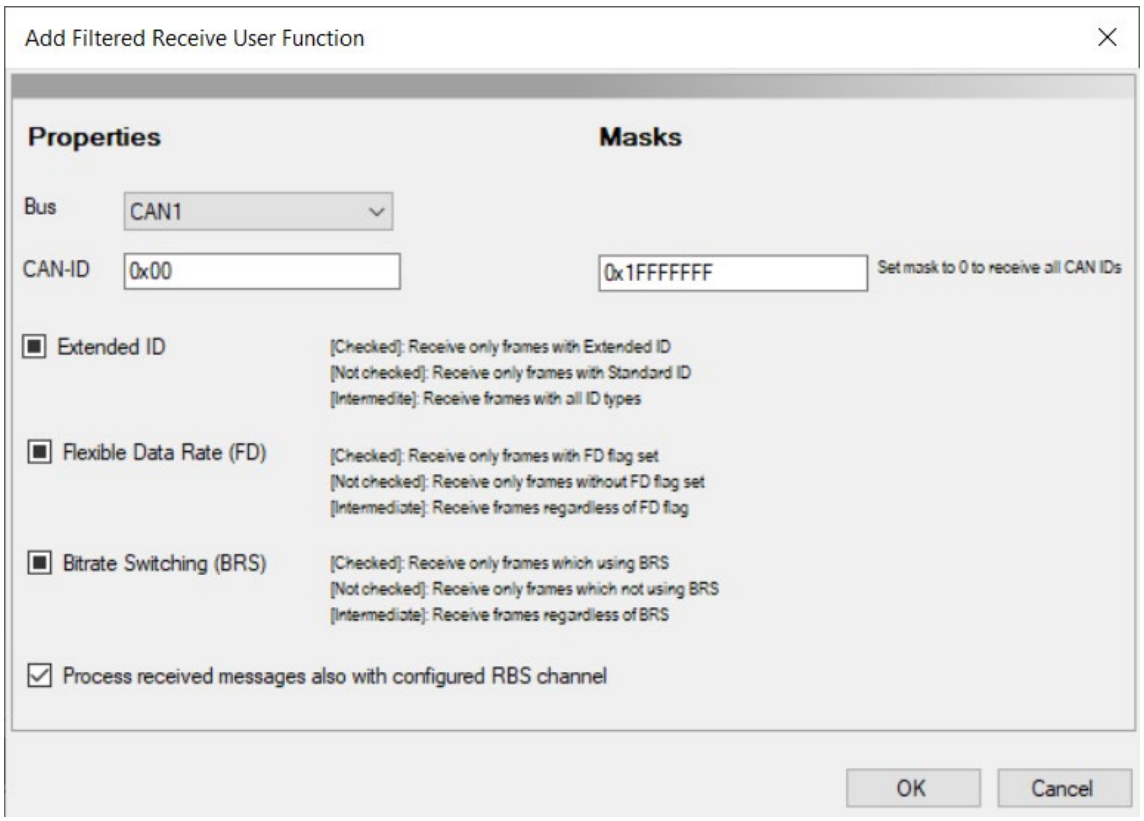


Figure 79: User Function Editor – Dialog “Add Filtered Receive User Function” for a CAN/CAN-FD Bus

The masks work as logical ‘AND Operation’ to the filter and can be used to allow a single value or range of values for the corresponding property.

Ethernet packets with following ether types can be received:

- 0x0800 (IPv4)
- 0x0806 (ArP)
- 0x86DD (IPv6)
- 0x8100 (VLAN)
- 0x88CC (LLdP)
- 0x8906 (FCoE)
- 0x8819 (CobraNet)
- 0x88E5 (MAC)
- 0x88F7 (PTP)

Example for an Ethernet Receive Function:

Source IP = 192.168.1.50 Mask= 255.255.255.255 -> All packages are received, which have the Source IP “192.168.1.50”

Source IP = 192.168.1.50 Mask= 255.0.0.0 -> All packages are received, which have the source IP starting with “192”

	NOTICE
	Multiple ‘Bus Receive User Functions’ can be added if the filters do not overlap. For example, two different ‘Bus Receive User Functions’ with following filters cannot be added: Function: Target IP = 192.168.1.20 Function: Target Port = 0x1234 These filters overlap because a range of conditions match for both functions. Creating the second function will be prevented. A possible solution is to create only a single User Function (e.g., with filter for Target IP) and check the Target Port within this User Function.

All packets which match a configured Bus Receive User Function filter will normally not be forwarded to the internal TCP/IP network stack. Therefore, these packets are not processed by software stacks like SOME/IP or ARP.

The checkbox “Process received packets also with internal TCP/IP stack” can be checked to pass a copy of the received packet to internal network stack.

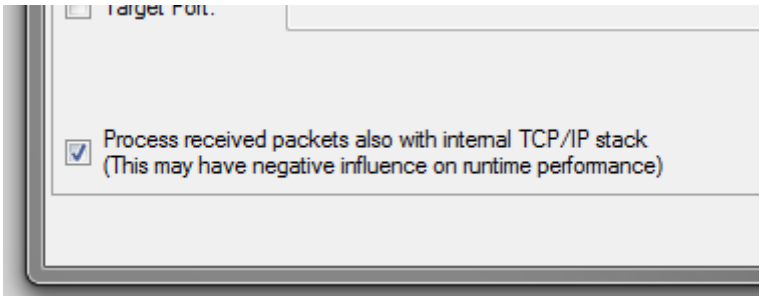


Figure 80: User Function Editor – Dialog “Add Filtered Receive User Function” – Option to duplicate packets

	NOTICE
	Only a copy of the received packet will be passed to the internal network stack. Any changes done in the Bus Receive User Functions will have no effect to internal network stack processing!

An Ethernet Receive Function is created with following input parameters:

```
void Eth_Bus_Rx_Function(u08* data, u32 length, u64 timestamp)
```

The payload and its length are defined with the parameters ‘data’ and ‘length’. The parameter timestamp has the receive time of the ethernet packet.

A CAN/FD receive function is created with following input parameters:

```
CAN_Bus_Rx_Function(u08* data, u08 length, u32 id, u08 ext, u08 fd, u08 brs, u64 timestamp)
```

The payload and its length are defined with the parameters ‘data’ and ‘length’. The ‘id’ parameter has the value of the CAN Frame Id. Over the ‘ext’ parameter it can be found whether the CAN message has an extended Id. The FD parameters keeps the flag ‘Flexible Data Rate’ and the BRS parameter keeps the flag ‘Bitrate Switching’.

	NOTICE
	The ‘timestamp’ parameter of a CAN Receive Function is only valid if the option ‘Analyzing’ (see chapter Base Settings CAN/CAN-FD) is activated for the CAN bus. In other cases, the parameter will have the value ‘0’

5.8.1.6 Service Provided Event User Functions

For Provided services, the following event functions can be created:

- **Event Transmit User Function**: This function is called before the event is sent by the provided service. It can be added by clicking on *'Add Event Transmit User Function'* within the context menu (see Figure 81).

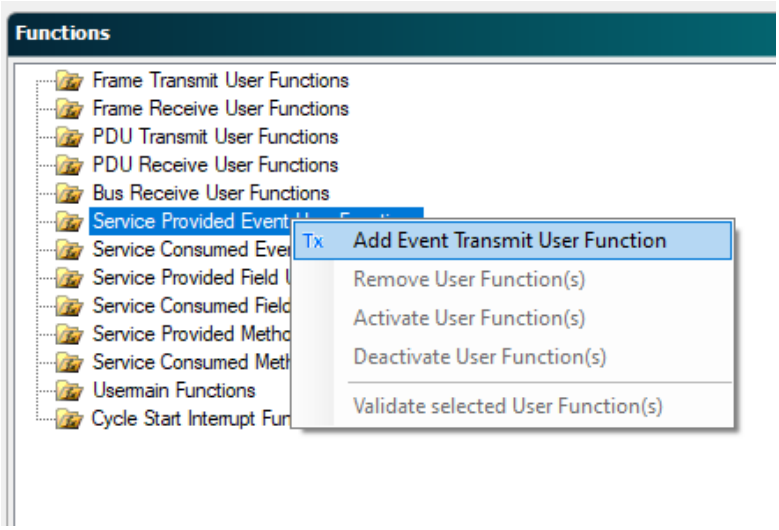


Figure 81: User Function Editor – Creating a "Event Transmit User Function"

The following dialog will be opened to select the event.

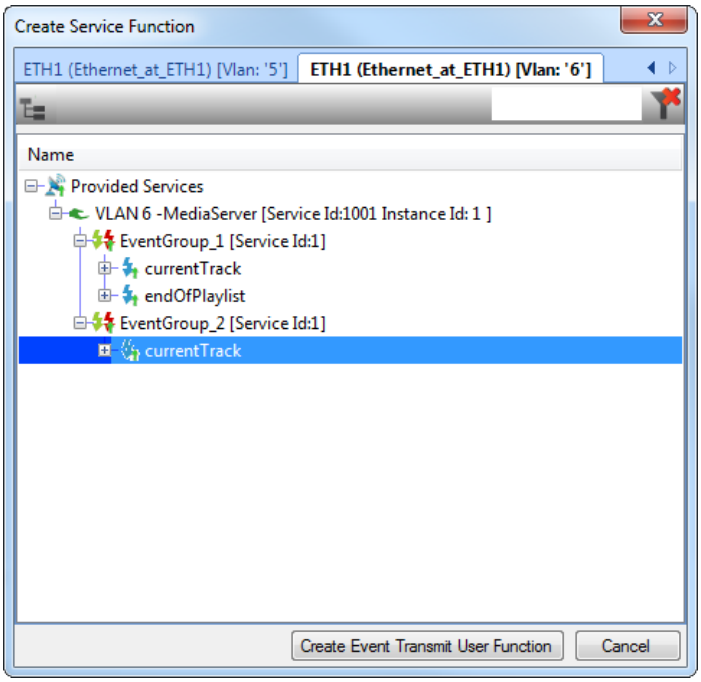


Figure 82: User Function Editor – "Dialog for (provided) event selection"

	NOTICE
	Each Service Provided Event User Function has an input parameter for every configured E2E profile. By dereferencing and setting the parameter to '0', the corresponding E2E method is disabled.

5.8.1.7 Service Consumed Event User Functions

For consumed services, the following event functions can be created:

- **Event Receive User Function**: This function is called, after the event is received from the consumer service. It can be added by clicking on *'Add Event Receive User Function'* within the context menu (see Figure 83).

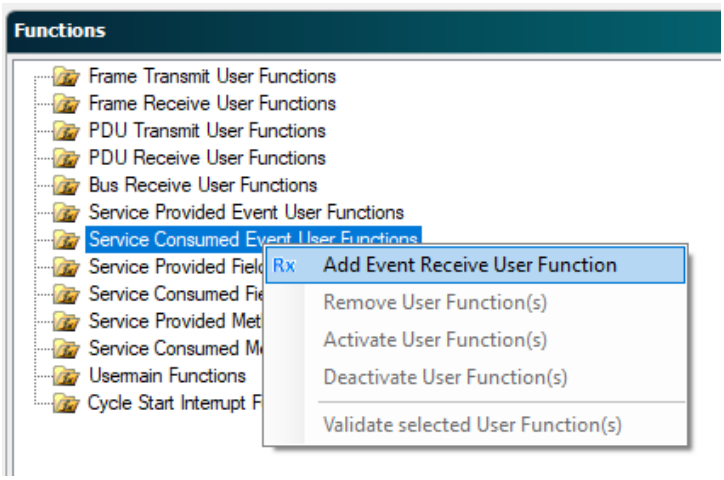


Figure 83: User Function Editor – Creating a “Event Receive User Function”

Following dialog will be opened to select the event.

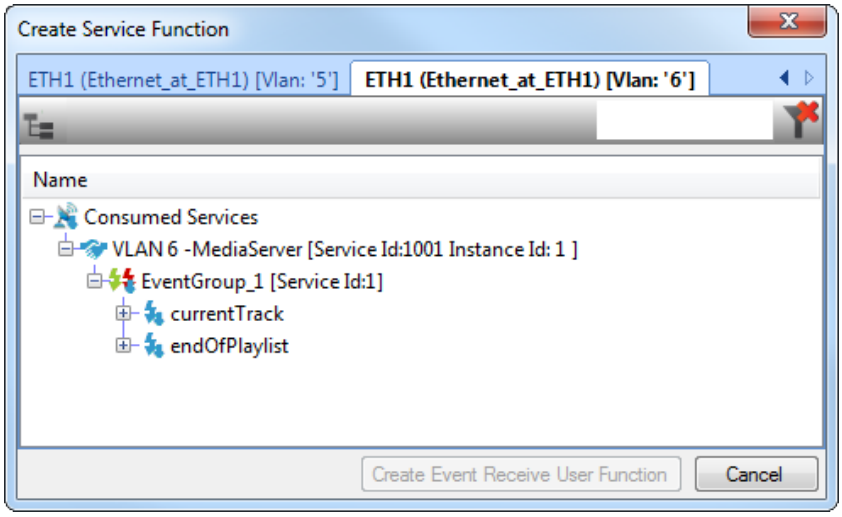


Figure 84: User Function Editor – “Dialog for (consumed) event selection”

5.8.1.8 Service Provided Field User Functions

For Provided services, the following field functions can be created:

- **Get Field Receive User Function**: This function is called after a ‘Get’ request for the field is received.
- **Set Field Receive User Function**: This function is called after a ‘Set’ request for the field is received.
- **Get Field Transmit User Function**: This function is called before a ‘Get’ response for the field is sent.
- **Set Field Transmit User Function**: This function is called before a ‘Set’ response for the field is sent.
- **Field Notifier Transmit User Function**: This function is called before a field notifier is sent.

These functions can be added over the context menu of the ‘Service Provided Field User Functions (see Figure 85).

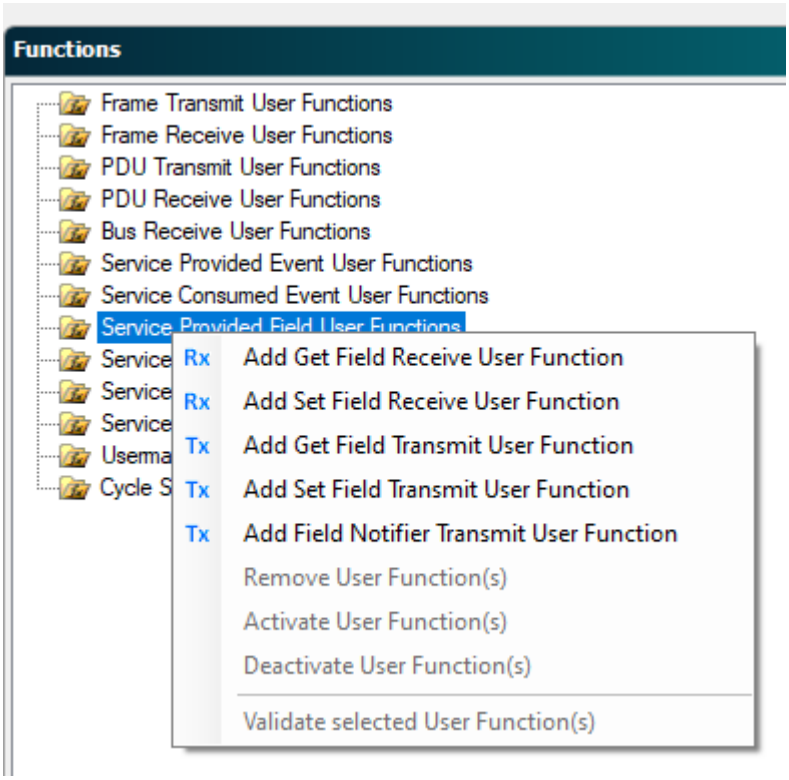


Figure 85: User Function Editor – Creating field functions on the provider side

Following dialog will be opened to select the field or notifier.

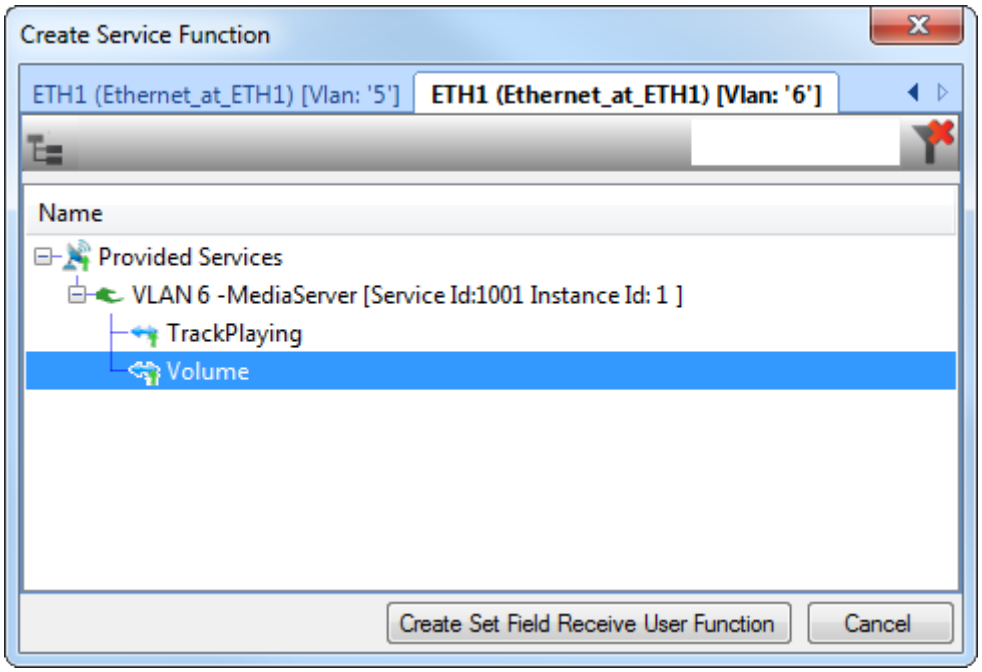


Figure 86: User Function Editor – “Dialog for (provided) field selection”

	NOTICE
	Each Field Notifier Transmit User Function has an input parameter for every configured E2E profile. By dereferencing and setting the parameter to ‘0’, the corresponding E2E method is disabled.

5.8.1.9 Service Consumed Field User Functions

For consumed services, the following field functions can be created:

- **Get Field Receive User Function**: This function is called after the ‘Get’ response for the field is received.
- **Set Field Receive User Function**: This function is called after the ‘Set’ response for the field is received.
- **Get Field Notifier Receive User Function**: This function is called after a notifier for the field is received.
- **Get Field Transmit User Function**: This function is called before the field ‘Get’ request is sent.
- **Set Field Transmit User Function**: This function is called before the field ‘Set’ request is sent.

These functions can be added over the context menu of the ‘Service Consumed Field User Functions’ (see Figure 87).

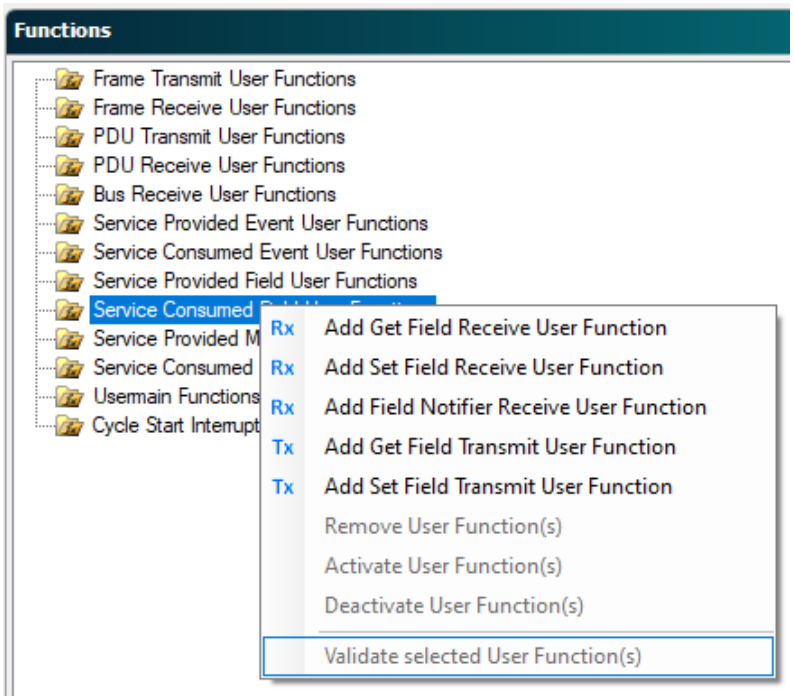


Figure 87: User Function Editor – Creating field functions on the consumer side

Following dialog will be opened to select the field or notifier.

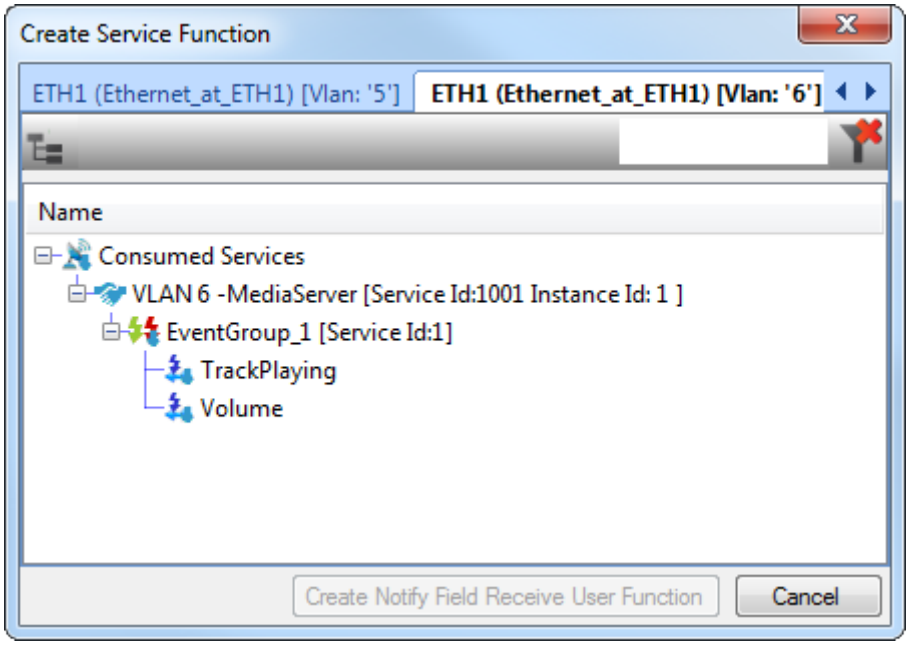


Figure 88: User Function Editor – “Dialog for (consumed) field selection”

	NOTICE
	Each Set Field Transmit User Function has an input parameter for every configured E2E profile. By dereferencing and setting the parameter to '0', the corresponding E2E method is disabled.

5.8.1.10 Service Provided Method User Functions

For provided services, the following method functions can be created:

- **Method Call Receive User Function**: This function is called after a method call is received. It can be added by clicking on *'Add Method Call Receive User Function'* within the context menu (s. Figure 89).
- **Method Response Transmit User Function**: This function is called before a method response is sent. It can be added by clicking on *'Add Method Response Transmit User Function'* within the context menu (s. Figure 89).

SOME/IP supports two method call types: Synchronous and asynchronous. If a method is called synchronous, then the consumer waits until it receives the response from the provider. If a method is called asynchronous, then the consumer does not wait at the response. The provider can send the response after a while. In this time the consumer can do other tasks.

FL3X Config handles every Method call as an asynchronous call. If a method call is received, then the function code contains the code for triggering the method response immediately. The user may delete this code and trigger the method response any time in any user function.

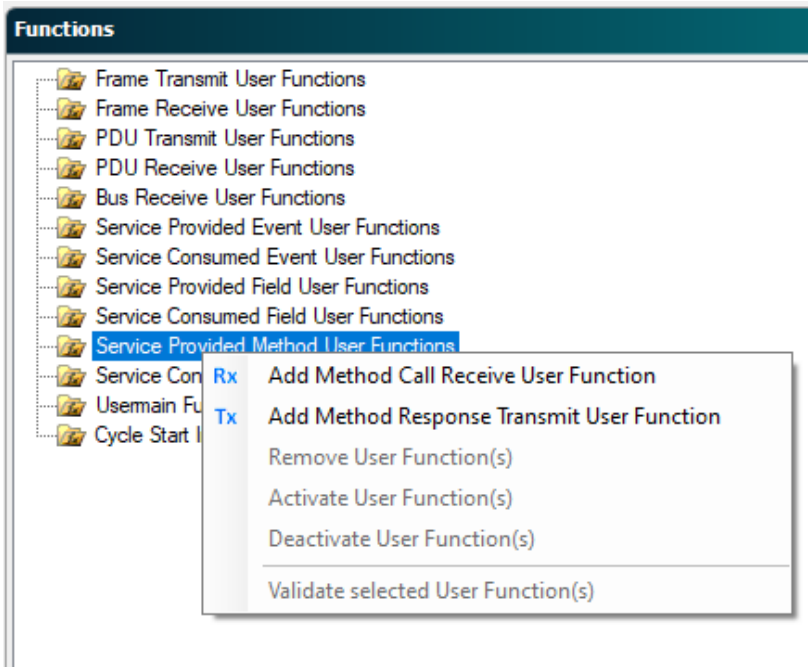


Figure 89: User Function Editor – Creating a "Method Call Receive User Function"

Following dialog will be opened to select the method.

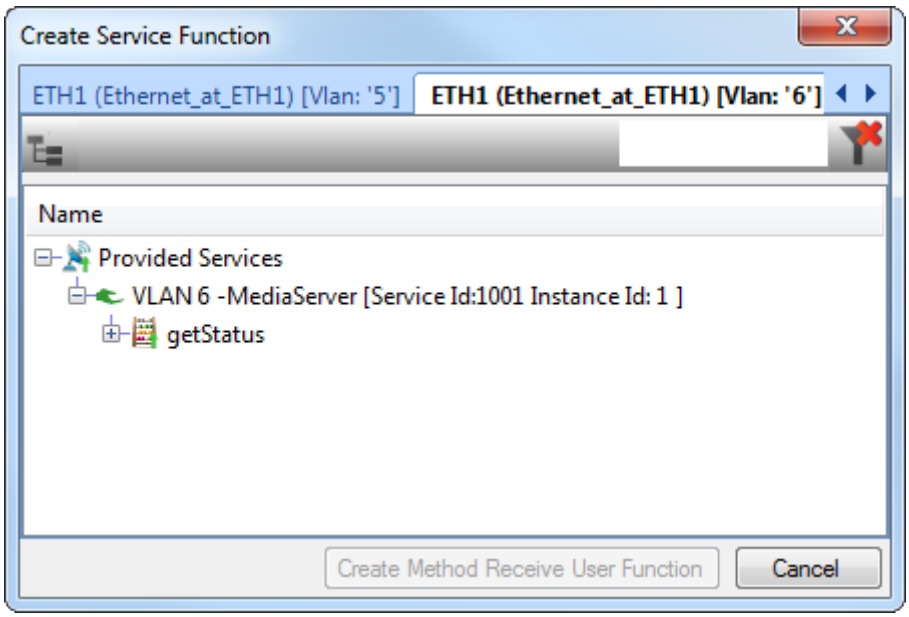


Figure 90: User Function Editor – “Dialog for (provided) method selection”

5.8.1.11 Service Consumed Method User Functions

For consumed services, the following method functions can be created:

- **Method Response Receive User Function**: This function is called after the response of a method call is received.
- **Method Call Transmit User Function**: This function is called before the method call request is sent.

These functions can be added over the context menu of the ‘Service Consumed Method User Functions’ (see Figure 91).

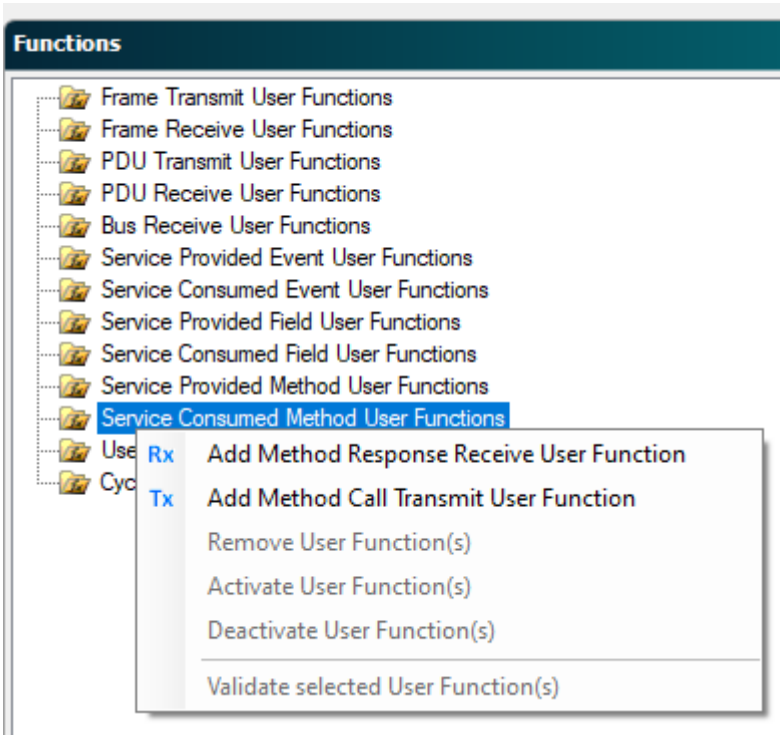


Figure 91: User Function Editor – Creating method functions on the consumer side

Following dialog will be opened to select the method.

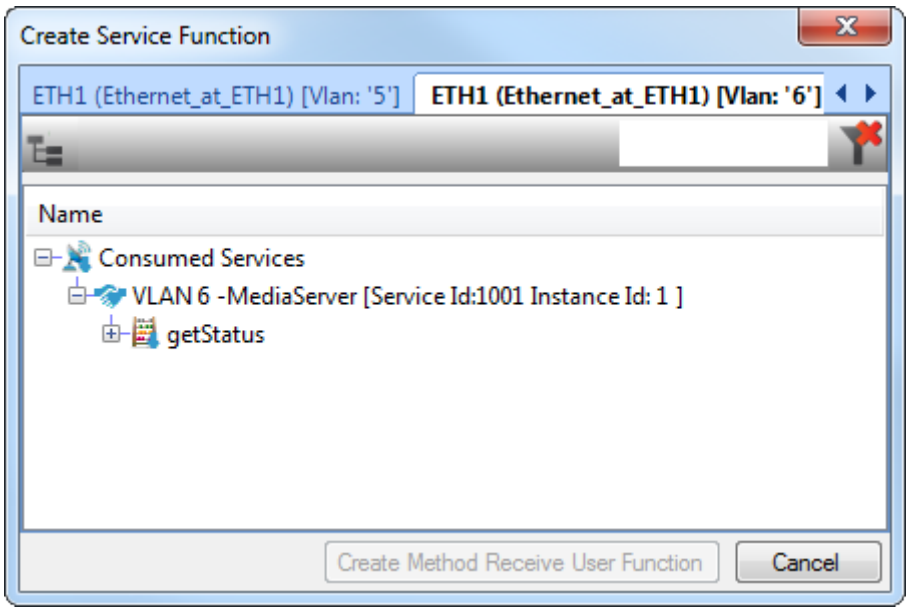


Figure 92: User Function Editor – "Dialog for (consumed) method selection"

	NOTICE
	Each Method Call Transmit User Function has an input parameter for every configured E2E profile. By dereferencing and setting the parameter to '0', the corresponding E2E method is disabled.

5.8.1.12 Usermain Function

Every RBS or Gateway can contain one "Usermain Function" which is cyclically called by a Timer. The time interval of the Timer can be set to a certain value (1ms to 500ms), which e.g. can be 100ms.

The "Usermain Function" can be added by clicking on *'Add Usermain Function'* within the context menu (s. Figure 93). The context menu is available by right clicking on the "Usermain Functions" folder.

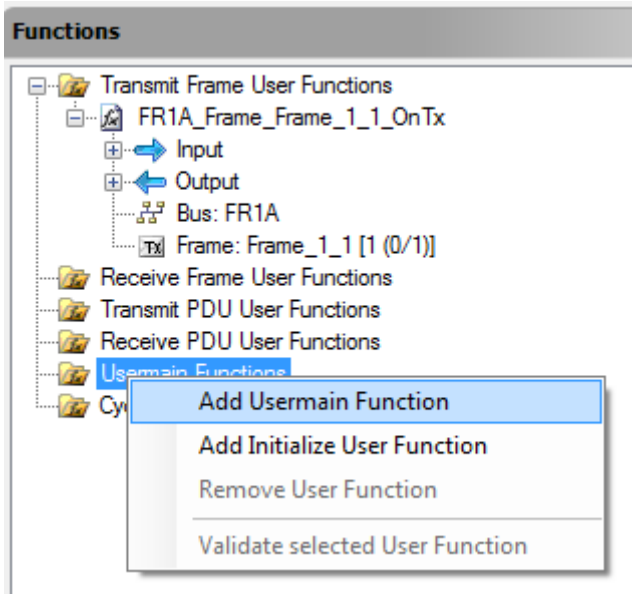


Figure 93: User Function Editor – Add "Usermain Function"

After this has been done a dialog appears (s. Figure 94).

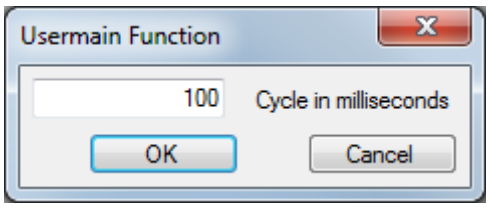


Figure 94: User Function Editor – Configure the period for the “Usermain Function”

Here the cycle time in milliseconds (1ms-500ms) can be set. Confirm the dialog with the **[OK]** button.

After that, a new empty “Usermain Function” appears in the “Source Code Editor”-section (see chapter 5.8.2.1.3 for more details). Additionally, the “Usermain Functions”-folder contains a new user function (s. Figure 68).

The “Usermain Function” can be removed by clicking on the option ‘*Remove User Function*’ within the context menu. The context menu is available by right clicking on the “Usermain Function”.

To validate the function, the option ‘*Validate Selected User Function*’ can be selected within the context menu.

5.8.1.13 Initialize User Function

Every project can contain a single “Initialize User Function”. This function will be called once when the project is started. This event can occur in parallel to other Element related User Functions (e.g. Frame Receive User Function). There is no guarantee that the Initialize User Function will be called before these other functions are called.

It can be used to initialize certain variables, call different imported initialize functions or to provide certain settings. To add the “Initialize User Function” the menu item ‘*Add Initialize Function*’ within the context menu shown in Figure 95 can be clicked. The context menu is available by right clicking on the “Usermain Functions” folder.

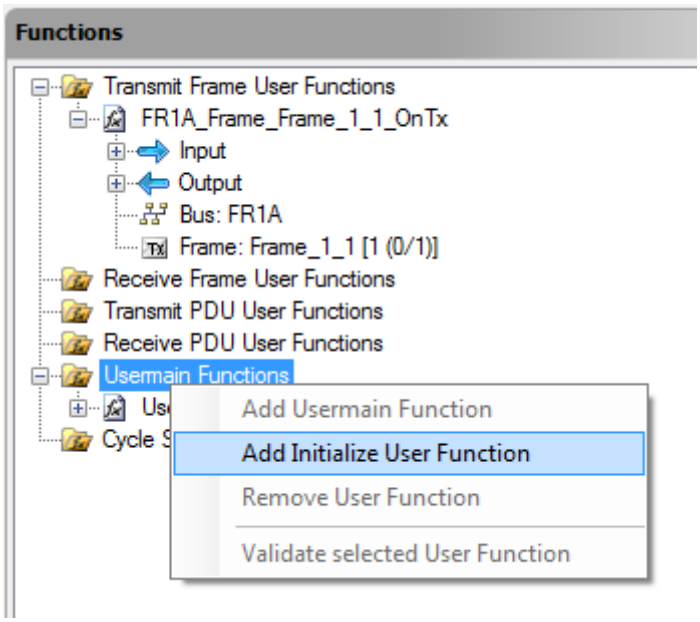


Figure 95: User Function Editor – Add Initialize User Function

After that, a new empty “Initialize User Function” appears in the “Source Code Editor”-section (see chapter 5.8.2.1.3 for more details). Additionally, the “Usermain Functions”-folder contains a new user function (s. Figure 68).

The “Initialize User Function” can be removed by clicking on the option ‘Remove User Function’ within the context menu. The context menu is available by right clicking on the “Initialize User Function”.

To validate the function, the option ‘Validate Selected User Function’ can be selected within the context menu.

5.8.1.14 Cyclestart Interrupt Function

A “Cyclestart Interrupt Function” can be written for each FlexRay bus which is used by the appropriate RBS or Gateway. The “Cyclestart Interrupt Function” is called immediately after a new FlexRay cycle begins.

New “Cyclestart Interrupt Functions” can be added by clicking on *‘Add Cyclestart Interrupt Functions’* within the context menu (s. Figure 96). The context menu is available by right clicking on the “Cyclestart Interrupt Functions”-folder.

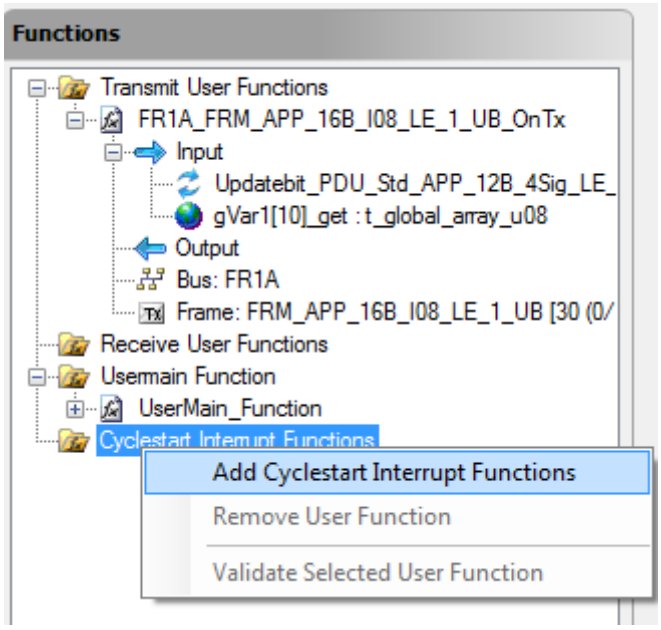


Figure 96: User Function Editor – Add “Cyclestart Interrupt Function”

After this has been done a window appears where the FlexRay busses can be selected (via checkboxes) for which appropriate user functions must be created (s. Figure 97).

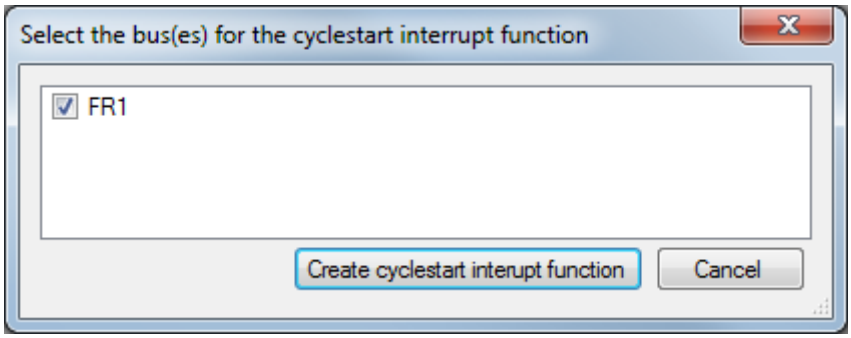


Figure 97: User Function Editor – Select bus to create “Cyclestart Interrupt Function”

Then the “*Create cyclestart interrupt function”*-button must be clicked. After that new empty “Cycle Start Interrupt Functions” are generated. In the “Source Code Editor” the last of them (concerning the tree-view of “Cyclestart Interrupt Functions”) is shown. Additionally, the “Cyclestart Interrupt Functions”-folder contains one or more user functions (s. Figure 68).

A “Cycle Interrupt Function” can be removed by clicking on the option ‘*Remove User Function*’ within the context menu. The context menu is available by right clicking on the appropriate “Cycle Interrupt Function”.

To validate a “Cycle Interrupt Function” the option ‘*Validate Selected User Function*’ can be selected within the context menu.

5.8.1.15 Enable/Disable Functions

User functions can be enabled/disabled via the option “*Deactivate/Activate selected User Function*” within the context menu (s. Figure 98). The User Function Editor cannot be closed via **[OK]** button if any enabled user function is corrupt or invalid.
Disabled user functions are not validated. They are not included in ‘Build project’.

If “Database update” (see chapter 5.14.2) has run through without success for a user function, then it will be automatically disabled to make it possible to access and change the code.

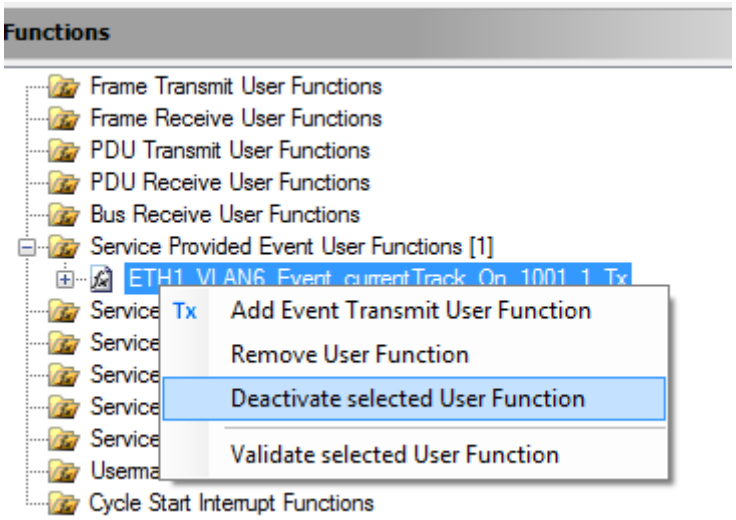


Figure 98: User Function Editor – Enable/Disable User Function

5.8.1.16 Validate All

The validation of all user functions which have been programmed or added to the current project can be triggered by clicking the *“Validate All”*-button (see Figure 68). This can be important, if new .c/.h- files have been added to the project.

Additionally, the validation of a single user function can be triggered by clicking on the ‘*Validate Selected User Function*’ option within the context menu of the appropriate user function.

After a validation has been triggered, the output of the “Validation Output”-section (s. more about that in chapter 5.8.3.9) will be updated. Independent of these possibilities a user function is always validated automatically. This happens a certain number of seconds after a user function has been edited.

5.8.1.17 Export Functions

To export all user functions, which have been programmed or edited, the “*Export All*”- button (see Figure 68) can be clicked. After that, the “Export User Functions” dialog appears, which is a standard dialog to save files. As can be seen in Figure 99 the file type is set to *.*.

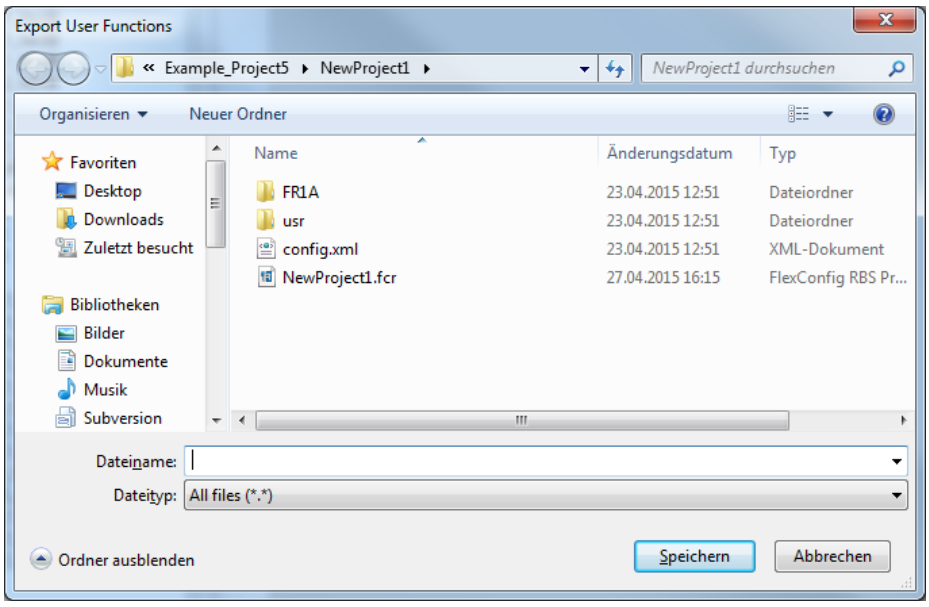


Figure 99: User Function Editor – “Export User Functions” dialog

After a directory and a filename has been selected, pressing the “*Save*”-button implies the generation of two files (*.h*, *.c*).

The Header file (*.h*) contains all definitions of the function headers of the user functions.

The C-code File (*.c*) contains the code of all user functions (especially the function bodies).

5.8.2 Source Code Editor

5.8.2.1 Code-Skeletons for the Different Types of User Functions

If a new user function within the “Functions”-section of the “User Function Editor” is created, a new user function appears in the “Source Code Editor”. How the automatically generated code skeleton looks like, depends on the type of the user function and if standard signal calculation is used within the RBS/gateway or not.

The programming language in which user functions can be programmed is “C”.

5.8.2.1.1 Code-Skeleton Frame Transmit User Function

An example of an automatically generated code skeleton for a user function of type “Frame Transmit User Function” is given in the following.

```
U08 FR1A_FRM_APP_16B_I08_LE_1_UB_OnTx (u08* enableCRC)
{
 /*
 To disable or enable the standard signal calculation, please use the
 parameters of the user function above.

 0 = disable calculation
 1 = enable calculation
 */

 return 1; /*0 = disables frame transmission; 1 = enables frame transmission*/
}
```

The function provides an input parameter for every standard signal and E2E calculations that is used in the RBS. If no standard signal or E2E calculation is defined, the input parameter is “void”.

In this example, a CRC (cyclic redundancy check) standard signal calculation has been used in the RBS.

To identify which parameter belongs to which calculation, the parameter name contains the affix of the standard signal (in this example “CRC”) or the name of the E2E calculation. Note that the E2E names depends on the E2E calculations that are included in the installed OEM package. Refer to chapter 4.5.4 (E2E Protection) for ways to identify which E2E profile is used to secure the PDU.

To enable/disable the standard signal or E2E calculation, the parameter must be dereferenced and the relevant value (“0” or “1”) assigned. With this parameter, the standard signal or E2E calculation can be enabled/disabled for all PDUs in the frame. Calculations for individual PDUs can be enabled/disabled in a “PDU Transmit User Functions” (see chapter 5.8.1.3).

Additionally, the “Frame Transmit User Function” provides a return parameter to control the transmission of the relevant frame. As can be seen in the comments a return value of “0” disables the transmission and a return value of “1” enables the frame transmission.



NOTICE

If the transmission of a frame is suppressed by returning “0” the standard signal or E2E calculation will still be processed!

To suppress the standard signal or E2E calculation, the appropriate parameter must explicitly be set to “0”!

5.8.2.1.2 Code Skeleton Frame Receive User Function

An example of an automatically generated code skeleton for a user function of type “Frame Receive User Function” is given in the following.

```
Void FR1A_FRM_APP_02B_I01_LE_2_NoUB_OnRx (void)
{
 /* FRM_APP_02B_I01_LE_2_NoUB.data[] : points to the frame data (u08*) */
 /* FRM_APP_02B_I01_LE_2_NoUB.length : contains the length of the frame in bytes
*/
```

```
 FRAME FRM_APP_02B_I01_LE_2_NoUB = get_Current_Rx_Frame_FR1A();
 /* Write your code here... */
}
```

As can be seen there are no input or return parameters within “Receive User Functions”. Instead, a line is provided where a local variable of type “Frame” is defined. This variable contains a pointer to the payload-data of the frame that is actually received, and the length of the frame. The variable contains the data resulting from the “get_Current_Rx_Frame_FR1A” call.

5.8.2.1.3 Code Skeleton PDU Transmit User Function

An example of an automatically generated code skeleton for a user function of type “PDU Transmit User Function” is given in the following.

```
U08 FR1A_PDU_pdu_Frame_1_1_OnTx (u08 updateBit, u08* enableCRC)
{
 /*
 The parameter updateBit contains the current state of PDU update bit
 0 = no changes since last Tx
 1 = new data present
 The updateBit parameter exists regardless whether the PDU has an update bit.
 */
 /*
 To disable or enable the standard signal calculation, please use the
 parameters of the user function above.

 0 = disable calculation
 1 = enable calculation
 */

 /*
 The return value contains the state of the update bit to use for Tx.
 The state can be 0 or 1 (!=0 is used as 1)
 The internal storage is updated with the return value.
 If the PDU does not have an update bit, the return value is ignored.
 */

 /* Write your code here... */
 return updateBit;

}
```

The input parameter “updateBit” is only shown for FlexRay PDUs. This parameter is deprecated and has no effect (reason: a PDU Transmit User function for a PDU with an update bit is only called, if the update bit is already “1”.

Similar to “Frame Transmit User Functions” the standard signal and E2E calculations can be enabled/disabled by call by reference parameters (“enableCRC” in this example).

In this function the standard signal calculation can be enabled/disabled for the selected PDU. If a “Frame Transmit User Function” exists for the frame containing the selected PDU, the calculation can be enabled/disabled for all PDUs in the frame. In this case the two parameters are evaluated by a logical AND expression. E.g. if the “enableCRC” parameter of the frame is set to “1” and the parameter of the PDU is set to “0”, the standard signal calculation will be disabled.

The return parameter of the function serves to set the state of the Update Bit. The Update Bit is set to the returned value immediately before the appropriate PDU is sent. Initially the return parameter is set to the value of the input parameter “updateBit”.

5.8.2.1.4 Code Skeleton PDU Receive User Function

An example of an automatically generated code skeleton for a user function of type “PDU Receive User Function” is given in the following.

```
Void FR1A_PDU_pdu_Frame_2_1_OnRx (void)
{
 /* parentFrame.data[] : points to the parent frame data (u08*) */
 /* parentFrame.length : contains the length of the parent frame in bytes */
 FRAME parentFrame = get_Current_Rx_Frame_FR1A();
```

```
 /* Be careful using the get_Current_Rx_Frame_{BusiId} method for multiple
instantiated PDUs */
 /* Be careful using the 'Get Frame' method to access multiple instantiated PDUs
*/

 /* Write your code here... */
}
```

Similar as the "Frame Receive User Function" this function does not have any input or output parameters.

The function body contains the call of the "get_Current_Rx_Frame" function. With the resulting "parentFrame"-variable the user can have access to the payload of the frame, where the PDU is contained in.

5.8.2.1.5 Code Skeleton UserMain Function

An example of an automatically generated code skeleton for a user function of type "UserMain Function" is given in the following.

```
Void UserMain_Function (void)
{
 /* Write your code here... */
}
```

As can be seen there are no input or return parameters within the "Usermain Function" and the function body is empty.

The function is called periodically within a certain amount of time which is set by the user in milliseconds (1-500 ms).

	NOTICE
	The setting of PDU Update Bits within the Usermain Function is not appropriate. This is because the Usermain Function is called asynchronously to the considered bus. Additionally, the PDU Update Bits are eventually set automatically, so that the setting in the Usermain Function is overwritten.

5.8.2.1.6 Code Skeleton Initialize Function

An example of an automatically generated code skeleton for a user function of type "Initialize Function" is given in the following.

```
Void Initialize_Function (void)
{
 /* Write your code here... */
}
```

As can be seen there are no input or return parameters within the "Initialize Function" and the function body is empty.

The "Initialize Function" can be used to initialize certain variables/settings or to call different initialize functions which have e.g. been imported.

5.8.2.1.7 Code Skeleton Cyclestart Interrupt Function

An example of an automatically generated code skeleton for a user function of type "Cyclestart Interrupt Function" is given in the following.

```
Void Cyclestart_Interrupt_Function_FR1 (u08 cycle_number)
{
 /* Write your code here... */
}
```

As can be seen there are no input or return parameters within the "Cyclestart Function" and the function body is empty.

The function is called every time a new FlexRay cycle is starting. From this results that the function is only available for busses of type FlexRay.

5.8.2.2 Accessing Data from the Database

After starting the Build Process, code will be generated, which contains a database.

The database primarily contains data (signals, PDUs, frames, Update Bits, etc.), which is sent by the RBS or gateway. Additionally, the database contains data, especially frames, which have been temporarily received.

To access this data, different functions are available, which are described in more detail in chapter 5.8.3.

5.8.2.3 Different Code Examples

5.8.2.3.1 Working with Defines

It is possible to define different Defines with the “Defines Editor” (see chapter 5.6.7). These Defines can be used for different purposes as for e.g. generating different code-parts depending on `#ifdef` instructions. Because the user functions are functions which are written in the programming language “C”, all pre-processor instructions which handle these defines are allowed.

The following code sample shows how a Define (“Define_Test”) can be used. The value of the Define is compared with a certain signal (“08Bit_Sig20_Req”). If both are equal the DO2 Pin is set to “1”, otherwise to “0”.

```
U08 FR1A_FRM_APP_16B_I08_LE_1_UB_OnTx (u08* enableCRC)
{
 /* Signal FR1A.FRM_1.PDU_1.08Bit_Sig20_REQ */
 u08 _08Bit_Sig20_REQ = get_FR1A_Signal_PduSigIn1();
 if (_08Bit_Sig20_REQ == Define_Test)
 {
       set_DIO_OUT_2_ON();
 }
 else
 {
       set_DIO_OUT_2_OFF();
 }
}
```

5.8.2.3.2 Working with Global Variables of Type Array

In the following example it is shown how to work with a global variable array (“gVar1”), especially how to set all values of the array. As can be seen in the comments the field “data” of “gVar1” is a pointer of type “u08*”. Due to that, the access via pointer directly changes the values within the internal database. A set function for global variable arrays is not available.

```
U08 FR1A_FRM_APP_16B_I08_LE_1_UB_OnTx (u08* enableCRC)
{
 /* gVar1.data[] : points to the array data (u08*) */
 /* gVar1.length : contains the number of the elements of the array (u32) */
 /* There is no 'set...' function required. All changes of the array data are
 saved automatically */
 t_global_array_u08 gVar1 = get_GlobalVar_gVar1();
 int k;
 for (k=0; k <gVar1.length; k++)
 {
       gVar1.data[k]=1;
 }
 return 1;
}
```

5.8.2.3.3 Suppressing the Transmission of a Frame

To suppress the transmission of a frame the return value of the appropriate “Frame Transmit User Function” has to be set to “0”. In this example this is done depending on a PDU Update Bit. But it is important to keep in mind, that the sending-behavior of the RBS or gateway also depend on the transmission settings (see chapter 5.6.7).

```
u08 FR1A_FRM_APP_16B_I08_LE_1_UB_OnTx (u08* enableCRC)
{
 /*
 To disable or enable the standard signal calculation, please use the
 parameters of the user function above.

 0 = disable calculation
 1 = enable calculation
 */
 /* PDU update Bit FR1A.Frm1. PDU_Std_APP_12B_4Sig_LE_UB_EB.UpdateBit
 u08Updatebit_PDU_Std_APP_12B_4Sig_LE_UB_EB                                    =
 get_FR1A_Updatebit_Pdu_e0f44ae7532c41ffa07c71c6e0c17615();
 if (Updatebit_PDU_Std_APP_12B_4Sig_LE_UB_EB == 1)
 {
       return 1;
 }
 else
 {
       return 0;
 }
}
```

5.8.2.3.4 Trigger the Transmission of a Frame

To trigger the transmission of a frame within a “UserMain_Function” the “set_*bn*_FrameTxEvent_*fn*” function must be called. In the following example this happens depending on the state of a certain signal.

Note that the behavior depends on the bus-type. For CAN and Ethernet channels, the frame TxEvent results in the immediate transmission of the frame.

For FlexRay channels, the transmission of frames is determined by the FlexRay cycles and their frame slots. A frame TxEvent guarantees that a data frame will be sent in the following slot for this frame.

Similarly, the transmission of LIN frames is determined by the schedule table that is controlled by the master node. For frames sent by the master node, the frame TxEvent guarantees that a data frame is sent at the following scheduled frame slot. For frames sent by a slave node, the time of transmission is controlled by the master. Therefore, a frame TxEvent does not result in an extra transmission and only the frame payload is updated.

```
Void UserMain_Function (void)
{
 /* Signal FR1A.FRM_1.PDU_1.04Bit_Sig01 */
 u08 _04Bit_Sig01 = get_FR1A_Signal_SigInstance1();
      if (_04Bit_Sig01 == 1)
 {
       /* Sets the Tx event for the frame (FRM_DRS_16B_I01_LE_1_NoUB) at FR1A */
       set_FR1A_FrameTxEvent_FRM_DRS_16B_I01_LE_1_NoUB_215_3_8();
 }
}
```

5.8.2.3.5 Programming a Counter

In this example a counter is programmed in two different ways. Because a function is called here, the content of every locally defined variable is lost after leaving the function. To prevent this behavior, a static variable can be declared (here “counter”). The information is kept in memory, even after leaving the function. This is done in the first few lines. If the counter reaches the value of 100 the DIO-Output2 is set to ‘1’ and the counter is reset to ‘0’. If the counter has a value unequal to 100 the DIO-Output2 is set to ‘0’.

This behavior can be programmed in a similar way by using a global variable (“counter2”). In this case the functions “get_GlobalVar_counter2” and “set_GlobalVar_counter2” are used.

```
Void Cyclestart_Interrupt_Function_FR1 (u08 cycle_number)
{
 /* counter with static variable */
 static u08 counter =0;
 if (counter == 100)
 {
       set_DIO_OUT_2_OFF();
       counter=0;
 }
```

```
 else
 {
       set_DIO_OUT_2_ON();
 }
 counter++;

      /* counter with global variable */
 u08 counter2 = get_GlobalVar_counter2();
 if (counter2 == 100)
 {
       set_DIO_OUT_2_ON();
       counter2=0;
 }
 else
 {
       set_DIO_OUT_2_OFF();
 }
 counter2++;
 set_GlobalVar_counter2(counter2);
}
```

5.8.2.4 Naming Conventions of Identifiers

Independent of the naming conventions within the “C”-programming language, the naming conventions of the application “FL3X Config” must also be considered.

For example, it is possible that a file is added to the current project, which has the name of a file which was generated by “FL3X Config” (e.g. “*gw.c*”). Additionally, it is possible that variables and macros are declared/used which have already been internally declared by “FL3X Config”. To prevent these situations, it is recommended to use a **unique prefix for all identifiers** which are used.

5.8.2.5 Useful Shortcuts for the Source Code Editor

Several useful shortcuts can be used within the Source Code Editor. These are listed in the table below (s. Table 72).

Shortcut	Description
CTRL-F	Similar as in other text-editors it is possible to press *CTRL-F* to search for certain string-expressions within a user function. If this button is pressed, a window pops up and certain settings, such as highlighting, match case, and whole word, are possible. The dialog also contains a replace tab s. *CTRL-H.*
CTRL-G	If this Shortcut is used, a dialog pops up, where a certain line-number can be entered. After pressing the “*OK*”-button, the cursor within the editor jumps to this line position.
CTRL-H	If this Shortcut is used, the same dialog as *CTRL-F* pops up. The only difference is that the “Replace” tab is activated. Using this dialog allows to search for certain string-expressions and replace them with other string-expressions. Different Buttons, like “*Replace all*”, “*Replace Previous*” and “*Replace Next*”, are available.

Table 72: User Function Editor – List of usable Shortcuts

It is also possible find the matching element for a function call. Mark a function call within the Source Code Editor, open a context menu via right-click and then click on “*Show Info*”. Now the corresponding element is highlighted in the section “Signals and Variables”.

5.8.3 Signals and Variables

In the “Signals and Variables”-section different functions can be added or removed to the currently edited user function. These different functions provide access to items like ECUs, frames, PDUs, signals, update bits, hardware signals (analog, digital) and so on.

As can be seen in Figure 100 these different items are accessible within a tree structure.

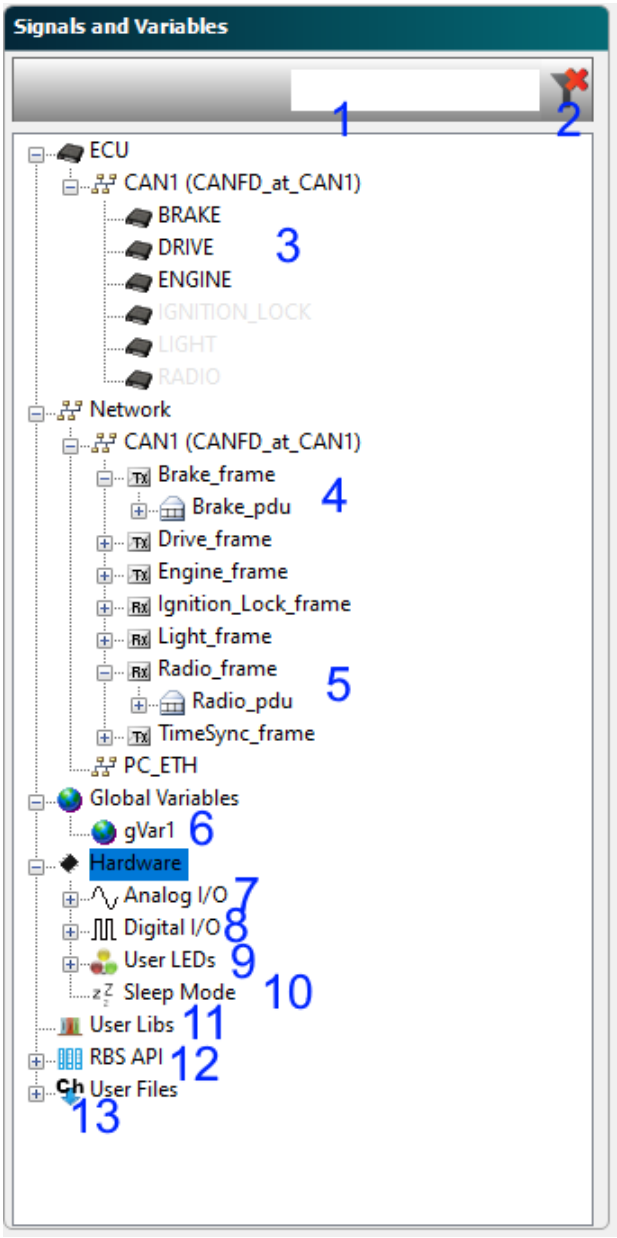


Figure 100: User Function Editor – section “Signals and Variables”

The description of the tree structure is given in the following Table 73.

No	Description
1	The search field can be used to filter the nodes shown in the signals and variables tree with the given text
2	Clears the filtering.
3	The **ECU** item shows all appropriate ECUs of all configured busses. The following points must be considered: • Only simulated ECUs, appropriate to the relevant bus, are shown not greyed out. • Only not simulated ECUs, appropriate to the relevant bus and have a sending behavior, are shown greyed out. • Not simulated ECUs, which do not have a sending behavior, are not shown.
4	The **Network** item shows all Transmit (**Tx**) Frame(s), appropriate to the relevant bus, and all child items (PDUs, signals, ...) are shown.
5	The **Network** item shows all Receive (**Rx**) Frame(s), appropriate to the relevant bus, and all child items (PDUs, signals, ...) are shown.
6	Shows all **Global Variables (s. chapter 5.8.3.3)**.
7	Shows all accessible analog input pins (**Analog I/O**).

No	Description
8	Shows all accessible digital input/output pins (**Digital I/O**).
9	Shows the items for controlling the device LEDs
10	The **Sleep Mode** item delivers functionality to set the sleep mode.
11	User Libs can be managed in this section (s. chapter 5.8.3.5)
12	API functions can be accessed in the section (s. chapter 5.8.6)
13	The item **User Files** shows all user files and folders, which have been added to the project (s. chapter 5.8.3.8).

Table 73: User Function Editor – Section “Signals and Variables”

The different main items (“ECU”, Network”, “Global Variables”, ...) within the tree structure and their subsequent child items are explained in more detail in the following chapters.

5.8.3.1 ECU

Within the “ECU” item different functions can be added or removed to the currently active user function to control (set on/off, get state) the selected ECU.

Only ECUs which are used for sending frames are editable and not greyed out. The other ECUs coming from the database, which are not used for sending are greyed out.

After right-clicking an ECU, which is not greyed out, a context menu is shown (see Figure 101).

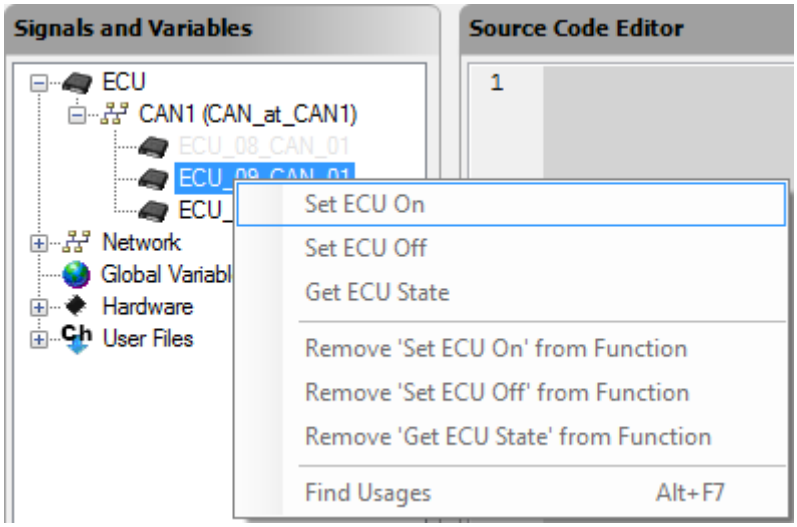


Figure 101: User Function Editor – context menu for ECUs

If one of the shown (not greyed out) menu-items is selected, the appropriate function will be applied to the currently edited user function. The different menu items are explained in more detail in the following table.

Menu Item	Description
‘Set ECU On’	Adds the function call `set_ECU_ON(x);` to the currently edited function. Where *x* is an internally generated Define, which specifies the appropriate ECU-ID. The parameter *x* is not editable! With this function the simulation of the selected ECU is switched on.
‘Set ECU Off’	Adds the function call `set_ECU_OFF(x);` to the currently edited function. Where *x* is an internally generated define, which specifies the appropriate ECU-ID. The parameter *x* is not editable! With this function the simulation of the selected ECU can be switched off.

Menu Item	Description
'Set ECU Default'	Adds the function call `set_ECU_DEFAULT(x);` to the currently edited function. Where *x* is an internally generated define, which specifies the appropriate ECU-ID. The parameter *x* is not editable! This function sets the default values of the signals contained in the frames sent by the selected ECU. See also chapter 5.9.
'Get ECU State'	Adds the following line `s08 sn_STATE = get_ECU_State(x);` to the currently edited function. Where *sn* is the short name of the ECU (parameter *x* s. '*Set ECU on/off*'). The function "get_ECU_State(*x*)" returns a value which can be: • -1: error occurred • 0: ECU is off • 1: ECU is on
'Remove 'Set ECU On' from function'	Removes all occurrences of the *'Set ECU On'* function within the currently edited user function.
'Remove 'Set ECU Off' from function'	Removes all occurrences of the *'Set ECU Off'* function within the currently edited user function.
'Remove 'Set ECU Default' from function'	Removes all occurrences of the *'Set ECU Default'* function within the currently edited user function.
'Remove 'Get ECU state' from function'	Removes all occurrences of the *'Get ECU state'* function within the currently edited user function.
'Find Usages'	Shows all user functions which use one of the three functions described in the first three lines of this table applied to the currently selected ECU (see chapter 5.8.3.9).

Table 74: User Function Editor – Description of the context menu for ECUs

5.8.3.2 Network

Within the "Network"-item different functions can be added or removed to the currently edited user function to access different data elements (frames, PDUs, signals, update bits, switches).

The direct child items of the "Network"-item are the different busses of the configured RBS. The direct child items of the busses are the different Rx/Tx frames which are received or sent by the RBS.

After right clicking a selected Tx frame a context menu is shown (see Figure 102).

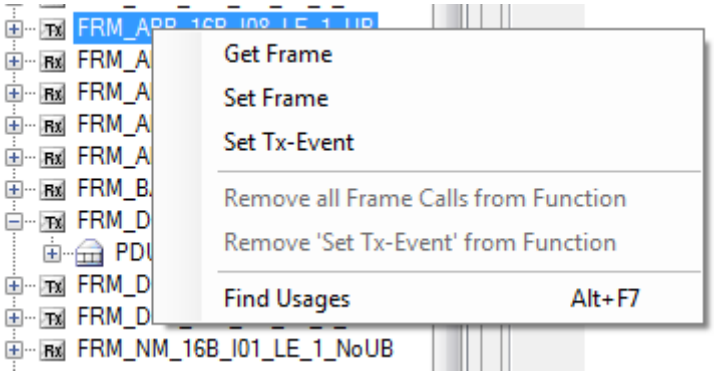


Figure 102: User Function Editor – context menu for transmit frames

The different menu items are explained in more detail in Table 75.

NOTICE

The menu items *'Get Frame'*, *'Set Frame'*, *'Set Tx-Event'* are only available for Tx-Frames! For Rx-Frames these menu items are not available!

It is only possible to get the information of the actual received Rx-Frame by using *'get_Current_Rx_Frame_bn'* (*bn*: bus name).

The child items of the Rx/Tx frames are eventually existing Update Bits and the relevant PDUs.

If the PDUs are normal PDUs they again contain child items which are signals. If the PDU is a multiplexed PDU the child items are a switch and the relevant switched PDUs.

If one of the child/descendant items of Rx/Tx frames (PDU, update bit, signal, switch) is right-clicked different context menus appear. For example, the context menu of a signal is shown in Figure 103.

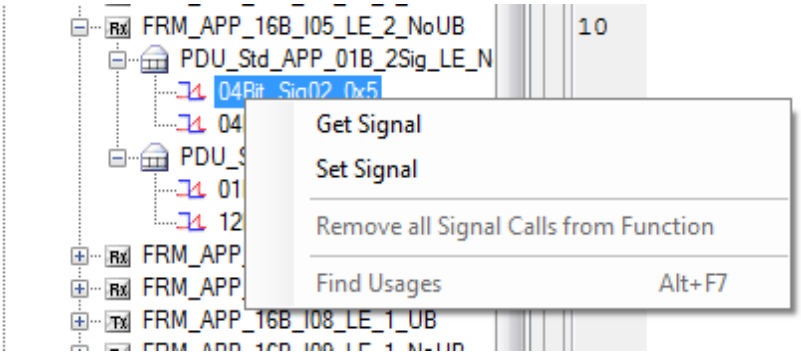
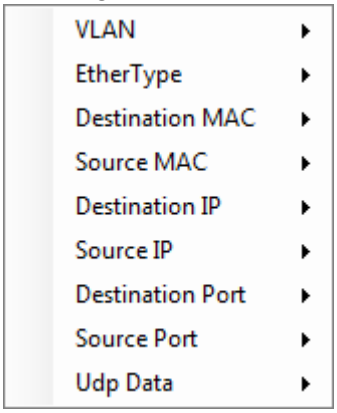


Figure 103: User Function Editor – context menu for signals

All menu items for the different items are explained in Table 75.

Menu Item	Description
'Set Bus On'	Adds the function call `set_BUS_ON(bn);` to the currently edited function. Where *bn* is the name of the selected bus. With this function, the selected bus will be activated. An activated bus will send and receive frames. If *bn* is a FlexRay bus, then all configured channels are activated.
'Set Bus Off'	Adds the function call `set_BUS_OFF(bn);` to the currently edited function. Where *bn* is the name of the selected bus. With this function, the selected bus will be switch off (deactivated). A deactivated bus will not send and receive any frames. If *bn* is a FlexRay bus, then all configured channels deactivated.
'Get Bus State'	Adds the following line `s08 bn_STATE = get_BUS_STATE(bn);` to the currently edited function Where *bn* is the name of the selected bus. The function "get_BUS_STATE(*bn*)" returns a value: • -1: error occurred • 0: Bus is off • 1: Bus id on
'Set Bus Tx On'	Adds the function call `set_BUS_TX_ON(bn);` to the currently edited function. Where *bn* is the name of the selected bus. With this function, the selected bus will be able to send frames again.

Menu Item	Description
'Set Bus Tx Off'	Adds the function call `set_BUS_TX_OFF(bn);` to the currently edited function. Where *bn* is the name of the selected bus. With this function the selected bus will not send any frames. Also the SOME/IP-SD und PTP TX messages will not be sent. But the bus can still receive frames.
'Set Bus Default'	Adds the function call `set_BUS_DEFAULT(bn);` to the currently edited function. Where *bn* is the name of the selected bus. This function is used to set the default values of the signals contained in the frames sent by the selected bus. See also chapter 5.9.
'Get Bus Tx State'	Adds the following line `s08 bn TX_STATE = get_BUS_Tx_STATE(bn);` to the currently edited function Where *bn* is the name of the selected bus. The function "get_BUS_Tx_STATE(*bn*)" returns a value: • -1: error occurred • 0: Bus Tx is off • 1; Bus Tx id on
'Set Bus Raw Tx'	Adds the following line `set_BUS_RAW_Tx(bn, data, length);` to the currently edited function Where *bn* is the name of the selected bus. The parameter 'data' is the pointer to the data to be sent. The parameter 'length' defines the length of the data to be sent. This function is only available for ethernet busses and PC Ethernet connector which is shown as 'PC_ETH' bus
Custom ETH Methods	This menu item is only available for ethernet busses and contains the following sub items:  Each sub item provides get & set methods for getting and setting the corresponding ethernet elements, which are useful if raw ethernet packages are sent or received. The comments displayed in the code editor for these functions describe the input parameters.

Menu Item	Description
'Get active Schedule Table'	Adds the following line `s08 bn_Active_Table_Index = get_bn_ACTIVE_SCHEDULE_TABLE();` to the currently edited function Where *bn* is the name of the selected bus. This function is only available for LIN busses and has only in master mode an effect. It returns the actual active schedule table. The returned '`bn_Active_Table_Index`' parameter can have a value which is normally defined in the comment of the function. Possible values for '`bn_Active_Table_Index`': - `Table_bn_NO_ACTIVE_TABLE` - `Table_bn_sn [created for each schedule table. Sn is the short name of a schedule table.]`
'Set active Schedule Table'	Adds the following line `set_bn_ACTIVE_SCHEDULE_TABLE(bn_Active_Table);` to the currently edited function Where *bn* is the name of the selected bus. This function is only available for LIN busses and has only in master mode an effect. It activates the selected schedule table.
'Get Frame' (only Tx frames!)	Adds the following line `FRAME bn_sn = get_bn_Frame_frtr();` to the currently edited function. Where *bn_sn* is the name of the locally generated variable. The following abbreviations have been used: *bn* is the name of the appropriate bus. *Sn* is the short name of the frame. *frtr* is the FIBEX ID of the appropriate frame-triggering. With the function get_*bn*_Frame_*frtr* the payload and the length of the frame is returned. The used type FRAME is a structure which contains the elements "data" and "length". Where "data" is a pointer of type (u08*) to the data of the frame and "length" is the length of the data in bytes. The 'length' property of the FRAME structure can be set to another value to change the length of a CAN, LIN, Ethernet tx-frame or FlexRay tx-frame in the dynamic segment. The maximum length is defined by the used database. **NOTICE** The 'length' property can not be changed for container frames. The length of container frame is calculated based on the size of its contained PDUs.
'Set Frame' (only Tx frames!)	Adds the following lines `FRAME bn_sn = get_bn_Frame_frtr();` `set_bn_Frame_frtr(bn_sn);` to the currently edited function. For the description of the first line and the appropriate abbreviations see *'Get Frame'* in this table. In the second line the data for the relevant frame is stored. The data is taken from the variable *bn_sn.*

Menu Item	Description
'Set Tx-Event' (only Tx frames! Context menu item of Frame)	Adds the following line `set_bn_FrameTxEvent_frtr();` to the currently edited function. The following abbreviations have been used: *bn* is the name of the appropriate bus. *frtr* is the FIBEX ID of the appropriate frame-triggering. With this function, it is possible to set an event to trigger the transmission of a frame. It is important to know that the transmission of a frame also depend on the transmission settings (see chapter 5.6.7).
'Set Default Values (only Tx frames! Context menu item of frame)	Adds the following line `set_bn_FrameDefault_frtr();` to the currently edited function. The following abbreviations have been used: *bn* is the name of the appropriate bus. *frtr* is the FIBEX ID of the appropriate frame-triggering. This function sets the default values of the signals of the selected frame. See also chapter 5.9.
'Remove all frame calls from function'	Removes all occurrences of the *'Get Frame'* and *'Set Frame'* functions within the currently edited user function.
'Remove 'Set Tx Event' from function' (context menu item of frame)	Removes all occurrences of the "Set Tx Event" function within the currently edited user function.
'Remove 'Set Default Values' from function' (context menu item of frame)	Removes all occurrences of the "Set Default Values" function within the currently edited user function.
'Get PDU Update Bit'	Adds the following line u08 Updatebit_*sn* = get_*bn*_Updatebit_*id*(); to the currently edited function. The following abbreviations have been used: *sn* is the short name of the PDU. *Bn* is the name of the appropriate bus. *Id* is the FIBEX ID of the appropriate PDU. The function is a getter of the current state of the relevant Update Bit, which is here assigned to the variable Updatebit_*sn*. The return value is 0: Update Bit is set. 1: Update Bit is not set. This function is available for PDUs with update bits (including Contained PDUs with update bits)
'Set PDU Update Bit'	Adds the following lines `u08 Updatebit_sn = get_bn_Updatebit_id();` `set_bn_Updatebit_id(Updatebit_sn);` to the currently edited function. For the description of the first line and the appropriate abbreviations see *'Get PDU Update Bit'* in this table. In the second line the Update Bit in the database is set to the value which is given by the variable Updatebit_*sn*. This function is available for PDUs with update bits (including Contained PDUs with update bits)
'Remove all 'PDU Update Bit'' calls from function"	Removes all occurrences of the 'Get/Set PDU Update Bit' functions within the currently edited user function.

Menu Item	Description
'Set Custom Timing'	Adds the following line: `set_bn_PDUCustomTiming_id(timing);` *timing* is the Timing to be used. The custom timing is affected by the value "Base Timing Factor" (see chapter 5.6.7.2 for more details). **NOTICE** Setting the custom timing for a Contained PDU may negatively affect the transmission of all Contained PDUs. It is highly recommended to not change the custom timing of a Contained PDU.
'Remove all 'Set Custom Timing' calls from function"	Removes all occurrences of the *'Set Custom Timing'* functions within the currently edited user function.
'Set Default Timing'	Adds the following line: `set_bn_PDUDefaultTiming_id();` The value "Used Cyclic Timing", which is defined in the transmissinon settings dialog will be set as PDU timing.
'Remove all 'Set Default Timing' calls from function"	Removes all occurrences of the *'Set Default Timing'* functions within the currently edited user function.
'Set Alternative Timing'	Adds the following line: `set_bn_PDUAlternativeTiming_id();` The value "Used Alternative Timing", which is defined in the transmissinon settings dialog will be set as PDU timing.
'Remove all 'Set Alternative Timing' calls from function"	Removes all occurrences of the ''*Set Alternative Timing'* functions within the currently edited user function.
'Get PDU'	Adds the following line `PDU bn_sn = get_bn_PDU_sn();` to the currently edited function. Where *bn_sn* is the name of the locally generated variable. The following abbreviations have been used: *bn* is the name of the appropriate bus *sn* is the short name of the PDU With the function get_*bn*_PDU_*sn* the contents and the length of the PDU is returned. The used type PDU is a structure which contains the elements "data" and "length". Where "data" is a pointer of type (u08*) to the data of the PDU and "length" is the length of the data in bytes
'Set PDU'	Adds the following lines `PDU bn_sn = get_bn_PDU_sn();` `set_bn_PDU_sn(bn_sn);` to the currently edited function. For the description of the first line and the appropriate abbreviations see *'Get PDU'* in this table. In the second line the data for the relevant PDU is stored. The data is taken from the variable *bn_sn.*
'Set PDU' (Contained PDU)	See "Set PDU" The current payload of this contained PDU is updated.
'Get PDU' (Contained PDU)	See "Get PDU" The current payload of this contained PDU is returned.

