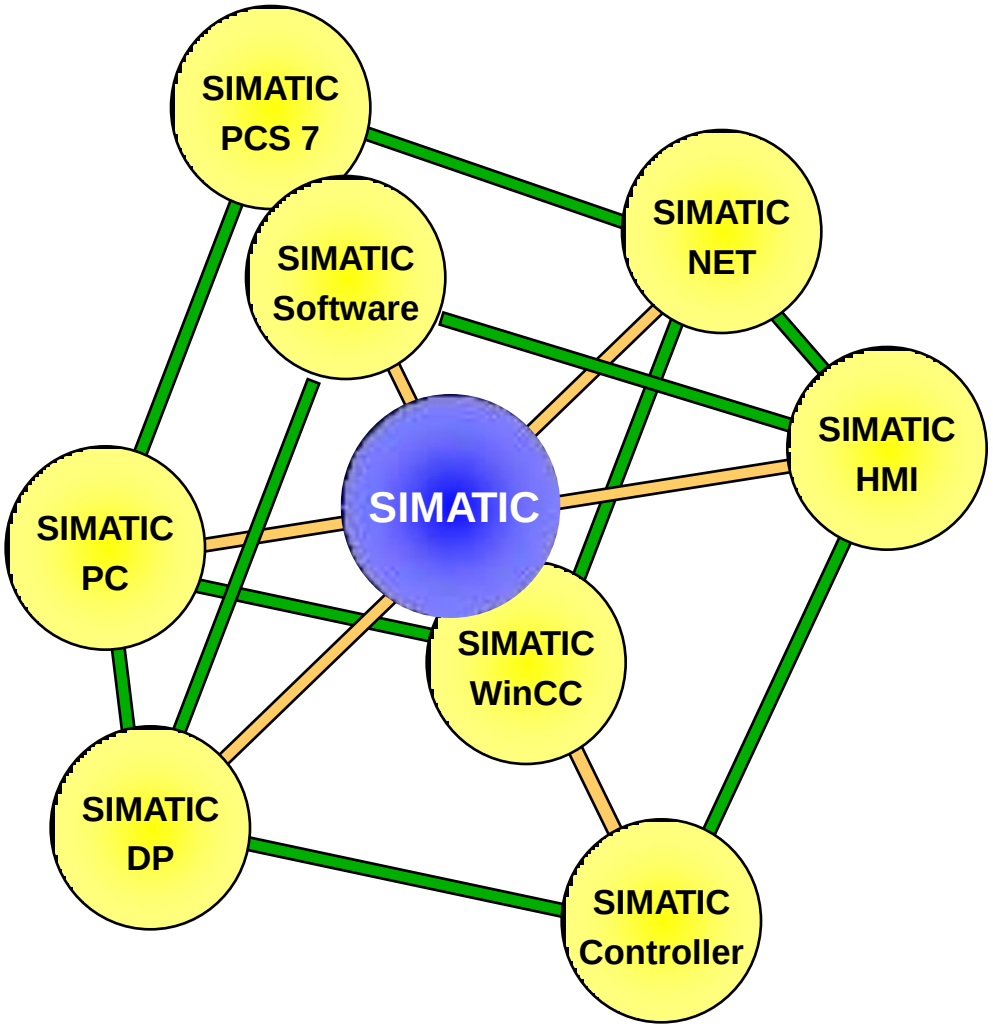
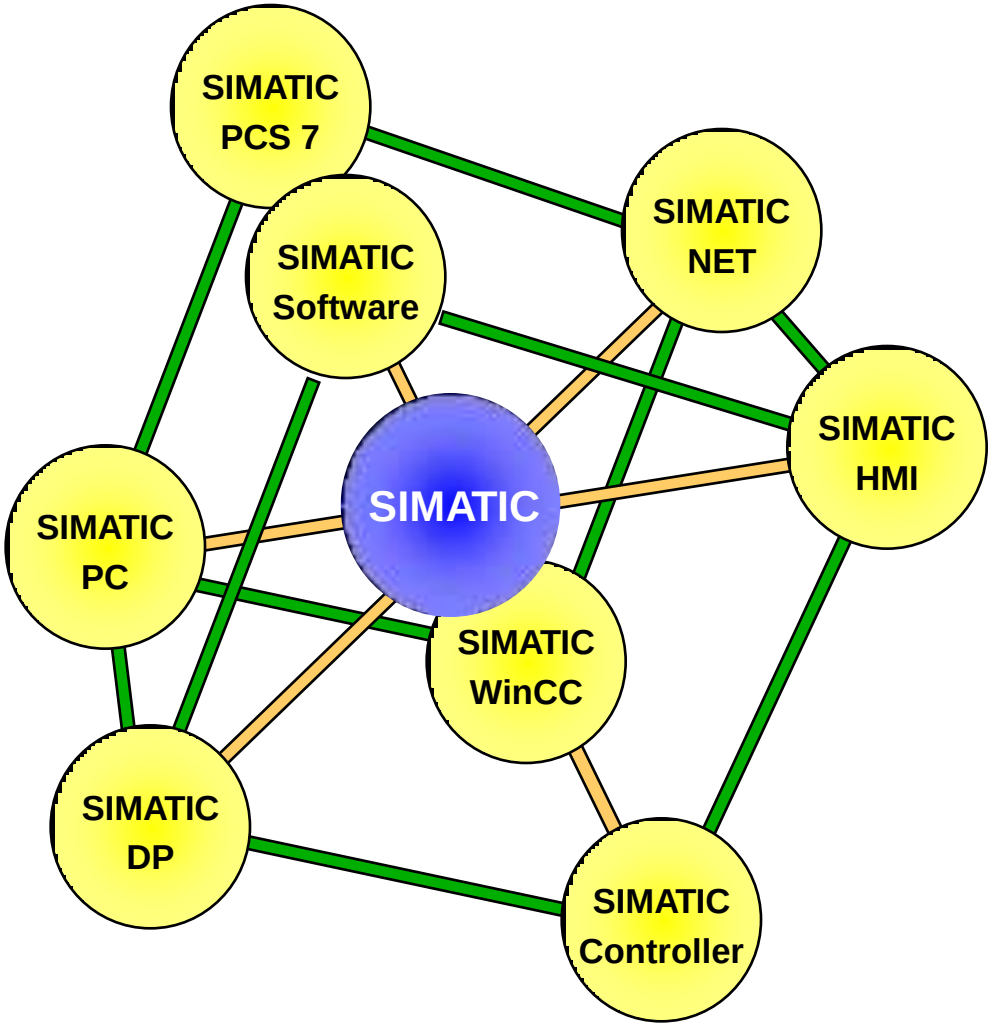
The SIMATIC S7 System Family



The SIMATIC S7 System Family



Objectives

Upon completion of the chapter the participant will ...

... have an overview of the SIMATIC S7 system family

... be familiar with the S7-300 and S7-400 automation systems

... have an overview of the modules available for these automation systems

... understand the concept of "Totally Integrated Automation" (T.I.A.)

... be familiar with the programming devices

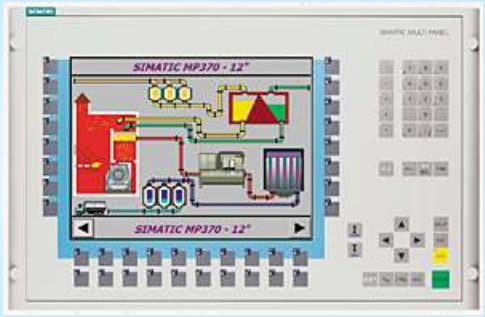
... be familiar with the PC requirements for working with STEP7

... be familiar with the tools of the STEP7 basic programming package

… have an overview of the most important tools used in the T.I.A. concept

SIMATIC S7 Overview

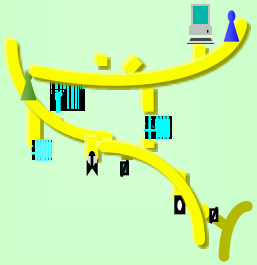
SIMATIC HMI



SIMATIC PG
SIMATIC PC

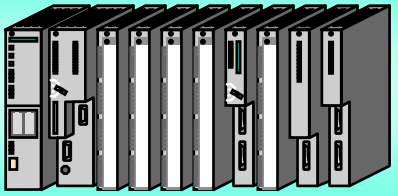
SIMATIC Software

SIMATIC NET



MPI Network
Industrial Ethernet
PROFIBUS
PROFINET

SIMATIC
Controller



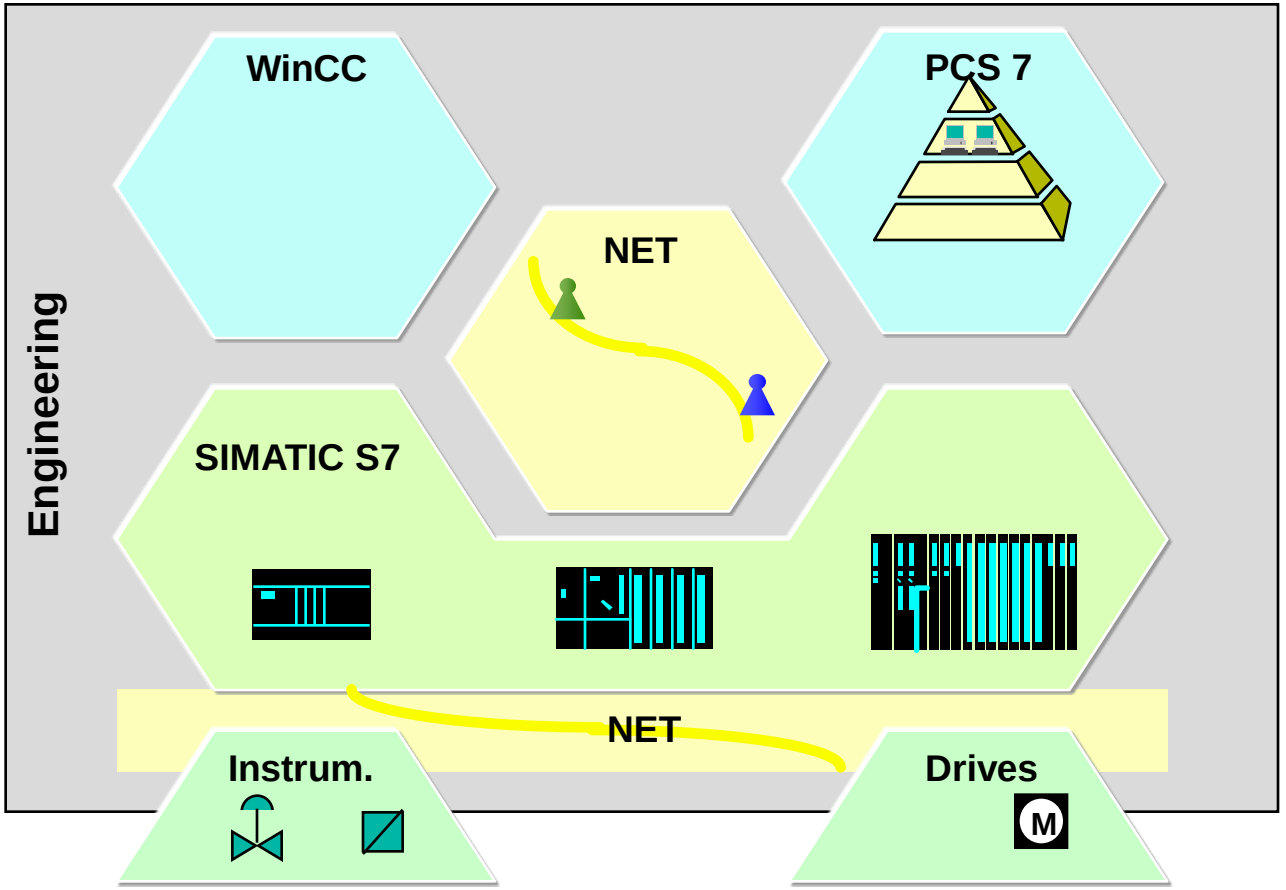
PC-based

PROFIBUS DP

AS-Interface

Date: 1/16/2012
File: PRO1_01E.4

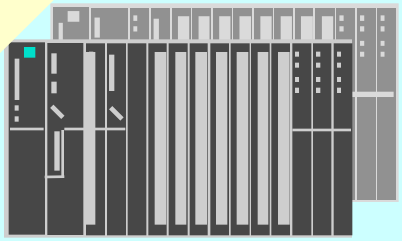
What does “Totally Integrated Automation“ Mean



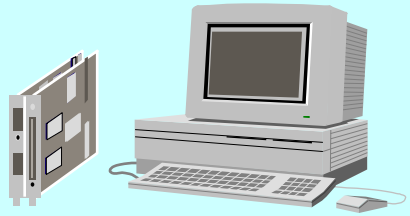
The SIMATIC S7/C7 and WinAC Controllers

Upper and middle performance range

modular



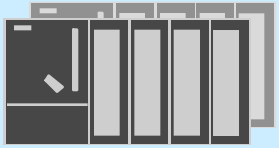
SIMATIC S7 - 400



SIMATIC WinAC Slot

Lower and middle performance range

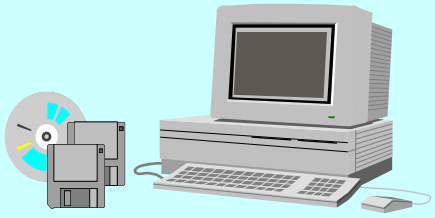
modular



SIMATIC S7 - 300

complete

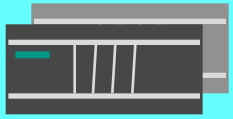
SIMATIC C7 – 6xx



SIMATIC WinAC Basis

Micro PLC

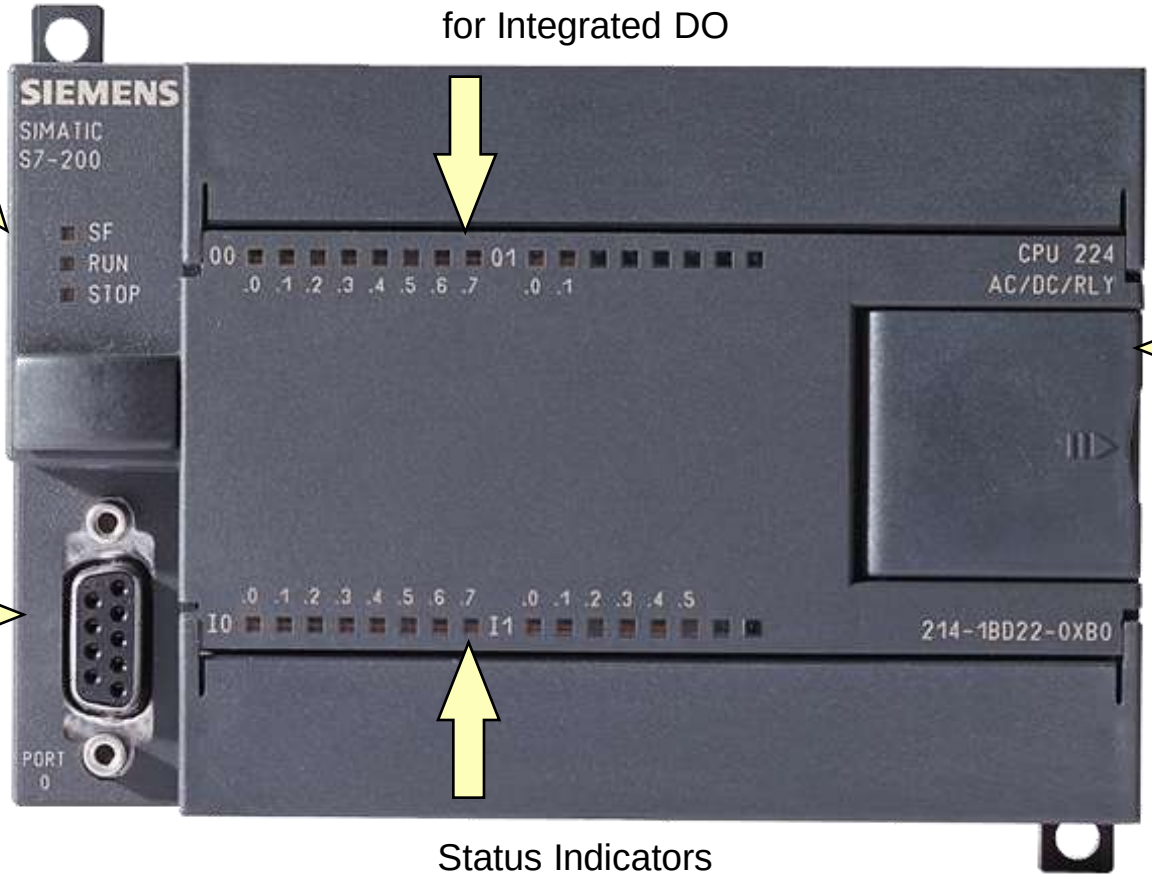
compact



SIMATIC S7 - 200

Logo

S7-200 Micro Controller



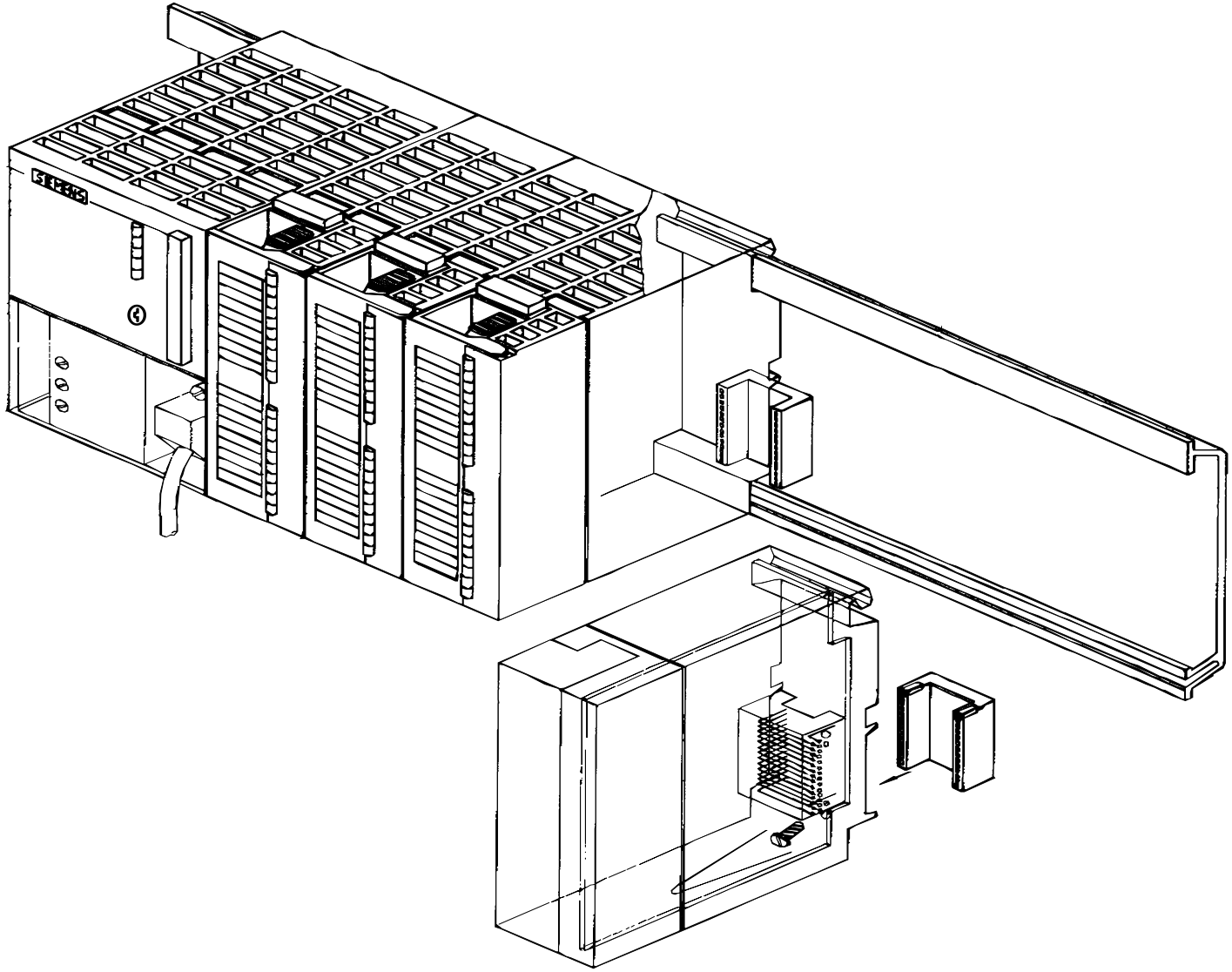
S7-200: Modules / Expansion Modules (EM)

I / O Modules

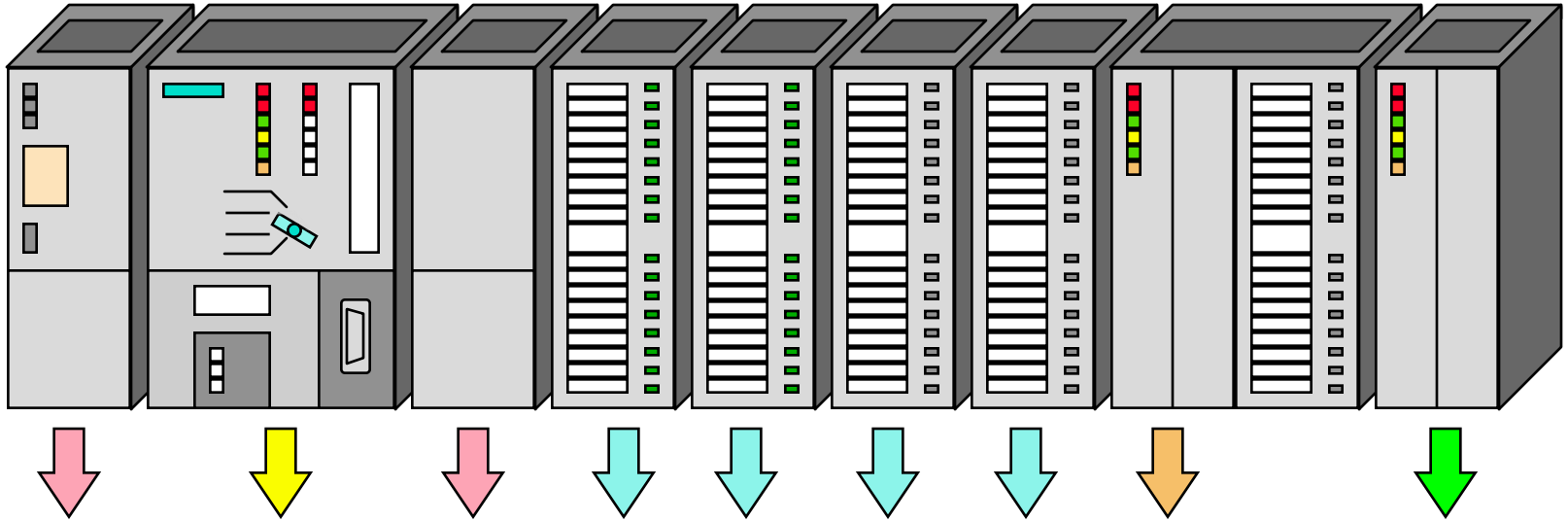
Function Modules (FM)

Communications Processors (CP)

