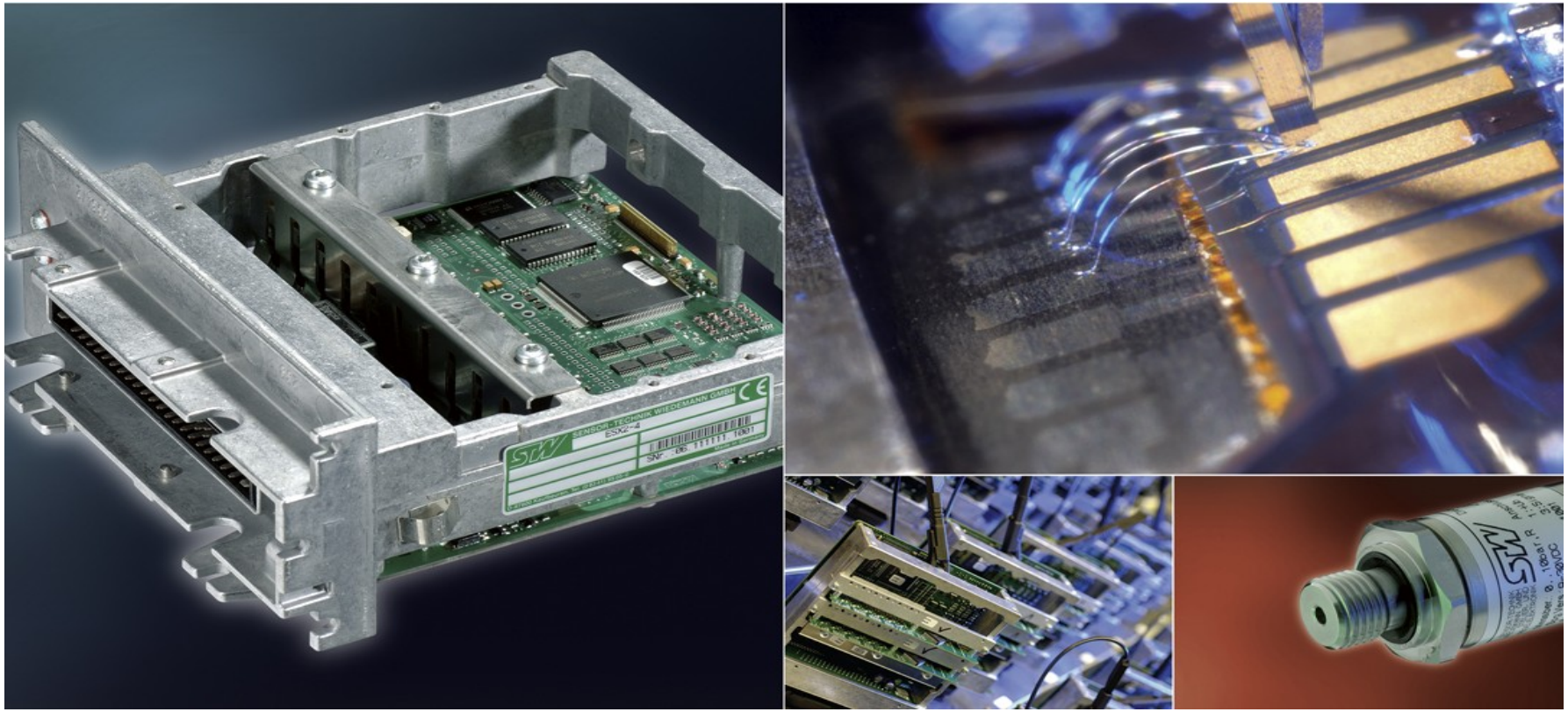
Training Course



CoDeSys V2.3

Topics

- Introduction
  - CoDeSys
  - IEC 61131
- IEC 61131-3 Programming languages
- Structure/Properties of a CoDeSys application
- Adaptions/Extensions by STW
- PLC Cycle
- Run-time system
- Tools
- Overview about supported ECU´s

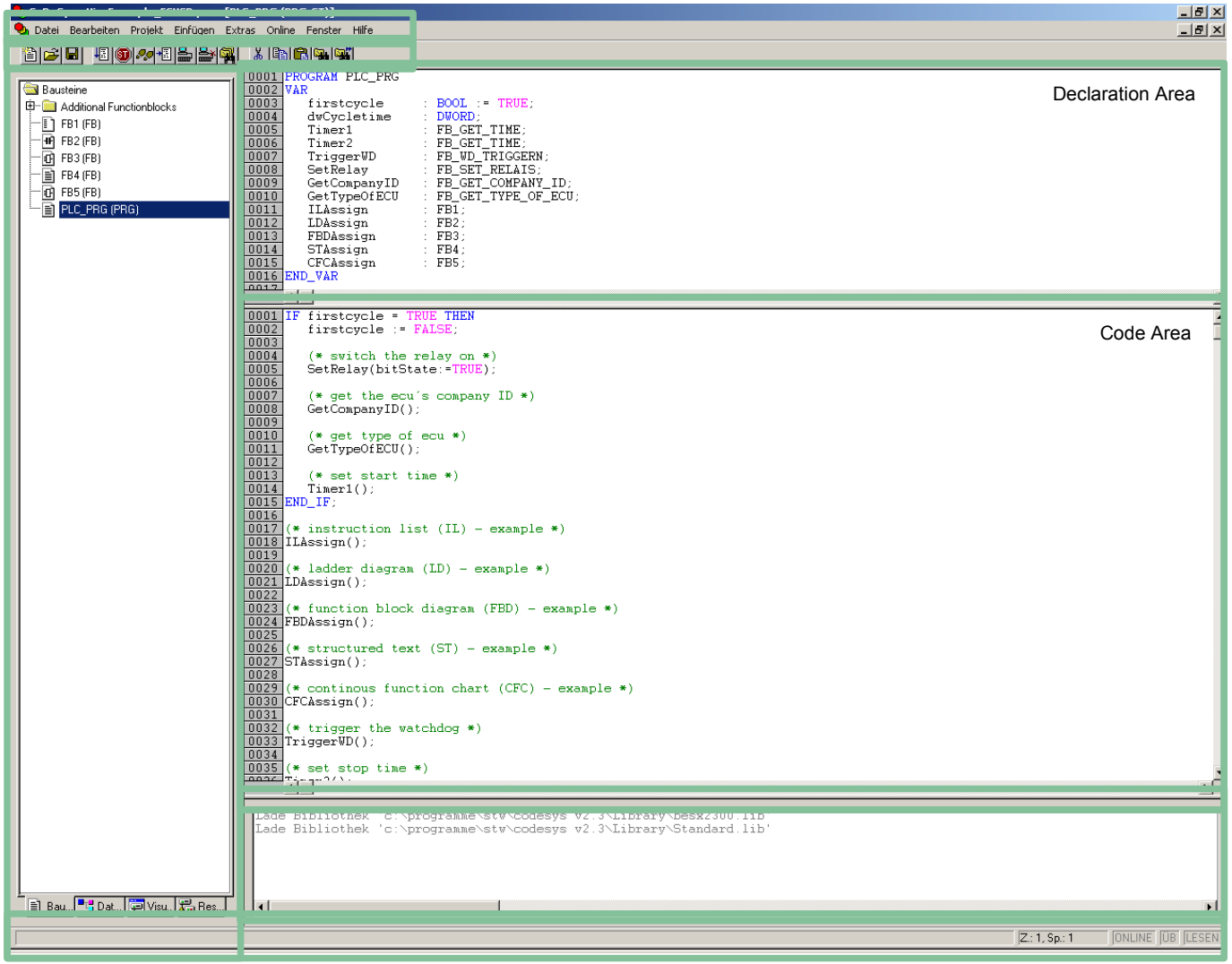
Introduction CoDeSys

- Product of the company 3S – Smart Software Solutions from Kempten
- Short cur for Controller Development System
- Complete integrated development environment (IDE) for PLC´s, consist of
  - Different editors (ST, FBD/CFC, SFC, IL, FUP, LD)
  - Built-in debugger
  - Tools for tests (Sampling Trace, Watch- and Recipe Manager)
  - Integrated visualization (HMI)
  - Gateway for communication with the PLC
- Conform to international IEC standard 61131-3
- Available for different versions of Microsoft Windows (98, W2K, XP)
- Actual at STW is version 2.3