Menu Item	Description
'Get Secured PDU'	Adds the following line `PDU bn_sn = get_bn_Secured_PDU_sn();` to the currently edited function. Where *bn_sn* is the name of the locally generated variable. The following abbreviations have been used: *bn* is the name of the appropriate bus *sn* is the short name of the PDU The function get_*bn*_Secured_PDU_*sn* accesses the payload and the trailer. It returns the payload and trailer data as a pointer of type (u08*) and the length of the data in bytes. This function is only available for secured PDUs.
'Set Secured PDU'	Adds the following lines `PDU bn_sn = get_bn_Secured_PDU_sn();` `set_bn_Secured_PDU_sn(bn_sn);` to the currently edited function. For the description of the first line and the appropriate abbreviations see *'Get Secured PDU'* in this table. In the second line the data (with trailer) for the relevant secured PDU is stored. The data is taken from the variable *bn_sn.* This function is available only for secured PDUs.
'Get Container PDU Raw' (Tx Container PDUs)	Adds the following line `CONTAINER_PDU FR1A_sn_ContainerIPdu = get_bn_Container_PDU_Raw_Pdus_PduId _ContainerIPdu();` to the currently edited function. Returns the current payload of the TX frame puffer. The field 'length' contains the maximum length of this container PDU.
'Set ContainerPDU Raw' (Tx Container PDUs)	Adds the following line `CONTAINER_PDU bn_sn_ContainerIPdu = get_bn_Container_PDU_Raw_Pdus_PduId_ContainerIPdu();` `set_bn_Container_PDU_Raw_Pdus_pduId (bn_pduId_ContainerIPdu);` to the currently edited function. Overwrites the payload of the TX frame puffer. Use the field "length" to set the size of the transmitted container PDU (and – depending of the bus type – the size of the transmitted frame). The limit for the size is the maximum size of the transmitted frame (all bus types).
AddToContainer (Contained PDU)	Adds the contained PDU to the transmission queue of the container PDU. Use the following syntax: `add_%BusID%_PDU_"Container PDU ShortName"();` Contained PDUs can have the following collection semantics: LAST-IS-BEST: The contained PDU can exist in the container PDU max. 1 time and contains the actual payload. QUEUED: The contained PDU can exist in the container PDU multiple times with different payloads. If the contained PDU has the collection type "LAST-IS-BEST" and the "*AddToContainer*" is used, the contained PDU will be added to the container with the actual payload, if it was not already added. If the container contains already the corresponding PDU, then just its payload will be updated. If the contained PDU has the collection type "QUEUED" and the "*AddToContainer*" is used, the contained PDU will be added to the container with the actual payload. The container can have the same contained PDU multiple times.

Menu Item	Description
'Set Tx-Event' (context menu item of PDU)	Adds the following line `set_bn_PDUTxEvent_sn();` to the currently edited function. The following abbreviations have been used: *bn* is the name of the appropriate bus. *Sn* is the id of the PDU With this function it is possible to set an event to trigger the transmission of a PDU. It is important to know that the transmission of a PDU also depend on the transmission settings (see chapter 5.6.7). This function does not automatically set the Update Bit of the corresponding PDU to "1". This must be done previously over *'Set PDU Update Bit'* function.
'Set Timeout' (context menu item of PDU)	Adds the following line `set_bn_PDUTxEvent_sn(to);` to the currently edited function. The following abbreviations have been used: *bn* is the name of the appropriate bus. *Sn* is the id of the PDU *to* is the timeout value This function is only available for PDUs, which are related with SOME/IP events. With this function it is possible to determine the timeout for an event. (E.g. The event will be sent immediately, if the timeout is 0)
'Set Default Values' (context menu item of PDU)	Adds the following line `set_bn_PDUDefault_sn();` to the currently edited function. The following abbreviations have been used: *bn* is the name of the appropriate bus. *sn* is the id of the PDU This function sets the default values of the signals of the selected PDU. See also chapter 5.9.
SetContainerRawTxEvent	Initiates the transmission of the selected container PDU. The container PDU will be transmitted with its current payload. This function can be used to bypass to default transmission behavior for container PDUs.
'Remove all PDU calls from Function'	Removes all occurrences of the *'Get PDU'* and *'Set PDU'* functions within the currently edited user function.
Remove 'Add to Container' functions (Contained PDU)	Removes all occurrences of the 'Add to Container' function within the currently edited user function.
Remove all Container PDU Calls from Function (Container PDU)	Removes all occurrences of the *'Get Container PDU Raw'* and *'Set Container PDU raw'* functions within the currently edited user function.
Remove 'Set Container Tx-Event Raw' from Function	Removes all occurrences of the '*Set Container Tx-Event Raw*' function within the currently edited user function.

Menu Item	Description
'Get Signal'	Adds the following line `type sn = get_bn_Signal_si();` to the currently edited function. The following abbreviations have been used: *type* of the signal variable (e.g. u08). *Sn* is the short name of the signal. *Bn* is the name of the appropriate bus. *Si* is the FIBEX-ID of the signal instance. The function call returns the current value of the appropriate signal. Signals up to 64 bits are supported.
'Set Signal'	Adds the following lines `type sn = get_Signal_bn_si();` `set_bn_Signal_si(sn);` to the currently edited function. For the description of the first line and the appropriate abbreviations see *'Get Signal'* in this table. In the second line the value of the signal in the database is set to the value of the variable *sn*. Signals up to 64 bits are supported.
'Get Physical Signal'	Adds the following line `double sn_physical = get_Physical_Signal_bn_si();` to the currently edited function. The following abbreviations have been used: *type* of the signal variable (e.g. double). *Sn* is the short name of the signal. *Bn* is the name of the appropriate bus. *Si* is the FIBEX-ID of the signal instance. The function call returns the current physical value of the appropriate signal. This function returns the calculated physical value of the signal if it has a supported compu-method (linear). In other cases, the internal value will be returned. Signals up to 64 bits are supported.
'Set Physical Signal'	Adds the following lines `type sn_physical = get_Physical_Signal_bn_si();` `set_Physical_Signal_bn_Signal_si(sn_physical);` to the currently edited function. For the description of the first line and the appropriate abbreviations see *'GetPhysical Signal'* in this table. In the second line the physical value of the signal in the database is set to the value of the variable *sn_physical*. This function sets the physical value of the signal if it has a supported compu-method (linear). In other cases, the used input parameter is considered as internal value. Signals up to 64 bits are supported.

Menu Item	Description
'Get Signal Update Bit'	Adds the following line u08 Updatebit_*sn* = get_*bn*_Updatebit_*id*(); to the currently edited function. The following abbreviations have been used: *sn* is the short name of the signal. *Bn* is the name of the appropriate bus. *Id* is the FIBEX ID of the appropriate signal. The function is a getter of the current state of the relevant Update Bit, which is here assigned to the variable Updatebit_*sn*. The return value is 0: Update Bit is set. 1: Update Bit is not set.
'Set Signal Update Bit'	Adds the following lines `u08 Updatebit_sn = get_bn_Updatebit_id();` `set_bn_Updatebit_id(Updatebit_sn);` to the currently edited function. For the description of the first line and the appropriate abbreviations see *'Get Signal Update Bit'* in this table. In the second line the Update Bit in the database is set to the value which is given by the variable Updatebit_*sn*.
'Set Default Value'	Adds the following line `type sn = get_bn_Default_si();` to the currently edited function. The following abbreviations have been used: *type* of the signal variable (e.g., u08). *sn* is the short name of the signal. *bn* is the name of the appropriate bus. *si* is the FIBEX-ID of the signal instance This function sets the default values of the selected signal. See also chapter 5.9.
'Remove all Signal Calls from function'	Removes all occurrences of the 'Get/Set Signal' functions within the currently edited user function.
'Remove all physical Signal Calls from function'	Removes all occurrences of the 'Get/Set Physical Signal' functions within the currently edited user function.
'Remove all 'Set Default Value' Calls from function'	Removes all occurrences of the 'Set Default Value' functions within the currently edited user function.
'Get Switch'	Adds the following line `u08 sn_lBit = get_bn_si();` The following abbreviations have been used: *sn* is the short name of the switch. *L* is the length in Bits. *Bn* is the name of the appropriate bus. *Si* is the FIBEX ID of the switch instance The function call returns the current value of the appropriate switch.

Menu Item	Description
'Set Switch'	Adds the following lines `u08 sn_lBit = get_bn_Switch_si();` `set_bn_Signal_si(_sn);` to the currently edited function. For the description of the first line and the appropriate abbreviations see *'Get Switch'* in this table. In the second line the value of the switch in the database is set to the value of the variable *sn_*lBit. Invalid switch values will be taken over without any change or validation!
'Remove all Switch Calls from function'	Removes all occurrences of the 'Get/Set Switch' function within the currently edited user function.
'Find Usages'	Opens a window that lists all user functions which use one of the *'Get/Set x'* functions of the selected item, where x can be a Frame, Tx Event, PDU, Signal, PDU Update Bit or Switch (see chapter 5.8.3.10).

Table 75: User Function Editor – Description of the context menu for section "Network"

NOTICE

If a project uses multiple instantiated PDUs (instantiation of a PDU in more than one frame), then the access to elements (signals, update bits, switches) related to the PDU is independent of the instantiation!

This is due to the fact, that a PDU is only stored once within the internal database.

The User Function Editor also supports service items (events, fields, notifiers, methods) and their complex datatypes:

Menu Item	Description
'Trigger Event'	Adds the function call `set_bn_TriggerEvent_sn_srvid_srvins_major_minor();` to the currently edited function. The following abbreviations have been used: *bn* is the name of the appropriate ethernet bus. *Sn* is the short name of the event. *Srvid* is the service id. *Srvins* is the service instance id. *major* is the major version of service interface *minor* is the minor version of service interface This function triggers the transmission of the corresponding event (for provided services).
'Trigger Notifier'	Adds the function call `set_bn_TriggerNotifyField_sn_srvid_srvins_major_minor();` to the to the currently edited function. The following abbreviations have been used: *bn* is the name of the appropriate ethernet bus. *Sn* is the short name of the field. *Srvid* is the service id. *Srvins* is the service instance id. *major* is the major version of service interface *minor* is the minor version of service interface This function triggers the transmission of the corresponding field notifier (for provided services).

Menu Item	Description
'Call Method'	Adds the function call `set_bn_TriggerCallMethod_sn_srvid_srvins_csid_major_minor();` to the to the currently edited function. The following abbreviations have been used: *bn* is the name of the appropriate ethernet bus. *Sn* is the short name of the method. *Srvid* is the service id. *Srvins* is the service instance id. *Csid* is the id of the consumed service instance *major* is the major version of service interface *minor* is the minor version of service interface This function triggers the transmission of the corresponding method call (for consumed services).
'Trigger Get Field'	Adds the function call `set_bn_TriggerGetField_sn_srvid_srvins_major_minor();` to the to the currently edited function. The following abbreviations have been used: *bn* is the name of the appropriate ethernet bus. *Sn* is the short name of the field. *Srvid* is the service id. *Srvins* is the service instance id. *major* is the major version of service interface *minor* is the minor version of service interface This function triggers the transmission of the corresponding field getter (for consumed services).
'Trigger Respond Get Field'	Adds the function call `set_bn_TriggerRespondGetField_sn_srvid_srvins_major_minor();` to the to the currently edited function. The following abbreviations have been used: *bn* is the name of the appropriate ethernet bus. *Sn* is the short name of the field. *Srvid* is the service id. *Srvins* is the service instance id. *major* is the major version of service interface *minor* is the minor version of service interface This function triggers the responding of the corresponding field getter (for provided services). The function is normally used in a "Get Field Receive User Function" by the provider for sending the value of the field.

Menu Item	Description
'Trigger Set Field'	Adds the function call `set_bn_TriggerSetField_sn_srvid_srvins_major_minor();` to the to the currently edited function. The following abbreviations have been used: *bn* is the name of the appropriate ethernet bus. *Sn* is the short name of the field. *Srvid* is the service id. *Srvins* is the service instance id. *major* is the major version of service interface *minor* is the minor version of service interface This function triggers the transmission for the corresponding field setter (for consumed services).
'Trigger Respond Set Field'	Adds the function call `set_bn_TriggerRespondSetField_sn_srvid_srvins_major_minor();` to the to the currently edited function. The following abbreviations have been used: *bn* is the name of the appropriate ethernet bus. *Sn* is the short name of the field. *Srvid* is the service id. *Srvins* is the service instance id. *major* is the major version of service interface *minor* is the minor version of service interface This function triggers the responding of the corresponding field setter with the actuel field value (for provided services). The function is normally used in a "Set Field Receive User Function" by the provider for sending the value of the field and makes less sense in other places

Table 76: User Function Editor – Description of the context menu for section "Network-Service Items"

The service elements may have simple (int, short) or complex data types (struct), In the User Function Editor it is possible to access to the last level of a complex data type.

Following example explains how to access to the data of a service item. We assume to have a service event with following properties (s. Figure 104):
Event name: "Position"
First input parameter: "Number" (a simple datatype e.g. u08)
Second input parameter: An array of the struct "Coordinate" with the length 3.
The struct "Coordinate" has 3 properties: "X", "Y" and an array of the datatype 'Info' with the length 5.

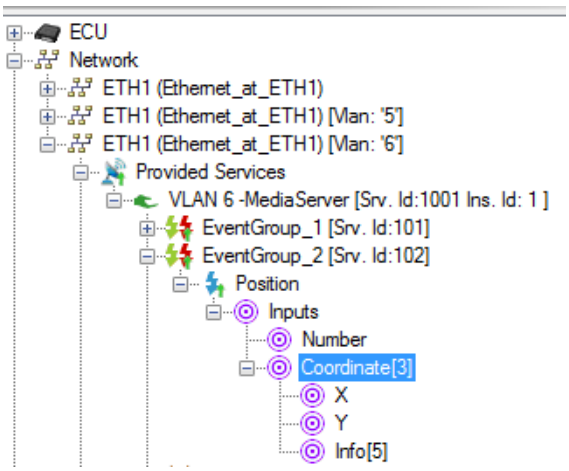


Figure 104: User Function Editor – Example data structure for an event

It is possible to access to the members “Number”, “X”, “Y” and “Info”. The “get”, “set”, “get Physical” and “set Physical” methods can be called for these members via the context menu.

Data types up to 64 bits are supported.

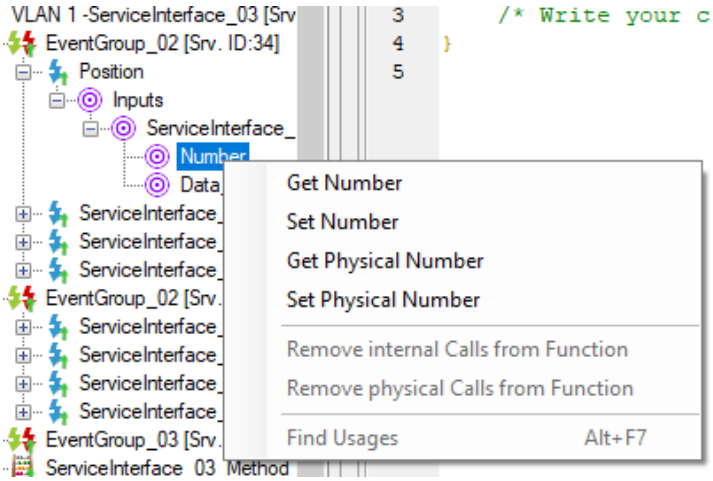


Figure 105: User Function Editor – Get/Set functions for data members

Following pattern is used for the get and set methods:

```
get/set_bn_iof_sn_parentn…parent1_mn_srvid_srvins_csi_major_minor(i1, i2..in);
```

Following pattern is used for the get and set methods:

```
get/set_Physical_bn_iof_sn_parentn…parent1_mn_srvid_srvins_csi_major_minor(i1,
i2..in);
```

The following abbreviations have been used:
bn: is the name of the appropriate ethernet bus.
Iof: can be the text “inputparameter”, “outputparameter” or “field”
sn: is the short name of the service element (field, event, method)
parentn…parent1 are just then available if the data type is the member of another data type
mn is the member name
srvid is the service id.
Srvins is the service instance id.
Csi is the consumed service instance id and. It is used just by consumed service methods
i1…in are just then available if the data member is an array
For the example above, following code will be generated for the get/set function. Please consider the code comments:

```
/*MediaServer [ SrvId:1001 Ins. Id: 1 Major: 1 Minor: 0] -> EventGroup_1 -> Position
-> Number*/
u08 Number = get_ETH1_inputparameter_Position_Number_1001_1_1_0();
```

```
set_ETH1_inputparameter_Position_Number_1001_1_1_0 (Number);

/*MediaServer [ SrvId:1001 Ins. Id: 1 Major: 1 Minor: 0]] -> EventGroup_1 -> Position -
> Coordinate -> X*/
/* 1. Parameter:  Index of 'Coordinate' */
u08 X = get_ETH1_inputparameter_Position_Coordinate_X_1001_1(0);
set_ETH1_inputparameter_Position_Coordinate_X_1001_1_1_0 (0,X);

/*MediaServer [ SrvId:1001 Ins. Id: 1 Major: 1 Minor: 0]] -> EventGroup_1 -> Position -
> Coordinate -> Y*/
/* 1. Parameter:  Index of 'Coordinate' */
u08 Y = get_ETH1_inputparameter_Position_Coordinate_Y_1001_1_1_0 (0);
set_ETH1_inputparameter_Position_Coordinate_Y_1001_1_1_0 (0,Y);

/*MediaServer [ SrvId:1001 Ins. Id: 1 Major: 1 Minor: 0]] -> EventGroup_1 -> Position -
> Coordinate -> Info*/
/* 1. Parameter:  Index of 'Coordinate' */
/* 2. Parameter:  Index of 'Info' */
u08 Info = get_ETH1_inputparameter_Position_Coordinate_Info_1001_1_1_0 (0,0);
set_ETH1_inputparameter_Position_Coordinate_Info_1001_1_1_0 (0,0,Info);
```

The get/set Physical function for data types returns or sets the current physical value of the appropriate data type, if it has a supported compu-method (linear). In other cases, the internal value will be returned or set.

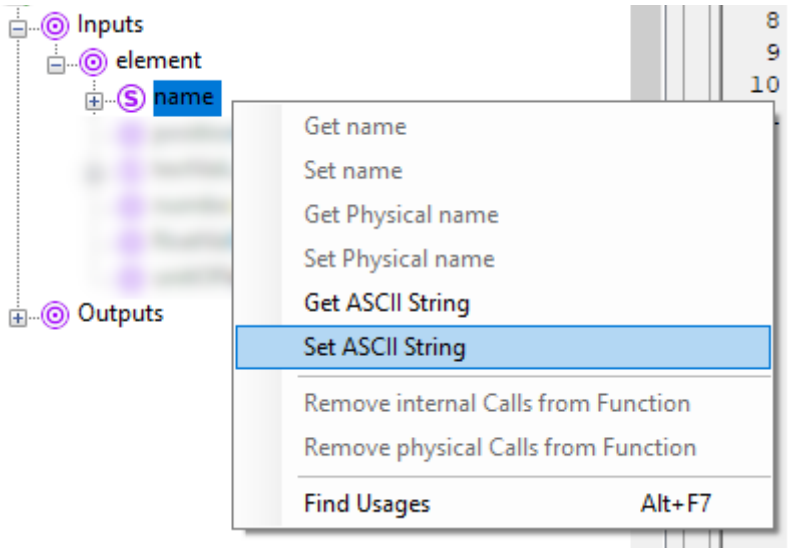


Figure 106 User Function Editor – Context menu for get/set ascii string functions

If the datatype is a string, the get/set ASCII string functions will be available. These functions allow you to get or set a char* that contains the content of the datatype. Note that currently, only ASCII, UTF-8, and UTF-16 strings are supported. When calling the functions on UTF-8 or UTF-16 strings, characters outside the ASCII range will be replaced by a placeholder symbol.

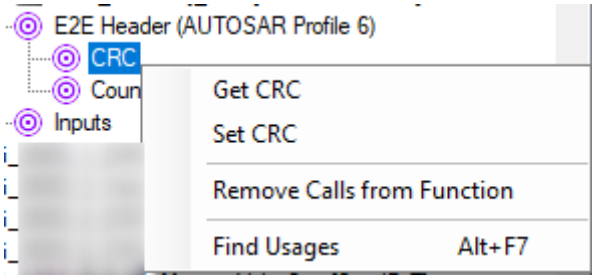


Figure 107: User Function Editor – Context menu for stacked-based E2E profiles

If an ethernet service item (e.g., event) is protected with a stacked-based E2E profile (e.g AUTOSAR E2E Profiles 4, 6, 7) it will have a “E2E Header” node by the signals and variables tree. This node contains the sub nodes “CRC” and “Counter”. The corresponding API functions are used to get or set the crc/counter values.

If an ethernet service item (e.g., event) is protected with a signal-based E2E profile (e.g AUTOSAR E2E Profiles 1, 2, 5), then the service item should normally have data types for CRC and Counter values. If these data types are missing in the database, then the CRC and Counter Signals are shown under the service item:

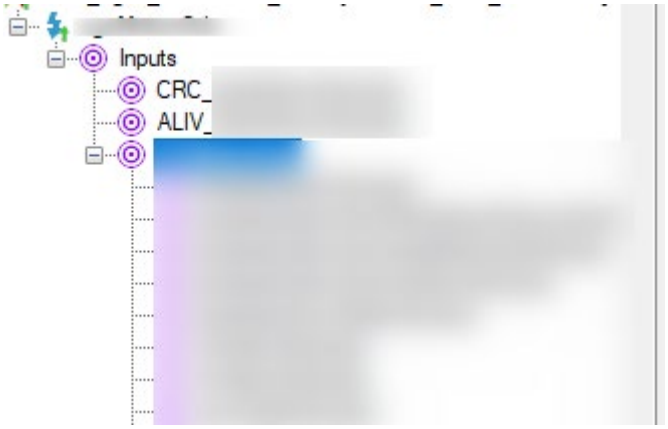


Figure 108: User Function Editor – CRC and Counter Signals with missing data types

The get function for CRC returns a u64 and the get method for the counter returns a u32 value. The get and set functions have the same structure as trigger service item functions and have as difference the special texts “CRC” and “CNT” instead of “Trigger”.

Arrays with dynamic sizes under service elements can be also configured via api functions. Following functions are available after right clicking on an array item:

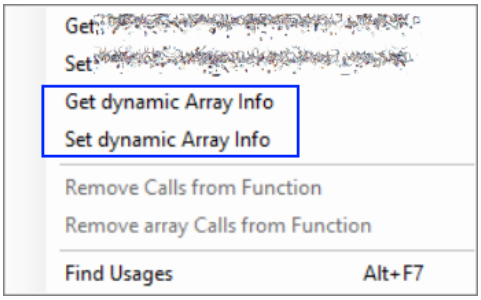


Figure 109: User Function Editor – context menu arrays with dynamic sizes

Menu Item	Description
'Get dynamic Array Info'	Adds the following line `ARRAYINFO sn_ArrayInfo = get_ArrayInfo…(…);` The following abbreviations have been used: *sn* is the short name of the data type. The function may have input parameters, if any parent node of the array is also an array. In this case the user must select the index of the parent data types. The added function has a name which is constructed with the combination of short names of the datatype (array) and its parents. The returned ARRAYINFO structure makes it possible to change the length of each array dimension. It has following properties: number_of_dimensions: Number of dimensions. Array_length: Array, which contains the actual count of elements for each dimension. The lengths can be changed for each array dimension. Array_min_length: Array, which contains minimum count of elements for each dimension. Array_max_length: Array, which contains maximum count of elements for each dimension.
'Set dynamic Array Info'	Adds the following line `set(…, ARRAYINFO);` The function may have input parameters, if any parent node of the array is also an array. In this case the user has to select the index of the parent data types. The added function has a name which is constructed with the combination of short names of the datatype (array) and its parents. The ARRAYINFO parameter is usually returned by the *'Get dynamic Array Info'* function.

5.8.3.3 Global Variables

To support the communication between the different user functions, global variables can be provided. For this purpose, the "Global Variables"-item exists. The child items of this item are all created global variables.

If the "Global Variables"-item is right clicked a context menu is shown (see Figure 110).

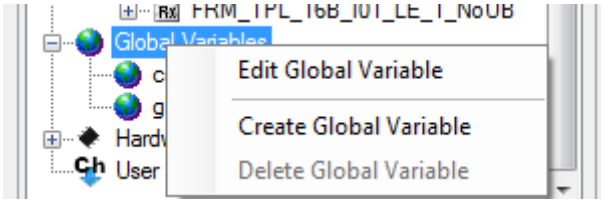


Figure 110: User Function Editor – context menu for all Global Variables

To create a new global variable, the option '*Create Global Variable*' must be clicked. After that the following dialog appears (see Figure 111), where the datatype, name, length (count) and the value of the global variable can be selected/entered.

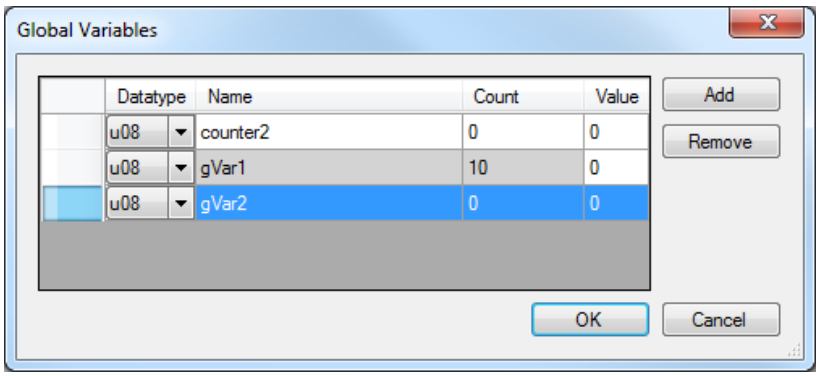


Figure 111: User Function Editor – “Global Variables” dialog

If the global variable shall be a normal global variable, the length (count) must be zero. Otherwise if a length is chosen greater than zero an array of global variables will be created. Within the dialog further global variables can be added by using the “*Add*”-button. Otherwise global variables can be removed by using the “*Remove*”-button.

To edit, remove or use the functionality of global variables within a user function, the context menu of the appropriate global variable must be opened (see Figure 112).

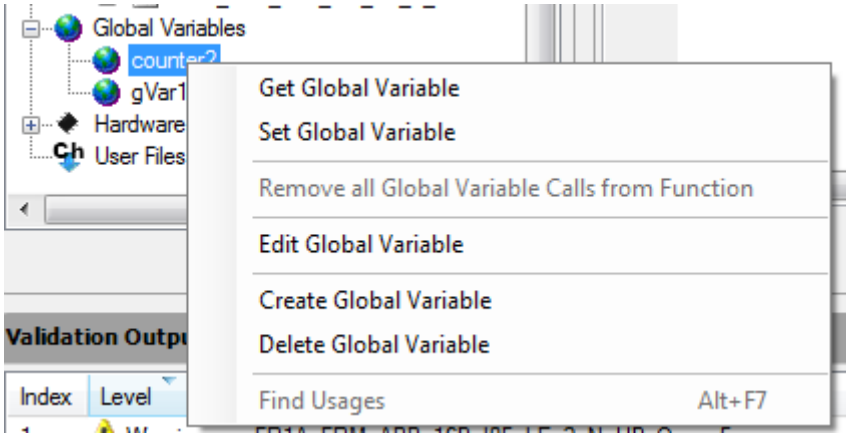


Figure 112: User Function Editor – context menu for a selected Global Variable

All context menu items are explained in Table 77.

Menu Item	Description
'Get Global Variable' (normal)	Adds the following line `type gn = get_GlobalVar_gn();` to the currently edited function. Where *type* is the selected type of the global variable and *gn* is the name of the global variable. The function call returns the current value of the appropriate global variable to a locally defined variable *gn*.
'Get Global Variable' (array)	Adds the following line `t_global_array_type gn = get_GlobalVar_gn();` to the currently edited function. Where *type* is the selected type of the global variable and *gn* is the name of the global variable. The function call returns the value of the appropriate global variable. The type of the return value is a struct that contains a "data" field and a "length" field. The "data" field is a pointer to a field of the selected type. Due to the type pointer of the "data" field a menu item "Set Global Variable" (array) is not necessary and not provided! The length of the array is given in the "length" field.
'Set Global Variable' (normal)	Adds the following lines `type gn = get_GlobalVar_gn();` `set_GlobalVar_gn(gn);` to the currently edited function. Where *type* is the selected type of the global variable and *gn* is the name of the global variable. For the description of the first line see "Get Global Variable" (normal) in this table. In the second line the value of the global variable is set to the value of the variable *gn*.
'Remove all Global Variable Calls from function'	Removes all occurrences of the "Get/Set Global Variable" function within the currently edited user function.
'Edit Global Variable'	Opens the "Global Variables" editor.
'Create Global Variable'	Opens the "Global Variable" editor and a new global variable is automatically generated (named "gVarx").
'Delete Global Variables'	Deletes the appropriate global variable from the "Global Variables"-folder. Requirement for the execution of this action is that the global variable is not already used within a user function!
'Find Usages'	Opens a window that lists all user functions which use one of the "Get/Set" functions of the selected global variable (see chapter 5.8.3.10).

Table 77: User Function Editor – Description of the context menu for Global Variables

	Information
	Accessing global arrays with the PC-Hardware Interface is restricted to the first 1024 elements. The global array can have more elements but only first 1024 will be returned. Accessing the global array via User Functions is not restricted.

	NOTICE
	FL3X Config Control does not support manipulations for global variables if the data type is 'float', 'double', 'u64' or 's64'. Such global variables are also not shown in FL3X Config Analyzer.

5.8.3.4 Hardware

The “Hardware”-folder of the “Signals and Variables”-section contains the direct child items “Analog I/O”, “Digital I/O” and “Sleep Mode”. As shown in Figure 100 these direct child items except “Sleep Mode” again contain child items like “AD_IN_x”, “Dox_off” or “Dox_on”.

By right clicking on one of these items e.g. “AD_IN_1” a context menu is shown (see Figure 113).

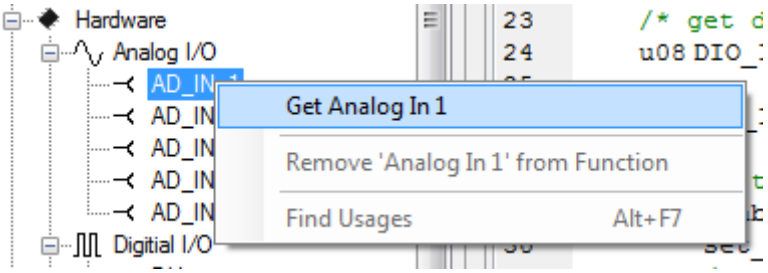


Figure 113: User Function Editor – context menu for “Hardware->Analog I/O”

All accessible menu items within the “Hardware” folder are explained in Table 78.

Menu Item	Description
‘Get Analog In x’	Adds the following line `u32 AD_IN_x = get_AD_IN_x();` to the currently edited function. Where *x* is the input-number of the ADI connector (1-5). The function call returns the value of the voltage at the appropriate ADI Pin. The value can be in the range 0x0 – 0x3FFF which corresponds to the voltage range 0V-55V (the exact PIN assignment is given in Table 79).
‘Remove ‘Analog In x’ from Function’	Removes all occurrences of the “get_AD_IN_*x*” function within the currently edited user function.
‘Get Digital In 1’	Adds the following line `u08 DIO_IN_1 = get_DIO_IN_1();` to the currently edited function. The function call returns a value of 0: no rising edge detected at DI 1 1: rising edge detected at DI 1 where DI 1 is the digital input 1 of the DIO connector (the exact PIN assignment is given in Table 80).
‘Remove ‘Get Digital In 1’ from Function’	Removes all occurrences of the “get_DIO_IN_1” function within the currently edited user function.
‘Get Is Digital In 1 High’	Adds the following line `u08 IS_DIO_IN_1_HIGH = get_DIO_IN_1_IS_HIGH ();` to the currently edited function. The function call returns a value of 0: The DI 1 is not on high level 1: The DI 1 is on high level
Remove ‘Get Is Digital In 1 High’ from Function’	Removes all occurrences of the “get_DIO_IN_1_IS_HIGH” function within the currently edited user function.
‘Get Is Digital In 1 Low’	Adds the following line `u08 IS_DIO_IN_1_LOW = get_DIO_IN_1_IS_LOW ();` to the currently edited function. The function call returns a value of 0: The DI 1 is not on low level 1: The DI 1 is on low level
Remove ‘Get Is Digital In 1 Low’ from Function’	Removes all occurrences of the “get_DIO_IN_1_IS_LOW” function within the currently edited user function.

Menu Item	Description
'Reset Digital In 1'	Adds the following line `reset_DIO_IN_1();` to the currently edited function. Calling this function resets the value which you get by "get_DIO_IN_1" to zero.
'Remove 'Reset Digital In 1' from function'	Removes all occurrences of the "reset_DIO_IN_1" function within the currently edited user function.
'Set Digital Out x Off'	Adds the following line `set_DIO_OUT_x_OFF();` to the currently edited function. Where *x* is the number of the output (2-3). Calling this function sets the Dox output to zero (the exact PIN assignment is given in Table 80).
'Remove 'Set Digital Out x Off' from Function'	Removes all occurrences of the "set_DIO_OUT_x_OFF" function within the currently edited user function.
'Set Digital Out x On'	Adds the following line `set_DIO_OUT_x_ON();` to the currently edited function. Where *x* is the number of the output (2-3). Calling this function sets the Dox output to one (the exact PIN assignment is given in Table 80).
'Remove 'Set Digital Out x Off' from Function'	Removes all occurrences of the "set_DIO_OUT_x_ON" function within the currently edited user function.
'Set Sleep'	Adds the following line `flextinys_enter_sleep_mode();` to the currently edited function. If this function is called, the device (FL3X Device-S, FL3X Device-L) immediately enters the sleep mode and power consumption is reduced to a minimum.
'Remove 'Set Sleep' from Function'	Removes all occurrences of the "flextinys_enter_sleep_mode" function within the currently edited user function.
'Find Usages'	Opens a window that lists all user functions which use one of the functions listed in this table (s. chapter 5.8.3.10).
Is_User_Led_%Color%_on/off	Functions to get the state of the User LED with the selected color.
User_Led_%Color%_on/off/toggle	Functions to set the state of the User LED with the selected color.

Table 78: User Function Editor – Description of the context menu for "Hardware" elements

NOTICE

The FL3X System-M do not support LED functions.

5.8.3.4.1 ADI-Connector (Analog)

There are five analog inputs available on the FL3X Device-L. To access the data of these inputs within a user function the menu item *'Get Analog In x'* (s. Table 78) can be clicked.

The pin assignment of the ADI connector is described in the following table.

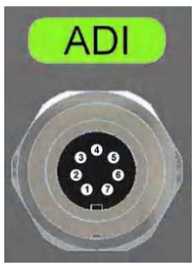
Pin out: Binder-Connector	Pin out: Sub-D (Solder-side view)	User Function Editor
		PIN 2 = AD_IN_1 PIN 3 = AD_IN_2 PIN 4 = AD_IN_3 PIN 5 = AD_IN_4 PIN 6 = AD_IN_5

Table 79: FL3X Device-L – Pin out of Analog-In

	NOTICE
	The FL3X System-XS and FL3X System-XS² support only AD_IN_2.

	Information
	The input can measure 0.05V to 55V which will lead to measured digital values from 0x000F (0.05V) to 0x3FFF (55V). The values below 0.05V (0-0.05V) will correspond to digital values of nearly 0x000F.

5.8.3.4.2 DIO-Connector (Digital)

There are one digital input and two digital outputs available via User Functions for FL3X Device. In order to access the data of these inputs/outputs within a user function the menu items *'Get Digital In 1'*, '*Reset Digital In 1'*, '*Set Digital Out x Off*', *'Set Digital Out x On'* can be used (s. Table 78).

Especially the resulting functions "set_DIO_OUT_x_ON/OFF" can e.g., be used as a trigger-output of the FL3X Device. They might be used to control measurement equipment like an oscilloscope.

The pin assignment of the DIO connector is described in the following table.

Pin out: Binder-Connector	Pin out: Sub-D (Solder-side view)	User Function Editor
		PIN 2 = DI1 PIN 3 = DO2 PIN 4 = DO3 PIN 6 = U_Bat PIN 7 = 5V -**DI1** (see FL3X Device-L Instruction for use [1]) -**DO2/DO3** (see FL3X Device-L Instruction for use [1])

Table 80: FL3X Device-L – Pin out of Digital I/O's

	Information
	The default range is 0-55V (ADI, DIO).

	NOTICE
	The FL3X Device-L/L² support different range configurations between ADI and DIO. This is not supported for the FL3X Device-S (the range configuration is always the same for ADI and DIO).

NOTICE
The devices FI3X Device L/L² support 5 ADI **AND** 4 DIO. The device FL3X Device-S supports 5 ADI **OR** 4 DIO. Every pin used as ADI is no longer available as a DIO. The devices FL3X Device-PXIe and FL3X Device-PCIe support only 2 DIO. The FL3X System-XS and the FL3X System-XS² do not support DO2_off and DO2_on. The FL3X System-M does not support ADI und DIO.

5.8.3.4.3 Sleep Mode, Hardware Settings (FL3X Device-L and FL3X Device-L²)

The Sleep and Wake-Up behavior of the FL3X Device-L/FL3X Device-L² can be set per software. The settings of the Wake-Up and Sleep behavior can be done within the Device Manager (see chapter 5.2.2).

Reference
For more information about the hardware of the FL3X Device-L or FL3X Device-L² please refer to: “3-0087-0A01-D19-FL3X Device-L Instruction for use” [1]

5.8.3.4.4 Sleep Mode, Hardware Settings (FL3X Device-S)

The Sleep and Wake-Up behavior of the FL3X Device-S can be set per software. The settings of the Wake-Up and Sleep behavior can be done within the Device Manager (see chapter 5.2.2).

Reference
For more information about the hardware of the FL3X Device-S please refer to: “3-0086-0A01-D09-FL3X_FL3X Device-S Instructions for Use” [3]

5.8.3.4.5 Sleep Mode, Hardware Settings (FL3X Device-PXIe, FL3X Device-PCIe)

The FL3X Device-PXIe and the FL3X Device-PCIe do not support Sleep and Wake-Up behavior. Any “set sleep” function call in User Functions is ignored at runtime. To stop all transmission for a bus (and therefore simulate a ‘sleep’ behavior), use the “Bus Tx Off” functions (see Table 75).

5.8.3.5 User Libraries

‘User Libs’ are common or customer specific user libraries, which are created by STAR ELECTRONICS for customer specific solutions. A ‘User Lib’ is a single file with the extension ‘*.ulib*’, which can be added to the project via context menu and may have multiple functions. User libraries are delivered with a document which describes it. After adding a user library, all its functions are available under its node.

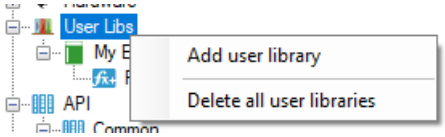


Figure 114: User Function Editor – context menu for “User Libs”

Menu Item	Description
‘Add user library’	Opens a file dialog box, where a *.ulib* file can be added to the “User Libs” node.
‘Delete all user libraries’	Removes all previously added user libs from the project.

A User Lib may have following context menu items:

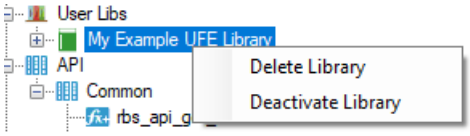


Figure 115: User Function Editor – context menu for a single “User Lib”

Menu Item	Description
'Delete Library'	Removes the selected user library.
'Deactivate Library'	Deactivates the selected user library. Its functions cannot be used in User Function Editor.
'Activate Library'	Activates a user library.

Tooltips can be used to get more information about each user library function:

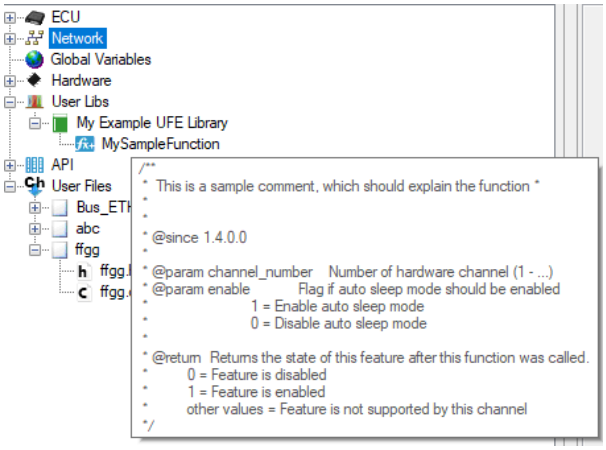


Figure 116: User Function Editor – Tooltip of user library functions

Per drag & drop or over context menu each function of the user lib can be added to the user code. The added function is commented out and must explicitly be activated in code with proper parameters.

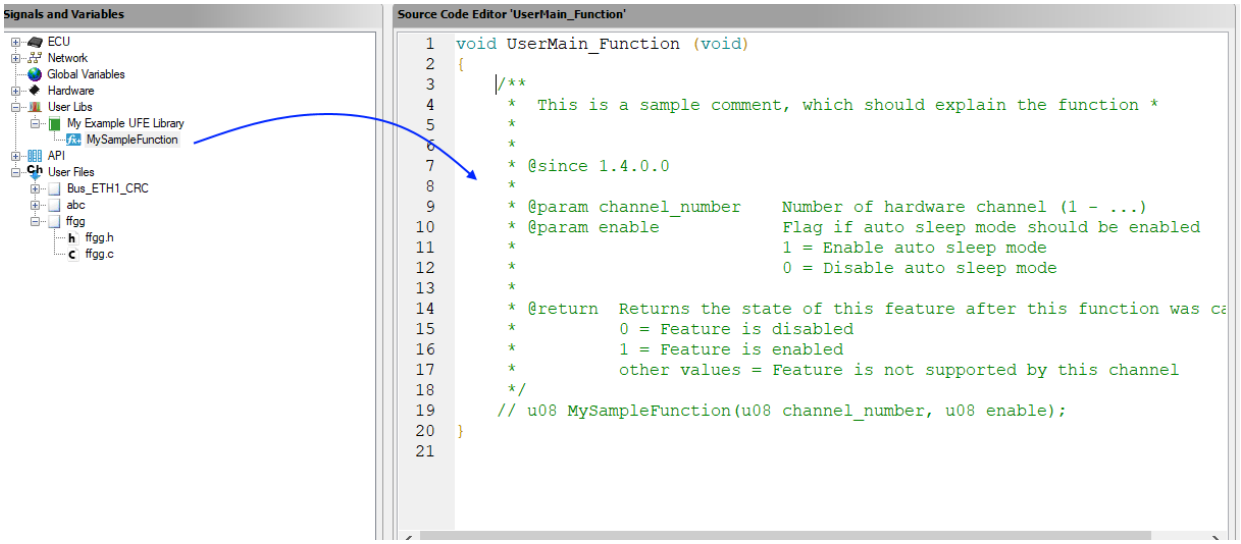


Figure 117: User Function Editor – Adding user lib function to code

NOTICE

User Functions can be modified with Visual Code if you have a FL3X Config Pro-License. Projects with ulibs are **not** supported for modification via Visual Code.

5.8.3.6 Math and Memory Functions

In the Math and Memory nodes, useful functions can be found that can be added to the code by drag and drop or via the context menu. The added code also contains a brief description of the function. The following functions are available:

Math:

- acos
- asin
- atan

- atan2
- ceil
- cos
- cosh
- exp
- fabs
- floor
- fmod
- frexp
- ldexp
- log
- log10
- modf
- pow
- sin
- sinh
- sqrt
- tan
- tanh

Memory:

- memcpy
- memmove
- memcmp
- memset
- strcpy
- strcat

5.8.3.7 RBS API

The ‘RBS API’ node contains multiple API-Functions, which can be used to access to the FL3X Device and get/set proper data. Tooltips can be used for the API-Functions for getting more information about them. An API-Function can be added per drag & drop or via context menu to the user code. The added code will also contain a sample usage of the corresponding function. See also chapter 5.8.6.

5.8.3.8 User Files

In some cases, it can be necessary to add additional .c/.h files (or folders with .c/.h files), to use additional functions.

The “User Files”-folder (s. Figure 100) contains all user files and folders which have been added to the project. Adding to the project means that the files/folders are copied to the internal “usr”-folder within the project folder.

All copied files, folders and subfolders are shown in tree view. As can be seen in Figure 100 for all .c/.h-files separate parent folders are created. The name of the parent folder is the name of the file without extension. In this way all .c/.h files with the same name are grouped.

If the “User Files”-folder will be right clicked, the following context menu appears (see Figure 118).

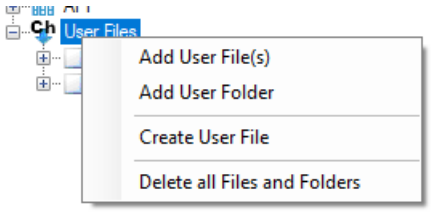


Figure 118: User Function Editor – context menu for “User Files”

The context menu appropriate to a user file within the “User Files”-folder will additionally contain the menu items *‘Edit File’* and *‘Delete File’*.

The context menu appropriate to an added folder will additionally contain the menu item *‘Delete Folder’*.

All menu items are described in detail in Table 81.

Menu Item	Description
‘Add User File(s)’	Opens a file dialog box, where *.h/.hpp/.hxx/.c/.cpp/.cxx/.cc* - files can be added to the “User Files”-folder.
‘Add User Folder’	Opens a dialog box, where a folder can be chosen, which will be copied to the project subfolder “usr”.
‘Create User File’	Opens a dialog, where a file name can be set. After closing the dialog, a header and a code file with the given name will be created and added as user files, whose contents can be filled/edited.
‘Delete all Files and Folders’	Removes all previously added files and folders from the project.
‘Delete Folder’	Deletes the selected folder. This can only be done with root folders, which have been added to the project!
‘Edit File’	The selected file is opened in an editor. Which editor will be used, can be set in the settings menu (see chapter 5.22.1.4).

Table 81: User Function Editor – Description of the context menu items for “User Files”

NOTICE

The user functions contained in included C++ files (header or code) are not supported directly in the User Function Editor. Because FL3X Config uses a C compiler, a wrapper header (.h) file is required to use the C++ functions.

NOTICE

If a folder has been added to the project none of the contained .c/.h files is used in the first step! To have access to variables, defines, functions within this folder structure, it is necessary to write a master header file or make it available. This must include all the functionality needed.

The master header file finally must be added to the “User Files” root-folder.

Example:

A user File “*File1.h*” following code:

```
u08 MyFunction1();…
```

A user File “*File1.c*” following code:

```
#include File1.h
u08 MyFunction1()
{
      return 0;
}
```

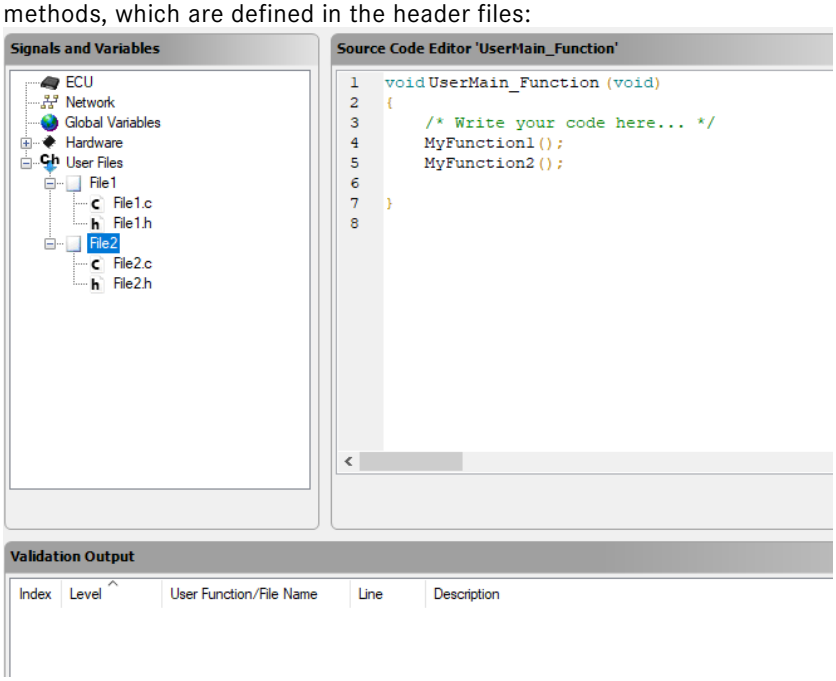
A user File “*File2.h*” following code:

```
u08 MyFunction2(); …
```

A user File “*File2.c*” following code:

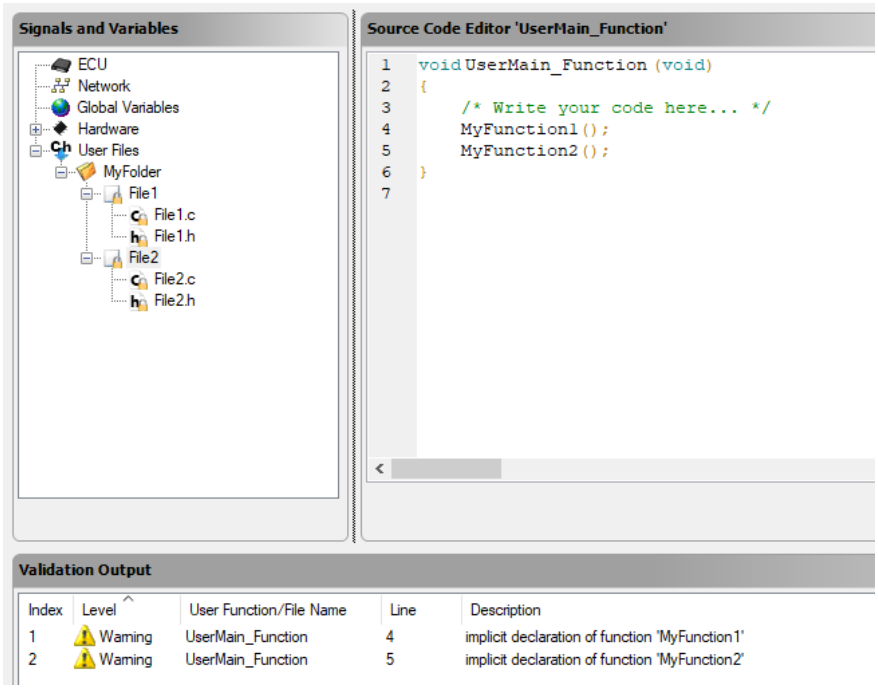
```
#include File2.h
u08 MyFunction2()
{
      return 0;
}
```

Case 1: The **user files** are added to the user function editor directly and any user function can call the methods, which are defined in the header files:



Case 2: We want to add a **folder**, which contains all files to the user function editor.

A folder “*MyFolder*” contains the files “*File1.h*”, “*File1.c*”, “*File2.h*” and “*File2.c*”. In this case the corresponding methods **cannot** be called (s. warnings in the following screenshot).

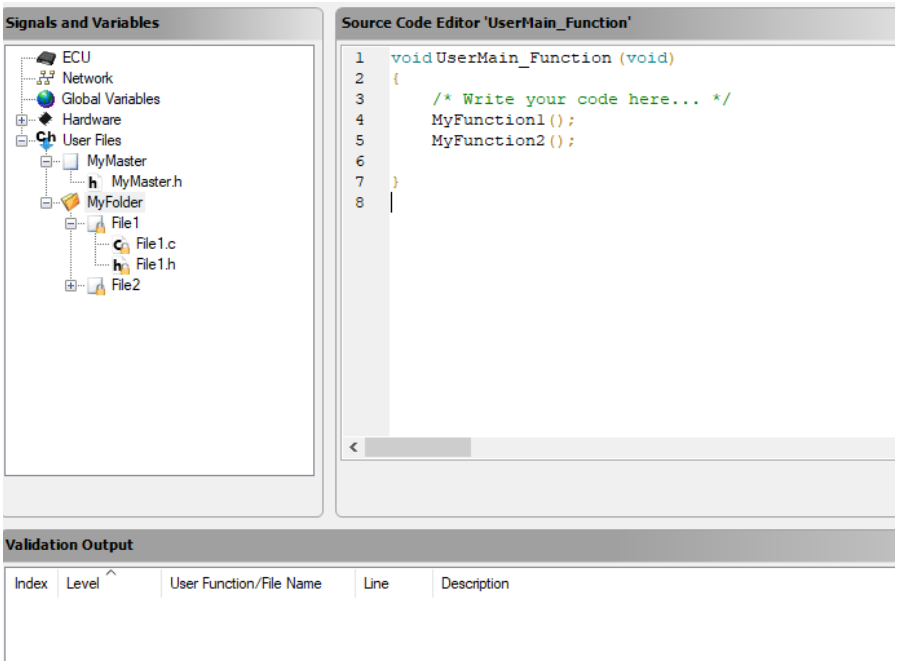


To solve this problem, we must create a new header file which have references to the header files in the “*MyFolder*” folder.

E.g., we create a new file “*MyMaster.h*” on the same level with the “*MyFolder*” folder with the following content:

```
#include "MyFolder\File1.h"
#include "MyFolder\File2.h"
```

Now we can add this file to the user function editor and call the methods:



	NOTICE
	The direct editing of the added *.c/.h*-file in their original paths does not have any influence on the project!

	NOTICE
	To prevent any naming conflicts, use your own prefix when defining a global variable or other elements in a user file. E.g., instead of "gVar1" (which is the default name for global variables created by the User Function Editor), you could use "mygVar1". There are prefixes that are not allowed for function: Do not create/use any function with prefix "get" or "set".

Internally all programmed user functions of the "User Function Editor" will be concatenated to one *.c*-file. In this file all header files, which have been added to the "User Files"-root-folder, will be included in alphabetical order. This must be kept in mind to prevent compilation errors.

5.8.3.9 Removing Functions

A function, which provides access to one of the items (signals, frames, global variables, ...) described above, can be removed by opening the context menu of the relevant item. After that, the *'Remove all x calls from Function'*-menu item can be applied (where *x* e.g., corresponds to "Signals"). If this has been done, all lines are removed, where functions of item *x* are called.

For example, if a signal is chosen all lines where the get and set functions of the signal are used, are removed. Additionally, all comments above the removed lines, are also removed until the next uncommented line appears. If the removing of comments is unwanted, an empty line between the comment and the code line should be inserted.

5.8.3.10 Find Usages

If and above all where the different functions of the different items (Frames, ECUs, signals, ...) are used, can be examined by using the context menu of the relevant item (e.g., signal) and then clicking the menu item *'Find Usages'*. After this has been done the following "Find Usages" window appears (see Figure 119).

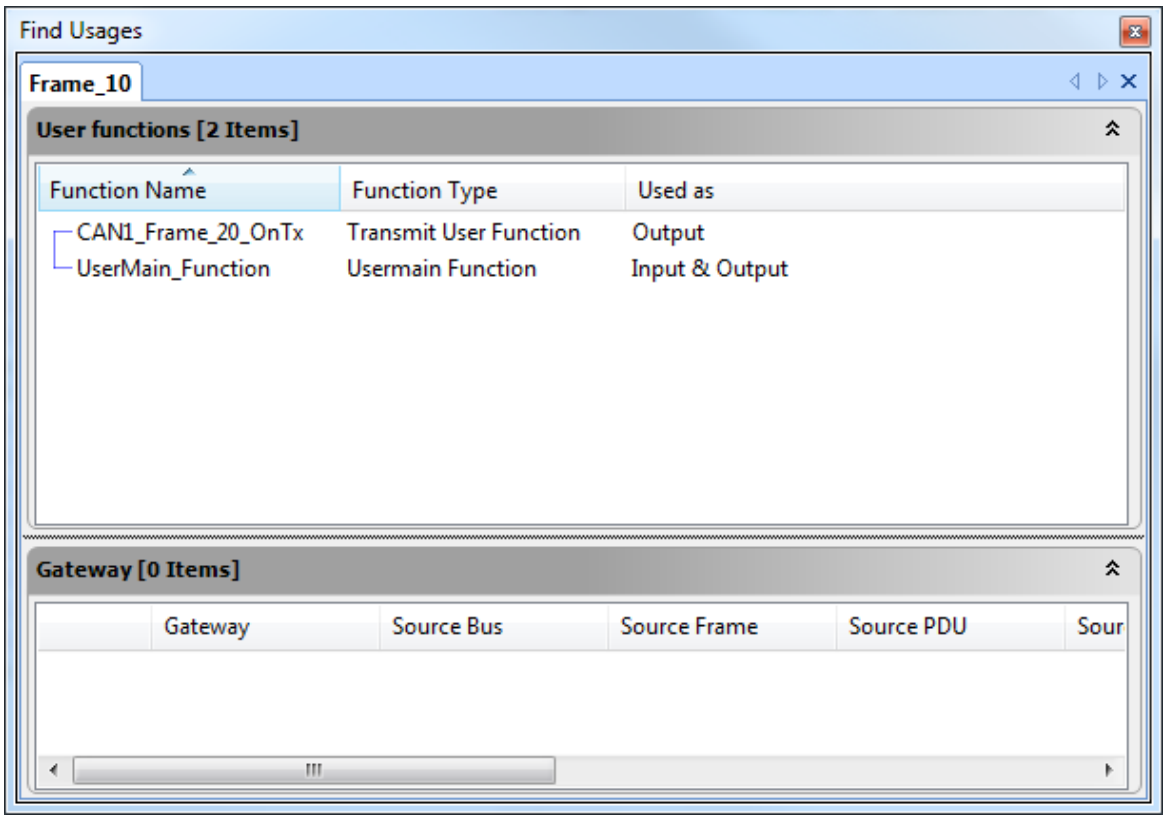


Figure 119: User Function Editor – “Find Usages” dialog

The columns of the list ‘user functions’ are described in Table 82.

Column Name	Description
“Function Name”	Name of the user function.
“Function Type”	The type of the function, which can be one of the following: • “Transmit User Function” • “Receive User Function” • “Usermain Function” • “Cyclestart Interrupt Function”
“Used as”	Specify the usage of the selected item. The information delivery is specified by “Input”. The setting of information is specified by “Output”.

Table 82: User Function Editor – Description of the different columns of the “Find Usages” window

After clicking on one of the shown user-functions the appropriate user function is selected and will be opened in the “Source Code Editor”.

The ‘Gateway’ list shows the routings which use the selected item as source or target.

5.8.4 Validation Output

The “Validation Output”-section within the “User Function Editor” is shown in Figure 120.

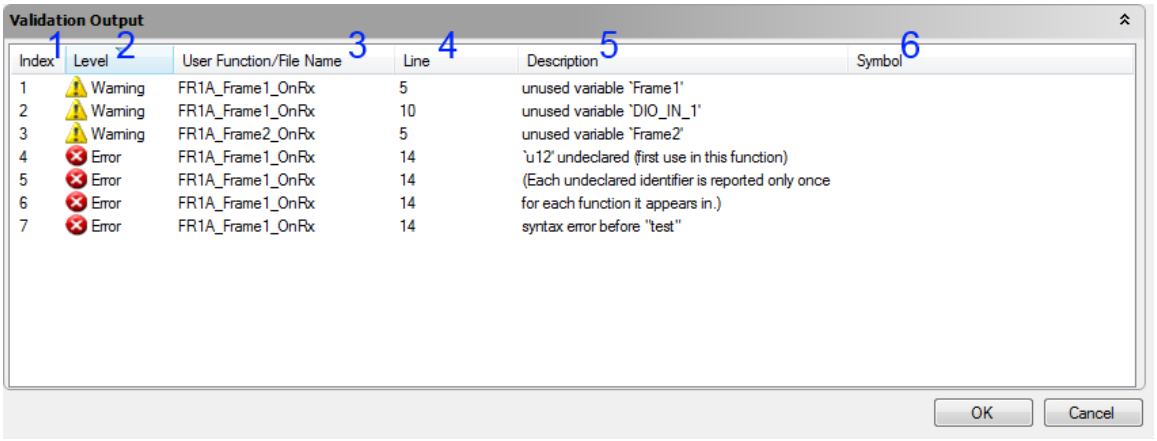


Figure 120: User Function Editor – section “Validation Output”

In this section, all messages of the internal validator are shown. The internal validator checks internal rules and makes use of the appropriate compiler (for FL3X Device-S or FL3X Device-L/L²).

These messages can be of type “Info”, “Warning” or “Error”. The description of Figure 120 is given in Table 83.

No	Description
1	The “**Index**” column specifies the number of the message. It is possible that one message gets more than one index due to the length of the description.
2	The “**Level**” column specifies the type of the information which can be “Info”, “Warning” or “Error”.
3	If the message is resulting from a user function the “**User Function/Filename**” column specifies the name of the appropriate user function. If the message is resulting from an added File (*.c/*.h) the “**User Function/Filename**” column specifies the name of the appropriate filename.
4	The “**Line**” column specifies the line number, which is appropriate to the message.
5	The “**Description**” column gives a detailed description of the message.
6	The “**Symbol**” column eventually gives additional information. E.g. if a function tries to access data of the internal database and the bus is not defined. If this is the case the identifier of this function will be shown here.

Table 83: User Function Editor – Description of the section “Validation Output”

The Output of the “Validation Output”-segment can be sorted. To do this one of the items “Level”, “User Function/File Name”, “Line”, “Description” or “Symbol” can be clicked. The sorting can be done in ascending or descending order. For example, to group the type of messages the “Level” item can be clicked.

If a selected message-line, resulting from a user function, is double clicked, the appropriate user function will be opened in the “User Function Editor”. Additionally, the appropriate line is marked.

If a selected message-line, resulting from an added file, is double clicked, the appropriate file opens in the appropriate editor. Which editor is used can be set by clicking ‘*Extras->Settings->Application->Open File Settings*’ (s. chapter 5.22.1.4).

NOTICE

The warning “Unused Variables” is not shown since FL3X Config Version 1.2 or later.

5.8.5 Merge User Functions

NOTICE

This feature requires the “FL3X Config Enterprise” license.
If this license is not available, the button “Merge User Function(s)” is disabled.

In some case, it may be necessary to import the user functions from another project, which is provided by the merge feature with a simple wizard.

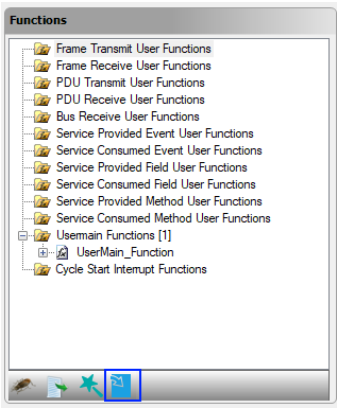


Figure 121: User Function Editor – Merge User Functions button

If you click on the “Merge User Function(s)” button, a dialog will open, which is used to select the configuration file (*.fcr*) of the project, whose functions should be imported. After this step, the bus assignment dialog will open, which is needed for mapping the busses of the imported project to the busses of the current project.

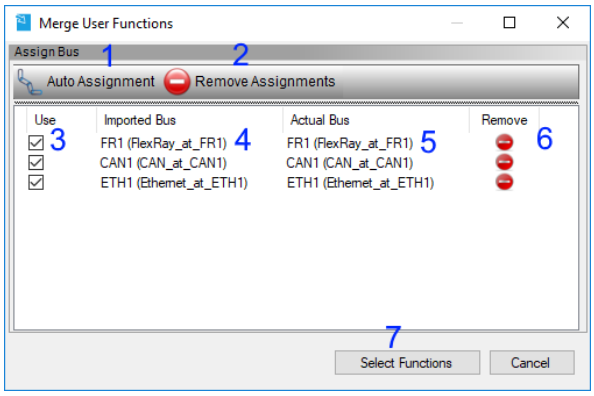


Figure 122: Merge User Functions – Bus Assignment

No	Description
1	When the bus assignment dialog is opened, an automatic assignment is suggested, which can be restored with this button. The bus types and the used databases are considered by automatic assignment.
2	This button removes all actual bus assignments.
3	The functions of the not selected/assigned busses are excluded by the import process.
4	This column shows the bus names of the busses, which are configured by the project with the user functions to be imported.
5	This column shows the bus names of the actual project. The cell values are editable. Clicking on a cell will open a dropdown list with the bus names, which may be used for the assignment.
6	This button removes the assignment on the corresponding row.
7	This button opens the dialog for selecting the user functions.

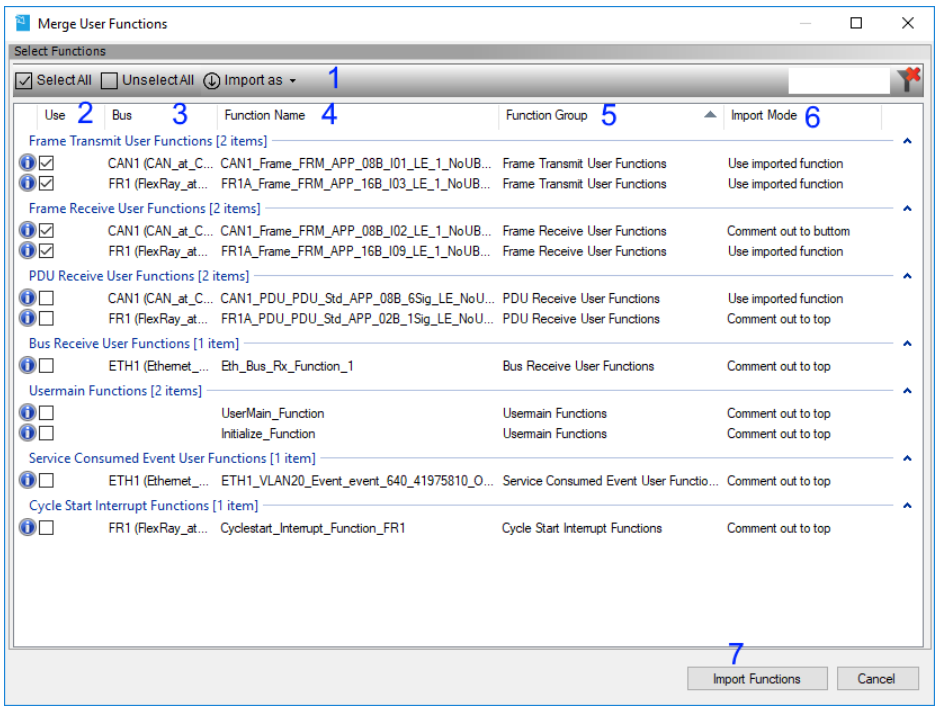


Figure 123: Merge User Functions – Select Functions

No	Description
1	Select all: Selects all functions for the import process Unselect all: The selection will be removed for each row Import as: This menu item changes the import mode of the selected rows
2	Each row can be manually selected. Not selected user functions will not be imported.
3	The bus name of the user function
4	The user function name
5	The user function group name
6	The import mode of each function can be individually set. Following modes are available: **Use imported function:** This is the default option. The function is created if it does not exist. The code will be imported/replaced. The state (enabled) and the properties (e.g. user main function cycle) of the imported function is used. **Comment out to bottom**: This option is only available if the current project has already the same function. The code of the current function will not be deleted. The code of the imported function will be added as comment to the bottom of the current function. **Comment out to top:** This option is only available if the current project has already the same function. The code of the current function will not be deleted. The code of the imported function will be added as comment to the top of the current function.
7	This button finishes the import process.

	Information
	A log file in csv format is shown after the import/merge process is finished.

	Information
	All imported/merged functions are validated after the import process is finished. Invalid user functions are deactivated. The log file also contains this information.

5.8.6 RBS API

The RBS API provides access to several functions via the User Function Editor. An API-Function can be added per drag & drop or via context menu to the user code. The added code will also contain a sample usage of the corresponding function. You can use the tooltips to see the description of an Api function.

API modules are sorted by groups:

- Clock: Contains API functions to get/set the time per clock group (RTC, PTP, ...) or use stopwatches
- Common: Contains some common function, for example reading the API version.
- IO: Contains functions to get/set/read analog and digital inputs/outputs.
- Log: Contains functions for logging of the FL3X Diagnostics Website (see chapter 8).
- Project: Contains functions to get project information running in the device.
- Replay: Contains functions to get/set running replay sequences.
- System: Contains functions to control or change the device's behavior.

	Reference
	For more information about the RBS API refer to: "RBS API Manual" [11] The document can be found in the Windows Start menu after installing FL3X Config.

5.8.7 Differences in Saving User Functions

FL3X Config saves the user functions in the project folder in two ways. Normally the file "UserFunctions.xml" keeps the meta data and the code of the user functions when the project is saved. If the "FL3X Config Enterprise" license is available, then the "UserFunctions.xml" file will contain only the meta data of the user functions and their code will be saved in separate .c files. This enables a better overview of the project files, which can be particularly necessary when several users work on user functions and version management tools are used to check the changes. Please consider the following points if you have a "FL3X Config Enterprise" license.

- The project directory will contain a folder with the name "UserFunctions".
- The folder "UserFunctions" may have many subfolders which are used to group the user functions by type (e.g. TxFrameFunctions, TxPduFunctions, etc.).
- Each user function will have a separate .c file, which contains the name of the function (with input and return parameters) and the code of the function.
- The name of the .c files should **not** be changed.
- The name of the user function stored the .c file should **not** be changed.
- The c. files contain a comment with the name of the function and its last state (active/inactive). This comment should **not** be changed.
- The code in the .c files can basically be changed via an external editor. When editing these files via an external editor, do not use any API function (e.g. get/set Signal/PDU/global var) which is not already used in the corresponding file. Doing this anyway, you will receive errors when you build your project with FL3X Config. If you want to add any API function to the files, add them first via User Function Editor and save your project in FL3X Config.

5.8.8 Supported Data Types

Following data types are supported in the User Function Editor.

Data Type	Base Data Type	Data Type	Base Data Type	Data Type	Base Data Type	Data Type	Base Data Type
s08	signed char	**vs08**	volatile signed char	**u08**	unsigned char	**vu08**	volatile unsigned char
int8		**vint8**		**uint8**		**vuint8**	
INT8		**VINT8**		**UINT8**		**VUINT8**	
int8_t		**vint8_t**		**uint8_t**		**vuint8_t**	
int8_T				**uint8_T**			
A_INT8				**A_UINT8**			
s16	signed short	**vs16**	volatile signed short	**u16**	unsigned short	**vu16**	volatile unsigned short
int16		**vint16**		**uint16**		**vuint16**	
INT16		**VINT16**		**UINT16**		**VUINT16**	
int16_t		**vint16_t**		**uint16_t**		**vuint16_t**	
int16_T				**uint16_T**			
A_INT16				**A_UINT16**			
s32	signed int	**vs32**	volatile signed int	**u32**	unsigned int	**vu32**	volatile unsigned int
int32		**vint32**		**uint32**		**vuint32**	
INT32		**VINT32**		**UINT32**		**VUINT32**	
int32_t		**vint32_t**		**uint32_t**		**vint32_t**	
int32_T				**uint32_T**			
int64_t				**A_UINT32**			
A_INT32							
boolean_T							
s64	signed long long	**vs64**	volatile signed long long	**u64**	unsigned long long	**vu64**	volatile unsigned long long
int64_T		**vint64_t**		**uint64_t**		**vuint64_t**	
A_INT64				**A_UINT64**			
				uint64_T			
A_ASCIISTRING	char	**A_UNICODE 2STRING**	char*	**A_FLOAT32**	float	**A_FLOAT64**	double
char_T		**rtString**		**real32_T**		**real_T**	
int24	int32	**uint24**	uint32	**A_BYTEFIELD**	uint8*	**A_BITFIELD**	uint32

Table 84: User Function Editor – Supported Data Types

5.9 Default Values

5.9.1 The Concept of Default Values

Default values are used to predefine the values of all frames, PDUs and signals, which are transmitted by the device.

The default values of frames, PDUs and signals can be changed with the “Default Value Editor” (see chapter 5.9.2).

The generated code uses an internal database to store the values of the frames, PDUs and signals sent by the RBS or gateway (see chapter 5.8.2.2). After the project has been built and flashed to the hardware, the internal database is initialized by using the default values configuration.

5.9.2 The Default Values Editor

The “Default Values Editor” provides the possibility to define the default values of signals within an RBS or gateway. The “Default Values Editor” can be opened by clicking *‘Edit -> Default Values Editor’* from the menu-bar.

All signals of the appropriate project (RBS or gateway) will be initialized with these values.

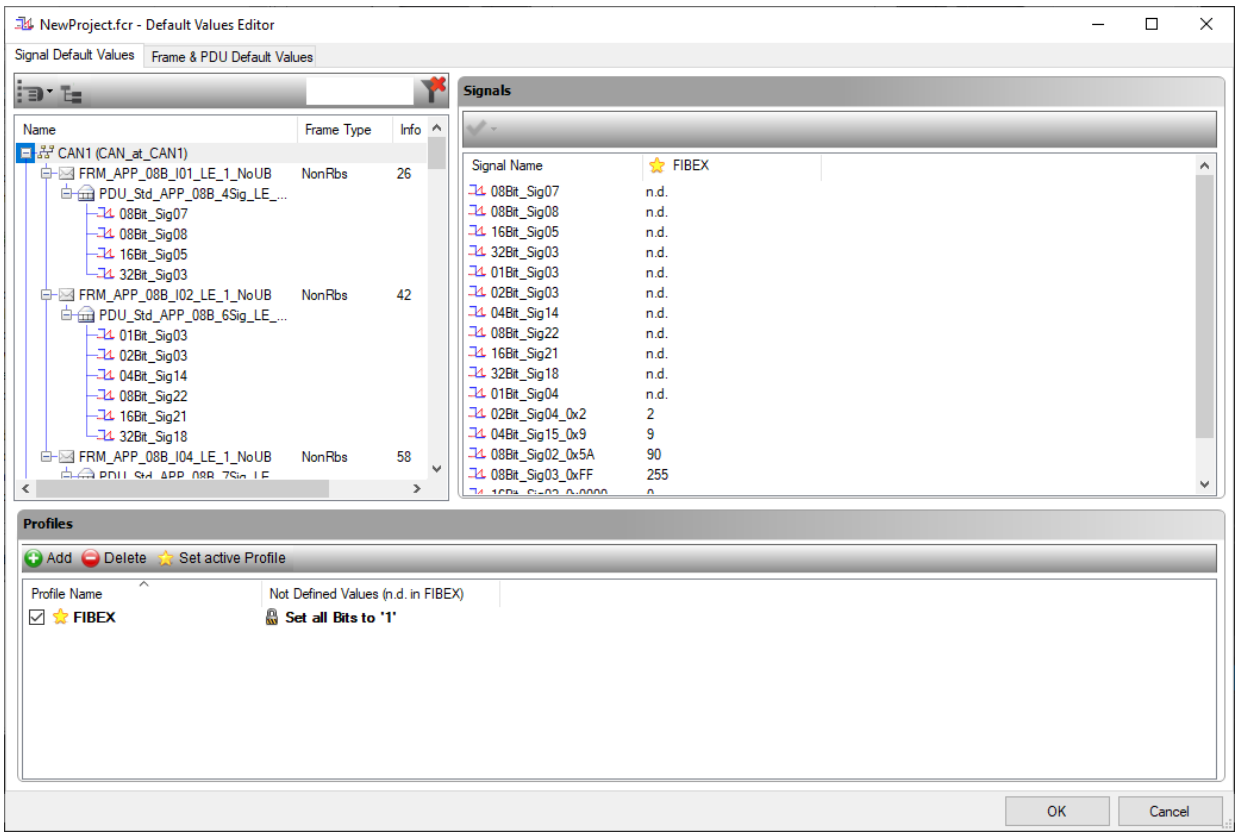


Figure 124: The “Default Values Editor”

As shown in Figure 124 the “Default Values Editor” contains 2 tabs, “Signal Default Values” and “Frame & PDU Default Values”.

5.9.2.1 Signal Default Values Tab

The “Signal Default Values” tab has 3 sections, which are described in more detail in the following chapters.

5.9.2.1.1 Network Section

In the section “Network”, all configured busses are shown. If one bus (e.g. CAN2) is selected, all signals of the appropriate bus are shown in the “Signals”-section.

Additionally, the tree view of the bus enables the user to select child-elements (e.g. frames, PDUs or signals). By the selection of one of these elements only the appropriate signals of the selected item are shown in the “Signals” section.

It is also possible to select more than one element (e.g. more than one frame) by using the “SHIFT”-key or the “CTRL”-key.

As with other dialogs, you can change the visible element levels in the network tree or do a text search.

If the PDU-View is activated, frames are not shown.

5.9.2.1.2 Profiles Section

The “Profiles” section contains all defined or available profiles. Every profile is a kind of list which contains all signals of the FIBEX files of the defined busses and their appropriate default values.

If the “Default Values Editor” is opened the first time within a project, only the FIBEX profile will be shown. This profile is set as active. If a profile is active the values of this profile will be taken for the initialization of the signals. A yellow star in front of the profile name shows that this profile is active.

Table 85 contains a description of each column in the “Profiles” section.

Column-Name	Description
Profile Name	Contains all Profile Names of the configured Profiles. The additional checkbox in each line manages the visibility of each profile in the “Signals” section. If the checkbox is activated the profile is shown. If the checkbox is deactivated the profile is not shown and thus not editable. The yellow star in front of one of the profiles shows that this profile is the active profile.
Not Defined Values (n.d. in FIBEX)	The default values of FIBEX, which are not defined, can be globally set for the appropriate profile in this column. There are two different possibilities. Either “Set all Bits to ‘1’” or “Set all Bits to ‘0’”. This can be done by clicking on the relevant field and choosing the relevant value. The “‘n.d.” value of the “FIBEX”-profile is not editable and is always set to “Set all Bits to ‘1’”.

Table 85: Default Values Editor – Columns of the section “Profiles”

Set Active Profile

To set a profile to the active profile, the profile must be selected using the left mouse button first. After the profile is selected, the “*Set active Profile*” button must be clicked.

Create New Profile

A new profile can be created, by clicking the “*New*” button in the “Profiles”-section. After this has been done, a new dialog opens (see Figure 125).

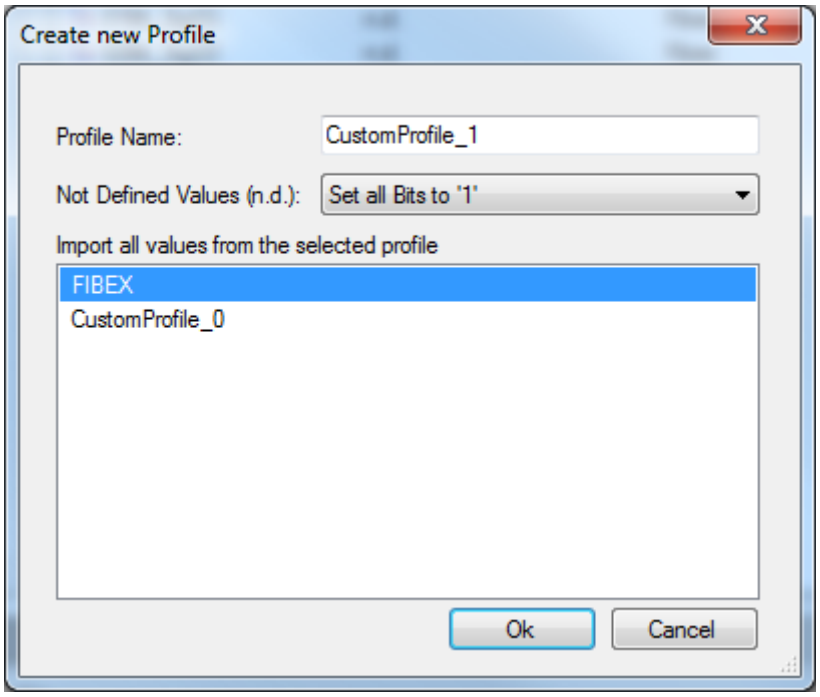


Figure 125: Default Values Editor – “Create new Profile” dialog

Within this dialog the name of the new profile can be edited in the “Profile Name” field. The default name is “CustomProfile_x”, where x is the number of the profile.

The initialization of default values, which are not defined in FIBEX, can be selected within the “Not Defined Values (n.d.)” field. This is either “Set all Bits to ‘1’” or “Set all Bits to ‘0’”.

To import the values from a previously defined profile, the relevant profile can be chosen in the “Import all values from the selected profile” field. If no profile is selected, the “*OK*” button cannot be activated. Initially, the “FIBEX” profile is selected.

If the “FIBEX” profile is selected all entries for the new custom profile are set to “FIBEX”.

Delete Profile

To delete a profile, the profile must be selected in the “Profiles” section first. After that, the *‘Delete’* button must be pressed. Only user defined profiles can be deleted.

If the deleted profile was active, the “FIBEX” profile will be set to the active profile.

5.9.2.1.3 Signals Section

In the “Signals” section, all signals are shown, which have been selected in the “Network” section. If the “Default Values Editor” is started the first time, the “Signals Section” only contains the columns “Signal Name” and “FIBEX”.

Additional columns will appear if new profiles are defined in the “Profiles” section.

The following Table 86 shows a description plus the tooltip-information of each of these columns. A tooltip appears if the user places the mouse-pointer on the relevant elements of the columns.

