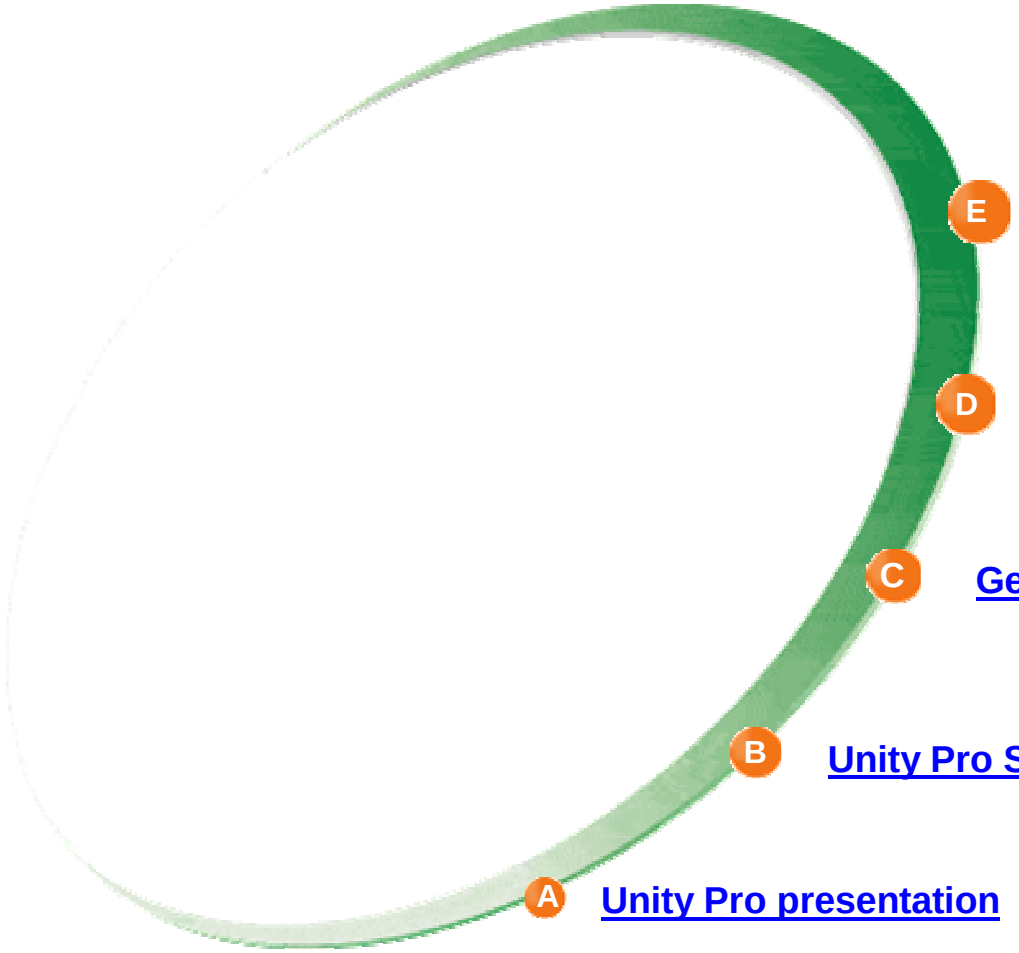
Basic Micro : Operating mode with Unity Pro S



[Operating modes](#)

[General advice to develop your application](#)

[Unity Pro Small](#)

[Unity Pro presentation](#)

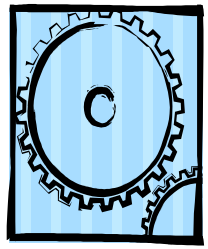
Unity Pro presentation

Unity Pro overview

- Unity Pro is a all in one software
  - For Quantum, Premium, Atrium and Modicon M340 platforms
- Scalability of the software is based on the supported platforms and not on features
  - Unity Pro Small supports only Modicon M340 PLCs
- Unity Pro for all the life cycle of your project

Design

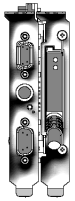
Debug



Operation

Maintenance

Selection guide



Unity Pro S	M	L	XL
Modicon M340	57-00, 57-10, 57-20	57-30, 57-40	57-50
	57-20	57-30	
		31110, 43412, 53414	65150, 65160, 67160

Offer structure

- Single and multi-seats

Up to 3 ……10 ……..100 users

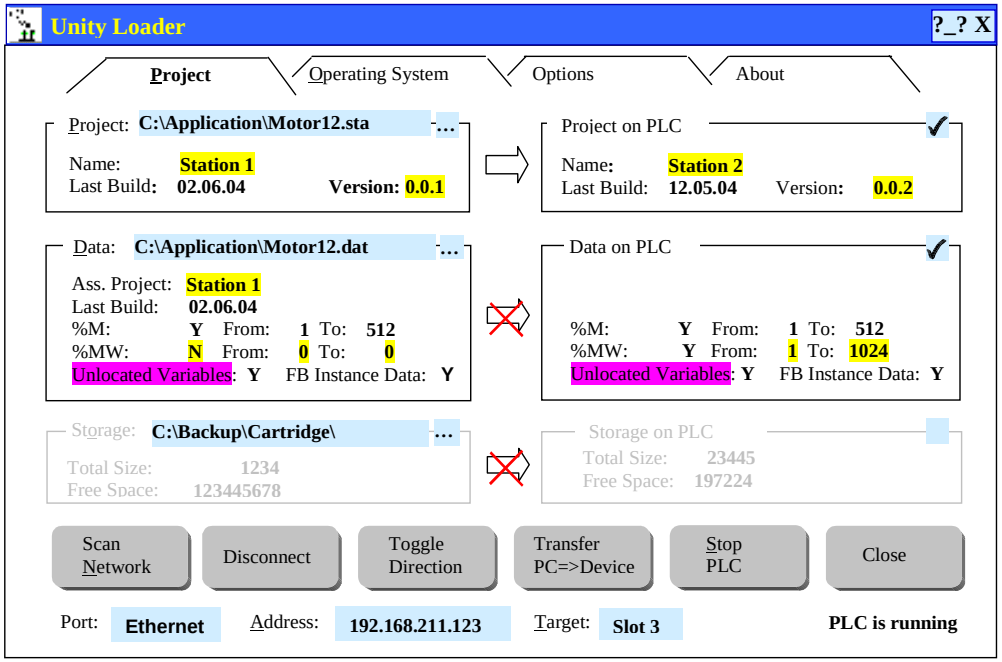
	Single	Group	Team	Facility
Unity Pro X Large	✓	✓	✓	✓
Unity Pro Large	✓	✓	✓	✓
Unity Pro Medium	✓	✓	✓	✓
Unity Pro Small	✓	✓	✓	

- Upgrades
  - Ease to make transition path from legacy software (PL7, Concept, ProWorx) for customers with active subscription only
- Education and Schneider Alliance special offer

Unity Loader tool

- A simple and easy-to-use tool to perform basic maintenance
  - Dedicated for operators that are not familiar with Unity-Pro
  - Stand alone tool that doesn't need Unity Pro on the computer
- Available only for Modicon M340 PLCs
- Simple and fast transfer
  - Upgrading a PLC can be done in one shot (except for firmware)
  - Transfer through USB port and Ethernet
  - Run / Stop PLC services

Unity Loader tool (cont)



- Basic operations are available through buttons : upload / download of :
  - Project
  - File of data
  - User files (data storage on memory card)
  - User Web pages (NOE module)
  - PLC and Ethernet module firmware

Unity Pro Small

Unity Pro Small overview

- The right tool for all phases of your mid range automation project
  - Selection and installation : « all in one » software for programming and debugging a complete application
  - Design : structured variables and 5 languages to develop your application program
  - Debug : PLC simulator and high level of debug services to test and debug your program
  - Operation : operator screens and diagnostic viewer to monitor and control your process
  - Maintenance : on-line modifications to update your application program
  - Openness : use of XML format

Design your program

Design

- Unity Pro Small is a full featured software package
- Mid range platforms benefit of the richness of Unity Pro features
  - 5 IEC languages
  - Standardization capabilities
    - Through data : unlocated variables, structures and arrays
    - Through program : sections of program, user function blocks, functional modules
  - Multitasking capabilities
    - Mast and Fast tasks (no Aux task)
    - Event triggered treatments
  - Functional view to map your application to the process
- With the possibility to bring more value through the openness
  - Interface to any client application through XML format

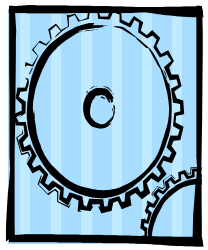
Debugging your application

Debug

- Embedded PLC simulator
  - No hardware constraints to debug your program
  - Provide the same execution capabilities as a PLC
  - Possibility to use function blocks of IO management library to simulate (write) %I, %IW, %ID and %IF inputs *(WRITE_INPUT_EBOOL, WRITE_INPUT_INT, …)*

- An unequalled set of debug services
  - Power flow animation for graphical languages
  - Breakpoint and step by step to test and debug the program
  - Watch point to know the real time value of a variable
  - SFC monitoring of step activity times
  - Color used during execution progress (boolean variables, steps, …)

Operate and maintain your application



Operation

Maintenance

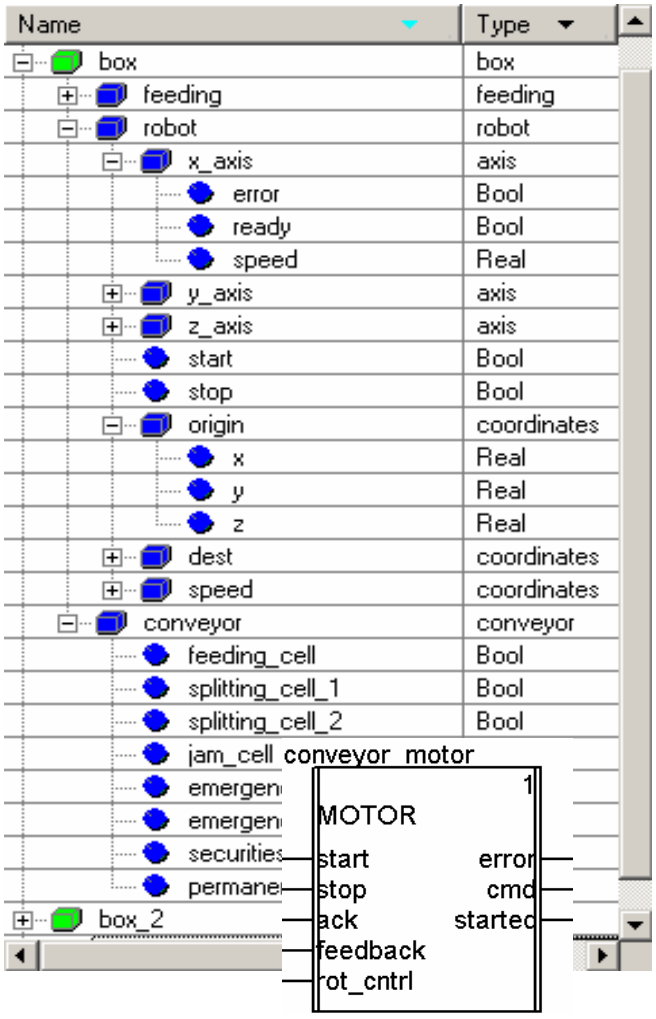
- Visualize and control the application with operator screens
  - Graphical and animated view of your process
- PLC and module diagnostic through configuration editor
- Integrated system and application diagnostic accessible through the built-in diagnostic viewer
  - No programming required
  - Display faults saved in the PLC with source time stamping
  - Fault cause analysis to find origin of process fault
  - Possibility to acknowledge
- Upward compatibility of the PLC applications
  - Last version of Unity Pro supports the installed base design with the previous versions

General advice to develop your application

Don’t reinvent the wheel each time

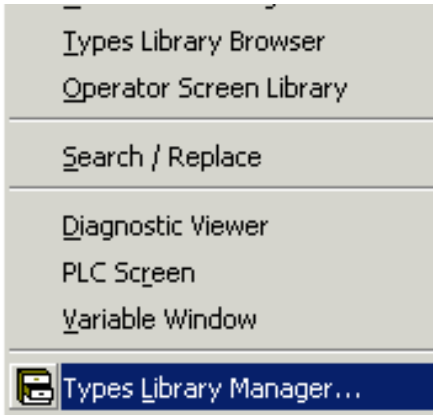
- Design and combined your standards to develop your application in short time
- Manage and share your standards
- Describe the process through functional approach
- Debug and adjust your application directly on your PC to reduce commissioning time
- Use embedded diagnostic to reduce downtime
- Manage all process data in the PLC

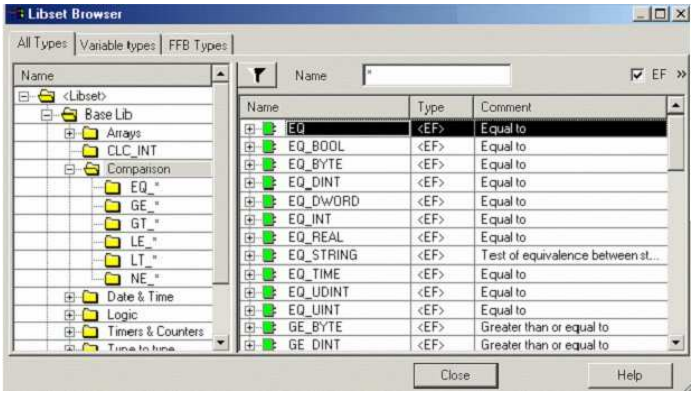
Combine your standards



- Arrange PLC data according to the process
  - Data structures and multi-dimensional arrays
  - Unlocated variables to create standard databases without memory mapping
  - Reusability simple through standards in libraries and import / export features
- Move to a component approach in design with Derived function blocks (DFB)
  - Create standards of your often used logic
  - Combine these standards to design more complex logic (nested DFB)
  - Protect your know-how

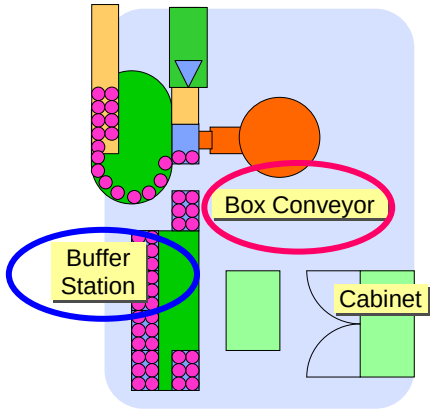
Share your standards

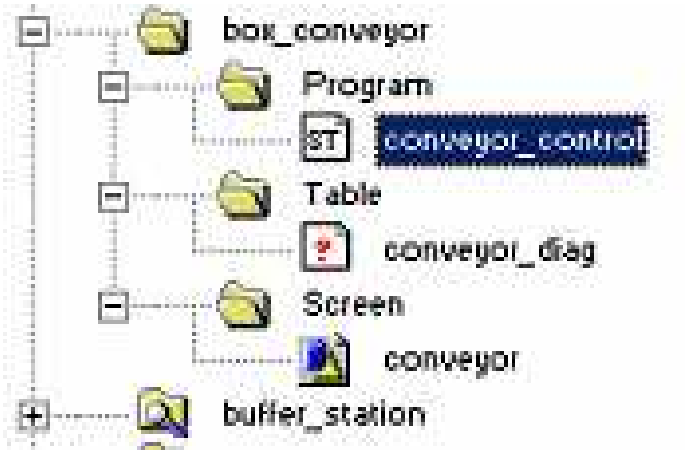


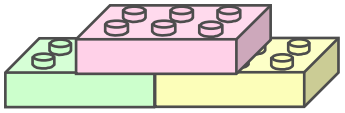


- Organize your standards in user libraries and families
  - Library manager is the repository of your standards
- Share your standards to other developers
  - Standards are accessible to all programmers
  - Be sure that your application contains the right version
    - Comparison of project to the library
    - Versioning to trace modifications
- Make modifications only to the model (type)
  - All instances are automatically updated

Think process and not PLC

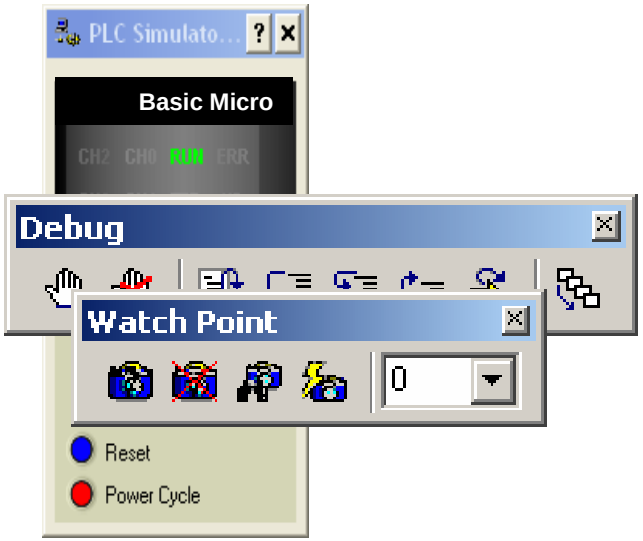


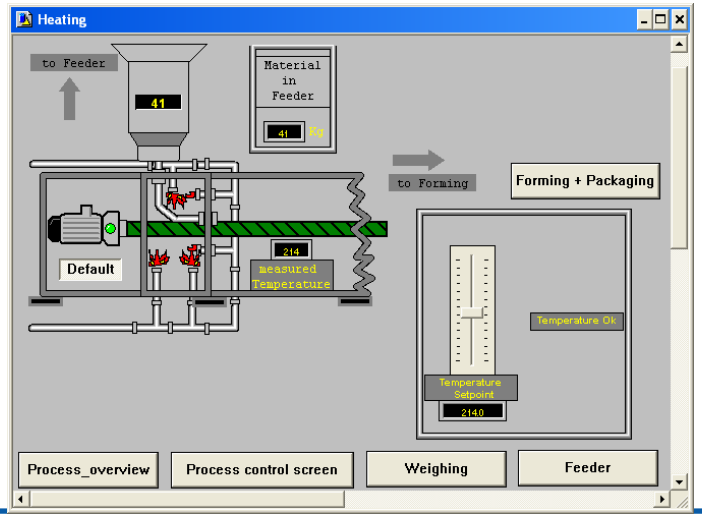




- Break down your application into functional modules close to the process
- Use modular functional modules to describe a complex machine
  - Tree description with nested modules
  - Easy to find the right information through the functional description of your application
- Reduce design time to create custom machines by combination of existing functional modules
  - Create standardized functional modules
  - Export / import modules
  - Wizard to manage variables when reusing a module

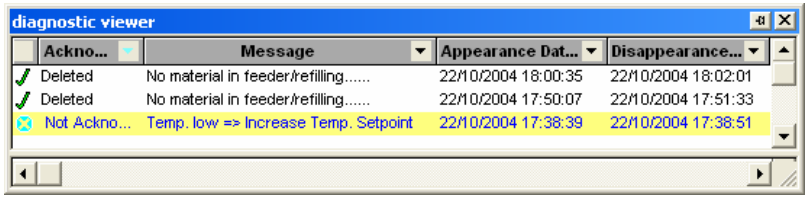
Debug and adjust the application in your PC





- Debug your program without hardware
  - Simulate the PLC on your PC
  - All debug tools are available
- Use embedded services to reduce commissioning time
  - Graphical operator screens to display the behavior of the machine or process
  - Quick access to additional documents about application through hyperlinks (documentation, wiring diagrams, …)

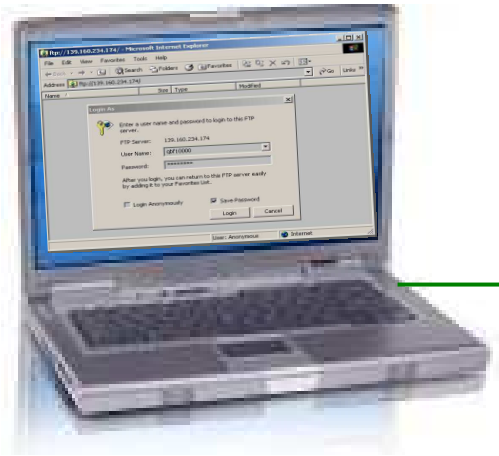
Reduce downtime and save costs during operation



- Use embedded diagnostic without programming
  - Hardware and program diagnostic are easy through clear messages
  - Choice of language for system messages
  - Time stamping and navigation to the root cause
- Access to advanced diagnostic through function blocks
  - To monitor a movement or any execution of the process
  - Fault cause analysis
- Diagnostic through Web pages accessible via Ethernet port

Store data in the PLC

Standard FTP client



Ethernet

FTP server

- Some memory cards can save data files accessible from the PLC application through function blocks
  - Create / delete files
  - Get / modify file attributes
  - Read / write data
- Use Ethernet / FTP to upload / download files
  - Memory card is managed as FTP server
  - Use Unity Loader or a standard tool *(Internet Explorer FTP Client)* to exchange with the memory card

Operating modes

Configuration required and installation

- PC configuration required to install Unity Pro
  - Nominal : PC 1.2 GHz / 512 MB RAM
  - Operating system (fully qualified) : Windows 2000 / Windows XP
- Same directory for all Schneider software
  - C:\program files\Schneider electric\xxx
- Installation procedure
  - Standard program install / uninstall of Windows
  - Right to use the software is associated to its registration (only 21 days free)
  - Unity Pro can coexist with PL7 or Concept and can be executed in same time
  - It's impossible to install 2 different versions of Unity Pro on the same PC

Unity Pro environment

- Multi instance mode
  - Several instances of Unity Pro can be executed in same time (same application or different applications)
  - If several instances of the same application only one instance opens the application in Read / Write mode. Other instances are in Read mode only

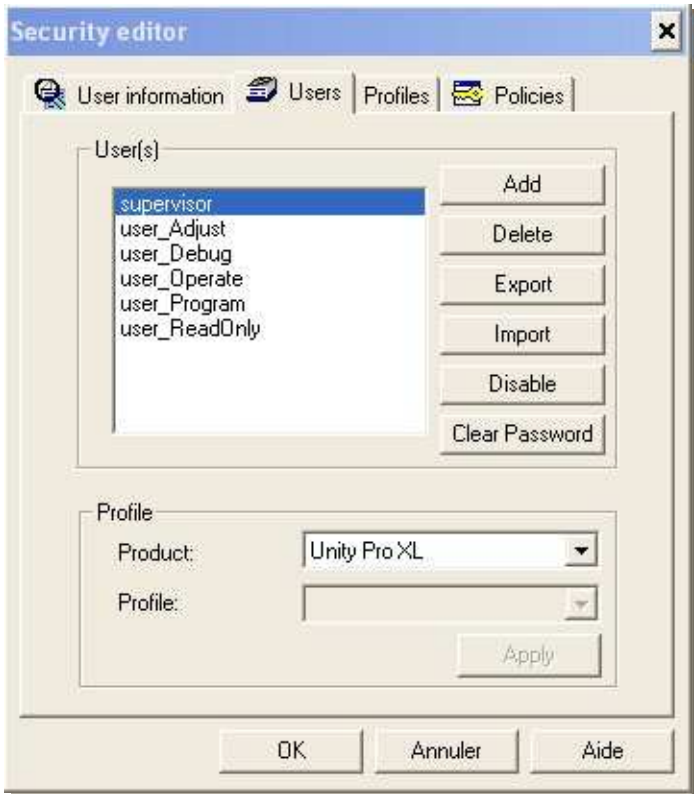
- Compatibility management
  - Ascending compatibility is guaranteed through archive (STA file) or XML format (XEF file)

- Multi language
  - Software can be installed in 6 languages with selection at install time
  - Installed languages can be changed (without reinstall) using an utility tool. Change is applied at the next launch of Unity Pro

Security editor

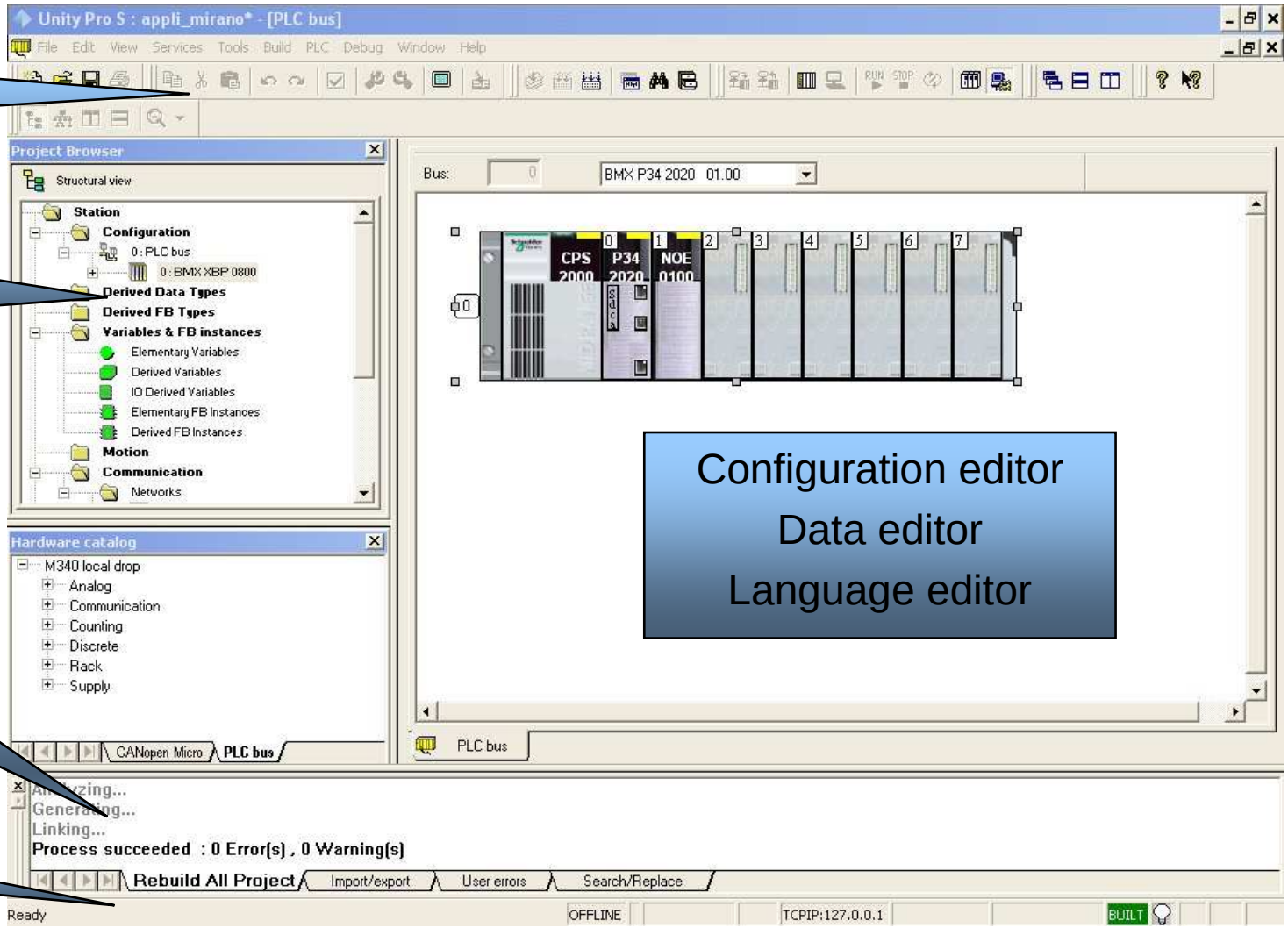
- Tool use to manage the access to Unity Pro.
- Security profiles are defined by the supervisor
- Policies tab define the security state (security off, security on with mandatory login, …) and enable / disable the auditing and confirmation
- Audit Log file managed by Windows security system save operation with audit attribute

