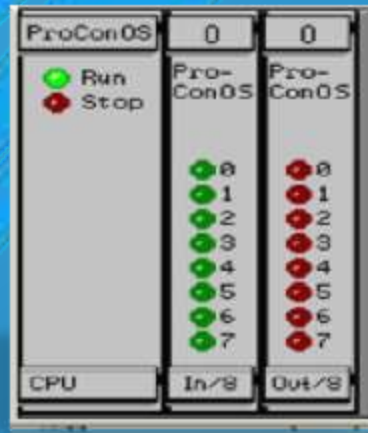




IEC 61131-3 Basics with MotionWorks IEC



Class Number: TRM011-MotionWorksIEC-Basic

Doc ID: eLV.MotionWorksIEC.01.Basic

Rev 1.10

Date: March 6, 2014



- *Everything but motion*
 - *In this class we won't have servos, axes, or make anything move*
 - *You'll manipulate bits, values, and time*
 - *You'll become proficient in the programming and navigation of MotionWorks IEC*
 - *You'll experience the IEC 61131-3 languages, data types, and programming structure*
 - *When you see the IEC logo, this means the page is directly related to IEC 61131-3*

Software Orientation

Mpiec Controllers vs PLC simulator
Mwiec versions, install, registration tips
Mwiec main screen, simple program editing

- *Download page*
 - *yaskawa.com/iecs*
 - *Installation instructions*
- *Windows XP Requirements*
 - *Service Pack 3*
 - *Dot Net 3.5*
- *Windows 7*
 - *Multi-Language Support*

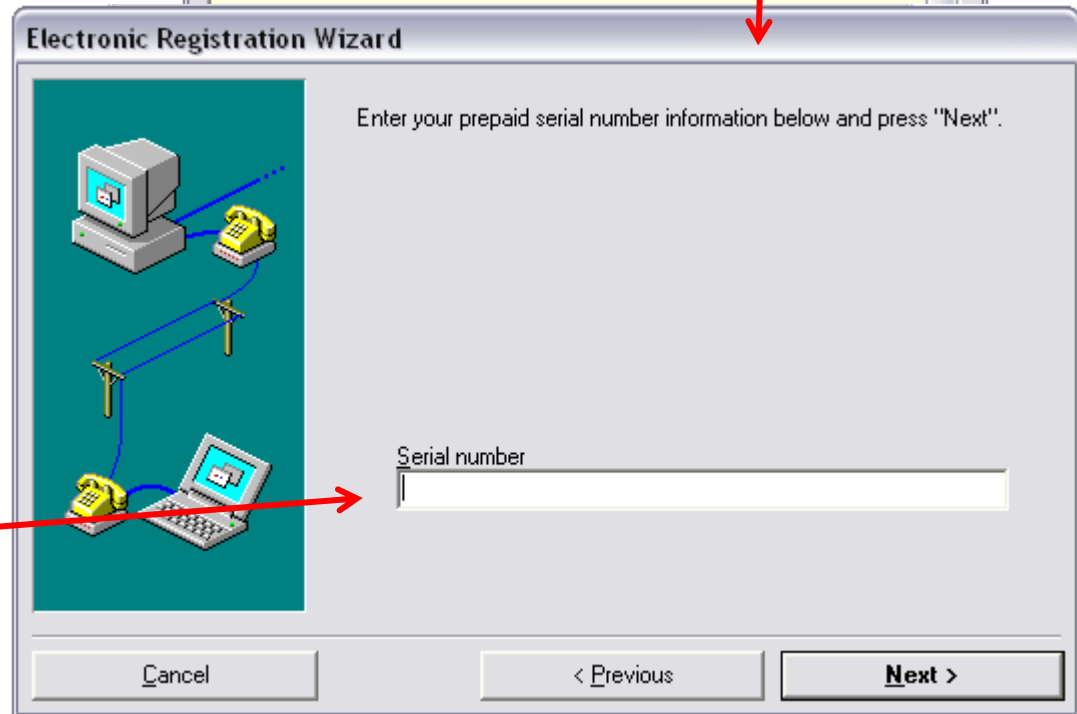
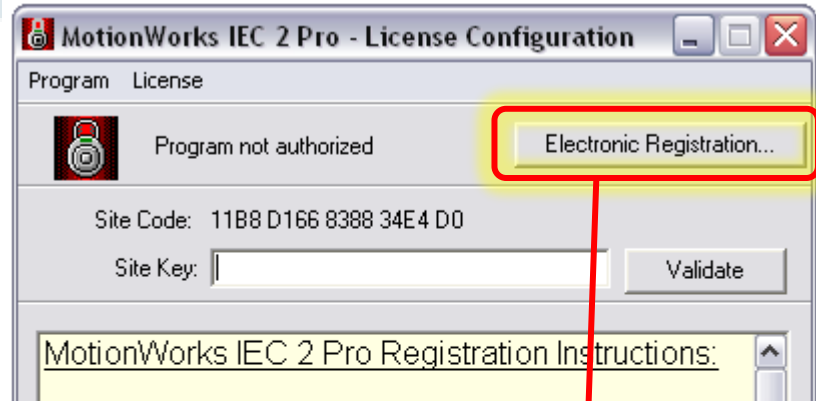


- *30-day Demo*
 - *Do not change windows clock*
- *Unlimited License*
 - *\$\$ Cost*
 - *Serial Number*
 - *Internet Connection*
 - *USB license option*

Sales Order Email contains Serial Number



See Video: <http://youtu.be/yGC2TmtP1bU?t=22s>



- *MotionWorks IEC “Express”*

- *Simplified Interface*
- *1 Execution Task*
- *Fundamental programming languages (ST, LD)*
- *Easy to use*
- *Easy to learn*
- *Low cost*
- *For simple applications requiring point-to-point, camming, gearing, ethernet*



- *MotionWorks IEC “Pro”*

- *Full options interface*
- *Multi-tasking operation possible*
- *All five IEC 61131-3 languages available (ST, LD, FBD, IL, SFC)*
- *Full featured*
- *Medium cost*
- *For the most demanding and high-performance applications*



- *Both Express and Pro can be simultaneously installed in one PC*
 - *Separate license required for each*
- *Express projects can be opened in Pro*
 - *Express Project is converted to Pro format*
 - » *Save project under a new name or as zip project first*
 - *Pro projects cannot be opened with Express*
- *Code can be copied and pasted between Pro and Express*
 - *Multiple Express / Pro open at the same time*
 - *Easiest to copy the entire POU*

Ethernet Communication via PC



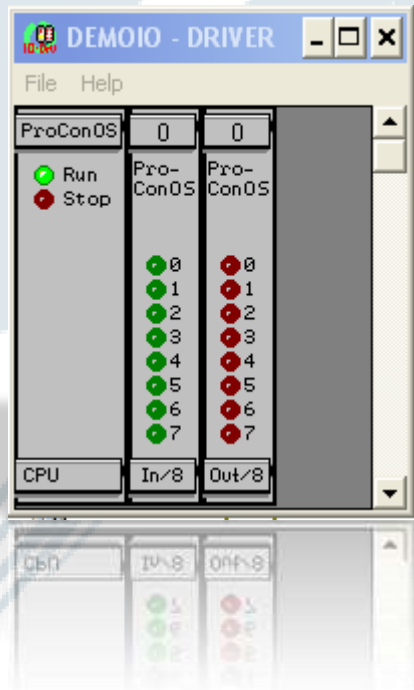
Web Server

- Confirm Ethernet communication
- Determine/Set IP address
- Troubleshoot and clear alarms
- Save/Restore Project Archive
- Test Motion
- Upgrade Firmware
- Requires Java

MotionWorks IEC

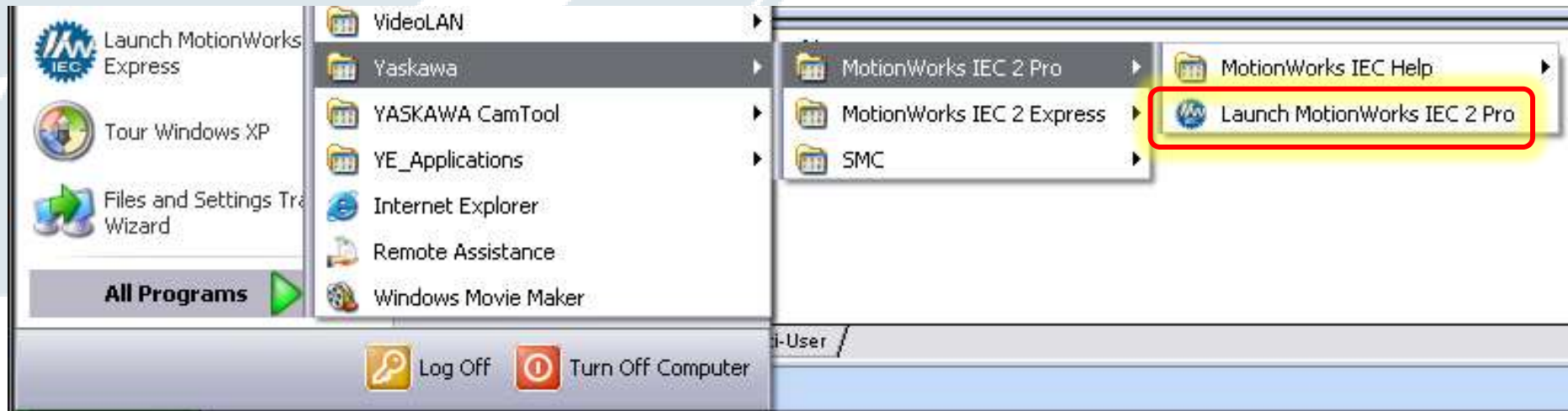
- Monitor and Edit Application Program
- Configure axes, IO, Network
- Troubleshoot and clear alarms
- Save/Restore Project Archive
- Test Motion





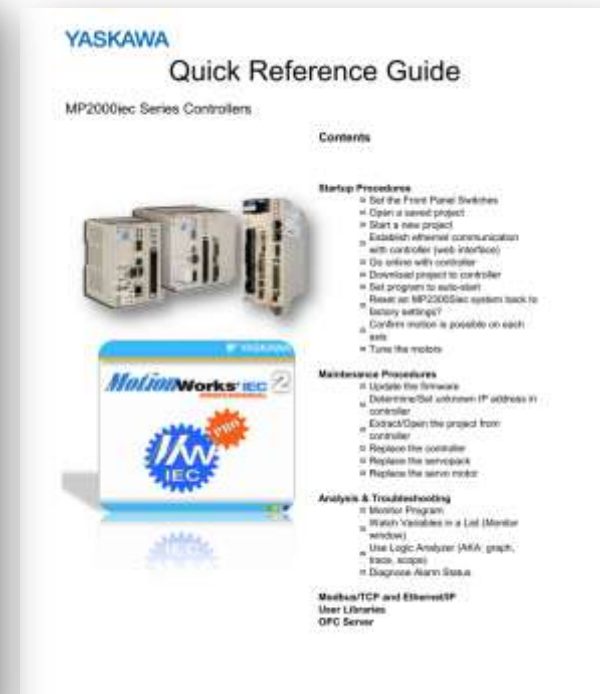
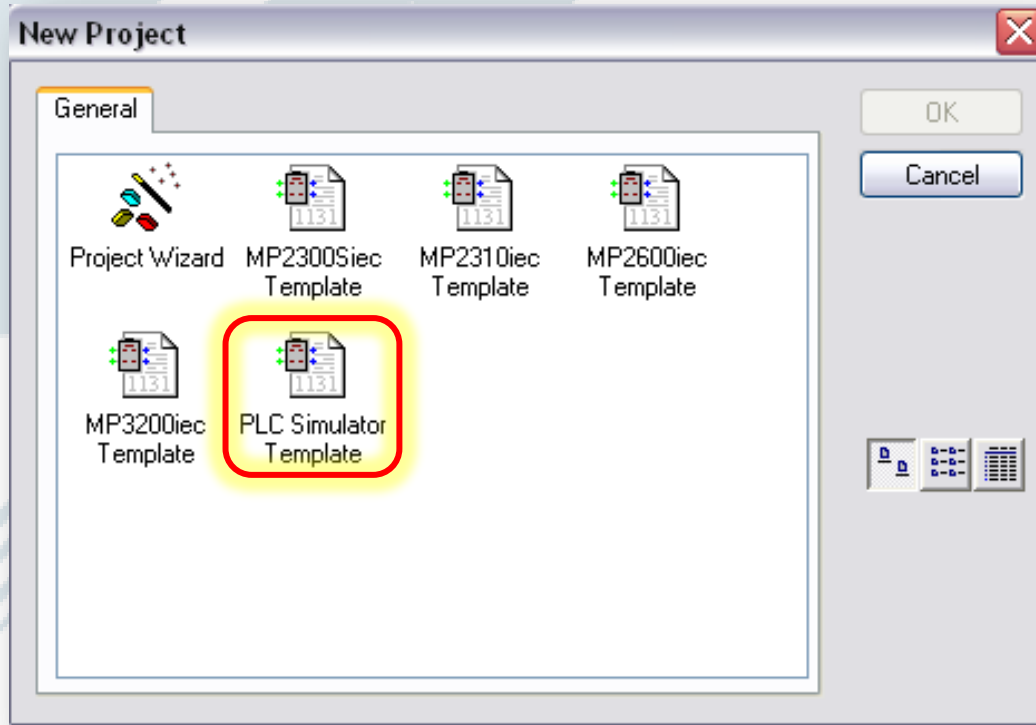
MotionWorks IEC

- Monitor and Edit Application Program
 - LD, ST, FBD, SFC languages
 - Tasks, Programs, Functions
 - I/O



Start a new project

- PLC Simulator Template
- File - Save Project As



•Enlarge programming space by toggling windows on/off
•Some windows only available when Programming Space is active
•Windows can be undocked

Project Tree Window

- Project : G:\Yaskawa\MWIEC_2_Pro\Projects\T
- Libraries
- Data Types
 - PLCTaskInfoTypes*
- Logical POU's
 - Main*
 - MainT
 - MainV*
 - Main*
- Physical Hardware*
- Configuration : PLC_Simulator*
- Resource : PLC_Simulator*
- Tasks
 - Task : CYCLIC
 - Main : Main*
- Global_Variables*
- ID_Configuration*

Edit Wizard

Group: All

Name	Description
Firmware FB	
Firmware functions	
Keyword	
Library FB	
Library function	
Network Templates	
User FB	
User function	

Message Window

Build | Errors | Warnings | Infos | PLC Errors | Print | Multi-User

For Help, press F1

G: >2GB



The screenshot displays the MotionWorks IEC 2 Pro software interface. The main window is titled "MotionWorks IEC 2 Pro - Software Orientation - [MAIN:Main]". It features a menu bar (File, Edit, View, Project, Build, Objects, Layout, Online, Extras, Window), a toolbar, and a Project Tree Window on the left. The Project Tree Window shows a project structure with folders for Libraries, Data Types, Logical POU's, Physical Hardware, Configuration, Resource, Tasks, and Global Variables. The Logical POU's folder is expanded, showing a "Main*" folder which contains three sub-items: "MainT", "MainV", and "Main". These three items are highlighted with a red box, and red arrows point from them to the text "Logical POU's" in the Programming Space. The Programming Space is a large white area with a blue header "Programming Space" and a list of items: "Logical POU's", "T = text comments", "V = variables", and "Code Pages". The Edit Wizard window on the right shows a list of functions with their descriptions. The status bar at the bottom indicates "MAIN:Main" and "2,0 G: >2GB".

Project Tree Window

- Project : G:\Yaskawa\MWIEC_2_Pro\Projects\T
 - Libraries
 - Data Types
 - PLC Task and Types
 - Logical POU's
 - Main*
 - MainT
 - MainV
 - Main
 - Physical Hardware
 - Configuration : PLC_Simulator*
 - Resource : PLC_Simulator*
 - Tasks
 - Task : CYCLIC
 - Main : Main*
 - Global Variables*
 - IO_Configuration*

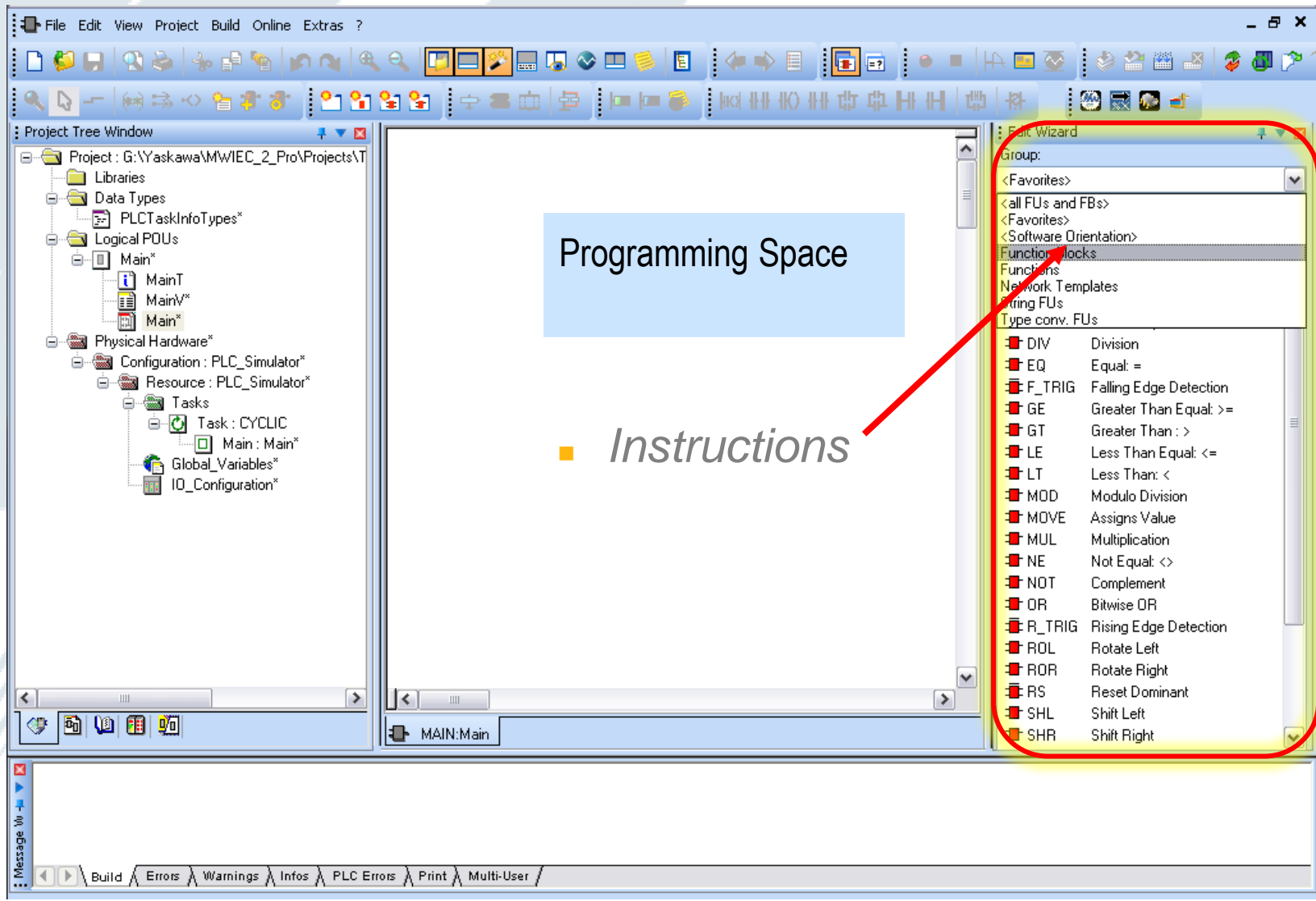
Programming Space

- Logical POU's
 - » T = text comments
 - » V = variables
 - » Code Pages

Edit Wizard

Group: <Favorites>

Name	Description
ADD	Addition
AND	Bitwise AND
CTD	Counter Down
CTU	Counter Up
CTUD	Counter Up/Down
DIV	Division
EQ	Equal: =
F_TRIG	Falling Edge Detection
GE	Greater Than Equal: >=
GT	Greater Than: >
LE	Less Than Equal: <=
LT	Less Than: <
MOD	Modulo Division
MOVE	Assigns Value
MUL	Multiplication
NE	Not Equal: <>
NOT	Complement
OR	Bitwise OR
R_TRIG	Rising Edge Detection
ROL	Rotate Left
ROR	Rotate Right
RS	Reset Dominant
SHL	Shift Left
SHR	Shift Right



The screenshot shows a software interface for editing ladder logic. It features several functional blocks:

- ADD** block: Inputs 'value1' and 'value2', output 'ENO'. A question mark is in the center.
- CTU_1** block: Inputs 'CU', 'RESET', 'PV'. Outputs 'Q', 'CV'. A red arrow points from the 'Edit Wizard' to the 'PV' input.
- F_TRIG_1** block: Input 'CLK', output 'Q'.
- MOVE** block: Input from 'F_TRIG_1', output 'ENO'. A question mark is in the center.

Annotations and panels:

- Add variables**: A blue box pointing to the 'value1' and 'value2' inputs of the ADD block.
- Drag from Edit Wizard**: A blue box pointing to the 'PV' input of the CTU_1 block.
- Add blocks by typing the name**: A blue box pointing to a search box containing 'F_TRIG'.
- Edit Wizard**: A panel on the right with a list of functions:

Name	Description
ADD	Addition
AND	Bitwise AND
CTD	Counter Down
CTU	Counter Up
CTUD	Counter Up/Down
DIV	Division
EQ	Equal: =
GE	Greater Than Equal: >=
GT	Greater Than: >
LE	Less Than Equal: <=
LT	Less Than: <
- Error**: A dialog box with a red 'X' icon and the text: "Execution number and highlight feedback may be wrong until next successful compilation!". An 'OK' button is highlighted.

Programming Quick Start

Function Block

Make, Download

Debug Mode

Ladder

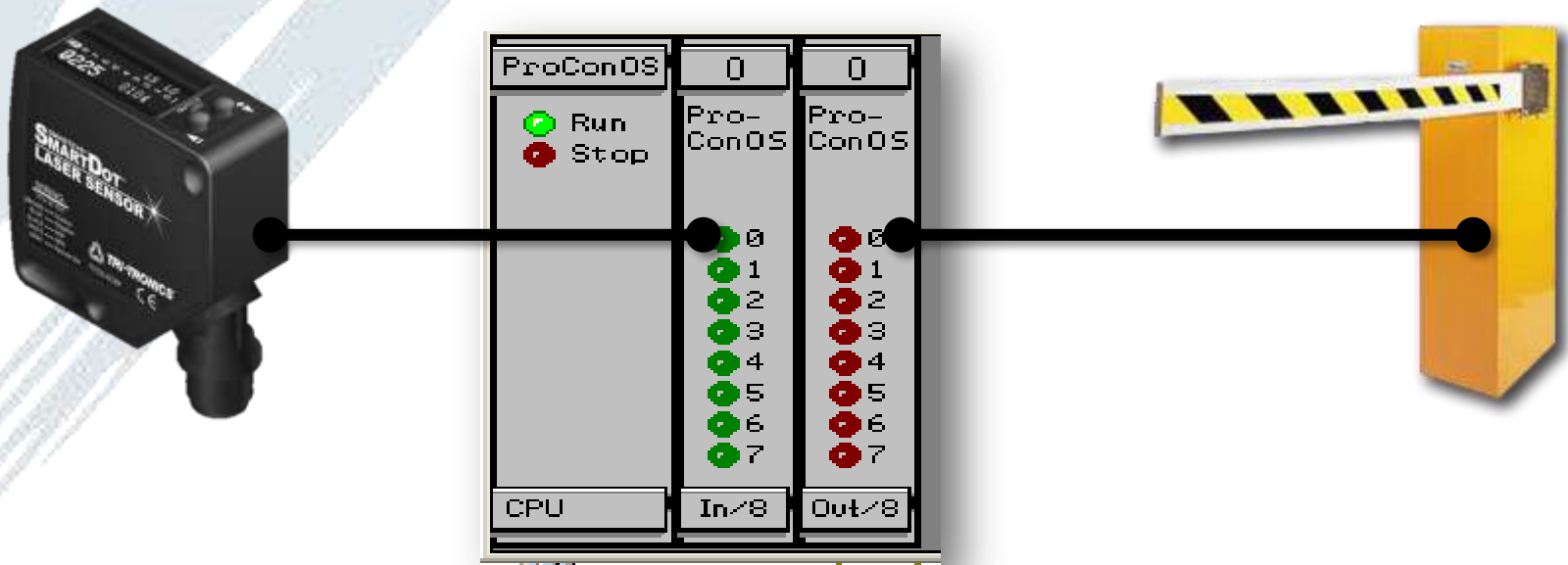
Variables and Properties

Variables and Properties

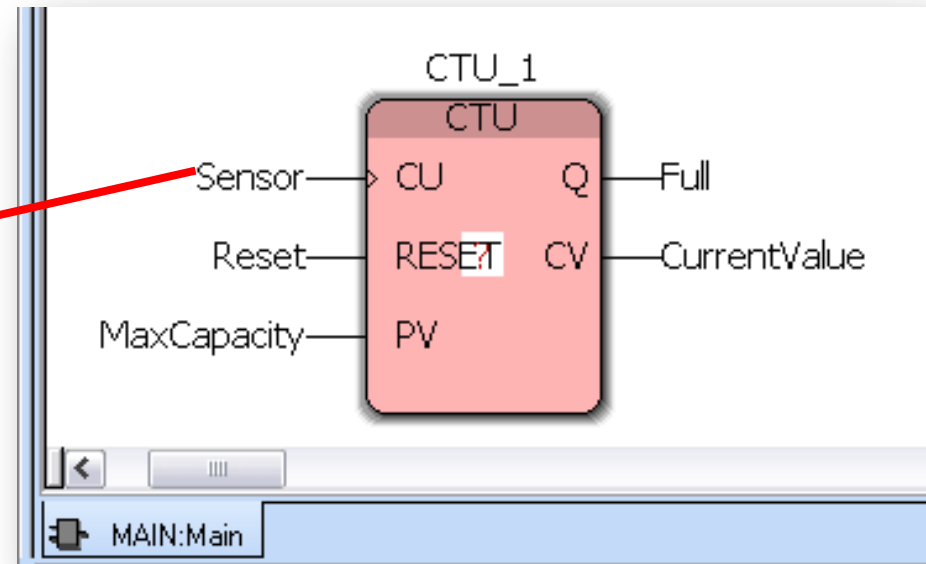
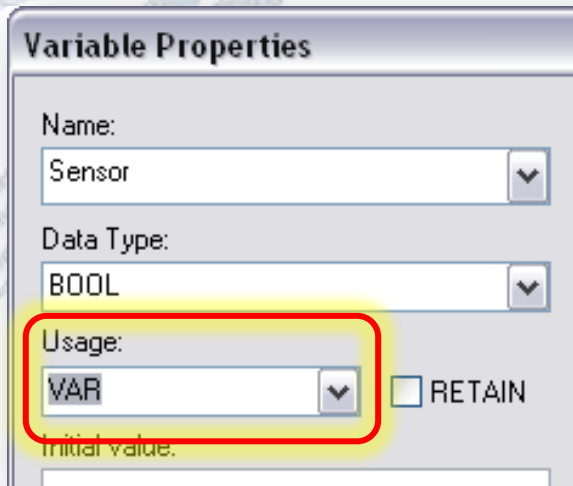
Ladder

Debug Mode

- *Learn the software by means of a simple program*
 - *Parking gate control*
 - *Many details to follow*



- *New Project – PLC Simulator Template*
- *POU “Main”*
 - *Counter UP block*
 - *Add variables (Usage: VAR)*
 - *Make, download, warm start*
 - *Debug Mode*
 - » *Enter values*

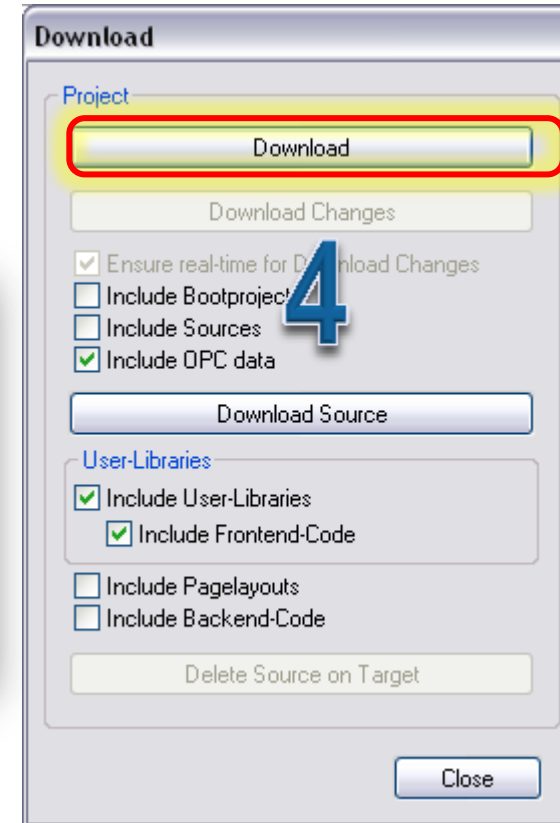
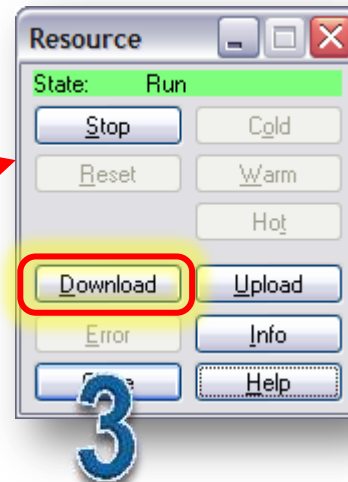
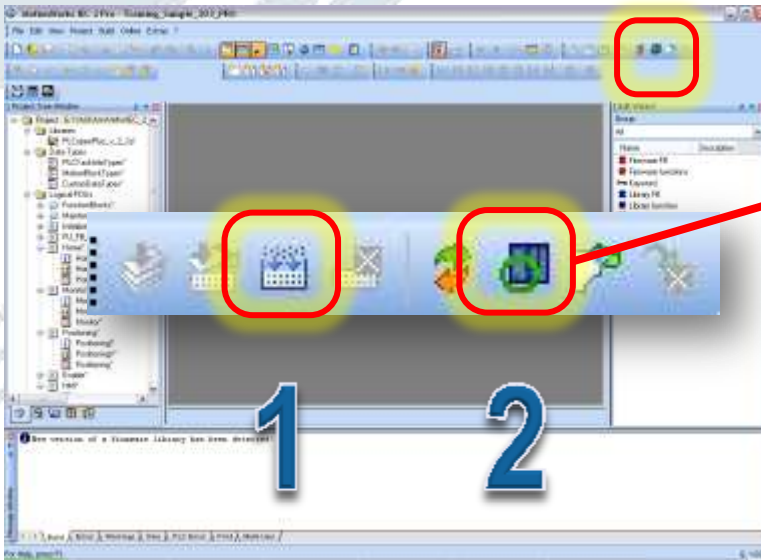


Add this code to the Main POU

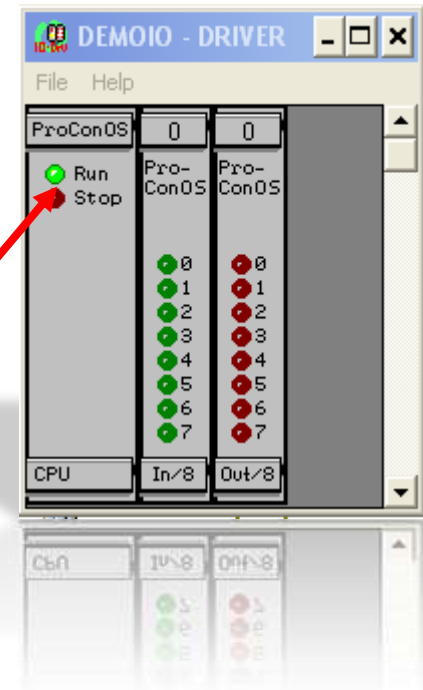
Steps to Download

1. Make (includes automatic "save")
2. Project Control → Resource
3. Download
4. Download

More details to come

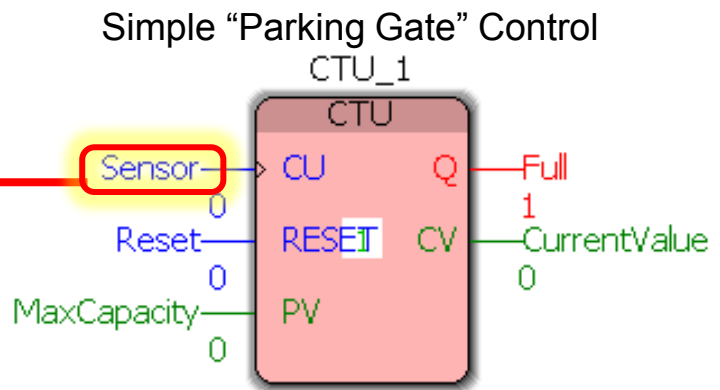
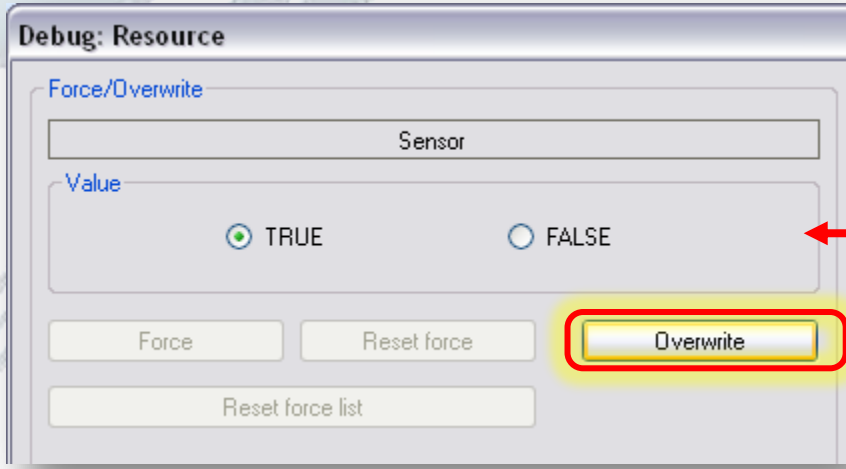
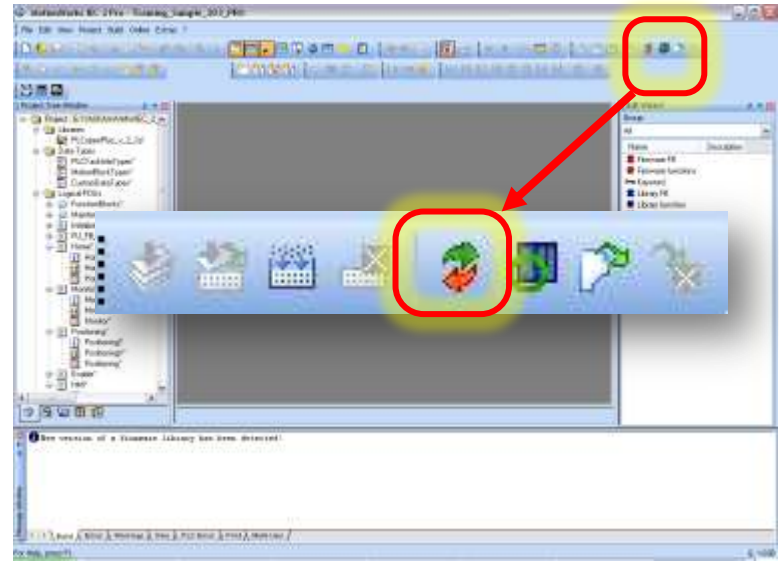


- *Warm Start*
 - *Normal "Power On" start*



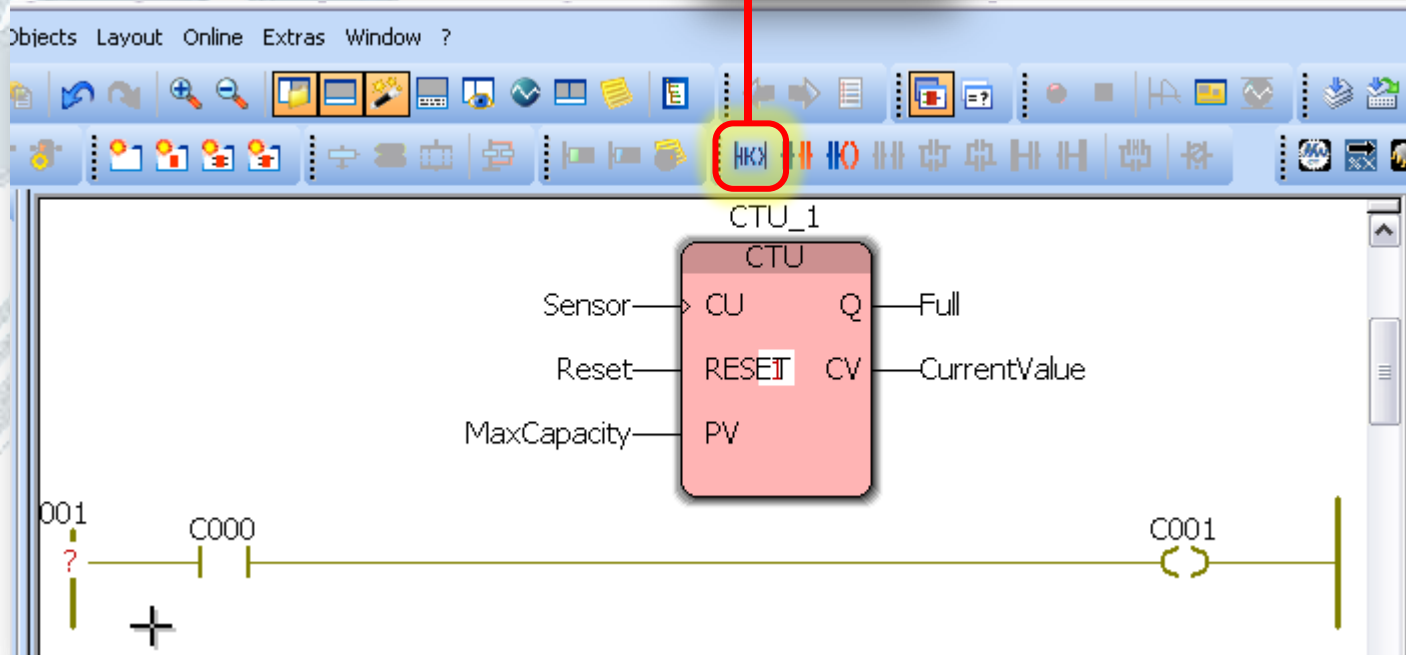
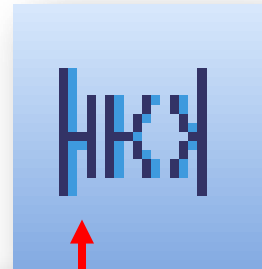
More details to come

- *Online – Debug*
- *Double click variables*
- *Overwrite*
- *Editing is disabled during debug*



Debug Mode (Disables Editing)

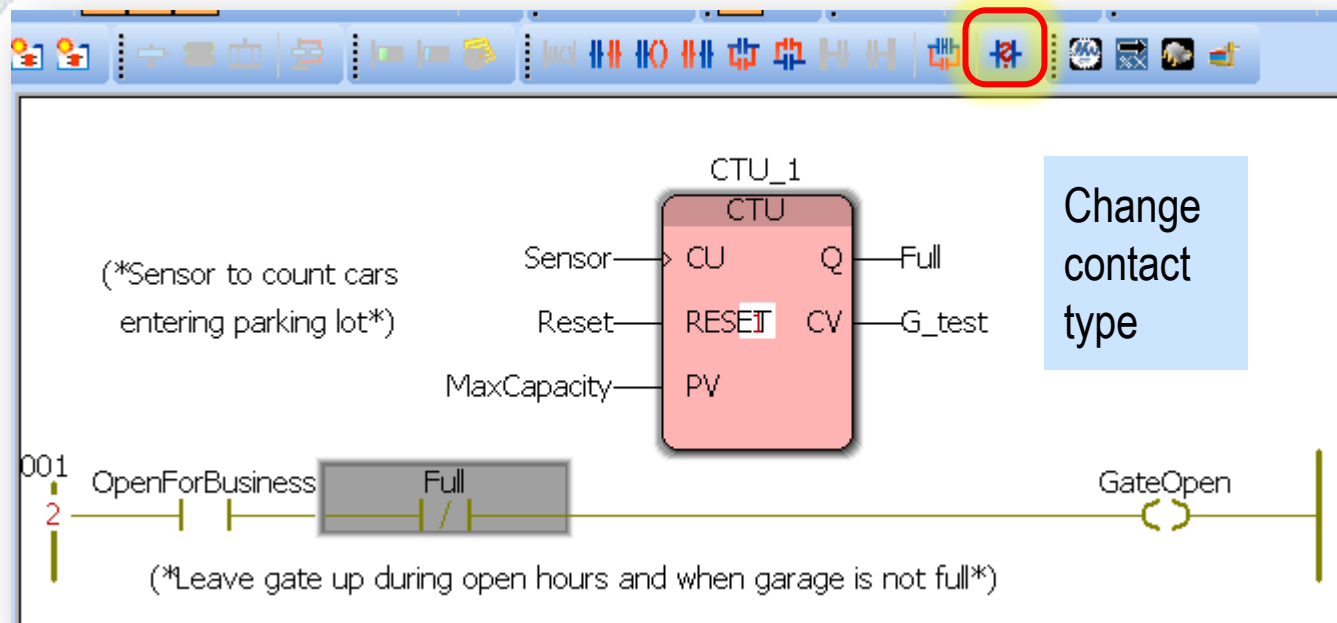
- *Simple Ladder*
 - *Network*
 - *Series contact, NC*
 - *Comments*
 - *Download changes*
 - *Debug mode*



- *Download Changes*
 - *Saves time*
 - *Includes*
 - » *Save*
 - » *Make*
 - » *Download*
 - *Does not require warm start*



Right-Click to
add comment





- *Variable Properties*
 - *Toggle Boolean Mode*
 - *Retain*
 - *Initial Value*

Initial Value

Retain last value after warm start

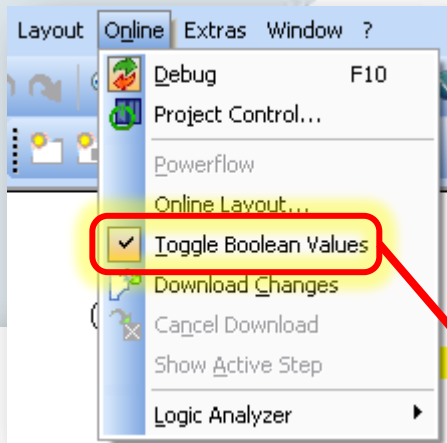
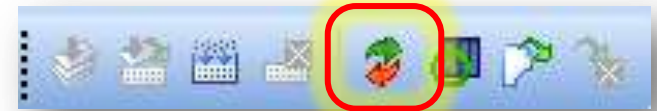
“Toggle Boolean” is a feature of debug mode

Project Tree Window

- Project : G:\Yaskawa\MWIEC
 - Libraries
 - Data Types
 - PLCTaskInfoTypes
 - Logical POU's
 - Main
 - MainT
 - MainV**
 - Main
 - Physical Hardware

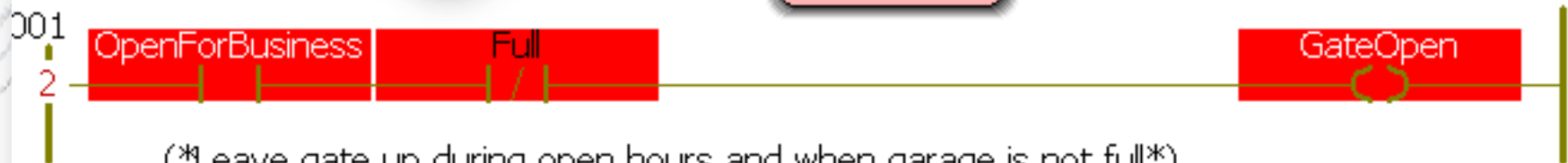
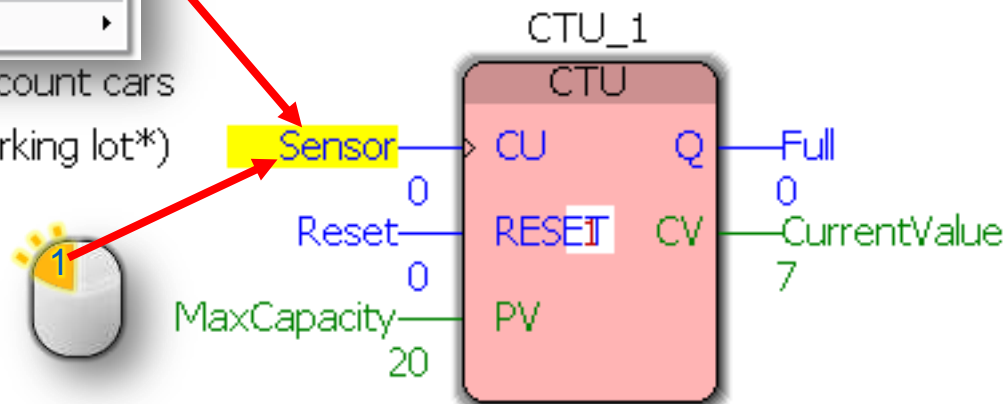
Name	Type	Usage	Description	Address	Init	Retain	PDD	OPC	TB
Default									
CTU_1	CTU	VAR				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
CurrentValue	INT	VAR				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Full	BOOL	VAR				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
GateOpen	BOOL	VAR				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
MaxCapacity	INT	VAR			20	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
OpenForBusiness	BOOL	VAR				<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Reset	BOOL	VAR				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Sensor	BOOL	VAR				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

- Toggle Boolean



Toggle Boolean – variable in debug mode is like a push-button

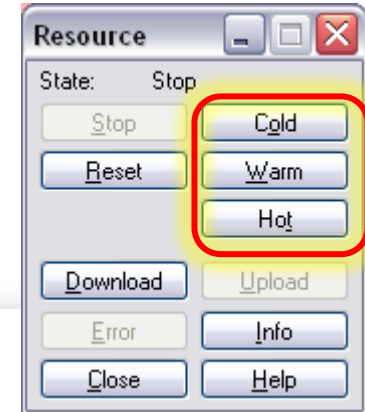
(*Sensor to count cars entering parking lot*)



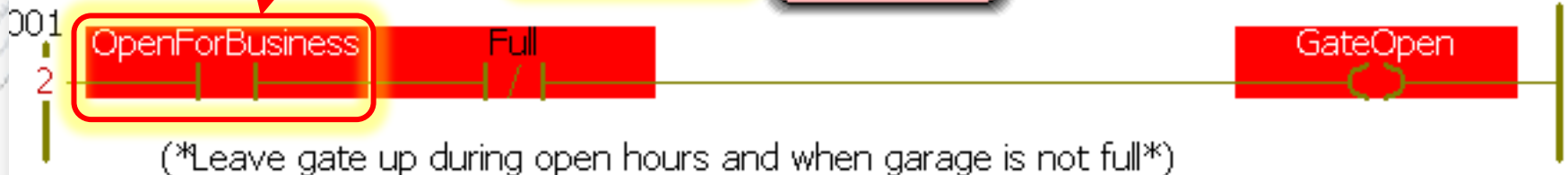
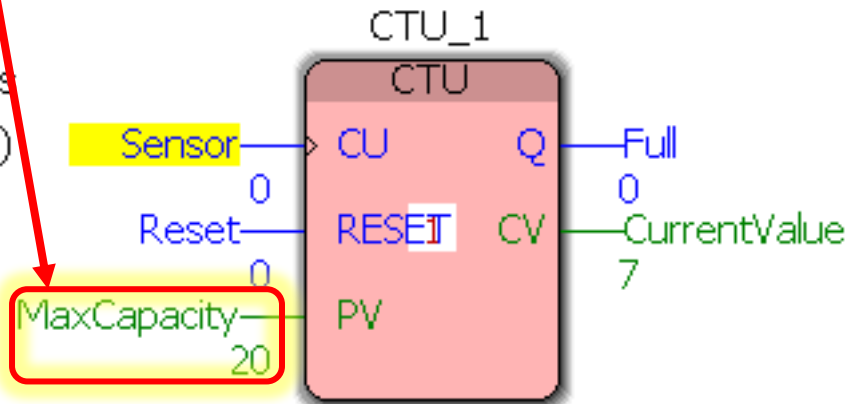
(*Leave gate up during open hours and when garage is not full*)

- Initial Value

Change the value in debug mode.
What happens after Warm start?
Cold start?



(*Sensor to count cars entering parking lot*)



(*Leave gate up during open hours and when garage is not full*)

Programming Quick Start #2

Task, Task Instance

Structured Text

F5 variable declaration

VAR & VAR_GLOBAL

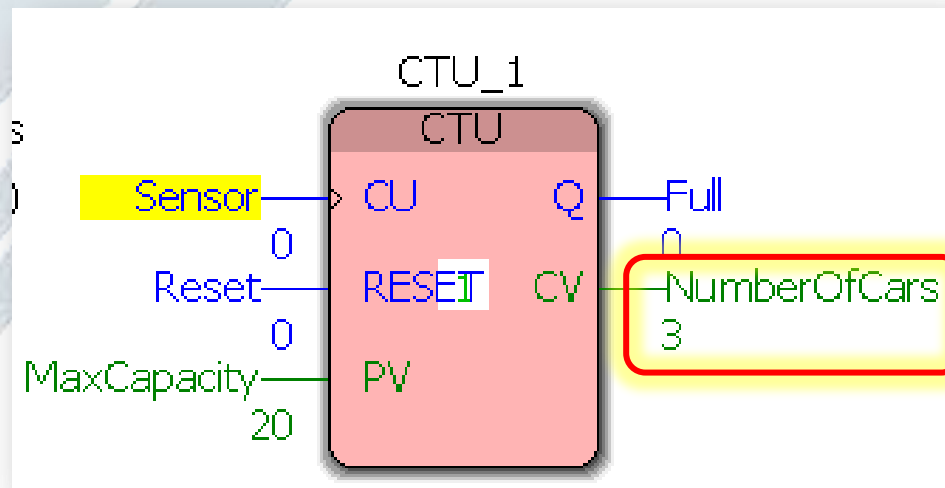
Data Type & Conversion

Data Type & Conversion

VAR & VAR_GLOBAL

F5 variable declaration

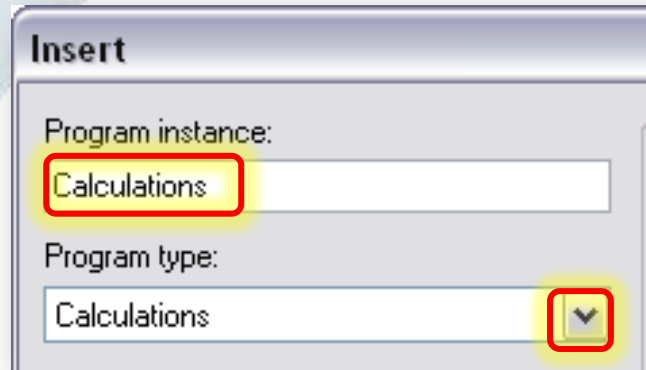
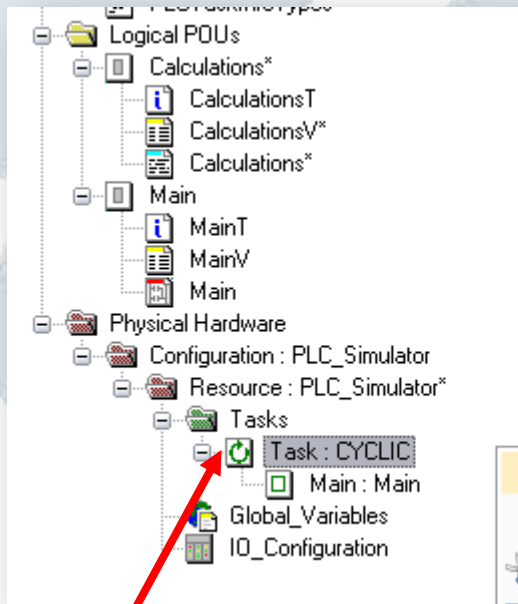
- *Calculations in a Structured Text program (ST POU)*
 - \$29.99 per car
 - » $Income = 29.99 * NumberOfCars$
 - *Ladder or Function block language also works*





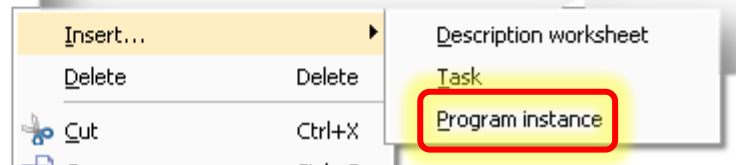
The screenshot illustrates the steps to create a new ST program in the IEC software. The 'Project Tree Window' on the left shows the project structure, with the 'Logical POU' folder selected. A context menu is open over this folder, and the 'Insert...' option is chosen. This opens the 'Insert' dialog box. In the 'Name' field, 'Calculations' is entered. Under the 'Type' section, 'Program' is selected. Under the 'Language' section, 'ST' is selected. A red arrow points from a mouse cursor to the 'Logical POU' folder in the tree.

- *Program POU runs in a Task*
 - *Manually add program instance*



Enter "Program instance"

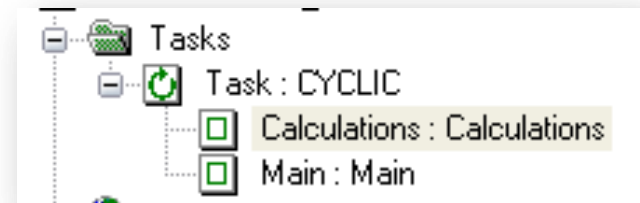
Choose "Program Type"



"Calculations" is not running in the task.