S7 - 300 Automation System

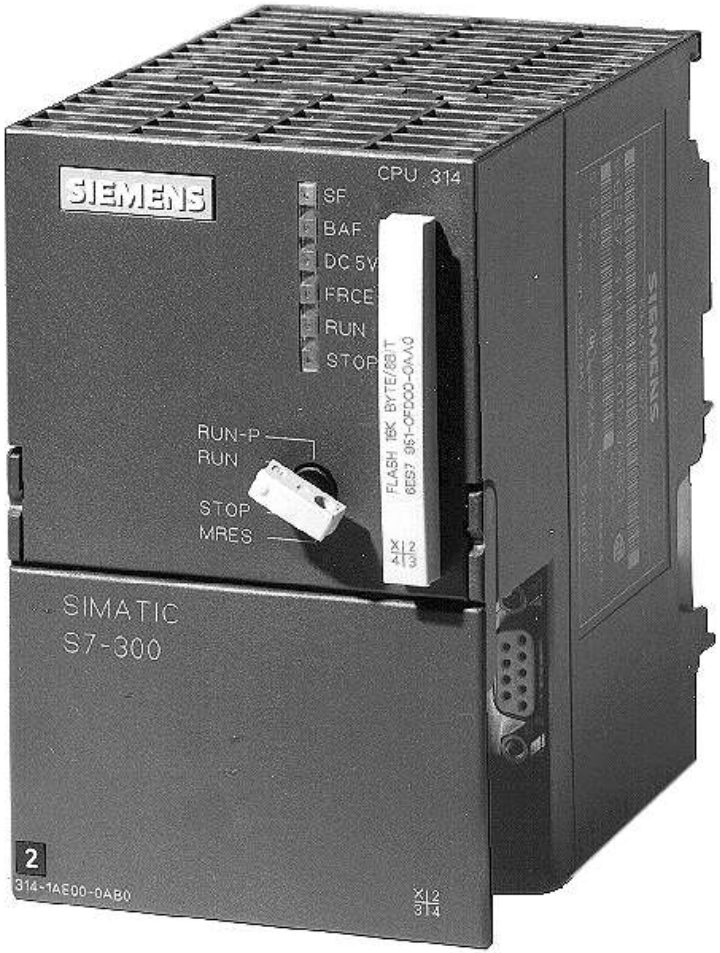


S7-300: Modules



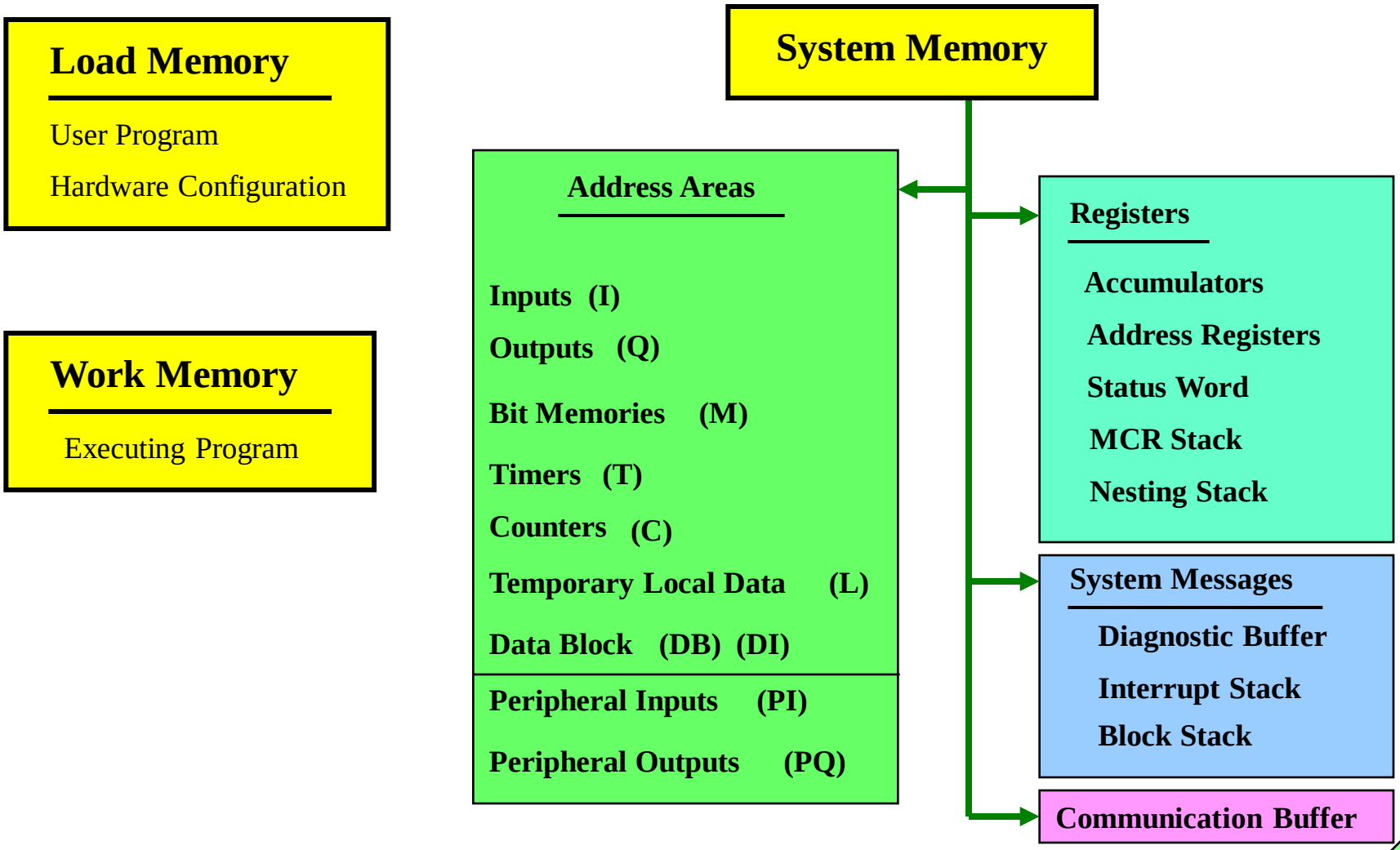
S7-300: CPU Design

CPU 314 until Oct. 2002

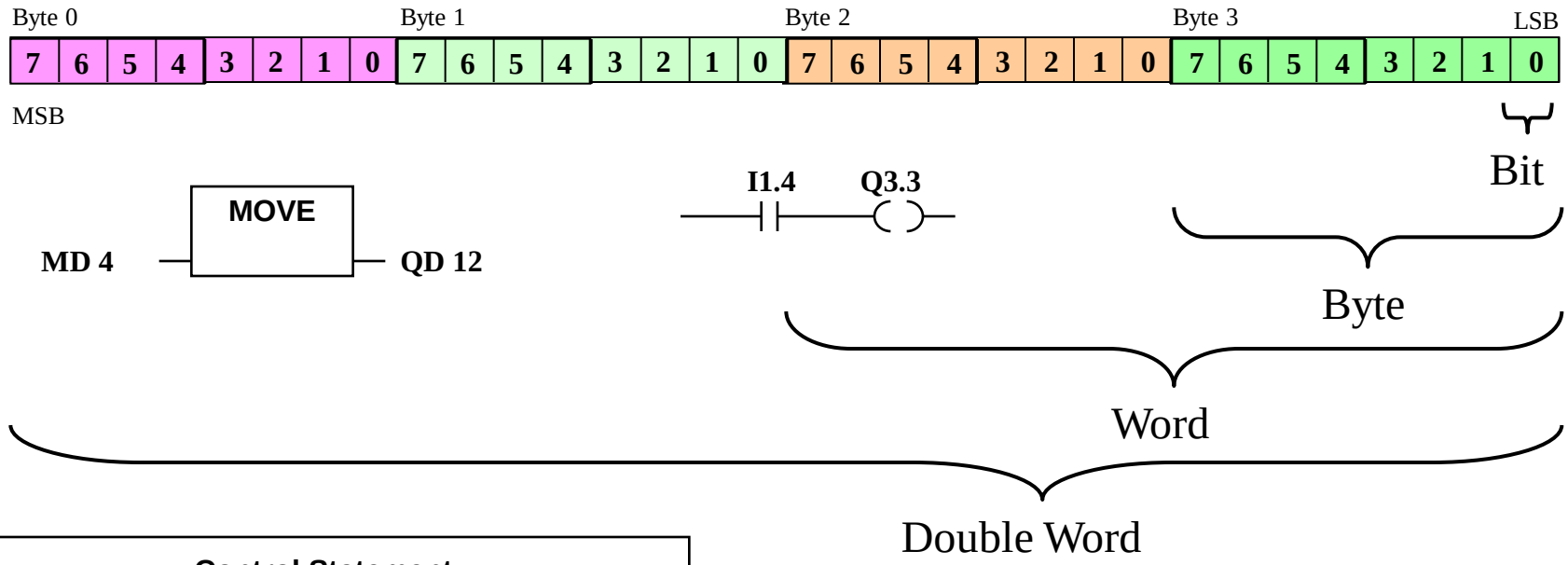


CPU 314 after Oct. 2002

CPU Memory Areas



Addressing



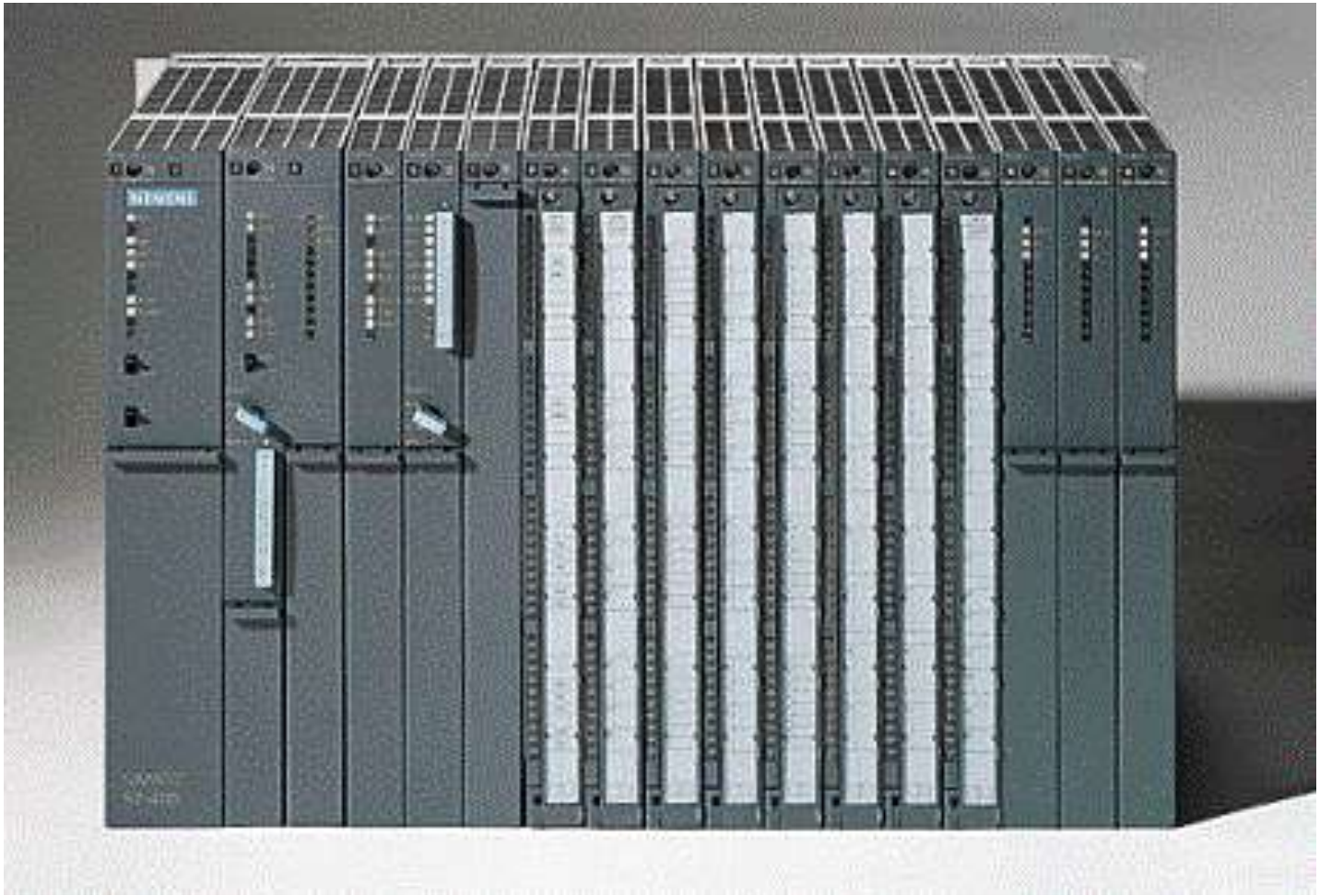
Control Statement		
A	*I*	*3.4*

Operational Part	Address Part	
A	*I*	*3.4*

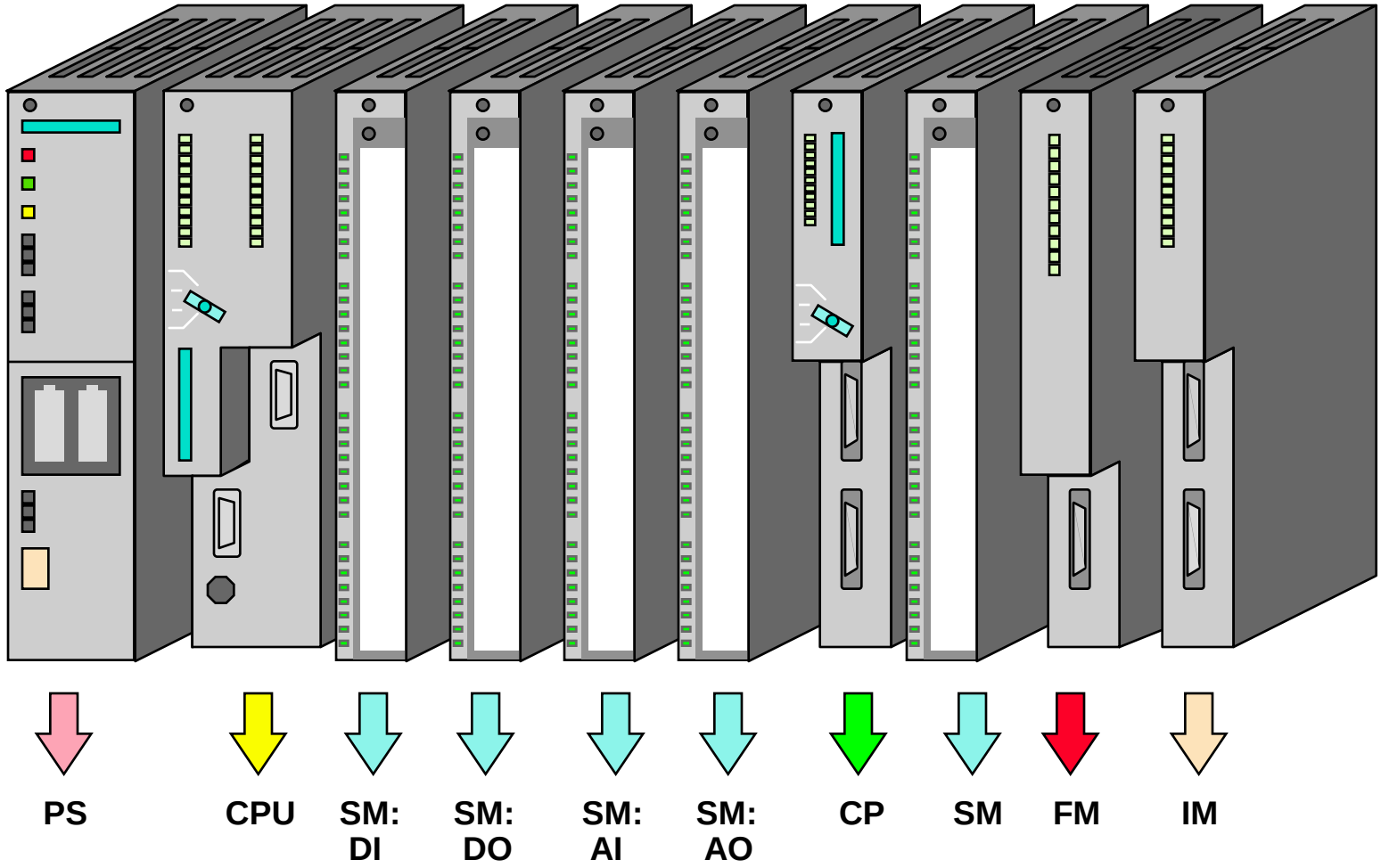
	Address Identifier	Parameter
	I	*3.4*

Parameter = Byte Address and Bit Address

S7-400 Automation System

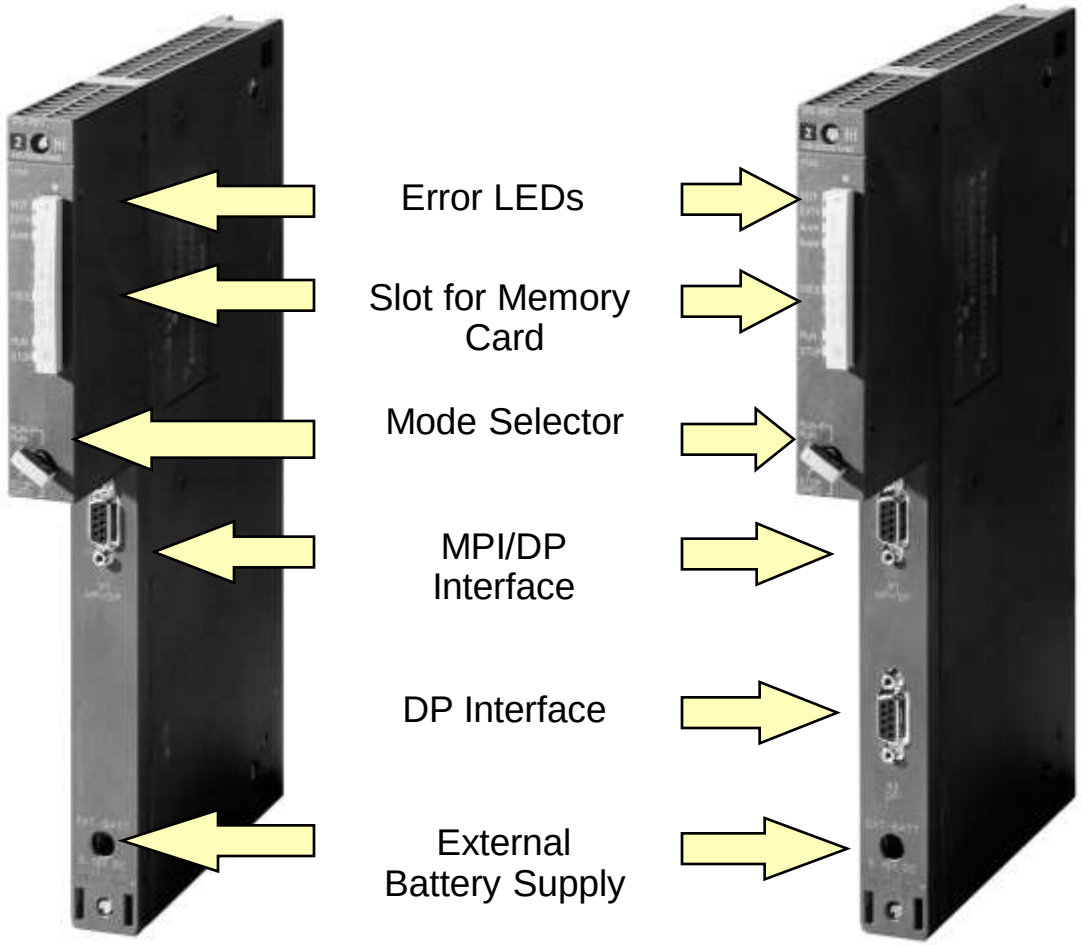


S7-400: Modules



Date: 1/16/2012
File: PRO1_01E.15

S7-400: CPU Design



PG/PC Requirements for Installing STEP 7

Operating Systems: Microsoft Windows 2000 Professional

Microsoft Windows XP Professional

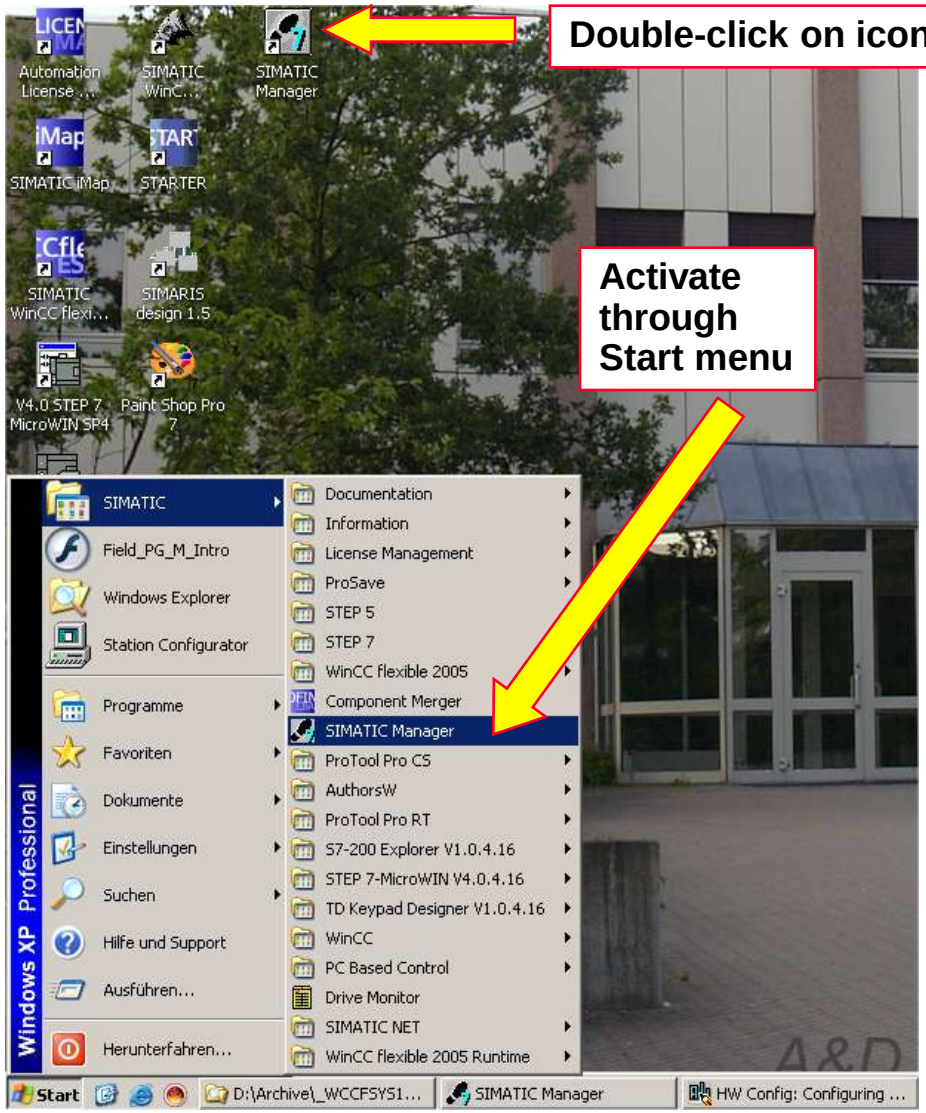
Memory on the Hard Drive: Depending on the installation, between 400 MB and 800 MB

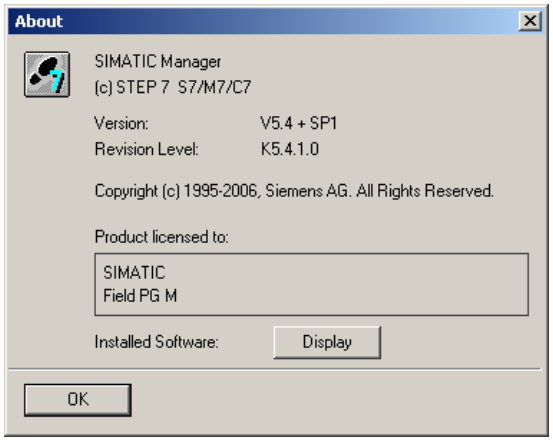
Mouse: Yes

Interfaces:

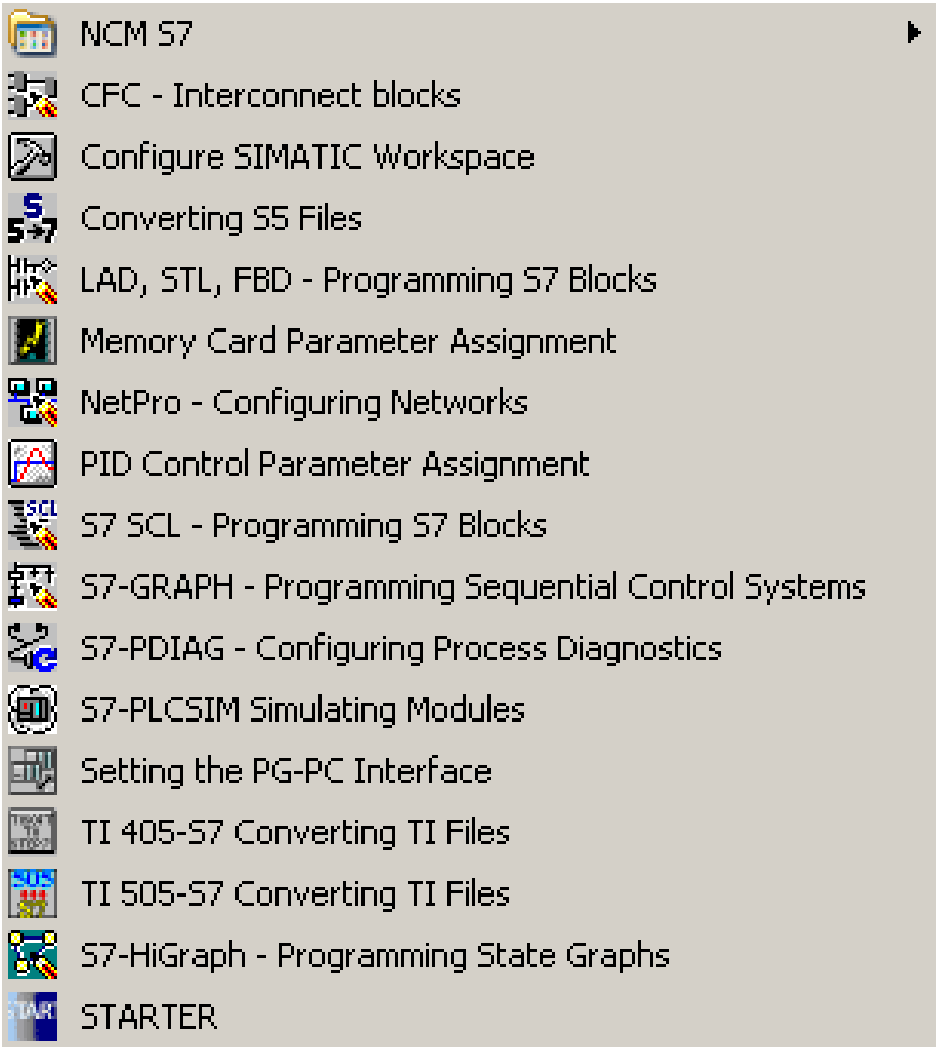
- CP5611 (PCI) or
- CP5512 (PCMCIA) or
- PC adapter (USB) or
- Industrial Ethernet Interface (NIC)
- Programming interface for Memory Card (optional)

Installing the STEP 7 Basic Package

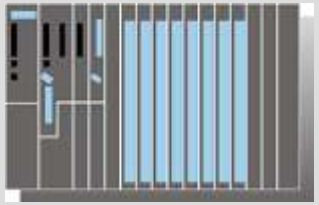




STEP 7 Tools

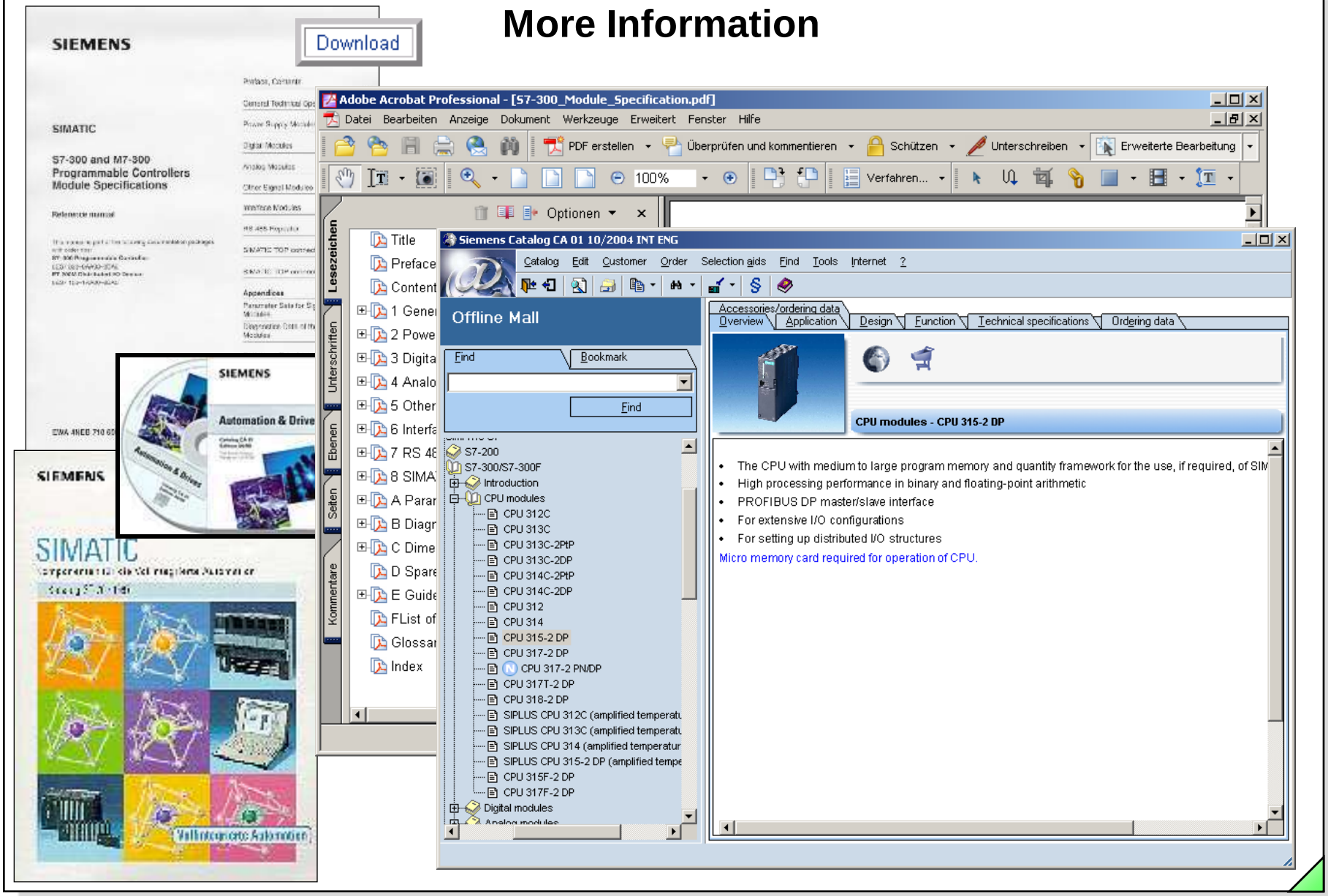


STEP 7 Software Packages (Examples)

Which PLC?	Necessary	Options
SIMATIC S7-200 	STEP 7 Micro/WIN	
SIMATIC S7-300 	STEP 7 Lite 	Tele Service; S7- PLCSIM
SIMATIC S7-300  SIMATIC S7-400 	STEP 7 	S7- SCL; CFC; S7- GRAPH; S7- HiGraph; Fuzzy Control; Standard PID Control; Modular PID Control; DOCPRO; TeleService; S7- PDIAG; S7- PLCSIM; PRODAVE MPI
STEP 7 Professional	STEP 7	S7- SCL; S7- PLCSIM; S7- GRAPH

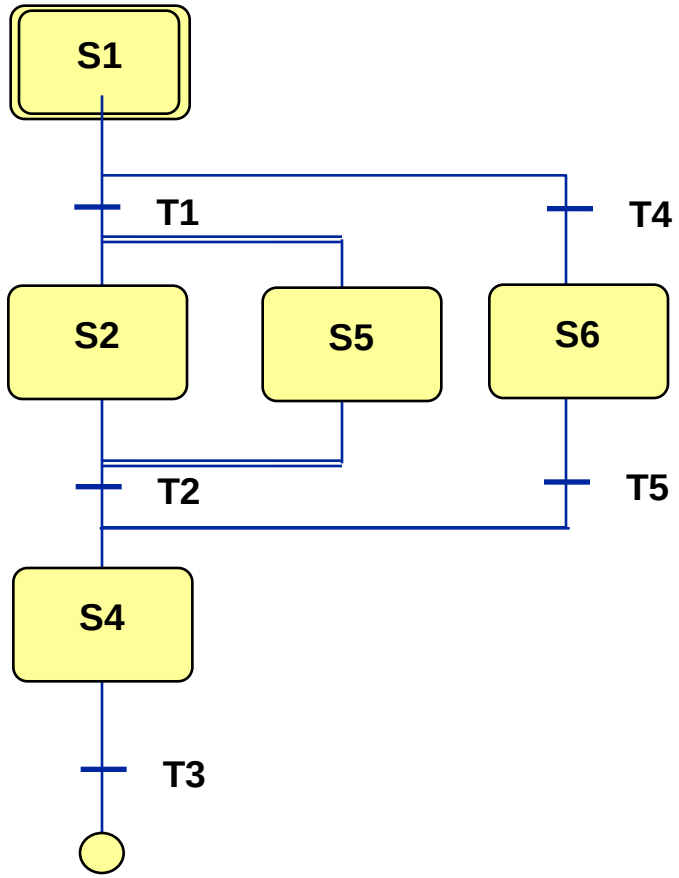
If You Want to Know More about the Topic “TIA“

More Information



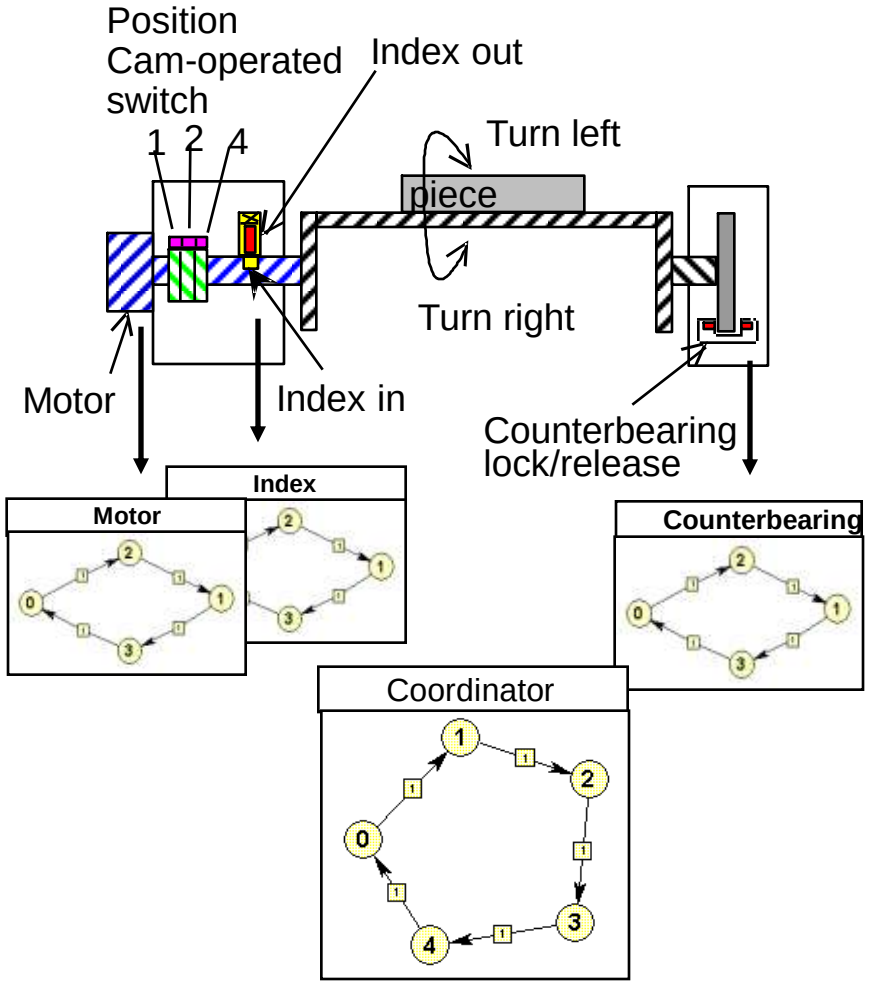
Programming Sequence Control Systems with S7- GRAPH

- **S7-GRAPH: The tool for programming sequence cascades**
  - Compatible with DIN EN 6.1131-3
  - Designed for the requirements of production engineering
  - Graphic division of the process into steps and transitions
  - Steps contain actions
  - Transitions check the conditions for switching to the next step
- **The following phases of automation can be optimized with S7-GRAPH:**
  - Planning, Configuring
  - Programming
  - Debugging
  - Commissioning
  - Maintenance, Diagnostics



The State Diagram Method S7- HiGraph

- **S7-HiGraph: The tool for programming using State Diagrams**
  - Division of the machine into functional units
  - Creating state diagrams for every function unit
  - States contain actions
  - State diagrams communicate using messages
- **The following phases of automation can be optimized with S7-HiGraph:**
  - Planning, Configuring
  - Programming and Debugging
  - Commissioning
  - Maintenance, Diagnostics
  - Supports reusability



Programming in the High Level Language S7- SCL

- **S7-SCL: High level language for creating PLC programs**
  - **Compatible with DIN EN 6.1131-3 (ST=Structured Text))**
  - **Certified according to "PLC open" Base Level**
  - **Contains all the typical elements of a high level language, such as operands, terms, control statements**
  - **PLC specifics are integrated, such as I/O access, timers, counters...)**

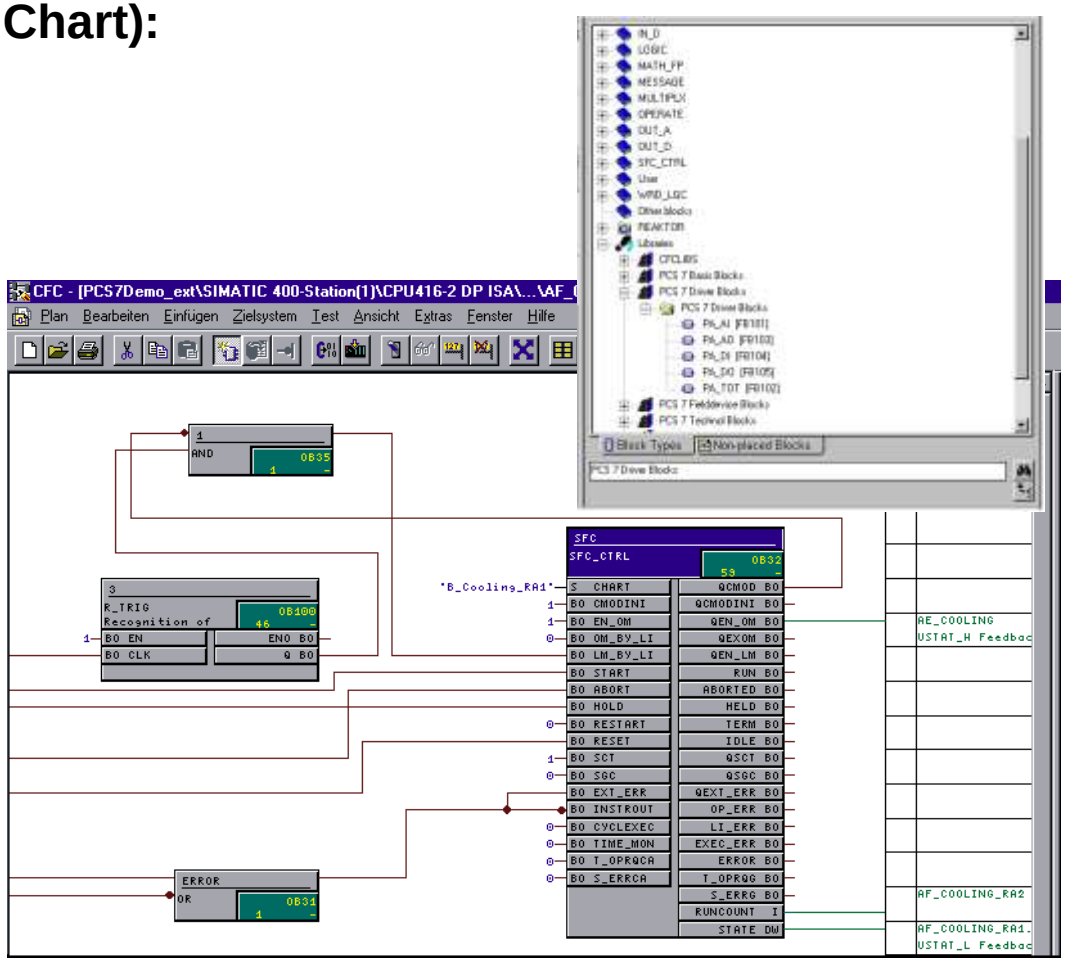
Advantages:

- **Well structured, easy to understand program**
- **For those knowlegeable in high level langugages**
- **For complex algorithms**

```
FUNCTION_BLOCK Integrator
VAR_INPUT
    Init       : BOOL;     // Reset output value
    x          : REAL;     // Input value
    Ta         : TIME;     // Sampling interval in ms
    Ti         : TIME;     // Integration time in ms
   olim        : REAL;     // Output value upper limit
   ulim        : REAL;     // Output value lower limit
 END_VAR

VAR_OUTPUT
   y   : REAL:= 0.0;      // Initialize output value with 0
END_VAR

BEGIN
    IF TIME_TO_DINT(Ti) = 0 THEN      // Division by ?
        OK := FALSE;
        y  := 0.0;
        RETURN;
    END_IF;
    IF Init THEN
        y:= 0.0;
    ELSE
        y := y+TIME_TO_DINT(Ta)*x/TIME_TO_DINT(Ti);
        IF y > olim THEN y := olim; END_IF;
        IF y < ulim THEN y := ulim; END_IF;
    END_IF;
END_FUNCTION_BLOCK
```

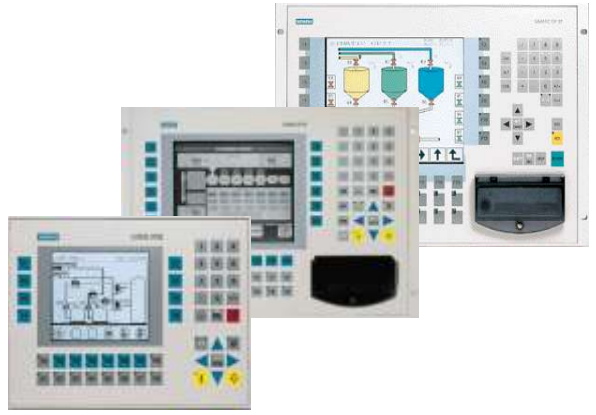
CFC - Tool for Graphic Interconnection of S7 Blocks

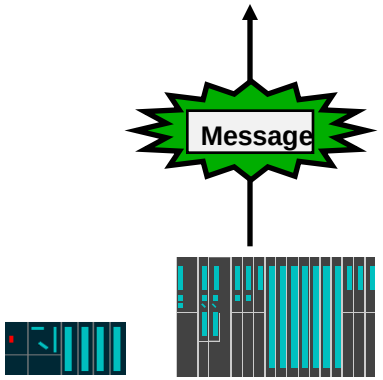
- **CFC (Continuous Function Chart): Tool for graphic creation of PLC programs**
  - **Blocks are placed on function charts and interconnected**
  - **Interconnection is possible:**
    - **between I/O fields**
    - **also to blocks in other charts**
  - **Sources and destinations are managed in the margins**
- **Advantages**
  - **Program creation for technologists**
  - **quick creation, testing and commissioning times**

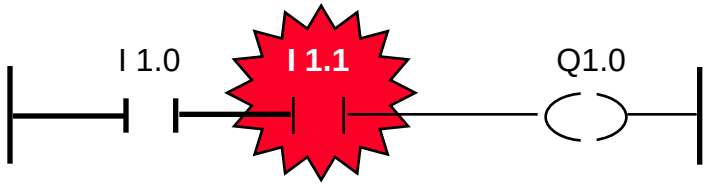


Process Diagnosis with S7- PDIAG

- **Process diagnosis: Detection of faults occurring outside the PLC**
  - **Sensor/actuator defective, movement faulty, ...**
- **S7- PDIAG: Tool for configuring the fault definition in STL, LAD, FBD**
  - **Integrated in the development environment**
  - **Simple formulation of fault monitoring and message texts (during and after the program session)**
  - **Fault detection and criteria analysis are conducted automatically**
  - **Comprehensive information for the operator on:**
    - type of fault
    - location of fault
    - cause of fault
- **Reduction of down-time**