Column-Name	Description	Tooltip
Signal Name	Contains all FIBEX-short-names of the signals, selected in the “Network”-section.	Information about the appropriate signalBit-position, Bit-Length, Compu-Method, ...)
FIBEX	Contains all default values of the signals from the appropriate FIBEX-File. If no default value is existent “n.d.” (not defined) will be shown. The values in this column are not editable!	Information about the min and max value of the appropriate signal.
Custom Profile (user defined name)	Contains all default values generated by the user. All these values are editable, by clicking on the appropriate field. The value “FIBEX” means that the value of this field is inherited from the “FIBEX”-column. Values which are “n.d.” in the “FIBEX”-column will be initialized with the appropriate “n.d.” value of the profile.	Information about the min and max value of the appropriate signal.

Table 86: Default Values Editor – Columns of the section “Signals”

As can be seen in Figure 124, a yellow star is shown in the header of one of the columns. A yellow star marks the column, which is set to the active profile.

An active profile means that the default-values for the signals of the appropriate RBS or gateway are taken from this profile.

The following popup ist shown if the cell under the active profile column is clicked.

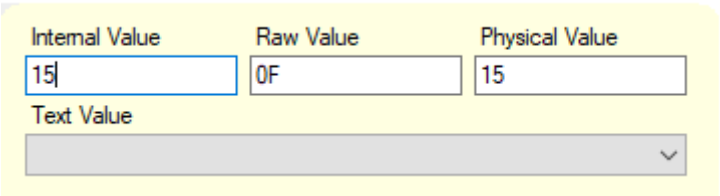


Figure 126: Default Values Editor – Change default value

You can change following values of the signal. Other values are calculated automatically after a value is set.

- Raw value: The hex value sent on the bus.
- Internal value: An ECU converts the received raw value to the internal value before converting it to physical value. E.g. if an 8-bit Signal is coded as “unsigned” and the raw value 0xFF is received, then the internal value will be 255. If this signal is code as “signed” then its internal value will be -1.
- Physical value: The internal value is converted to physical value using the compu-methods.
- Text value can also be selected if the signal has a text table in compu-methods.

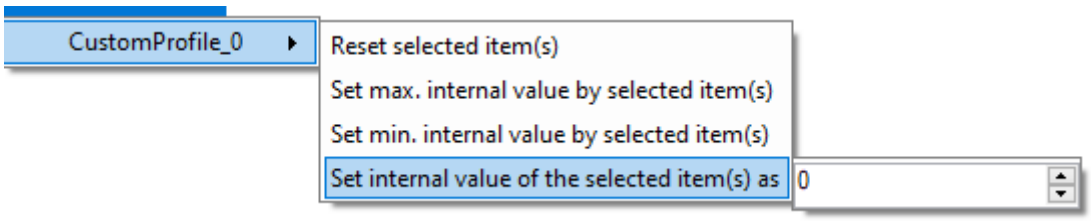


Figure 127: Default Values Editor – Context menu in signals section

You can select multiple signals and set/reset their internal values at once.

	NOTICE
	If time synchronization frames (see chapter 5.10) are configured, default values have no effect on them, because they are always overridden with the data of the time synchronization.

5.9.2.2 Frame & PDU Default Values Tab

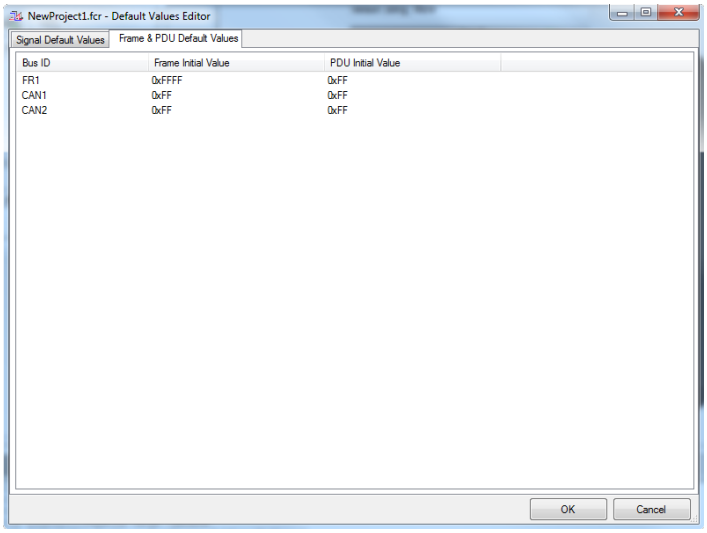


Figure 128: Default Values Editor – Tab “Frame & PDU Default Values”

The default values of the frames and PDUs can be changed for each configured bus. The frame and PDU payload are configured as hexadecimal number. Table 87 contains a description of each column in the “Frame & PDU Default Values” – tab.

Column-Name	Description
Bus ID	This is the bus ID. The corresponding connector can be seen in the “Device Configuration” section.
Frame Initial Value	The initial raw payload of the frames can be edited in this column.
PDU Initial Value	The initial raw payload of the PDUs can be edited in this column.

Table 87: Default Values Editor – Columns of the tab “Frame & PDU Default Values”

	Information
	The default values for the project or the project templates can also be changed in the “Initial Value Settings”. Note that this setting applies to all busses. (see chapter 5.22.3.4)

5.10 Secure Onboard Communication

Secure Onboard Dialog is used to assign the PDU keys to the SecOC PDUs, which are needed for the authentication mechanism.

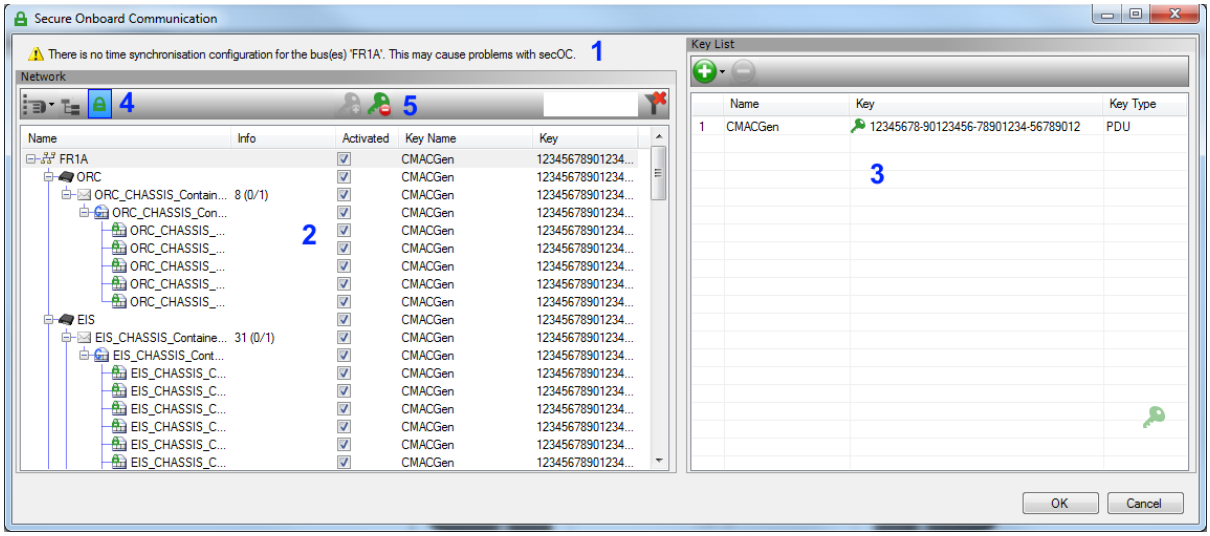


Figure 129: “Secure Onboard Communication” dialog – Example for FlexRay

No	Description
1	For some OEMs, a valid configuration for Secure Onboard Communication also needs a configuration for “Time Synchronization”. If there is no “Time Synchronization” configuration in the project, this warning is shown.
2	This list contains the configuration for all “Secure Onboard” related elements (PDUs).
3	This list contains all available keys in the project. Use the “*Add*” or “*Remove*” button to add/remove a key to the list.
4	The SecOC filter button can be used to show just the SecOC PDUs.
5	Buttons for assigning PDU keys or removing the assignments.

Table 88: “Secure Onboard Communication” dialog

There are several ways to assign the PDU keys to the SecOC PDUS.

- Drag & Drop: To assign a specific key to an element, select the key in **3** and move it per drag & drop on the network element (**2**). E.g. if you assign a key to an ECU, it will be automatically assigned for all elements of this ECU. You can also drag a network item and drop it on a PDU key.
- Context Menu: After selecting the network item (**2**) and the PDU key (**3**) you can open the context menu of the network item and select ‘Map key to network item’.
- Buttons: After selecting the network item (**2**) and the PDU key (**3**) you can use the assignment button (**5**)

If you want to remove an assignment, you can use the ‘delete key’, “context menu” or ‘remove assignment button (**5**) ‘E.g. if you remove the assignment by an ECU, it will be automatically removed for all elements of this ECU.

Information
For better runtime performance, use a template which supports AES/SecOC (see chapter 2.8 to find FPGA images with AES/SecOC support).

NOTICE
A valid configuration for Secure Onboard Communication also requires an OEM specific handling of the SecOC messages (freshness value: timestamp/counter). OEM-specific configuration options for synchronization of SecOC messages (freshness value: timestamp/counter) are described in corresponding “Additional Information” documents for certain OEMs.

NOTICE
SecOC verification is one of the first steps when a frame is received and SecOC authentication is the last action when a frame is transmitted. **RBS:** RBS<DefaultValue> ->TX User Function -> Signal Manipulation -> Standard Signal Handling -> **SecOC** -> RBS_TX_on_Bus. **Gateway:** GW_RX_on_Bus -> RX Frame User Function-> **SecOC** -> Rx PDU User Function (if SecOC valid) -> Gateway routing + scaling -> TX-User Function -> Signal Manipulation (via FL3X Config Control) -> Standard Signal Handling -> **SecOC** -> GW_TX_on_Bus.

NOTICE
The SecOC calculation for ‘RBS RX’ and ‘Non RBS’ frames is only executed if they are actively received and processed by a gateway routing or a Rx user function.

If SecOC is not activated for a PDU (no key is assigned to the PDU), then the PDU trailer will have a default value, which can be defined in the settings dialog (Project -> Initial Values Settings). The option “Route the trailer of the SecOC PDUs” in the settings dialog (Project -> Common Settings) affects gateways. The following figure explains which SecOC Trailer of a PDU is used.

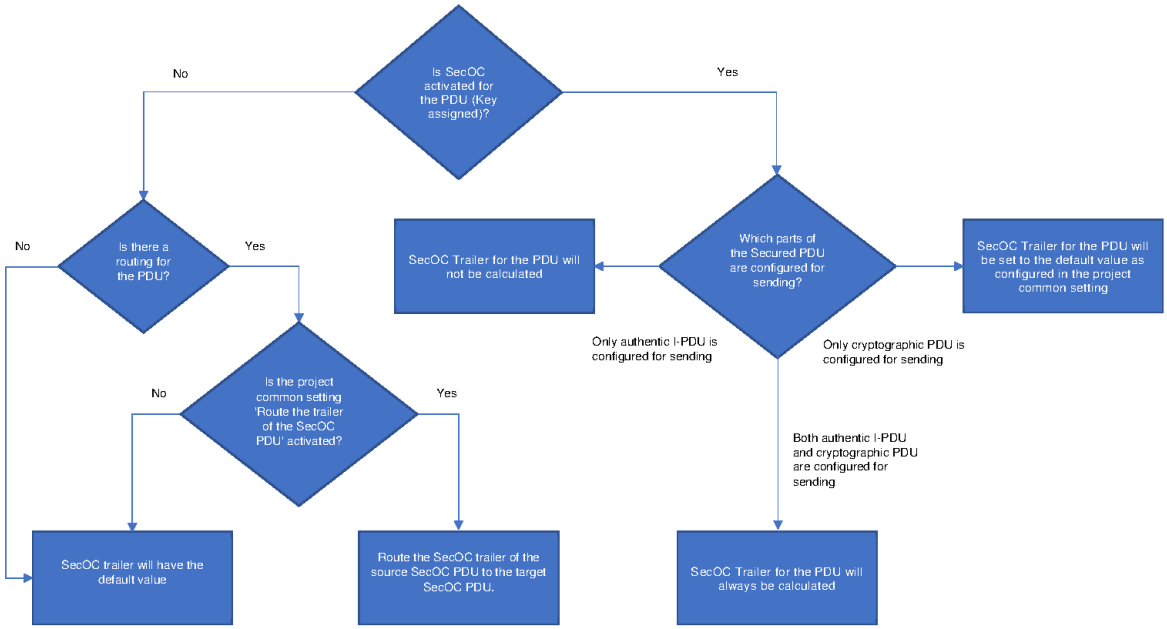


Figure 130: Flowchart for SecOC

5.11 Time Synchronization

This dialog (see Figure 131) configures time synchronization according to IEEE PTP and AUTOSAR .

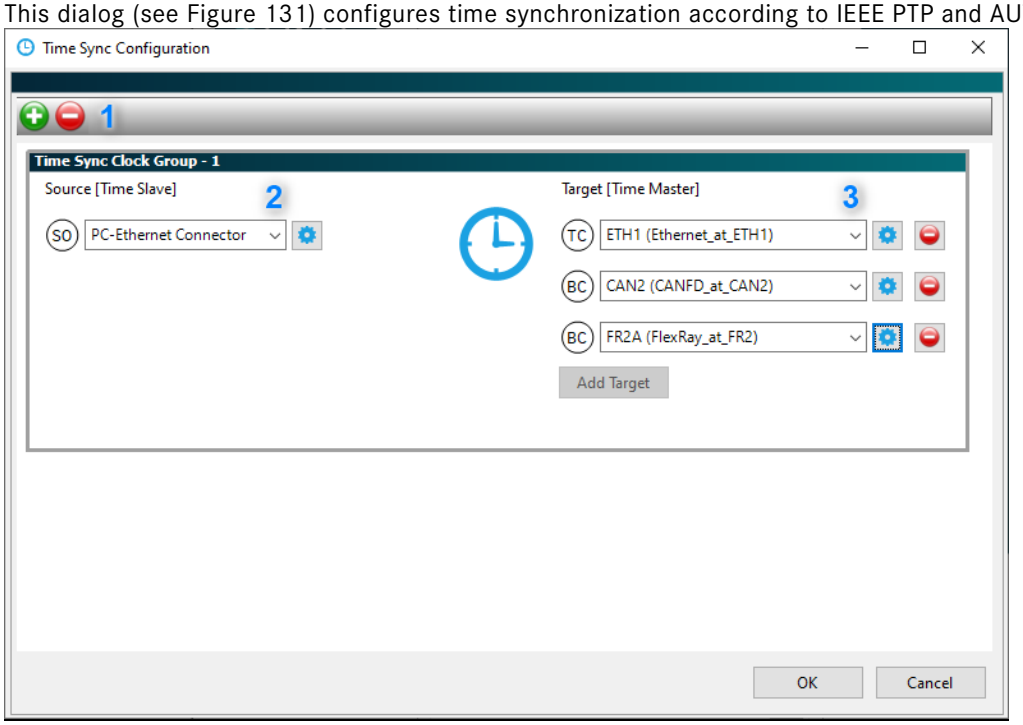


Figure 131: Time Sync Configuration – Example for CAN, Ethernet, FlexRay

No	Description
1	Use the “*Add*” and “*Remove*” button to add/remove a “Time Sync” configuration. A “Time Sync” configuration consists of a source (Time Slave, see 2) and target (Time Master, see 3). It is possible to configure several independent “Time Sync” configurations per project.
2	The source of a single “Time Sync” configuration. You can either select the device internal clock, PC-Ethernet Connector, or a configured bus (which supports time synchronization) as source.
3	The target of a single “Time Sync” configuration. You can select the PC-Ethernet Connector or any configured bus (which supports time synchronization and is not used for time synchronization yet) as target. Use the “*Edit*” button to configure the time synchronization parameters for each selected bus.

Table 89: Time Sync Configuration

The behavior of the configured Time Sync channel is shown in front of the channel selection drop-down. The following behaviors are possible:

Icon	Name	Description	Conditions
	Slave Only	The synchronized time is received from the network/bus and stored	Only for Source / Slave configurations of real-busses (not Device Internal Clock)
	Monotonic Clock	The clock time is generated by a monotonic counter starting by zero (0) at device start	Only for Source / Slave configuration of Device Internal Clock
	Grandmaster Clock	The generated clock time is distributed to the network/bus	Only for Target /Master configurations of real-busses when Source / Slave side is configured as Monotonic Clock
	Boundary Clock	The synchronized time is distributed to the network/bus independently of the packet timing as seen on the slave side. If the slave side was synchronized once the time will still be distributed independently	Only for Target / Master configurations of real-busses when Source / Slave side is configured as Slave Only Clock
	Transparent Clock	The synchronized time is distributed to the network/bus by using the same packet timing as seen on the slave side (time distribution is tunneled). The internal processing offset is added to the outgoing packets according to bus specific protocol. If the slave side stops synchronizing the time distribution will also stop	Only for Target / Master configurations of Ethernet channels when Source / Slave side is also configured as an Ethernet channel. Transparent Clock behavior must be explicit configured inside the Target / Master configuration dialog. If conditions are no longer met after making changes inside the Time Sync dialog, this behavior may change automatically to Boundary Clock

Table 90: Time Sync Configuration – Clock behavior



NOTICE

The transmission of “Time Synchronization Frames” is regulated by the TimeSync configuration. If the TimeSync configuration prevents the transmission of a “Time Synchronization Frame” at runtime, the following steps are **not** executed for this “Time Synchronization Frame”:
- User Functions
- Manipulation
- Standard Signal Handlings

	NOTICE
	“Synchronized Time Bases” is supported according to AUTOSAR 4.3. TimeSync according to older AUTOSAR versions is not guaranteed. “Offset Time Bases” is not supported explicitly.

5.11.1 Common Configuration

Parameter	Description	Source [Time Slave]	Target [Time Master]
Sync-Loss-Timeout [ms]	After timeout clock group is considered as asynchronous.	Yes	No
Sync-Loss-Threshold [ms]	If the timestamp exceeds the threshold, clock group is considered as asynchronous.	Yes	No
Sync-Period [ms]	Cyclic period of sync frame transmissions	No	Yes
Global time master	Enables global time master mode	No	Yes
Domain ID	ID of clock group domain	Yes	Yes
CRC validated	CRC validation is enabled for incoming time sync related frames	Yes	No
CRC secured	Frames will be secured with CRC	No	Yes
Sync Data-IDs	CRC data IDs for sync frame	Yes	Yes
Follow-Up Data-IDs	CRC data IDs for follow-up frame	Yes	Yes
User Bytes	Enables transmission of user bytes	No	Yes
User-Byte-Values	Values of user bytes	No	Yes

Table 91: Time Sync Configuration – Common Parameters

5.11.2 CAN Configuration

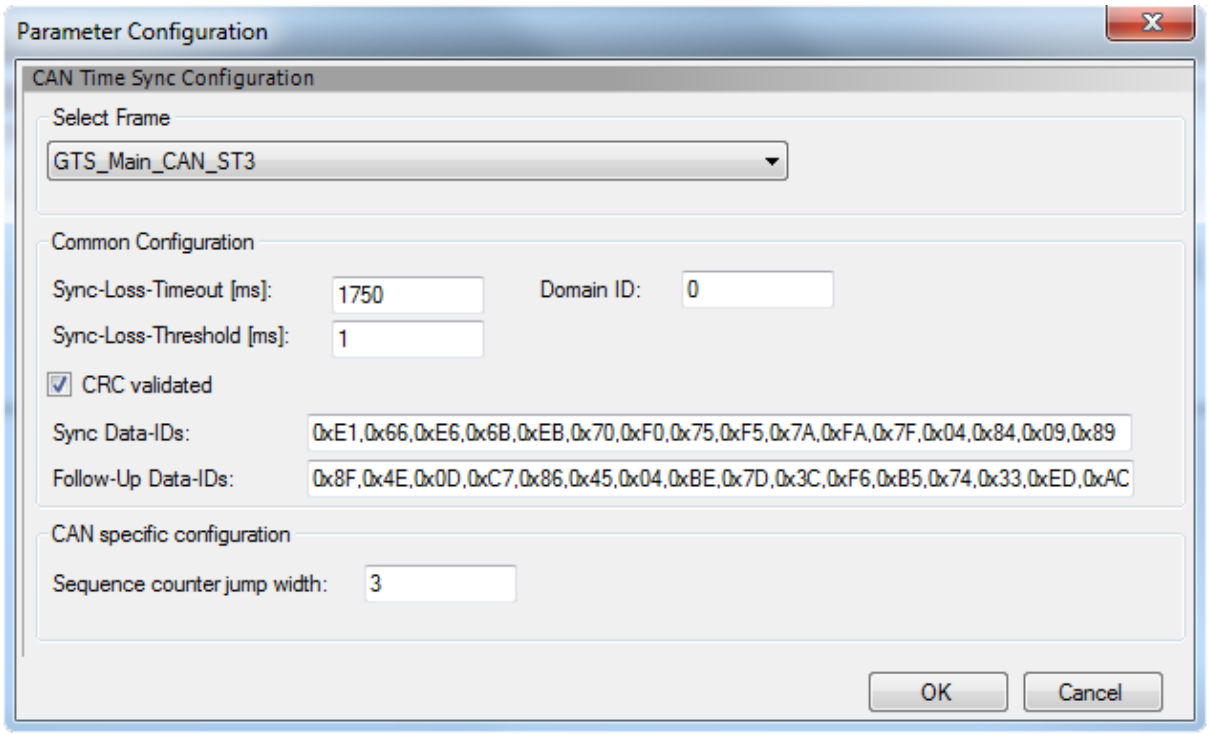


Figure 132: Time Sync Configuration – CAN Master Configuration

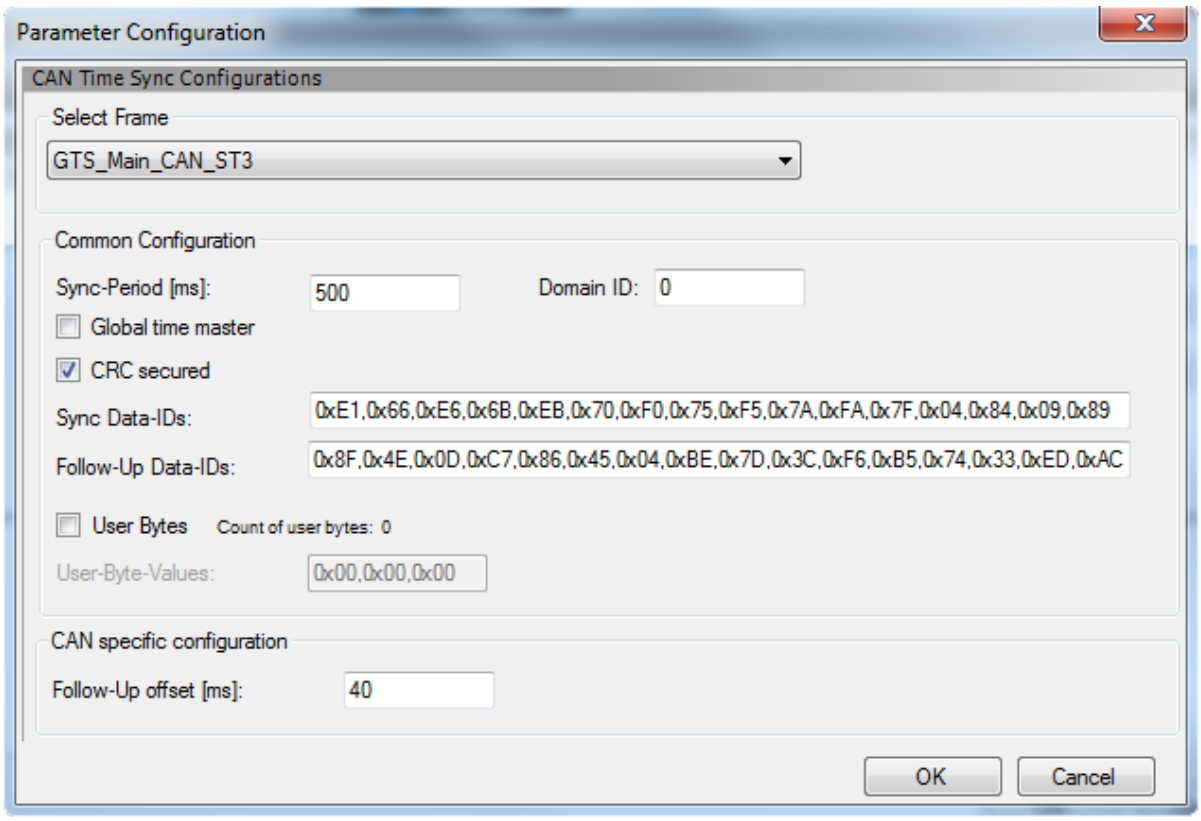


Figure 133: Time Sync Configuration – CAN Slave Configuration

Parameter	Description	Source [Time Slave]	Target [Time Master]
Select Frame	Time sync related frame from database	Yes	Yes
Sequence counter jump width	Number of accepted sequence counter jumps	Yes	No
Follow-Up offset [ms]	Interval between follow-up frame and sync frame	No	Yes

Table 92: Time Sync Configuration – CAN specific Configuration

5.11.3 Ethernet Configuration

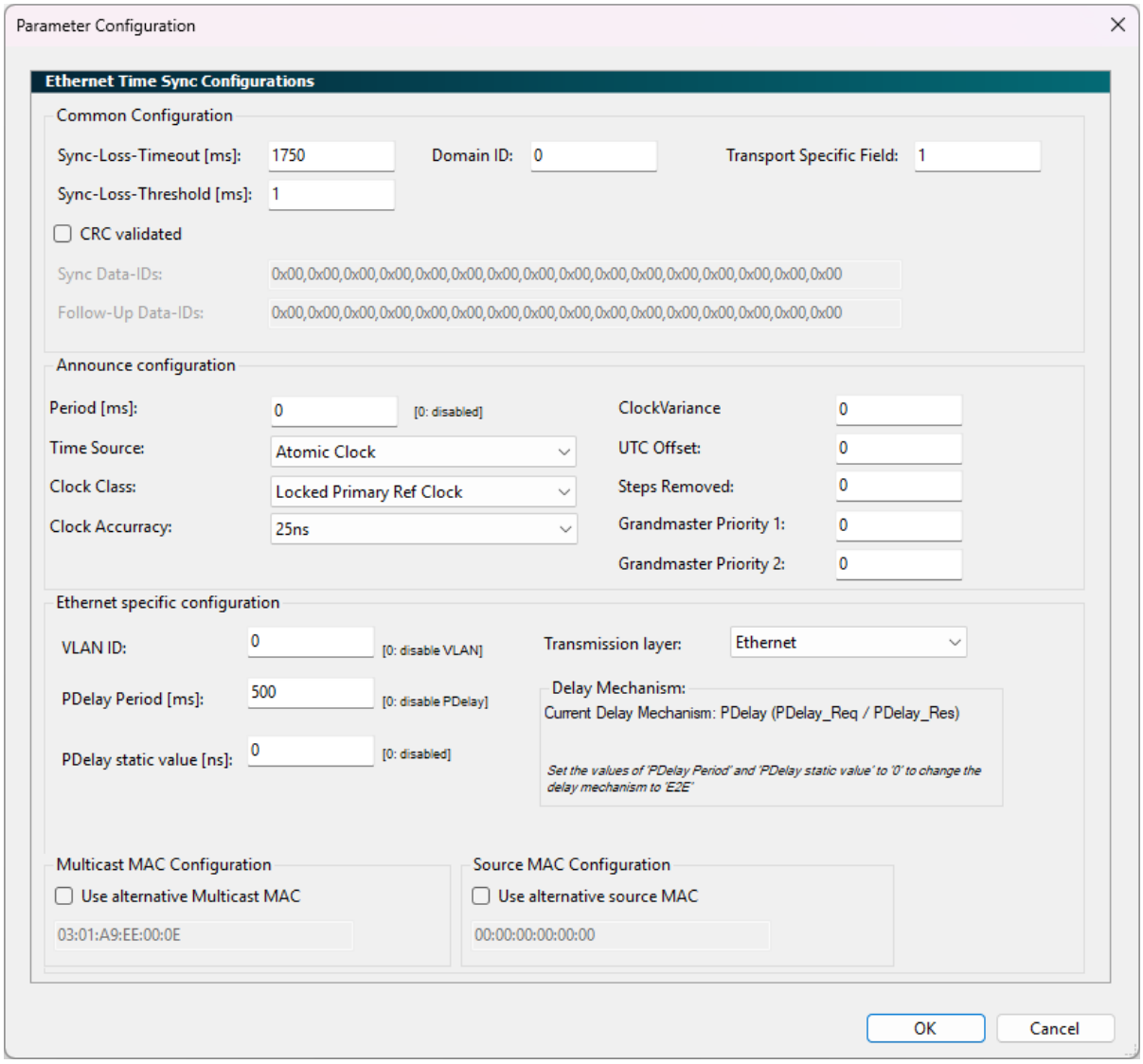


Figure 134: Time Sync Configuration – Ethernet Slave Configuration

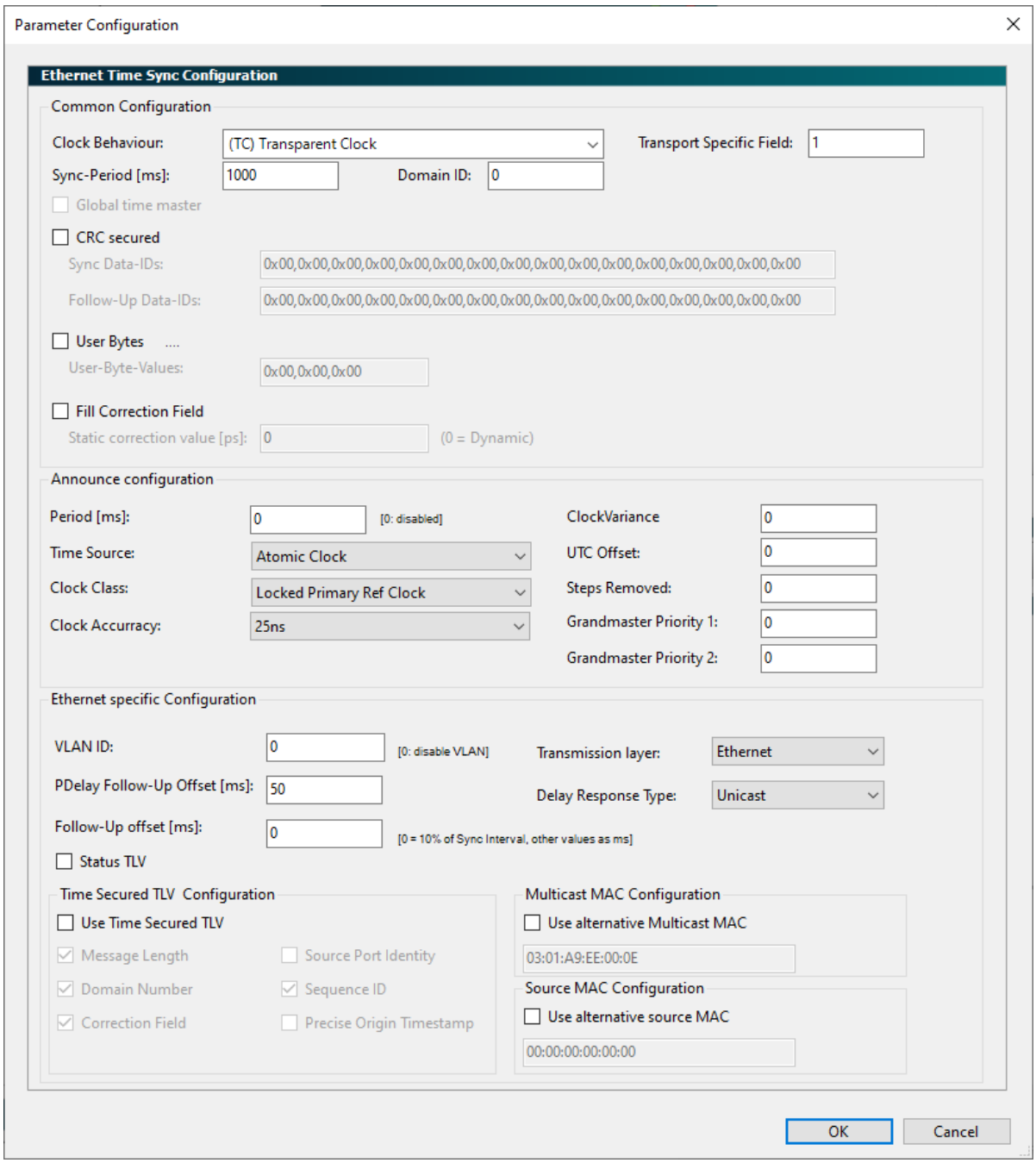


Figure 135: Time Sync Configuration – Ethernet Master Configuration

Parameter	Description	Source [Time Slave]	Target [Time Master]
Clock Behaviour	Selection of used clock behaviour	No	Yes
Fill Correction Field	If checked the correction field will be filled to distribute the Transparent Clock offset. Only applicable when clock behaviour is configured as Transparent Clock	No	Yes
Static correction value [ps]	If set != 0 the correction field will be set to the given explicit value in picoseconds. If set to 0 the offset is calculated automatically	No	Yes
VLAN ID	ID of VLAN tag. 0 disables VLAN tagging.	Yes	Yes
Pdelay Period [ms]	Period of Pdelay measurement	Yes	No
Pdelay static value [ns]	Fixed preconfigured Pdelay value	Yes	No
Pdelay Follow-Up Offset [ms]	Interval between Pdelay follow-up and sync frame	No	Yes

Parameter	Description	Source [Time Slave]	Target [Time Master]
Transmission layer	Selection between Ethernet and Ipv4	Yes	Yes
Status TLV	Enables transmission of Status TLV	No	Yes
Use Time Secured TLV	Enables transmission of Time Secured TLV	No	Yes
Message Length	Message Length will be included in the CRC calculation	No	Yes
Domain Number	Domain Number will be included in the CRC calculation	No	Yes
Correction Field	Correction Field will be included in the CRC calculation	No	Yes
Source Port Identity	Source Port Identity will be included in the CRC calculation	No	Yes
Sequence ID	Sequence ID will be included in the CRC calculation	No	Yes
Precise Origin Timestamp	Precise Origin Timestamp will be included in the CRC calculation	No	Yes
Delay Response Type	A PTP slave can send 'Delay Request' messages (as an alternative to PDelay Request messages). This setting defines the way of responding 'Delay Request' messages.	No	Yes
Use alternative Multicast MAC	Allows the user to change the MAC address	Yes	Yes

Table 93: Time Sync Configuration – Ethernet specific Configuration

	Information
	The fields under the announce configuration are special parameters that are used for PTP-Announce messages. See IEEE 802.1AS for further details.

5.11.4 FlexRay Configuration

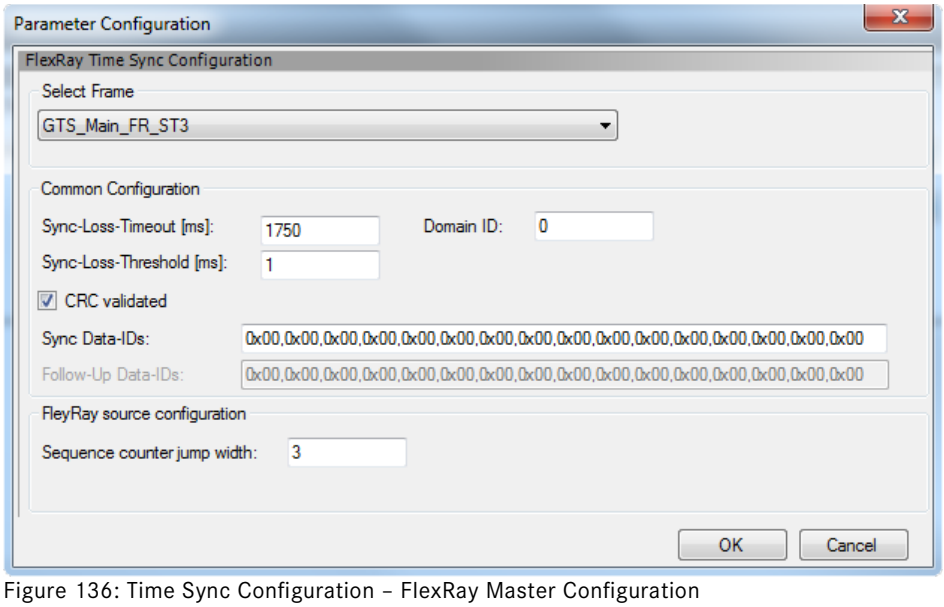


Figure 136: Time Sync Configuration – FlexRay Master Configuration

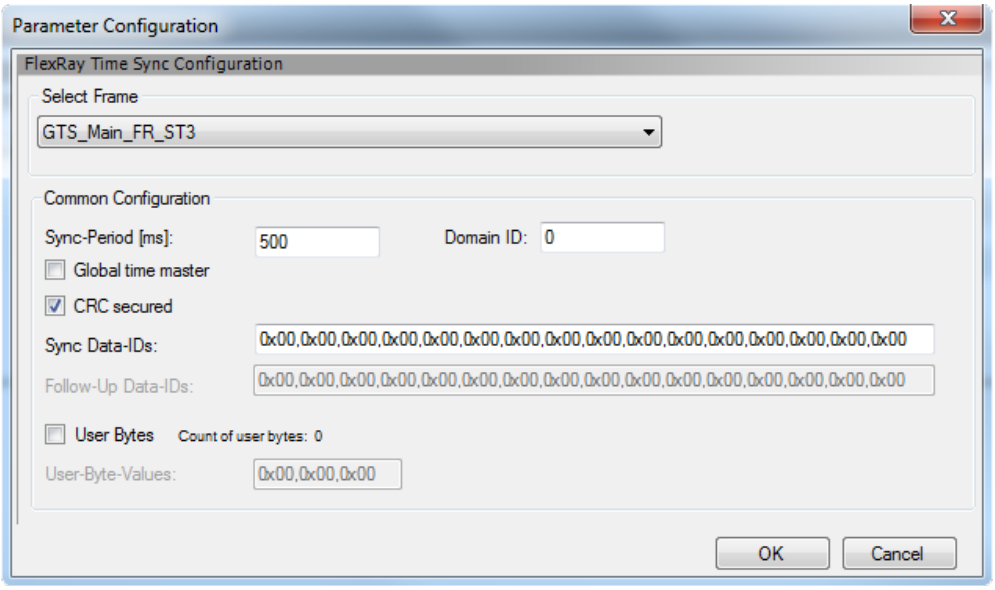


Figure 137: Time Sync Configuration – FlexRay Slave Configuration

Parameter	Description	Source [Time Slave]	Target [Time Master]
Select Frame	Time sync related frame from database	Yes	Yes
Sequence counter jump width	Interval between follow-up frame and sync frame	Yes	No

Table 94: Time Sync Configuration – FlexRay specific Configuration

5.12 Custom E2E-Profiles

The Custom E2E-Profiles dialog provides the ability to create user specific E2E profiles which can be used to protect PDUs, signal groups or SOME/IP elements.

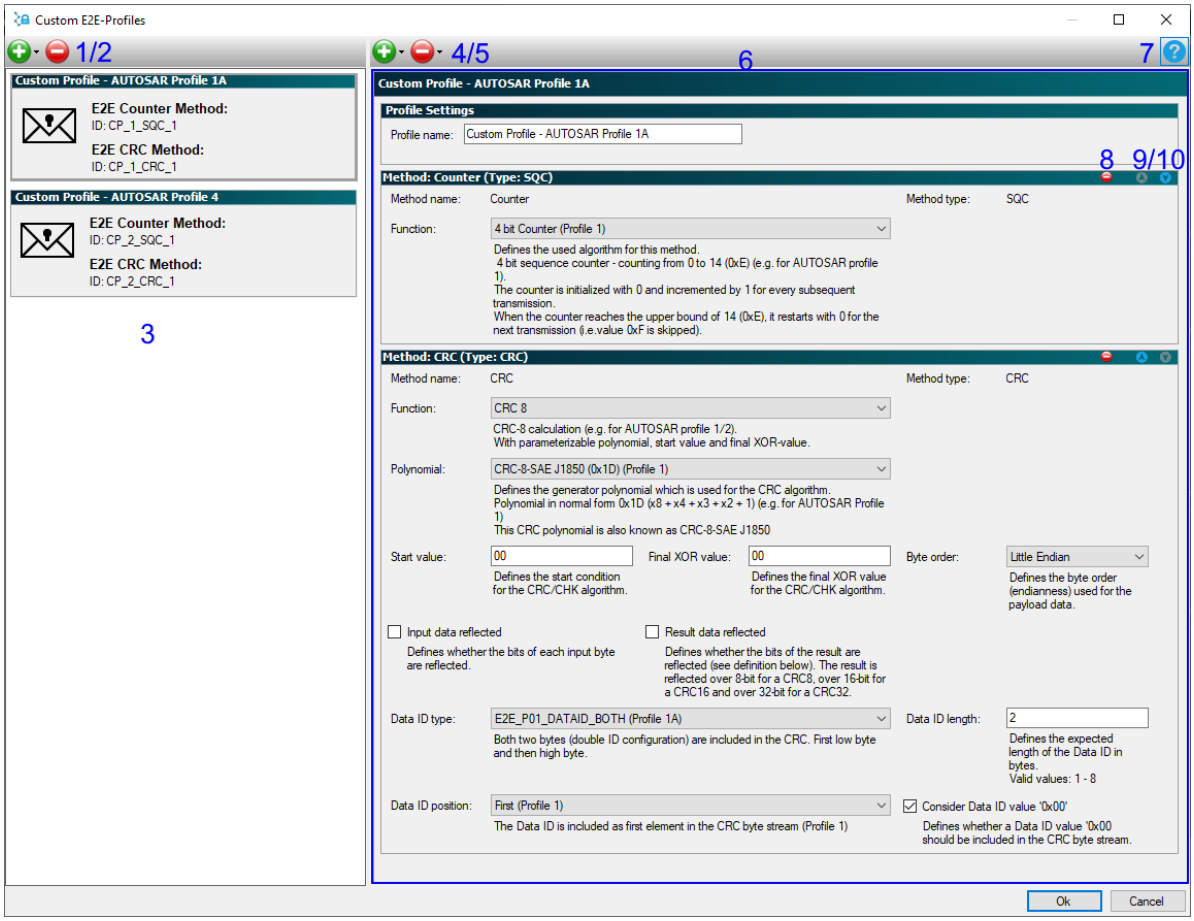


Figure 138: “Custom E2E-Profiles” dialog

No	Description
1	Shows a list of supported AUTOSAR E2E profiles that can be used as template for a custom E2E profile to add (see also chapter 5.12.1).
2	Deletes the selected (3) custom E2E profile.
3	The list of configured custom E2E profiles.
4	Adds a new method for the sequence counter, CRC calculation or Checksum calculation, if the selected template supports it (see also chapter 5.12.2.1).
5	Deletes the selected method from the custom E2E profile.
6	Shows the methods and parameters of the selected custom E2E profile. (See also chapter 5.12.2 and 5.12.2.1).
7	Shows or hides additional information for all parameters.
8	Deletes the selected method from the custom E2E profile.
9/10	Move the method up/down. The order of the methods specifies the order in which the calculations are processed. Normally the counter is calculated first and then the CRC (including the counter value).

Table 95: Description of the Custom E2E-Profiles dialog

	Information
	Th configured custom E2E profiles are shown in all Busses under the E2E tab for mapping them to the corresponding database items. For more details about protecting elements (PDUs, signal groups, events, …) in an RBS/Gateway see chapter 5.6.4.1.

The custom E2E profiles consist of a chain of methods. The order of the methods specifies the order in which the calculations are processed. Each method specifies a function and additional parameter to calculate a value based on the signals/payload of the protected element (e.g. PDU, signal group, event, …). The result of the calculation is used e.g. as counter or CRC value.

There are two different kinds of custom E2E profiles:

- Signal based profiles (see chapter 5.12.2)
  The result of the calculation is directly written to configured signal of the protected element.
- Stack based profiles (see chapter 5.12.3)
  The result of the calculation is used to fill a configured header e.g. via the SOME/IP stack.

5.12.1 Templates for Custom E2E-Profiles

The following profiles can be used as template:

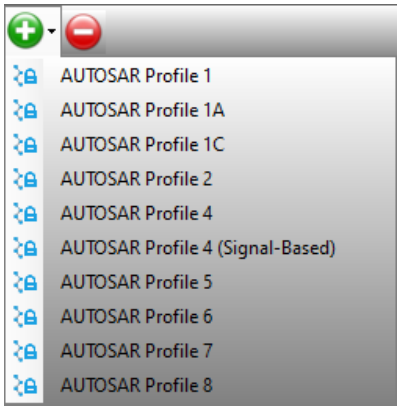


Figure 139: Custom E2E – Available templates for Custom E2E-Profiles

The templates

- AUTOSAR Profile 4
- AUTOSAR Profile 6
- AUTOSAR Profile 7
- AUTOSAR Profile 8

are designed as “stack based profiles” e.g. to protect SOME/IP elements (Events, Fields, Methods).

The template

- AUTOSAR Profile 4 (Signal-Based)

is designed to be used for database representations where the AUTOSAR Profile 4 header fields are mapped to individual signals within a PDU or signal group.

5.12.2 Configure a Signal Based Custom E2E-Profile

The following methods/parameters are available for a signal based custom E2E-Profiles:

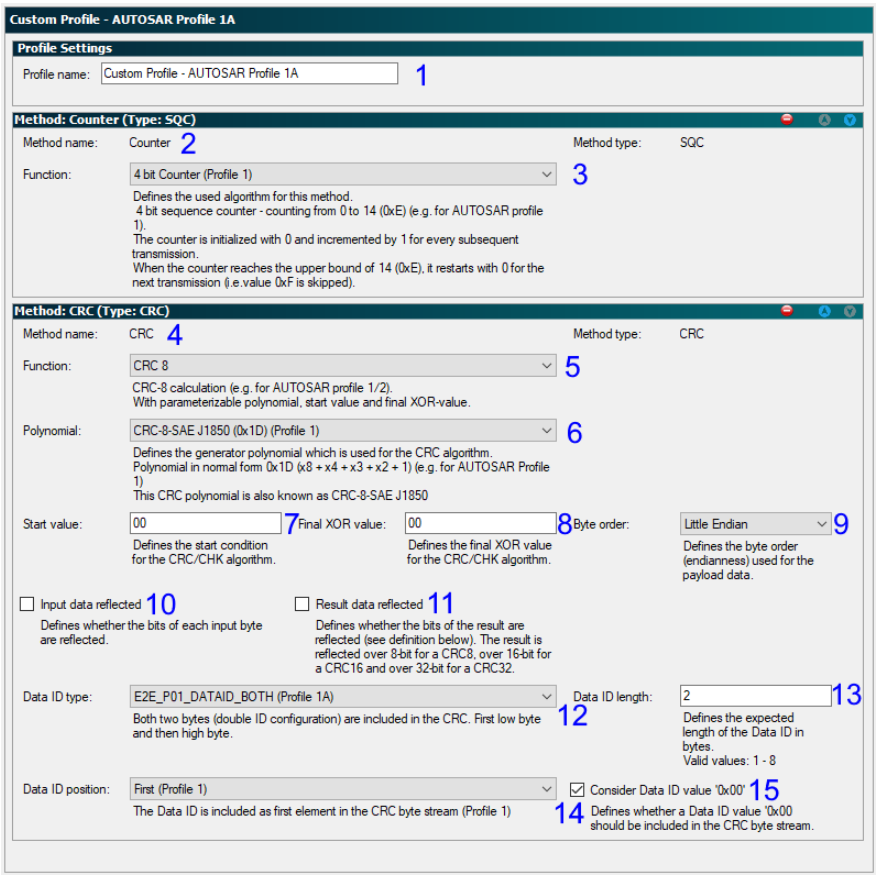


Figure 140: Custom E2E –parameters of a signal based custom E2E-Profile

No	Description
1	Name: The unique name of the custom profile.
2	Method: Counter (Type: SQC) The parameters of the counter method/calculation.
3	Function: Specified the function which is called to calculate the next counter value. See chapter 5.12.4.2.
4	Method: CRC (Type: CRC) The parameters of the CRC method/calculation.
5	Function: Specifies the function which is called to calculate the CRC value. See chapter 5.12.4.4.
6	Polynomial: Defines the polynomial to use for the CRC algorithm. See chapter 5.12.4.6.
7	Start value: Defines the start (initial) value of the CRC/CHK algorithm.
8	Final XOR value: Defines the final XOR value of the CRC/CHK algorithm.
9	Byte order: Defines the byte order (endianness) used for the payload data.
10	Input data reflected: Defines whether the bits of each input byte are reflected.
11	Result data reflected: Defines whether the bits of the result are reflected. The result is reflected over • 8 bit for a CRC8 • 16 bit for a CRC 16 • 32 bit for a CRC 32 • 64 bit for a CRC 64
12	Data ID type: Defines the type of the Data ID (handling of the Data ID value). See chapter 5.12.4.7.
13	Data ID length: Defines the expected length of the Data Id value in bytes. Valid values are 1 - 8
14	Data ID position: Specifies the position of the Data Id in the CRC input stream. See chapter 5.12.4.8.
15	Consider Data ID value ‘0x00’: Defines whether a Data ID value ‘0x00’ should be included in the CRC input stream.

	Information
	A signal based custom E2E profile can be completely modified. It is also possible, for example to use only counter methods only a single checksum calculation. Any warning that is shown intend to aid. Warnings do not prevent the user from using an “invalid” profile configuration.

5.12.2.1 Additional Methods for AUTOSAR Profile 4 (Signal-Based)

The following methods/parameters are available to calculate the additional header fields “Length” and “DATA-ID” of the “AUTOSAR Profile 4 (Signal-Based)”:

Method: Length

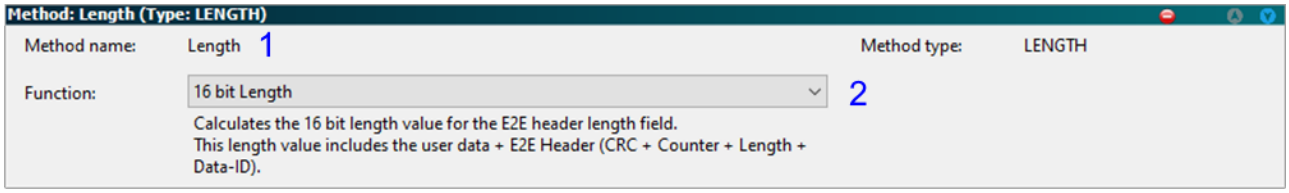


Figure 141: Custom E2E – method “Length” for custom “AUTOSAR Profile 4 (Signal-Based)”

No	Description
1	Method: Length (Type: LENGTH) The parameters of the length method/calculation.
2	Function: Specified the function which is called to calculate the length of the protected payload. See chapter 5.12.4.1.

Method: DATA-ID

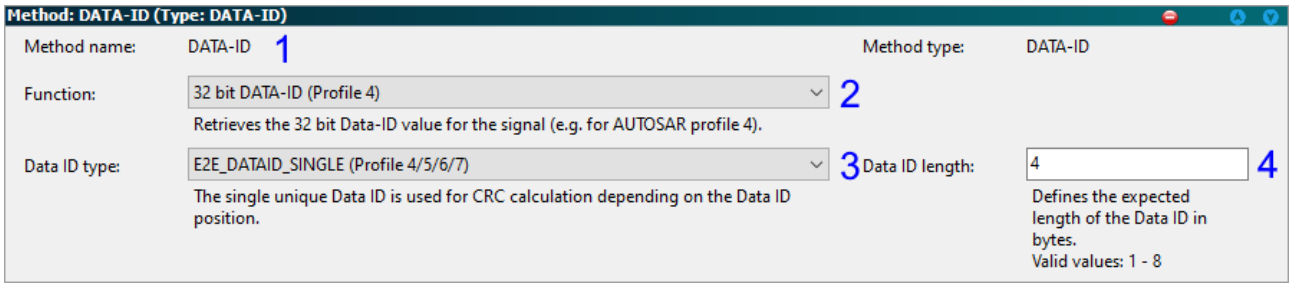


Figure 142: Custom E2E – method “DATA-ID” for custom “AUTOSAR Profile 4 (Signal-Based)”

No	Description
1	Method: DATA-ID (Type: DATA-ID) The parameters of the DATA-ID method/calculation.
2	Function: Specifies the function which is called to retrieve the DATA-ID of the CRC calculation. See chapter 5.12.4.3.
3	Data ID type: Defines the type of Data ID (handling of the Data ID value). See chapter 5.12.4.7.
4	Data ID length: Defines the expected length of the Data Id value in bytes. Valid values are 1 - 8

In contrast to the Data ID configuration options available in the CRC method, the settings

- Data ID position
- Consider Data ID value ‘0x00’

are not provided.

These parameters are implicitly defined by the value of the DATA-ID signal itself.

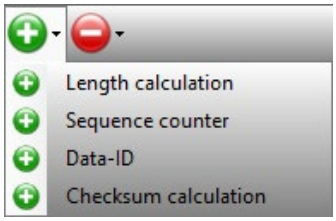
Restrictions: Custom AUTOSAR Profile 4 (Signal-Based)

- Only the Data ID type E2E_DATAID_SINGLE is supported.
  No additional Data ID formats are available within this custom profile.
- The Data ID type and Data ID length configured in the CRC method and the DATA-ID method must be identical.
  Both methods must use matching Data ID parameters to ensure correct and consistent end-to-end protection behavior.
- Data ID settings defined in the CRC method have no effect.
  The following configuration parameters of the CRC method are ignored:
  - Data ID type
  - Data ID length
  - Data ID position
  - Consider Data ID value 0x00

  These values are implicitly determined by the DATA-ID method and cannot be overridden via CRC configuration.

5.12.2.2 Additional Methods for Custom E2E-Profiles

Following additional methods can be added to a signal based custom profile:



- Length calculation
  - The number of 'length' methods in a Custom E2E profile is limited. There must be '0' or '1' 'length' methods.
  - The 'length' method should be located above a CRC method because the calculation should be performed before the CRC calculation. This is because normally the calculated length signal is part of the CRC calculation. Calculation of the counter after the CRC will lead to an invalid CRC value
  - More details can be found in chapter 5.12.4.1.
- Sequence counter
  - The number of counter methods in a Custom E2E profile is not limited.
  - The counter methods should be located above a CRC method because the calculation should be performed before the CRC calculation. This is because normally the calculated counter signal is part of the CRC calculation. Calculation of the counter after the CRC will lead to an invalid CRC value. When the counter is calculated after the CRC, the counter signal must be excluded from the CRC calculation via the Data Layout settings of the protected element (see chapter 5.6.4.1).
  - More details can be found in chapter 5.12.4.2.
- DATA-ID
  - The number of 'DATA-ID' methods in a Custom E2E profile is limited. There must be '0' or '1' DATA-ID methods.
  - The 'DATA-ID' method should be located above a CRC method because the calculation should be performed before the CRC calculation. This is because normally the calculated DATA-ID signal is part of the CRC calculation. Calculation of the counter after the CRC will lead to an invalid CRC value.
  - More details can be found in chapter 5.12.4.3
- CRC calculation
  - The number of CRC methods in a Custom E2E profile is limited. There must be '0' or '1' CRC method.
  - More details can be found in chapter 5.12.4.4.
- Checksum calculation
  - The number of CHK methods in a Custom E2E profile is limited. There must be '0' or '1' CHK method.
  - The CHK Method must be
    - the single method of the Custom E2E profile
    - or in combination with leading counter methods
      This is because normally the calculated counter signal is part of the CHK calculation. Calculation of the counter after the CHK will lead to an invalid CHK value.
    - or located below the CRC method because the calculation must be performed after the CRC method.

This is because the CHK method includes the complete payload of the protected element except the CHK signal.
In this case the calculated CHK signal must be the last signal of the protected element because currently excluding the CHK signal works only fine under this condition.

- The CHK method to calculate a simple checksum. As opposed to an CRC method the checksum method cannot specify a polynomial. All other method parameters are the same as for a CRC method.
  More details can be found in chapter 5.12.4.5.

5.12.3 Configure a Stack Based Custom E2E-Profile

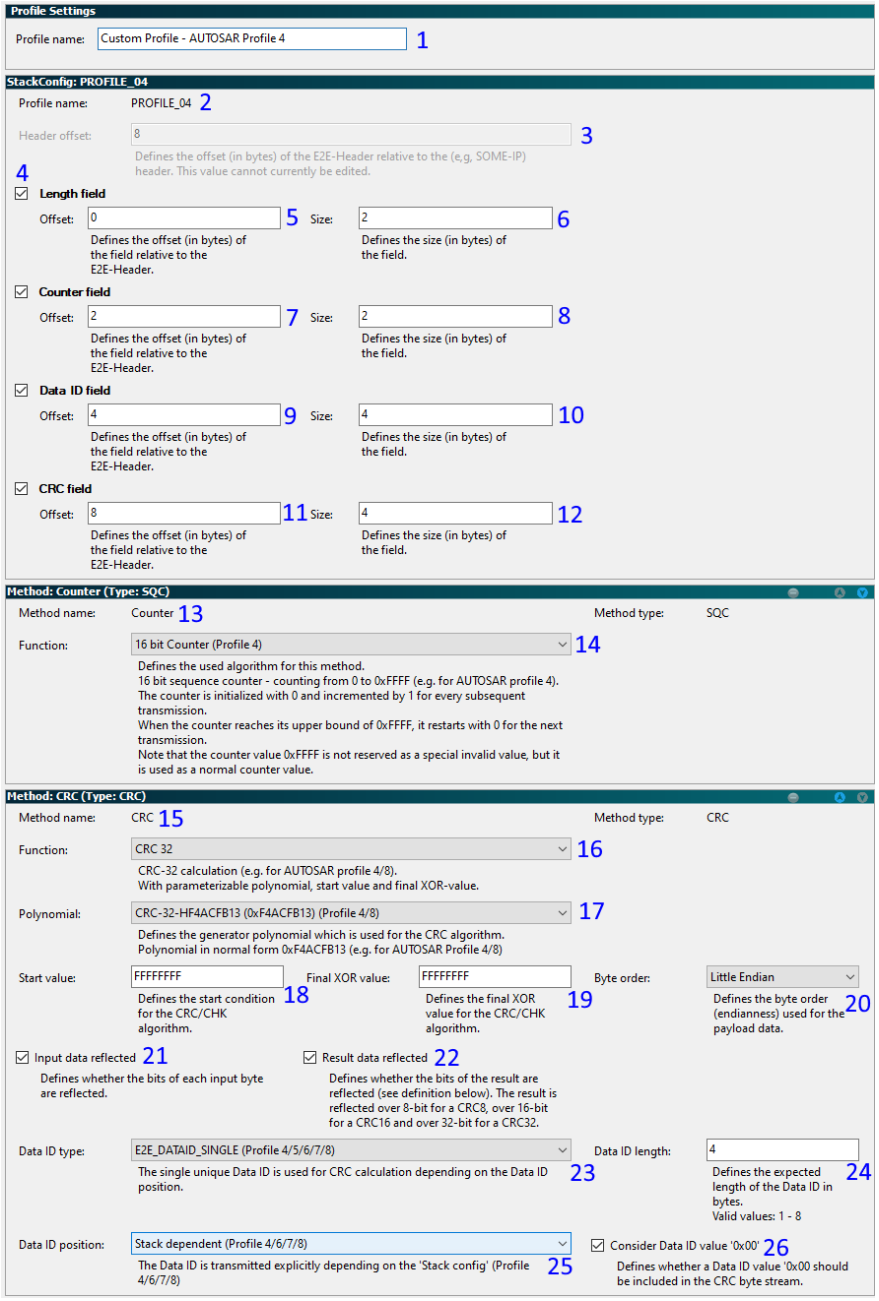


Figure 143: Custom E2E –parameters of a stack based custom E2E-Profile

No	Description
1	Name: The unique name of the custom profile.
2	Stack Config: PROFILE_ID The parameters of the stack (header/header fields) configuration.
3	Header offset: Defines the Offset (in bytes) of the E2E-Header relative to the (e.g. SOME/IP) header. Currently this parameter is read only and fixed to 8 bytes.
4	Enables/Disables a complete field. Disabling a field will set the offset and size to zero. A disabled filed will not be validated
5/6	Length field Offset: Defines the offset (in bytes) of the length field relative to the start of the E2E-Header. Size: Defines the size (in bytes) of the length field.
7/8	Counter field This field contains the calculated counter value. Offset: Defines the offset (in bytes) of the counter field relative to the start of the E2E-Header. Size: Defines the size (in bytes) of the counter field. A size of zero means that the CRC value is not transmitted.
9/10	Data ID field Profile 4/6: This field contains the explicit transmitted Data ID. Profile 5: The field value should be zero since the profile 5 does not transmit the Data ID value explicit. Offset: Defines the offset (in bytes) of the Data ID field relative to the start of the E2E-Header. Size: Defines the size (in bytes) of the Data ID field.
11/12	CRC field Offset: Defines the offset (in bytes) of the CRC field relative to the start of the E2E-Header. Size: Defines the size (in bytes) of the CRC field. A size of zero means that the CRC value is not transmitted.
13	Method: Counter (Type: SQC) The parameters of the counter method/calculation.
14	Function: Specified the function which is called to calculate the next counter value. See chapter 5.12.4.1.
15	Method: CRC (Type: CRC) The parameters of the CRC method/calculation.
16	Function: Specifies the function which is called to calculate the CRC value. See chapter 5.12.4.4.
17	Polynomial: Defines the polynomial to use for the CRC algorithm. See chapter 5.12.4.6.
18	Start value: Defines the start (initial) value of the CRC/CHK algorithm.
19	Final XOR value: Defines the final XOR value of the CRC/CHK algorithm.
20	Byte order: Defines the byte order (endianness) used for the payload data.
21	Input data reflected: Defines whether the bits of each input byte are reflected.
22	Result data reflected: Defines whether the bits of the result are reflected. The result is reflected over • 8 bit for a CRC8 • 16 bit for a CRC 16 • 32 bit for a CRC 32 • 64 bit for a CRC 64
23	Data ID type: Defines the type of the Data ID (handling of the Data ID value). See chapter 5.12.4.7.
24	Data ID length: Defines the expected length of the Data Id value in bytes. Valid values are 1 - 8
25	Data ID position: Specifies the position of the Data Id in the CRC input stream. See chapter 5.12.4.8.
26	Consider Data ID value ‘0x00’: Defines whether a Data ID value ‘0x00’ should be included in the CRC input stream.

	NOTICE
	Stack based profiles does not provide additional methods because currently only a fix set of stack fields are provided.

5.12.3.1 Constraints for Stack Based Custom E2E-Profiles

The following constraints apply to the stack based custom E2E-Profiles and their fields:

- The complete stack header length is limited to 32 bytes.
  Therefore, all fields (Offset/Size) must not exceed this limit.
- The size of each field must match a real datatype.
  Therefore the filed size is limited to the following values: [0, 1, 2, 4, 8]
- A Field size of '0' means that the field is not present in the header.
  The Length field does not support the '0' value.

5.12.4 Parameter Description

5.12.4.1 Length Function

The following length functions are available:

- 16 bit Length
  Returns the byte length of the protected payload.
  The Length includes user data + E2E Header (CRC + Counter + Length + DATA-ID).

5.12.4.2 Counter Function

The following counter functions are available:

- 4 bit Counter (Profile 1)
  4 bit sequence counter - counting from 0 to 14 (0xE) (e.g. for AUTOSAR profile 1).
  The counter is initialized with 0 and incremented by 1 for every subsequent transmission.
  When the counter reaches the upper bound of 14 (0xE), it restarts with 0 for the next transmission (i.e. value 0xF is skipped).
- 4 bit Counter (Profile 2)
  4 bit sequence counter - counting from 0 to 15 (0xF) (e.g. for AUTOSAR profile 2).
  The counter is initialized with 0 and incremented by 1 for every subsequent transmission.
  When the counter reaches its upper bound of 15 (0xF), it restarts with 0 for the next transmission.
  Note that the counter value 0xF is not reserved as a special invalid value, but it is used as a normal counter value.
- 16 bit Counter (Profile 4)
  16 bit sequence counter - counting from 0 to 0xFFFF (e.g. for AUTOSAR profile 4).
  The counter is initialized with 0 and incremented by 1 for every subsequent transmission.
  When the counter reaches its upper bound of 0xFFFF, it restarts with 0 for the next transmission.
  Note that the counter value 0xFFFF is not reserved as a special invalid value, but it is used as a normal counter value.
- 8 bit Counter (Profile 5/6)
  8 bit sequence counter - counting from 0 to 0xFF (e.g. for AUTOSAR profile 5/6).
  The counter is initialized with 0 and incremented by 1 for every subsequent transmission.
  When the counter reaches its upper bound of 0xFF, it restarts with 0 for the next transmission.
  Note that the counter value 0xFF is not reserved as a special invalid value, but it is used as a normal counter value.
- 32 bit Counter (Profile 7/8)
  32 bit sequence counter - counting from 0 to 0xFFFFFFFF (e.g. for AUTOSAR profile 7/8).
  The counter is initialized with 0 and incremented by 1 for every subsequent transmission.
  When the counter reaches its upper bound of 0xFFFFFFFF, it restarts with 0 for the next transmission.

Note that the counter value 0xFFFFFFFF is not reserved as a special invalid value, but it is used as a normal counter value.

5.12.4.3 DATA-ID Function

The following DATA-ID functions are available:

- 32 bit DATA-ID (Profile 4)
  Returns the 32 bit DATA-ID used for the E2E protection of the protected element.

5.12.4.4 CRC Function

The following CRC functions are available:

- CRC 8
  CRC-8 calculation (e.g. for AUTOSAR profile 1/2).
  With parameterizable polynomial, start value and final XOR-value.
- CRC 16
  CRC-16 calculation (e.g. for AUTOSAR profile 5/6).
  With parameterizable polynomial, start value and final XOR-value.
- CRC 32
  CRC-32 calculation (e.g. for AUTOSAR profile 4/8)
  With parameterizable polynomial, start value and final XOR-value.
- CRC 64
  CRC-64 calculation (e.g. for AUTOSAR profile 7).
  With parameterizable polynomial, start value and final XOR-value.

5.12.4.5 Checksum (CHK) Function

The following CHK functions are available:

- 8 bit checksum calculation with parameterizable start value and final XOR-value.
  Calculates the checksum via bytewise XOR over the payload data.
  The parameter Input-/Result data reflected are not supported (the parameters are available but have no effect).
- 4 bit CHK J1993 with parameterizable start value and final XOR-value.
  E.g. to create a profile to calculate the checksum of the SAE J1939 protocol.

5.12.4.6 Polynomial

The following polynomials are available:

- CRC-8-SAE J1850 (0x1D) (Profile 1)
  Defines the generator polynomial which is used for the CRC algorithm.
  Polynomial in normal form 0x1D (x8 + x4 + x3 + x2 + 1) (e.g. for AUTOSAR Profile 1)
  This CRC polynomial is also known as CRC-8-SAE J1850
- CRC-8-H2F (0x2F) (Profile 2)
  Defines the generator polynomial which is used for the CRC algorithm.
  Polynomial in normal form 0x2F (x8 + x5 + x3 + x2 + x + 1) (e.g. for AUTOSAR Profile 2)
  This CRC polynomial is also known as CRC-8-SAE J1850
- CRC-32-HF4ACFB13 (0xF4ACFB13) (Profile 4/8)
  Defines the generator polynomial which is used for the CRC algorithm.
  Polynomial in normal form 0xF4ACFB13 (e.g. for AUTOSAR Profile 4/8)
- CRC-16-H1021 (0x1021) (Profile 5/6)
  Defines the generator polynomial which is used for the CRC algorithm.
  Polynomial in normal form 0x1021 (e.g. for AUTOSAR Profile 5/6)
- CRC-64 (ECMA - 0x42F0E1EBA9EA3693) (Profile 7)
  Defines the generator polynomial which is used for the CRC algorithm.
  Polynomial in normal form 0x42F0E1EBA9EA3693 (e.g. for AUTOSAR profile 7)
  This CRC polynomial is also known as CRC-64 (ECMA)

- CRC-8-H9B (0x9B) (Profile WCDMA8)
  Defines the generator polynomial which is used for the CRC algorithm.
  Polynomial in normal form 0x9B (e.g. for WCDMA8)

5.12.4.7 Data ID Type

The following Data ID types are available:

- E2E_P01_DATAID_LOW (Profile 1)
  Only the low byte is included, and the high byte is never used.
  This is equals to the situation if the Data IDs (in a given application) are only 8 bits.
- E2E_P01_DATAID_BOTH (Profile 1A)
  Both two bytes (double ID configuration) are included in the CRC. First low byte and then high byte.
- E2E_P01_DATAID_ALT (Profile 1B)
  Depending on parity of the counter (alternating ID configuration) the high and the low byte is included.
  For even counter values the low byte is included and for odd counter values the high byte is included.
- E2E_P01_DATAID_NIBBLE (Profile 1C)
  The high nibble of high byte of Data ID is not used (it is 0x0), as the Data ID is limited to 12 bits, the low nibble of high byte of Data ID is transmitted explicitly and covered by CRC calculation when computing the CRC over Data.
  The low byte is not transmitted, but it is included in the CRC computation as start value depending on the Data ID position.
- E2E_P02_DATAID_MULTIE (Profile 2)
  The specific Data ID used for CRC calculation is selected from a pre-defined Data ID list with 16 Data IDs using the value of the Counter as an index.
  Each data (e.g PDU, signal group), which is protected by a CRC owns a dedicated Data ID list.
- E2E_DATAID_SINGLE (Profile 4/5/6/7/8)
  The single unique Data ID is used for CRC calculation depending on the Data ID position.

5.12.4.8 Data ID Position

The following Data ID positions are available:

- First (Profile 1)
  The Data ID is included as first element in the CRC byte stream (Profile 1).
- Last (Profile 2/5)
  The Data ID is included as last element in the CRC byte stream (Profile 2/5).
- CRC start value
  The Data ID is used as CRC 'Start value'.
  The configured 'Start value' parameter is not considered.
- Signal defined (Profile 4/6/7/8 Signal-Based)
  This option specifies that the DATA ID is represented by a signal (in the CRC payload) and must not be considered in the CRC calculation itself. Therefore, following parameters are obsolete and only available for compatibility reasons:
  - Data ID type
  - Data ID length
  - Consider Data ID value ‘0x00’

  In most cases, the DATA-ID signal is then calculated using the DATA-ID method.
- Stack dependent (Profile 4/6/7/8)
  This option is only available for stack based profiles.
  The Data ID is transmitted explicitly depending on the 'Stack config' (Profile 4/6/7/8).
  See chapter 5.12.3 Configure a Stack Based Custom E2E-Profile

5.13 RBS Based Replay

The RBS based Replay dialog allows to load mf4 log files and replay it for configured busses. Preconditions:

- A valid mf4 log file must be available, which is supported by FL3X Config.
  Mf4 files have many different options for logging data. Not all variants of mf4 files are supported by RBS Based Replay. FL3X Config supports the mf4 files, which are logged by the STAR devices and the sorted Vector CANoe mf4 files, which contain complete communication.
- The RBS busses must be configured before using the RBS Based Replay.
- Before the project is compiled and flashed to the device:
  - The option “Replay mode” must be activated (see Figure 144).
  - The ID-Defines must be generated. E.g. the ‘Always create the ID-Defines’ option can be used. See chapter 5.22.1.1 for more information.

A brief explanation:

After clicking on the **[Start]** button, each entry in the mapped mf4 channel is analyzed. If a frame (e.g., CAN/FlexRay) or a service item (event, field, method) is found for the configured and assigned FL3X Config busses, then the ID-Defines file is analyzed to get the proper function addresses (set data, trigger event). If the relevant information is found in the ID-Defines, the item data (payload, timestamp, item meta information, function addresses) is sent to the connected device. The device buffers the received data and tries to send it with the given timestamp. The feature is called “RBS based Replay” because an RBS must be configured to send the replay data. The transmission settings are important for this feature. If the frames of an RBS are configured with a Cyclic or CyclicAndRouting transmit behavior, this will also affect the replay. The resulting transmit behavior at runtime will not match the loaded mf4 file. For most use cases, it is necessary to change the transmission settings for all replay elements to ‘routing’ (see chapter 5.6.7 for more details).

In ‘RBS Based Replay’ dialog, it is possible to load multiple Mf4 files and use their channels. A FL3X Config Bus can be mapped to one or multiple mf4 channels.

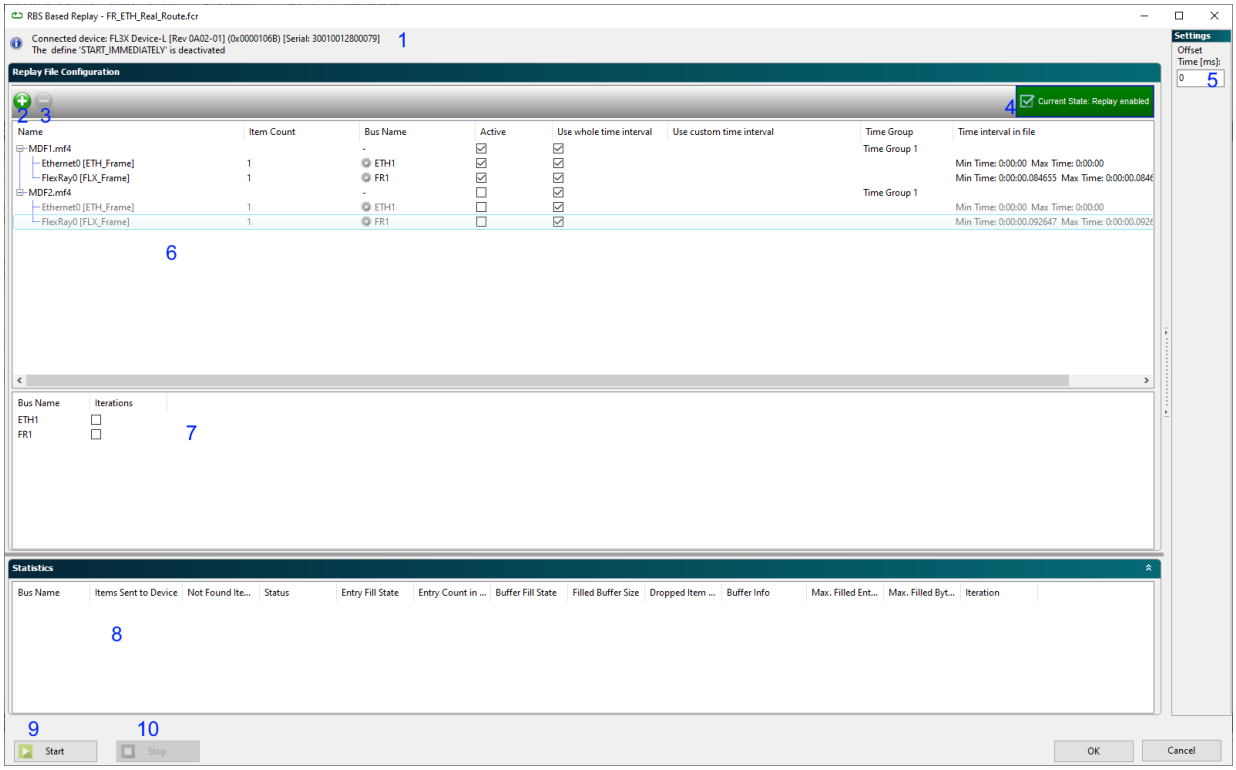


Figure 144: “RBS based Replay” dialog

<table>
<tr><th>No</th><th>Description</th></tr>
<tr><td>1</td><td>The information and any warnings about the device connection status are shown in this area.</td></tr>
<tr><td>2</td><td>Adds a mf4 file. Only a single mf4 file is supported by the RBS based Replay dialog. The add button will be enabled if no mf4 file was added yet and the ‘Replay mode’ (see No 4) is activated.</td></tr>
<tr><td>3</td><td>Remove the loaded mf4 file.</td></tr>
<tr><td>4</td><td>Replay mode: Adding a mf4 file is allowed by activated ‘Replay mode’. If the project is compiled and flashed with activated ‘Replay mode’, then each configured RBS does not start automatically after power on. This is ensured by deactivating the ‘START_IMMEDIATELY’ user defines (see chapter 5.6.7.8) when the ‘Replay mode’ is activated for the first time. Even if it is possible to reactivate the ‘START_IMMEDIATELY’ again, there should not be done any manual change to the user defines to avoid any negative effect on the RBS based replay.</td></tr>
<tr><td>5</td><td>Offset time determines the time to wait after the replay is started. If necessary, this time can be used for the bus synchronization or SOME/IP Service Discovery.</td></tr>
<tr><td>6</td><td>This list shows the information about the loaded mf4 file and lets the user map the FL3X Config bus with the mf4 channel.
Columns:
<table>
<tr><th>Column name</th><th>Editable</th><th>Description</th></tr>
<tr><td>Name</td><td>No</td><td>Name of the mf4 file and the loaded channel groups</td></tr>
<tr><td>Item Count</td><td>No</td><td>Count of the items, which are part in the mf4 channel groups.</td></tr>
<tr><td>Bus Name</td><td>Yes</td><td>This is an editable column, which should be used to map the FL3X Config bus to the mf4 channel group.</td></tr>
<tr><td>Active</td><td>Yes</td><td>This option determines whether the log file should be replayed on the corresponding mapped bus.</td></tr>
<tr><td>Use whole time interval</td><td>Yes</td><td>This option determines whether all messages of the mf4 channel should be used</td></tr>
<tr><td>Use custom time interval</td><td>Yes</td><td>This option is editable only if the ‘Use the whole interval’ option is disabled.
Editing this cell opens a view where the start and end time can be entered.</td></tr>
<tr><td>TraceSync</td><td>Yes</td><td>This option is editable for the mf4 file entry and is used to group the files, which should have the same time base. The mf4 file with the earliest creation date determines the start time. Data of the other files in the same time group are added to a list with time offsets to send all data sequentially. As time offset, the creation data of those files are used, which are saved in the Mf4 file. Data of the other files with different time groups are sent parallel.</td></tr>
<tr><td>Time interval in file</td><td>No</td><td>The start and end time of the logging (based on the timestamps in the loaded mf4 file) are shown in this cell</td></tr>
<tr><td>Measurement Start Time</td><td>No</td><td>The start time of the loaded mf4 file.</td></tr>
</table></td></tr>
<tr><td>7</td><td>This list shows the configured FL3X Config buses, which are mapped to the mf4 channels on the list above and are used for setting the iterations.
Columns:
<table>
<tr><th>Column name</th><th>Editable</th><th>Description</th></tr>
<tr><td>Iterations</td><td>Yes</td><td>This option determines whether the log file should be replayed in an endless loop or just one time.</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>No</th><th>Description</th></tr>
<tr><td>8</td><td>This list shows the statistics information after the replay is started.
Columns:
<table>
<tr><th>Column name</th><th>Description</th></tr>
<tr><td>Bus Name</td><td>Name of FL3X Config bus, where the replay is running.</td></tr>
<tr><td>Items Sent to Device</td><td>The count of the items which are sent to the device. These are the messages which are sent by the device. These items are stored in a buffer in the device and replayed due to timestamps. The ratio of the number of elements found to the total number of elements may not seem plausible for Ethernet elements, as the actual elements are decoded from the raw data. Furthermore, individual dummy packets may be sent to the device, which are required for time synchronization</td></tr>
<tr><td>Not Found Items</td><td>The count of the items, which are either not defined in the RBS configuration or have no information in the ID-Defines file.</td></tr>
<tr><td>Status</td><td>Actual replay status in the device.</td></tr>
<tr><td>Entry Fill State</td><td>Visualizes the entry fill state in the buffers. E.g. if the buffer allows 100 frames and it contains 50 frames, the column shows a progress bar filled 50%.</td></tr>
<tr><td>Entry Count in Buffer</td><td>Number of entries in the buffer.</td></tr>
<tr><td>Buffer Fill Size</td><td>Visualizes the buffer fill state E.g. if the buffer allows 100 KB and X frames and the total size of the buffered frames are 20 KB, the column shows a progress bar filled 20%.</td></tr>
<tr><td>Filled Buffer Size</td><td>Show the filled buffer size in bytes.</td></tr>
<tr><td>Dropped Item Count</td><td>Show the number of items which could not be sent from the device.</td></tr>
<tr><td>Max Filled Entry Count</td><td>Max count of frames, which the buffer had.</td></tr>
<tr><td>Max Filled Byte Count</td><td>Max memory occupied from the frames in the buffer in bytes.</td></tr>
<tr><td>Iteration</td><td>Displays the info how many times the content of the log file has sent to the device. If the ‘iterations’ option is enabled, the count will increase each time the content is sent to the device.</td></tr>
</table></td></tr>
<tr><td>9</td><td>Starts the replay</td></tr>
<tr><td>10</td><td>Stops the replay</td></tr>
</table>

	Information
	- FL3X Config Analyzer (Windows [9]) must be opened **before** opening the RBS based Replay dialog if you want to use both dialogs at the same time. - RBS based Replay always influences the cyclic transmission timing of FlexRay.