Insert a program instance.

Result



- Use *F5* to declare variables

```
1 Income := NumberOfCars * LREAL#29.99;
```



Variable Properties

Name: Income

Data Type: LREAL

Usage: VAR RETAIN

Variable Properties

Name: NumberOfCars

Data Type: LREAL

Usage: VAR RETAIN

Format for "literal" (constant) values is <datatype>#<value>

Download Changes



- Why is “NumberOfCars” not the same value in both POU’s?

The screenshot illustrates the relationship between global and local variables in a PLC program. It is divided into four main sections:

- Top Left: Global Variable Declaration**

Name	Online value	Type	Usage	Descript
Default				
CTU_1		CTU	VAR	
CurrentValue	0	INT	VAR	
Full	FALSE	BOOL	VAR	
GateOpen	TRUE	BOOL	VAR	
MaxCapacity	20	INT	VAR	
NumberOfCars	5	INT	VAR	
OpenForBusiness	TRUE	BOOL	VAR	
Reset	FALSE	BOOL	VAR	
Sensor	FALSE	BOOL	VAR	
- Top Right: Ladder Logic**

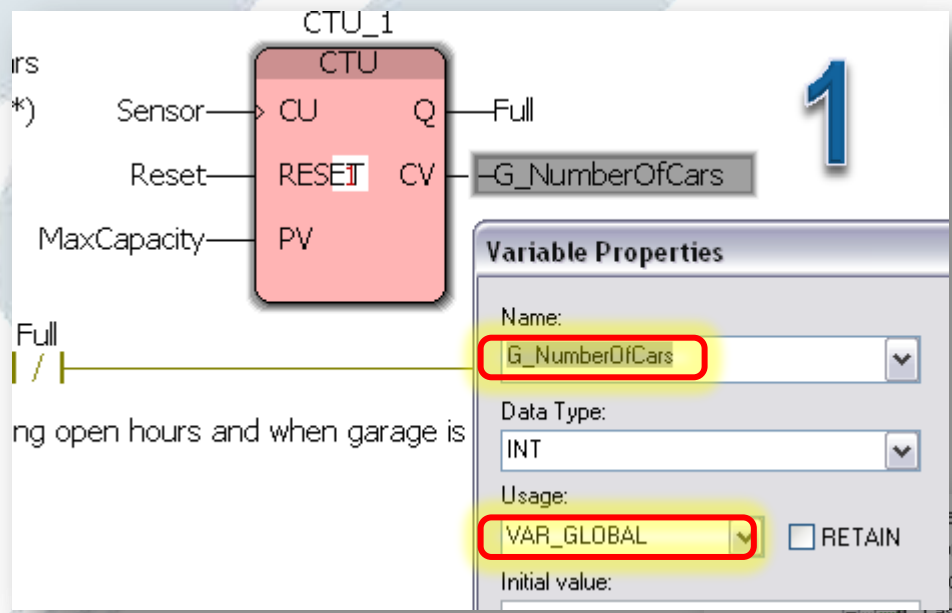
Diagram of a Counter Up (CTU) block labeled CTU_1. The 'CV' (Current Value) input is connected to a variable named 'NumberOfCars', which is highlighted with a red circle and has a tooltip showing its value as 5. The 'PV' (Present Value) input is connected to 'MaxCapacity' (value 20). The 'Q' (Output) is labeled 'Full'.
- Bottom Left: Local Variable Declaration**

Name	Online value	Type
Default		
Income	0.0000000	LREAL
NumberOfCars	0.0000000	LREAL

Open local variable list
- Bottom Right: Ladder Logic**

Diagram of a calculation block. The output is calculated as `Income := NumberOfCars * LREA`. A tooltip for the 'NumberOfCars' variable in this block shows its value as 0.0000000. A blue box with the text 'Hover mouse for tooltip' points to this tooltip.

- *Use New Global Variable in both POUs*
 - *G_NumberOfCars*
 - *Global Variable List*



CTU_1
CTU

Sensor → CU Q → Full

Reset → RESET CV → G_NumberOfCars

MaxCapacity → PV

Full

ng open hours and when garage is

1

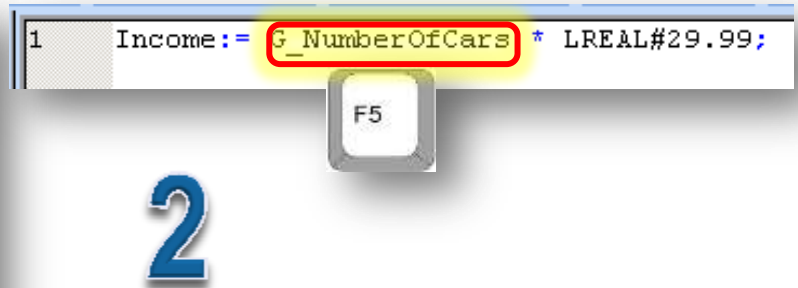
Variable Properties

Name: G_NumberOfCars

Data Type: INT

Usage: VAR_GLOBAL RETAIN

Initial value:

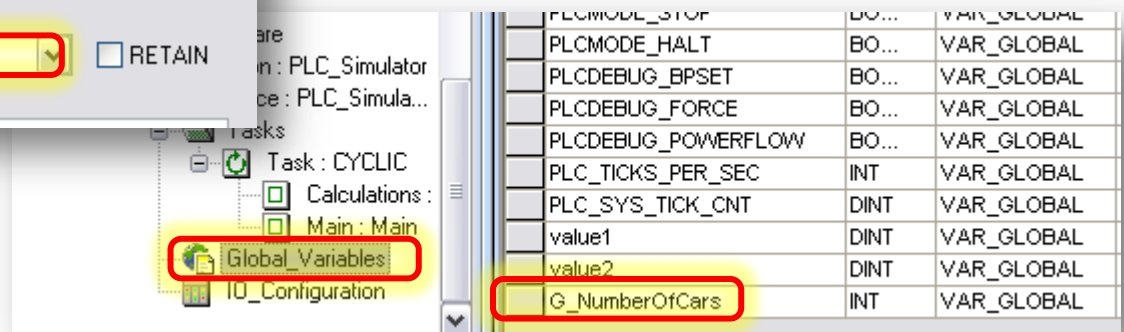


1 Income := G_NumberOfCars * LREAL#29.99;

F5

2

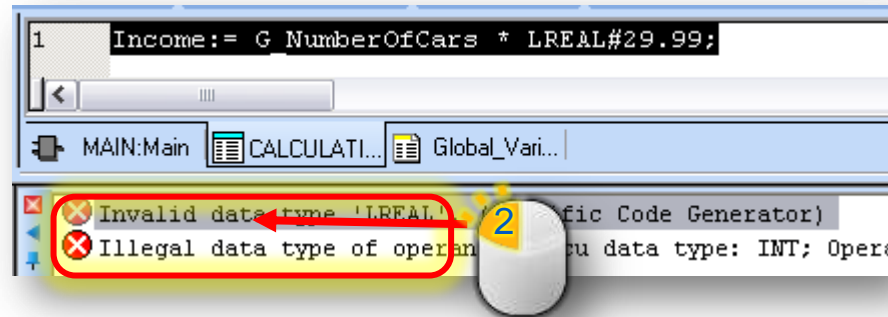
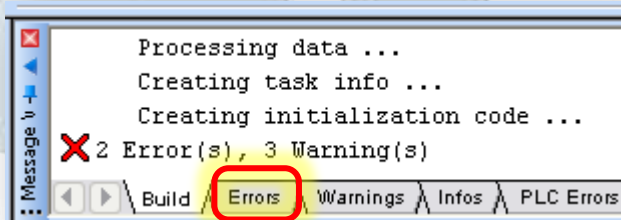
Best Practice: **G_ prefix**



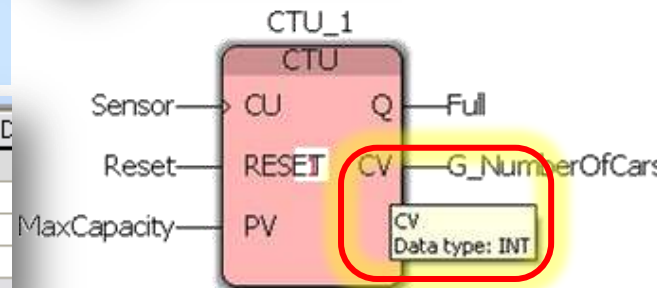
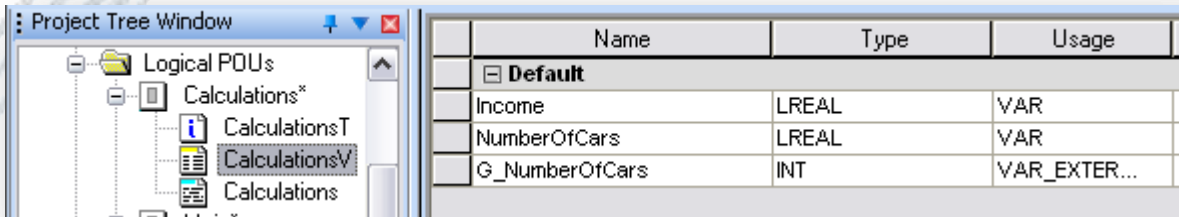
Variable Name	Data Type	Usage
PLC_MODE_STOP	BO...	VAR_GLOBAL
PLC_MODE_HALT	BO...	VAR_GLOBAL
PLC_DEBUG_BPSET	BO...	VAR_GLOBAL
PLC_DEBUG_FORCE	BO...	VAR_GLOBAL
PLC_DEBUG_POWERFLOW	BO...	VAR_GLOBAL
PLC_TICKS_PER_SEC	INT	VAR_GLOBAL
PLC_SYS_TICK_CNT	DINT	VAR_GLOBAL
value1	DINT	VAR_GLOBAL
value2	DINT	VAR_GLOBAL
G_NumberOfCars	INT	VAR_GLOBAL

- *IEC 61131-3 “Strict Data Typing”*
 - *Operations on data with same data type only*
 - *Convert data type in code; no automatic conversion*

Download Changes



INT cannot multiply by LREAL




CV output datatype is INT

- *Use Conversion Functions in code*
 - *Best Practice required by IEC 61131-3*

```

1  (* Result as LREAL *) :=INT_TO_LREAL((* IN as INT *));
2
3  Income := LREAL#29.99 * NumberOfCars;
```

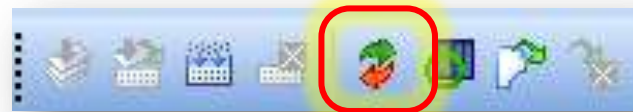


Edit Wizard

Group:

Type conv. FUs

Name	Description
<input type="checkbox"/> INT_TO_LINT	
<input checked="" type="checkbox"/> INT_TO_LREAL	Converts INT to LRI
<input checked="" type="checkbox"/> INT_TO_REAL	Converts INT to RE.
<input type="checkbox"/> INT_TO_SINT	Converts INT to SINT



1	8.0000000	NumberOfCars :=INT_TO_LREAL(G_NumberOfCars);
2		
3	239.9200000	Income := LREAL#29.99 * NumberOfCars;

HELP DOCUMENTATION

Programming System
IEC 61131 blocks
Right-Click Help
Contextual Help Button

Contextual Help Button
Right-Click Help

- *CHM Documentation Installed*

- *MotionWorks IEC*

- » *MPReadMe001.chm*

- *Help on Functions and Function Blocks*

- » *PLC_FBfun001.chm*

- *Search for *.CHM*

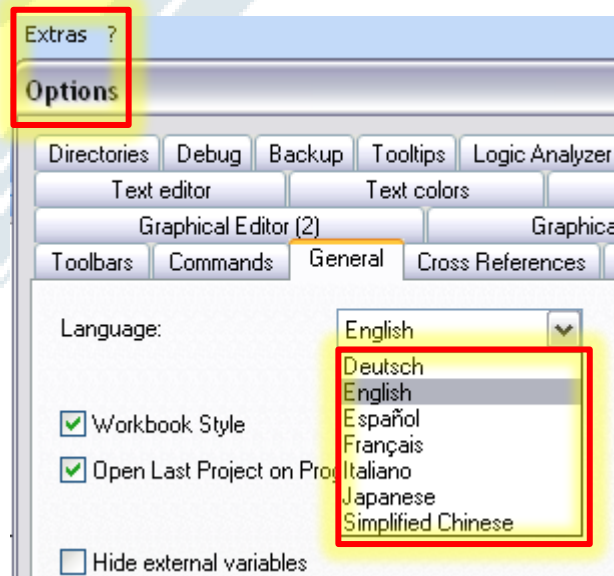
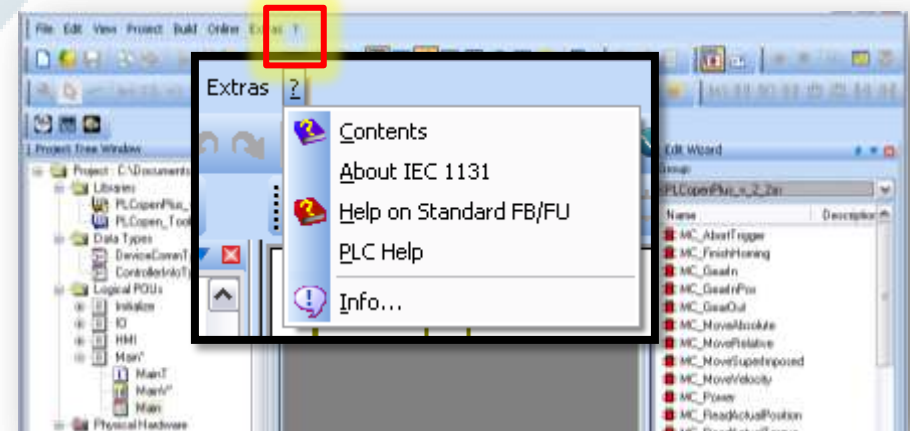
- » *C:\Program Files (x86)\Yaskawa*



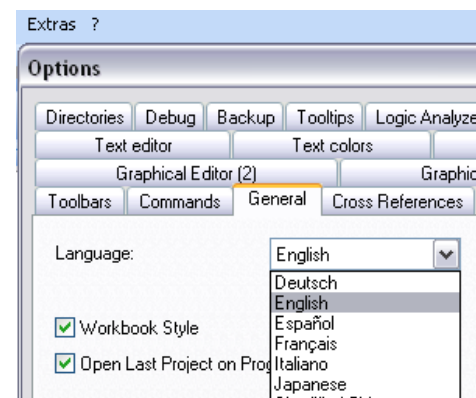
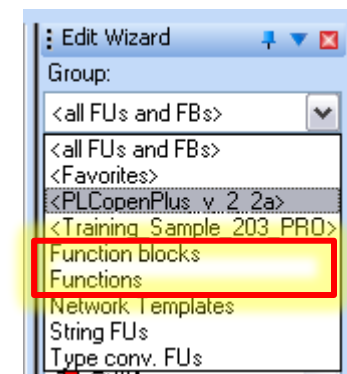
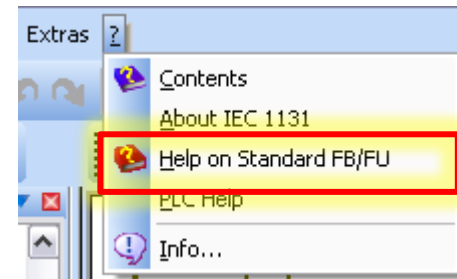
001 English
081 Japanese
086 Chinese



- *Programming System*
- *IEC 61131-3 Basics*
 - *“About IEC 1131”*
- *General PLC*
 - *“PLC Help”*
- *Available in several languages*
 - *Extras – Options – General – Language*

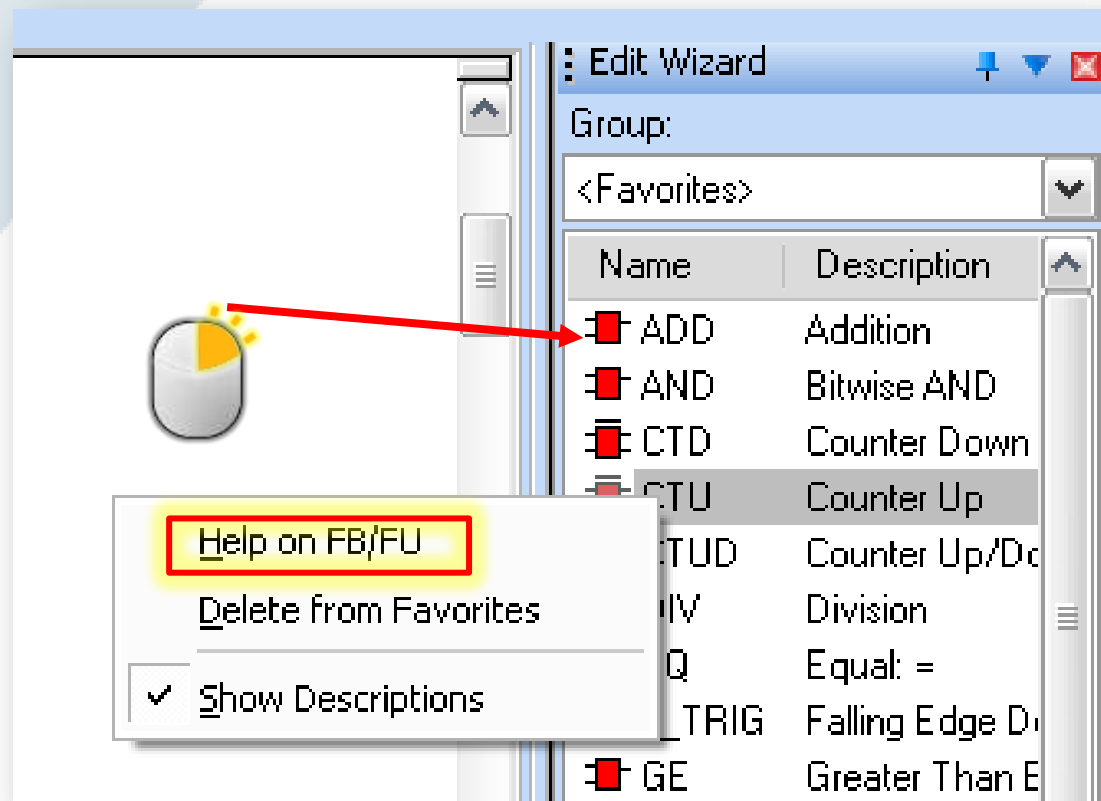


- *Standard IEC 61131-3 blocks*
 - *Add, subtract, timer, pulse, ...*
- *Applies to These Groups in Edit Wizard*
 - *Function Blocks*
 - *Functions*
 - *String FUs*
 - *Type conv. FUs*
- *Available in Several Languages*
 - *Extras – Options – General - Language*



Help File
PLC_FBfun001.chm

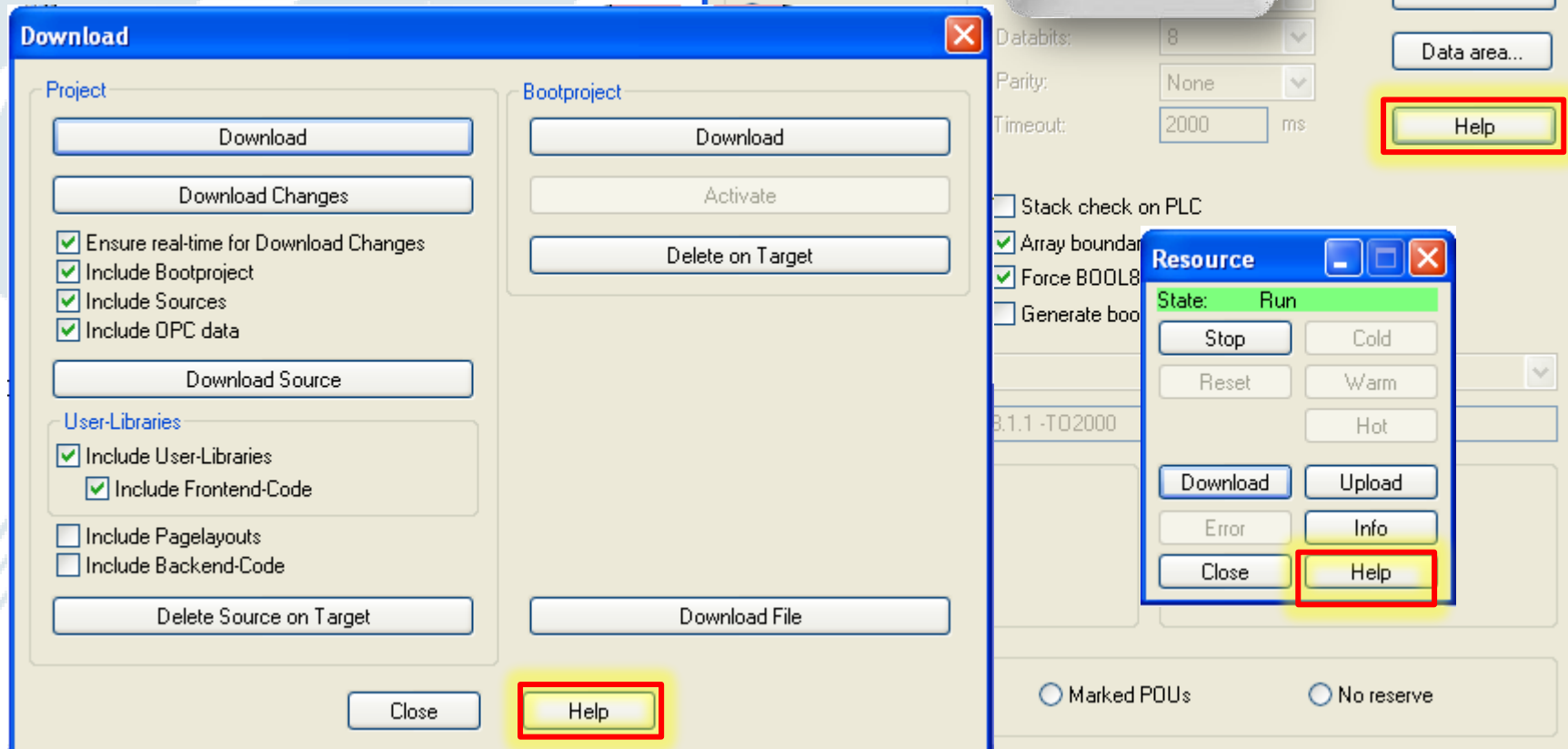
- *Right click a block for help*
 - *In the code*
 - *In the edit wizard*



- Use the Help Button and F1 key

Help

F1
(help)

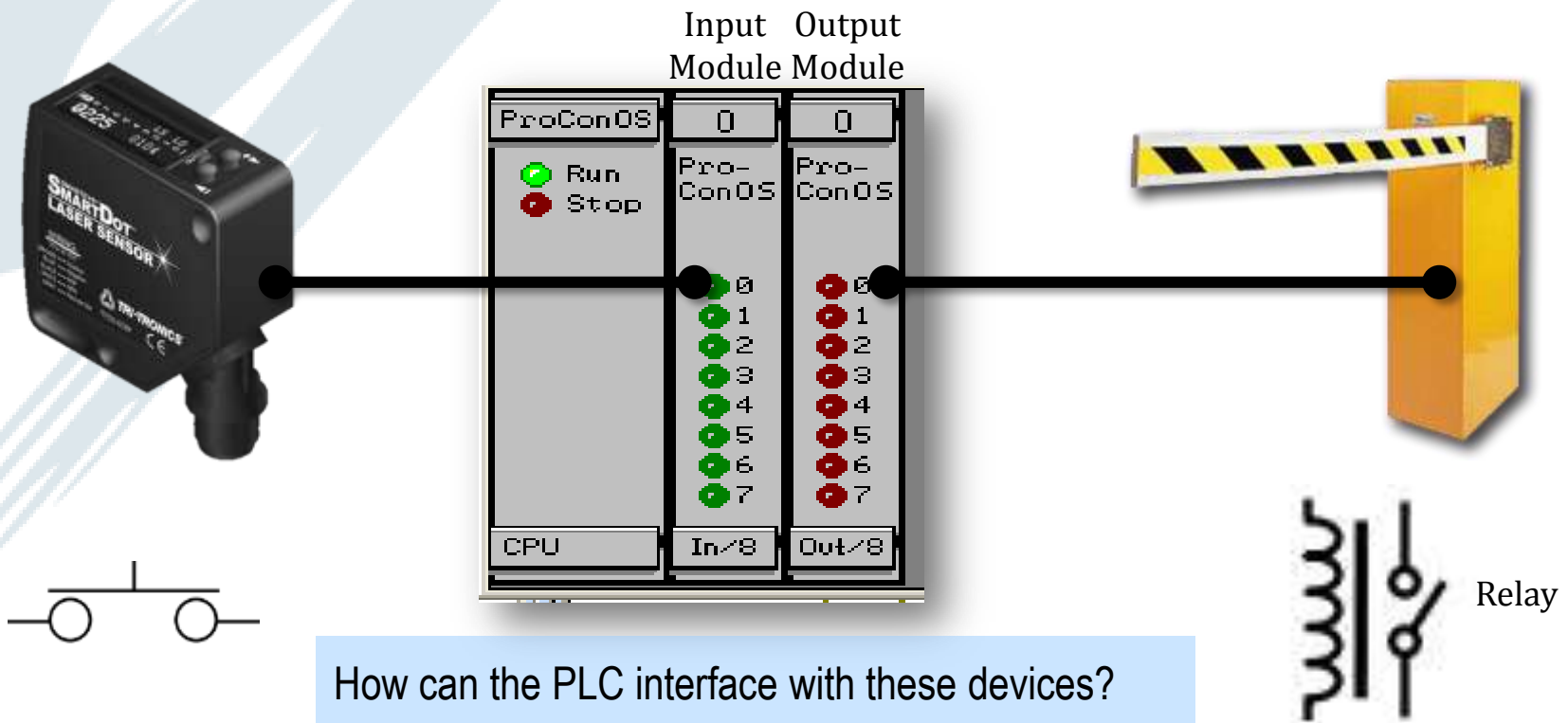


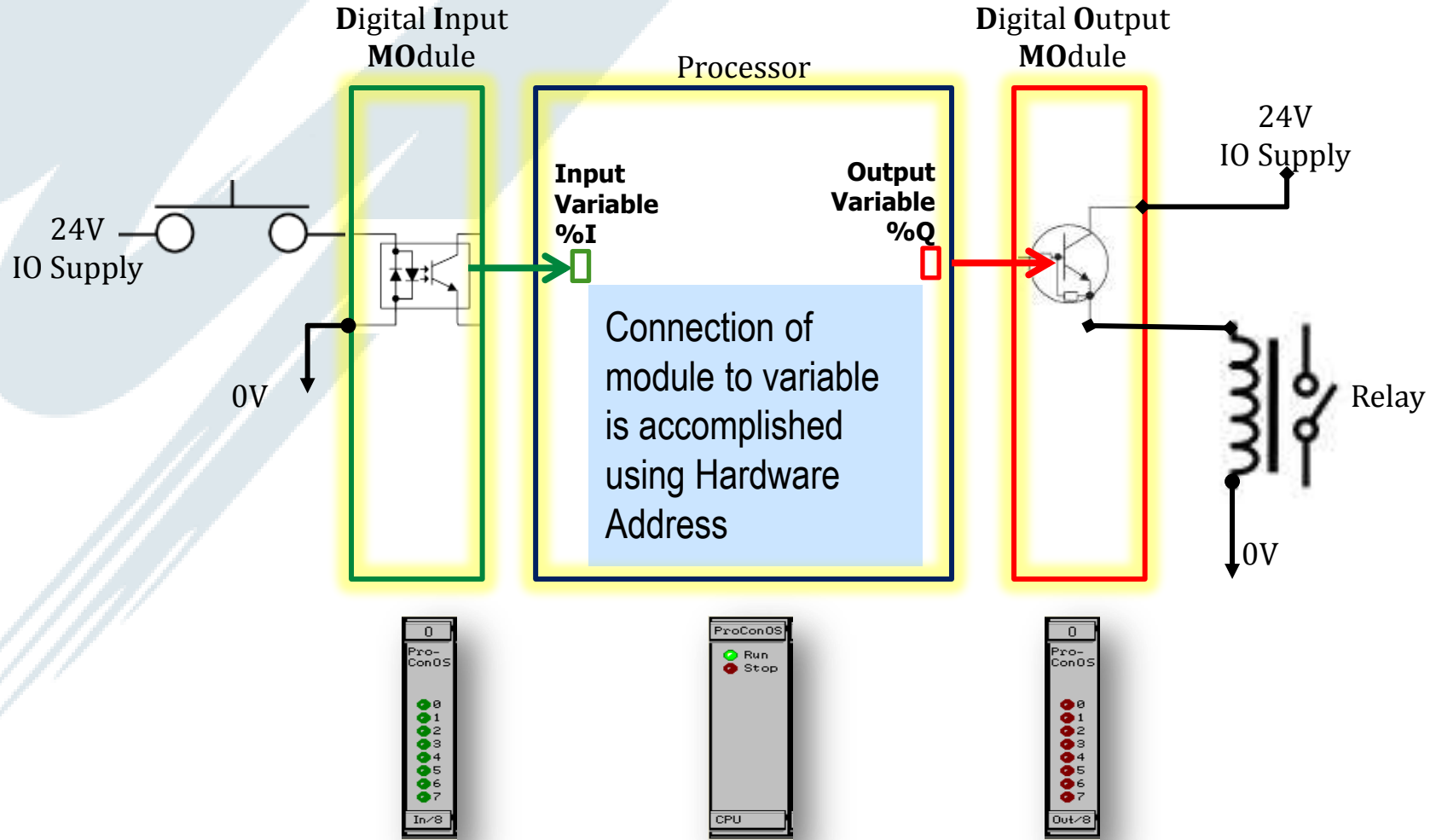
Using I/O

I/O Module Concept
I/O Variable Addressing
Configuration
Best Practice
Execution Sequence
Task Assignment

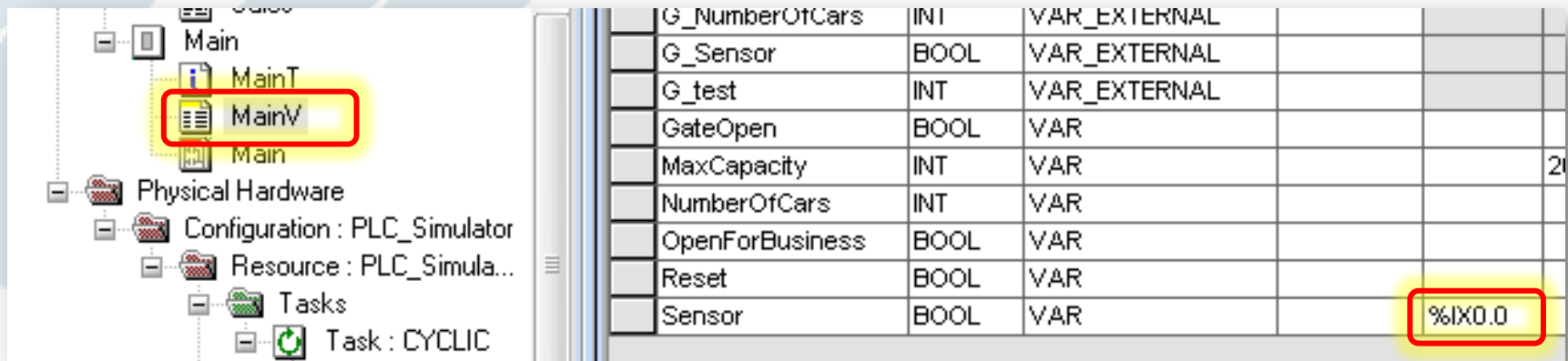
Task Assignment
Execution Sequence

- *Parking Gate Example Application*
 - *Connect physical sensor to PLC input*
 - *Connect physical gate to PLC output*





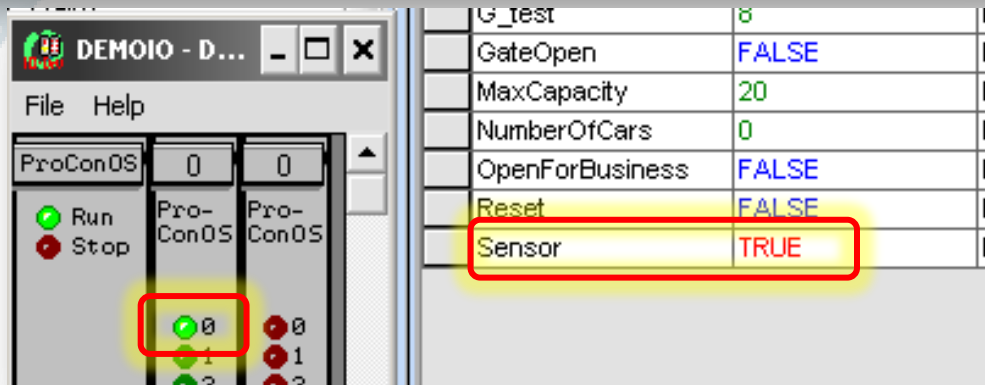
- Any variable can be given a hardware address
 - Make “sensor” an input variable with address %IX0.0
 - In debug mode, turn the simulator input on/off



The screenshot shows the project tree on the left with 'MainV' highlighted in a red box. The variable declaration table on the right is as follows:

G_NumberOfCars	INT	VAR_EXTERNAL		
G_Sensor	BOOL	VAR_EXTERNAL		
G_test	INT	VAR_EXTERNAL		
GateOpen	BOOL	VAR		
MaxCapacity	INT	VAR		20
NumberOfCars	INT	VAR		
OpenForBusiness	BOOL	VAR		
Reset	BOOL	VAR		
Sensor	BOOL	VAR		%IX0.0

The value '%IX0.0' in the 'Sensor' row is highlighted with a red box.



The screenshot shows the 'DEMOIO - D...' simulator window. The variable values are displayed in a table:

G_test	8	IN
GateOpen	FALSE	B
MaxCapacity	20	IN
NumberOfCars	0	IN
OpenForBusiness	FALSE	B
Reset	FALSE	B
Sensor	TRUE	B

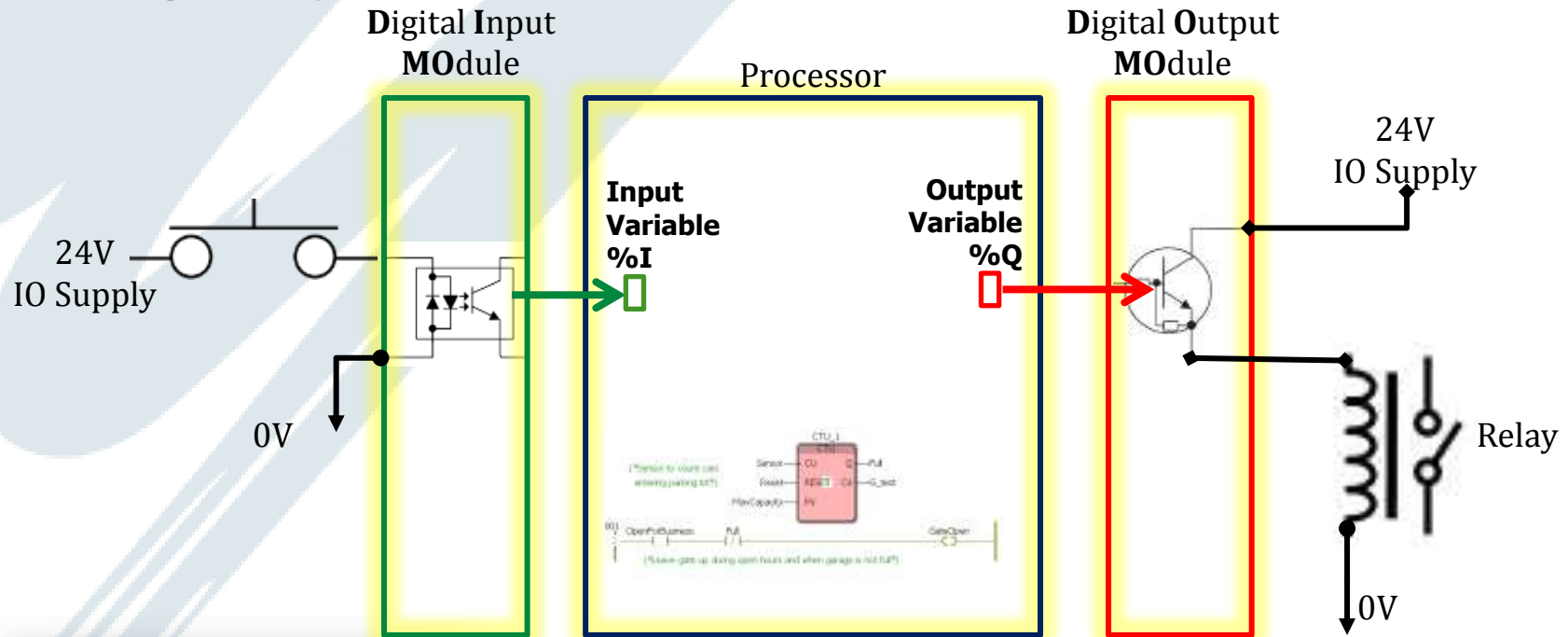
The 'Sensor' row is highlighted with a red box. Below the table, the physical hardware status is shown with a red box around the '0' value:

ProConOS	0	0
Run	0	0
Stop	0	0
0	0	0
1	1	1
2	2	2

Where does %IX0.0
come from?

- Explanation coming next

- *Connection of hardware to software*
 - *IO Driver*



```

    Physical Hardware
    - Configuration: PLC_Simulator
      - Resource: PLC_Simula...
        - Tasks
          - Task: CYCLIC
            - z: Calculation...
            - Main: Main
        - Global_Variables
        - IO_Configuration
    
```



- IEC 61131-3 Byte level addressing

%IX 43210 .7

Data Size

X – Bit (1 bit)
B – Byte (8 bits)
W – Word (16 bits)
D – Double (32 bits)
L – Long (64 bits)

Location Prefix

%I – Input Location
%Q – Output Location
%M – Memory Location

Bit Indicator

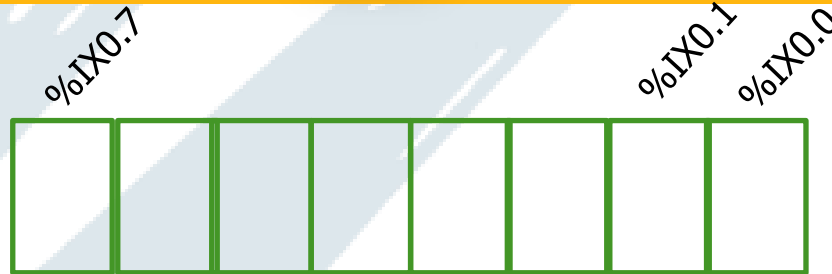
.0 to .7
*Only for Bit (X)
 data size*

Memory Address

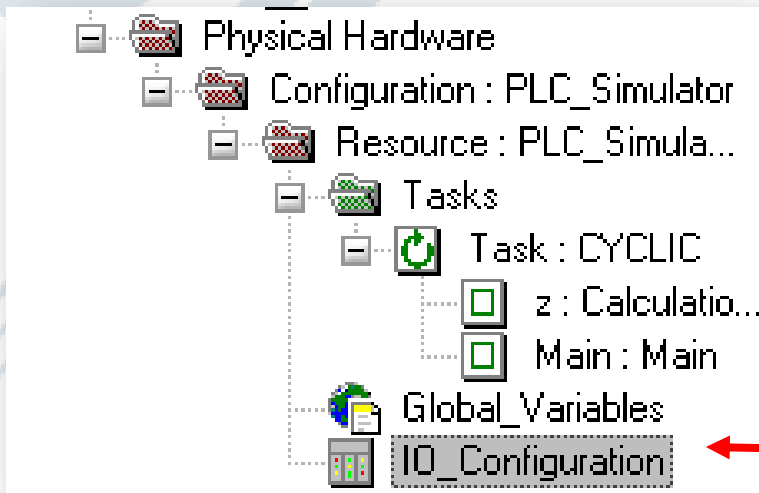
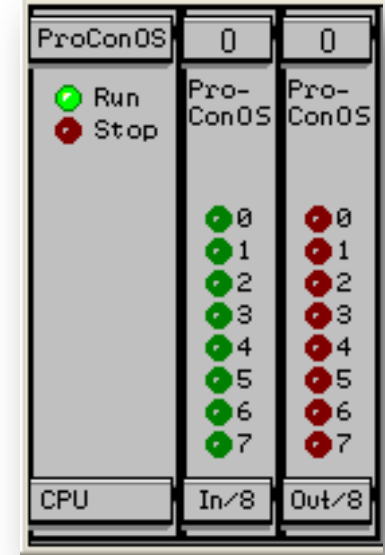
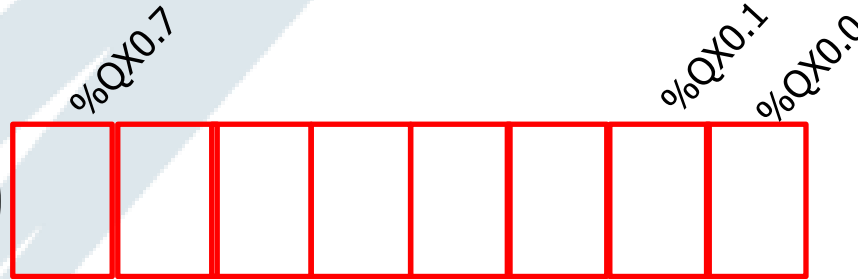
0 to max supported
 Byte level addressing



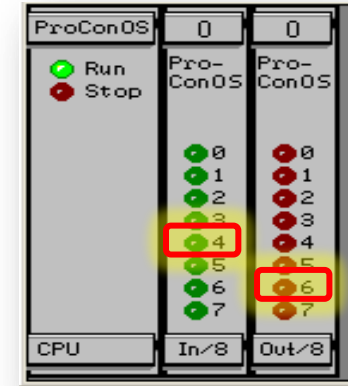
%IB0



%QB0

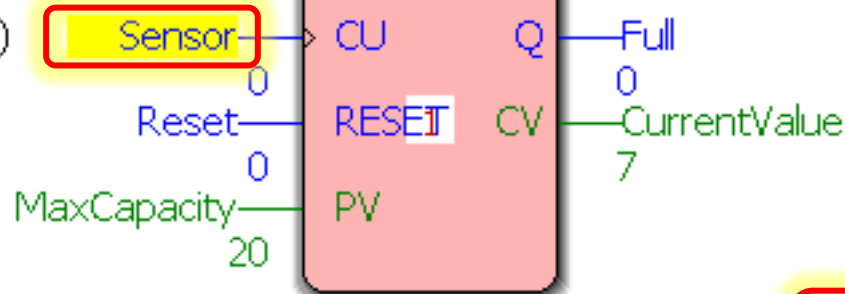


- Sensor** corresponds to input 4 on the simulator. What address is required?
 - `%IX0.4`
- GateOpen** corresponds to output 6 on the simulator. What address is required?
 - `%QX0.6`



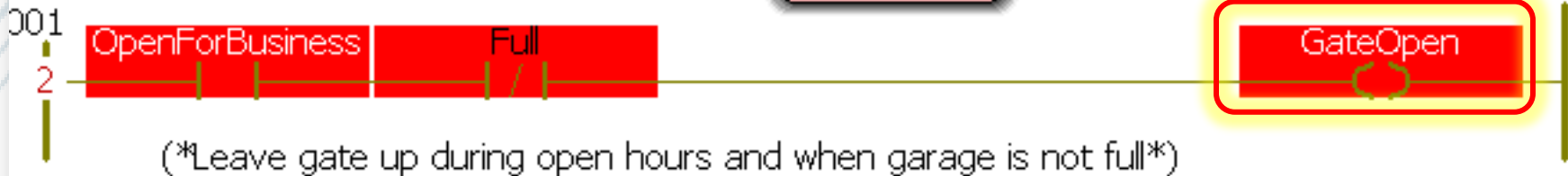
1

(*Sensor to count cars entering parking lot*)



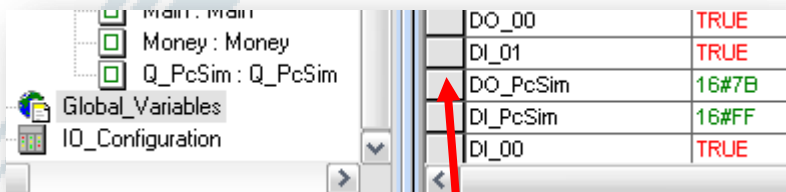
Make it work!

2

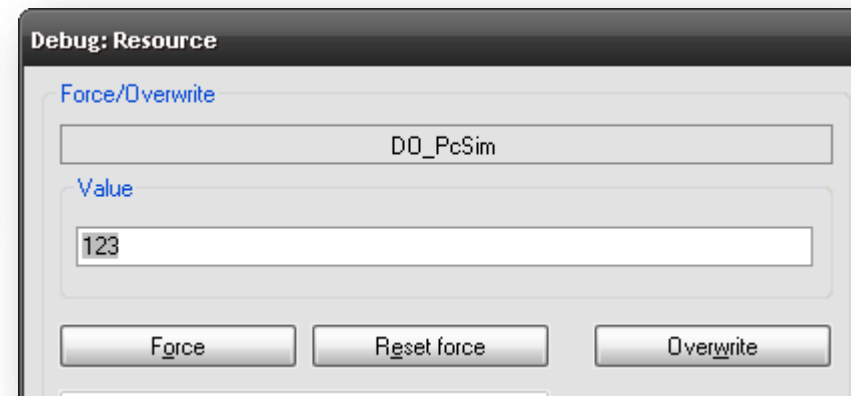


- Create an input variable **DI_PcSim** in *Global_Variables*
 - All 8 inputs as a byte
- Create output variable **DO_PcSim** in *Global_Variables*
 - All 8 outputs as a byte
- Read inputs in debug mode
 - What is the value with all inputs on?
- Overwrite outputs in debug mode
 - How can you enter hex values directly?