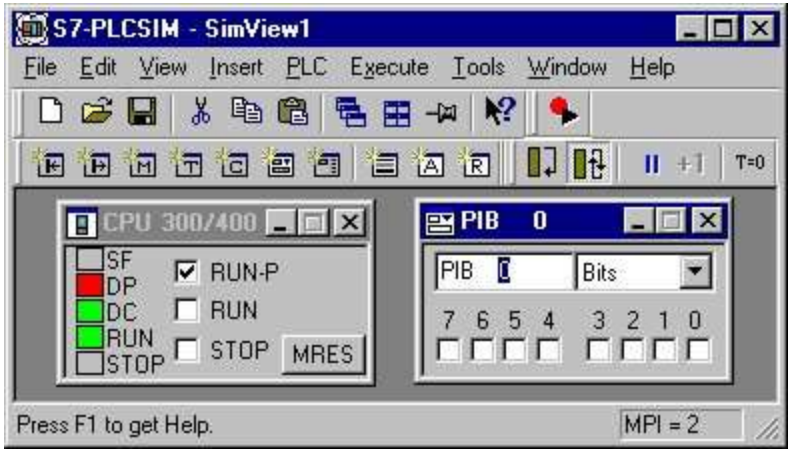




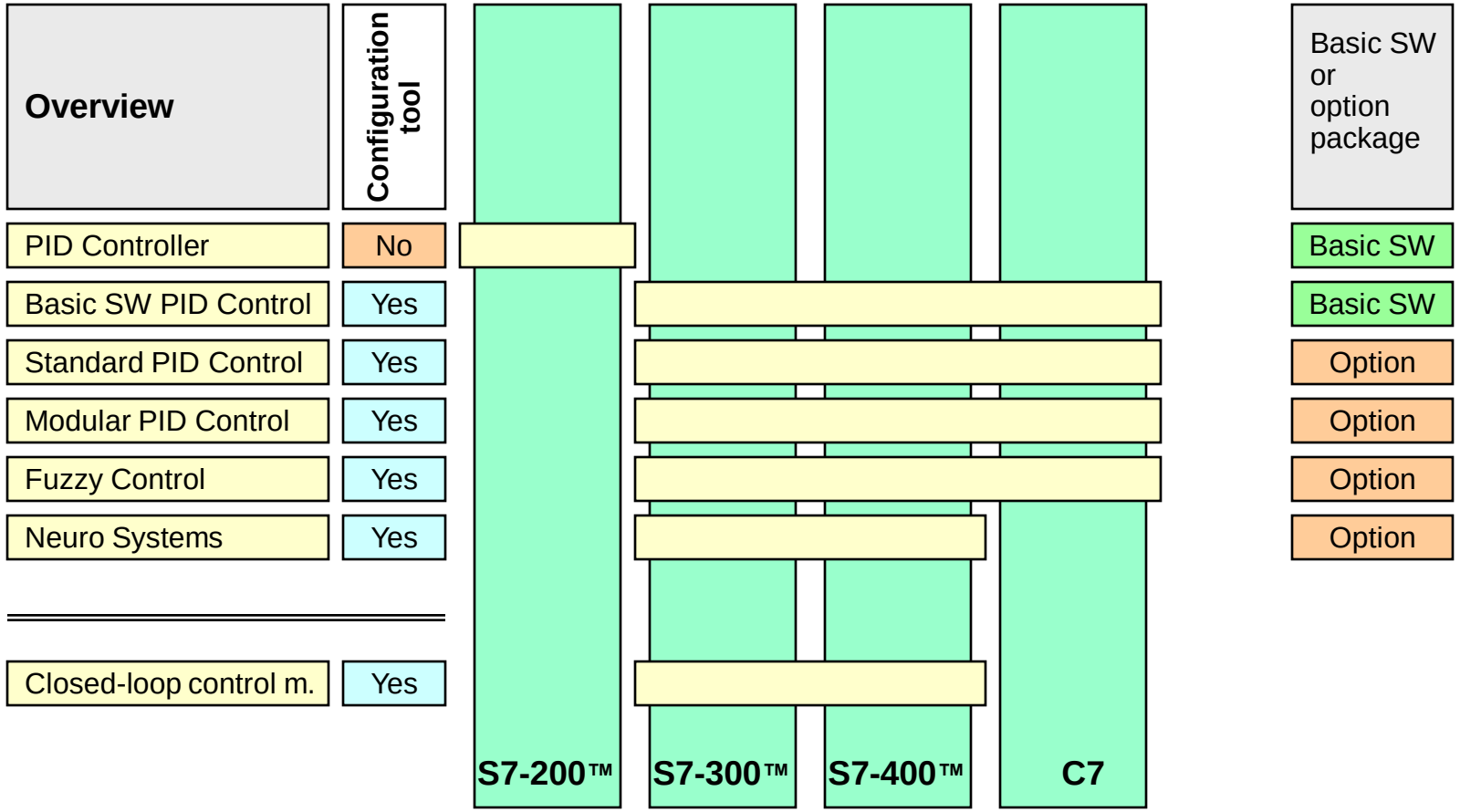


Testing User Programs with S7- PLCSIM

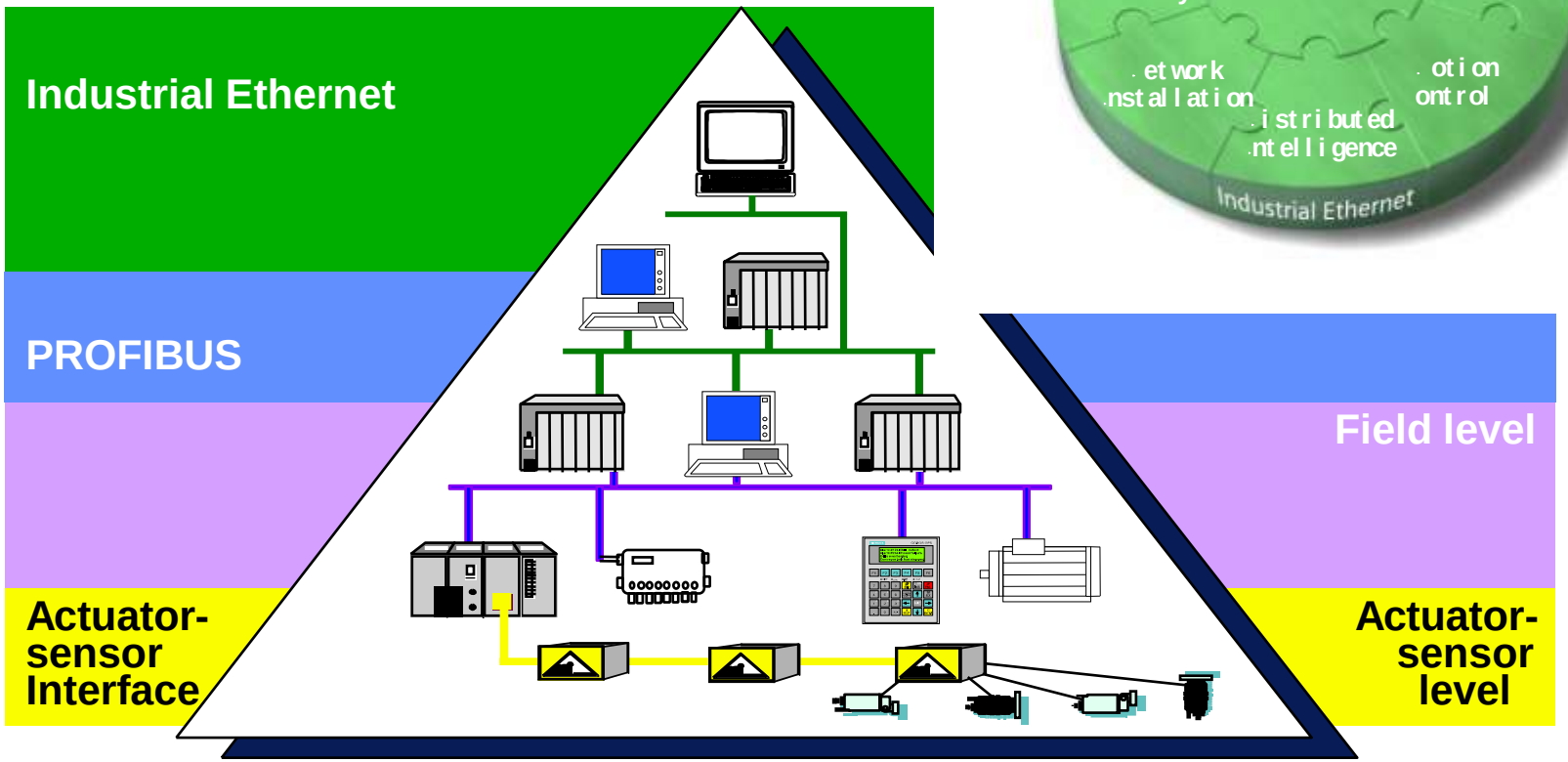
- **S7-PLCSIM: Simulation software for offline testing of PLC programs**
  - **Functional program test**
    - on a simulated CPU
    - with display/modify I/O
  - **Testing of user blocks in**
    - LAD, FBD, STL, S7-SCL,
    - S7-GRAPH, S7-HiGraph, CFC
    - S7-PDIAG, WinCC
- **Advantages**
  - **Faults can be detected early and eliminated**
  - **Many tests are already possible in the office without the final hardware**

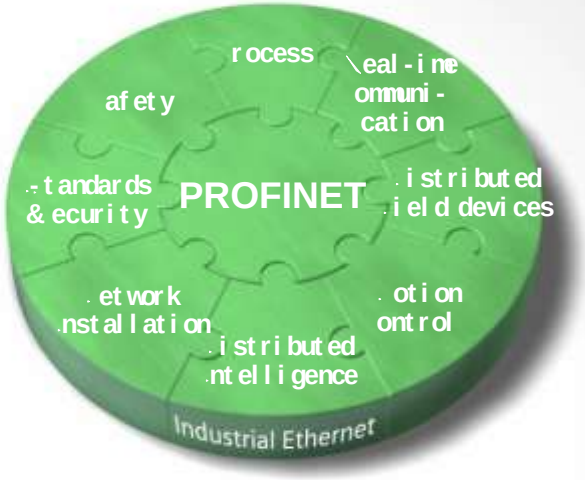


Runtime Software for Closed-loop Control Engineering Tasks



Communicating with SIMATIC NET





Operator Control and Process Monitoring with SIMATIC® HMI

Process visualization system SIMATIC WinCC

Process visualization system SIMATIC WinCC flexible

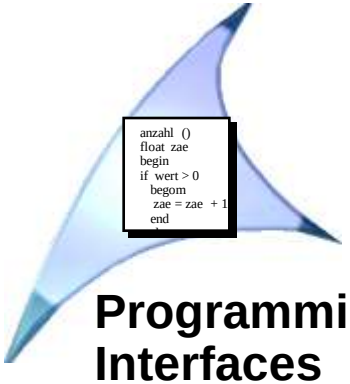
SIMATIC PG, PC

SIMATIC Panels, PGs, PCs

Process Visualization and Operator Control with WinCC

Alarm Logging
(Message System)

Process visualization

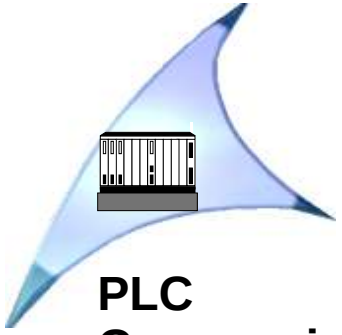


Programming
Interfaces

Tag Logging
(Archiving)

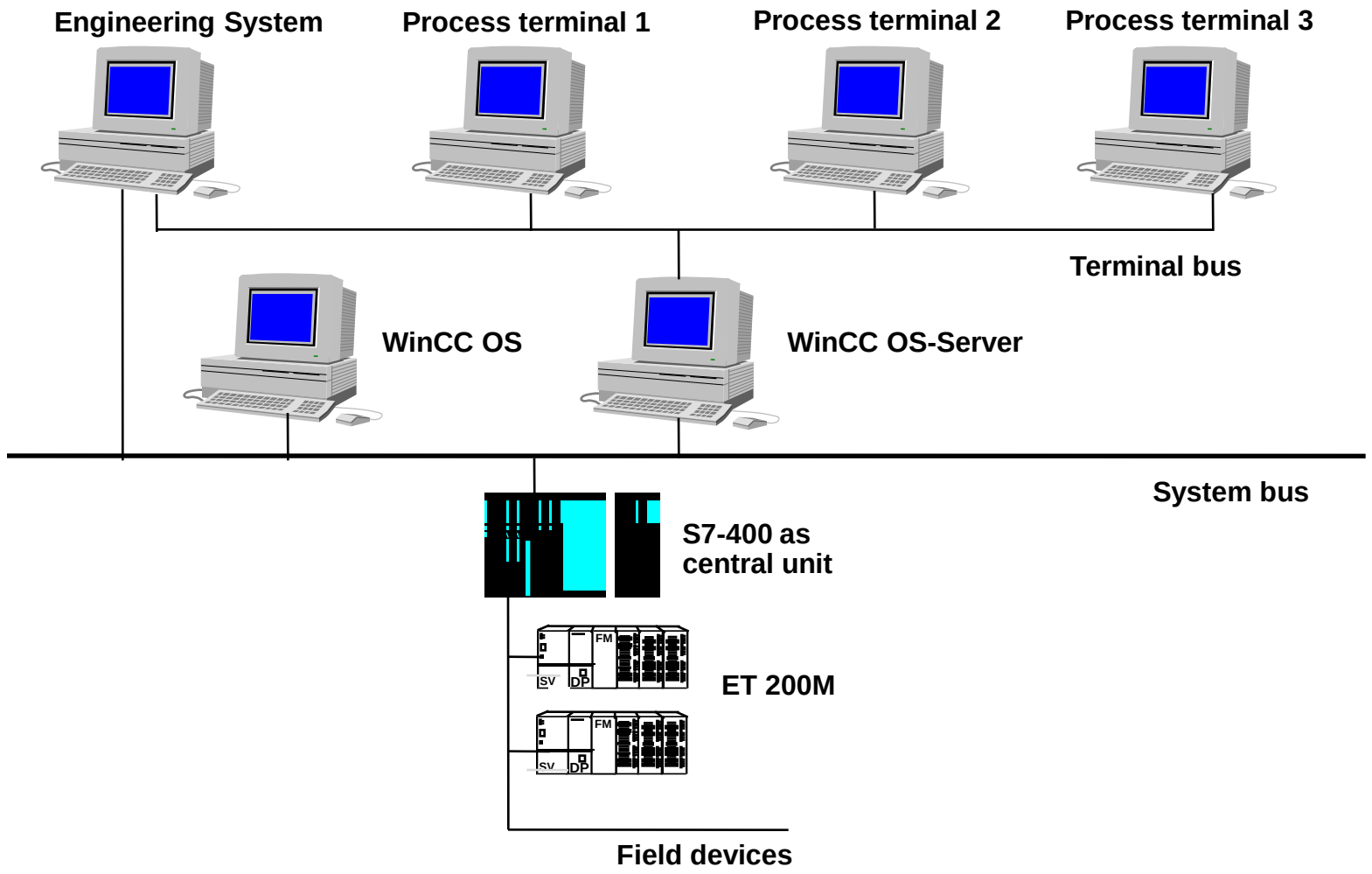
Standard
Interfaces

Report Designer
(Report System)



PLC
Communication

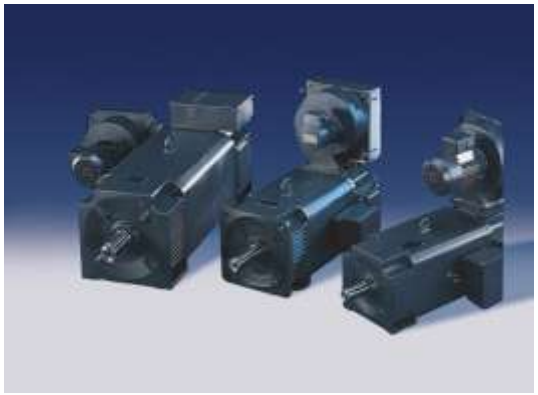
Process Automation with SIMATIC® PCS 7



DRIVES Technology

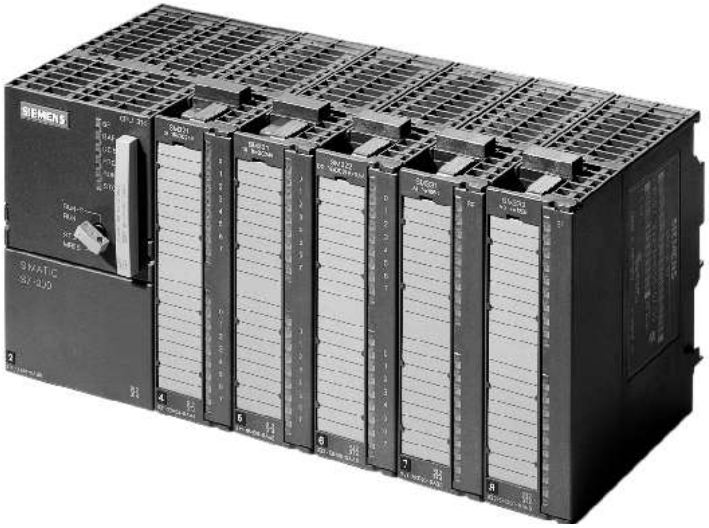
Consistent drives spectrum for all applications

- **From standard drives with 100 W up to large drives with 50 MW**
- **Motion and vector control**
- **Technology-specific closed-loop controllers**



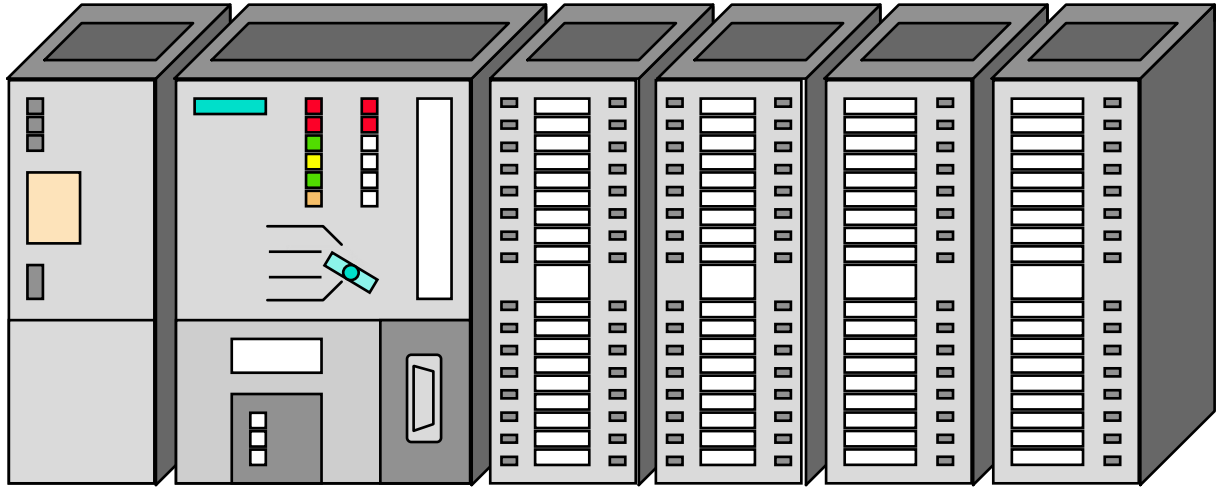
SINUMERIK® and SINAMICS Drives

Training Units and Addressing



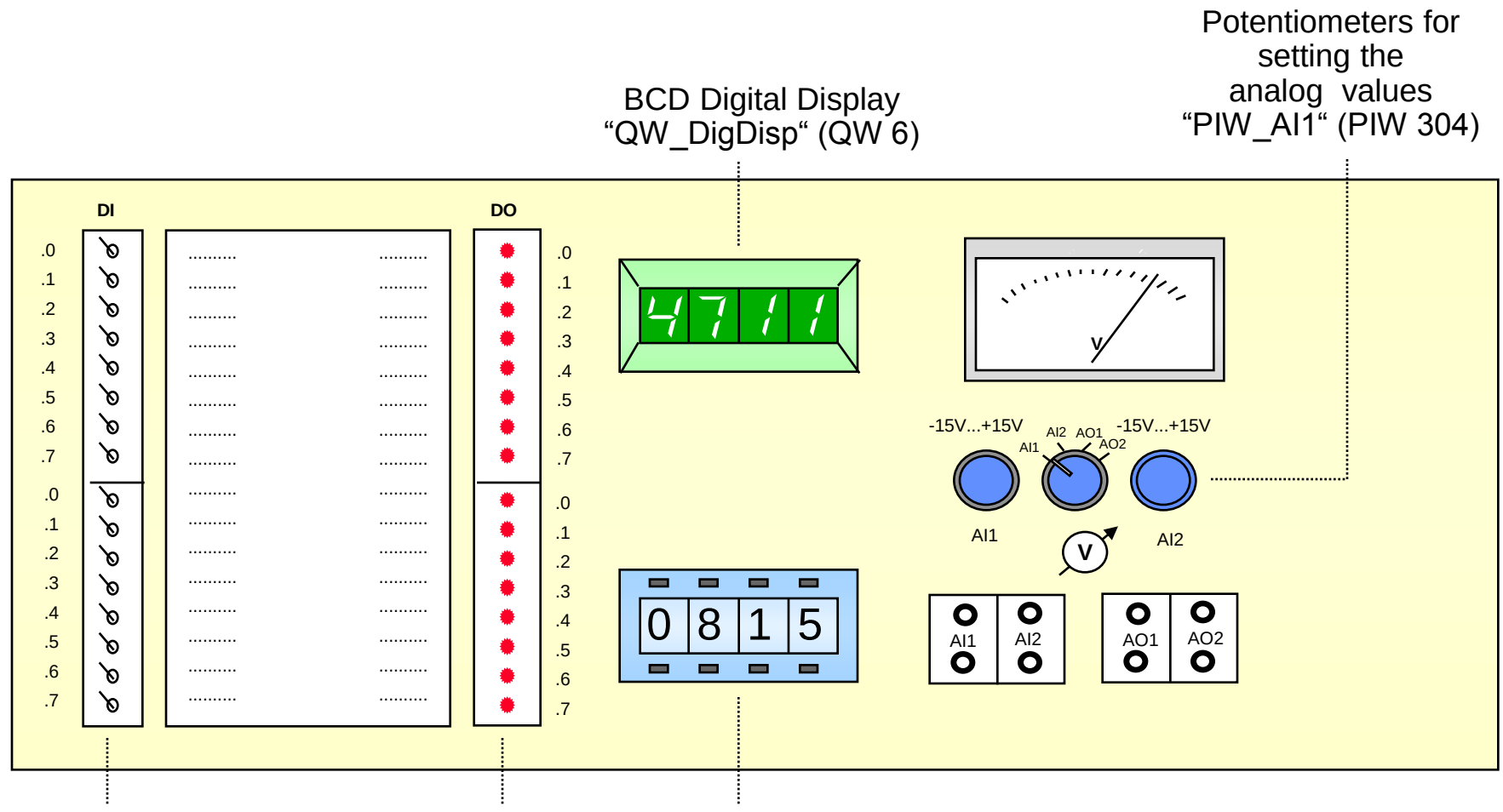
Training Area Setup with S7 - 300

Configuration of the S7 - 300 Training Unit

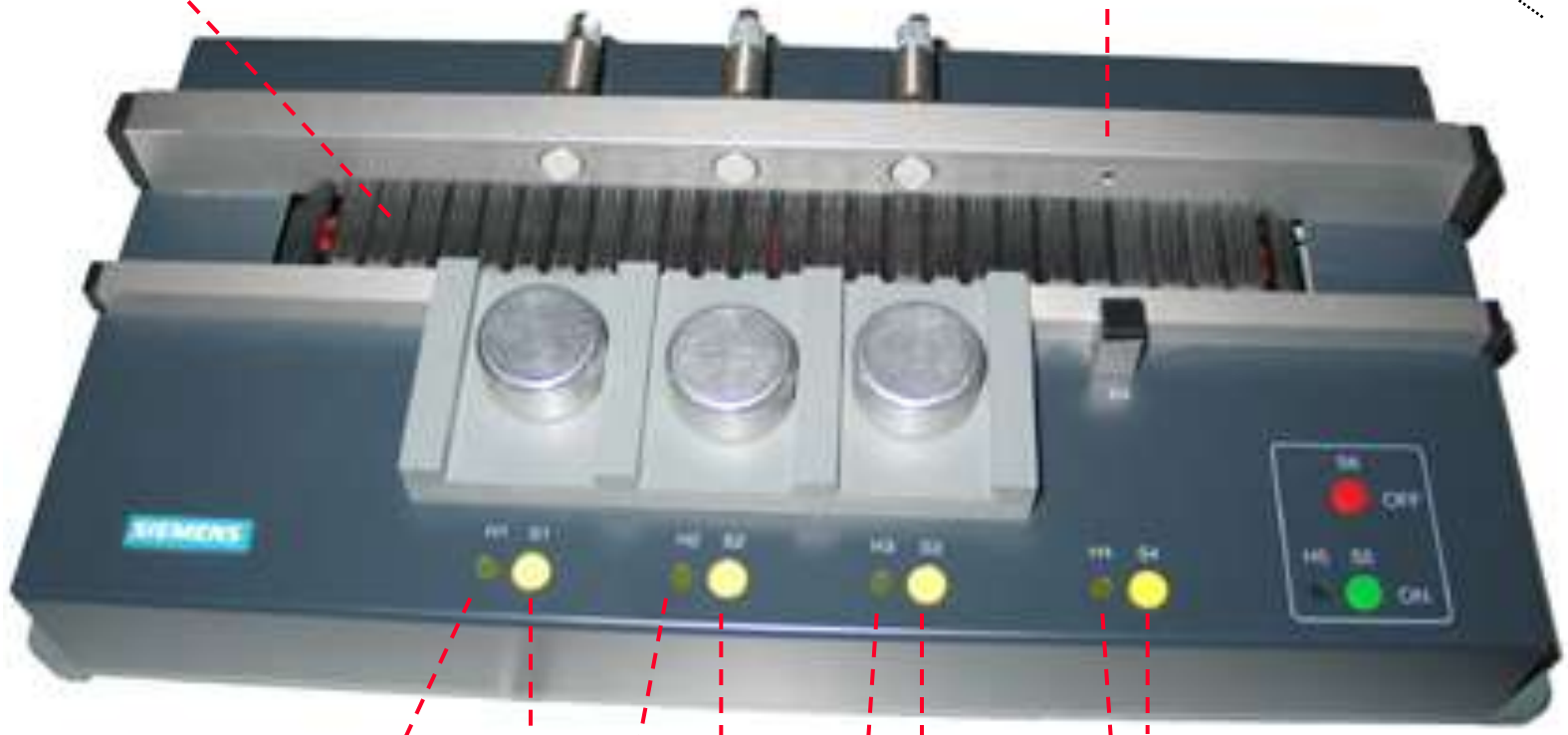


Module	-->	PS	CPU	DI 32	DO 32	DI8/DO8	AI 2
Slot No.	-->	1	2	4	5	6	7
I/O Address	-->			0	4	8	304

The Simulator



The Conveyor Model



Complete Symbol List (1)

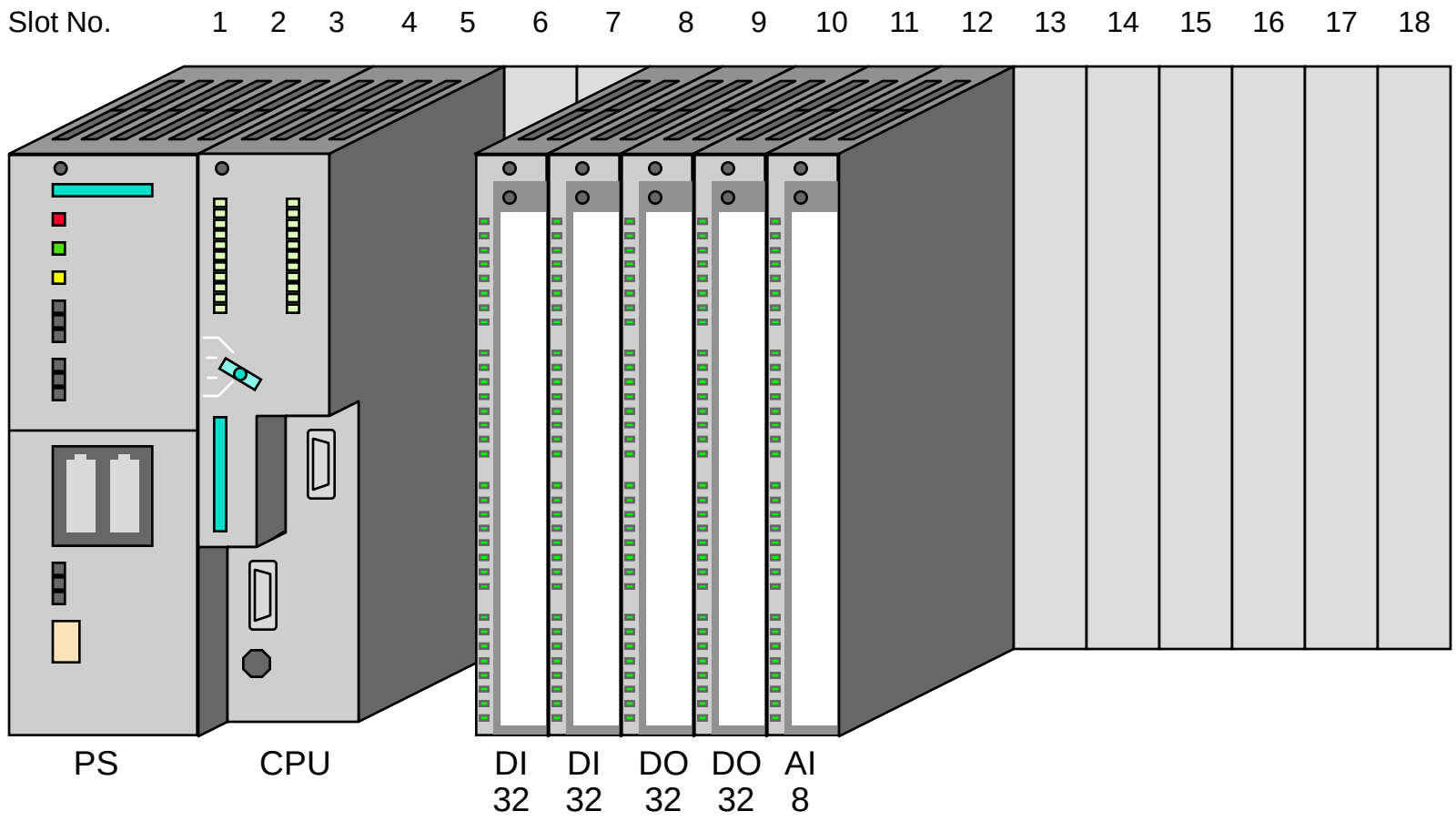
Complete Symbol List (2)

Complete Symbol List (3)

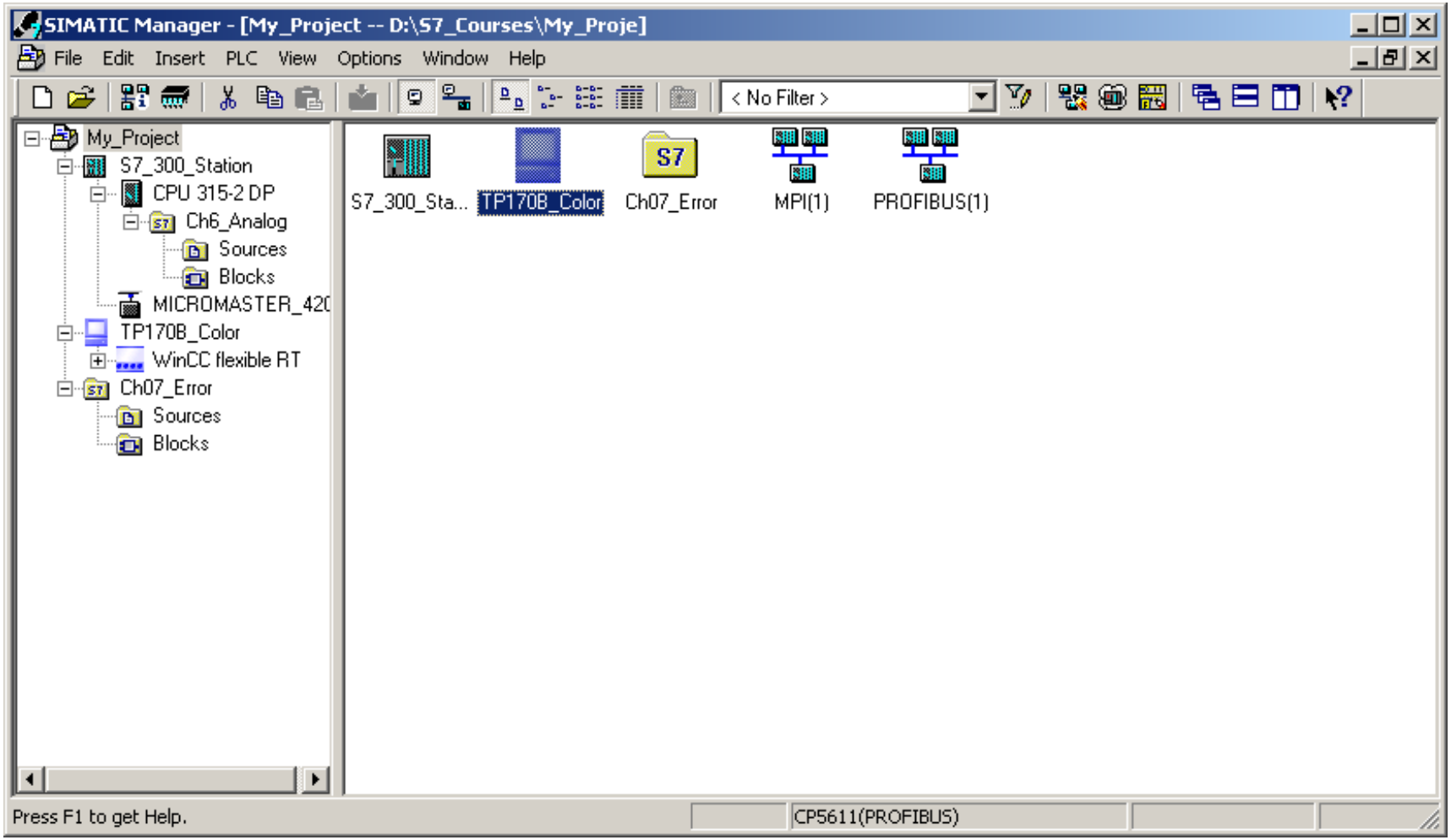
If You Want to Know More

Training Area Setup with S7- 400

Configuration of the S7-400 Training Unit



The SIMATIC Manager



Objectives

Upon completion of the chapter the participant will ...