	NOTICE
	If any Tx-Item is used in a routing, it will be ignored by the RBS based Replay (and not be sent with the data from mf4 file).

5.14 Database Update

There are two different types to update a database used in the project.

1. Database update to change the database of an RBS or a (transparent) gateway to different version of the database. See chapter 5.14.2
2. Database update to reconvert a database to use new features (e.g. necessary if the internal converter was updated). This will update all RBS/Gateway configurations which use parts of the reconverted database. See chapter 5.14.3

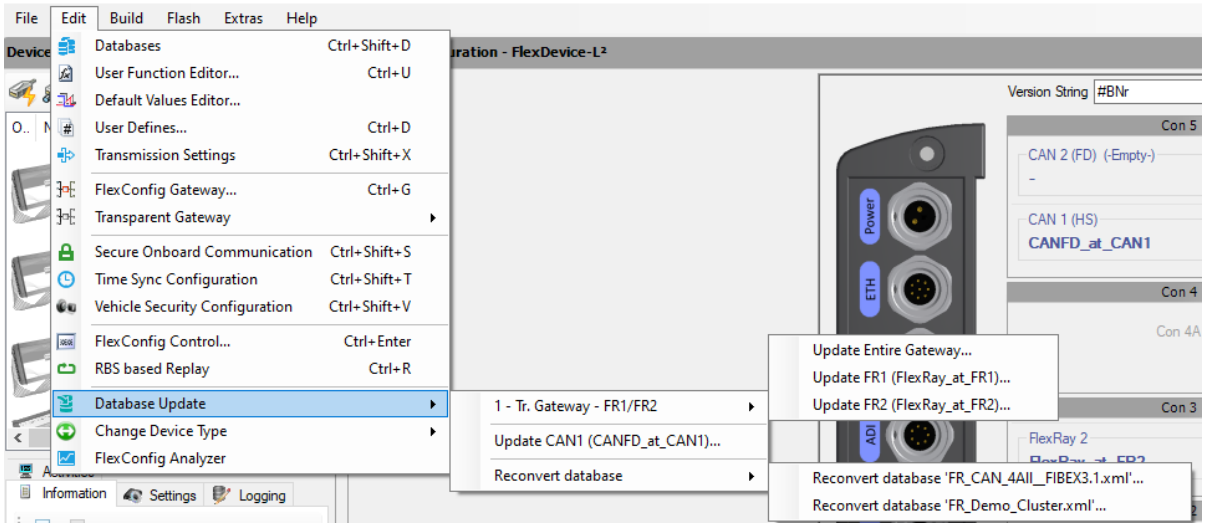


Figure 145: “Database Update” menu

During the update process, it is attempted to keep the existing configuration and assign the elements from the ‘old’ to the ‘new’ database (depending on the short name, timing, or unique ElementID (ho: OID)). A detailed log output (in csv format) shows the update results.

5.14.1 Database Update and E2E Profiles / OEM dependent Standard Signal Handlings)

During a Database update, the E2E-Profiles and OEM dependent Standard Signal Handlings will also be updated based on the new database.

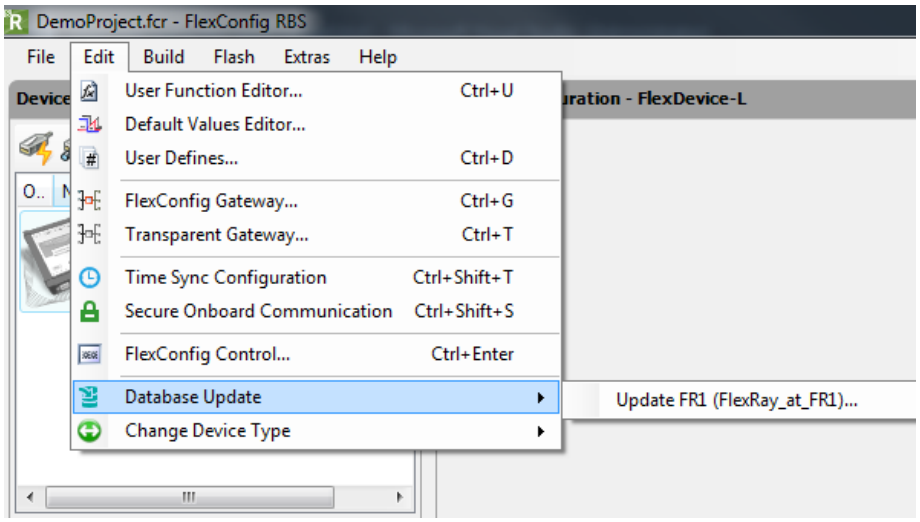
More information can be found in the OEM package documentation.

5.14.2 Database Update to Change the Used Database

This kind of database update is used to change the database of an RBS or a (transparent) gateway to a different version of the database. This feature needs at least the FL3X Professional license.

5.14.2.1 Change the Database for an RBS

In the menu bar open the entry ‘*Edit*’ -> ‘*Database Update*’.



A Database update is always performed only for one connector (RBS and gateway). It is possible to update a transparent gateway (see chapter 5.14.2.2) at once.

The “Change database” dialog (see Figure 146) will open after the selection of the connector.

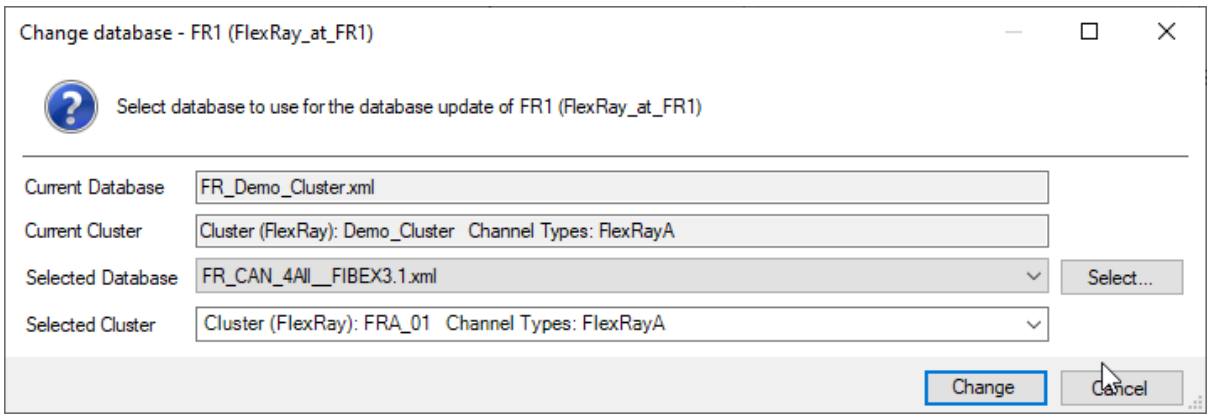


Figure 146: Database Update – “Change database” dialog

Firstly, the database which should replace the current used database, must be selected via the Dropdown “*Selected Database*”. With the button “*Select...*” the Database Picker (see chapter 5.5.1) can be used to select a project database or to add and select a new database. Then, if there are multiple clusters in the selected database, select the new cluster via the Dropdown “*Selected Cluster*”.

Then start the update process by clicking the button “*Change*”. This will open the “Update Database” dialog (see Figure 147) and start the update process.

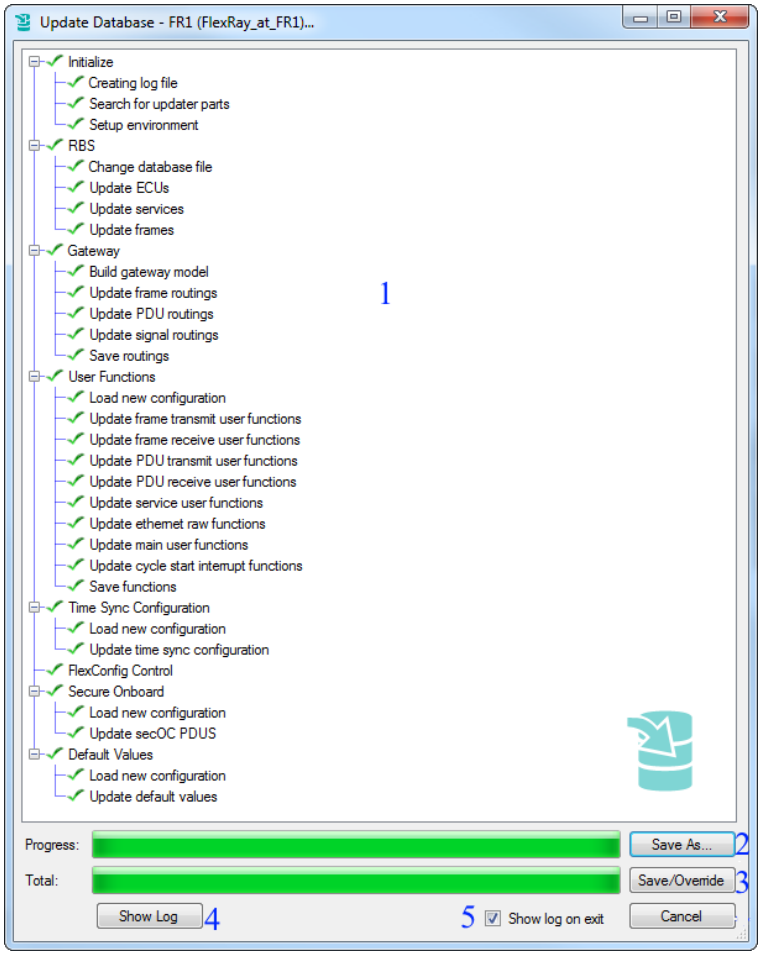


Figure 147: Database Update – “Update Database” dialog

No	Description
1	Shows the project items to be updated Following icons are shown: : Item was successfully updated : Item was updated but there are warnings : Item was not updated (the update was not successful) : Item update in progress
2	The updated project can be saved under a new name after clicking on this button
3	Clicking on this button will apply the changes to the current project
4	Shows the log output in csv format
5	The log file will be opened after closing this dialog if this option is activated

Table 96: Database Update – “Update Database” dialog

NOTICE

If there are any routings with scalings (see chapter 6.1.5), then these scalings are not changed/updated by a Database update! All scalings must be updated manually!

NOTICE

If the project is created for the FL3X Device L/L² with the FPGA “0x1061, then a change of the start-up/sync Frames of the real ECUs can cause Problems, because just one start-up/sync frame can be sent from the first FlexRay bus.

5.14.2.2 Change the Database for Transparent Gateway

The Database update for a transparent gateway corresponds to the Database update of an RBS. However, the following differences exist:

ECUs, which exist only in the new database, will automatically be assigned to gateway side “A”.

The two RBS configurations are updated according to the guidelines of a Transparent Gateway.

All routings of gateway are updated according to the guidelines of a Transparent Gateway.

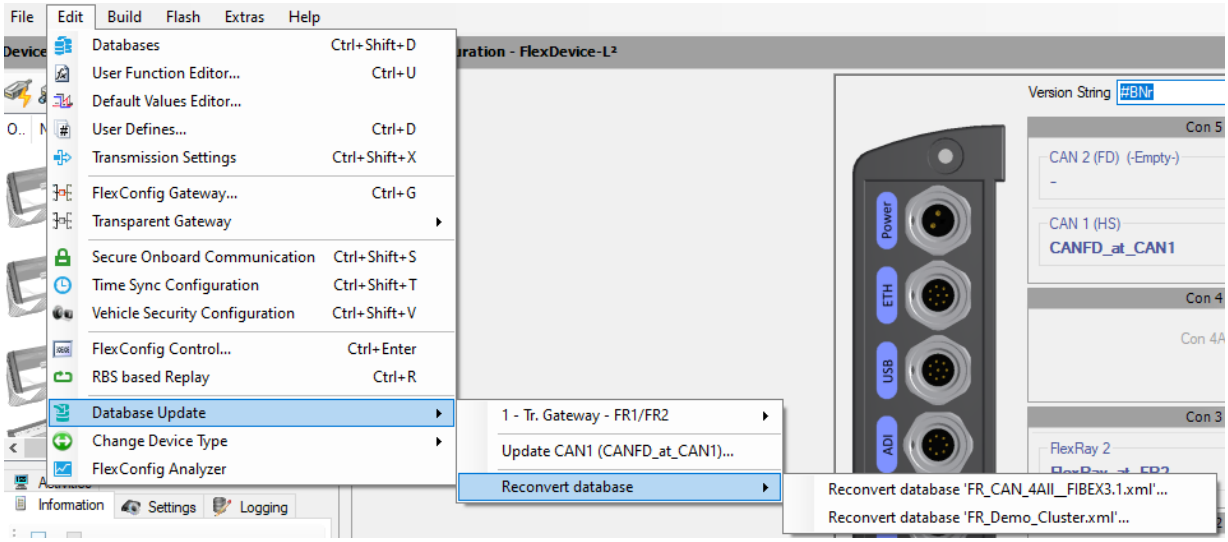
NOTICE

In case of a Transparent Gateway it is strictly recommended to update both connectors at once. An update of a single connector of a transparent gateway is not recommended!

5.14.3 Database Update to Reconvert a Database

This kind of database update is used to reconvert a database. This will update all RBS/Gateway configuration which uses parts of the reconverted database.

In the menu bar open the entry ‘*Edit*’ -> ‘*Database Update*’ -> *Reconvert database*’.



	Information
	Reconvert is only possible for used databases. If a database is added to the project but not used and therefore not loaded reconvert is not possible and the Reconvert entry will be disabled.

A database reconvert is always performed for the whole database and all RBS/gateway configurations. The “Reconvert database” dialog (see Figure 148) will open after the selection of the database.

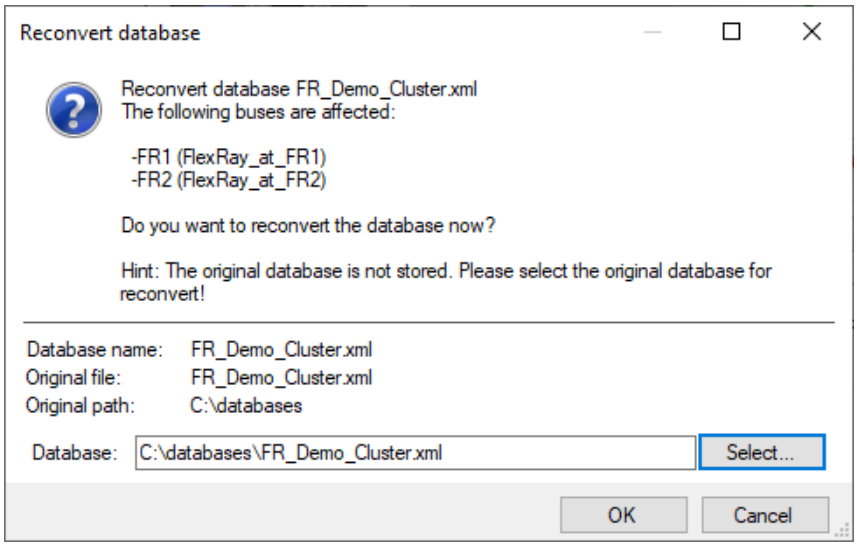


Figure 148: Database Update – “Reconvert database” dialog

The “Reconvert database” dialog shows which database is being reconverted and which buses are affected.

If an automatic reconversion is not possible (e.g. because the *db.dat* is missing), the lower section of the dialog is enabled. With the button “*Select...*” the original database can be chosen from the file system.

	NOTICE
	A warning will be shown when the file name of a manually selected original file does not match the original file name. The selected file does not match the name of the original database! Please ensure that the selected database is correct. In this case the database name in the project will NOT be updated to the new database name!

Start the reconvert process by clicking the **[OK]** button. This will open the “Update Database” dialog (see Figure 147 in chapter 5.14.2.1) and start the update process.

5.15 Change Device Type

The change of the device type can be done via *'Edit->Change Device Type'* menu item. See Figure 149 for an example. Firstly, select your new device type (**1**), then the revision of the device (**2**) and finally the wanted FPGA image (**3**).

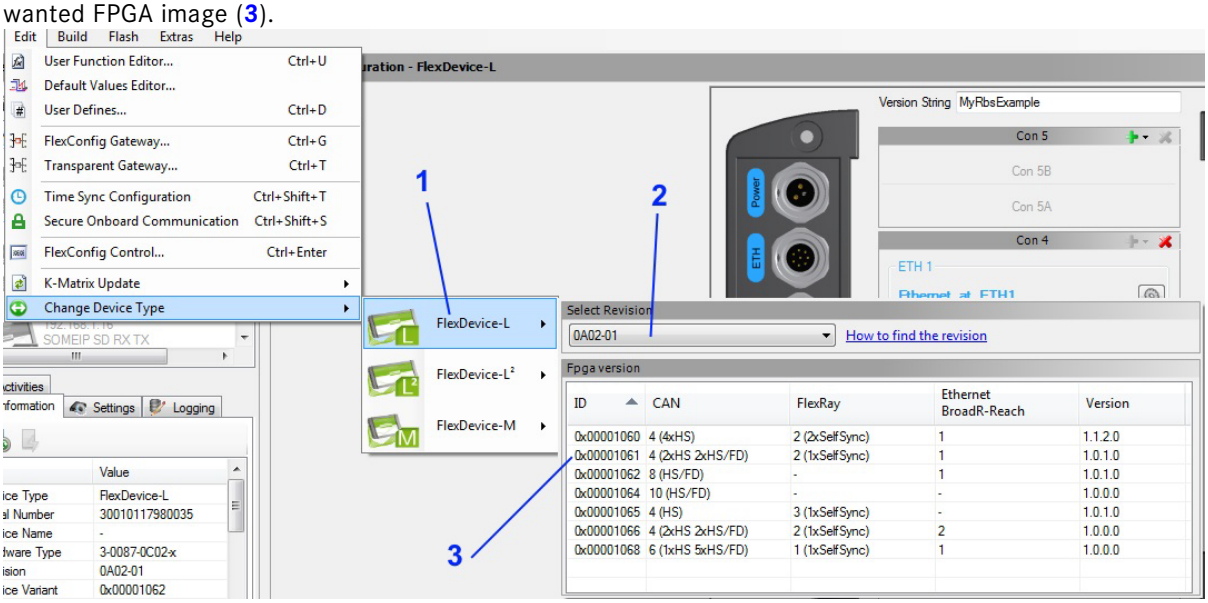


Figure 149: Change Device Type

Due to the reason that every device type supports different features, as for example different connectors, the device type change will imply the creation of a new project. If there are still unsaved changes in the actual project, the following dialog box appears first (see Figure 150).

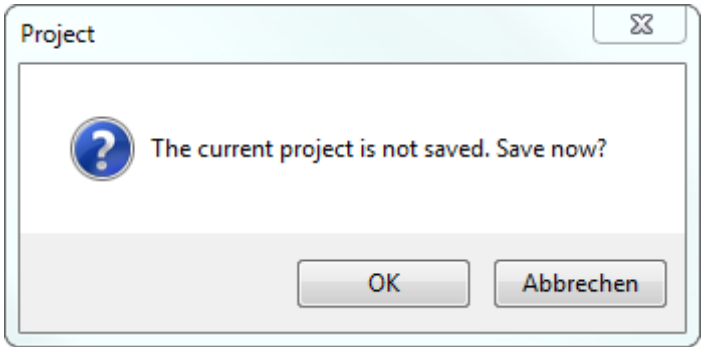


Figure 150: Save Project changes dialog

It must be confirmed that all actual changes are saved. If the '*Cancel*' button is clicked, the changing of the device will be aborted!

In the next step, the "Change Device..." window (see Figure 151) will be opened.

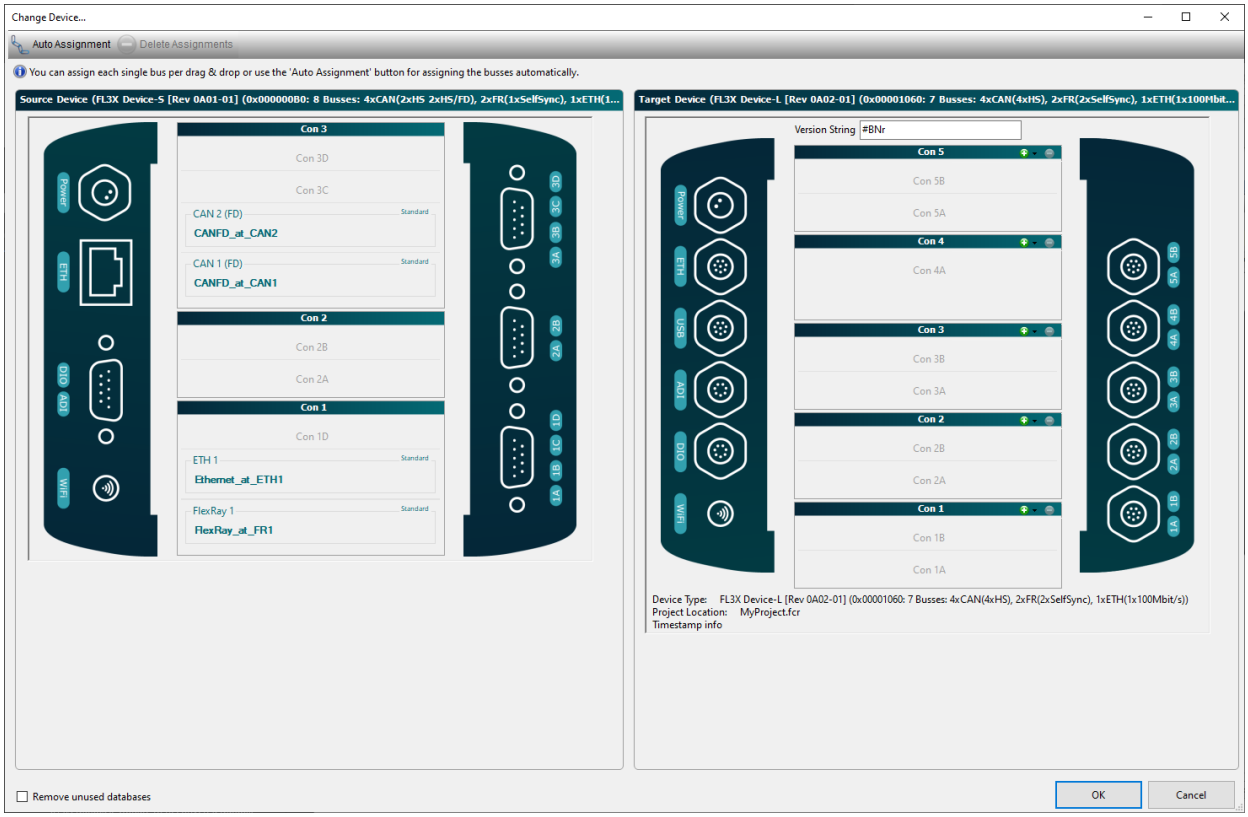


Figure 151: “Change device ...” dialog (here source FL3X Device-S, target FL3X Device-L)

In this window assignments between the current device (“Source”) and the new device (“Target”) can be made. Assignment means that the bus of the source connector is mapped to an appropriate target connector.

	NOTICE
	If FL3X System-M is used as the target device, it is necessary to make the slot configuration before assigning any connectors.

To create or delete assignments the following options are available:

1. Use the **[Auto Assignment]** button to create the assignments of the connectors automatically. Auto Assignment button is only then functional, if at least the FL3X Professional license is available.
2. Assignments can be made by hand via drag & drop. If a source bus is clicked and the mouse is slightly moved, all proper target connectors are shown with a green frame. The source bus must be dragged to the target connector until the mouse pointer appears. After that the selected bus can be dropped to the target.
   The following symbols can appear while a source bus is being assigned:
   - Assignment is not possible. Move the mouse pointer over this symbol to see more information.
   - Assignment is possible with some limitations. Move the mouse pointer over this symbol to see more information.
3. The delete button (red cross) of the target bus can be pressed to delete the appropriate assignment.
4. All assignments can be deleted by using the ‘*Delete Assignments*’ button.

Overview of supported features per FL3X Device variant: see chapter 2.8 Supported Target Hardware.

NOTICE
It is possible that one or more assignments could fail during “Auto Assignment”, as the target device might not support certain bus types or features.

NOTICE
The changing of a device will delete the user files with the extension “*.a*”.

NOTICE
All active user functions are deleted if they are created for or use parameters of the unassigned buses. Inactive user functions will not be deleted. If active ‘user main’ and ‘initialize’ functions use parameters of the unassigned busses, they will be deactivated.

After all assignments are made, the *“OK”*-button can be clicked. In the next step a dialog appears where a new project must be created. The dialog is nearly like the dialog in chapter 5.4.1.1, but in this case the project type and the device type must not be chosen again. Finally, all assignments which have been made are applied to the new project and the project will be opened.

NOTICE
After using “Change Device Type”, the project must be built again!

NOTICE
Any used hardware API functions (e.g., access to DIO or User LED) have no effect if the new device does not support them. - Unsupported functions are not removed from the project when the device type is changed. - There is no hint or warning if any unsupported functions are used in the project. For example, the FL3X System-M devices do not support any ADI/DIO or LED function calls via User Function. Such functions calls are ignored at runtime and have no effect.

5.16 Create Partner Project

Information
This feature requires the FL3X Config Enterprise license (see chapter 3.1.1).

NOTICE
This feature may still have issues and is marked as ‘Beta’.

The ‘Create Partner Project’ feature can be used for creating an partner project. The partner project has the opposite configuration to the current project. For example, if the current project has an ECU_A configured as ‘Real ECU’, then the partner project has the ECU_A configured as ‘Simulated ECU’.

The partner project can be used for an automated FL3X Device test. The partner project can be flashed to a separate device to provide the data required by the actual device or an ECU modelling. Please contact the STAR ELECTRONICS support team for more information about testing via FL3X Device.

A partner project always creates the file *ProjectData.xml* and the folder *ID_Defines* during compiling (independent from the global settings).

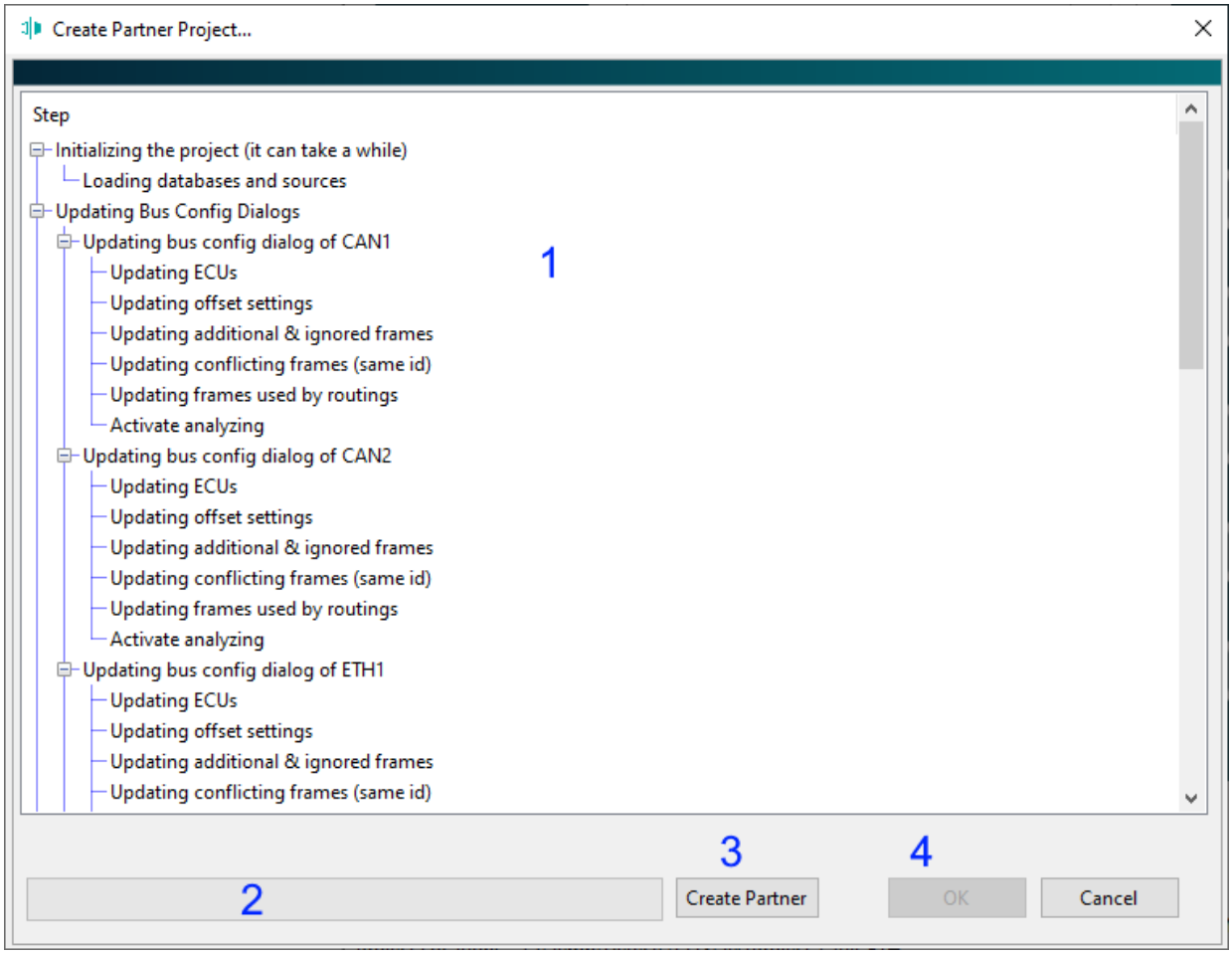


Figure 152: “Create Partner Project”

No	Description
1	Shows the process steps
2	Shows creation progress
3	Starts creation of the partner project
4	Opens a dialog for saving the partner project. The partner project will be opened with a new instance of FL3X Config.

Table 97: “Create Partner Project” dialog

Step	Action by creation of partner project
Initializing the project	The same databases are used as in the original project
Bus Configuration	The ECUs in the lists of real ECUs and ECUs to simulate are changed. The ECUs in the ignored list remain without any change. Offset settings for a FlexRay bus are deleted. Additional Frames added by the user are deleted. The frames which are sent by the original device are ignored (Frame IDs are taken into account in this case). Missing sync/startup. frames for a FlexRay bus are added. Transceiver option by an ETH Bus is changed (Master/Slave) Consumed and provided services are changed The “Analyzing” option will be activated. Also the frames which are used by routings as Rx are automatically simulated.
Tx Behavior	Only the items used by routings are sent cyclic. Other items will not be sent. The transmission settings can be changed manually after creating the partner project, if this behavior is not wanted

Step	Action by creation of partner project
User Defines	No change (same as original project)
Gateway	All routings are deleted.
User Functions	All user functions are disabled.
Default Values	The custom default values (edited by the user) are deleted. Database default values are used.
Manipulations	All manipulations in FL3X config Control are deleted
Time Sync Configuration	A time master bus in the original project is created a time slave bus. The device internal clock is used in this case as time master.
SecOC Configuration	No change (same as original project)

Table 98: “Create Partner Project” steps

5.17 Support for Visual Studio Code

The support for Visual Studio Code allows to edit existing User Functions and build the project outside of the FL3X Config. It requires the installation of an extension to Visual Studio Code. This extension also provides a way to insert code snippets into the User Functions (e.g., to set a specific signal value).

This feature is only then functional, if FL3X Enterprise license is available.

To use this feature, it is required to activate the setting “Generate Files for Visual Studio Code” (see chapter 5.22.1.1).

5.17.1 Install the FL3X Config Integration Extension

To install the extension:

1) Open the extensions tab in Visual Studio Code and click on the button with the three dots.

2) Click the last element in the list “*Install from VSIX...*” (see also Figure 153).

3) Select the *.vsix* extension package.

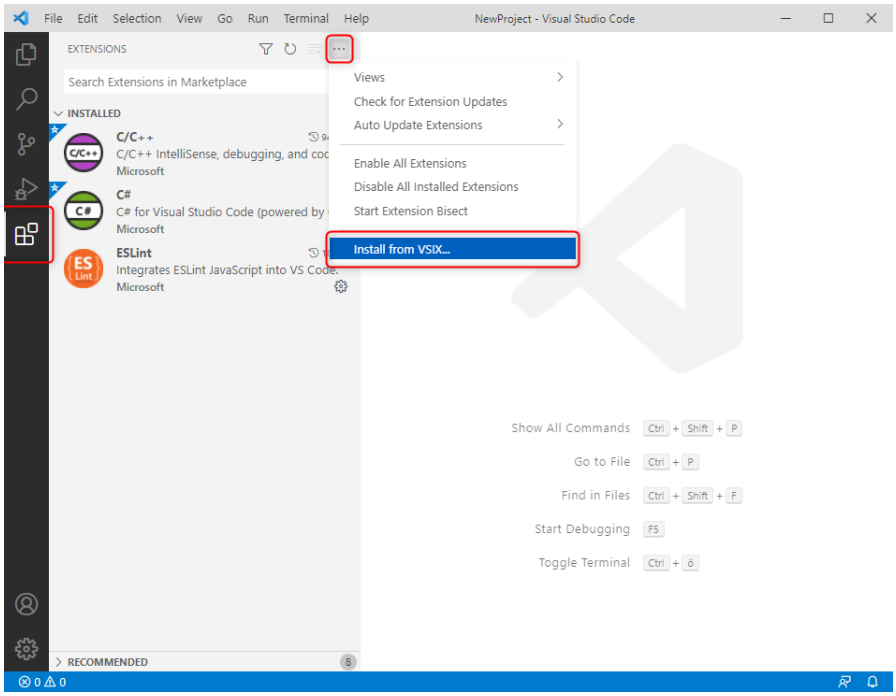


Figure 153: Installing the Visual Studio Code extension

	Information
	The Visual Studio Code extension package (**.vsix* file) can be found in the folder *C:\Program Files\StarCooperation\FL3X Config %VERSION%\VSCode*

5.17.2 Open a FL3X Config Project

To open a FL3X Config project, select “File/Open Folder…” in Visual Studio Code. Now select the project folder (the folder containing the *.fcr* file). When the project is loaded, it should look like Figure 154.

The User Functions can be found inside the User Functions folder. Currently only editing of existing functions is supported. So, adding files outside of FL3X Config will not work and only the **.c* Files inside of the User Functions folder can be edited.

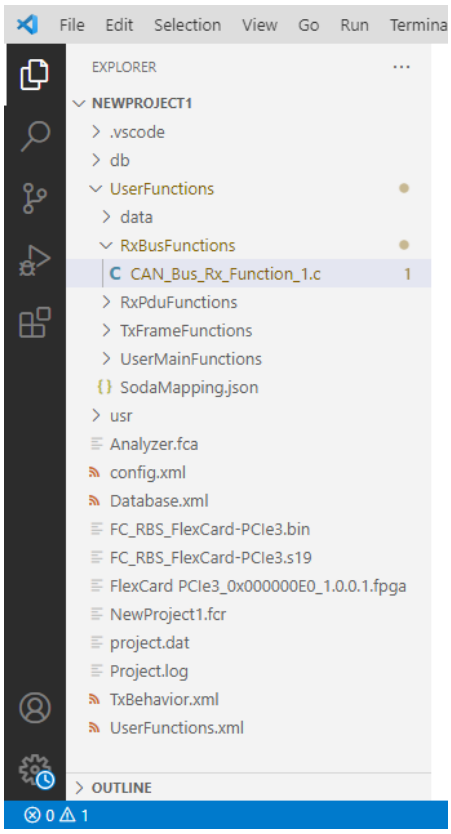


Figure 154: Visual Studio Code – FL3X Config Integration –FL3X Config project folder

After a project was loaded, a message like Figure 155 should be shown to indicate that the extension was successfully initialized, and the code snippet suggestions can be used.

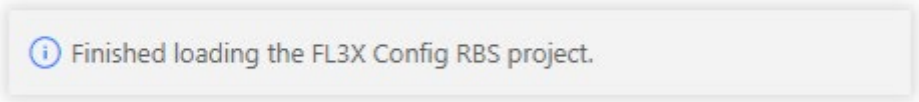


Figure 155: Visual Studio Code – FL3X Config Integration –FL3X Config project initialized

5.17.3 Using Code Snippet Suggestions

In the User Function Editor there is a menu on the left that shows different code snippets that can be inserted into the functions.

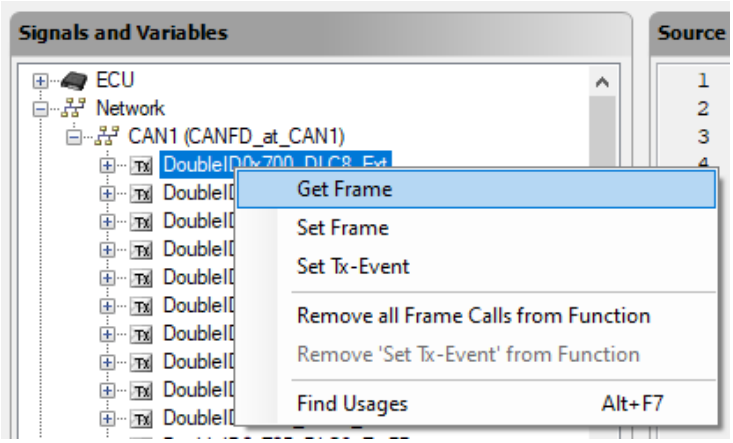


Figure 156: User Function Editor - Code Snippets selection

To make the same functionality available in Visual Studio Code the extension is needed. With the extension installed the user can navigate through the same structure as seen in the User Functions Editor by typing in “RBS” followed by the names of each level. This can be seen in the following image where there are three code snipped suggestion or the next level of the selected frame (the PDU pdu_DoubleID0x700_DLC8_Ext).

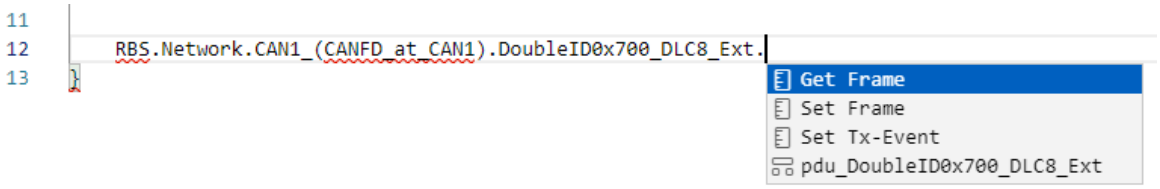


Figure 157: Visual Studio Code – FL3X Config Integration - Code Snippets example 1

When typing in the word RBS followed by a dot six suggestions should be visible like in the following image. If not try pressing the keys for the “Trigger Suggest” Command (the default keyboard shortcut should be Ctrl + Space).

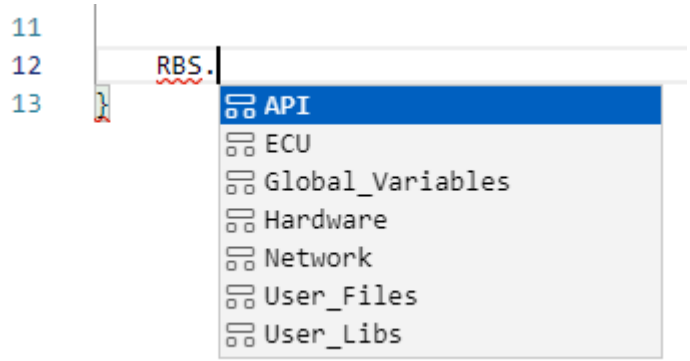


Figure 158: Visual Studio Code – FL3X Config Integration - Code Snippets example 2

After that you can navigate through the suggestions with the arrow keys and pressing dot or enter.

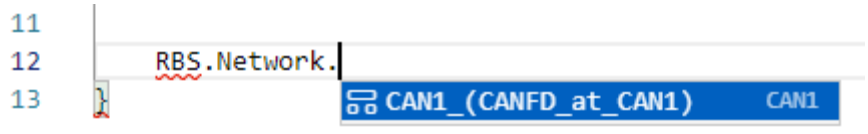


Figure 159: Visual Studio Code – FL3X Config Integration - Code Snippets example 3

In this example we reached a point where there are code snippets that can be inserted. The three suggestions “Set Bus On”, “Set Bus Tx Off” and “Set Bus Tx On” are all code snippets that can be inserted into our function by pressing enter (Keybinding for the “acceptSelectedSuggestion” command). We can also see the code that will get inserted at our current position.

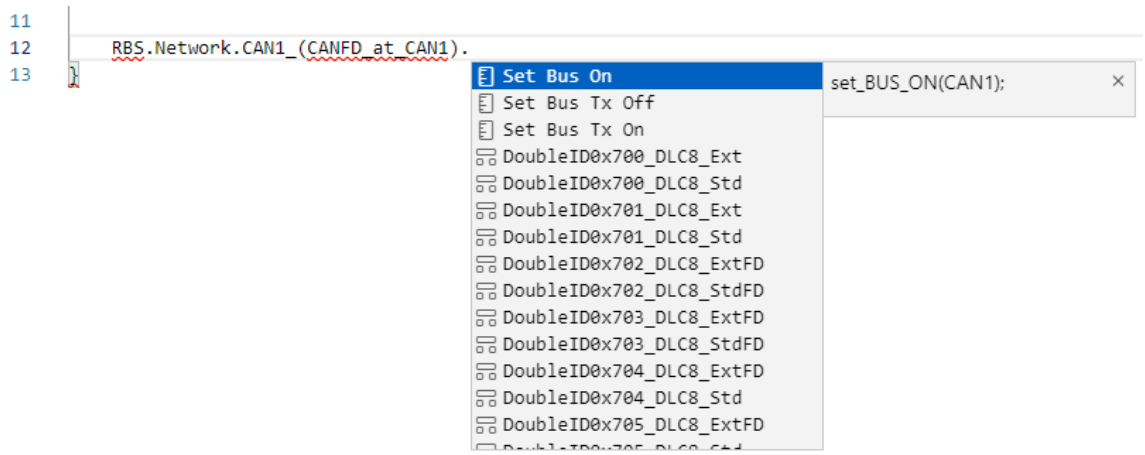


Figure 160: Visual Studio Code – FL3X Config Integration - Code Snippets example 4

After pressing enter, the code will get inserted, and it will look something like this.

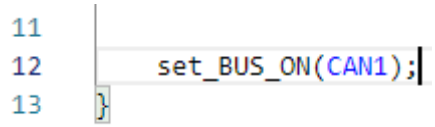


Figure 161: Visual Studio Code – FL3X Config Integration - Code Snippets example 5

5.17.4 Building the Project inside Visual Studio Code

To build the project, run the “Build FL3X Config Project” task. Navigate to “Terminal/Run Build Task…” and select the “Build FL3X Config Project” task. After that, the console on the bottom should open and the build should begin.
If the build task is not shown, be sure that the Visual Studio Code files are generated for the project (see chapter 5.22.1.1).

	NOTICE
	FL3X Config will override the files edited outside of the User Function Editor. To avoid losing changes it is HIGHLY recommended to **not** open the project parallel in FL3X Config and Visual Studio Code.

5.18 Handling for ECU Families

A FIBEX file has always a fixed configuration for all ECUs, which are used in the cluster. In some cases, it is not known which type of an ECU will be used (depending on the available hardware), until the cluster is realized. For example, 2 Versions of an ECU can be available, and it is not known, which version will be used until the cluster is implemented. This scenario can be considered in a FIBEX file. The FIBEX file contains in this case an ECU family for different versions of an ECU.

When ECU families are included in a FIBEX file that is imported by FL3X Config, the user must select which type of ECU should be used for the RBS/Gateway configuration. The following dialog (see Figure 162) is shown after importing a FIBEX file which contains ECU families.

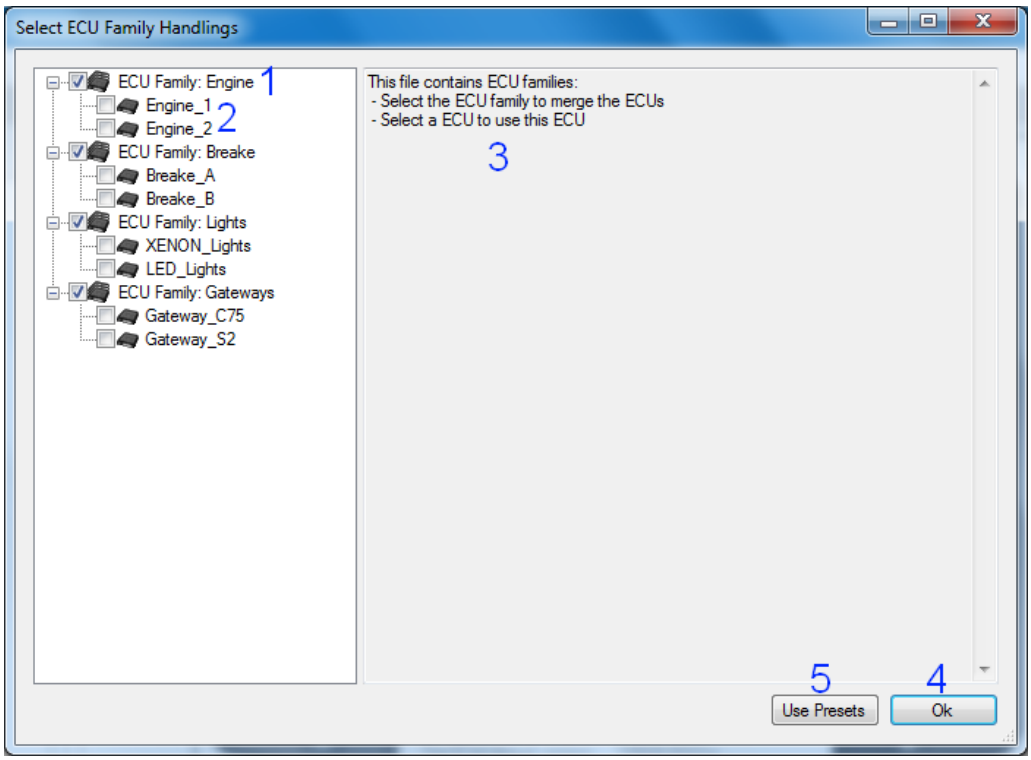


Figure 162: Selecting ECU-Family Handlings

No	Description
1	Name of the ECU family.
2	List of the ECUs, which belong to the ECU family
3	Information about the selected ECU / ECU family.
4	Click the "*Ok*" button to close the dialog and use the selected configuration. Clicking the [X] button has the same effect.
5	Click the "*Use Presets*" button to reset the settings to the predefined configuration. The dialog will not be closed.

5.18.1 ECU Family Example

For this example, the following configuration is assumed: The ECU family "FAM1" contains the ECUs "ECU1" and "ECU2". The ECUs have the following configuration:

ECU name	Slot Nr. (FlexRay)	Tx Frame
ECU1	1	FRA
	2	FRB
	4	FRX
ECU2	1	FRA
	3	FRC
	4	FRY

When the FIBEX file containing this ECU configuration is loaded, the dialog "Select ECU Family Handlings" (see Figure 162) will show a node "FAM1" with two child nodes "ECU1" and "ECU2". To continue, the user has three options:

Option 1: Select ECU family.

The ECU family is initially selected. If the user clicks the "*OK*" button, the configuration of both ECUs will be merged. In this case, the following configuration will be used for the RBS:

ECU to be simulated	Slot Nr. (FlexRay)	Tx Frame
ECU (merged)	1	FRA
	2	FRB
	3	FRC

No frame will be sent in slot 4, because both ECUs (ECU1 / ECU2) have a different frame configuration in this slot.

Option 2: Select ECU1.

ECU to be simulated	Slot Nr. (FlexRay)	Tx Frame
ECU1	1	FRA
	2	FRB
	4	FRX

Option 3: Select ECU2

ECU to be simulated	Slot Nr. (FlexRay)	Tx Frame
ECU2	1	FRA
	3	FRC
	4	FRY

5.19 Handling for PDU Update Bits

Case RBS: A PDU is **not** used as routing target in FL3X Config Gateway Editor.

Case Gateway: A PDU is used as routing target in FL3X Config Gateway Editor.

5.19.1 Handling for PDU Update Bits – Case RBS

The Update Bit of a PDU is initialized and sent with the value '1'.
Based on the cyclic timing of the PDU, the Update Bit is sent with value '1' (cyclic timing elapsed) or sent with value '0' (cyclic timing **not** elapsed).

5.19.2 Handling for PDU Update Bits – Case Gateway

The Update Bit of a target PDU is initialized with '0'.

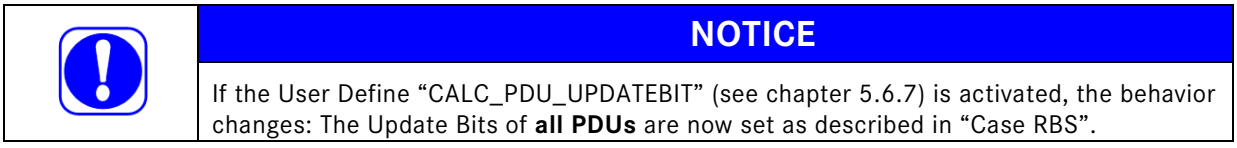
Each time a source PDU is received with its Update Bit set to '1', the Update Bit of the corresponding target PDU is also set to '1'.

After a target PDU was sent with Update Bit '1', its Update Bit is set to '0'.

If the transmission of a target PDU should be triggered via User Functions, the PDU Update Bit must be set manually.

Source PDU = A PDU that is configured as routing source in FL3X Config Gateway Editor.

Target PDU = A PDU that is configured as routing target in FL3X Config Gateway Editor.



5.19.3 Handling for Signal/SignalGroup Update Bits – Case Gateway

Update Bits of signals and signal groups are not considered by routings (FL3X Config Gateway).
Signal Update Bits are just shown in User Function Editor, where it is possible to change their values.

5.20 Handling for PDU-Container

PDU-Container have a dynamic layout, which consist of contained PDUs.

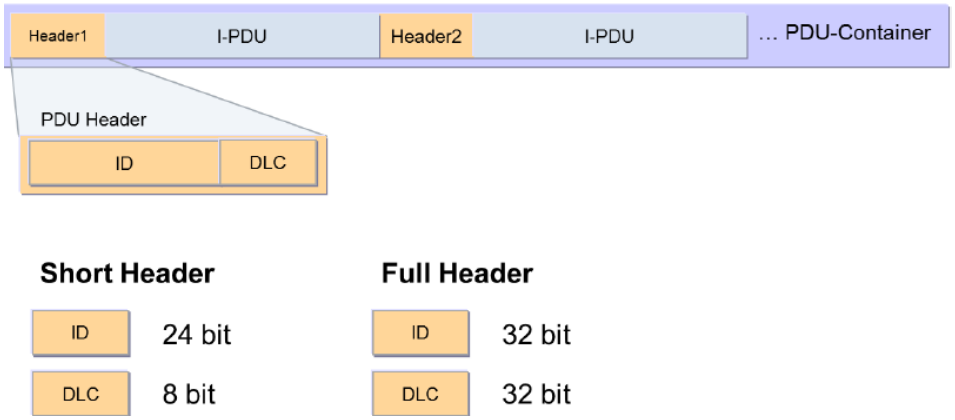


Figure 163: PDU-Container Layout [AUTOSAR Specification I-PDU Multiplexer]

A contained PDU has the property “collection semantics” which can have following values:

- Last is best: Just the latest payload of the contained PDU is added to the PDU-Container (there is a maximum of one PDU instance per PDU-container).
- Queued: Each new payload of the contained PDU is added to the PDU-Container (there could be several instances of a PDU within a PDU-container).

Both variants are supported for RBS and Gateway.

	NOTICE
	A container PDU with a fixed layout does not have any contained PDUs with header information. Such container PDUs are shown in FL3X Config with the icon  . A container PDU with a fixed layout behaves like a normal frame which has PDUs without update bits.

The following flowchart describes, how the PDU-Container are sent:

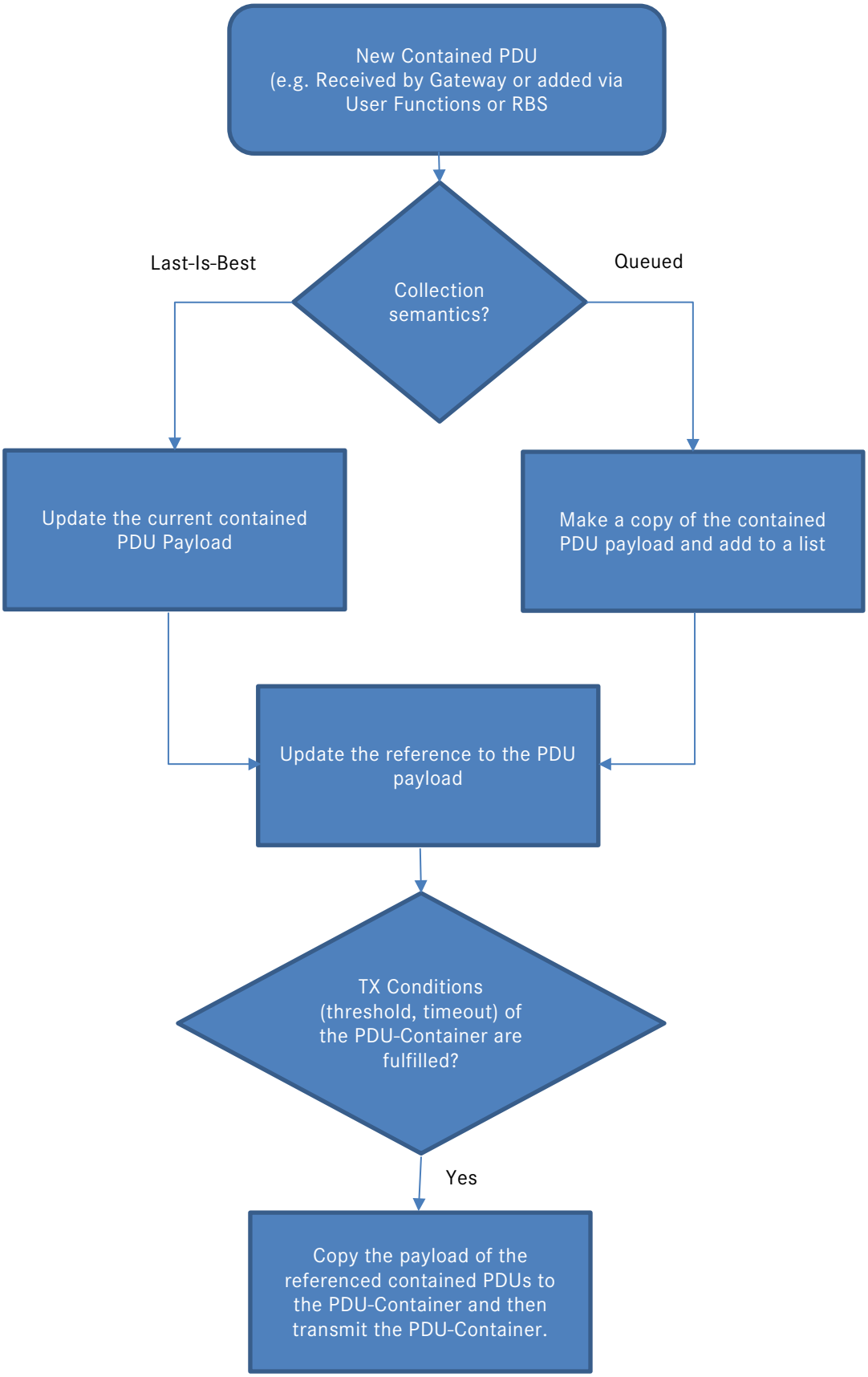


Figure 164: Flowchart for container PDU handling

The following issues must be considered for container/contained PDUs:

- User Function Editor:

- After a container PDU is received, the Rx-PDU function is called for each contained PDU. If a contained PDU is ‘queued’ and exists multiple times in the container, then the Rx-PDU function is called for each instance.
- The ‘Add to container’ function adds a copy of the actual payload of a ‘queued’ PDU to the container. If the contained PDU is “last is best”, then the ‘Add to container’ function tries to add the contained PDU to the container, if it does not already exist and update the payload if it is already added.
- Before a container PDU is sent, the Tx-PDU function is called for each contained PDU. If a contained PDU is ‘queued’ and exists multiple times in the container, then the Tx-PDU function is called for each instance.
- Normally Rx-Frame user functions are called before the Rx-PDU user functions. This is not the case for container PDUs. The Rx-PDU user functions of contained PDUs are called before the Rx-Frame user function.
- Normally Tx-Frame user functions are called after the Tx-PDU user functions. This is not the case for container PDUs. The Tx-PDU user functions of contained PDUs are called after the Tx-Frame user function.

- FL3X Config Gateway:
  - The routing behavior can be different depending on the side (source/target) of the container PDU and the transmissions settings. If the source side is a contained PDU and transmission behavior is ‘routing’, then receiving the contained PDU will trigger each time sending the target Frame. If a ‘queued’ contained PDU has multiple instances in the container, each instance triggers the target frame.
  - If a source side of a routing is a ‘last is best’ contained PDU and it is received multiple time times too fast, it may be routed just one time with the last payload.
  - In User Function Editor it is not possible to change the payload of the container PDU via the get/set functions for the signals of the contained PDUs, if the container PDU is used in a Byte-array based container PDU routing. Manipulating such signals is also not possible in FL3X Config Control. The “Add to Container” function is also not available for the contained PDUs of such container.
- Bus Configuration:
  - All instances of the contained PDUs are considered by checking the standard signal handlings, if “Calculation correct” is used as run level.
- FL3X Config Control:
  - The manipulation duration ‘event’ has just one time an effect on the transmission of the container,
  - After a Frame-Loss manipulation ends, it will be tried to send the pending contained PDUs.
  - The “event” term by the manipulation duration for a ‘queued’ contained PDU means adding the queued PDU to the container and not sending the contained PDU.
- Transmission Settings:
  - The transmission settings of the contained PDUs determines how the container PDU is filled and sent. To send container PDUs two conditions must be satisfied:

    1- The contained PDUs must be added to the container
    2- The container PDU must be triggered for sending.

    These actions can be done in user function editor via ‘add to container’ and ‘set Tx event’ functions. The sending behavior can be easily configured in transmission settings dialog. The property “Pdu Update Behavior” is responsible for adding the contained PDU to the container. The property “Frame Trigger Mode” is responsible for sending the frame/container.
    Sending the container can be cancelled, if it was triggered, but it contains no payload.

    Example 1:
    The contained PDU has a cyclic Timing of 100 ms. Both properties “Pdu Update Behavior” and “Frame Trigger Mode” are configured as ‘cyclic’.
    => The contained PDU will be sent each 100ms.

    Example 2:
    The contained PDU has a cyclic Timing of 100 ms. The property “Pdu Update Behavior” is configured as “Routing” and the property “Frame Trigger Mode” is configured as ‘Cyclic”.
    => The contained PDU may be sent each 100ms, but if no routing takes place, then the PDU will not be sent.

    Example 3:
    The contained PDU has a cyclic Timing of 100 ms. The property “Pdu Update Behavior” is configured as “Cyclic” and the property “Frame Trigger Mode” is configured as ‘Routing”.
    => In case of “queued” each 100ms the PDU is added to the container, but it will just be sent if a routing happens.

(e.g. if the routing happens each 400ms -> each 400ms the container is sent with 4 instances)
=> In case of 'last is best', each 100ms the contained PDU in the container is updated.
Info: if the routing happens each 10ms, the contained PDU still will be sent each 100ms. (No data, no tx)

Example 4:
The contained PDU has a cyclic Timing 100 ms. Both properties "Pdu Update Behavior" and "Frame Trigger Mode" are configured as 'event'.
=> The contained PDU is only sent if a routing takes place (Cyclic timing is not considered in this case).

5.21 Automotive Ethernet Sleep/Wake-up Management (TC10)

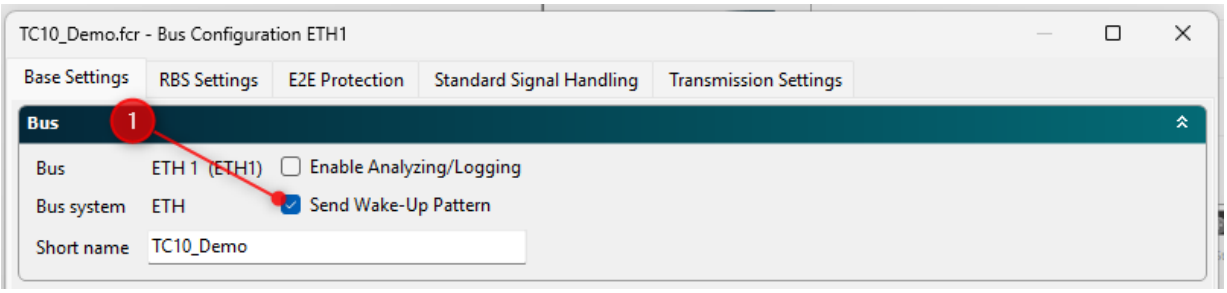
TC10 is an Automotive Ethernet mechanism specified by the OPEN Alliance that manages device power states. It defines how nodes enter low-power sleep mode and how they can be selectively woken up when needed. To use TC10 Sleep/Wake functionality, the hardware (transceivers) must support this feature.

Wake-Up Patterns (WUP) and Sleep Requests can be sent and received to wake network nodes or put them into sleep mode.

	NOTICE
	• TC10 features are currently only supported for 10BASE-T1S bus configurations • Sleep Requests are currently not supported (10BASE-T1S only supports Wake-up by specification)

5.21.1 Automatic Transmission of Wake-up Request

Wake-Up requests can be sent automatically when a configured Automotive Ethernet network link is established (BusOn). To enable this feature, the 'Send Wake-Up Pattern' checkbox must be selected.

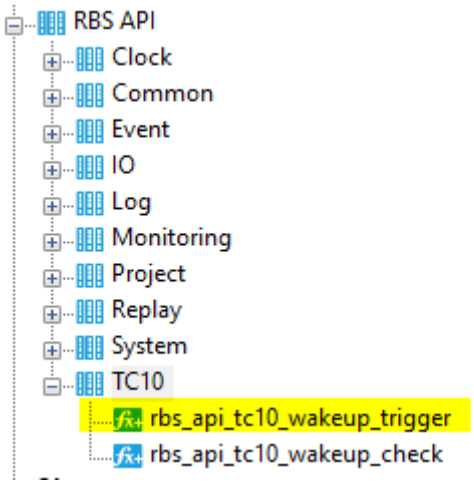


BusOn is triggered when the Automotive Ethernet link is established. The following events can influence this behavior:

- Device startup
- Calling "set_BUS_ON(...)" inside the "User Function Editor" after the network was disabled before (e.g. "set_BUS_OFF(...)")
- Bus state control via FL3X Config Control

5.21.2 Manual Transmission of Wake-Up Requests

It is also possible to manually transmit a TC10 Wake-Up request on the network. This can be done using the RBS API function **`rbs_api_tc10_wakeup_trigger()`** within the 'User Function Editor'. Once this function is called, a Wake-Up request will be transmitted. For more details, refer to the RBS API documentation.

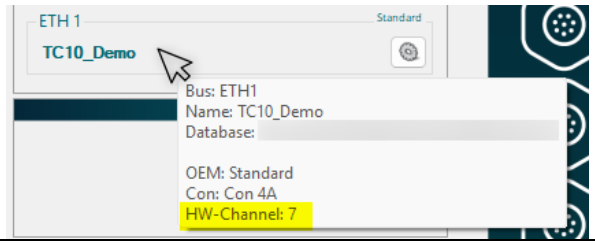


Source Code Editor 'UserMain_Function'

```
1  void UserMain_Function (void)
2  {
3    /**
4     * Trigger the transmission of a Wakeup Request according to TC10
5     *
6     * @since 1.14.0.0
7     * @param   channel_number      Number of hardware channel (1 - ...)
8     *
9     * @return  If return value is error the wakeup request will no be sent
10    *          RBS_API_ERROR_CODE_OK = Wakeup Request was sent
11    *          RBS_API_ERROR_CODE_BAD_PARAMETER = Wrong or unknown channel number
12    *          RBS_API_ERROR_CODE_NOT_SUPPORTED = Given channel or hardware does not support TC10 wakeup requests
13    */
14   rbs_api_tc10_wakeup_trigger(7 /* Channel ETH1 on CON4A */);
15 }
```

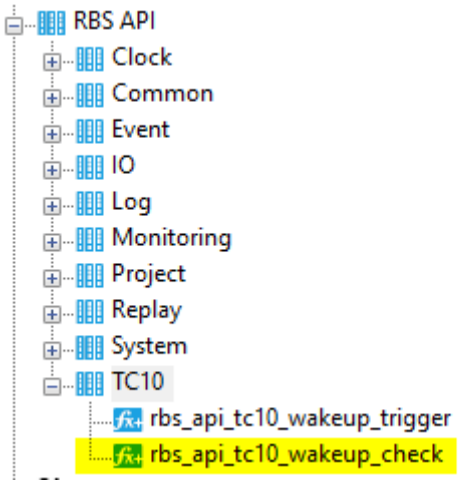
Information

The Hardware Channel number can be determined by looking at the Tooltip when hovering over the desired Bus configuration



5.21.3 Reception of Wake-Up Requests

The RBS API also provides a function to check whether a Wake-Up request has been received. This check is self-clearing and will report a given Wake-Up request only once, until a new request arrives. The transceiver hardware cannot distinguish between externally received and self-transmitted Wake-Up requests. As a result, the RBS API will also report Wake-Up requests that were triggered by the device itself. For additional details, refer to the RBS API documentation.



Source Code Editor 'UserMain_Function'

```
1  void UserMain_Function (void)
2  {
3     // Only check for Wake-Up Requests when ETH1 is Off
4     if(0 /* Off */ == get_BUS_State(ETH1))
5     {
6        if(RBS_API_ERROR_CODE_OK == rbs_api_tc10_wakeup_check(7 /* Channel ETH1 on CON4A */))
7        {
8           set_BUS_ON(ETH1); // Wake-Up Request received, turn ETH1 on
9        }
10    }
11 }
12
```

NOTICE
The function call used to check for a received Wake-Up request is time-consuming and may negatively impact runtime performance. Therefore, it should only be invoked at a moderate interval that meets the application's requirements, or in situations where detecting Wake-Up requests is critical (e.g. when the bus was previously turned off).

5.22 Settings

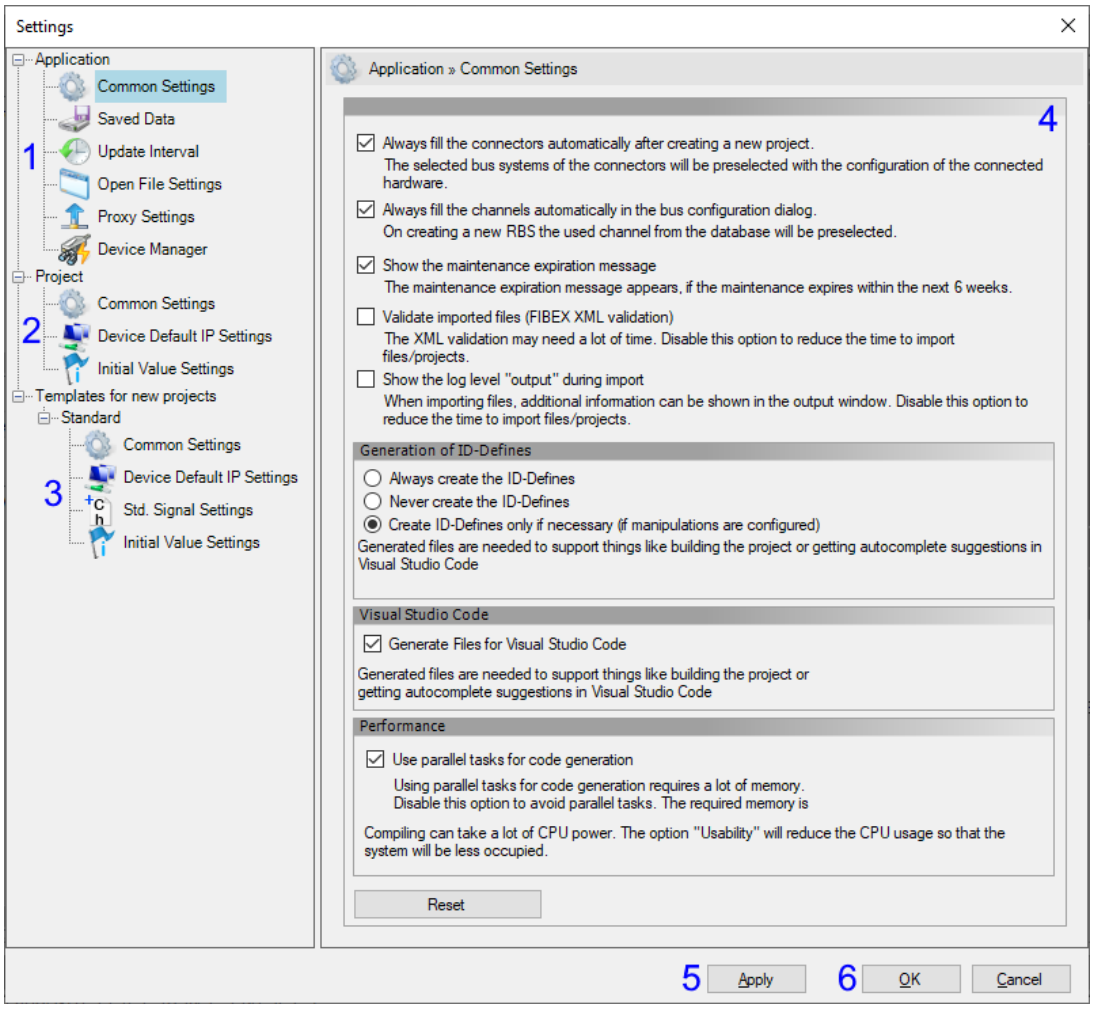


Figure 165: Settings Dialog

No	Description
1	Section for the application settings.
2	Section for the settings of the active project.
3	Section for template settings. Templates define the initial configuration for new projects.
4	This area contains the settings of the active section.
5	The “*Apply*” button saves the current changes without closing the dialog. This button will be enabled after each valid change.
6	The “*OK*” button saves current changes and closes the dialog. This button will be enabled after each valid change. The “*Cancel*” button closes the dialog without saving the changes

5.22.1 Application Settings

5.22.1.1 Common Settings

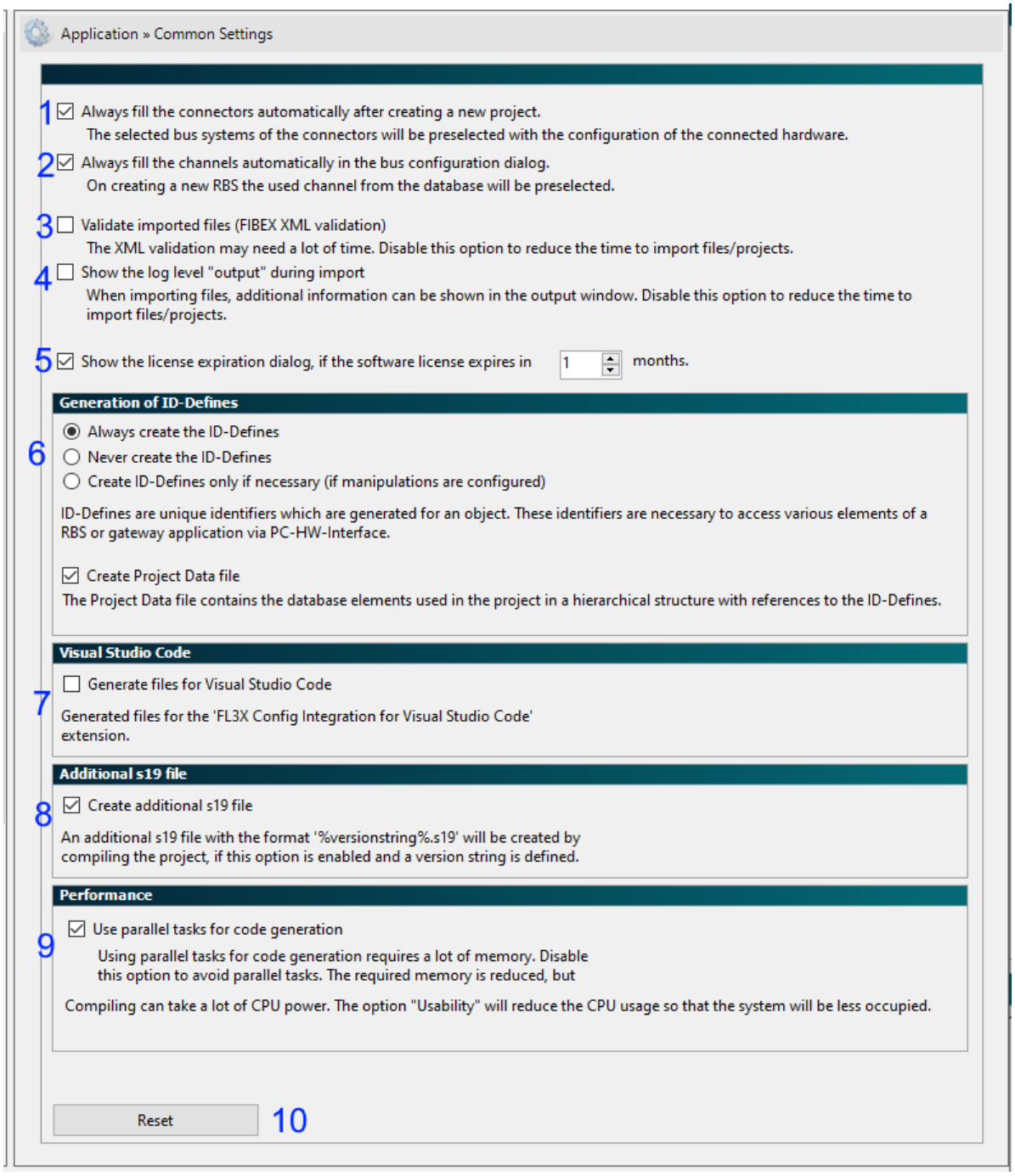


Figure 166: Application Settings – Common Settings

No	Description
1	If this setting is activated, all connectors will be automatically filled with the bus names after a project is created. The filled values depend on the busses, which are supported by the target device. A connected hardware device is required to fill out the connectors successfully.
2	If this setting is activated, the channel (drop down list) in the connector configuration dialog will be filled automatically after selecting a database and if the database has just one proper cluster for the selected connector. If the database has more proper clusters, then the channel will be left empty.
3	If this setting is activated, all loaded FIBEX files are validated. Errors are shown in the output section of the Main Window.
4	If this setting is activated, additional details will be shown in the output section while loading a database.
5	If this setting is activated, the software license will be checked each time on starting the application. If the software license expires within configured time, a dialog is shown with the expiring information and option to contact the sales team (s. chapter 4.5.1 for more information). ,
6	This option defines, whether the ID-Defines should be created. ID-Defines are used for manipulations (e.g. via FL3X Config Control and for RBS based Replay). Creating the ID-Defines may need a long time. - Always create ID-Defines: The ID-Defines are always created. - Never create ID-Defines: The ID-Defines are not created. This option can be used if you do not use FL3X Config Control, PC-HW-Interface [2] or RBS Based Replay. - Create ID-Defines only if necessary: ID-Defines are generated, if any manipulation is defined in the FL3X Config Control dialog or RBS Based Replay is activated (see chapter 5.12). The “Create Project Data file” option is enabled if the ID-Defines can be created. Activating this option creates a “Project Data file” by compiling the project. A Project Data file is a hierarchical xml file that contains the database elements used in the project and references to the ID-Defines. The ProjectData.xsd schema file can be found in the installation path of FL3X Config, which describes the xml structure.
7	If this setting is activated, additional files will be generated that are required for the Visual Studio Code Support. For more information see chapter 5.16.
8	When this setting is enabled, a copy is saved separately from the .s19 file that is created after a successful compile. - The text of "Version string" (shown in the main window) is used as the file name. - If the text "Version String" contains invalid characters for file names (e.g.: '?','/') they will be replaced with '_'. - If the text of "Version String" is '#BNr', after each successful compilation a new file will be created with the build number as the name.
9	If this setting is activated, parallel tasks are used for code generation (build project). Disable this option to reduce memory usage while increasing the ‘build project’ time.
10	The **[Reset]** button can be clicked to restore the default settings of the application.

Table 99: Application Settings – Common Settings

5.22.1.2 Saved Data

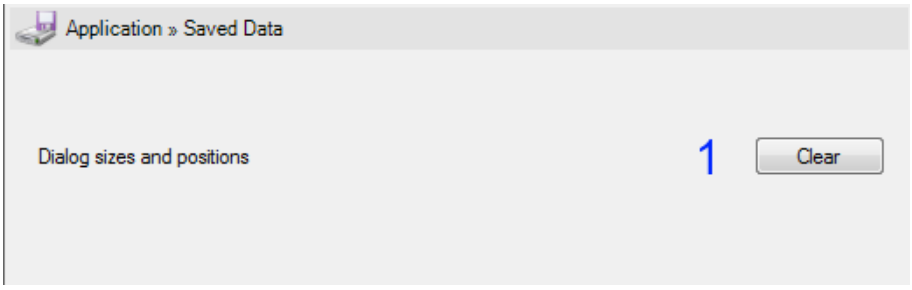


Figure 167: Application Settings – Saved Data

No	Description
1	Click on this button, to load the default GUI properties (size, location…).

Table 100: Application Settings – Saved Data

5.22.1.3 Update Interval

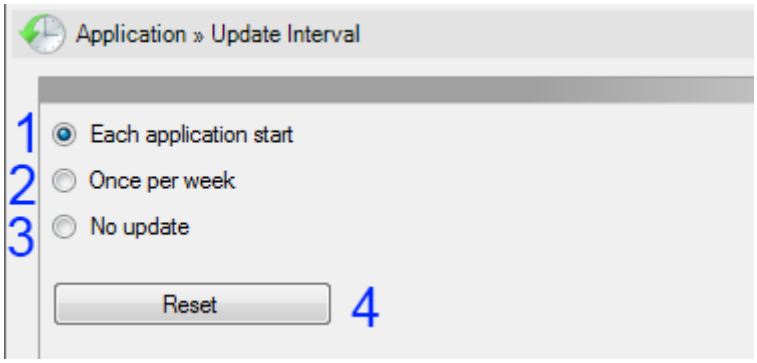


Figure 168: Application Settings – Update Interval

No	Description
1	The process for searching the license and application updates will be started each time, after the application is launched.
2	The process for searching the license and application updates will be started only once a week.
3	The process for searching the license and application updates will never be started.
4	Reset to default.

Table 101: Application Settings – Update Interval

Information
The process for searching the license and application updates can be manually started over the menu entry “*Help*” -> “*Check For Updates*”.

5.22.1.4 Open File Settings

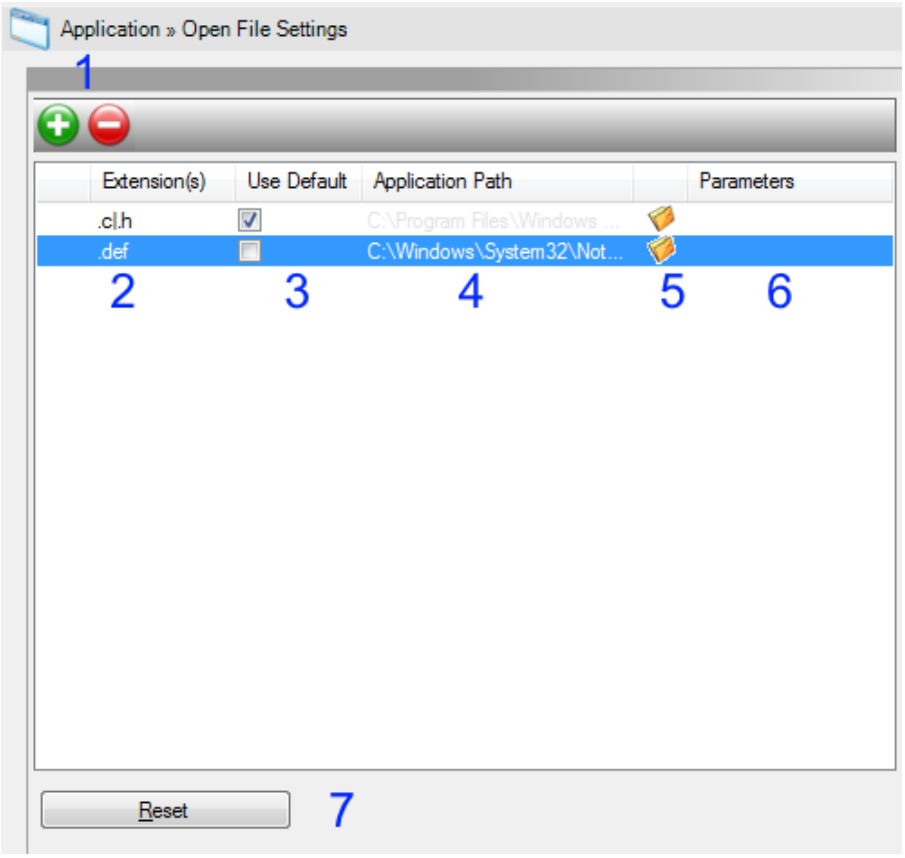


Figure 169: Application Settings – Open File Settings

The “Open File settings” dialog can be used to open files (*.c*, *.h*) with a selected text editor. These *.c*, *.h* files are the external files, which have been added within the “User Function Editor”.