Overview IEC 61131

- International standard, based on
  - Grafcet (France) and IEC 848
  - DIN 40719 (Germany)
  - NEMA ICS-3-304 (US)
  - DIN 19239 (Germany)
  - VDI 2880 (Germany)
- Structured in 5 parts
  - Part 1 – general informations
  - Part 2 – equipment and test requirements
  - Part 3 – Program languages
  - Part 4 – user guidelines
  - Part 5 - communication

Overview CoDeSys IDE



Menu Bar
Tool Bar

Object Organizer

Working Area

Message Window

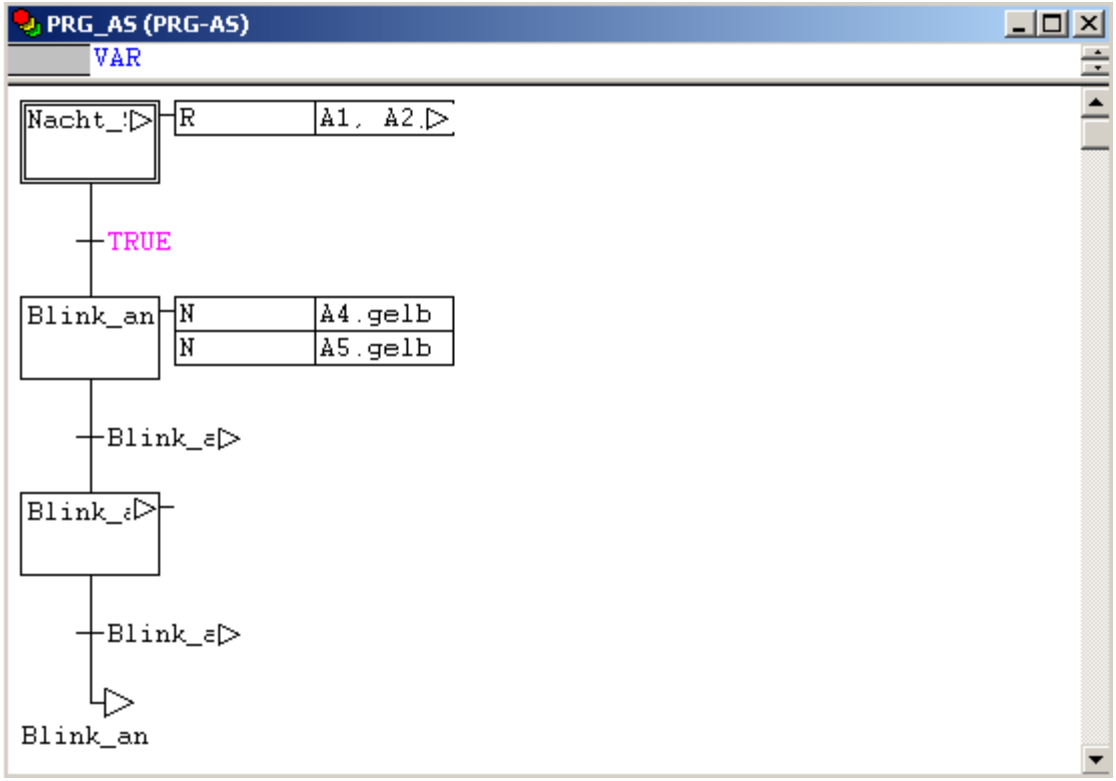
Status Bar

The IEC 61131-3 Programming languages

- Sequential function chart (SFC)
- Graphical languages
  - Ladder diagram (LD)
  - Function block diagram (FBD)
  - Continuous function chart (CFC)
- Textual languages
  - Instruction list (IL)
  - Structured Text (ST)
- Common elements of all languages
  - Program organisation
  - Program Organisation Unit (POU)
  - Data types
  - Standard – Operators, - Functions

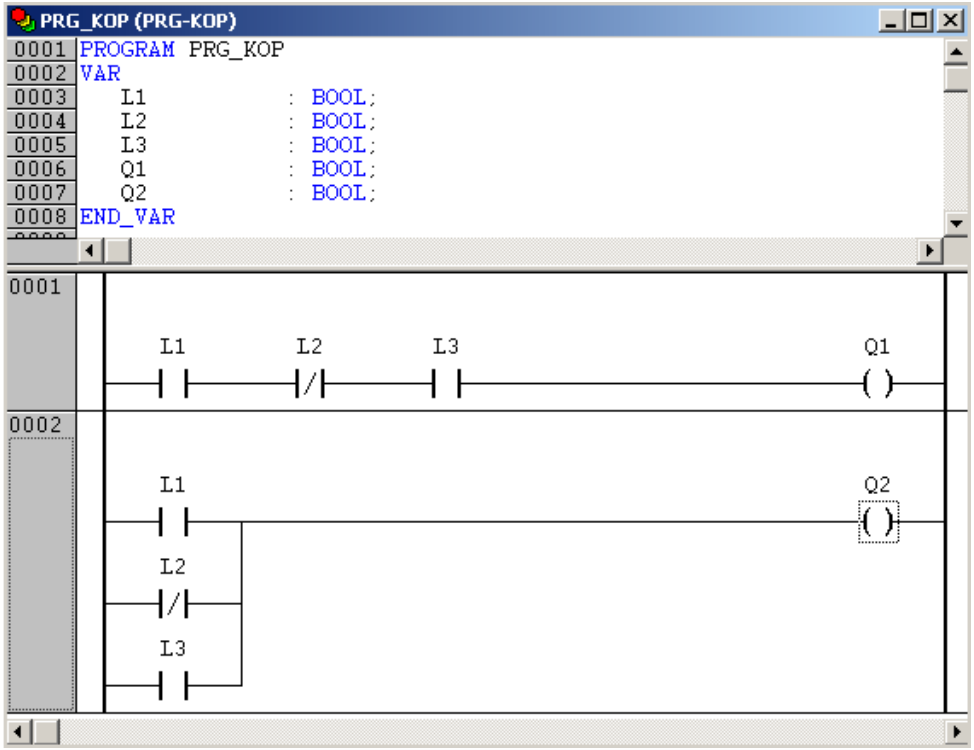
Programming languages
Sequential Function Chart (SFC)

- Graphical language
- Consists of steps and transitions
- The step contains the program
- SFC can not be replaced by any other language