... understand the project structure in the SIMATIC Manager

… understand the function "accessible nodes"

... be familiar with the offline / online view in the SIMATIC Manager

... be familiar with the STEP7 Standard Libraries

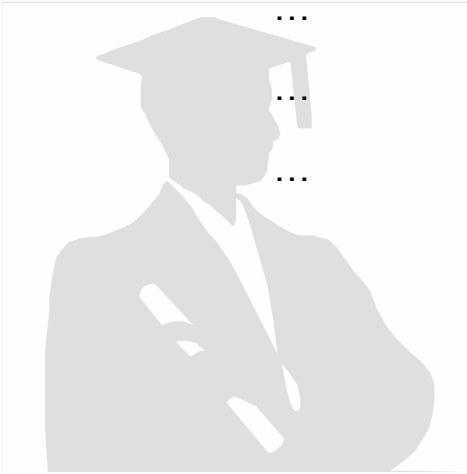
... be familiar with the STEP7 help system

... be able to create and manage a project with the SIMATIC Manager

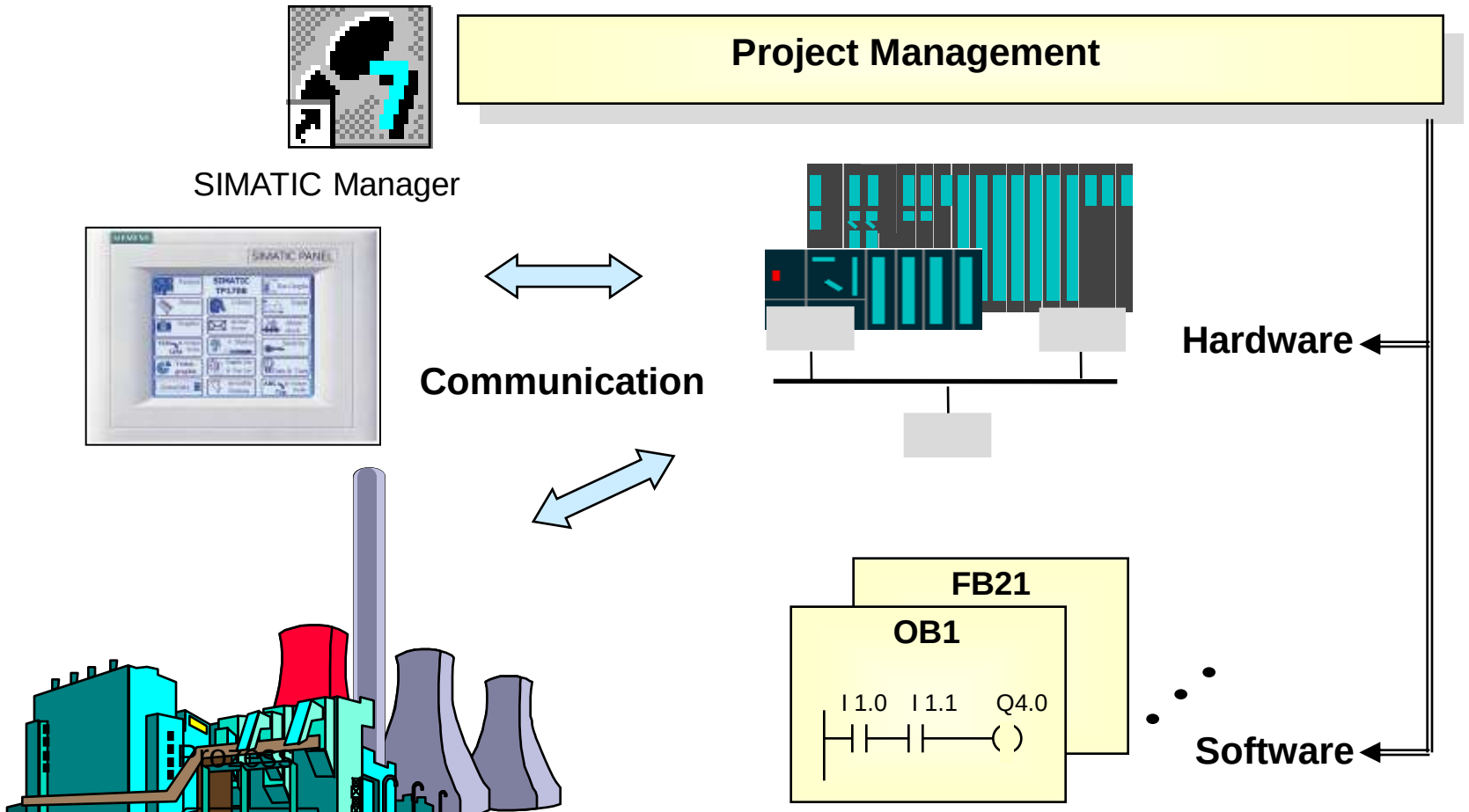
… be able to set the PG´s interface

… be able to erase an MMC

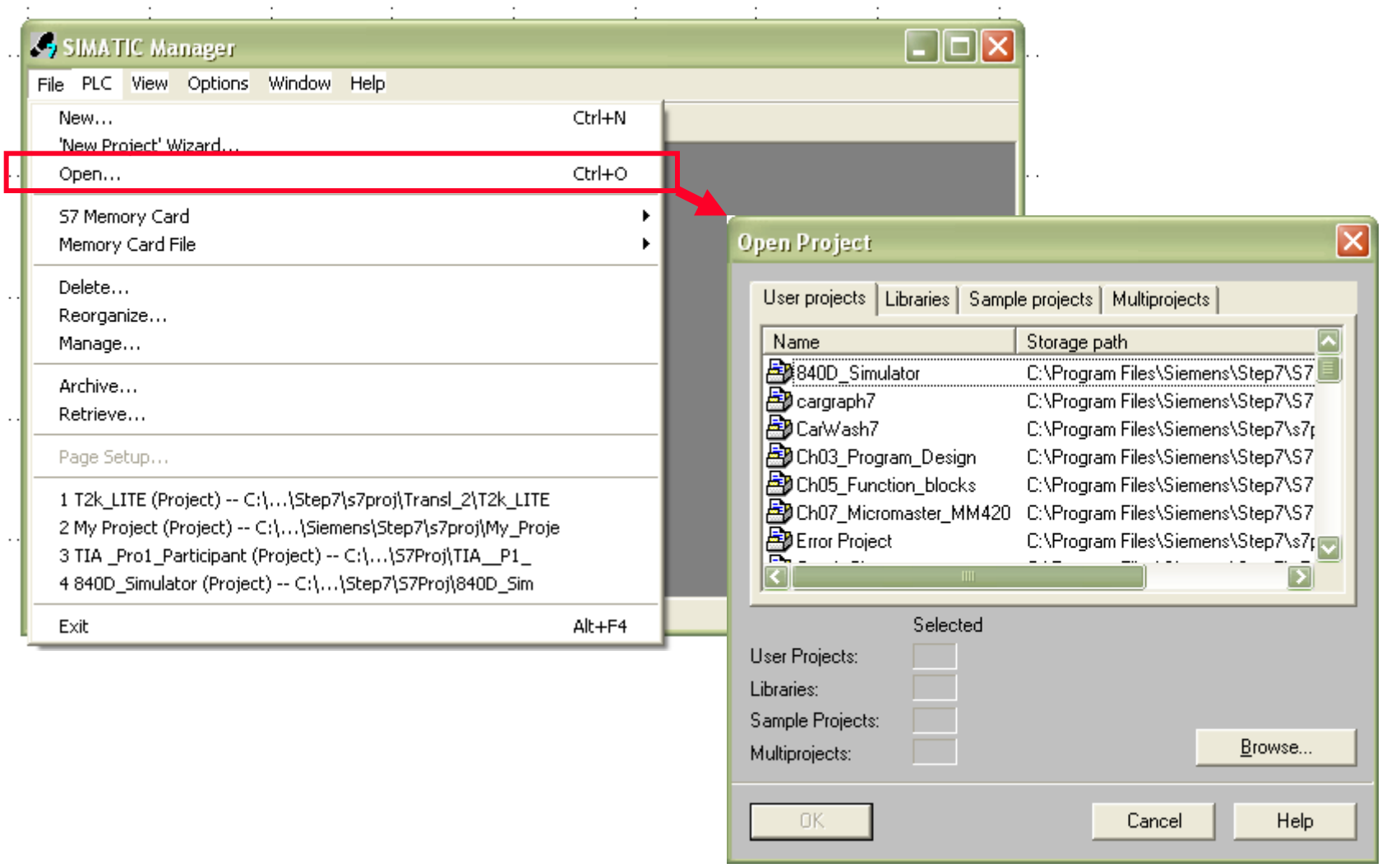
… be able to perform an S7 CPU memory reset



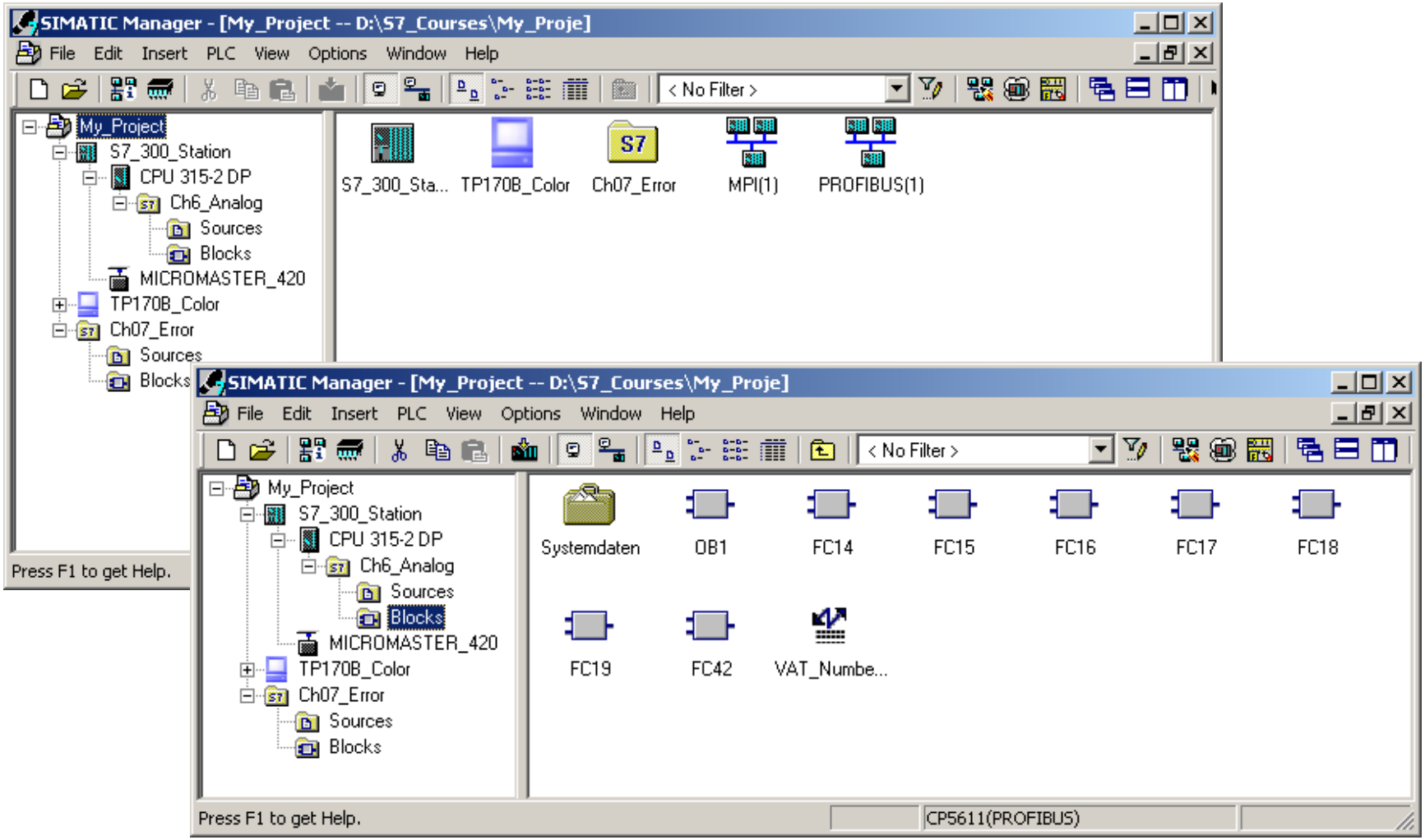
From Process to Project



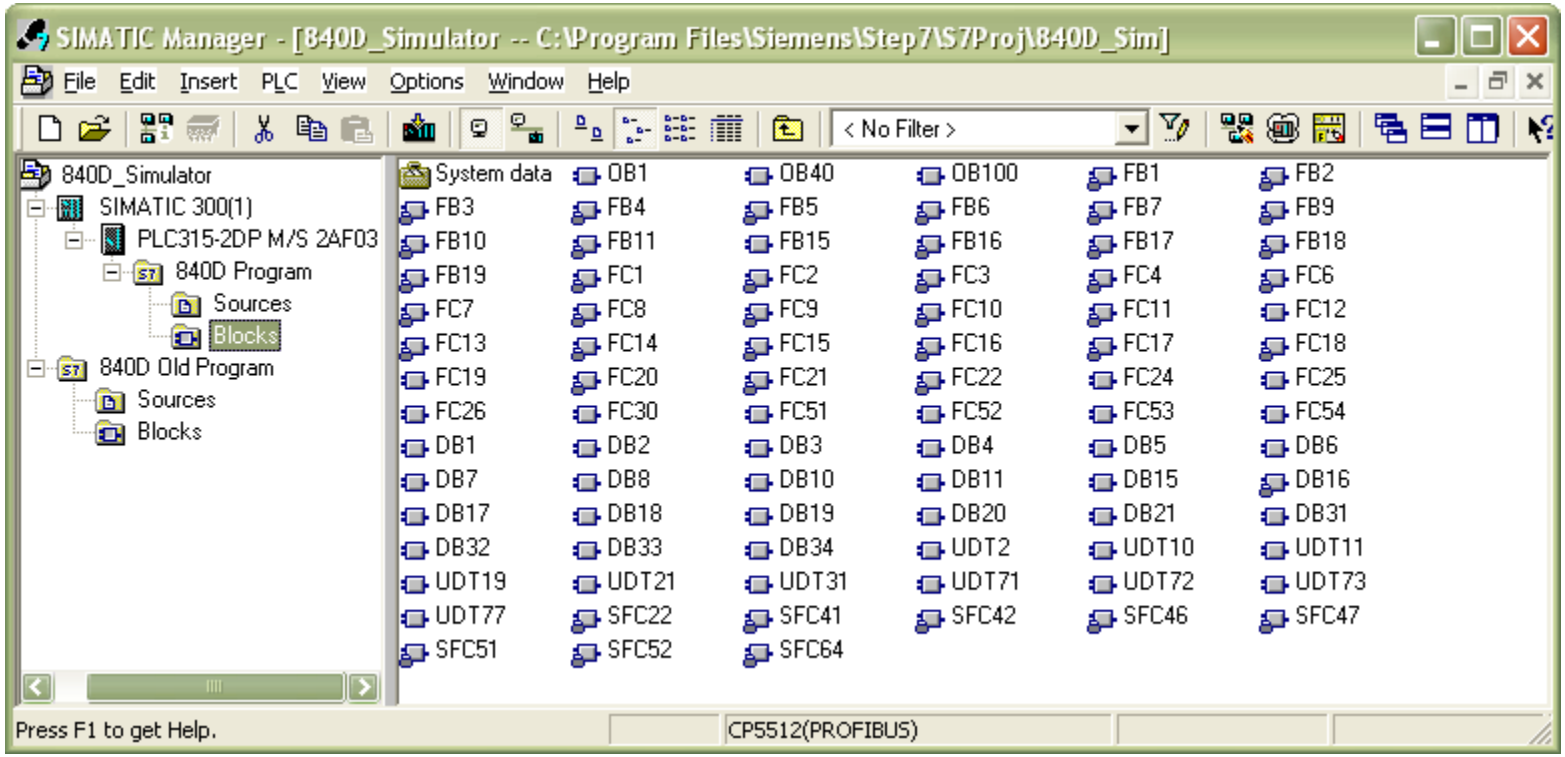
Opening a Project, Deleting a Project, and Project Save As



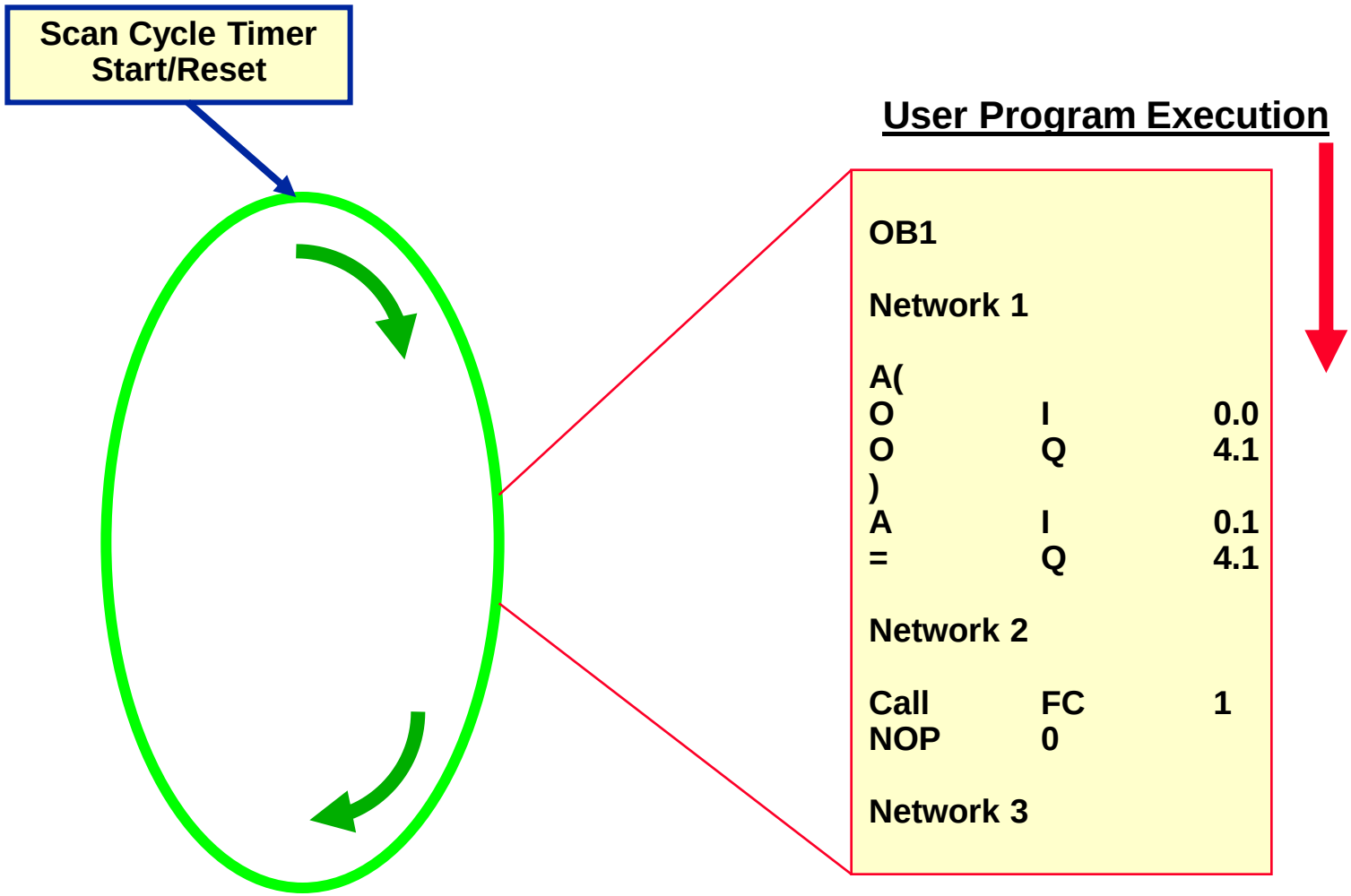
STEP 7 Project Structure



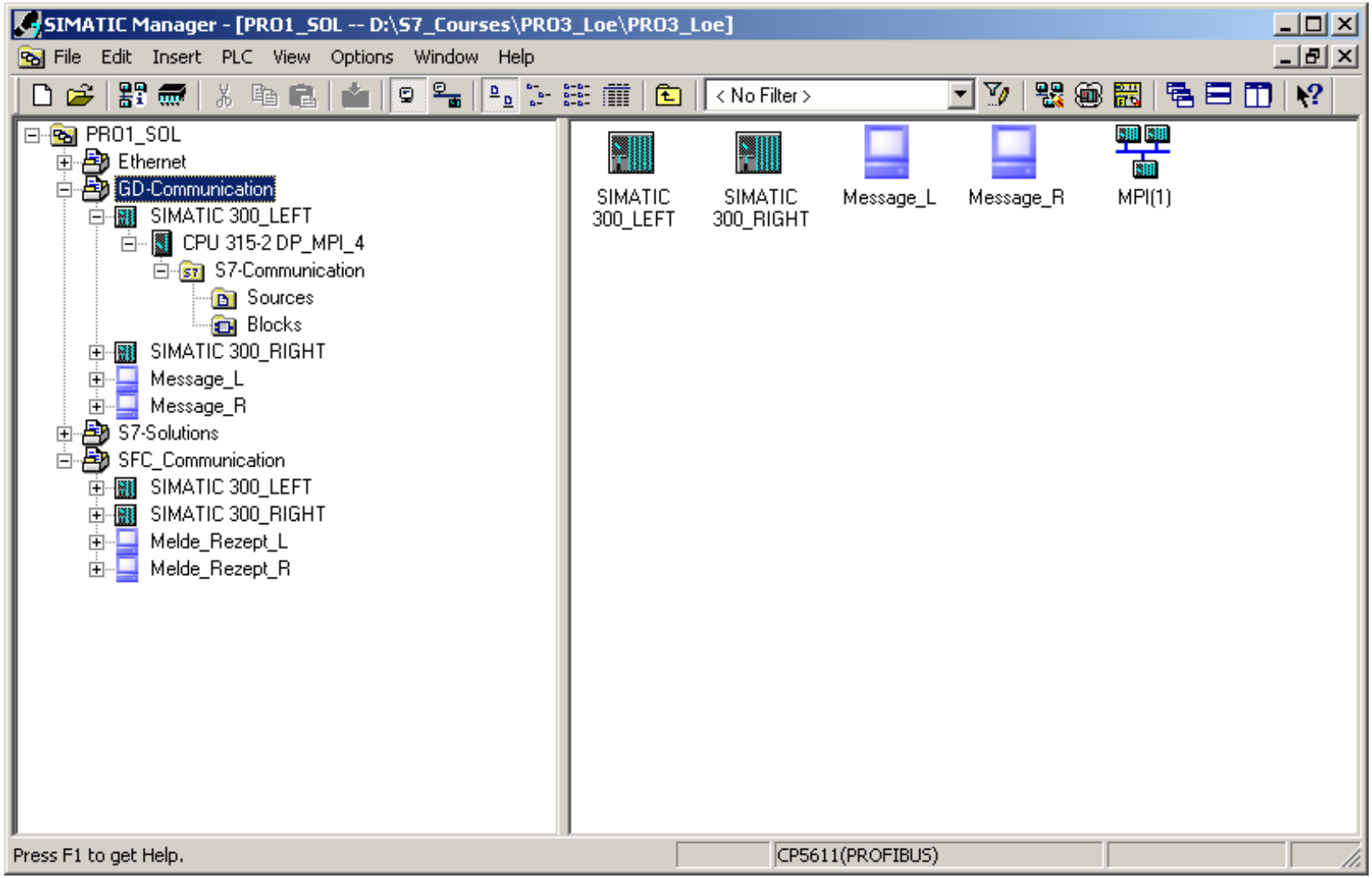
S7 Program Blocks



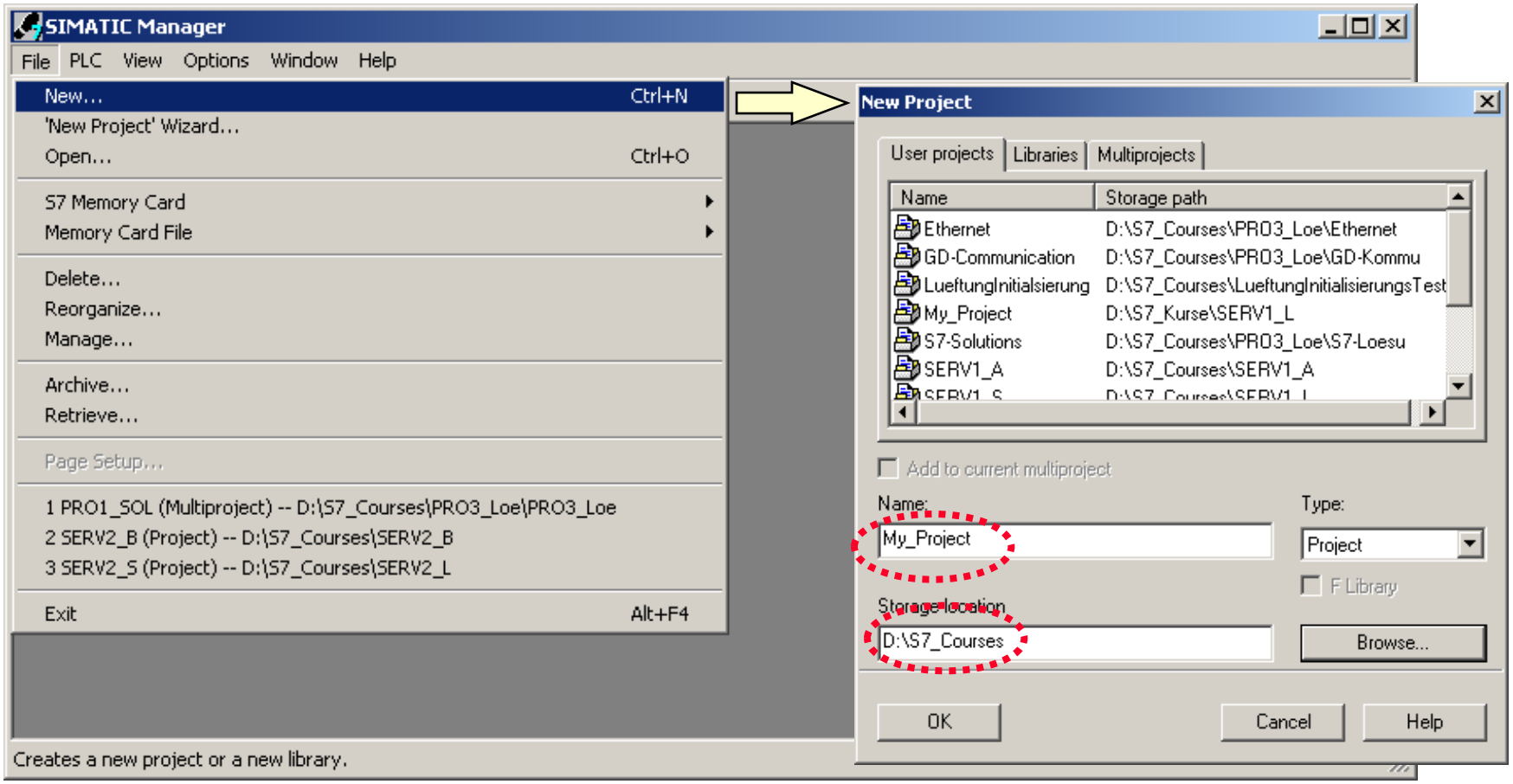
Cyclic Program Processing



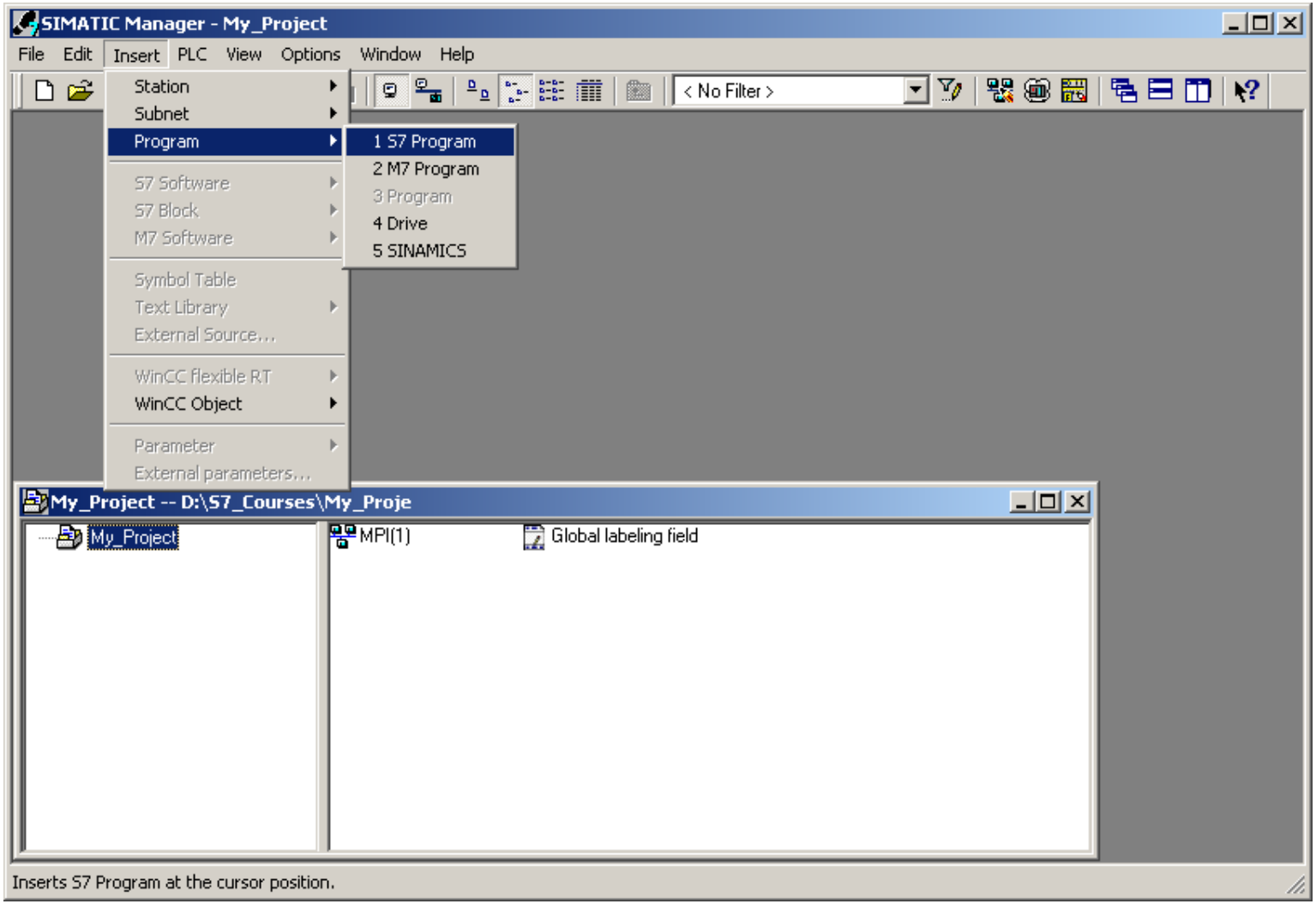
Interesting Facts about Multiprojects



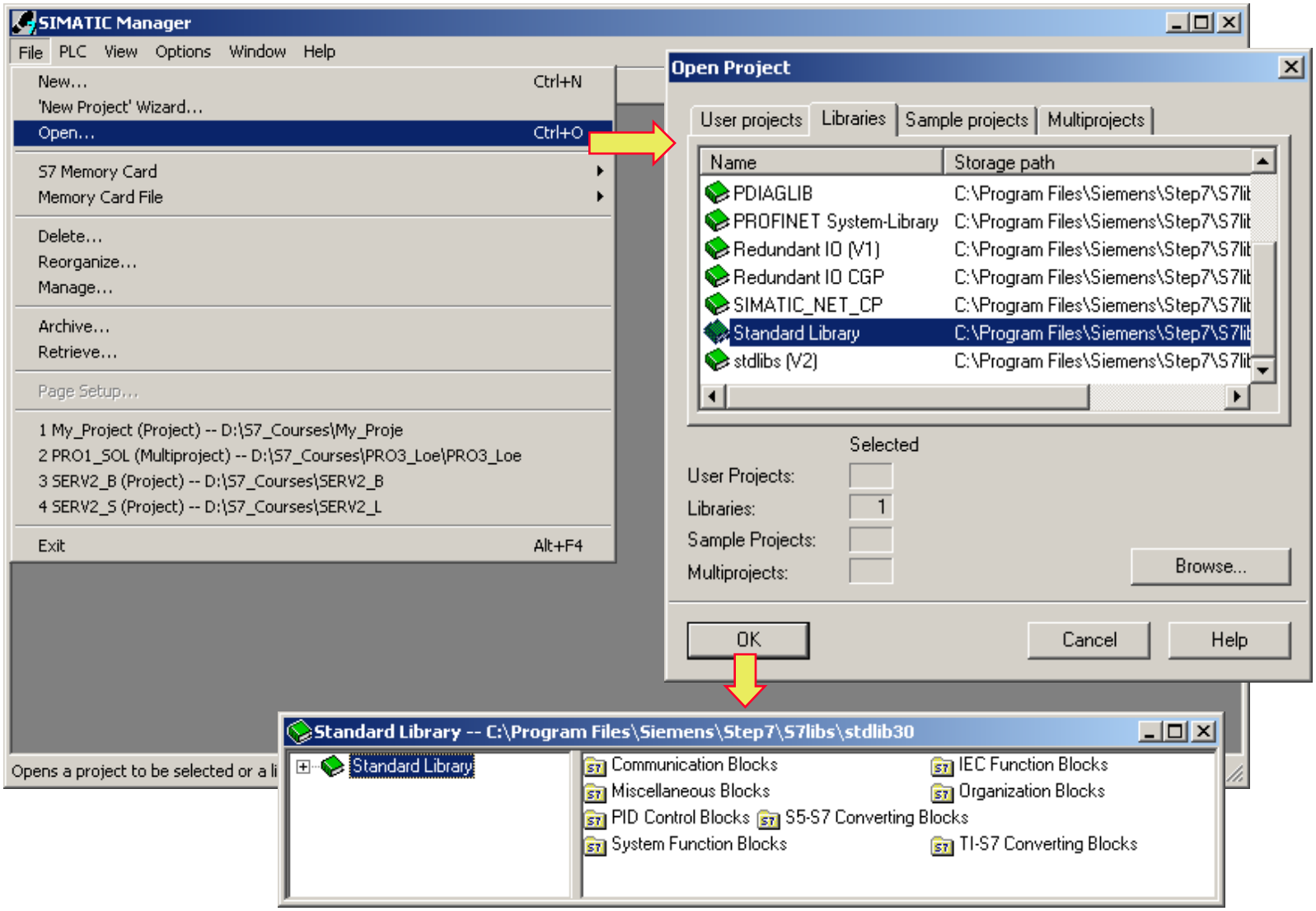
Creating an S7 Project



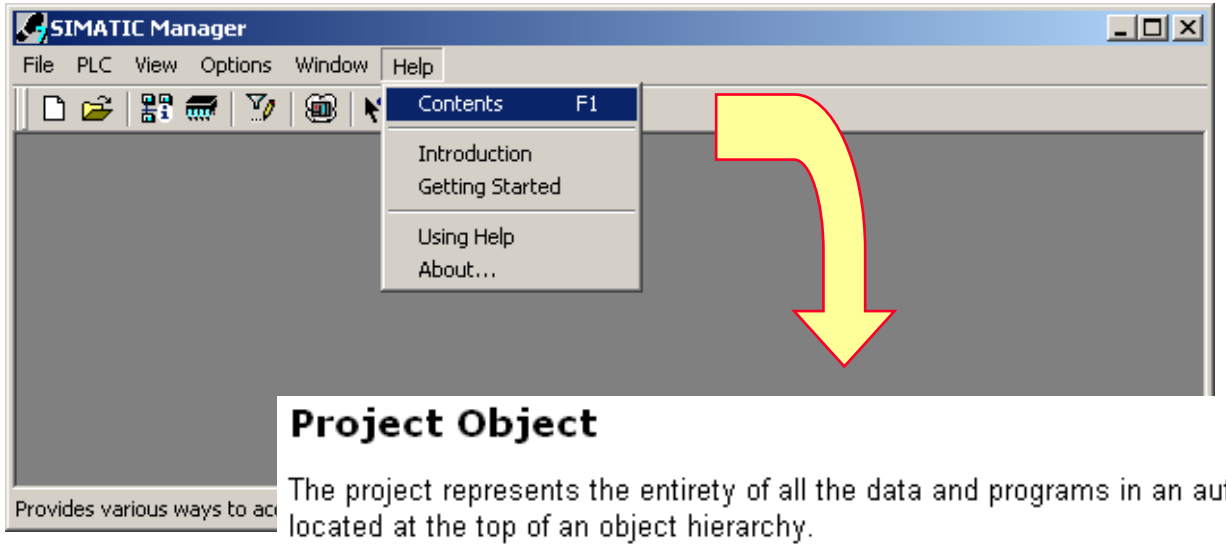
Inserting an S7 Program



Standard Library



STEP 7 Help System



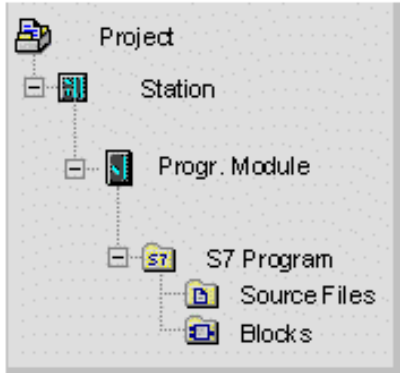
Press Hot-key F1

Project Object

The project represents the entirety of all the data and programs in an automation solution, and is located at the top of an object hierarchy.