Security editor (cont)



- Users tab defines the list of users (predefined users and users created by supervisor) and assign the profiles to the users
- Profiles define the list of rights (predefined profiles and profiles created by supervisor)
- Predefined users and profiles are not modifiable

User interface



Menus and toolbars

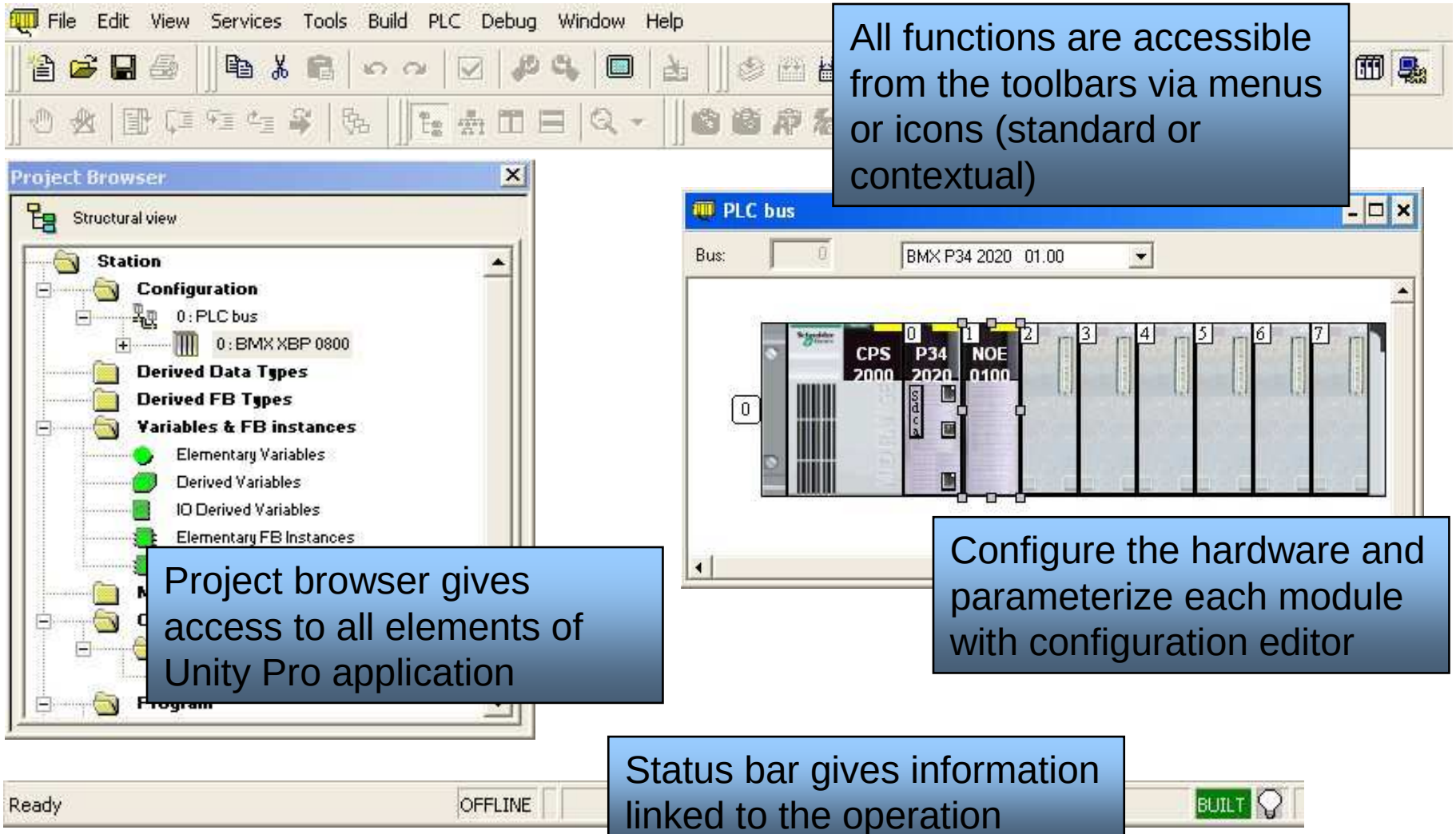
Application browser

Configuration editor
Data editor
Language editor

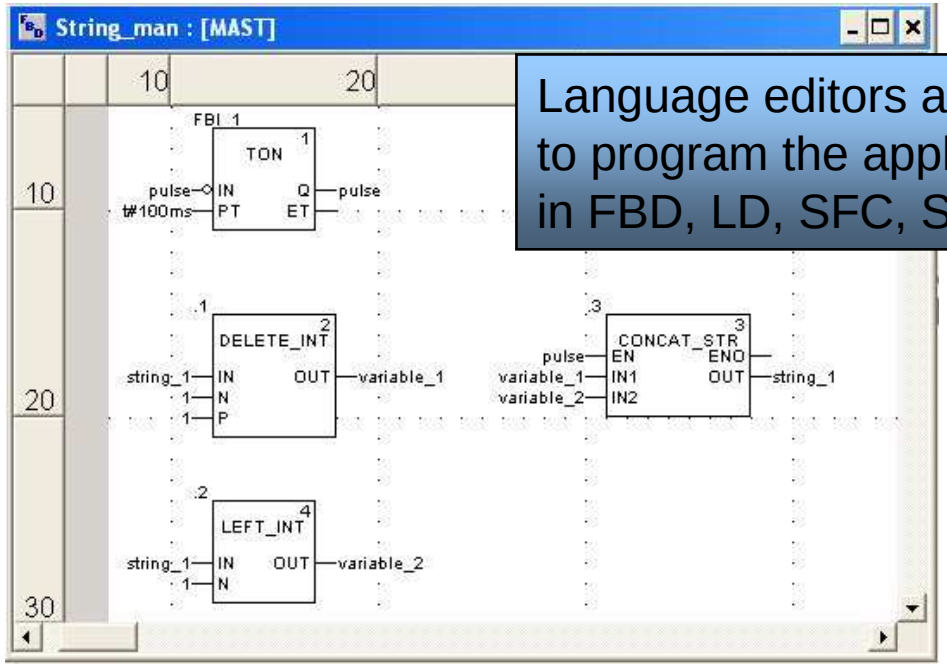
Output windows

Status bar

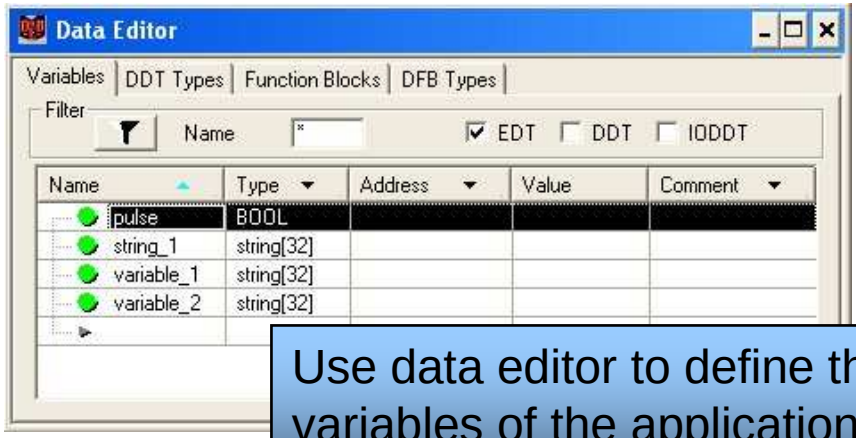
User interface (cont)



User interface (cont)



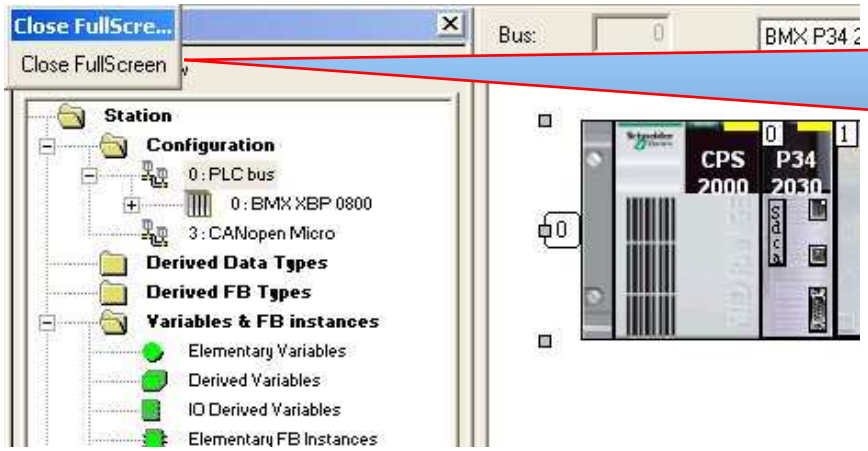
Language editors allow you to program the application in FBD, LD, SFC, ST or IL



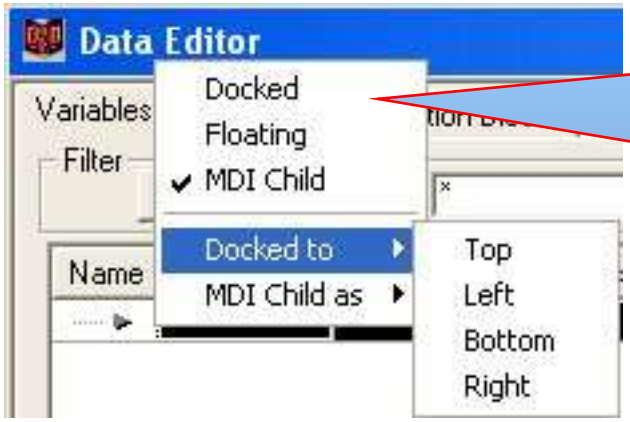
Use data editor to define the variables of the application : types and instances

Output windows give result of some functions : build, import / export, search

Ergonomics



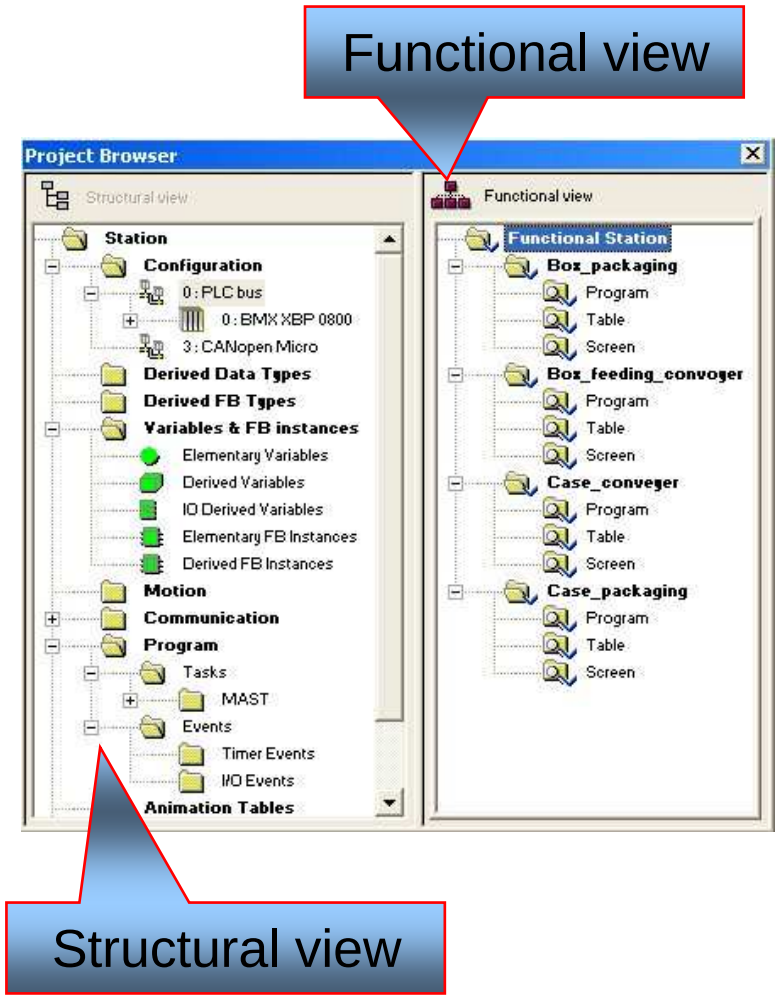
All editor can be switched from normal to full screen mode (to enlarge the space dedicated for edition)



Some windows (animation tables, data editor) can be :

- docked (to anchor in positions outside the application window)
- or floating (to be always visible in foreground).

Project browser



- Project browser proposes 2 different views to present and structure your project in tree-structured format
  - Structural view
  - Functional view
- With the structural view the user can access and manage the different elements of the application (hardware configuration, variables, program, documentation, …)
- Functional view enables the user to structure the application into functional modules

Structural view

Hardware configuration

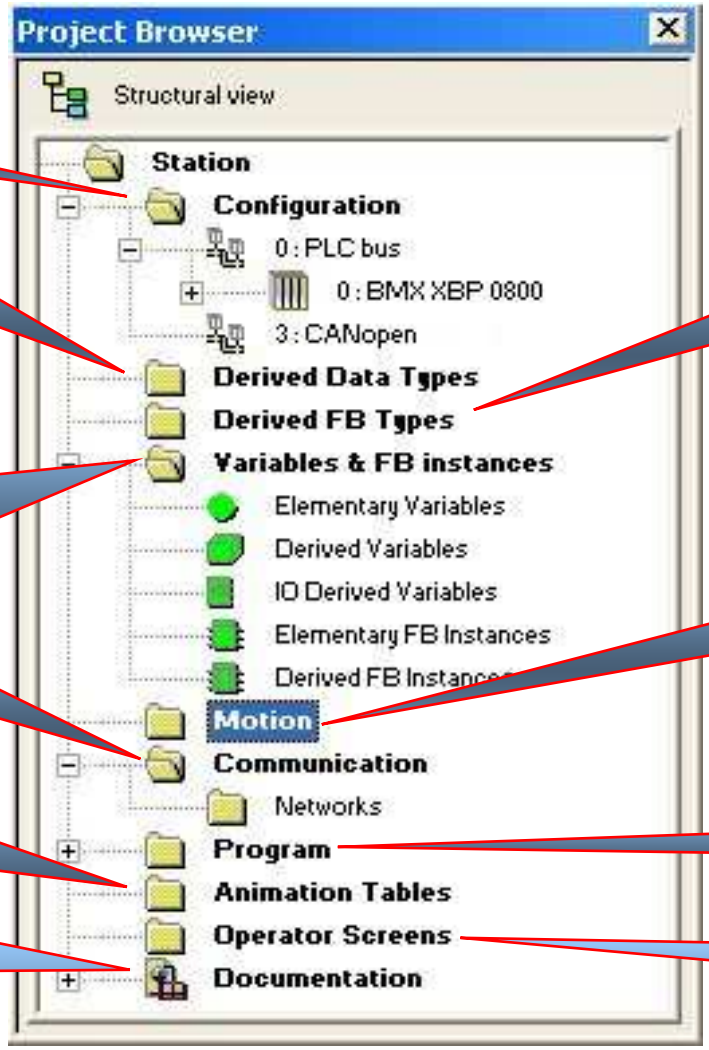
Type of structured variables

Elementary variables and instances of function blocks

Logical networks configuration

Table of user variables

Project documentation



Type of user function blocks

Logical motion axis configuration

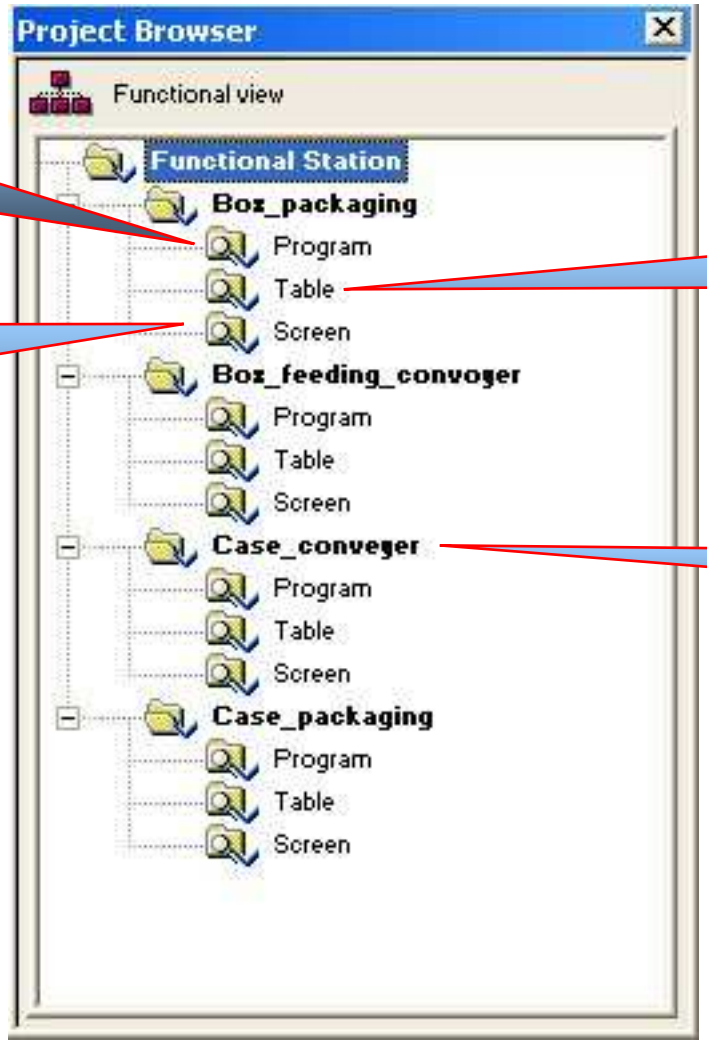
Program structure

Graphical screens

Functional view

Program sections of the functional module

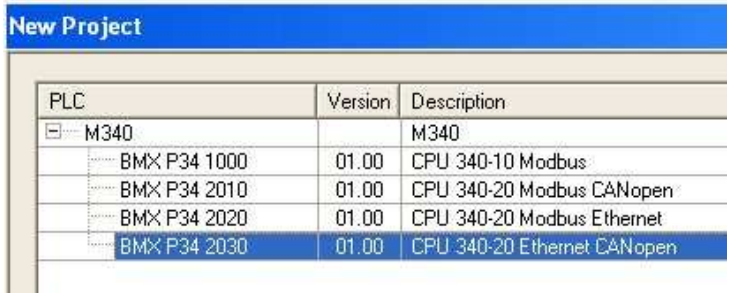
Operator screens of the functional module



User variables of the functional module

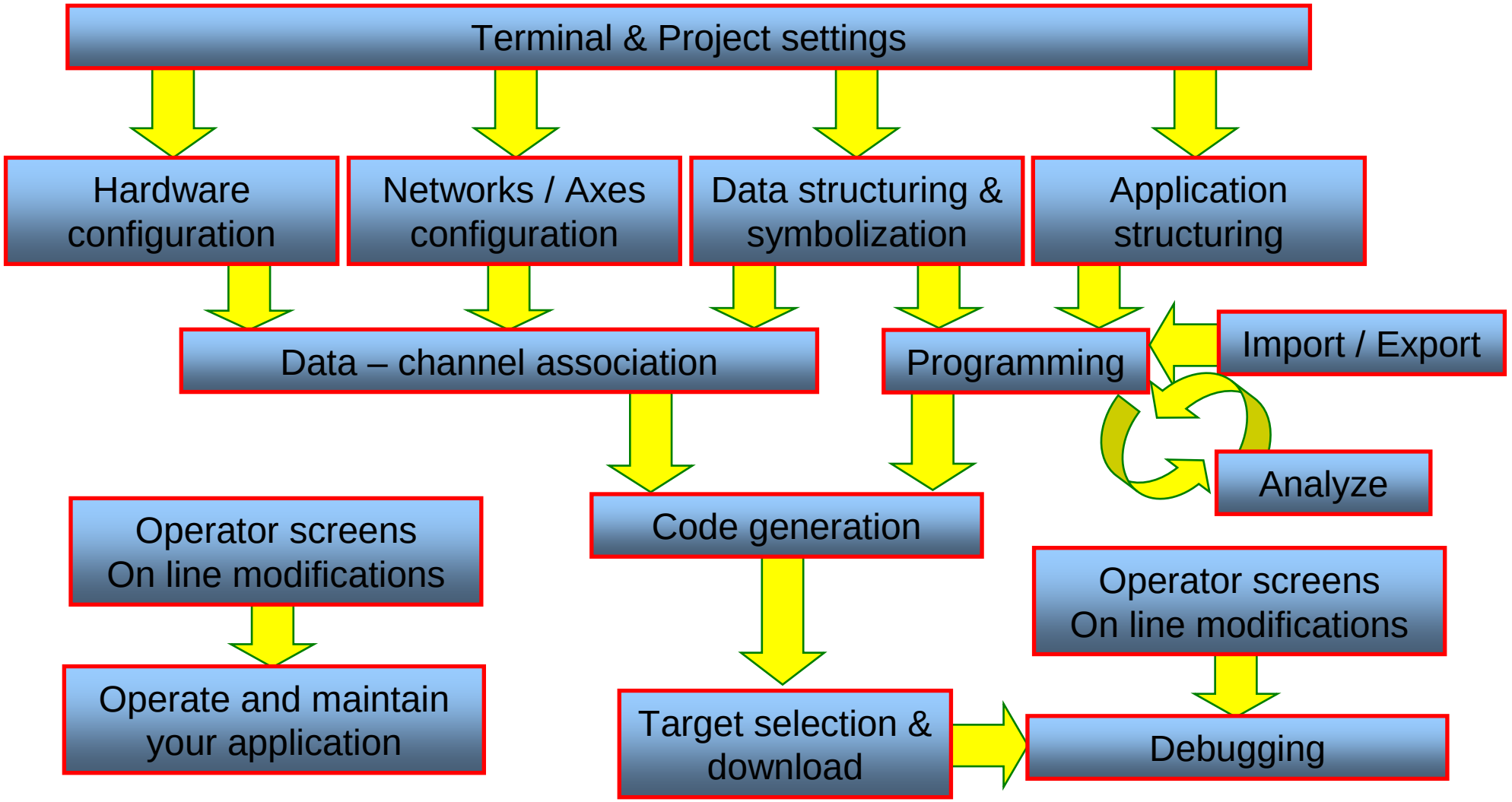
Functional module

Start a new project



- Run Unity Pro from *Program / Schneider Electric / Unity Pro*
- Open a new project
- Select the family (if necessary) and the processor (*i.e. BMX P34 2030*)

Methodology to develop a new application

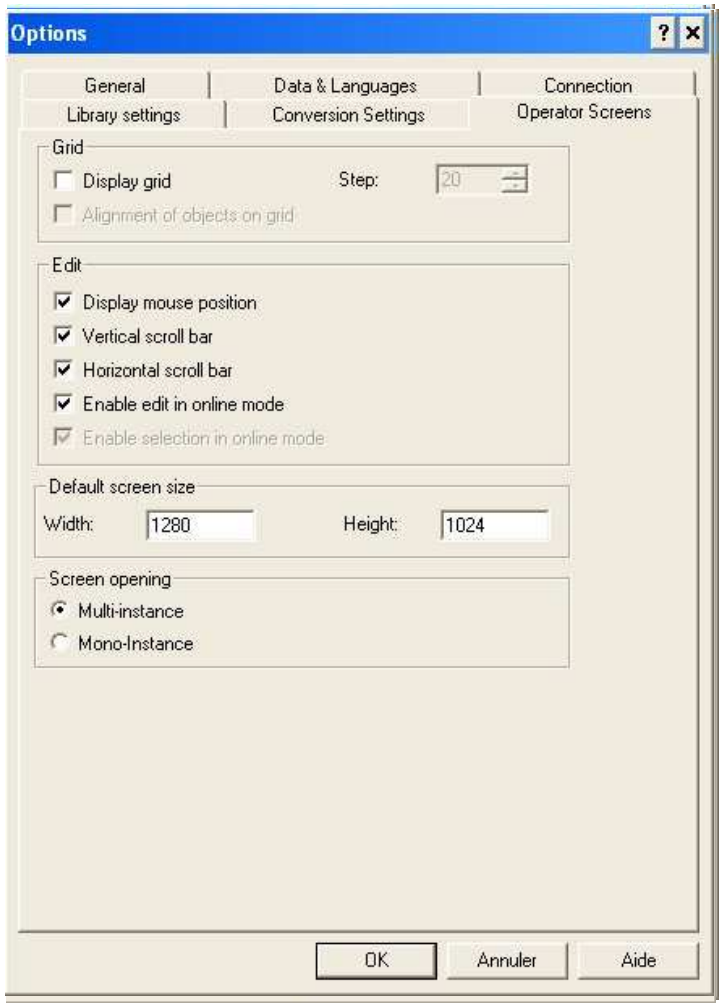


Methodology to develop an application (cont)