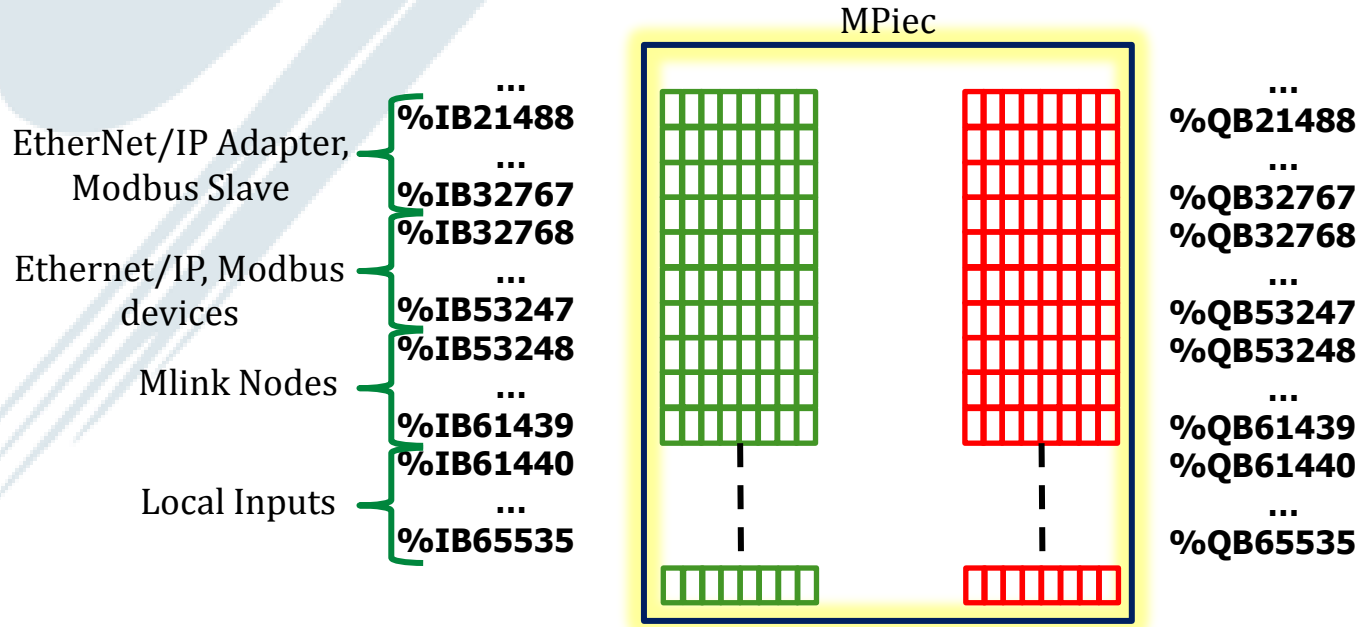
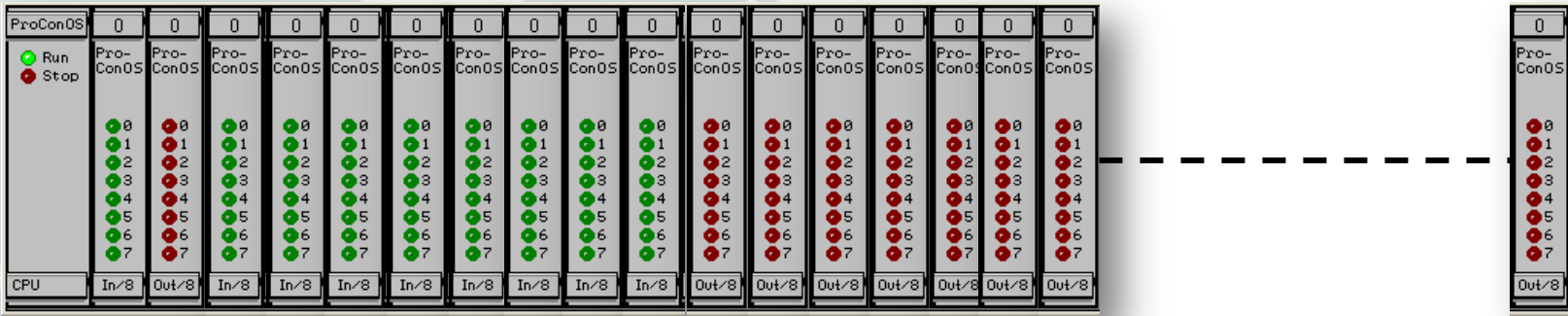
What happens in debug mode if there is no IO module configured at the address of the variable – for example try %IB2



DO_00	TRUE
DI_01	TRUE
DO_PcSim	16#7B
DI_PcSim	16#FF
DI_00	TRUE

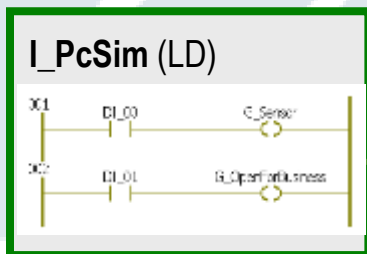


- Large number of input modules possible in real PLC

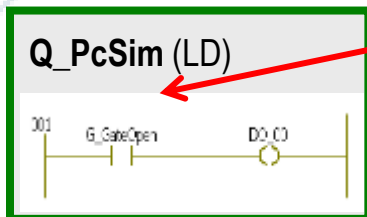
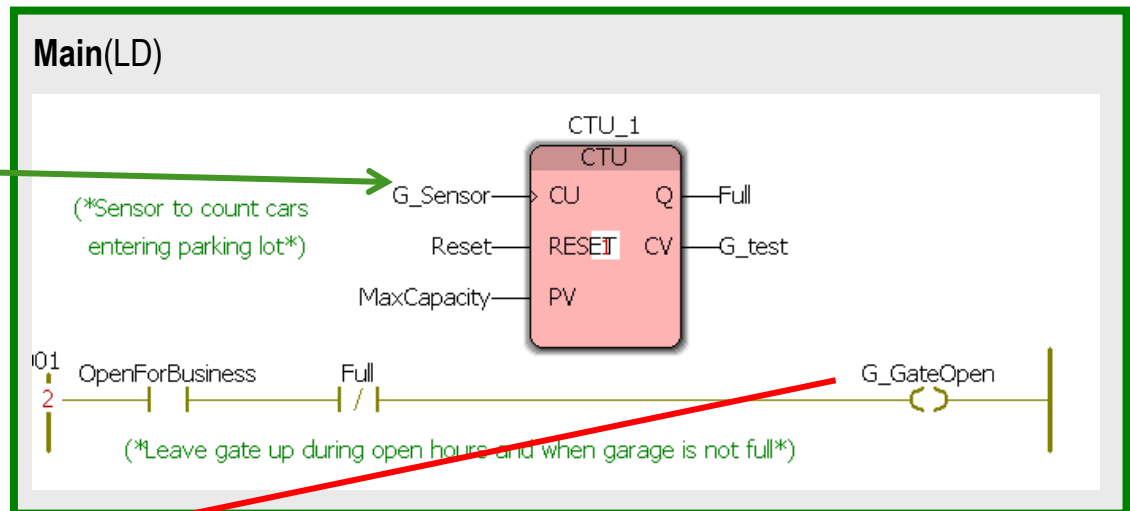


MotionWorks IEC controls the addresses and creates many IO variables automatically for the MPiec controllers

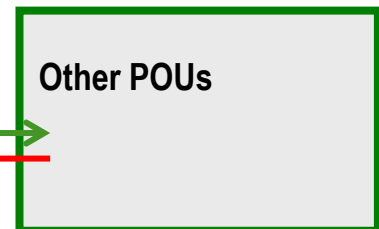
- **Generic IO variables in Global_Variables list**
 - DI_00, DI_01, etc.
 - DO_00, DO_01, etc.
- **Program IO variables in separate program POU's**
 - POU's easily reused with other IO hardware – modularity
 - » Example: move from one HMI to another – addresses may change
 - 1 or 2 POU's affected by hardware change
 - Transfer I/O data through global variables



G_Sensor transfers data from input 0



G_GateOpen transfers data to output 0



- Project Update Checklist

- Global Variables in Code

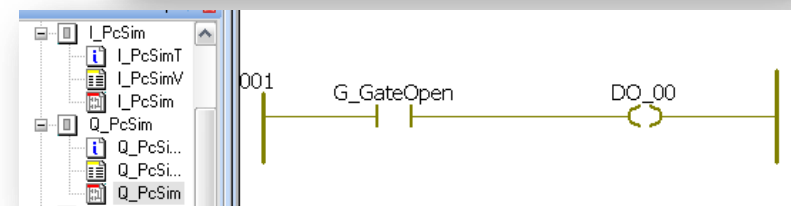
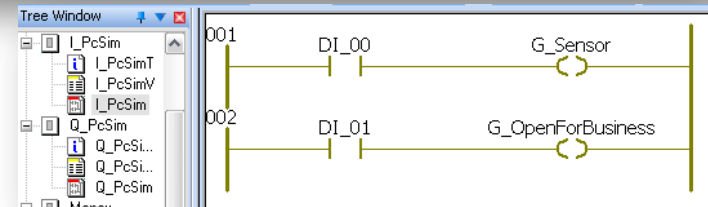
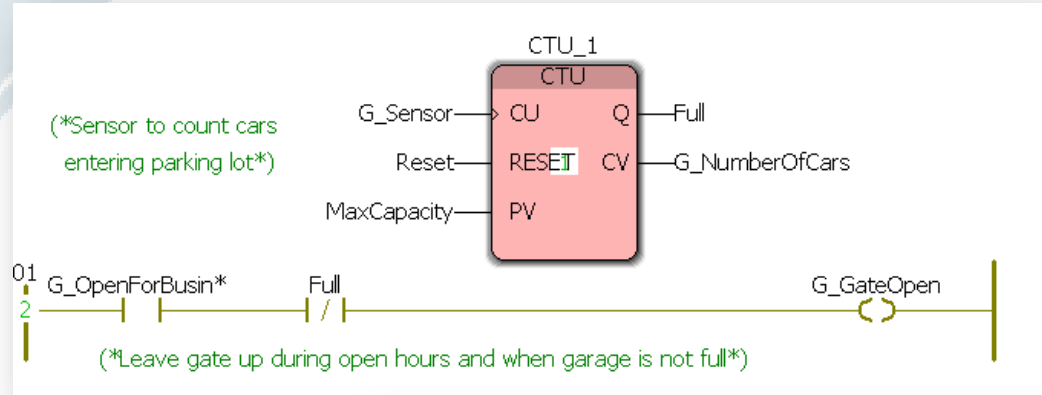
- » *G_Sensor*
- » *G_OpenForBusiness*
- » *G_GateOpen*

- Global IO Variables

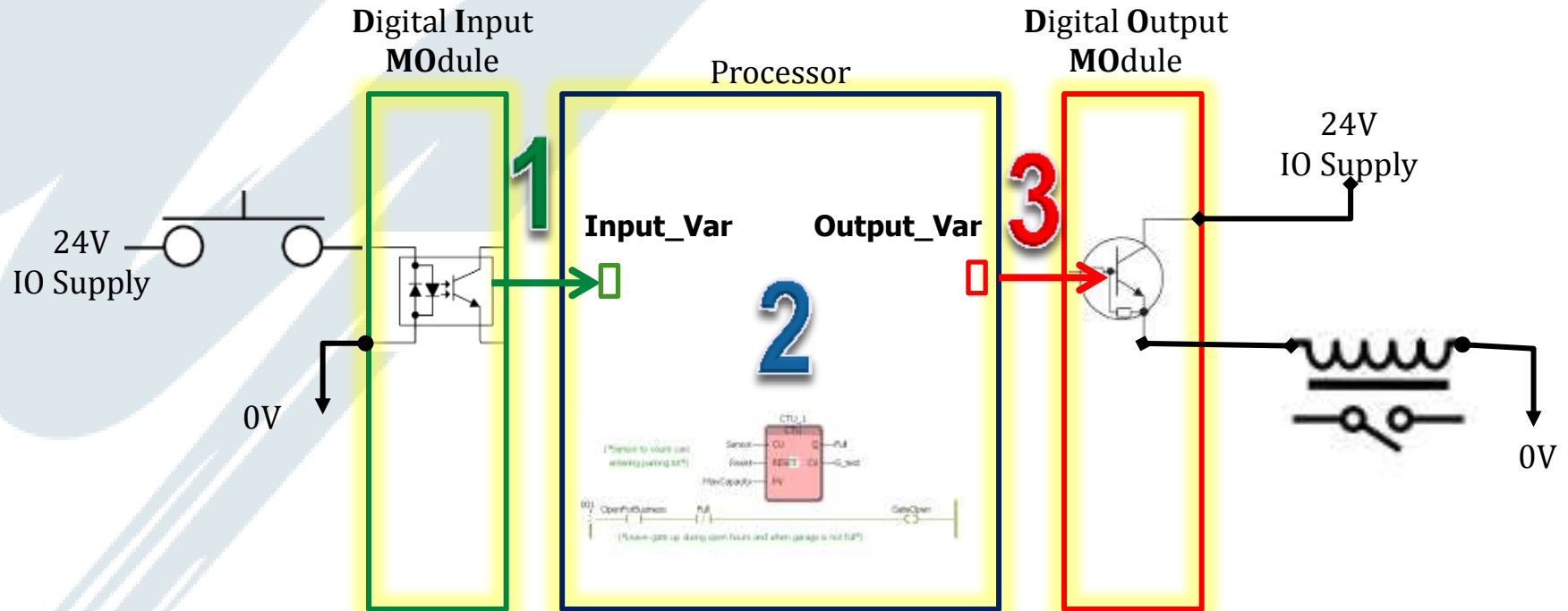
- » *DI_00, DI_01*
- » *DO_00*

- LD Program POU's

- » *Q_PcSim*
- » *I_PcSim*
- » *Instance in Task*

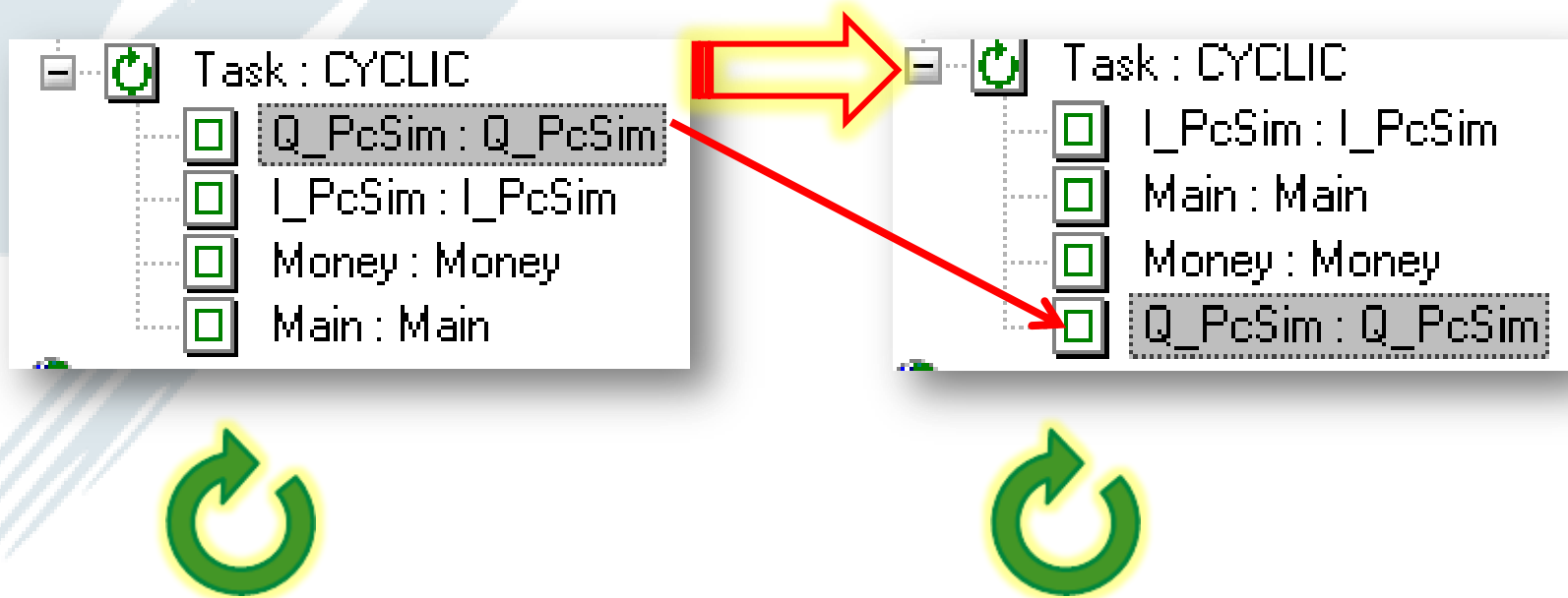


Task	Symbol	DataType	Variable Type	Address
G_Sensor		BOOL	VAR_GLOBAL	
G_OpenForBusiness		BOOL	VAR_GLOBAL	
G_NumberOfCars		INT	VAR_GLOBAL	
G_GateOpen		BOOL	VAR_GLOBAL	
DO_00		BOOL	VAR_GLOBAL	%QX0.0
DI_01		BOOL	VAR_GLOBAL	%IX0.1
DI_00		BOOL	VAR_GLOBAL	%IX0.0

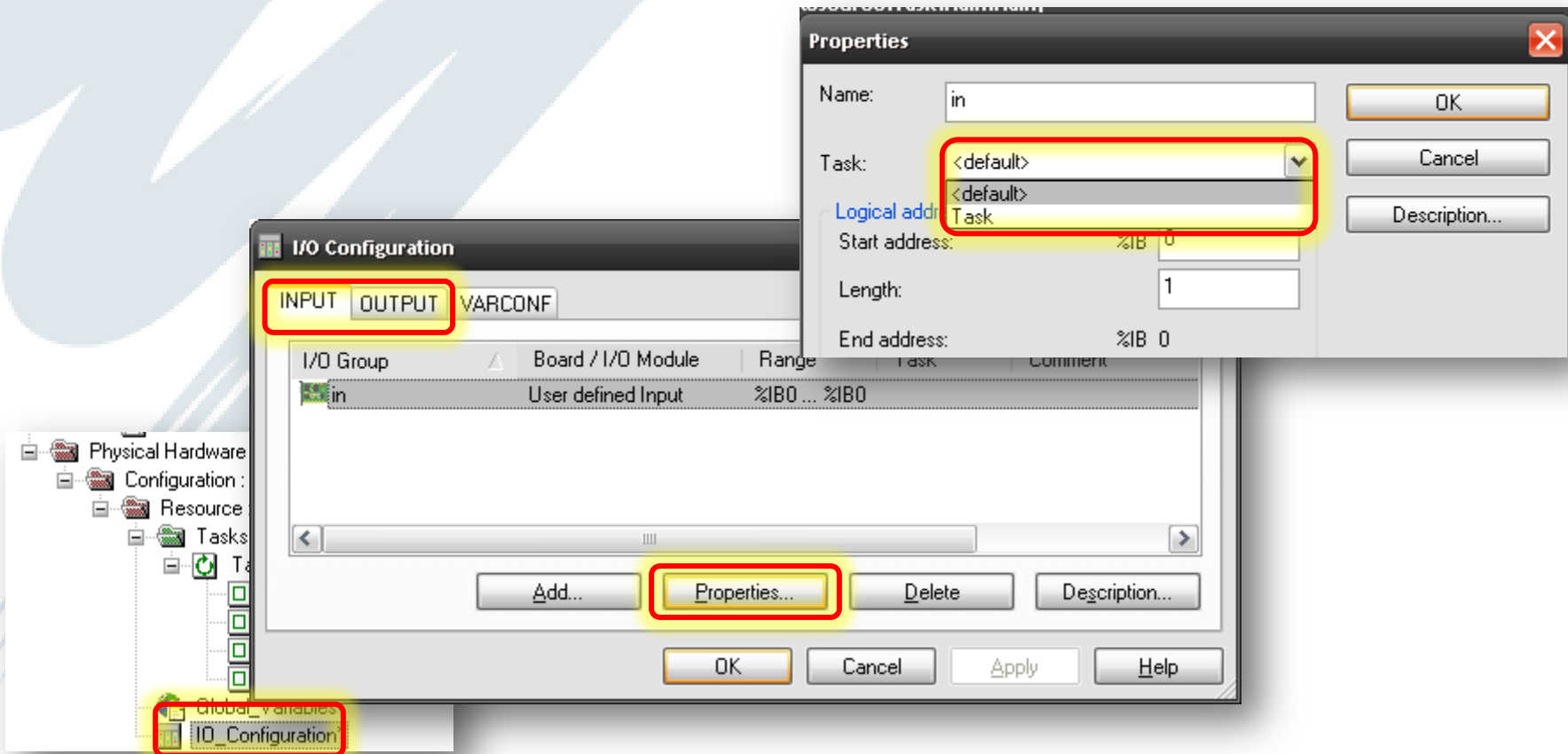


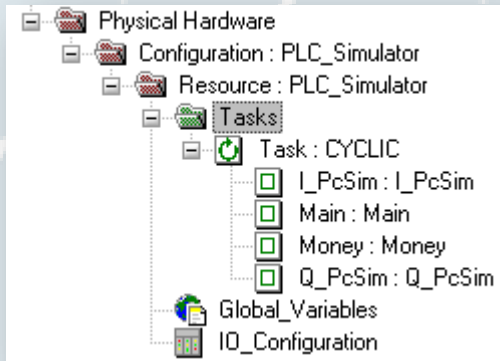
1. Input variables are read from input modules
2. Processor solves application code, updating output variables from top to bottom of task
3. Output variables are written to output modules

- *Order of POU instances in task affect operation*
 - *Read Input Variables to global variables at top of task list*
 - *Write Output Variables at bottom of task list*

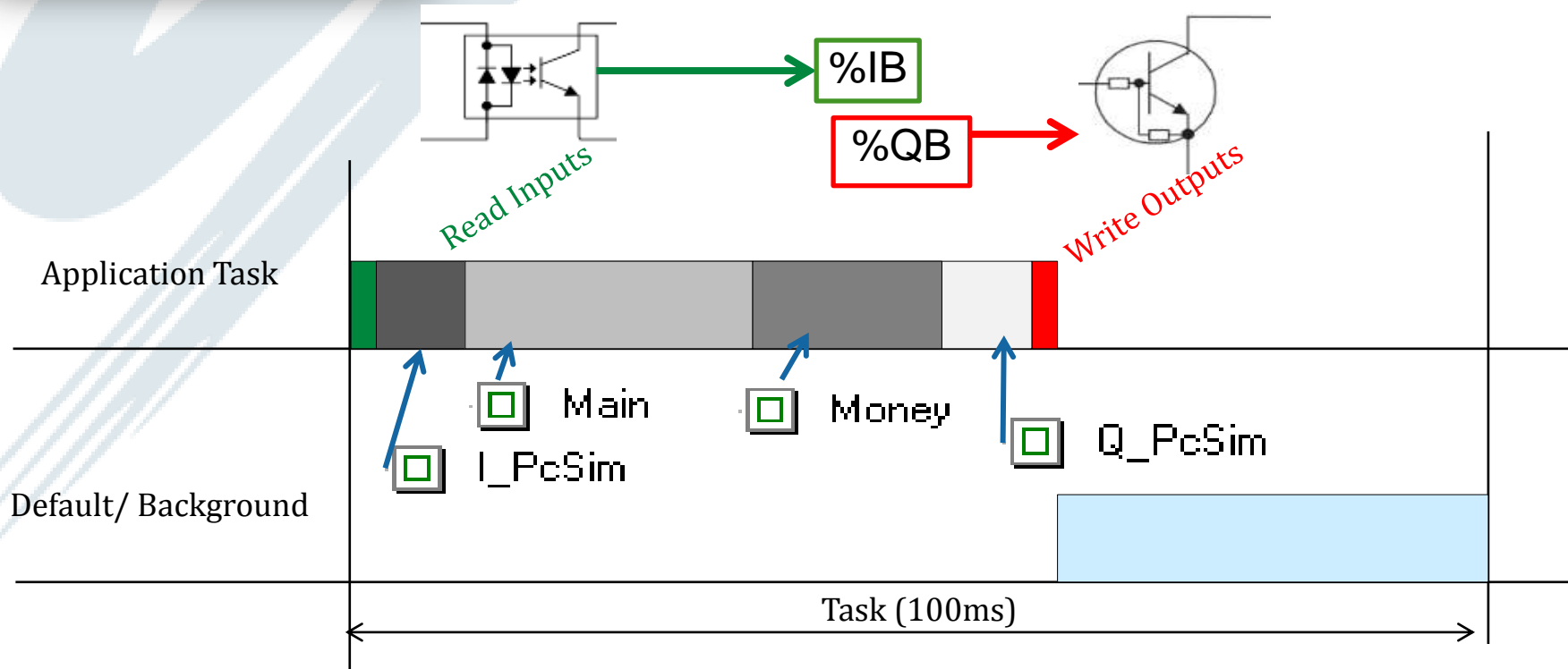


- Assign IO to the task in which they are used
 - Physical Hardware – IO_Configuration: Properties - task





- *Inputs are read into %IB before task execution starts*
- *Task is executed*
- *%QB are written to outputs immediately after task execution ends.*



Monitoring

Force I/O variables
LREAL Value Display
Watch Window
Tooltip
Cross Reference

Cross Reference

Tooltip

WATCH WINDOW

Monitor Global Variables

G_NumberOfCars	1	INT	VAR_GLOBAL		
G_GateOpen	TRUE	BOOL	VAR_GLOBAL		
DO_PcSim	16#01	BYTE	VAR_GLOBAL		%QB0
DO_00	TRUE	BOOL	VAR_GLOBAL		%QX0.0
DI_PcSim	16#03	BYTE	VAR_GLOBAL		%IB0
DI_01	TRUE	BOOL	VAR_GLOBAL		%IX0.1
DI_00	TRUE	BOOL	VAR_GLOBAL		%IX0.0

Force boolean to
TRUE
FALSE

Or Reset Force

Debug: Resource

Force/Overwrite

DO_00

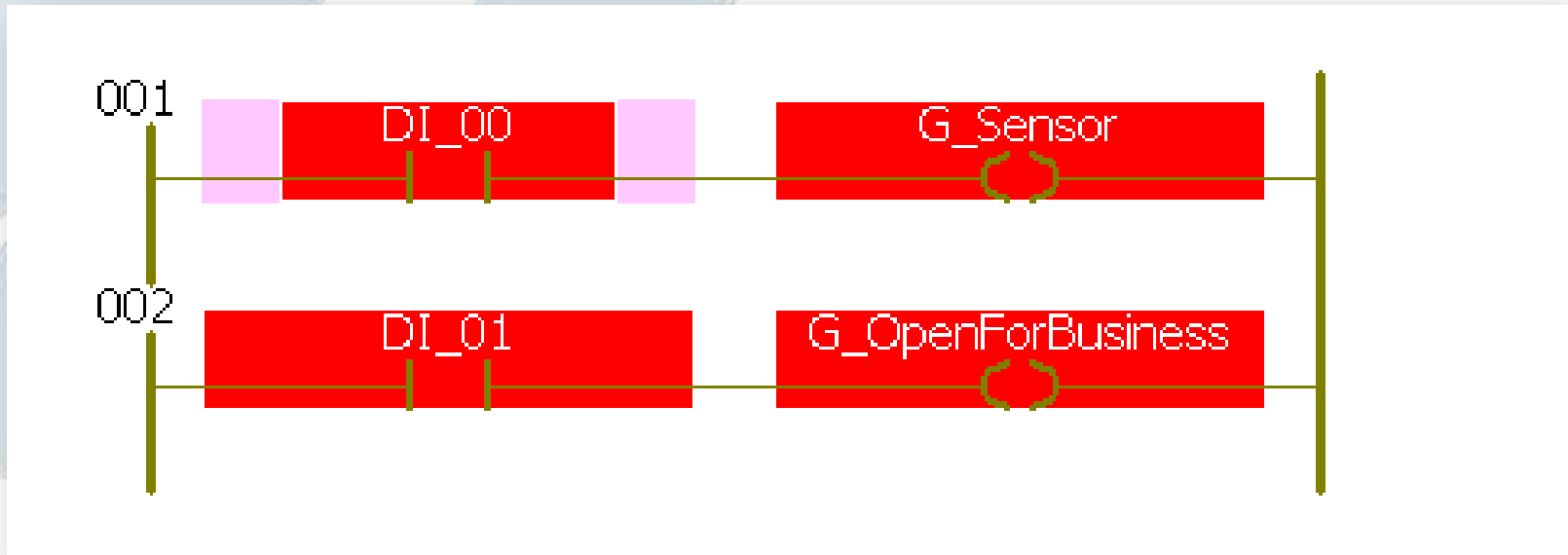
Value

TRUE FALSE

Force **Reset force** **Overwrite**

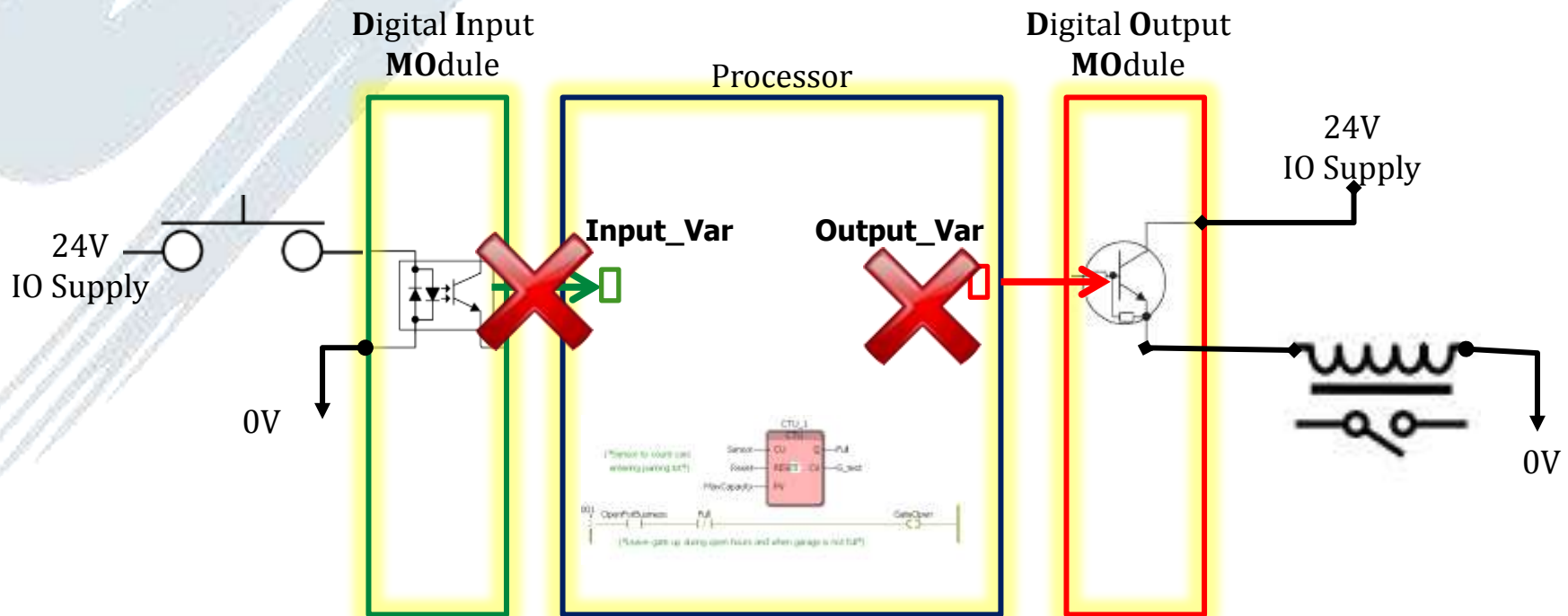
Reset force list

- *Pink Highlight*



DO_PcSim	16#00	BYTE	VAR_GLOBAL	%QB0
DO_00	FALSE	BOOL	VAR_GLOBAL	%QX0.0
DI_PcSim	16#7B	BYTE	VAR_GLOBAL	%IB0
DI_01	TRUE	BOOL	VAR_GLOBAL	%IX0.1
DI_00	TRUE	BOOL	VAR_GLOBAL	%IX0.0

- *What is “Forcing” and an I/O variable?*
 - *Variable in processor is held at a specific value regardless of code execution*
 - *Processor does not see true input state*
 - *Output does not reflect result of code*



- *Debug Mode*
 - *Info - Force*

Debug: Resource

Force/Overwrite

DO_00

Value

IRUE FALSE

Force **Reset force**

Reset force list

Close **Info** **Help**

Resource: Resource

Resource POU's **Force**

Resource: Resource

Resource POU's Force

Variables name	Instance Path	Address	Value
<input type="checkbox"/> DO_00	Global variables	%QX 0.0	FALSE
<input type="checkbox"/> DI_PcSim	Global variables	%IB 0	123
<input type="checkbox"/> DI_00	Global variables	%IX 0.0	TRUE

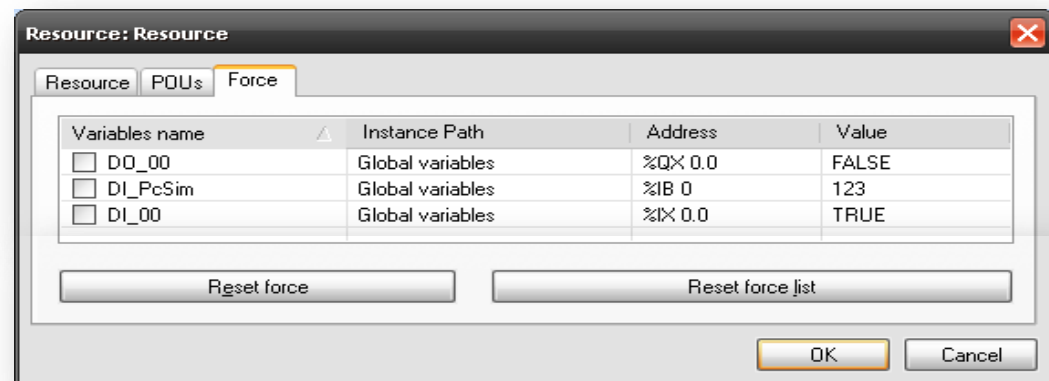
Reset force **Reset force list**

OK **Cancel**

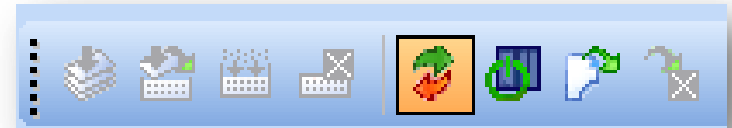
OK **Cancel**

Time between transmissions: 10 ms

- *Limitations during Force*
 - Force state is not saved in the project
 - Force DOES survive warm start and cold start
 - Task instances cannot be deleted or rearranged
 - Task instances CAN be copied
 - » Click and drag will create a copy
 - » Best not to edit code during force



- Add line to “Money” POU
 - $Tax := Income * TaxRate;$
- Value Display
 - Debug Mode
 - Double click any variable
 - Adjust Precision
 - Close



```
29.99 Income := LREAL#29.99 * NumberOfCars;  
2.40 Tax := Income * TaxRate;
```



Valuedisplay

Standard
 Decimal
 Hexadecimal
 Binary

REAL values

Width: Precision:

IEEE format

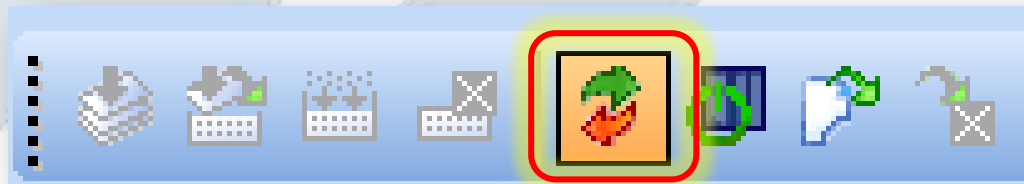
Help

- *Tooltip shows data of variable*
 - *Debug Mode*
 - *Hover over variable*
- *Especially useful in ST code*

```
2.40 Tax := Income * TaxRate;
```

```
TaxRate := 0.08
```

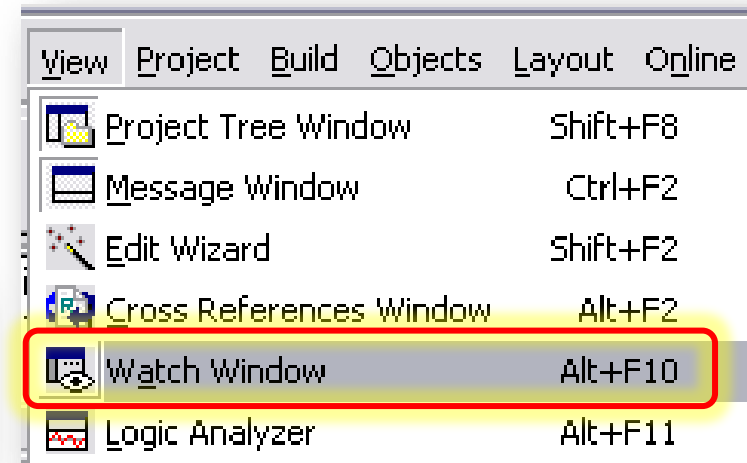
- *Display Watch Window*
 - *Debug*
 - *View – Watch Window*



Variable	Value	Default value	Type
..... NumberOfCars	1.00		LREAL
..... G_NumberO...	1		INT
..... Income	29.99		LREAL
..... TaxRate	0.08		LREAL
..... Tax	2.40		LREAL
..... G_Sensor	TRUE		BOOL
..... MaxCapacity	10		INT
..... DI_01	TRUE		BOOL
..... DI_00	TRUE		BOOL

Watch Window

Watch 1 | Watch 2 | Watch 3 | Watch 4 | Watch 5

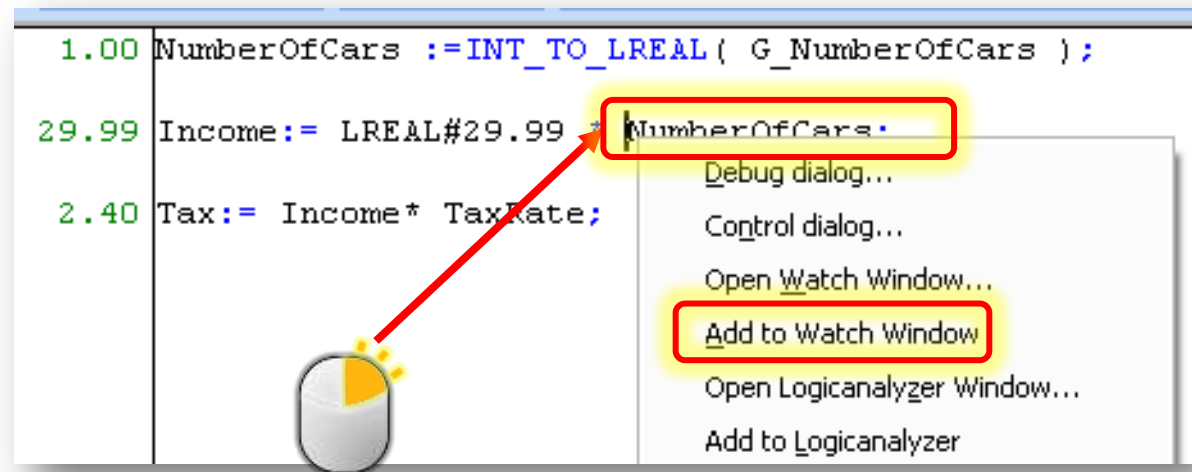
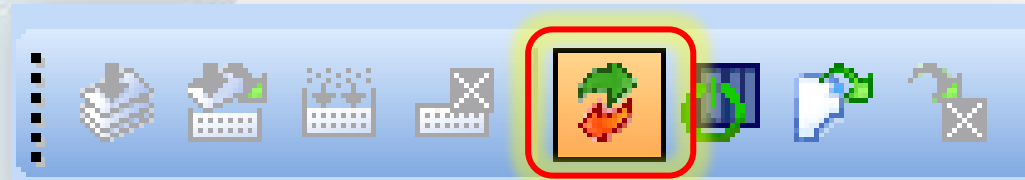


The Watch Window is only available in DEBUG MODE

Tab names can be changed

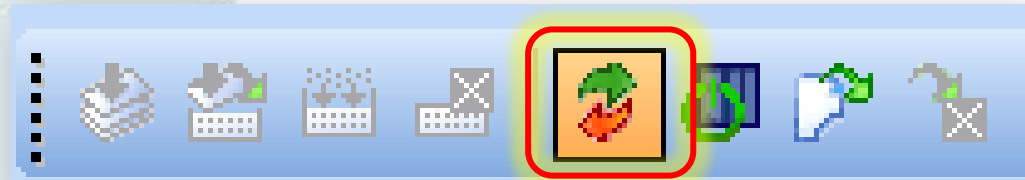
- *Right-Click Method*

- *Debug ON*
- *Right-Click Variable in Code*
- *Right-Click Variable in Variable List*
- *Choose “Add to Watch Window”*



- *Drag & Drop Method*

- *Debug ON*
- *Drag and drop from code only*
 - » *Variable List Not supported*



CTU_1
CTU

(*Sensor to count cars entering parking lot*)

G_Sensor 1 CU Q Full 0

Reset 0 RESET CV

MaxCapacity 10 PV

001 G_OpenForBusin* Full

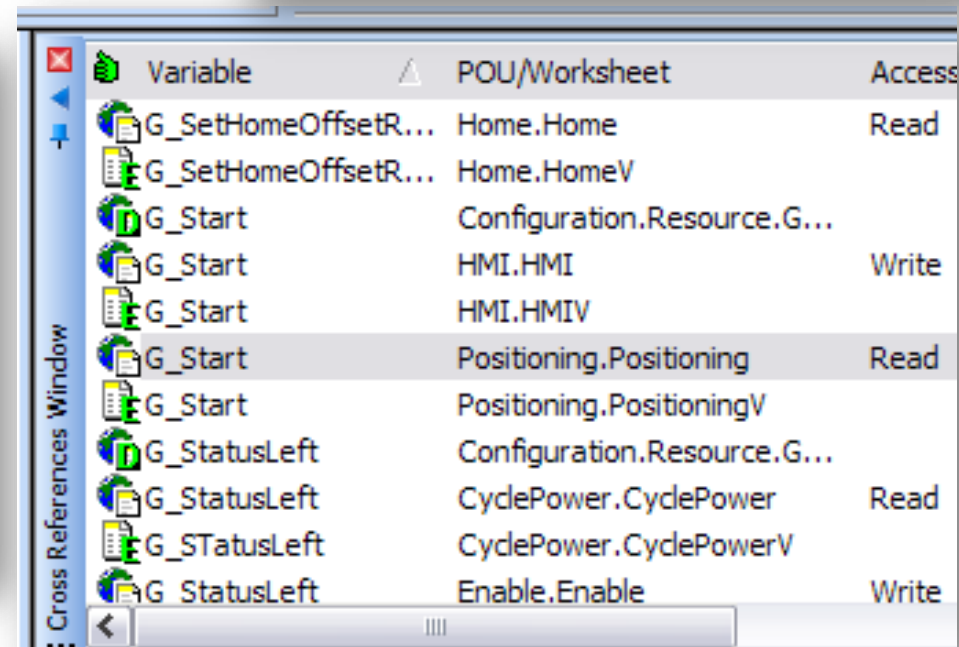
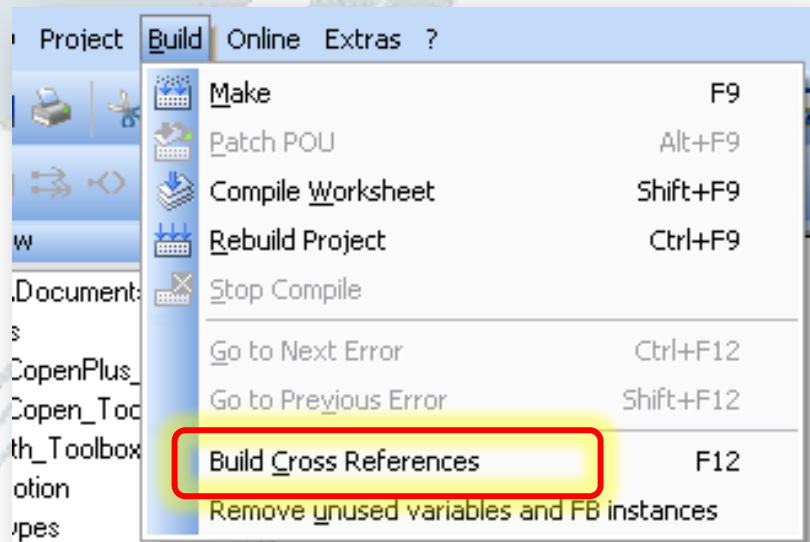
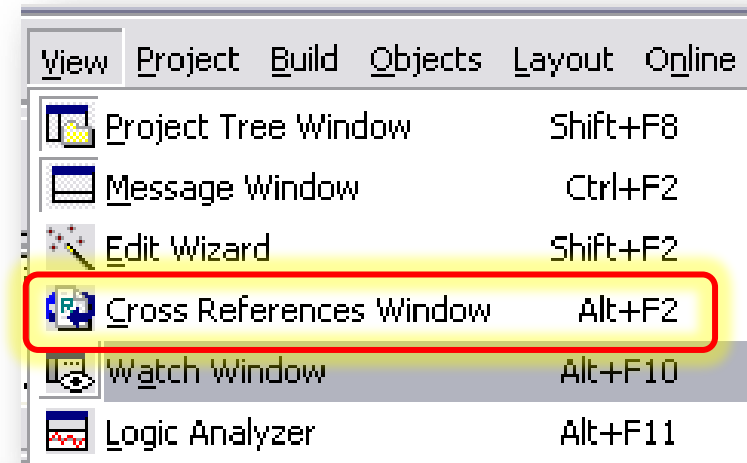
2

(*Leave gate up during open hours and when garage i

Watch Window

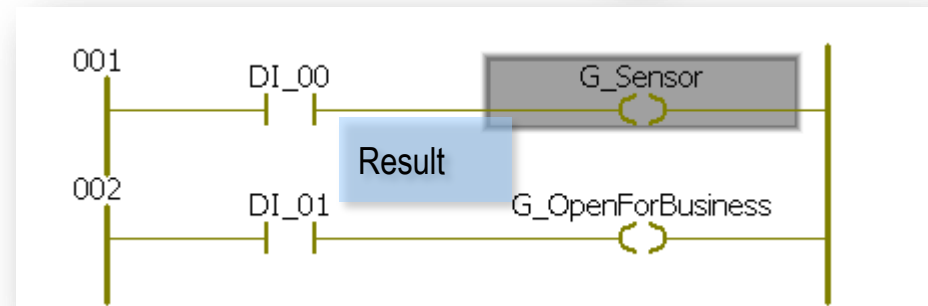
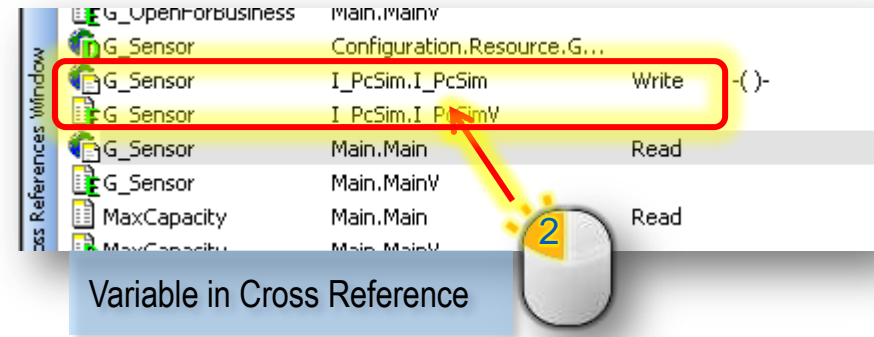
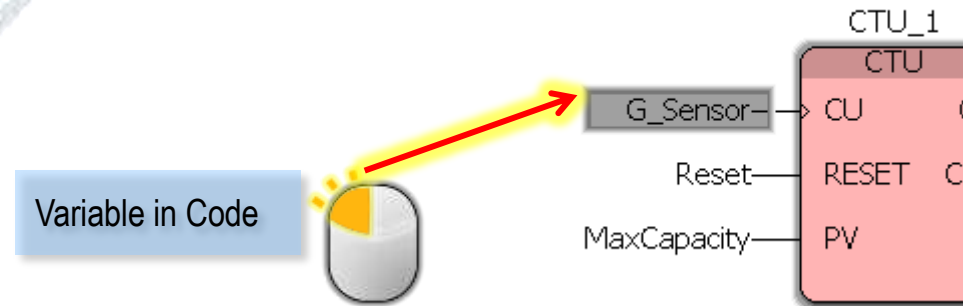
Variable	Value	Default value	Typ
NumberOfCars	1.00		LRE
G_NumberOf...	1		INT
Income	29.99		LRE
TaxRate	0.08		LRE
Tax	2.40		LRE
G_Sensor	TRUE		BOC
MaxCapacity	10		INT
DI_01	TRUE		BOC
DI_00	TRUE		BOC

- *Display Cross Reference*
 - *View – Cross References Window*
- *Build Cross Reference*
 - *Build – Build Cross References*



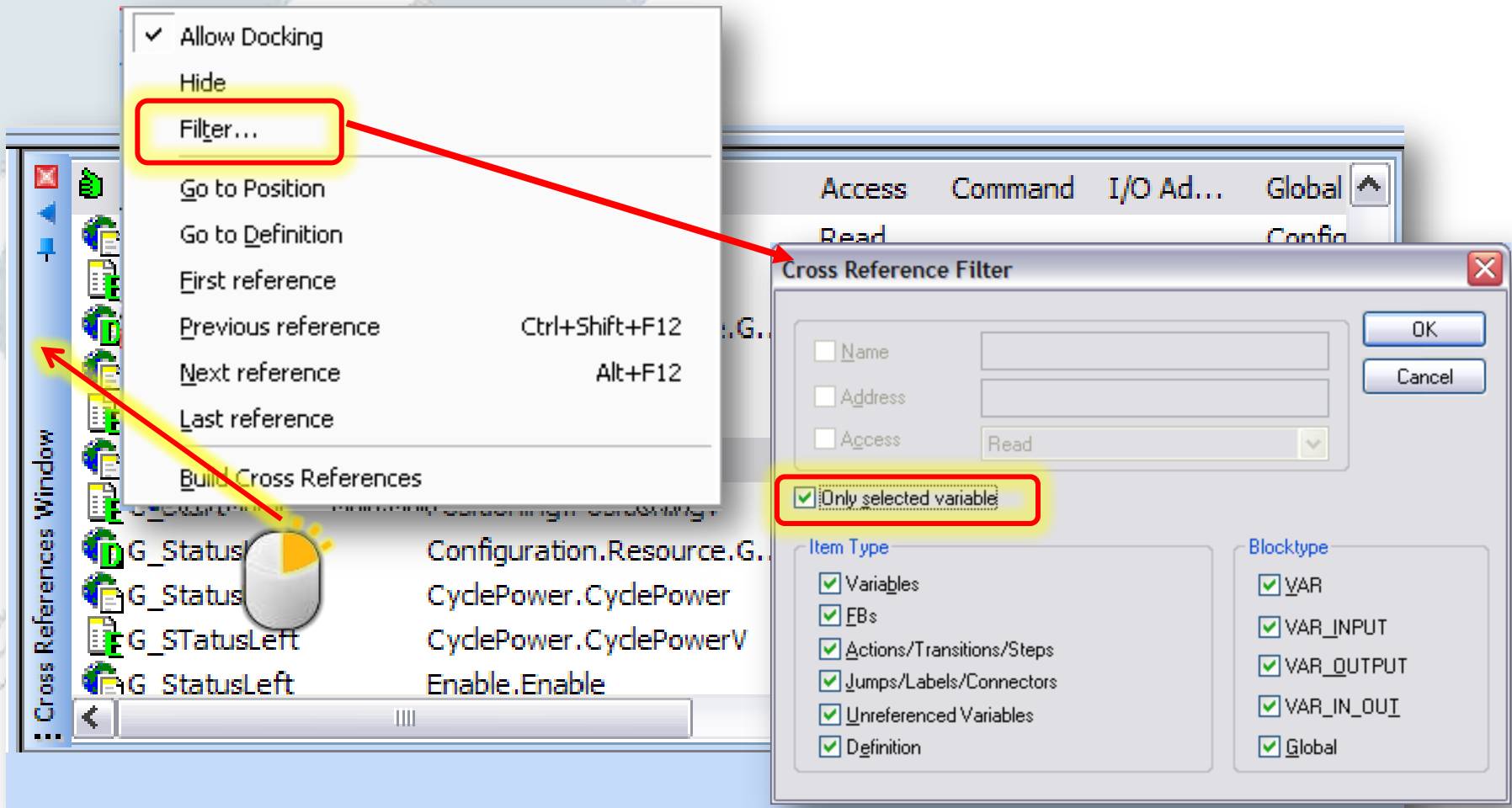
■ *Navigation*

- *Click on variable in code*
 - » *Cross reference window updates*
 - » *Read or Write access indicated*
- *Double-Click to open variable from Cross Reference*
 - » *Code worksheet opens*



- *Filter*

- *Right-Click (title bar) – Filter*



IEC 61131-3 Programming Overview

Standard
Languages
Data Types
Software Model
Task Execution

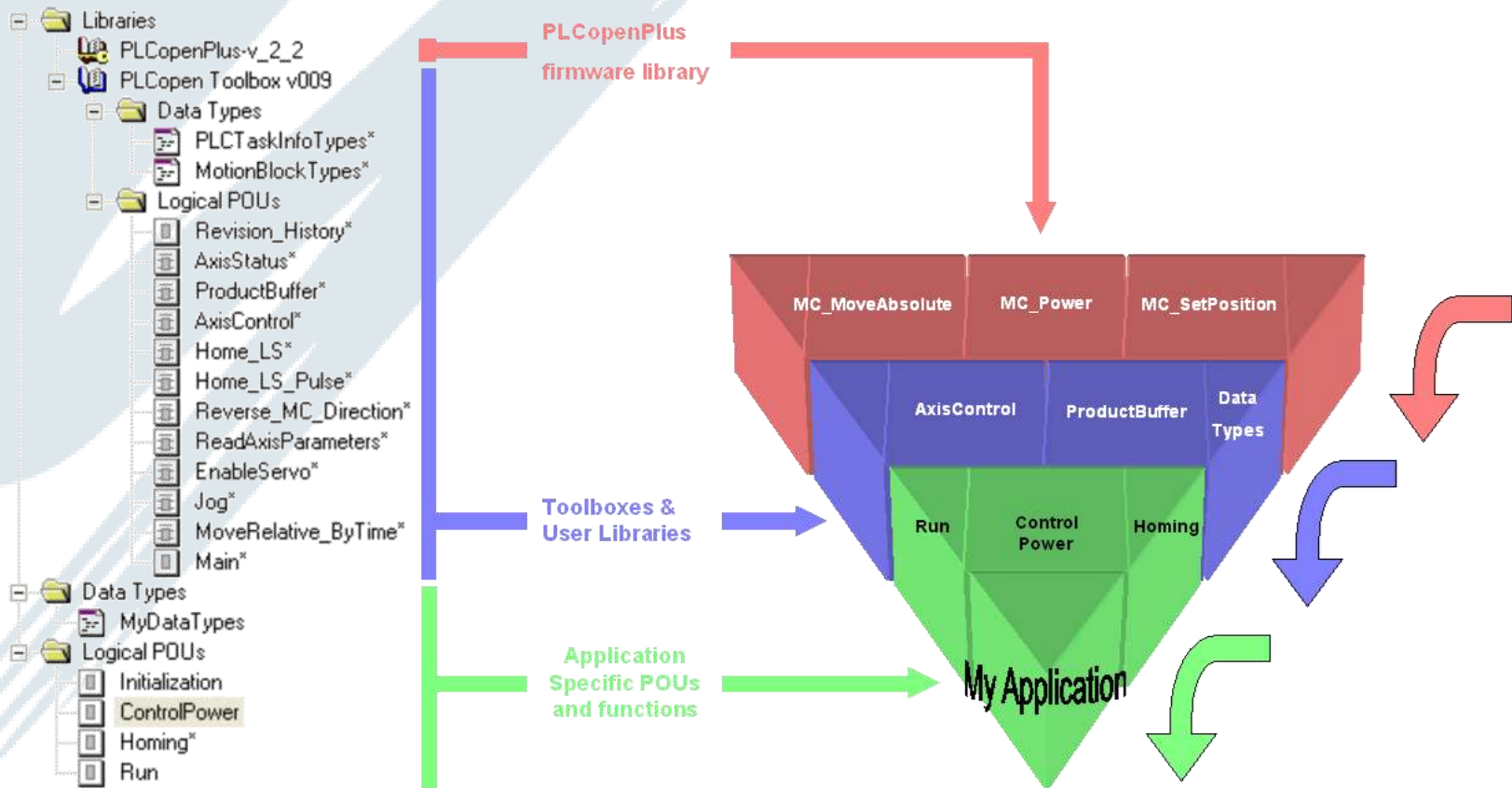
TASK EXECUTION
SOFTWARE MODEL

- *IEC = International Electrotechnical Commission*
 - *World standards organization*
 - *Founded in 1906*
 - *International Electrical and Electronic Standards*
 - *50+ Participating countries*



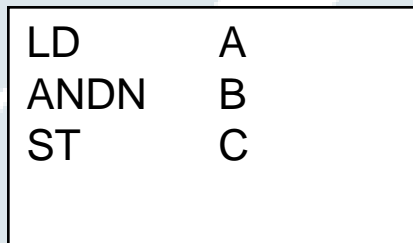


- *Yaskawa Tech Note: TN.MCD.08.130*

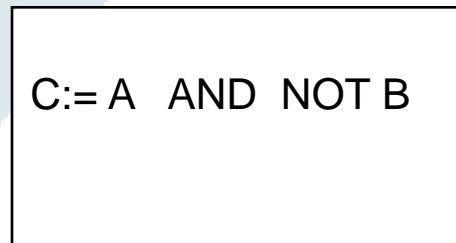


All Functions, Function Blocks, and DataTypes from the libraries are available for the application.