No	Description
1	The “+”, “-” buttons add/remove a new setting.
2	In the “Extension(s)” column the extensions of the files, which are opened can be edited. If more than one file-extension is used, each extension must be separated by “|” e.g. (.c|.h).
3	The “Use Default” column contains a checkbox for each line. If the checkbox is activated, all appropriate files are opened with the OS (Operating System) specific application. If the checkbox is not activated the application is chosen, which has been selected. (s. 4, 5, 6).
4	The “Application Path” specifies the folder and application (editor), which will be used to open the added, external files in the “User Function Editor”. The path can be edited directly within this column or added with the browse button (s. 5).
5	The browse button can be pressed to select the appropriate folder and application (editor) within a file-dialog box.
6	The chosen application can be called with an application-specific set of parameters. These parameters can be provided in this column. For example, the “-n10” parameter of the Notepad++ Editor will induce that the editor will jump to the specified line 10. “FL3X Config” provides two control sequences which can be used to specify a parameter set. These are: *%FileName,* this control sequence provides the appropriate, context-specific filename including the absolute path in quotation marks. *%Line,* this control sequence provides the appropriate, context-specific line number. Example: An error occurs within an external file (*C:\Example\my_file.c*). The error including the line and filename is displayed in the “Output” section of the “User Function Editor”. The selected application is the notepad++ editor and the “Parameters” column contains the expression *%FileName -n%Line.* If the user double-clicks on the displayed error line, notepad++ opens the relevant file and jumps to the appropriate line. If an application is selected and no parameters are provided, the control sequence *%FileName* is used implicitly. So, if an external file is opened in the “User Function Editor”, the file will be opened within the relevant editor, starting the first line.
7	If the ‘*Reset*’ button is clicked, the original/default settings will be provided again.

Table 102: Application Settings – Open File Settings

	Information
	FL3X Config detects if Notepad++ is installed. As long the ‘Open File Settings’ are not changed, FL3X Config will change the ‘Open File Settings’ automatically to use Notepad++ as the default editor.

5.22.1.5 Proxy Settings

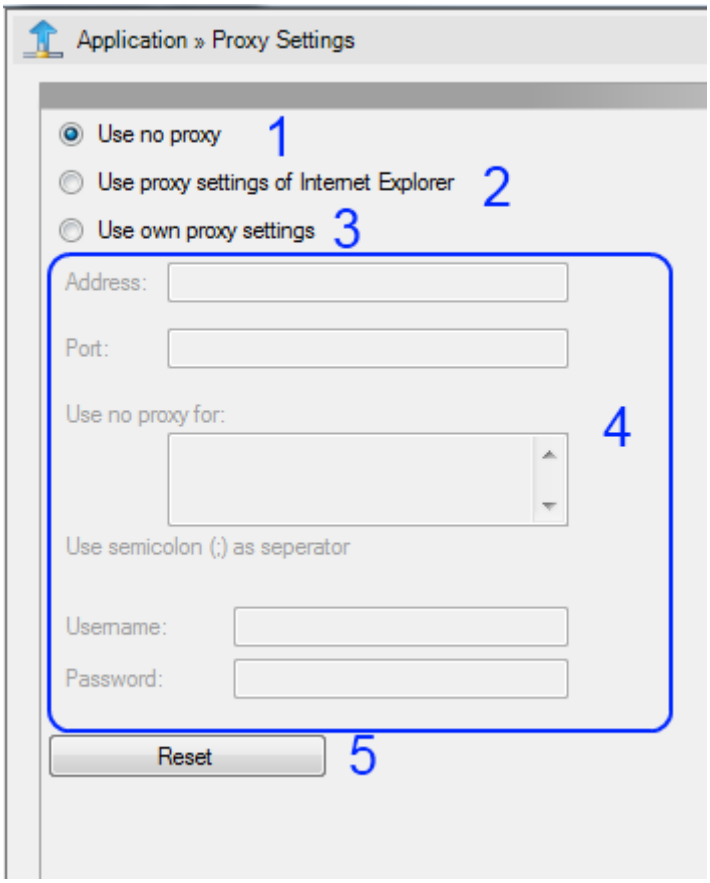


Figure 170: Application Settings – Proxy Settings

Proxy settings should be configured to permit the connection to the license and application update servers.

No	Description
1	Use this setting if no proxy is needed for a connection to an external URL.
2	Use this setting if the proxy configuration should be taken from the Internet Explorer.
3	Use this setting and configure the proxy settings manually.
4	Fields for the setting the proxy configuration. They will be enabled if the option 3 is selected.
5	The default configuration will be selected if you click on this button.

Table 103: Application Settings – Proxy Settings

5.22.1.6 Device Manager

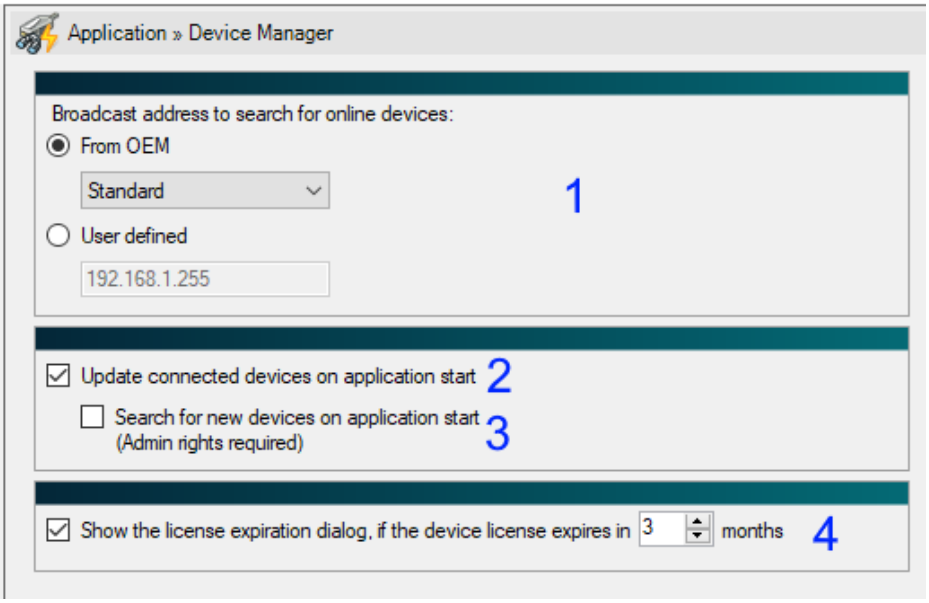


Figure 171: Application Settings – Device Manager

No	Description
1	This setting specifies the broadcast IP-Address which is used to search for devices in the Device Manager. There are two options available. The first option (“From OEM”) uses the default broadcast IP-Address of a specific OEM. The second option (“User defined”) allows to enter the broadcast IP-Address manually.
2	Activate this setting to reconnect each device in the device list when the application is started.
3	Activate this setting to search for new devices when the application is started. A search requires that the application is started as local administrator.
4	Activate this setting if the ‘device expiring dialog’ should be shown automatically (see also chapter5.2.6.1). If this setting is activated: • The runtime license of each device will be checked on application start. The setting ‘Update connected devices on application start’ must be activated too. • The runtime license of each device will be checked if any device is changed, or a new device is added.

5.22.2 Project Settings

NOTICE
The entries for the project settings will not be shown if there is no active project!

5.22.2.1 Common Settings

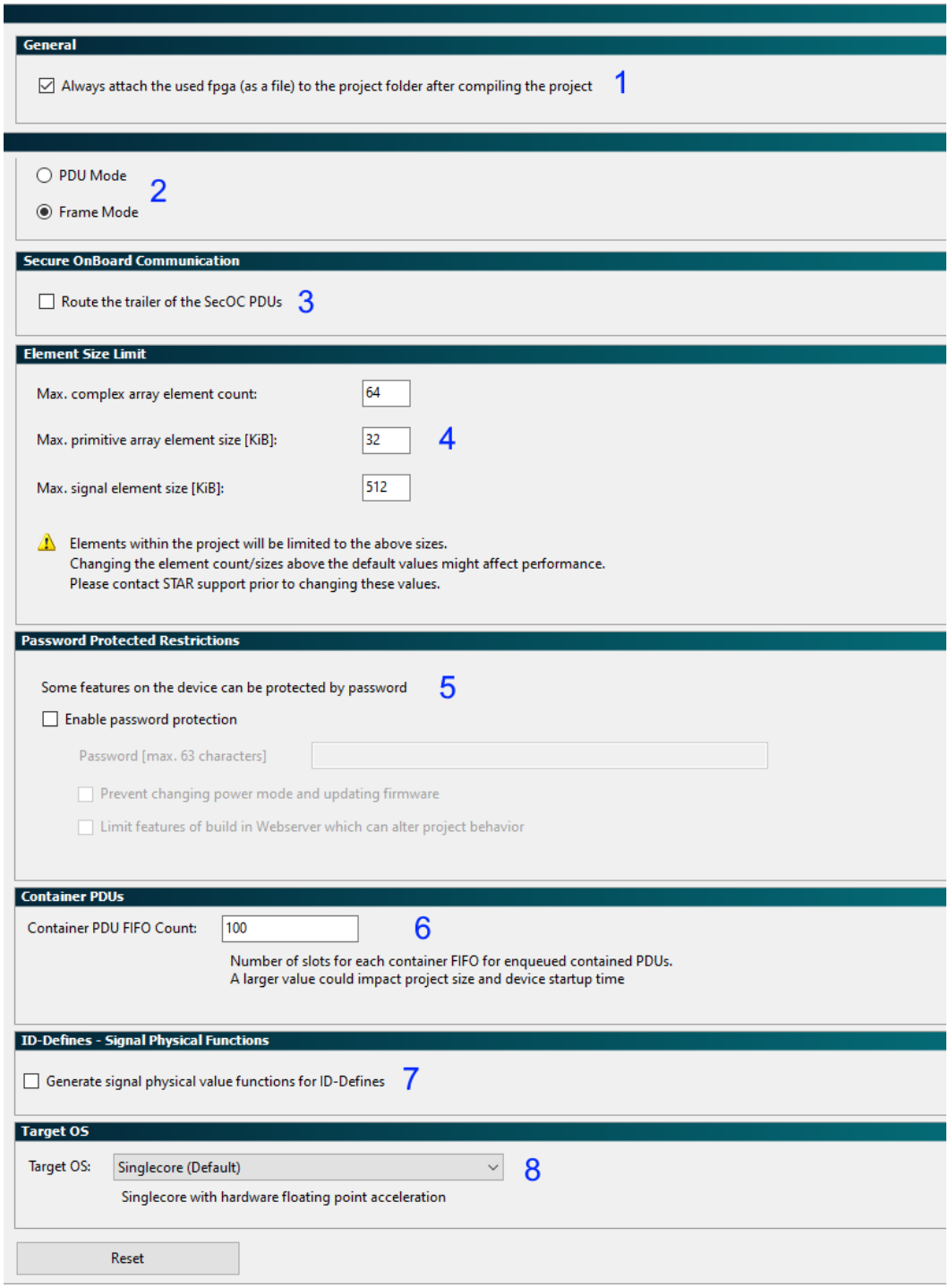


Figure 172: Project Settings – Common Settings

No	Description
1	If this option is activated, the used FPGA image is also saved to the project folder, each time the project is built. The FPGA image file is the same as if the image was exported through '*Extras -> Export FPGA-Images*' (see chapter 5.4.5.2).
2	The view mode for the FL3X Config Gateway editor for the current project. This setting affects the items which are displayed and the method for the creation of routings. See chapter 6.1.4 The view mode also affects the displayed items by 'FL3X Config Control' dialog.
3	This option enables routing of the SecOC trailer from the source PDU to the target PDU. See chapter 5.10 for more information.
4	Large arrays and signals may cause performance problems. Maximum lengths are used for such elements which can be configured in this section. These options can be also changed by template settings. The following sample element is used to explain these settings: Event1 -DataType1 [100] -DataType2(int) -DataType3(int) -DataType4(int) [1000] The "DataType1" is a complex data type (struct) and is a dynamic array with a max length of 100. The "DataType4" is a primitive array (in this case integer) with a max length of 1000. "Max. complex array element count" specifies the number of complex array elements like "Datatype1", that can be internally processed by the device. The total size of the array is still limited by the following setting "Max signal element size". The "Max. primitive array element size" affects the site of primitive arrays like "DataType4". The "Max signal element size" setting considers the total size of the element "Event1".This value does not specify the length that can be sent, but merely the maximum size that can be processed internally. The value of this setting has priority. This means that the dynamic arrays will be sent with a length of "0" if another valid setting results in a total size that exceeds this value. Please also note that for AUTOSAR PDUs, there is a further internal limitation of 32KiB maximum size which takes precedence.
5	This section allows you to set a password for the project, which restricts some operations with the device and the FL3X Diagnostics Website (see chapter 8). To unlock password protection or enable this feature at runtime, the FL3X Diagnostics Website can be used. Be aware that: - The password is stored unencrypted in the source code and memory. - The password is transmitted unencrypted when entered. The option "prevent changing power mode..." prevents the device to be restarted via PC-HW-Interface and avoids flashing a new software. The option "Limit feature..." avoids accessing following pages of the FL3X Diagnostics Website: - Disk - Ethernet - SOME/IP - VSS - SoK - SecOC - IPsec - Transport Protocols - Wifi - Logging - Threads
6	The default value for the internal FIFOs used for Container PDUs. Before you change this value, please contact the *STAR ELECTRONICS* support team.

No	Description
7	If this option is activated, the get/get physical value functions will be generated and included in the ID-Defines.
8	The target OS specifies for which operating system the project is build. Different options, based on the device type (FL3X Device or FL3X System), are available. See also chapter 5.22.4.

Table 104: Project Settings – Common Settings

5.22.2.2 Device Default IP Settings

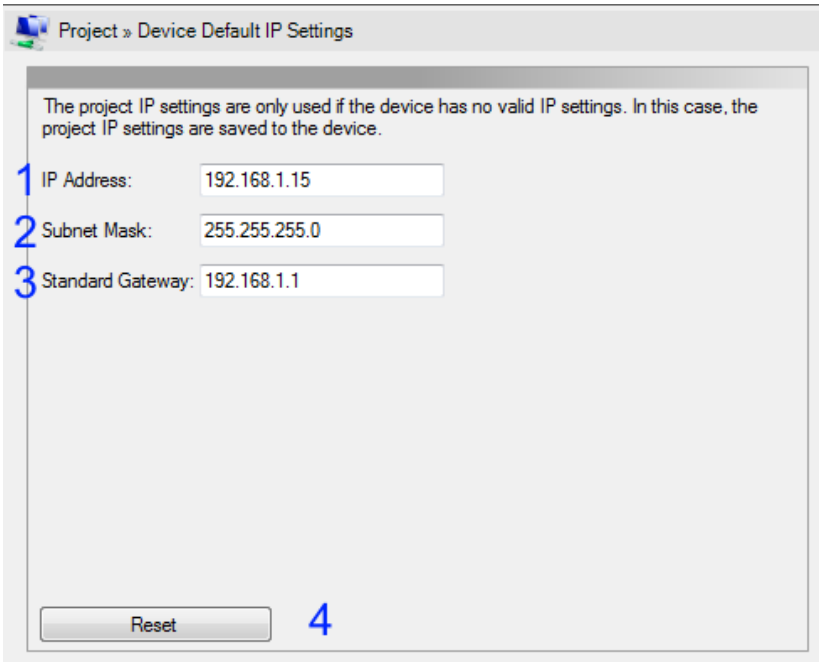


Figure 173: Project Settings – Device Default IP-Settings

No	Description
1	Default IP-Address of the device for the current project. The Default IP-Address is used for devices where the IP-Address is unknown to the application (e.g. a project has been flashed with an earlier version of FL3X Config).
2	Subnet mask of the Default IP Address.
3	Standard gateway address of the current project.
4	Sets the default Fall-back IP settings of the project.

Table 105: Project Settings – Device Default IP-Settings

5.22.3 Templates for New Projects

When a new project is created, its initial settings are taken from the templates, which are defined in this section. The settings of a current project can be changed any time and can be different from the template settings.

5.22.3.1 Common Settings

The common settings used for new projects. See Table 104 for more information about these settings.

5.22.3.2 Device Default IP Settings

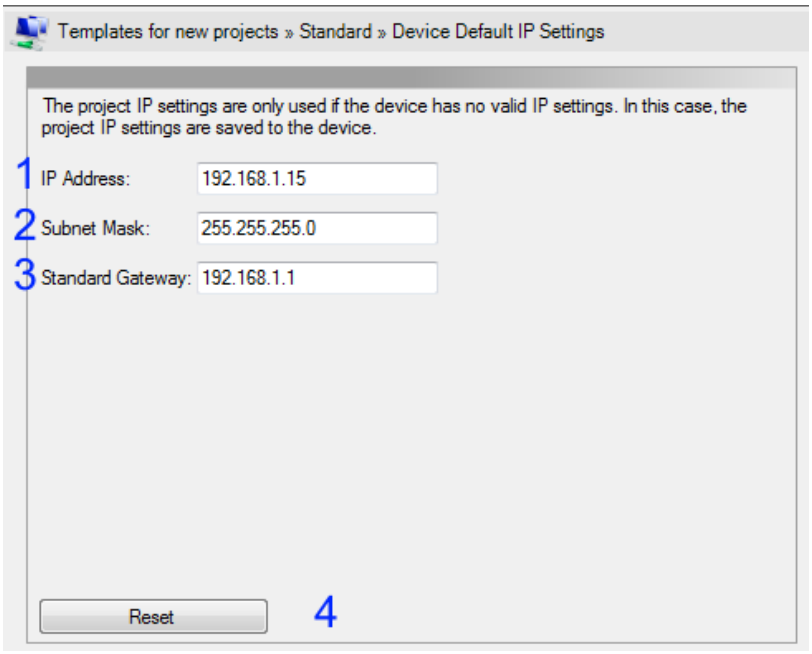


Figure 174: Template Settings – Device Default IP-Settings

No	Description
1	Default IP-Address of the device for the new project.
2	Subnet mask of Fall-back IP Address for the new project.
3	Standard gateway address of the new project.
4	Sets the default IP settings of the template.

Table 106: Template Settings – Device Default IP-Settings

5.22.3.3 Standard Signal Settings

You can set the default source and header files and the functions, which can be used for the standard signal calculation. Each time an RBS or transparent gateway is created, the default settings will be used automatically.

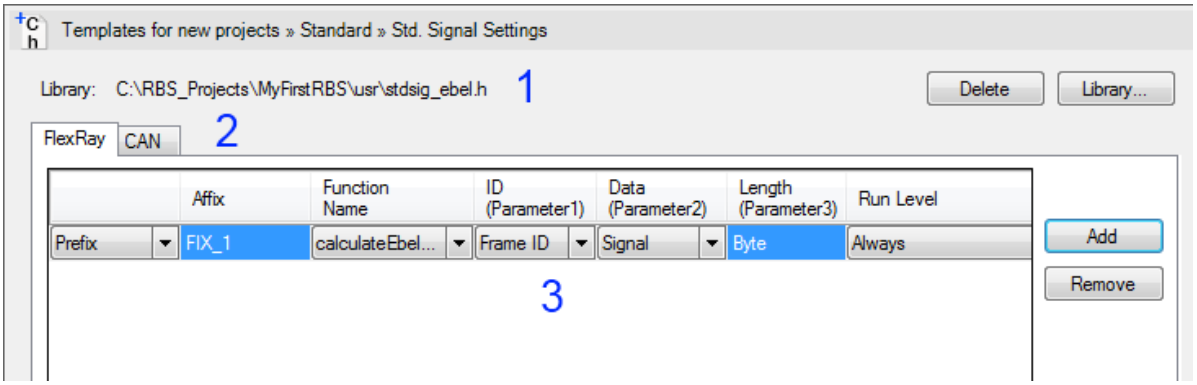


Figure 175: Template Settings – Standard Signal Handling

No	Description
1	Here you can see the selected user library. The functions of this library will be available for the standard signal calculation.
2	Here you can select the bus, for which the configuration should be applied
3	In this area you can specify the settings for the appropriate standard signals. This can be e.g. the affix, function name, and so on. A detailed description for the setting of these parameters is given in Table 53.

Table 107: Template Settings – Standard Signal Handling

5.22.3.4 Initial Value Settings

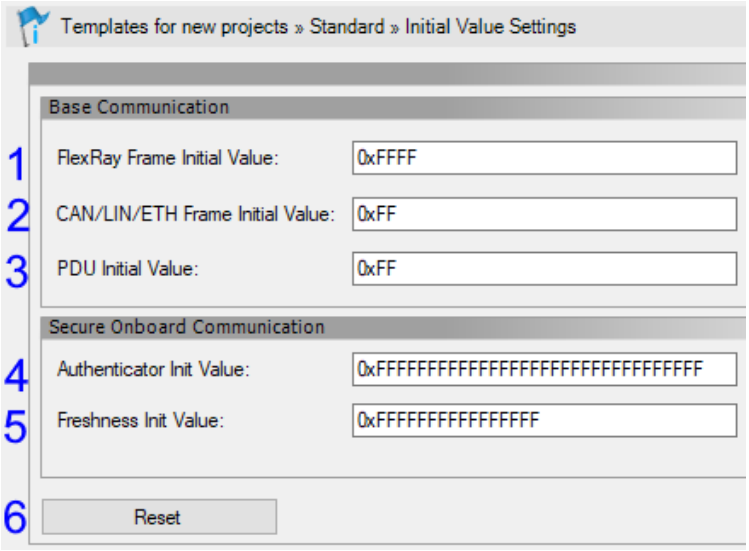


Figure 176: Template Settings – Initial Value Settings

No	Description
1	Initial raw payload of FlexRay frames.
2	Initial raw payload of CAN, LIN, Ethernet frames.
3	Initial raw payload of PDUs.
4	Initial value of the secure onboard authenticator value. Only used for Secure Onboard Communication.
5	Initial value of the secure onboard freshness value. Only used for Secure Onboard Communication.
6	Sets the default initial values.

Table 108: Template Settings – Initial Value Settings

	NOTICE
	These settings only apply to bus configurations that are added afterwards. The default values of existing bus configurations must be changed manually.

	Information
	The “Default Values Editor” can also be sued to change the initial values of a bus (see chapter 5.9.2).

5.22.4 Target OS

The target OS specifies for which operating system the project is build.

The target OS can be used/configured in

1. Project Settings (see chapter 5.22.2.1)
   To specify for which target OS the project is build.
2. Project Template (see chapter 5.22.3.1)
   To specify the default settings for new projects.
3. Build Menu (see chapter 5.4.3.3)
   To build the project for a specific target OS.

Different Target OS are available for each device type (FL3X Device or FL3X System):

- Default
  Build the project for the device type.
  Available for devices which does not support different target OS (e.g. FL3X System)

- Singlecore (Default)
  Build the project for the device type with singlecore support.
  This option is default for the most devices which supports singlecore and multicore builds.
- Multicore
  Build the project for the device type and enables the multicore feature of the CPU.
- Virtual FL3X Device for Windows
  This option is available for the most device types.
  Build the project as Virtual FL3X Device for Windows will result in a *.dll* with the RBS/Gateway functionality which can be executed on Microsoft Windows.
  See chapter 5.22.4.1
- Virtual FL3X Device for ADTF
  Build the project as a virtual device which can be used as ADTF plugin.
  This option is available for the most device types.
  See chapter 5.22.4.1.

5.22.4.1 Virtual FL3X Device for Windows

The Virtual FL3X Device for Windows is a compile option available for the most device types (see chapter 5.4.3.3 Build Project for Target OS and chapter 5.22.4 Target OS).

Build the project as a virtual device which can be executed on Microsoft Windows.
When the project is build, the following files will be saved in the folder *%PROJECTPATH%\out\Virtual_FL3X_Device_Windows*:

- *Virtual_FL3X_Device.dll*: DLL which contains the project logic and implements the RBS API.
- *Virtual_FL3X_Device.lib*: Import library for the *Virtual_FL3X_Device.dll*.
- *Virtual_FL3X_Device_CLI.exe*: CLI for Testing and example functionality. The user can input CAN/CAN-FD frames as an asc file into the RBS through CLI. The RBS is also able to generate a similar asc file as output.
  For more information, run the executable with the parameter '-h':
  `Virtual_FL3X_Device_CLI.exe -h`
- *Project.dat*
- *Build.info*

	Reference
	More information about the Virtual Device Interface can be found in the document: RBS API Manual [11]

	NOTICE
	Each instance of a Virtual FL3X Device for Windows requires a runtime license. Please contact STAR ELECTRONICS support team for more information.

	NOTICE
	Not all features are supported to build for Virtual FL3X Device for Windows. This may lead in compilation errors. The following features are **not** supported for Virtual FL3X Device for Windows: - FlexRay - SOME/IP Configuration - LIN - SENT - Time Synchronization

5.22.4.2 Virtual FL3X Device for ADTF

Build the Project as the Virtual FL3X Device for ADTF which will result in a Windows DLL with the Virtual Device Interface [12]. This DLL contains the project logic and can be used for simulation without using real hardware devices.

Following files will be saved in the folder *%PROJECTPATH%\out\Virtual_FL3X_Device_ADTF* after compiling a project:

- *FC_RBS.dll*: DLL which contains the project logic and the implementation of the RBS API.
- *FC_RBS_CLI.exe*: CLI tool to process input log files with FL3X Config configuration logic and write resulting data into output log file.
  Output will be written to *<InputTrace>.out.asc*
  For more information, run the executable without any parameter:
  FC_RBS_CLI.exe
- *FC_RBS_GAF.description*
- *FC_RBS_GAF.dll*
- *FC_RBS_GAF_Properties.xml*
- *FC_RBS_WIBU.dll*
- *Project.dat*
- *Build.info*

	Reference
	More information about the Virtual Device Interface can be found in the document: FL3X Config Virtual Device Interface [12]

	NOTICE
	Each instance of a Virtual FL3X Device for ADTF requires a runtime license. Please contact STAR ELECTRONICS support team for more information. If no virtual runtime license is available, the virtual device runs in demo mode for 30 seconds.

	NOTICE
	Not all features are supported to build for Virtual FL3X Device for ADTF. This may lead in compilation errors. The following features are not supported for Virtual FL3X Device for ADTF: - SOME/IP Configuration - LIN - SENT - Time Synchronization

5.23 Multi OEM Support

FL3X Config supports bus configurations with different OEM specific settings. E.g., a project can contain multiple busses. And each bus has a different OEM-specific setting.

The prerequisites are:

- The project must be created with the project type "Standard".
- At least one OEM package is installed.

If these conditions are fulfilled, then the database selection dialog will show a dropdown list to select the OEM settings for this bus.

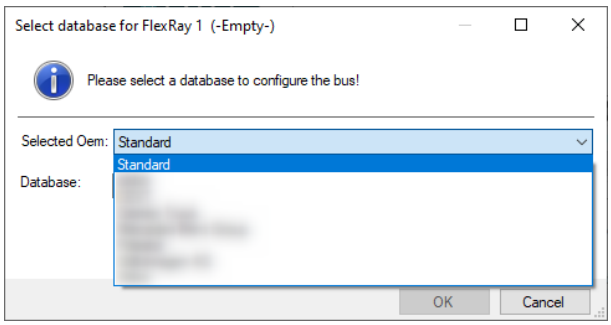


Figure 177: Multi OEM Support – Select OEM for Bus

A similar dropdown list is also shown in the Transparent Gateway dialog:

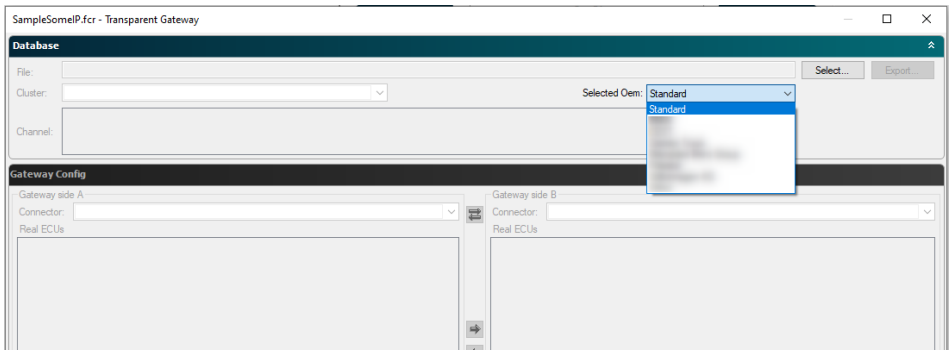


Figure 178: Multi OEM Support – Select OEM for Transparent Gateway

Feature	Behavior
View Mode	View Mode (Frame/PDU) is inherited from the bus OEM. This is just relevant for creating a Transparent Gateway. In other cases, the view mode of the project is used.
E2E	The E2E configuration of the selected OEM is automatically used for the corresponding bus.
Frame/PDU Initial Values	Frame and PDU initial values are OEM specific. Each configured bus takes the values from the selected OEM. These values can be edited in 'Default Values' dialog.
Secure Onboard	Secure Onboard (SecOC) settings are inherited from the selected OEM, if there are OEM-specific SecOC settings.
Standard Signal Handling	It is not possible to configure standard signal handlings with different files per bus. The settings are inherited from the 'Standard' OEM.
FL3X Config Control	OEM specific manipulation handlings are inherited from the selected OEM for each bus.

Table 109: FL3X Config Gateway – Description of the sections

5.24 OEM Specific Content

Please read the additional information provided for each OEM package.

6 FL3X Config Gateway

FL3X Config Gateway is part of FL3X Config. It is used to generate firmware code for the devices FL3X Device-S or FL3X Device-L/L² out of FIBEX files. The generated code can be used for gateway applications for FlexRay, CAN, LIN or Ethernet.

6.1 Gateway

6.1.1 General

Open the FL3X Config Gateway dialog with *'Edit->FL3X Config Gateway...'* to configure the gateway.

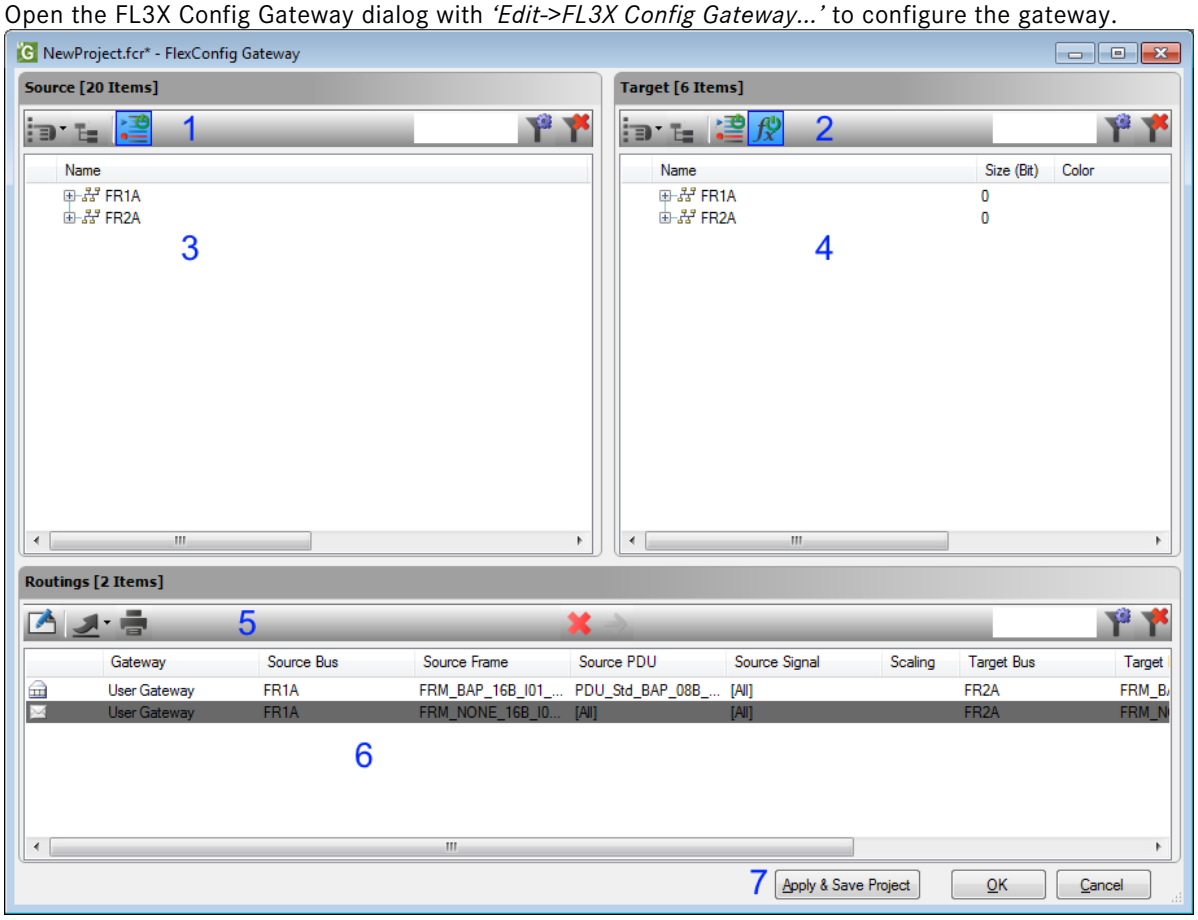


Figure 179: "FL3X Config Gateway" editor

No	Description
1	Toolbar for the "Source" section (see 6.1.1.1 Source Toolbar)
2	Toolbar for the "Target" section (see 6.1.1.2 Target Toolbar)
3	The "Source" section which contains the entire Network data of the configured bus channels.
4	The "Target" section, which contains just the configured frames to be sent.
5	Toolbar for the "Gateway Routings" section (see 6.1.1.3 Gateway Routings Toolbar).
6	The "Gateway Routings" section shows the configured routings.
7	The "*Apply & Save Project*" button saves all changes made with the gateway editor.

Table 110: FL3X Config Gateway – Description of the sections

The FL3X Config Gateway will load the databases automatically and you will see all configurations made for the bus channels in the "Source" and "Target" section. Now you are able to define a routing from "Source" section to the "Target" section per drag & drop, with the route button or via the context menu.
If there is no routing for a signal, the manipulation behavior of this signal is like signal manipulation for RBS (see chapter 6.2.6). Please check chapter 6.1.12.1 for the limitations for the signal routings.

	Information
	After selecting any item (except Bus and ECU) on the source or target section, all matching items, which are suitable for routings will be shown on the other side with little flags. If a signal is selected, then the green flag on the other section means that the signal layout (also considering the Compu-Methods) is the same and the short name of the signal also match (The short name of one element contains the other one or they are the same). An orange flags means that the signal can be mapped, but it has a different layout, or the short name does not match. If a PDU or a frame is selected, then the green flag on the other side means that either the OIDs are same, or the short names of the elements are matching.

	Information
	Showing suitable items with flags and jumping to the matching items will not work if the items are not expanded.

	Information
	For the items shown in the FL3X Config Gateway editor there are tooltips providing detailed information. For example, all frames of a multiple instantiated PDU are shown within the tooltip.

6.1.1.1 Source Toolbar



Figure 180: FL3X Config Gateway – Toolbar for the “Source” section

No	Description
1	View button: Over the view button you can choose, which source cluster elements are shown
2	Expand/Collapse button expands or collapses all source elements
3	Jump to target button: If this button is activated, then the first matching target element will be automatically selected in the target list, after choosing a source element. There is no matching for Busses and ECUs.
4	Text-Filter field: Only elements containing the entered text are shown in the source list. See no. 5.
5	Source button is used to select the columns which are considered by the text-filter. Selecting less columns will improve the performance
6	Clear button resets the text-filter

Table 111: FL3X Config Gateway – Toolbar of the source section

	Information
	If the “*Jump to source*” button or the “*Jump to target*” button is activated, and more than one matching item is found, then you can use the “F2” key to jump to the next matching item.

	Information
	Showing suitable items with flags and jumping on the matching items will not work if the items are not expanded.

6.1.1.2 Target Toolbar



Figure 181: FL3X Config Gateway – Toolbar for the “Target” section

No	Description
1	View button: Over the view button you can choose, which target cluster elements are to be shown
2	Expand/Collapse button expands or collapses the whole target elements
3	Jump to source button: If this button is activated, then the first matching source element will be automatically selected in the source list, after choosing a target element. There is no matching for Busses and ECUs
4	Scaling button: If this button is activated, then the scaling dialog will be shown after routing signals, which must be scaled and have supported Compu-Methods.
5	Text-Filter field: Only elements containing the entered text are shown in the target list. See no. 6.
6	Source button is used to select the columns, which are considered by text-search. Selecting less columns will improve the performance
7	Clear button resets the text-filter

Table 112: FL3X Config Gateway – Toolbar of the “Target” section

Information
If the “*Jump to source*” button or the “*Jump to target*” button is activated, and more than one matching item is found, you can use the “F2” key to jump to the next matching item.

Information
Showing suitable items with flags and jumping on the matching items will not work, if the items are not expanded.

6.1.1.3 Gateway Routings Toolbar



Figure 182: FL3X Config Gateway – Toolbar for the “Gateway routings” section

No	Description
1	“Edit Routing” button opens the scaling dialog if a signal routing is selected. If a frame or a PDU routing is selected, then a dialog will be opened, in which the selected routing can be spitted to its child elements.
2	“Export” button is used to export the actual content of any section as a *.csv* or *.xlsx* file.
3	“Print” button is used to print the actual content of any section.
4	“Delete” button removes the selected routing(s).
5	“Route” button is used to create a routing for the selected source and target element.
6	“Create child routings” button creates routing(s) for the lower level items of the selected source and target elements. This button is activated just when frames or PDUs are selected on source and target sections, which have the same layout.
7	Text-Filter field: Only elements containing the entered text are shown in the target list. See no. 8.
8	Source button is used to select the columns, which are considered by the text-search. Selecting less columns will improve the performance
9	Clear button resets the text-filter

Table 113: FL3X Config Gateway – Toolbar of the “Gateway routings” section

6.1.2 Routings

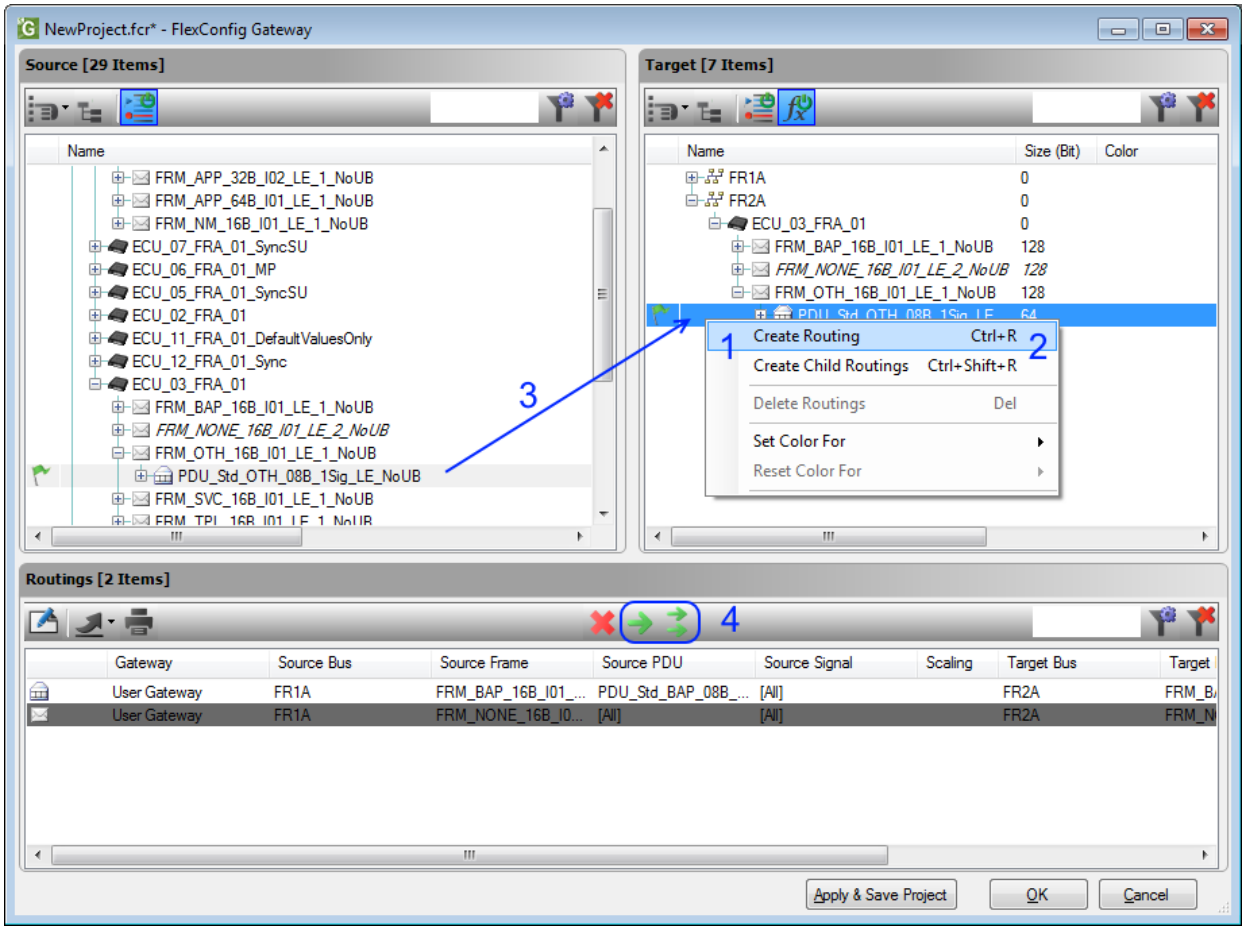


Figure 183: FL3X Config Gateway – Create a routing

You have several options for creating routings. Firstly, one source and one suitable target item must be selected. After the selection, you can

- Create a routing over route buttons (**4**). If the button with multiple arrows is activated, you can create routings for the child items of the selected elements.
- Create a routing per drag & drop (**3**)
- Create a routing over the context menu (**1**) by the source or the target section
- Use shortcuts (**2**). "*Ctrl+R*" is used for creating a routing

The following routing types are principally possible.

Routing type	Information	View
Byte-array based frame routing	A frame can be routed to another frame, without considering the layouts. The whole payload of the source frame will be copied to the target frame. The source frame cannot be longer than the target frame. Except for FlexRay static segment, the length of the target frame will always be adapted to the length of the source Frame (e.g. if a CAN-Frame is received with length of 4Bytes, the target frame will also be sent with 4 Bytes).	Frame View
PDU based frame routing	A frame can be routed to another frame, if the layouts are same. The payload of each source PDU will be copied to the target PDU. If a target FlexRay frame is in the dynamic segment, the length will always be adapted to the length of the source Frame.	Frame View

Routing type	Information	View
Byte-array based container PDU routing	A container PDU can be routed to another container PDU. The whole payload of the source container PDU will be copied to the target container PDU. In user function editor it is not possible to use the get/set functions for the signals of the contained PDUs, if the container PDU is used in a Byte-array based container PDU routing. Manipulating such signals is also not possible in FL3X Config Control. The “Add to Container” function is also not available for the contained PDUs of such container.	Frame View PDU View
PDU routing	The payload of the source PDU will be copied to the target PDU. The PDUs must have the same length.	Frame View PDU View
Signal routing	The payload of the source signal will be copied to the target signal. Please check chapter 6.1.12.1 for the limitations for the signal routings	Frame View PDU View
String routing	The string will be routed and, if needed, converted between different encodings. If the length of the source string is too long, the string will be truncated to fit in the target string. Please check chapter 6.1.12.5 for the limitations of string routing.	Frame View PDU View

Table 114: FL3X Config Gateway – Supported types of routings

If a routing is created for child items of a source and a target element (PDU or Frame), the following dialog will be shown when the routing for the last child item is created:

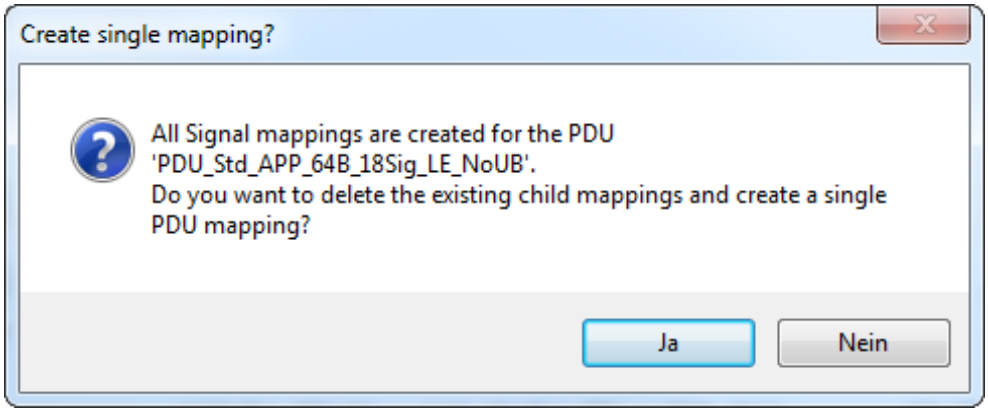


Figure 184: FL3X Config Gateway – Create parent routing

If you click on the “*Yes*” button, all created child routings will be deleted, and a single routing will be created for the parent item. This dialog will not be shown for a PDU, if any child signal routing is already scaled.

	Information
	The source and target items which are used by a routing will have another text format and the checkbox in the column “Used in Routing” will be checked. If an item exists in the source list and in the target list and this item is used only as source by a routing, then it will be marked as “Used in Routing” only in the source list. If an item exists in the source list and in the target list and this item is used only as target by a routing, then it will be marked as “Used in Routing” only in the target list.

6.1.2.1 Frame Routings

Following frame types are shown by the gateway editor:

No	Description
RbsTx	The RBS transmits this frame.
RbsRx	This frame is transmitted by a real ECU and can be received by the RBS.

No	Description
NonRbs	There is no active RBS configuration to transmit or receive this frame. For example, frames which are exchanged only between simulated ECUs have this type.

Table 115: FL3X Config Gateway – Supported frame types

	Information
	You can create byte array-based routings by holding down the "*Alt*" key and clicking on the route button. The content of the source frame/container PDU will be routed without considering the layouts. If the "*Alt*" key is pressed, you cannot use the context menu to create a routing. In this case you can use the route button or "drag & drop". If the "*Alt*" key is pressed, you cannot see the information, which is shown by the route button. In this case you can use "drag & drop".

	NOTICE
	If a frame or a container PDU is used as a target by a byte-array-based routing, then the PDUs and the signals of it cannot be used by user functions or FL3X Config Control. Additionally, Manipulations and Standard Signal Handling cannot be applied to byte array based routed frames.

	NOTICE
	The gateway editor permits to create a routing, whose source frame is also sent by the RBS (Tx Frame -> Tx Frame). Depending on the value of the define "USE_EVENT_TIMINGS_FOR_GW" (see Table 69) an event to send the source frame cannot be fired always automatically. You must use user functions to set the send event for the source frame in this case.

6.1.2.2 PDU Routings

Following routings are possible for a PDU (see chapter 6.1.12.2 for the limits of PDU routings):

		Target					
		Standard PDU	Multiplexed PDU	Switched PDU	Static PDU	Contained PDU	Container PDU
Source	Standard PDU	✓	-	✓	✓	✓	-
	Multiplexed PDU	-	✓	-	-	-	-
	Switched PDU	✓	-	✓	✓	✓	-
	Static PDU	✓	-	✓	✓	✓	-
	Contained PDU	✓	-	✓	✓	✓	-
	PDU-Container	-	-	-	-	-	✓

Table 116: FL3X Config Gateway – Supported routings for PDUs

		Target	
		Signal	Switch Code
Source	Signal	✓	✓
	Switch code	✓	✓

Table 117: FL3X Config Gateway – Supported routings for signals of multiplexed PDUs

	Information
	The switch of a multiplexed PDU can be used in routings as a signal.

	Information
	PDUs can be used in a routing either with other PDUs or service items (events, fields/notifiers, methods).

	NOTICE
	Invalid switch values of multiplexed PDUs are routed without any change.

	NOTICE
	The switch code of a multiplexed PDU is not updated automatically, if switched PDUs are used individually as routing target.

6.1.2.2.1 Multiple Instantiated PDUs

	NOTICE
	Routings for all instances of multiple instantiated PDUs or for their signals are supported only within the PDU view. All source instances of the items will be routed to all target instances in this view. Routings for single instances of a multiple instantiated PDU can only be created in Frame view.

6.1.2.2.2 Container PDUs

A container PDU can be routed to another container PDU if the count and layout of contained PDUs are matching. If there are routings with contained PDUs, the following rules are applied:

		Target		
		Contained PDU Last-Is-Best	Contained PDU Queued	Normal PDU
Source	Contained PDU Last-Is-Best	PDU is added to target container or replaced (1 instance)	PDU is added to target container (multiple instances)	PDU is routed
	Contained PDU Queued	PDU is added to target container (last instance)	PDU is added to target container (all instances)	PDU is routed (last instance)
	Normal PDU	PDU is added to target container or replaced (1 instance)	PDU is added to target container (multiple instances	PDU is routed

6.1.2.3 SOME/IP Service-Related Routings

	Information
	Service items (events, fields/notifiers, methods) can either be used in a routing with other service items or with PDUs.

	Information
	Data types of service items can be used in a routing with other data types or signals.

The source side of a routing must be a service item, which is being received by the FL3X Device.

Following service items can be used as **source**:

Service Item	Information
Consumed Event	Event and data can be routed
Consumed Field Notifier	Notifier and field data can be routed
Provided Field Getter	Getter call can be routed
Provided Field Setter	Field data and setter call can be routed
Provided Method	Input parameters of the method are considered. Method call can be routed.
Consumed Method	Return parameters of the method are considered. Method response can be routed.
Consumed Field Getter	Getter response and data can be routed.
Consumed Field Setter	Setter response and data can be routed if the setter has a response.

The target side of a routing must be a service item, which is being sent by the FL3X Device.

Following service items can be used as **target**:

Service Item	Information
Provided Event	Event will be triggered
Provided Field Notifier	Notifier will be triggered
Consumed Field Getter	Getter call will be triggered
Consumed Field Setter	Setter call will be triggered
Consumed Method	Input parameters of the method are considered. The method will be called
Provided Method	Return parameters of the method are considered. Method response will be routed.
Provided Field Getter	Getter response will be routed.
Provided Field Setter	Setter response will be routed if the setter has a response.

The following data types are supported by the routings:

- Data types which are at the lower level of the data structure (they have no child items).
- Data types which have only one child, which is an array
- String data types which are shown with the icon Ⓢ

	Information
	Field notifier, setter and getter are not displayed with separate nodes in gateway editor. The direction (provided/consumed) and access permission (write, read, notifier) of the field determine the behavior of the routing.

	NOTICE
	A routing which calls a field getter/setter or a method, does not automatically include the response routing. For example, if a provided method is routed to consumed method, the method call will be triggered on the consumer side, but the response of the method will not be automatically sent to the provider side. This means 2 routings are needed in this case: 1. Provided Method -> Consumed Method (Method call is routed) 2. Consumed Method ->Provided Method (Method response is routed)

6.1.3 Filter

In the FL3X Config Gateway editor text or column filters can be used.

NOTICE
If any cluster element is filtered on the source or target section (via text-filter or column filter), all other cluster elements, which belong to the filtered element, will also be shown.

6.1.3.1 Text Filter

Every section (source/target/routings) has its own textbox on the toolbar for text filtering. See chapters 6.1.1.1, 6.1.1.2 and 6.1.1.3 for more details.

Information
You can use wildcards to make more specific text filtering. For example, if you enter the text "*abc***def*" to the text-filter field, then all elements will be shown, which begin with the text "*abc*" and end with the text "*def*".

6.1.3.2 Column Filter

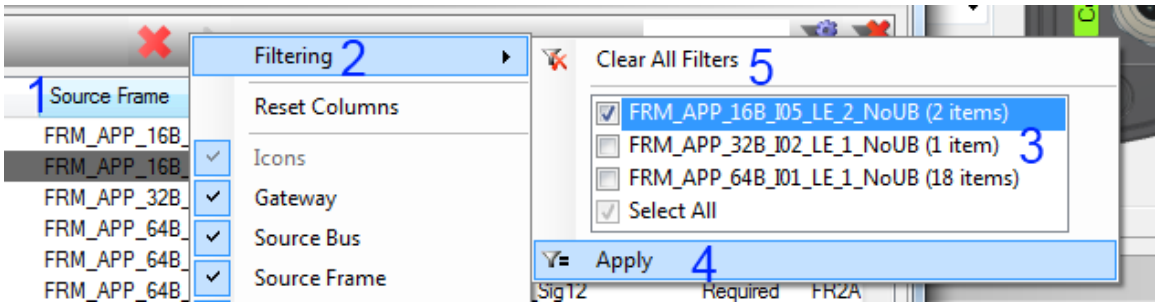


Figure 185: FL3X Config Gateway – Column filtering

- Right-click on the column title (**1**), which you want to filter
- In the context menu click on "*Filtering*" (**2**)
- Select the desired items (**3**)
- Click on "*Apply*" to activate the filter (**4**)

If you want to deactivate the column filtering

- Right-click on the title of any column (**1**)
- In the context menu click on "*Filtering*" (**2**)
- Click on "Clear All Filters" (**5**)

Here is some more information about filtering in FL3X Config.
In most cases, FL3X Config uses "TreeListView" as list type: This is a tree view with additional columns. It contains parent and child nodes. For example, TreeListView is used in:

- Gateway Editor -> Network elements
- FL3X Config Control -> Network elements
- Default Values Editor -> Network elements

In some cases, FL3X Config uses "Listview": This is a normal table with multiple columns. For example, Listview" is used in:

- Gateway Editor -> Routings
- FL3X Config Control -> list of global variables
- Default Values Editor -> List of profiles

Column filtering shows different behaviors in different list types. Please consider the following limitations of column filtering:

- All child elements of a filtered element are also shown by a TreeListView without considering whether they match the filter or not.

- All parent elements of a filtered element are also shown by a TreeListView without considering whether they match the filter or not.
- The filter view of a TreeListView (which is opened by a right click on a column header) is based on the current state of the list. This means, only currently shown elements can be filtered. It may be necessary to expand more elements to see them by the filter view.
- A ListView always shows all elements in the filter view.

6.1.4 Display Items

The cluster elements and the columns, which are shown in the gateway editor, depend on the “Gateway Editor View” which is selected for the project (see chapter 5.22.2.1). Following “Gateway Editor View” modes are available:

- PDU-View
- Frame-View

Frames and corresponding columns will not be shown, if the PDU-View is activated. Just one instance of multiple instantiated PDUs will be shown in the PDU-View.

To display the desired cluster elements, you can use the “*View*” button of the toolbars (source /target sections, see Figure 186).

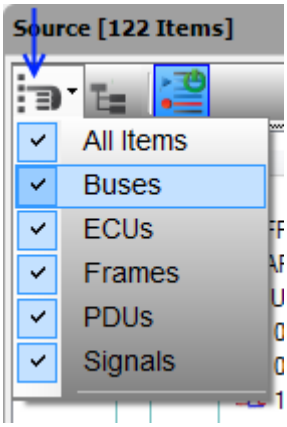


Figure 186: FL3X Config Gateway – Cluster view button

6.1.4.1 Configure Columns

Gateway editor shows several columns for source, target, and routing list views. The source and target lists show among others following columns:

Used In Routing: This column can have following values:

Not Routed:	The element is not routed
Routed:	The element is routed
Child Routed:	A child element is routed
Parent Routed:	A parent element is routed

Routing Type: This column shows the routing type for the routed element.

Info: This column shows the frame id for the CAN and LIN frames. It shows the frame id, base cycle, and repetition for the FlexRay frames.

You can also show/hide the columns in any section of the FL3X Config Gateway editor.

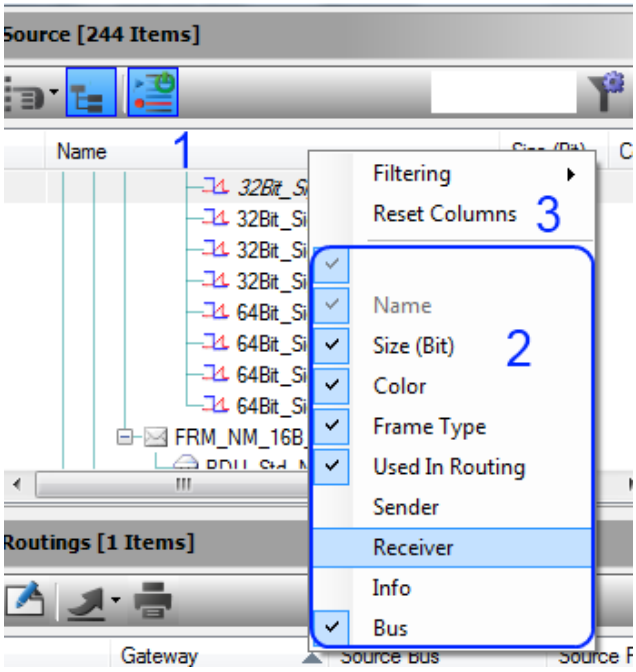


Figure 187: FL3X Config Gateway – Show/Hide columns

To show the desired columns

- Right-click on the title of any column (**1**)
- Select the columns, which should be shown or hidden (**2**)

To display the default column view by any section

- Right-click on the title of any column (**1**)
- Click on "*Reset Columns*" (**3**)

If you want to change the order of a column, click on the title of it and use drag & drop for moving it to another position.

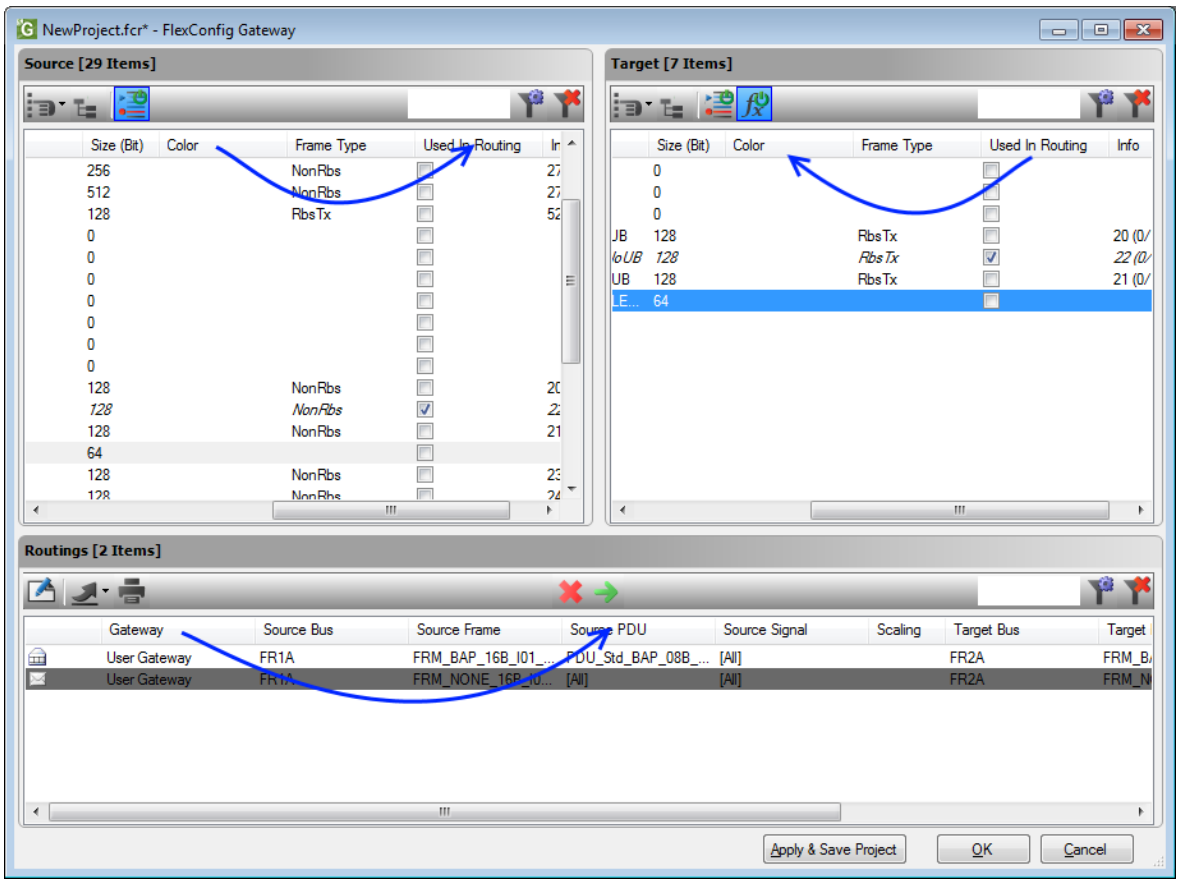


Figure 188: FL3X Config Gateway – Change column order

You can **sort** each column by clicking on its title.

6.1.4.2 Highlight Items with a Color

You can set the background color for the selected items on any section:

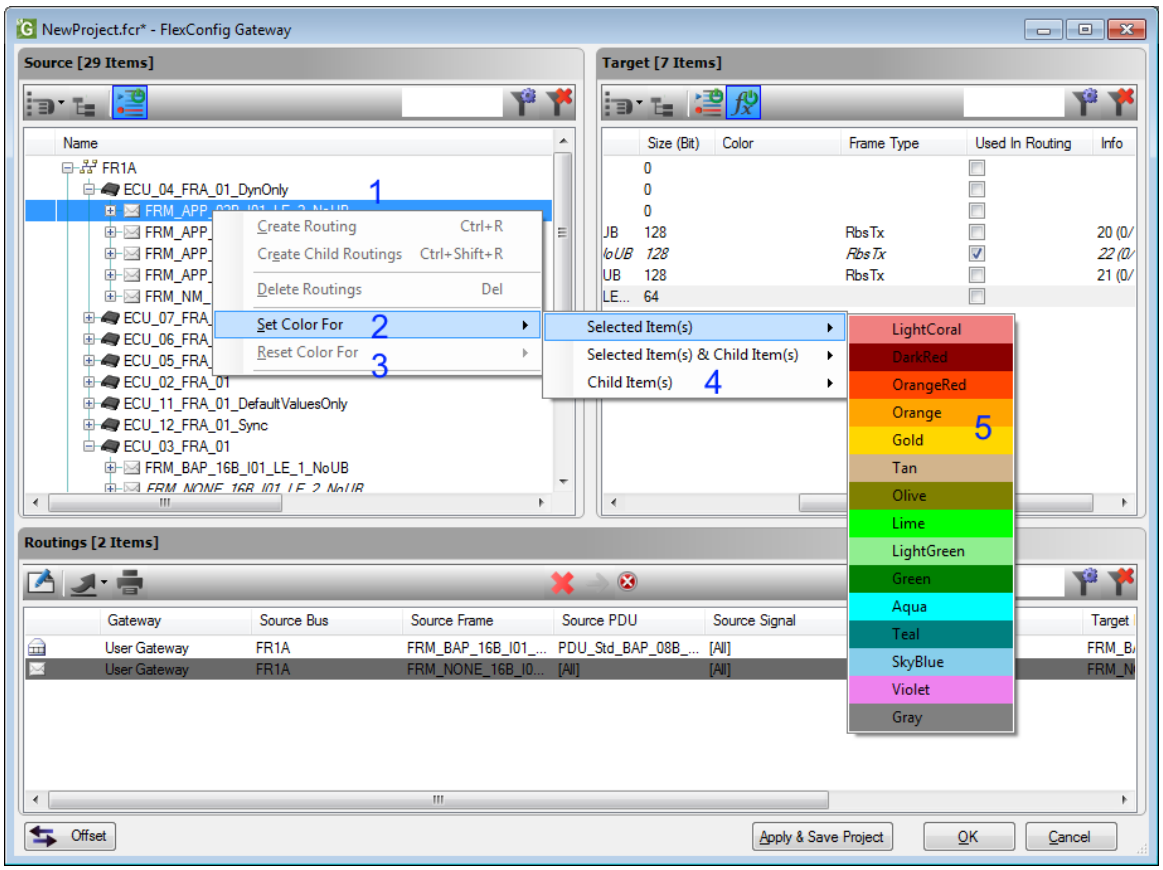


Figure 189: FL3X Config Gateway – Set color to highlight items

To set the background color:

- Select the item(s) which should be colored (**1**) and right-click on a selected item
- Select "*Set Color For*" from the context menu (**2**)
- Select the items to be colored (**4**)
- Select the color (**5**)

You can reset the color in the same way (**3**).

	Information
	If the color of a source or a target item is changed, then the same item on the other side will get the same color.

6.1.5 Scaling Signals

You can enable or disable the scaling for signal routings by clicking on the scaling button in the toolbar of the target section (see Figure 190).

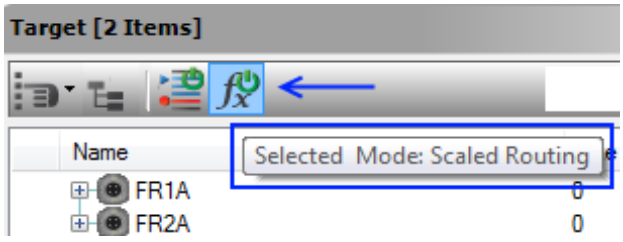


Figure 190: FL3X Config Gateway – Enable/Disable scaling mode

If the scaled mode is enabled and the signals have different layouts or/and different Compu-Methods, then the "Signal Scaling" editor will be opened when a routing is created. Please check chapter 6.1.12.1 for the limitations with signal routings.

The “Signal Scaling” editor can also be opened after creating a signal routing by double-clicking on the routing or clicking on “*Edit Scaling*” via the context menu.

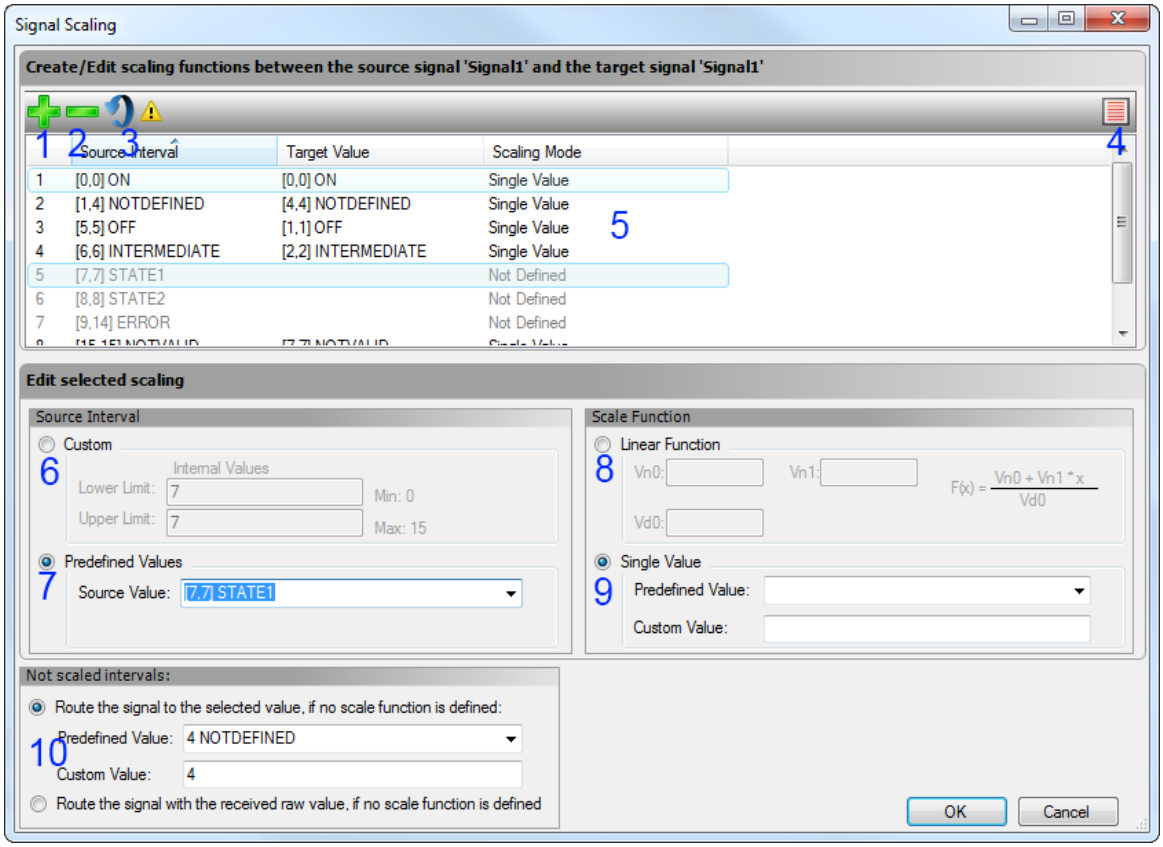


Figure 191: FL3X Config Gateway – “Signal Scaling” dialog

No	Description
1	Adds a new (not used) source interval. If all source intervals are used, then adds the biggest source value.
2	Deletes the selected interval(s).
3	Recreates the possible intervals with scaling.
4	This button opens an information dialog (layout and Compu-Methods) about the source and the target signal.
5	The section for source intervals and their scaling. Target value shows the calculated target interval with using the source interval and the scaling function. Scaling mode shows the type of the scaling function: Single Value: Source interval is scaled to single value Linear Function: Source interval is scaled with a linear function Not Defined: No scaling function is defined
6	In this section, you can define a custom source interval.
7	In this section, you can select a predefined source interval. Predefined source intervals are either the values of text tables or constraint by the Compu-Methods
8	In this section, you can define a linear scaling function.
9	In this section, you can select a predefined target value or enter a single custom value. Predefined values are the constraint by the Compu-Methods or values of text tables of the target signal. If a text-table value of the target signal has a range (for example [5, 10] is “ON”), then the minimum value of this range will be shown (as [5,5] “ON”).
10	In this section, you can set the action for the source signal values, which belong to none of the defined source intervals. They can be routed with the received raw values or routed as another (predefined/custom) single value.

Table 118: FL3X Config Gateway – “Signal Scaling” dialog options

If you define a custom interval and click on the field for the lower or the upper limit, then the following mask will be opened:

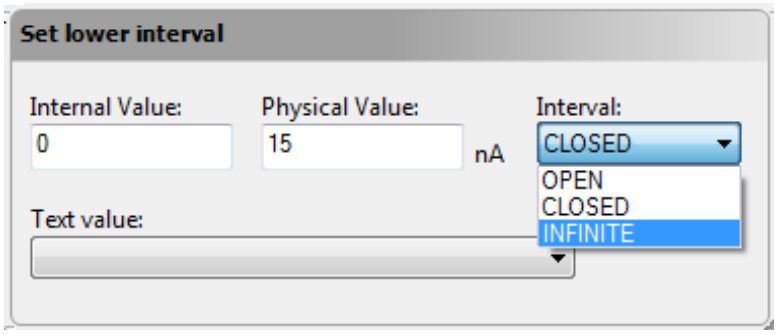


Figure 192: FL3X Config Gateway – "Signal Scaling" editor – Interval mask

Value	Description
CLOSED	The interval includes the selected value.
OPEN	The interval does not include the selected value
INFINITE	The selected value is not considered. Depending on the configured side (lower or upper limit), the minimum or maximum internal value of the signal will be used.

Table 119: FL3X Config Gateway – "Signal Scaling" editor – Interval types

The "scaling" column in the gateway routings section shows different values depending on the scaling status:

Value	Description
-	Scaling is not possible. The source or target signal is larger than 32 bits.
Required	The routing must be scaled (because of the different Compu-Methods or bit lengths).
Not Required	Scaling is not required. Signals have the same layout and Compu-Methods.
Yes	At least one interval of the source signal is already scaled.
Empty	If a routing is not a signal routing, then no value will be shown.

Table 120: FL3X Config Gateway – Scaling status

	Information
	Intervals, for which no scaling can be suggested, are shown in grey color. There are the following reasons: • The source signal has a linear Compu-Method, but no matching target scaling can be found. • The source signal has a text-table value, but no matching target text-table value can be found. • The target signal has a text-table value, but no matching source text-table value can be found. Two unused intervals can be assigned to each other via drag & drop.

	NOTICE
	Scaling suggestions can be made only for the signals which have the Compu-Method types "Linear", "Identical" or "Text-table".

	NOTICE
	Source intervals and the computed target values are based on internal values and not on raw or physical values.

NOTICE
If you click on the “*Cancel*” button in the “Signal Scaling” editor while creating a routing, no routing will be created.

NOTICE
The “Target value” column does not show any decimal values. The calculated decimal values are rounded to smaller integer values.

6.1.6 Auto Routing

Auto routing mechanism helps you to create proper routings between two busses with different configurations.

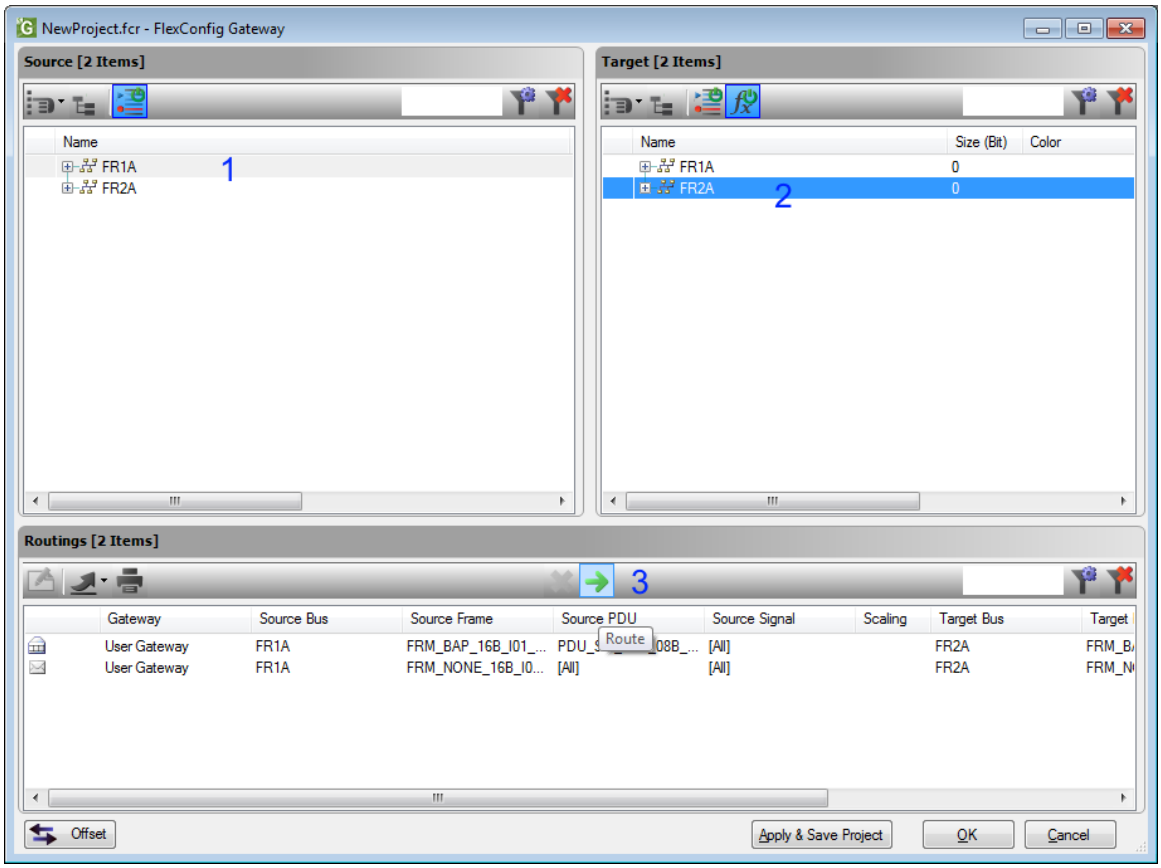


Figure 193: FL3X Config Gateway – Open the “Create Autoroutings” dialog

To open the “Create Autoroutings” dialog select the source and target busses (**1** & **2**) and click on the route button (**3**).

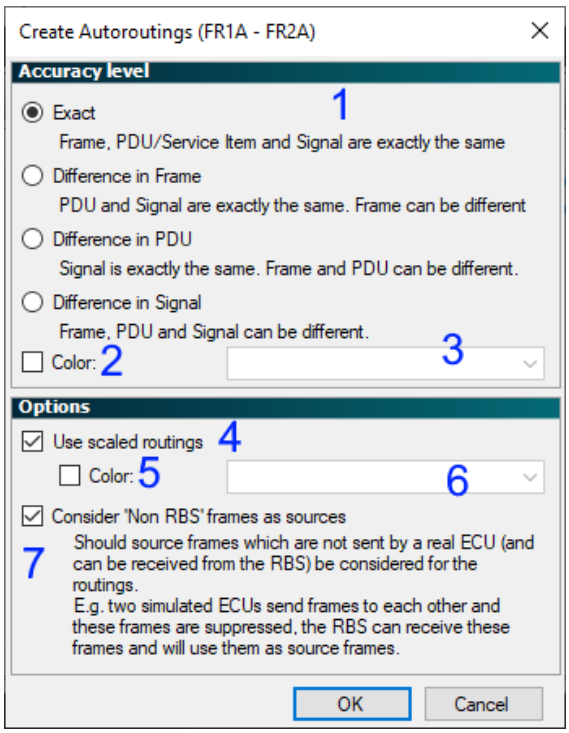


Figure 194: FL3X Config Gateway – "Create Autoroutings" dialog without ethernet buses.

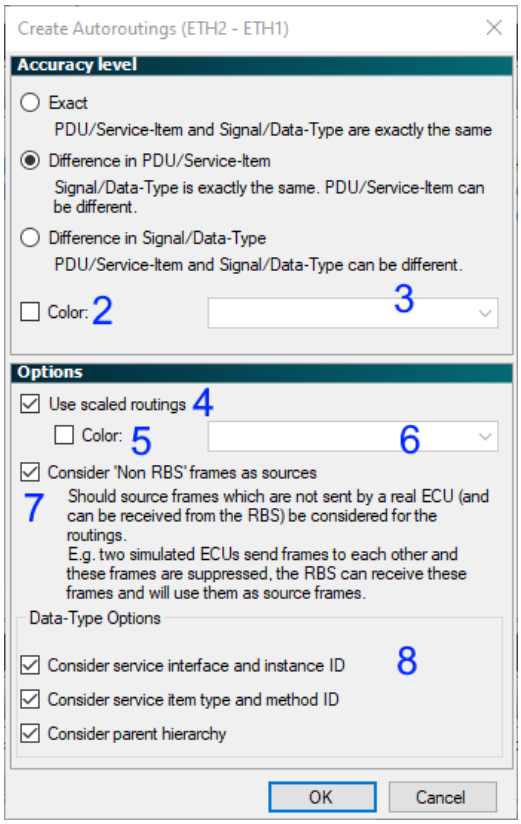


Figure 195: FL3X Config Gateway – "Create Autoroutings" dialog with ethernet buses.

No	Description
1	Options for finding the matching items. This option can differ depending on the selected view (PDU view or Frame view).
2	Enable to set a specific color for created routings.
3	List of colors. The selected color will be used for the created routings
4	Enable to use scaled signal/data type routings (see chapter 6.1.5).

No	Description
5	Enable to set a specific color for created scaled routings.
6	List of colors. The selected color will be used for the scaled signal routings.
7	Option to consider source frames without a real sender (based on the RBS configuration).
8	These options are available, if at least one side is an ethernet bus and the 'exact' option is not selected. • *Consider service interface and instance ID*: If checked, the service interface Identifier and the provided service instance ID must be same. • *Consider service Item type and method ID*: The service item type (event, method, field) and method IDs must match. • *Consider parent hierarchy*: The count of parents and their short names must match for the data types.

Table 121: FL3X Config Gateway – "Create Autoroutings" dialog

	Information
	The auto routing mechanism does not delete any manually created routings.

6.1.7 Split a Routing

You can split existing routings and create child routings (for the items at the lower levels). Following operations are possible:

- Split a PDU routing to signal routings
- Split a PDU based frame routing to PDU routings.

You can open the context menu by a routing and select "*Create Child Routings*" or double click on the routing to split it.

