


Features and improvements V3.4 SP3

CoDeSys

Public
18.03.2011

Smart Software Solutions

We software Automation.

Contents

Batch mode / Scripting

- CoDeSys V3 now includes a scripting language based on Python.
- This means you have the same functionality as in CoDeSys V2.3 plus the advantages of a real programming language combined with the vast possibilities of the Python standard library.
- Applications:
  - Automatic import of project parts
  - Automatic compilation and download of PLC applications

Python as scripting language

Advantages

- Easy syntax
  - Existing CoDeSys V2.3 batches can easily be converted
- Support of complex program structures
  - structured, object-oriented and functional programming is possible
  - comprehensive scripts and script libraries can be realized
- Widespread use
- Link for more information: http://www.python.org/

Comparison between V2 and V3

- CoDeSys V2.3 batch mode – very common - but
  - No standard
  - No variables
  - No control structures (except of subroutines)
  - No synchronous online functions
- CoDeSys V3
  - Python as scripting language
  - Powerful programming structures (variables, loops)
  - Comprehensive API functions
    - Access to CoDeSys objects
    - Optional – synchronous online functions
    - Python standard library (file access etc.)

Scripting – libraries

System

- PLC environment:
  Trace
  Activate user interface
- User requests
- Wait

Project

- Open, save, close
- Search objects and list
- Print, export and import
- Delete, rename and move
- Properties

LibMan

- List of libs
- Add or delete libraries

Device

- Add, insert or plug device
- Device properties:
  Communi-cation
  Simulation

Online

- Login, logout, start, stop
- Download, Boot application
- Read, write and force variables
- Status query:
  PLC status
  Communication abort

Scripting – execute

- Via task menu

- Via command line

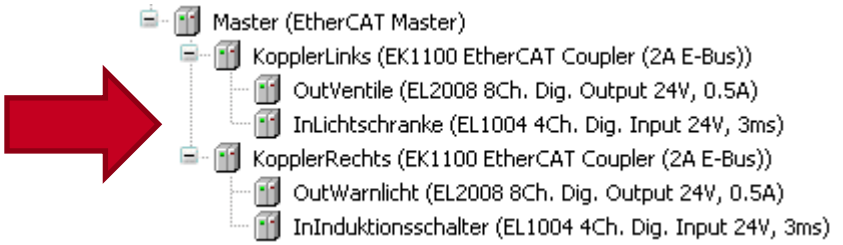
CoDeSys --profile="CoDeSys V3.4 SP3" --runscript="*path_to_sample_script.py*"
[--enablescripttracing] [--noUI]

Scripting – example

File generation of a fieldbus configuration based on a description file

```
Master (EtherCatMaster)
    KopplerLinks (EK1100)
        OutVentile (EL2008)
        InLichtschranke (EL1004)
    KopplerRechts (EK1100)
        OutWarnlicht (EL2008)
        InInduktionsschalter (EL1004)
```

```
print "Current project."
proj = projects.primary
for dev in proj.get_children():
    if dev.is_device:
        f = open("C:/Users/d.hess/Desktop/Konfiguration.txt", "r")
        for line in f:
            tok = line.split(' ')
            print tok[1], tok[0]
                if "(EtherCatMaster)" in tok[1]:
                dev.add(tok[0], DeviceID(64, "0000 0001", "3.4.2.0"))
                subnodes = dev.get_children()
                master = subnodes[len(subnodes) - 1]
                elif "(EK1100)" in tok[1]:
                master.add(tok[0], DeviceID(65, "2_044C2C5200110000",
"Revision=16#00110000"))
                subnodes = master.get_children()
                coupler = subnodes[len(subnodes) - 1]
                elif "(EL2008)" in tok[1]:
                coupler.add(tok[0], DeviceID(65, "2_07D8305200100000",
"Revision=16#00100000"))
                elif "(EL1004)" in tok[1]:
                coupler.add(tok[0], DeviceID(65, "2_03EC305227080000",
"Revision=16#27080000"))
print "script finished."
```



Definition of PLCopen XML

- Exchange of function blocks, libraries and projects between different tools, e.g.
  - Debugging Tools
  - Simulators
  - Documentation tools
  - Modelling tools
- Transport without losing information
  - Logical information
  - Graphical information
  - Manufacturer specific information
- Source of the XML codes unimportant
  - XML description is complete
  - Filtering of the required information via the importing tool

PLCopen XML

A “native“ import/export in CoDeSys already exists, but

- it is hardly human readable
- it includes constraints, which have to be observed
- it is only “documented“ by the export
- it can be different from version to version

… on the other hand it is always…

- complete

… but then again not suitable for all applications.

```
<Single Name="TypeGuid" Type="System.Guid">6f9dac99-8de1-4efc-8465-
68ac443b7d08</Single>
      <Array Name="EmbeddedTypeGuids" Type="System.Guid">
        <Single Type="System.Guid">a9ed5b7e-75c5-4651-af16-
d2c27e98cb94</Single>
        <Single Type="System.Guid">3b83b776-fb25-43b8-99f2-
3c507c9143fc</Single>
      </Array>
      <Single Name="Timestamp" Type="long">633439547036984297</Single>
    </Single>
    <Single Name="Object" Type="{6f9dac99-8de1-4efc-8465-68ac443b7d08}"
Method="IArchivable">
      <Single Name="SpecialFunc" Type="{0db3d7bb-cde0-4416-9a7b-
ce49a0124323}">None</Single>
      <Single Name="Implementation" Type="{3b83b776-fb25-43b8-99f2-
3c507c9143fc}" Method="IArchivable">
        <Single Name="TextDocument" Type="{f3878285-8e4f-490b-bb1b-
9acbb7eb04db}" Method="IArchivable">
          <Array Name="TextLines" Type="{a5de0b0b-1cb5-4913-ac21-
9d70293ec00d}">
            <Single Type="{a5de0b0b-1cb5-4913-ac21-9d70293ec00d}"
Method="IArchivable">
              <Single Name="Id" Type="long">22</Single>
```

PLCopen XML

Therefore we now have a new PLCopen XML import/export which is …

- standardized
- human readable
- defined for text and graphical languages
- with CoDeSys specific enhancements (e.g. devices, interfaces, methods)
- also available in other IEC 61131-3 tools

… and even though it is…

- not really complete

… it is suitable for almost all applications.