1. Define the settings
   - Work station settings accessible via *Tools / Options*
   - Project settings accessible via *Tools / Project settings*
2. Configure the hardware and the networks (racks, modules,…)
3. Define and edit the variables (elementary variables, arrays, structures, instances of function blocks,…)
4. Structure the application (tasks, sections, functional modules)
5. Edit the sections of program (Ladder, FBD, …)
6. Generate the code (build)
7. Choose the target (PLC or simulator) and transfer the application (download)
8. Debug the application (breakpoint, step by step, …)

Define the settings

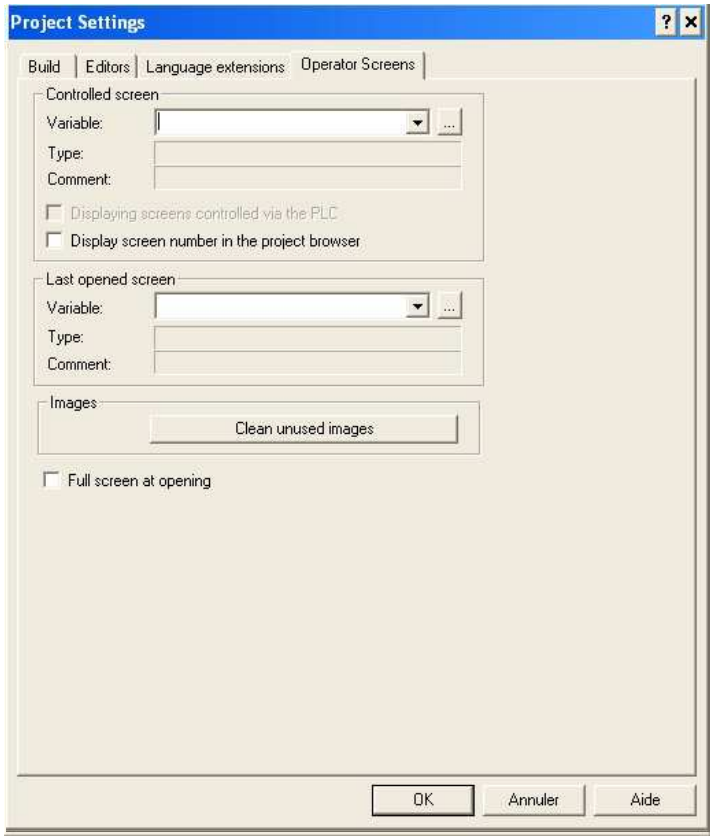
Work station options



- General to define general settings of the project and working directories
- Data & languages tab configures settings for entering data
- Connection defines options relating to connection of terminal to the PLC
- Library settings tab gives information about global library
- Conversion settings tab define settings for conversion
- Operator screens defines runtime screens

Define the settings

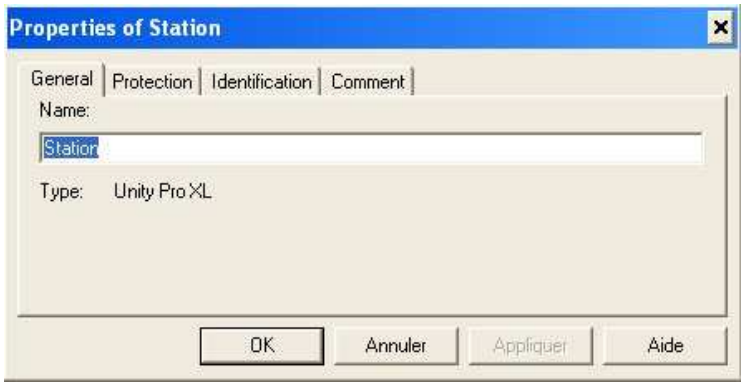
Project settings



- **These settings are save in the application**
- Build configures the generation of the project
- Editors defines the features for graphical languages (Ladder and Function Block Diagram)
- Language extensions contains settings for enabling expansions to IEC standard.
- Operator screens defines management of the screen by the operator

Define the settings

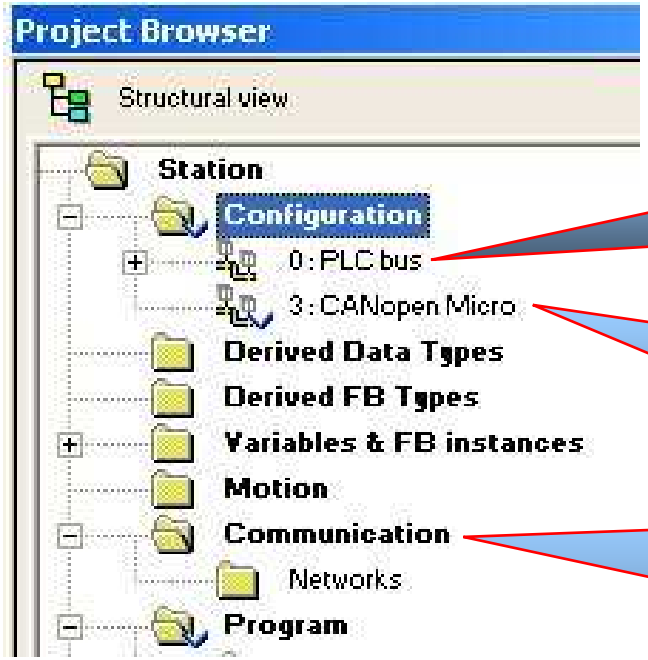
Project properties



- Accessible by **right-clicking** on station folder
- General defines the name of the project (*Station or Functional station by default*)
- Protection activates the protection of the program sections. This operation needs a password
- Identification identifies the project (current version, last rebuild all and last partial build)
- Comment associates a comment to the project

Configuration editor

- Accessible from structural view

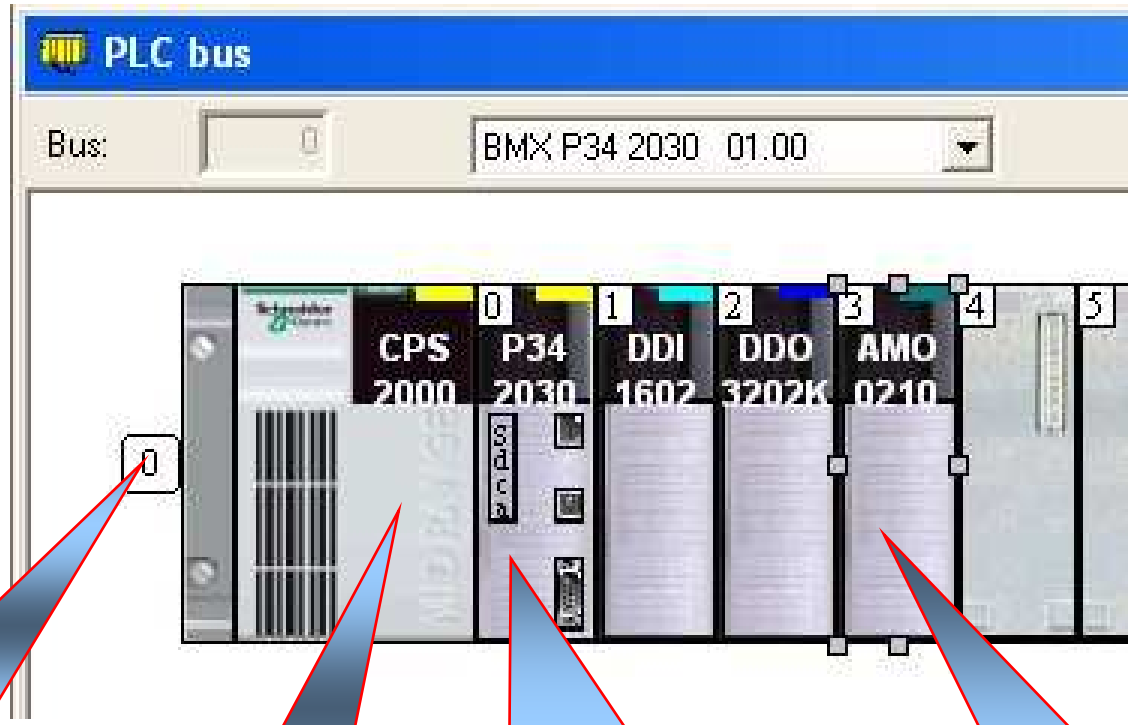


Configure the racks of the local bus

Configure the field bus : CANopen Micro

Link communication hardware to logical network Ethernet

Local bus configuration



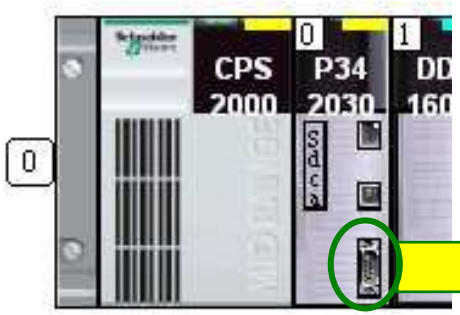
Choose the rack

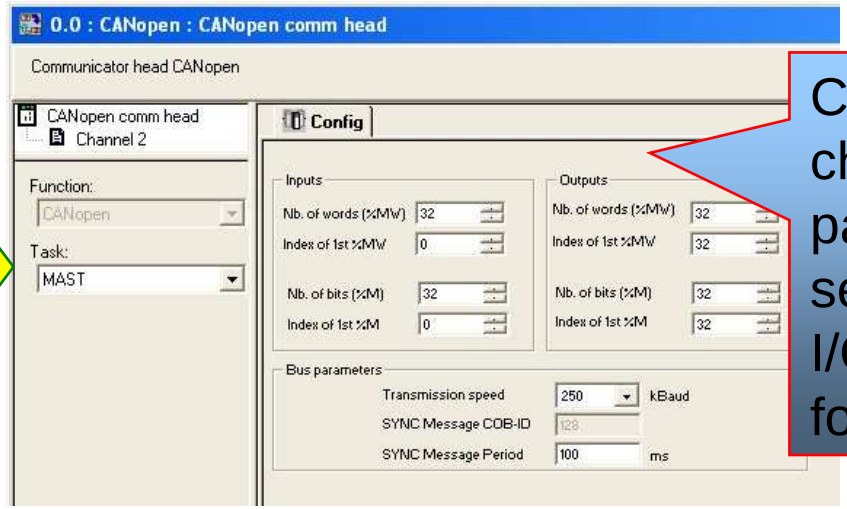
Define power supply module

Replace processor (if necessary

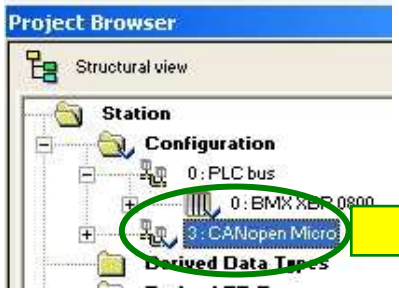
Define modules (drag and drop from catalog)

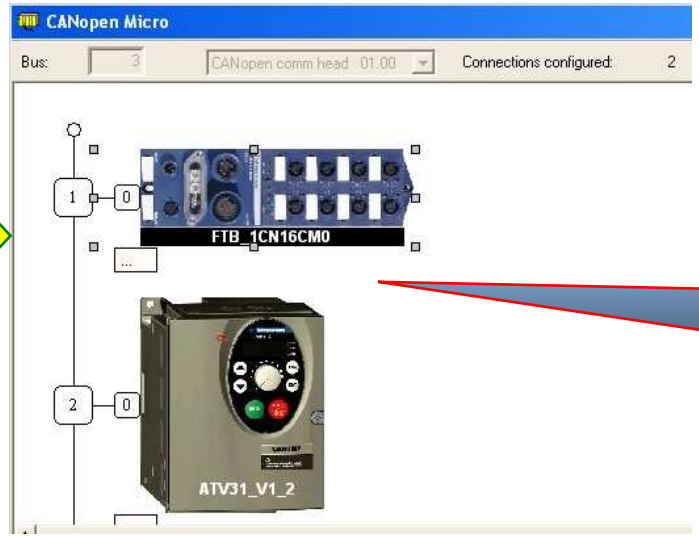
CANopen bus configuration





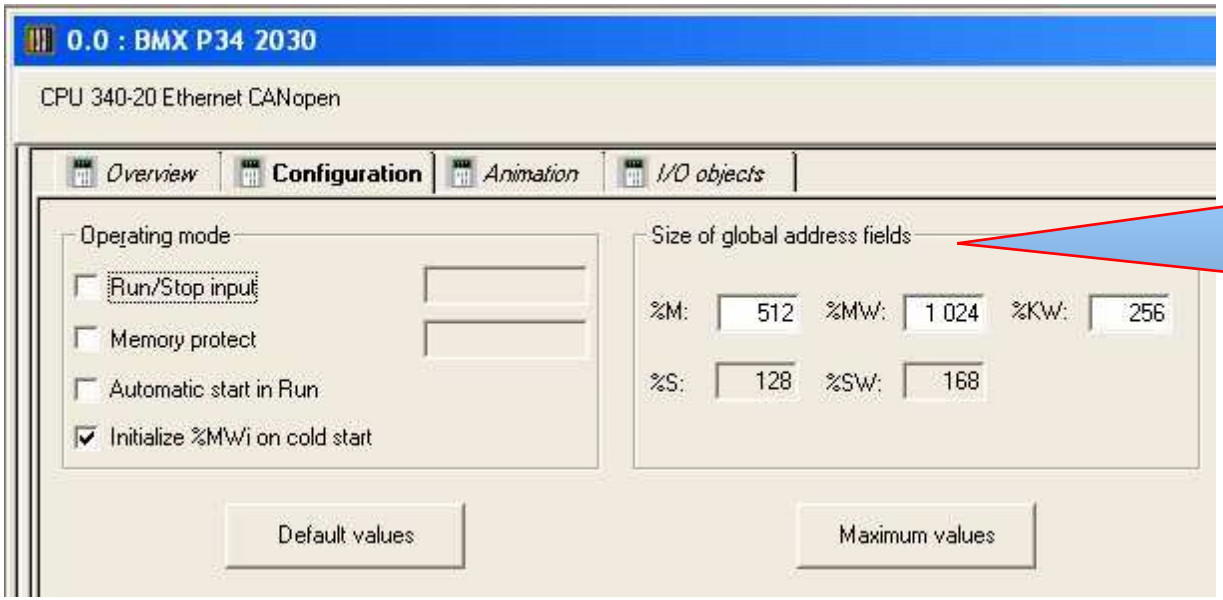
Configure CANopen channel (bus parameters, task which set rate of update for I/O, range of addresses for I/O)



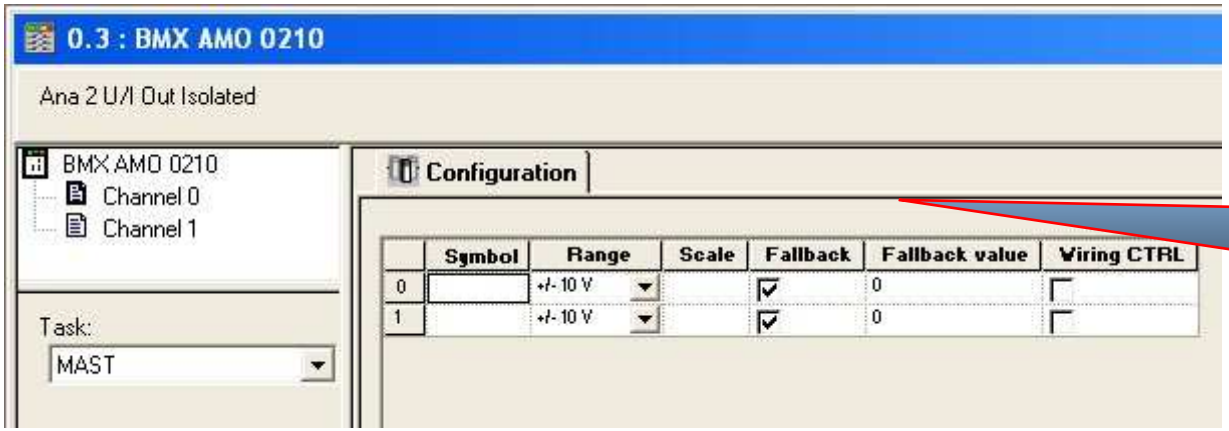


Configure CANopen fieldbus (drag and drop from catalog)

Processor and modules configuration

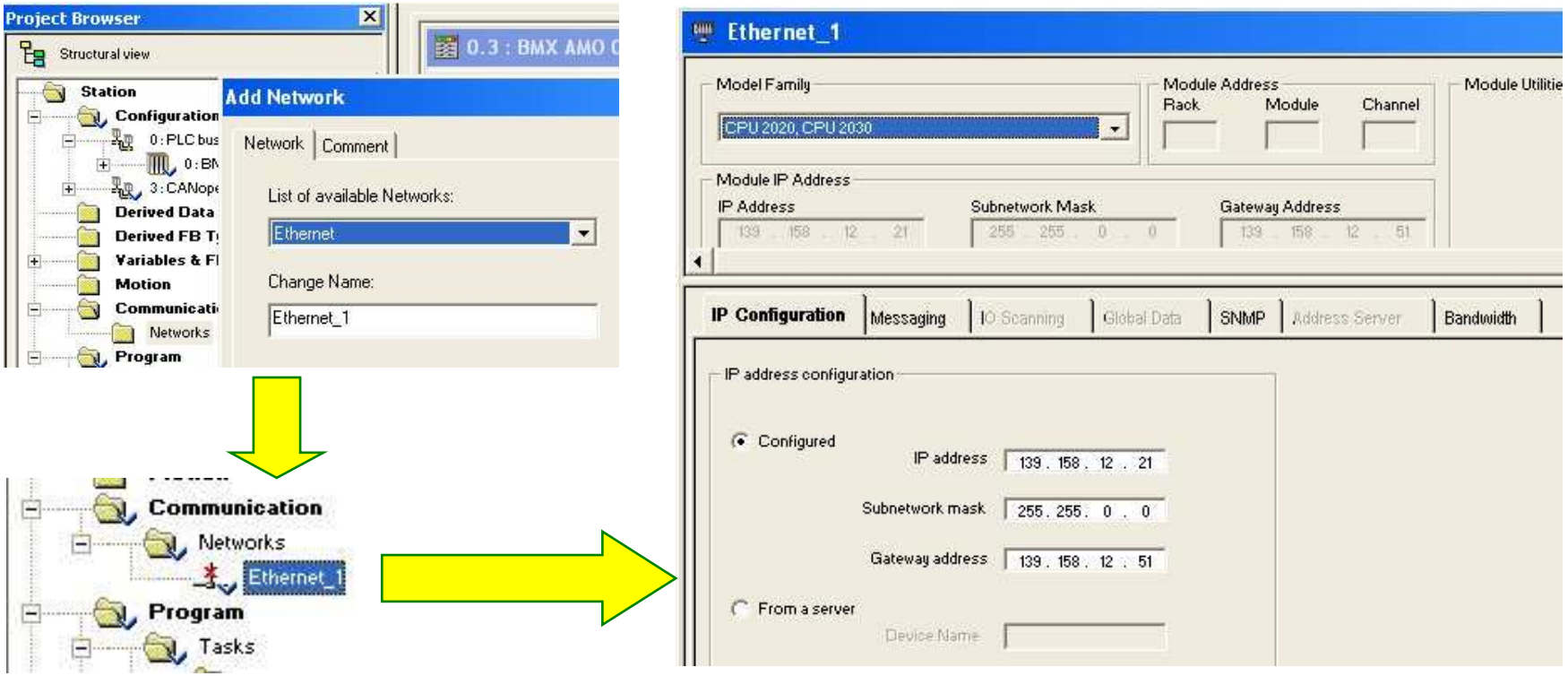


Configure processor (select operating mode, define global objects of the application : bits and words)



Configure each module (task, parameters of each channel)

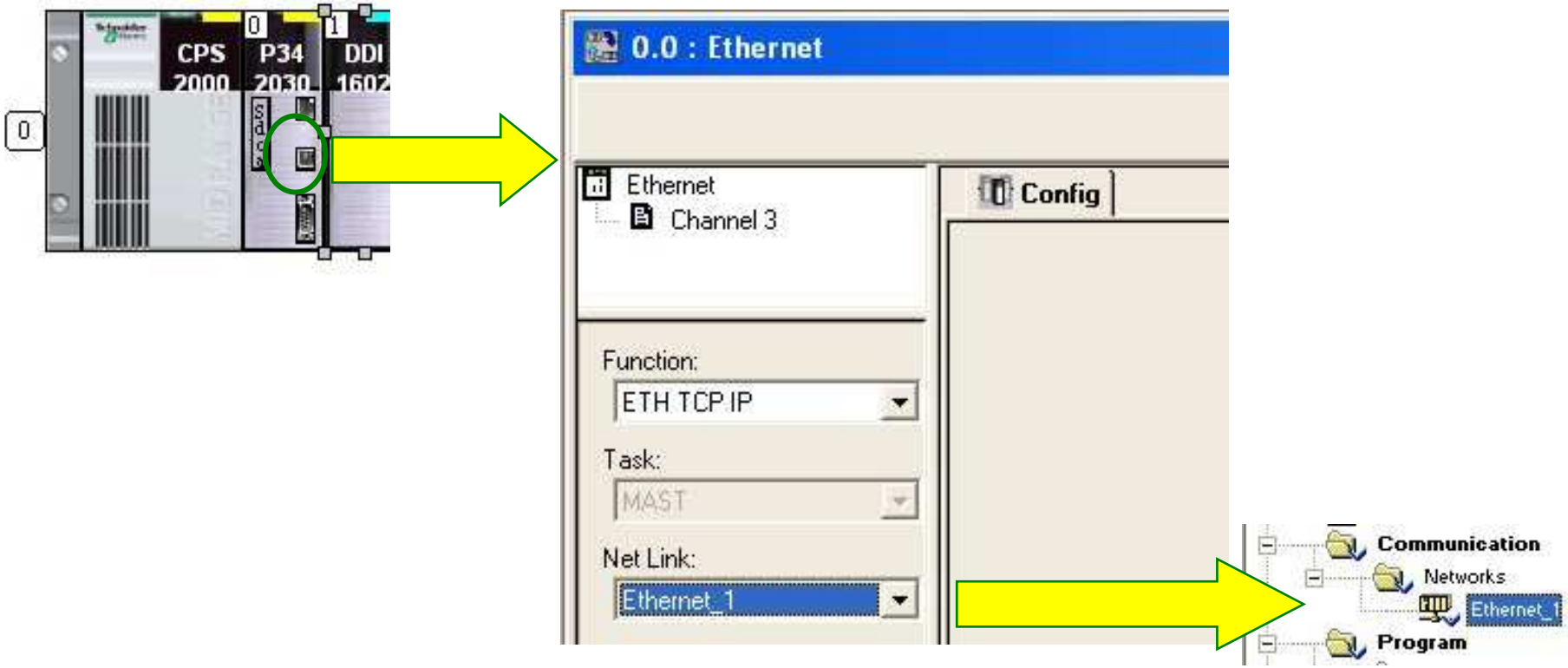
Configure Ethernet network



1 – Create a logical network (Ethernet_1)

2 – Configure the logical network

Configure Ethernet network (cont)



3 – Access Ethernet communication port (or module)

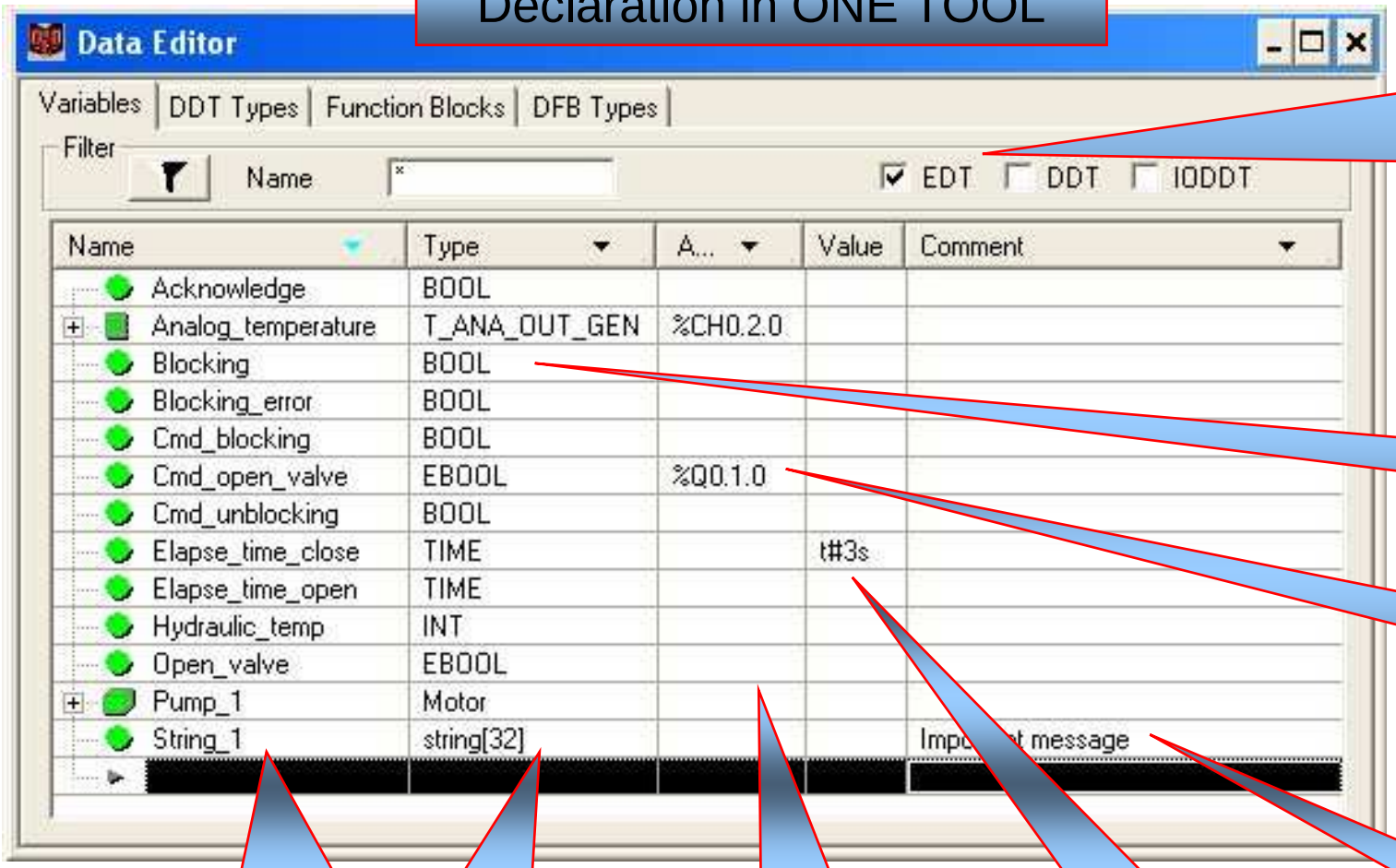
4 – Associate channel to logical network