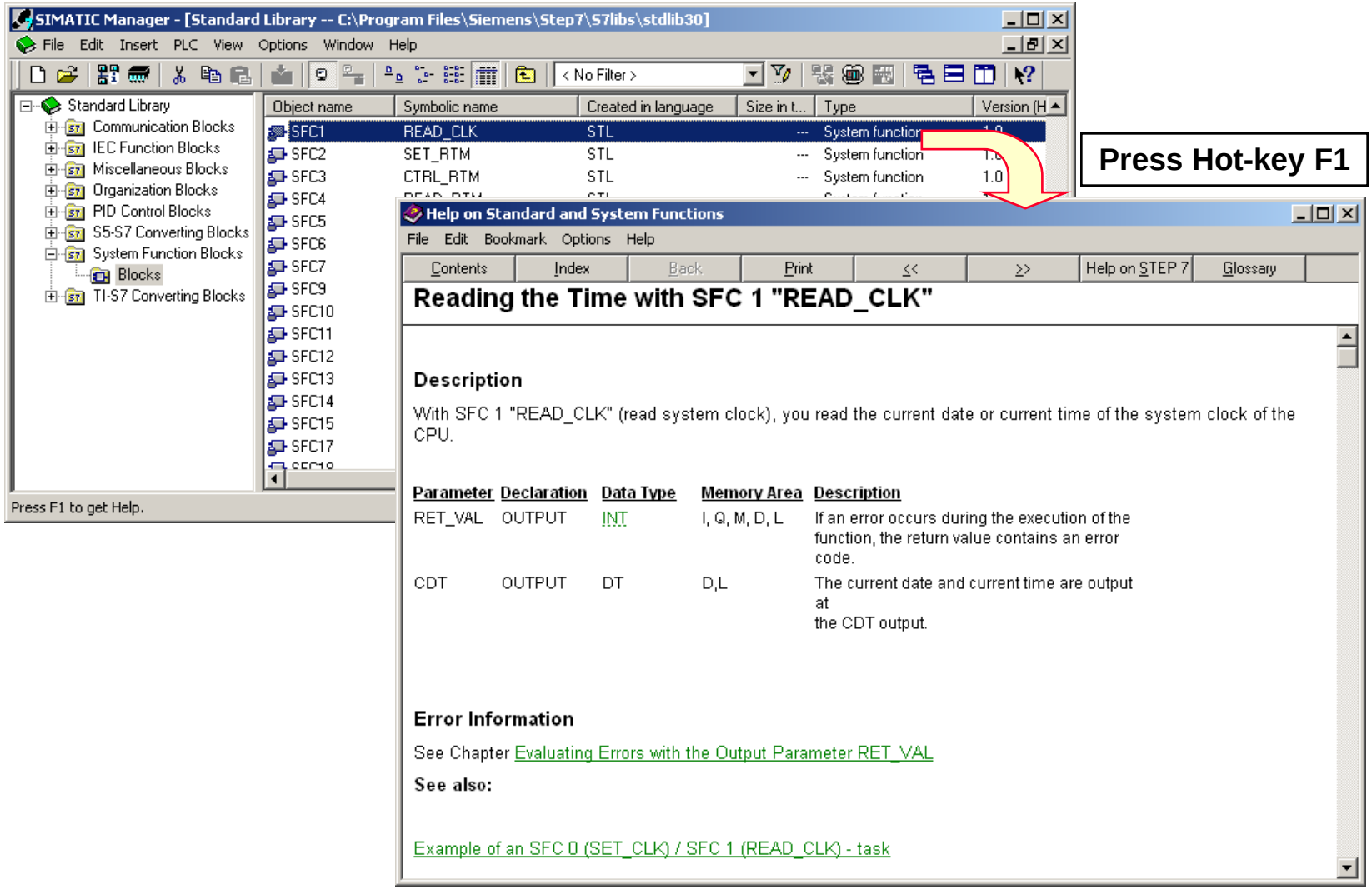
Position in the Project View

Example



Project Object
Station Object
Programmable Module Object
S7/M7 Program Object
Source File Folder Object
Block Folder Object

Context Sensitive Help in STEP 7



Setting the PG/PC Interface

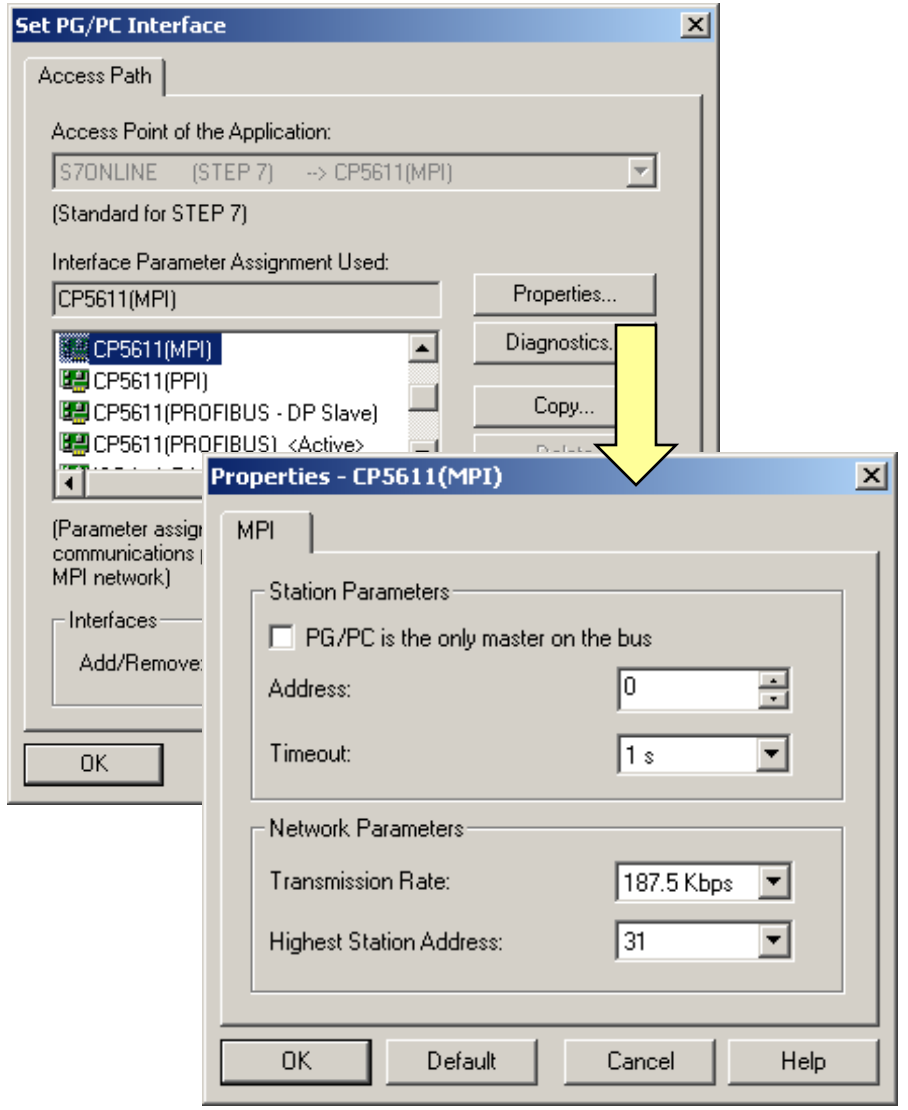
STEP 7

Access Point of the Application: S7ONLINE

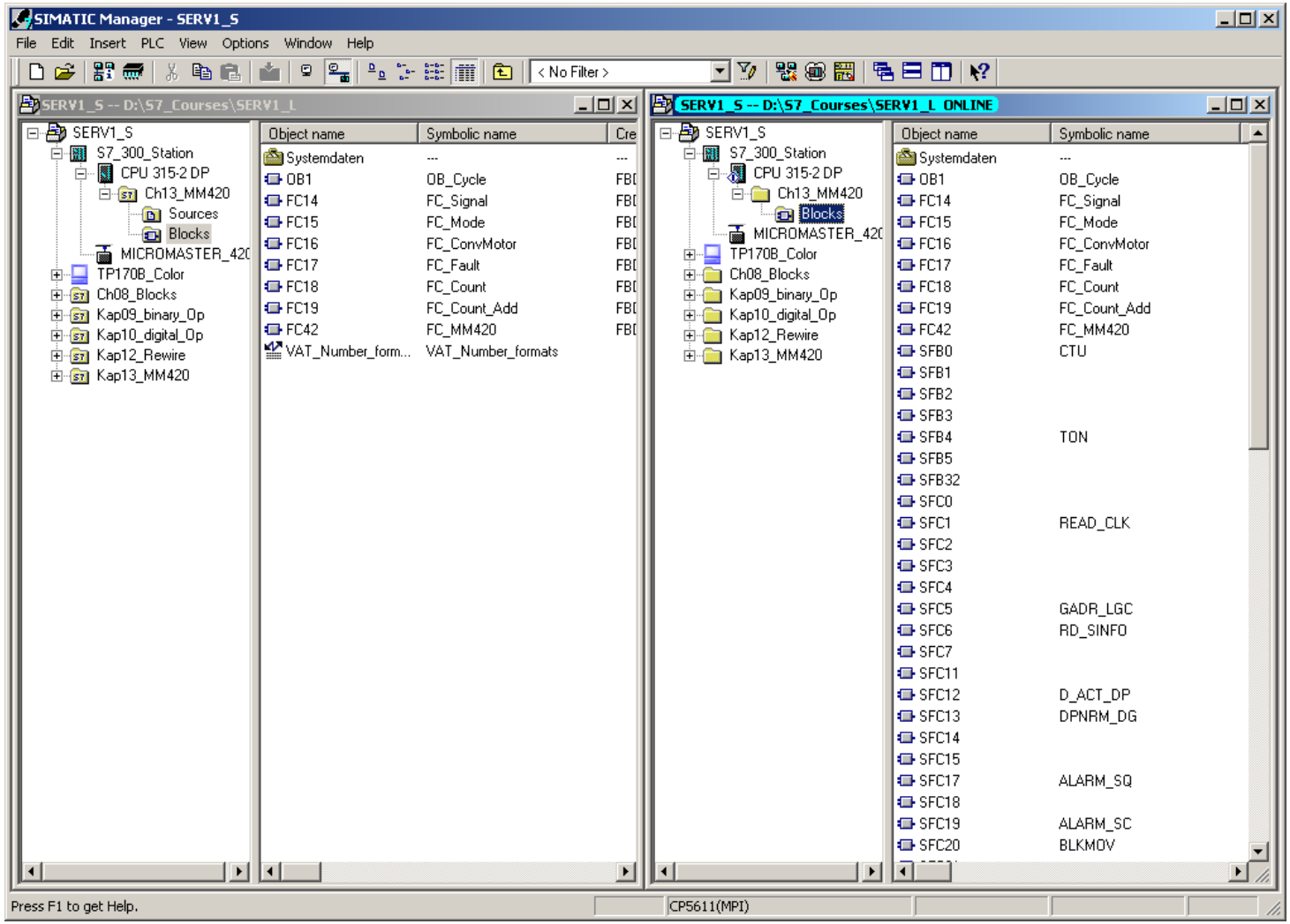
CP5611

Interface Parameter Assignment Used: CP5611 (MPI)

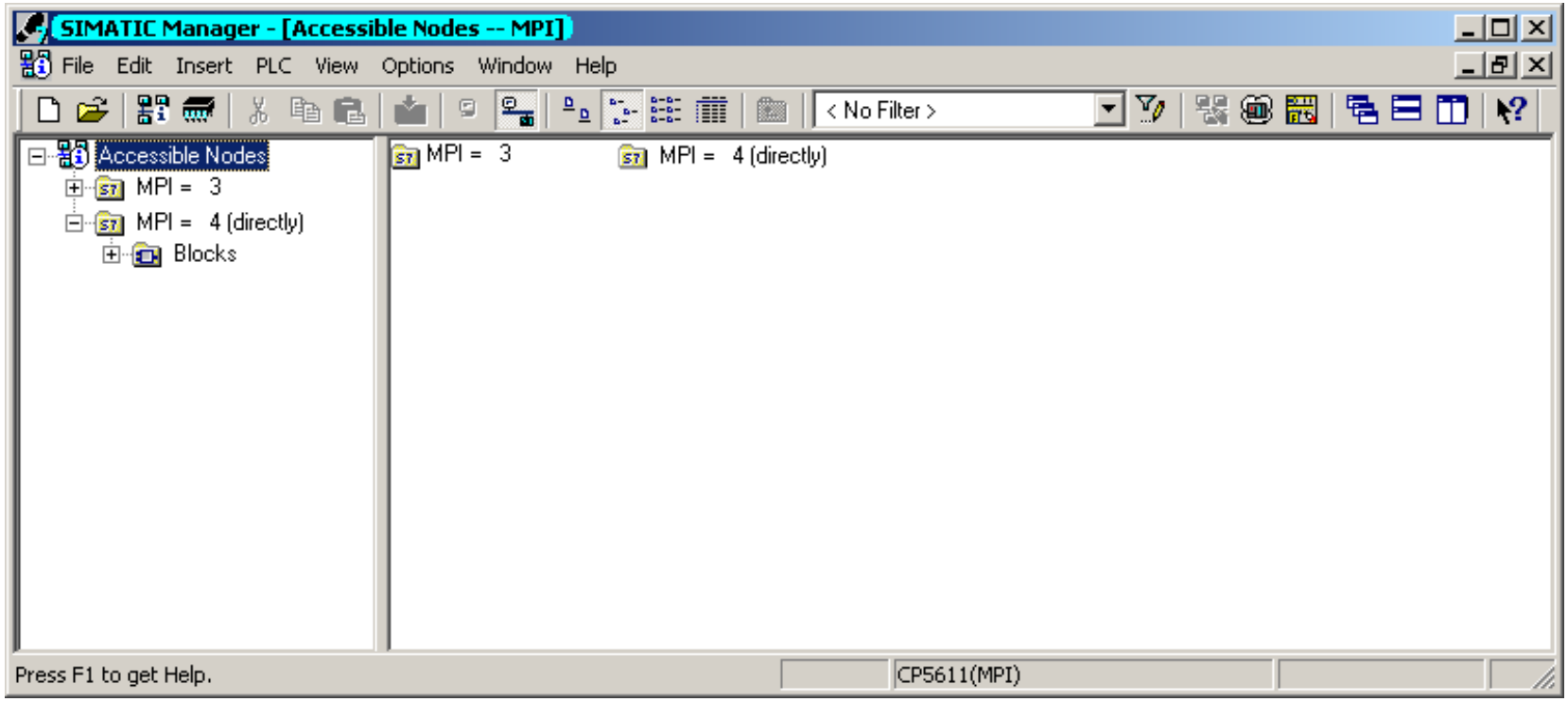
CPU



Offline / Online View in the SIMATIC Manager



Online Connection using "Accessible Nodes"



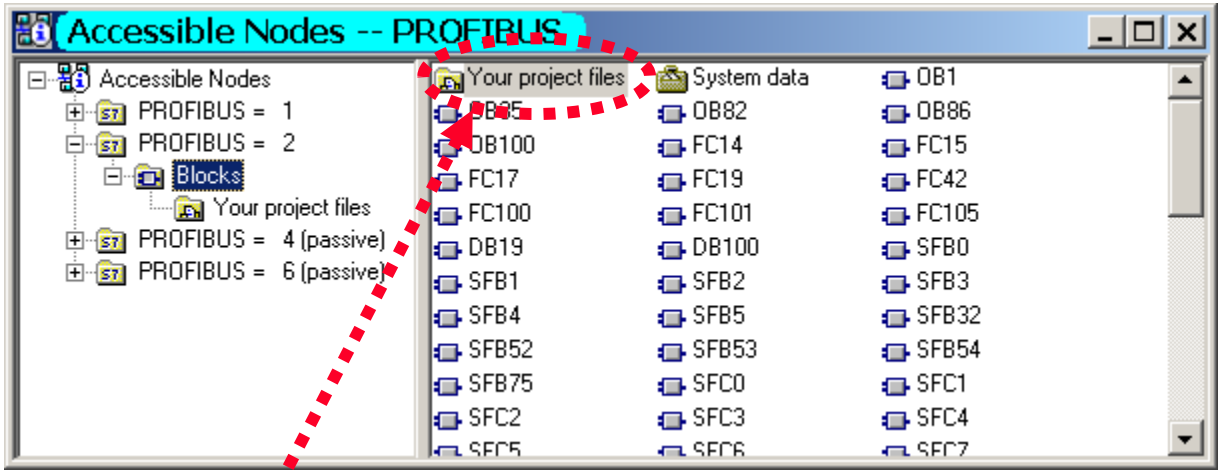
Erasing Data Stored on the MMC

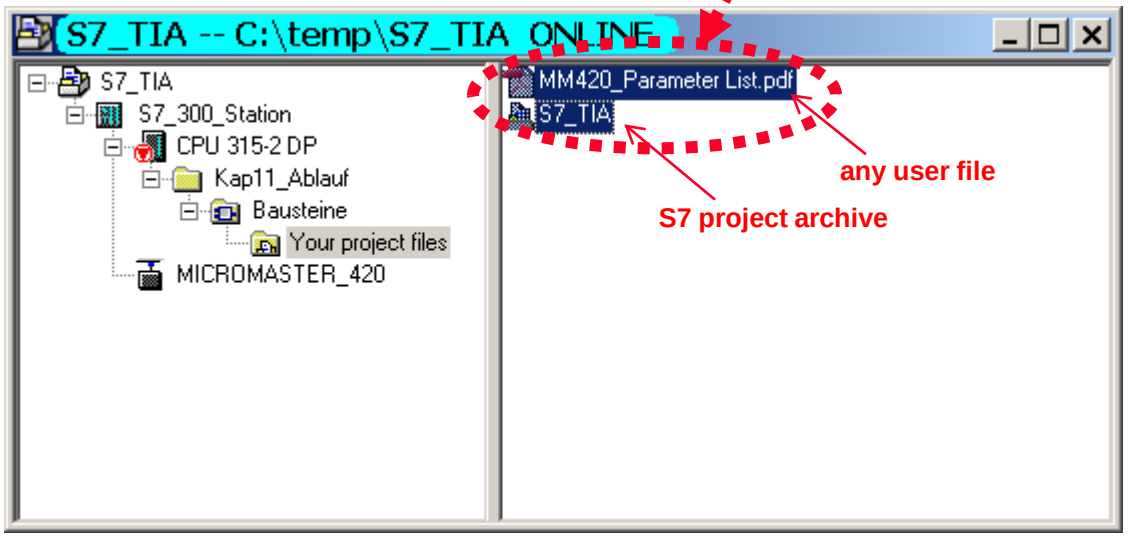
Accessible Nodes:

Icon

or

Menu option "PLC → display accessible nodes"





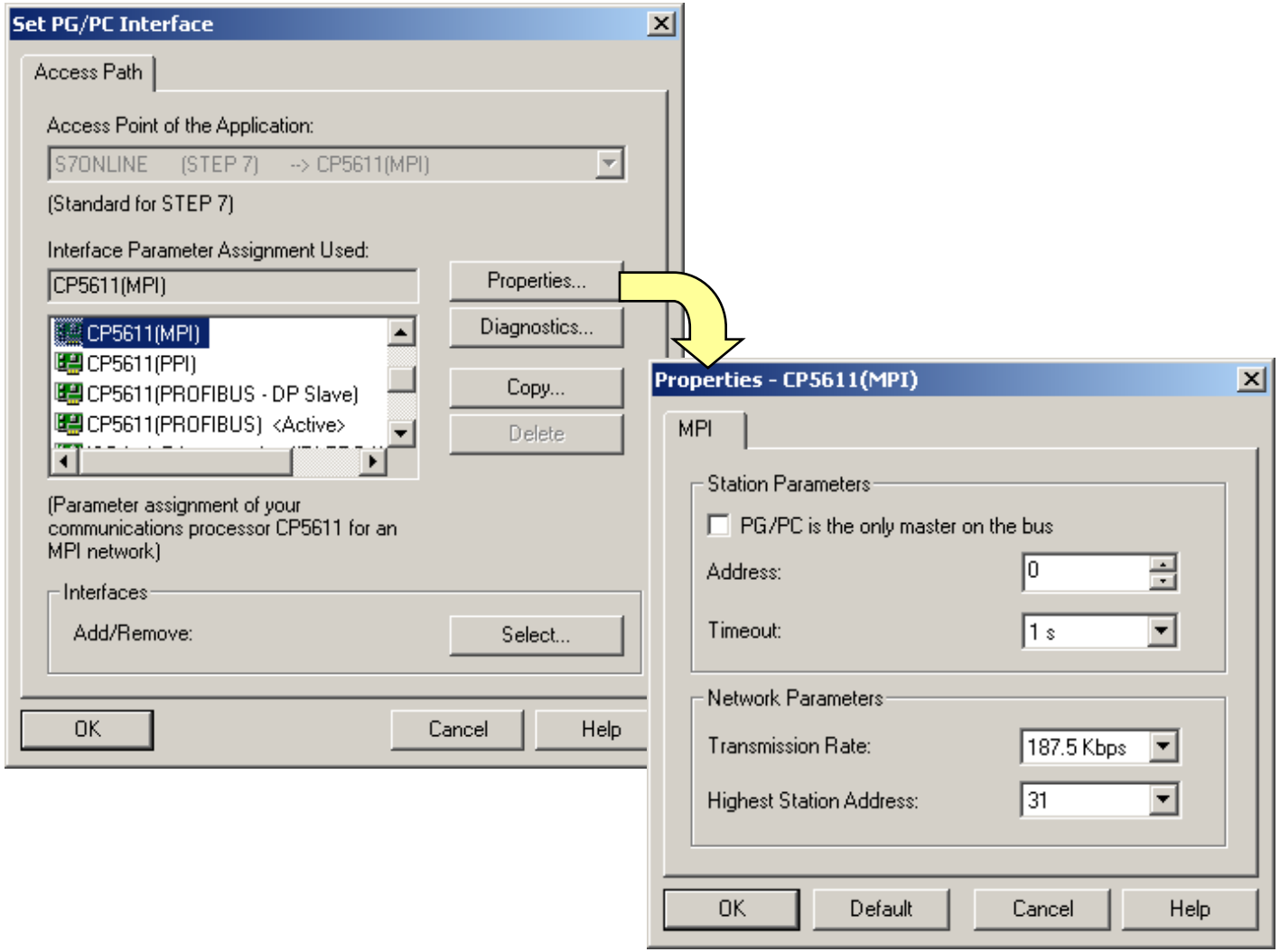
Project – Online – View:

Icon

or

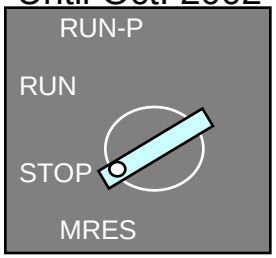
Menu option "View → online"

Exercise 1: Setting the PG Interface to MPI

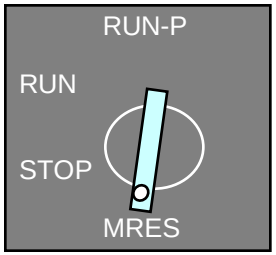


Exercise 2: Performing a CPU Memory Reset and a Warm Restart

Until Oct. 2002

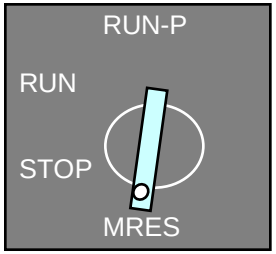


1. Set the mode selector switch to STOP

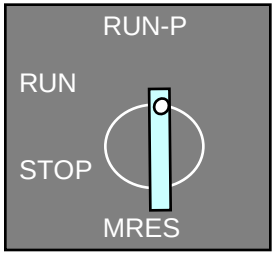


2. Hold (Press) the mode selector switch in the MRES position until the STOP LED has flashed twice slowly. Release the mode selector switch so that it returns to the STOP position.

within 1 sec

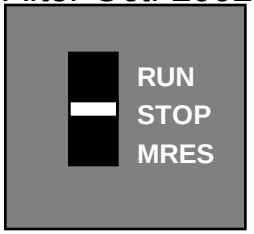


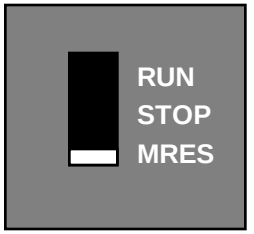
3. Turn (press) the mode selector switch to the MRES posi--tion once more until the STOP LED begins to flash quickly. Release the mode selector switch so that it returns to the STOP position.

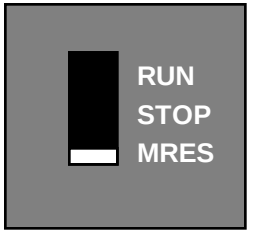


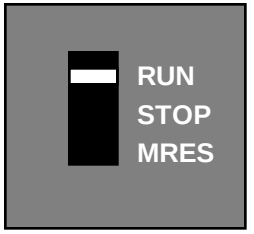
4. Set the mode selector switch to the RUN-P (RUN) position. (A warm restart is carried out in the transition from STOP to RUN/RUN-P)

After Oct. 2002

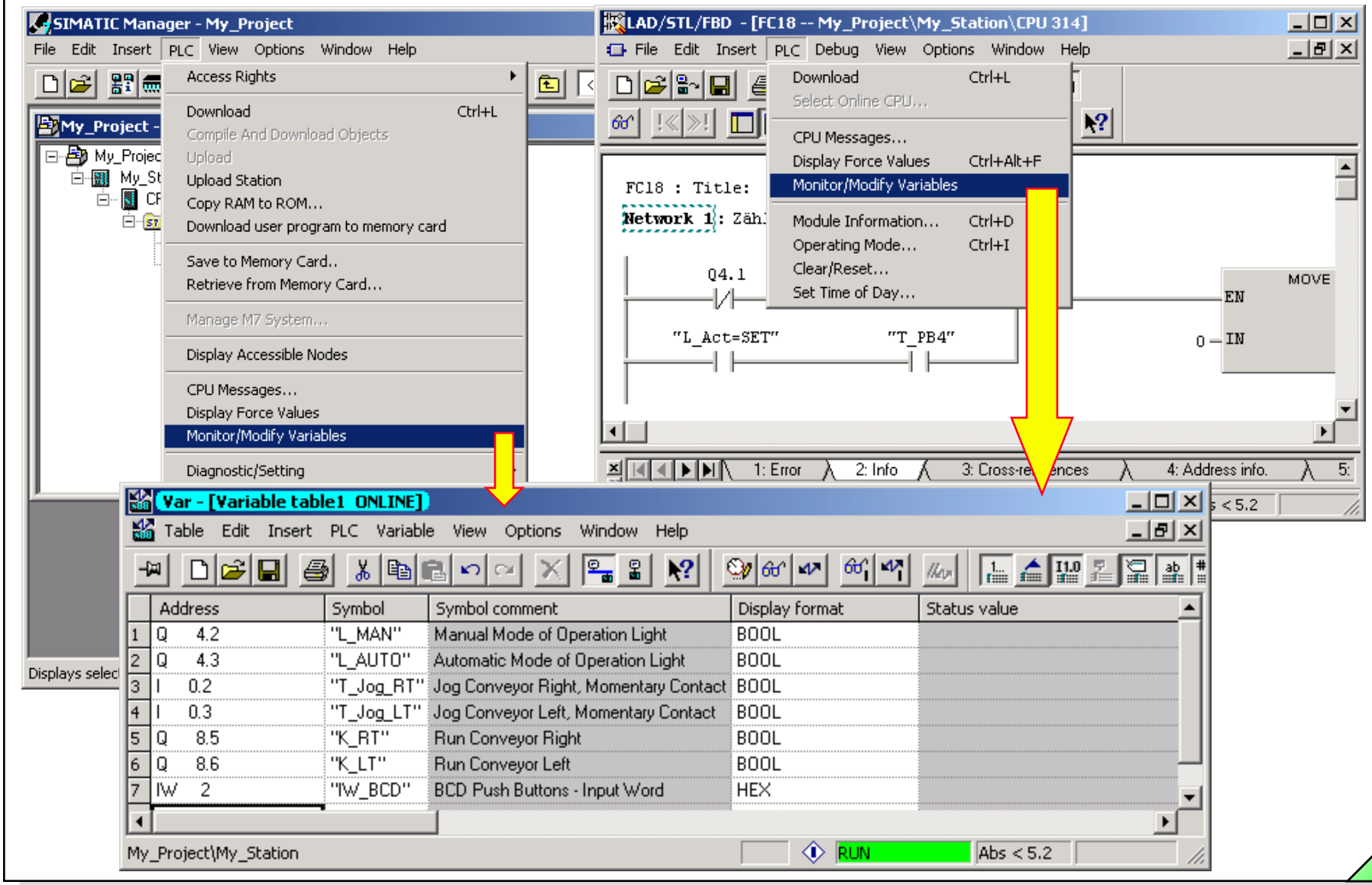




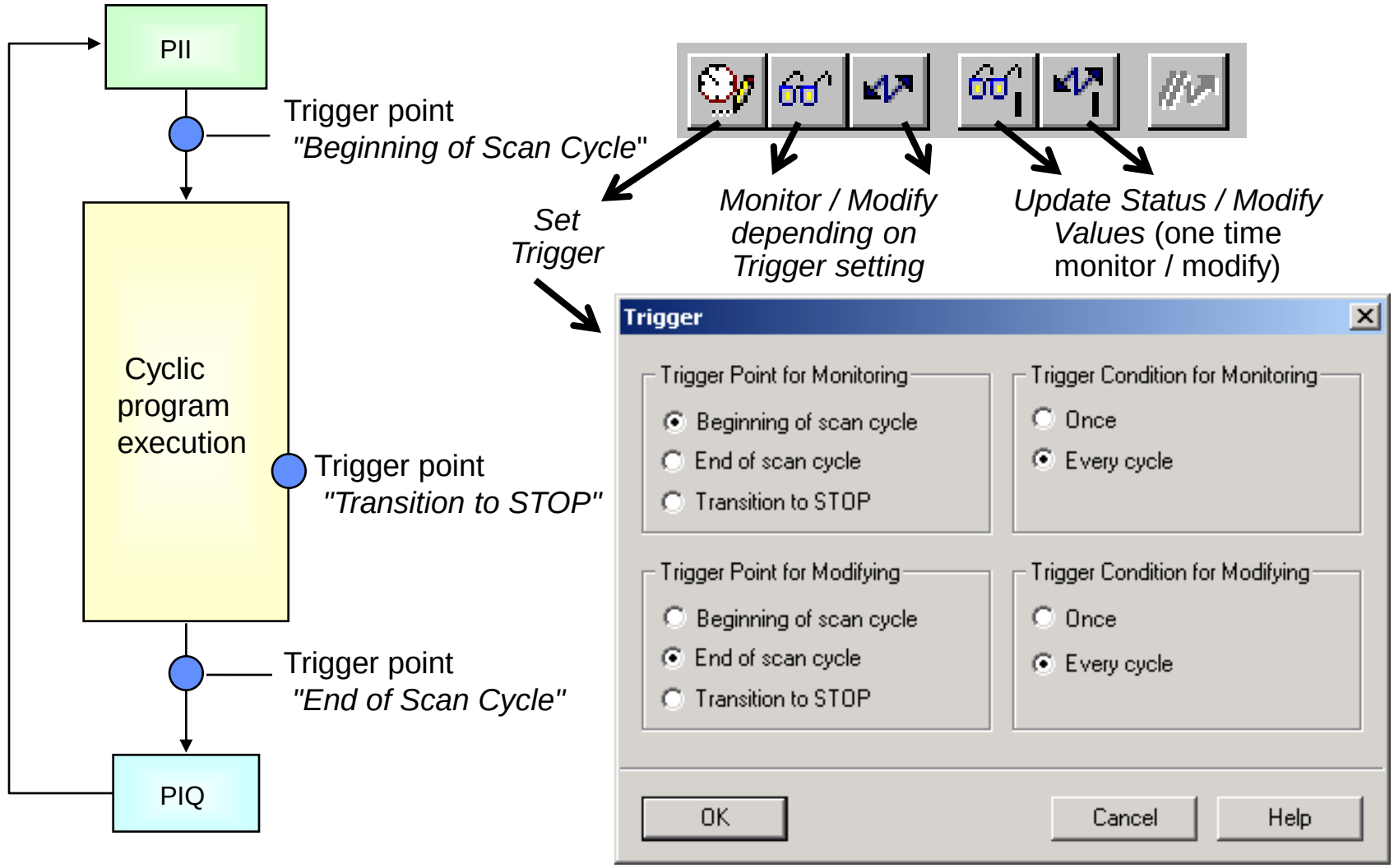




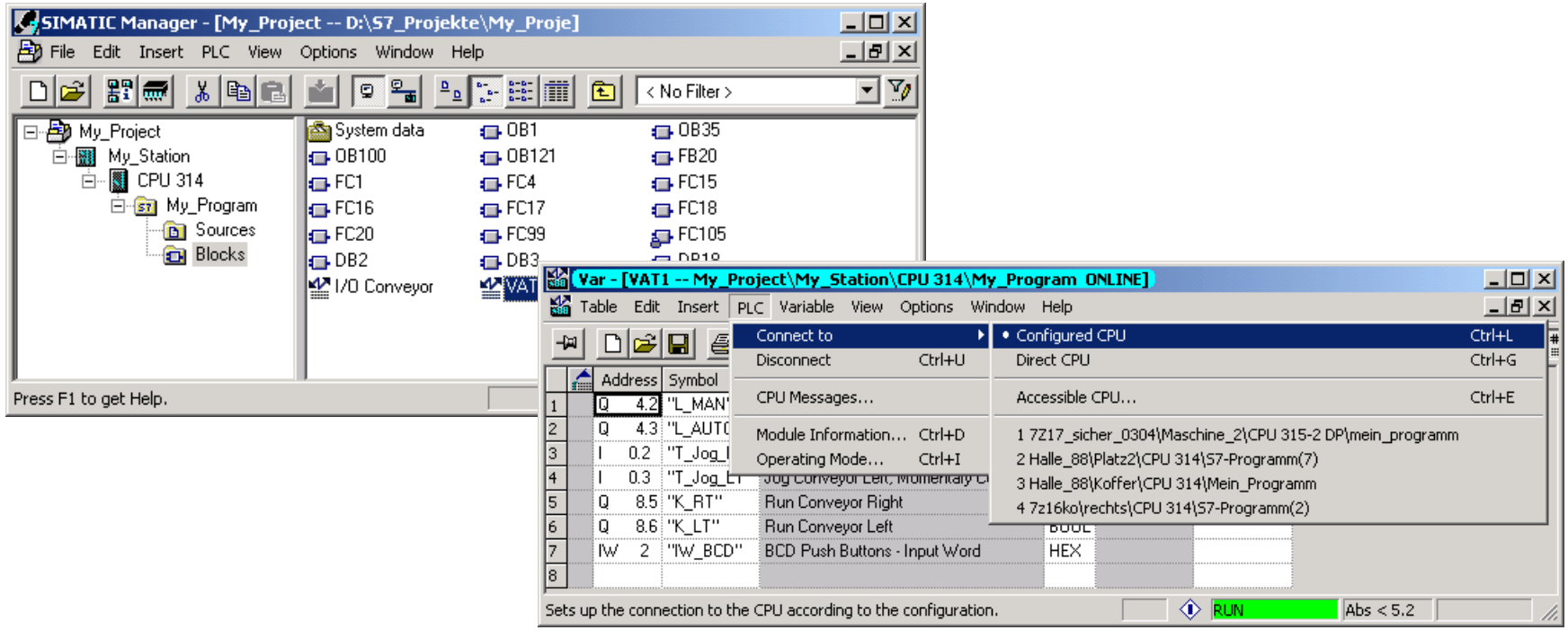
Calling the "Monitor/Modify Variables" Tool