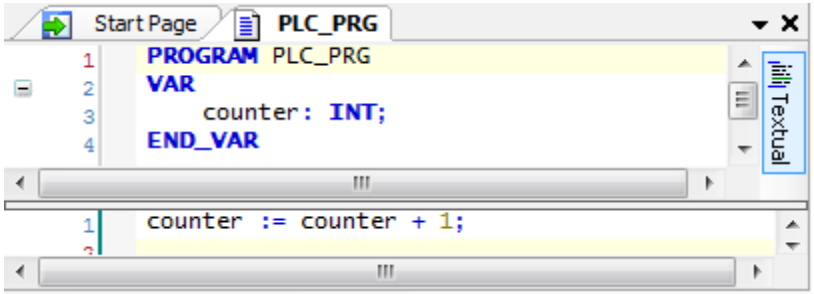
PLCopen XML

Applications

- Exchange format between IEC programming tools
  - Data exchange between different development platforms
  - Parallel usage of different programming environments
- Interface for “suppliers“ of graphical or logical information
  - From higher-level engineering tools, generating IEC code / data
  - Possibility to keep the connection to some elements
- Interface for “consumers“ of graphical or logical information
  - Examples: Validation tools, compilers, SCADA/HMI tools, documentation tools, translation tools
  - Filters the required information from the complete XML file
- Distribution format for function block libraries

PLCopen XML

Example of a PLCopen XML export




```
<pou name="PLC_PRG" pouType="program">
  <interface>
    <localVars>
      <variable name="counter">
        <type>
          <INT />
        </type>
      </variable>
    </localVars>
  </interface>
  <body>
    <ST>
      <xhtml xmlns="http://www.w3.org/1999/xhtml">counter := counter + 1;
</xhtml>
    </ST>
  </body>
</pou>
```

IEC program: Simple counter

PLCopen XML file

PLCopen XML in CoDeSys

- Realized object types:
  - POUs (incl. object-oriented enhancements)
  - Interfaces (CoDeSys-specific enhancement)
  - Actions
  - Methods (CoDeSys-specific enhancement)
  - Properties (CoDeSys-specific enhancement)
  - Transitions (CoDeSys-specific enhancement)
  - Global variables
  - Data types
  - Tasks (incl. Unions)
  - Devices (not complete, CoDeSys-specific enhancement)
  - Applications (not complete, CoDeSys-specific enhancement)
  - Project information (CoDeSys-specific enhancement, only export)
- Realized programming languages:
  - ST
  - FBD
  - CFC

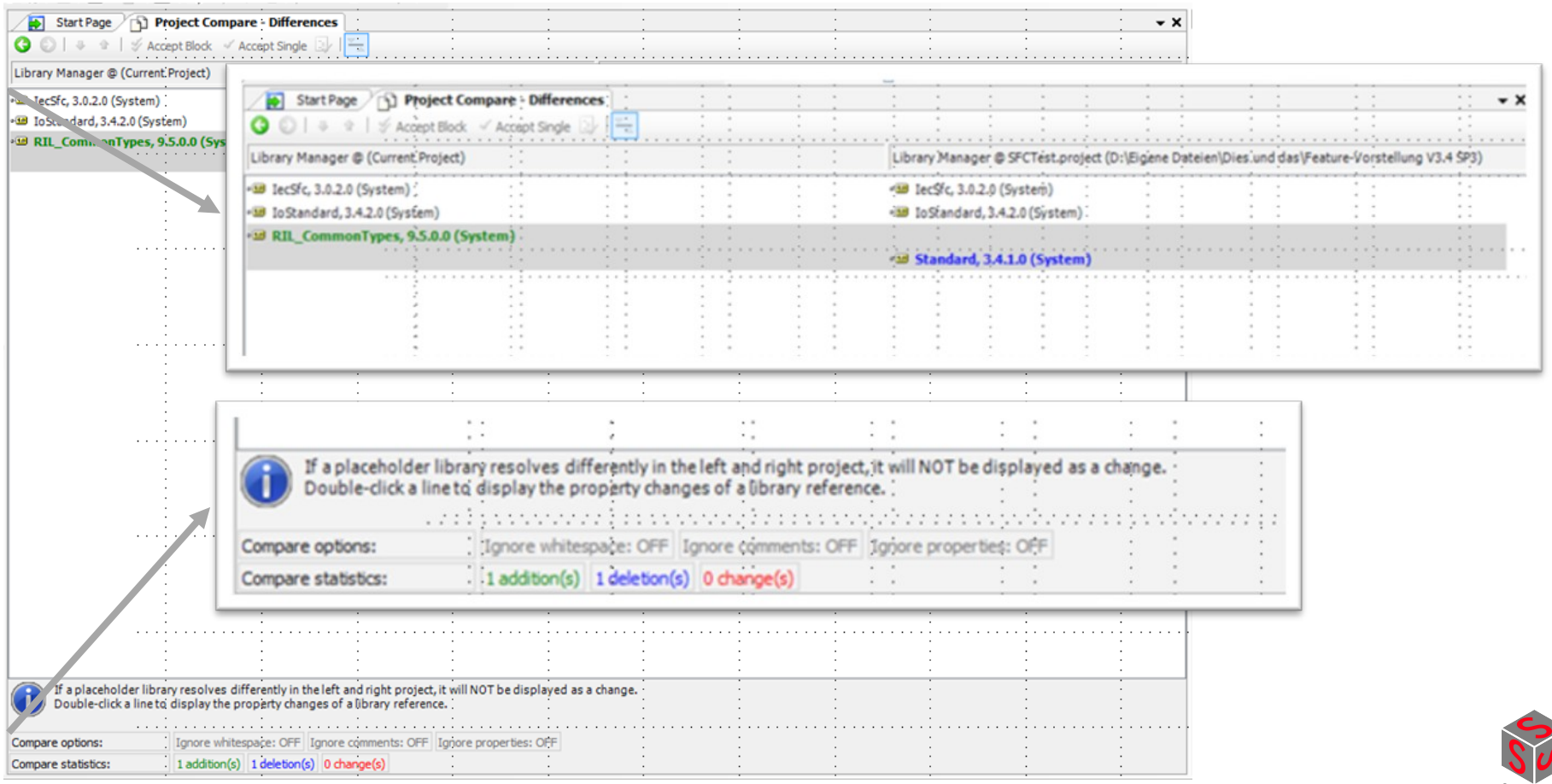
PLCopen XML - comparison

	"Native" XML	PLCopenXML
Human readable?	no	yes
Usable for external tools?	yes, only with automation platform	yes
Loss free?	yes	no
Complete?	yes	Enhancements in the project format are not directly available in PLCopen XML

PLCopenXML is not suitable for an exact storage of a CoDeSys project.