Variables overview

- A variable said unlocated is define by a symbolic name (32 characters) and a type. It's not possible to know the position in memory or to set cyclically an unlocated variable
- A variable also mapped to an I/O module or associated to a memory reference is said to be a located variable
- A function blocks can use public variables (accessible by the function block and the application program) or private variable (only accessible by the function block itself)
- An IODDT (Input Output Derived Data Type) designates a structure representing the channel of a PLC module
- Constants cannot be modified by the program during execution
- Unity Pro software provides Elementary Data Types (BOOL, EBOOL, INT, WORD, REAL, String, …) or Derived Data Types (Array, structure)

Editing variables

Declaration in ONE TOOL



Direct filter (on type, names)

Funnel filter (on properties)

Unlocated variable

Located variable

Symbolic name

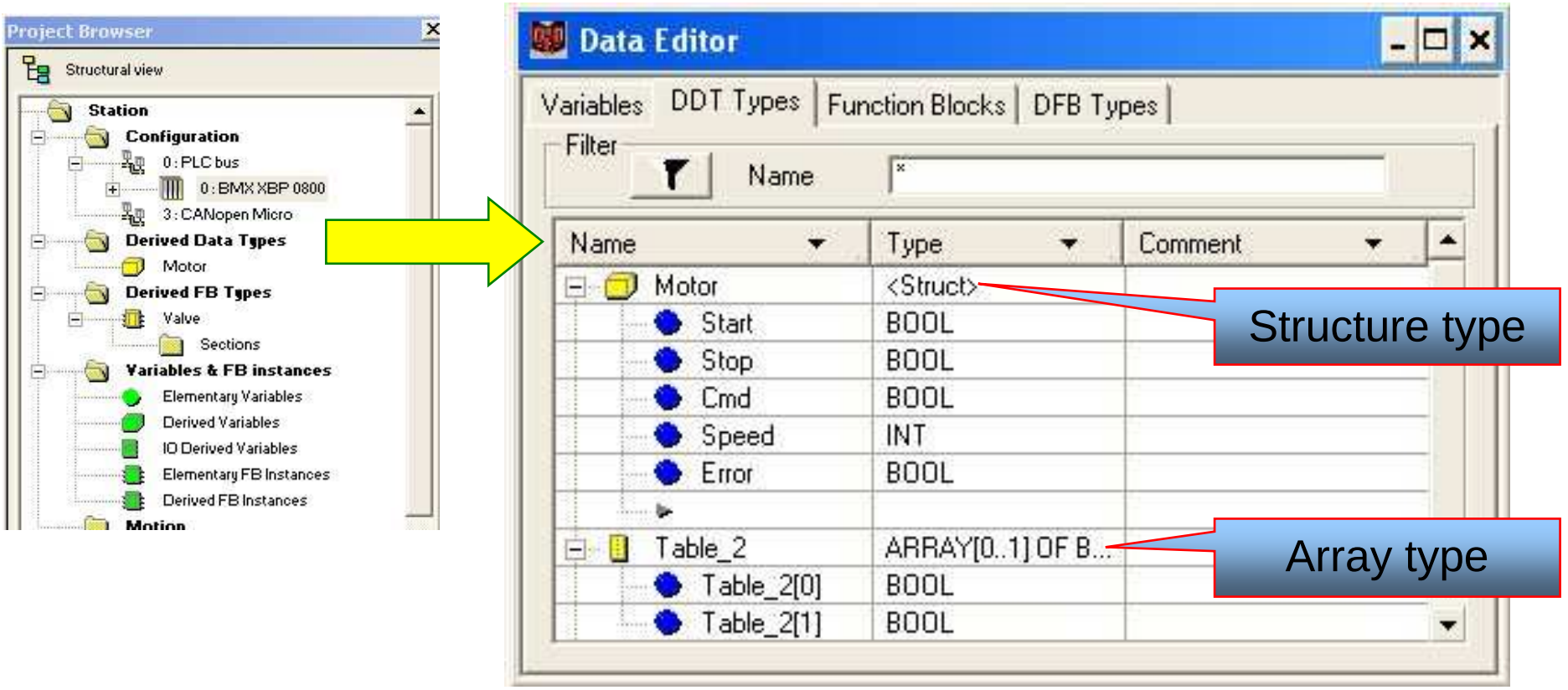
Data type

Address

Initial value

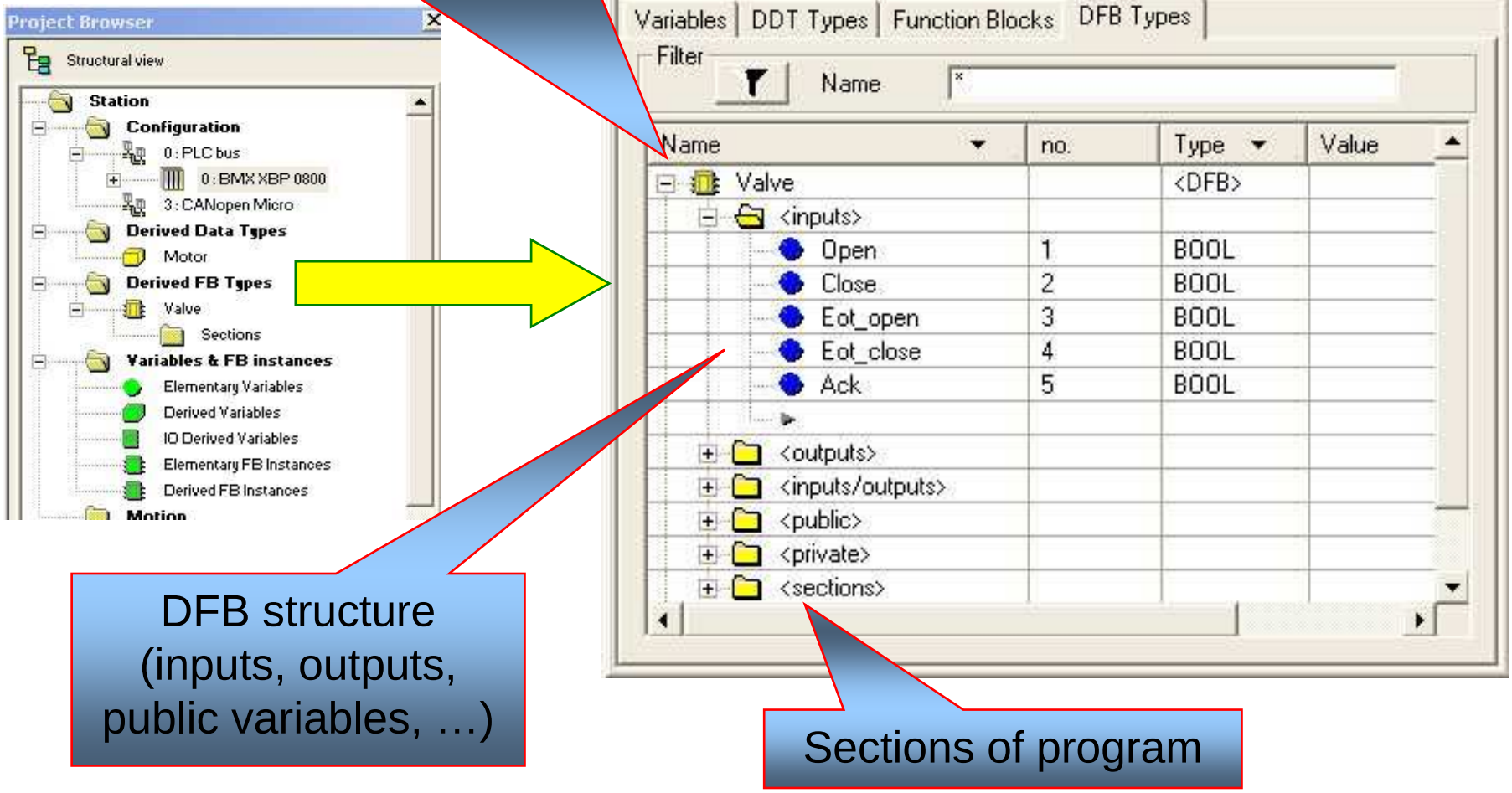
Comment

DDT types


Project Browser
Structural view
Station
Configuration
0 : PLC bus
0 : BMX XBP 0800
3 : CANopen Micro
Derived Data Types
Motor
Derived FB Types
Valve
Sections
Variables & FB instances
Elementary Variables
Derived Variables
IO Derived Variables
Elementary FB Instances
Derived FB Instances
Motion
Data Editor
Variables
DDT Types
Function Blocks
DFB Types
Filter
Name
*
Name
Type
Comment
Motor
<Struct>
Start
BOOL
Stop
BOOL
Cmd
BOOL
Speed
INT
Error
BOOL
Table_2
ARRAY[0..1] OF B...
Table_2[0]
BOOL
Table_2[1]
BOOL
Structure type
Array type

DFB types

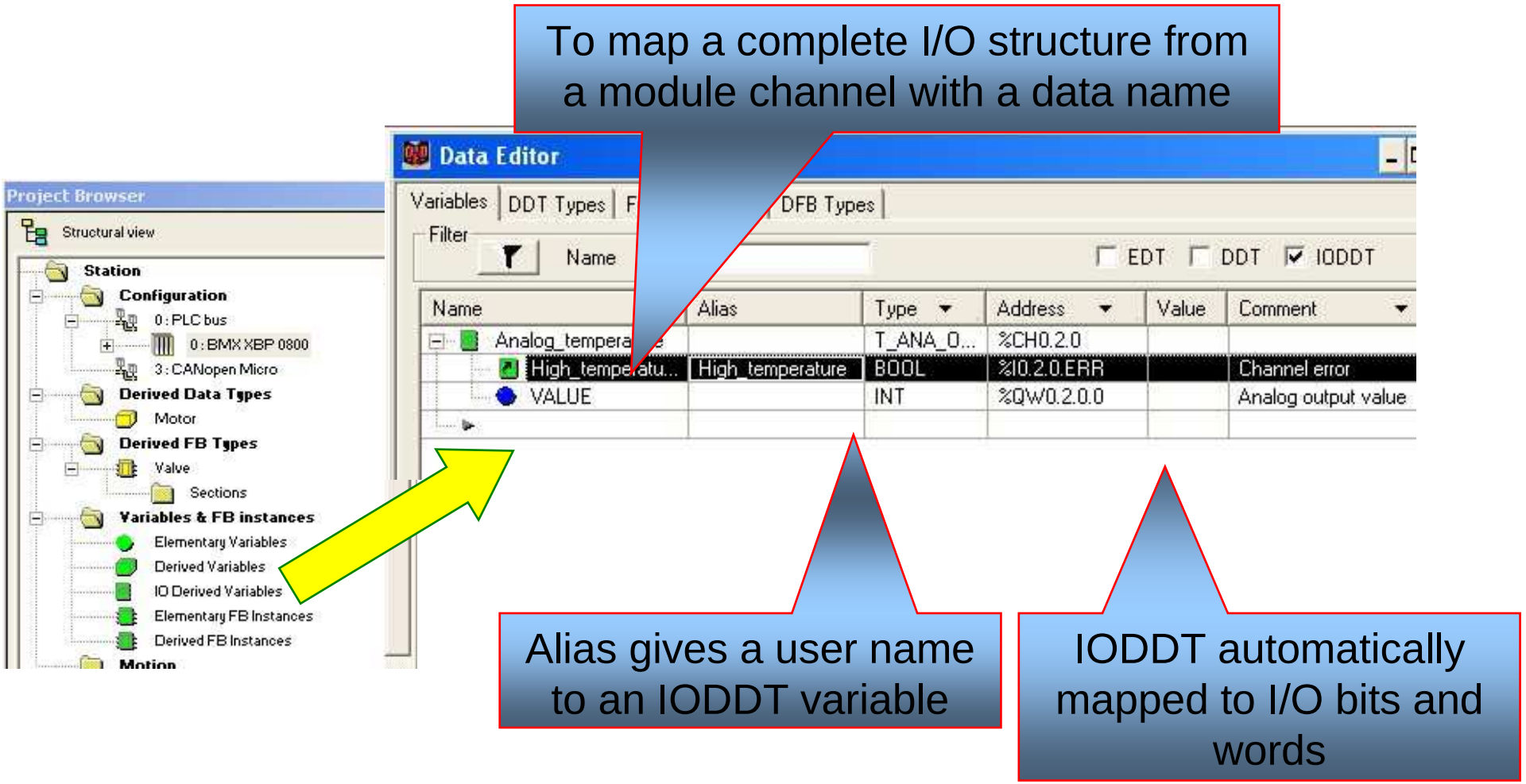
Analyze DFB type



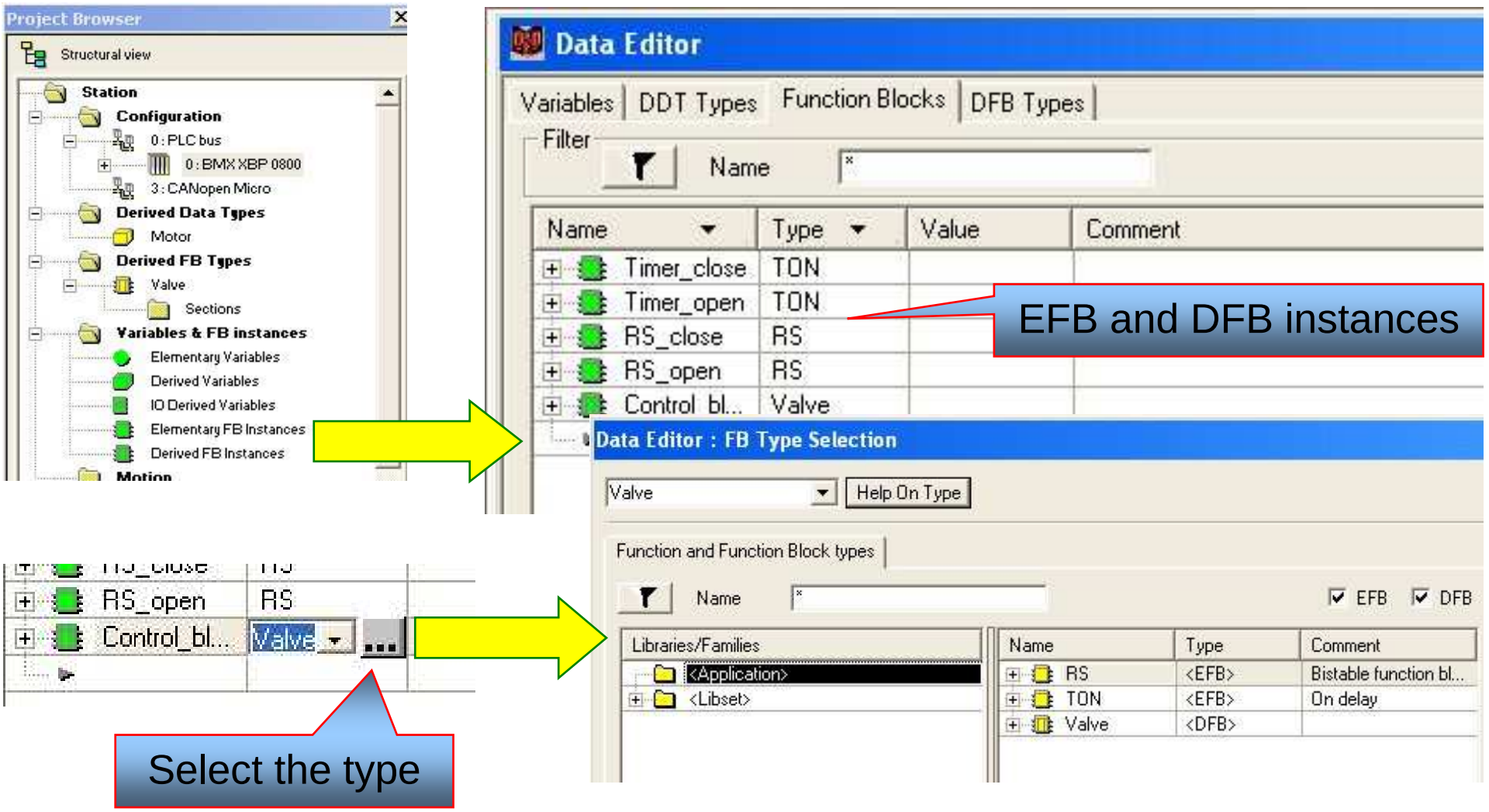
DFB structure (inputs, outputs, public variables, …)

Sections of program

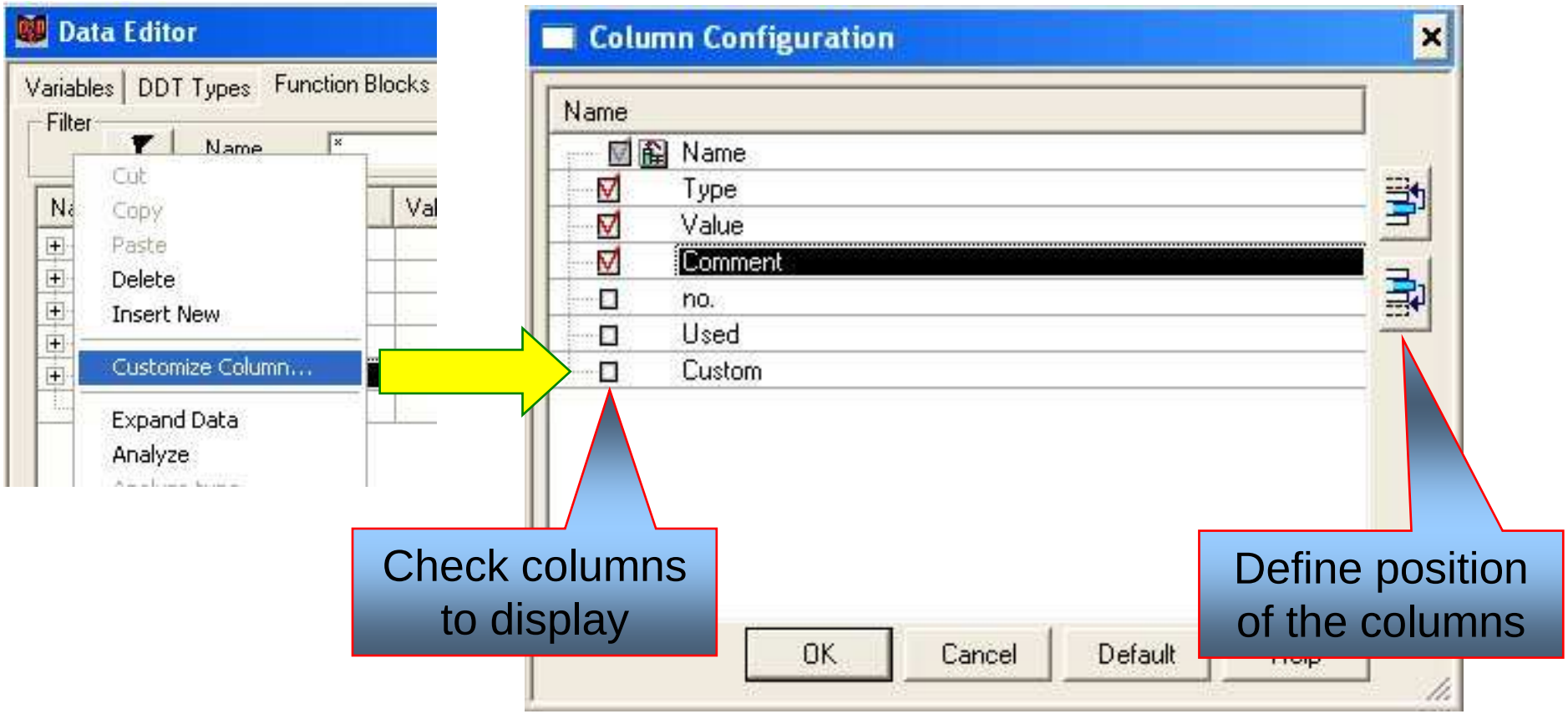
IODDT


To map a complete I/O structure from a module channel with a data name
Project Browser
Structural view
Station
Configuration
0 : PLC bus
0 : BMX XBP 0800
3 : CANopen Micro
Derived Data Types
Motor
Derived FB Types
Valve
Sections
Variables & FB instances
Elementary Variables
Derived Variables
IO Derived Variables
Elementary FB Instances
Derived FB Instances
Motion
Data Editor
Variables
DDT Types
DFB Types
Filter
Name
EDT
DDT
IODDT
Name
Alias
Type
Address
Value
Comment
Analog_tempera
T_ANA_0...
%CH0.2.0
High_temperatu...
High_temperature
BOOL
%I0.2.0.ERR
Channel error
VALUE
INT
%QW0.2.0.0
Analog output value
Alias gives a user name to an IODDT variable
IODDT automatically mapped to I/O bits and words

Function block instances

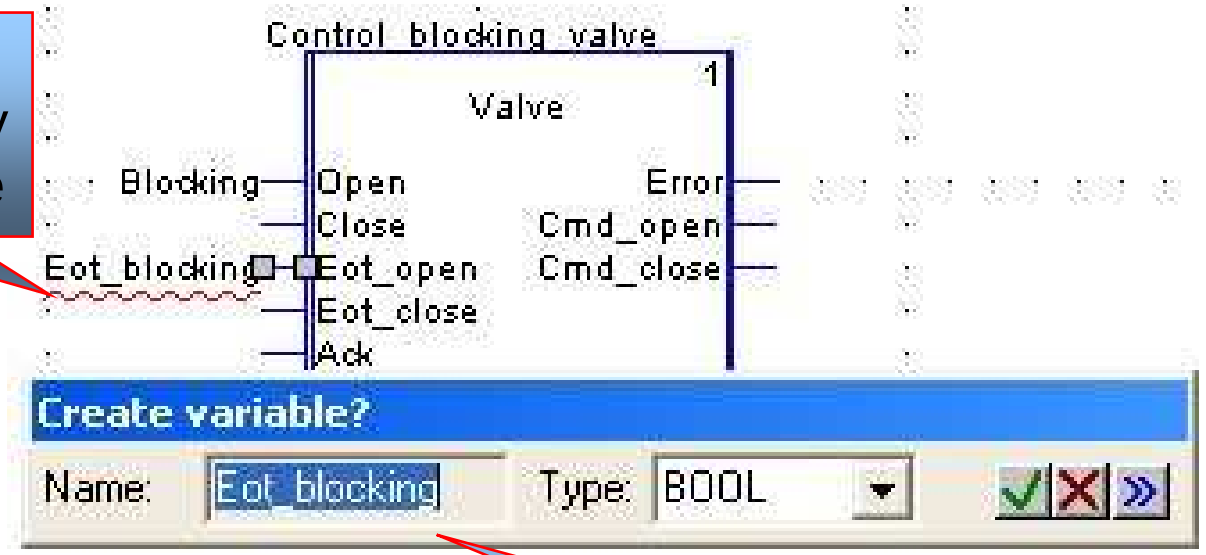


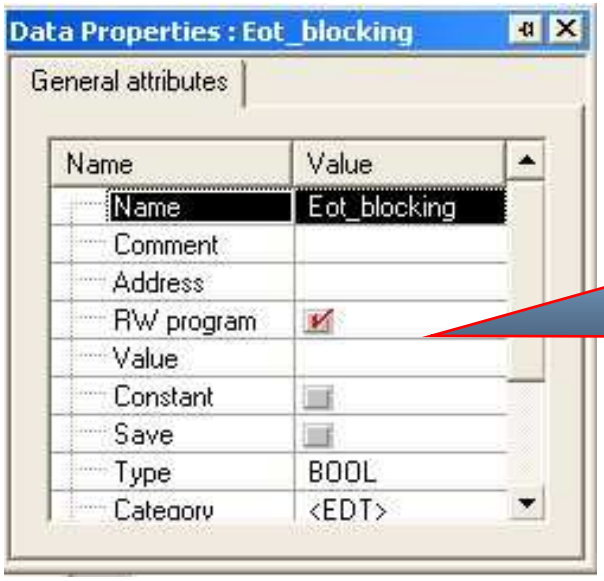
Configure columns to display



Editing a variable through program editor

Smart analyzer check availability of variable name

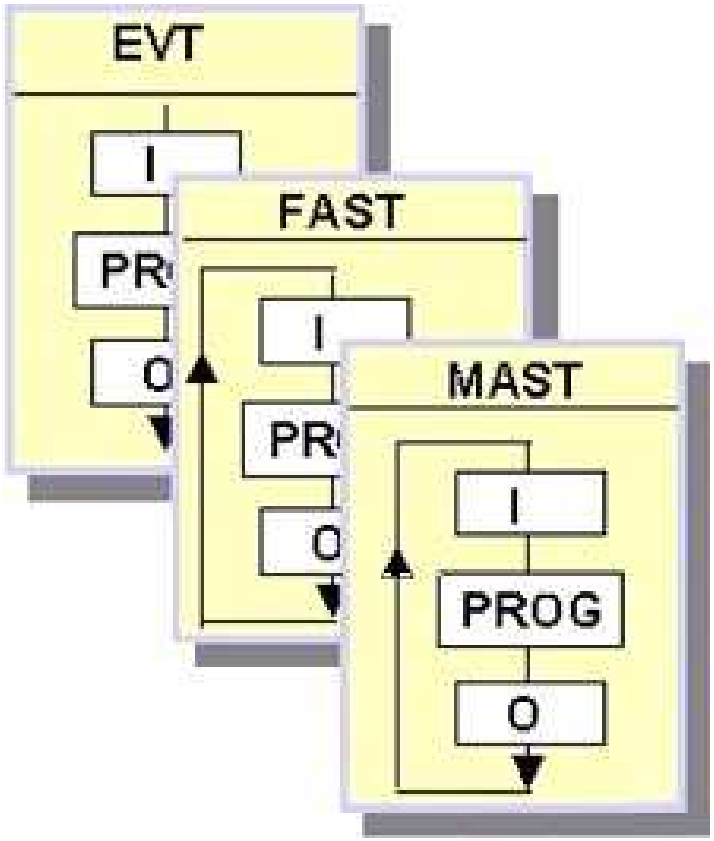




Display or modify variable attributes inside the editors

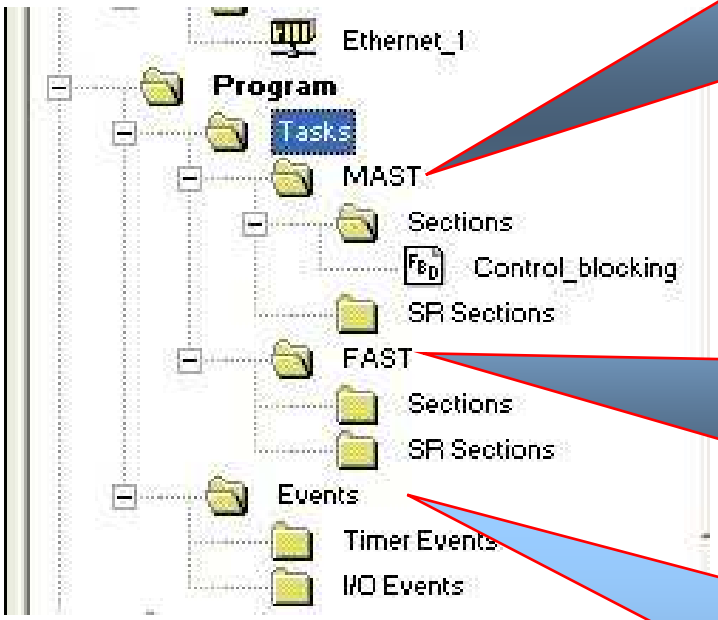
Popup (window) to create variable if it doesn't exist