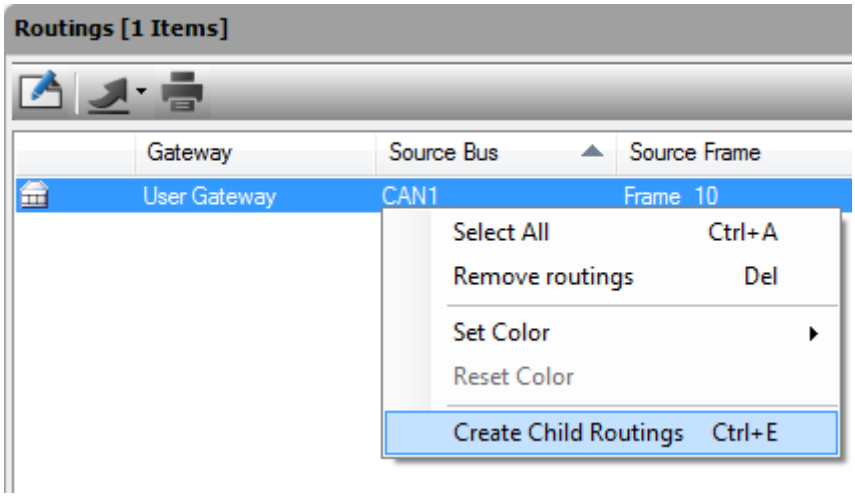


Figure 196: FL3X Config Gateway – Create child routings of an existing routing

The following dialog will be shown:

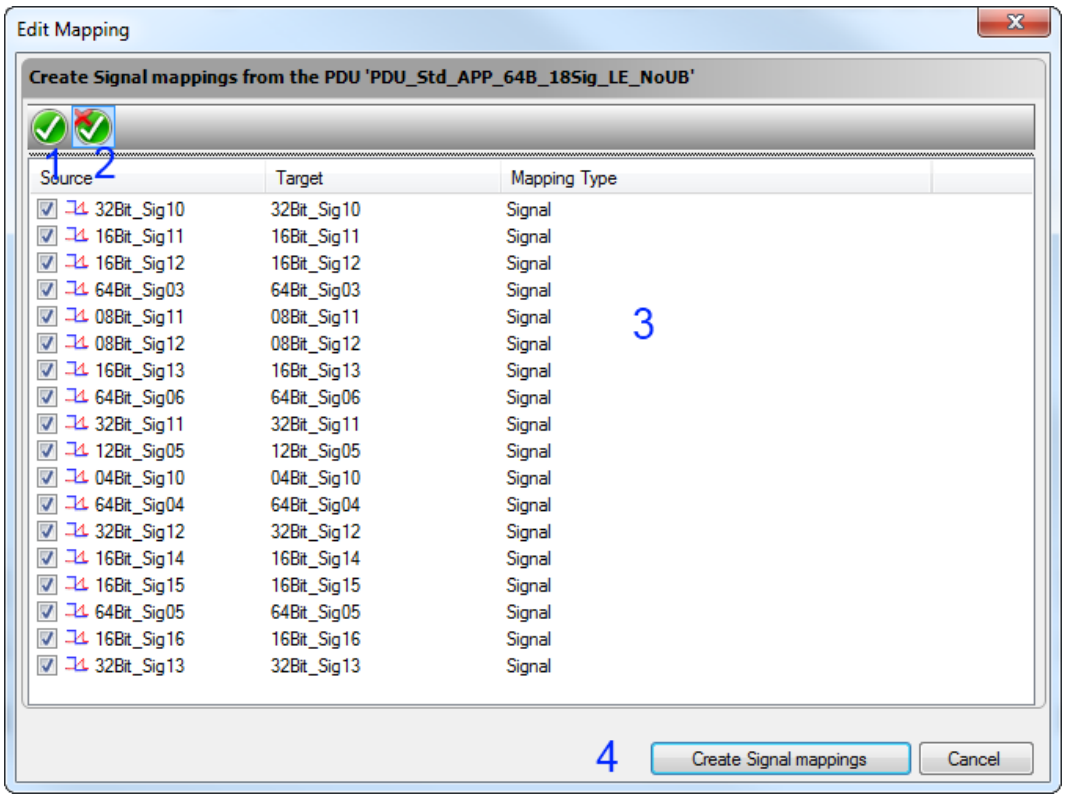


Figure 197: FL3X Config Gateway – Dialog to create child routings

No	Description
1	Toolbar button to select all items.
2	Toolbar button to deselect all items.
3	List of child items, for which routings should be created.
4	Button to create child routings.

6.1.8 Delete Routings

You can select the routings on the routings list, which you want to delete and click on the "*Delete*" button in the toolbar. Another alternative would be to open the context menu and select "*Remove routings*" or to press the "*Delete*" key.

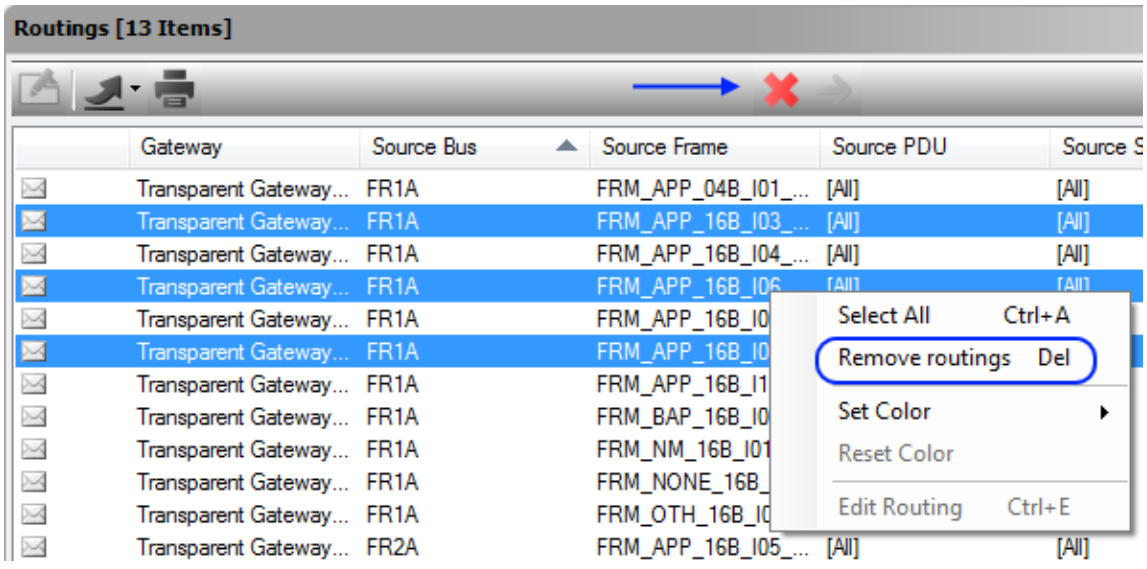


Figure 198: FL3X Config Gateway – Delete routings in the routings list

If you want to delete the routings for the items on the source or target sections, select the item, and click the "*Delete*" key or open the context menu and click on "*Delete Routings*":

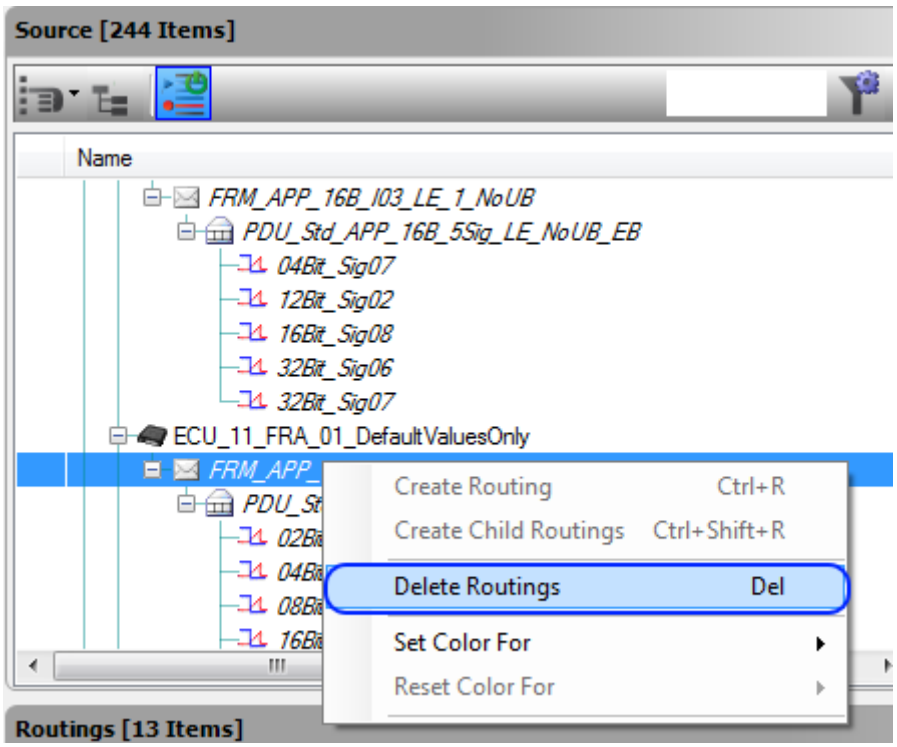


Figure 199: FL3X Config Gateway – Delete routings in the source or target list

The following dialog will be opened which shows the routings to be deleted:

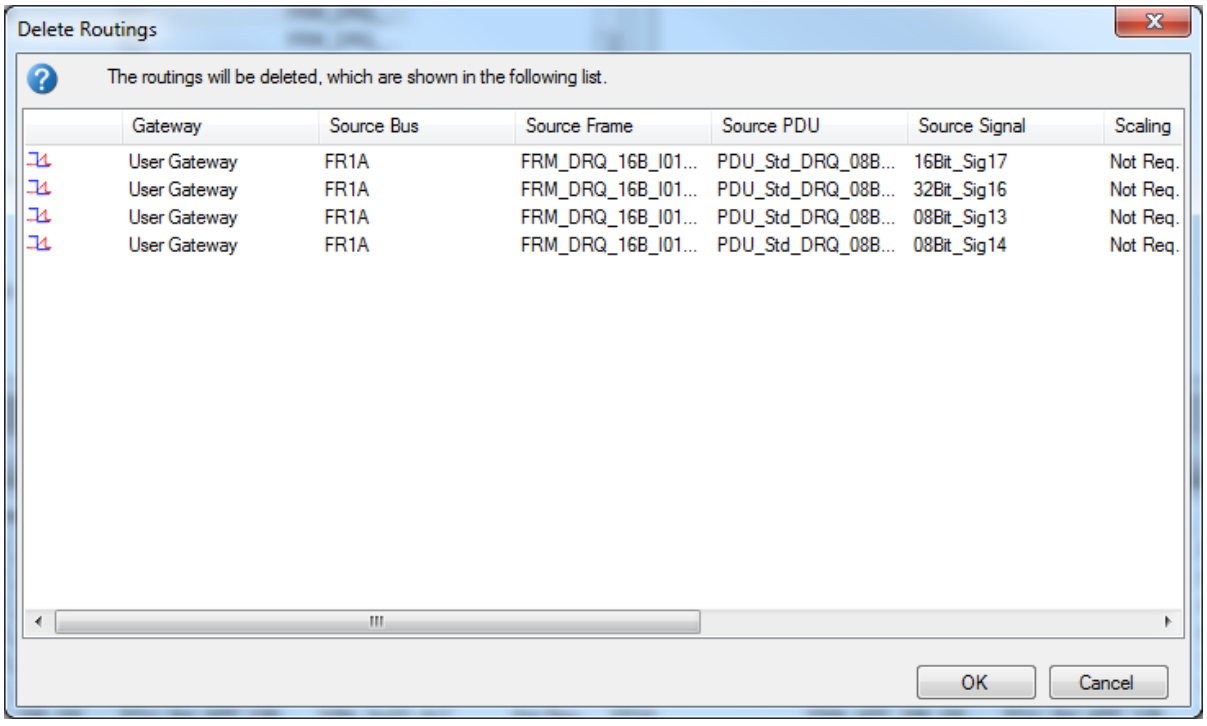


Figure 200: FL3X Config Gateway – “Delete Routings” dialog

	Information
	Selecting “*Delete Routings*” on the source section will delete all routings, which use the selected items or their child items as source. Selecting “*Delete Routings*” on the target section will delete all routings, which use the selected items or their child items as target.

NOTICE
“Delete Routings” dialog will also be shown if you try to create a routing for 2 items, whose child items have already routings.

6.1.9 Export Items

With the button “Export” (see Figure 201), you can export the content of any section in the FL3X Config Gateway editor. This purpose of this feature is to export information about items based on a user defined selection (e.g. export only the columns “Source Bus” and “Target Bus” of the list “Routings”).

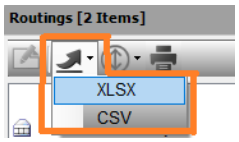


Figure 201: FL3X Config Gateway – Export Items

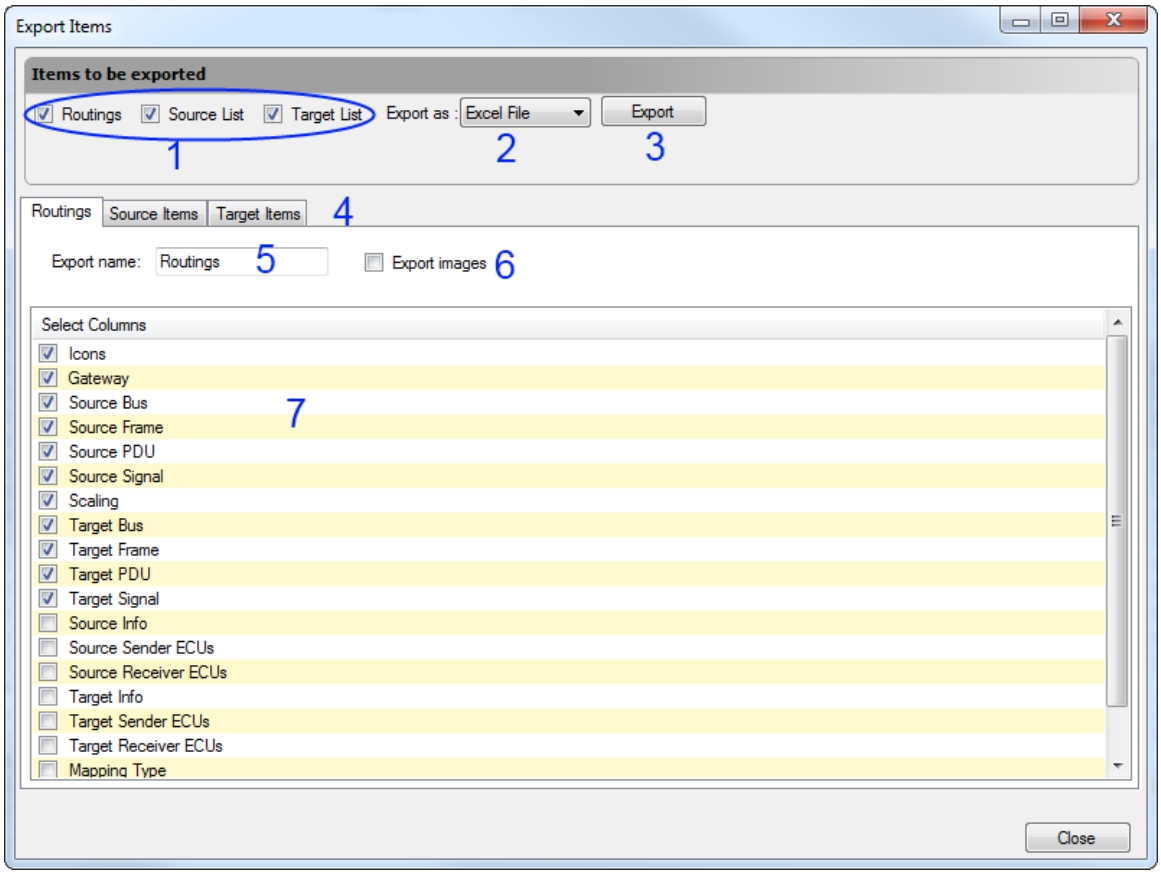


Figure 202: FL3X Config Gateway – “Export Items” dialog

No	Description
1	Select the sections which should be exported. The “Routings” section is always selected as default.
2	The export file type.
3	The “*Export*” button starts the export.
4	Configuration which content of each section should be exported.
5	The name, which is entered here, will be used as work sheet name, if you export the content as an excel file. If you export the content as a csv-file, then this name will be added to the file name.
6	Enable to export the images (only available for export to excel).
7	In this section the column names are shown, which are activated on the corresponding list. You can select/unselect more columns. The information by the selected columns will be exported

Table 122: FL3X Config Gateway – “Export Items” dialog options

NOTICE
The file format "*xlsx*" is used for exporting to Microsoft Excel. If your office version does not support this format, you can download updates and converters for an earlier version of Microsoft Excel.

6.1.10 Print

You can print the content of any section in the FL3X Config Gateway editor.

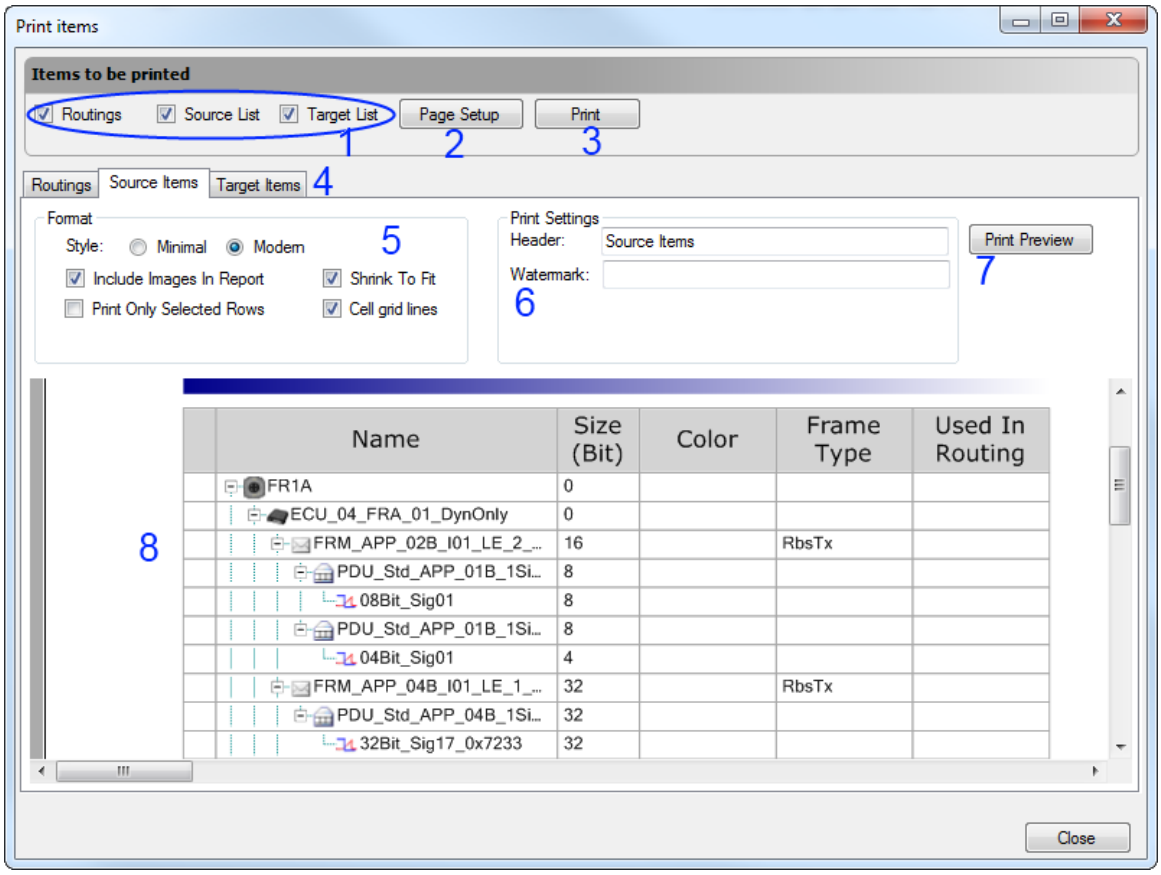


Figure 203: FL3X Config Gateway – "Print Items" dialog

No	Description
1	Select the sections which should be printed. The "Routings" section is always selected as default.
2	The "*Page Setup*" button opens a dialog for the page setup.
3	The "*Print*" button starts the print.
4	Configuration which content of each section should be printed.
5	Configuration options for the print style.
6	Options for the header and the watermark.
7	The "*Print Preview*" button opens a dialog to preview the content that will be printed.
8	This section shows the current content that will be printed.

Table 123: FL3X Config Gateway – "Print Items" dialog options

Information
If multiple lists are selected for printing, then following order will be used: Routings list -> Source list -> Target list.

NOTICE
The actual view of the selected list(s) will be printed. This means, that the actual size and the order of the columns should be customized to prevent any problems.

6.1.11 Import and Export of Routings

NOTICE
This feature requires the “FL3X Config Enterprise” license. If this license is not available, the buttons “Import Routings” and “Export Routings” are disabled.

In addition to the feature “Export Items” (see chapter 6.1.9), there is the feature “Import/Export Routings”. The main difference between these two features is, that “Import/Export Routings” works with a defined format specially to support importing. E.g. the name and order of the columns is predefined and cannot be changed by the user.

Reference
More information about this feature can be found in the document 3-01140A01-D017_FL3X_Config_Guidelines_for_Routing_tables. Use the template document were all necessary columns for a proper import are defined: 3-01140A01-D016_Template_routing_table.xlsx Both documents can be found via “Start Menu –> FL3X Config %version%”.

To import or export routings, click the “Import/Export Routings” button (see Figure 204).

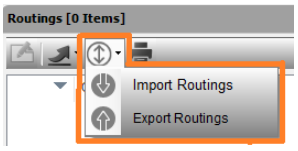


Figure 204: FL3X Config Gateway – Import/Export Routings

The following dialog is shown after the import/export menu item is clicked:

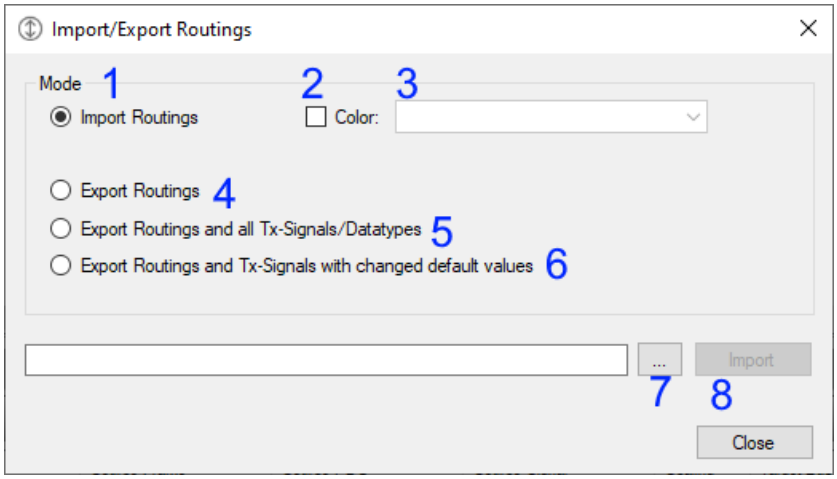


Figure 205: FL3X Config Gateway – Import/Export Routings Dialog

No	Description
1	Import mode
2	Activate this checkbox if you want to have the imported routings shown in another color.
3	Dropdown list for colors
4	Just the routed items will be exported.
5	The routed items will be exported. The exported file will also contain the list of all signals, which are shown in the target list.
6	The routed items will be exported. The exported file will also contain signals of the target list, whose default values are changed.
7	Select the file path (for import/export) via this button
8	Import/Export button

Table 124: FL3X Config Gateway – “Import/Export Items” dialog options

6.1.12 Limits of a Gateway

This chapter contains restrictions and limitations for gateways.

	NOTICE
	Names and ho:OIDs of frames, PDUs and signals are **not** checked when a routing is created.

6.1.12.1 Limits for Signal Routings

	NOTICE
	Signals, which are larger than 32-bits (see chapter 2.8.3), and whose length is not evenly divisible by 8, are not supported.

Routing signals with different length

It is possible to route signals with different length. If the scaling mode is not activated, then the length of the source signal cannot be larger than the length of the target signal. If you activate the scaling mode, then length of the source signal can also be bigger than the length of the target signal.

Scaling mode

Signals cannot be scaled if the length of the source or the target signal is larger than 32 bits.

Signed integer to unsigned integer

When a signed integer signal is routed to a larger unsigned integer signal, and both signals have a length less than or equal to 32 bits, then the unsigned integer value which is interpreted by the compiler will be sent instead of the raw value of the signed integer signal.

Example: Routing from s08 to u16

The s08 value is internally casted to s16, so s08 “0xFE” becomes s16 “0xFFFE”. Then the s16 value is internally casted to u16 so the value now is u16 “0xFFFE”.

When a signed integer signal is routed to a larger unsigned integer signal and any signal length is bigger than 32 bits, then the raw value will be copied, without considering the sign of the source value.

Example: Routing from s16 to u64

The value “0xFFE7” will be routed as “0x000000000000FFE7”

6.1.12.2 Limits for PDU Routings

For direct PDU routings, both PDUs must have the same size. The PDUs should have the same layout. This means, that they must have the same number of signals with the same length and order. If the routed PDUs have the same size but not the same layout, a warning is shown in “FL3X Config Gateway Editor”.

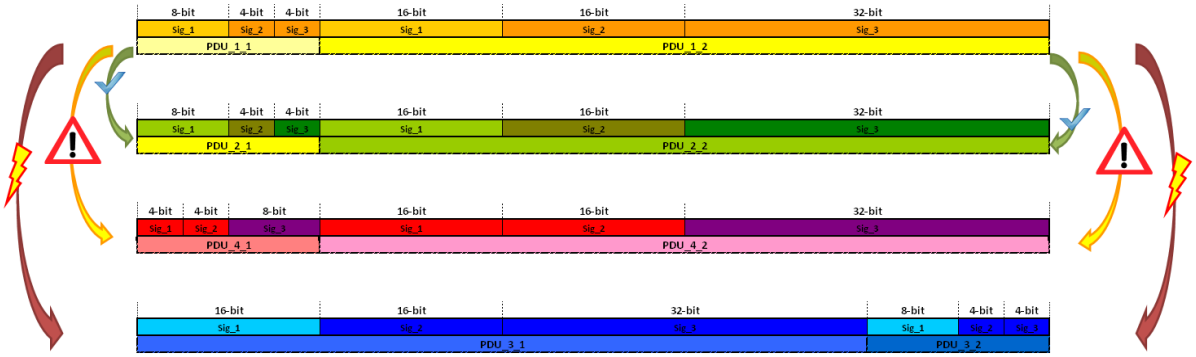


Figure 206: Routing of PDUs with different PDU layout

	NOTICE
	A direct PDU routing is not signal based. Direct PDU routings perform only a byte copy from the source PDU to the target PDU (byte ‘0’ of the source PDU is copied to the byte ‘0’ of the target PDU, byte ‘1’ of the source PDU is copied to byte ‘1’ of the target PDU and so on). E. g. PDU_A has signals Sig_1A (bit position ‘0’, 8-bit length) and Sig_1B (bit position ‘8’, 8-bit length) and a PDU_B has the signals Sig_2B (bit position ‘0’, 8-bit length) and Sig_2A (bit position ‘8’, bit-length ‘8’). With a ‘direct PDU routing’ the signal Sig_1A would be routed to signal Sig_2B and signal Sig_1B would be routed to signal Sig_2A.

For multiplexed PDUs this means:

- The switches should have the same layout.
- The dynamic segments should have the same layout.
- The number of switch codes should be identical.
- The assignment of switch codes and switched PDUs should be identical.
- The switched PDUs should have the same layout.
- Both multiplexed PDUs should have the same number of static PDUs with the same layout.

If a switched-PDU, which has multiple segments, should be routed to a normal PDU, then the routing will only work correctly if following conditions are satisfied:

- The start bit position of every segment must be aligned at a byte boundary.
- The length of each segment must be evenly divisible by 8.

6.1.12.3 Limits for Frame Routings

A frame cannot be routed directly to another frame. If a routing between two frames is created in “FL3X Config Gateway Editor” PDU routings are created. The limits for PDU routings are described in chapter 6.1.12.2. In addition, the routed frames must consist of an equal number of PDUs with identical size and order.

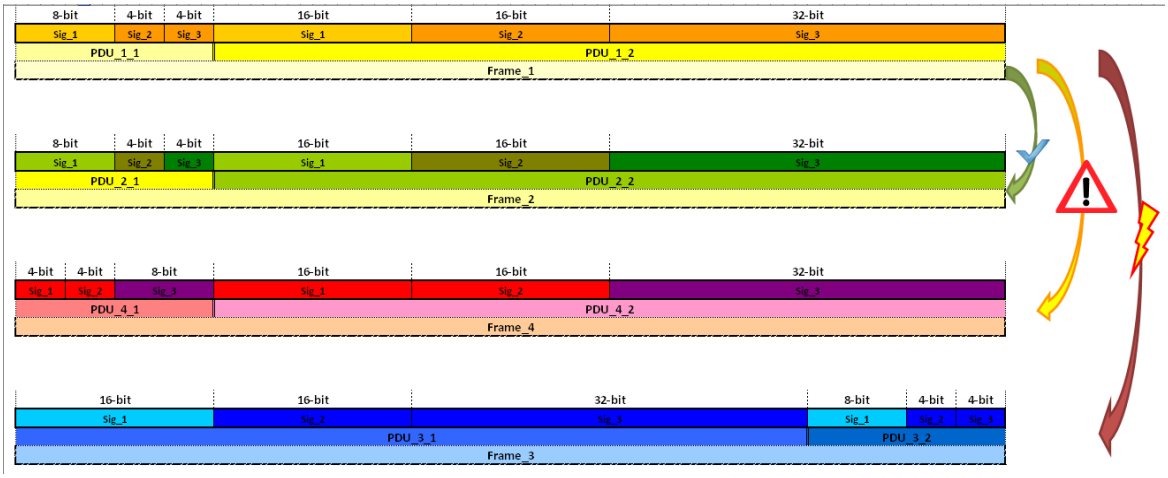


Figure 207: Routing of frames with different frame layout

Example:

Frame1:
(PDU1_1: sig1_8bit, sig2_4bit, sig_3_4bit), (PDU1_2: sig1_16bit, sig2_16bit, sig_3_32bit).

Frame2:
(PDU2_1: sig1_8bit, sig2_4bit, sig_3_4bit), (PDU2_2: sig1_16bit, sig2_16bit, sig_3_32bit).

Frame3:
(PDU3_2: sig1_16bit, sig2_16bit, sig_3_32bit), (PDU3_1: sig1_8bit, sig2_4bit, sig_3_4bit).

Frame4:
(PDU4_1: sig1_4bit, sig2_4bit, sig_3_8bit), (PDU4_2: sig1_16bit, sig2_16bit, sig_3_32bit).

Frame1 can be routed to Frame2.

Frame1 != Frame3 => Frame1 cannot be routed to Frame3. The PDUs have the same size, but the order of the PDUs is different.

Frame1 can be routed to Frame4. Only the order and size of the PDUs is relevant. The PDU layout (order/size of the signals) within the PDUs PDU1_1 and PDU4_1 is different, but not considered here. A warning is shown in the “FL3X Config Gateway Editor” in this case.



NOTICE

If time synchronization frames (see chapter 5.10) are configured, routings to these frames will not work, because the routed data will always be overwritten with the time synchronization data.

6.1.12.4 Limits for SOME/IP Routings

Routings with SOME/IP elements (event, method, field) may not work properly, if the source and the target elements use different E2E Profiles.

6.1.12.5 Limits for String Routings

Currently, only strings with encoding UTF-8 and UTF-16 are supported. String routings are not supported for other encodings. If a direct string routing is not possible, a signal routing (array -> array) can be used instead.

6.2 Transparent Gateway

The transparent gateway is used to split one or more ECUs from the cluster and reconnect them via a gateway. The gateway is transparent to the cluster. All ECUs work as normal. With the gateway the user has the possibility to control the routed data. So, it is possible to change the data in different ways or hide complete frames and ECUs (see chapter 6.2.5).

The Transparent Gateway supports the bus systems CAN and FlexRay. For FlexRay there is a configurable offset between the two FlexRay clusters at each side of the transparent gateway described in the following part.

6.2.1 Transparent Gateway in the Main Window

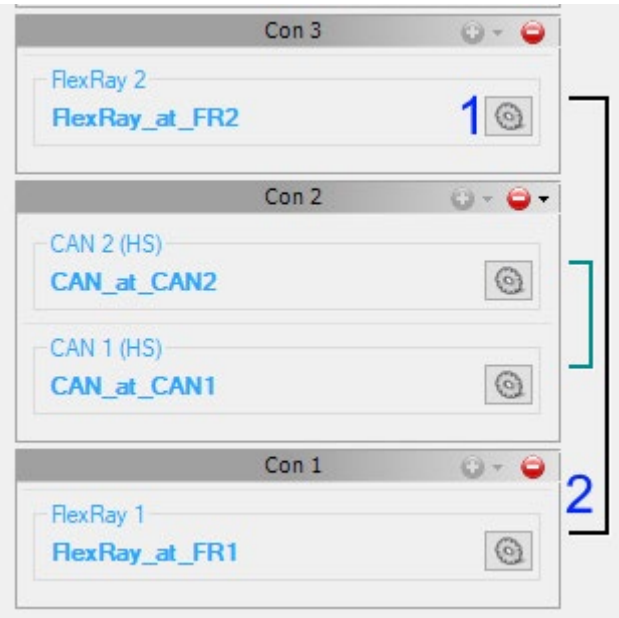


Figure 208: Main window with transparent gateway

After a transparent gateway has been created, the connectors of the “Main Window” will look similar to the ones shown in Figure 208. The connectors used by a transparent gateway are graphically connected with a connection line containing a gateway symbol.

No	Description
1	Using the settings button of the appropriate connector provides the possibility to change the actual settings.
2	The gateway indicator shows which connectors are used for the transparent gateway.

	NOTICE
	If the transparent gateway is created for a FL3X System-M and the used connectors are on different slots, then dashed lines are shown by the connectors to visualize the transparent gateway.

6.2.2 Edit Connectors of a Transparent Gateway

If a connector of a transparent gateway shall be edited, the ‘*Settings*’ button has to be pressed (see Figure 208). After this has been done a warning dialog pops up (see Figure 209).

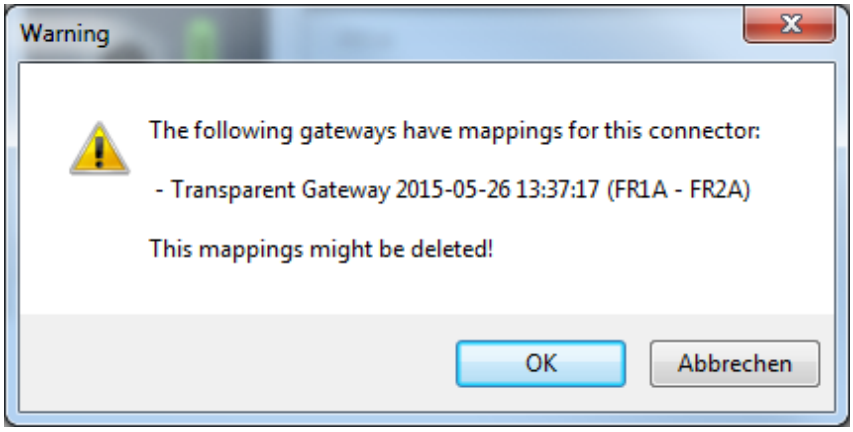


Figure 209: Warning dialog of transparent gateway

To continue the editing of the connector it is necessary to press the '*OK*' button. The warning dialog pops up, because changes of the connector configuration can lead to unwanted behavior.

When frames were removed during editing the connector, the corresponding gateway routings will also be removed automatically. When frames were added during editing the connector, **no** gateway routings will be added.

	NOTICE
	Be careful with manual changes of the connector configuration. A transparent gateway may not work as intended after those changes! For example, if the FlexRay synchronization configuration is manually changed for both sides of a transparent gateway, the cluster Synchronization may not work any longer.

6.2.3 Remove a Transparent Gateway Configuration

To remove a transparent gateway, the delete button (red cross, see Figure 208) of an involved connector can be used. After pressing the delete button a pop-up dialog opens, which must be confirmed. The pop-up dialog gives information about what items will additionally be removed (e.g. gateway routings based on this bus, and so on).

When one bus of a Transparent Gateway is removed, the settings which were set by the transparent gateway feature (User Defines, etc.) will be set to the default value.

6.2.4 Limits of a Transparent Gateway

The gateway is a PDU based gateway which means that frames without PDUs cannot be routed from one gateway side to the other.

The transparent gateway will send these frames (if they are within the FlexRay static segment) but the affected payload bytes are not routed.

The following limit affects only transparent gateways with FlexRay:
Synchronization of clusters with InCycle-Offset (see item **7** of the table in chapter 6.2.5) and Cycle Offset (see item **10** of the table in chapter 6.2.5) is only possible if the gateway side with the offset (e.g. gateway side B in Figure 210) is connected to a cluster with less than two coldstart / synchronization nodes (ECUs). If the configuration has two or more coldstart / synchronization nodes at the gateway side with the offset, a warning is shown.

There are more limits for a Transparent Gateway, see chapter 6.1.

	NOTICE
	For limitations about available message buffers (FlexRay) see chapter 5.6.2.1.

6.2.5 Create a Transparent Gateway

The other way is to create a new project by using the '*File->New Project*' item. After this has been done, a Transparent Gateway can be created by using '*Edit->Transparent Gateway->Create new...*'.

A Transparent Gateway can be created only for empty busses. An existing RBS cannot be used in a Transparent Gateway. It is possible to create multiple Transparent Gateways.

The following Transparent Gateway dialog will appear:

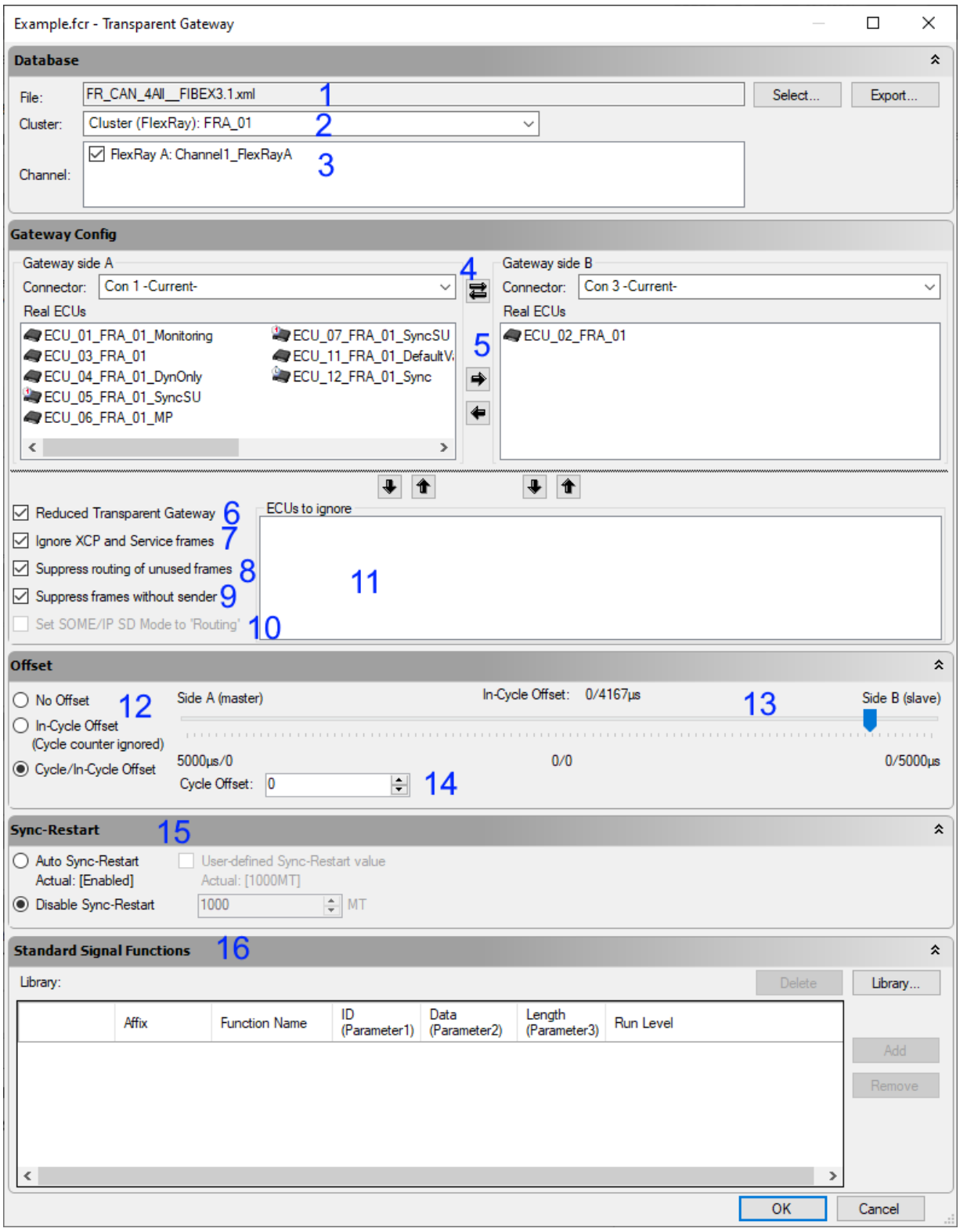


Figure 210: Transparent Gateway configuration dialog

The Transparent Gateway dialog (see Figure 210) provides all settings for a Transparent Gateway.

No	Description
1	Shows the selected database for the transparent gateway. With the button *"Select..."* the "Database Picker" (see chapter 5.5.1) is shown to select the database used for the transparent gateway.
2	With this drop-down box a cluster of the FIBEX file can be selected. Only CAN and FlexRay clusters are supported.
3	In the "Channels" section one or more channels can be selected. E.g. if a cluster with FlexRay A and FlexRay B channel is selected, both or one of the channels can be selected.
4	Using the  button the connectors can be reversed. The connectors themselves can be configured with the drop-down boxes for the gateway side A and B.
5	The ECU lists showing the ECUs on gateway side A and B. The ECUs can be switched from side A to B and back with the buttons  and  or by drag & drop.
6	When "*Reduced Transparent Gateway*" is checked, the resulting RBS configuration will only have "Real ECUs". "Simulated ECUs" are not configured. This means that only frames which are explicitly sent/received by the real ECUs will be received, cached, routed and sent. This will reduce the required runtime performance of a transparent gateway.
7	When "*Ignore Service and XCP frames*" is checked, all XCP, TPL and Service frames will be ignored. They will not be sent by the underlying RBS and they will not be handled by the gateway. This option is like the RBS option "*Ignore Service and XCP frames*". (See 5.6.3 RBS Settings)
8	When "*Suppress routing of unused frames*" is checked, unused frames (frames only sent and received by ECUs at ONE gateway side) will not be routed. This option is like the RBS option "*Suppress frames between simulated ECUs*". (See 5.6.3 RBS Settings)
9	When "*Suppress frames without sender*" is checked, frames without sender ECUs are not considered by the transparent gateway.
10	The option "*Set SOME/IP SD Mode to 'Routing'*" is only available for ethernet gateways. If this option is activated, the option "SOME/IP SD Mode" is set to "Routing" for all provided services. If this option is deactivated, the option "SOME/IP SD Mode" is set to "StandAlone" for all provided services. The option can be changed later for each service in the "Service" tab of the Transmission Settings" dialog (see chapter 5.6.7.5).
11	This field contains all ECUs of the transparent gateway which will be ignored. ECUs can be added or removed to this field via drag & drop.
12	There are three options available: -No Offset: There is no synchronization between the two FlexRay clusters. In-Cycle Offset and Cycle Offset are not used. -In-Cycle Offset: Only the In-Cycle Offset is used (see item **8**). The Cycle Offset is not used. -Cycle/In-Cycle Offset: Both In-Cycle Offset, and Cycle Offset are used.
13	In FlexRay clusters the In-Cycle-Offset configures the offset of the synchronized clusters (gateway side A and B). 0/4166µs means that the left side (side A) is 4166µs **before** the right side (side B). Or in other words, the right side (side B) has a time offset of 4166µs to the left side (side A). Be careful with this option – if the configured offset is too short, the hardware may not be able to route the configured frames.
14	An additional Cycle Offset can be defined. A Cycle Offset of '0' (default) means, that the two FlexRay clusters are regulated to the same cycle. Example Configured Cycle Offset =,3': Current Cycle Con1: '8' Current Cycle Con3: '5'

No	Description
15	With the settings made in this section, the restart of the synchronization can be triggered. The following settings are possible: • Auto Sync-Restart is enabled, using an automatically calculated Sync Restart value. In this case the “User-defined Sync-Restart value” check box is not activated. • Auto Sync-Restart is enabled, and the Sync Restart value can be entered. • Sync-Restart is disabled If FlexRay cluster synchronization is active, and the absolute difference between the configured offset and the measured offset is equal or greater than the Sync Restart value and the SYNC-RESTART option is enabled, the synchronization will be restarted.
16	Standard Signal Functions: this is the same functionality as in chapter 5.6.5.

Table 125: Transparent Gateway configuration dialog

	Information
	If the transparent gateway is created for a FL3X System-M, then the connector list will also show the card index and card short name.

	NOTICE
	If the “Transparent Gateway” dialog is confirmed with *“OK”* the project configuration is updated. Frames which are used in user functions, are **not** ignored, even if their sender-ECU is put to the list “Ignored ECUs”.

	NOTICE
	If a service field has a setter with no response, the setter request is forwarded, and the response is ignored. When a service field has a setter with a response, two routings are created - one for the setter request and one for the setter response.

	NOTICE
	If the database contains frames, which have no sender ECUs, then these frames will be simulated on both sides of the transparent gateway (see chapter 5.6.2.1) Routings are also created for frames without sender ECUs. If such routings are unwanted, use the Gateway Editor to remove them. The handling of frames without sender ECUs also depends on the user define ‘USE_EVENT_TIMING_FOR_GW’ (see chapter 5.6.7).

6.2.6 FlexRay Synchronization used in a Transparent Gateway

Two FlexRay busses used in a Transparent Gateway are synchronized (if the option is not disabled). This synchronization is done via configuration and runtime control of the FlexRay communication controller of the slave side. The parameters, for this synchronization, are configured as described in Table 125, item **8**, **9** and **10**).

To achieve the configured synchronization and the chosen offset value, it is necessary for the device to initiate the FlexRay bus start-up. To guarantee this behavior, the maximum number of Sync/Start-Up frame sent by ECUs connected with the slave side of the gateway should be 1. If there are more than 1 Sync/Start-Up frames sent by other ECUs connected to the slave side of the gateway, the device is not able to control the configured offset.

This is the reason that the warning ‘Cycle synchronization cannot be guaranteed’ is shown in the Transparent Gateway editor, if the slave side has more than one external sync-node.

	Information
	A short explanation about the FlexRay offset correction: Each sync-node on a FlexRay bus evaluates drift values (difference between the real and expected cycle duration), The Fault Tolerant Midpoint (FTM) algorithm is used for the offset correction. All measured drift values are collected, sorted, and averaged (The lowest and highest values are not considered, if 3-7 sync-nodes exist. If more than 7 sync-nodes are available, then the 2 lowest and 2 highest values are ignored), The average is used for the offset correction. This process shows if there are fewer external ECUs, there is a better chance for the device to correct the offset. If 'Sync-Restart' is enabled (see Table 125) and the maximum of one external sync-node is present, then the slave side of the device can stop the synchronized network and restart it again. This is done multiple times until the configured offset is achieved. See also chapter 5.6.6.2.

6.3 Differences between RBS and Gateway

An RBS always transmits the default signal values defined in a FIBEX database, or the signal values configured by the user in the "Default Values Editor" (see chapter 5.4.2.3).

A (Transparent) Gateway routes only signals with defined gateway routings. The signal routings are based on events. Only received signals are routed.

In both cases transmitted signal values can be affected by the User Functions (see chapter 5.8), signal manipulations (see chapter 7.1) and by the user defined standard signal calculation algorithms (see chapter 5.6.4).

There is a predefined processing order. Please note that the standard signal handling always affects the signal values at last.

	NOTICE
	**RBS:** RBS<DefaultValue> ->TX User Function -> Signal Manipulation -> Standard Signal Handling -> RBS_TX_on_Bus. **Gateway:** GW_RX_on_Bus -> RX User Function -> Gateway routing + scaling -> TX-User Function -> Signal Manipulation -> Standard Signal Handling -> GW_TX_on_Bus. If a SecOC configuration is part of the project, please see chapter 5.10.

Due to the different possibilities concerning the standard signal calculation, the manipulation of a signal eventually effects the standard signal calculation.

The only exception is Standard Signal Handling in combination with manipulation. If a manipulation is activated for a signal that also has a configured signal calculation (Standard Signal Handling), then the execution of the signal calculation will automatically be switched off during the manipulation.

7 FL3X Config Control

7.1 Manipulations and Sequences

Manipulations are used to manipulate transmitted entities such as signals, PDUs, frames or global variables in different ways. The activation and deactivation of manipulations may depend on definable events.

To configure manipulations and sequences, you must open the “FL3X Config Control” dialog. This can be done via the menu *‘Edit -> FL3X Config Control’*.

The graphical user interface is divided into two main sections (see Figure 211). The top section allows to create, delete, update and to run manipulations. The bottom section is used to create groups of manipulations (sequences) and to execute such a group.

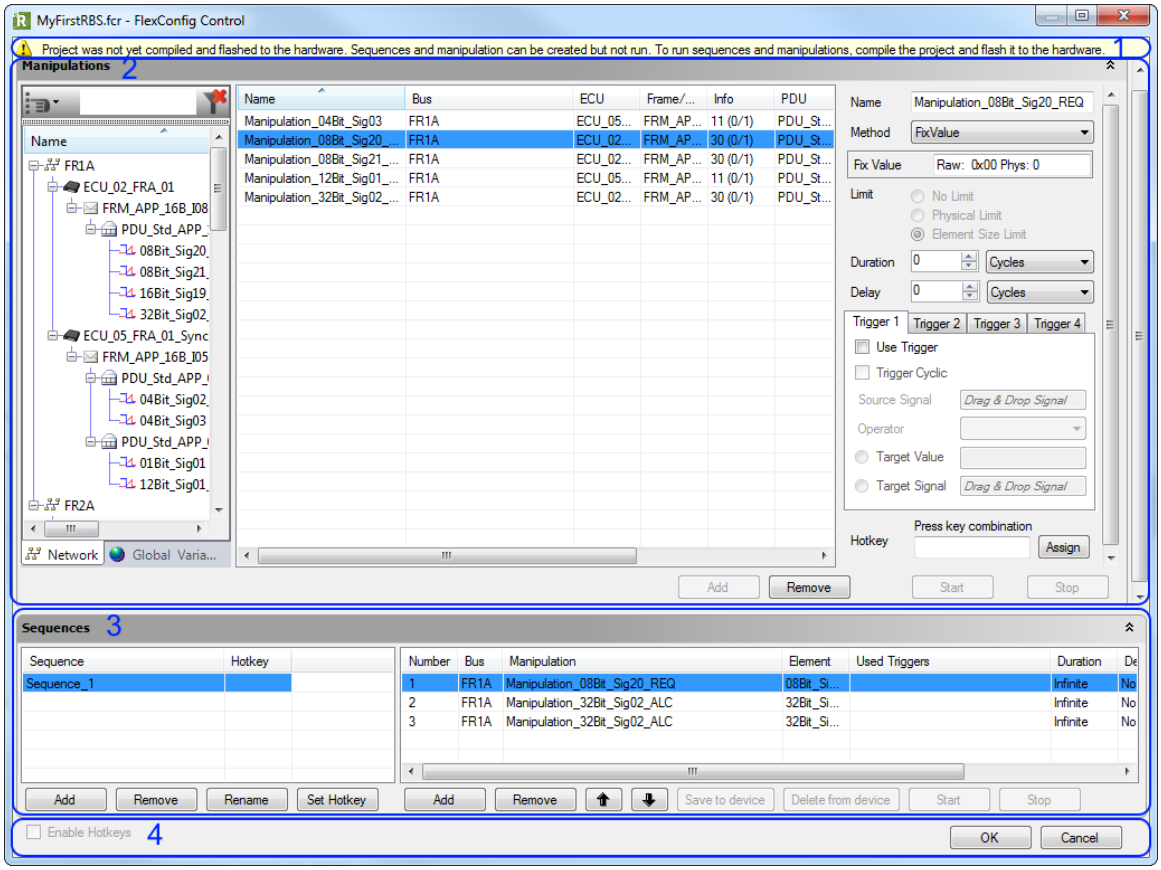


Figure 211: FL3X Config Control

No	Description
1	In this section of the “FL3X Config Control” editor different warnings are displayed. For example, if a project has not yet been built and flashed to the project the warning seen in Figure 211 is displayed. Another example: If a manipulation sequence is already or still running on the device and “FL3X Config Control” is started, then the warning “*A manipulation or sequence is already active. Cannot resolve which manipulation or sequence is active*” is displayed.
2	Section to configure and execute manipulations.
3	Section to configure and execute sequences (see chapter 7.1.10)
4	Enables/Disables Hotkey detection which allows starting sequences and manipulations via shortcut (see chapter 7.1.11) This option is **not** saved when the dialog is closed via the button *“OK”*.

No	Description
	Information
	The FL3X Config Manipulation Executor is a separate tool, which is also installed with FL3X Config. It can be used to load and execute the manipulations. The FL3X Config Manipulation Executor does not need to load the bus databases, which make the load time very short. Because of this, changes to manipulations is very limited. Adding of manipulations is not supported for FL3X Config Manipulation Executor. Please read the “3-01140A01-D015-FL3X Config Manipulation Executor – User Manual” [10] for more information.

	NOTICE
	Do not open a FL3X Config Project (**.fcr*) with FL3X Config and FL3X Config Manipulation Executor at the same time.

7.1.1 Manipulations

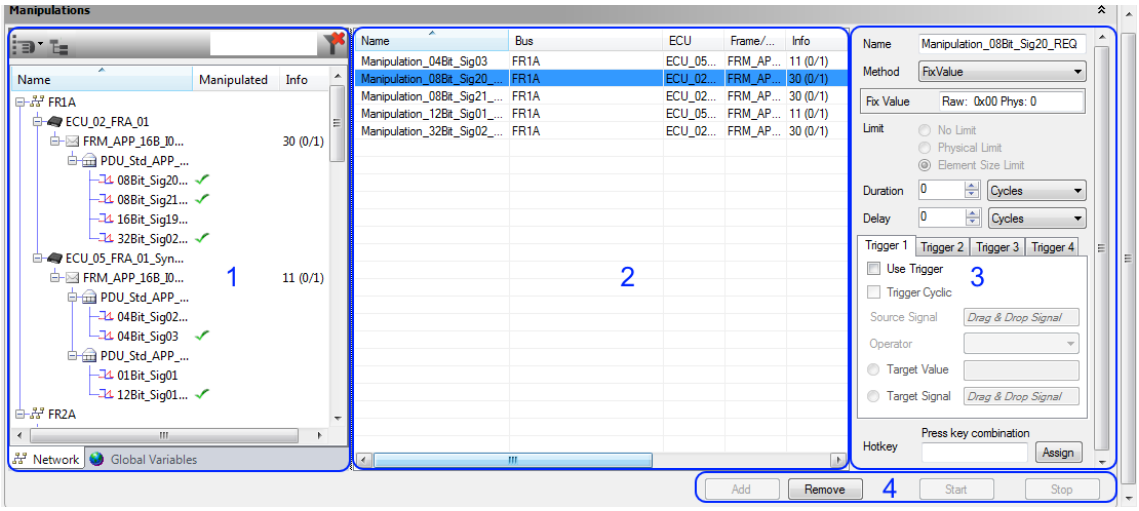


Figure 212: FL3X Config Control – Manipulation control

The manipulation section is divided into four parts (see Figure 212).

No	Description
1	Manipulable Entities: Shows which elements can be manipulated. The elements are grouped into *‘Network’’* and *‘Global Variables’*. Each group has the following columns: - Name: Name of the network element or the global variable - Manipulated: Shows, whether a manipulation exist for the element - Info: Additional information for the frame
2	List of configured manipulations: Shows an overview of already configured manipulations.
3	Manipulation configuration: Allows to configure the selected manipulation. The options are described later.
4	Manipulation actions: Allows to perform an action on the selected manipulation such as adding, deleting or start/stop.

	NOTICE
	Manipulations for signed signals other than 8, 16 and 32 bits can lead to unexpected results. See Troubleshooting chapter 12.3 FL3X Config Control (Point 2 Manipulations for signed signals have unexpected results) for more information

7.1.1.1 Toolbar



Figure 213: FL3X Config Control – Toolbar for the “Manipulable Entities” section

No	Description
1	View button: Over the view button you can choose which source cluster elements are shown
2	Expand/Collapse button expands or collapses all source elements
3	Text-Filter field: Only elements containing the entered text are shown in the list
4	Clear button resets the text-filter

Table 126: FL3X Config Control – Toolbar of the “Manipulable Entities” section

7.1.1.2 Display Items

The cluster elements, which are shown in ‘Manipulable Entities’ list, depend on the “Gateway Editor View” which is selected for the project (see chapter 5.22.2.1). Following “Gateway Editor View” modes are available:

- PDU-View
- Frame-View

Frames will not be shown, if the PDU-View is activated. Just one instance of multiple instantiated PDUs will be shown in the PDU-View.

	Information
	Service items (events, fields, methods) and data types are shown for both views.

To display the desired cluster elements, you can use the “*View*” button of the toolbars (source /target sections, see Figure 186).

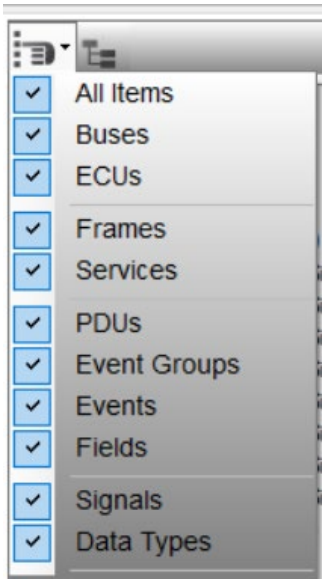


Figure 214: FL3X Config Control – Cluster view button

	NOTICE
	Subsequent PDUs (static PDUs and switched PDUs) of multiplexed PDUs are not supported for manipulations!

NOTICE
If a standard signal is manipulated, then a new value will not be calculated (manipulated value will be used as the signal value).

7.1.1.3 Add a Manipulation

A manipulation can be added in three ways:

Via drag & drop: Select an entity for which a manipulation should be defined and hold the left mouse button and move the mouse pointer to the manipulation table and then release the left mouse button.

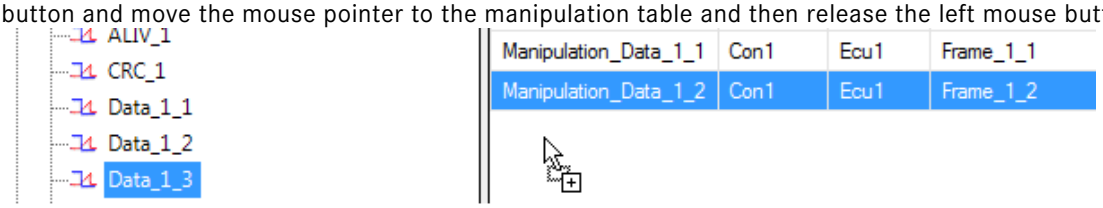


Figure 215: FL3X Config Control – Create a manipulation via drag & drop

Via *"Add"* button: Select an entity for which a manipulation should be defined and click on the button *"Add"* (within the section "Manipulations").

Via context menu: Select an entity for which a manipulation should be defined and right click to open the context menu. You can also just select the item and click on the "Insert" button

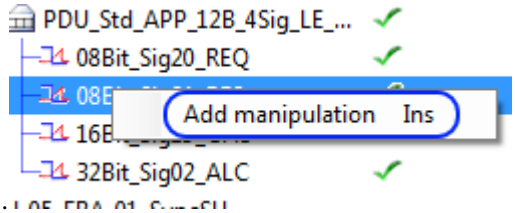


Figure 216: FL3X Config Control – Create a manipulation via context menu

Once the manipulation was added, it will be automatically selected, and the manipulation settings can be configured.

Please note that not all entities can be used for a manipulation. For details on this restriction see chapter 7.1.3.

7.1.1.4 Remove a Manipulation

To remove a manipulation, select the manipulation in the list of configured manipulation and click on the button *"Remove"*.

You can also open the context menu and click on 'Remove manipulation' element, to delete the selected manipulation. Another alternative is to select the manipulation and press "Delete" key.

If the manipulation is used within one or more sequences, the manipulation will also be removed from the sequence(s).

7.1.1.5 Clone a Manipulation

In some cases, it can save time to make a copy of an existing manipulation and modify it, instead of creating a new one.

You can open the context menu and click on 'Clone manipulation' element to create a copy of the selected manipulation. The new item will have a postfix "_Copy" by its name. Each additional copy will add a number to the name.

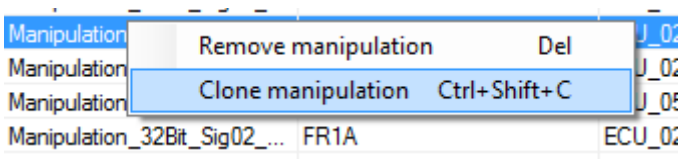


Figure 217: FL3X Config Control – Clone a manipulation via context menu

7.1.1.6 Configure a Manipulation

To configure a manipulation a manipulation must be selected first. The following settings can be configured (see Figure 218):

No	Description
1	The name of the manipulation. The default name is the concatenation of “Manipulation_” and the name of the entity (e.g. signal short name).
2	The manipulation method. Depending on the selected method different configurable options are available. All available manipulation methods and their options are described in more detail in chapter 7.1.3.1.
3	Depending on the manipulation method (“Factor”, “Offset”, “Drift”, “Ramp”) a “Limit” parameter can be set here. If the “Limit” parameter is set to “No Limit” an overflow can happen. Details are described in chapter 7.1.2.
4	Configures the duration (execution time) of the manipulation. A duration of ‘0’ means that the manipulation is executed infinitely. A complete description can be found in chapter 7.1.6.
5	Configures the delay. A complete description can be found in chapter 7.1.6.
6	Configures the trigger which is used. For a detailed description of this feature please see chapter 7.1.7.
7	Configures the hotkey which is used to start a manipulation. Starting a manipulation via hotkey is only possible if the *“Enable Hotkeys”* option is activated (see chapter 7.1.11).

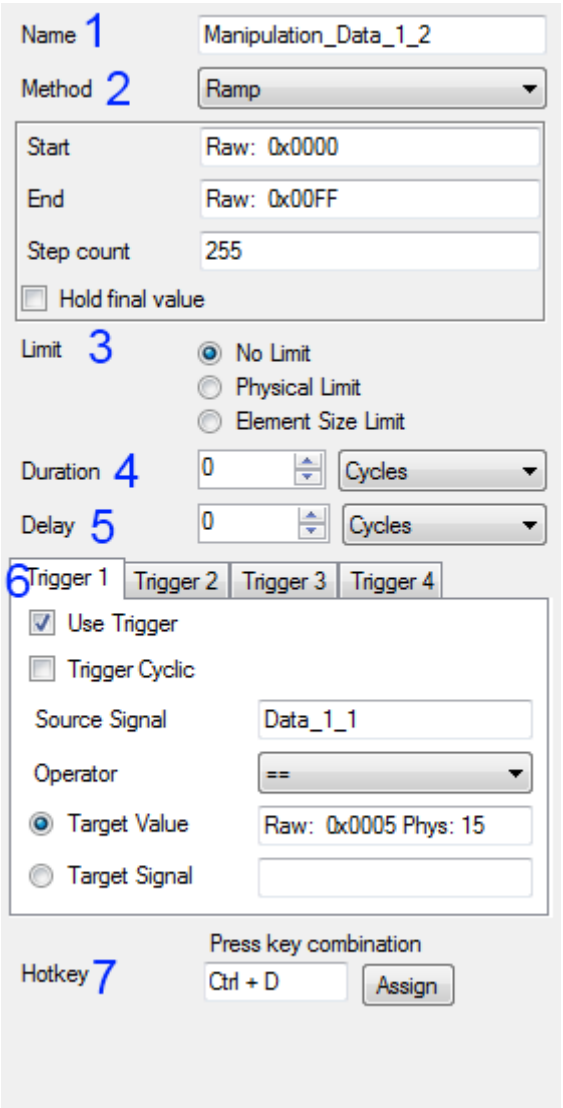


Figure 218: FL3X Config Control – Manipulation configuration options

7.1.1.7 Start a Manipulation

To start a manipulation, select the manipulation and click on the button *"Start"*. Another alternative is to double-click on the manipulation.

To indicate that the manipulation is executed the visual appearance of this manipulation will be changed (see Figure 219 and Figure 220) Please note that only one manipulation can be executed at the same time. To execute multiple manipulations at the same time, use manipulation sequences (see chapter 7.1.10). An active manipulation can't be edited.

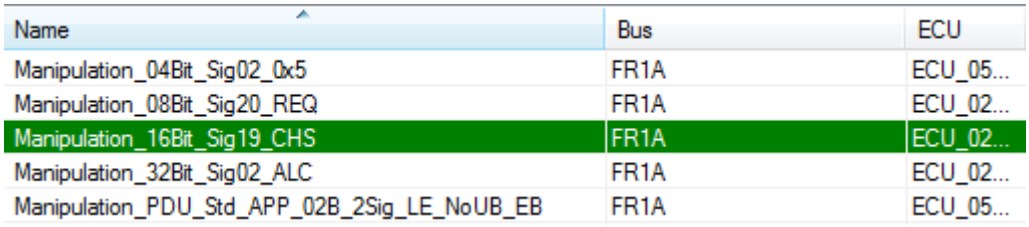


Figure 219: FL3X Config Control – Manipulation is running and is selected

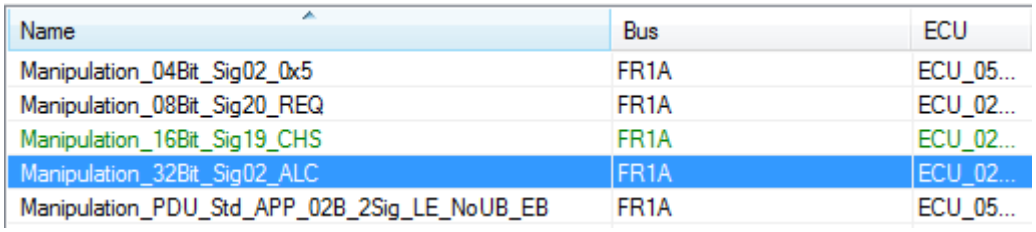


Figure 220: FL3X Config Control – Manipulation is running and is not selected.

7.1.1.8 Stop a Manipulation

To stop a manipulation, just select the manipulation and click the *"Stop"* button. Another alternative is to double-click on the running manipulation.

7.1.1.9 Save a Manipulation to the Device

It is not possible to save a single manipulation on the device. The manipulation must be added to a manipulation sequence first (see chapter 7.1.10.9). All saved manipulations are automatically activated after power on. A separate activation of the saved manipulations via FL3X Config Control is not necessary.

7.1.2 Limits for Signal Values

Manipulation types such as ramp and drift allow a value restriction.

In this example, a ramp is configured for an 8-bit signal with 0 – 500 in 50 steps.

No Limit: This means that both over-and under-flows may happen
(in Example: 0, 10, 20 … 250, 4, 14 …244)

Physical Limit: limits the signal to the predefined value from the database
(in Example: 10, 20, 30 … 240, 245, 245 …245)

Element Size Limit: limits the signal to the Minimum / maximum value that can be achieved by this signal (data type dependent).
(in Example: 0, 10, 20 … 250, 255, 255 …255)

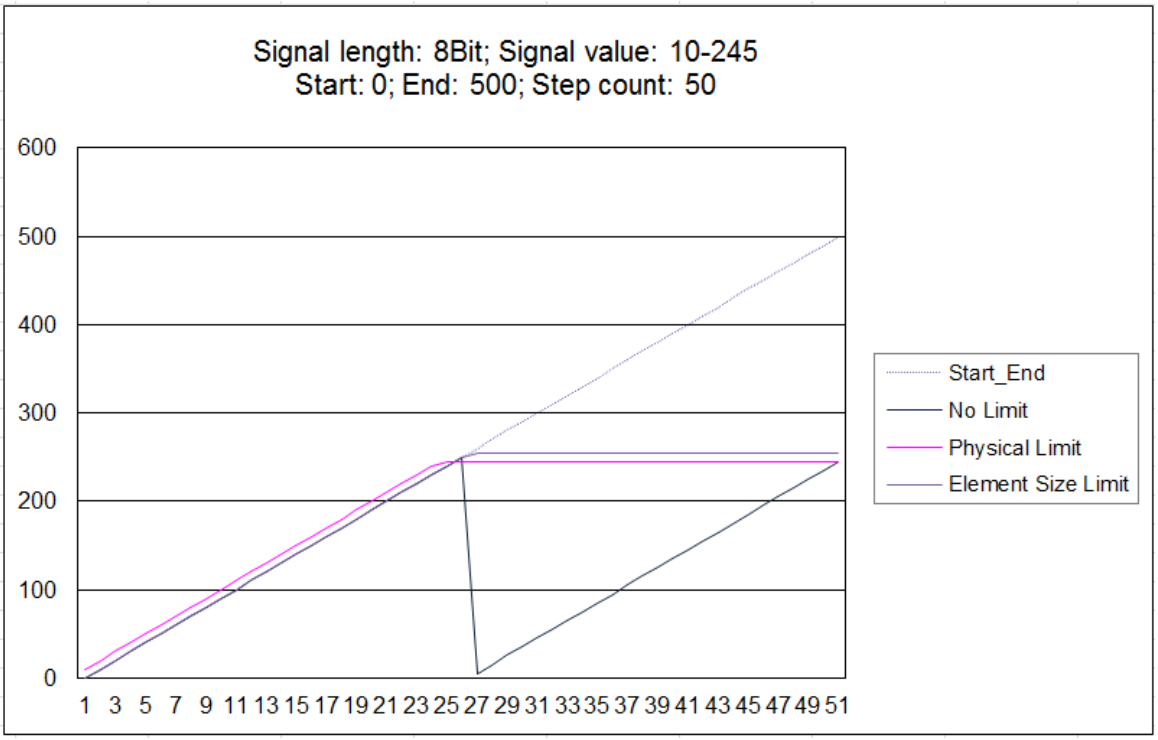


Figure 221: Manipulations – Example for value limits

The following table describes the limits, which are supported for the different signal manipulations. If several limitations are together, then in each cell is indicated which is determining for the manipulation limit.

Manipulation	No Limit	Physical Limit	Element Size Limit
Loss (Signal)	X (No Limit)		
Factor (Signal)	X	X	X
Freeze (Signal)	X (No Limit)		
Fix Value (Signal)	X (Element Size Limit)		
Offset (Signal)	X	X	X
Drift (Signal)	X	X	X
Ramp (Signal)	X	X	X
Disable M. (Signal)	X (No Limit)		
Global Variable	X	X (Element Size Limit)	
Switch	X	-	X

Table 127: Manipulations – Value Limits depending on manipulation type

NOTICE

Manipulations are not available for the signals larger than 32 Bits.

7.1.3 Available Elements for Manipulation

Not all manipulation methods can be applied to every element. The following table shows the elements and their supported manipulation methods. Please note that some elements may be greyed out. This means that they cannot be used for manipulation (e.g. if the signal bit length exceeds the 32-bit limit, see chapter 2.8.3).

Element	Manipulations
ECU	Loss, Disable Manipulation
Frame	Loss, Disable Manipulation
PDU	Loss, Disable Manipulation
Switch	Freeze, FixValue, Disable Manipulation

Element	Manipulations
Signal	Loss, Factor, Freeze, FixValue, Drift, Offset, Ramp, Disable Manipulation
Signal Update Bit	Loss, Freeze, FixValue, Disable Manipulation
Signal Group Update Bit	Loss, FixValue, Disable Manipulation
Global variables	FixValue Attention: Global variables must be defined in the “User Function Editor” (see chapter 5.8.3.3) before they can be manipulated.

Table 128: FL3X Config Control – Available elements for manipulation

7.1.3.1 Available SOME/IP Elements for Manipulation

The following manipulations are available for SOME/IP.

Element	Manipulations
Event	Loss, Disable Manipulation
Field	Loss, Disable Manipulation
Method – Call	Loss, Disable Manipulation
Method – Response	Loss, Disable Manipulation
Data Types	Loss, Factor, Freeze, FixValue, Drift, Offset, Ramp, Disable Manipulation

Table 129: FL3X Config Control – Available SOME/IP elements for manipulation



NOTICE

Only UINT8, UINT16, and UINT32 data types are supported.



NOTICE

Manipulations are not available for static/dynamic arrays or unions.

7.1.4 Manipulation Methods

The following chapter describes for each method the configuration options and the action which takes place if the manipulation is executed.



NOTICE

Manipulation methods have been developed for gateways. Manipulations, which are performed for an RBS, may differ from the desired / expected behavior.

7.1.4.1 Loss

Allow to stop transmitting entities. Depending on the manipulated element the actions differ slightly.

Element	Action
ECU	All frame transmissions from this ECU are stopped. However, if the ECU sends a sync or Sync/Start-Up frame, the frame will be still sent as a NULL frame to guarantee the synchronization of the FlexRay cluster.
Frame	The transmission of the frame is stopped (FlexRay frames in dynamic segment, CAN frames, LIN frames). If the manipulated frame is a FlexRay frame in static segment, NULL frames are sent instead of the frame.
PDU	The frame with the PDU is transmitted as configured. But all bytes of the PDU are set to ‘0x0’.
Signal	The frame with the signal is transmitted as configured. But all bits of the signal are set to ‘0x0’.

Element	Action
Signal Update Bit	The Signal Update Bit is set to 0x0.
Signal Group Update Bit	The Signal Group Update Bit is set to 0x0.
Switch	Not supported
Global Variable	Not supported

Table 130: Manipulation ‘Loss’ – Effect on manipulated elements

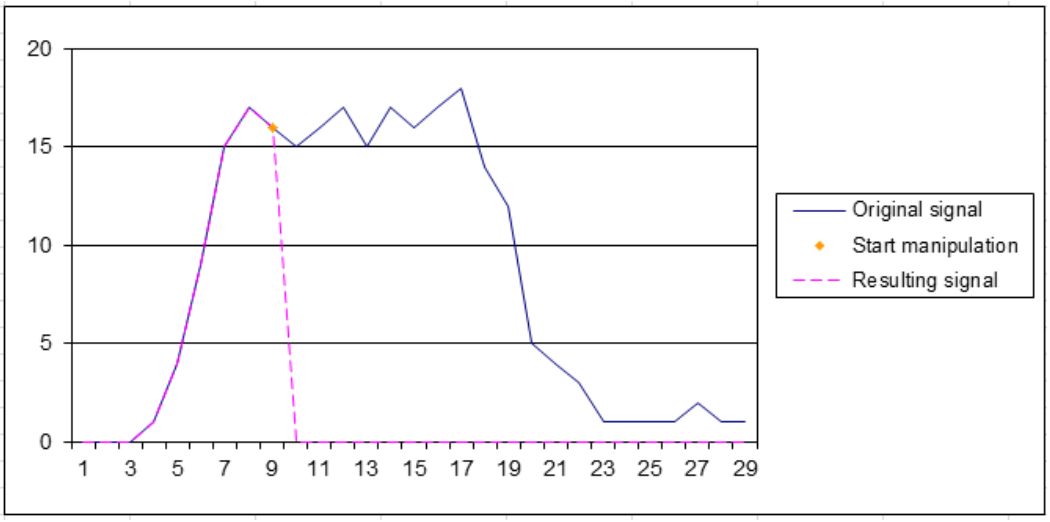


Figure 222: Example: Manipulated signal with method “Loss”

NOTICE

If PDU Update Bits are used and a signal manipulation with the method “Loss” for a PDU is executed, the PDU Update Bits are also set to ‘0’.

Signal manipulation for Signal Update Bits is not supported.

7.1.4.2 Disable Manipulation

Stops all ongoing manipulations for the selected element and restores the original state for the element. The duration of this manipulation is always ‘0’.

NOTICE

The method “Disable Manipulation” stops all manipulations for an element completely! Even if a trigger is used in combination with this method, other manipulations for the affected element are no longer performed.

To manipulate an affected element again, **all** manipulations must be stopped and restarted again.

For example, this manipulation type can be used in a manipulation sequence in combination with a trigger condition. The trigger condition is used to enable/disable the manipulation without any interaction via FL3X Config Control.

Element	Action
ECU	All manipulations of this element are stopped. All frames with their original values are transmitted.
Frame	All manipulations of this element are stopped. The frame is transmitted with the original value.
PDU	All manipulations of this element are stopped. The PDU is transmitted with the original value.
Signal	All manipulations of this element are stopped. The signal is transmitted with the original value.
Signal Update Bit	All manipulations of this element are stopped. The signal Update Bit is transmitted with the original value.

Element	Action
Signal Group Update Bit	All manipulations of this element are stopped. The signal Update Bit is transmitted with the original value.
Switch	All manipulations of this element are stopped. The switch is transmitted with the original value.
Global Variable	Not supported

Table 131: Manipulation ‘Disable Manipulation’ – Effect on manipulated elements

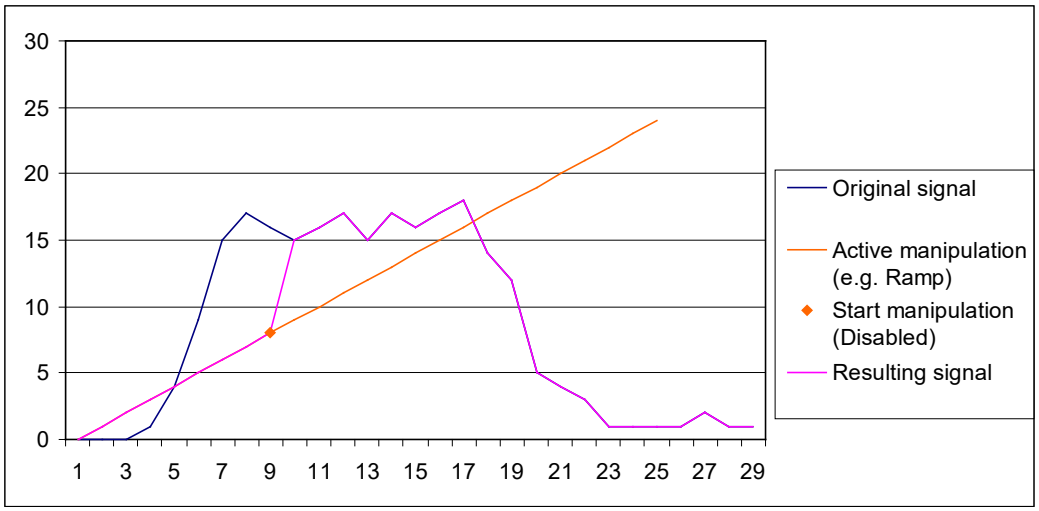


Figure 223: Example: Manipulated signal with method “Disable Manipulation”

7.1.4.3 Factor

The raw signal value is multiplied by the given factor. An overflow of the resulting raw value depends on the Limit configuration (see chapter 7.1.2). This means that the resulting raw value will be casted to the value range of the manipulated signal.