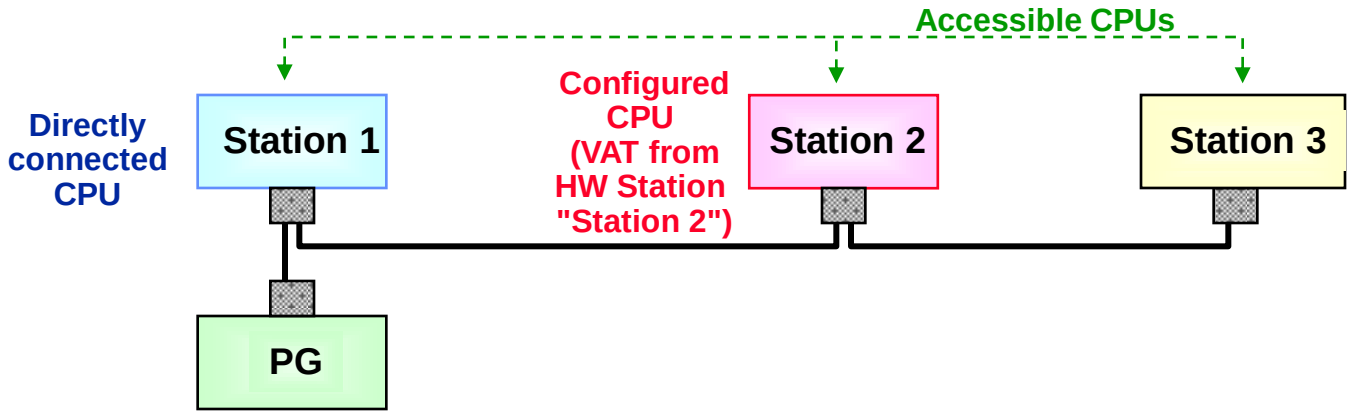


Establishing Trigger Points for "Monitor/Modify Variable"

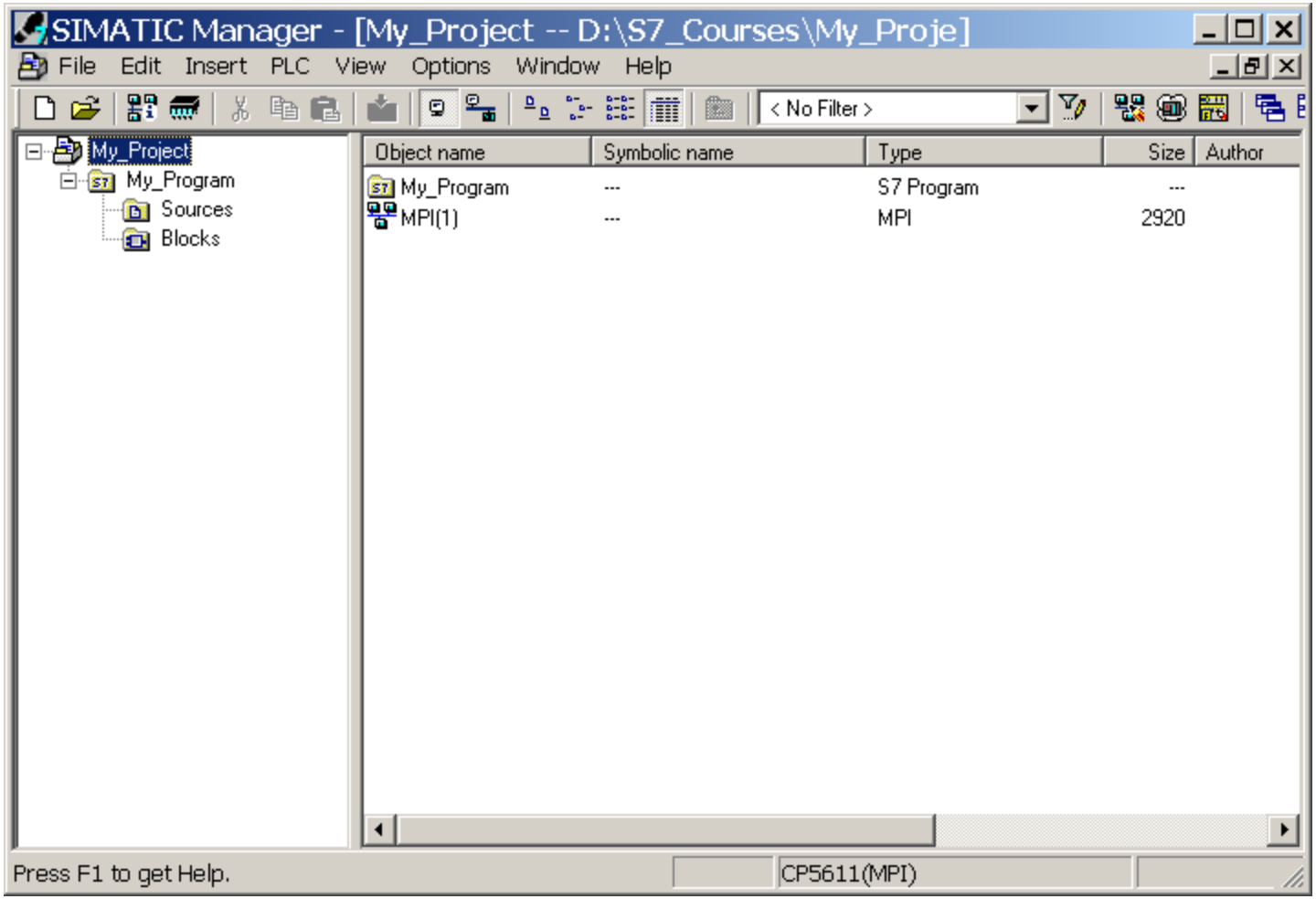


Opening a Variable Table and Establishing a Connection to the CPU



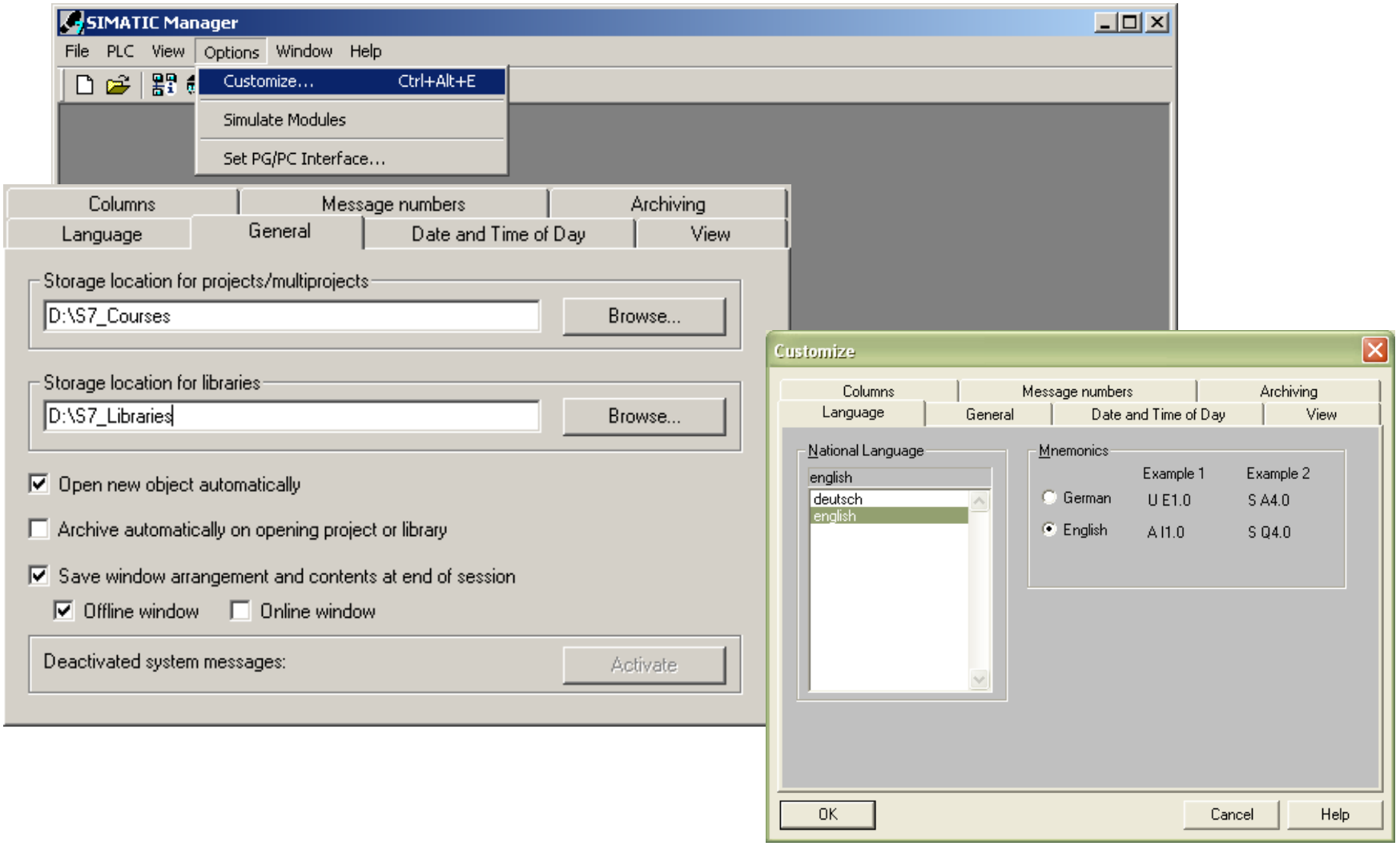


Exercise 3: Creating a Project

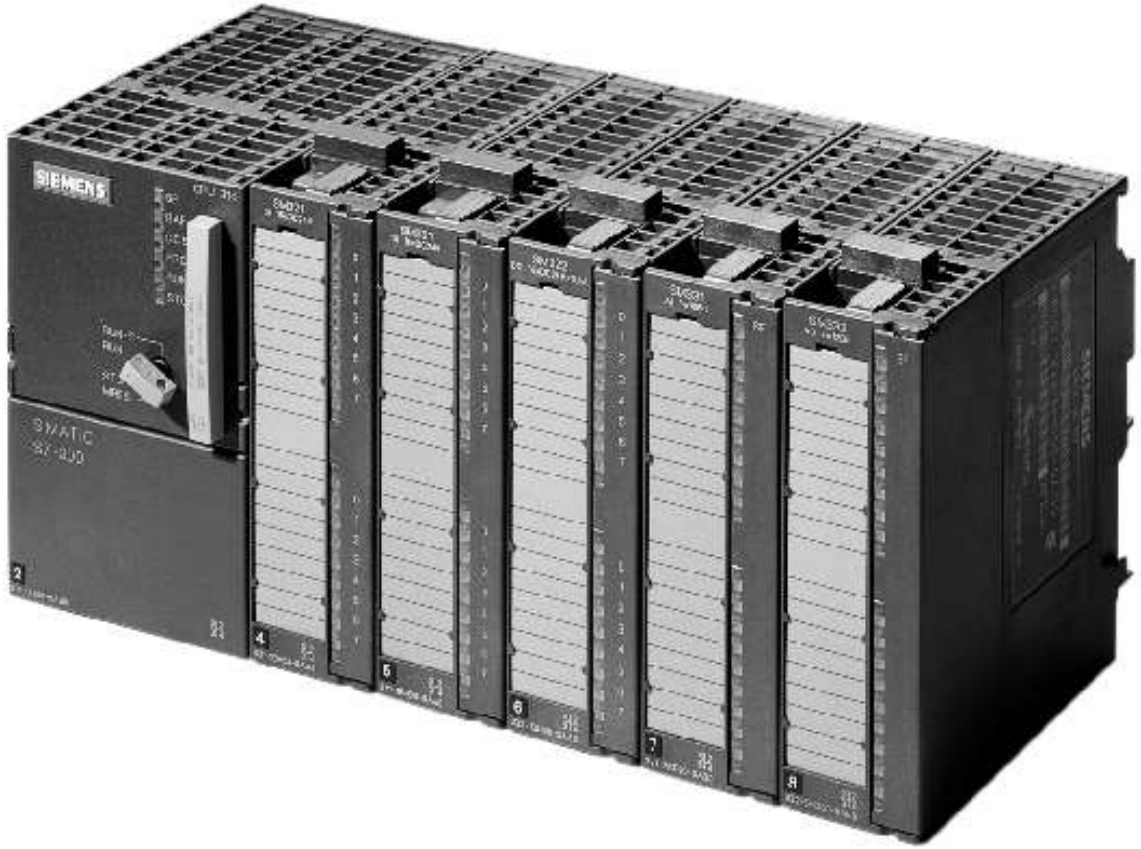


If You Want to Know More

SIMATIC Manager Customizing Options



Hardware Configuration



Objectives

Upon completion of this chapter the participant will ...

... be able to create a setpoint configuration and assign parameters to it

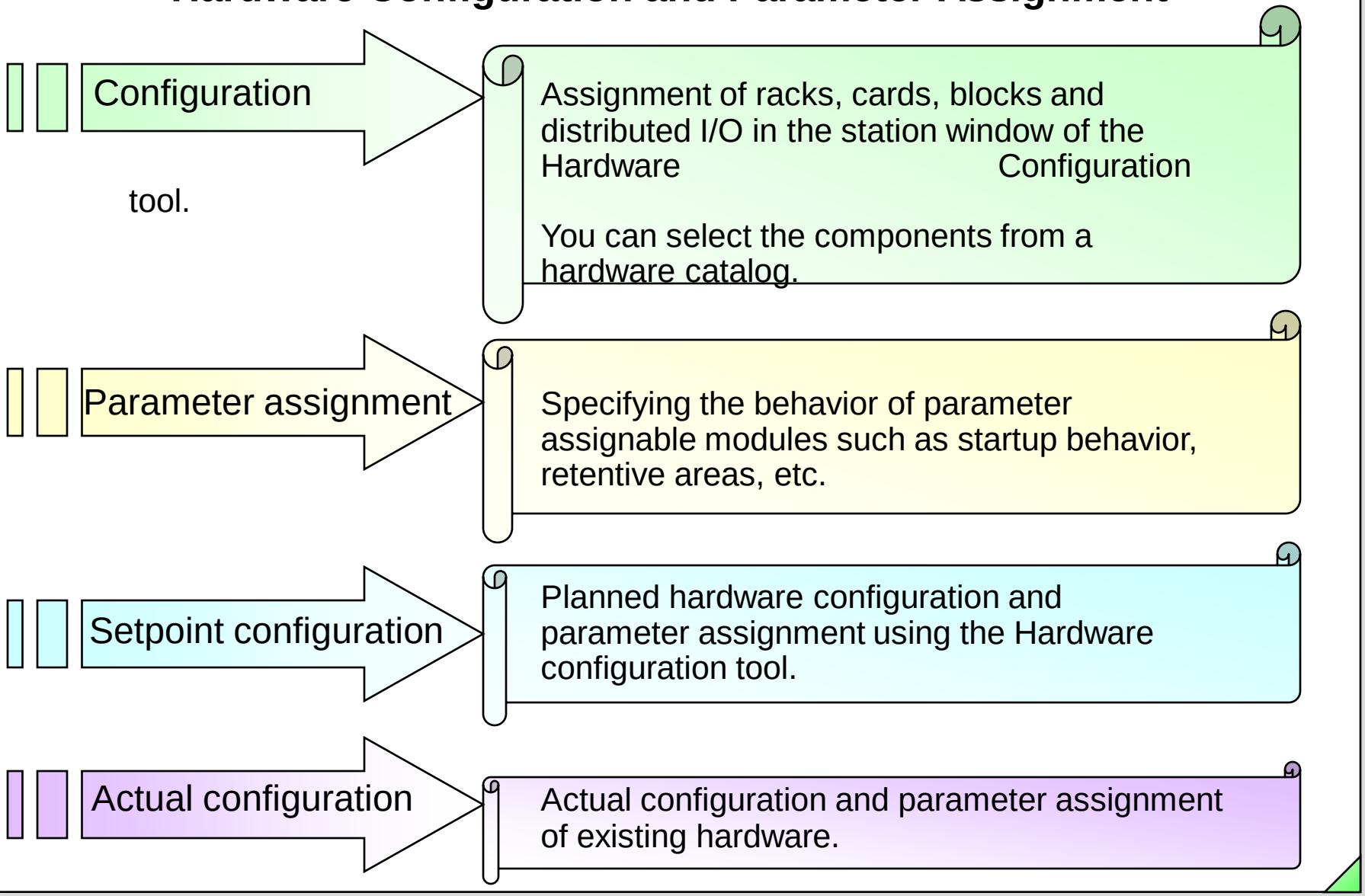
... be able to read out an actual configuration and assign parameters to it

... be familiar with the addressing of S7-300 input and output modules

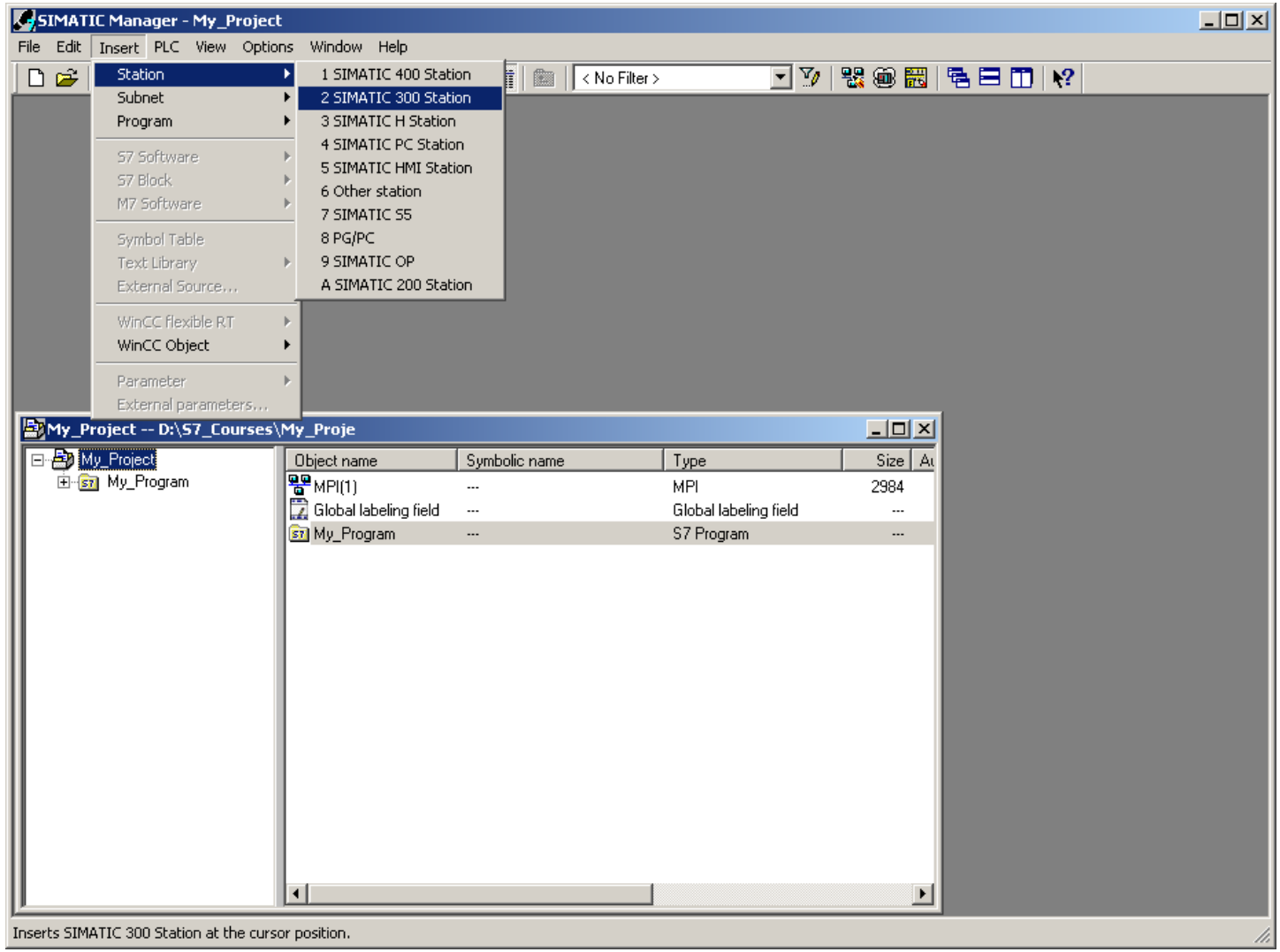
... be able to adapt a given hardware configuration to an existing hardware

… as an option, be familiar with the Catalog CA01 hardware Import/Export function

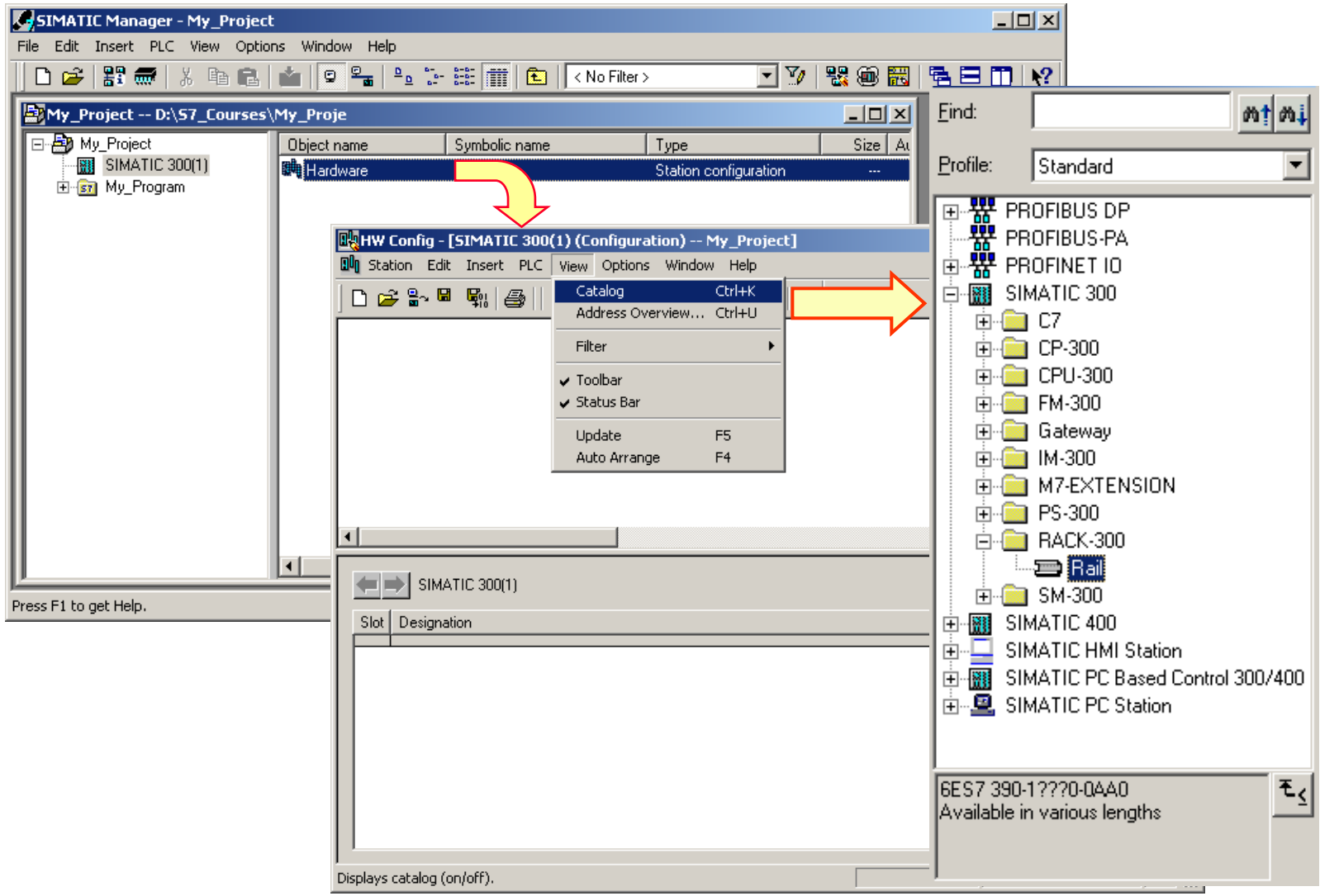
Hardware Configuration and Parameter Assignment



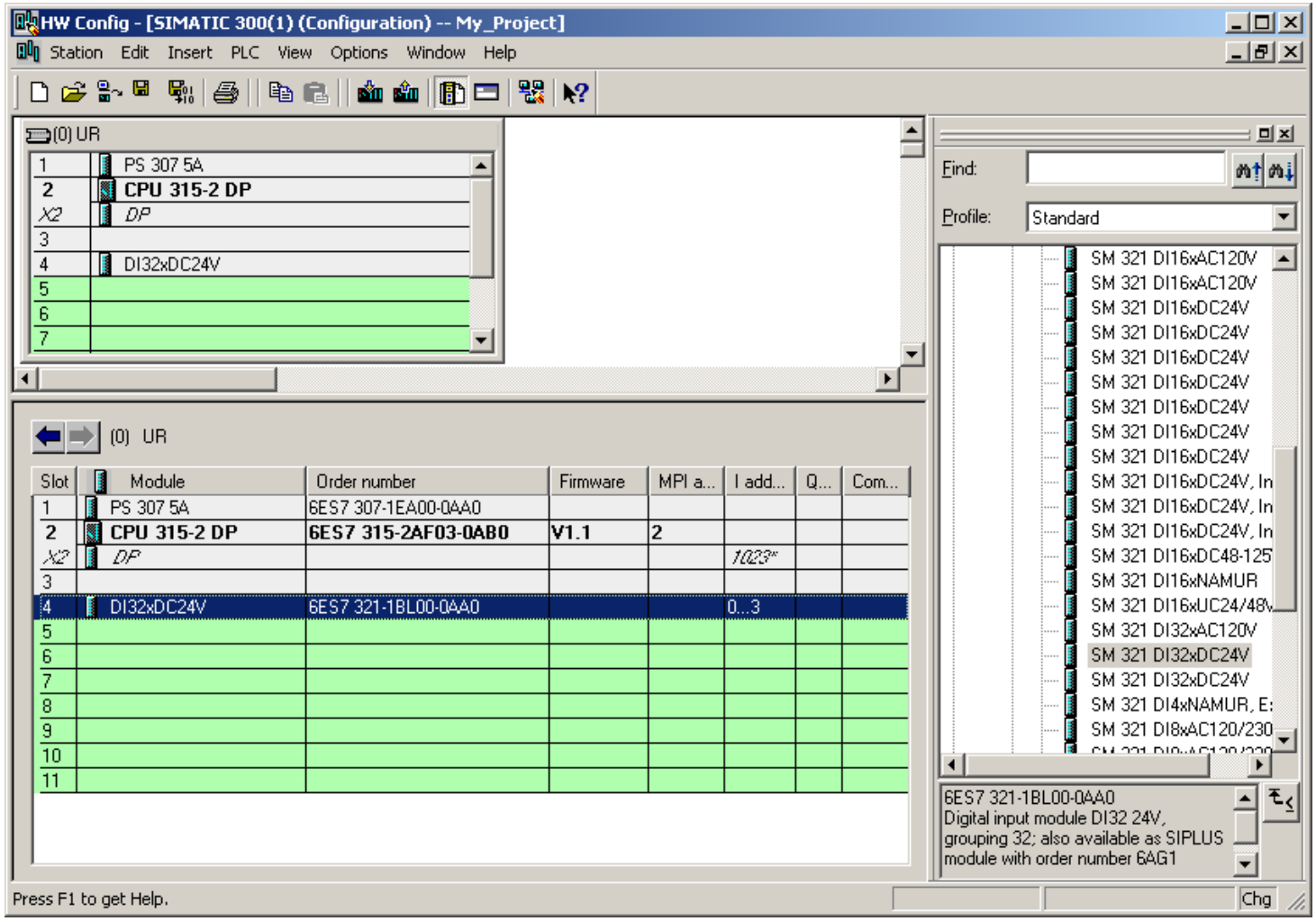
Inserting a Station



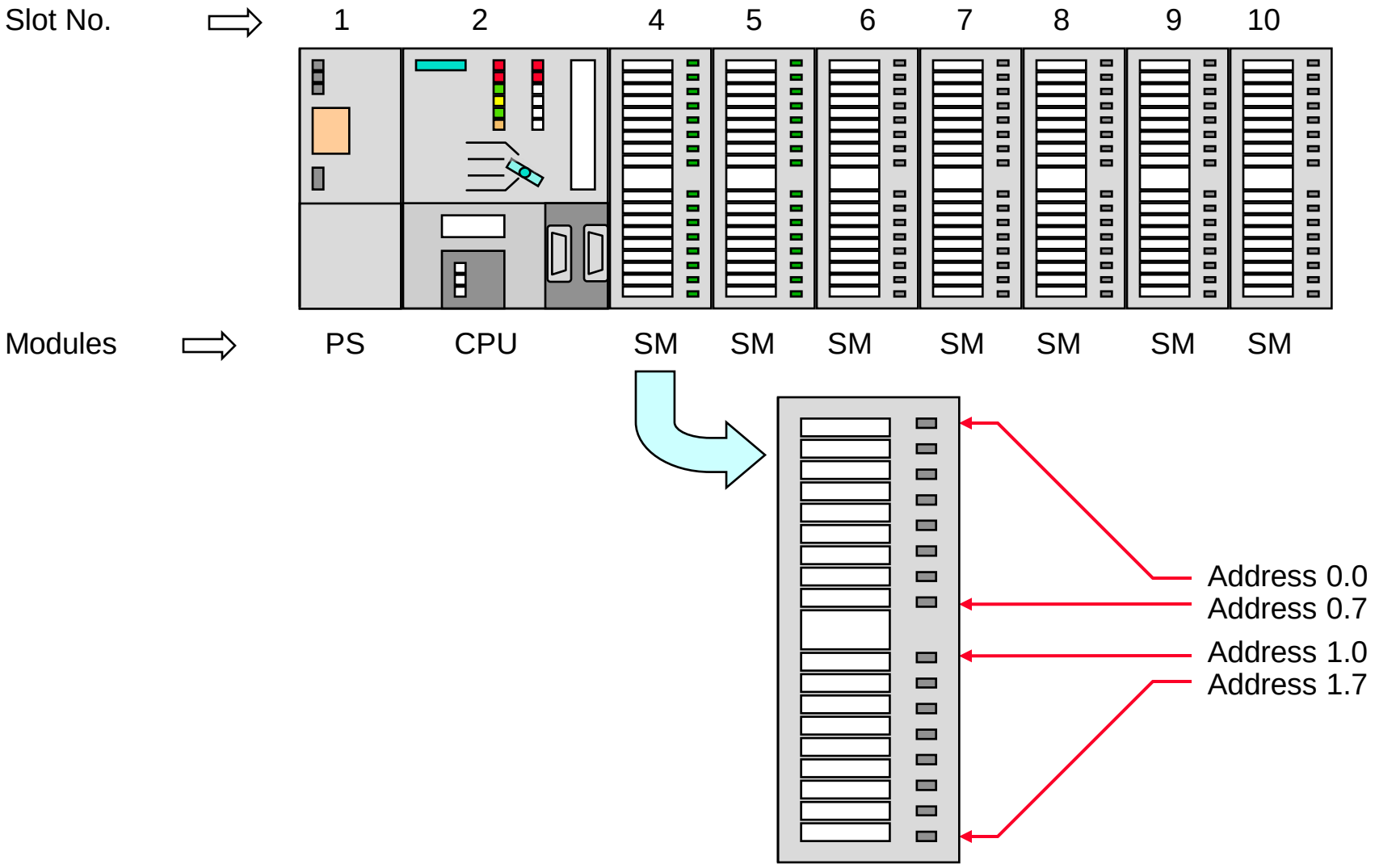
Starting the Hardware Configuration Editor