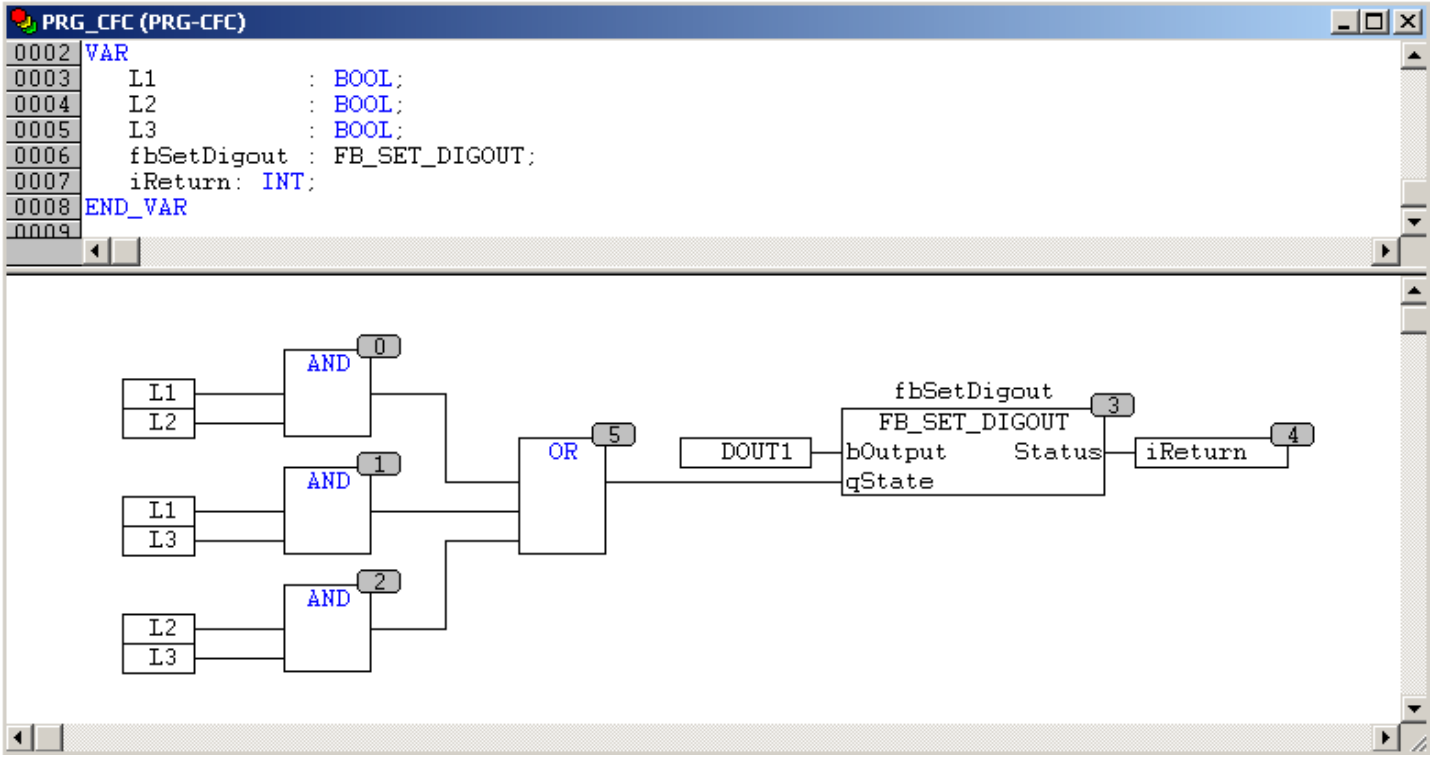
Programming language
Ladder Diagram (LD)

- Graphical oriented
- Network oriented
- Available for almost all classical PLC´s
- Hard to program with analogue values



Programming languages
Continuous Function Chart (CFC)

- Graphical language
- Consists of blocks and operands
- Free placement of connections, blocks, etc.
- Easy to understand



Programming language Instruction List (IL)

- Textual language
- Assembler like
- Best known language in Europe
- All operations work on a special register (accumulator, LD, ST)
- Easy to understand when the program is small

Programming language Structured Text (ST)

- Textual language
- Pascal oriented
- High-level language
- Recommendation by STW

F_CONVERT_DATA_TO_SDO (FUN-ST)

```
0001 FUNCTION F_CONVERT_DATA_TO_SDO : tSDOFrame
0002 VAR_INPUT
0003    pbData    : POINTER TO ARRAY [0..7] OF BYTE;
0004 END_VAR
0005 VAR
0006 END_VAR
```

```
0001 (* convert array [0..7] of byte to SDO frame format *)
0002 F_CONVERT_DATA_TO_SDO.bSCS        := pbData^[0];
0003 F_CONVERT_DATA_TO_SDO.wIndex      := BYTE_TO_WORD(pbData^[1]) + SHL(BYTE_TO_WORD(pbData^[2]), 8);
0004 F_CONVERT_DATA_TO_SDO.bSubIndex   := pbData^[3];
0005 F_CONVERT_DATA_TO_SDO.dwData      := BYTE_TO_DWORD(pbData^[4])          +
0006                                      SHL(BYTE_TO_DWORD(pbData^[5]), 8)  +
0007                                      SHL(BYTE_TO_DWORD(pbData^[6]), 16) +
0008                                      SHL(BYTE_TO_DWORD(pbData^[7]), 24);
```

Type of POU´s

- Function
  - A lot of inputs
  - One result value
  - No memory
- Function Block
  - Inputs and outputs
  - Has memory
  - Must be used indirect via instances
- Program
  - A global defined function block

Function

- Unable to store old values
- Local variables are initialized before each function call
- Function name is the variable of the return value >> functions must be typed!

Function Block

- All internal variables store their values
- Several inputs and outputs possible
- Function blocks are not used directly, they are always called through instances (copies)

Call of POU´s - Function

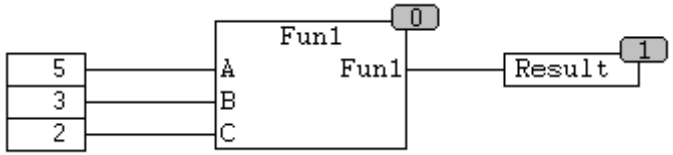
- Example:
  - Function “Fun1 : INT”
  - 3 inputs (INT): A, B, C
- Call at ST:

```
Result := Fun1(5, 3, 2);
```

- Call at IL:

```
LD    5
Fun1   3,2
ST    Result
```

- Call at CFC:



Call of POU´s – Function Block

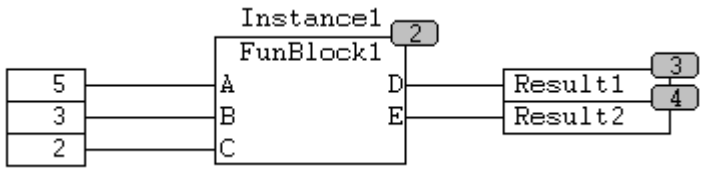
- Example:
  - Function block “FunBlock1”
  - 3 inputs (INT): A, B, C
  - 2 outputs (INT): D, E
  - Instance: “Instance1”
- Call at ST:

```
Instance1(A:=5, B:=3, C:=2);
Result1 := Instance1.D;
Result2 := Instance1.E;
```