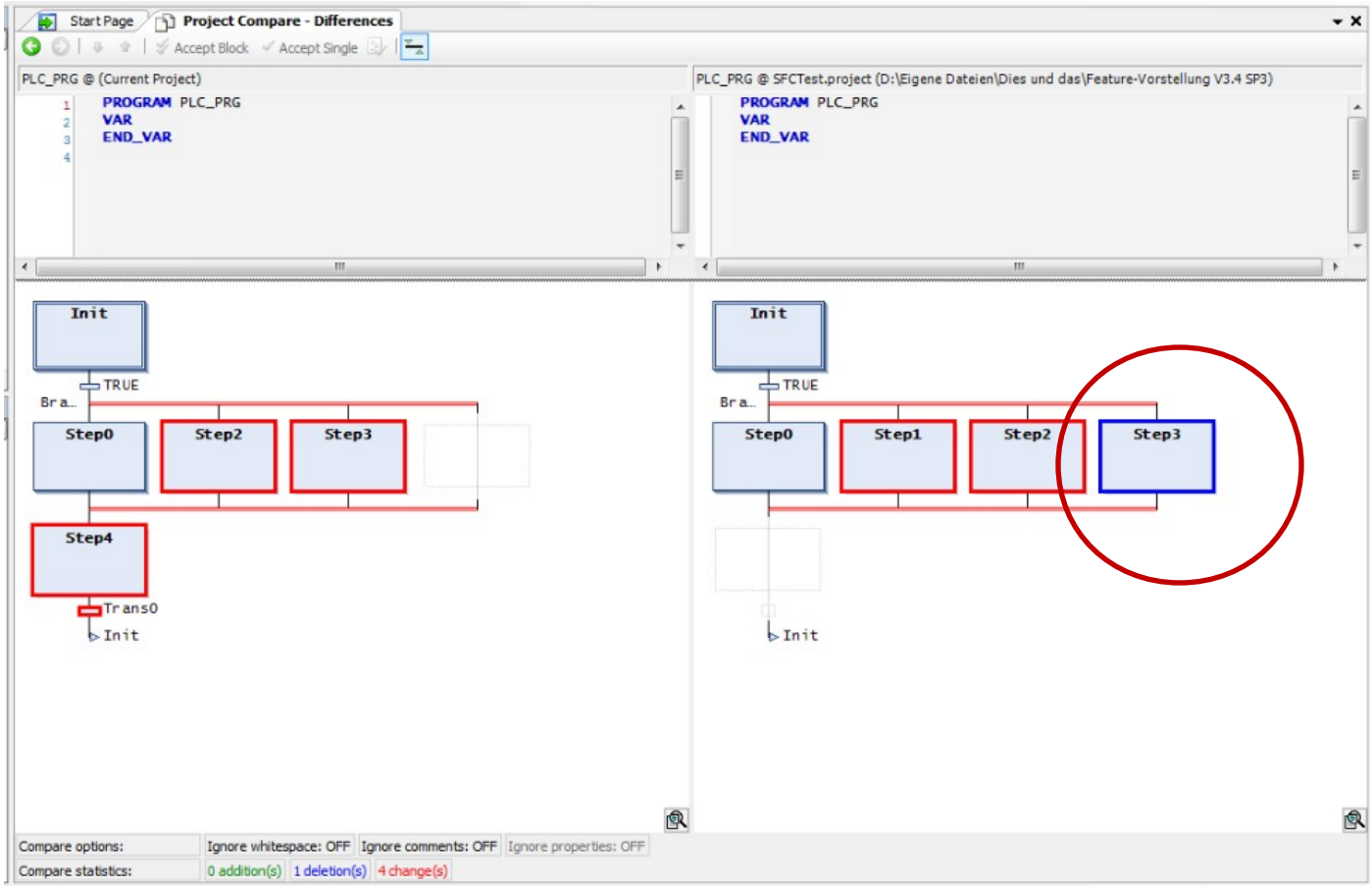
Engineering: User interface and editors

Project comparison: Graphical diff editor for library manager



Engineering: User interface and editors

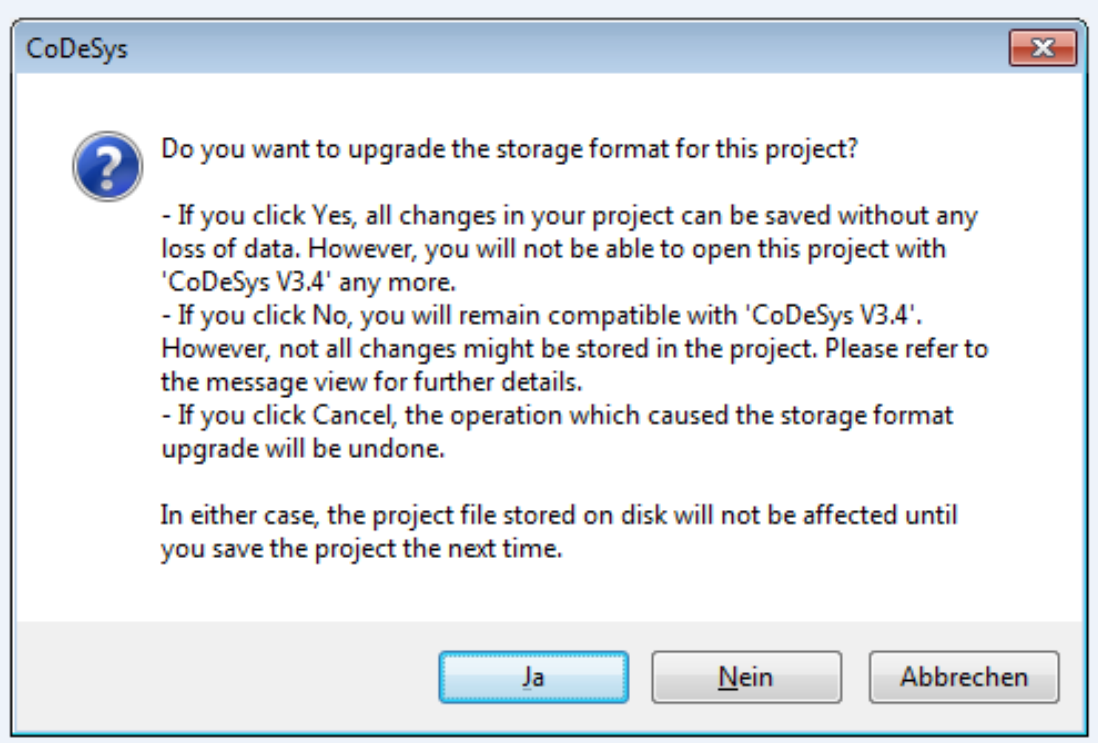
Project comparison: Graphical diff editor for SFC



Engineering: User interface and editors

Project compatibility: Improved usability

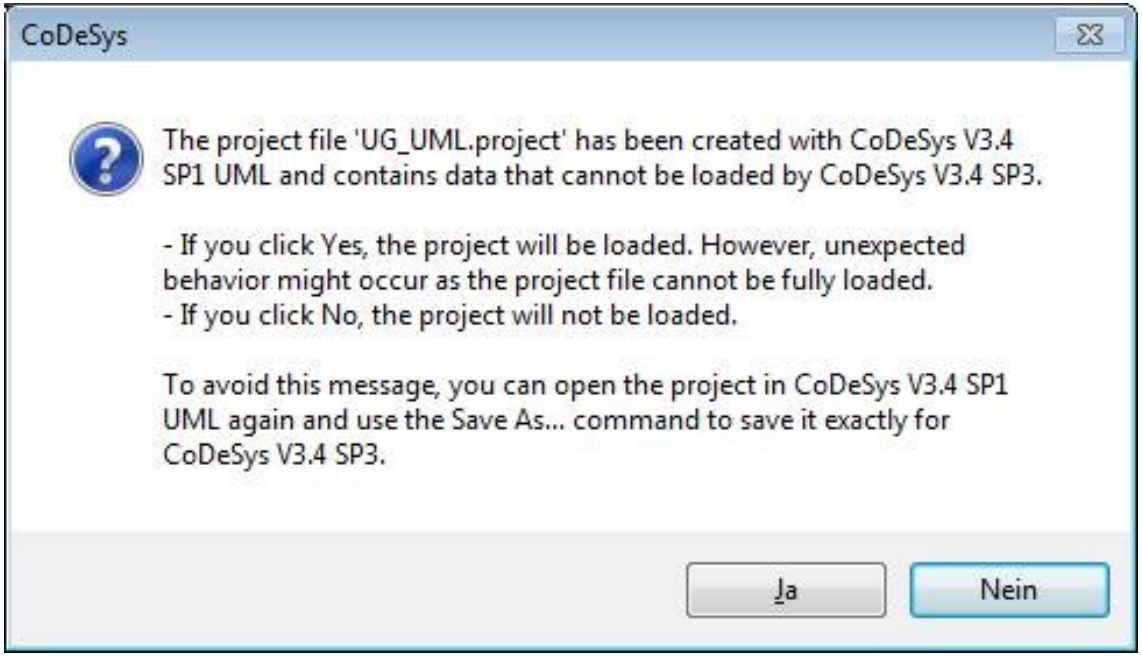
- Edit operations which would require a project storage format upgrade can be undone with the function “Cancel”.