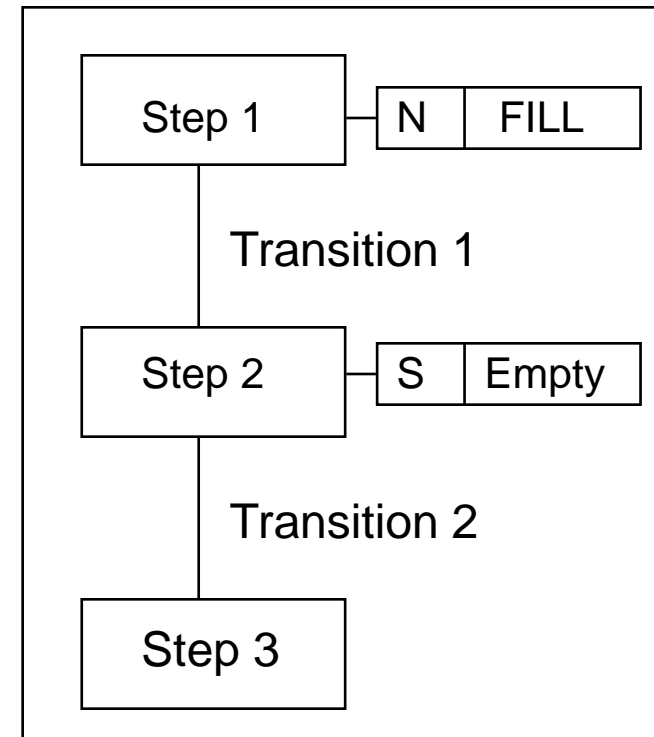
Instruction List (IL)



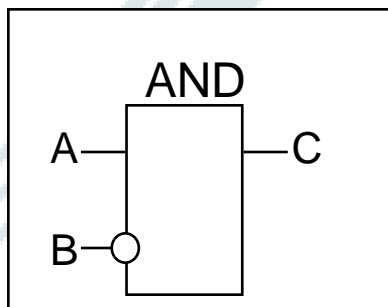
Structured Text (ST)



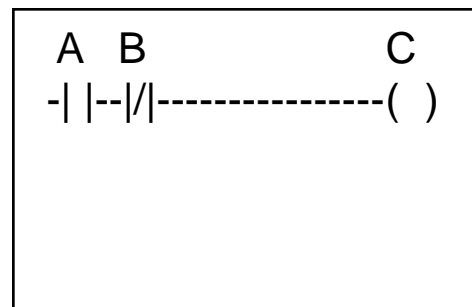
Sequential Function Chart (SFC)



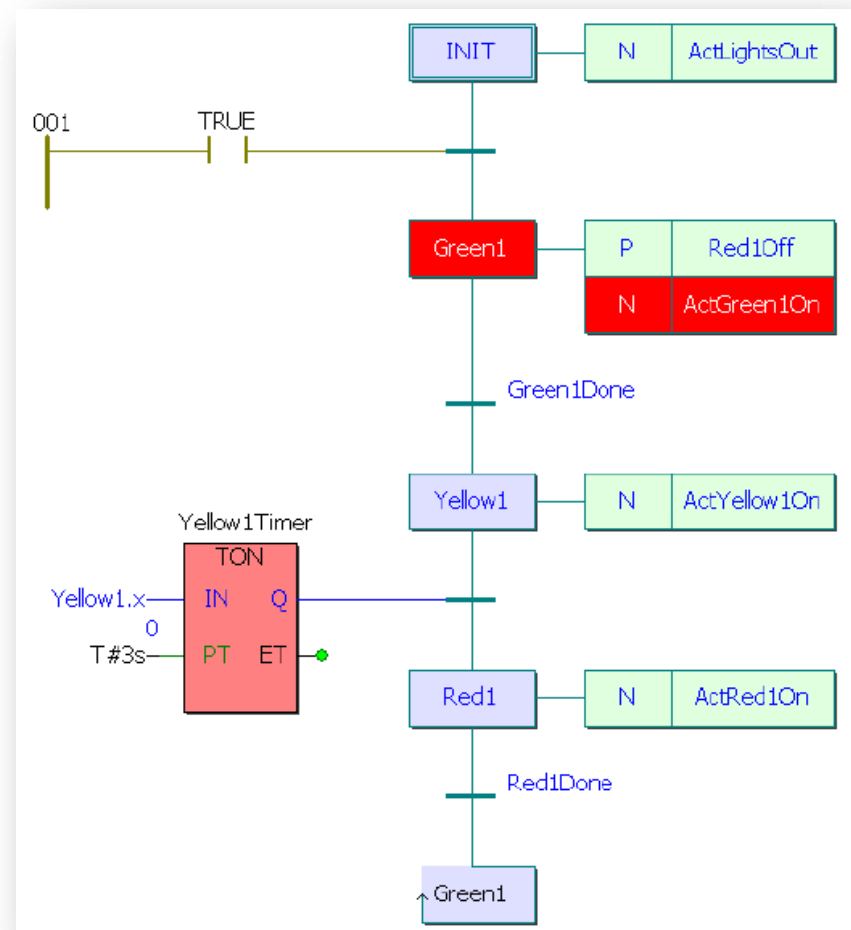
Function Block Diagram (FBD)



Ladder Diagram (LD)

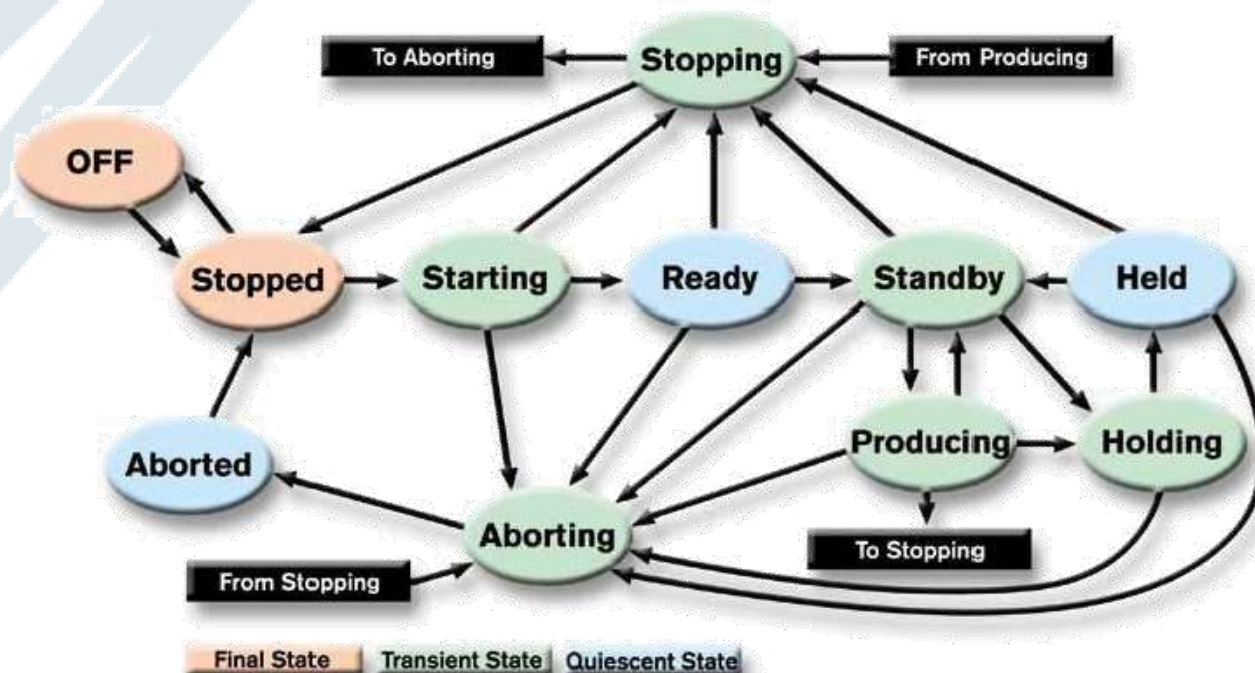


- A graphical programming language to describe control sequences in graphical form
- A tool for “top-down” analysis and representation of a control sequence
- Shows the main states of a program










Traffic Light Example

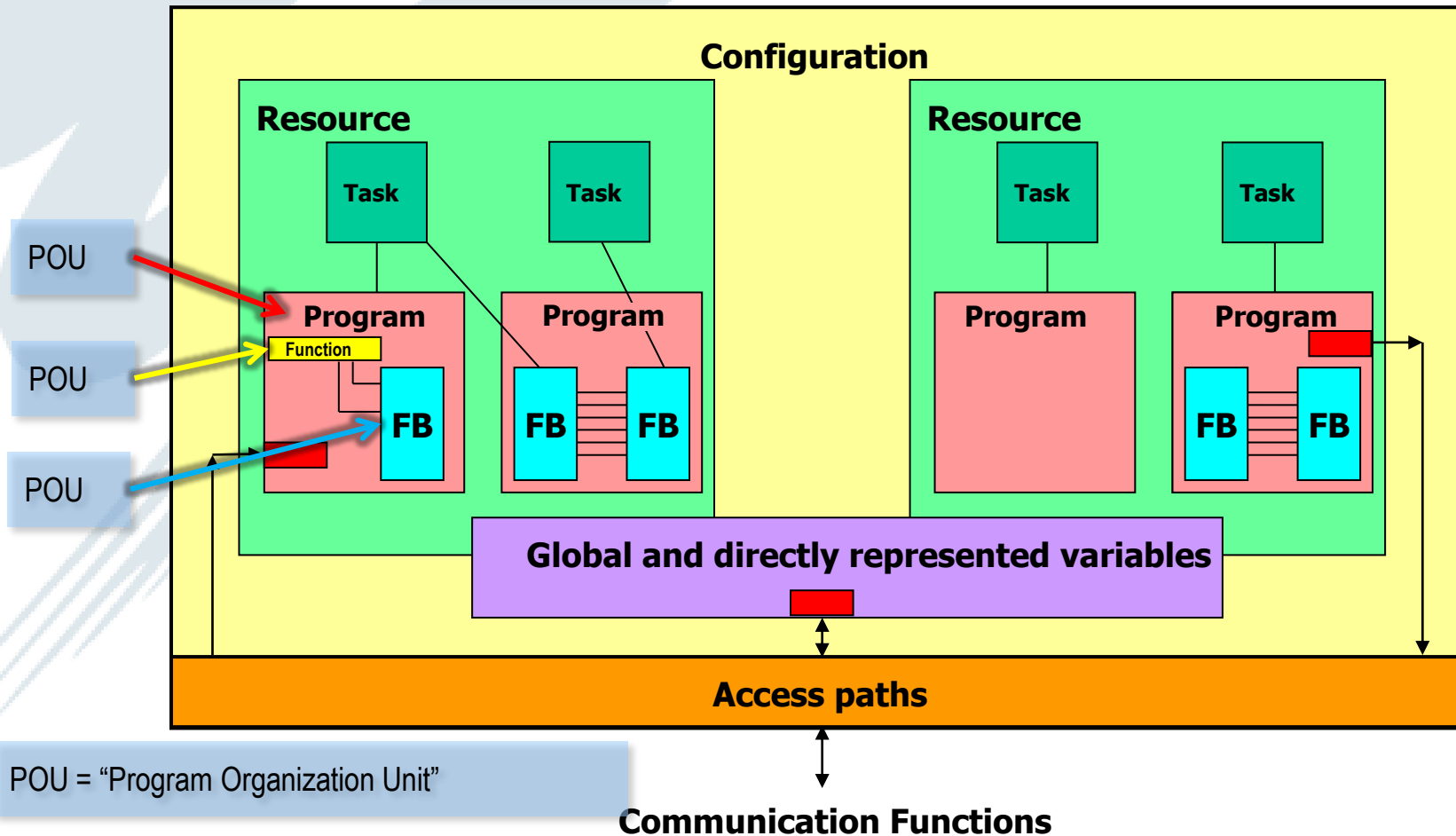
- When there is a definite sequence for the machine to follow
- When it is advantageous to think of the machine operation as different states of operation
- When someone who is not familiar with your code will have to work with it. Look at SFC and quickly understand how the machine works



- Data Types standard with IEC 61131-3*

No.	Keyword	Data Type	Bits
1	BOOL 	Boolean	1
2	SINT	Short integer	8
3	INT 	Integer	16
4	DINT	Double integer	32
5	LINT	Long integer	64
6	USINT	Unsigned short integer	8
7	UINT 	Unsigned integer	16
8	UDINT	Unsigned double integer	32
9	ULINT	Unsigned long integer	64
10	REAL	Real numbers	32
11	LREAL 	Long reals	64
12	TIME 	Duration	32
13	DATE	Date (only)	16
14	TIME_OF_DAY or TOD	Time of day (only)	32
15	DATE_AND_TIME or DT	Date and time of day	64
16	STRING	Character string	8
17	BYTE 	Bit string of length 8	8
18	WORD 	Bit string of length 16	16
19	DWORD	Bit string of length 32	32
20	LWORD	Bit string of length 64	64

- IEC hardware must operate according to this standard*



Configuration



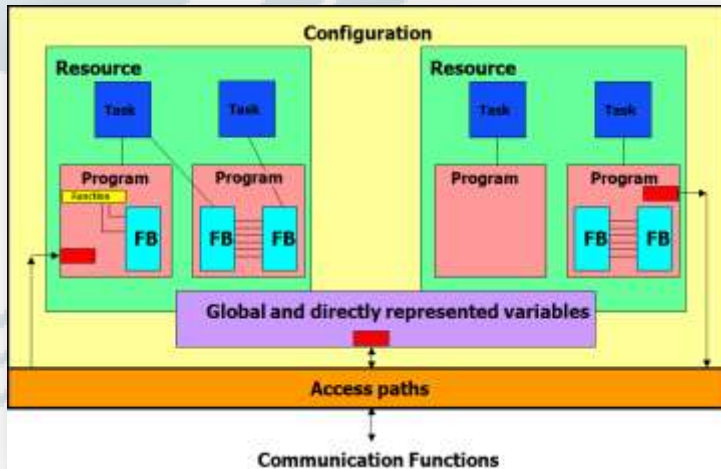
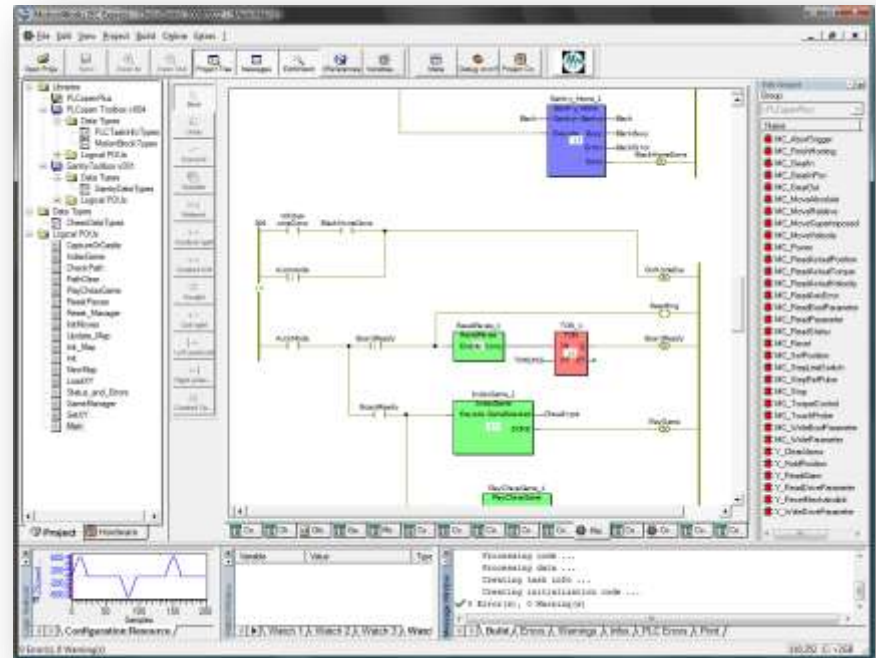
MotionWorks IEC does not yet support an entire configuration with multiple controller resources

Configuration

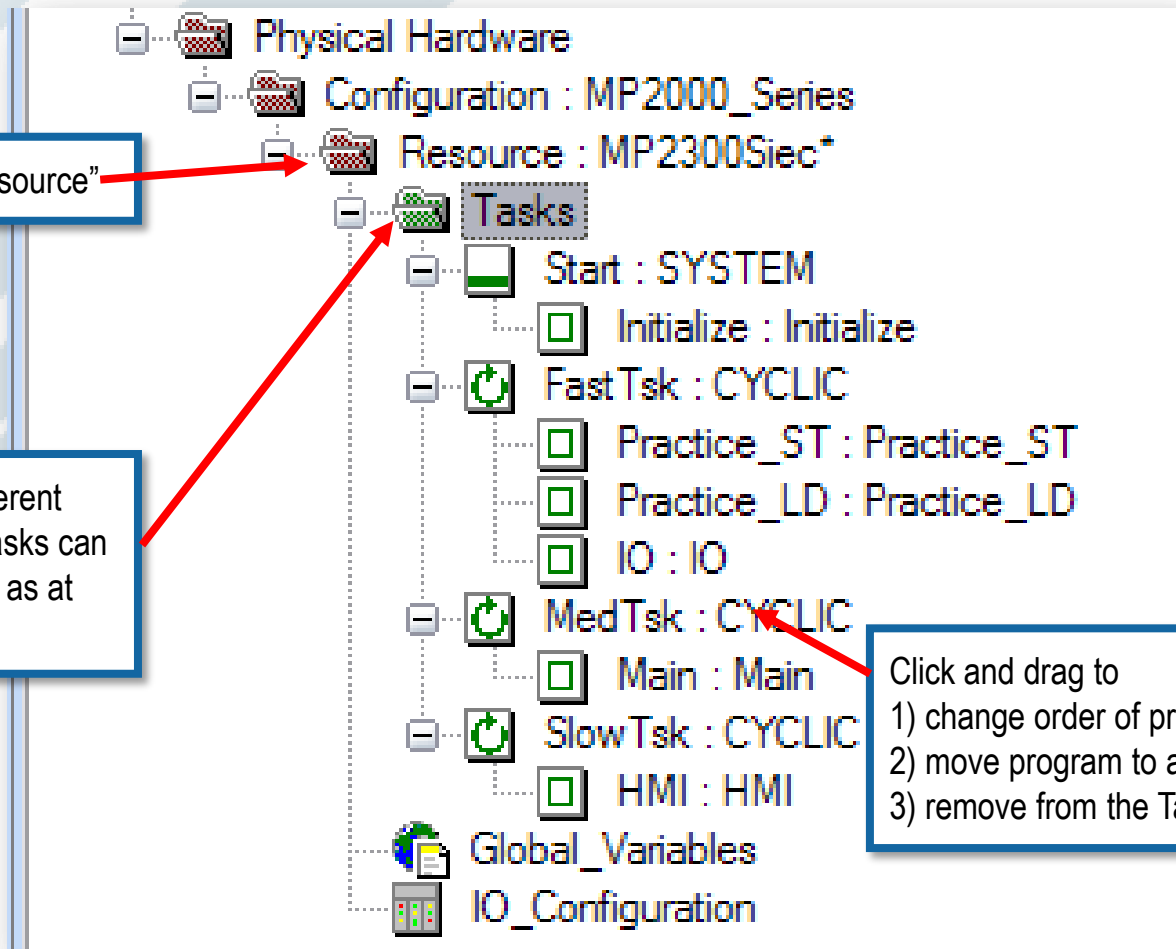
Resource



- Controller “Resource” must execute code according to IEC standards



- Task Details*



The controller is the "Resource"

Multiple tasks run at different update rates. System tasks can be set to run once, such as at WarmStart

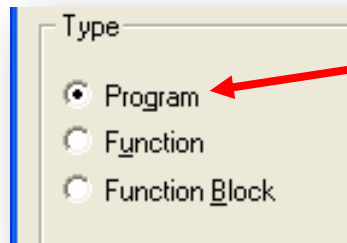
Click and drag to

- 1) change order of program execution
- 2) move program to another task
- 3) remove from the Task list

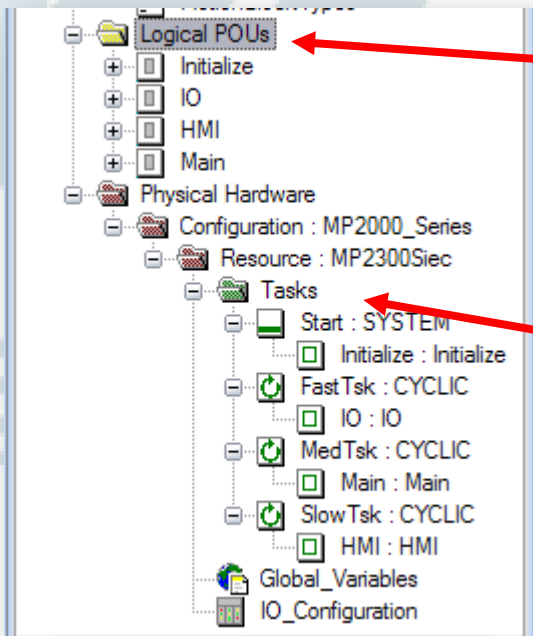
- *Program Organization Unit (POU)*

- *3 types of POU*

- » *Program*
 - » *Function*
 - » *Function Block*

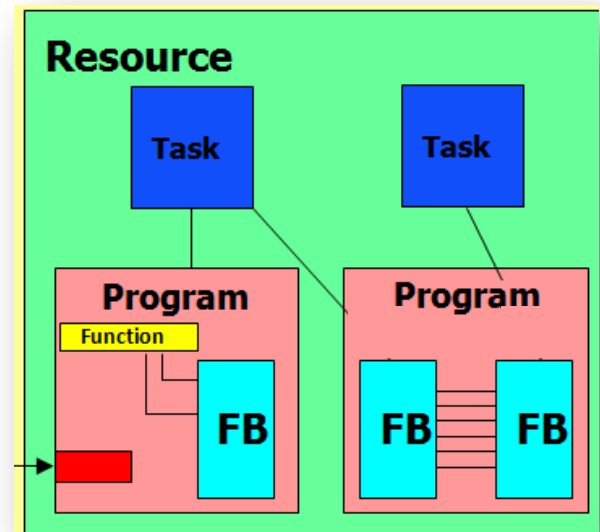


Only Program POUs can run in a task. Functions and Function Blocks run in a program.



Logical POUs contain the code pages

Physical Hardware shows where the **Program POUs** run

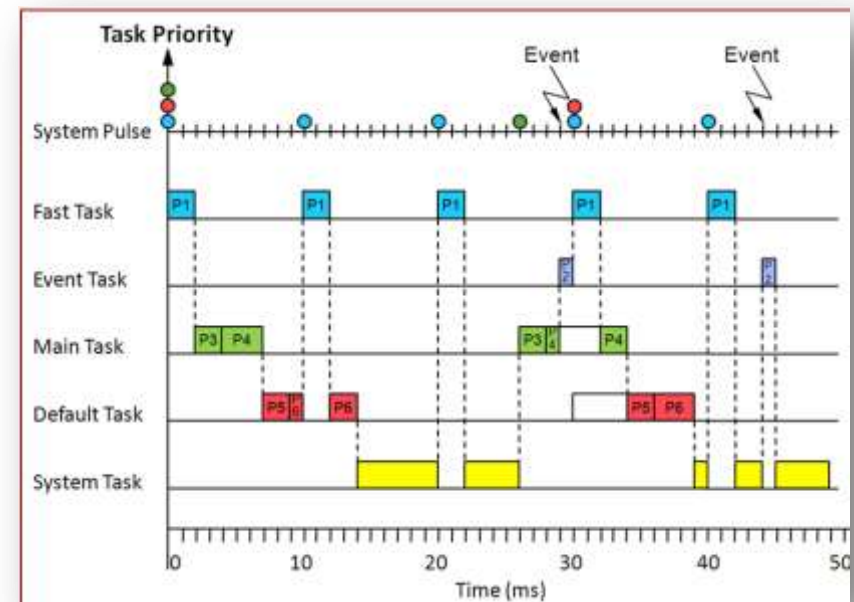


Multi Task

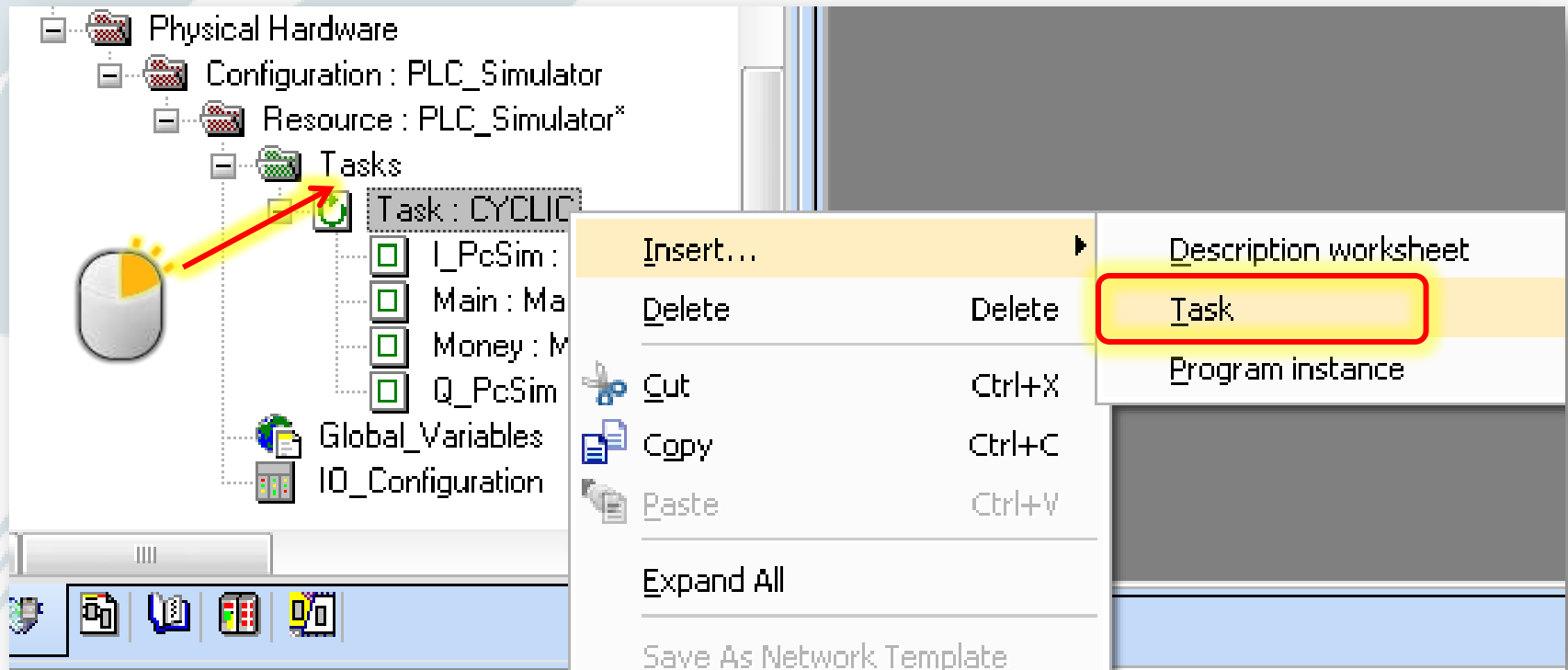
New Task
Task Types
System Task, Default Task
Cyclic Task
Slow Task
Task Priority
Watchdog
Task Properties/Settings

Task Properties/Settings
Watchdog

- **4 Task Types**
 - System, cyclic, default, event
- **Up to 16 Tasks**
 - Yaskawa implementation
- **Task Interval for Cyclic Tasks**
 - Lowest interval depends on PLC
 - » Simulator ~ 10 ms
 - » MP2000iec ~ 1 ms
 - » MP3000iec ~ 0.125 ms
- **Task Priority for Cyclic Tasks**
 - 0 = highest priority (runs first)
 - 15 = lowest priority (runs last)
- **Task Execution**
 - Read Inputs %I
 - Program POU
 - Write outputs %Q



- *Class Project*
- *Right-Click Task Folder*
- *Insert → Task*





Default

- Executes after all other tasks complete



Cyclic

- Repeats at given "interval".
 - » Slow interval for non-critical tasks.
 - » Fast interval for machine IO and motion
- Priority
 - » Which cyclic task is done 1st, 2nd, 3rd, etc



Event

- Executes 1 time when condition is met



System

- Executes 1 time when the system reaches a given state
 - » Warm Start
 - » Cold Start

Insert

Name:

Program type:

Task type:

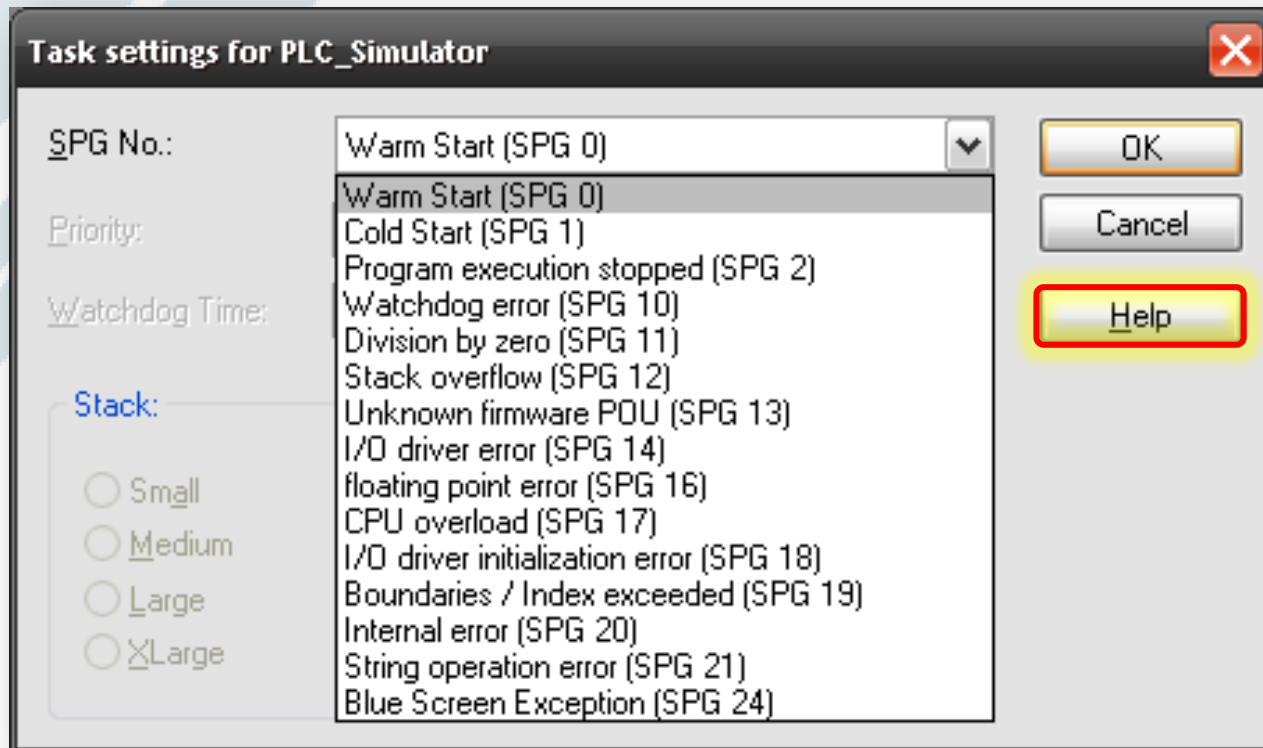
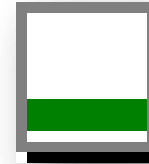
Configuration
 Resource
 Task
 Program instance
 Description
 Variables
 FB instance

Mode:

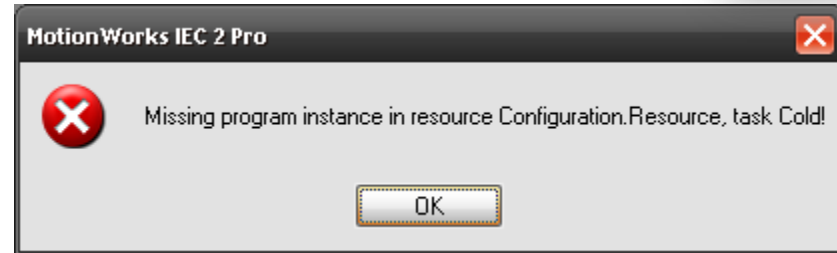
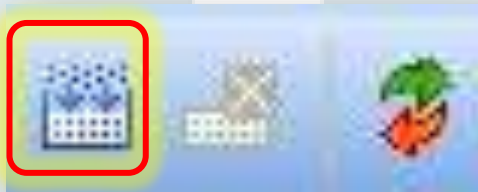
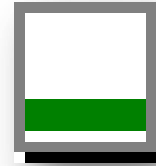
Insert
 Append

Exclude from compilation

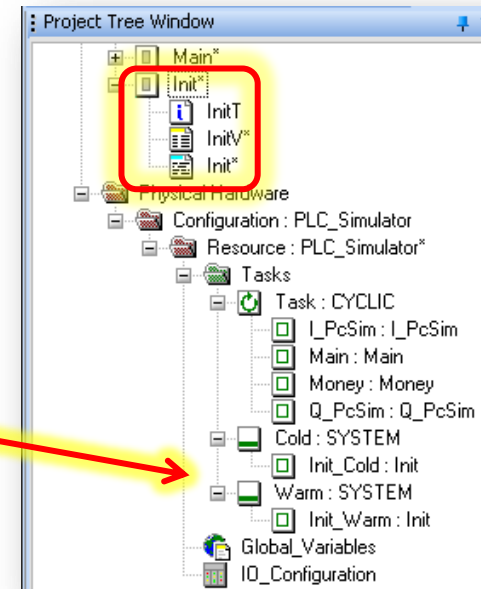
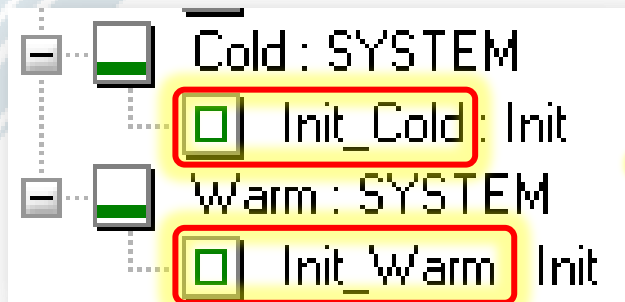
- Use system tasks to execute a program POU triggered by specific system conditions
 - Warm Start
 - Cold Start
 - Errors



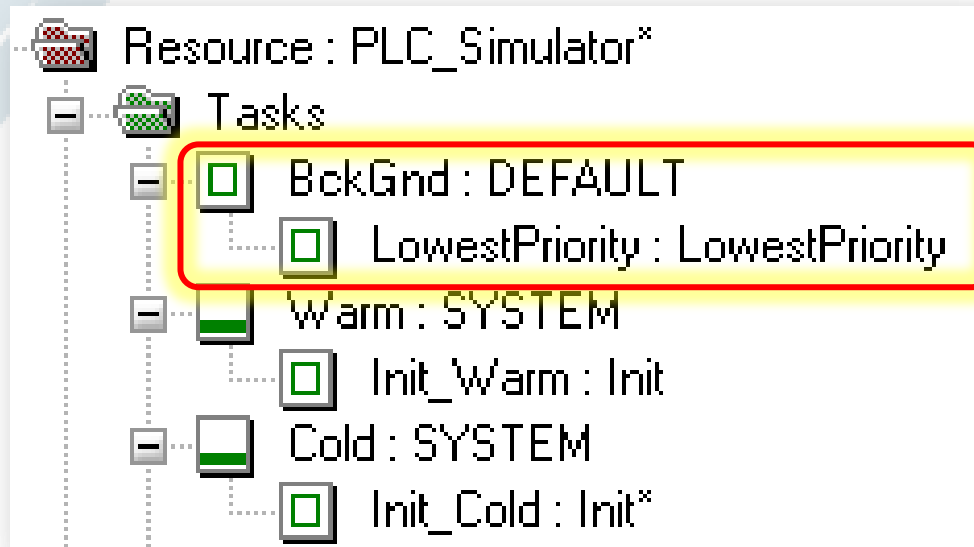
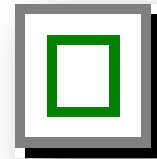
- Create a warm start system task named “warm”
- Create a cold start system task named “cold”



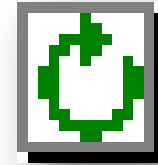
- Create a blank ST POU named “Init”
 - Insert program instance to Cold task
 - Insert program instance to Warm task



- Create a Default task named “BckGnd”
 - 500ms watchdog
- Create a blank ST program POU named “LowestPriority”
 - Insert program instance in BckGnd task



- Create a cyclic task named “Slow”
- Why have another cyclic task?
 - Optimize code execution
 - Critical POU's in fast tasks (ex: motion & machine IO)
 - Non-time-critical POU's in slow tasks (ex: user interface)



Insert [Close]

Name:

Program type:

Task type:

Exclude from compilation

Type

Configuration

Resource

Task

Program instance

Description

Variables

FB instance

Mode:

Insert

Append

OK

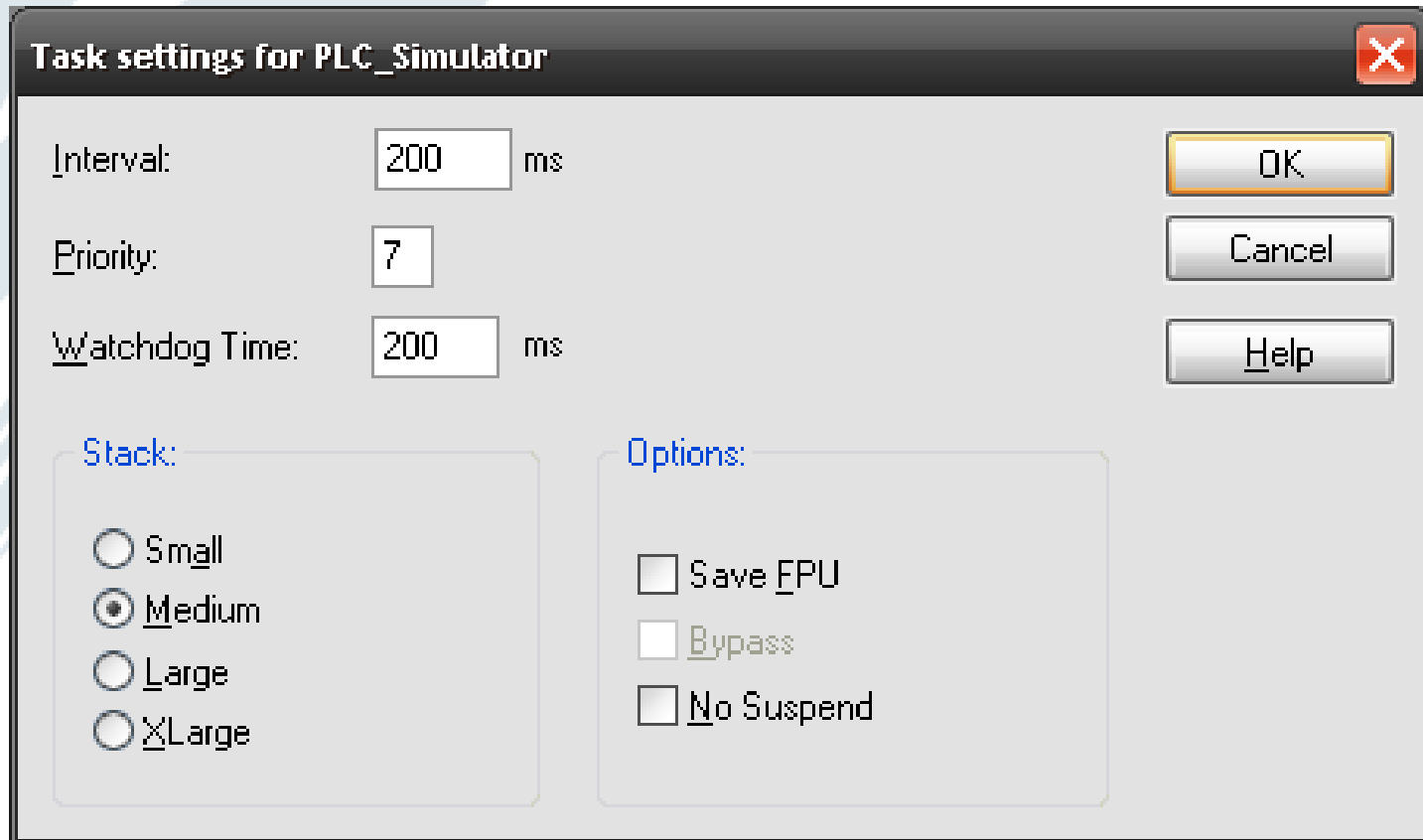
Cancel

Help

- *Set interval time = 200 ms*
- *Watchdog Time = Interval*
- *Priority = 7*

Why Priority 7?

To leave room for higher priority tasks in the future (“medium” etc)



Task settings for PLC_Simulator

Interval: ms

Priority:

Watchdog Time: ms

OK

Cancel

Help

Stack:

Small

Medium

Large

XLarge

Options:

Save FPU

Bypass

No Suspend

- *Tasks are executed sequentially in order of priority*
 - *0 = first, 15 = last*
 - » *Software does allow entry up to 31, but don't do it! Controller internally executes hidden tasks beyond priority 15. You want your code to execute before these, so keep it to 15 and below*
- *Similar to prioritizing office work*
 1. *Phone Calls*
 2. *Emails*
 3. *Daily Report*
 4. *Weekly Report*
 5. *Monthly Report*
 6. *Yearly Goals*



You wouldn't work first on the yearly goal while refusing to answer phone calls and emails!

- *Maximum time allowed for task to complete*
 - *Use same setting as Interval*

Task settings for PLC_Simulator

Interval: ms

Priority:

Watchdog Time: ms

If the task does not complete in time, the PLC will stop with Watchdog error

Resource

State: Stop

Stop Cold

Reset Warm

Hot

Download Upload

Error Info

Close Help

```

Message Window
[Error] Watchdog exceeded in task '3'!
[Error] Exception detected in usertask '3'!
[Error] Watchdog exceeded in task '2'!
[Error] Exception detected in usertask '2'!
[Error] Watchdog exceeded in task '1'!
[Error] Exception detected in usertask '1'!
[Error] CPU load: 100%!
[Error] PLC stopped!
Build Errors Warnings Infos PLC Errors
  
```


- *What are the settings of the existing “task”?*
 - *Right click, settings*
 - *Change to 20 ms*

The image shows a software interface for configuring tasks. On the left, a tree view displays the project structure: Physical Hardware > Configuration : PLC_Simulator > Resource : PLC_Simulator > Tasks. A right-click context menu is open over the 'Task : CYCLIC' entry, with the 'Settings...' option highlighted. A red arrow points from the mouse cursor to this task. On the right, the 'Task settings for PLC_Simulator' dialog is shown. It contains three input fields: 'Interval' set to 20 ms, 'Priority' set to 0, and 'Watchdog Time' set to 20 ms. The 'Interval' field is highlighted with a red box. Below these fields, there is a 'Stack' section with radio buttons for 'Small', 'Medium' (selected), 'Large', and 'XLarge'.

- *Change name to “Fast”*

The screenshot shows a software interface with a task tree on the left and a properties dialog on the right. The task tree is expanded to show a 'Task : CYCLIC' item, which is highlighted. A red arrow points from a mouse cursor to this task. A context menu is open over the task, with 'Properties...' selected. The properties dialog is titled 'Fast' and has a 'Name' field containing 'Fast' and a 'Description' field. The 'Name' field is highlighted with a red box.

Physical Hardware

- Configuration : PLC_Simulator
 - Resource : PLC_Simulator*
 - Tasks
 - Slow : CYCLIC
 - Task : CYCLIC
 - I_PcSim : I_Pc...
 - Main : Main
 - Money : Money
 - Q_PcSim : Q_P...
 - Cold : SYSTEM
 - Init_Cold : Init...
 - Warm : SYSTEM
 - Init_Warm : Init...

Context Menu:

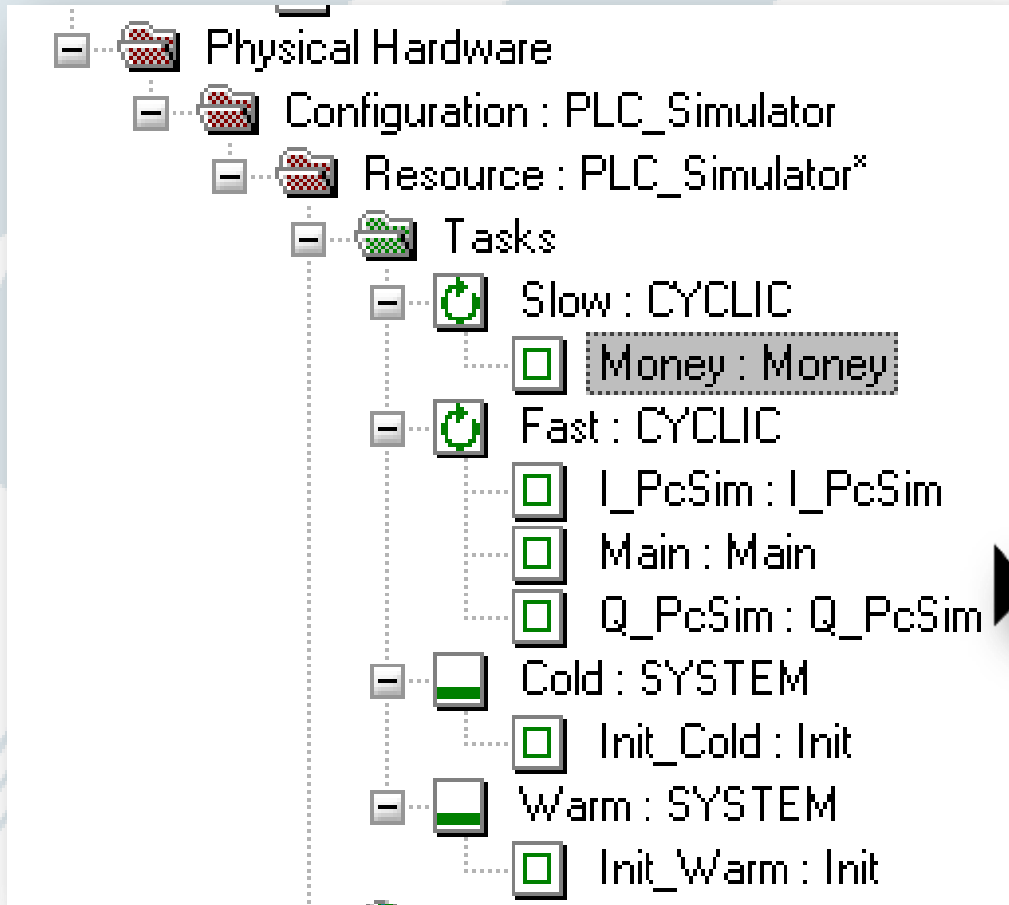
- Insert...
- Delete Delete
- Cut Ctrl+X
- Copy Ctrl+C
- Paste Ctrl+V
- Expand All
- Save As Network Template
- Define Placeholders
- Properties...
- Settings...

Properties Dialog:

Name	Type
Name:	Fast
Description:	

OK

- *Click and drag to move*

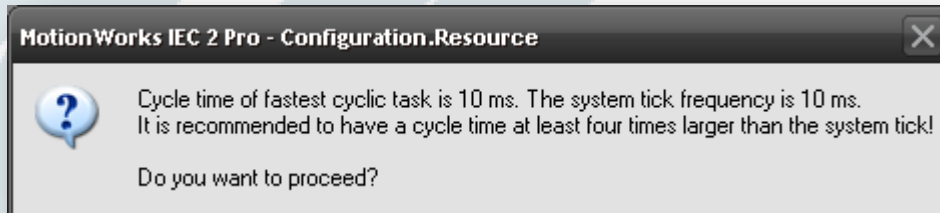


What is the Slow task cycle?

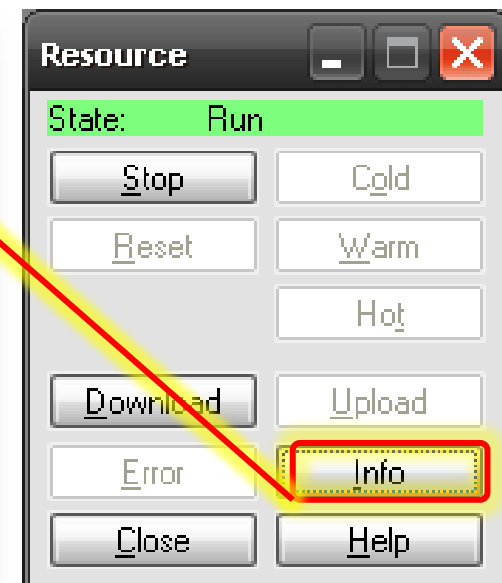
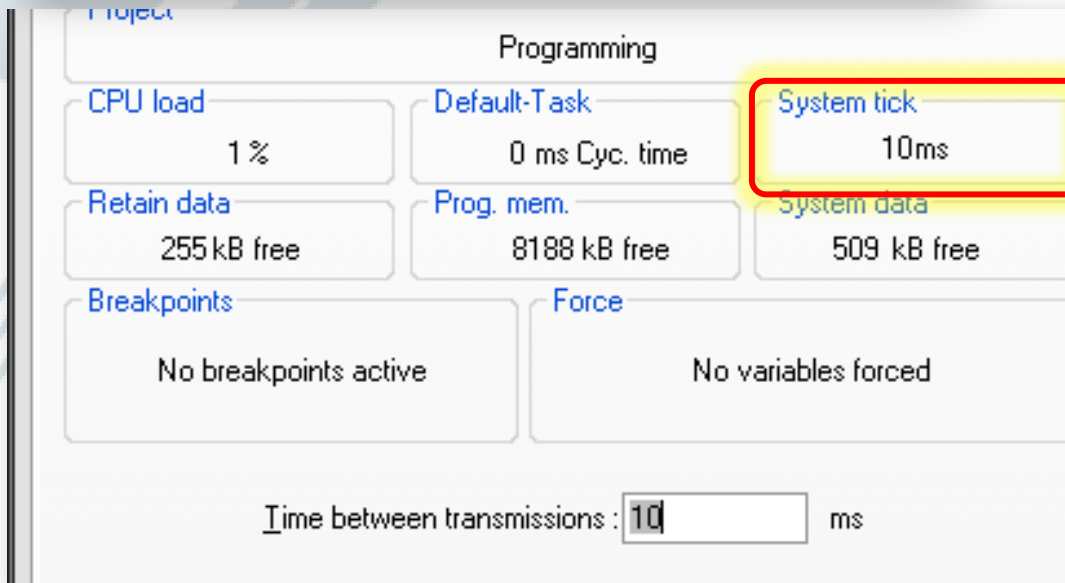
What is the Fast task cycle?

Which POU can be executed at slower cycle?

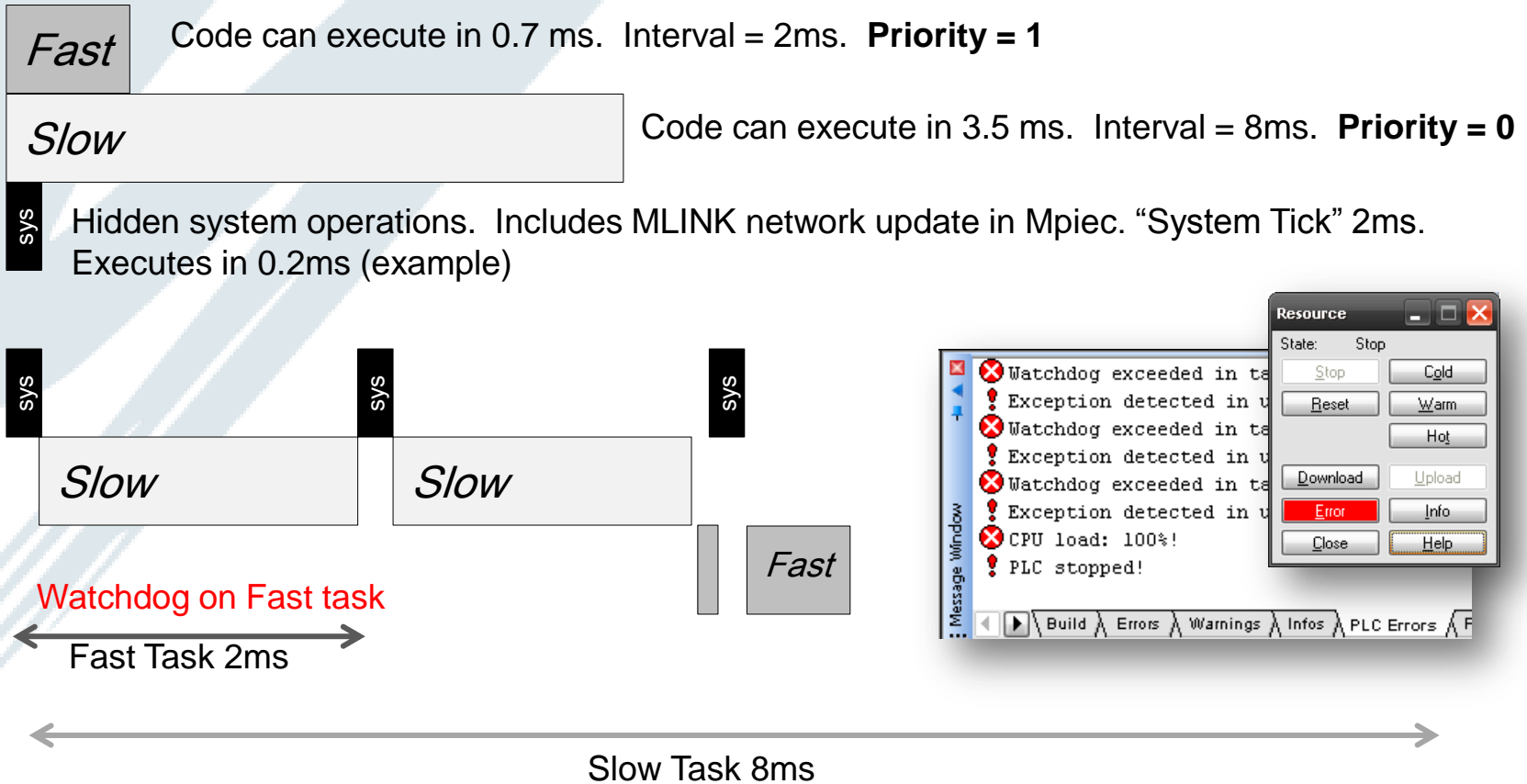
- *Fastest possible task interval*
- *Set task interval as integer multiple of system tick*
- *MPiec System Tick = Mechatrolink network update*



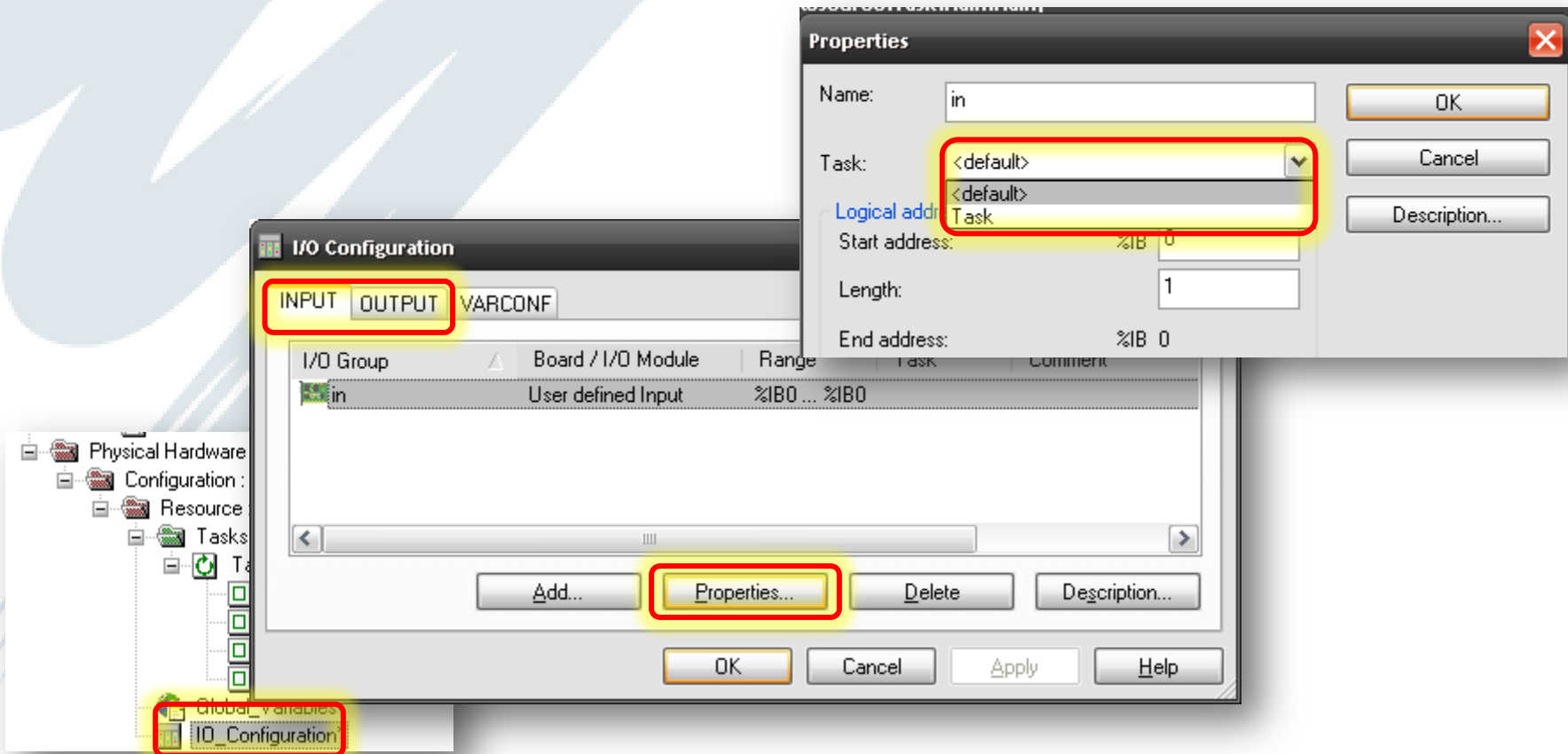
OK to click accept



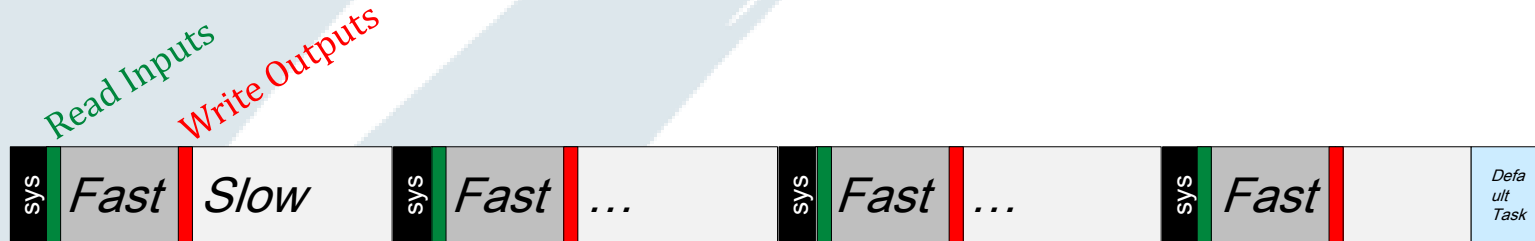
- What happens with incorrect priority?