- Call at IL:

```
CAL   Instance1(A:=5, B:=3, C:=2)
LD    Instance1.D
ST    Result1
LD    instance1.E
ST    result2
```

- Call at CFC:



Call of POU´s - Program

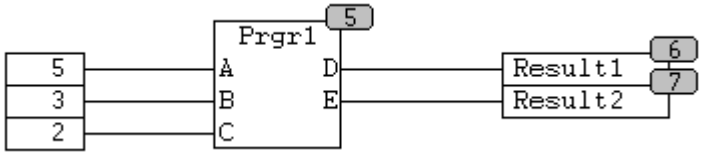
- Example:
  - Program “Prgr1”
  - 3 inputs (INT): A, B, C
  - 2 outputs (INT): D, E
- Call at ST:

```
Prgr1(A:=5, B:=3, C:=2);
Result1 := Prgr1.D;
Result2 := Prgr1.E;
```

- Call at IL:

```
CAL   Prgr1(A:=5, B:=3, C:=2)
LD    Prgr1.D
ST    Result1
LD    Prgr1.E
ST    Result2
```

- Call at CFC:



Predefined POU´s (Libraries)

- A library consist of objects, which can be used in different projects
- The user can easily generate and use his own libraries with CoDeSys
- The manufacturer can deliver encrypted libraries
- Examples:
  - besx23xx.lib >> ESX BIOS library
  - Standard.lib >> Standard library
  - StwUtilities.lib >> STW Utilities library
  - KEFEX.lib >> ESX-KEFEX library

Standard Data Types

- BOOL
- SINT, INT, DINT
- USINT, UINT, UDINT
- BYTE, WORD, DWORD
- STRING
- REAL, LREAL
- TIME, TOD, DATE, DATE_AND_TIME (DT)
- STRUCT

CoDeSys Operators

- Assignment
- Boolean operators
- Analogue operators
- Comparison operators
- Selectors
- Type conversion
- Real operators
- Shift operations
- Special operators

Assignment

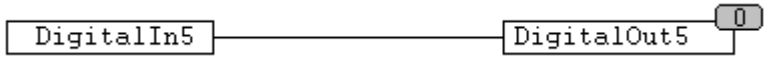
- Instruction List (IL)

```
LD    DigitalIn1
ST    DigitalOut1
```

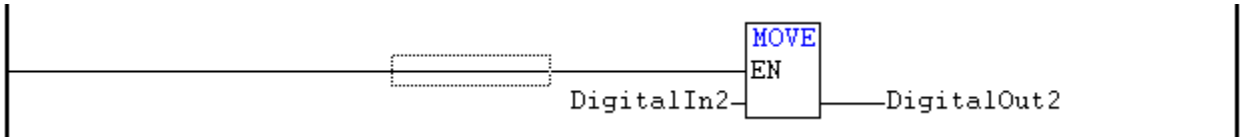
- Structured Text (ST)

```
DigitalOut4 := DigitalIn4;
```

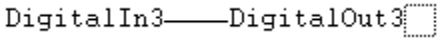
- Continuous Function Chart (CFC)



- Ladder Diagram (LD)



- Function Block Diagram (FBD)



Boolean Operators

- Operators
  - AND
  - OR
  - XOR
  - NOT
- AND, OR and XOR are extensible (Number of inputs may be extended)

Analogue operators

- IL, FBD, CFC, LD, ST
  - ADD
  - SUB
  - MUL
  - DIV
  - MOD
- ST
  - +
  - -
  - *
  - /
  - MOD
- Works with ANY_NUM

Comparison operators

- IL, FBD, CFC, LD, ST
  - EQ
  - NE
  - GE
  - GT
  - LE
  - LT
- ST
  - =
  - <>
  - >=
  - >
  - <=
  - <
- Works with ANY_NUM

Selectors

- MIN returns the lesser of two values
- MAX returns the greater of two values
- SEL binary selection
- MUX Multiplexer
- LIMIT Limit of a value

Type conversion

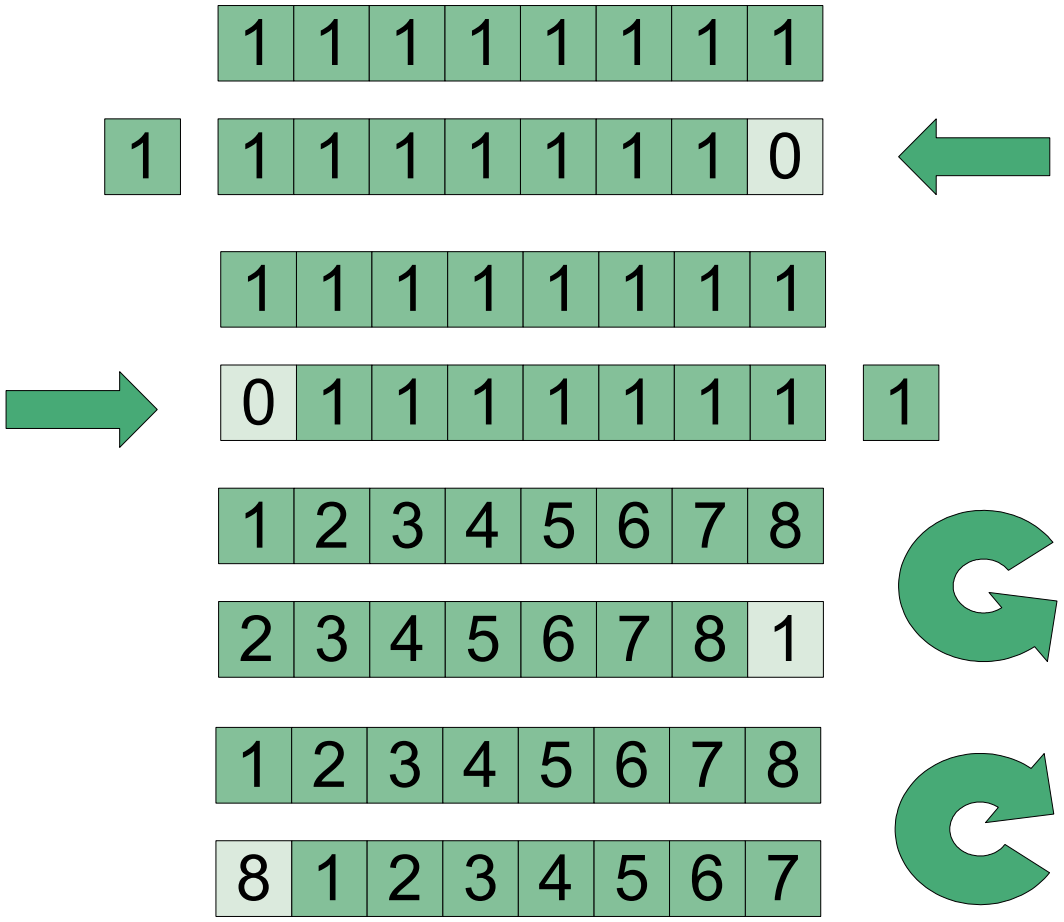
- In general: Function “<type>_TO_<type>”
- The number of conversions has been kept as low as possible
- If possible, conversions should be implicitly
- Example
  - INT_TO_BYTE
  - WORD_TO_BYTE
  - DWORD_TO_BYTE
  - UDINT_TO_SINT