Engineering: User interface and editors

Project compatibility: Improved usability

- Projects created with a new CoDeSys version can no longer be opened without having to confirm a warning message. In previous versions, the user was often unaware of a possible data loss under certain circumstances.



Engineering: User interface and editors

Package Manager:

- Library profiles are now supported
- Setting of options is now supported

Minor usability improvements:

- Project archives can now be extracted even if a second instance of CoDeSys is running. Library update messages are now displayed in the Message view instead of in a dialog box.
- The editor caption now shows *POU.Name* instead of *Name* for actions, methods, properties of transitions.

Online change

New functionality

- Online change with optimized jitter
  - Reduction of jitter through optimized change procedure
  - Display of changes
- Alternative storage allocation algorithm with a complete download during online change:
  - No additional loading of the boot project
  - No memory fragmentation
  - CRC (cyclic redundancy check) of the program code is possible

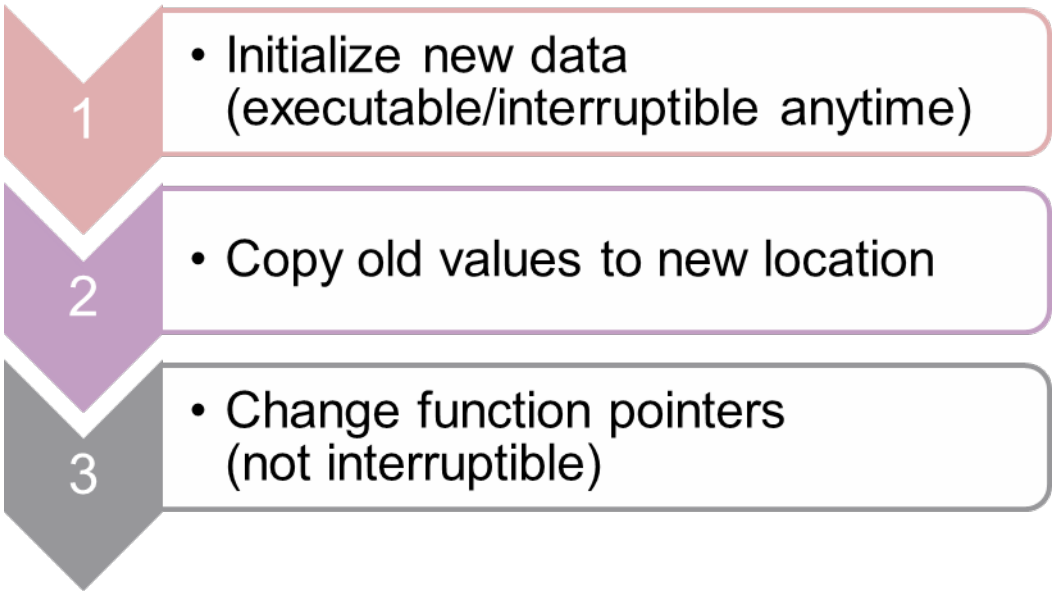
Online change

Requirements:

- In order to use this new feature a runtime update is required and the device description settings "runtime_features//optimized_online_change" have to be configured.

Online change

Process simplified:



Idea:

1 Easy call up

2 Call up in blanking interval; repeat if interrupted

3 Stop task scheduling in blanking interval and then execute

→ Minimum jitter

Online change

Restrictions:

- Initialization, cannot be done while the task is running, if an
  - initialization is not constant
  - FB_EXIT is necessary
  - FB_INIT contains cross references to moved code
  - FB_INIT is called in the IEC-Cycle
  - FB_INIT contains virtual method calls

Online change

Real process

1. • “Constant“ initialization (concurrent)
2. • Copy code for “constant“ initialization (repeatable)
3. • Initialization and copy code for non-constant initialization (interrupt-safe)
4. • Function pointer initialization (interrupt-safe)

Online change

Conclusion:

- All online change procedures are divided into executable, interruptible and non-interruptible steps.
- The different function block instances will be differentiated, and external references will be verified. An external reference is an access to global variables and external calls.
- Then the runtime system can call up the different parts.

Online change

Examples for online change events without jitter:

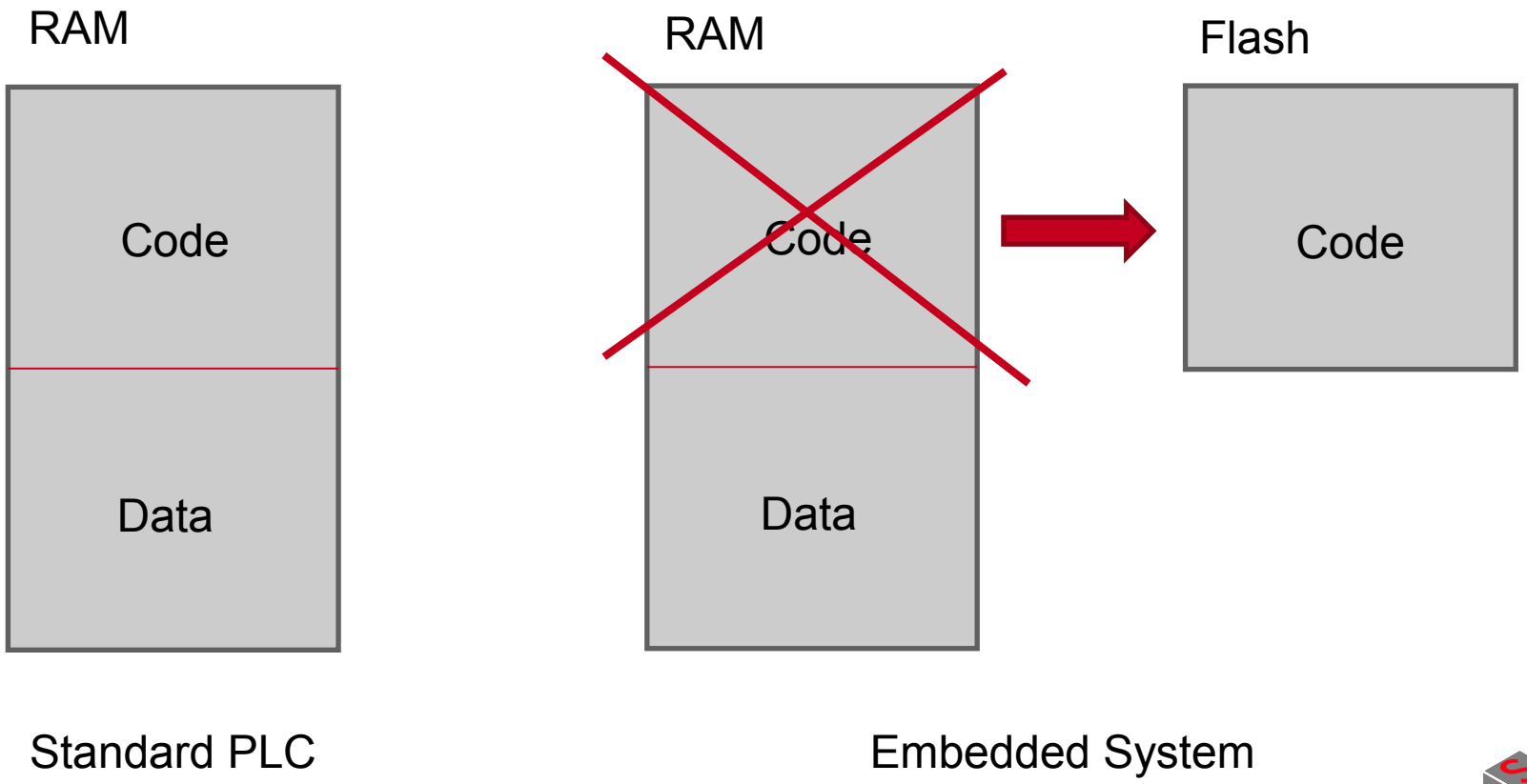
- Change size of existing arrays (even really big ones)
- Add new data of any volume
- Call new functions (a complete new call tree will have no effect on the running IEC-tasks)
- Changes in old style function blocks without FB_Init or FB_Exit-methods.