Application structure



- In different tasks
  - Single task (only MAST) or multitask (MAST and FAST)
  - Even tasks : IO event (EVT) and TIMER event
- In execution mode
  - Cyclic execution (only for MAST task)
  - Periodic execution
- In sections
  - Tasks are structured in sections of program
- In subroutines (SR)
  - MAST and FAST tasks can contain subroutines. A subroutine is called by a section or another subroutine

Tasks



MAST

Contains sections and subroutines

Programmed in LD, FBD, IL, ST or SFC

Cyclic or periodic execution (0 to 255 ms)

Controlled by watch dog and system bits & words

FAST

Contains sections and subroutines

Programmed in LD, FBD, IL or ST

Periodic execution (1 to 255 ms)

Controlled by watch dog and system bits & words

EVT and TIMER

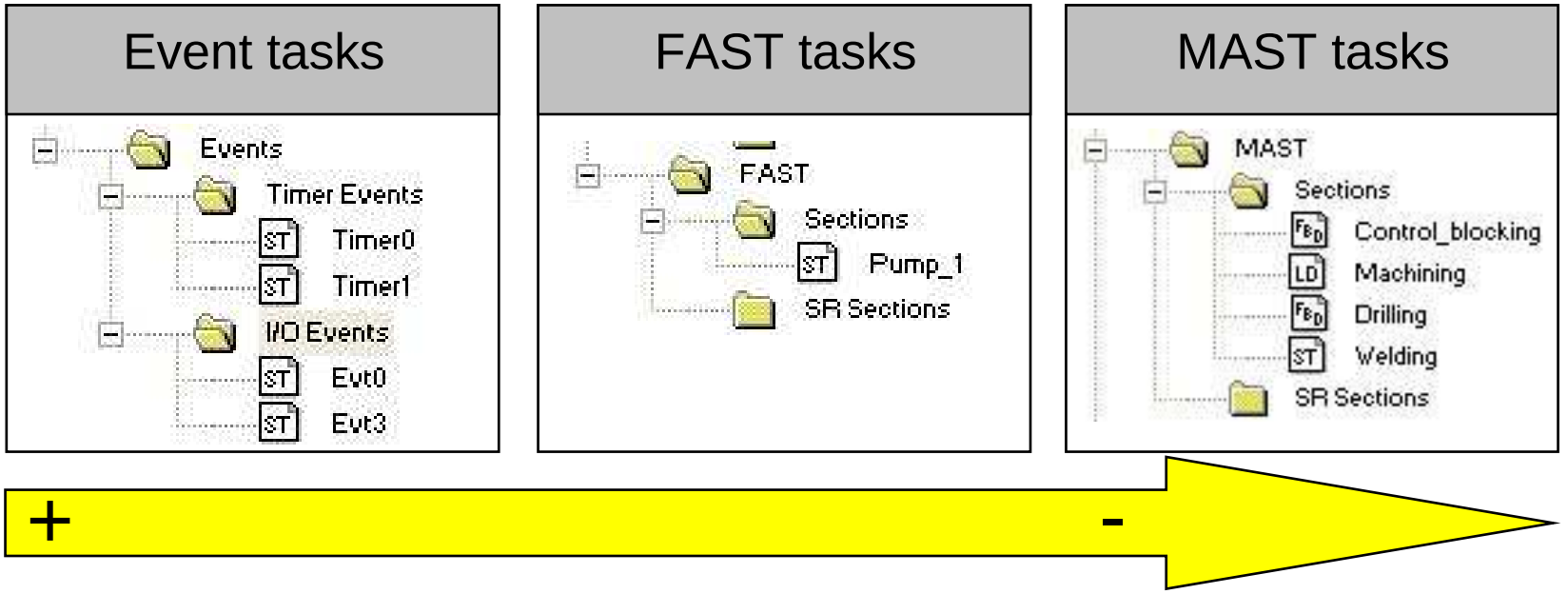
To reduce response time of application program

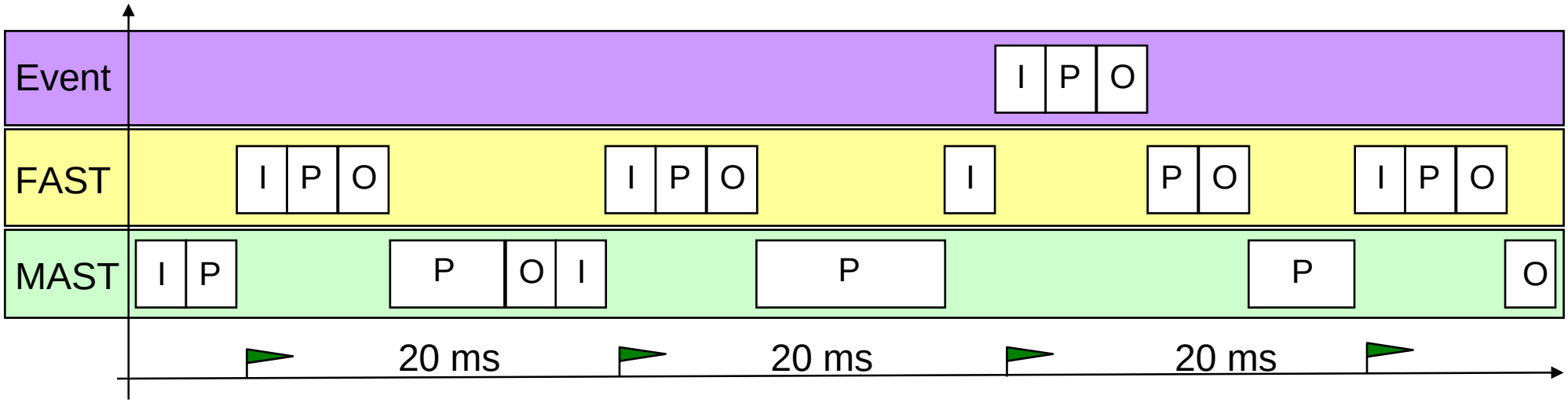
Single section programmed in LD, FBD, IL or ST

EVTi : events come from I/O modules

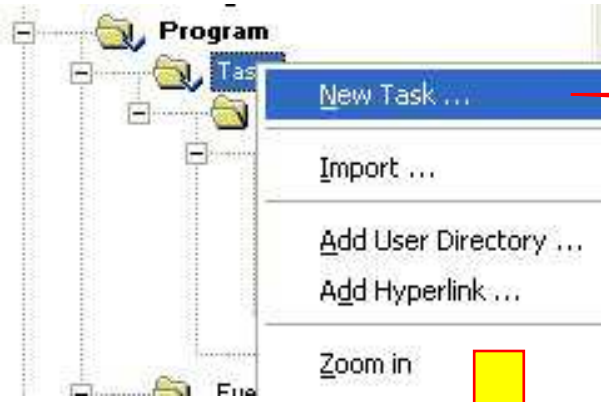
TIMERi : events come from event timers

Priority and execution





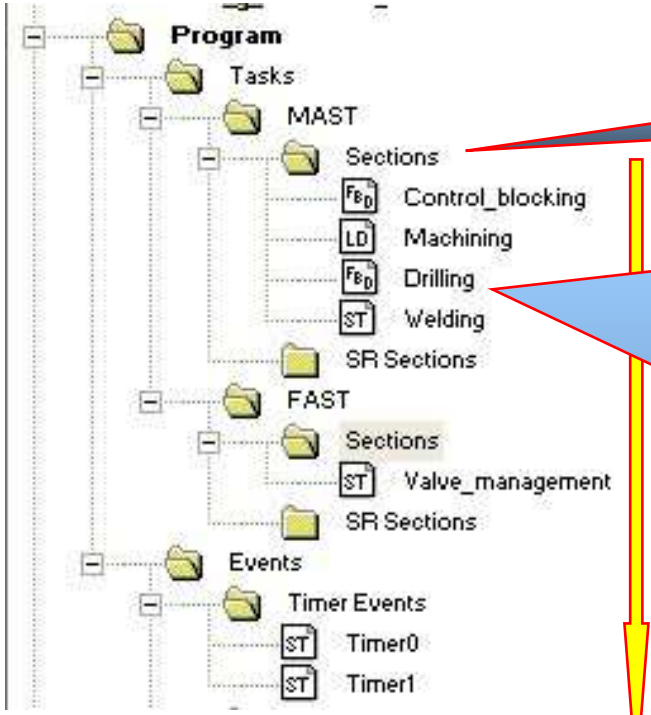
Add FAST task



Right-click => New task

Select the period and the watchdog

Sections



Unlimited number of sections

Attributes of each section

Name : 32 characters maximum

Language : LD, FBD, IL, ST or SFC

Associated task : MAST, FAST, Event

Condition (optional) : validation bit for execution

Localization : functional module containing the section

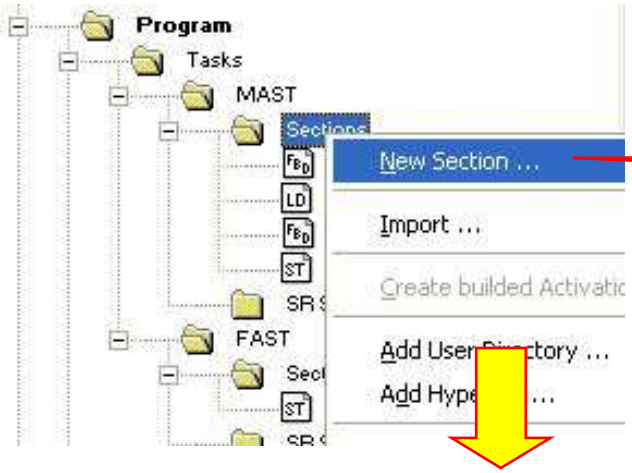
Protection : read only or no read/write

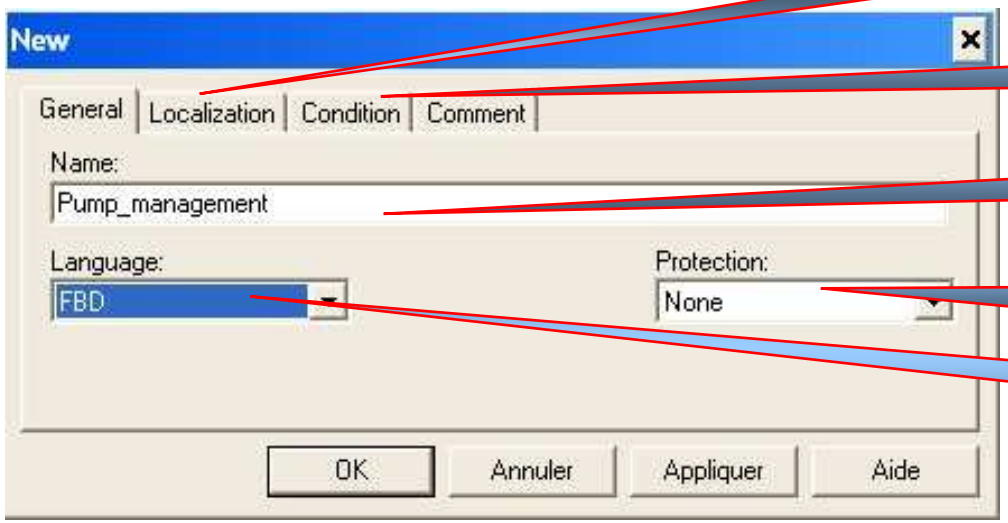
Comment : 256 characters maximum

Order of execution

Order in which the sections appear in the browser

Add new section





Right-click > New section

Localize the section

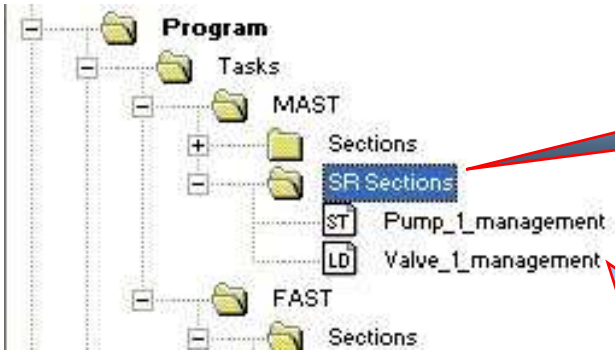
Condition execution

Enter the section name

Protect the section

Select the language

Subroutines



Unlimited number of subroutines

Attributes of each subroutine

Name : 32 characters maximum

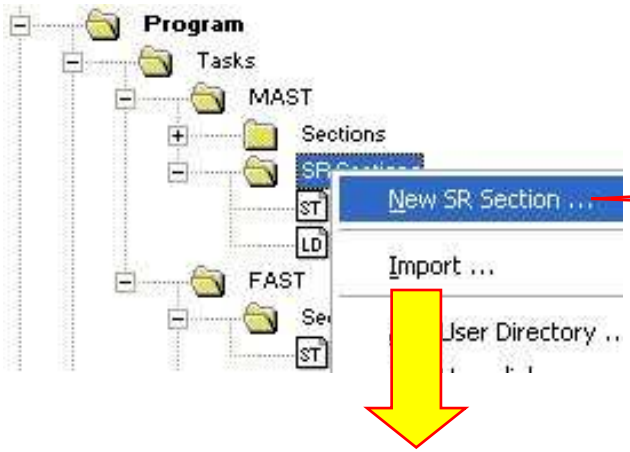
Language : LD, FBD, IL, ST

Associated task : MAST, FAST

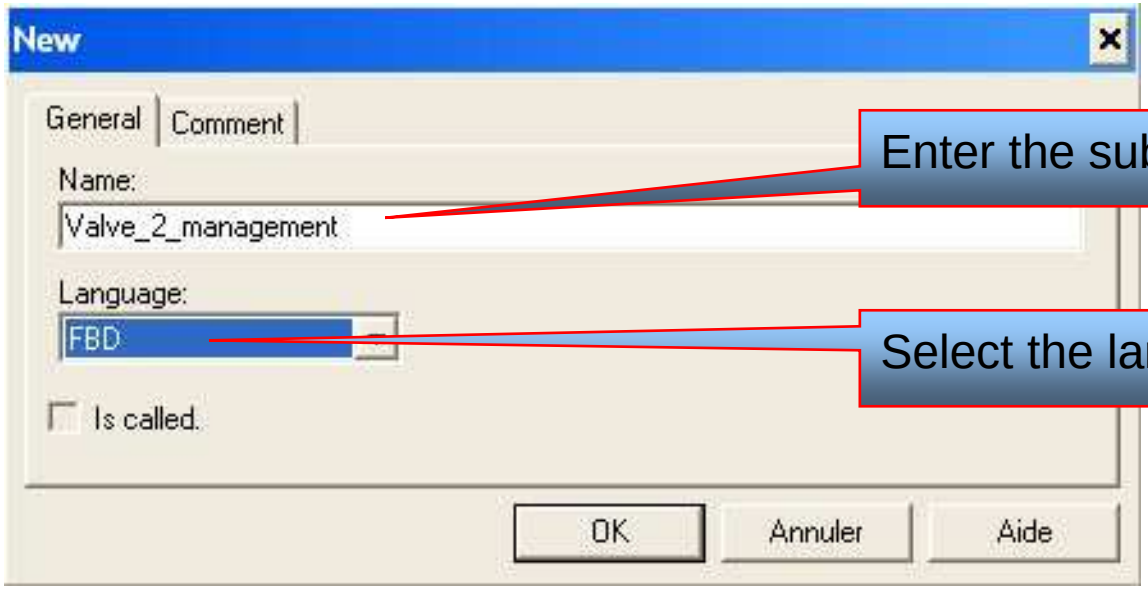
Call is carried out from a section or another subroutine. Number of nesting is limited to 8. A subroutine should not call itself (recursive)

Subroutine is linked to a task. It cannot be called from sections or subroutines of the other task

Add new subroutine



Right-click > New SR section



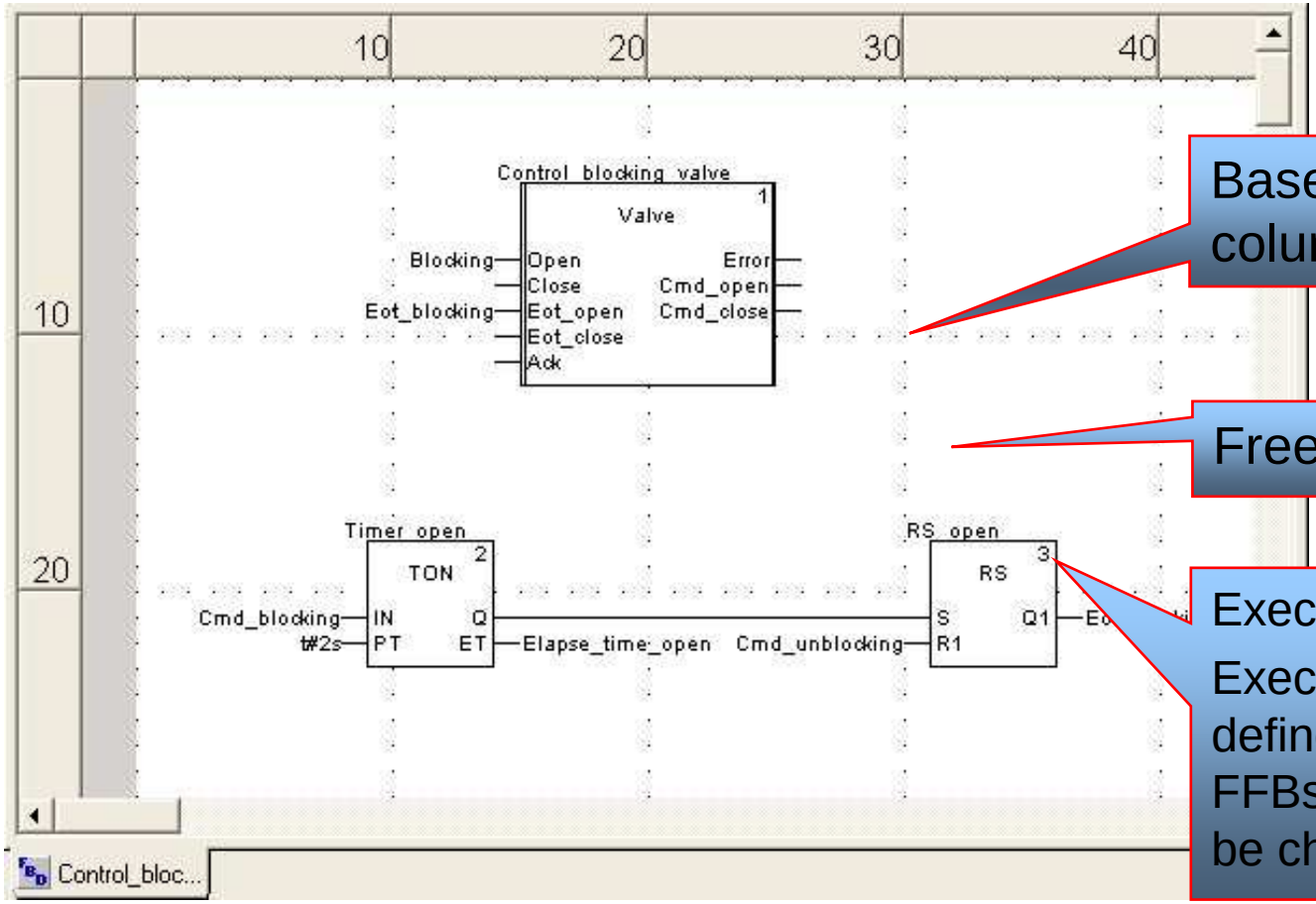
Enter the subroutine name

Select the language

FBD language overview

- Function Block Diagram (FBD) is a graphical language that is data flow oriented
- Complies with IEC 61131-3 standard
- Particularly suitable for continuous or discrete control applications
- FBD program uses instances of elementary and derived function blocks linked each other
- FBD programming is not cell oriented

FBD editor



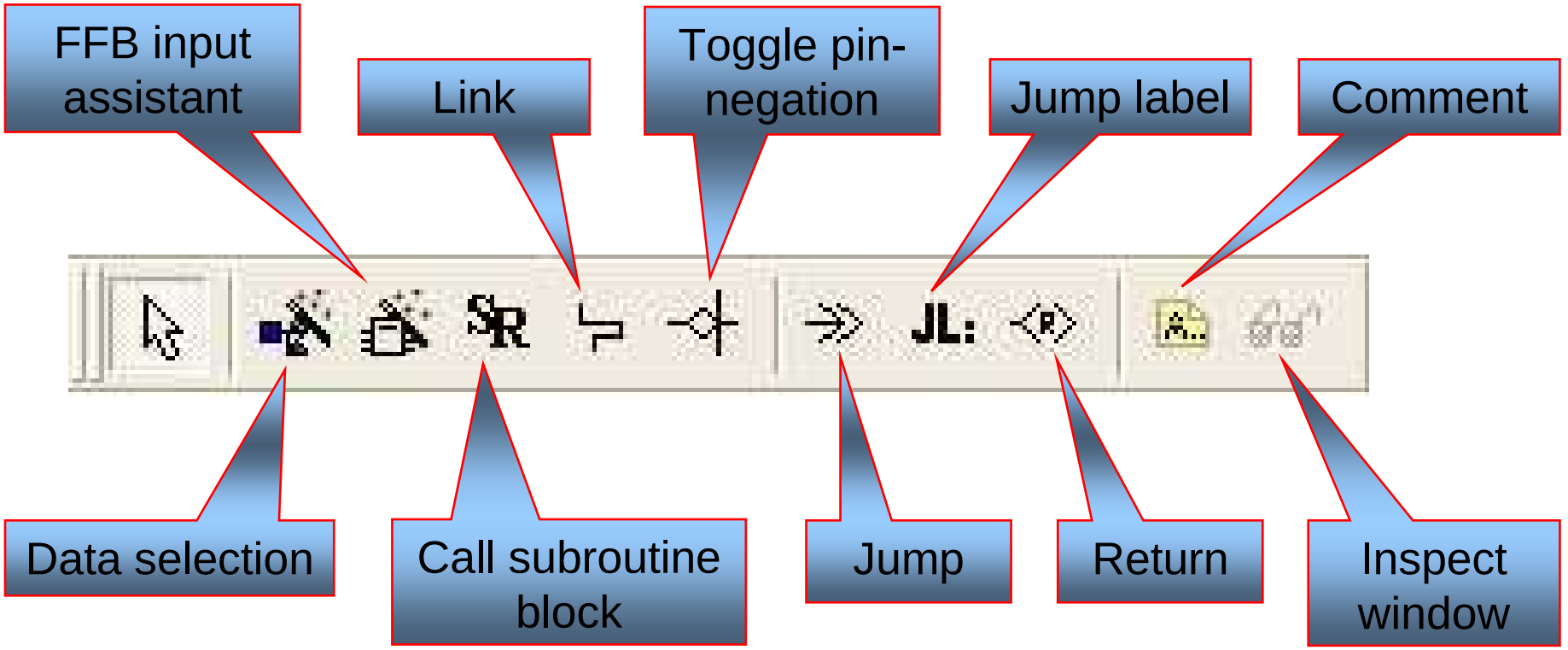
Based on a grid (36 columns x 24 rows)