Real operations

- ABS returns the absolute value of a number
- SQRT returns the square root of a number
- LN returns the natural logarithm of a number
- LOG returns the logarithm of a number in base 10
- EXP (e^x) returns the exponential function
- Trigonometric function (SIN, COS, TAN, ASIN, ACOS, ATAN)
- EXPT (y^x) exponentiation of a variable with another variable

Shift operations

- SHL (shift left)
- SHR (shift right)
- ROL (rotate left)
- ROR (rotate right)



Special operations

- ADR returns the address
- SIZEOF returns the size
- Content operator “^” for pointers

Additional language constructs

- Array (maximum 3D)
- Structure
- Enumeration
- Alias
- Pointer

Data and addresses

- Validity
  local (within a POU) or global (for all POU´s)
- Style of declarations
  text, table and automatic variable declaration
- Kind of addresses
  inputs, outputs and markers

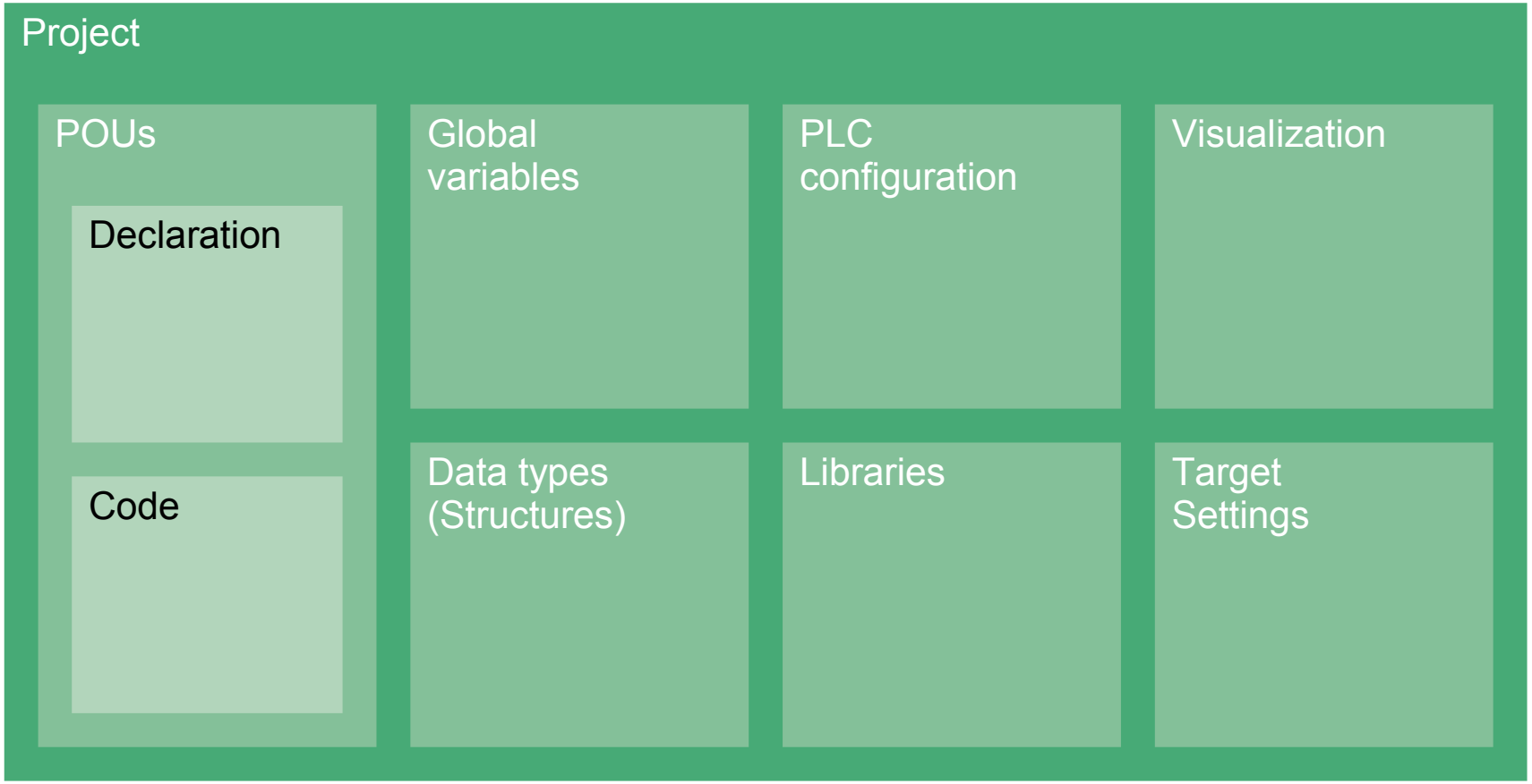
Address syntax

- Every address starts with the percentage sign “%”
- Area prefix
  - I Input
  - Q Output
  - M Marker
- Size
  - X single bit
  - B byte (8 bits)
  - W word (16 bits)
  - D dword (32 bits)
- Examples
  - %IW1
  - %QX2.0
  - %MD16

Identifier syntax

- Letters and numbers
- Must start with a letter
- Only single underscores
- No spaces
- No IEC keywords/operands
- Capital letters are significant
- Example:
  - Otto, OTTO, otto, oTTO, OTto, otTO
  - Valve1
  - A_very_long_identifier

Structure of a CoDeSys application



CoDeSys Project Properties

- Saved in one file (file extension .pro)
- Consists of program organisation units (POU´s)
- Starts with the POU named PLC_PRG
- Is cyclically executed

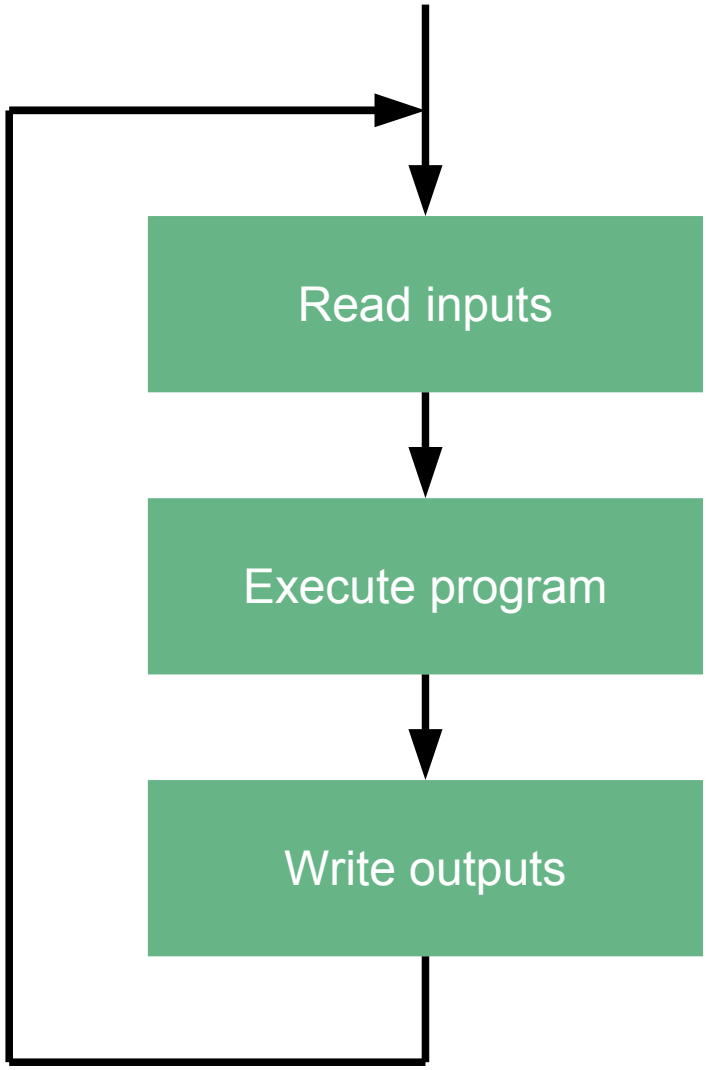
Adaptions/Extensions by STW

- Run-time system
- PLC configuration
- ECU BIOS libraries
- “Utility” libraries
- Target Support Packages
- Communication protocol
- Installation routine (InstallShield Setup)
- Wizard for CAN interface installation and automatic configuration