Transfer of symbol config as XML file

- Transfer of the “symbol configuration“ as XML file onto the controller (now contains area/offset/bit and name of direct addresses)
- Symbol configuration is required for external symbol access to IEC variables
- Advantage:
  - no code generation on PLC
  - no use of internal memory

Run from Flash for embedded systems

- Run from Flash for embedded systems



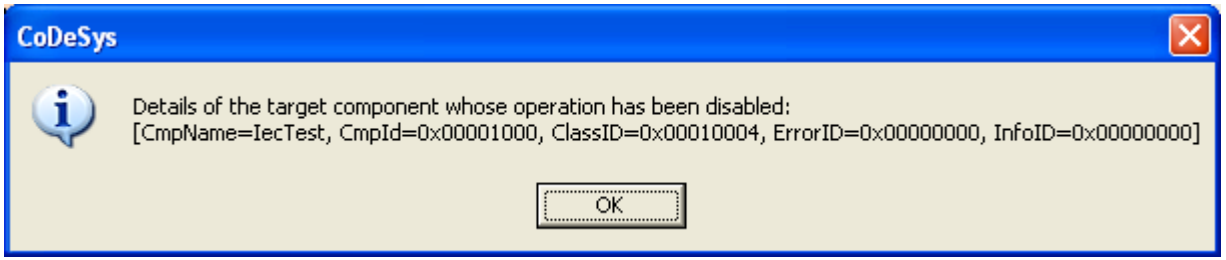
Run from Flash for embedded systems

- Advantages:
  - Especially for embedded systems, in respect of the reduced memory requirements for the RAM

- Disadvantages:
  - No debugging possible
  - No online change possible

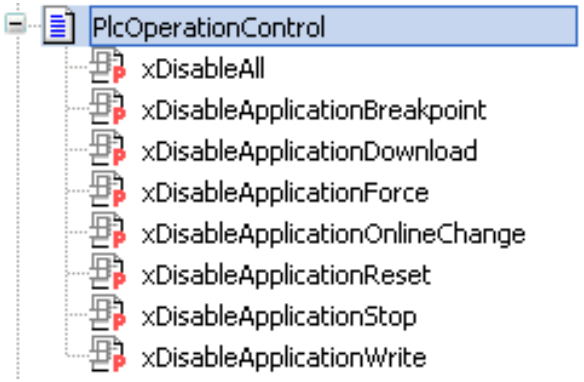
CoDeSys message box

- Generation of message boxes through logger entries or IEC applications during online operation



Operation control to increase security

- Access to the PLC can be controlled via application in case of a critical condition to ensure a safe plant operation.
- Application is able to prevent following operations, depending on the status
  - Online Change
  - Force
  - Breakpoints
  - Reset
  - Stop
  - Download
- Operations can be deactivated directly from IEC or C
- Delete of application and reset origin is always possible
- A message will be displayed in CoDeSys