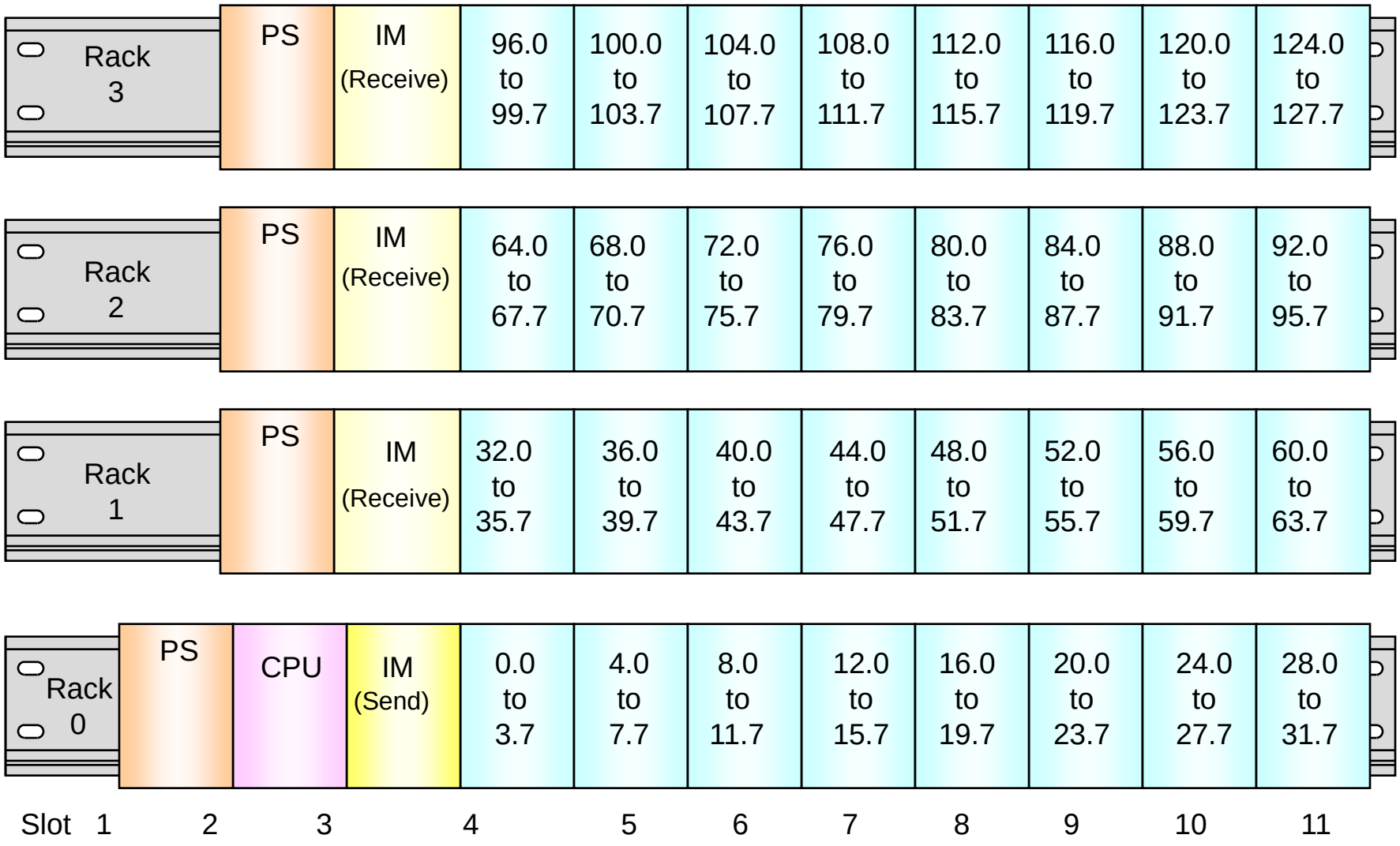
Generating a Hardware Setpoint Configuration



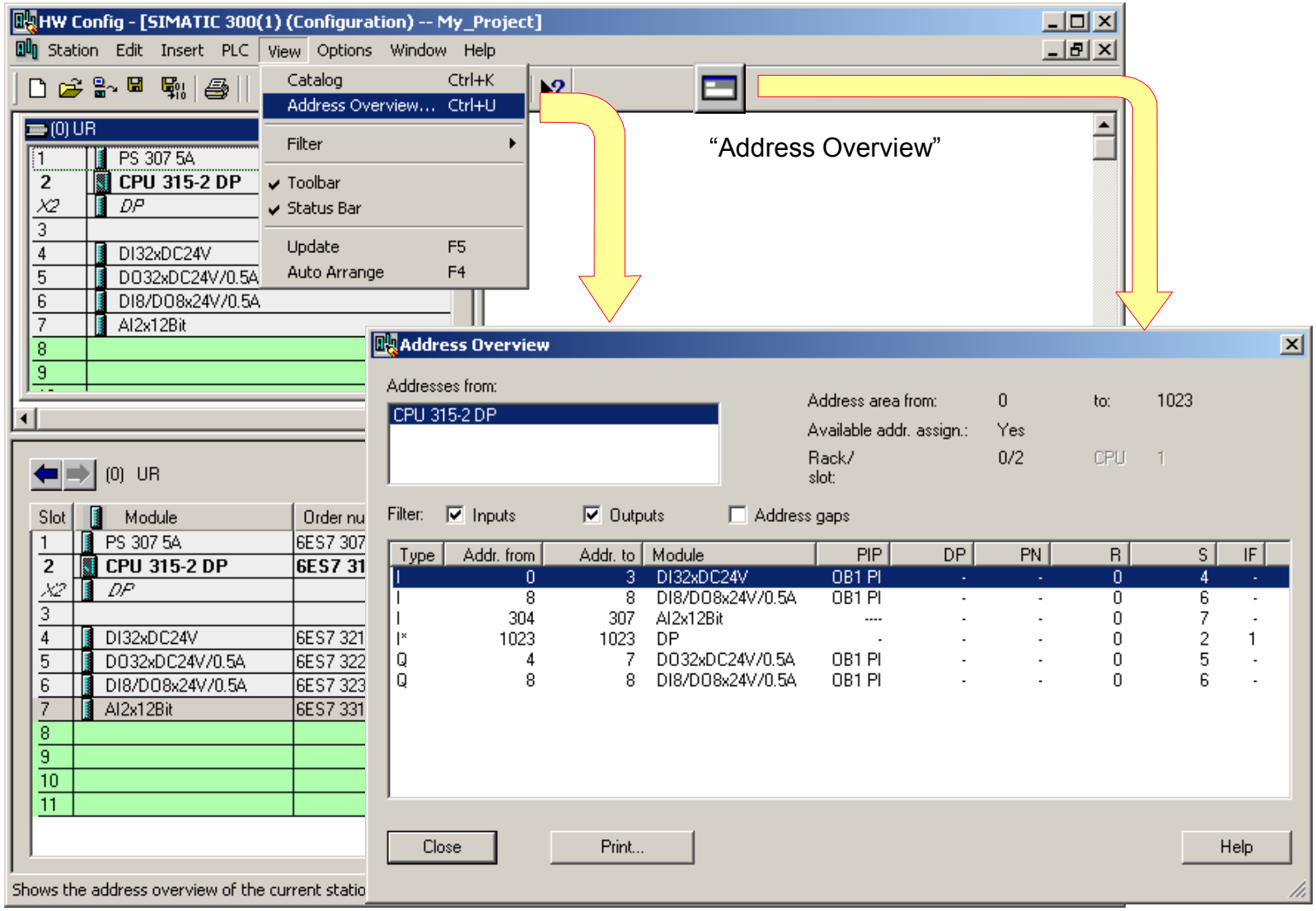
Addressing S7-300 Modules



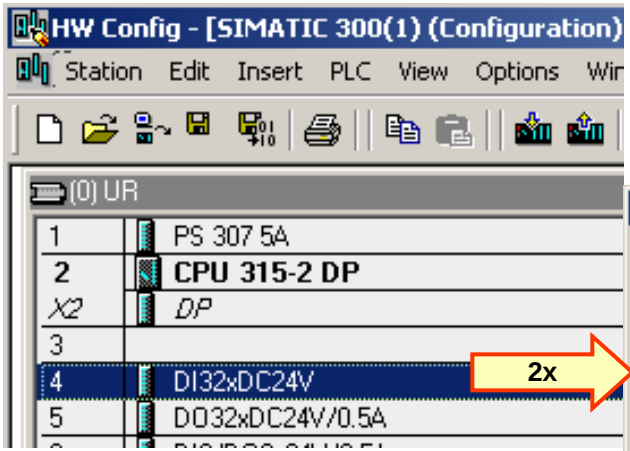
DI/DO Default Addressing in Multi-Tier Configurations

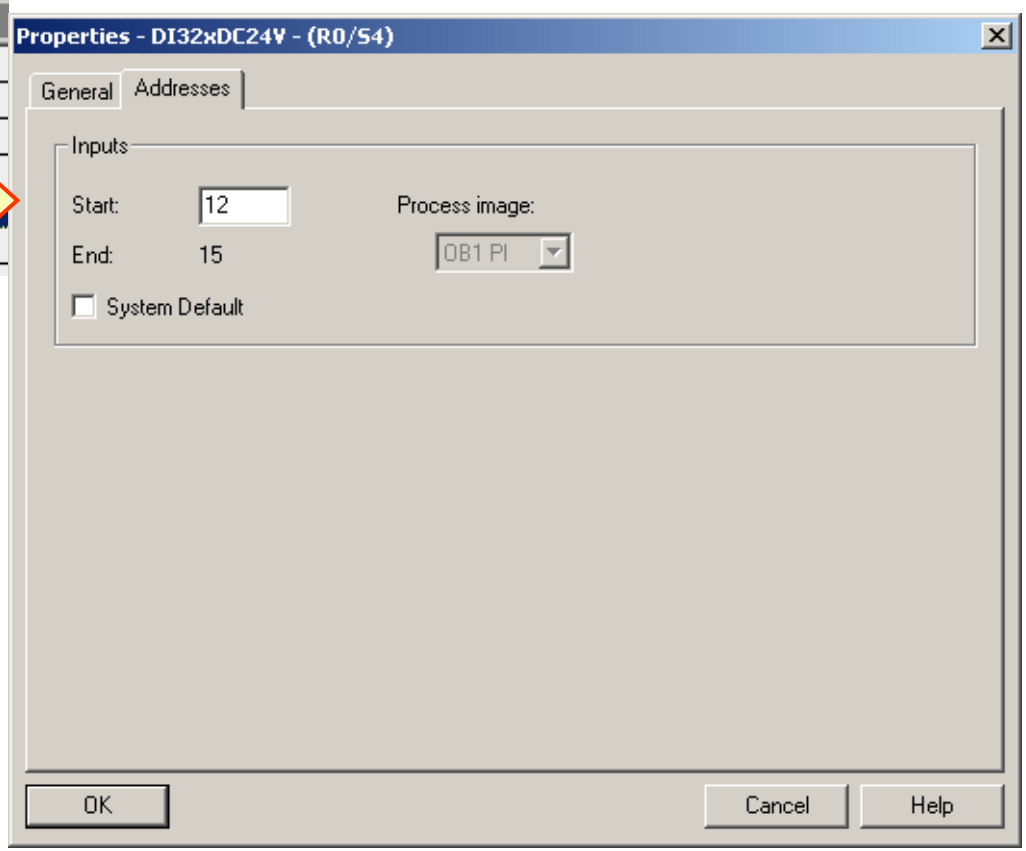


Module Address Overview

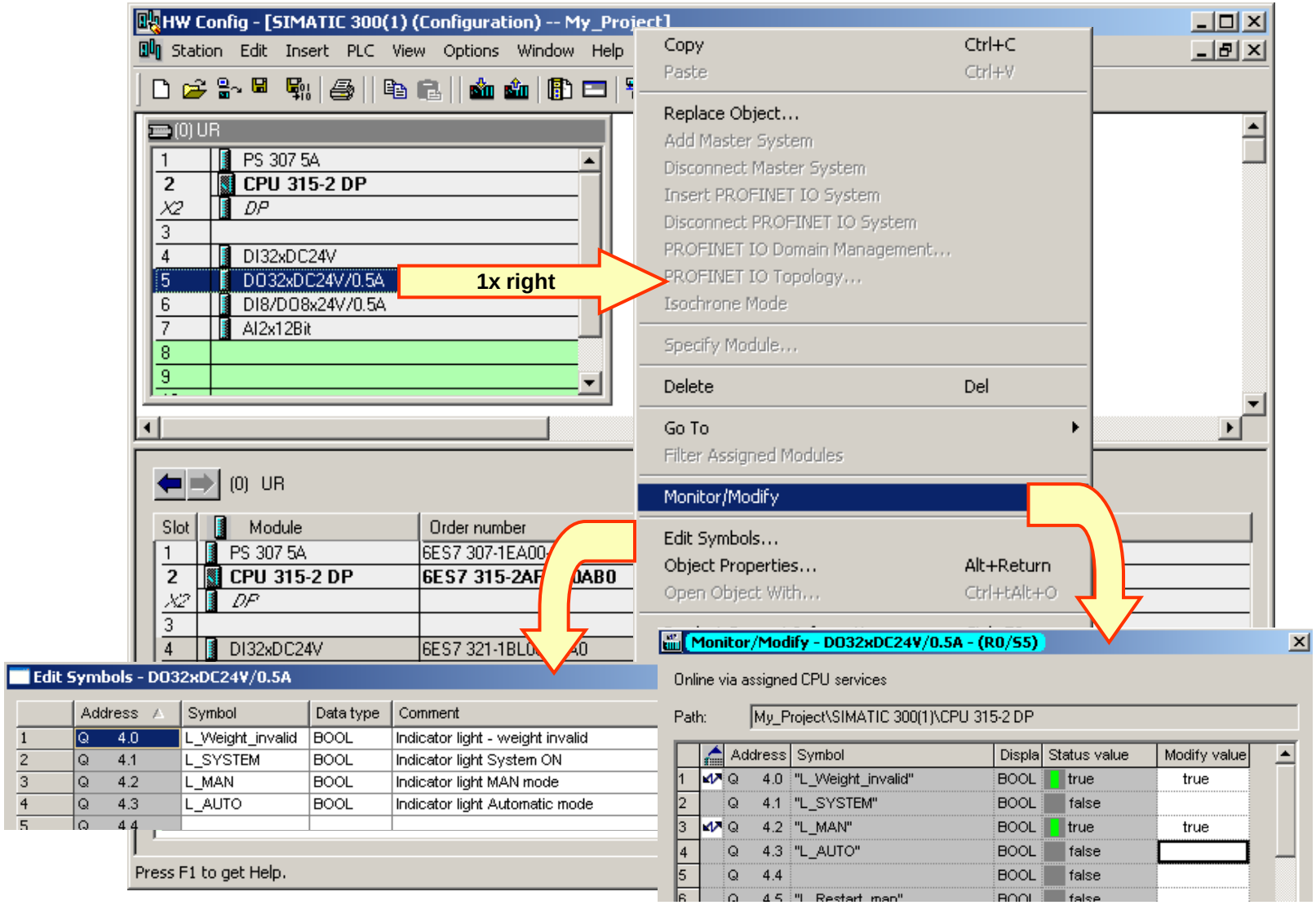


Variable Addressing

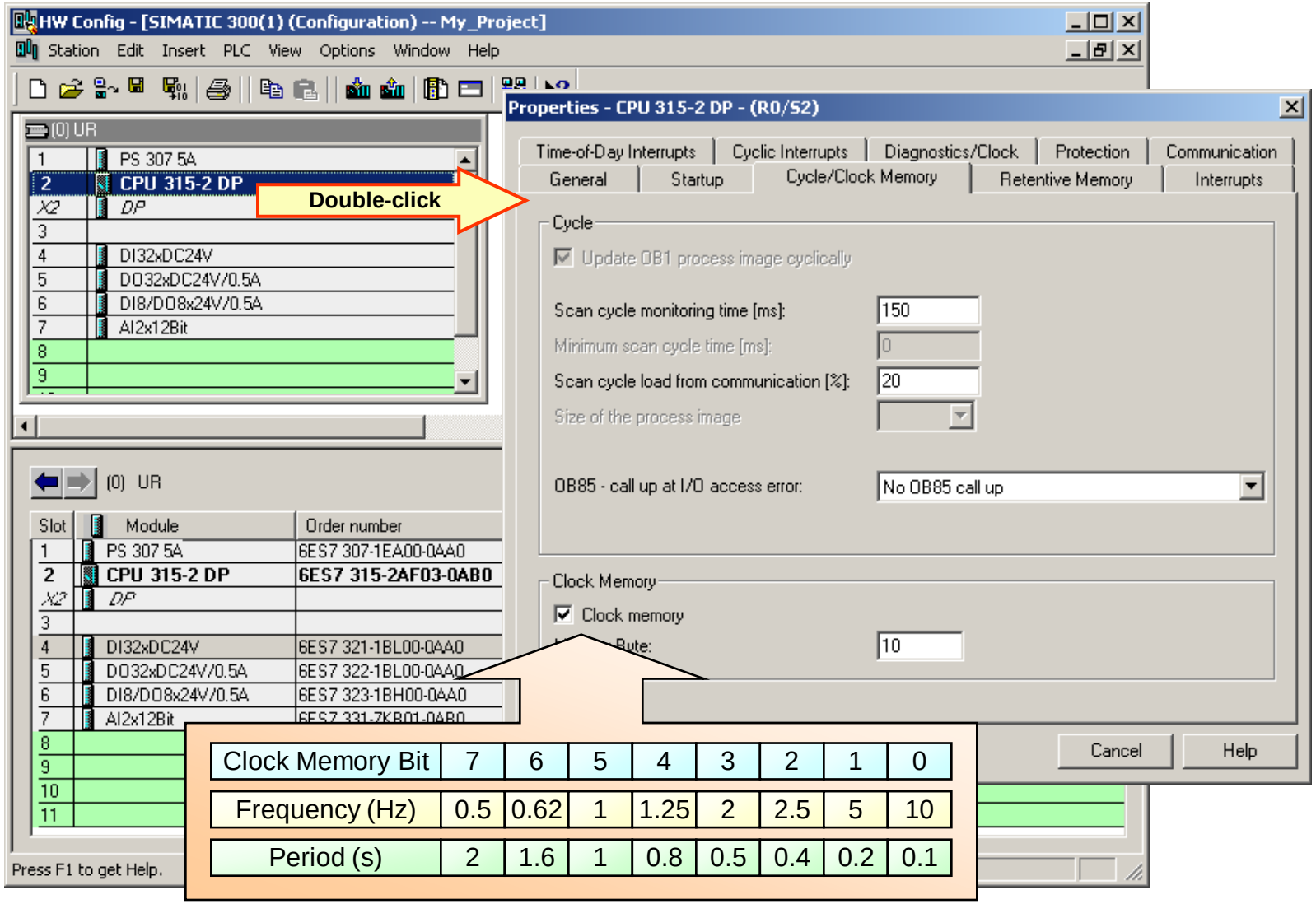




Edit Symbols, Monitor/Modify Variables

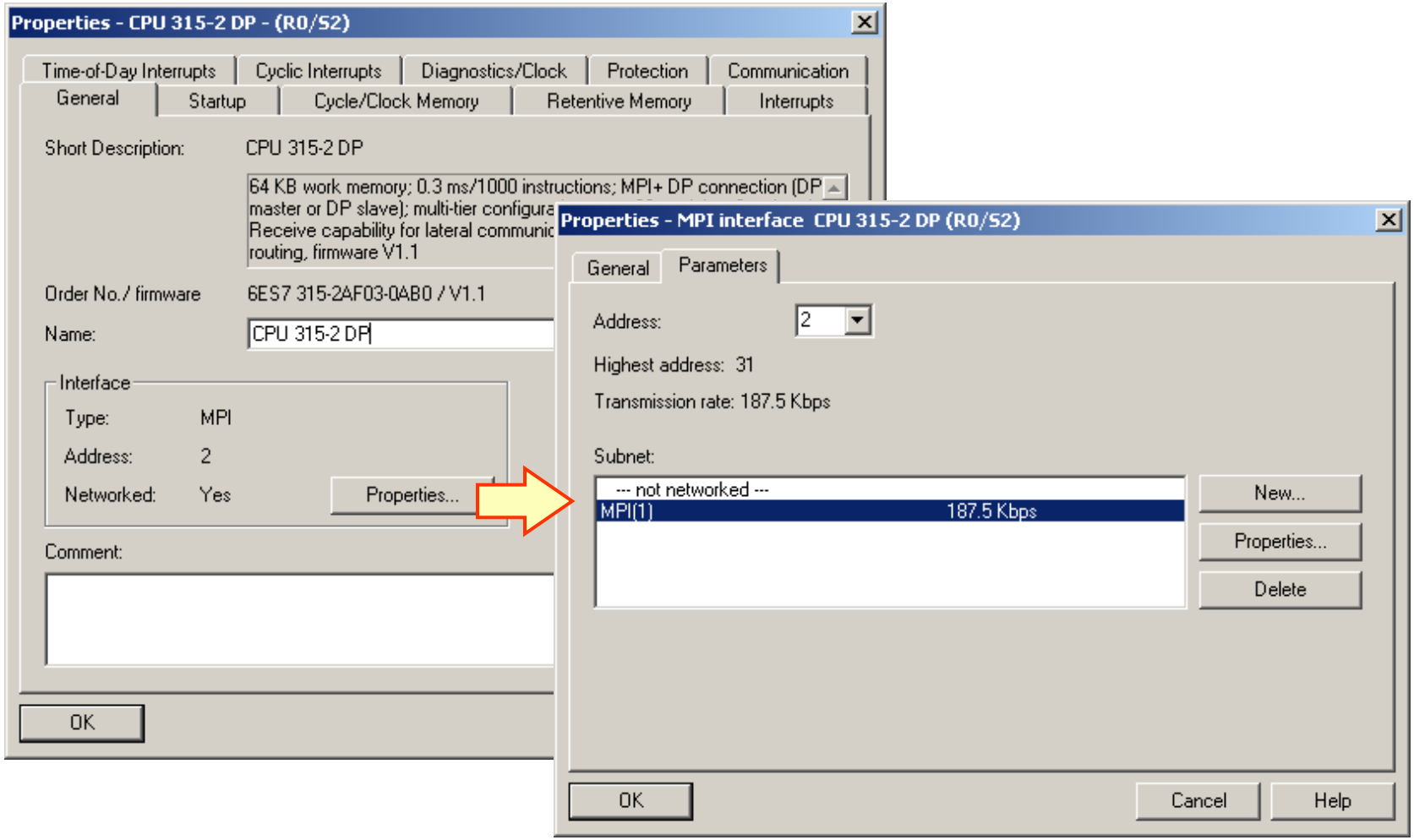


CPU Properties: Cycle/Clock Memory

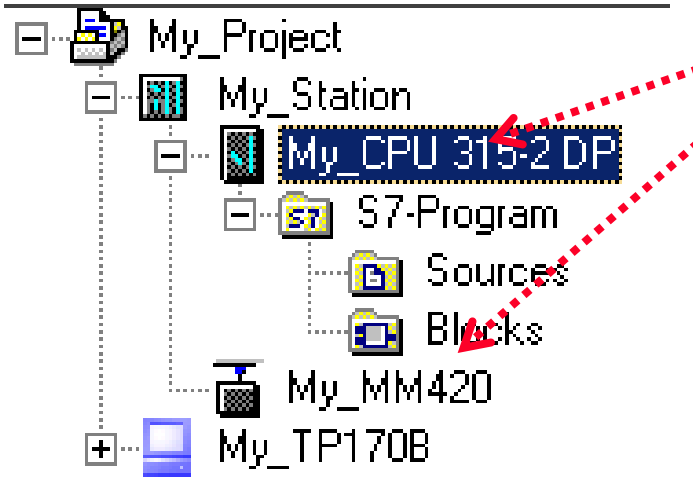


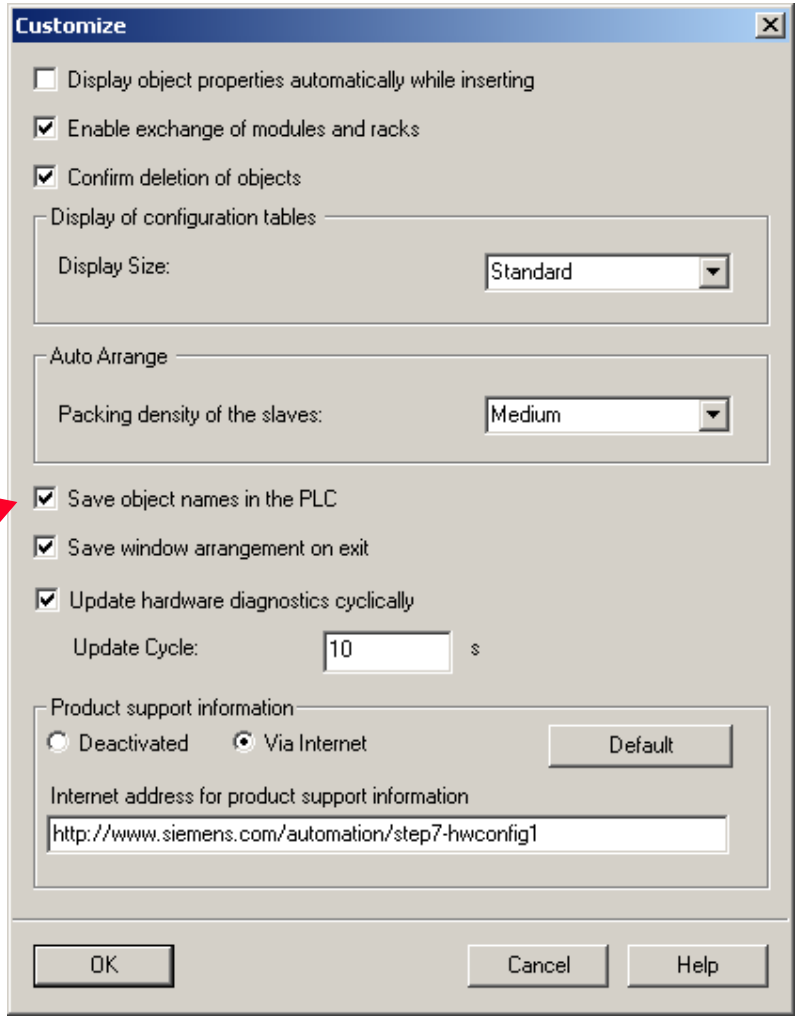
Clock Memory Bit	7	6	5	4	3	2	1	0
Frequency (Hz)	0.5	0.62	1	1.25	2	2.5	5	10
Period (s)	2	1.6	1	0.8	0.5	0.4	0.2	0.1

CPU Properties: General (amongst others MPI Address)

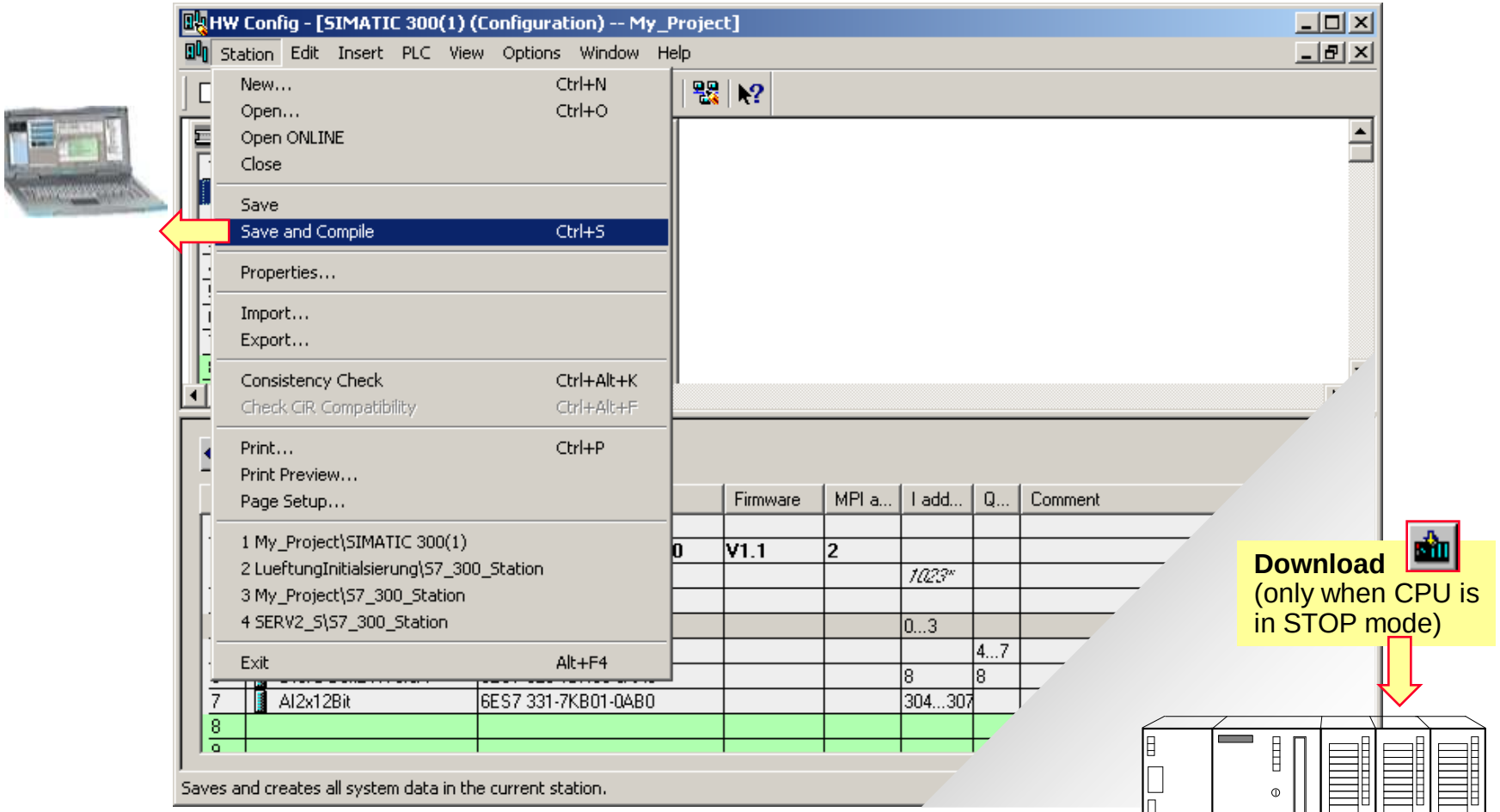


Saving Station Names in the CPU

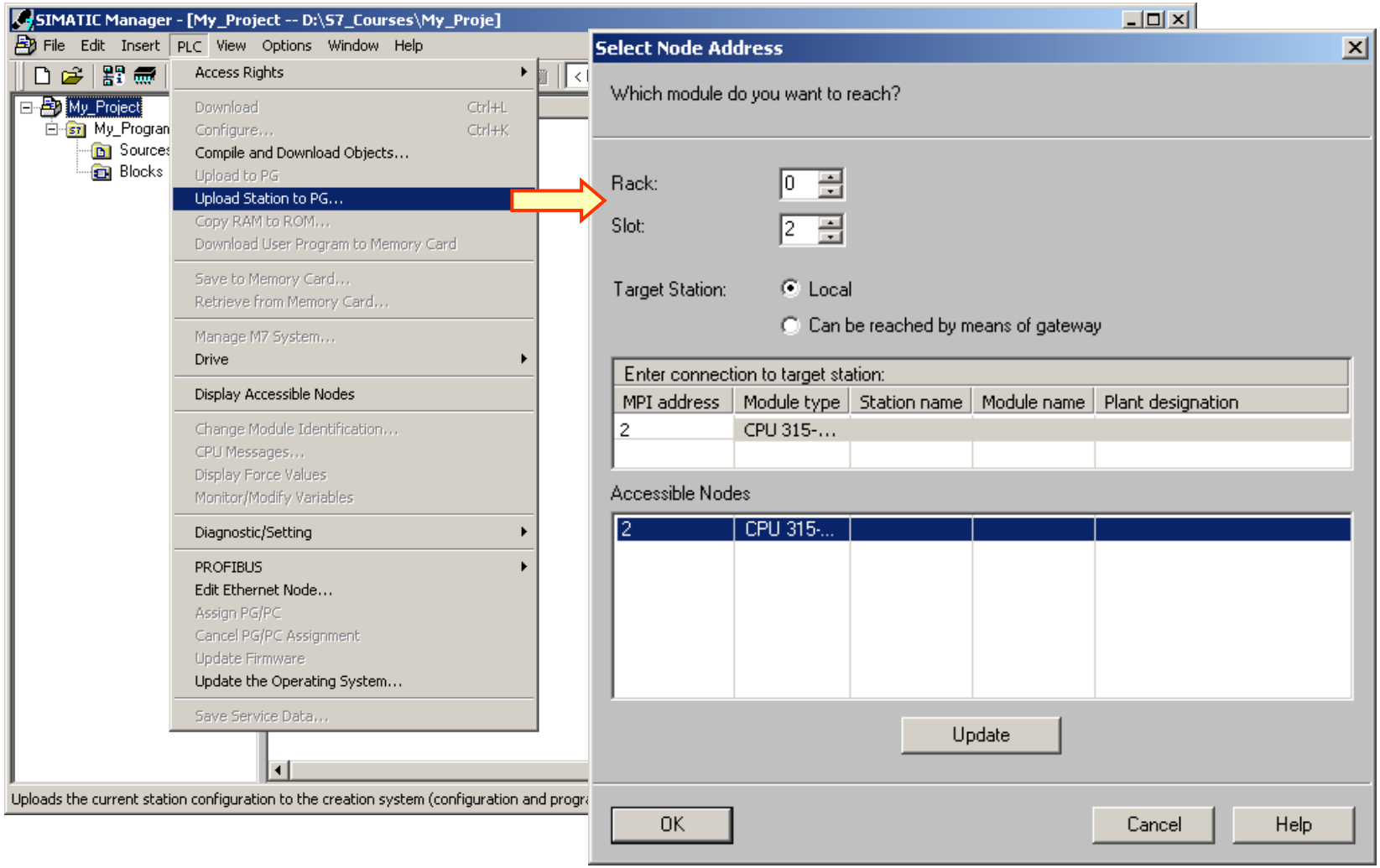




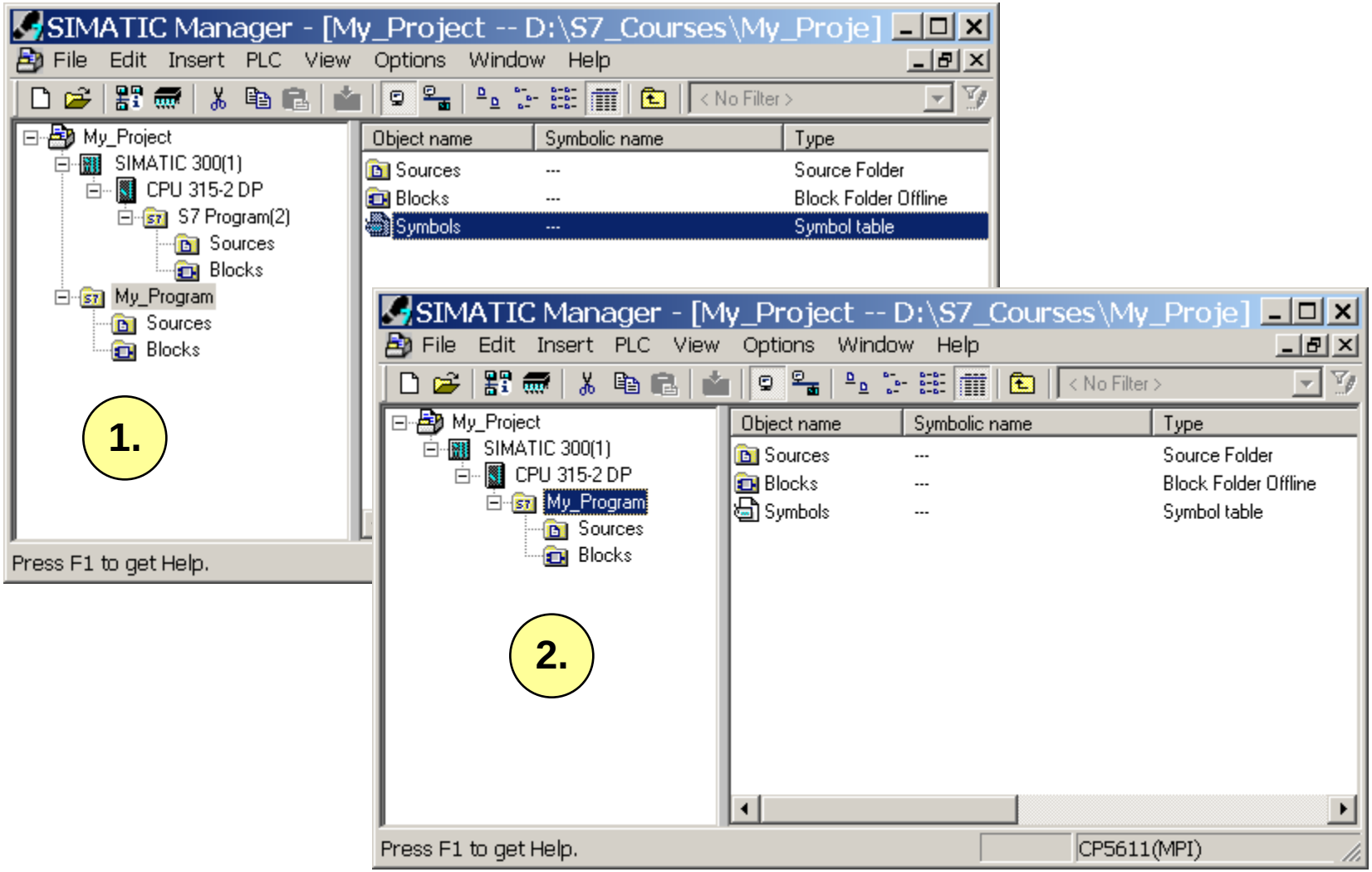
Saving the Setpoint Configuration and Downloading it in Modules