- Assign IO to the task in which they are used
 - Physical Hardware – IO_Configuration: Properties - task

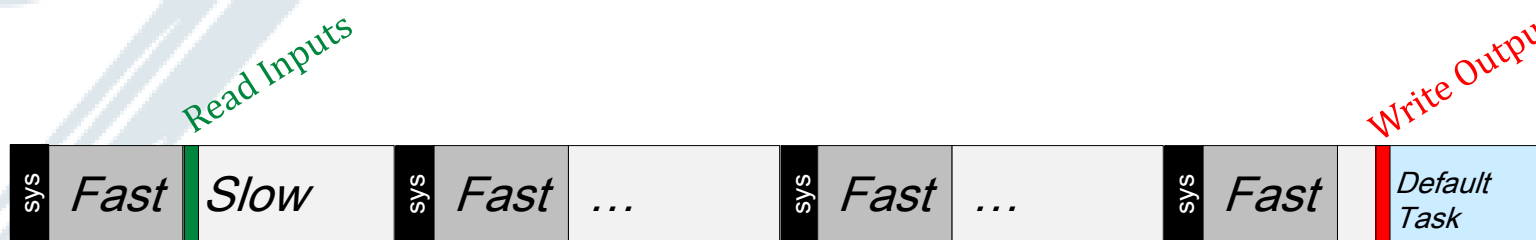


- IO configuration assigned to **Fast** task



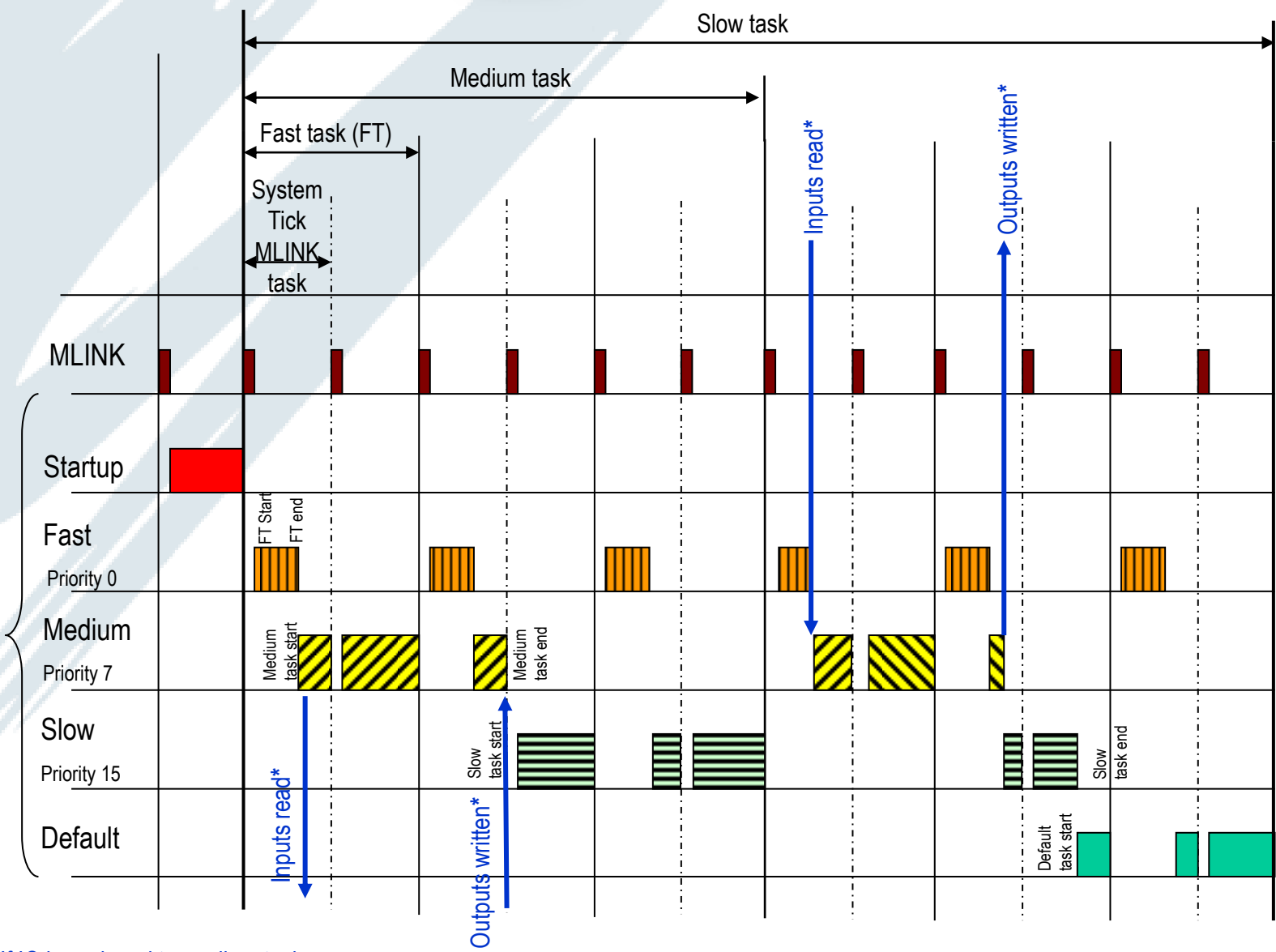
What if an input changes state rapidly?

- IO configuration assigned to **Slow** task



Inputs not read until task execution begins

IO update takes processor time



* If IO is assigned to medium task

Logic Analyzer

Overview

Add Variables

Trigger Setup

Zoom, Cursors

Zoom, Cursors

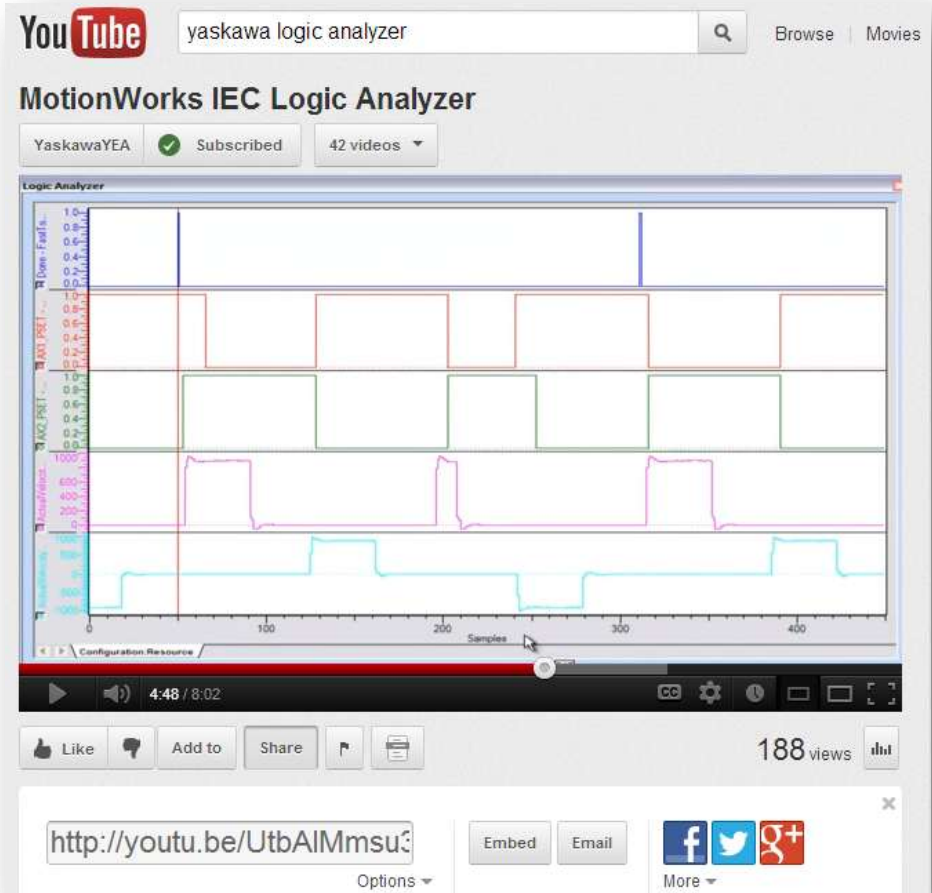
Trigger Setup

- **Quick Reference Guide**

- **Document:**
[QRG.MP2000iecSeries.01](#)
- **Search Yaskawa.com:** [QRG](#)

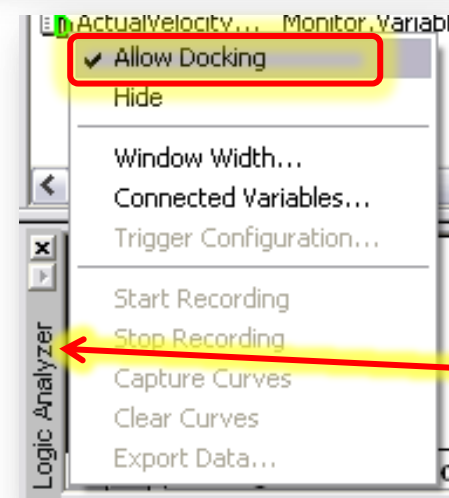
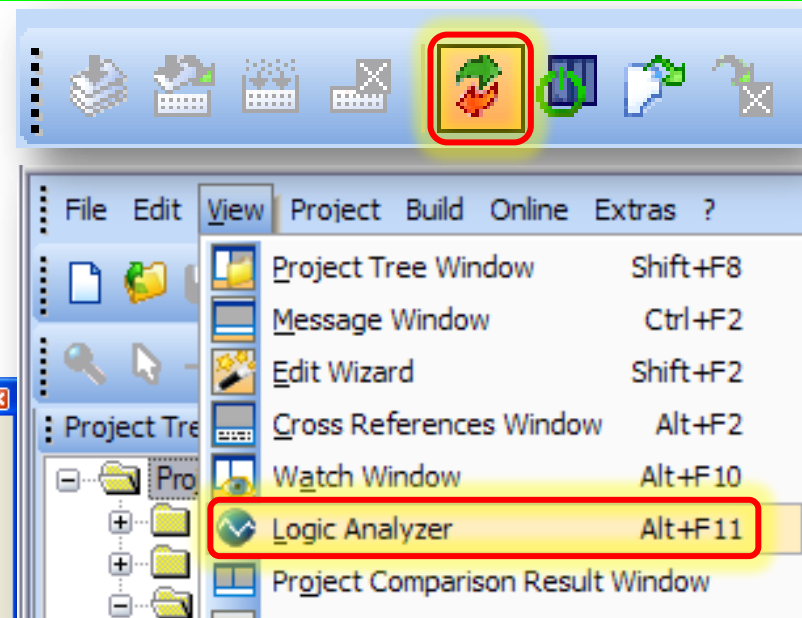
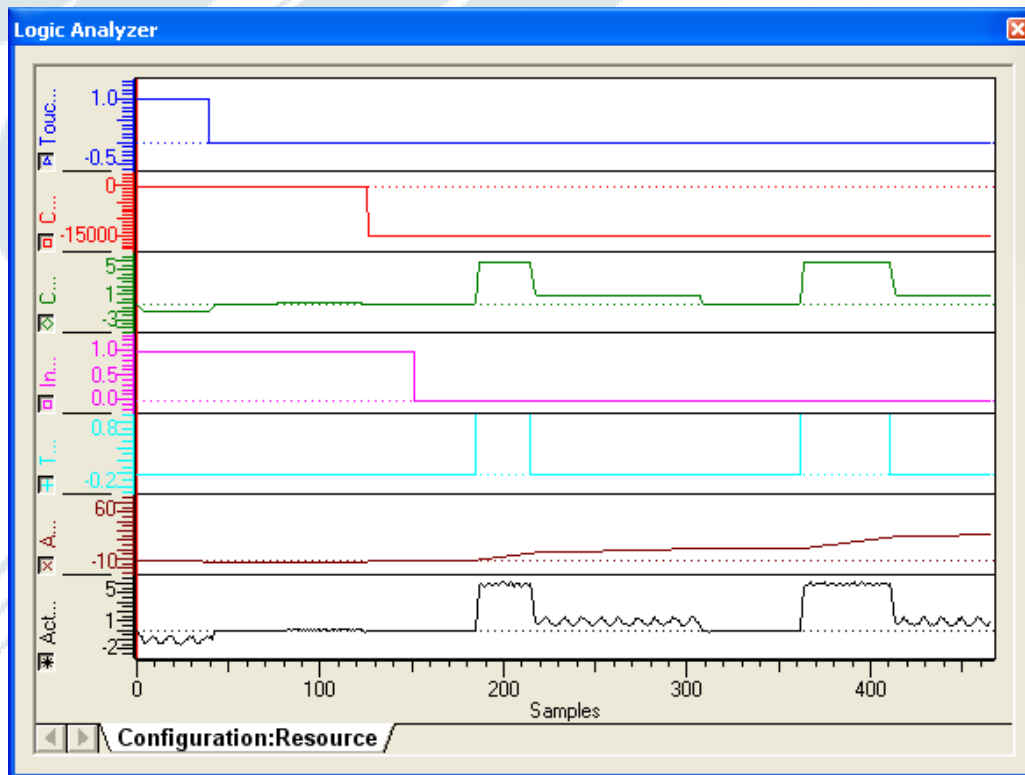
- **eLearning Video on Yaskawa.com**

- **Document:**
[eLV.MotionWorksIEC.01.LogicAnalyzer](#)
- **YouTube Video**
 - **Search:** [Yaskawa Logic Analyzer](#)

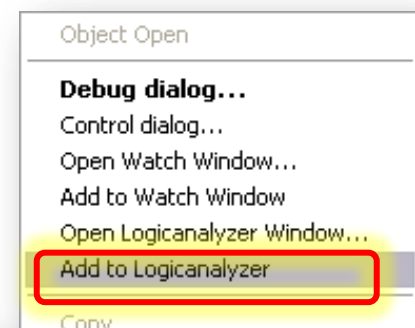
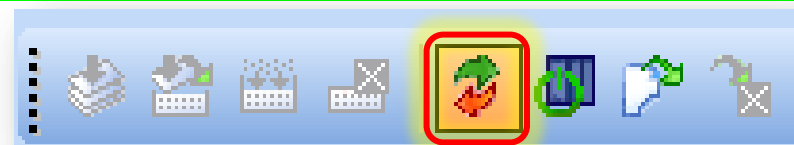


- *Features*

- 16 variables
- Sample at task update rate
- With MotionWorks IEC open
- Debug mode

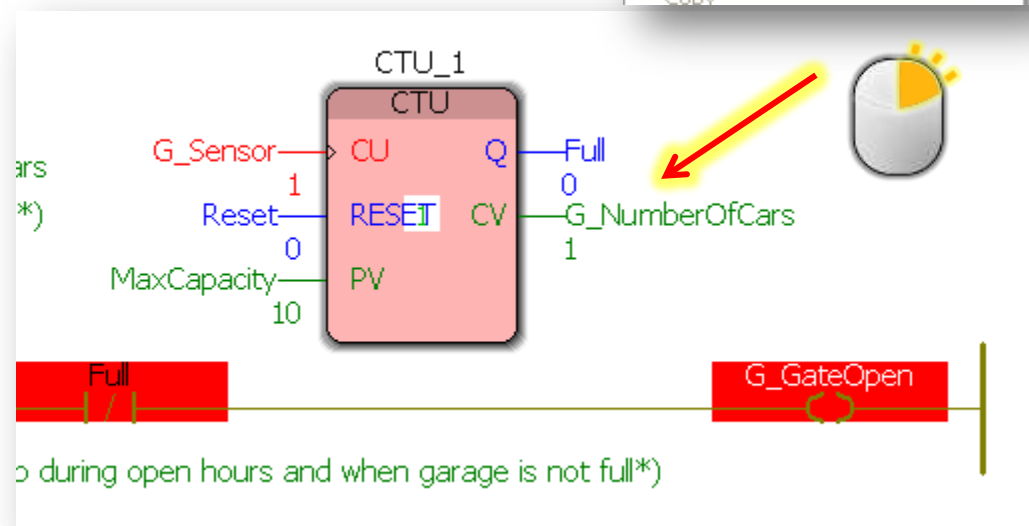


- *Add variables to Logic Analyzer*
 - *Debug ON*
 - *Right-Click Variable*
 - » *From Code*
 - » *From List*
 - *Add to Logicanalyzer*

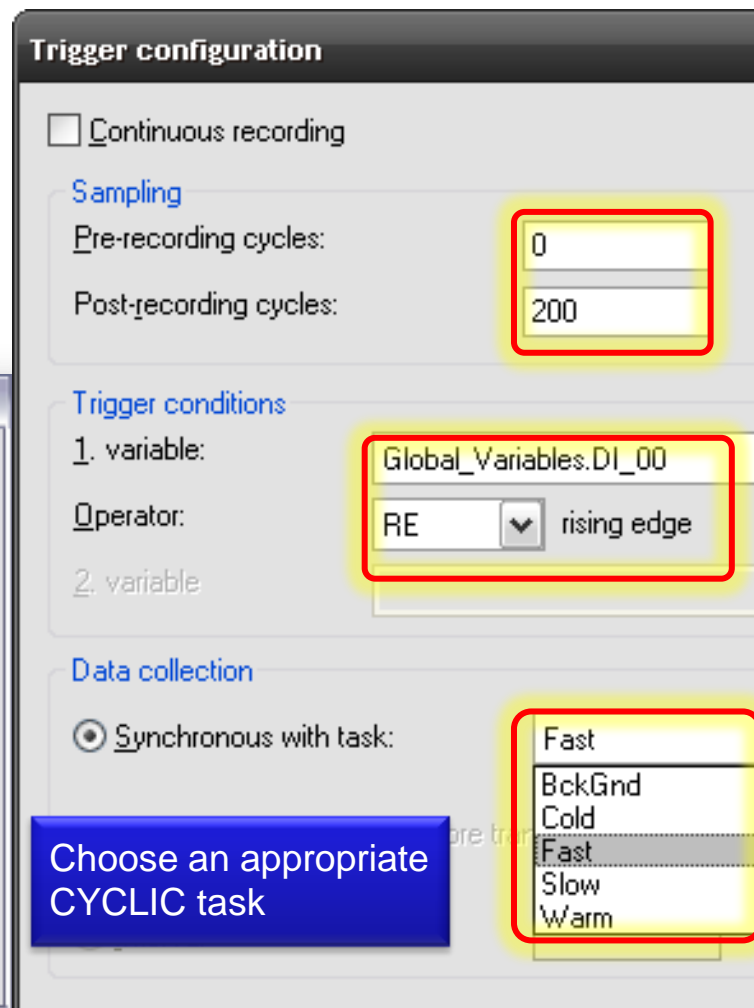
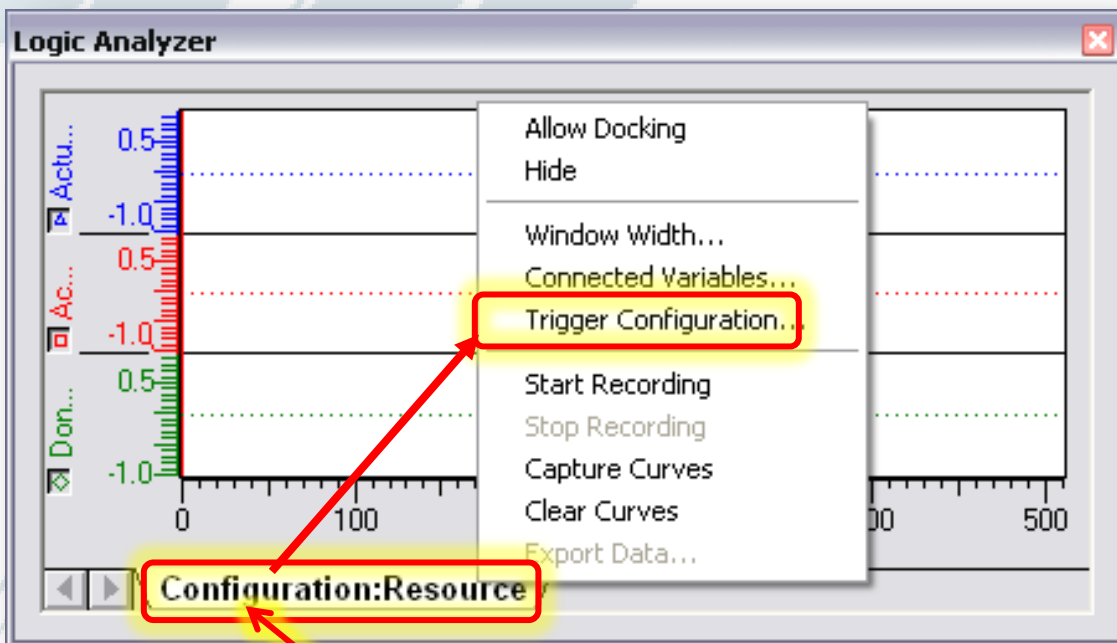


Add To Logic Analyzer

- `G_NumberOfCars`
- `DI_00`
- `G_GateOpen`
- `DO_00`

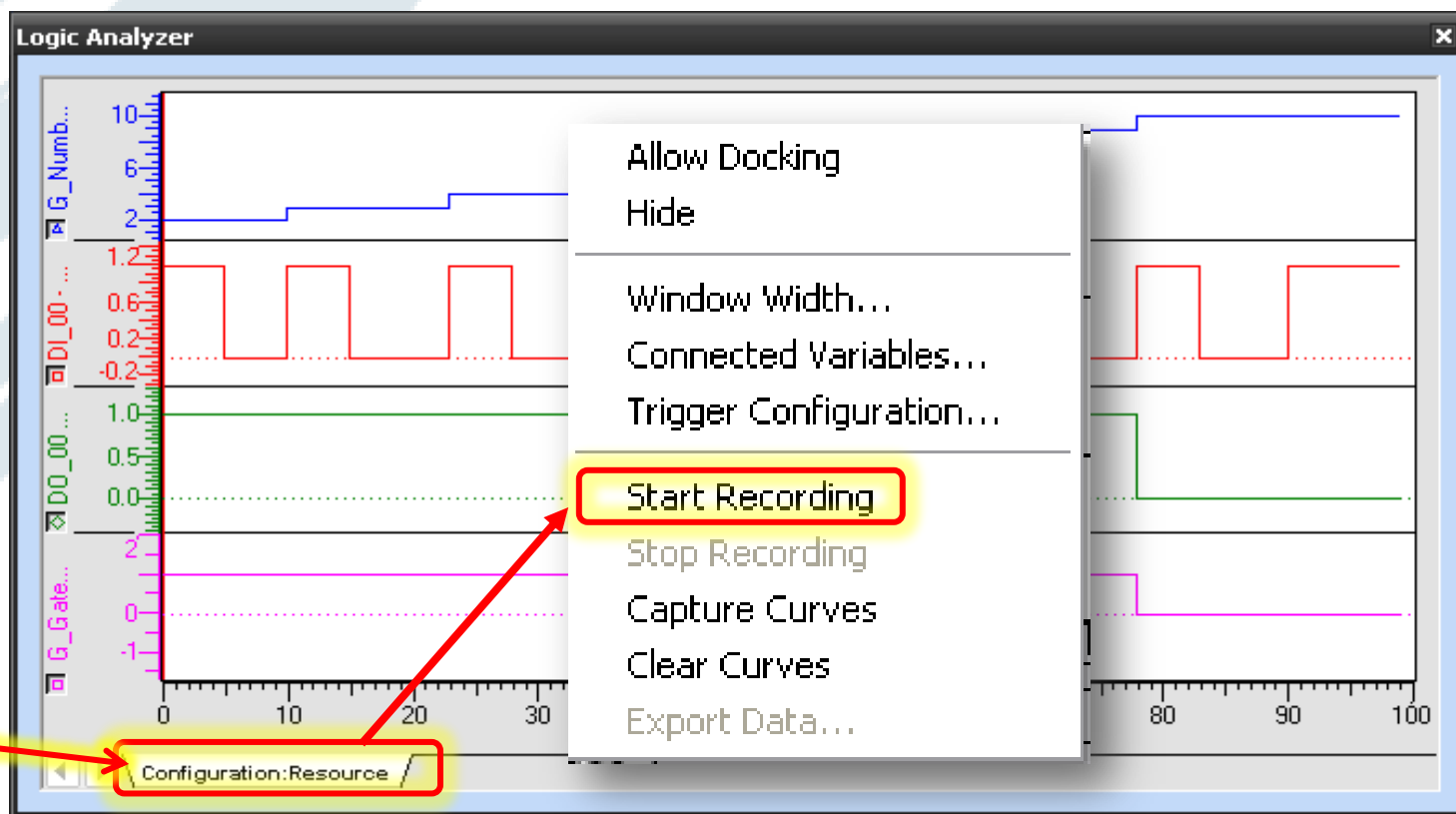
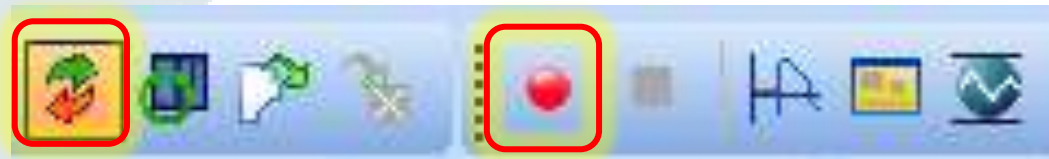


- *Right-click Tab*
 - *Sampling*
 - *Trigger condition*
 - *Task*



Data will be captured over how much time? (cycles... task time)

- *Trigger Configured*
- *Debug ON*
- *Start Recording*
- *Toggle DI_00*

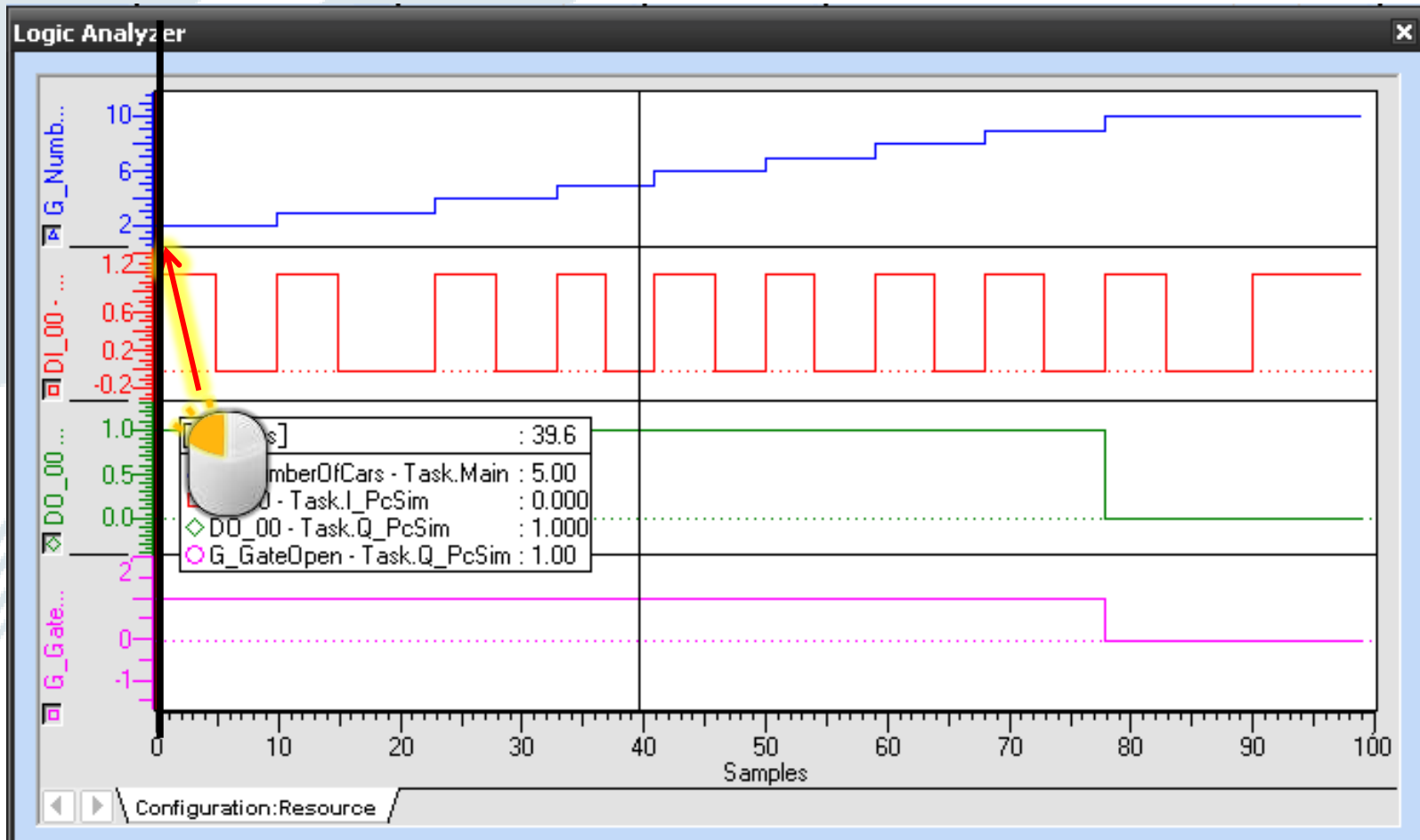


- *Mouse cursor over axis*
- *Right-Click and drag*
- *Left-Click and drag*



- *One Vertical Cursor*
- *One Horizontal Cursor*

Verify the correct number of “cars”
though DI_00 before gate closes.



Program Data Transfer

Compile

Download/Upload

Cold/Warm/Hot Start

Power-On Sequence

Power-On Sequence

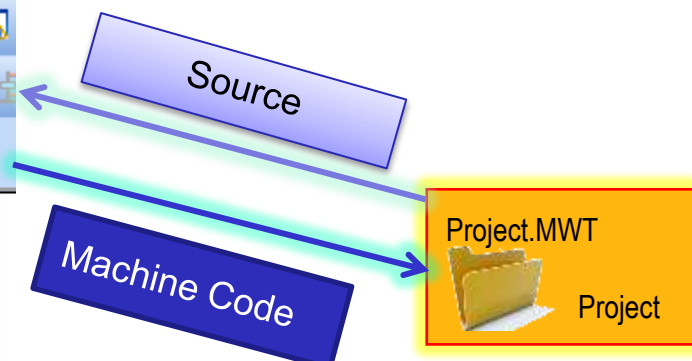
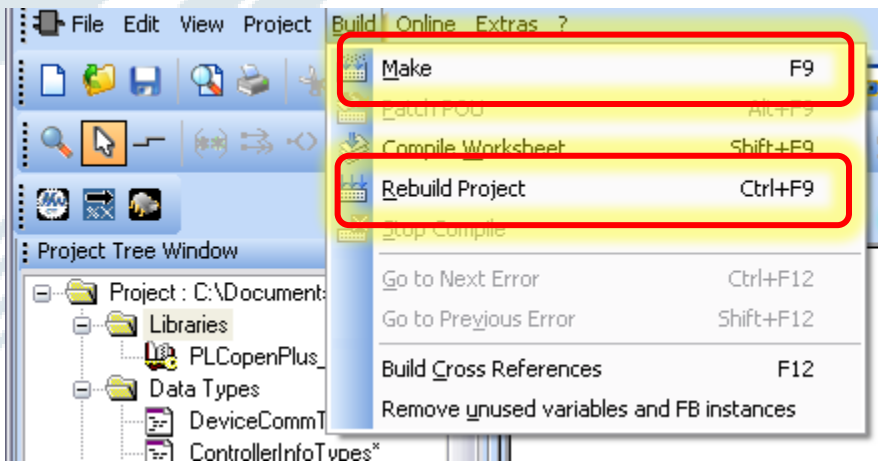
- **“MAKE” (F9)**

- Saves project source
- Compiles changes only
- Faster
- Normal Usage



- **“Rebuild Project” (Shift+F9)**

- Saves project source
- Compiles entire project
- Slower
- Periodic Usage



- Click Errors Tab
- Double-Click Error
 - Location of Error
- Fix Error
- Make

Name	Type	Usage	Description
Default			
Income	LREAL	VAR	
NumberOfCars	LREAL	VAR	
G_NumberOfCars	INT	VAR_EXTER...	
Tax	LREAL	VAR	
TaxRate	LREAL	VAR	
n	INT	VAR	
m	INT	VAR	
PLC_SYS_TICK_CNT	DINT	VAR_EXTER...	
DummyDINT	DINT	VAR	
max	INT	VAR	
MyINT	INT	VAR	
Value1	LREAL	VAR	
Value2	LREAL	VAR	
MyLREAL	LREAL	VAR	

MONEY:M... MoneyV:M... MAIN:Main I_PCSIM:I... Q_PCSIM

No matching global variable found for 'Money:G_NumberOfC...
 No matching global variable found for 'Main:G_NumberOfC...

Build Errors Warnings Infos PLC Errors Print Multi-User

```

----- Compiling variables -----
Global_Variables
----- Linking POU -----
----- Generating IEC Code -----
Collecting POUs used by RESOURCE 'Resource' ...
Generating IEC code for RESOURCE 'Resource' ...

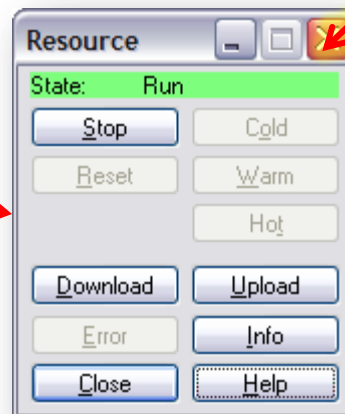
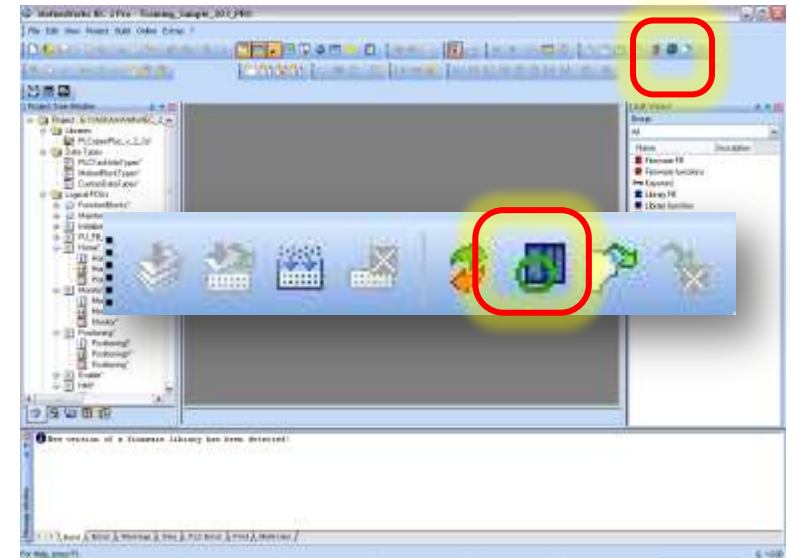
```

✘ 2 Error(s), 0 Warning(s)

Build Errors Warnings Infos PLC Errors Print Multi-User

Project Control Button

- Connection to the controller
 - » Normal “online” status is **green**
 - » If **Error** button appears red, click it to view/resolve error
- Close the Resource Window
 - » Tasks cannot be deleted or changed while open
 - » Avoid Errors “com channel is open”



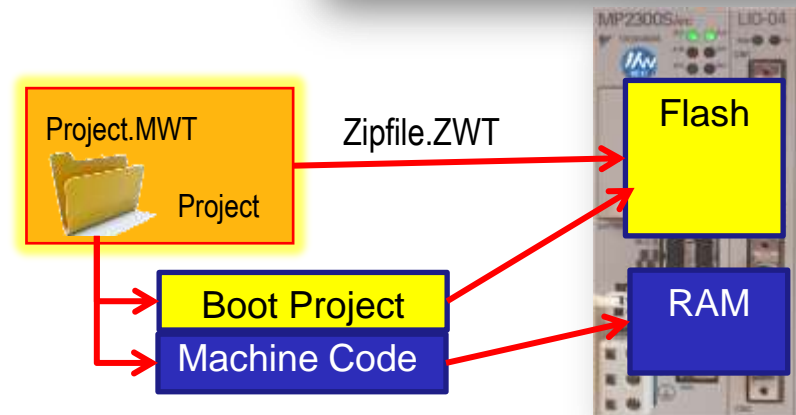
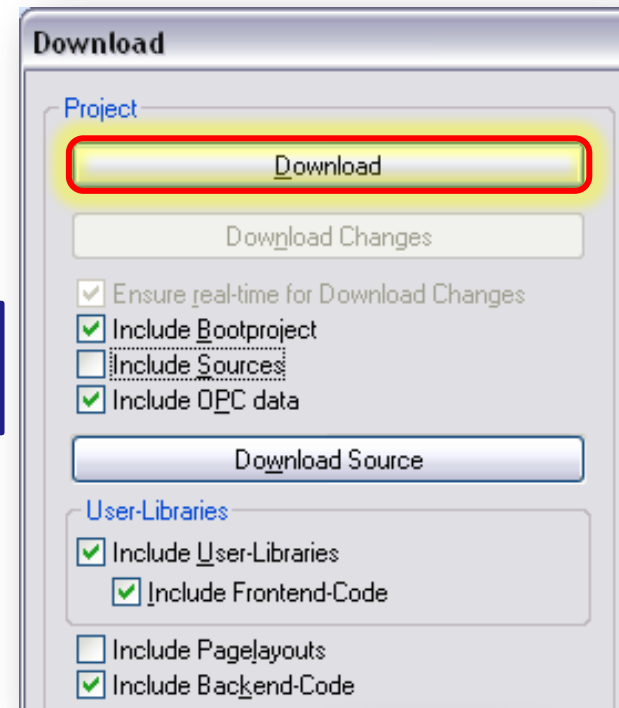
Always Close



ProConOS	0	0
Run	Pro-Con05	Pro-Con05
Stop		
0	0	0
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
CPU	In/8	Out/8

- **Project**
 - Machine code
 - Stored in RAM
 - Stop CPU
- **Boot Project**
 - Machine code
 - Stored in flash
 - Loads to RAM at power up
 - Executes at power up
- **Sources**
 - Zipped offline project folder
 - Stored to flash
 - Future upload
 - Download time slow
 - Intellectual property

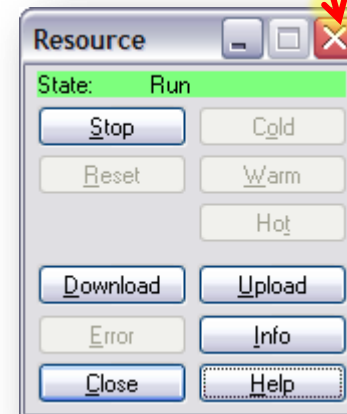
Uncheck items for faster download



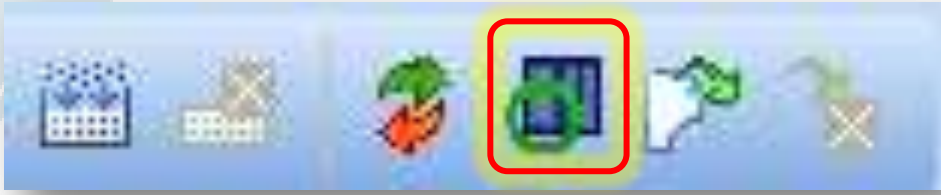
- *Download Changes Includes:*
 - *MAKE all edited POU's*
 - » *Automatic SAVE*
 - *Download Changes*
 - *CPU does not stop*
 - *Limitation (Not Supported)*
 - » *Hardware Change*
 - » *No project on controller*
 - » *Other*
- *Works by memory swap*
 - *Compiled project must be <1/2 PLC RAM capacity*



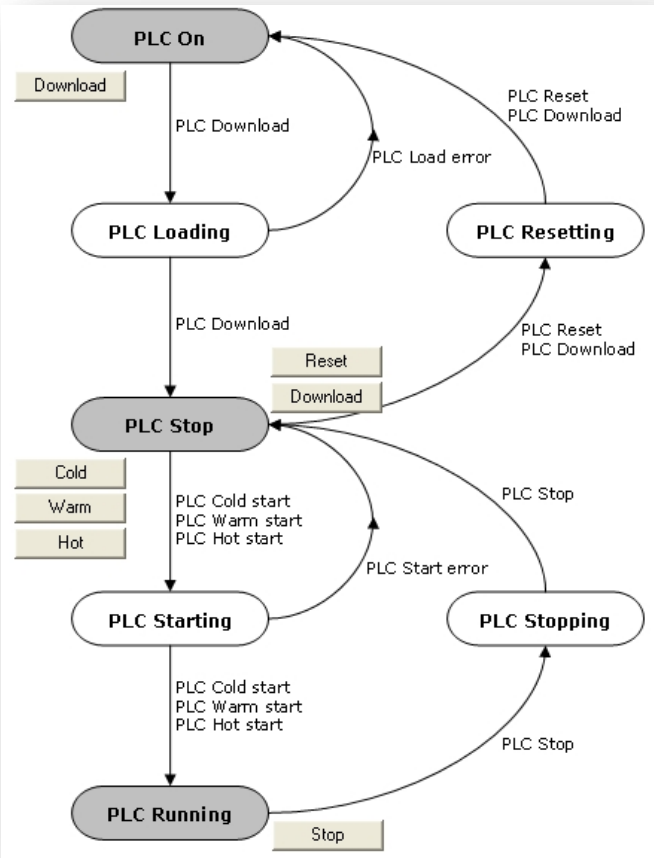
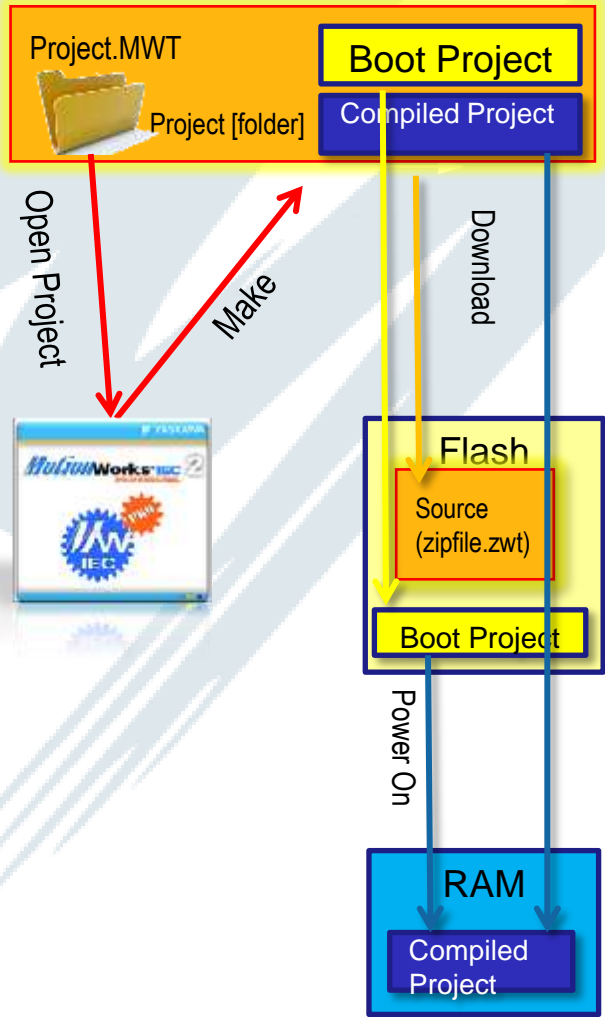
Always Close



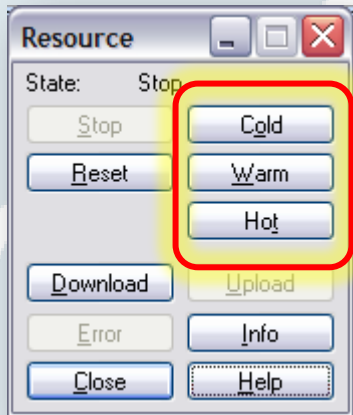
Download changes is FAST and EASY



Project Control



Start CPU: Cold, Warm, or Hot



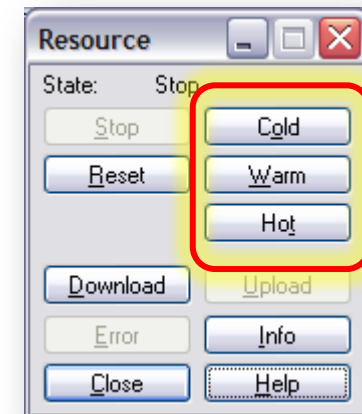
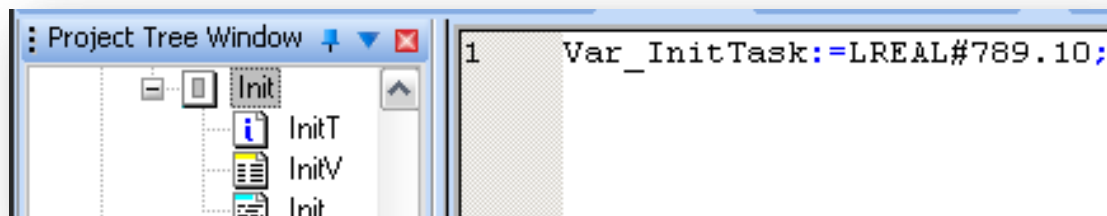
	Cold Start (Machine Commission)	Warm Start (Power On)	Hot Start
Retain Variables Initialized	Yes	No	No
Non-Retain Variables Initialized	Yes	Yes	No
ColdStart Task Runs	Yes	No	No
WarmStart Task Runs	No	Yes	No

1. Add variables to global list

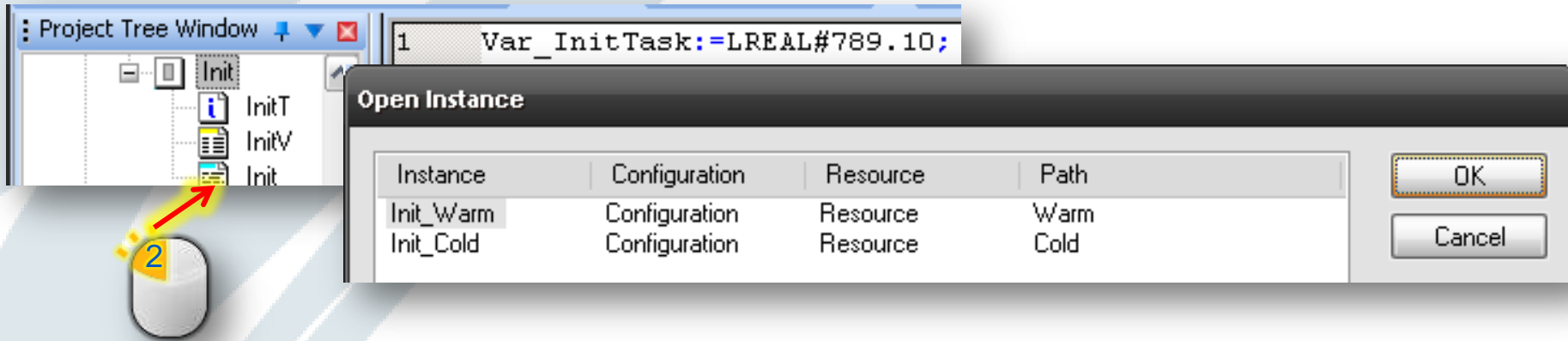
Name	Type	Usage	Init	Retain
System				
Var_Retain	INT	VAR_GLOBAL	123	<input checked="" type="checkbox"/>
Var_NonRetain	INT	VAR_GLOBAL	456	<input type="checkbox"/>
Var_InitTask	LREAL	VAR_GLOBAL		<input type="checkbox"/>

- Download Changes
- Debug Mode
- Stop and start controller using Cold/Warm/Hot

2. Add code to Init POU

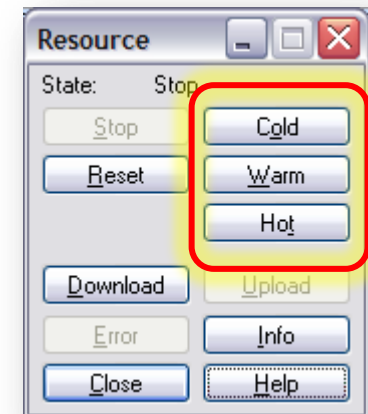


- *Program POU instance used in more than one task*



- *Change Online Values*
 - *Stop and start controller*

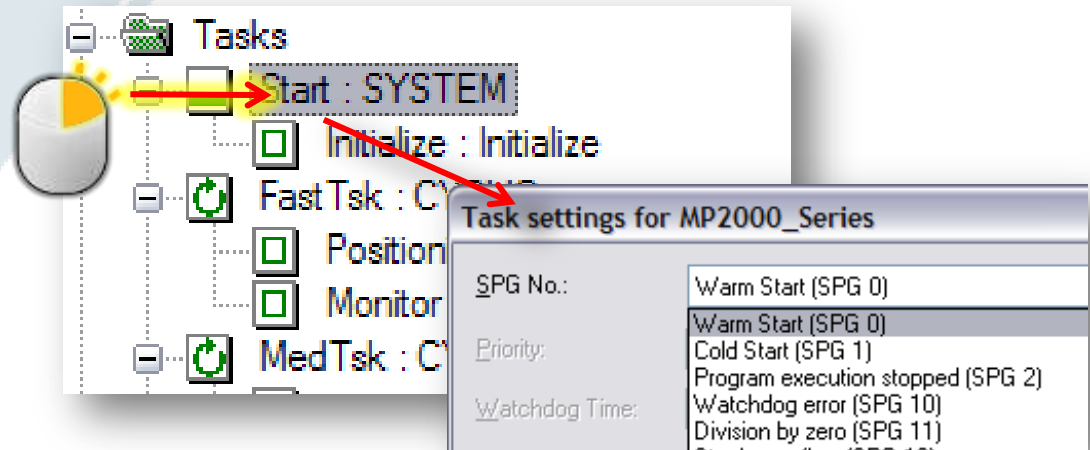
Name	Online value	Type	Usage	Init
System				
Var_Retain	123	INT	VAR_GLOBAL	123
Var_NonRetain	456	INT	VAR_GLOBAL	456
Var_InitTask	789.10	LREAL	VAR_GLOBAL	





- Initialize Data at CPU Start

- Variable "Init" Value
- Warm Start Task
- Cold Start Task



Name	Online value	Type	Usage	Description	Address	Init	Retain
Default							
HomeVel	55.000	LREAL	VAR			45.0	<input type="checkbox"/>
HomeAccDec	360000.000	LREAL	VAR			360_000.0	<input type="checkbox"/>
HomeOffsetLeft	-332.800	LREAL	VAR			20.0	<input checked="" type="checkbox"/>
HomeOffsetRight	-126.000	LREAL	VAR			0.0	<input checked="" type="checkbox"/>
HomePosition	0.000	LREAL	VAR			0.0	<input type="checkbox"/>
G_AbsResetDoneRight	FALSE	BOOL	VAR_EXT...				<input type="checkbox"/>

Why mark a variable "Retain"?