Free form editor

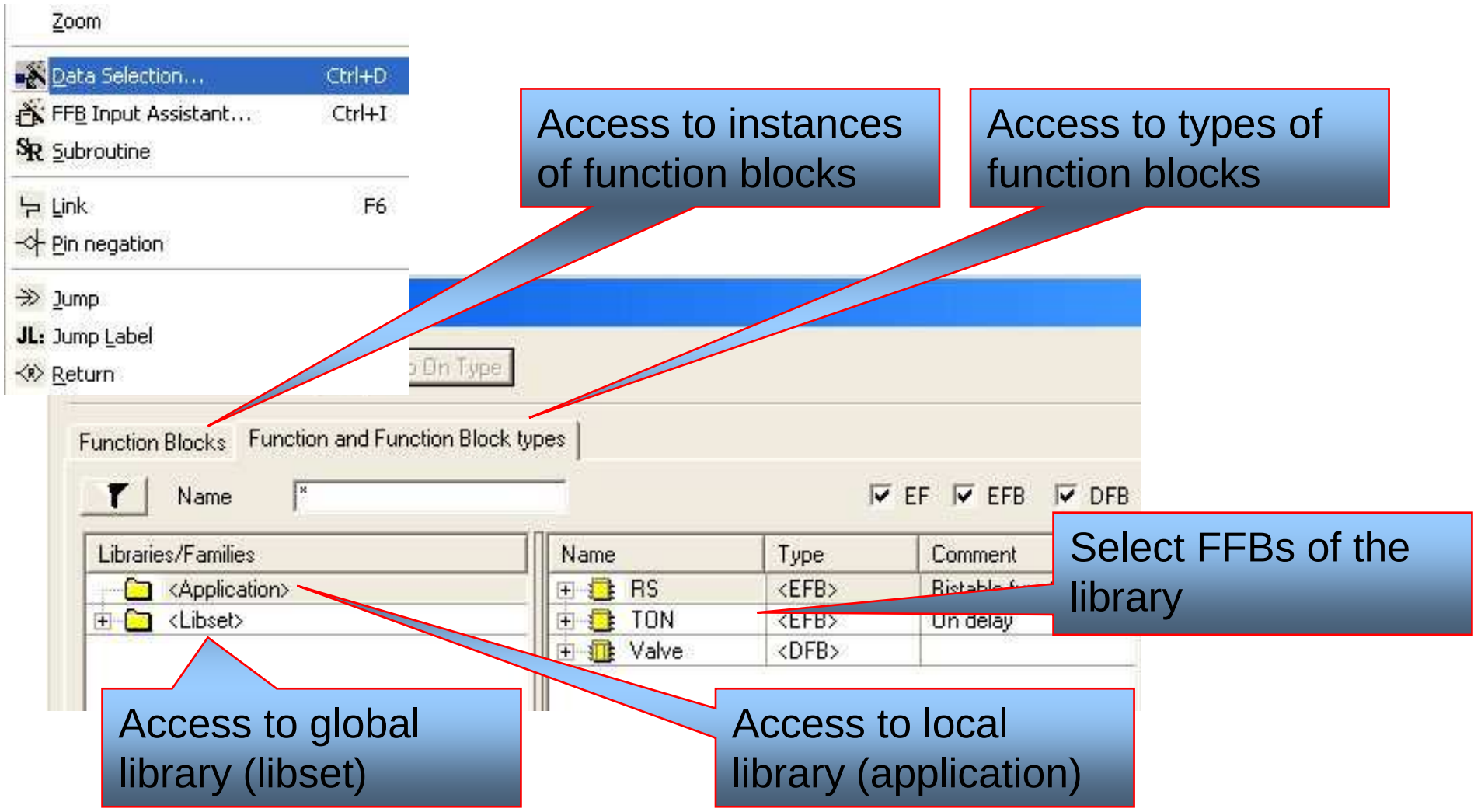
Execution number

Execution sequence is defined by position of FFBs and links but can be changed

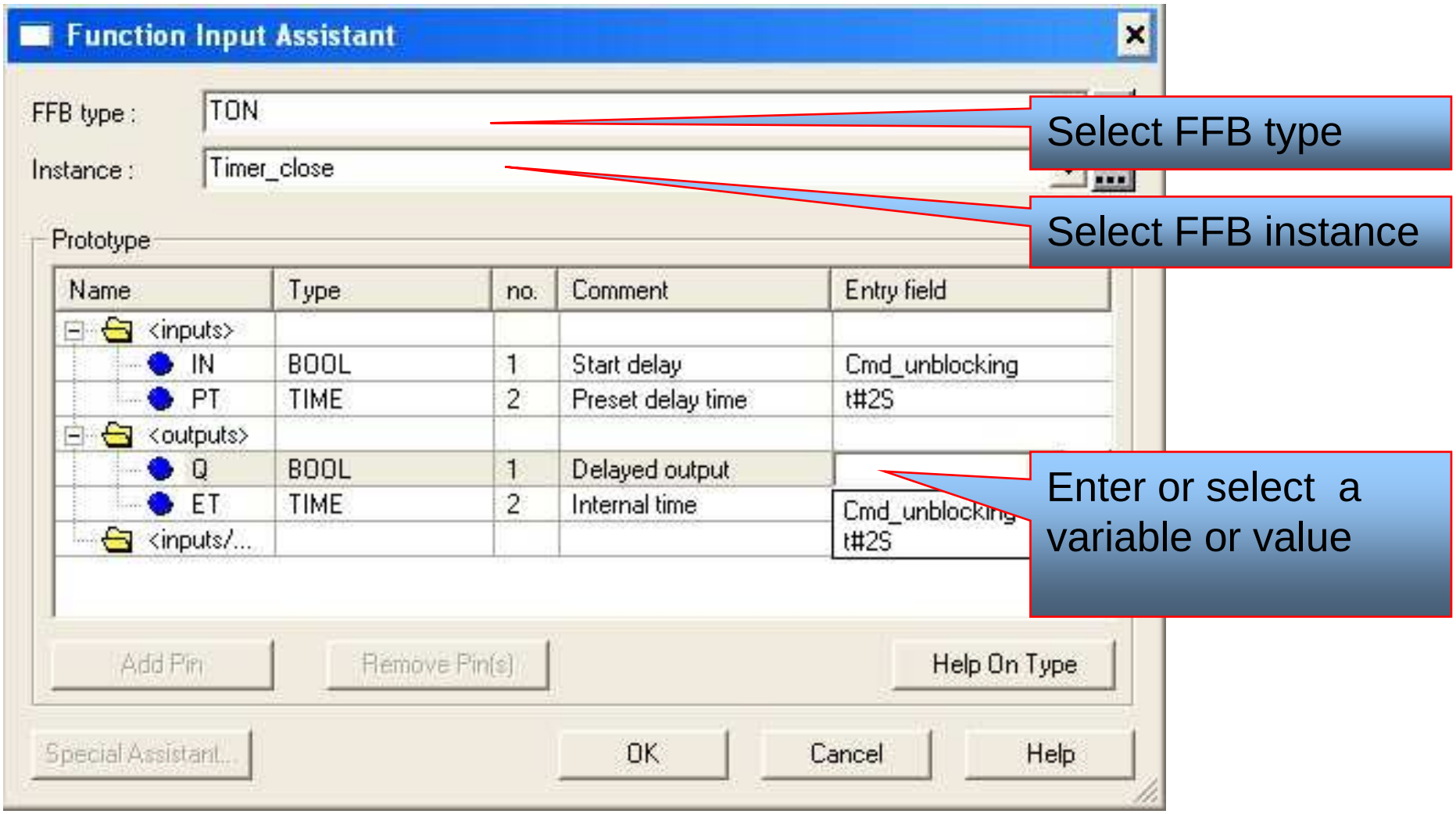
FBD specific toolbar



Select a FFB


Zoom
Data Selection... Ctrl+D
FFB Input Assistant... Ctrl+I
Subroutine
Link F6
Pin negation
Jump
JL: Jump Label
Return
Access to instances of function blocks
Access to types of function blocks
Function Blocks
Function and Function Block types
Name
EF
EFB
DFB
Libraries/Families
<Application>
<Libset>
Name
Type
Comment
RS
<EFB>
TON
<EFB>
On delay
Valve
<DFB>
Select FFBs of the library
Access to global library (libset)
Access to local library (application)

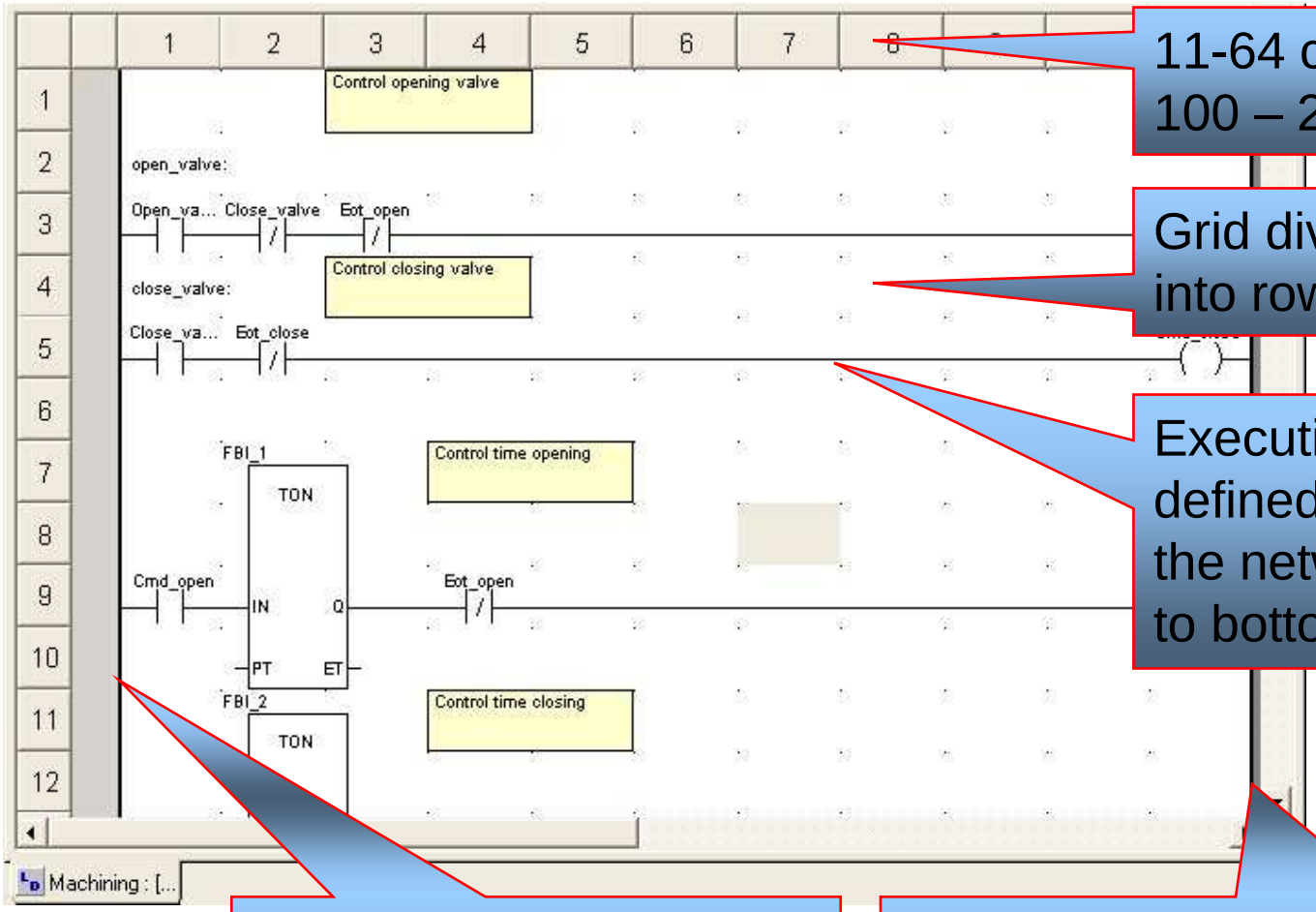
Use FFB input assistant



LD language overview

- Ladder Diagram (LD) corresponds to a rung for relay switching. Left power rail corresponds to the phase and right power rail corresponds to the neutral of the rung
- Complies with IEC 61131-3 standard
- A group of objects linked together with no link to other objects (excluding power rail) is called a network
- LD programming language is cell oriented (only one object can be placed in each cell)
- Process sequence is determined by the data flow within the section. Networks connected to the left power rail are processed from top to bottom

LD editor



11-64 columns and 100 – 2000 lines

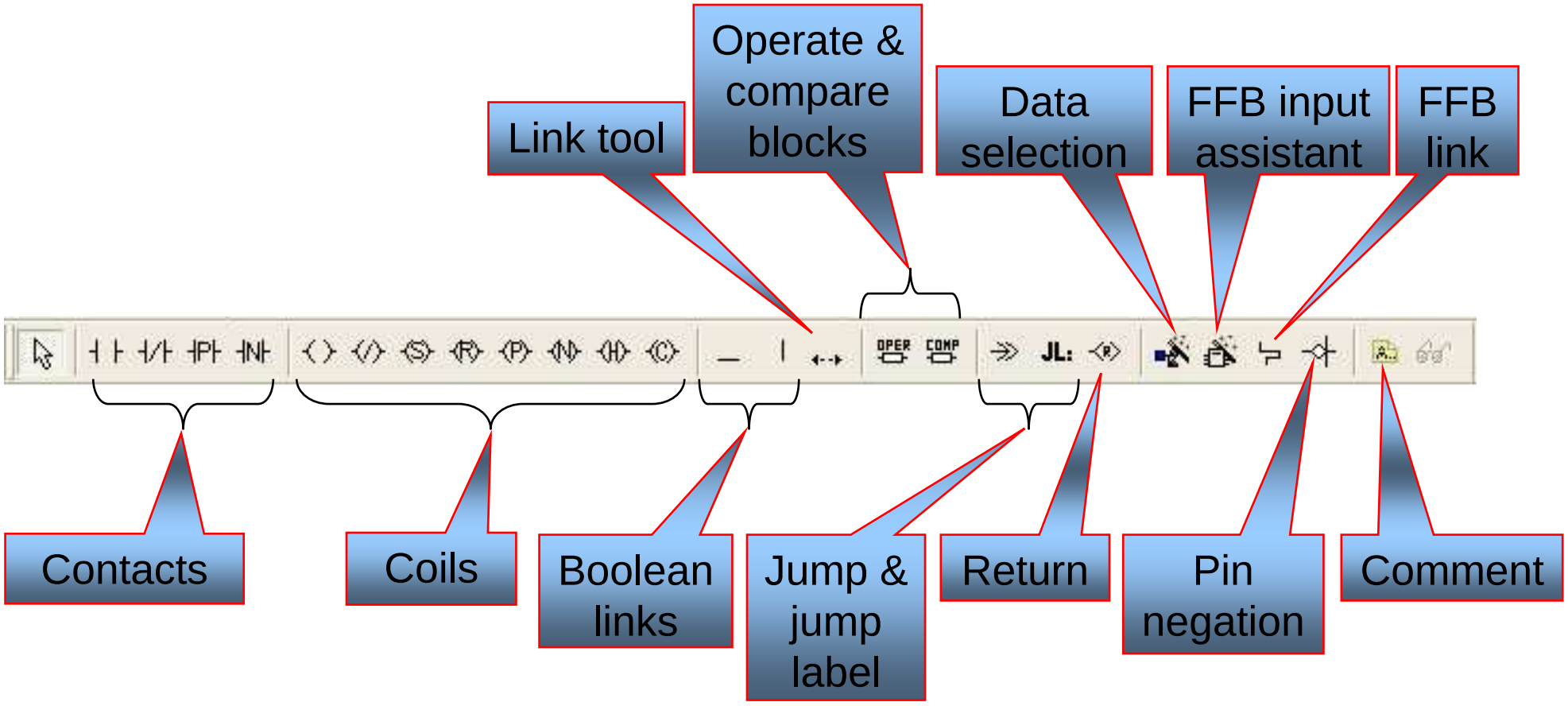
Grid divides section into rows and columns

Execution sequence is defined by position of the networks (from top to bottom)

Left power rail (phase)

Right power rail (neutral)

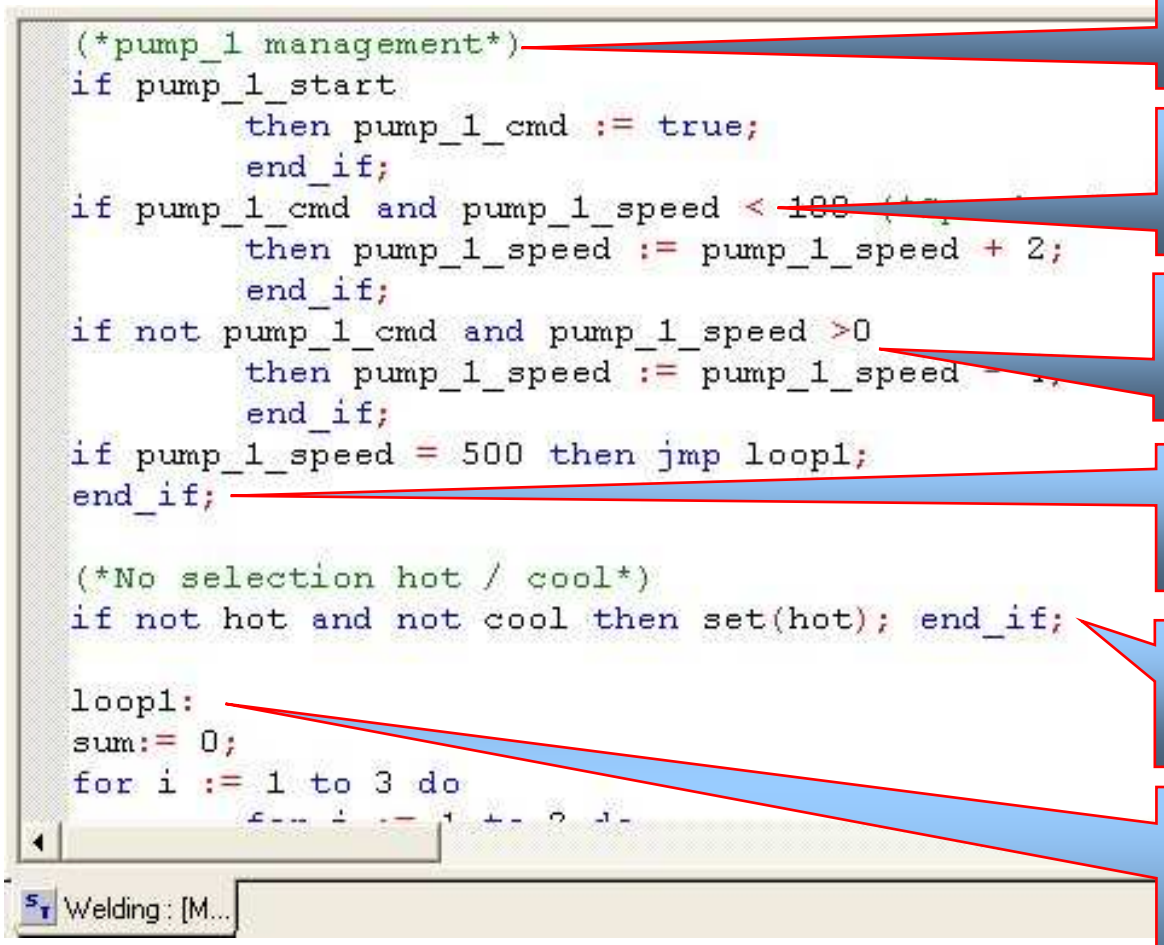
LD specific toolbar



ST language overview

- Structured Text (ST) is a computer language using comprehensive range of constructs for assigning values to variables, calling FFBs, creating expressions
- Complies with IEC 61131-3 standard
- Used to write structured logical and numerical processing programs (nested control constructs)
- Easy to learn and use
- Particularly suitable for programming complex functions like arithmetic functions, ….
- Structured Text program is a sequence of lines

Language reference



Comment : additional information

Operator : defines operation performed

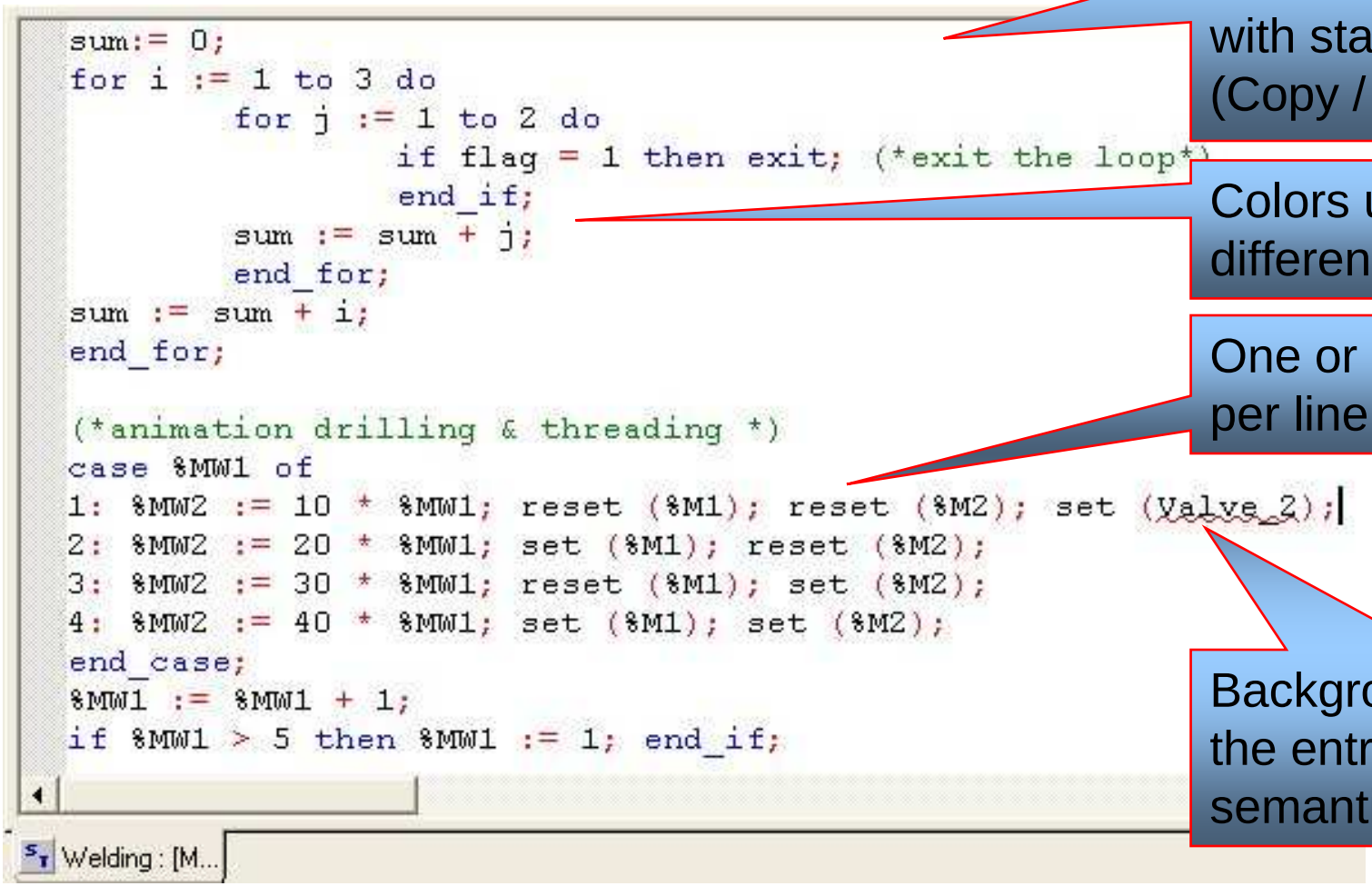
Operand : object that operator acts upon

Control construct (i.e. if then else)

Instruction : ended by semicolon

Label : pinpoints a program block (ended by colon)

ST editor



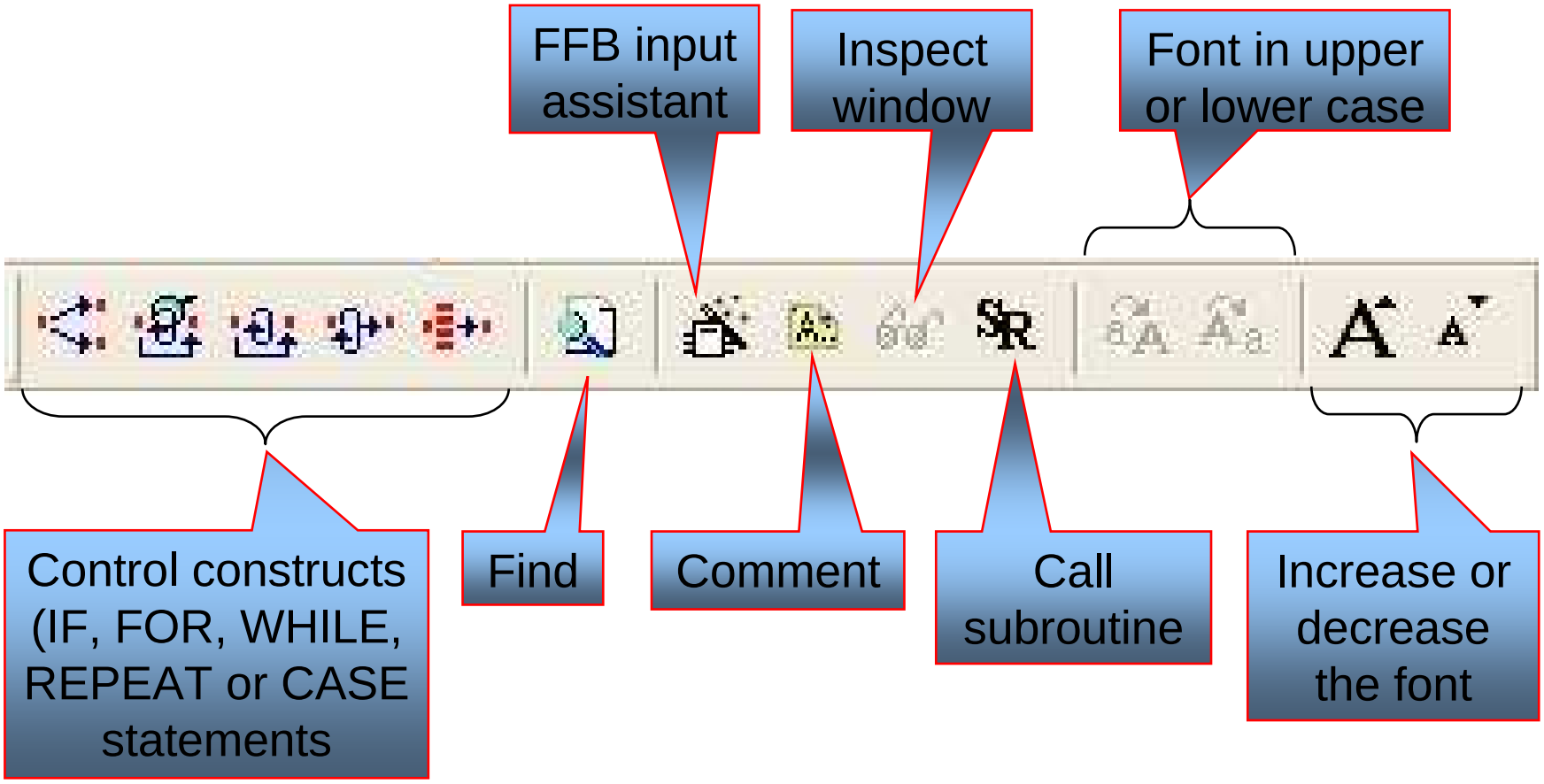
Text editor (WordPad) with standard functions (Copy / Paste, Tab,…)