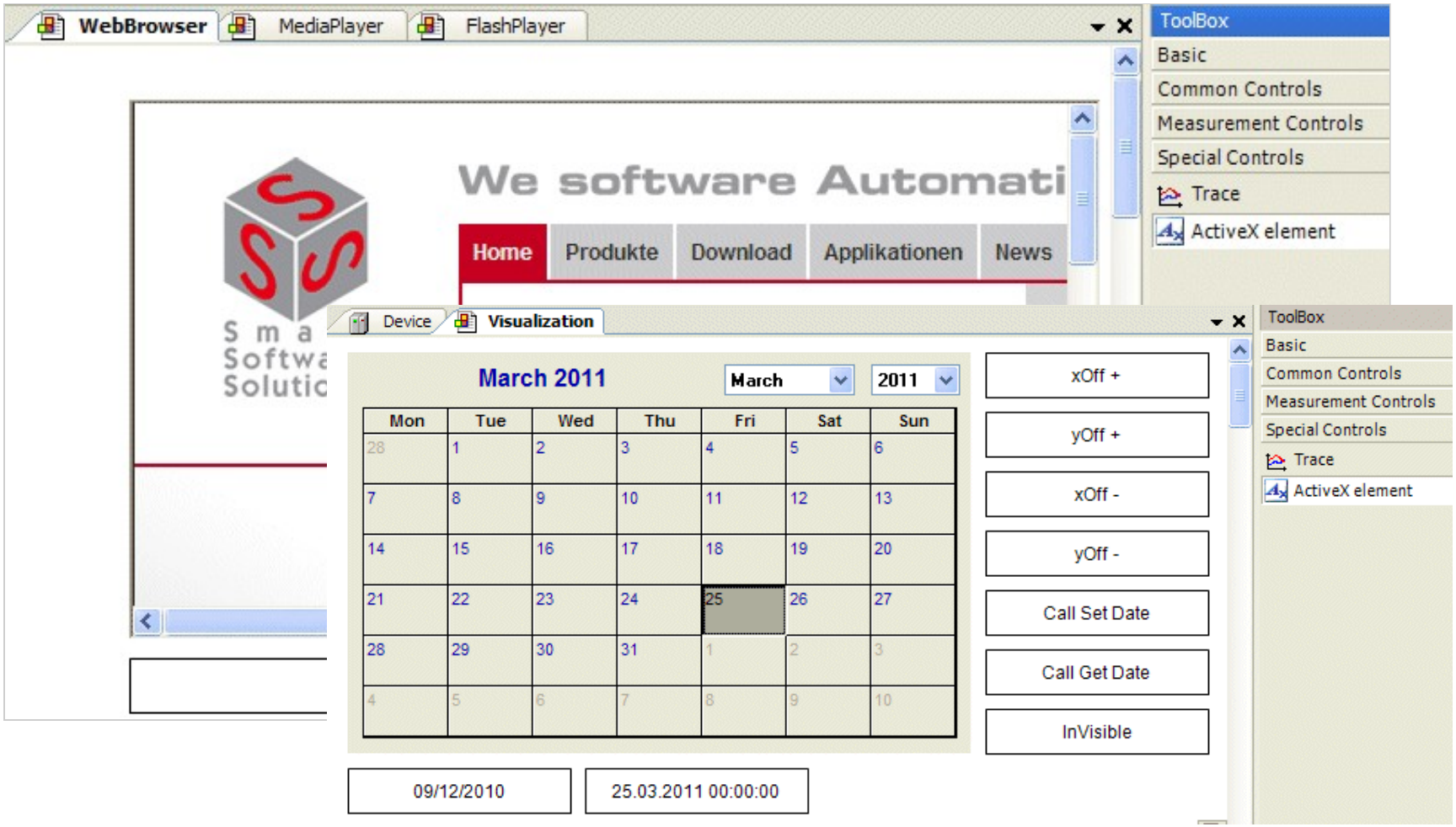


Visualization

- Visualization element “NativeControl”:
  For connection to ActiveX controls for Windows platforms or platform specific objects, like camera objects directly within a visualization.
- Platform-dependent controls in a visualization can be used via NativeControl (ActiveX element within the toolbox). Examples are web browser, calendar, media player, flash player, ...
- IEC variables can be passed to the element and the element can send back a result via a common interface.

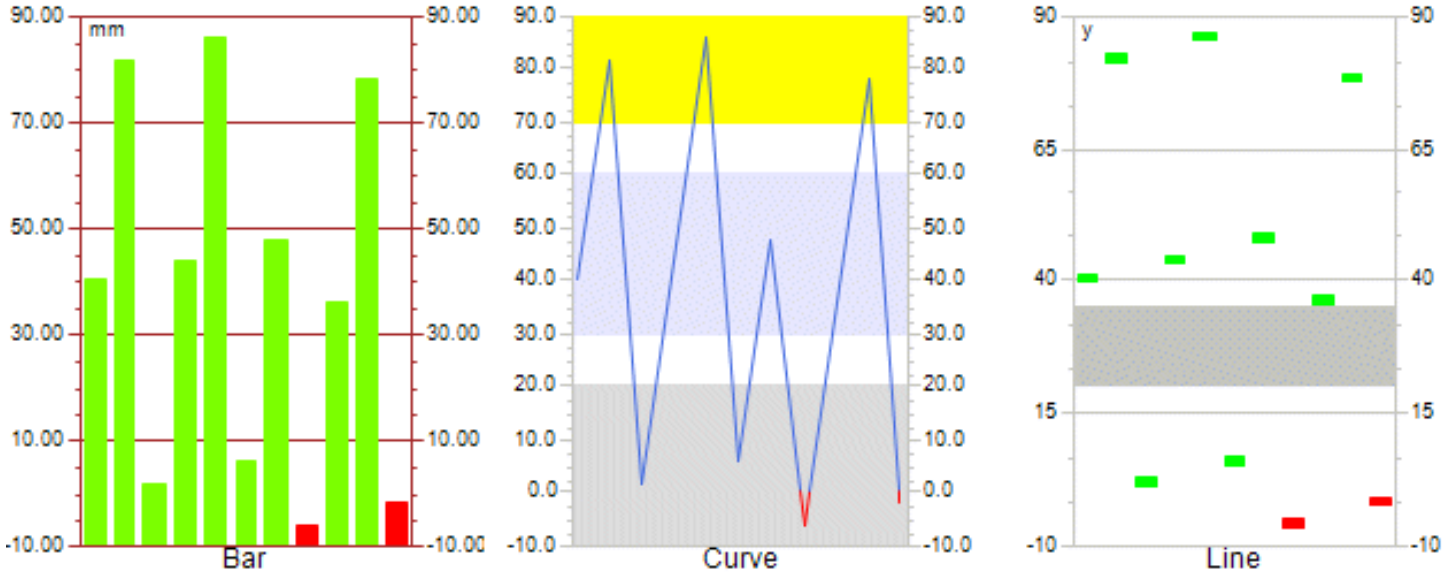
Visualization

- ActiveX element in the toolbox



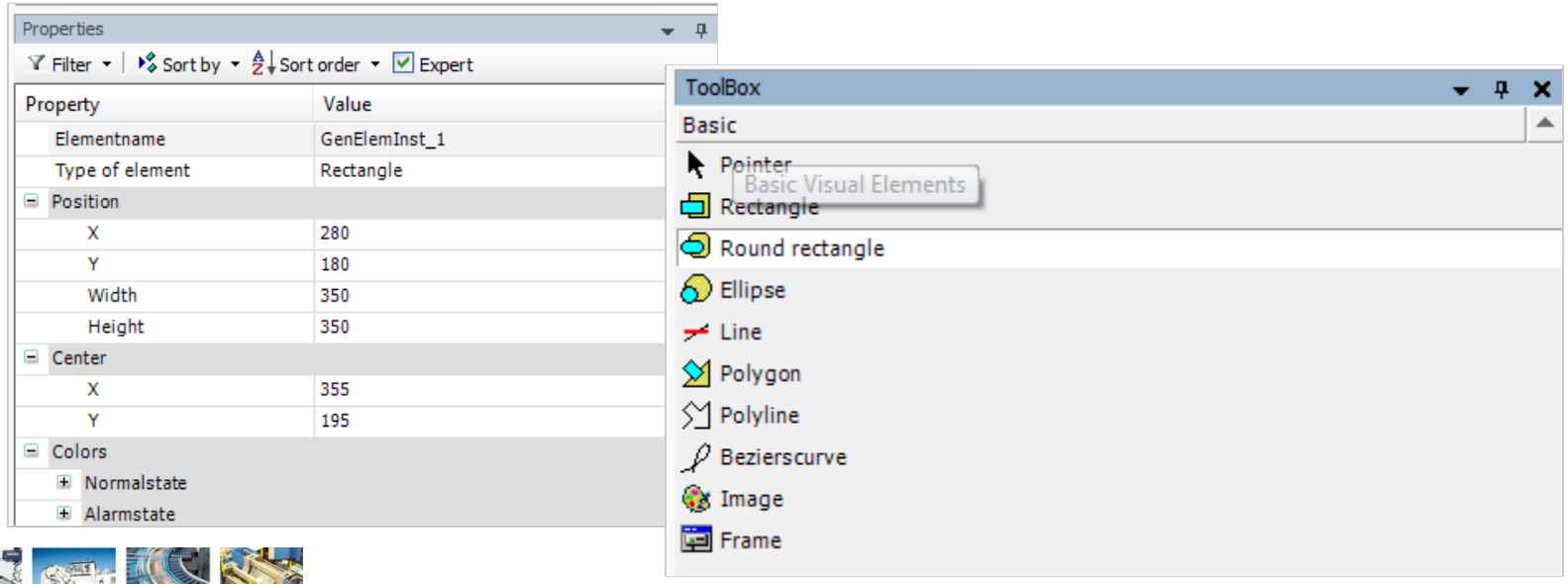
Visualization

- Visualization element “Histogram” now available – display of a numerical array as bargraph, curve or line.



Visualization

- When using the VisualElementToolkit to develop a visualization the texts of the different elements can now be supplied in different languages.
- The element properties as well as the toolbar are now available in German and English language.
- Other languages will follow.