Supported STW ECU´s

- ESX®
- ESX®-Light
- ESX®-micro (special variations which are supported in CoDeSys!)
- ESX® expansion boards (aka babyboards)
- ESX®-DIOS/DIOM over COL2/YDIOS or 3S CANopen stack

The way a PLC works



Function from the run-time system

- Read inputs
- Execute the application
- write/set outputs
- initialize/configure the ECU in depend on the PLC configuration
- Interface to the corresponding ECU BIOS functions
- Communication handler
- Debug handler
- Run-time system is ECU specific

CoDeSys Tools

- Library Manager
- PLC Configuration
- Gateway Server
- Sampling Trace
- Watch- and Recipe Manager
- Visualisation

Library Manager

- Shows all libraries at the actual project
- A library POU can be handled just the same way as own defined POU´s
- Libraries can be added to a project or removed from a project
- Libraries, which are part of the distribution, can be found in “<CoDeSysInstallDir>\Library”
- The right ECU BIOS library will be inserted in dependence of the selected target support package
- Unused POU´s of a library has the text colour grey and the used one´s are black

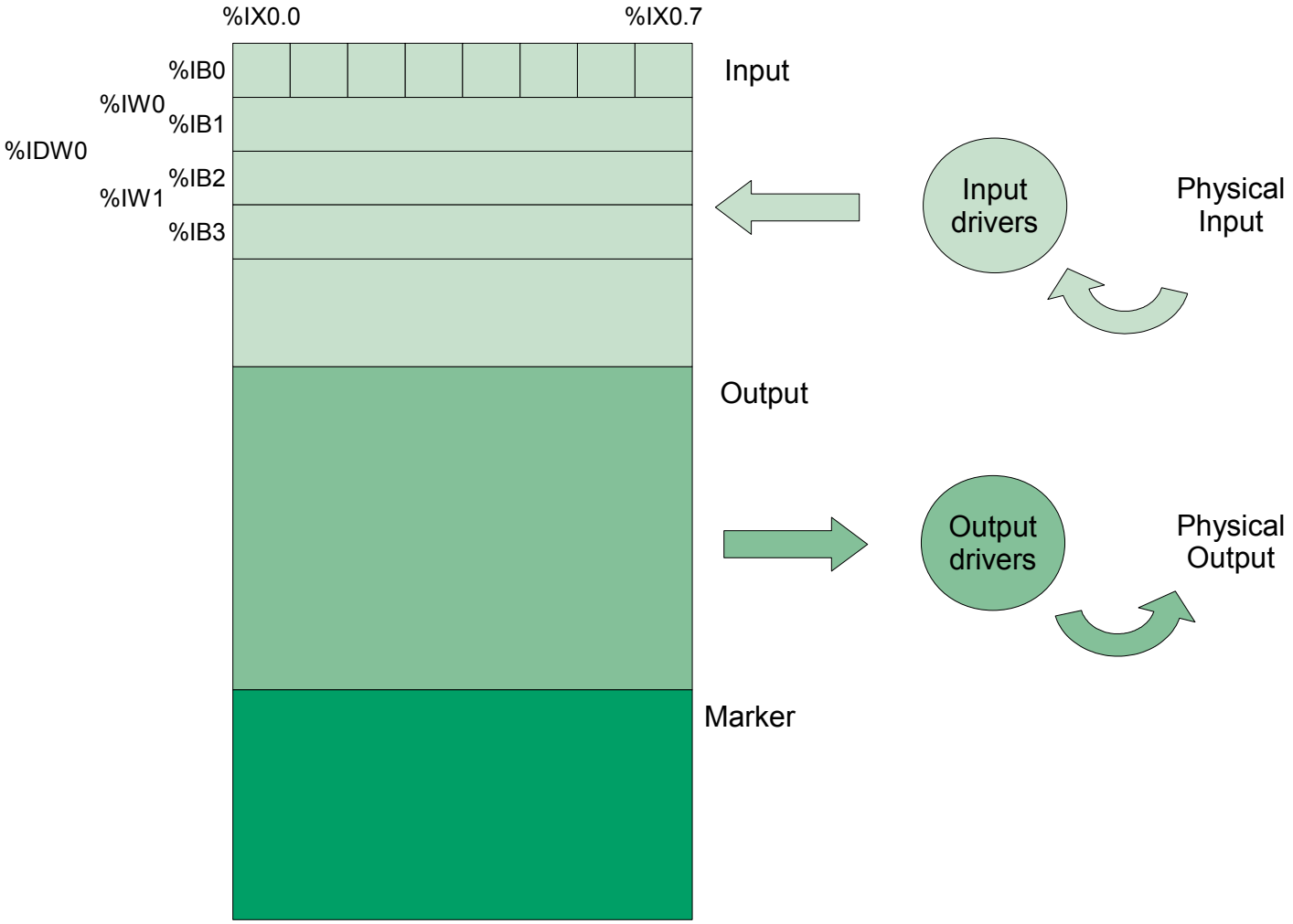
Available Libraries (Selection)

- ECU dependent libraries
  - ESX® BIOS (besx23xx.lib)
  - ESX®-Light BIOS (besl23xx.lib)
  - ESX®-micro BIOS (besu23xx.lib)
  - ESX®-DIOS/DIOM (YDIOS.lib)
- General libraries
  - CANopen L2 (col2.lib)
  - Useful stuff (StwUtilities.lib)
  - Standard library (appropriated by STW)
  - KEFEX library (kefex.lib)