Colors used to define different objects

One or more instructions per line (separated by ;)

Background analysis of the entry (syntax and semantics check)

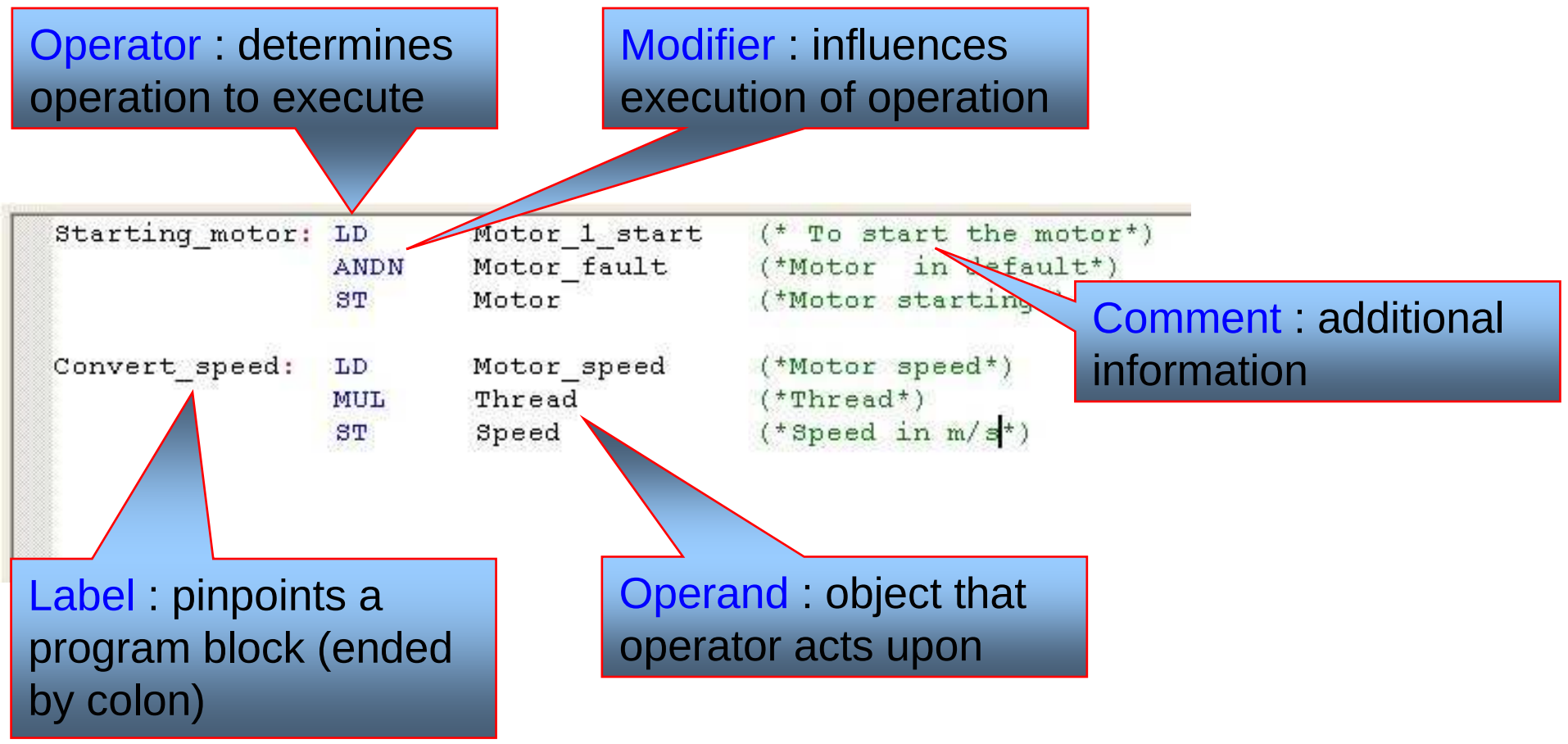
ST specific toolbar



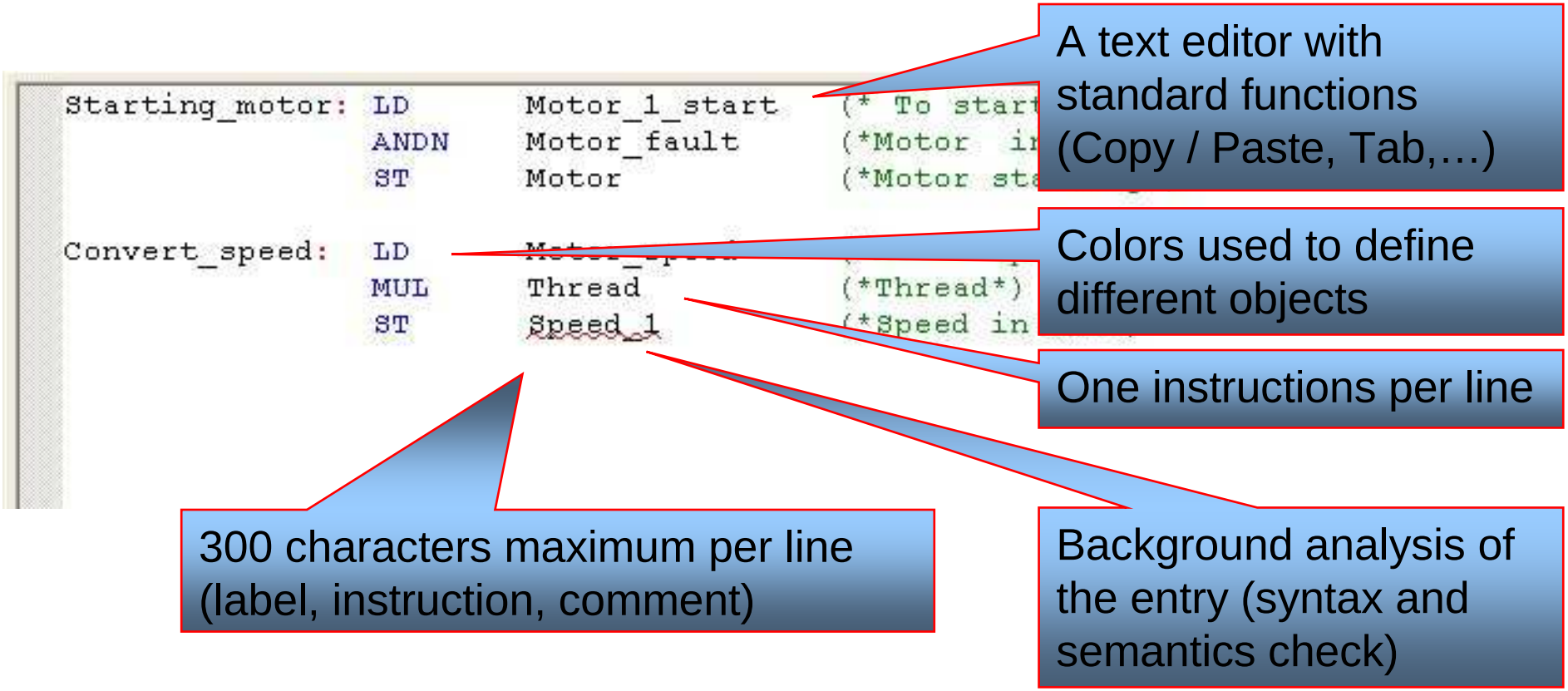
IL language overview

- Instruction List is a Boolean machine language for numerical and logical processing
- Complies with IEC 61131-3 standard
- Optimized and fast code for critical sections
- Ideal for simple applications
- Easy to learn but the program is difficult to read and understand
- Difficult to follow the program flow
- IL program is composed of a series of instructions

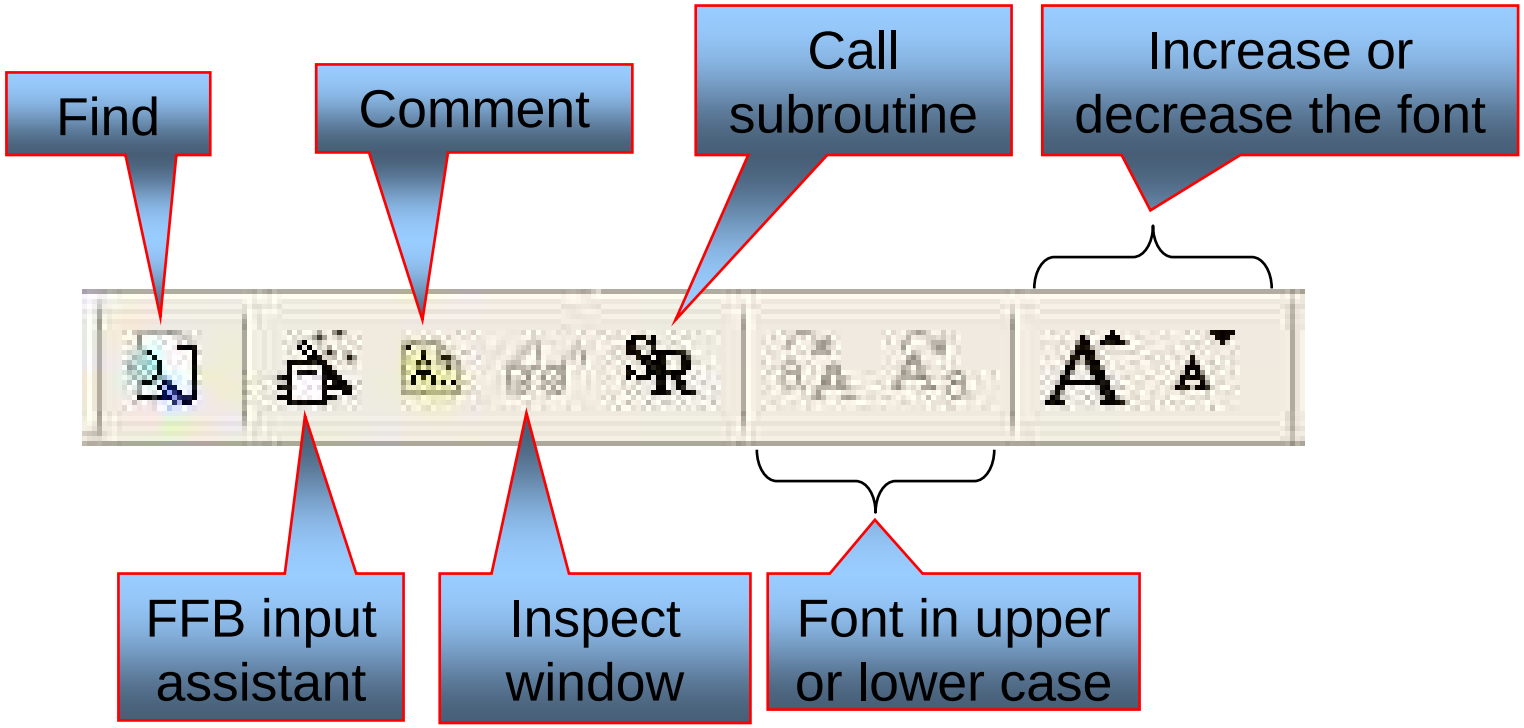
Language reference



IL editor


Starting_motor: LD Motor_1_start (* To start
ANDN Motor_fault (*Motor
ST Motor (*Motor
Convert_speed: LD
MUL Thread (*Thread*)
ST Speed_1 (*Speed in
A text editor with standard functions (Copy / Paste, Tab,…)
Colors used to define different objects
One instructions per line
300 characters maximum per line (label, instruction, comment)
Background analysis of the entry (syntax and semantics check)

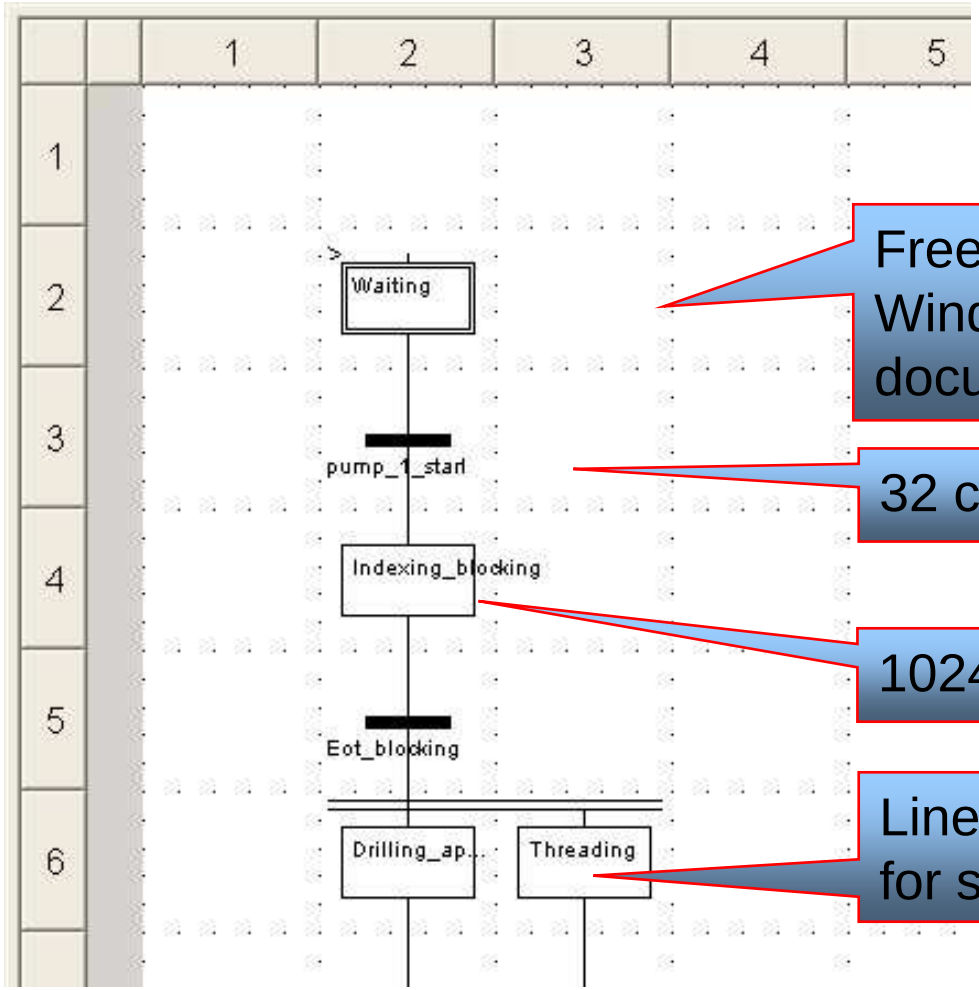
IL specific toolbar



SFC language overview

- Sequential Function Chart is a graphic method to represent a sequential control system by using sequence of steps and transitions
- Complies with IEC 61131-3 standard
- Step is a command or action that is either active or inactive
- Flow passes from one step to the next through a conditional transition that is either true or false
- Sequential program is composed of SFC sections, action sections and transition sections
- SFC sections are only allowed in MAST task
- SFC section contains one or several SFC graphs

SFC editor



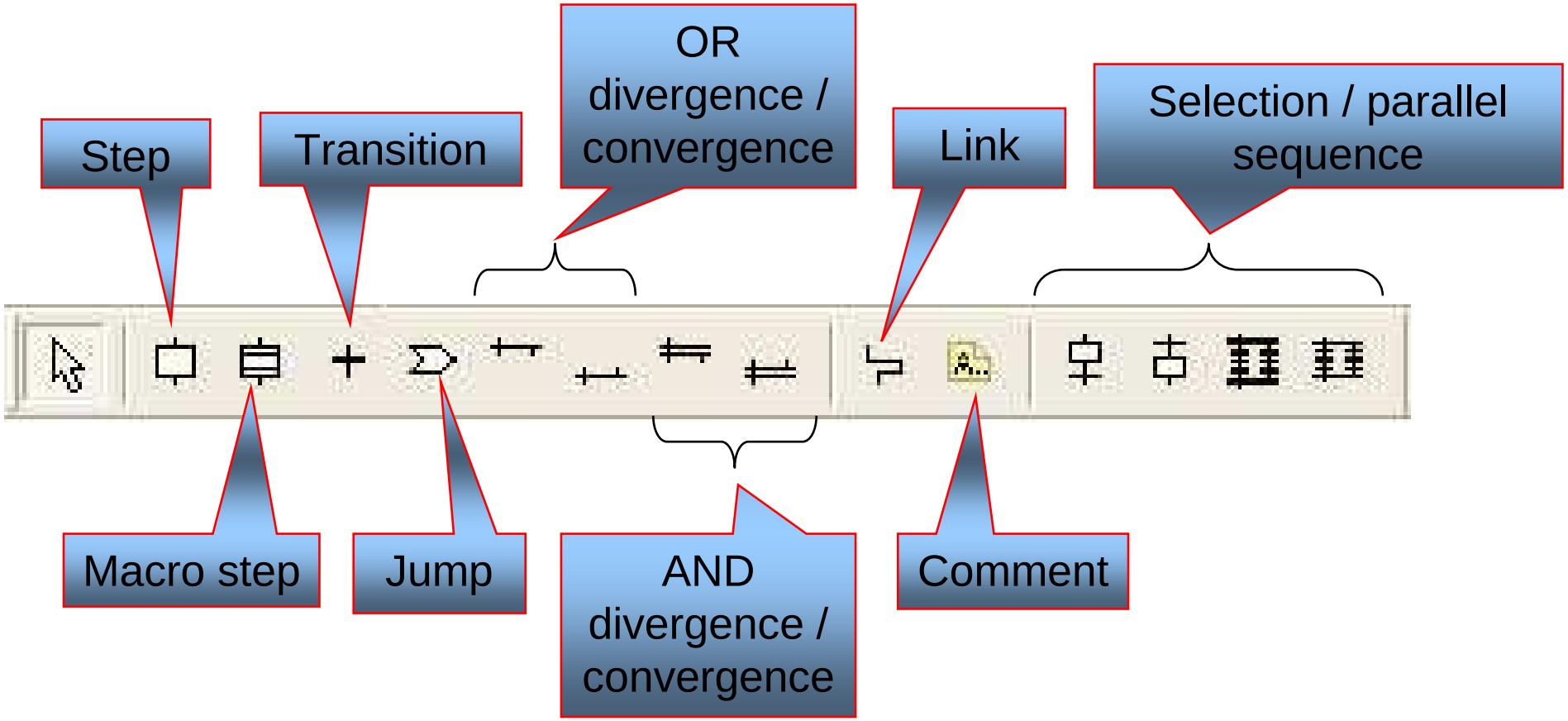
Freeform editor based on Window multiple document interfaces

32 columns and 200 rows

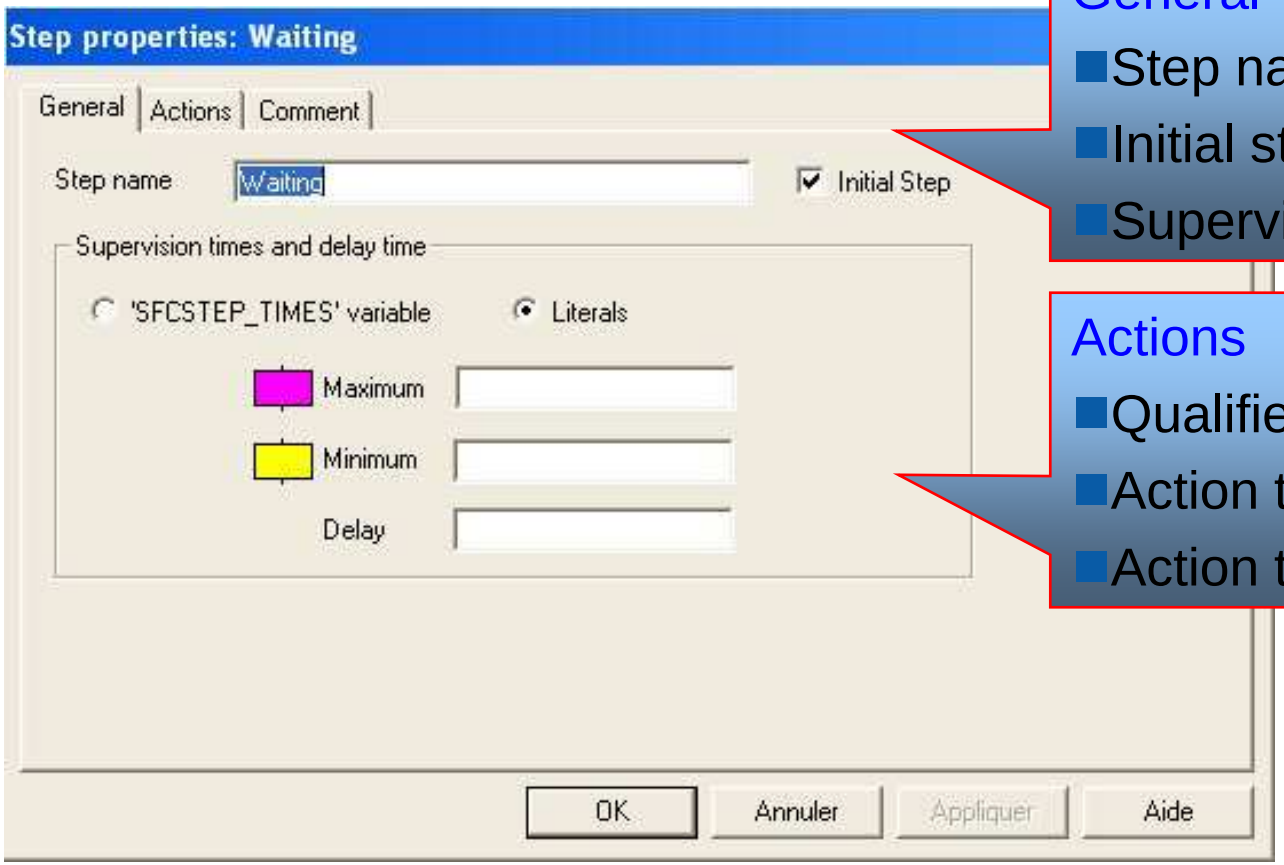
1024 steps maximum

Lines are not dedicated for steps or transitions

SFC specific toolbar



Step properties



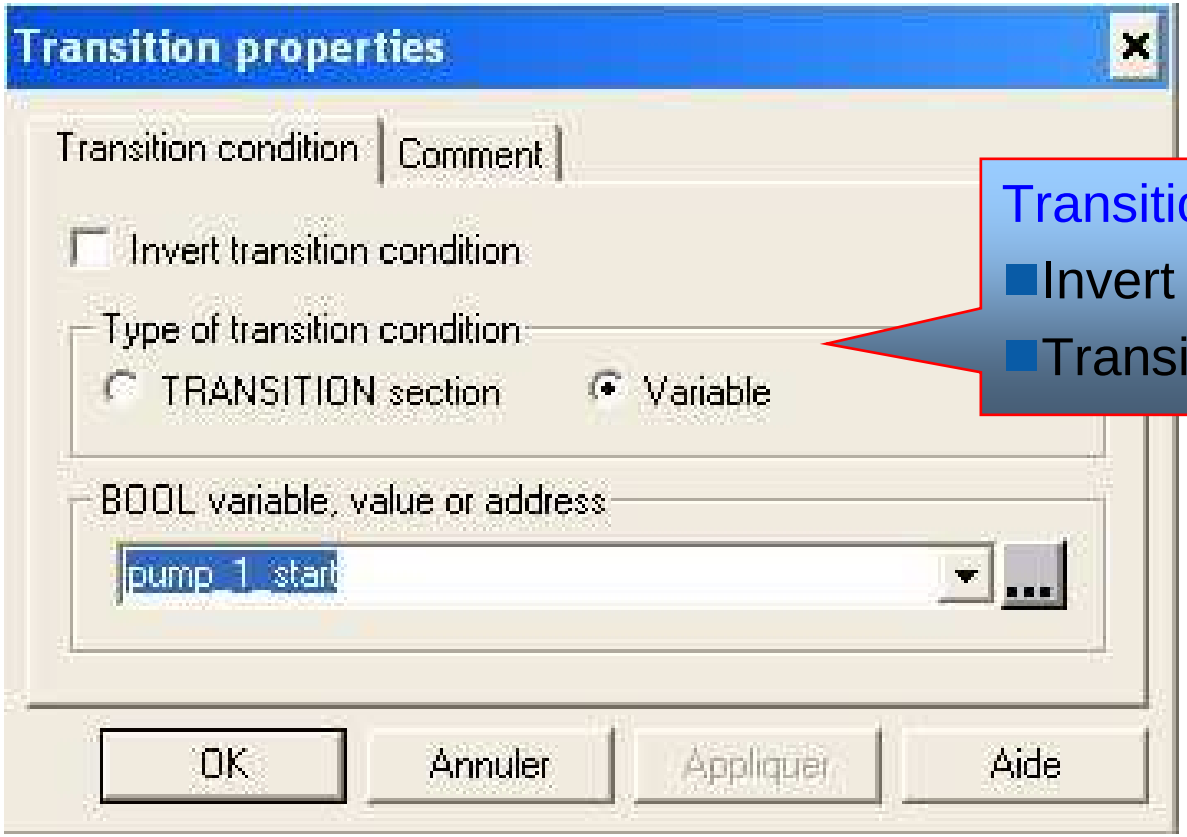
General

- Step name
- Initial step attribute
- Supervision and delay times

Actions

- Qualifier to control actions
- Action time
- Action type (variable or section)