- Home Offsets, Part Counters and other variables have to keep their values when the power is off.
- The controller uses the lithium battery to keep these variable values "alive."

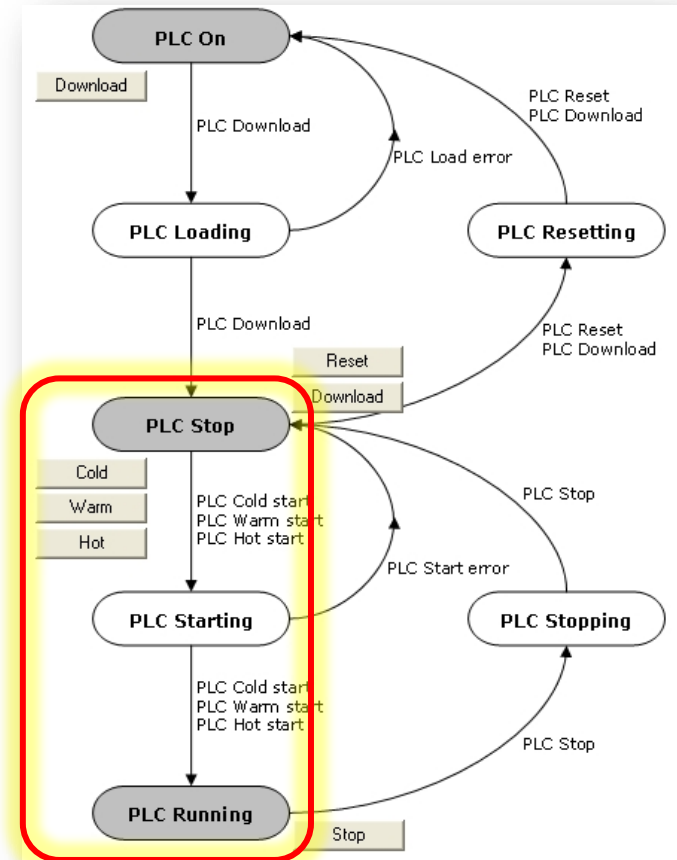


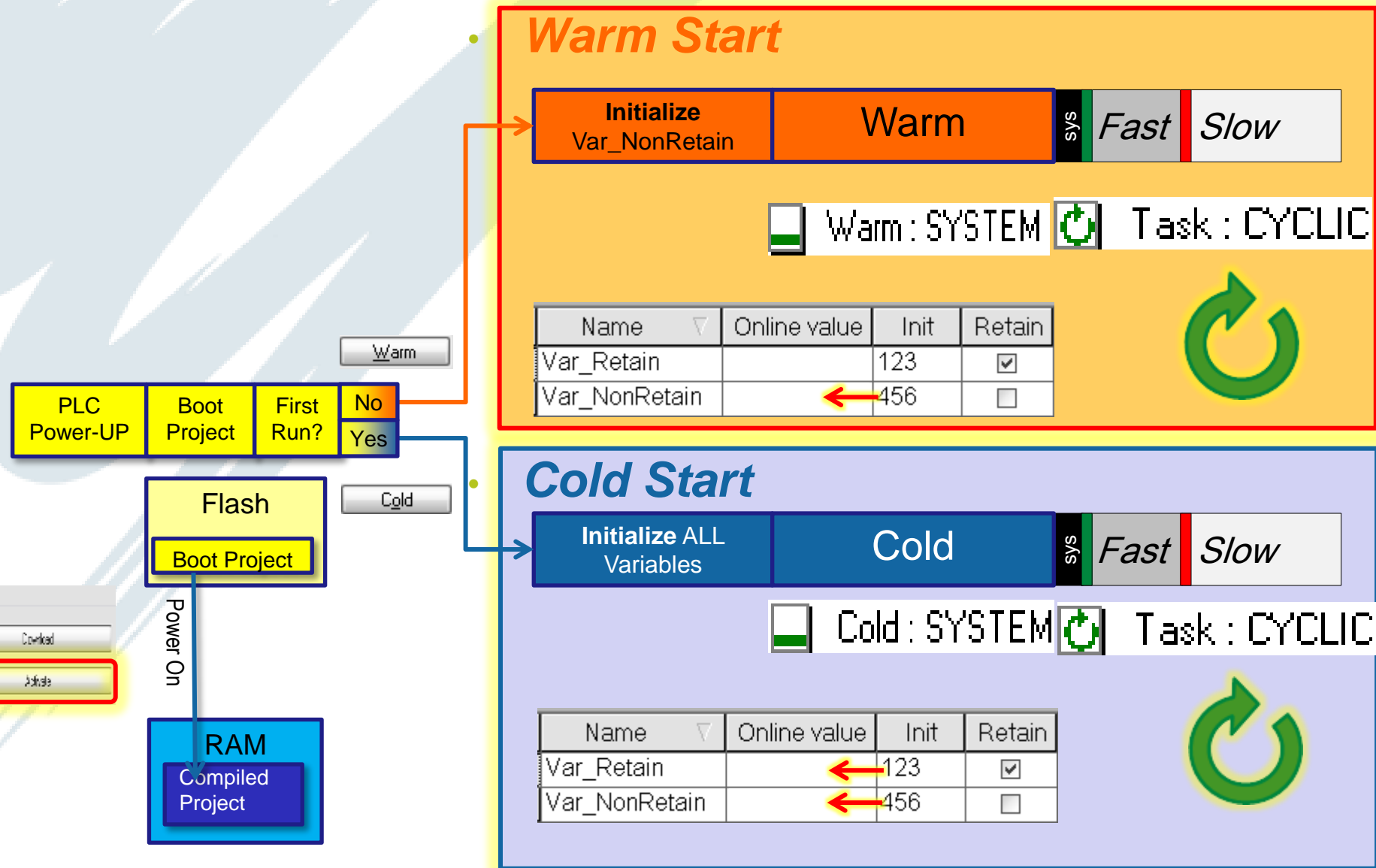
- *Normal PLC Power-On Execution Sequence*

- *Load Boot Project*
 - » *From FLASH to RAM*
- *Warm Start*
 - » *Initialize non-retain variables*
 - » *Run Warm system task 1x*
 - » *Run Cyclic Tasks*

- *Simulator has no “power on”*

- *Boot project is activated manually*
- *Cold or warm is activated manually*





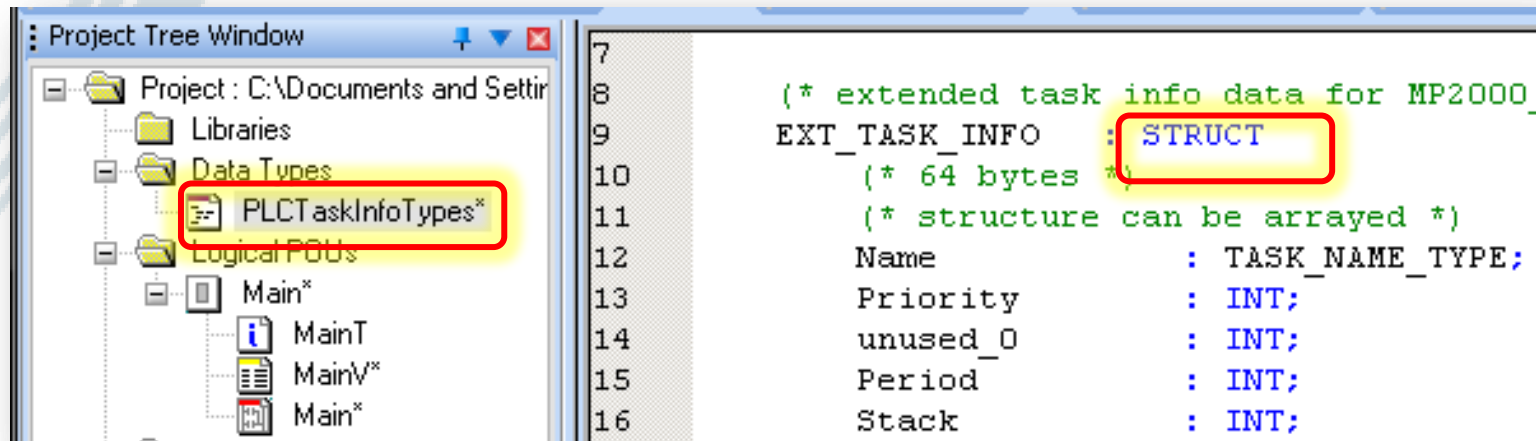
Variables

Name Rules
Data Type
Global/Local
Literal, Keyword
Variable Usage
Renaming Variables
Local and Global with same name
Variable List Organization
Global List Cleanup
Remove Unused Instances

Remove Unused Instances
Global List Cleanup
Variable List Organization

- *Alphanumeric characters and underscore “_”*
 - *Abc_123*
- *Capitalization is displayed but not important*
 - *Var1 = var1 =VAR1*
- *30 character maximum*
 - *Structures and “dot notation” for longer names*
 - » *Ex: FeedAxis.Home.CreepSpeed*
 - » *Declare structures in a Data Types worksheet*

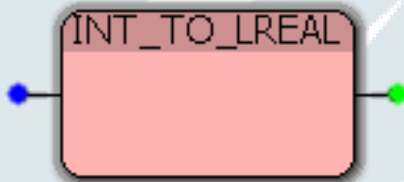
These rules and syntax are VERY strict.
There are no exceptions!



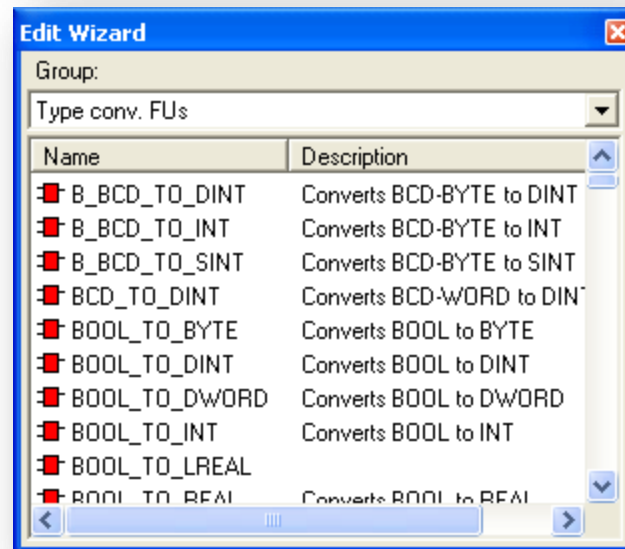
The screenshot displays a software interface with two main panels. On the left is the 'Project Tree Window' showing a hierarchical view of a project. The project is located at 'C:\Documents and Settings\...'. The tree structure includes 'Libraries', 'Data Types', and 'Logical POU's'. Under 'Data Types', a folder named 'PLCTaskInfoTypes*' is highlighted with a red box. Below it, there is a 'Main*' folder containing 'MainT', 'MainV*', and another 'Main*'. On the right is a code editor window showing a ladder logic program. The code is as follows:

```
7
8      (* extended task info data for MP2000_
9      EXT_TASK_INFO : STRUCT
10     (* 64 bytes *)
11     (* structure can be arrayed *)
12     Name           : TASK_NAME_TYPE;
13     Priority        : INT;
14     unused_0       : INT;
15     Period         : INT;
16     Stack          : INT;
```

- *Convert the data type*
 - *Type Conversion Functions*
 - *No automatic conversion*



```
1      3.0000000 NumberOfCars :=INT_TO_LREAL( G_NumberOfCars );
2
3      89.9700000 Income:= LREAL#29.99 * NumberOfCars;
```



Literals

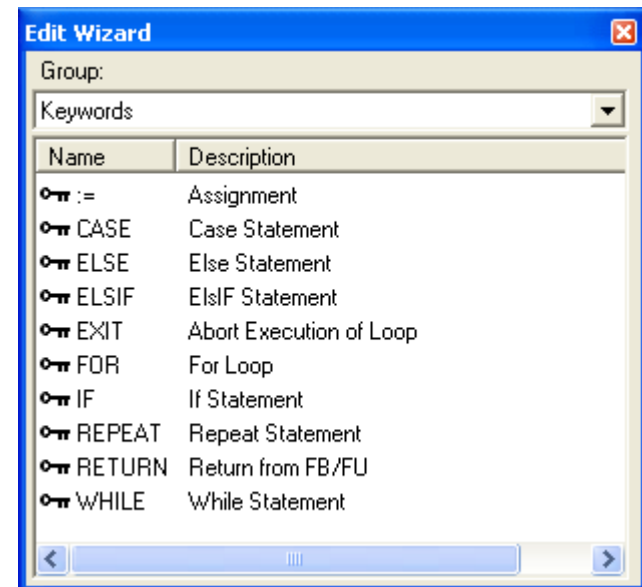
- *REAL#1.0*
- *INT#3*
- *TIME#2s*

Keywords

- *List in “edit wizard”*
- *Keywords only apply to ST POU's*

Notice the decimal point – never an “implied” decimal point

Notice ALL CAPS to define a literal value



The screenshot shows the 'Edit Wizard' dialog box with the 'Keywords' group selected. The list contains the following items:

Name	Description
:=	Assignment
CASE	Case Statement
ELSE	Else Statement
ELSIF	ElsIF Statement
EXIT	Abort Execution of Loop
FOR	For Loop
IF	If Statement
REPEAT	Repeat Statement
RETURN	Return from FB/FU
WHILE	While Statement



- *With Alphanumeric Characters in Variable Name*
 - *Servo_01*
 - *_01Servo*

- *Place Holder Within Data (Cosmetic Only)*
 - *Initial Values*
 - » *100_000_000.0*
 - *Literals*
 - » *LREAL#100_000_000.0*



- View Local Variables

The screenshot shows the 'Project Tree Window' on the left with 'Practice_LD*' selected. On the right, a table lists local variables with their names, types, and usage. The 'Usage' column is highlighted with a red box.

Name	Type	Usage	Description
[-] Default			
MC_MoveRelative_1	MC_MoveRelative	VAR	
Start1	BOOL	VAR	
Distance1	LREAL	VAR	
Speed1	LREAL	VAR	
G_AccDec	LREAL	VAR_EXTER...	

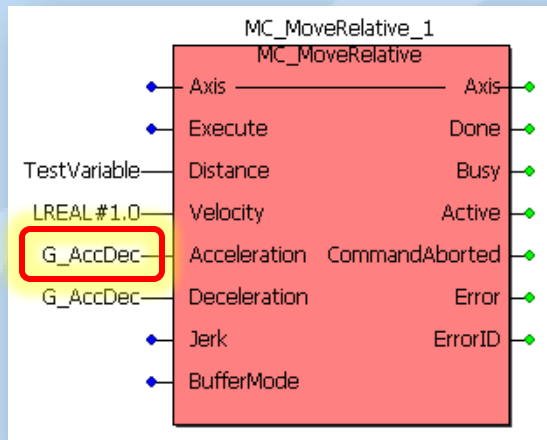
- View Global Variables

The screenshot shows the 'Global Variables' section in the project tree on the left. On the right, a table lists global variables. The 'Usage' column for 'G_AccDec' is highlighted with a red box.

[-] User Variables			
G_AccDec	LREAL	VAR_GLOBAL	
[-] <SGDV Rotary> - Sigma-V Rotary Servo Amplifier - 1:1 (* Moc			

- Create global and local variables

Practice_LD



Variables: Practice_LD

Name	Type	Usage
Default		
MC_MoveRelative_1	MC_MoveRelative	VAR
TestVariable	LREAL	VAR
G_AccDec	LREAL	VAR_EXTER...
TestInt	INT	VAR

Best Practice:
Name global variables
with prefix G_

Practice_ST

```

MC_MoveRelative_1(Axis:=
(* ANY *),Execute:=(* BOOL *),
Distance:= TestDistance (* LREAL *),
Velocity:= LREAL#1.0 (* LREAL *),
Acceleration:= G_AccDec (* LREAL *),
Deceleration:= G_AccDec (* LREAL *),
Jerk:=(* LREAL *),
BufferMode:=(* INT *));
(* ANY *) :=MC_MoveRelative_1.Axis:
    
```

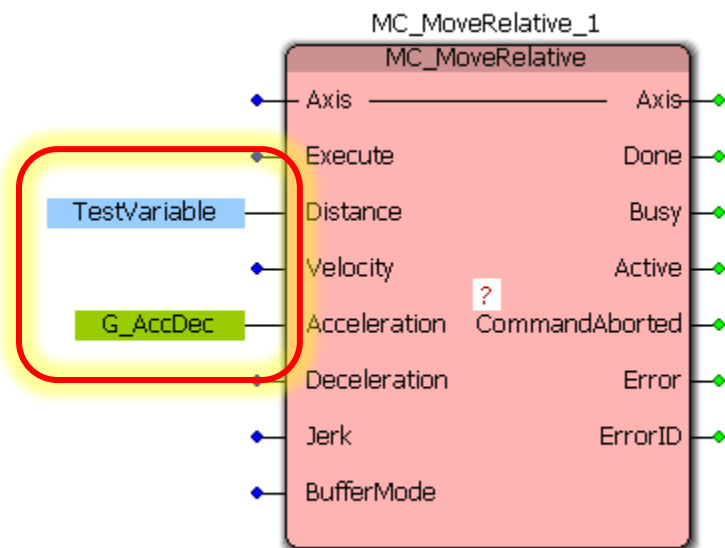
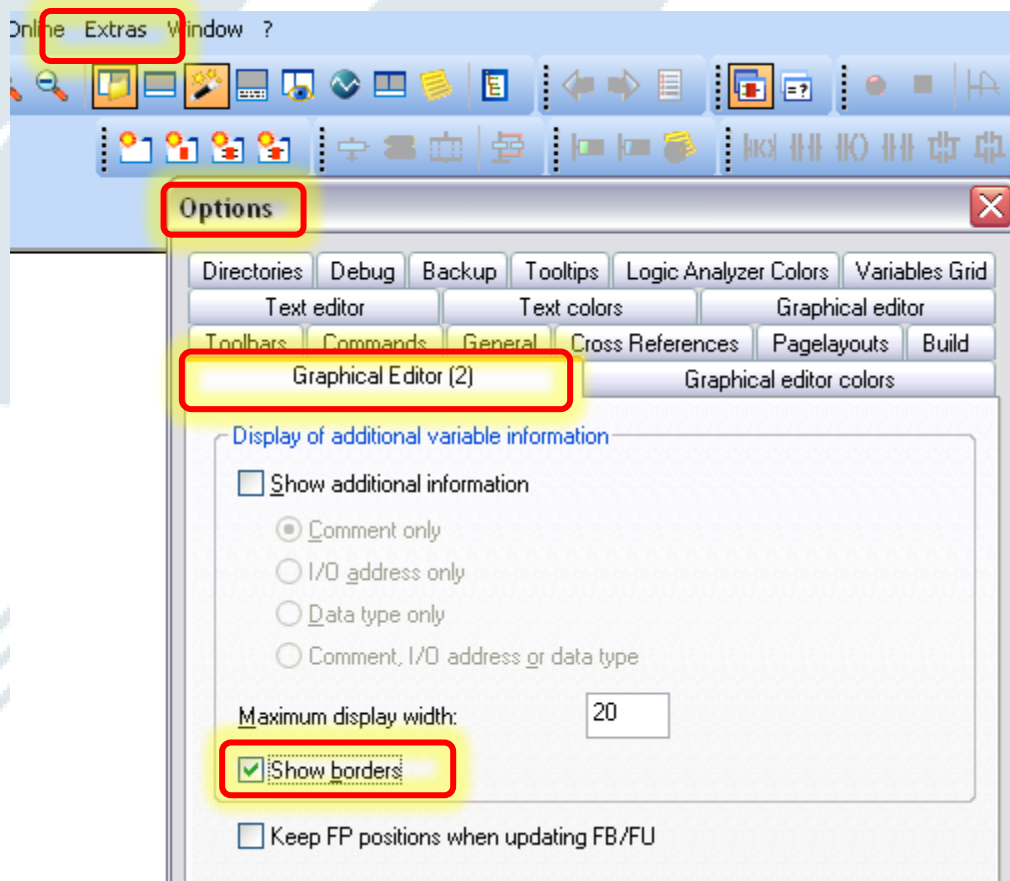
Variables: Practice_ST

Name	Type	Usage
Default		
MC_MoveRelative_1	MC_MoveRelative	VAR
TestDistance	LREAL	VAR
G_AccDec	LREAL	VAR_EXTERNAL

GLOBAL_VARIABLES

Name	Type	Usage
MO1_DI_15	BOOL	VAR_GLOBAL
MO1_DO_00	BOOL	VAR_GLOBAL
MO1_DO_01	BOOL	VAR_GLOBAL
MO1_DO_02	BOOL	VAR_GLOBAL
MO1_DO_03	BOOL	VAR_GLOBAL
MO1_DO_04	BOOL	VAR_GLOBAL
MO1_DO_05	BOOL	VAR_GLOBAL
MO1_DO_06	BOOL	VAR_GLOBAL
MO1_DO_07	BOOL	VAR_GLOBAL
MO1_DO_08	BOOL	VAR_GLOBAL
MO1_DO_09	BOOL	VAR_GLOBAL
MO1_DO_10	BOOL	VAR_GLOBAL
MO1_DO_11	BOOL	VAR_GLOBAL
MO1_DO_12	BOOL	VAR_GLOBAL
MO1_DO_13	BOOL	VAR_GLOBAL
MO1_DO_14	BOOL	VAR_GLOBAL
MO1_DO_15	BOOL	VAR_GLOBAL
User Variables		
G_AccDec	LREAL	VAR_GLOBAL

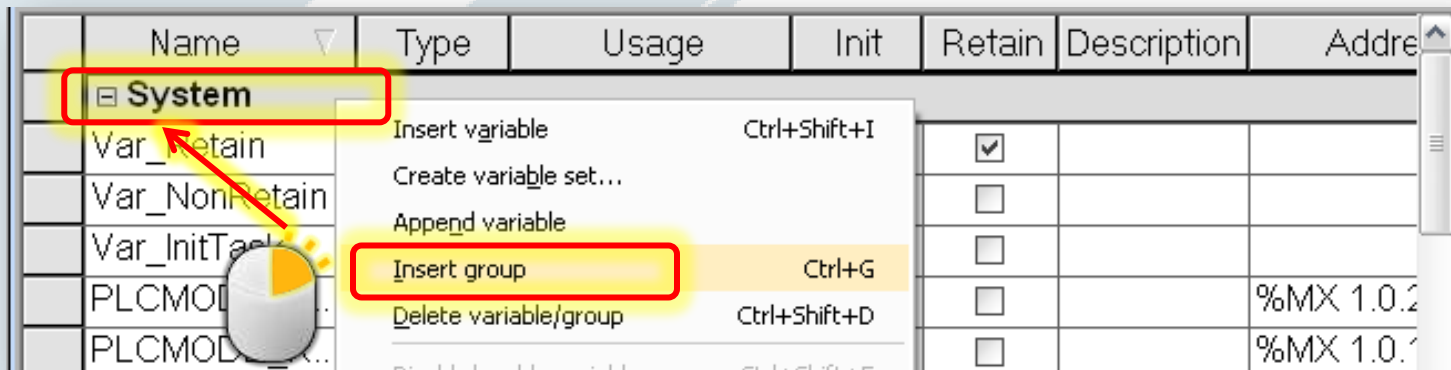
- Use “Show Borders” for color confirmation of global and local
 - Extras, Options, Graphical Editor (2)



- **GLOBAL or LOCAL?**
 - Use LOCAL unless global is required
 - Suggested Convention
 - » Type G_ in front of all global variables
 - G_EnableRight
 - G_EnableLeft
 - » Visual Confirmation when looking at global list

Name	Type	Usage	D
System			
<LIO-01> - 16 DI / 16 DO Sinking + 1 Pulse Latch I/O Modu			
User Variables			
LeftMotor	AXIS_REF	VAR_GLOBAL	
RightMotor	AXIS_REF	VAR_GLOBAL	
VirtualMotor	AXIS_REF	VAR_GLOBAL	
G_EnableRight	BOOL	VAR_GLOBAL	
G_EnableVirtual	BOOL	VAR_GLOBAL	
ExternalEncoder	AXIS_REF	VAR_GLOBAL	
G_Start	BOOL	VAR_GLOBAL	
G_StopLeft	BOOL	VAR_GLOBAL	
G_StopRight	BOOL	VAR_GLOBAL	
G_ResetLeft	BOOL	VAR_GLOBAL	
G_ResetRight	BOOL	VAR_GLOBAL	
G_ResetController	BOOL	VAR_GLOBAL	
<SGDV Rotary> - Sigma-V Rotary Servo Amplifier - 1:1 (f			
AX1_SI1_POT	BOOL	VAR_GLOBAL	P
AX1_SI2_NOT	BOOL	VAR_GLOBAL	N
AX1_SI3_DEC	BOOL	VAR_GLOBAL	D
AX1_SI4_EXT1	BOOL	VAR_GLOBAL	E
AX1_SI5_EXT2	BOOL	VAR_GLOBAL	E
AX1_SI6_EXT3	BOOL	VAR_GLOBAL	E
AX1_BRK	BOOL	VAR_GLOBAL	B
AX1_HBB	BOOL	VAR_GLOBAL	H
AX1_SI0_I012	BOOL	VAR_GLOBAL	C
AX1_SI1_I013	BOOL	VAR_GLOBAL	C
AX1_SI2_I014	BOOL	VAR_GLOBAL	C
AX1_SI3_I015	BOOL	VAR_GLOBAL	C
AX1_ALM	BOOL	VAR_GLOBAL	A

- *Insert Variable Groups*



- *Rename*
 - *User Variables*
 - *PC Simulator*

Name	Type	Usage
User Variables		
PC Simulator		
System		

Variable Groups facilitate organization of the global variables

- *Global variables in the wrong group?*
 - *Example: Check LIO-01 Group*
 - *Drag variables to appropriate group*

Click and drag row header to highlight multiple rows

Click and drag row header **again** to move the group of variables

Red Line appears while dragging

	Name ▾	Type	Usage
<input type="checkbox"/>	User Variables		
<input type="checkbox"/>	PC Simulator		
<input type="checkbox"/>	System		
<input type="checkbox"/>	Var_Retain	INT	VAR_GLOBAL
<input type="checkbox"/>	Var_NonRetain	INT	VAR_GLOBAL
<input type="checkbox"/>	Var_InitTask	LREAL	VAR_GLOBAL
<input type="checkbox"/>	PLCMODE_S...	BOOL	VAR_GLOBAL
<input type="checkbox"/>	PLCMODE_R...	BOOL	VAR_GLOBAL

- Create the variable “G_DelayTime” in the Init POU

The screenshot displays the SIMATIC Manager interface. On the left, the 'Variable Properties' dialog box is open, with the 'Name' field set to 'G_DelayTime', 'Data Type' set to 'TIME', and 'Usage' set to 'VAR_GLOBAL'. The 'RETAIN' checkbox is unchecked. In the center, the 'Project Tree Window' shows the project structure, with 'User Variables' highlighted in blue. On the right, a ladder logic diagram shows the variable declaration 'G_DelayTime := T#2s;' in a yellow box, with a red box around it and a mouse cursor hovering over it. A blue callout box points to the 'User Variables' folder in the project tree, stating: 'Global Variable Groups are for organizational purposes only and do not affect operation'. Another blue callout box points to the variable declaration in the ladder logic, stating: 'More information coming on time format T#2s'. At the bottom, a portion of a ladder logic diagram is visible, showing 'PLCMODE STOP' and 'User Variables' highlighted in blue.

- *Global List Has Mistakes*

- *Delete them*

User Variables		
VirtualMotor	AXIS_REF	VAR_GLOBAL
V001	BOOL	VAR_GLOBAL
V000	BOOL	VAR_GLOBAL
RightMotor	AXIS_REF	VAR_GLOBAL
LeftMotor	AXIS_REF	VAR_GLOBAL
k	BOOL	VAR_GLOBAL
G_StopRight	BOOL	VAR_GLOBAL

- *Change local list usage to “var”*

» *Find using MAKE*

Name	Type	Usage	Description
Default			
MC_Stop_1	MC_Stop	VAR	
MC_Stop_2	MC_Stop	VAR	
k	BOOL	VAR_EXTERN	
leftMotor	AXIS_REF	VAR	
rightMotor	AXIS_REF	VAR	
G_StopLeft	BOOL	VAR	

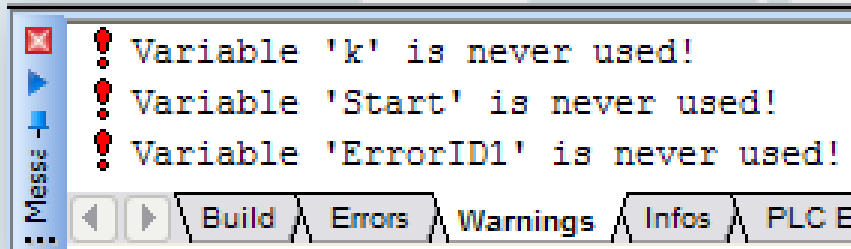
Global_Vari... Reset:Reset ResetV:Reset StopV:Stop

No matching global variable found for 'Stop:k' in resource 'Resource'!
 No matching global variable found for 'Reset:V000' in resource 'Resource'!

TIP: If you have a lot of variables to update...

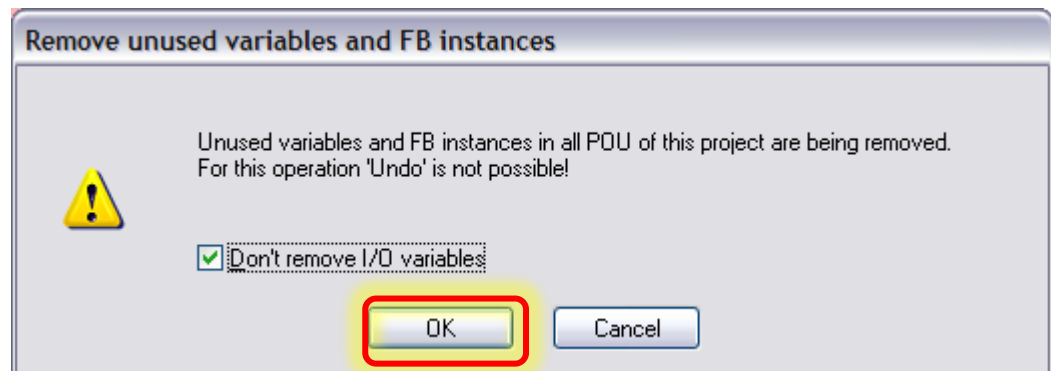
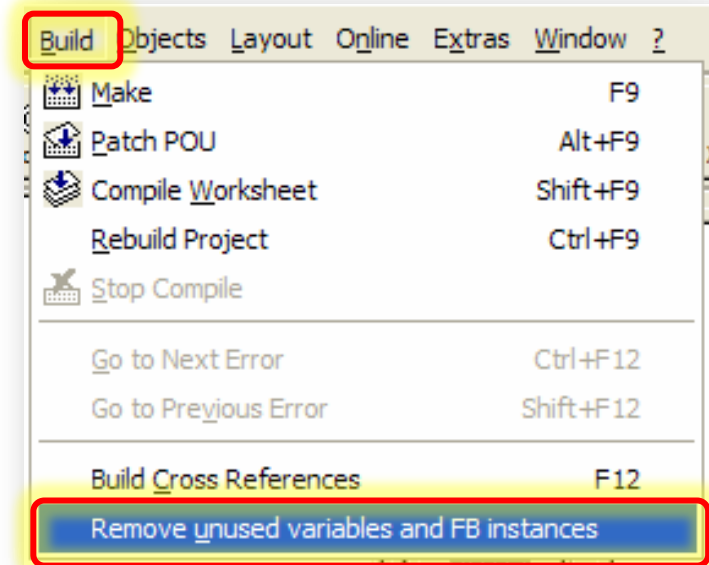
1. Click on the Usage field
2. Type the letter 'v'
3. Click the next one

- *Warning – Variable is never used!*



- *Build menu*

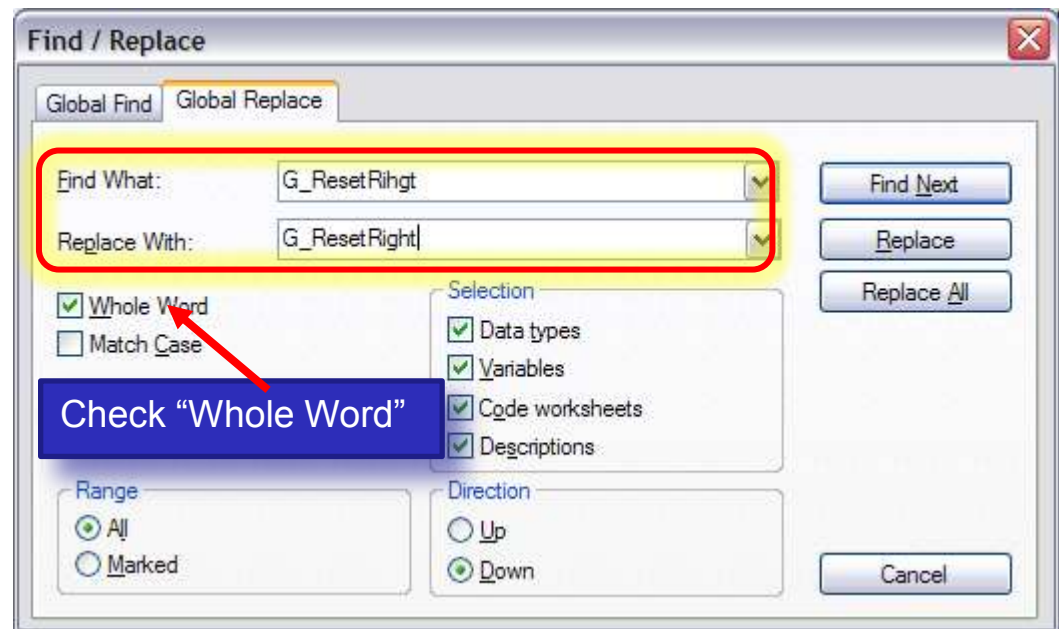
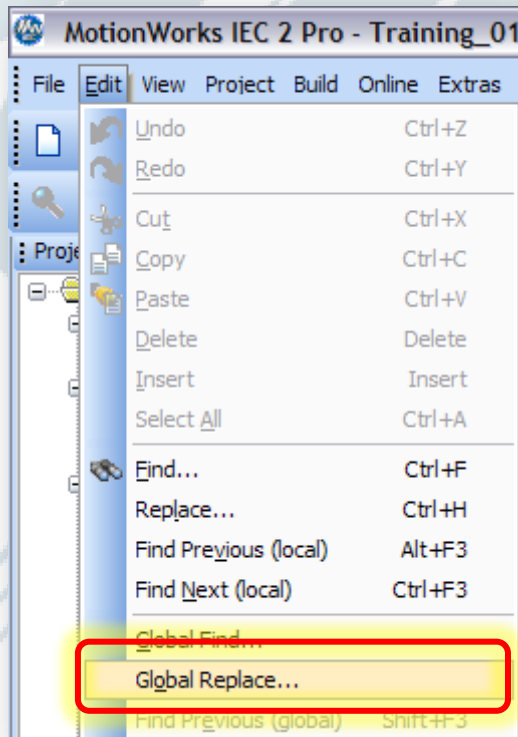
- » *“Remove unused variables and FB instances”*
 - *“Unused” means not used in a POU*
- » *MAKE completes with no warnings*
- » *Global variables NOT removed*



Rename Variables using Global Replace

- Same name required
 - » Local Variable List
 - » Usage in Code

EXAMPLE:
Rename due to spelling error or naming convention



- **Common Mistake**

- Create global variable (without G_)
 - » Mistake – wanted local variable
- Create local variable with same name
 - » CAUTION “New Local” or “Use Global”

Variable Properties

Name: Done

Data Type: BOOL

Usage: VAR_GLOBAL **Mistake !**

Initial value:

CAUTION

A global variable with the same name already exists in the selected global worksheet!

Create new local variable

Use global variable

Apply setting above and don't ask again.
(Reactivating this dialog can be done by 'Extras'-'>'Options'-'>'Build'.)

OK Cancel

User Variables			
LeftMotor	AXIS_REF	VAR_GLOBAL	
RightMotor	AXIS_REF	VAR_GLOBAL	
VirtualMotor	AXIS_REF	VAR_GLOBAL	
G_EnableRight	BOOL	VAR_GLOBAL	
G_EnableVirtual	BOOL	VAR_GLOBAL	
ExternalEncoder	AXIS_REF	VAR_GLOBAL	
G_Start	BOOL	VAR_GLOBAL	
G_StopLeft	BOOL	VAR_GLOBAL	
G_StopRight	BOOL	VAR_GLOBAL	
G_ResetLeft	BOOL	VAR_GLOBAL	
G_ResetRight	BOOL	VAR_GLOBAL	
G_ResetController	BOOL	VAR_GLOBAL	
V000	BOOL	VAR_GLOBAL	
Busy			
Done			

- **Solution**

- » Cancel the CAUTION
- » DELETE global variable

Timers and Sequencing

Overview

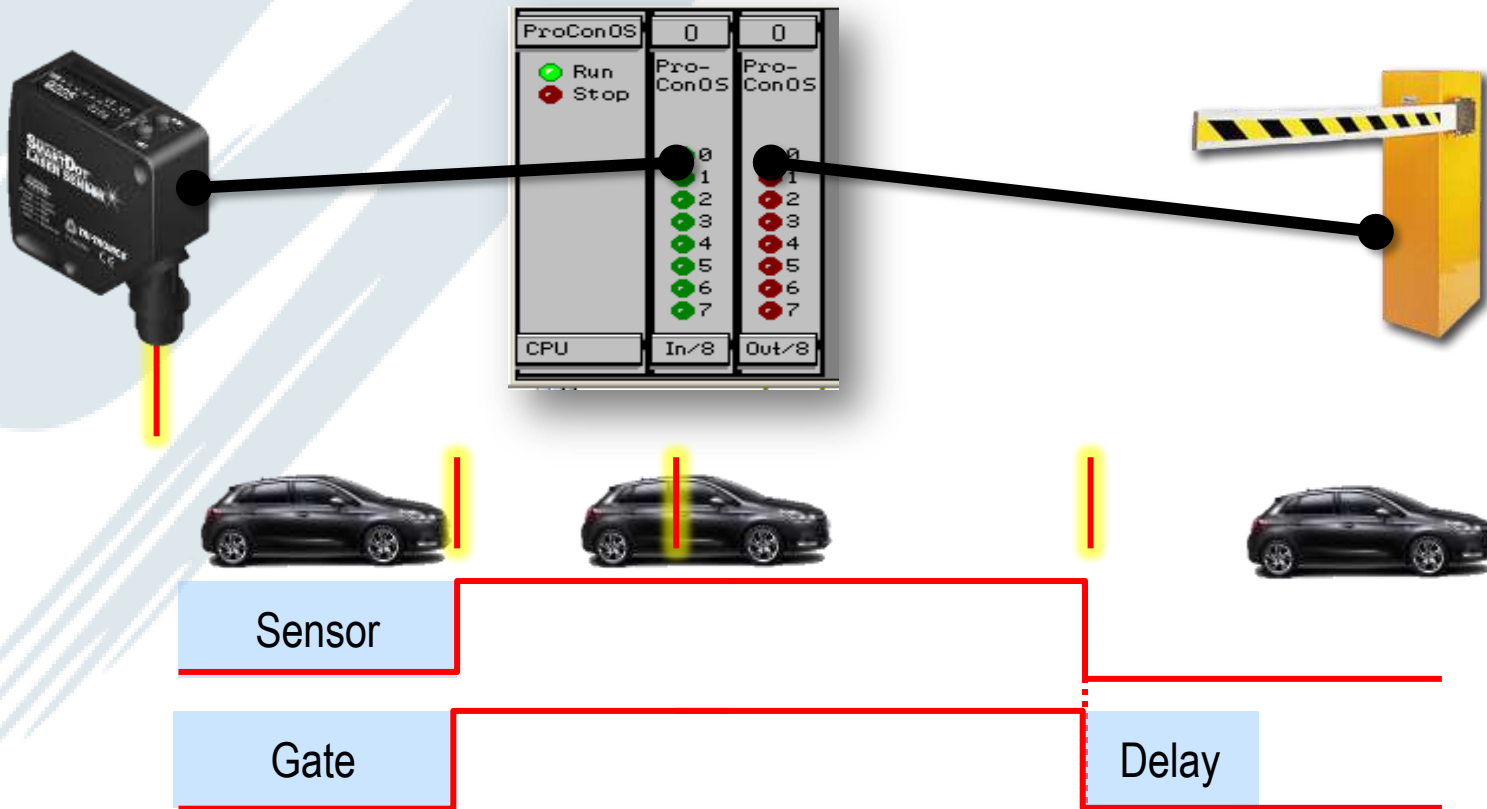
IEC 61131-3 Timers

Application

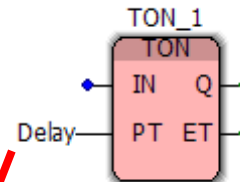
Sequencing

sequencing

- *Application*
 - *Timer to close gate after sensor releases*



- *Right-click Help for parameter definition*
- *TIME data type has a special format*
 - *Initial values must be expressed as a literal*
 - **CORRECT**
 - » T#5s
 - » T#5.0s
 - » Time#5s
 - **NOT CORRECT**
 - » 5
 - » 5.0



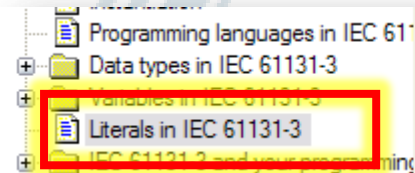
Variable Properties

Name: Delay

Data Type: TIME

Usage: VAR RETAIN

Initial value: T#2s



Help File
MPReadMe001.chm

Duration literals

Duration data can be represented in hours, minutes, seconds, milliseconds and in combination of these formats.

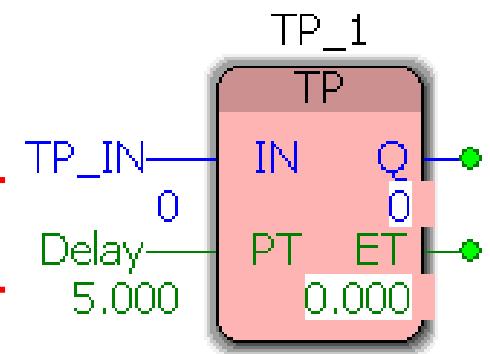
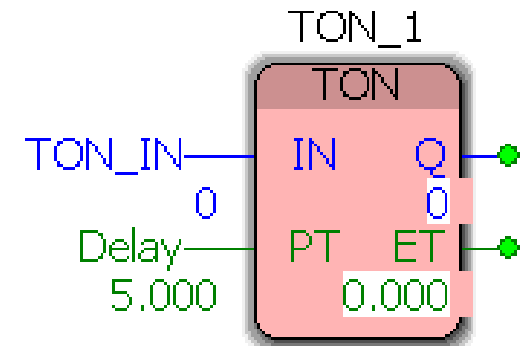
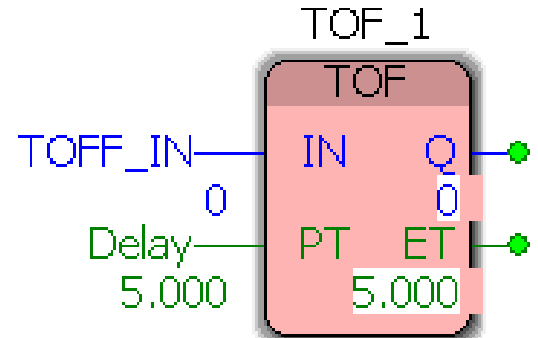
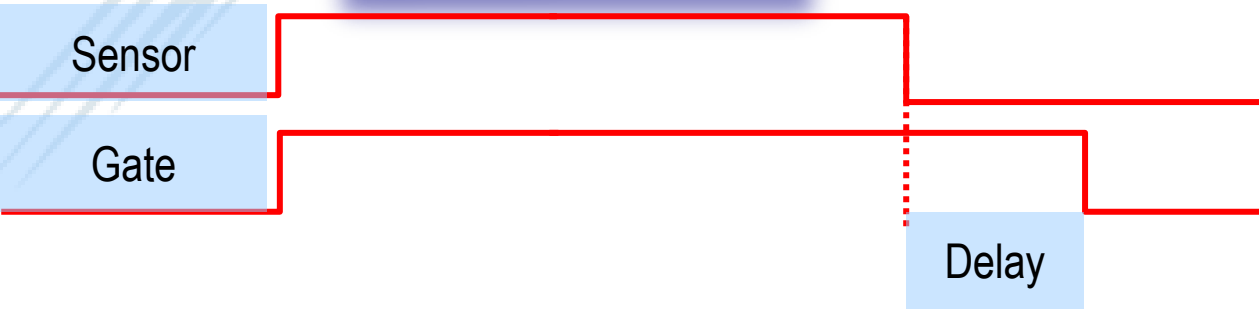
Type	Examples
Short prefix	T#14ms t#14ms t#12m18s3.5ms T#25h_15m t#25h_15m
Long prefix	TIME#14 ms time#14ms TIME#25h_15m time#25h_15m



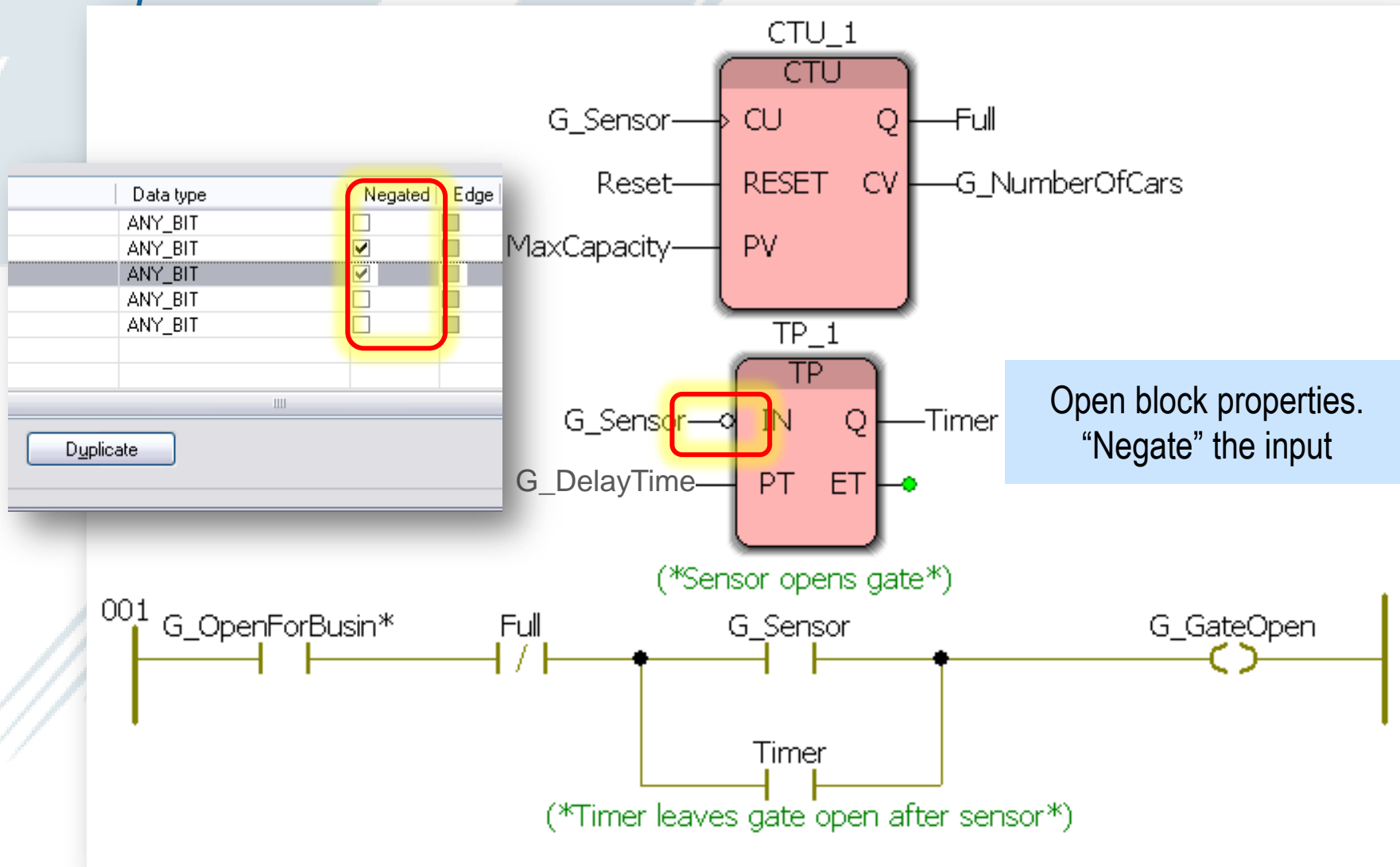
- Create new LD POU “Timers”
- Run in Fast task
- Add each timer

What happens if you change the value of IN during operation?

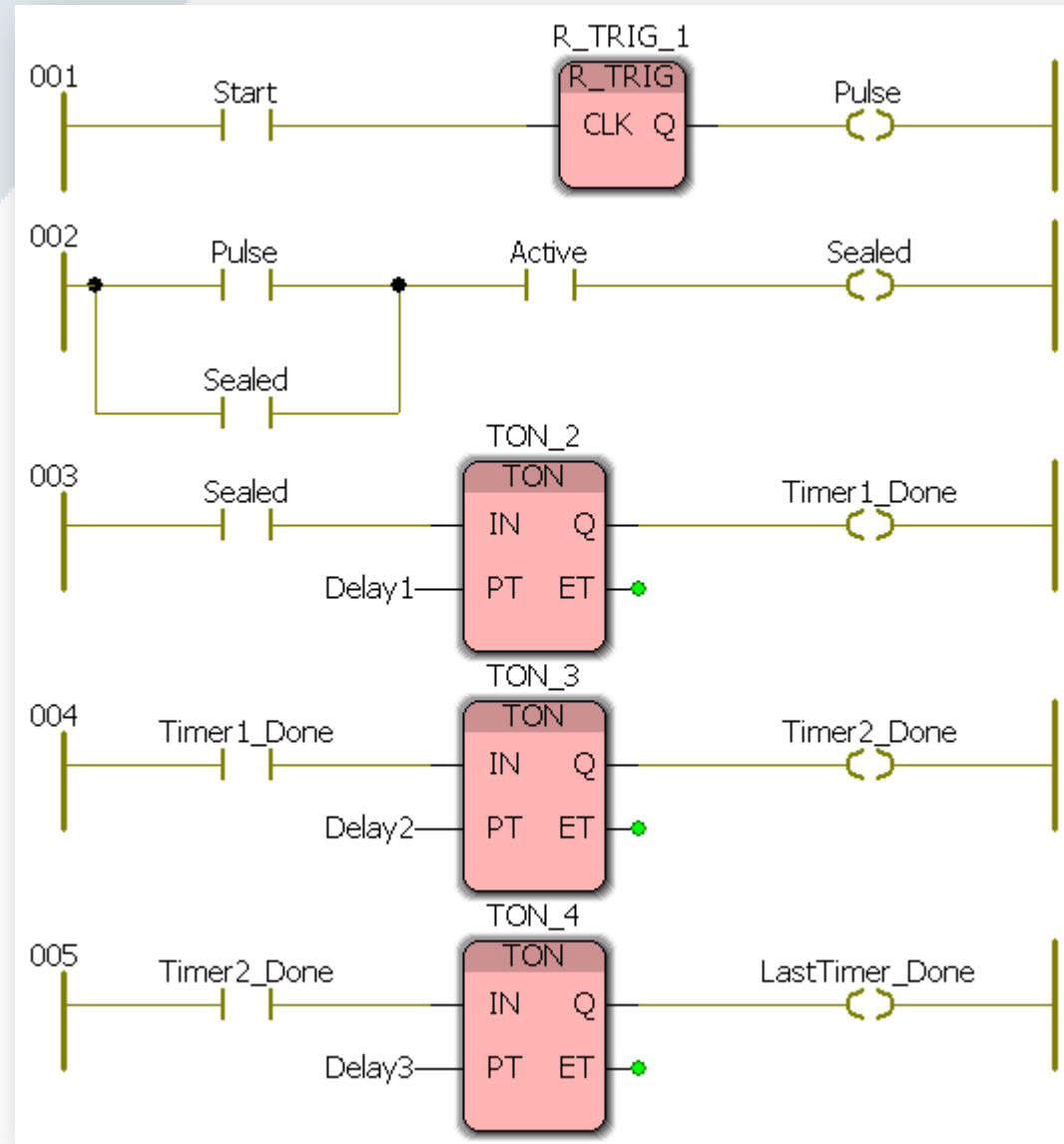
Which timer is appropriate for the application?