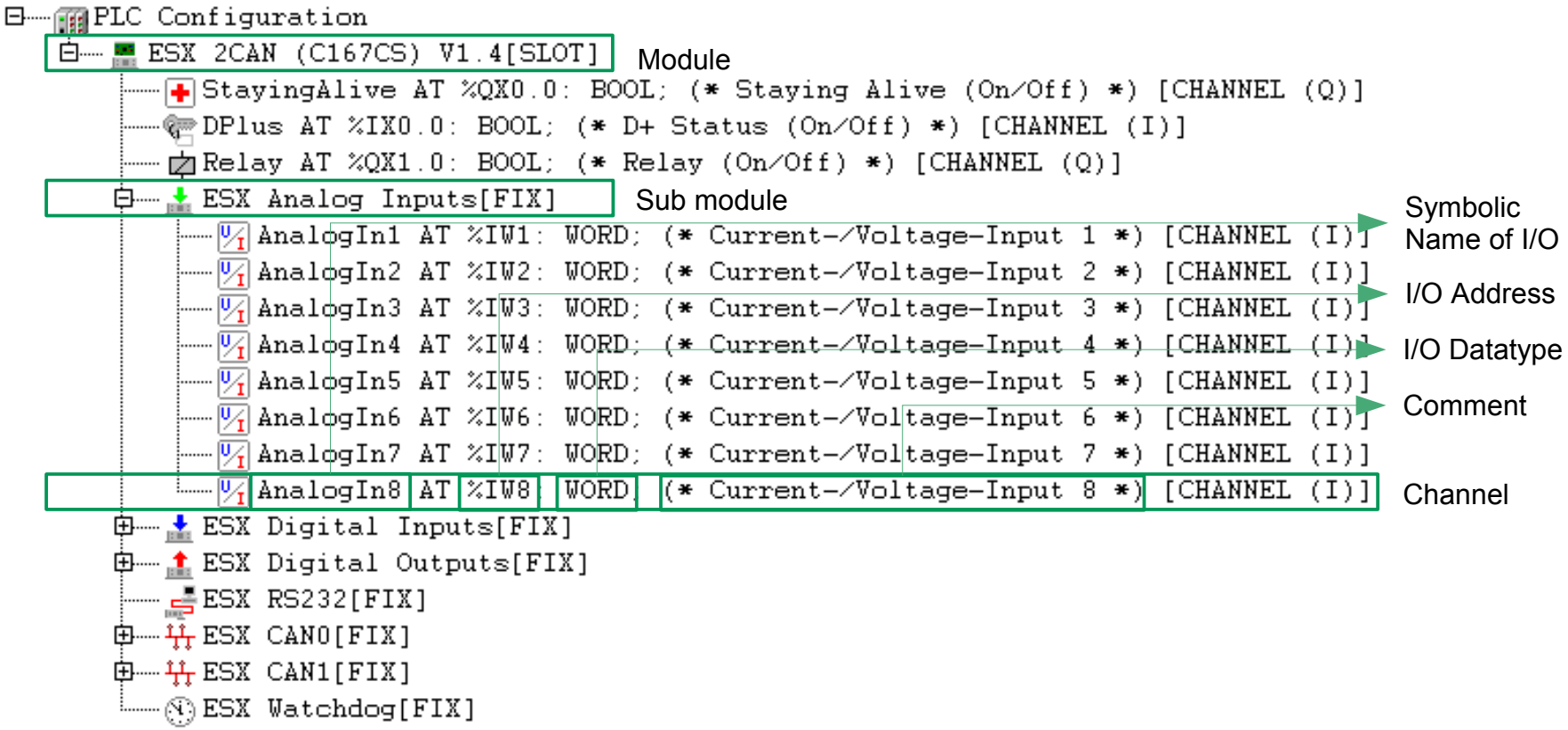
PLC configuration

- Configuration from ECU functions
- Lay down the symbolic name for in-/output
- Address definition in input/output process image
- ECU is described by a module
- Functional groups are sum up in sub modules
- Channels describe each individual input/output
- Modules and channels can be configured about parameters

PLC configuration – process image



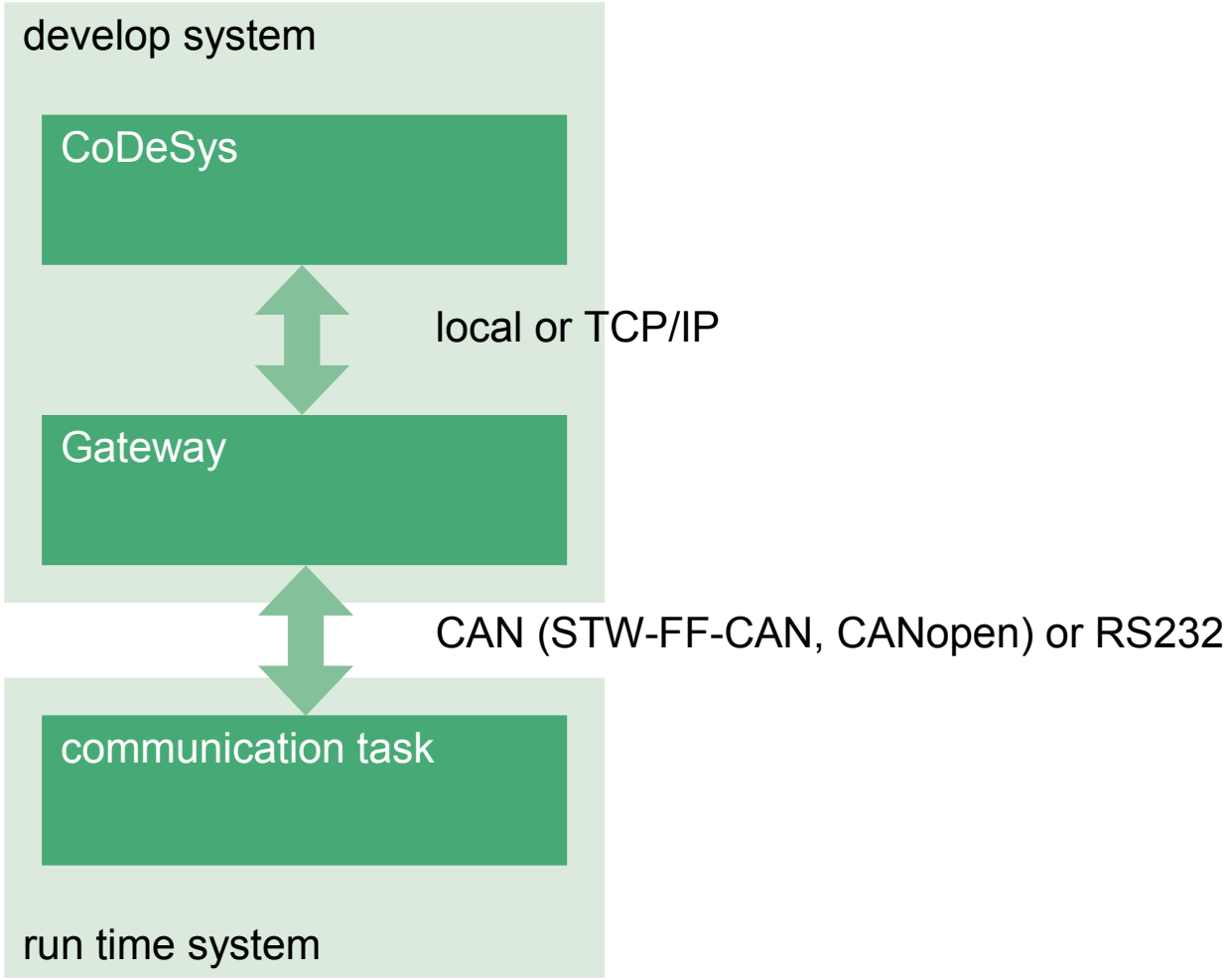
PLC configuration - example



Gateway Server

- Gateway between CoDeSys IDE and run-time system
- Can be either used “local” or via “TCP/IP”
- Channels describe the way of connection
- Adjustments were saved for each project
- Supported protocols
  - STW-FF-CAN (block orientated CAN protocol)
  - CANopen DSP302 (SDO CAN protocol)
  - RS232 (serial protocol)