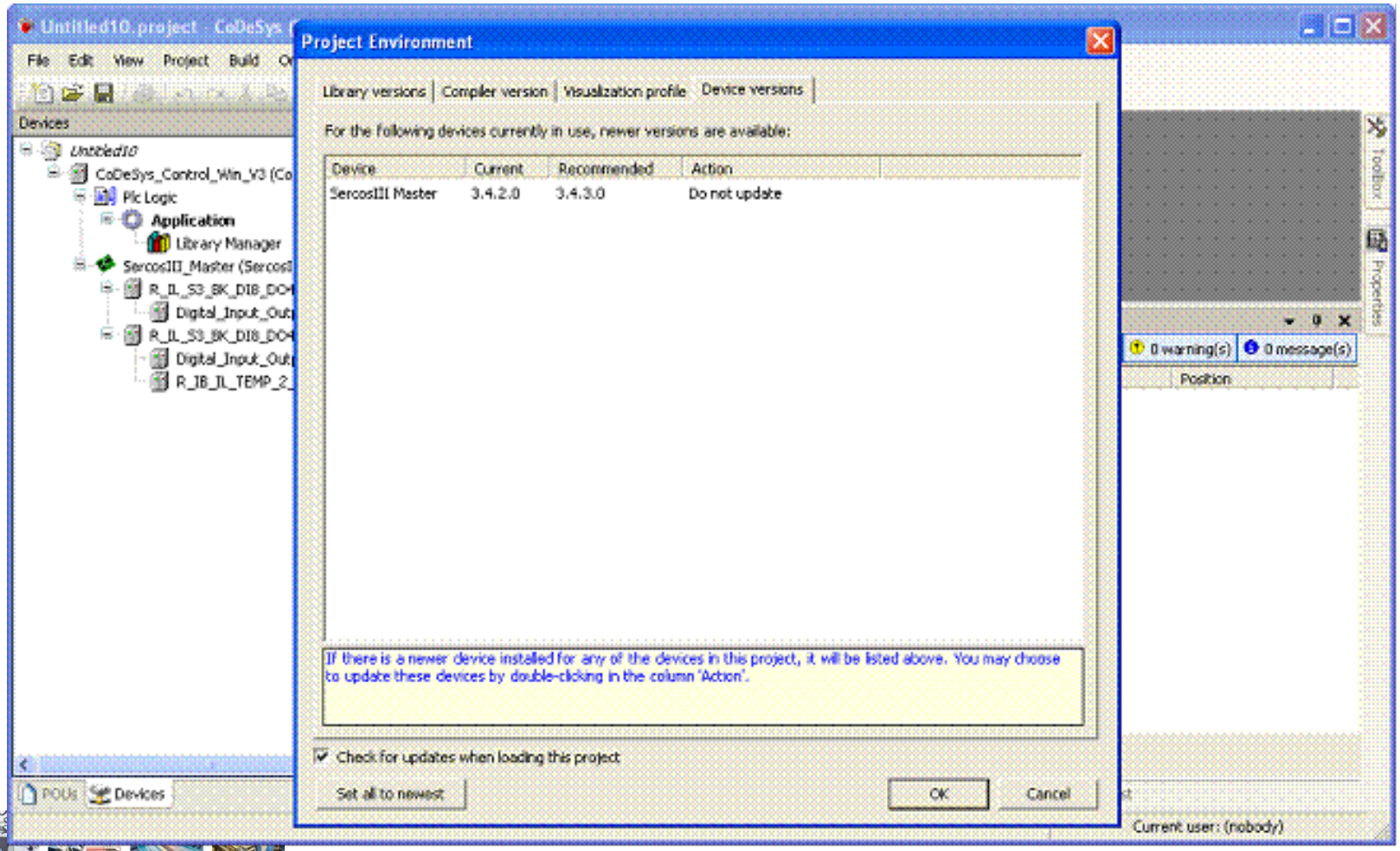


Fieldbus technology

- When opening a project an update of all devices will be offered.
- Ethernet over EtherCAT (EoE)

Fieldbus technology

- Device update when opening a project

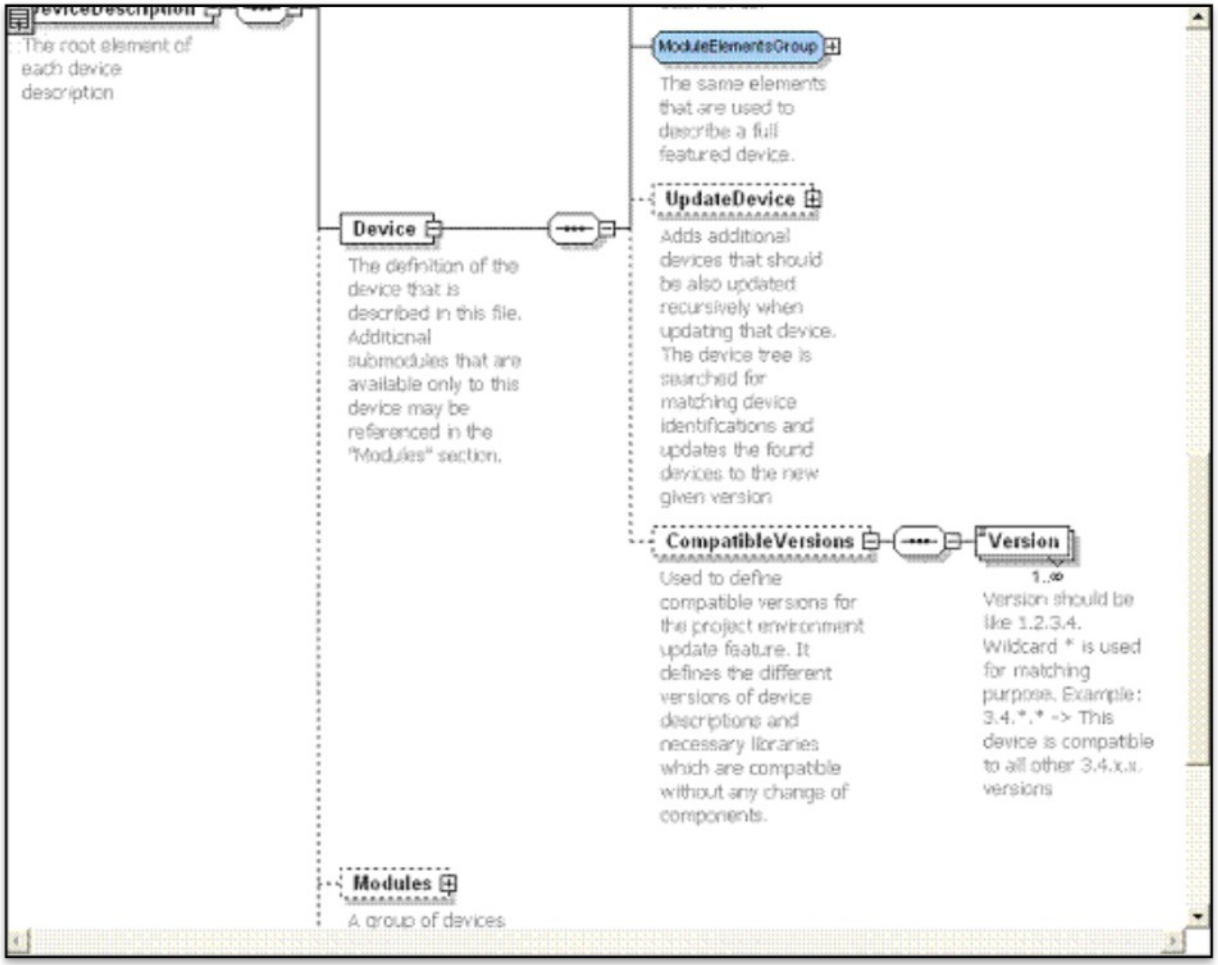


Fieldbus technology

- Device update when opening a project
  - Functionality is identical to functionality for libraries, visualizations and compilers.
  - Which one of the older versions is to be exchanged must be defined in the device description.
  - Only the supplier of the respective device can estimate the compatibility and can give a recommendation.