- Implement timer modification to MAIN



- *Use ladder “seal-in” circuit*



Arrays

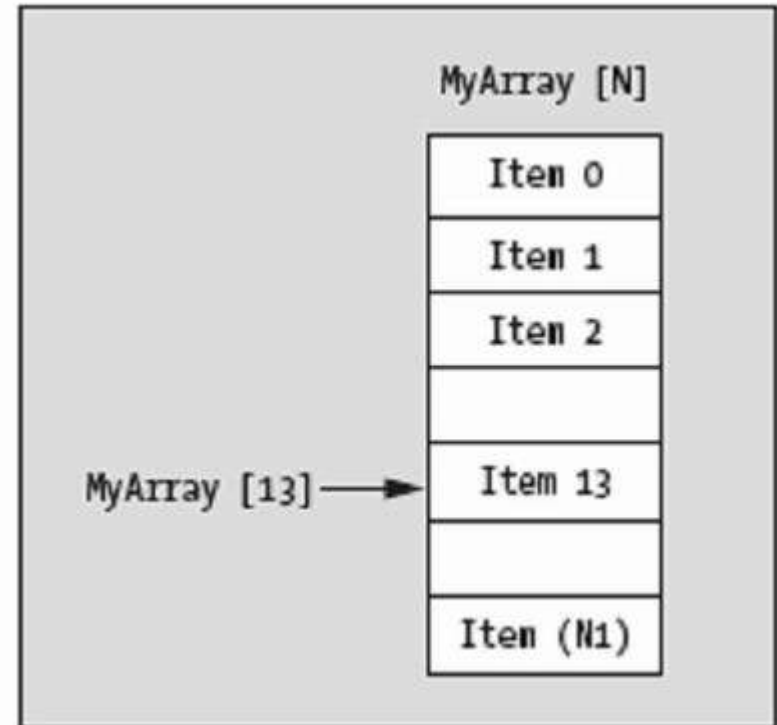
Definition

Create Data Type

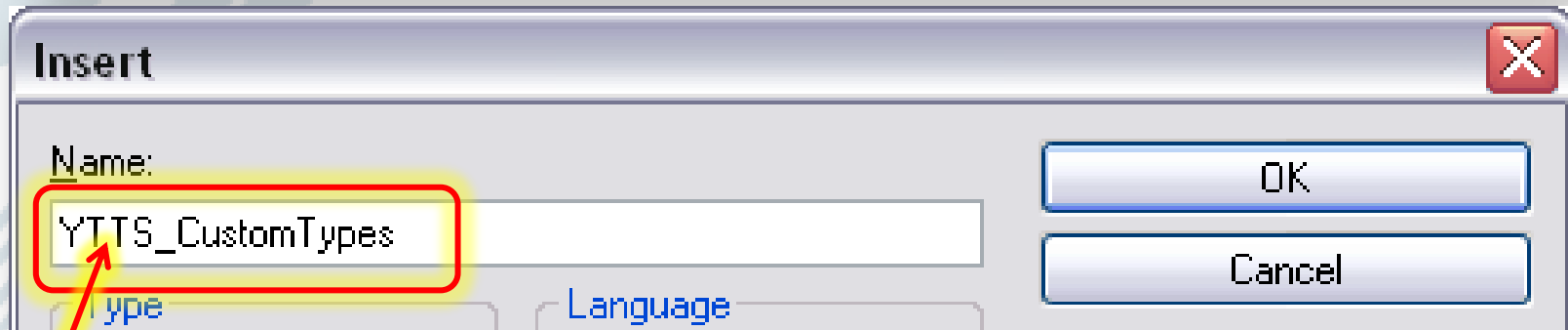
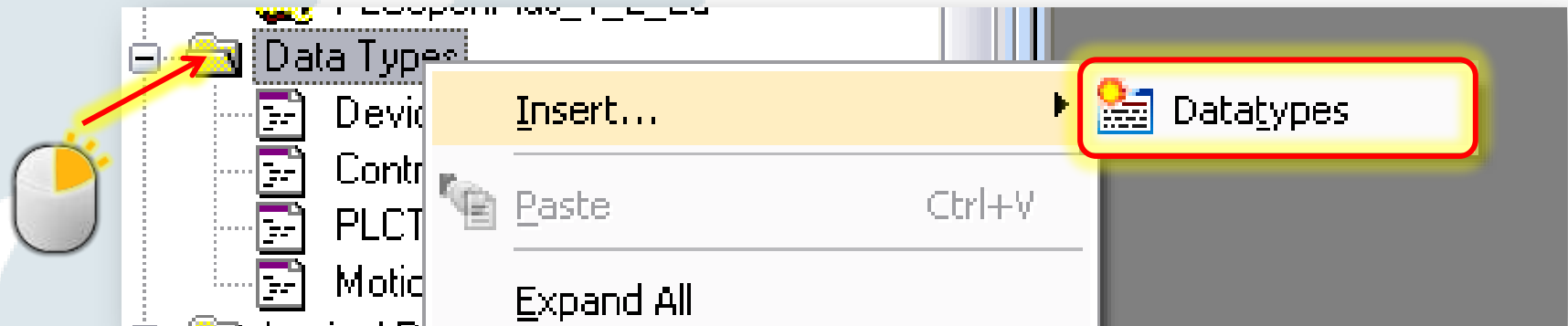
Usage and Syntax

- *A Single Variable*
- *Multiple elements*
- *Same data type*
- *Values accessed using index*

- *Using an Array*
 - *Create array data type*
 - *Create a variable with this data type*
 - *Array element index*
 - » *MyArray[13]*



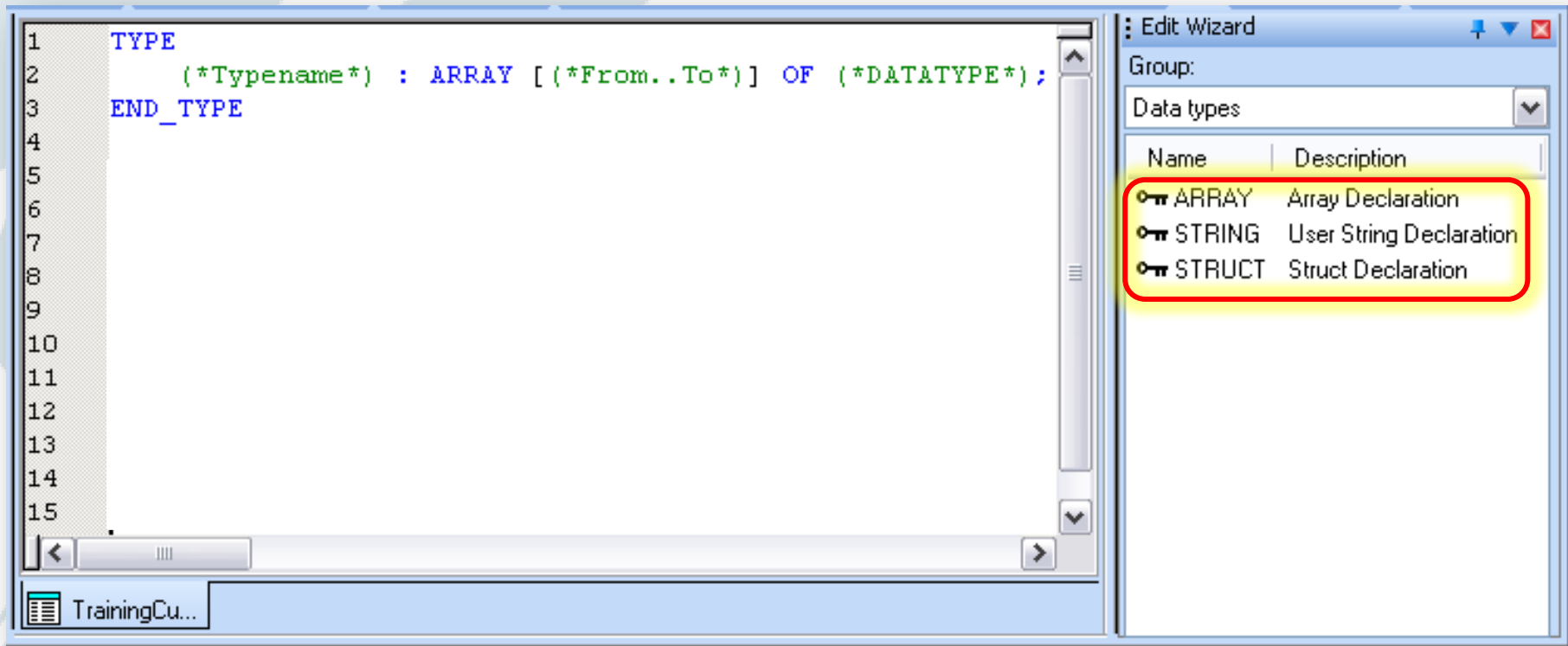
- *Arrays and Structures Defined as Data Type*
 - *Derived data types*



Best Practice: Prefix data type name

1. Avoid name conflict
2. Organization

- *Add ARRAY Data Type From Edit Wizard*



The screenshot displays a software interface with a code editor on the left and an 'Edit Wizard' dialog box on the right. The code editor shows the following code:

```
1 TYPE
2     (*Typename*) : ARRAY [(*From..To*)] OF (*DATATYPE*);
3 END_TYPE
4
5
6
7
8
9
10
11
12
13
14
15
```

The 'Edit Wizard' dialog box has a 'Group:' dropdown set to 'Data types'. Below this is a table with two columns: 'Name' and 'Description'. The table contains three entries, with the first one highlighted by a red box:

Name	Description
ARRAY	Array Declaration
STRING	User String Declaration
STRUCT	Struct Declaration

- *Compile (Make)*

```
TYPE
```

```
    YTTS TrainArray : ARRAY [0..7] OF WORD;
```

```
END_TYPE
```

- *Create Variable With Data Type*

	Name	Type	Usage
	+ System		
	- User Variables		
	G_TestVariable1	TrainArray	VAR_GLOBAL
	G_TestVariable2	TrainStruct	VAR_GLOBAL

- *Use Variable In Code*
 - *Array Element []*

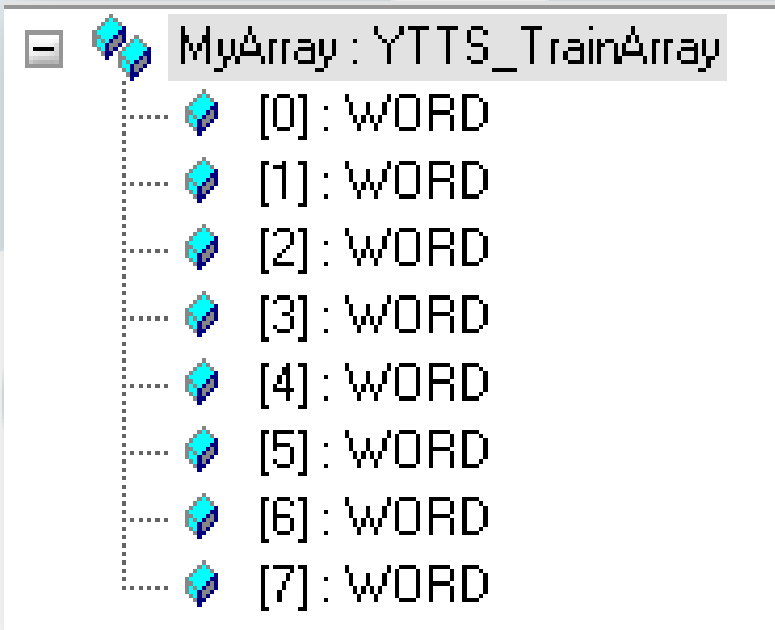
Variable Properties

Name:
G_TestVariable1[0]

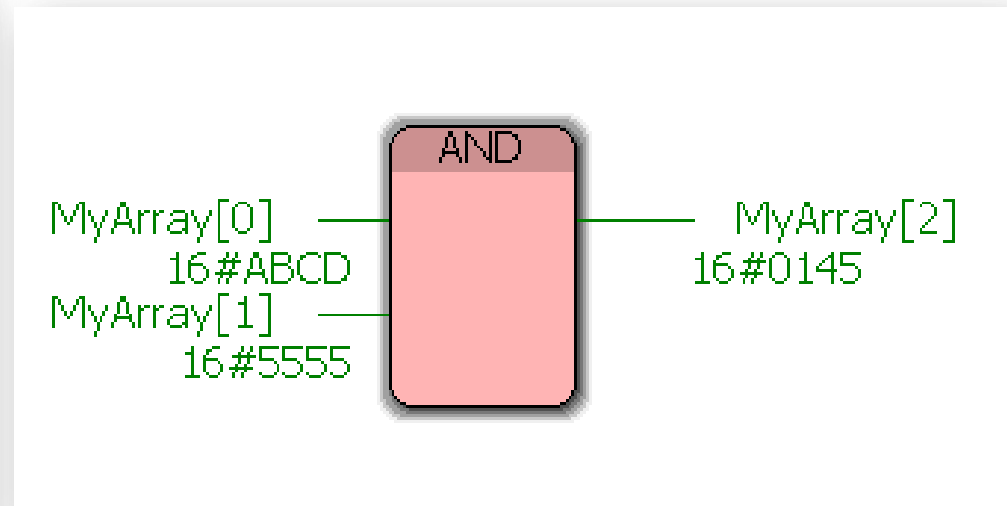
Data Type:
TrainArray

Usage:
VAR_GLOBAL RETAIN

- *Add to Watch Window*



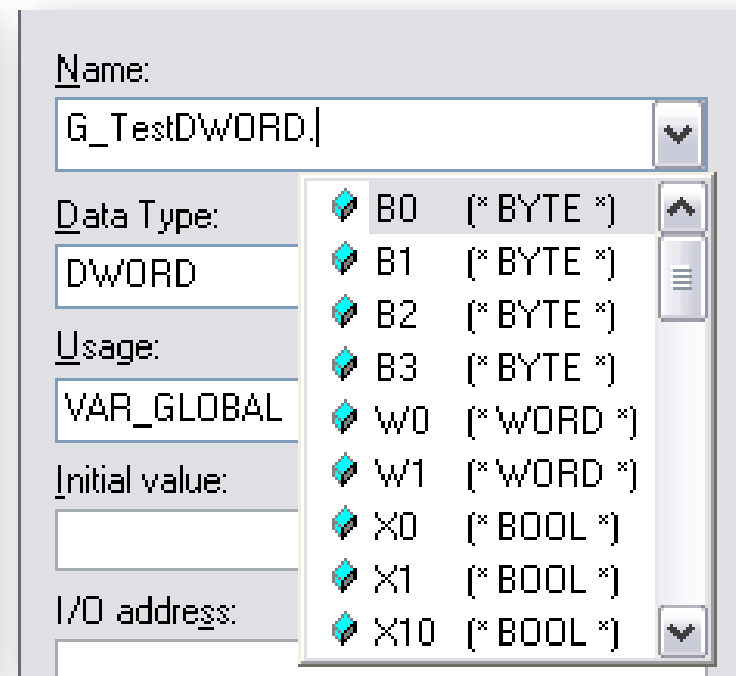
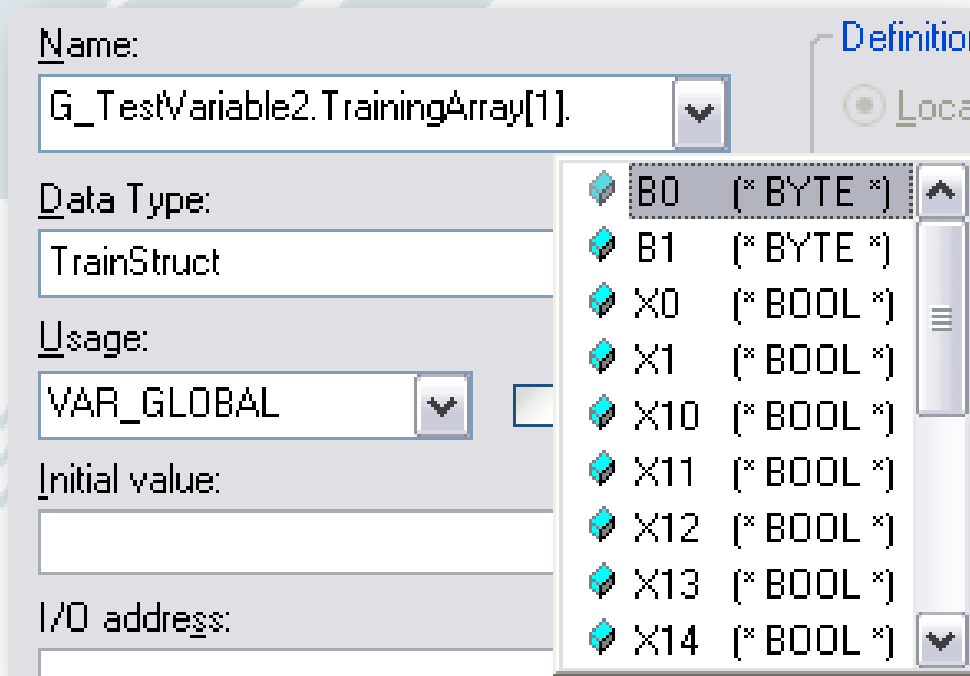
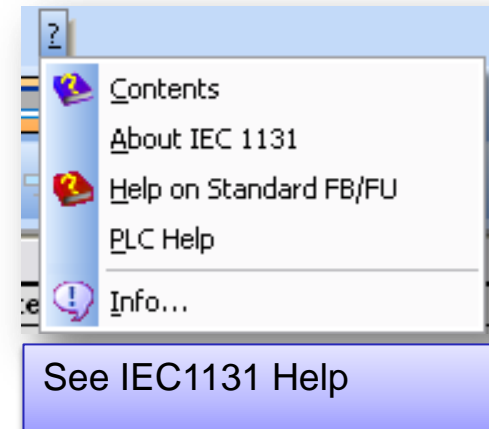
- *Use in Code*



```

n := 0; (* n is an integer *)
MyArray[n] := Var1 ; (* apply value to array element *)
n := n + INT#1; (* increment index variable *)
    
```

- “Bit String”
 - *BYTE, WORD, DWORD*
 - *Default Debug Display: Hexadecimal (0-F)*
 - *Use “dot” (.) to specify bit, byte, etc*



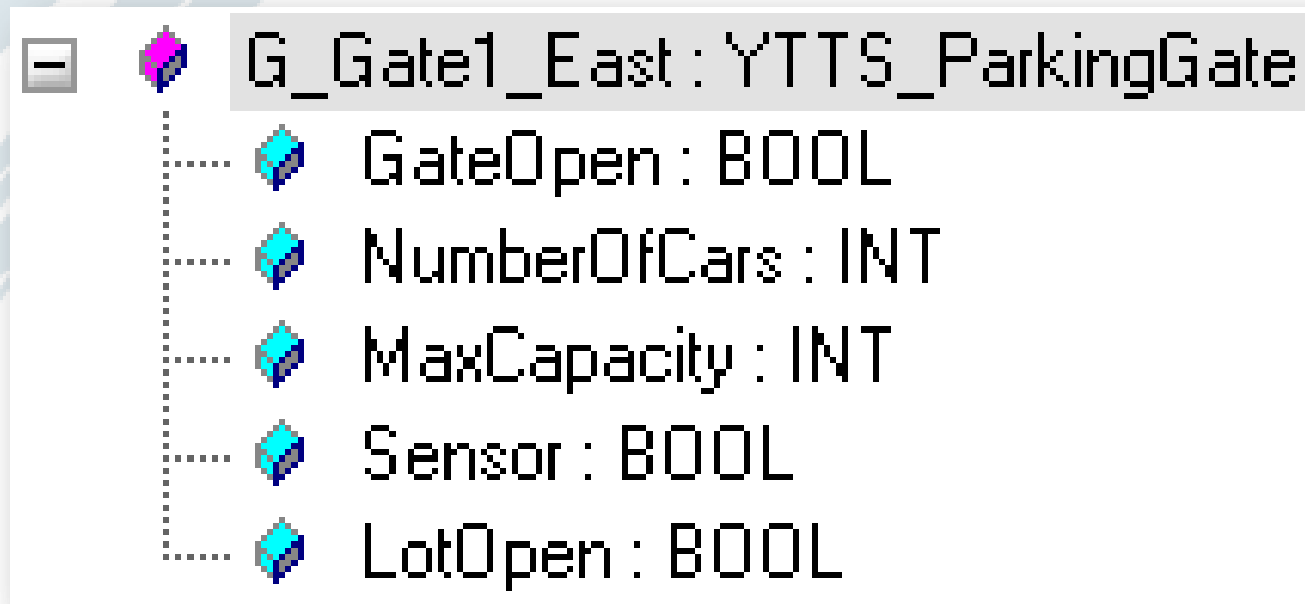
Structures

Definition

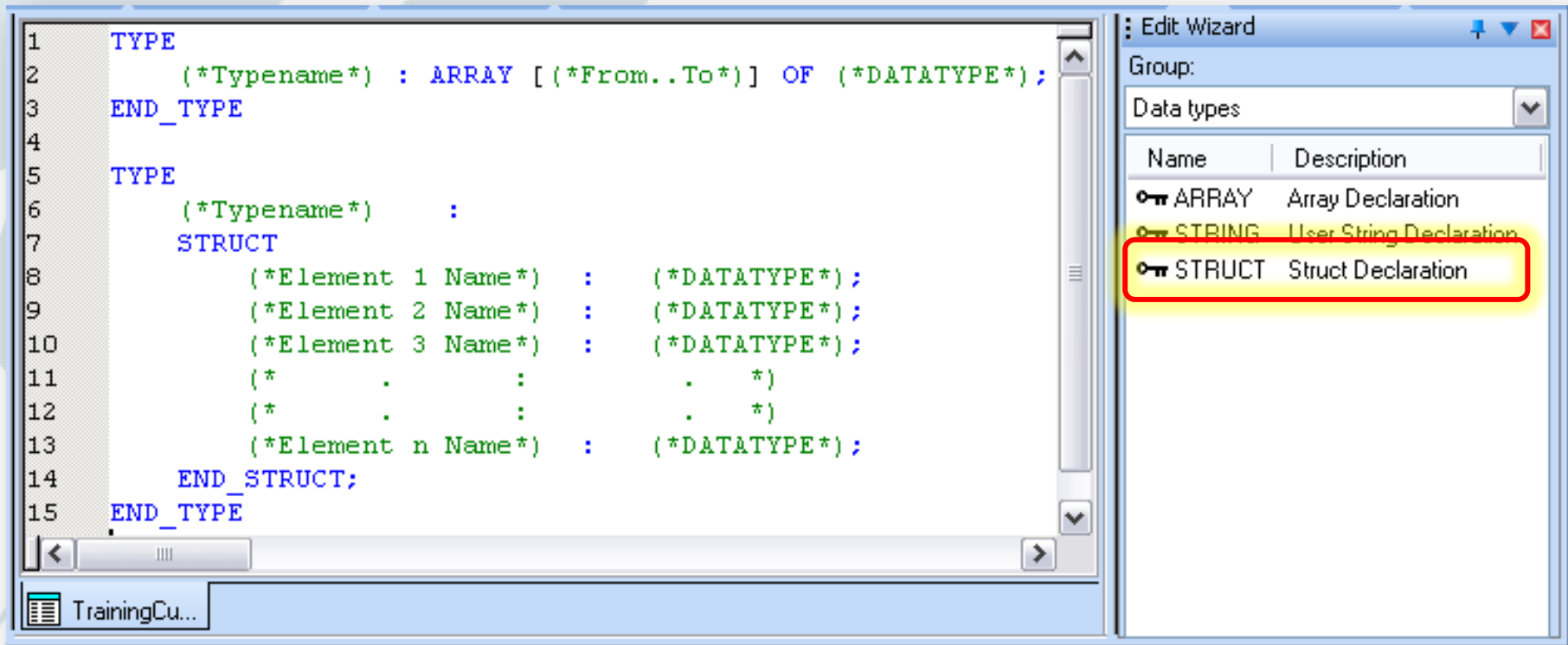
Create Data Type

Usage and Syntax

- *A set of data with mixed data types*
- *Similar to a file directory structure*
- *Organize Data*
- *Complex names (DOT notation)*
 - *G_Gate1_East.Sensor*



- *Add STRUCT Data Type From Edit Wizard*



The screenshot displays a software interface with two main panels. The left panel shows a code editor with the following code:

```
1 TYPE
2   (*Typename*) : ARRAY [(*From..To*)] OF (*DATATYPE*);
3 END_TYPE
4
5 TYPE
6   (*Typename*)      :
7   STRUCT
8     (*Element 1 Name*) : (*DATATYPE*);
9     (*Element 2 Name*) : (*DATATYPE*);
10    (*Element 3 Name*) : (*DATATYPE*);
11    (*      .      :      .      *)
12    (*      .      :      .      *)
13    (*Element n Name*) : (*DATATYPE*);
14  END_STRUCT;
15 END_TYPE
```

The right panel is titled "Edit Wizard" and shows a "Data types" list. The list has two columns: "Name" and "Description". The items are:

Name	Description
ARRAY	Array Declaration
STRING	User String Declaration
STRUCT	Struct Declaration

The "STRUCT" entry is highlighted with a red rectangle. The "Group:" dropdown is set to "Data types".

- *Compile (Make)*

```
TYPE
    YTTS_TrainArray : ARRAY [0..7] OF WORD;
END_TYPE
TYPE
    YTTS_TrainStruct :
    STRUCT
        Array8 : YTTS_TrainArray;
        Frequency : LREAL;
        RunCommand : BOOL;
        Parameter : UINT;
    END_STRUCT;
END_TYPE
```

- *Create Variable In List*

	Name	Type	Usage
	+ System		
	- User Variables		
	G_TestVariable1	TrainArray	VAR_GLOBAL
	G_TestVariable2	TrainStruct	VAR_GLOBAL

- *Use Variable In Code*

Variable Properties

Name: G_TestVariable2

Data Type: TrainStruct

Usage: VAR_GLOBAL

MyElement (* LREAL *)
 MyNumber (* UINT *)
 RunCommand (* BOOL *)
 Speed (* LREAL *)
 TrainingArray (* TrainArray *)

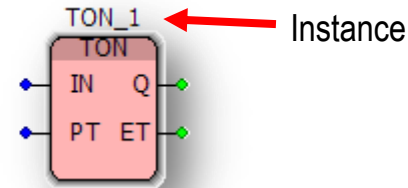
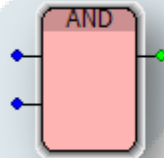
Functions & Function Blocks

Comparison
Bitwise Functions
Duplicate & Negate Inputs
EN/ENO

EN/ENO

duplicate & negate inputs

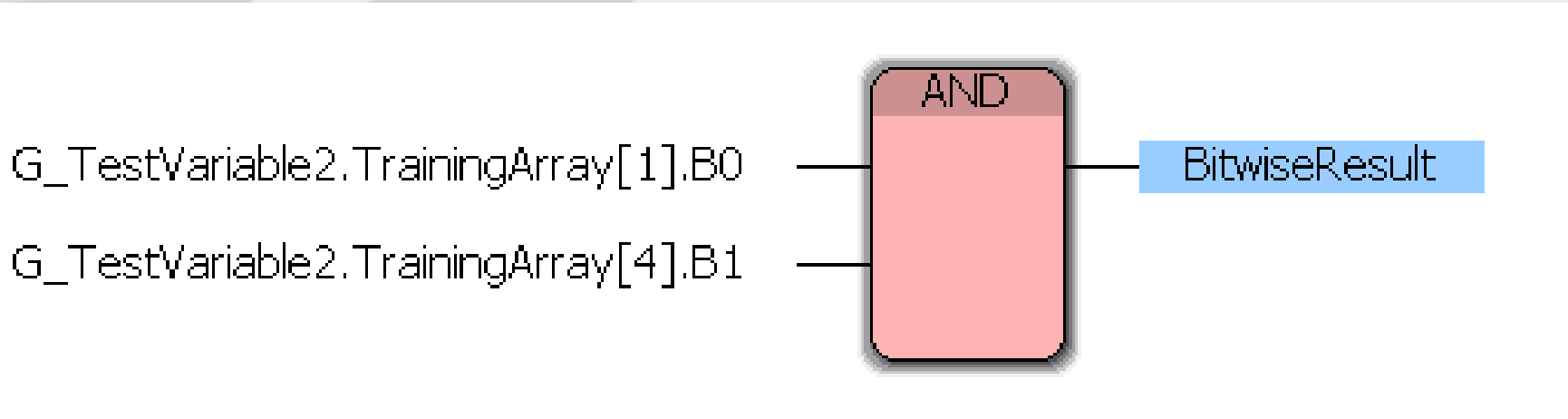
Function Block Definition (IEC 61131-3)



Function (FU)	Function Block (FB)
Executes in 1 scan	May require several scans to execute.
Executes without an "instance" of memory, as indicated by the icon	Each FB runs in a separate "instance" of memory, as indicated by the icon
Single Output required	Multiple Outputs possible
Must connect output and all inputs for compile	Only Var_In_Out must be connected to compile
Output value depends only on the function inputs	Output value may depend on values that are not part of the function block input
Function Blocks may not be used inside functions*	Other function blocks may be used within the function block*
Most efficient use of processor and RAM*	May use slightly more processor power and RAM to accomplish the same task*
VAR (local) variables only*	VAR_GLOBAL allowed, but not recommended*

* For user created FU or FB

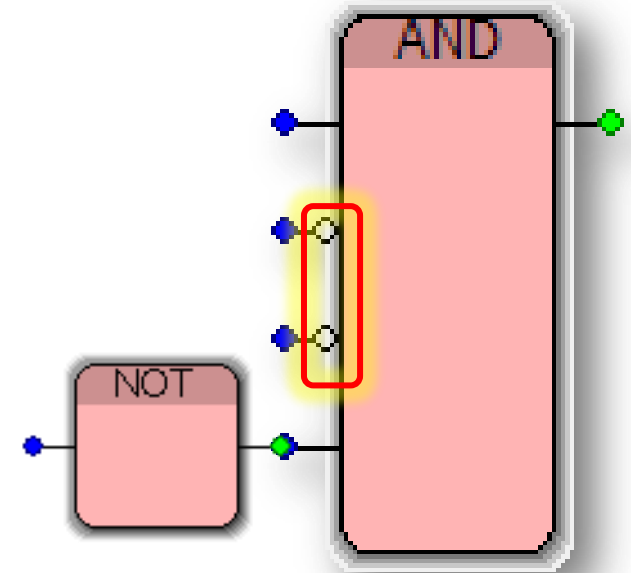
- *Functions operate at bit level*
- *Inputs require same data type*



	Bit (X)							
Data	X7	X6	X5	X4	X3	X2	X1	X0
Input 1	1	0	0	0	1	0	0	0
Input 2	0	0	1	0	1	0	0	1
Input n								
Output	0	0	0	0	1	0	0	0

- **Negate**
 - Same as NOT function
 - Inverts ALL bits

- **Duplicate**
 - Add Inputs
 - Does not apply to all Functions



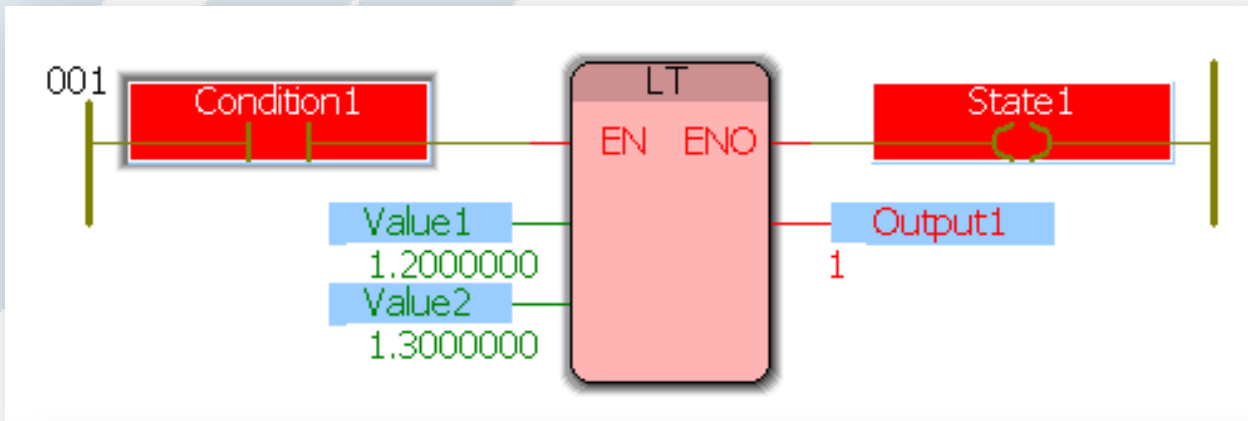
	Data type	Negated	Edge
	ANY_BIT	<input type="checkbox"/>	<input type="checkbox"/>
	ANY_BIT	<input checked="" type="checkbox"/>	<input type="checkbox"/>
	ANY_BIT	<input checked="" type="checkbox"/>	<input type="checkbox"/>
	ANY_BIT	<input type="checkbox"/>	<input type="checkbox"/>
	ANY_BIT	<input type="checkbox"/>	<input type="checkbox"/>

||||

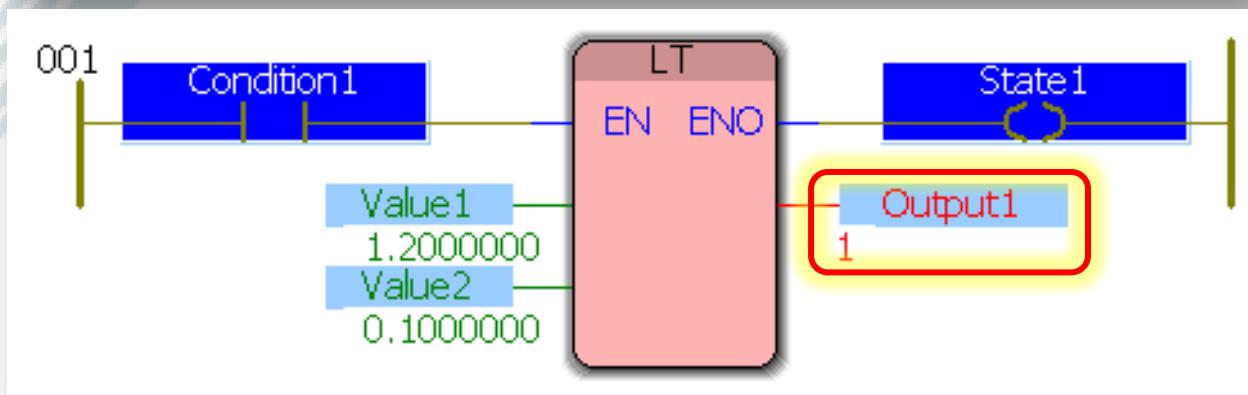
Duplicate

- **Definition: Enable / Enable Output**
 - Enable or disable execution
 - EN disconnect = Always enable execution
 - Normal operation → EN = ENO
 - Function error → EN = ON, ENO=OFF

See Yaskawa FAQ number
MTN-7PCQK4

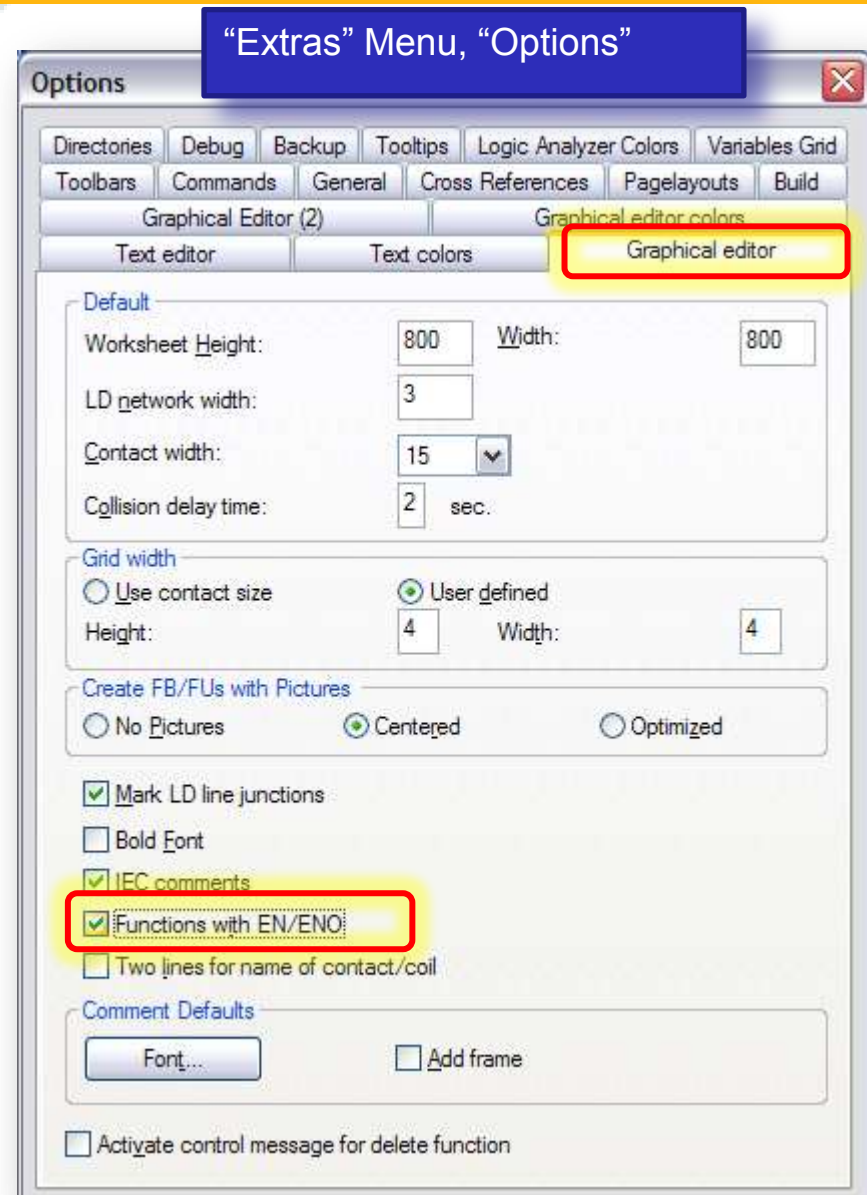
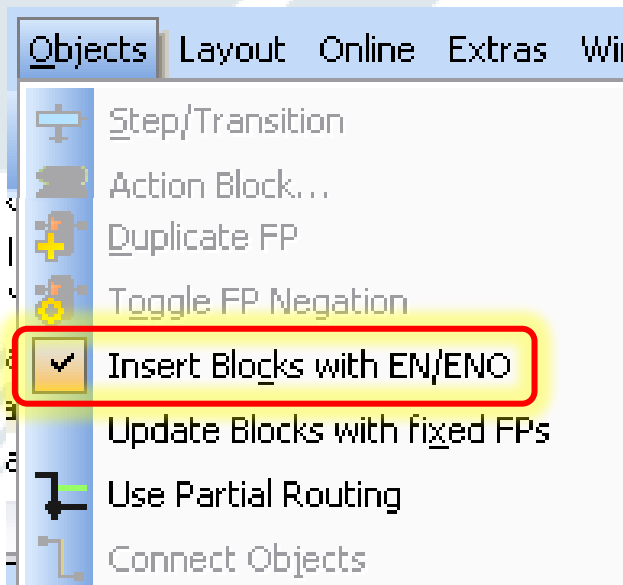


Function Enabled
Execution = TRUE
Output updates every scan

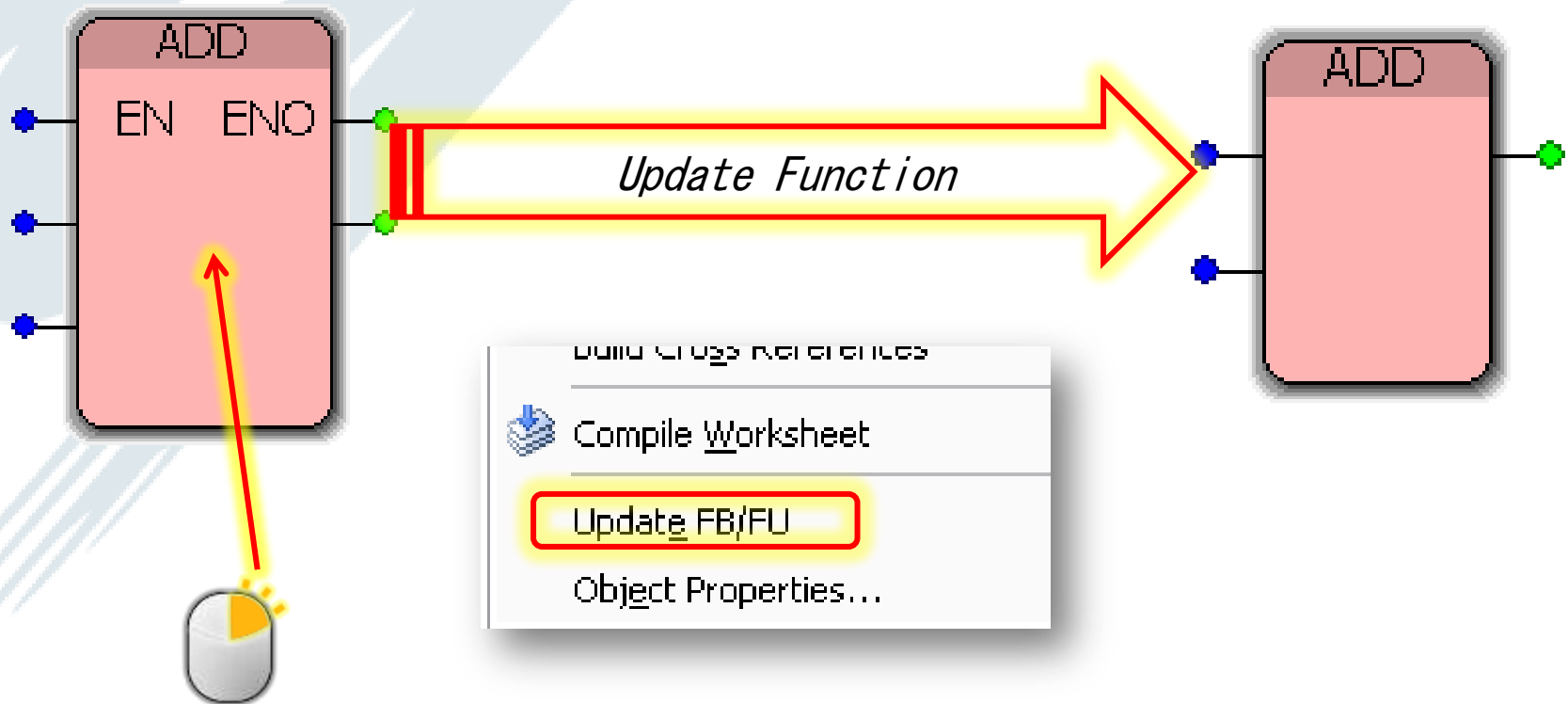


Function Disabled
Execution = FALSE
Output remains at previous state

- **Software Option**
 - Applies to all worksheets and new worksheets
 - Affects new functions only
- **Worksheet Option**
 - Active worksheet only
 - Affects new functions only



- *Manual Update of EN/ENO For Existing Functions*



User Function

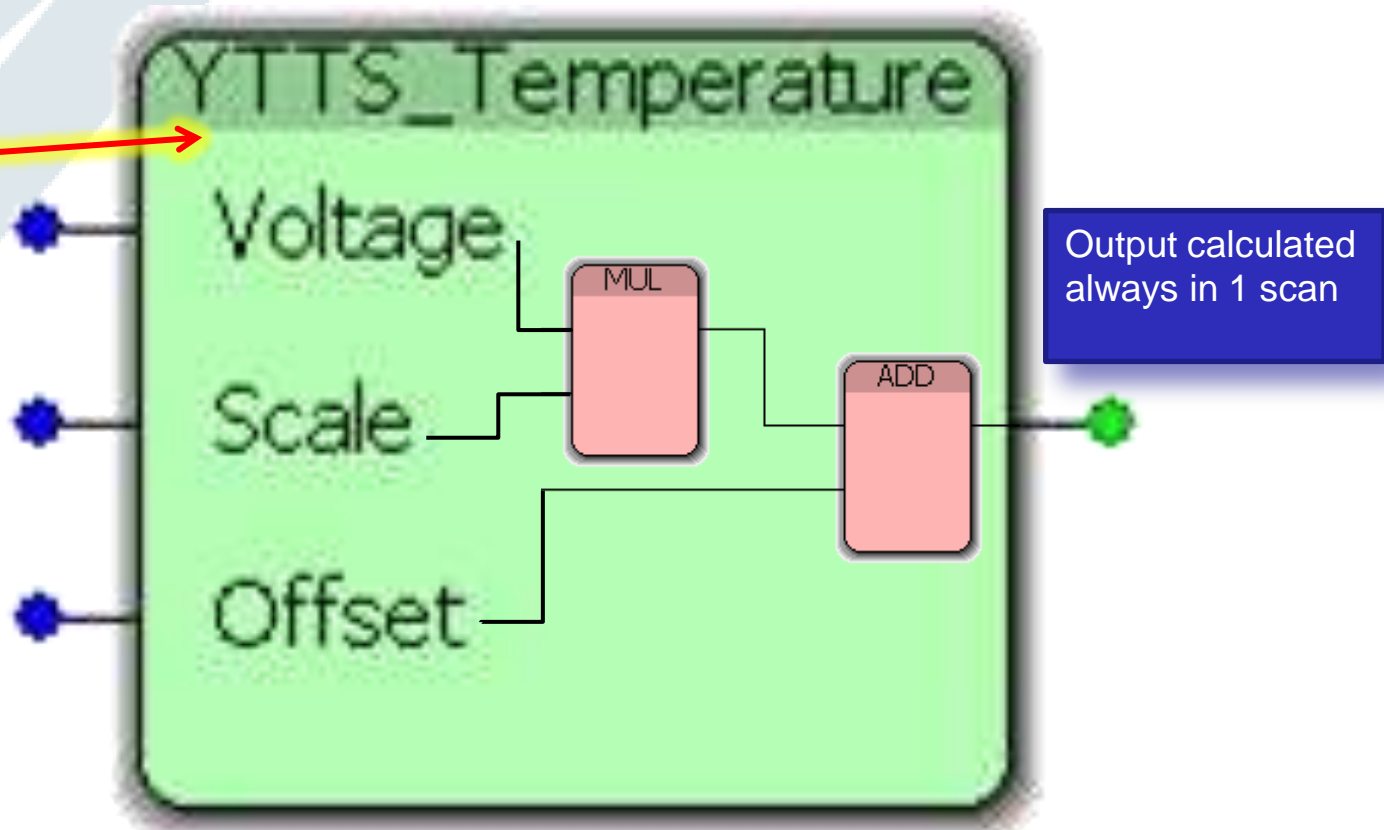
Overview & Concept
Write Code
Function in Program POU
PowerFlow
Function Troubleshooting

Function Troubleshooting

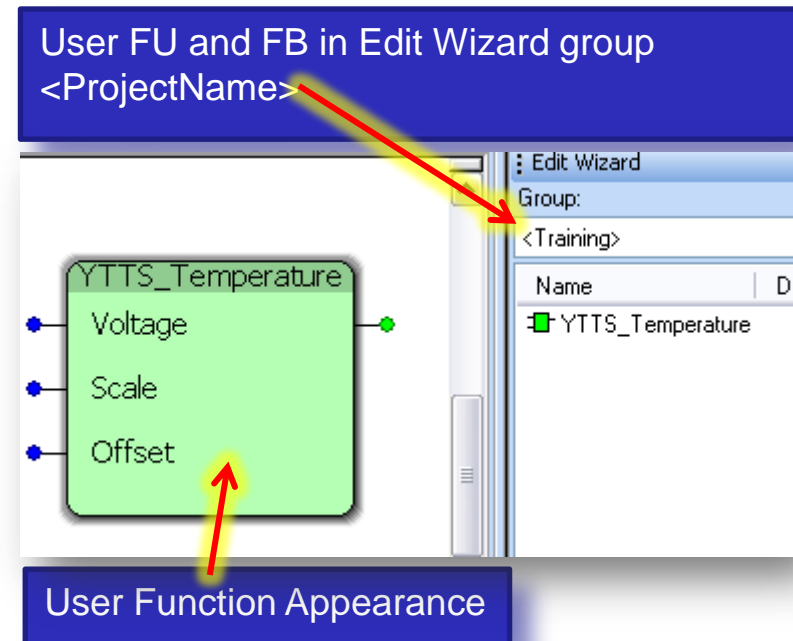
- *YTTS_Temperature*
 - *Convert analog input voltage to temperature*
 - » *Based on scale and offset*
 - » *Linear equation: $Y=mX+b$*

Best Practice:
Prefix function
name

1. Avoid name conflict
2. Organization



- *Write a User Function POU*
 - *Convert analog input voltage to temperature*
 - » *Based on scale and offset*
 - » *Linear equation: $Y=mX+b$*
 - *Temperature*
 - » *output of function*
 - *Scale*
 - » *input “slope” of function*
 - *Voltage*
 - » *raw input voltage*
 - *Offset*
 - » *calibration offset*



- *New Function POU*

- *Name:*
 - » *YTTS_Temperature*
- *Type:*
 - » *Function*
- *Language:*
 - » *LD*
- *Datatype of return value:*
 - » *LREAL*

Insert

Name: YTTS_Temperature

Type

Program

Function

Function Block

Action

Transition

Step

Worksheet

Language

IL

SI

SFC

FBD

LD

FFLD

MSFC

VAR

Data Types

Description

Use Reserve

Mode

Insert

Append

Datatype of return value (Function name will be the VAR. OUTPUT):

LREAL

PLC type: <independent>

Processor type: <independent>

OK

Cancel

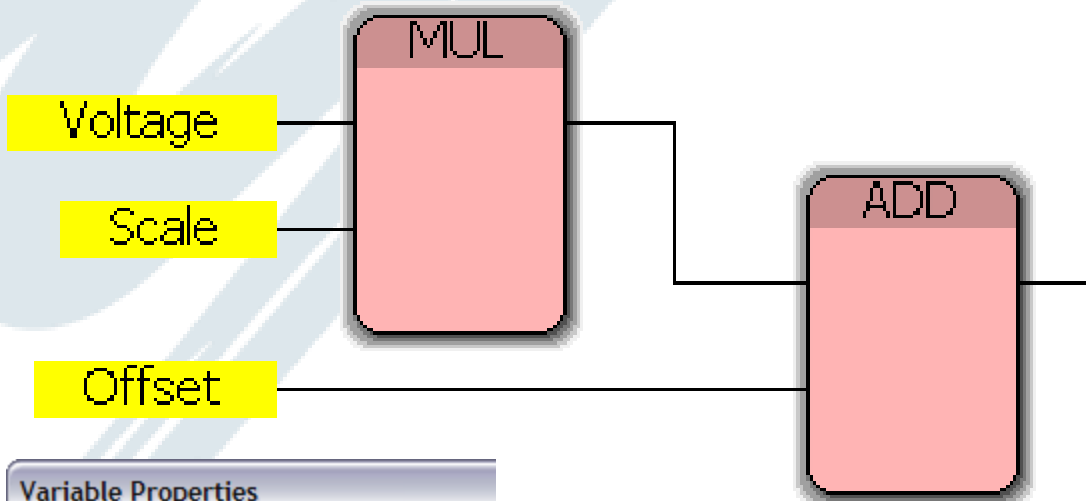
Help

- *Inputs*

- Usage: VAR_INPUT
- Datatype: LREAL

- *Output*

- Choose function name



Another variable may not use this name!