Element	Action
ECU	Not supported.
Frame	Not supported.
PDU	Not supported.
Signal	The raw or physical value of the signal is multiplied by the given factor.
Signal Update Bit	Not supported.
Signal Group Update Bit	Not supported.
Switch	Not supported
Global Variable	Not supported

Table 132: Manipulation ‘Factor’ – Effect on manipulated elements

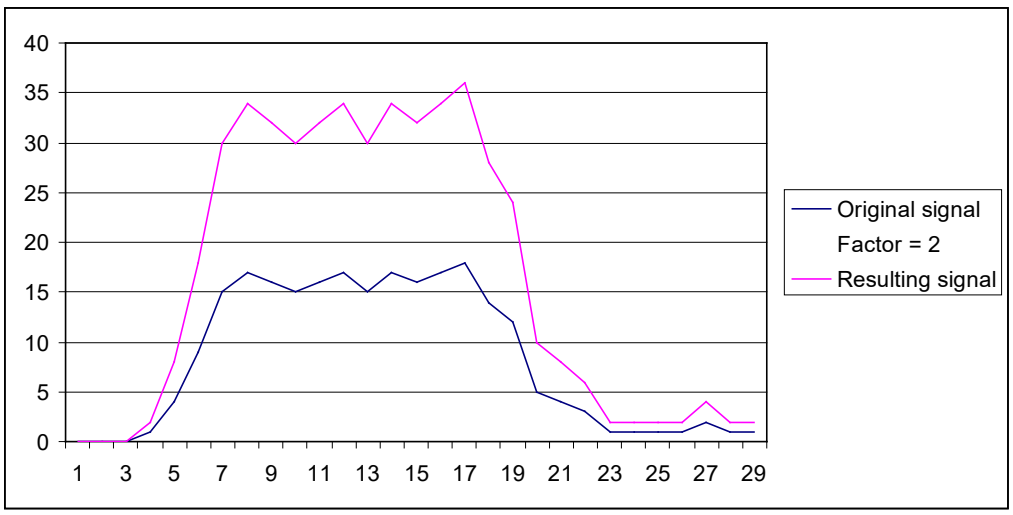
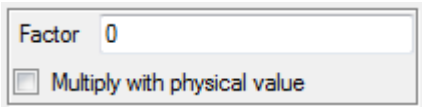


Figure 224: Example: Manipulated signal with method “Factor” (raw)

If the option “Multiply with physical value” is enabled, then the multiplication is performed with the physical signal value. If the option “Multiply with physical value” is not selected, the manipulation is performed with the raw value of the signal. The option “Multiply with physical value” is only available if there is a linear (identical) Compu-Method available for the affected signals.



	Information
	By a multiplication with physical values, the specified factor can be both a negative or positive integer, as well as a decimal number.

7.1.4.4 Freeze

The current signal value is frozen for the duration of the manipulation.

Element	Action
ECU	Not supported.
Frame	Not supported.
PDU	Not supported.
Signal	The current signal value is frozen for the duration of the manipulation.
Signal Update Bit	The current signal Update Bit value is frozen for the duration of the manipulation.
Signal Group Update Bit	Not supported.
Switch	The current switch value is frozen for the duration of the manipulation.
Global Variable	Not supported

Table 133: Manipulation ‘Freeze’ – Effect on manipulated elements

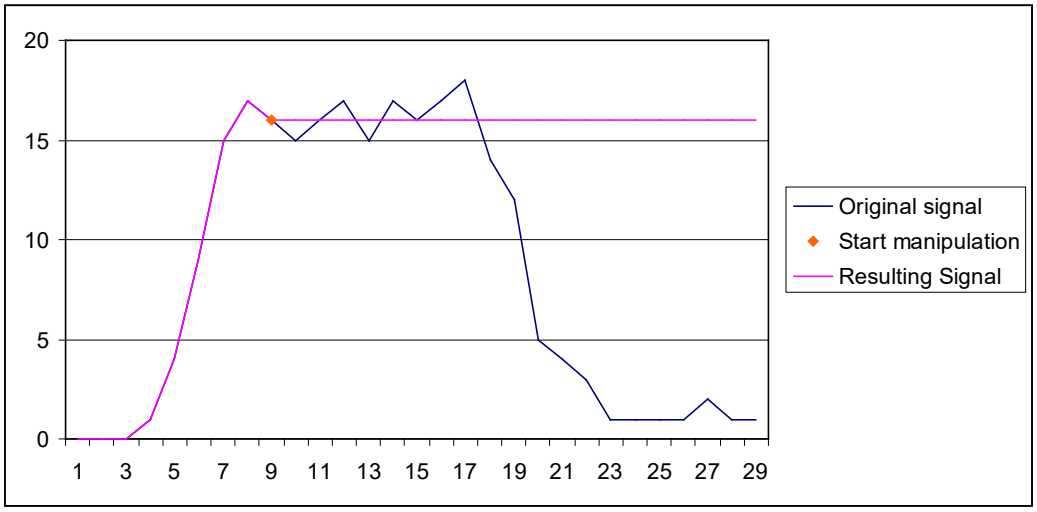


Figure 225: Example: Manipulated signal with method ‘Freeze’

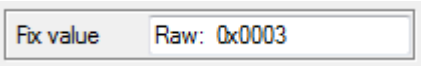
The following table shows another example of the manipulation ‘Freeze’ within a gateway (the values are not the same as in Figure 225).

Event counter	RX Signal value	TX Signal value with freeze (duration '1')	TX Signal value with freeze (duration '3')
1	1	1	1
2	2	2	2
3	3	3	3
Start manipulation command			
4	4	4	4
5	5	5	4
6	6	6	4
7	7	7	7
8	8	8	8

Table 134: Manipulation 'Freeze' – Example for a gateway

7.1.4.5 FixValue

This manipulation specifies a fix value for the manipulated element.



Element	Action
ECU	Not supported.
Frame	Not supported.
PDU	Not supported.
Signal	The signal value is sent with the configured value.
Signal Update Bit	The Signal Update Bit value is sent with the configured value.
Signal Group Update Bit	The Signal Group Update Bit value is sent with the configured value.
Global Variable	The global variable value is sent with the configured value. The duration of this manipulation is always "1 Event".
Switch	The switch value is sent with the configured value.

Table 135: Manipulation 'FixValue' – Effect on manipulated elements

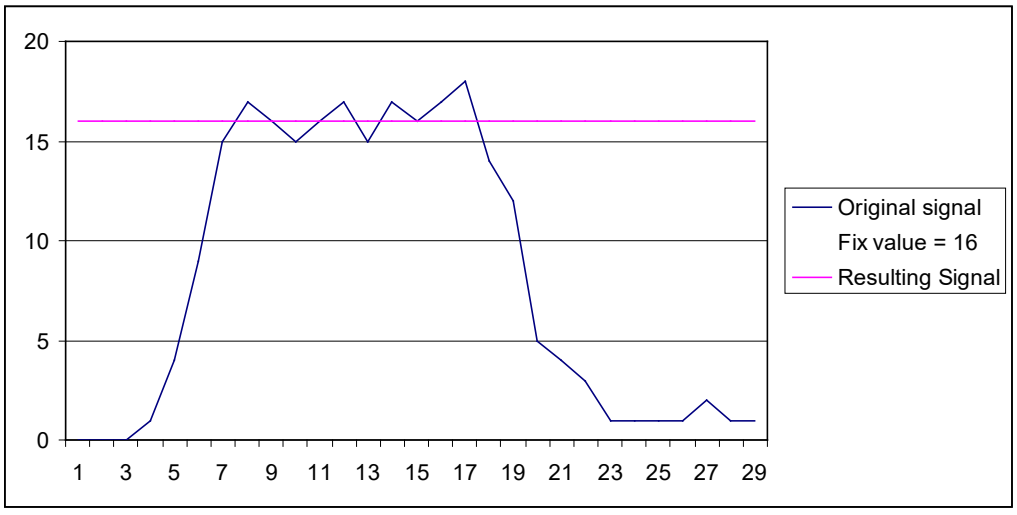
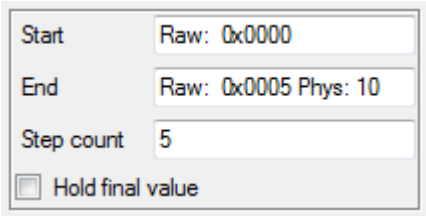


Figure 226: Example: Manipulated signal with method "FixValue"

7.1.4.6 Drift

This manipulation specifies a drift for the manipulated element.



Start: The start value of the drift.

End: The end value of the drift.

Step count: The count of steps to go from start to end.

Hold final value: The end value of the drift is hold until the manipulation ends.

Element	Action
ECU	Not supported.
Frame	Not supported.
PDU	Not supported.
Signal	A drift is added to the signal value before it is sent.
Signal Update Bit	Not supported.
Signal Group Update Bit	Not supported.
Global Variable	Not supported

Table 136: Manipulation ‘Drift’ – Effect on manipulated elements

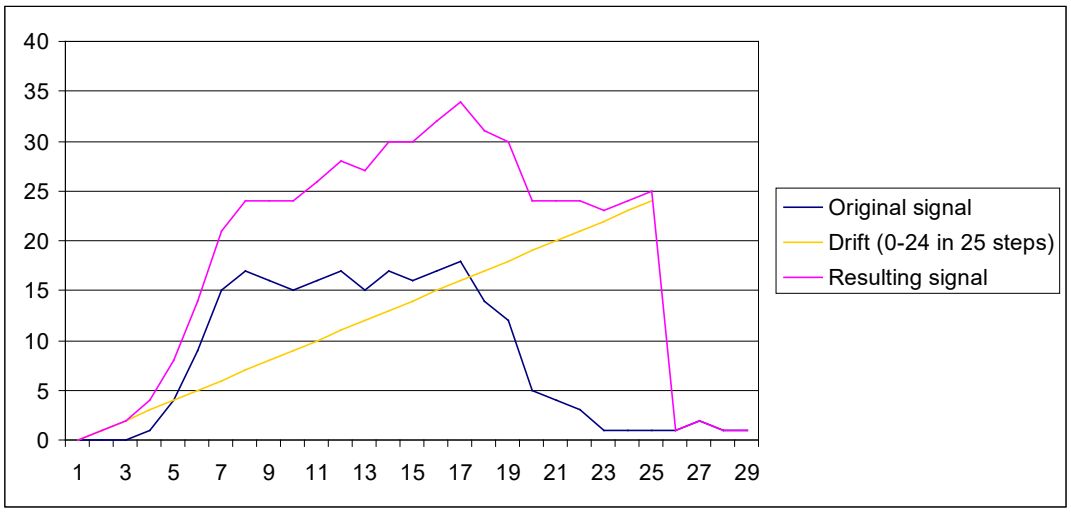


Figure 227: Example: Manipulated signal with method “Drift” (Duration: 25, Delay: 0, Hold final value: not active)

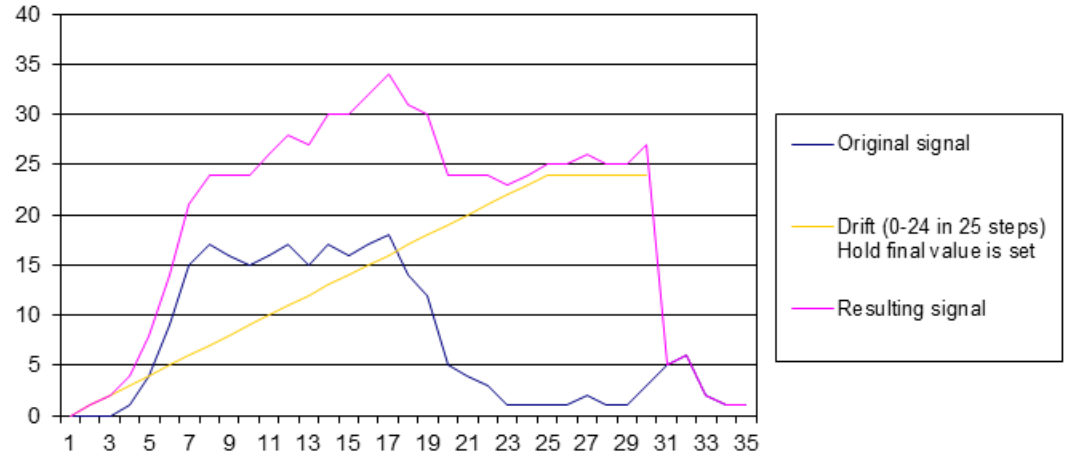


Figure 228: Example: Manipulated signal with method “Drift” (Duration 30, Delay: 0, Hold final value: active)

NOTICE

It is not possible to set the “step count” to ‘0’ via the FL3X Config Control.

If a manipulation “drift” is created with “step count” = ‘0’ via the “PC-Hardware-Interface”, then the manipulation “drift” behaves like a manipulation “offset” with the ‘End’ value.

NOTICE

The parameter “Duration” defines the number of events with active manipulation.

The parameter “Step count” defines the number of steps between the start value and the end value and is used to calculate the step width. Therefore, the step width is calculated as follows:
step width = (End value – Start value) / (Step count)

The resulting manipulation also considers the start value as a step. Thus, a drift from 0 to 24 with the step width 1 requires a “Duration” of 25 steps.

See the following examples.

Examples for step count

Start: 10 End: 40 Step count: 3 Duration: 4 Delay: 1

Event (n)	1	2	3	4	5
Step	0 This is not a step of the drift, it’s part of the delay	1	2	3	4
Drift value	0	10	20	30	40
Incoming signal value	5	15	25	35	45
Manipulated signal value	0	25	45	65	85

Start: 10 End: 30 Step count: 2 Duration: 6 Delay: 0

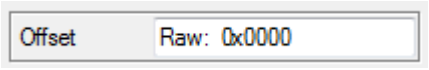
Event (n)	1	2	3	4	5	6
Step	1	2	3	1	2	3
Drift value	10	20	30	10	20	30
Incoming signal value	5	15	25	35	45	55
Manipulated signal value	15	35	55	45	65	85

Start: 10 End: 30 Step count: 2 Duration: 6 Delay: 0 Hold final value: active

Event (n)	1	2	3	4	5	6
Step	1	2	3			
Drift value	10	20	30	30	30	30
Incoming signal value	5	15	25	35	45	55
Manipulated signal value	15	35	55	65	75	85

7.1.4.7 Offset

This manipulation specifies an offset to the manipulated element.



NOTICE

Positive and negative values are possible as offset. But negative offsets can be only set for physical values.

If no value limitation (“No Limit”) is configured and an under-or overflow take place by the offset, then the under-or overflow always refer to the data type of the signal (e.g. unsigned 16-bit). Possibly physical limits defined for the signal are not considered.

The default value for the limit is “No limit”.

Element	Action
ECU	Not supported.
Frame	Not supported.
PDU	Not supported.
Signal	An offset is added to the signal value before it is sent.
Signal Update Bit	Not supported.
Signal Group Update Bit	Not supported.
Global Variable	Not supported

Table 137: Manipulation ‘Offset’ – Effect on manipulated elements

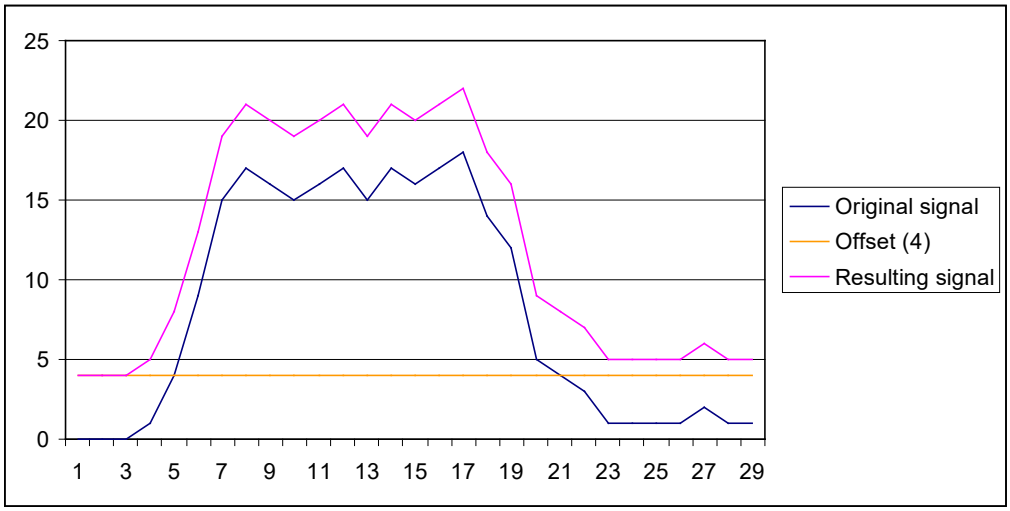
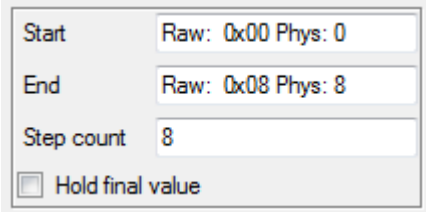


Figure 229: Example: Manipulated signal with method “Offset”

7.1.4.8 Ramp

This manipulation specifies a ramp for the manipulated element.



Start: Specifies the start value of the ramp.

End: Specifies the end value of the ramp.

Step count: Specifies the count of steps to go from start to end.

Hold final value: The end value of the ramp is hold until the manipulation ends.

Element	Action
ECU	Not supported.
Frame	Not supported.
PDU	Not supported.
Signal	A ramp is sent as signal value.
Signal Update Bit	Not supported.
Signal Group Update Bit	Not supported.
Global Variable	Not supported

Table 138: Manipulation ‘Ramp’ – Effect on manipulated elements

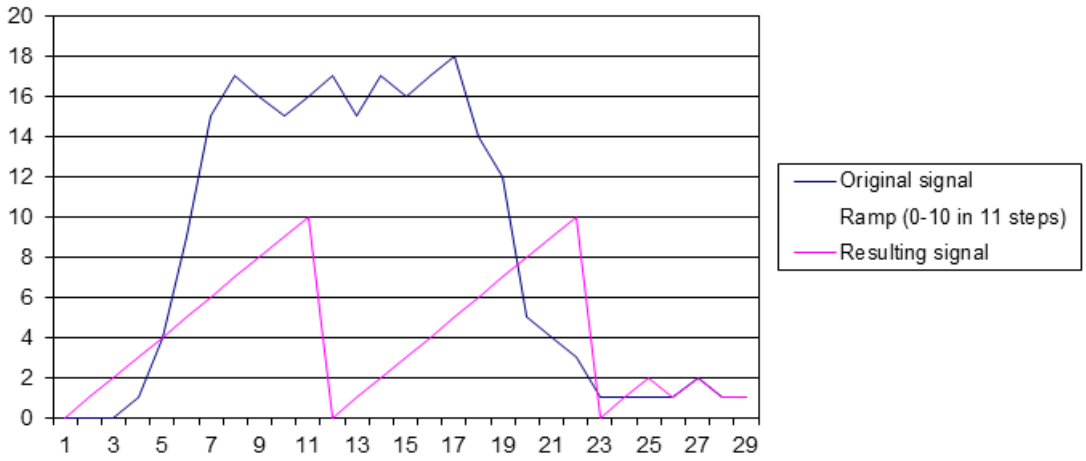


Figure 230: Example: Manipulated signal with method “Ramp” (Duration: 25, Delay: 0)

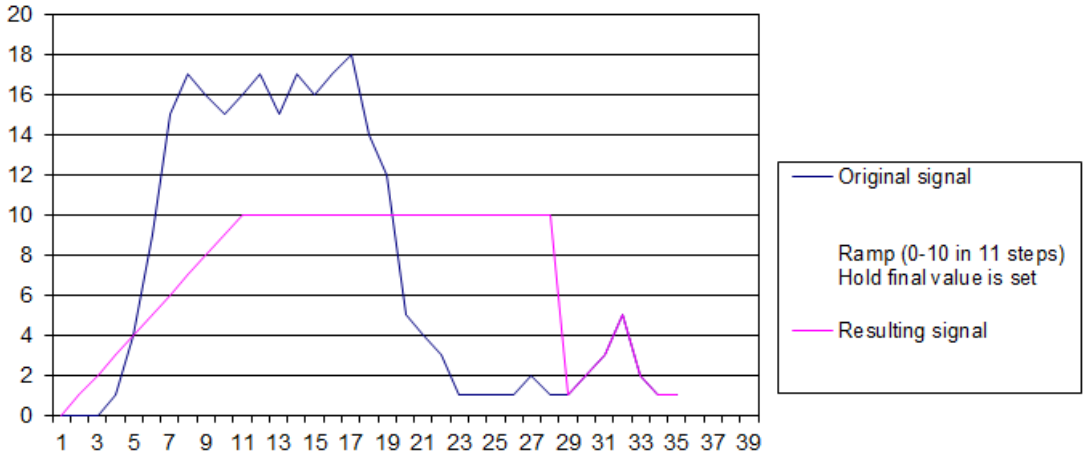


Figure 231: Example: Manipulated signal with method “Ramp” (Duration: 28, Delay: 0, Hold final value: active)

NOTICE

It is not possible to set the “step count” to ‘0’ via the FL3X Config Control.

If a manipulation “ramp” is created with “step count” = ‘0’ via the “PC-Hardware-Interface”, then the manipulation “ramp” behaves like a manipulation “FixValue” using the ‘End’ value.

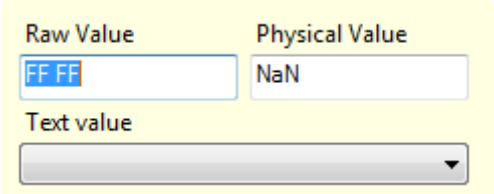
	NOTICE
	The parameter “Duration” defines the number of events with active manipulation. The parameter “Step count” defines the number of steps between the start value and the end value and is used to calculate the step width. Therefore, the step width is calculated as follows: step width = (End value – Start value) / (Step count) The resulting manipulation also considers the start value as a step. So, a ramp from 0 to 24 with the step width 1 needs a duration of 25 steps. See the following example.

Example for step count

Start: 0 End: 20 Step count: 4 Duration: 5 Delay: 0

Event (n)	1	2	3	4	5
Manipulated signal value	0	5	10	15	20

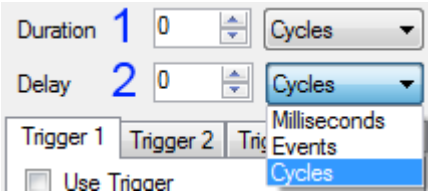
7.1.5 RAW Value Editor



The user can enter a raw or a physical value of a signal. The “Raw Value” is displayed as a hexadecimal value. The “Physical Value” is displayed as a decimal value. The “Text Value” shows special meanings of the value (as defined in the database) or the matching TEXTTABLE content if available.

Base of the conversion between raw and physical value is the Compu-Method defined in the FIBEX file. If there is now Compu-Method defined for the signal, raw value and physical value are the same.

7.1.6 Duration and Delay



No	Description
1	The parameter “Duration” defines the number of events/of cycles/of milliseconds the manipulation is active. “Cycles” is equivalent to FlexRay Cycles. Events are “transmit events” of the corresponding entity. For example, if a FlexRay frame in static segment is configured with base cycle ‘0’ and cycle repetition ‘2’, the matching event for the manipulation is every second cycle. A duration of ‘0’ means infinite number of events. The manipulation does not stop automatically. It must be stopped via the *“Stop”* button or by activating another single manipulation. For FlexRay clusters, the duration can be set to ‘events’, ‘cycles’ or ‘milliseconds’. For CAN clusters, only ‘milliseconds’ and ‘events’ are available. In case of ECU manipulations in a FlexRay network the duration can be configured in ‘cycles’ or ‘milliseconds’ and in a CAN or LIN network only in ‘milliseconds’.
2	The parameter “Delay” defines the number of events the manipulation is delayed. The time base for the delay is the start of the manipulation. For FlexRay clusters, the delay can be set to ‘events’, ‘cycles’ or ‘milliseconds’. For CAN clusters, only ‘milliseconds’ is available. In case of ECU manipulations in a FlexRay network, the delay can be configured in ‘cycles’ or ‘milliseconds’ and in a CAN or LIN network only in ‘milliseconds’.

NOTICE

The manipulation is disabled for the duration of at least one event after the configured duration expires.

NOTICE

The duration and the delay are defined separately for each manipulation. If multiple manipulations are combined to a so called “manipulation sequence” (see chapter 7.1.10), the duration/delay configuration of each manipulation is anyway independent from the other manipulations.

NOTICE

Independent of the duration setting of a manipulation, the shown state within FL3X Config Control is only updated after a send event has taken place! E.g. the duration of a manipulation is only set to one event and only one send event occurs, the manipulation state will stay active (green)!

Example for Duration and Delay

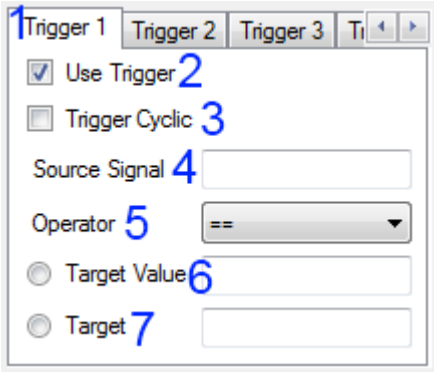
Manipulation A) Duration: 4 Delay: 0

Manipulation B) Duration: 3 Delay: 2

Event (n) n=1: start of manipulation	1	2	3	4	5	6
Manipulation A) active	X	X	X	X	-	-
Manipulation B) active	-	-	X	X	X	-

7.1.7 Trigger

A trigger can be defined to activate a manipulation depending on the trigger condition, for example if a signal value is equal to a raw value.



No	Description
1	Up to 4 triggers are available. Each trigger must be enabled with this option first. See chapter 7.1.8 for more details.
2	Activate this checkbox to enable trigger functionality for the current manipulation.
3	Cyclic check of the trigger condition (see chapter 7.1.9 for more details)
4	The “Source Signal” parameter defines the trigger source. Any signal that is transmitted on the same Connector as the manipulated entity can be selected. Each time the source signal is transmitted, its value is checked if it matches the trigger condition. The signal can **only** be selected via drag & drop from the manipulable items tree (see also chapter 7.1.1.2)!
5	The operator defines the comparison mode of the trigger. The following operators are available to create a trigger condition: “**Edge**”: Use this operator to implement a falling or rising edge. A falling edge corresponds to “Target Value” > “Hysteresis”. And a rising edge corresponds to “Target Value” < “Hysteresis” (see the Notice box at the end of this chapter) “**==**”: The “Source Signal” must be equal to the “Target Value”/”Target” “**!=**”: The “Source Signal” must not be equal to the “Target Value”/”Target” “**>**”: The “Source Signal” must be greater than the “Target Value”/”Target” “**<**”: The “Source Signal” must be smaller than the “Target Value”/”Target” “**>=**”: The “Source Signal” must be greater or equal than the “Target Value”/”Target” “**<=**”:The “Source Signal” must be smaller or equal than the “Target Value”/”Target”
6	Except for the operator “Edge”, all operators allow to select “Target Value” or “Target”. “Target Value” means that a raw value entered by the user is used for the trigger condition.
7	“Target” means that the value of the selected signal is used for the trigger condition. The signal can only be selected via drag & drop from the manipulable items tree (see also chapter 7.1.1.2)!

All manipulations have a TX-Event-based behavior. I.e. the verification of pre-configured trigger conditions, the calculation of delay and duration of a manipulation (regardless of the chosen time units for the delay and the duration) always take place right before and only for transmitted elements (signal / PDU / frame) which are to be manipulated. Therefore, it is only possible to detect if a trigger condition is fulfilled when the manipulated element is sent. If the trigger condition is fulfilled at the TX-event, then the calculation of the delay and / or duration of the manipulation start from this TX-event.

Information

Once a trigger source signal is received, its value is stored in the database of the device (FL3X Device-S, FL3X Device-L/L²). This value remains until a new trigger source signal is received.

Example:
Preconditions: A signal of a FlexRay frame in the dynamic segment is defined as the trigger source signal for the triggered signal manipulation. The operator "==" and a raw value entered by the user are used for the trigger condition. The duration and the delay of the signal manipulation are set to '0'. The signal manipulation is activated if the value of the trigger source signal is equal to the raw value entered in the *"Target Value"* field.

Action: A FlexRay frame in the dynamic segment was received only once and the trigger source signal value, e.g. '0x10', was stored in the database. If the target value of the trigger will now be set to '0x10' and the corresponding signal manipulation is started, the trigger condition will be fulfilled immediately. Even if the corresponding trigger source signal was received one hour ago.

NOTICE

The trigger configuration is defined separately for each manipulation. If multiple manipulations are combined to a so called "manipulation sequence" (see chapter 7.1.10), the trigger configuration of each manipulation is anyway independent from the other manipulations.
The exceptional case is if there are multiple manipulations for one signal. Then the order of the manipulations within the sequence affect which trigger condition will be used (see chapter 7.1.10.6).

NOTICE

If a signal value is used as trigger condition for a manipulation and there is a gateway routing for this signal, then the trigger condition is only checked when a routing event occurs.

Example:

- The Signal 'Sig1' is routed via FL3X Config Gateway from CAN 1 to CAN2.
- The Signal 'Sig2' is transmitted cyclic at CAN2
- There is an active manipulation for 'Sig2' (FixValue = 3) with the trigger condition 'Sig1 == 1'

Example:

Routing 'Sig1'	-	-	7	-	**1**	-	-
Output 'Sig2'	5	5	5	5	**3**	**3**	**3**

(After the 'Sig1' has the value '1', the trigger starts the manipulation.)

NOTICE

Signal Update Bits cannot be used as trigger source.

NOTICE
Three values are required to trigger an “edge” operator! E.g. for a rising edge, the source signal value must meet the following conditions: 1) Source signal < Target value 2) Target value < Source signal < Hysteresis 3) Source signal > Hysteresis

7.1.8 Linked Trigger Conditions

There are up to 4 triggers that can be linked together. If triggers are linked, each trigger condition needs to be fulfilled before the manipulation is active.

Example: Trigger 1 AND Trigger 2 AND Trigger 3 AND Trigger 4 become “true” if all their conditions are fulfilled.

Trigger 1 AND Trigger 2 are active				
Trigger 1	Trigger 2			Manipulation is active
True	False			False
True	True			True
Trigger 1 AND Trigger 2 AND Trigger 3 AND Trigger 4 are active				
Trigger 1	Trigger 2	Trigger 3	Trigger 4	Manipulation is active
True	False	True	False	False
False	True	True	True	False
True	True	True	True	True

7.1.9 Trigger Modes

Normally, a trigger condition is checked each transmit event of the corresponding entity (Signal / PDU / Frame). If the trigger condition is fulfilled, it is not checked any more until the manipulation ends. But if the option “Trigger Cyclic” is activated, it doesn’t matter if the condition is fulfilled, the check is preformed anyway.

Example: Trigger1 (Option “Trigger Cyclic” disabled) becomes true, if Signal1 = “0”;

Trigger2 (Option “Trigger Cyclic” enabled) becomes true, if Signal1 = “1”;

(Both Triggers are checking the same Signal)

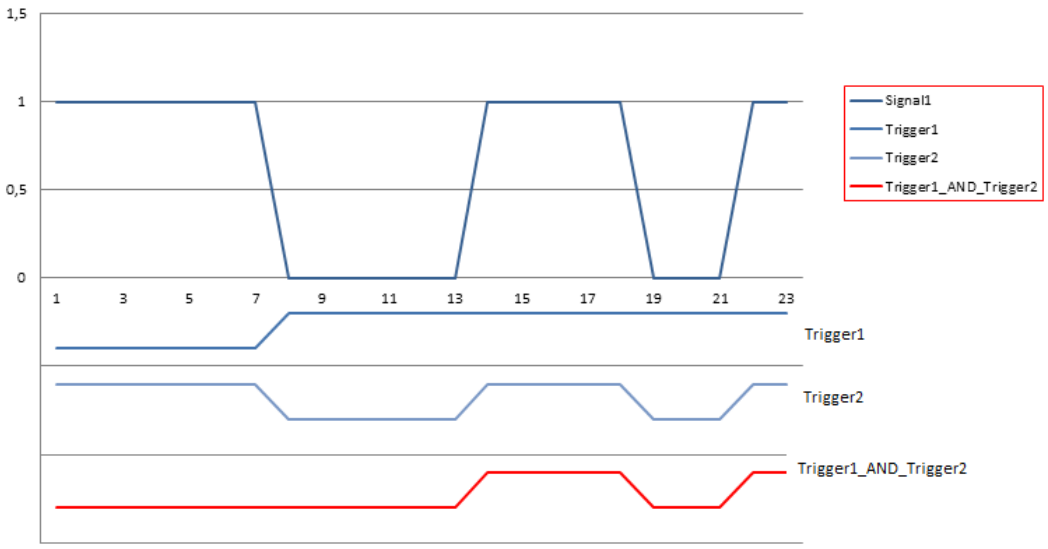


Figure 232: Example for the option “Trigger Cyclic”

7.1.10 Manipulation Sequences

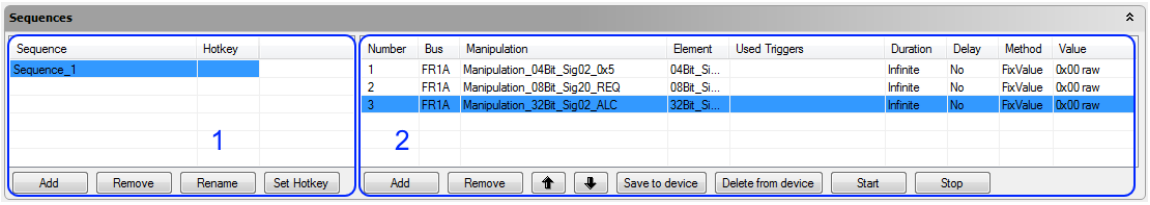


Figure 233: FL3X Config Control – Sequence control

No	Description
1	List of configured/stored sequences *"Add"* button: adds a new sequence (see chapter 7.1.10.1) *"Remove"* button: removes the selected sequence (see chapter 7.1.10.2) *"Set Hotkey"* button: Assign a shortcut to a sequence (see chapter 7.1.11)
2	List of manipulations of the selected sequence *"Add"* button: Adds the selected manipulation from the list of configured manipulations (see chapter 7.1.10.4) *"Remove"* button: removes the selected manipulation from the sequence (see chapter 7.1.10.5) Sort buttons (arrows up and down): moves the selected manipulation up and down (see chapter 7.1.10.6) *"Start"* button: starts the selected sequence (see chapter 7.1.10.7) *"Stop"* button: stops the active sequence (see chapter 7.1.10.8)

NOTICE

It is not possible to add more than 255 manipulations to a sequence.

7.1.10.1 Add a Sequence

To add a new sequence, the *'Add'* button must be pressed. Another alternative is to open the context menu and click on the 'Add new Sequence' element. Pressing the "Insert" key while the sequence list has the focus will also add a new sequence.

7.1.10.2 Remove a Sequence

To remove a sequence, the sequence must be selected and the '*Remove'* button must be clicked. Another alternative is to open the context menu and click on the 'Remove Sequence' element. Pressing the "Delete" key will also delete the selected sequence.

7.1.10.3 Clone a Sequence

In some cases, it can save time to make a copy of an existing sequence and modify it, instead of creating a new one.

You can open the context menu and click on 'Clone sequence' element to create a copy of the selected sequence.

If you select the option "Use same Manipulation", then the new sequence will be an exact copy of the original sequence (with the same manipulations).

If you select the option "Use copy of Manipulation", a copy of each manipulation of the original sequence will be created and the new sequence will contain these copied manipulations.

The new sequence will have a postfix "_Copy" by its name. Each additional copy will add a number to the name.

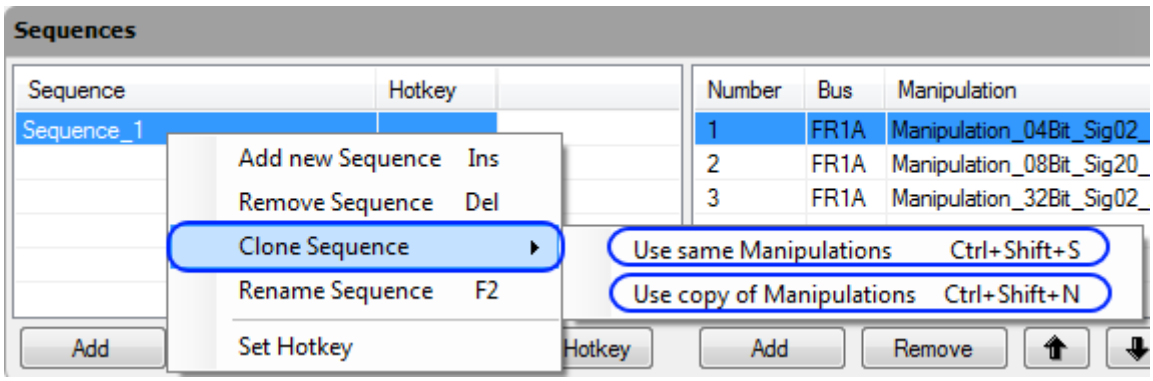


Figure 234: FL3X Config Control – Clone a sequence via context menu

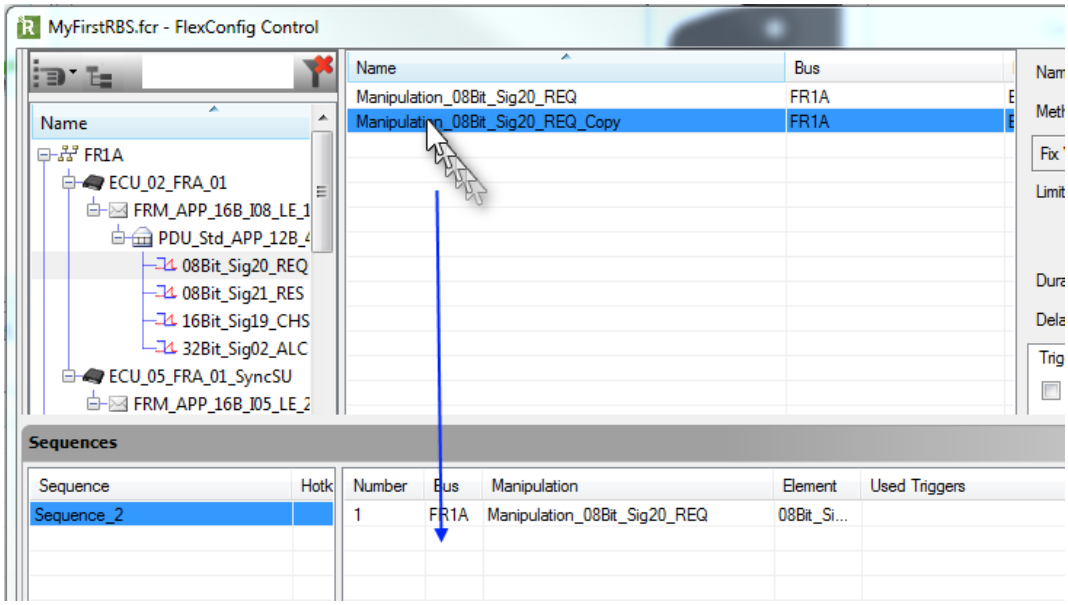
If the removed sequence was previously stored to the device (s. chapter 7.1.10.9), the sequence will remain in the memory of the device. After removing a previously stored sequence it is still possible to stop the sequence via the '*Stop*' button.

7.1.10.4 Add a Manipulation to a Sequence

To add a manipulation to a sequence, the relevant sequence must be selected first.

There are two different ways to add a manipulation:

Via drag & drop: A manipulation must be selected from the list of configured manipulations. The left mouse button must be held, and the mouse pointer must be moved to the sequences list. After that, the left mouse button must be released.



	Information
	If you keep the "Ctrl" key pressed during drag & drop, then a copy of the selected manipulation is added to the list of the manipulations, after the selected manipulation is added to the sequence

Via *'Add'* Button: The manipulation must be selected from the list of configured manipulations and the '*Add*' button of the "Sequences" section must be clicked.

The 'sequence' column by the manipulations list shows the sequences, to which the manipulation is added.

7.1.10.5 Remove a Manipulation from a Sequence

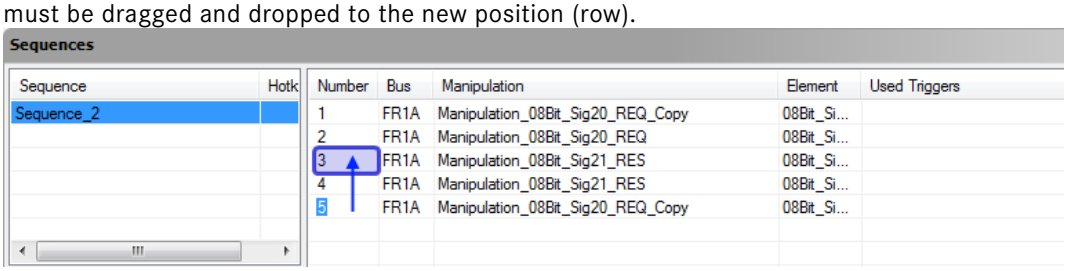
To remove a manipulation from a sequence, the manipulation of the sequence must be selected. After that, the 'Remove' button of the "Sequences" section must be pressed. Another alternative is to press "Delete" key after selecting the manipulation to be removed.

7.1.10.6 Sort Manipulations of a Sequence

The order of the manipulations within a sequence relate to the priority of each manipulation. The manipulation with the Number 1 (see Figure 233, column “Number”) has the highest priority. The priority is only relevant if there is more than one manipulation for a specific entity (signal, frame, etc.).

The manipulations of a sequence can be sorted in two ways:

Via drag & drop: A manipulation of the appropriate sequence must be selected. The selected manipulation must be dragged and dropped to the new position (row).



Via *“Up”* and *“Down”* Button: A manipulation of the appropriate sequence must be selected. After that, the ‘*Up*’ or ‘*Down*’ buttons can be pressed to place the manipulation at the correct place.

7.1.10.7 Start a Sequence

To start a sequence, the sequence must be selected and the ‘*Start*’ button must be pressed. Another alternative is to double-click on the sequence.

To indicate that the sequence is executed the visual appearance of this sequence and its manipulations will be changed (see Figure 235). Only one sequence can be executed at the same time. An active sequence cannot be edited.

The ‘*Start*’ button is greyed out if a manipulation is running on the device.

NOTICE

It is possible that a manipulation is running which has previously been saved on the device!

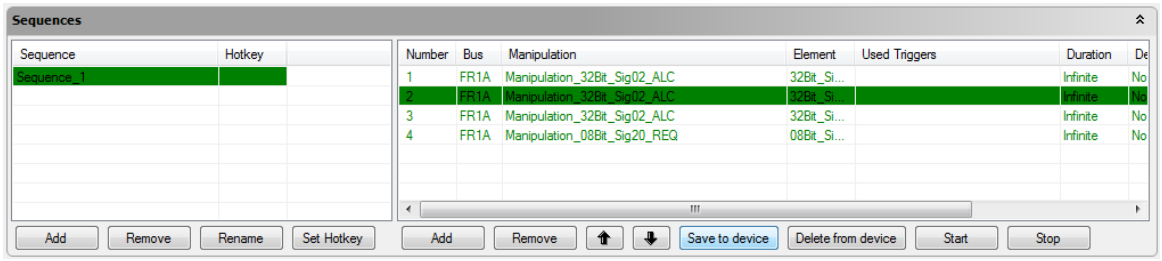


Figure 235: FL3X Config Control – Active sequence

7.1.10.8 Stop a Sequence

To stop a sequence, the sequence must be selected and the ‘*Stop*’ button must be clicked. Another alternative is to double-click the active sequence.

The ‘*Stop*’ button in the sequence area has the same effect as the ‘*Stop*’ button in the single manipulation area. All active manipulations running on the connected device are stopped.

7.1.10.9 Save a Sequence to Device

To save a sequence to the flash memory of the device (FL3X Device-L/L²) the sequence must be selected and after that the ‘*Save to device’* button must be pressed.
After the sequence has been saved to the device, the device restarts and the stored sequence runs immediately. If the device is restarted (e.g. by power off and on) the sequence stored in the flash memory will be restarted again.

NOTICE

The saving of a sequence to a device implicitly overwrites a previously stored sequence!

NOTICE

It is not recommended to save a sequence to the device if a manipulation is still running!

7.1.10.10 Delete a Sequence from Device

To delete all sequences from the device, the '*Delete from device'* button must be pressed.

7.1.11 Hotkey

It is possible to configure hotkeys for each manipulation and sequence.

The hotkeys are used to start a manipulation or sequence. Using a hotkey is only possible if the *"Enable Hotkeys"* option is activated.

NOTICE

The option "*Enable Hotkeys*" is deactivated when the dialog is closed (via *"OK"* or *"Cancel"*). If the "FL3X Config Control" dialog is reopened, the option must be set again.

7.2 Difference between Signal Manipulations within an RBS and a (Transparent) Gateway

The following listing assumes that no User Functions are used. However, if User Functions are used and if signal values are affected by these User Functions, the operand "FIBEX-DefaultValues" must be replaced by "User Function-Result" in the formulae.

RBS: A signal manipulation takes place on a "FIBEX-DefaultValue".
In case of the manipulation method "Factor" the value of a transmitted signal is set to "FIBEX-DefaultValue * FactorValue".
For the manipulation method "Offset" the signal value "FIBEX-DefaultValue + OffsetValue" is transmitted.
The manipulation method "Drift" sets the signal value to "FIBEX-DefaultValue + DriftValue".
In case of the manipulation method "Freeze" the "FIBEX-DefaultValue" is sent.

For the manipulation methods "FixValue", "Ramp" and "Loss" the behavior of the signal manipulation within an RBS is the same as for a (Transparent) Gateway.

NOTICE

A signal manipulation rewrites the signal values stored in the SODA-database of the device (FL3X Device-L/L²). For that reason, the signal value of the last signal manipulation will be sent by the RBS even though the manipulation execution is stopped.

(Transparent) Gateway: A signal manipulation always takes place on a value of the latest mapped signal value.

7.3 Special Case: Using the Same PDU Instance in Multiple Frame Triggerings

In case that a PDU instance is used in multiple frame triggerings, the manipulation behavior depends on the applied routings. The following example explains this:

- A User gateway routes PDU_1_A of the Frame_1_A from gateway side "A" to the PDU_1_B of gateway side "B".
- The PDU_1_B is transmit at the gateway side "B" in 4 different frames (frame triggerings) with the following order: Frame_1_B, Frame_2_B, Frame_3_B and Frame_4_B.
- All signals of the PDU_1_A are sent on gateway side "A" with the constant value '0'.
- A signal of the PDU "PDU_1_B" is manipulated.

	Input value	Defined routings (from PDU_1_A to ...)	Manipulation method	Manipulated value	Manip. Result after TX-Event 1	Manip. Result after TX-Event 2
PDU_1_A (Frame_1_A)	0	-	-	-	-	-
PDU_1_B (Frame_1_B)	-	X	Offset (+1)	0	1	1
PDU_1_B (Frame_2_B)	-	-	Offset (+2)	1	3	3
PDU_1_B (Frame_3_B)	-	-	Offset (+3)	3	6	6
PDU_1_B (Frame_4_B)	-	-	-	-	6	6

Table 139: Example for a PDU-Instance used in multiple frames (single routing)

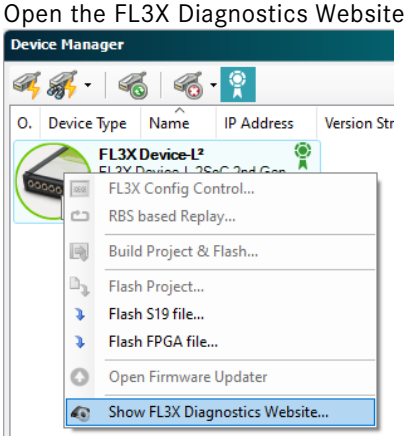
	Input value	Defined routings (from PDU_1_A to ...)	Manipulation method	Manipulated value	Manip. Result after TX-Event 1	Manip. Result after TX-Event 2
PDU_1_A (Frame_1_A)	0	-	-	-	-	-
PDU_1_B (Frame_1_B)	-	X	Offset (+1)	0	1	1
PDU_1_B (Frame_2_B)	-	X	Offset (+2)	0	2	2
PDU_1_B (Frame_3_B)	-	X	Offset (+3)	0	3	3
PDU_1_B (Frame_4_B)	-	X	-	-	0	0

Table 140: Example for a PDU-Instance used in multiple frames (multiple routing)

7.4 Manipulation of Global Variables via FL3X Diagnostics Website

Global Variables can be modified at runtime via FL3X Diagnostics Website (see chapter 8). This option is always enabled even if the generation of ID-Defines is disabled.

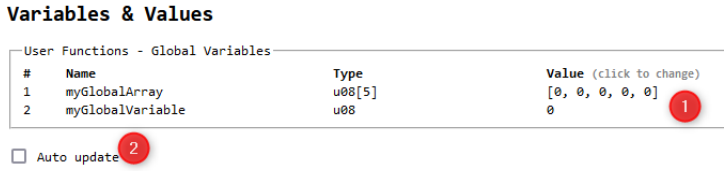
1. Open the FL3X Diagnostics Website (see chapter 8).



2. Switch to the tab “Variables”



3. All configured global variables are shown with name, type and current value.

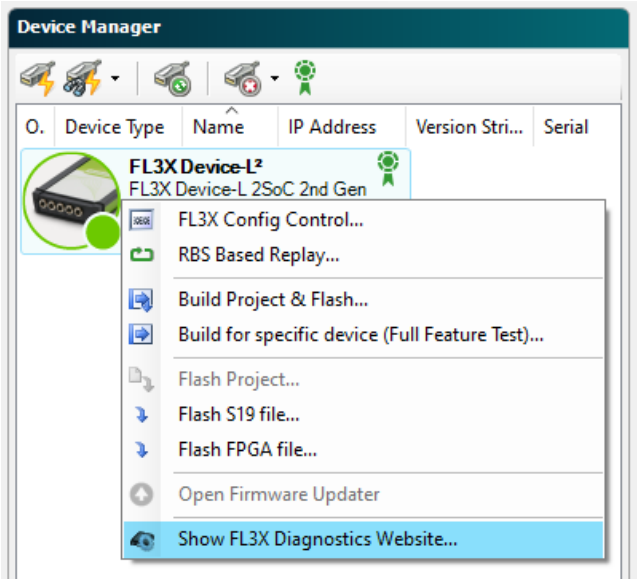


(1) Click on the current variable value to enter a new value which will be applied after ‘Enter’ was typed on the keyboard.
(2) Variable values are cyclically updated after the “Auto update” checkbox is enabled.

8 FL3X Diagnostics Website

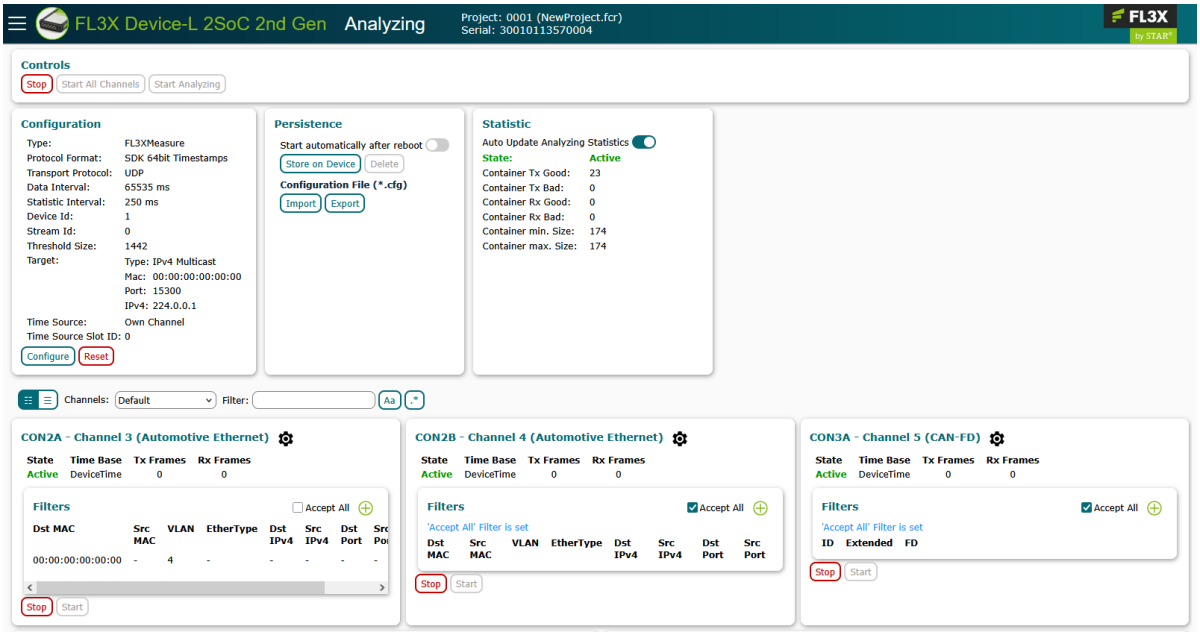
The FL3X Diagnostics Website is a website that shows the current state of a device and offers configuration options that are either possible or not possible via FL3X Config such as flashing, tracing, communication states or statistics.

The FL3X Diagnostics Website can be opened by entering the IP-Address of the device into a web browser. You can also click on the device in the device manager window of FL3X Config and select "*Show FL3X Diagnostics Website..*".



8.1 Analyzing

The Analyzing page allows the user to set up the Analyzing functionality on the device, configure each individual channel that is either an analyzing channel (i.e. not configured for RBS via FL3X Config) or an RBS channel configured via FL3X Config that has the "Analyzing Enabled" flag enabled for analyzing, enabling / disabling individual channels for Analyzing and start or stop Analyzing.



8.1.1 Configuration for Analyzing as ASAM CMP Probe

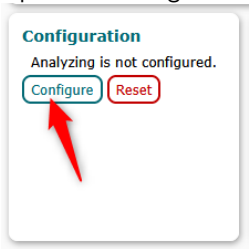
To configure the device for ASAM CMP (Capture Module Protocol) the following steps must be carried out:

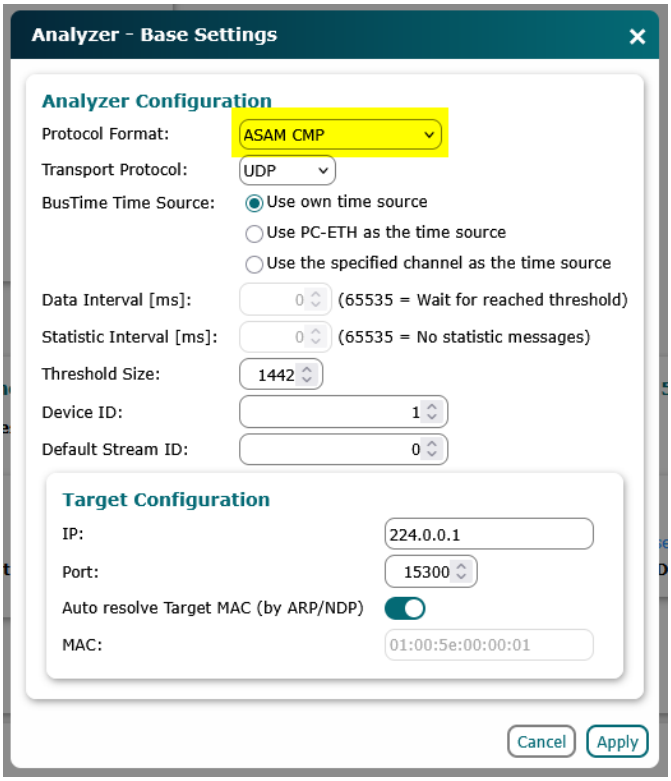
1. Configure Analyzer Base Settings (see chapter 8.1.1.1)
2. Configure Channels (see chapter 8.1.1.2)

3. Add Channel Filters (see chapter 8.1.1.3)
4. Start Channels (see chapter 8.1.1.4)
5. Start Analyzer (see chapter 8.1.1.5)

8.1.1.1 Configure Analyzer Base Settings

Open the configuration dialog and make sure to select “ASAM CMP” as protocol format.

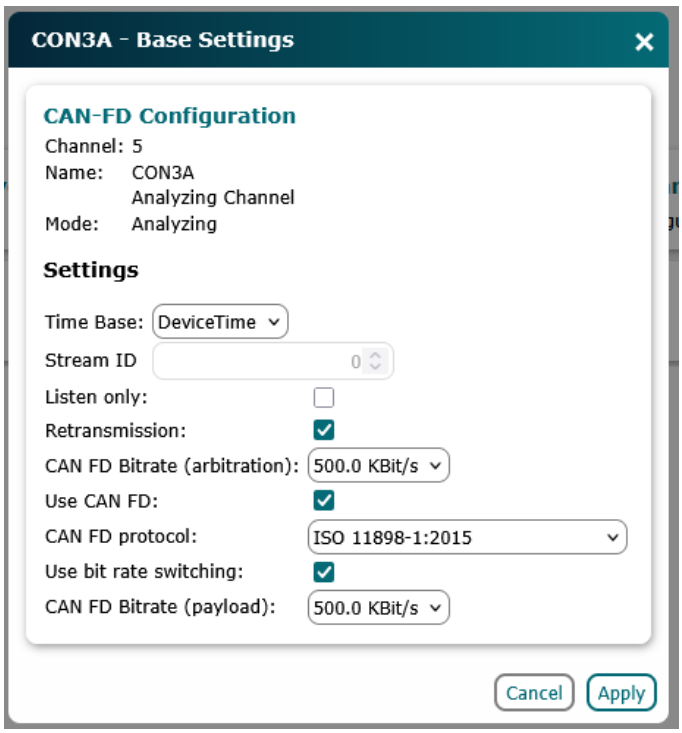




8.1.1.2 Configure Channels

Open channel configuration dialog for each channel which should be configured and apply the required settings.





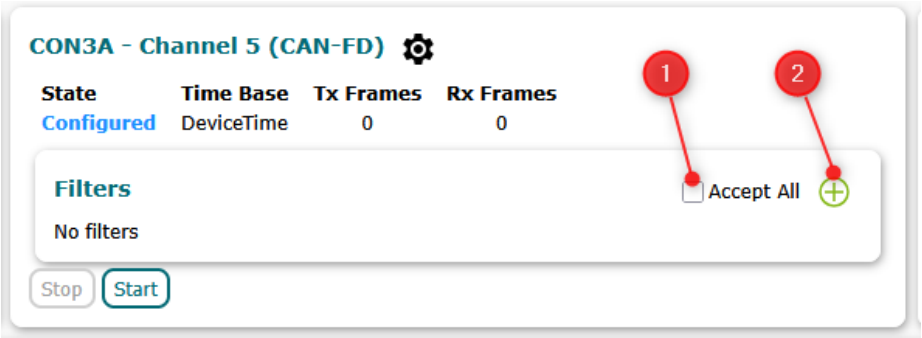
The available settings differ due to the channels bus type.

Currently CAN/CAN-FD, FlexRay and Ethernet are supported for ASAM CMP protocol format.

8.1.1.3 Add Channel Filters

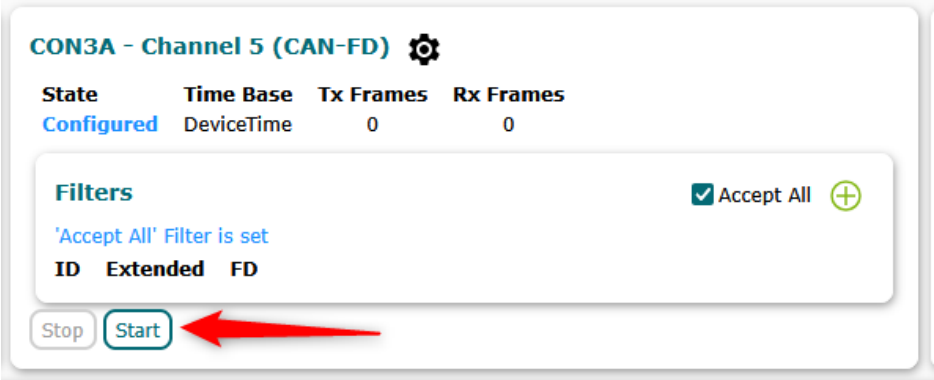
To allow reception of data, add required filters for each configured channel.

It's possible to add an "Accept All" filter to receive any message (1) or add an explicit filter to receive all frames matching the given settings (2)



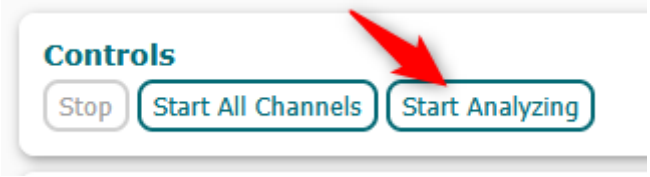
8.1.1.4 Start Channels

After configuration all required channels must be started

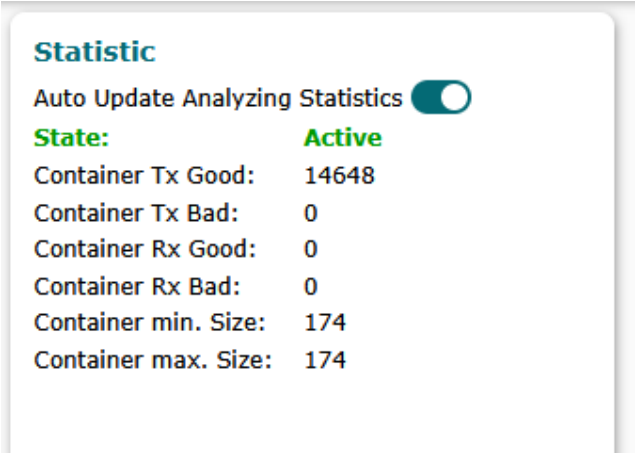


8.1.1.5 Start Analyzer

To start the Analyzer the [Start Analyzing] button must be clicked.

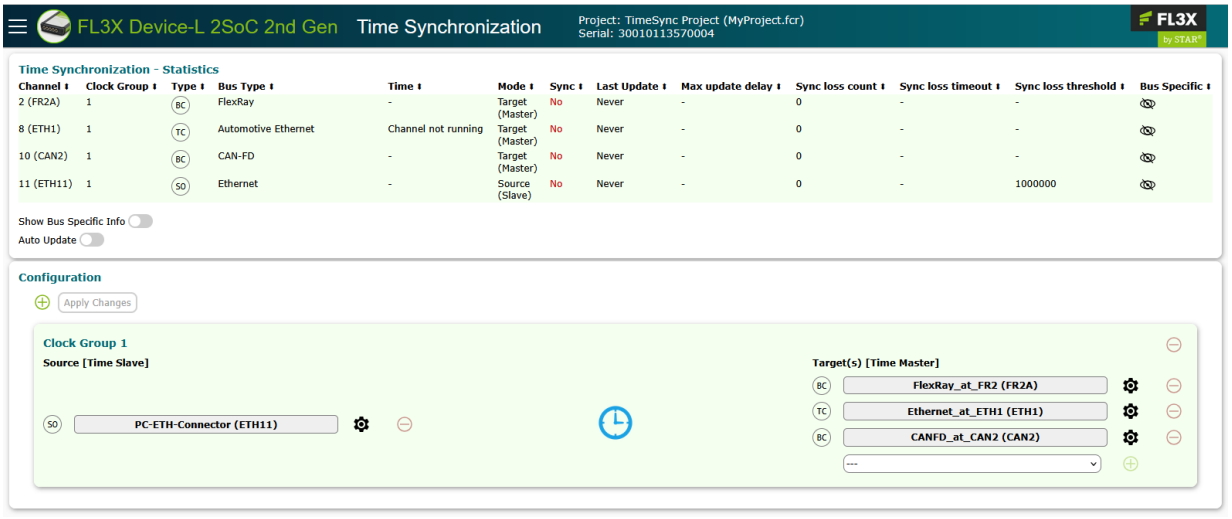


The state should be changed to “Active” and – in case of real bus communication – the receive and packet counters should increase.



8.2 Time Synchronization

The Time Synchronization page allows the visualization and configuration of Time Synchronization according to IEEE PTP or AUTOSAR (similar to the Tim Sync Configuration in FL3X Config, see chapter 5.11).

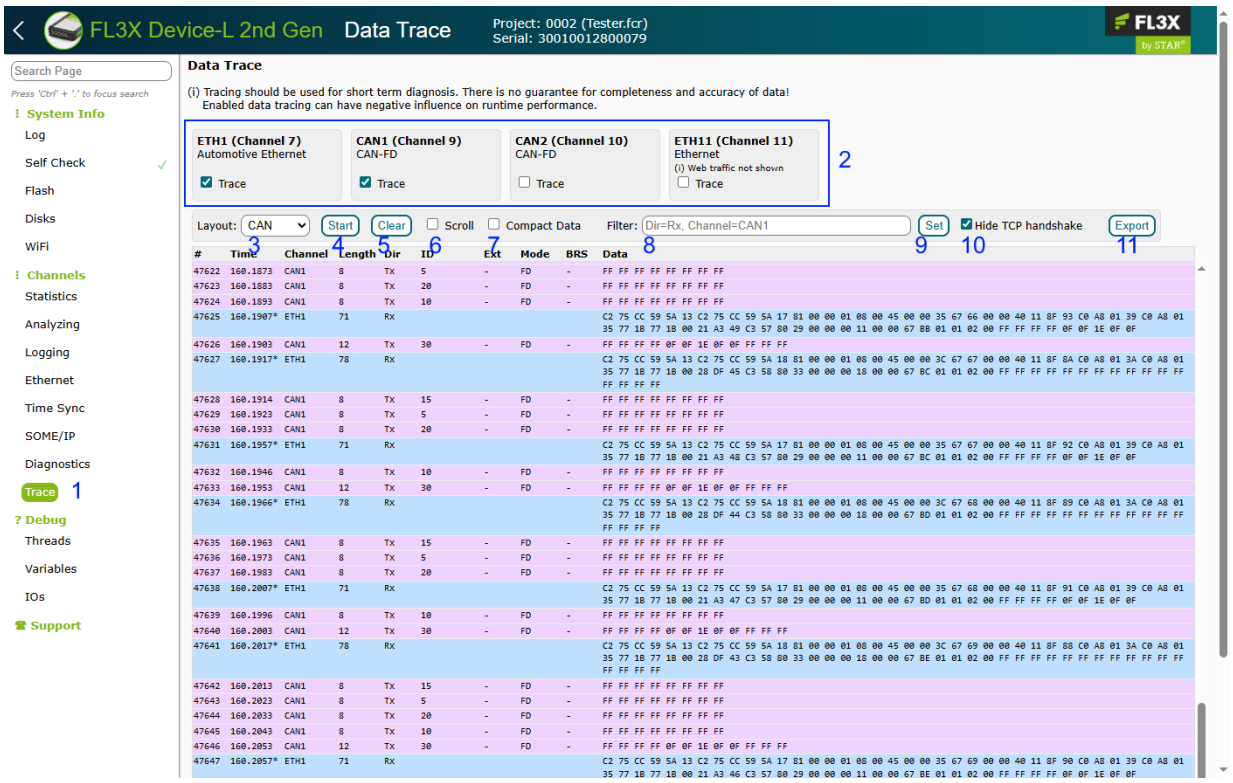


(Automotive) Ethernet channels which are not part of the FL3X Config Time Sync Configuration can be configured for Time Sync or changed, except for the PC-ETH-Connector, which may be reconfigured even if it was part of the FL3X Config Time Sync Configuration. Other bus types like CAN-FD cannot be configured but are visible and read-only if configured as part of the FL3X Config Time Sync Configuration.

All changes made by the Time Sync Website are permanently stored on the device and will be restored after a power cycle.

8.3 Trace

The trace page allows the user to show the data traffic of the selected connectors in a quick way without using a database and interpreting the packages. For extensive analysis the tool FL3X Config Analyzer can be used.



No	Description
1	Link for opening the trace tab.
2	List of all project buses. The buses can be selected here whose data traffic should be shown.
3	List for selecting the suitable layout (with different columns)
4	Starts showing the trace data
5	Clears the list of packets
6	Enables/disables auto scroll
7	Long data (e.g. ETH packets) can be cut if this option is activated.
8	Field for custom filter
9	Activates the entered filter
10	Shows/hides TCP handshake
11	Exports the data to an ascii (*.asc*) file.

Table 141: FL3X Diagnostics Website – Trace Tab

NOTICE

The timestamps shown in the trace list are purely software timestamps that only reflect the time the message was delivered to the communication controller.

This means that multiple messages can have the same timestamp. Accuracy cannot be guaranteed.

9 FL3X Config Project Format

FL3X Config stores all files and sub-folders of a single project in a project folder. This project folder is created together with the project file (**.fcr*). The project file is stored in this folder. Normally the project folder and the project file have the same name. But the project folder name can be different and is not considered in any way.

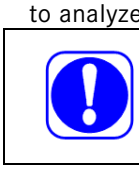
The content (files and folders) of the project folder is written by the FL3X Config. With some exceptions any change to files or folders will be overwritten by FL3X Config (every time the project is saved).

9.1 Files and folders of a FL3X Config Project

The following table describes the files and folders located in the project folder.

Meaning of the colums

- File/Folder: The name of the file or folder in the project folder.
- Required: Describes if the file or folder is required for the project
  - Essential –Essential to the project.
  - Optional – The project can be used without this file or folder. Some functionality can be restricted without this file or folder.
  - Generated – The file or folder is generated (e.g., at save project, Database Update, build project).
- Config: Used for project configuration.
- Build: Generated as result when building a project.
- Support: Information about the project and configuration history. This information helps our support team to analyze the project.



NOTICE

When using a VCS (Version Control System) such as Subversion or GIT it is recommended to include at least the “Essential” files and folders to the VCS.

File/Folder	Required	Config	Build	Support	Info
.fcr file	Essential	X			- Project configuration
.xml files (config.xml, Database.xml, DefaultValuesX.xml, GatewayMapping.xml, Manipulation.xml, UserFunctions.xml, TxBehavior.xml, E2EProfiles.xml…)	Essential	X			- Additional project configuration
~~bus folders (e.g. FR1)~~	~~Essential~~	~~X~~			~~- Bus specific information~~ Since FlexConfig RBS version 5.5: Replaced by the ‘db folder’
db folder	Essential	X			- Folder with the project databases
.xml file in db folder	Essential	X			- The imported database
.dat files in db folder	Optional	X		X	- Information about the original database - Needed to provide automatic database updates - Needed to provide support by STAR ELECTRONICS GmbH & Co. KG
usr folder	Essential	X			- Additional user C-Code for the project
UserFunctions folder	Essential	X			- This folder is only created when the FL3X Config Enterprise license is available. This folder contains subfolders with .c files (the user function code)
.fca	Essential	X			- FL3X Config Analyzer project information

File/Folder	Required	Config	Build	Support	Info
.bin file	Generated Normally not needed		X		- The compiled project as *.bin file - Created during build
.s19 file	Generated		X		- The compiled project as .s19 file - Used to flash the project to a device - Created during build
.fpga file	Generated		X		- The needed FPGA image - Used to flash the FPGA image to a device - Created during build
.dat file	Generated		X	X	- Project information about the compiled project - Created during build - Needed to provide support by STAR ELECTRONICS GmbH & Co. KG
out	Generated		X		- Out folder when build the project for one of the Virtual FL3X Devices
out\Virtual_FL3X_Device_Windows	Generated		X		- Out folder for Virtual FL3X Device for Windows builds - Contains different generated files for the Virtual Device support and integration
out\Virtual_FL3X_Device_ADTF	Generated		X		- Out folder for Virtual FL3X Device for ADTF builds - Contains different generated files for the ADTF support and integration
ID_Defines folder	Generated		X		- ID-Defines for Manipulations and PC-Hardware-Interface communication. - Created during build
.log file	Generated			X	- Information about the used FL3X Config version and used OS - Needed to provide support by STAR ELECTRONICS GmbH & Co. KG
.csv file	Generated			X	- Information about “Database Update” or “User Function import”. - Needed to provide support by STAR ELECTRONICS GmbH & Co. KG
.vscode folder	Generated	X			- Visual Studio Code integration - Contains the files and settings to edit and compile the User Functions via Visual Studio Code - This folder is only created when the setting “Settings – Application – Common Settings – Generate Files for Visual Studio Code” is set.

Table 142: Files and folders of a FL3X Config project

9.2 Ignored Files and Folders

The files and folders defined below are not modified by FL3X Config when loading or saving a project. With this behavior, the following use cases are supported:

- different VCS (Version Control Systems) such as SVN or GIT
- a folder for project documentation

There is a different behavior based on folder hierarchy.

First level files and folders (directly located in the project folder)

- Names starting with '.'
  Except '.vscode' which is provided by the feature "Visual Studio Code integration".
  See chapters
  5.17 Support for Visual Studio Code
  9.1 Files and folders of a FL3X Config Project
- The folder named "Docs"

At all levels (project folder and all sub folders)

- Hidden files and folders (file attribute)
- Files and folders with the following names:
  - *.git*
  - *.svn*
  - *.gitignore*
  - *.ignore*

10 FL3X Config Command Line Application

Some functions of FL3X Config are also available via the Windows command line. The following actions are available:

- Update projects
  The project files are updated to the latest FL3X Config version.
  A conversion of the included database files is also done, if necessary.
- Build projects
  This is the same action as via menu *'Build -> Project...'* (see chapter 5.4.3.1).
- Flash projects and binaries to connected devices
  This is the same action as via menu *'Flash...'* (see chapter 5.4.4).

To use the FL3X Config command line application:

1. Open a Windows command line (Start -> run "cmd").
2. Change the path to the FL3X Config installation path (the default is *C:\Program Files\StarCooperation\FL3X Config %VERSION%)*
3. The command line application name is *FL3XConfigConsole.exe*

The available actions and their parameters are not explained in this document. Please run the FL3X Config command line application with the parameter "`-help`" ("`FL3XConfigConsole.exe -help`") to show more information about the available actions and how to use them.
Using the FL3X Config command line application is optional. Every action of the FL3X Config command line application can also be done with the FL3X Config GUI application.

11 Device LEDs, Update Device License

This chapter contains general information about supported devices in relation to FL3X Config.

11.1 FL3X Device-S, FL3X Device-L & FL3X Device-L² LEDs

The FL3X Device-S, FL3X Device-L and FL3X Device-L² have some LEDs which are used to inform the user about the running application. More details about the LEDs can be found in [1] and [3].

Connector	Bus system	LED	Meaning
Con 1-5 (Con 1-3 for FL3X Device-S)	FlexRay	Red	The system is not synchronous and not in start-up state.
			There is an unknown/wrong/defect FL3X Tiny in the related FL3X Tiny-Slot.
			The bus is switched off due to calling the “BUS_OFF” function.
		Green	1. Blue DIO LED off: No failure. The system runs synchronous. 2. Blue DIO LED toggles: The project configuration and the related FL3X Tiny configuration are correct, but the system is not running.
		Red & Green	The start-up of the FlexRay communication is in progress.
		Off	There is no FL3X Tiny in the slot or the channel is not configured with a database
Con 1-5 (Con 1-3 for FL3X Device-S)	Ethernet		See the “instructions for use” document of the corresponding device.
Con 1-5 (Con 1-3 for FL3X Device-S)	CAN	Red	There is a wrong/defect FL3X Tiny in the slot.
			The bus is switched off due to calling the “BUS_OFF” function.
		Green	1. Blue DIO LED off: No failure. 2. Blue DIO LED toggles: The project configuration and the related FL3X Tiny configuration are correct, but the system is not running.
		Off	There is no FL3X Tiny in the slot or the channel is not configured with a database.
Con 1-5 (Con 1-3 for FL3X Device-S)	LIN	Red	LEDs are not used for LIN. Except Con 1 of FL3X Device-S: both LEDs are always on.
		Green	

Connector	LED	Meaning
ADI	Orange (ON)	A manipulation via FL3X Config Control (PC-HW-Interface [2]) is active
ADI	(OFF)	Signal manipulation is not active
ADI	Blue (TOGGLE)	If the factory reset button is pressed for 3 seconds, the blue LED will blink. This signalizes that the factory image is loaded.

Connector	LED	Meaning
DIO	Blue (ON)	During all flash operations (application update, firmware update, saving manipulation sequences and deleting manipulation sequences)

Connector	LED	Meaning
DIO	Blue (TOGGLE)	1. Firmware is not up to date. Please update the firmware. 2. FL3X Device hardware is unknown. 3. FL3X Tiny configuration is wrong (at the Connector where the red LED is on). 4. Hardware licenses do not match the project. 5. An error has occurred during device initialization.

Connector	LED	Meaning
Button (Between ADI and Wi-Fi for FL3X Device-S)	Green	FlexRay only: The configured offset (“InCycle Offset” only or “InCycle Offset” + “Cycle Offset”, see chapter 6.1 and 6.2) between side A and side B of a Gateway is reached.
Button (Between ADI and Wi-Fi for FL3X Device-S)	Blue	If the factory reset button is pressed for 3 seconds, the blue LED will blink. This signalizes that the factory image is loaded.
Button (Between ADI and Wi-Fi for FL3X Device-S)	Red	If the red LED will blink (and no user code activated it), an internal firmware error has occurred. If the red LED is continuously on (and no user code activated it), an internal firmware error has occurred. Please contact the *STAR ELECTRONICS* support team.

Connector	LED	Meaning
Wi-Fi	Blue	Wi-Fi is active
Wi-Fi	Orange	Wi-Fi is connected

Connector	LED	Meaning
SD-Card (Only FL3X Device-L/L² Rev3)	Orange	An SD-Card is recognized.
SD-Card (Only FL3X Device-L/L² Rev3)	Blue (TOGGLE)	SD-CARD write access. Hint: Do not remove the SD-Card while the blue LED toggles.

Connector	LED	Meaning
Power	Green (ON) Yellow (ON)	The FL3X Device is powered
Power	Green (TOGGLE) Yellow (TOGGLE)	The FL3X Device is in sleep mode.
Power	Green (OFF) Yellow (OFF)	The FL3X Device is not powered

11.2 FL3X Device-PXIe & FL3X Device-PCIe LEDs

The FL3X Device-PXIe/FL3X Device-PCIe have some LEDs which are used to inform the user about the running application.

Connector	Bus system	LED	Meaning
Con 1-5	FlexRay	Red	The system is not synchronous and not in start-up state.
			There is an unknown/wrong/defect FL3X Tiny in the related FL3X Tiny-Slot.
			The bus is switched off due to calling the “BUS_OFF” function.
		Green	1. Blue DIO[2] LED off: No failure. The system runs synchronous. 2. Blue DIO[2] LED toggles: The project configuration and the related FL3X Tiny configuration are correct but the system is not running.
		Red & Green	The start-up of the FlexRay communication is in progress.
		-	There is no FL3X Tiny in the slot.
Con 1-5	Ethernet		See the “instructions for use” document of the corresponding device.
Con 1-5	CAN	Red	There is a wrong/defect Tiny in the slot.
			The bus is switched off due to calling the “BUS_OFF” function.
		Green	1. Blue DIO[2] LED off: No failure. 2. Blue DIO[2] LED toggles: The project configuration and the related FL3X Tiny configuration are correct but the system is not running.
		-	There is no Tiny in the slot.
Con 1-5	LIN	-	LEDs are not used for LIN.

Please note that the DIO connectors and LEDs are slightly different to the FL3X Device family (see Figure 236).



Figure 236: FL3X Device-PXIe – DIO connectors

Connector	LED	Meaning
DIO[1]	Orange (ON)	A manipulation via FL3X Config Control (PC-HW-Interface [2]) is active.
DIO[1]	(OFF)	Signal manipulation is not active.
DIO[1]	Blue (TOGGLE)	If the factory reset button is pressed for 3 seconds, the blue LED will blink. This signalizes that the factory image is loaded.

Connector	LED	Meaning
DIO[2]	Blue (ON)	During all flash operations (application update, firmware update, saving manipulation sequences and deleting manipulation sequences).
DIO[2]	Blue (TOGGLE)	1. Firmware is not up to date. Please update the firmware. 2. FL3X Device hardware is unknown. 3. Tiny configuration is wrong (at the Connector where the red LED is on). 4. Hardware licenses do not match the project. 5. An error has occurred during device initialization.

LED position	LED	Meaning
User-LEDs	Green	FlexRay only: The configured offset (“InCycle Offset” only or “InCycle Offset” + “Cycle Offset”, see chapter 6.1 and 6.2) between side A and side B of a Gateway is reached.
User-LEDs	Blue	If the factory reset button is pressed for 3 seconds, the blue LED will blink. This signalizes that the factory image is loaded.
User-LEDs	Red	If the red LED will blink (and no user code activated it), an internal firmware error has occurred. If the red LED is continuously on (and no user code activated it), an internal firmware error has occurred. Please contact the *STAR ELECTRONICS* support team.

NOTICE

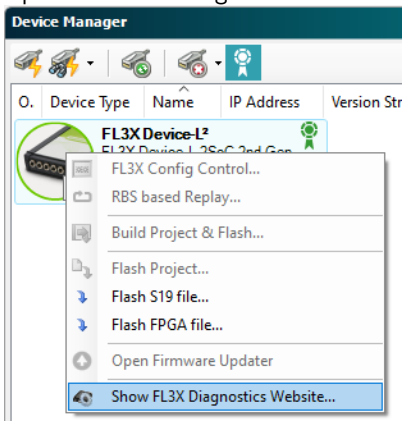
FL3X System-M devices do not have LEDs.

11.3 Update Device License

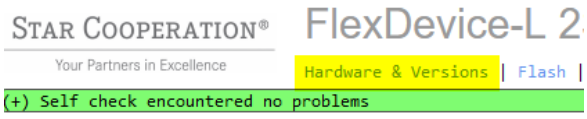
There are different ways to update the license on the device. If the device supports the FL3X Diagnostics Website (see chapter 8), it is possible to update the license directly via this website. Otherwise, the tool “FL3X Update” is required (see chapter 11.3.2).