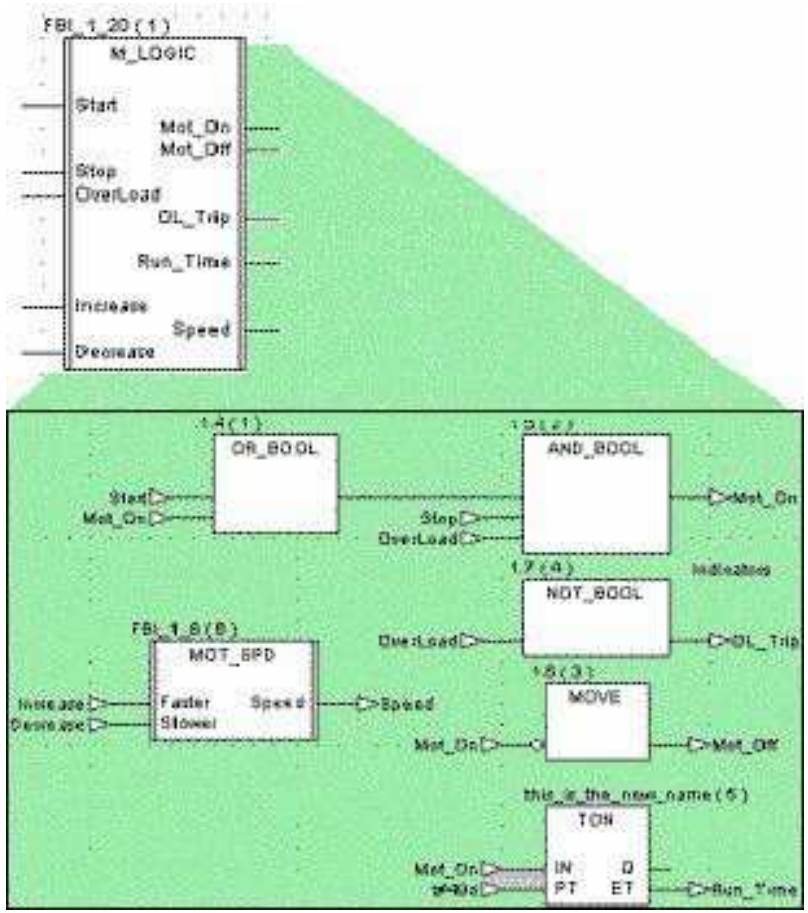
Transition properties



Transition condition

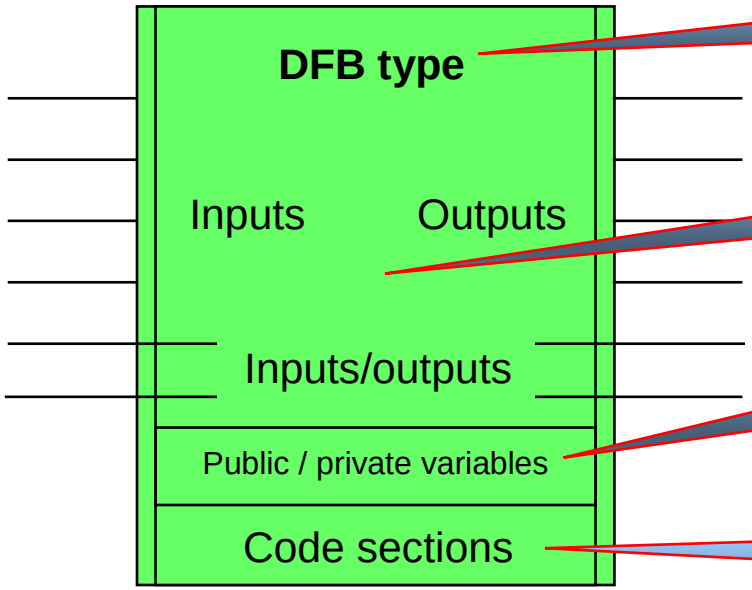
- Invert transition condition
- Transition type (variable or section)

DFB overview



- DFB is a user logic encapsulated in a reusable block
  - to structure the application, simplify the program creation, improve readability, make debugging easier and protect your know-how
- The designer programs and debugs the DFB function block model called “DFB Type”
  - DFB can be protected and exported / imported between designer and users
  - DFB is local to an application or global in library
- The end user creates an image of this block called “DFB Instance" and use it on the application
  - DFB code is loaded only once in the CPU

DFB structure



Name : 32 characters max

Inputs/outputs parameters

Public or private internal variables

One or several program sections

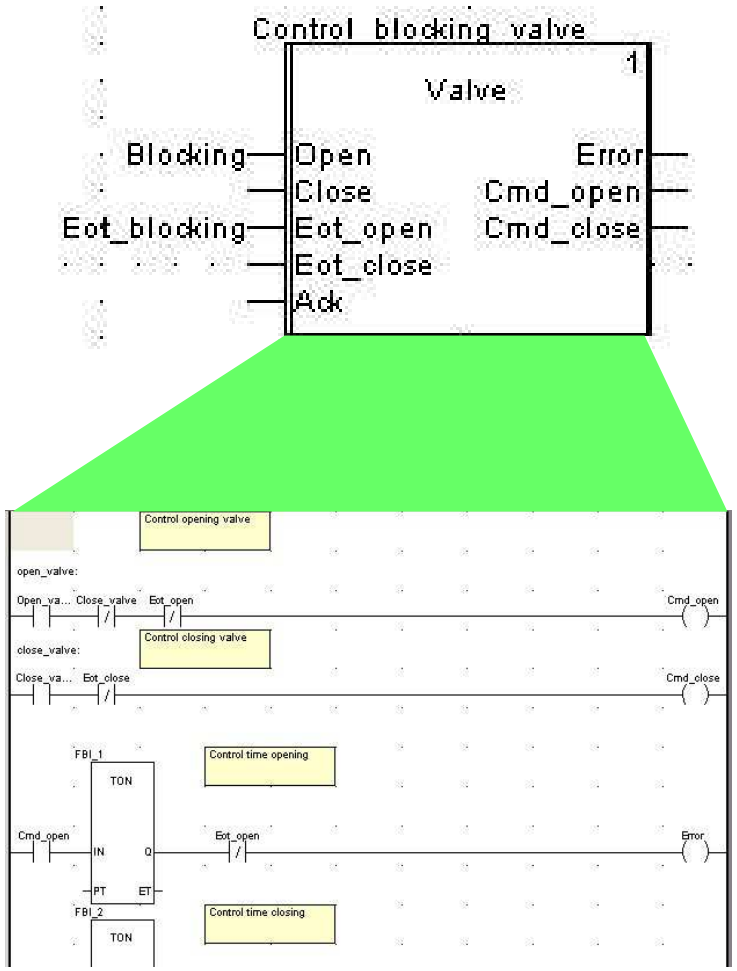
Comment : 1024 characters max

Descriptive file and protection level

I/O parameters and internal variables

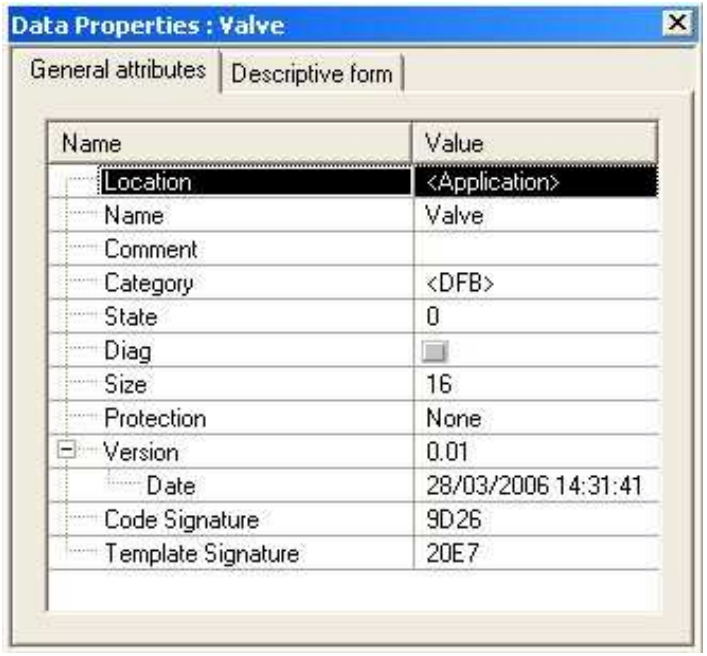
- Inputs : 32 max
  - In read only by the DFB
  - To transfer values from application to DFB
- Outputs : 32 max
  - In write only by the DFB
  - To transfer values from DFB to application
- Inputs / outputs : 32 max
  - In read / write by the DFB
  - To transfer values to DFB, modify and return them to application
- Private variables : unlimited
  - Only used by DFB (i.e. : intermediate variable)
- Public variables : unlimited
  - Used by DFB and application or user in adjust mode
  - Values modified by program or adjustment can be saved as initial values by setting %S94

DFB program sections



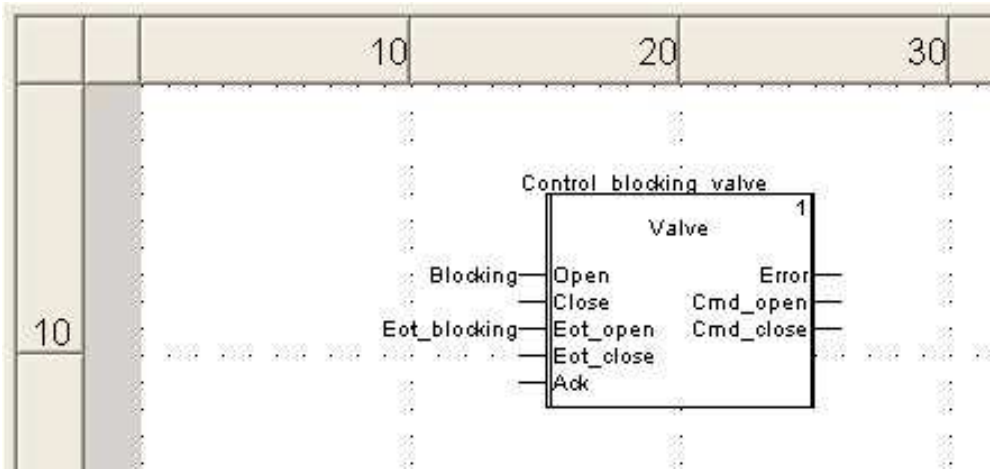
- Structured in program sections
  - IEC compliant : only one section
  - Not IEC compliant : several sections
- Section
  - Symbolic name
  - Written in LD, IL, ST, FBD
  - Validation condition
  - Protection (no, write, read / write)
  - Comment : 256 characters max
- Use only parameters and variables defined for function block or system bits and words

DFB properties



- General attributes
  - Name
  - Comment : 1024 characters max
  - Protection : None, read only, no read & write *(independent of application protection)*
  - Diag : to define a user diagnostic DFB
- Descriptive form
  - To edit descriptive file

Use DFB instance




```
if start then
Motor_2 (Open := Blocking(*BOOL*),
        Close := not blocking(*BOOL*),
        Eot_open := Eot_blocking(*BOOL*),
        Eot_close := eot_unblocking(*BOOL*),
        Ack := Acknowledge(*BOOL*),
        Error => Error(*BOOL*),
        Cmd_open => Cmd_open(*BOOL*),
        Cmd_close => Cmd_close(*BOOL*));
        end_if;
```

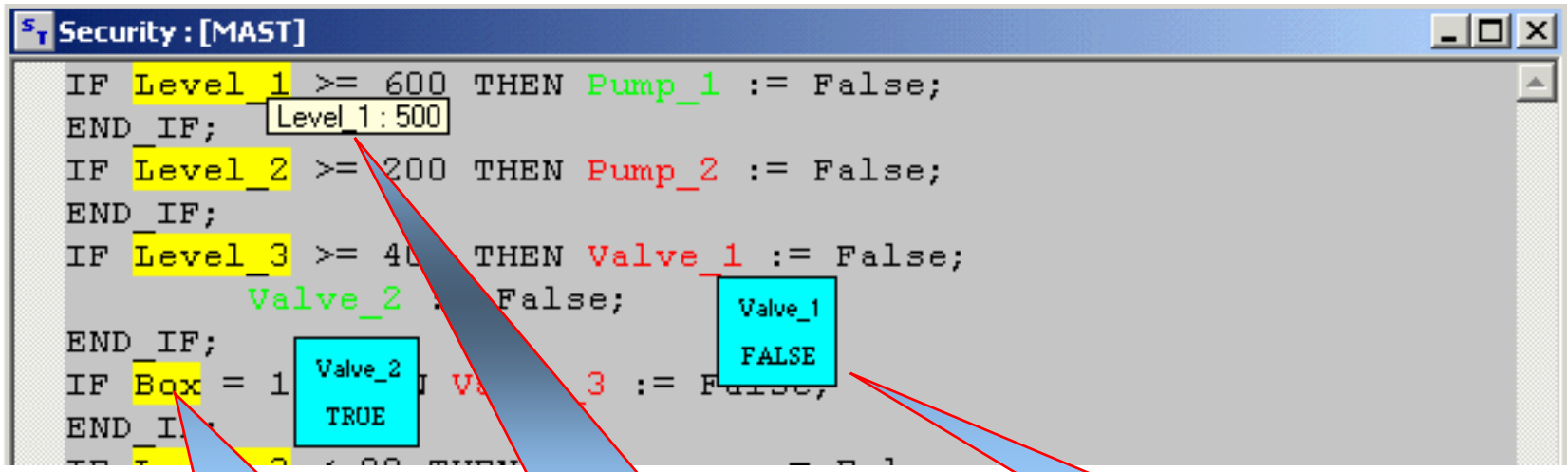
- DFB instance
  - Used in all application tasks except event tasks and SFC transitions
  - Connect inputs / outputs to application variables of same type
- DFB instance used as
  - A standard block in LD and FBD languages
  - An elementary function in ST or IL languages

Debugging the program

To shorten debugging and commissioning time :

- Dynamic animation of the program
- Inserting a breakpoint in the program and executing it step by step
- Inserting a watch point in the program
- Using display boxes attached to variables
- Using animation tables or operator screens to drive inputs / outputs
- Using cross reference to see usage of a variable

Dynamic animation and display boxes



Dynamic animation

Boolean and analog variables are animated with colors (green, red or yellow)

Tool tip displays value when cursor is flying above a variable

Inspect window displays value of a variable. Color of the window is defined by comparison of value and thresholds (yellow, blue, magenta)

Breakpoint and watch point

Breakpoint to stop execution of program and inspect code behavior

Security : [MAST]

```
IF Level_1 >= 600 THEN Pump_1 := False;
END_IF;
IF Level_2 >= 200 THEN Pump_2 := False;
END_IF;
IF Level_3 >= 400 THEN Valve_1 := False;
        Valve_2 := False;
END_IF;
```

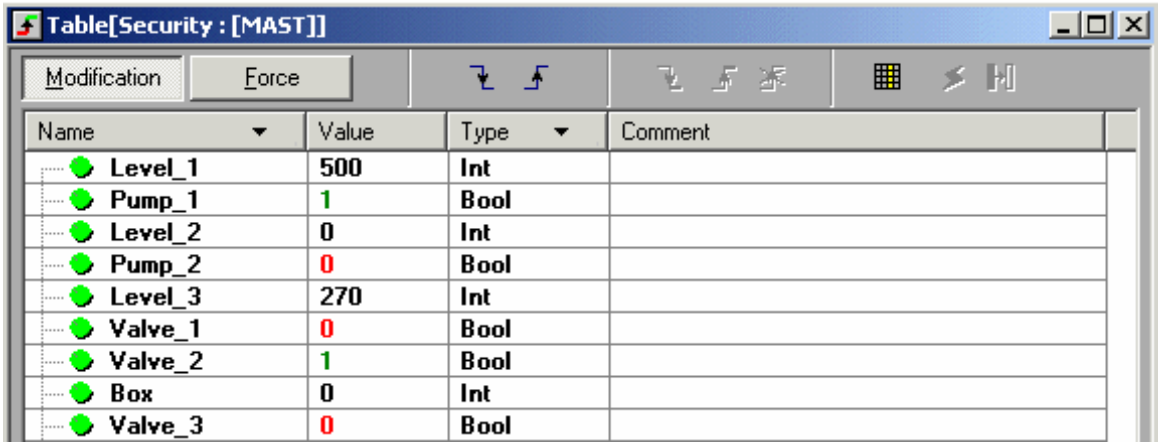
Watch point to inspect variables exactly when a program line is executed

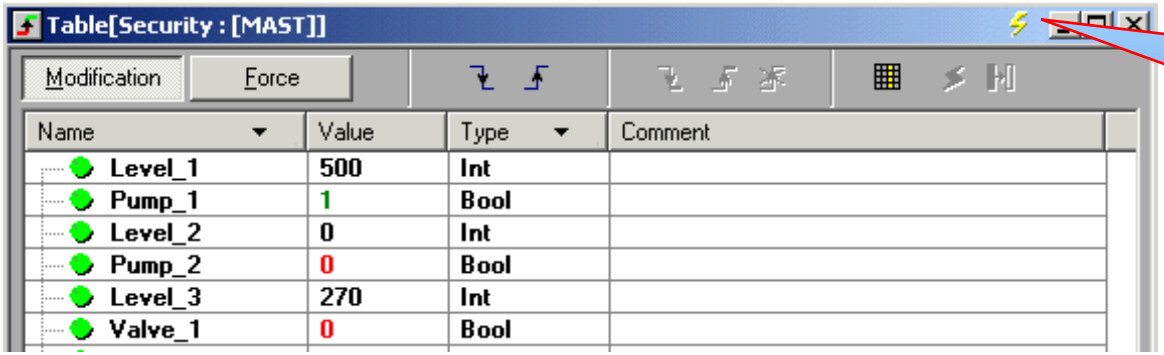
Security : [MAST]

```
IF Level_1 >= 600 THEN Pump_1 := False;
END_IF;
IF Level_2 >= 200 THEN Pump_2 := False;
END_IF;
IF Level_3 >= 400 THEN Valve_1 := False;
        Valve_2 := False;
END_IF;
```

Animation table

Animation table to modify the value of variables and to force Boolean variables

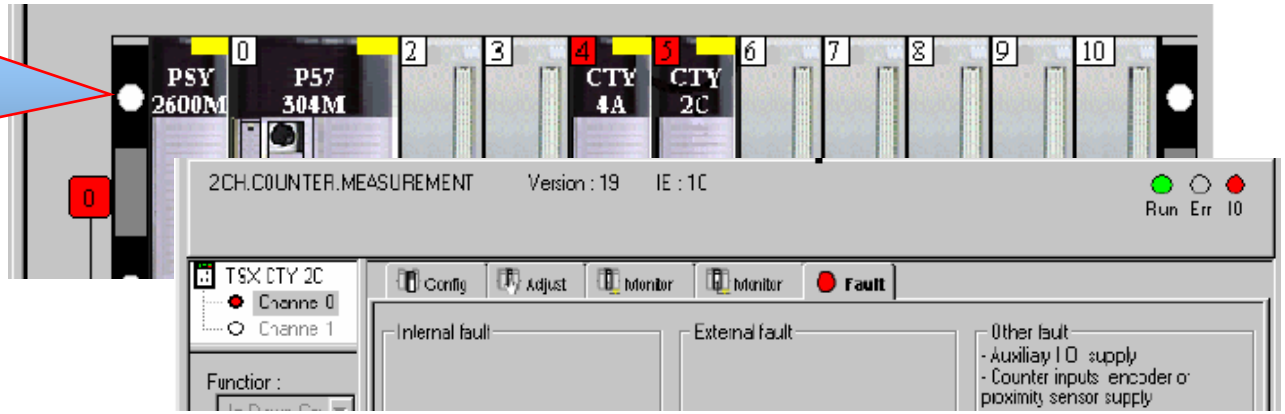




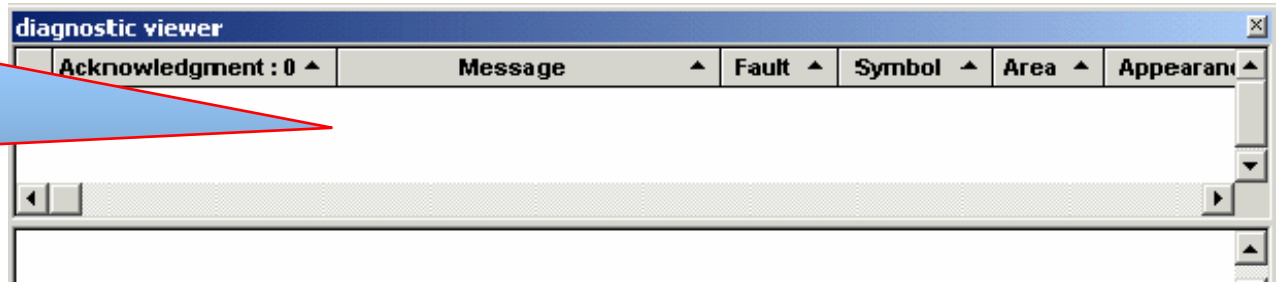
Animation table can be synchronize on watch point

Diagnostic in run time

Module fault reporting in configuration screen of the rack



Diagnostic viewer to display system and application defaults from the diagnostic buffer of the PLC



Operator screens overview

- Control and monitoring for first and second level diagnostic
  - Displaying real time state of a machine / process
  - Run time screens customized to needs of operator

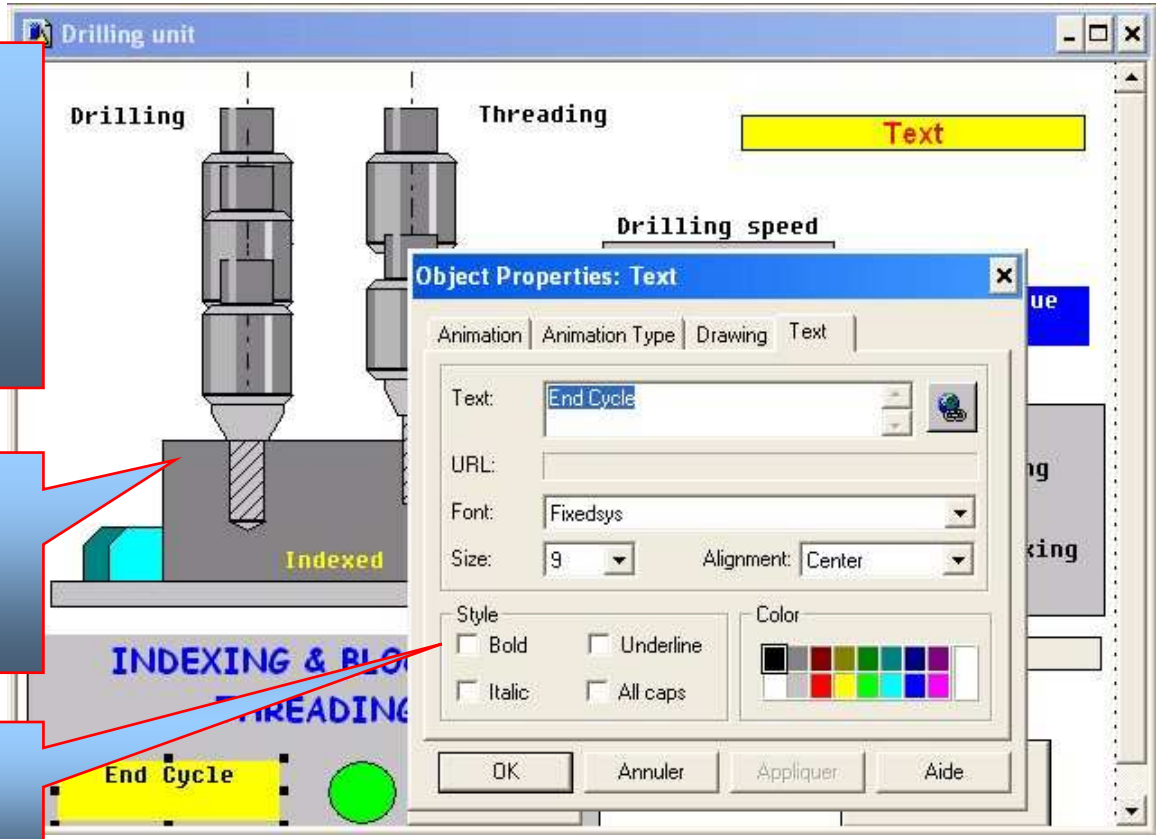
- Completely integrated in Unity Pro
  - Use same browser, data editor, cross reference tool, …
  - Use existing application resources
  - Run time screens are part of PLC application file
  - No need for additional hardware like communication module
  - Easy to create screens : toolbar, predefined objects, library of objects, configuration dialog boxes, …
  - Easy to create animations : visibility, flashing, bar-graph, trend diagrams, predefined animated objects, …

Graphic editor and toolbar

Specific toolbar to create objects (rectangle, curve, button, cursor, …), insert images and edit objects (group, align, …)

Graphic editor to create screen using toolbar and library objects

Dialog box to configure objects and animations



Utilities

- Import / export in XML format
  - Accessible from structural and functional views
  - Export type is determined by extension file (XEF for global application, XLD for Ladder section, XDB for DFB type, …)
  - Management of conflicts during import (wizard)
- Documentation of the application
  - Build and print the documentation of the project
  - Define title page and general information page
  - Define footer of the pages
  - Define documentation content (include headings)
- Hyperlink to external documents
  - In project browser or from a comment filed
- Converters to transform legacy application (PL7, Concept, …) into Unity Pro application