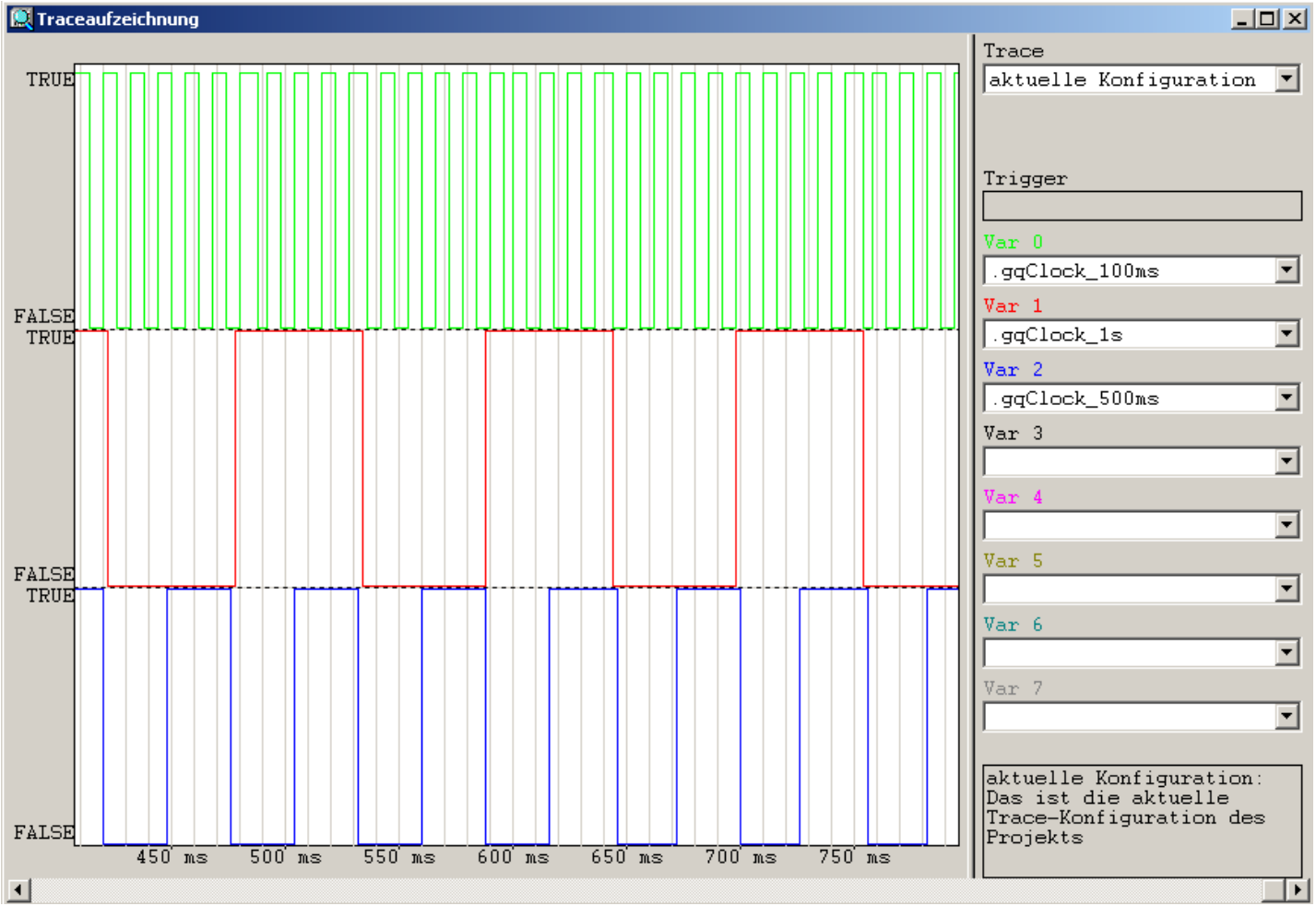
Gateway server



Sampling Trace

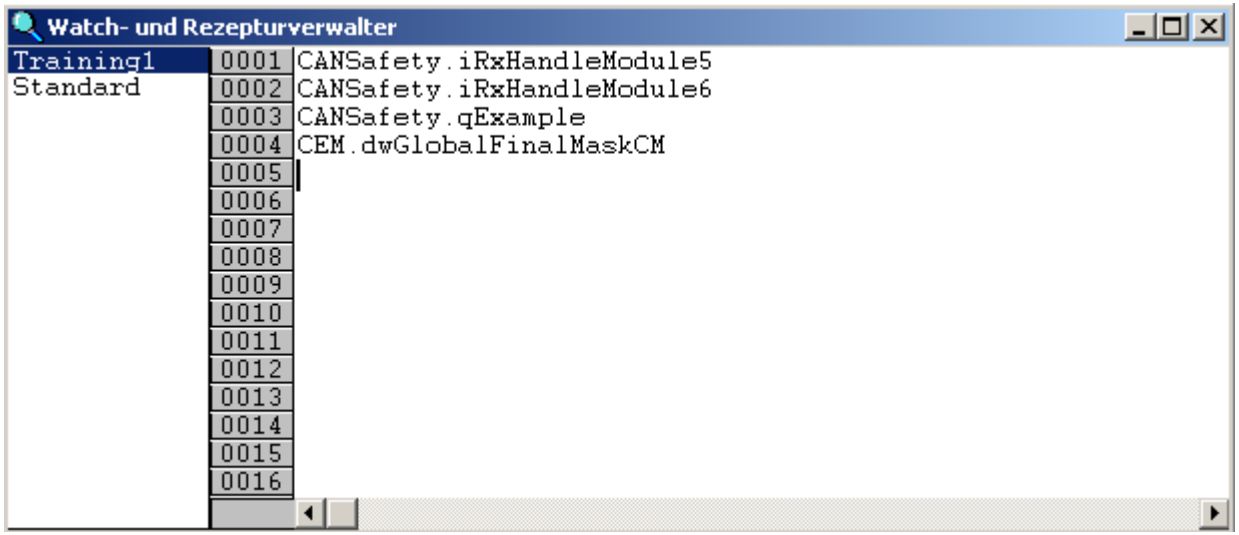
- Recording of values from variables in a defined period
- Maximum 20 variables can be recorded at the same time
- Per variable 500 values can be recorded (depend on the data type of the variable)
- Size of the trace buffer is limited at run-time system onto 4990 byte
- Trace configuration can be saved or loaded (.tcf)
- Trace recordings can be saved or loaded (project format .trc or XML-format .xml)
- Management of further trace configurations is possible

Sampling Trace - Example



Watch- and Recipe Manager

- Displaying of selected variables with the values
- Modification of variables values
- Management of more than one watch list is possible
- Internal and/or external storing is possible
- Example:



Visualisation

- Visualisation editor is a part of CoDeSys IDE
- Watch or serve application data
- Force variables
- If is possible to display the visualization without CoDeSys IDE (CoDeSysHMI)
- Configuration is done via special dialogues
- Visualization objects can be used multiple (place holder concept)
- Possible to control the PLC exclusively from the visualisation

Visualisation - Example

Additional informations

- STW CoDeSys V2.3 User Guide
- CoDeSys built-in help system
- Homepage of 3S: www.3s-software.com

Questions?

Thank you for attention!