Fieldbus technology

- Entry in the .xml file:



Fieldbus technology

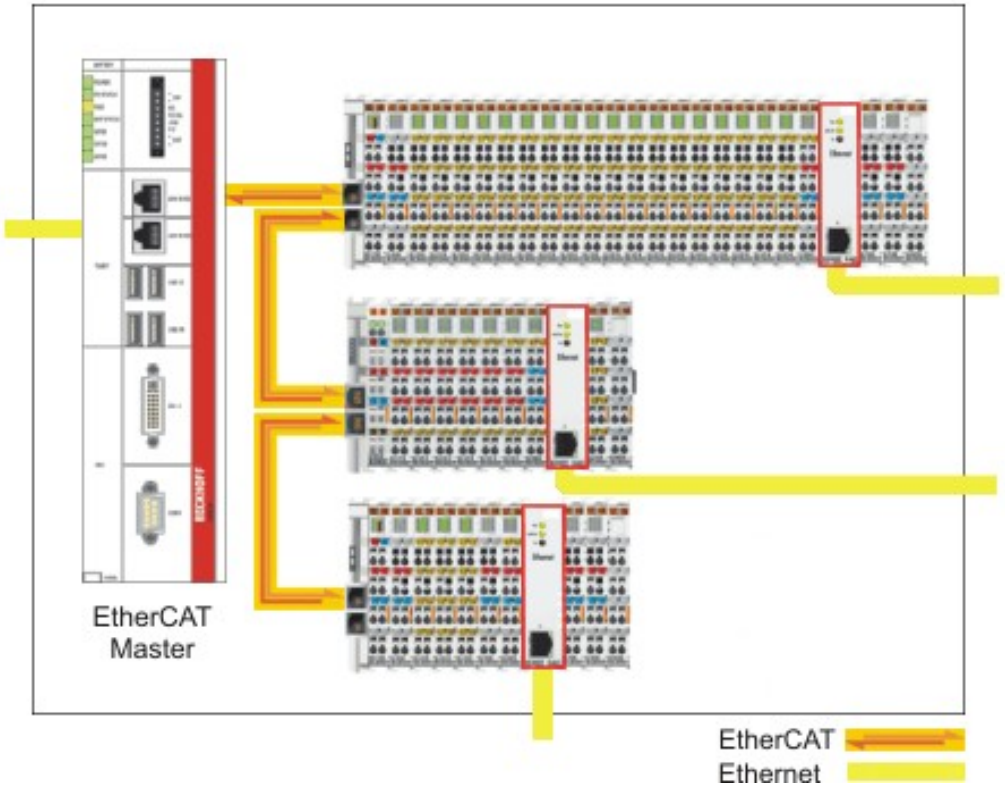
- **Definition: Ethernet over EtherCAT (EoE)**
- Any kind of Ethernet device can be connected within the EtherCAT segment via a switch port. The Ethernet frames are tunneled through the EtherCAT protocol, like it is known from internet protocols (e.g. TCP/IP, VPN, PPPoE (DSL))
- The EtherCAT network is completely transparent for the Ethernet devices and the EtherCAT realtime properties will not be affected.

Fieldbus technology

- Extension for EtherCAT: EoE (Ethernet over EtherCAT)
- Special switch port terminal (EL6601) or devices with EoE protocol (e.g. Indradrive CS) will be supported
- It is possible to connect standard network devices to the switch port terminal, like printer, PC …
- It is also possible to connect a Modbus TCP slave and to use our Modbus TCP stack without needing an additional network card.

Fieldbus technology

- EL6601 – Switch port terminal for Ethernet from Beckhoff

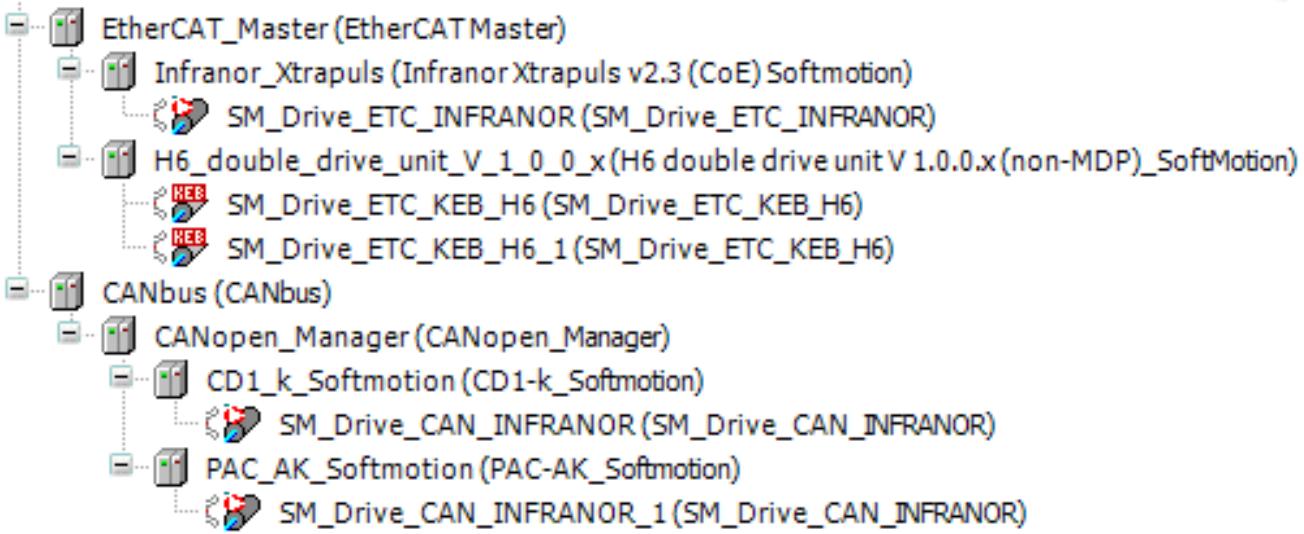


Fieldbus technology

- Indradrive Cs uses EoE for configuration purposes and to download firmware.
- Indraworks can therefore be connected to the servo controller via EtherCAT and no serial data communication is necessary.

The following drives are supported from Version V3.4 SP3:

- KEB H6 Dual (ETC)
- Infranor Xtrapuls PAC (ETC)
- Infranor Xtrapuls PAC (CAN)
- Infranor cd1-k (CAN)



CAM editor

- This feature displays the exact initialization code of a CAM table.
- It helps the customer to understand the way CAM tables are defined in CoDeSys.
- CoDeSys V2.3 offers this feature too.