YTTS_Temperature

Variable Properties

Name: Offset

Data Type: LREAL

Usage: VAR_INPUT RETAIN

Initial value:

Variable Properties

Name: V000

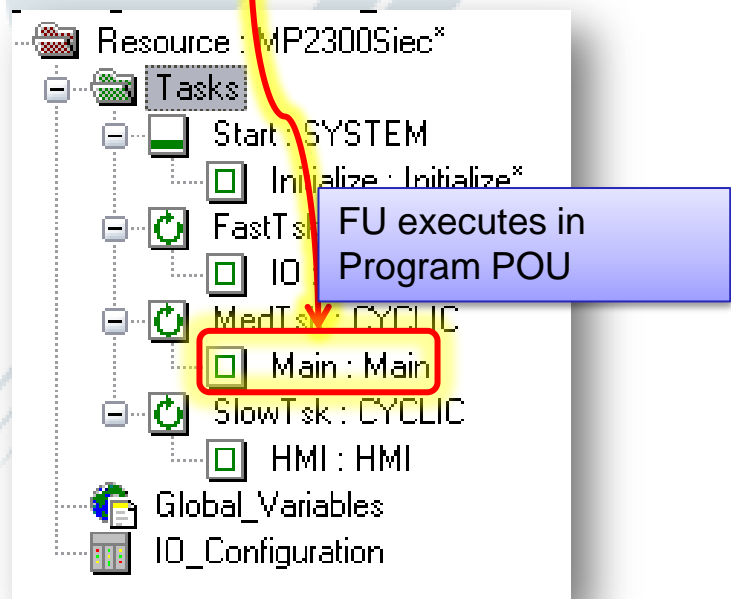
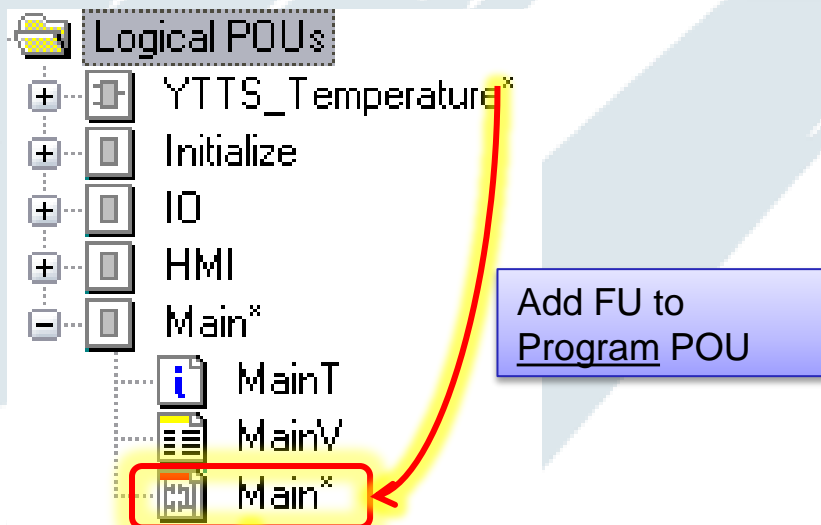
ExternalEncoder

LeftMotor _

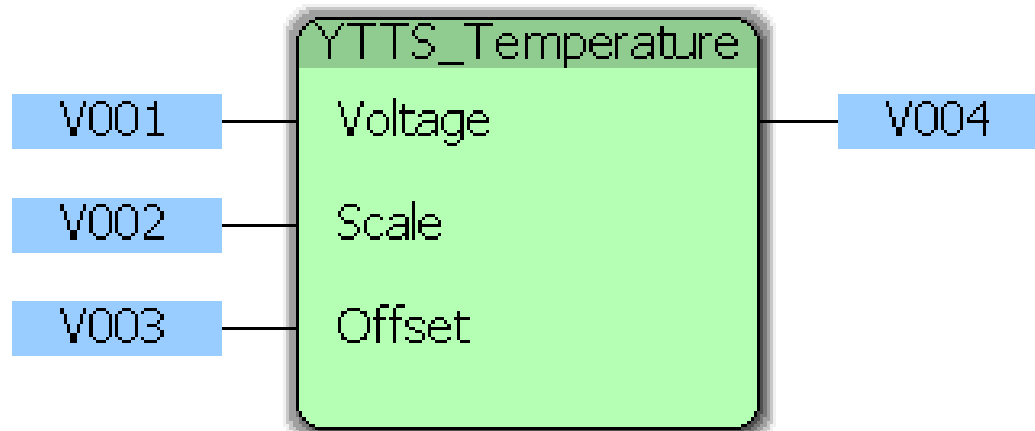
Scale

Voltage

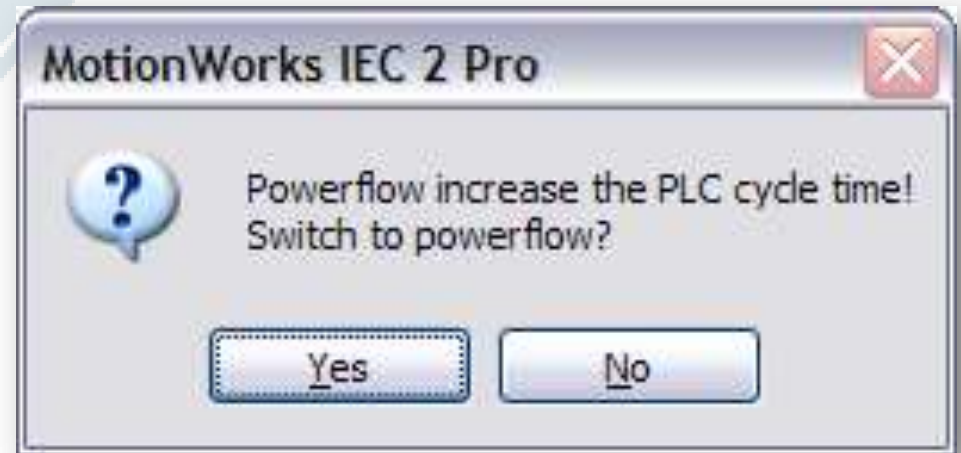
YTTS_Temperature



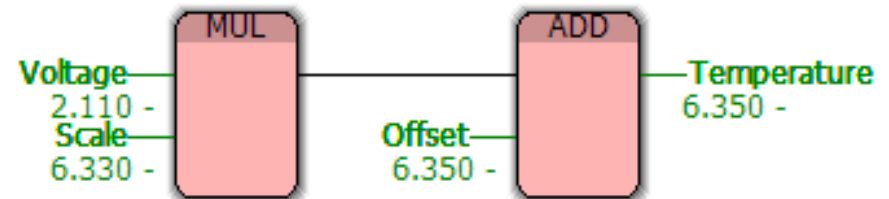
- **MAKE**
- *Execute YTTS_Temperature in Main Program POU*
 - Open logical POU "Main"
 - Add Function
 - » Edit wizard group <project name>
 - Connect variables
 - Download Changes
 - Confirm output



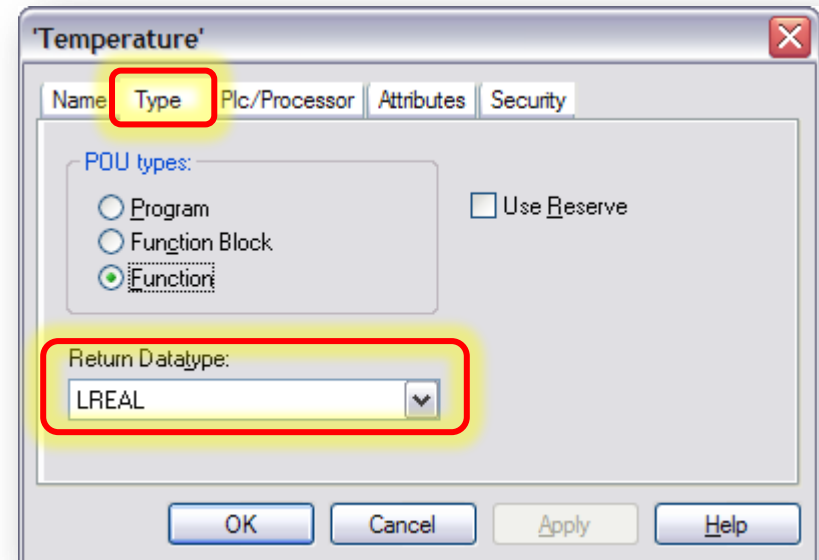
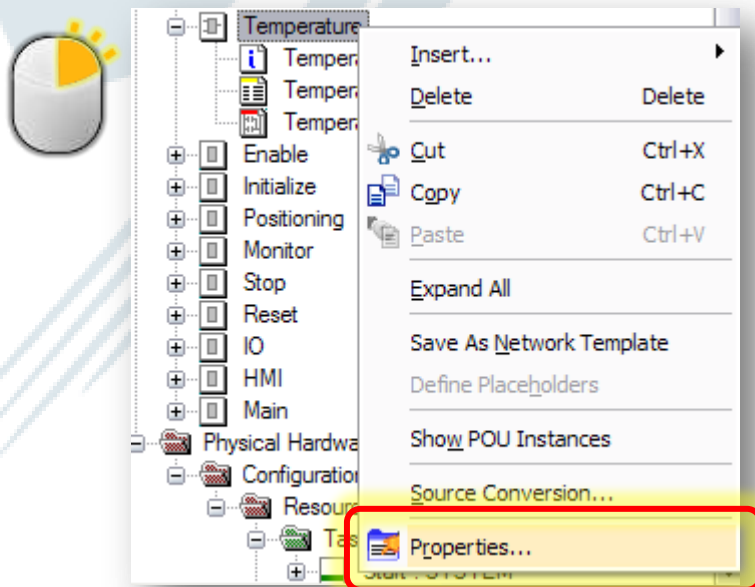
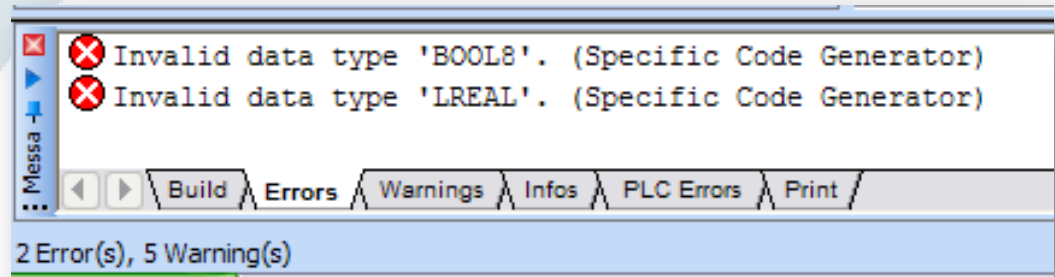
- *PowerFlow Prompt*
 - *Debug feature*
 - *Condition: function worksheet open*
 - *ST language support*
 - » *monitor lines executed*
 - *LD , FBD not supported*
 - » *Incorrect display*
 - » *Answer "No"*



Click **No** to disable this feature



- *Return Datatype*
 - *Return Datatype may be incorrect*

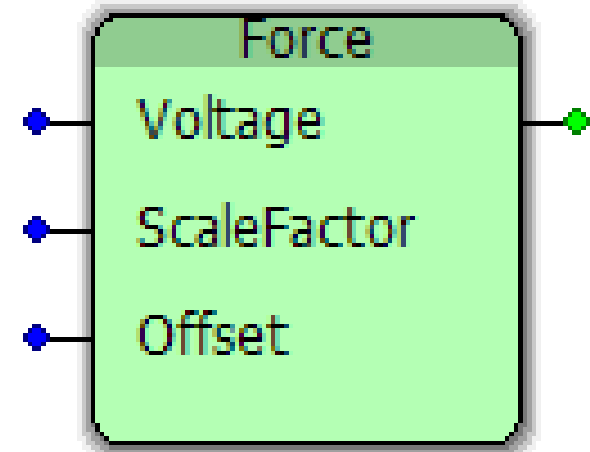


Create the same function using ST language

- Force = Voltage * ScaleFactor + Offset

HINTS:

1. This can be done in 1 line of ST
2. Use F5 to declare variables in ST
3. Refer to the Initialize POU for syntax;



User Function Block

Overview & Concept
Write Code
Implement Instance
Troubleshooting
Enable Input
Customized Help

Customized Help
Enable Input

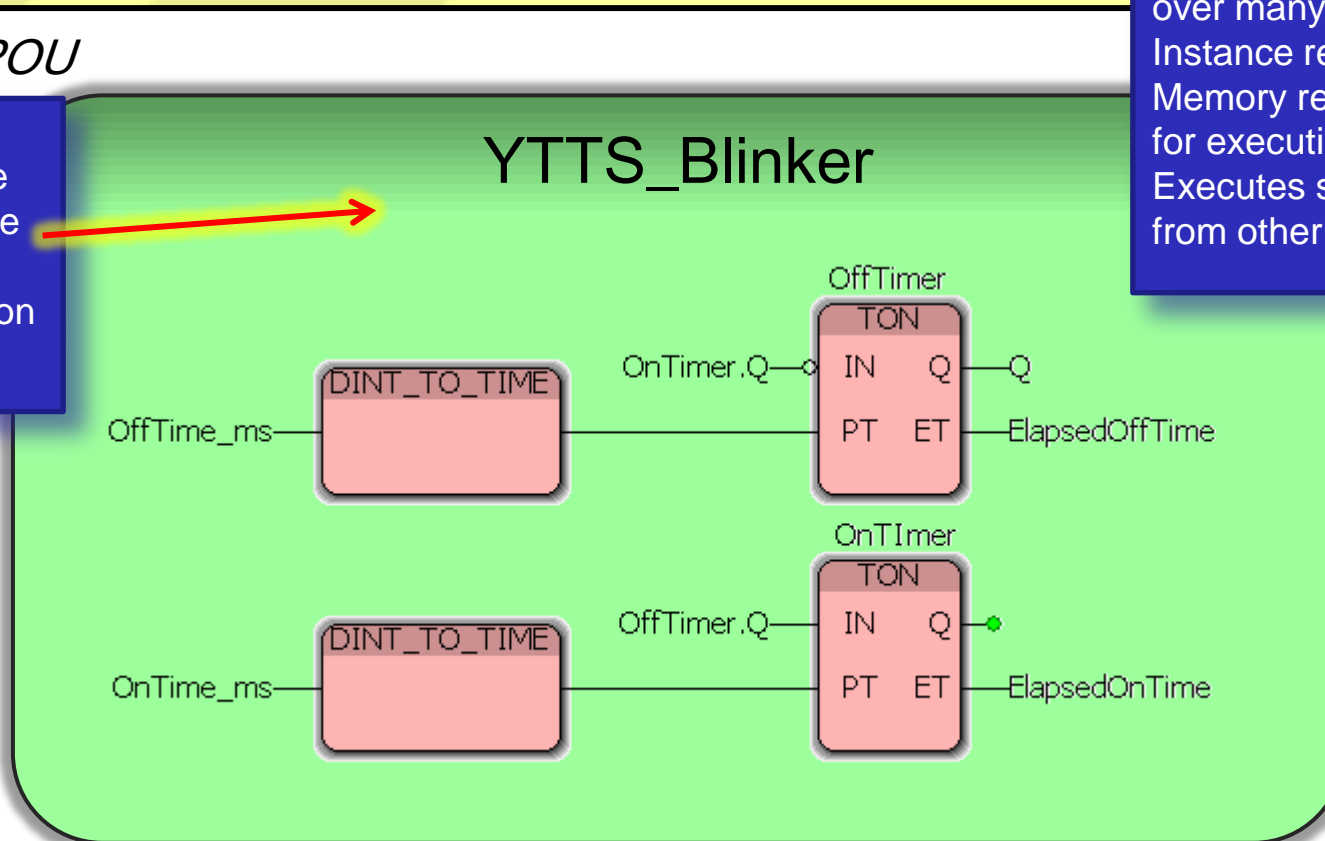
- *YTTS_Blinker*
 - *Run in program POU*

• *Main POU*

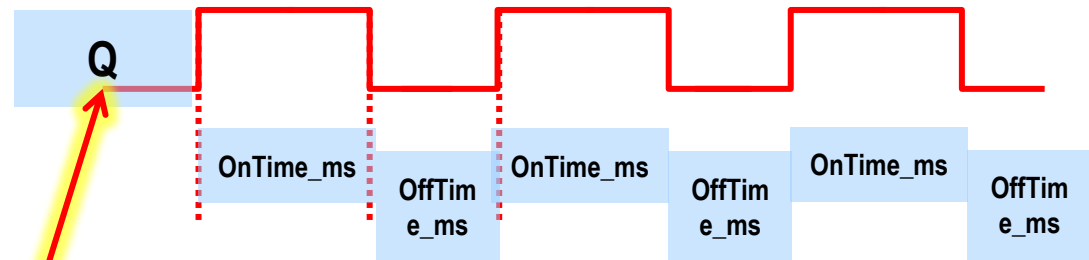
Best Practice:
Prefix FB name

1. Avoid name conflict
2. Organization

Outputs calculate over many scans.
Instance required.
Memory reserved for execution.
Executes separately from other instances



- Write a User Function Block POU
 - “Q” output cycles according to input OnTime and OffTime
 - Inputs
 - » OffTime_ms
 - » OnTime_ms
 - Outputs
 - » Q



The screenshot shows the SIMATIC Manager software interface. On the left, a function block named YTTTS_Blinker_1 is shown with two input ports: OffTime_ms and OnTime_ms. The output port is labeled Q. A red arrow points from the Q output port to the timing diagram above. On the right, the Edit Wizard dialog is open, showing a list of function blocks. The list includes ADD, AND, CTD, CTU, and CTUD. The description for each block is provided.

Name	Description
ADD	Addition
AND	Bitwise AND
CTD	Counter Down
CTU	Counter Up
CTUD	Counter Up/Down

User Function Block

Write Function Block Code

YTTS_BlinkerV:YTTS_Blinker

Name	Type	Usage	Init	Description	Address	Retain
Default						
OffTimer	TON	VAR				<input type="checkbox"/>
OnTimer	TON	VAR				<input type="checkbox"/>
ElapsedOffTime	TIME	VAR				<input type="checkbox"/>
ElapsedOnTime	TIME	VAR				<input type="checkbox"/>
OffTime_ms	DINT	VAR_INPUT	DINT#1000			<input type="checkbox"/>
OnTime_ms	DINT	VAR_INPUT	DINT#1000			<input type="checkbox"/>
Q	BOOL	VAR_OUTPUT				<input type="checkbox"/>

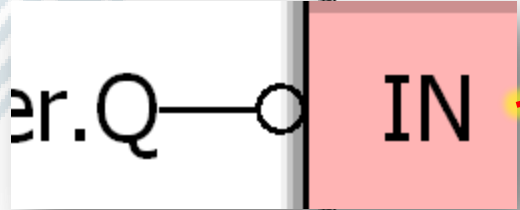
YTTS_BLINKER:YTTS_Blinker* - Configuration.Resource.Fast.Main.YTTS_Blinker_1.YTTS_Blinker

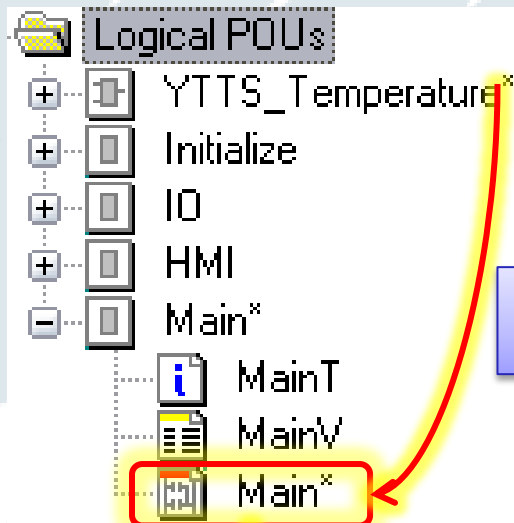
*.Q notation for direct connection

Instance name can be changed

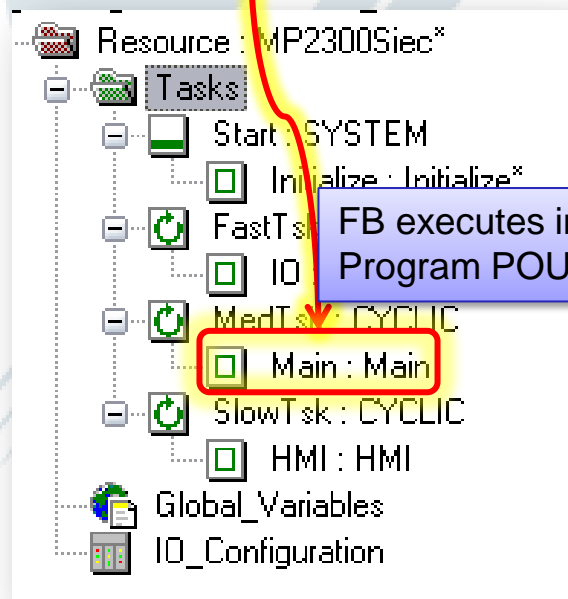
HINT: Check the Type, Usage, Init and Retain of each variable

Best Practice: No Global Variables in a Function Block





Add FB to Program POU



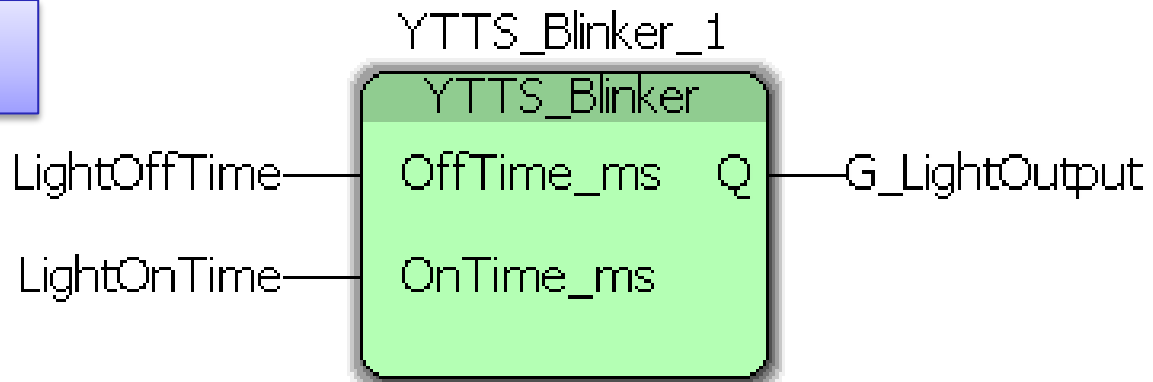
FB executes in Program POU

Execute YTTT_Blinker

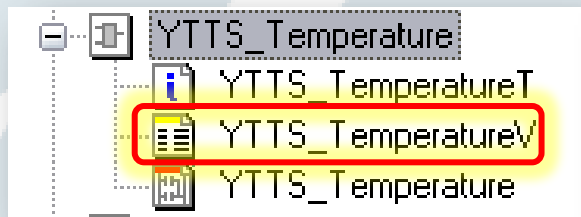
- Add to Main Program POU
- Connect variables
- Confirm output

Extra Credit

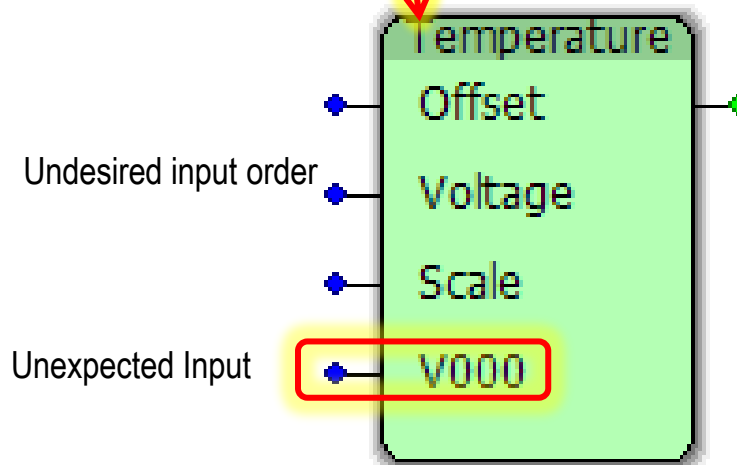
- Drive simulator output
- Use best practice for IO



- *Local Variable List*
 - *Defines Inputs and appearance*

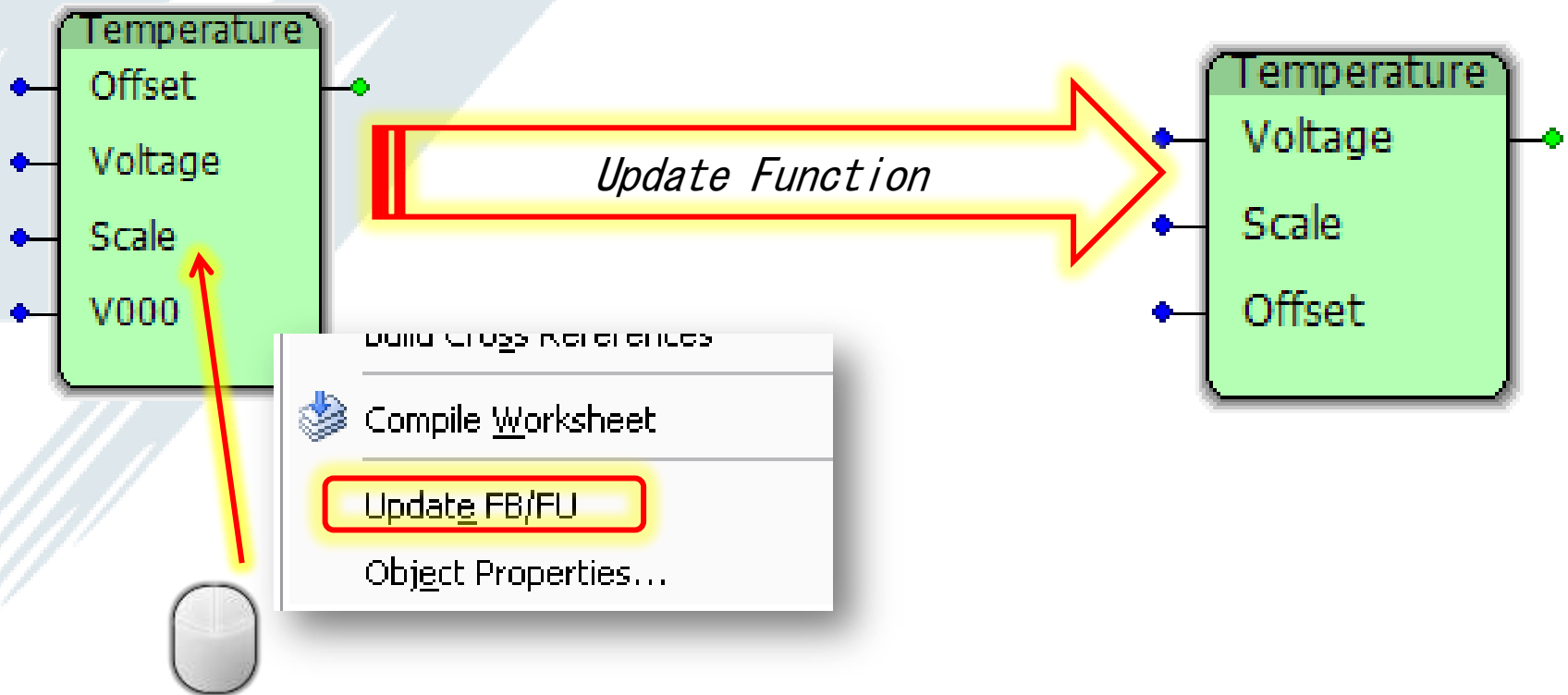


Name	Type	Usage	Description
[-] Default			
Offset	LREAL	VAR_INPUT	Temperature offset for calibration
Voltage	LREAL	VAR_INPUT	Raw input voltage to controller
Scale	LREAL	VAR_INPUT	Scale factor to convert volts to temperature
V000	BOOL	VAR_INPUT	oops! I didn't mean to add this variable!

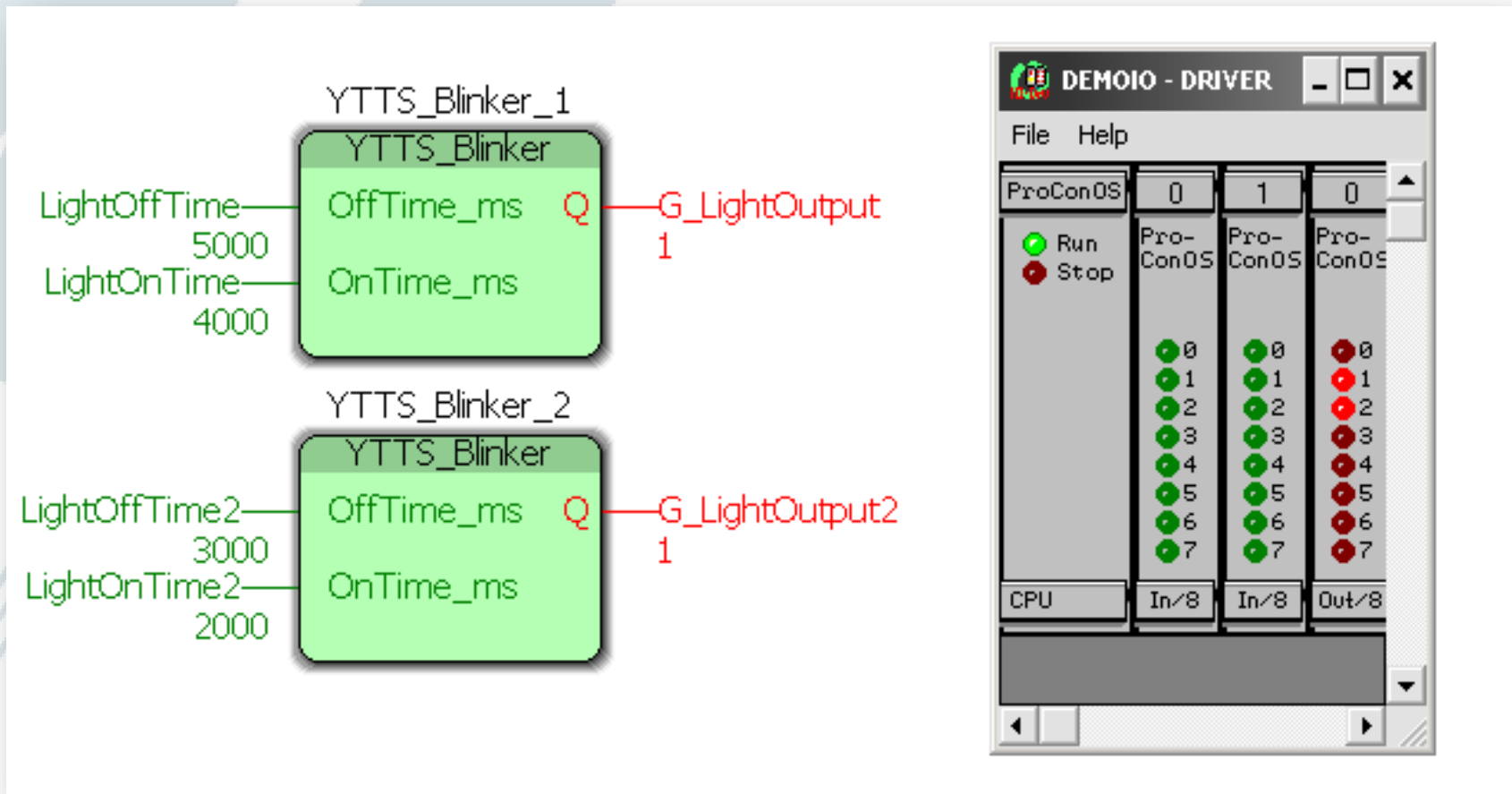


Order in list = Order of appearance

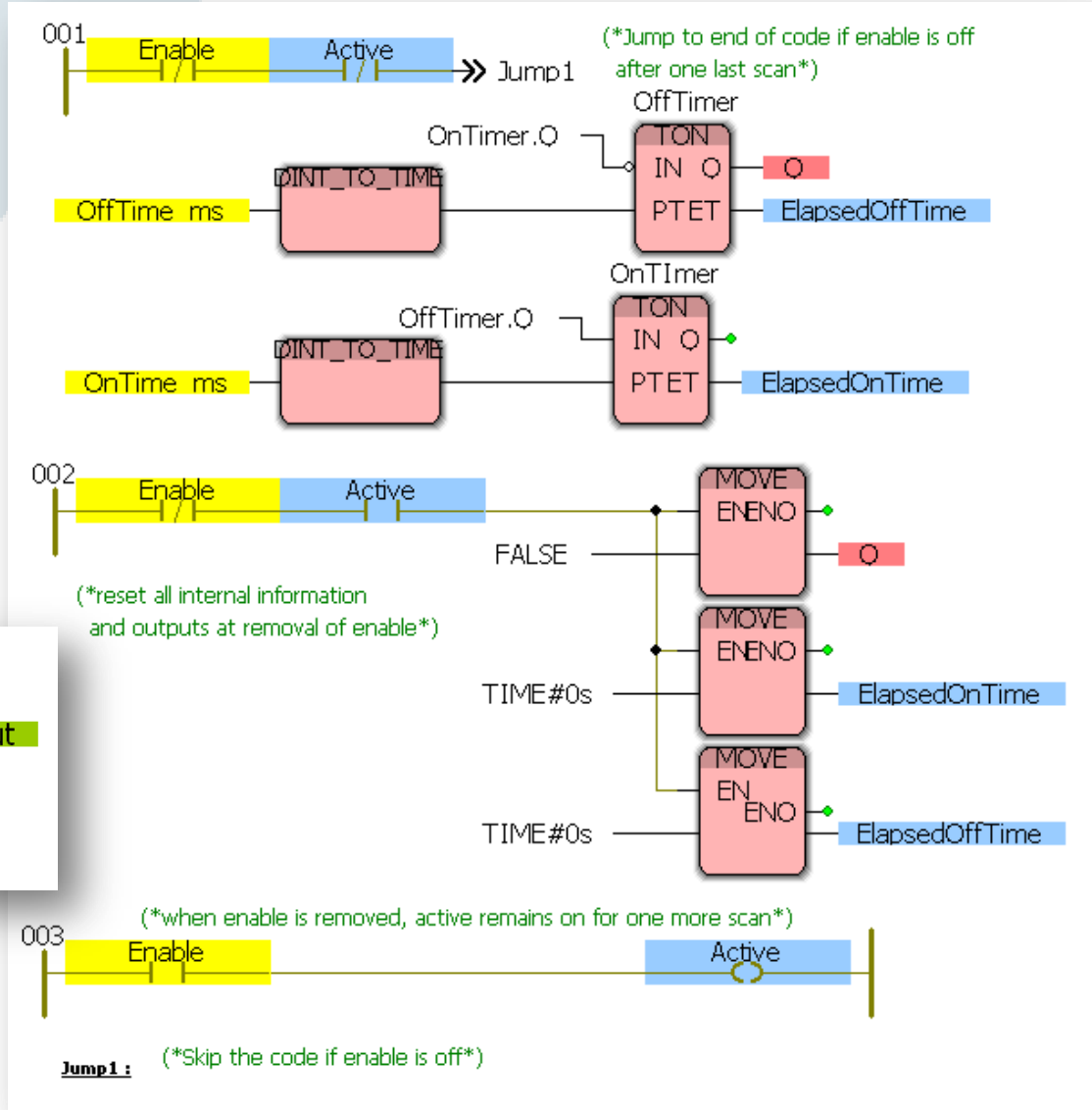
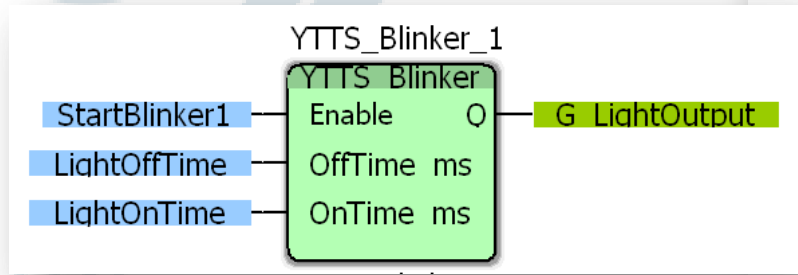
- *Make Project*
- *Right-Click*
 - *Update FB/FU*



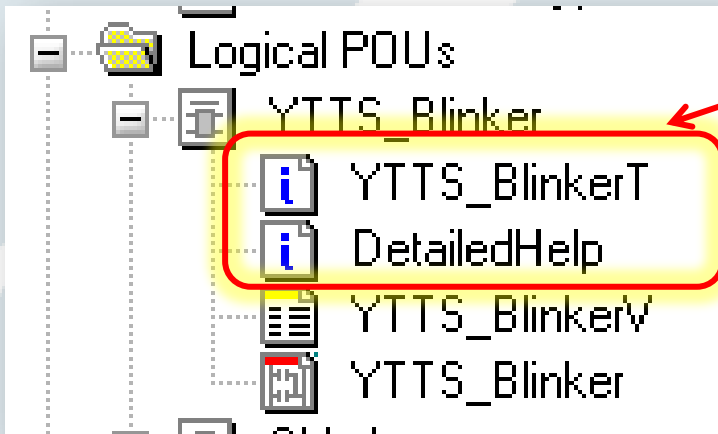
- *Insert YTTS_Blinker into the Main program POU*
 - *Cause Two outputs to blink independently*



- *Jump to Label*
 - For efficiency
- *Active*
 - To clear data after execute



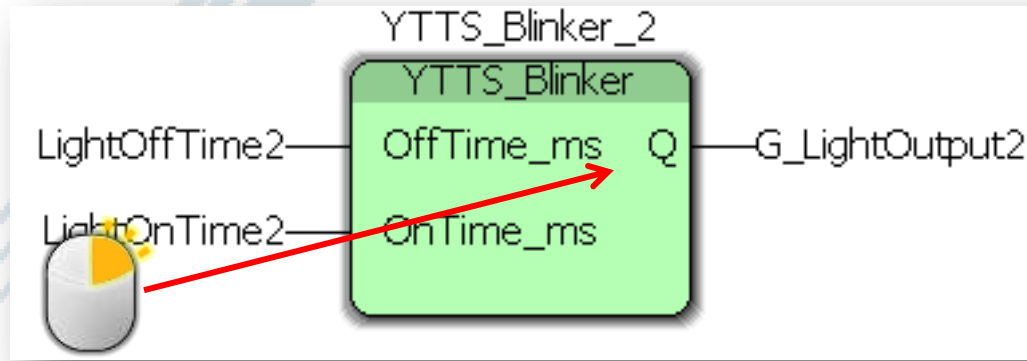
- Option 1: Description Worksheet in Function Block Program



Help = Text from "Description Worksheet "

Help of POU YTTTS_Blinker

"Q" output cycles according to

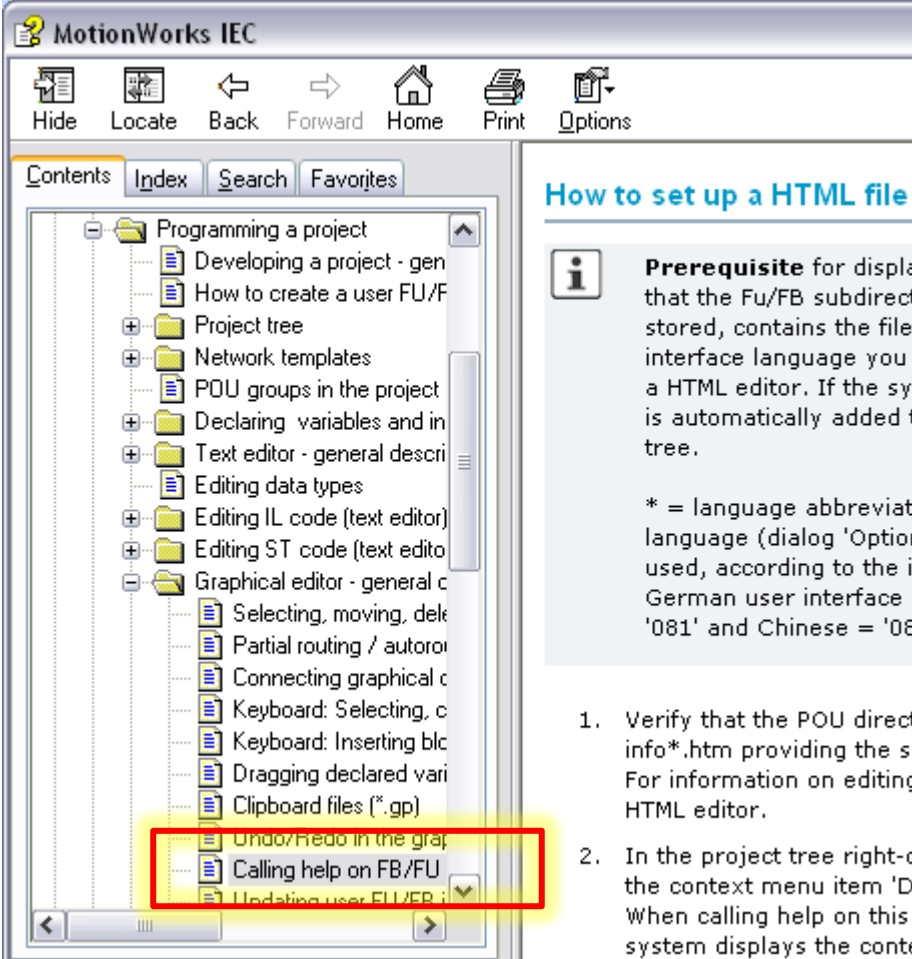


Help on FB/FU

Add to Favorites

Show Descriptions

- *Option 2: *.HTML file*
 - *See Motionworks IEC Help*
 - *Search "HTML"*
 - *Graphics*
 - *Language Codes*



The screenshot shows the MotionWorks IEC help interface. The title bar reads "MotionWorks IEC". The navigation bar includes icons for Hide, Locate, Back, Forward, Home, Print, and Options. Below the navigation bar are tabs for Contents, Index, Search, and Favorites. The main content area is divided into two panes. The left pane shows a tree view of the help content, with the following items listed: Programming a project, Developing a project - gen, How to create a user FU/F, Project tree, Network templates, POU groups in the project, Declaring variables and in, Text editor - general descri, Editing data types, Editing IL code (text editor), Editing ST code (text edito, Graphical editor - general c, Selecting, moving, del, Partial routing / autoroi, Connecting graphical c, Keyboard: Selecting, c, Keyboard: Inserting blc, Dragging declared vari, Clipboard files (*.gp), Undo/Redo in the gra, Calling help on FB/FU, and Updating user FU/EB. The "Calling help on FB/FU" entry is highlighted with a red box. The right pane shows the article "How to set up a HTML file". It includes an information icon and a prerequisite section: "Prerequisite for displ that the Fu/FB subdirect stored, contains the file interface language you a HTML editor. If the sy is automatically added t tree." Below this is a note: "* = language abbreviat language (dialog 'Optio used, according to the i German user interface '081' and Chinese = '08". At the bottom of the right pane, there are two numbered steps: "1. Verify that the POU direct info*.htm providing the s For information on editing HTML editor." and "2. In the project tree right-c the context menu item 'D When calling help on this system displays the conte".

User Libraries

Process Overview

Create a Library

Use a Library

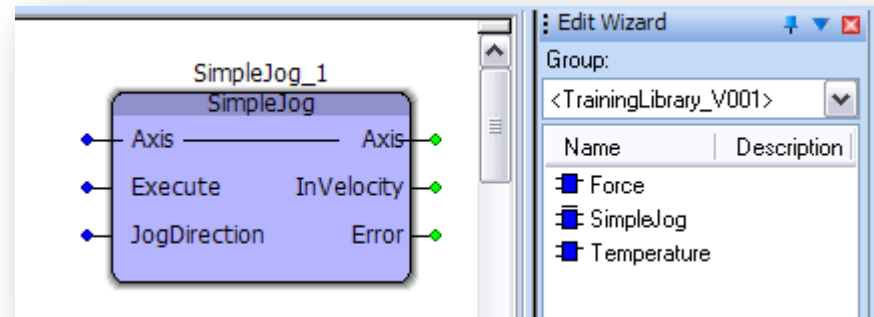
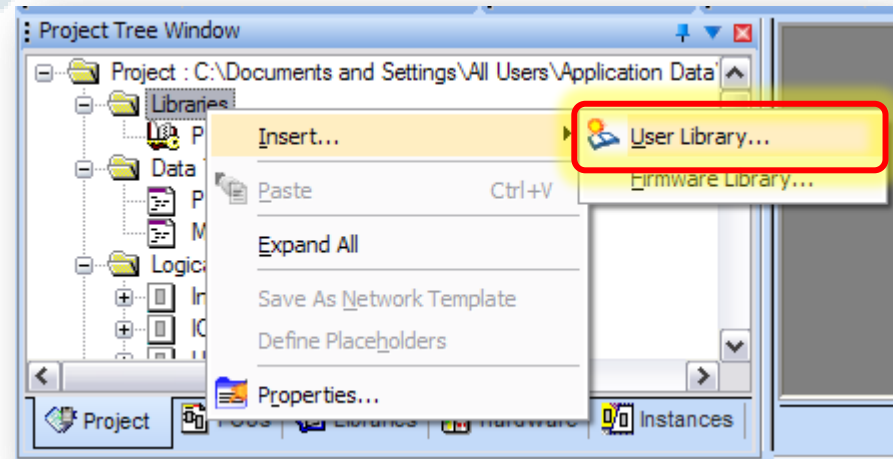
Zip / Unzip

Delete Datatypes

Delete Datatypes

Zip / Unzip

- *Insert Another Project*
 - *Library = any project*
 - **.mwt (or *.mwe)*
- *Library Data Imported*
 - *User FU & FB POU's*
 - *Program POU's*
 - *Data Types*
 - *NOT global variables!*
- *Organization*
 - *Specific projects for library use*
 - *Revision number in project name*
 - *Prefix (ex: YTTS_)*



- Refer to Quick Reference Guide
 - Save project in "Libraries" folder under new name
 - Clean up and make the project
 - Save as ZIP for reference

YASKAWA
Quick Reference Guide
MPiC Series Controllers

Contents

Startup Procedures

- Set the Front Panel Switches
- Open a saved project
- Start a new project:
 - Establish ethernet communication with controller (web interface)
- Go online with controller
- Download project to controller
- Set program to auto-start
- Reset an MP23025ec system back to factory settings?
- Confirmation is possible on each axis
- Tune the motors

Maintenance Procedures

- Update the firmware
- Determine/Set unknown IP address in controller
- Extract/Open the project from controller
- Replace the controller
- Replace the servopack
- Replace the servo motor

Analysis & Troubleshooting

- Monitor Programs
- Watch Variables in a List (Monitor window)
- Use Logic Analyzer (AKA: graph, trace, scope)
- Diagnose Alarm Status

Modbus/TCP and Ethernet/IP User Libraries OPC Server

User Libraries

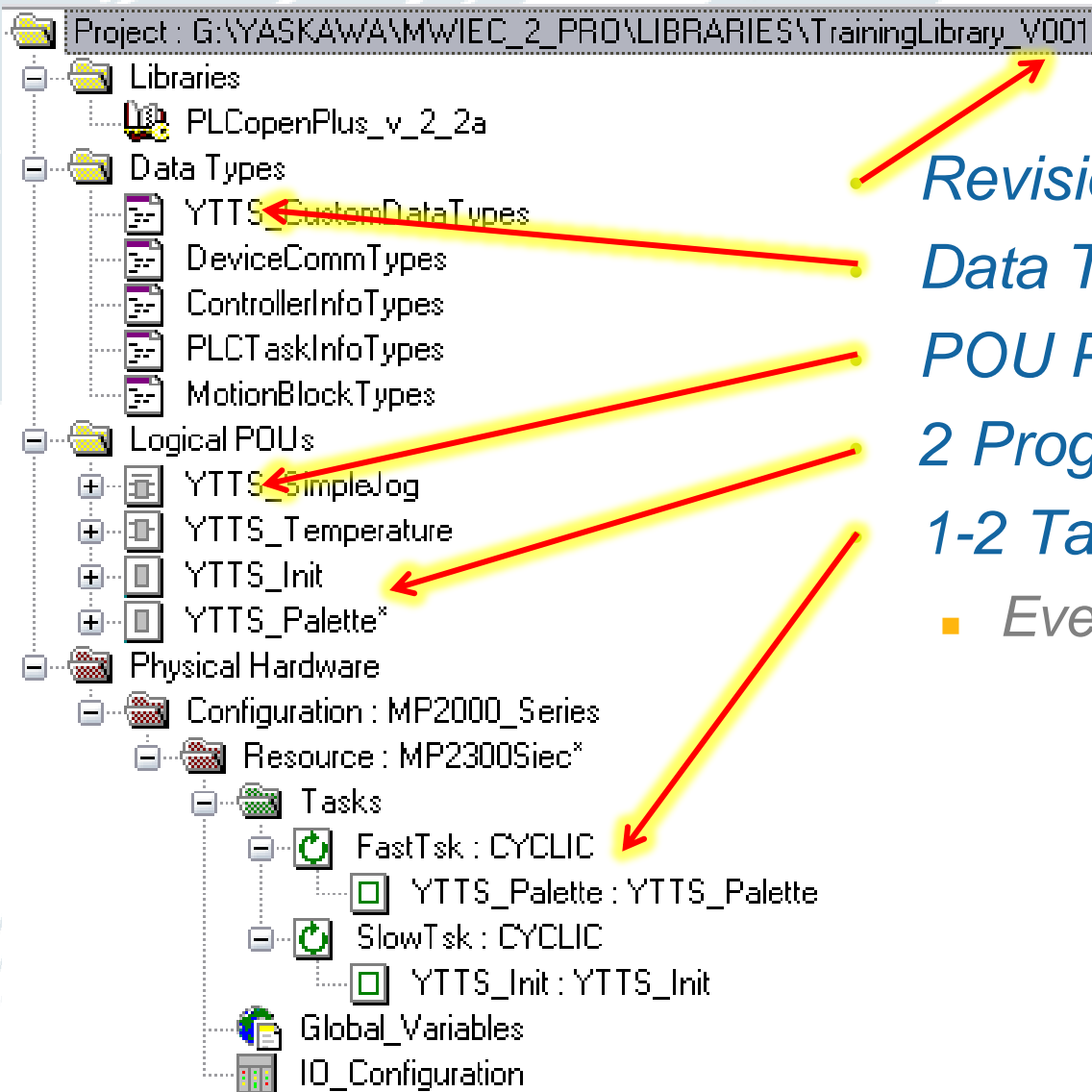
Overview

User Libraries are just normal project files imported into another project file as a library.
"Application Code Toolboxes" from Yaskawa are simply User Libraries created at Yaskawa

Create a Library

Step Description	Detail
1 Rename the project and save in the Libraries folder (Optional, Best Practice)	File -> Save As.File -> Save Project As / Zip Project As... Navigate to Libraries folder Edit the file name to reflect the intended usage as a user library and revision control. For example, "TrainingLibrary_V001"
2 Delete unnecessary POU's, Tasks, Datatypes (Optional, Best Practice)	In project tree leave one LD program POU with an instance of each user FU/FB. Leave one ST program POU with data initialization. Leave one task to run both programs (for compile).
3 Prefix all POU's, Tasks, Datatypes (Optional, Best Practice)	Example Prefix: "YTTS_". Typical to leave LD programs YTTS_Palette and YTTS_Init running in YTTS_Tsk, using YTTS_CustomDatatypes.
4 MAKE the library project	Click "MAKE" and resolve all errors. A project that has only been renamed requires a new "MAKE"
5 Save the library project as ZIP (Optional, Best Practice)	File -> Save Project As / Zip Project As... Navigate to the Libraries folder (for organization purposes) Choose ZIP as the file type Best Practice: Do NOT change the name when zipping. Change the name in Step 1 and re-MAKE first. Under "Zip Options" check boxes for "User Libraries" and "Front-end Code". Click the Zip button. <i>Single portable library file with revision name is produced</i>

The user library contains all POU's from the library project. Even program POU's can be reused. More common and useful is to only reuse FU and FB.



Revision In Project Name

Data Type Prefix

POU Prefix

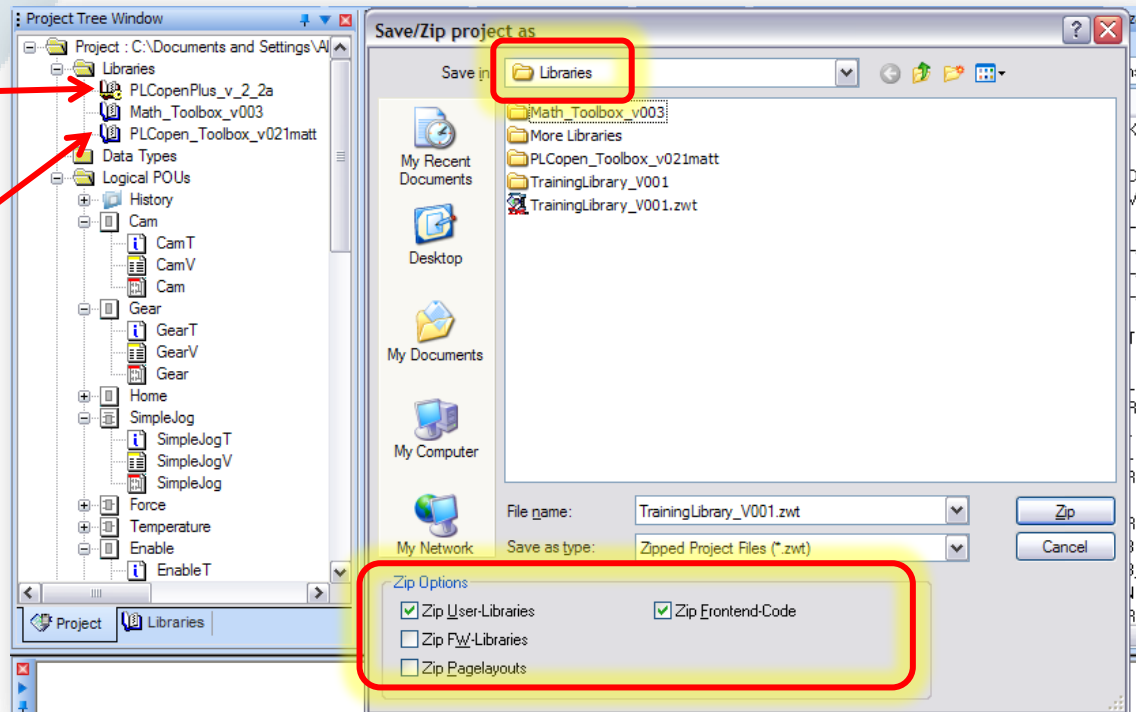
2 Program POU's

1-2 Tasks

- *Every task requires a program*

Zip to Libraries Folder

- Keep Organized
- Single File is portable
- **FW-Libraries (red)**
 - Ex: PLCopenPlus
 - Part of MWiec
- **User Libraries (Blue)**
 - Zip these libraries in this project
- **Frontend-Code**
 - Zip the result of "Make"
- **Pagelayouts**
 - Not used by Yaskawa




Default check "Zip User-Libraries" and "Zip Frontend-Code"

- *New Project*
 - *Insert user library*
 - *Add user function and function block to a POU*

Refer to the Quick Reference Guide



Use a Library

Step/Description	Detail
1 Acquire a ZWE (Express) or ZWT (Pro) file	Use your own, or download from Yaskawa.com Product Page Follow links to save the file In Windows Explorer, copy the file to C:\Documents and Settings\All Users\Documents\MotionWorks IEC xxx\Libraries (For organization purposes)
2 Unzip the library project to the library directory	In MotionWorks IEC... File-> Open Project / Unzip Project Click "Yes" to unzip to the Library directory (File was copied here in previous step) or click "No" if opening directly from CD or Download folder "Skip All" to Extracting Firmware Libraries dialog "Yes to All" to Overwrite Page Layout
3 Check for Dependent Libraries	Project Tree -> Project Tab, Expand Libraries folder  Yaskawa Toolbox Take note of any User Libraries, indicated by the "blue book" icon.
4 Start new / open existing project	File -> New, or File -> Open
5 Insert the Library and any dependent libraries	In Project Tree, "Project" tab, R-Click "Libraries" -> Insert -> User Libraries Navigate to find the Library (if you unzipped it to the "libraries" folder, you will see it right away) Also insert any dependent libraries noted in Step 3
6 Delete duplicate project data types	In Project Tree, "Project" tab, expand "Data Types" folder for both the user library and the project library. Delete any duplicates of "PLCTaskInfoTypes" or "MotionBlockTypes" from the project library.

The user library contains all POUs from the library project. Even program POUs can be reused. More common and useful is to only reuse FU and FB.

ENGINEERING EXPERTISE

EASY TO
WORK WITH  YASKAWA™

QUALITY
PRODUCT



TECHNOLOGICAL
INNOVATION