11.3.1 Update Device License via FL3X Diagnostics Website

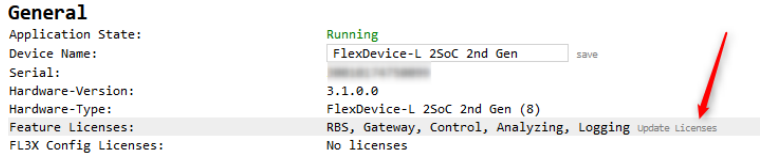
1. Open the FL3X Diagnostics Website via the Device Manager or directly in your Web Browser.



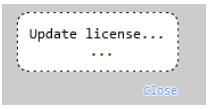
2. Switch to the tab “Hardware & Versions”.



3. Click on “Update License” and select your new license file.



4. Wait until the licenses will be updated. An automatic device reboot will be performed afterwards.



11.3.2 Update Devices via FL3X Update

To update licenses or FPGA images on the device (FL3X Device-S, FL3X Device-L/L², FL3X Interface-PMC, FL3X Device-PCIe, FL3X Device-PXIe3), you need the tool “FL3X Update”. FL3X Update can be downloaded separately.

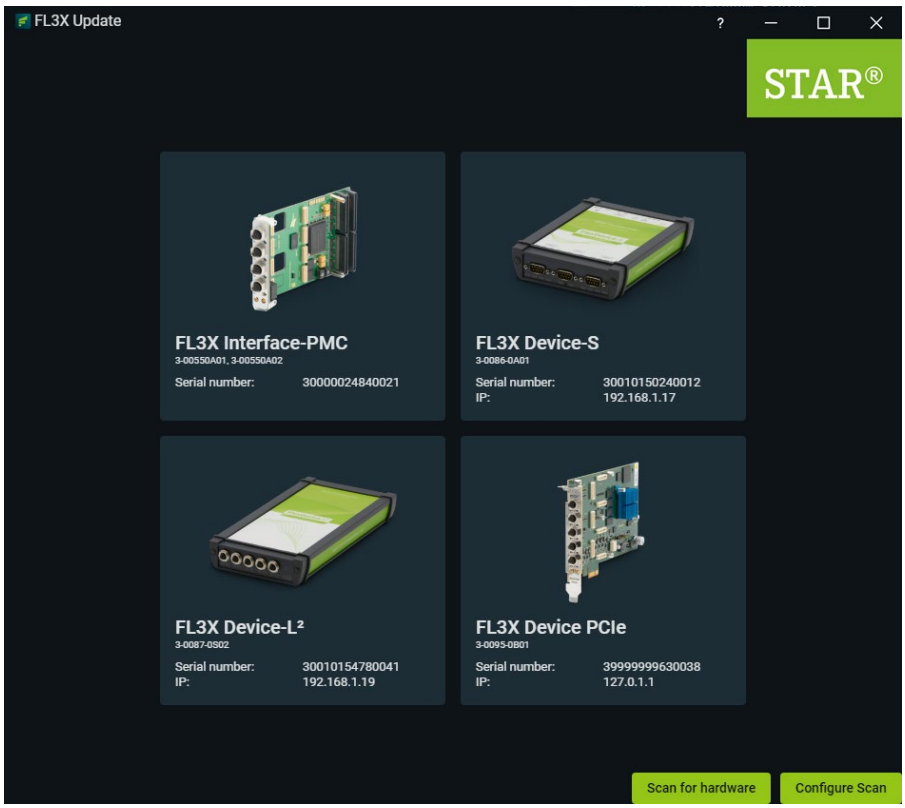


Figure 237: FL3X Update - Device Selection

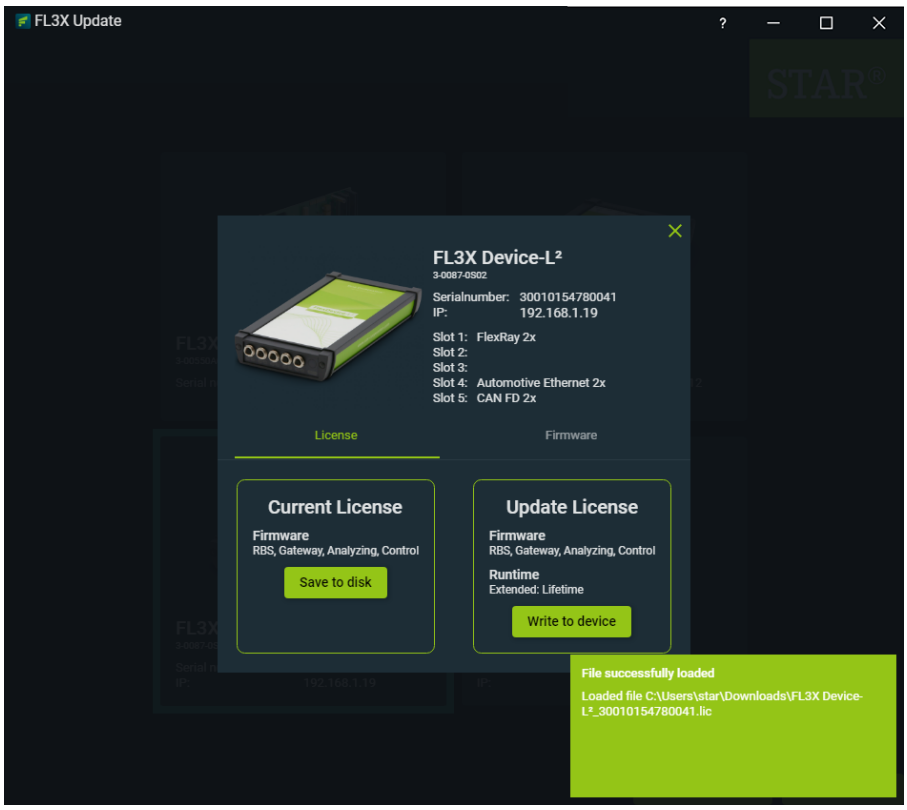


Figure 238: FL3X Update - License View

11.3.3 Write a License File to the Device

The following steps must proceed to write a license file to the device.

1. Start FL3X Update.
2. Scan for hardware and click on the device which needs an update (the screen will show all connected devices which are found, if more than one).
3. Click on the “*Update*” button.
4. Load the license file either via Drag & Drop or by clicking the update area to open a file dialog box.
5. If the file is valid, the “*Write to device*” button will be activated.
6. Click on the “*Write to device*” button.

For more detailed instructions and screenshots on writing a license and information on updating the FPGA images, there is a FL3X Update User Manual which is also included in the delivery package.

12 Troubleshooting

This chapter contains some frequently asked questions about FL3X Config.

12.1 FL3X Config – General

1 FL3X Config cannot be started.

The program crashes with the following error message.

Windows client with language set to “English”:

“FL3X Config has stopped working
A problem caused the program to stop working correctly. Please close the program.
-> Close the Program”

Windows client with language set to “German” (Deutsch):

“FL3X Config funktioniert nicht mehr
Ein Problem hat die richtige Ausführung dieses Programms verhindert. Schließen Sie das Programm.
-> Programm schließen”

There are some possible solutions to solve this error:

1. One reason for this error could be that there is more than one “CodeMeter Runtime Kit*” installed.
   To make FL3X Config run again the following steps are necessary:
   - uninstall FL3X Config.
   - open ‘*Control Panel -> Programmes and Features*’ and uninstall all “CodeMeter Runtime Kit*” entries.
   - install FL3X Config again.
2. Another reason for this error could be that the installed “CodeMeter User Runtime” is too old.
   To update the “CodeMeter User Runtime”:
   - close any open error messages that are shown by FL3X Config
   - open www.wibu.com and download the latest “CodeMeter User Runtime for Windows”
   - install the latest “CodeMeter User Runtime for Windows”
   - start FL3X Config again

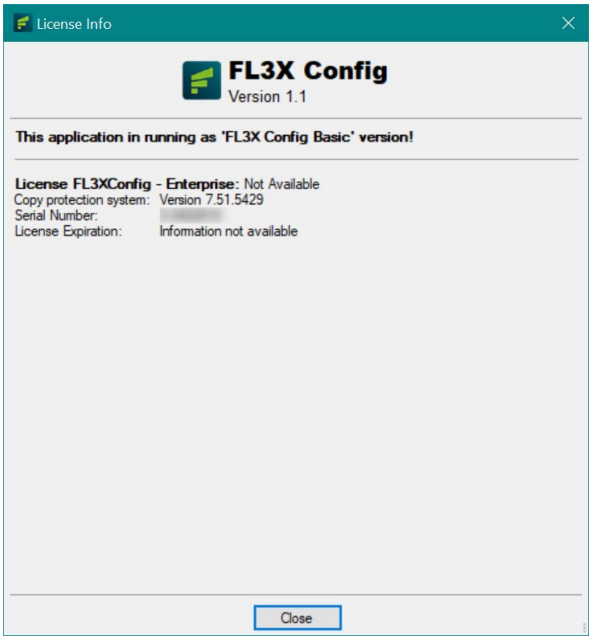
2 Dongle cannot be detected.

WIBU dongles are used starting with the Version 2.5.5.0 of FL3X Config. Dongles for the older Versions will not function any more for the new versions of the software. Please contact the *STAR ELECTRONICS* sales department for more information (see “Contact information” at the beginning of this document).

3 License not available

1. If no license is available, please check if the dongle (container)/license is visible the CodeMeter WebAdmin.
   http://localhost:22352/dashboard.html
2. Sometimes there are other WIBU dongles available in the network which can prevent FL3X Config detecting the correct license. In this case it can halp to deactivate the “Automatic Server Search” of the WIBU Runtime:
   1. Open the CodeMeter WebAdmin – Server Search List:
      http://localhost:22352/configuration/server_search_list.html
   2. Delete the ‘Automatic Server Search’ entry
   3. Click the [Apply] button

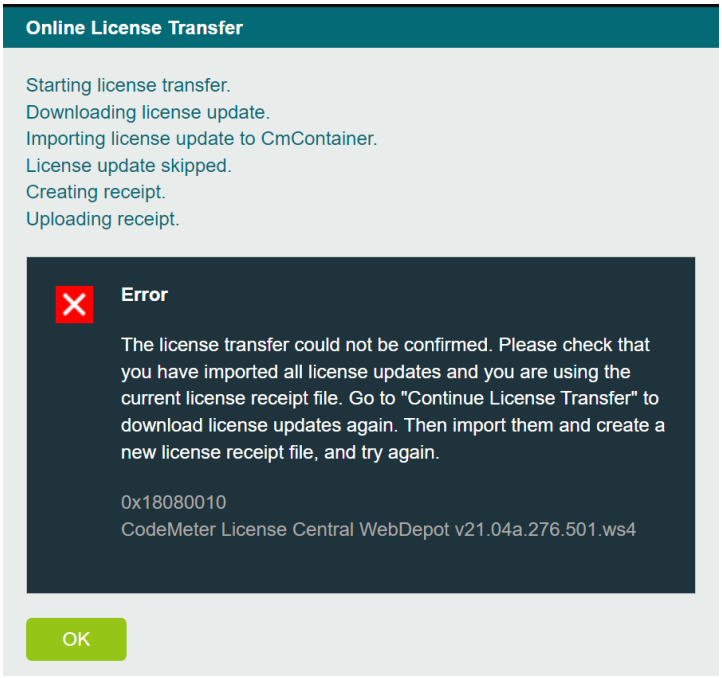
3. Sometimes the license is not available because the certified time of a dongle is outdated:



To update the certified time see : 5 How to update the ‘Certified Time’ of a CmContainer (Dongle).

4 Web Depot - License transfer could not be confirmed

If the license transfer (WebDepot https://license.star-cooperation.com) fails with the error 0x18080010



Pleas ensure that certified time of license (CmContainer/Dongle) is up to date.

See: 5 How to update the ‘Certified Time’ of a CmContainer (Dongle)

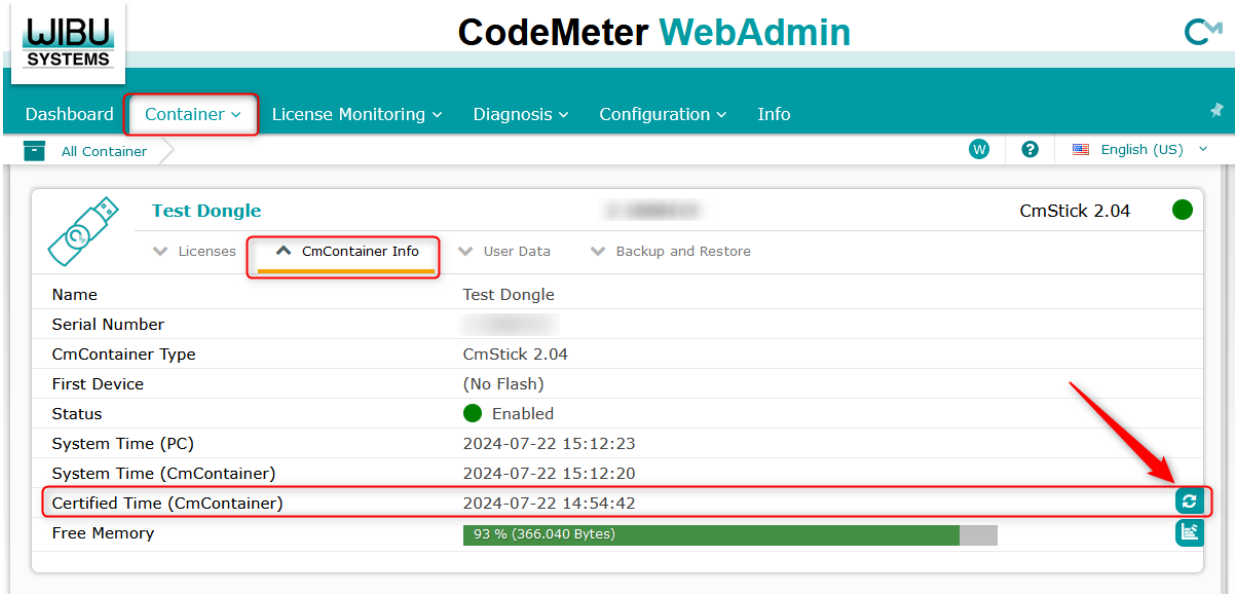
5 How to update the ‘Certified Time’ of a CmContainer (Dongle)

The license requires a certified time. It is possible that the certified time in the CmContainer (dongle) has expired and needs to be renewed. Follow the steps to update the certified time:

Prerequisite: An Internet connection must be available

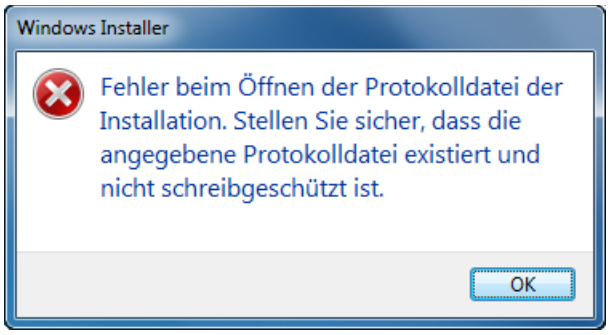
1. Open WebAdmin on the computer connected to the dongle. E.g.: http://localhost:22352/dashboard.html
2. Navigate to the CmContainer (Dongle).
   Menu “Container”

3. Select the CmContainer Info
4. Update the ‘Certified Time’ by click at the update button:



6 Uninstall fails (windows installer)

Uninstalling FL3X Config and/or the OEM Packages fails with the following error message:



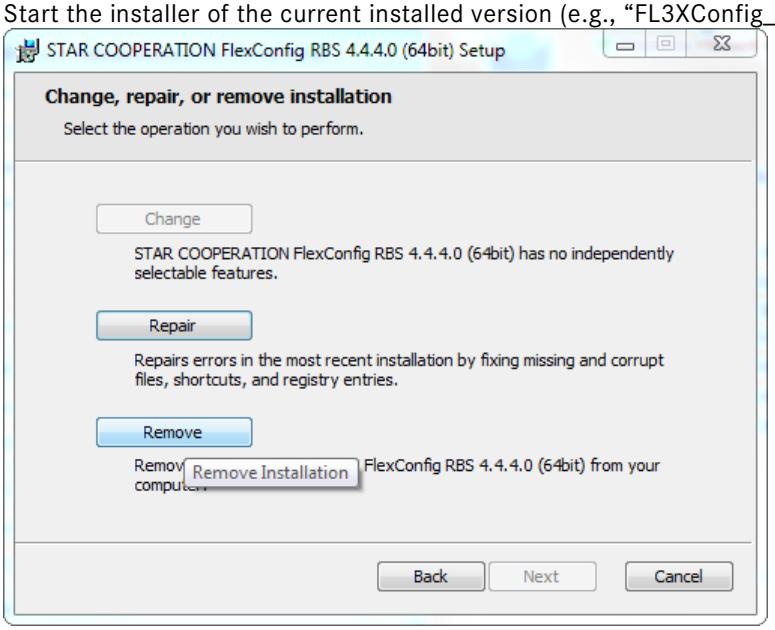
Use following steps to repair the installation service:

1. Open CMD with administrator rights (Windows-Key -> type “cmd” -> right click at cmd.exe and use “Run as administrator”)
2. Enter following calls:
   ```
   msiexec /unregister
   msiexec /regserver
   ```
3. Delete the content of the temp folders:
   C:\Windows\Temp\
   C:\Users\%user_name%\AppData\Local\Temp
4. Restart the PC

After these steps, uninstalling should work.

If not, try uninstalling via the installer:

1. Start the installer of the current installed version (e.g., “FL3XConfig_V1.0_64Bit.msi”)



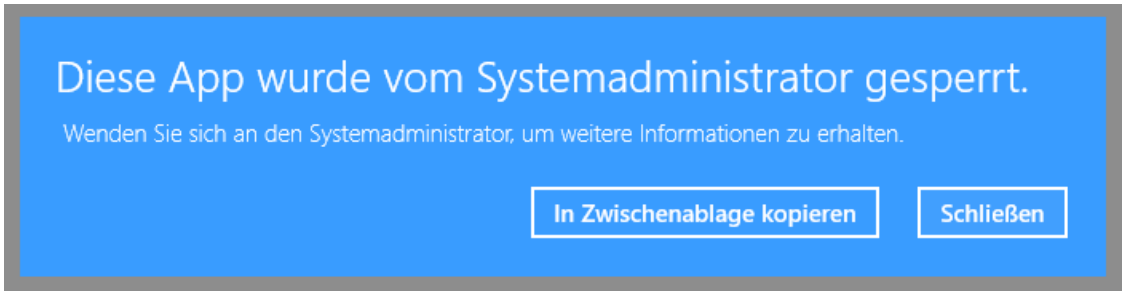
2. Use the button [Remove] to uninstall the software.

7 Problems with downloading the automatic application update.

If an automatic application update does not work properly, you must check if the proxy settings are correct. For details about the correct proxy settings, please contact your IT administration.

8 Problems with installing the automatic application update.

If the application update fails with a message as:



Maybe a system policy prevents to install the software. Please contact your system administrator.

The downloaded setup files are saved in the folder

C:\Users\%UserName%\AppData\Local\StarCooperation\FL3XConfig\Update

Installer names:

- *FL3XConfig_Setup_V%version%_64Bit.exe*
- *FL3XConfig_%OEM-Name%_Package_V%version%_64Bit.msi*

You can also try copying the setup files to a different location and running it from there.

9 Accessing the device via multiple instances of FL3X Config.

Principally it is permitted to access a device from different instances of FL3X Config. It is for example possible to start FL3X Config multiple times on the same PC or on different PCs (if enough licenses are available) and access the same device.

But you should not flash the device, if any other instance of FL3X Config executes a manipulation on the same device at the same time. This can damage the device.

10 Newly connected device not visible in FL3X Config.

A device has been physically connected to the network. After that the device is searched via the “*Search Devices*”-button but is not displayed in the devices-list.

There are different possibilities for this behavior. The most probable reason is that the device has another IP-Address than the network where devices are searched for. So, make sure that the IP-Address of the newly connected device is available with the actual Search Device IP Settings. Another reason can be that there is already a device existent within the network, which has the same IP-Address.

11 No network connection to the device via PC-ETH.

There are several possible reasons for this:

- Defective cable
- Defective connector (RJ45 or M9) on the FL3X Device
- Incorrectly configured network settings (see label on the back side of the FL3X Device)
  - IP address/speed differences
  - DHCP is not supported by the FL3X Device
- Defective hardware component on the FL3X Device related to Ethernet

12 The device name of the device cannot be displayed.

This should only happen if the project running on the device was built with an old FlexConfig RBS version.

To solve this issue, you can build and flash the appropriate project with the current version of FL3X.

13 Read settings shows “Not Supported”.

If you click the “*Read settings*” button, FL3X Config sends PC-Hardware-Interface commands to get information from the connected device (FL3X Device-S, FL3X Device-L/L²). These commands are only implemented for RBS/Gateway applications created by FL3X Config. So, if there are any other applications (e.g. the FL3X Device-L application delivered by default) running on the device, the result “Not Supported” is returned.
Flash the device with an RBS/Gateway application created by FL3X Config and then the “*Read settings*” operation should work as intended.

14 An expected window is not shown.

The last position and the size of many windows are saved automatically after closing them. Any change by the screen(s) can cause a problem with displaying a window (It can be shown in an area which cannot be displayed by the screen). In this case you must clear the content of the file, which keeps GUI properties.

If you can open the global settings dialog (*Extras ->Settings*), then you can click on the “*Clear*” button on *Application->Saved Data* window.

You can also delete the configuration file manually which is saved on the path “*C:\Users\%User-Name%\Documents\StarCooperation\FlexConfigRbsGUIState.xml*”

15 Items with the length ‘0’

Signals, PDUs and frames with the length ‘0’are not supported by FL3X Config. Such items are not shown in GUI.

16 VCS - Version management for FL3X Config project folders

To use a VCS (e.g. SubVersion, Git) to track and control the changes of the FL3X Config project folder see chapter 9.1.

17 GIT

See chapter 9.1.

18 SVN / Subversion

See chapter 9.1.

19 Files added to the project folder by the user are changed when the project is saved

See chapter 9.2 Ignored Files and Folders.

20 Tooltips are cut or disappear too fast

Most of the lists which contain database items support tooltips. You can use the key combination [Ctrl] + [Alt] + [T] for opening the last shown tooltip (of lists) in a new window, which let you also copy the content.

21 Tooltips occupy the display and prevent working with the GUI

At low resolutions it may be difficult to work in some dialogs (e.g. gateway editor) when tooltips are displayed upon mouse movements. The tooltips can be temporarily deactivated using [F7] key.

22 Required device configuration is not available for new project/change device

If a required device configuration/FPGA image is missing (via “Change Template” button), check the following list.
If the required FPGA image is listed here, please **contact** the **STAR ELECTRONICS Support team**.

Device Type	Image ID
FL3X Device-L²	0x1094108F / 0x2094208F (0x94/0x8F)
FL3X Device-L²	0x10941185 / 0x20942185 (0x94/0x185)
FL3X Device-L²	0x109C1184 / 0x209C2184 (0x9C/0x184)
FL3X Device-L²	0x109D1186 / 0x209D2186 (0x9D/0x186)

12.2 Building and Flashing Issues

1 Compile error after installing a new version of FL3X Config and compiling the project

1. Function calls to access database elements (e.g. ‘SetTxEvent’ for a frame) can be different between two versions of FL3X Config. When a project is opened, such function calls are recognized and changed/updated automatically.
   But source code in the user files is neither checked nor updated. This means that all function calls within user files must be updated **manually**.
2. Compile errors can occur because of using databases converted/imported with older FL3X Config versions. For some older imports, it cannot be determined whether a database update is necessary. Therefore, the application does not request a database update. After performing a manual database update, the project can be compiled without errors.

2 FL3X Config crashed after starting the build process.

It is possible that you do not get a direct response of the application after starting the build process. This behavior can theoretically have the reason that FL3X Config crashed.

But it is more probable that the project build time is very long (more than 30 minutes) due to the complexity of the project. If the progress bar still shows activity, then the application has not crashed yet.

3 Error message “Project too large”.

If building a project abort with an error “project too large”, then the configured project needs more resources than available.

The only way to solve this problem is to reduce the complexity of the given project. One possibility is to use the User Defines “DON’T_USE_X_MANIPULATION”, to switch off different types of manipulations. Another possibility is to disable all manipulations. See chapter 5.6.7 for more details.

4 Error message “Error during compiling the project.”, due to “size exceeds limit” problem.

It is possible that an error message box with the information “Error during compiling the project” pops up. If additionally, the error message “Internal build error: The size of the created output file exceeds the limit…” appears in the “Output”-section of the main window, the project consumes too many resources.

The only way to solve this problem is to reduce the complexity of the given project. A possibility is to use the User Defines “DON’T_USE_X_MANIPULATION”, to switch off different types of manipulations. Another possibility is to disable all manipulations. See chapter 5.6.7 for more details.

5 Display driver error while building a project.

While building a project, your PC may temporarily hang or become unresponsive, and you receive the following error message: “Display driver stopped responding and has recovered”.

Microsoft has published an article about this issue: http://support.microsoft.com/kb/2665946

6 Project Build Error with message ‘region ext_sram is full’.

A possible reason for this is that the project contains too large arrays.
E.g. an Array with 1234567890 elements (defined in the User Function Editor) will lead to this error.

7 Building of a project takes very long

For projects with large database files (e.g. an Ethernet database file >500 Mbyte), the code generation needs a lot of RAM. If the needed RAM exceeds the available physical memory, Microsoft Windows automatically uses the hard drive to provide more memory. Unfortunately, if this happens, the whole process becomes very slow. So the 'build project' process takes very long.
To reduce the needed RAM for 'build project', try to turn off the setting "Use parallel tasks for code generation" (see chapter 5.22.1.1 for more information).

8 Building of a project finished with errors due to "make.sys"

This problem may occur when security software (like antivirus software) checks project files during compile time.

Try rebuilding the project.

To avoid this problem, set the base folder of all projects to the exception list of the antivirus software.

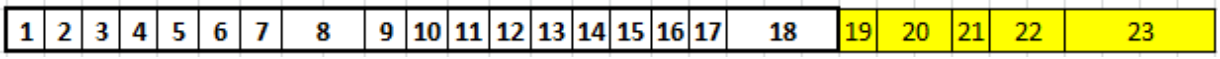
9 Building of a project finished with a warning – Compiling successfully finished!

Maybe there are not enough message buffers available. This problem is explained with an example.

An RBS or gateway is configured to send 150 frames. When building the project, the build process is performed and a warning is shown at the end, that only 128 of 150 could be sent and that 22 Message will not be sent.
However, to build the Project without the warning it may not be sufficient to remove the 22 frames from the send list, because now the error text says that only 114 of 138 frames could be sent and that 24 Messages will not be sent.

The reason for this behavior is that the message buffer length depends on the frame payload length, while there is only limited space in memory for all message buffers.

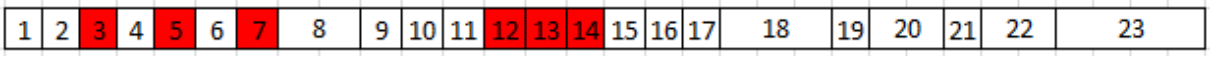
For example:



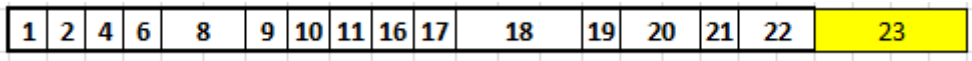
The resulting error text is:

```
"The controller {Controller Name} could not instantiate more than 18
message buffers, but 23 are required.
The following 5 messages will not be sent:
Message Buffer Index: 19; Type: {Buffer type}
Message Buffer Index: 20; Type: {Buffer type}
Message Buffer Index: 21; Type: {Buffer type}
Message Buffer Index: 22; Type: {Buffer type}
Message Buffer Index: 23; Type: {Buffer type}"
```

Now the following frames (labelled red) are removed from the sender list:



Even without the removed frames, the available memory space is insufficient.



The resulting error text is:

```
"The controller {Controller Name} could not instantiate more than 15
message buffers, but 16 are required.
The following 1 messages will not be sent:
Message Buffer Index: 16; Type: {Buffer type}"
```

In addition, ten message buffers are used as receive FIFO. So, the total numbers of required message buffers are "number of TX-Frames" + 10.

10 Building of a project finished with errors due to not conform FIBEX files.

Using not conform FIBEX-Files for defining busses can result in build errors.

For example, the length of a Signal is 8 bits, but the base data type of the appropriate coding is “UINT16”. This can lead to a build error of the form:

“error: negative width in bit-field”

If the length of a Signal is 0 will result in an error of the form:

“error: zero width for bit-field ‘xy’”

11 Building of a project finished with errors, related to user functions, after changing the device.

If user functions make use of functions coming from internal files (e.g. “*dio.h*”, “*Flextiny.h*”), errors could occur after changing the device. Since the content of these files is device dependent, the access to functions within these files is incompatible.

To circumvent this problem, these function calls must be removed or replaced.

12 Building of a project finished with errors, related to a user library, after using a new FL3X Config release.

In most cases, user libraries are compatible with new FL3X Config versions. Sometimes it may be necessary to update the user library for a new FL3X Config version.

Please contact STAR ELECTRONICS support team if compile errors occur with the user library after updating to a new FL3X Config version.

13 Building of a project finished with “error: duplicate case value” in “soda_FRx_y.c”

This can occur when several frames with the same absolute timing (slot ID, base cycle, cycle repetition) are configured on the receiver side. These collisions can arise through Rx-user-functions, get_Signal-calls in user functions and gateway mappings.

14 Error message “error: zero width for bit-field ‘%signal instance%’.

If building a project aborts with an error “zero width for bit-field ‘%signal instance%’”, then a signal is defined (in a database) with the length ‘0’. Such signals are not supported.

15 A timeout is reported while flashing a project.

The reason for this problem may be a change of the project’s network address via “*Edit*” -> “*Hardware Settings*”.

After the flash operation, the application checks if an update of the target’s FPGA image is necessary. But after the flash operation is complete, the target hardware will response on the configured (new) network address while the FL3X Config application tries to contact the hardware on the previous network address.

To re-establish the connection from the FL3X Config application to the hardware, try the following steps:

The IP address changed in the same sub net:
e.g. 192.168.1.15 (Default) to 192.168.1.33
-> Update the target address (e.g. in the “Read Settings” dialog) to the new hardware network address.

The complete IP address block changed:
e.g. 192.168.1.15 (Default) to 192.168.55.24
-> You must change the network configuration of your network device. Set the network address of your network device to the same subnet as the target hardware (e.g. 192.168.55.1). To change the configuration of your network device, administrator rights may be required.

If there is no change of a network address, please perform a power reset of the hardware device and flash the project again. If the problem still occurs, please contact the *STAR ELECTRONICS* support team.

16 Flashing not possible

When using the Windows Defender, FL3X Config must be allowed to use all local ports.

17 Flashing failed with error message “Invalid Device Type”

The error message “Invalid device Type” indicates that the project does not match to the selected device. It is important that the device revisions match.
E.g. if a project is created for FL3X Device-L² revision 1 (3-0087-0S**01**) but the target device is a FL3X Device-L² revision 2 (3-0087-0S**02**), this error message is shown.

18 Flashing failed with error message “[#0] File corrupted”

The error message “[#0] File corrupted]” indicates that the used network device allows “Jumbo frames”. FL3X Config does not support “Jumbo frames” for flashing.

Check the network device settings. The setting “Jumbo frames” should be set to “disabled”. To change the configuration of your network device, administrator rights may be required.
For some network devices, the name of the setting is “Jumbo Packet”.

19 Flashing Error message appears incorrectly when flashing via WIFI

If a project is flashed via WiFi on the FL3X System-XS/-XS² or FL3X System-M, the user incorrectly receives the following error message:

“Flashing failed!
Reason: The application image was downloaded but could not check for FPGA image update.
Reason: The device could not be reached before the timeout period (1000ms) expired!”

However, the project was successfully flashed to the device. To check this, reconnect the FL3X System-XS or FL3X System-XS² manually via WiFi and access the device's diagnostics page.

Here you can see that the project has been successfully flashed.

To check the correct FPGA image version, open the diagnostics page and check via “System Info” -> “FPGA1” whether a warning message is displayed.
If so, the appropriate FPGA image must be flashed manually via the “Flash” tab.

20 FL3X Device L is not accessible after flashing a s19 file created with the version 4.0.2.0.

FL3X Device-L can be (rarely) not accessible after flashing an .s19 file which is created with the version 4.0.2.0. You will probably see an “unknown device” image by the device manager if you try to connect to the device. A power reset should solve this problem.

21 A Project which was created with the “FlexXCon_midget-toolset” can’t be flashed.

The Project File (*.s19) includes some information for which FL3X Device variant it is build. If this information is missing, the flashing progress will be stopped with the message “...wrong Device”.

Please contact the *STAR ELECTRONICS* support team.

12.3 FL3X Config Control

1 Starting of a manipulation not possible

Due to downwards compatibility problems, it is possible that a manipulation cannot be started. This can happen if the project running on the device was built with an older version of FL3X Config.

To solve this problem, just rebuild (menu *‘Build -> Build and Download Project...’*) the old project with the latest FL3X Config version and then flash it to the hardware.

2 Manipulations for signed signals have unexpected results

Manipulations for signed signals other than 8, 16 and 32 bits can lead to unexpected results. This limitation affects FL3X Config Control and direct manipulations via PC-HW-Interface [2].

Please use User Functions to manipulate such signals.

1. Add a global variable to use as manipulation flag.
2. Create a PDU Tx User Function for the PDU which includes the signal to manipulate.
3. Implement the manipulation in this User Function
   Exampe code:

```
static u08 steps = 0;
U08 gVarManipFlag = getGlobalVarManipFlag();
// e.g. manipulation should be performed for 10 iterations
if(gVarManipFlag && steps < 10){
            s16 sigToManipulat = getSignalX();
            signalToManipulate = DoManipulation(signalToManipulate, steps);
            steps++;
            setSignalX(signalToManipulate);
}
else
```

```
{
            // reset manipulation flag and step count
            manipFlag = 0;
            setGlobalVarManipFlag(manipFlag);
            steps = 0;
}
```

4. Create a manipulation for the global variable (e.g. `gVarManipFlag`)
5. Perform the manipulation to trigger the manipulation code in the User Function.

12.4 User Functions

1 Which values have empty areas in a frame or PDU?

Empty areas, such as parts of a frame with no Signals or PDUs, are initialized with the default FRAME/PDU initialization values. These can be changed per bus in '*Extras -> Default Values Editor...*' or globally in '*Extras -> Settings*'. See chapters 5.9 and 5.22.3.4 for more details.

2 What happens in case of invalid switch codes of a multiplexed PDU?

RBS:
The current payload of the multiplexed PDU in the SODA-database is sent by the RBS if the switch code is invalid.

Gateway:
The current payload of the last routed dynamic PDU in the SODA-database is sent by the RBS, if the switch code is invalid.

If the switch signal is manipulated (via FL3X Config Control or user functions), then the behavior described above is applied depending on the project type.

3 Disabling Standard Signal handling (e.g. CRC calculation) in User Functions does not work.

For example, in a project standard signal handling (CRC calculation) is configured for signals with prefix "CRC_". Now a Tx-User Function for a FRAME_A (including a signal "CRC_sig") is created. Within this User Function the standard signal handling is disabled using the parameter of the User Function (see chapter 5.8.2.1).

However, it seems that the CRC calculation is executed anyway. This effect may occur, if a signal is used in more than one frame. E.g. the signal "CRC_sig" is included in frame "FRAME_A" and included in frame "FRAME_B" (the PDU may be referenced to both frames). If the CRC calculation is now disabled for frame "FRAME_A", it will be done for frame "FRAME_B" anyway. In this case it is necessary to disable standard signal handling in all frames with the included signal.

4 Error "syntax error before '?' token" appears after validating a User Function.

Copy & paste source code from other sources within the User Function Editor can lead to the error "syntax error before '?' token". This is because the User Function Editor only supports ASCII characters. Copying other character sets (e.g. UTF8) can lead to this error. So only use ASCII characters if you copy source code from other sources.

5 Behavior of the User Function call "SetTxEvent" changed for container PDUs

Since the enhanced support for container PDUs in FlexConfig RBS 4.3, the transmission of container PDUs must be triggered with the "SetContainerRawTxEvent" call.

The "SetTxEvent" call will trigger the frame (container PDU) only if the container PDU has any PDUs in its queue.

Attention: Existing projects will not be updated/modified automatically. The user functions must be modified manually.

12.5 FL3X Device – General

1 How to find the revision of a device

The information about the device revision can be found on the back side of the device. There should be a label that looks like this:

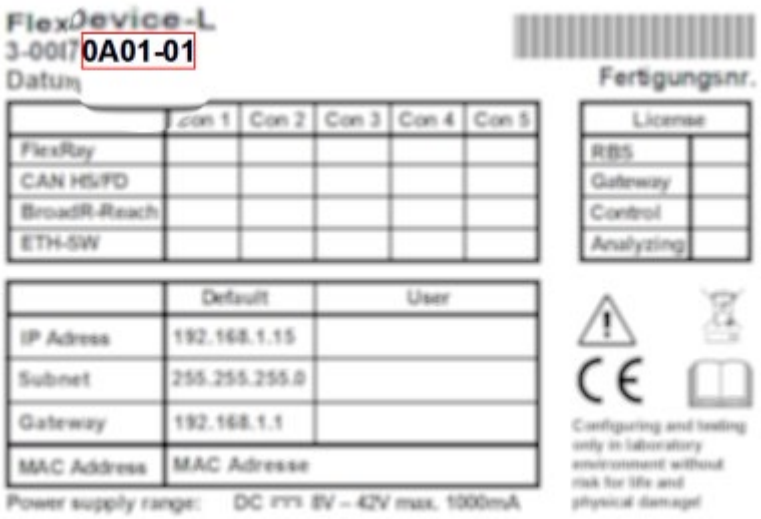


Figure 239: FL3X Device revisions – label with device information

2 How to connect to a device via WiFi

Find the device in your WiFi list. Each FL3X Device which supports WiFi will be shown with the name “FL3X Device-X-YYYYY”. X is the type of the device and YYYYY represents the last 5 digits of the serial number.

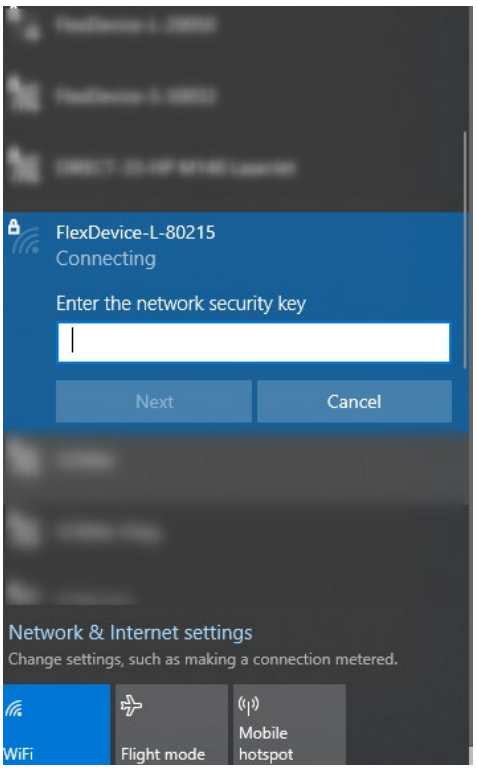


Figure 240: Example for connecting a FL3X Device via WiFi

Enter the **complete serial number** as security key, which is found on the label on back side of the device. The connection should be established.

NOTICE
The FL3X Diagnostics Website (see chapter 8) of FL3X Device L/L² Revision 2 and FL3X Device S can be accessed via IP-Address 198.51.100.50. The FL3X Diagnostics Website of a FL3X Device L/L² Revision 3 can be accessed via IP-Address 172.30.1.1

3 Device does not go into sleep mode (independent of sleep function call).

If the sleep function (s. chapter 5.8.3.4) is called within a user function, it is possible that the device does not go into sleep mode.

The problem can result from the connection of an Ethernet cable. To prevent this behavior, disconnect the Ethernet cable from the device.
Also check if the FL3X Tiny configuration matches the project (every FL3X Tiny slot should match the bus type of the selected FPGA image even if the slot is not used).

4 Communication/programming of the device is not possible.

It may happen that the communication with the device is not possible, e.g. the action “Flash -> Current project” aborts with an error message. If the green Ethernet LED is flashing, the device and the PC network card cannot agree on an Ethernet speed. Until now, this problem was only noticed with the “Intel 82579LM Gigabit Network” chipset.

The following workaround may solve this problem.

PLEASE NOTE: To use this workaround, you must have administrator rights on the PC.

Configure the PC network card with the option “10Mbps Full-Duplex” instead of “Auto Negotiation” (Link Speed).

5 There is no communication possible with the device after a firmware update.

Please contact the *STAR ELECTRONICS* support team.

6 DIO-IN does not work as expected.

The DIO IN just triggers on an “Rising Edge”, not between “0” and “1” (see chapter 5.8.3.4.2).

7 The FL3X Diagnostics Website cannot be opened via web browser

A configured proxy in the web browser may avoid accessing the FL3X Diagnostics Website.

8 WiFi network of FL3X Device does not appear after a firmware update

After flashing a *.s19*/*.fpga* file to a device via WiFi, the WiFi is disabled.

Restart the device by turning off the power supply. This only applies to devices with integrated WiFi extension.

Affected devices:

- 3-0087-0A02-01
- 3-0087-0S02-01

12.6 FL3X Device – Runtime Errors

1 FL3X Config project does not run on the selected device.

It is possible that a FL3X Config project does not run on the selected device due to problem with the FPGA-image. E.g. the FPGA image is broken due to an interrupted flash process.

Please perform the following steps, to restore the FPGA image:

1. Activate the factory image as described in the FL3X Device-L Instruction for use [1] or in the FL3X Device-S Instructions for Use [3].
2. Start the PC-Hardware-Interface Demo-GUI. This application is part of the PC-HW-Interface delivery package.
3. Connect to your device (via “TCP Open”)

4. Search for the Button "*Download FPGA...*" and click it.
5. Select the file "*FL3X Device_L_0x0000**0060**_%Version%.fpga*" (for FL3X Device-L device revision **1**) or the file "*FL3X Device_L_0x0000**1060**_%Version%.fpga*" (for FL3X Device-L device revision **2**) or the file "*FL3X Device_S_0x0000**00B0**_%Version%.fpga*" (for FL3X Device-S device revision **1**) and click "*Open*". This file is included in the delivered software package.
6. Wait until the blue DIO LED stops flashing. This process can last about three minutes.
7. When the blue DIO LED no longer permanently on, perform a power reset. You should now be able to flash and run a FL3X Config project to your device again.

If the project will not run with the actual needed FPGA-Image-Type, please check if an older version of the FPGA-Image version is required. If an older version is required, perform a downgrade of the FPGA Image. Please contact Star Electronics Support team for more information.

2 No FlexRay synchronization at runtime (FlexRay / FlexRay gateway).

This problem affects FlexRay / FlexRay Gateways where the cycle/in-cycle synchronization between the two gateway sides is used.

Most likely, the reason is one (or more) connected coldstart ECUs on the slave side of the gateway.
To make the synchronization between the two FlexRay bussed more efficiently, the following steps may help:

1. Open the "Bus Configuration" dialog of the slave bus and open "Offset Settings".
2. Activate "Disable Sync-restart" at the menu point "Sync- Restart".
3. For the fallowing steps, it is necessary that the Expert Settings Key is set at the Windows "Registry Editor":
   Navigate to HKEY_LOCAL_MACHINE -> SOFTWARE -> StarCooperation -> FlexConfig RBS
   Add a new String Value with
   - Value name: ViewExpertSettings
   - Value data: true
4. Restart FL3X Config
5. Open the "Expert Settings" dialog and edit the following points:
   1. UseOffsetControlBooster = true
      This configuration enables the second Communication Controller (CC) to be used for offset control.
      Notice: this option just has an effect if there are two CCs on this connector available.
   2. UseBusLock = true
      This configuration prevents any normal FlexRay synchronization on the slave bus by transmitting CAS symbols. This is done as long the master bus isn't synchronized.
      Now it is guaranteed that the FL3X Device starts the slave bus, and not the connected ECU.
      Notice: This behavior is not conforming to the FlexRay specification, so an ECU may switch to an error state.
   3. RateCorrectionDampingValue = (value -1).
      This configuration modifies a parameter for the CCs used for offset control.
      Notice: Changing this value affects the FlexRay bus communication. This could result in ECUs switching to an error state.
      Notice: Try to decrease this value step by step!

3 FlexRay frame collision for Transparent Gateways

Some databases contain frames which have no explicit sender ECU but multiple receiver ECUs. If a Transparent Gateway configuration – created with such a database – contains at least one receiver ECU on both sides, then the created RBS are configured to send those frame on both sides.
This may result in a FlexRay frame collision (two participants are trying to send in the same slot) on one side of the Transparent Gateway (because the real sender ECU is connected to this side).
To solve this issue, open the Bus Configuration Dialog of the side with the collision an move the frame to the ignore list.

12.7 FL3X Tiny/FPGA Issues

1. CAN/CAN-FD with FL3X Device-L/FL3X Device-L²

The available FPGA images with CAN/CAN-FD support are different in the numbers of CAN busses per connector (e.g., the FPGA image 0x106F has **two** CAN-FD busses on connector 5 while the FPGA image 0x1161 has **four** CAN-FD busses on connector 5).

There are two different variants of FL3X Tinys for CAN/CAN-FD. A variant with **two** CAN/CAN-FD busses per connector (3-00881E01) and a variant with **four** CAN/CAN-FD busses per connector (3-00881G02).

Be aware that **not** all combinations of FPGA image and FL3X Tiny are compatible. Please check the following cases:

Case 1: FPGA supports 2x CAN on a connector and the connector is mounted with a Tiny with 2x CAN

Case 2: FPGA supports 2x CAN on a connector and the connector is mounted with a Tiny with 4x CAN

Case 3: FPGA supports 4x CAN on a connector and the connector is mounted with a Tiny with 4x CAN

Case 4: FPGA supports 4x CAN on a connector and the connector is mounted with a Tiny with 2x CAN

Case	Funtionality	Restrictions	Support for Cable 3-00341L02 (1M9 -> 2DSub)	Support for Cable 3-00342S01 (1M9 -> 4DSub)
1	Full functional Bus_A -> over M9 pins 3&4 Bus_B -> over M9 pins 5&6	-	Yes (Bus_A & Bus_B)	Yes (Bus_B & Bus_C) No (Bus_A & Bus_D)
2	Partly functional Bus_A -> over M9 pins 1&2	No support for Bus_B	No	Yes (Bus_A) No (Bus_B, Bus_C & Bus D)
3	Full functional Bus_A -> over M9 pins 1&2 Bus_B -> over M9 pins 3&4 Bus_C -> over M9 pins 5&6 Bus_D -> over M9 pins 7&8	-	Yes (Bus_B & Bus_C) No (Bus_A & Bus_D)	Yes (Bus_A, Bus_B, Bus_C & Bus_D)
4	**Not** functional	If Bus_C or Bus_D is configured, the project does not start (Wrong physical layer)	No (Bus_A, Bus_B)	No (Bus_A, Bus_B, Bus_C, Bus_D)

13 Ordering Information

13.1 FL3X Config

Product	Description	Ordering number
FL3X Config Professional 1 year rent	Software license (see chapter 3.1)	3-V1141C01-01
FL3X Config Professional 2 year rent	Software license (see chapter 3.1)	3-V1141D01-01
FL3X Config Professional 3 year rent	Software license (see chapter 3.1)	3-V1141E01-01
FL3X Config Enterprise 1 year rent	Software license (see chapter 3.1)	3-V1141G01-01
FL3X Config Enterprise 2 year rent	Software license (see chapter 3.1)	3-V1141H01-01
FL3X Config Enterprise 3 year rent	Software license (see chapter 3.1)	3-V1141I01-01
FL3X Config Standard Runtime 1 year	Runtime license (see chapter 3.2)	3-01142A01-01
FL3X Config Standard Runtime 2 year	Runtime license (see chapter 3.2)	3-01142B01-01
FL3X Config Standard Runtime 3 year	Runtime license (see chapter 3.2)	3-01142C01-01
FL3X Config Standard Runtime 4 year	Runtime license (see chapter 3.2)	3-01142D01-01
FL3X Config Standard Runtime 5 year	Runtime license (see chapter 3.2)	3-01142E01-01
FL3X Config Standard Runtime 6 year	Runtime license (see chapter 3.2)	3-01142F01-01
FL3X Config Standard Runtime 7 year	Runtime license (see chapter 3.2)	3-01142G01-01
FL3X Config Standard Runtime Lifetime	Runtime license (see chapter 3.2)	3-01142O01-01
FL3X Config Standard Runtime Sync	Synchronize two or more Standard Runtime licenses to the same expiration date.	3-01142Q01-01
FL3X Config Extended Runtime 1 year	Runtime license (see chapter 3.2)	3-01142H01-01
FL3X Config Extended Runtime 2 year	Runtime license (see chapter 3.2)	3-01142I01-01
FL3X Config Extended Runtime 3 year	Runtime license (see chapter 3.2)	3-01142J01-01
FL3X Config Extended Runtime 4 year	Runtime license (see chapter 3.2)	3-01142K01-01
FL3X Config Extended Runtime 5 year	Runtime license (see chapter 3.2)	3-01142L01-01
FL3X Config Extended Runtime 6 year	Runtime license (see chapter 3.2)	3-01142M01-01
FL3X Config Extended Runtime 7 year	Runtime license (see chapter 3.2)	3-01142N01-01
FL3X Config Extended Runtime Lifetime	Runtime license (see chapter 3.2)	3-01142P01-01
FL3X Config Extended Runtime Sync	Synchronize two or more Extended Runtime licenses to the same expiration date.	3-01142R01-01

13.2 Accessory Parts

Product	Description	Ordering number
FL3X Dongle	Dongle to host software licenses	3-V0709A01-01
Customer specific parts		Please contact *STAR ELECTRONICS GmbH & Co. KG*

13.3 Related Documents

Document	Description	Ordering number
[1] FL3X Device-L Instruction for use	General description of the FL3X Device-L/L². FL3X Device-L/L² is a hardware platform to run generated RBS or gateway projects.	3-00870A01-D19
[2] PC-HW-Interface API Documentation	Describes the PC-HW-Interface implemented for the device. This interface can be used to control a running Remaining Bus Simulation.	3-99990C01-D04
[3] FL3X Device-S Instructions for Use	General description of the FL3X Device-S. FL3X Device-S is a hardware platform to run generated RBS or gateway projects	3-00860A01-D09

Document	Description	Ordering number
[4] FL3X Config mobile User Manual	User Manual for FL3X Config mobile	Please contact *STAR ELECTRONICS GmbH & Co. KG*
[5] FL3X Tiny 3 Family Instructions for Use	General description of the FL3X Tiny 3 Family. The FL3X Tiny 3 Family is a series of small interface modules with physical layer drivers for various bus systems.	3-00880Z01-D01
[6] FL3X Device-PXIe/PCIe Instructions for Use	General description of the FL3X Device-PXIe/PCIe. FL3X Device-PXIe/PCIe is a hardware platform to run generated RBS or gateway projects.	3-00940A01-D10
[7] FL3X Config – Device access via PC-HW-Interface	Examples for the PC-HW-Interface	3-01140A01-D019
[8] FL3X Config – Tutorial	A tutorial for FL3X Config	3-01140A01-D018
[9] FL3X Config Analyzer (Windows) – User Manual	User Manual for FL3X Config Analyzer (Windows)	3-01140A01-D014
[10] FL3X Config Manipulation Executor – User Manual	User Manual for FL3X Config Manipulation Executor	3-01140A01-D015
[11] RBS API Manual	General description of the RBS API	3-01140A01-D077
[12] FL3X Config Virtual Device Interface	General description of the FL3X Config Virtual Device Interface	3-01140A01-D076
[13] FL3X Config FAQ Software Runtime Licenses	FAQ for software runtime licenses	3-01140A01-D010
[14] FL3X System-M Instructions for Use	General description for FL3X System-M	3-01050A01-D11
[15] FL3X System-XS/XS² Instructions for Use	General description for FL3X System-XS/XS²	3-01040A01-D11
[16] FL3X Interface-PMC Instructions for Use	General description for FL3X Interface-PMC	3-00550P01-D05
[17] FL3X Config Tester User Manual	User manual for FL3X Config Tester	3-01140A01-D119

Additional Resources

In addition to the documents and software listed in this manual, further tools, firmware packages, and supplementary documentation may be required depending on the use case.

All additional downloads are provided on the official product website:

www.flex-product.com

14 Appendix

14.1 Acronyms and Abbreviations

Item	Definition
µS	Microsecond
API	Application Programming Interface
ASAM	Association for Standardization of Automation and Messuring Systems
AUTOSAR	AUTomotive Open System Architecture
BD	Bus driver
BP	Bus plus
BM	Bus minus
CAN	Controller Area Network
CANdb	A CANdb file (*.DBC) is used to store information about an entire CAN network
CH	Channel
CRC	Cyclic Redundancy Check
E2E	(End to End) This protocol defines abstract mechanisms for providing end-to-end communication
ECU	Electronic Control Unit
EMC	Electromagnetic Compatibility
ESD	Electro Static Discharge
Fibex	Field Bus Exchange
FIFO	First In First Out
FPGA	Field Programmable Gate Array
FR	FlexRay
ID	Identifier
K-Line	Bidirectional single-wire bus for data transmission
LIN	Local Interconnect Network
NC	Not Connected
Node	An ECU or a network node
Mdf, Mf4	Measurement Data Format (A popular bus log file format)
OEM	Original Equipment Manufacturer
PCB	Printed Circuit Board
PDU	A single unit of information transmitted among peer entities of a computer network.
PL	Physical Layer
PTP	The Precision Time Protocol
RBS	Remaining Bus Simulation
S19	This file is used to store the program code of microcontrollers in embedded systems
SDK	Software Development Kit
SENT	(Single Edge Nibble Transmission) is a serial interface
SOME/IP	Scalable Service-Oriented Middleware over IP
SPI	Serial Peripheral Interface
Sync	Synchronisation
UART	Universal Asynchronous Receiver / Transmitter
VLAN	Virtual LAN
VCS	Version Control System
XML	Extensible Markup Language

14.2 List of Tables

Table 1: Supported FPGA images – FL3X Device-S 19
Table 2: Supported FL3X Tinys – FL3X Device-S 19
Table 3: Supported FPGA images – FL3X Device-L 20

Table 4: Supported FL3X Tinys – FL3X Device-L 21
Table 5: Supported FPGA images – FL3X Device-L² 21
Table 6: Supported FL3X Tinys – FL3X Device-L² 22
Table 7: Supported FPGA images – FL3X Device-PXIe 23
Table 8: Supported FL3X Tinys – FL3X Device-PXIe 23
Table 9: Supported FPGA images – FL3X Device-PCIe 23
Table 10: Supported FL3X Tinys – FL3X Device-PCIe 24
Table 11: Supported FPGA images – FL3X Interface-PMC (FL3X System-M) 24
Table 12: Supported FL3X Tinys – FL3X Interface-PMC (FL3X System-M) 24
Table 13: Supported FPGA images – FL3X Interface-PCIe (FL3X System-M) 24
Table 14: Supported FL3X Tinys – FL3X Interface-PCIe (FL3X System-M) 25
Table 15: Supported device variant– ‘PCIe SFP Quad (LREC9714HF-4SFP)‘ module 25
Table 16: Supported SFP-Modules – PCIe SFP Quad (LREC9714HF-4SFP) 25
Table 17: Supported device variant– ‘PCIe SFP+ Dual (LREC9812BF-2SFP+)‘ module 25
Table 18: Supported SFP-Modules – PCIe SFP+ Dual (LREC9812BF-2SFP+) 26
Table 19: Supported device variant– ‘Mini PCIe SFP Single (DeLOCK 95267)‘ module 26
Table 20: Supported SFP-Modules – Mini PCIe SFP Single (DeLOCK 95267) 26
Table 21: Supported device variant– ‘CMI 8xDI 8xDO (CMI-DIO02/UB1018)‘ module 26
Table 22: Supported device variant– ‘CMI 10G Dual Port 10GBASE-T (CMI-10GLAN03)‘ module 27
Table 23: Supported FPGA images – FL3X System-XS 27
Table 24: Supported FPGA images – FL3X System-XS² 27
Table 25: Differences between the FL3X Config versions. 32
Table 26: List of features that require a Runtime License 34
Table 27: Description of the FL3X Config Main Window 45
Table 28: Description of the “Device Manager” section (Main Window) 47
Table 29: Description of the keys in the “Information” tab 51
Table 30: Description of the “Settings” tab 53
Table 31: Description of the activities in the “Activities” tab 54
Table 32: Description of the “New Project” dialog 63
Table 33: Description of the “License Information” dialog 68
Table 34: Description of the “Database Manager” dialog 69
Table 35: Description of the “Database Picker” dialog 70
Table 36: Description of the items of the “Device Configuration” section 71
Table 37: FL3X System-M – Configure slots 72
Table 38: Description of the Bus Configurations dialog – “Base Settings” (CAN/CAN-FD) 75
Table 39: Base Settings for CAN – Supported bit rates for CAN 75
Table 40: Description of the Bus Configurations dialog – “Base Settings” (FlexRay) 77
Table 41: Description of the Bus Configurations dialog – “Base Settings” (Ethernet) 78
Table 42: Description of the Bus Configurations dialog – “Base Settings” (Ethernet) – 10BASE-T1S 79
Table 43: Base Settings for Ethernet – “Static ARP Table” configuration 80
Table 44: Description of the Bus Configurations dialog – “Base Settings” (LIN) 83
Table 45: Bus Configuration – “RBS Settings” for CAN/CAN-FD 85
Table 46: Bus Configuration – “RBS Settings” for CAN – “Frame Settings” Dialog 85
Table 47: Bus Configuration – “RBS Settings” for FlexRay 87
Table 48: Bus Configuration – “RBS Settings” for FlexRay – “Edit Sync Frames” Dialog 88
Table 49: Bus Configuration – “RBS Settings” for Ethernet 92
Table 50: Bus Configuration – “RBS Settings” for LIN 95
Table 51: Bus Configuration – tab “E2E Protection” 97
Table 52: Tab “E2E Protection” – “Edit Protected Element” dialog 101

Table 53: Bus Configuration – “Standard Signal Handling” 104
Table 54: Bus Configuration – “Standard Signal Handling” – function prototype 104
Table 55: Bus Configuration – “Run Levels” 106
Table 56: Bus Configuration – “Offset Settings” 107
Table 57: Recommended Offset Control configuration for FPGA-Images 110
Table 58: Transmission Settings 110
Table 59: Transmission Settings – Common Settings 111
Table 60: Transmission Settings – “Frame” Settings 112
Table 61: Transmission Settings – “Frame Destination” Settings 112
Table 62: Transmission Settings – “PDU” Settings 115
Table 63: Transmission Settings – “Container” Settings 116
Table 64: Transmission Settings – “Service Items” Settings 117
Table 65: Transmission Settings – “Services” Settings 120
Table 66: Transmission Settings – “Preconfigured Service Discovery” Settings 120
Table 67: Transmission Settings – “Network Endpoints” Settings 121
Table 68: Description of performance optimization options in Transmission Settings 122
Table 69: Automatically created User Defines 123
Table 70: User Function Editor – Section “Functions” 126
Table 71: User Function Editor – Limitations for PDUs 127
Table 72: User Function Editor – List of usable Shortcuts 150
Table 73: User Function Editor – Section “Signals and Variables” 152
Table 74: User Function Editor – Description of the context menu for ECUs 153
Table 75: User Function Editor – Description of the context menu for section “Network” 163
Table 76: User Function Editor – Description of the context menu for section “Network-Service Items” 165
Table 77: User Function Editor – Description of the context menu for Global Variables 171
Table 78: User Function Editor – Description of the context menu for “Hardware” elements 173
Table 79: FL3X Device-L – Pin out of Analog-In 174
Table 80: FL3X Device-L – Pin out of Digital I/O’s 174
Table 81: User Function Editor – Description of the context menu items for “User Files” 178
Table 82: User Function Editor – Description of the different columns of the “Find Usages” window 181
Table 83: User Function Editor – Description of the section “Validation Output” 182
Table 84: User Function Editor – Supported Data Types 186
Table 85: Default Values Editor – Columns of the section “Profiles” 188
Table 86: Default Values Editor – Columns of the section “Signals” 189
Table 87: Default Values Editor – Columns of the tab “Frame & PDU Default Values” 191
Table 88: “Secure Onboard Communication” dialog 191
Table 89: Time Sync Configuration 194
Table 90: Time Sync Configuration – Clock behavior 194
Table 91: Time Sync Configuration – Common Parameters 195
Table 92: Time Sync Configuration – CAN specific Configuration 196
Table 93: Time Sync Configuration – Ethernet specific Configuration 199
Table 94: Time Sync Configuration – FlexRay specific Configuration 200
Table 95: Description of the Custom E2E-Profiles dialog 201
Table 96: Database Update – “Update Database” dialog 217
Table 97: “Create Partner Project” dialog 222
Table 98: “Create Partner Project” steps 223
Table 99: Application Settings – Common Settings 237
Table 100: Application Settings – Saved Data 237
Table 101: Application Settings – Update Interval 238

Table 102: Application Settings – Open File Settings 239
Table 103: Application Settings – Proxy Settings 240
Table 104: Project Settings – Common Settings 244
Table 105: Project Settings – Device Default IP-Settings 244
Table 106: Template Settings – Device Default IP-Settings 245
Table 107: Template Settings – Standard Signal Handling 245
Table 108: Template Settings – Initial Value Settings 246
Table 109: FL3X Config Gateway – Description of the sections 249
Table 110: FL3X Config Gateway – Description of the sections 250
Table 111: FL3X Config Gateway – Toolbar of the source section 251
Table 112: FL3X Config Gateway – Toolbar of the “Target” section 252
Table 113: FL3X Config Gateway – Toolbar of the “Gateway routings” section 252
Table 114: FL3X Config Gateway – Supported types of routings 254
Table 115: FL3X Config Gateway – Supported frame types 255
Table 116: FL3X Config Gateway – Supported routings for PDUs 255
Table 117: FL3X Config Gateway – Supported routings for signals of multiplexed PDUs 255
Table 118: FL3X Config Gateway – “Signal Scaling” dialog options 263
Table 119: FL3X Config Gateway – “Signal Scaling” editor – Interval types 264
Table 120: FL3X Config Gateway – Scaling status 264
Table 121: FL3X Config Gateway – “Create Autoroutings” dialog 267
Table 122: FL3X Config Gateway – “Export Items” dialog options 270
Table 123: FL3X Config Gateway – “Print Items” dialog options 271
Table 124: FL3X Config Gateway – “Import/Export Items” dialog options 272
Table 125: Transparent Gateway configuration dialog 280
Table 126: FL3X Config Control – Toolbar of the “Manipulable Entities” section 284
Table 127: Manipulations – Value Limits depending on manipulation type 288
Table 128: FL3X Config Control – Available elements for manipulation 289
Table 129: FL3X Config Control – Available SOME/IP elements for manipulation 289
Table 130: Manipulation ‘Loss’ – Effect on manipulated elements 290
Table 131: Manipulation ‘Disable Manipulation’ – Effect on manipulated elements 291
Table 132: Manipulation ‘Factor’ – Effect on manipulated elements 291
Table 133: Manipulation ‘Freeze’ – Effect on manipulated elements 292
Table 134: Manipulation ‘Freeze’ – Example for a gateway 293
Table 135: Manipulation ‘FixValue’ – Effect on manipulated elements 293
Table 136: Manipulation ‘Drift’ – Effect on manipulated elements 294
Table 137: Manipulation ‘Offset’ – Effect on manipulated elements 296
Table 138: Manipulation ‘Ramp’ – Effect on manipulated elements 297
Table 139: Example for a PDU-Instance used in multiple frames (single routing) 307
Table 140: Example for a PDU-Instance used in multiple frames (multiple routing) 307
Table 141: FL3X Diagnostics Website – Trace Tab 313
Table 142: Files and folders of a FL3X Config project 315

14.3 List of Figures

Figure 1: CodeMeter Control Center 33
Figure 2: List of features that require a Runtime License 35
Figure 3: FL3X Config – Installation wizard 37
Figure 4: FL3X Config – Installation wizard – Options menu 38
Figure 5: FL3X Config – Version Selector 40
Figure 6: FL3X Config – Software license expire dialog 41

Figure 7: FL3X Config Main Window (after starting FL3X Config) 44
Figure 8: FL3X Config Main window after loading a project (selected device: FL3X Device-L2) 45
Figure 9: FL3X Config Main Window – “Device Manager” section 46
Figure 10: “Device Manager” section – “Information” tab 48
Figure 11: “Device Manager” section – “Settings” tab 52
Figure 12: “Device Manager” section – “Activities” tab 53
Figure 13: “Device Manager” section – “Logging” tab 54
Figure 14: Flashing the current project over Ethernet to the selected device 55
Figure 15: Flashing a .s19 file over Ethernet to the selected device 56
Figure 16: Device Manager – section “Hardware Licenses” 57
Figure 17: Runtime license expire dialog 58
Figure 18: Device Manager - Firmware Update available 59
Figure 19: Firmware Update Dialog 59
Figure 20: Firmware Update - Available Updates 60
Figure 21: FL3X Config Main Window – “Output”-section 60
Figure 22: Menu bar 60
Figure 23: The *‘File’* menu 61
Figure 24: The “New Project”-dialog 62
Figure 25: The *‘Edit’* menu 63
Figure 26: The sub menu entries for Transparent Gateways 64
Figure 27: The *‘Build’* menu 65
Figure 28: The *‘Build – Project for’* menu 66
Figure 29: The *‘Flash’* menu 66
Figure 30: The *‘Extras’* menu 67
Figure 31: The *‘Help’* menu 67
Figure 32: License Information Dialog 68
Figure 33: Database Manager dialog 69
Figure 34: Database Picker dialog 70
Figure 35: “Device Configuration”-section (here for FL3X Device-L) 71
Figure 36: FL3X System-M – Configure slots 72
Figure 37: Info dialog for FlexRay/ETH/LIN bus 73
Figure 38: Database selection dialog for CAN bus 73
Figure 39: Bus Configuration – “Base Settings” (CAN-FD) 74
Figure 40: Bus Configuration – “Base Settings” (FlexRay) 76
Figure 41: Bus Configuration – “Base Settings” (Ethernet) 77
Figure 42: Bus Configuration – “Base Settings” (Ethernet) -10BASE-T1S Parameters 79
Figure 43: Base Settings for Ethernet – “Static ARP Table” configuration dialog 80
Figure 44: Base Settings for Ethernet – “Alternate IP Configuration” dialog 81
Figure 45: Base Settings for Ethernet – “TTL/Hop Limit Configuration” dialog 82
Figure 46: Bus Configuration – “Base Settings” (LIN) 82
Figure 47: Bus Configuration – “RBS Settings” for CAN-FD 84
Figure 48: Bus Configuration – “RBS Settings” for CAN – “Frame Settings” Dialog 85
Figure 49: Bus Configuration – “RBS Settings” for FlexRay 86
Figure 50: Bus Configuration – “RBS Settings” for FlexRay – “Edit Sync Frames” dialog 88
Figure 51: Bus Configuration – “RBS Settings” for Ethernet (tab “Services” in front) 89
Figure 52: Bus Configuration – “RBS Settings” for Ethernet (tab “Frames” in front) 92
Figure 53: Bus Configuration – “RBS Settings” for LIN 93
Figure 54: Bus Configuration – Add Frames Dialog 96
Figure 55: Bus Configuration – Tab “E2E Protection” 97

Figure 56: Tab “E2E Protection” – Edit a protected element 97
Figure 57: Tab “E2E Protection” – Assign an element to a E2E profile 98
Figure 58: Tab “E2E Protection” – Context menu of an E2E protected Element 98
Figure 59: Tab “E2E Protection” – Delete E2E profile assignment from protected element 98
Figure 60: Tab “E2E Protection” – “Edit Protected Element” dialog 99
Figure 61: Tab “E2E Protection“ – Import menu 101
Figure 62: Bus Configuration – “Standard Signal Handling” 103
Figure 63: Bus Configuration – “Standard Signal Handling” – “Open User File” dialog 104
Figure 64: Bus Configuration – “Offset Settings” 106
Figure 65: Transmission Settings – “Frame” Tab 110
Figure 66: User Defines 122
Figure 67: User Function Editor 124
Figure 68: User Function Editor – Section “Functions” 125
Figure 69: User Function Editor – Add “Frame Transmit User Function” 127
Figure 70: User Function Editor – Frame selection to create a “Frame Transmit User Function” 127
Figure 71: User Function Editor – Add “Frame Receive User Function” 128
Figure 72: User Function Editor – Frame selection to create a “Frame Receive User Function” 129
Figure 73: User Function Editor – Add “PDU Transmit User Function” 130
Figure 74: User Function Editor – PDU selection to create a “PDU Transmit User Function” 130
Figure 75: User Function Editor – Add “PDU Receive User Function” 131
Figure 76: User Function Editor – PDU selection to create a “PDU receive User Function” 131
Figure 77: User Function Editor – Creating a “Bus Receive User Function” 132
Figure 78: User Function Editor – Dialog “Add Filtered Receive User Function” for an Ethernet Bus 132
Figure 79: User Function Editor – Dialog “Add Filtered Receive User Function” for a CAN/CAN-FD Bus 133
Figure 80: User Function Editor – Dialog “Add Filtered Receive User Function” – Option to duplicate packets 134
Figure 81: User Function Editor – Creating a “Event Transmit User Function” 135
Figure 82: User Function Editor – “Dialog for (provided) event selection” 135
Figure 83: User Function Editor – Creating a “Event Receive User Function” 136
Figure 84: User Function Editor – “Dialog for (consumed) event selection” 136
Figure 85: User Function Editor – Creating field functions on the provider side 137
Figure 86: User Function Editor – “Dialog for (provided) field selection” 137
Figure 87: User Function Editor – Creating field functions on the consumer side 138
Figure 88: User Function Editor – “Dialog for (consumed) field selection” 138
Figure 89: User Function Editor – Creating a “Method Call Receive User Function” 139
Figure 90: User Function Editor – “Dialog for (provided) method selection” 140
Figure 91: User Function Editor – Creating method functions on the consumer side 140
Figure 92: User Function Editor – “Dialog for (consumed) method selection” 141
Figure 93: User Function Editor – Add “Usermain Function” 141
Figure 94: User Function Editor – Configure the period for the “Usermain Function” 142
Figure 95: User Function Editor – Add Initialize User Function 142
Figure 96: User Function Editor – Add “Cyclestart Interrupt Function” 143
Figure 97: User Function Editor – Select bus to create “Cyclestart Interrupt Function” 143
Figure 98: User Function Editor – Enable/Disable User Function 144
Figure 99: User Function Editor – “Export User Functions” dialog 144
Figure 100: User Function Editor – section “Signals and Variables” 151
Figure 101: User Function Editor – context menu for ECUs 152
Figure 102: User Function Editor – context menu for transmit frames 153
Figure 103: User Function Editor – context menu for signals 154
Figure 104: User Function Editor – Example data structure for an event 166

Figure 105: User Function Editor – Get/Set functions for data members 166
Figure 106 User Function Editor – Context menu for get/set ascii string functions 167
Figure 107: User Function Editor – Context menu for stacked-based E2E profiles 167
Figure 108: User Function Editor – CRC and Counter Signals with missing data types 168
Figure 109: User Function Editor – context menu arrays with dynamic sizes 168
Figure 110: User Function Editor – context menu for all Global Variables 169
Figure 111: User Function Editor – “Global Variables” dialog 170
Figure 112: User Function Editor – context menu for a selected Global Variable 170
Figure 113: User Function Editor – context menu for “Hardware->Analog I/O” 172
Figure 114: User Function Editor – context menu for “User Libs” 175
Figure 115: User Function Editor – context menu for a single “User Lib” 175
Figure 116: User Function Editor – Tooltip of user library functions 176
Figure 117: User Function Editor – Adding user lib function to code 176
Figure 118: User Function Editor – context menu for “User Files” 178
Figure 119: User Function Editor – “Find Usages” dialog 181
Figure 120: User Function Editor – section “Validation Output” 182
Figure 121: User Function Editor – Merge User Functions button 183
Figure 122: Merge User Functions – Bus Assignment 183
Figure 123: Merge User Functions – Select Functions 184
Figure 124: The “Default Values Editor” 187
Figure 125: Default Values Editor – “Create new Profile” dialog 188
Figure 126: Default Values Editor – Change default value 189
Figure 127: Default Values Editor – Context menu in signals section 190
Figure 128: Default Values Editor – Tab “Frame & PDU Default Values” 190
Figure 129: “Secure Onboard Communication” dialog – Example for FlexRay 191
Figure 130: Flowchart for SecOC 193
Figure 131: Time Sync Configuration – Example for CAN, Ethernet, FlexRay 193
Figure 132: Time Sync Configuration – CAN Master Configuration 195
Figure 133: Time Sync Configuration – CAN Slave Configuration 196
Figure 134: Time Sync Configuration – Ethernet Slave Configuration 197
Figure 135: Time Sync Configuration – Ethernet Master Configuration 198
Figure 136: Time Sync Configuration – FlexRay Master Configuration 199
Figure 137: Time Sync Configuration – FlexRay Slave Configuration 200
Figure 138: “Custom E2E-Profiles” dialog 201
Figure 139: Custom E2E – Available templates for Custom E2E-Profiles 202
Figure 140: Custom E2E –parameters of a signal based custom E2E-Profile 203
Figure 141: Custom E2E – method “Length” for custom “AUTOSAR Profile 4 (Signal-Based)” 204
Figure 142: Custom E2E – method “DATA-ID” for custom “AUTOSAR Profile 4 (Signal-Based)” 205
Figure 143: Custom E2E –parameters of a stack based custom E2E-Profile 207
Figure 144: “RBS based Replay” dialog 212
Figure 145: “Database Update” menu 215
Figure 146: Database Update – “Change database” dialog 216
Figure 147: Database Update – “Update Database” dialog 216
Figure 148: Database Update – “Reconvert database” dialog 218
Figure 149: Change Device Type 219
Figure 150: Save Project changes dialog 219
Figure 151: “Change device …” dialog (here source FL3X Device-S, target FL3X Device-L) 220
Figure 152: “Create Partner Project” 222
Figure 153: Installing the Visual Studio Code extension 223

Figure 154: Visual Studio Code – FL3X Config Integration –FL3X Config project folder....224
Figure 155: Visual Studio Code – FL3X Config Integration –FL3X Config project initialized....224
Figure 156: User Function Editor - Code Snippets selection....225
Figure 157: Visual Studio Code – FL3X Config Integration - Code Snippets example 1....225
Figure 158: Visual Studio Code – FL3X Config Integration - Code Snippets example 2....225
Figure 159: Visual Studio Code – FL3X Config Integration - Code Snippets example 3....225
Figure 160: Visual Studio Code – FL3X Config Integration - Code Snippets example 4....226
Figure 161: Visual Studio Code – FL3X Config Integration - Code Snippets example 5....226
Figure 162: Selecting ECU-Family Handlings....227
Figure 163: PDU-Container Layout [AUTOSAR Specification I-PDU Multiplexer]....229
Figure 164: Flowchart for container PDU handling....230
Figure 165: Settings Dialog....235
Figure 166: Application Settings – Common Settings....236
Figure 167: Application Settings – Saved Data....237
Figure 168: Application Settings – Update Interval....238
Figure 169: Application Settings – Open File Settings....238
Figure 170: Application Settings – Proxy Settings....240
Figure 171: Application Settings – Device Manager....241
Figure 172: Project Settings – Common Settings....242
Figure 173: Project Settings – Device Default IP-Settings....244
Figure 174: Template Settings – Device Default IP-Settings....245
Figure 175: Template Settings – Standard Signal Handling....245
Figure 176: Template Settings – Initial Value Settings....246
Figure 177: Multi OEM Support – Select OEM for Bus....249
Figure 178: Multi OEM Support – Select OEM for Transparent Gateway....249
Figure 179: “FL3X Config Gateway” editor....250
Figure 180: FL3X Config Gateway – Toolbar for the “Source” section....251
Figure 181: FL3X Config Gateway – Toolbar for the “Target” section....252
Figure 182: FL3X Config Gateway – Toolbar for the “Gateway routings” section....252
Figure 183: FL3X Config Gateway – Create a routing....253
Figure 184: FL3X Config Gateway – Create parent routing....254
Figure 185: FL3X Config Gateway – Column filtering....258
Figure 186: FL3X Config Gateway – Cluster view button....259
Figure 187: FL3X Config Gateway – Show/Hide columns....260
Figure 188: FL3X Config Gateway – Change column order....261
Figure 189: FL3X Config Gateway – Set color to highlight items....262
Figure 190: FL3X Config Gateway – Enable/Disable scaling mode....262
Figure 191: FL3X Config Gateway – “Signal Scaling” dialog....263
Figure 192: FL3X Config Gateway – “Signal Scaling” editor – Interval mask....264
Figure 193: FL3X Config Gateway – Open the “Create Autoroutings” dialog....265
Figure 194: FL3X Config Gateway – “Create Autoroutings” dialog without ethernet buses.....266
Figure 195: FL3X Config Gateway – “Create Autoroutings” dialog with ethernet buses.....266
Figure 196: FL3X Config Gateway – Create child routings of an existing routing....267
Figure 197: FL3X Config Gateway – Dialog to create child routings....268
Figure 198: FL3X Config Gateway – Delete routings in the routings list....268
Figure 199: FL3X Config Gateway – Delete routings in the source or target list....269
Figure 200: FL3X Config Gateway – “Delete Routings” dialog....269
Figure 201: FL3X Config Gateway – Export Items....270
Figure 202: FL3X Config Gateway – “Export Items” dialog....270

Figure 203: FL3X Config Gateway – “Print Items” dialog 271
Figure 204: FL3X Config Gateway – Import/Export Routings 272
Figure 205: FL3X Config Gateway – Import/Export Routings Dialog 272
Figure 206: Routing of PDUs with different PDU layout 273
Figure 207: Routing of frames with different frame layout 274
Figure 208: Main window with transparent gateway 276
Figure 209: Warning dialog of transparent gateway 276
Figure 210: Transparent Gateway configuration dialog 278
Figure 211: FL3X Config Control 282
Figure 212: FL3X Config Control – Manipulation control 283
Figure 213: FL3X Config Control – Toolbar for the “Manipulable Entities” section 284
Figure 214: FL3X Config Control – Cluster view button 284
Figure 215: FL3X Config Control – Create a manipulation via drag & drop 285
Figure 216: FL3X Config Control – Create a manipulation via context menu 285
Figure 217: FL3X Config Control – Clone a manipulation via context menu 285
Figure 218: FL3X Config Control – Manipulation configuration options 286
Figure 219: FL3X Config Control – Manipulation is running and is selected 287
Figure 220: FL3X Config Control – Manipulation is running and is not selected. 287
Figure 221: Manipulations – Example for value limits 288
Figure 222: Example: Manipulated signal with method “Loss” 290
Figure 223: Example: Manipulated signal with method “Disable Manipulation” 291
Figure 224: Example: Manipulated signal with method “Factor” (raw) 292
Figure 225: Example: Manipulated signal with method ‘Freeze’ 292
Figure 226: Example: Manipulated signal with method “FixValue” 293
Figure 227: Example: Manipulated signal with method “Drift” (Duration: 25, Delay: 0, Hold final value: not active) 294
Figure 228: Example: Manipulated signal with method “Drift” (Duration 30, Delay: 0, Hold final value: active) ... 294
Figure 229: Example: Manipulated signal with method “Offset” 296
Figure 230: Example: Manipulated signal with method “Ramp” (Duration: 25, Delay: 0) 297
Figure 231: Example: Manipulated signal with method “Ramp” (Duration: 28, Delay: 0, Hold final value: active) 297
Figure 232: Example for the option “Trigger Cyclic” 302
Figure 233: FL3X Config Control – Sequence control 303
Figure 234: FL3X Config Control – Clone a sequence via context menu 304
Figure 235: FL3X Config Control – Active sequence 305
Figure 236: FL3X Device-PXIe – DIO connectors 320
Figure 237: FL3X Update - Device Selection 322
Figure 238: FL3X Update - License View 323
Figure 239: FL3X Device revisions – label with device information 334
Figure 240: Example for connecting a FL3X Device via WiFi 334

STAR ELECTRONICS GmbH & Co. KG
A Company of the STAR COOPERATION Group
Jahnstrasse 86
73037 Goeppingen
Germany
Phone: +49 (0) 7031 6288-5656
sales-ee@star-cooperation.com
www.flex-product.com