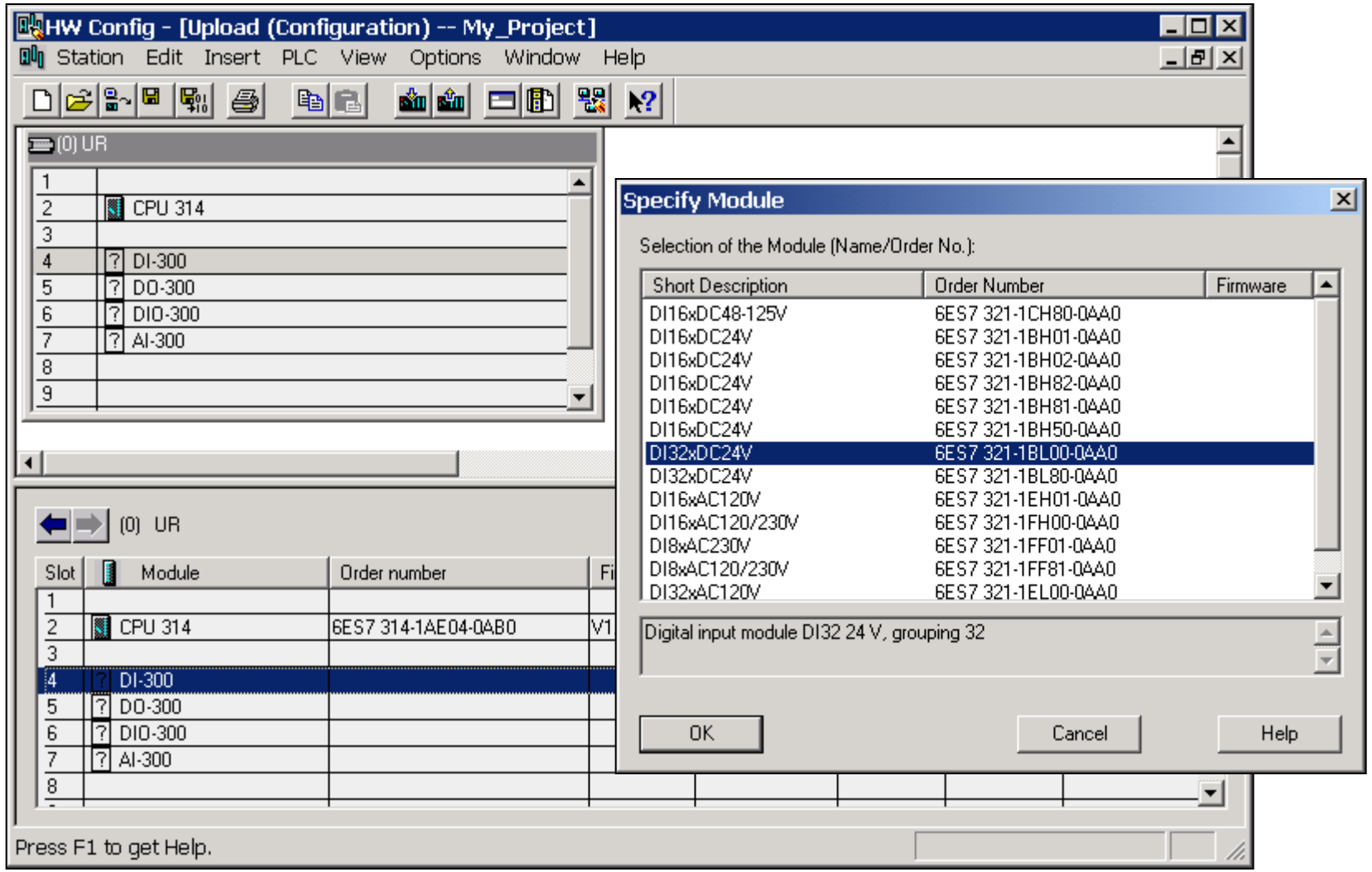
Uploading the Actual Configuration to the PG



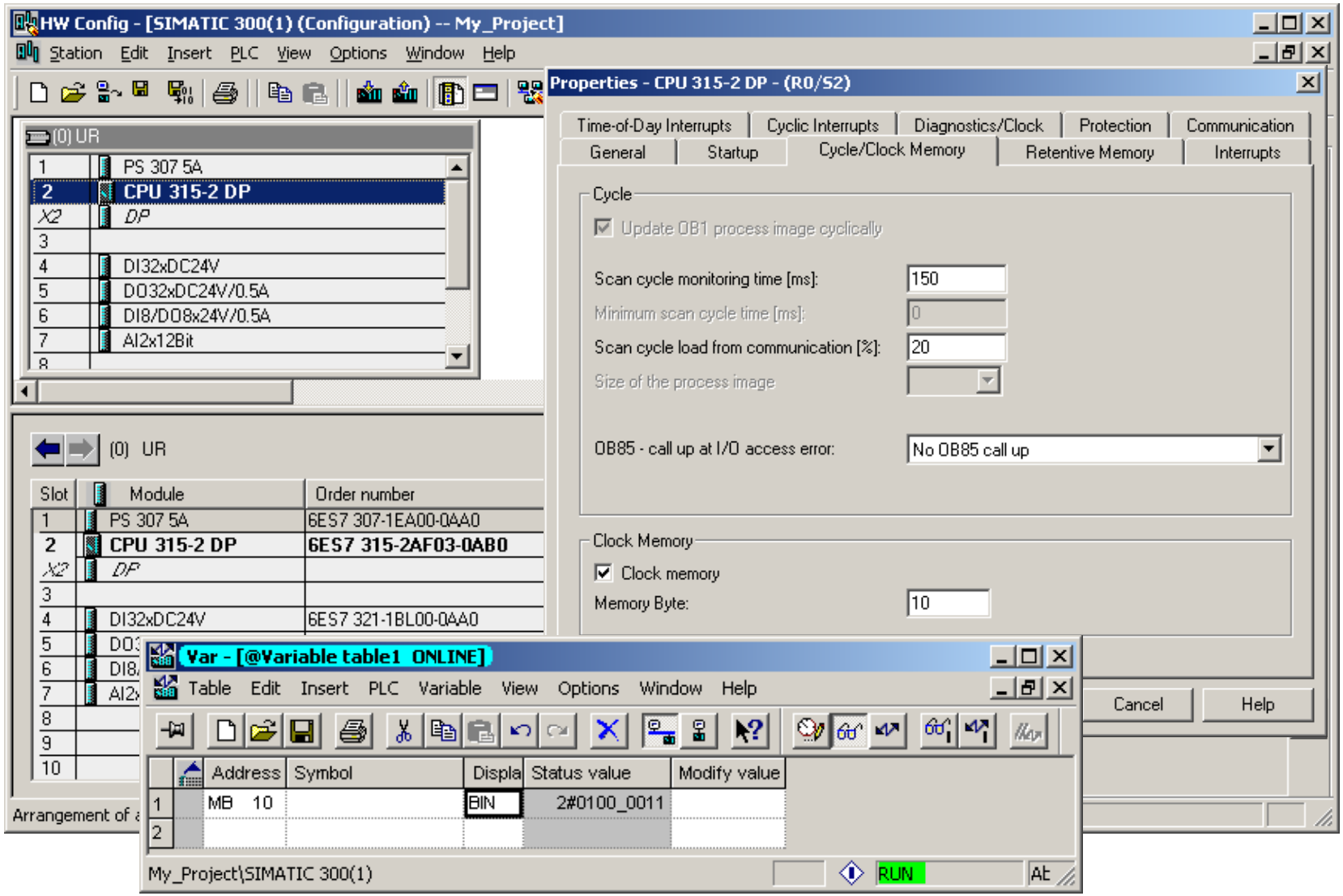
Exercise 1: Creating a HW Station



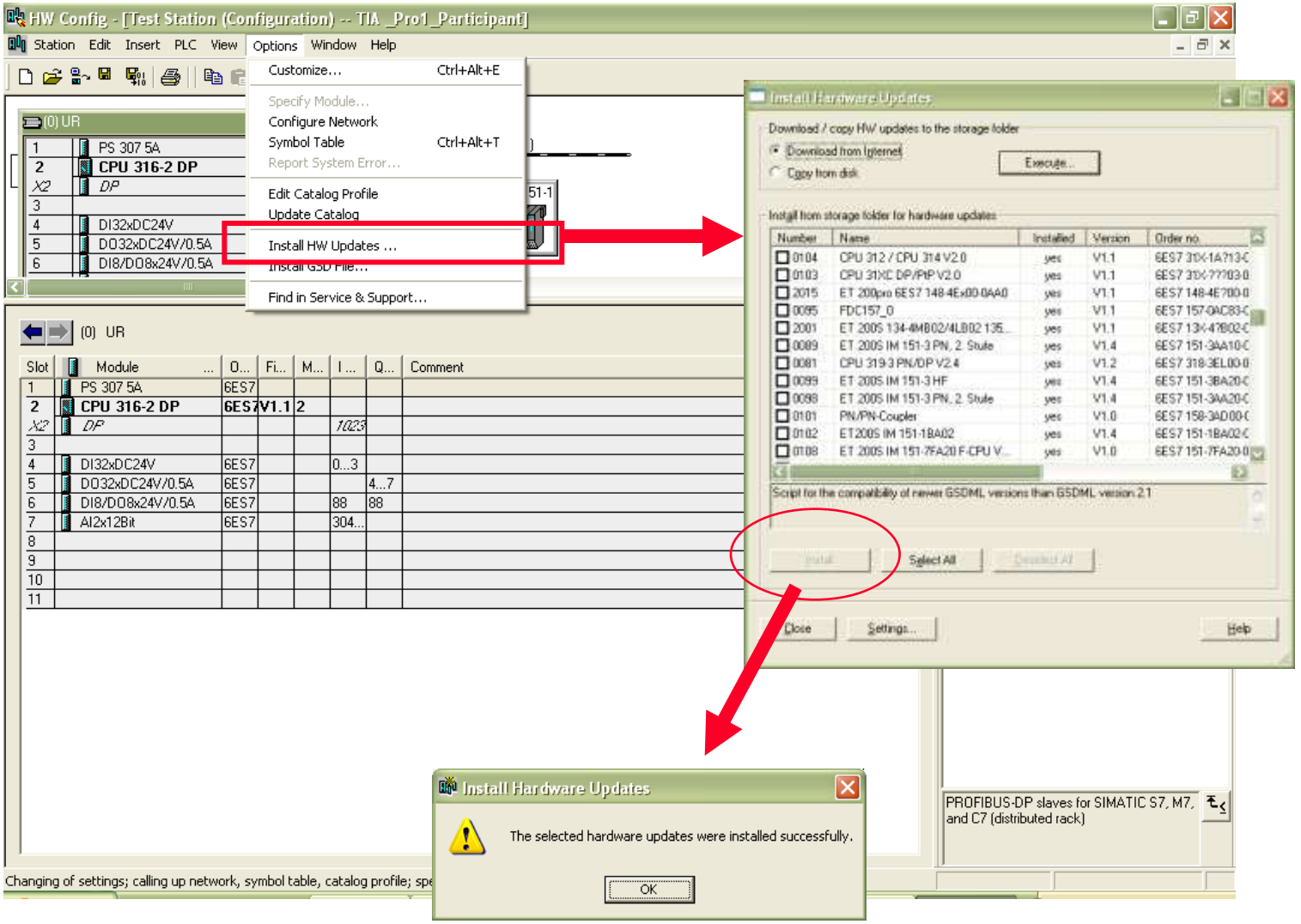
Exercise 2: Adapting the Actual Configuration



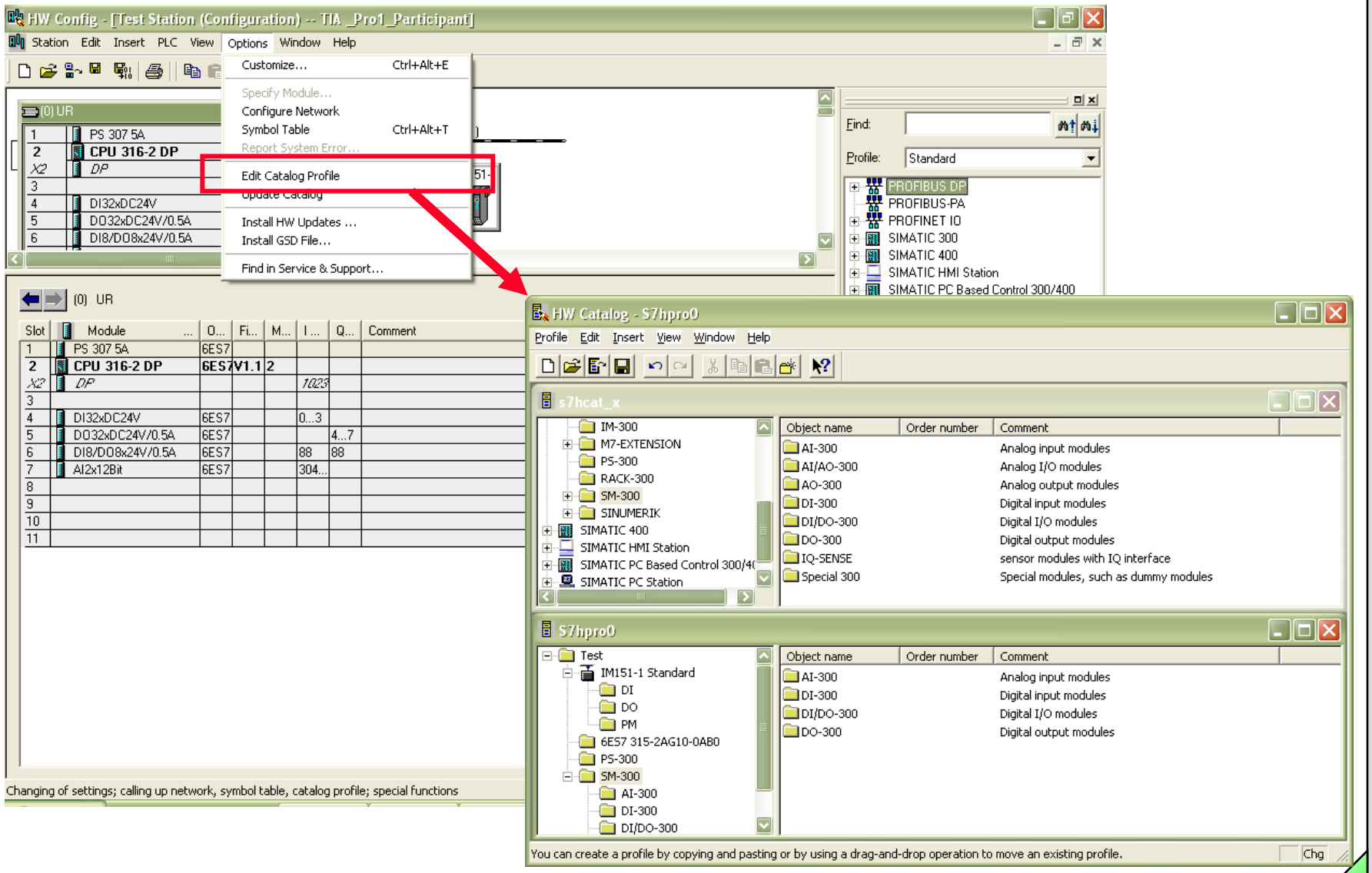
Exercise 3: Assigning Parameters to the CPU Clock Memory and Testing It



Hardware Configuration Editor Catalog Update

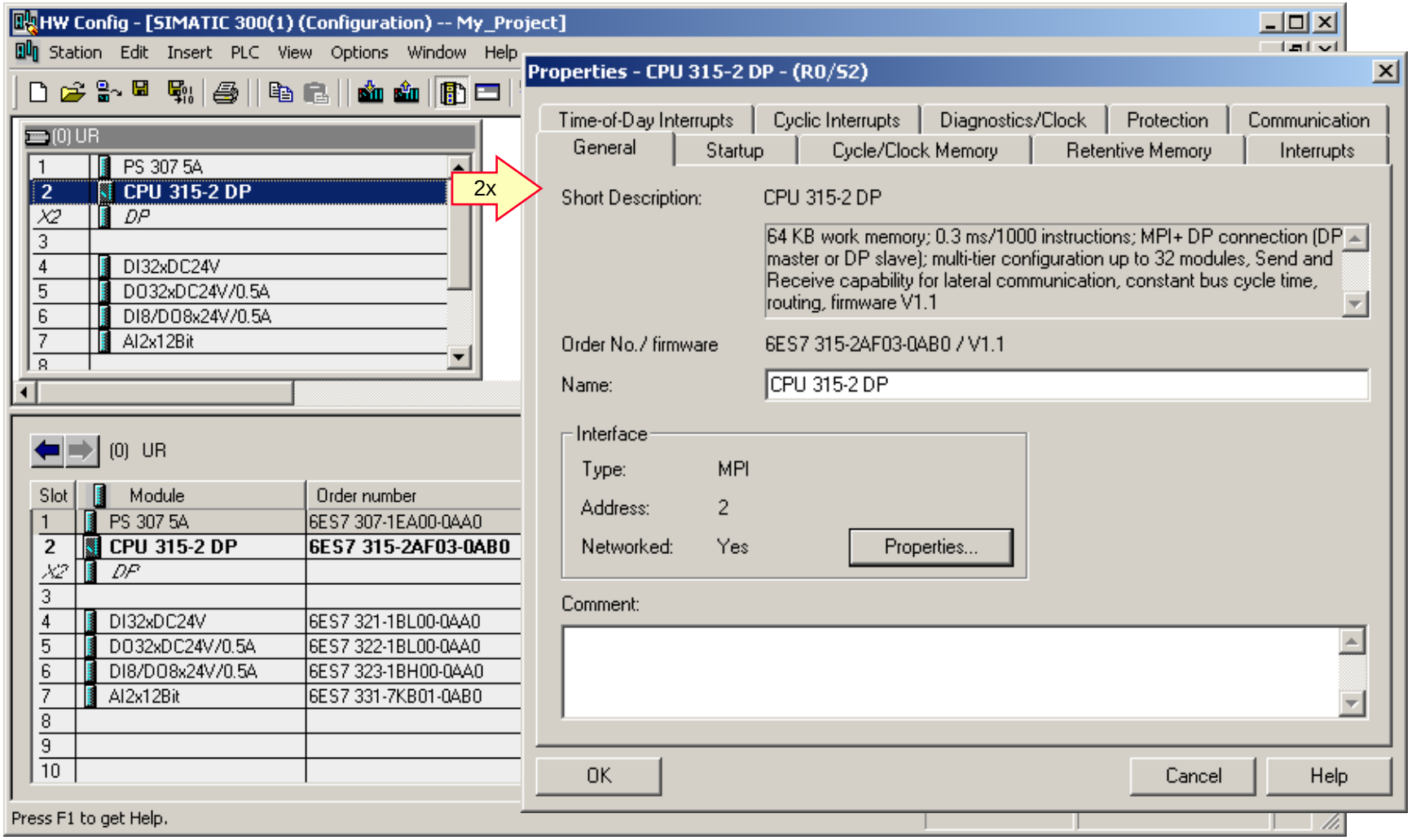


Edit Hardware Profile

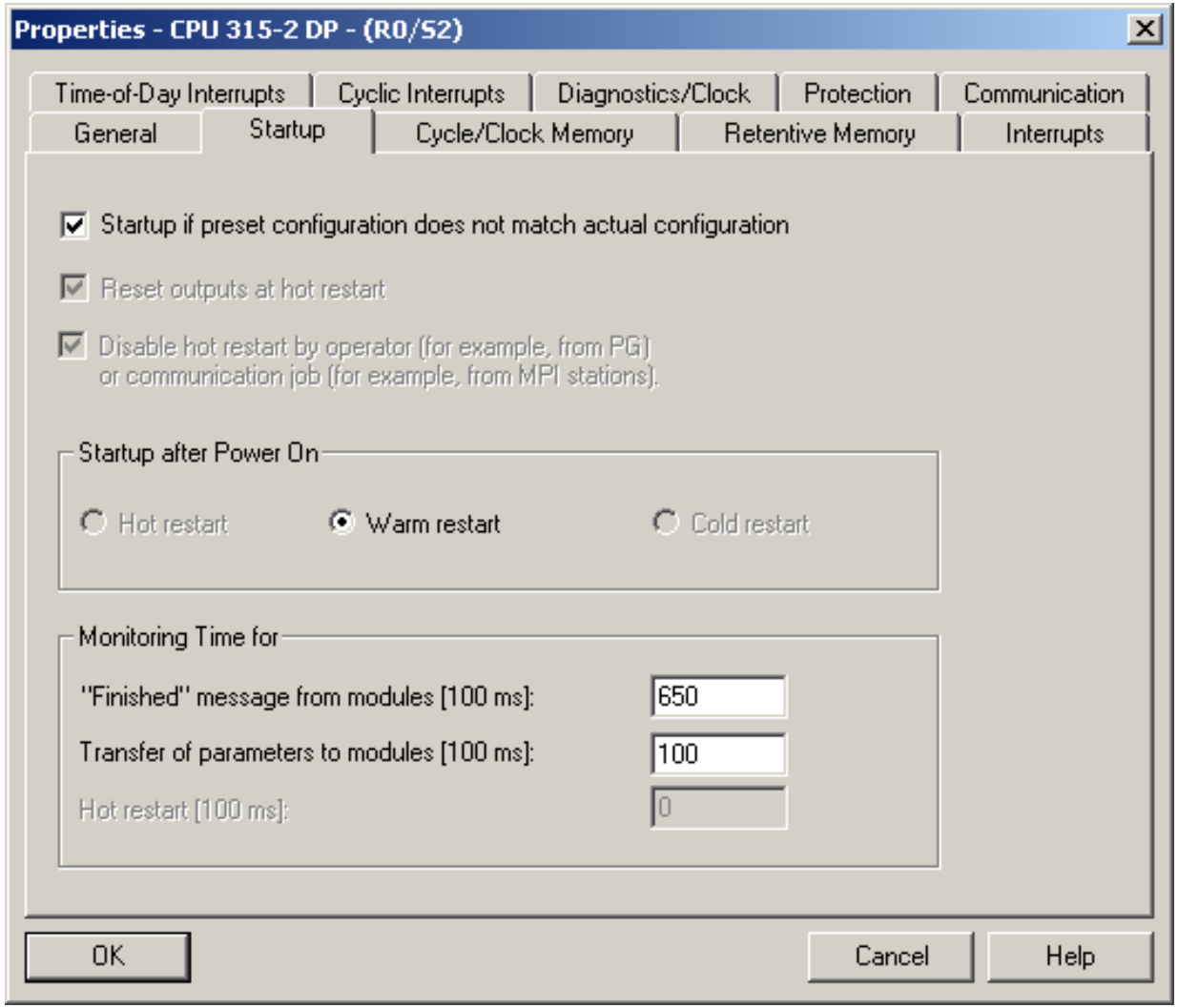


If You Want to Know More

CPU Properties

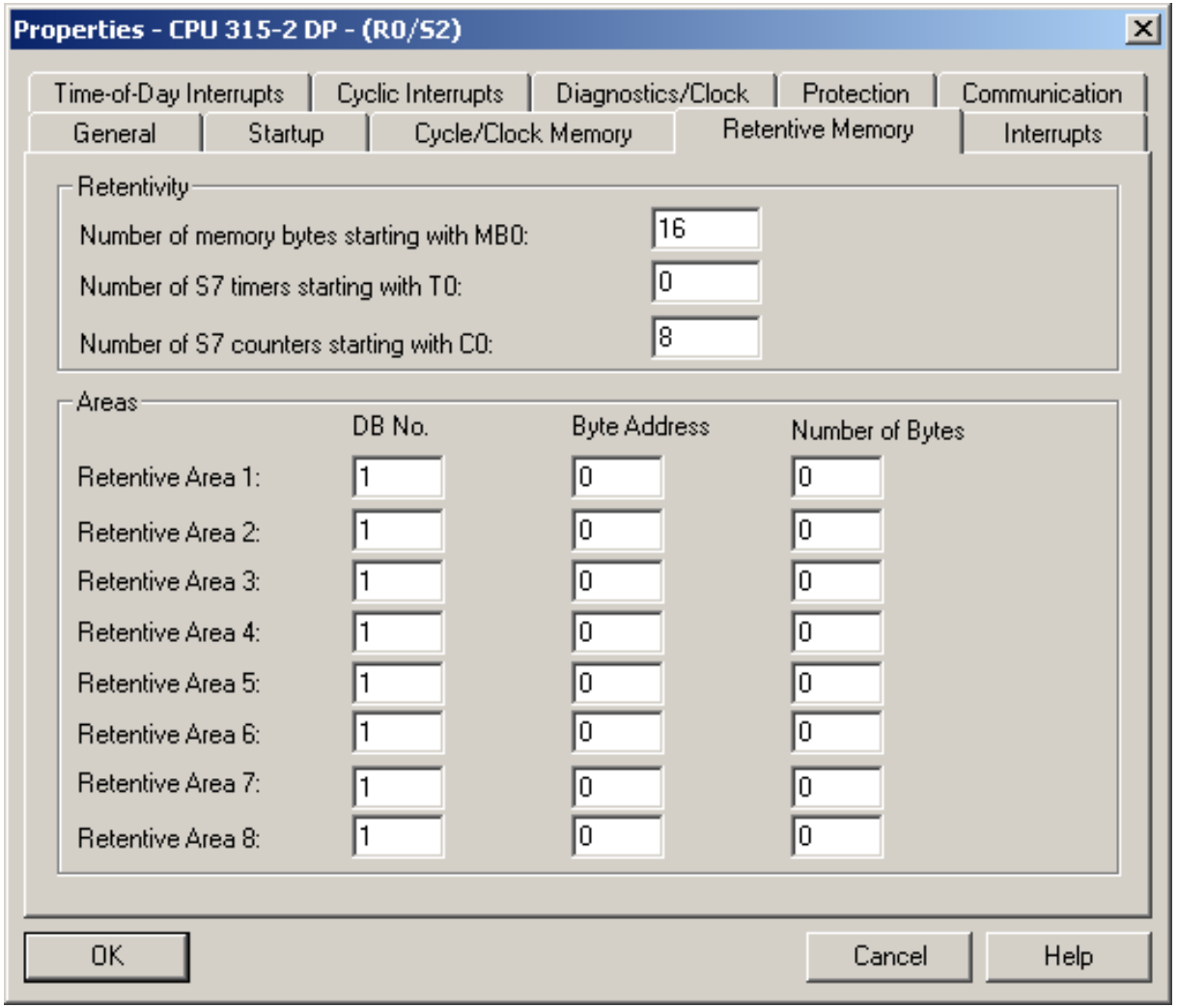


CPU Properties: Startup

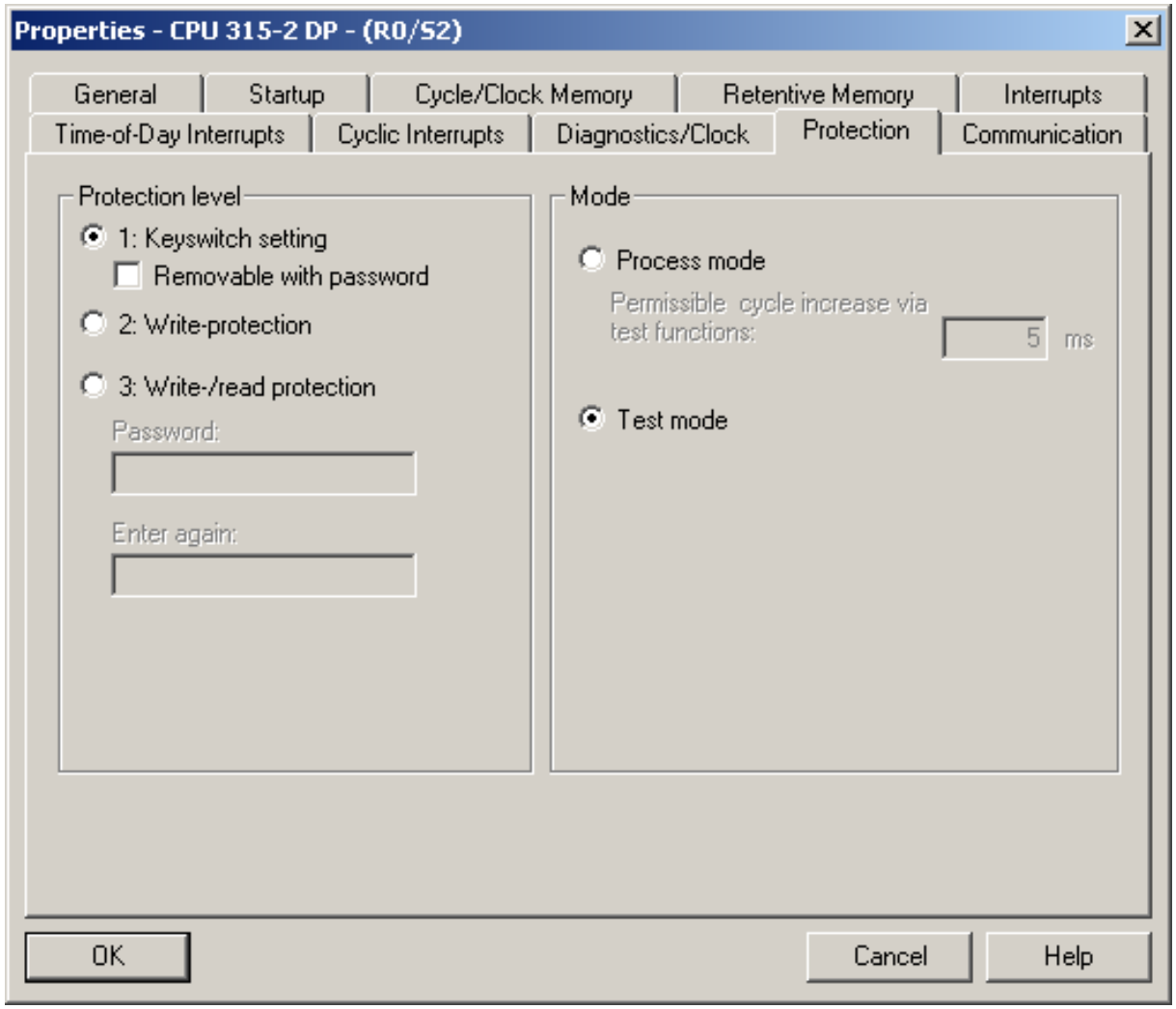


CPU Properties: Retentive Memory

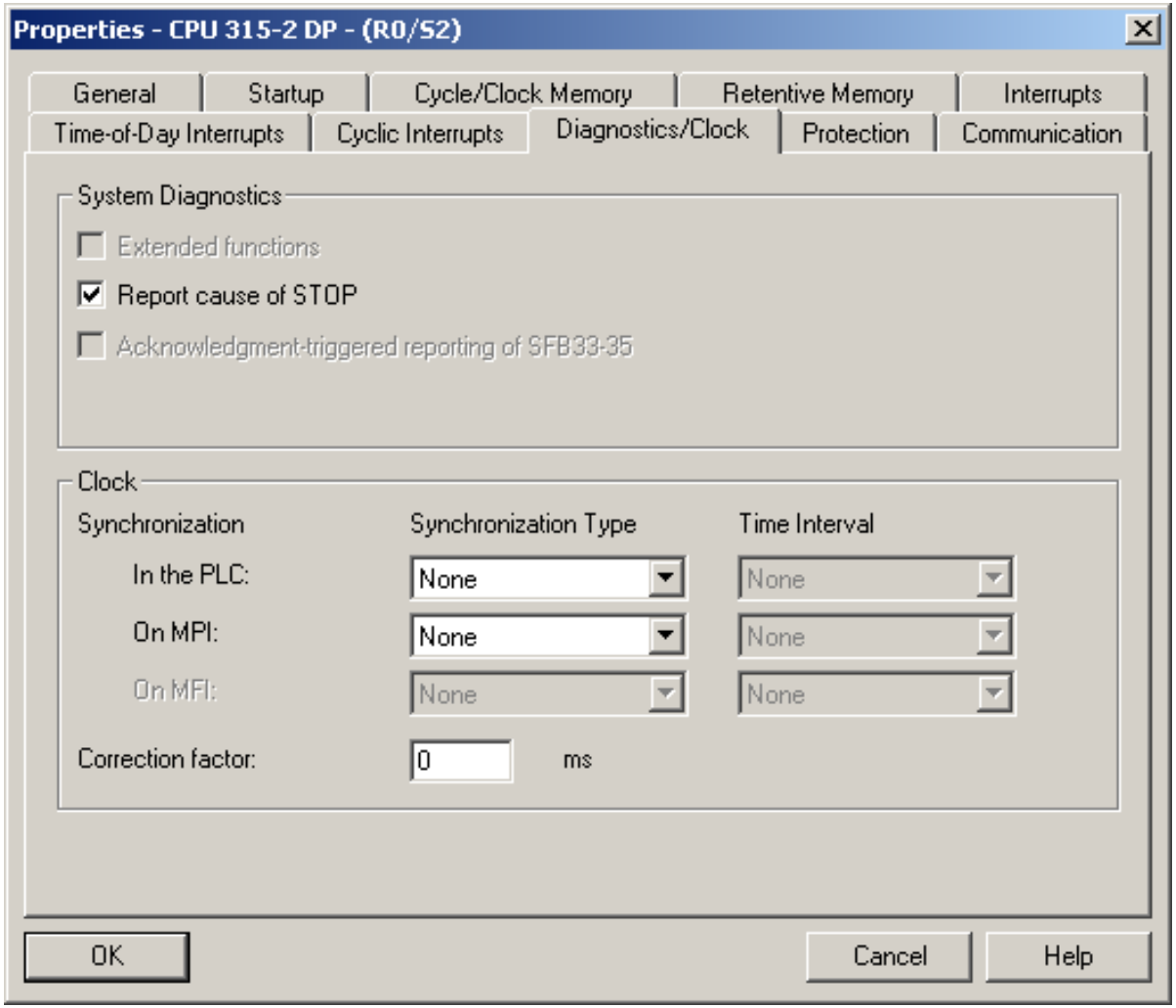
Relevant only for CPUs until Oct. 2002 that have no backup battery



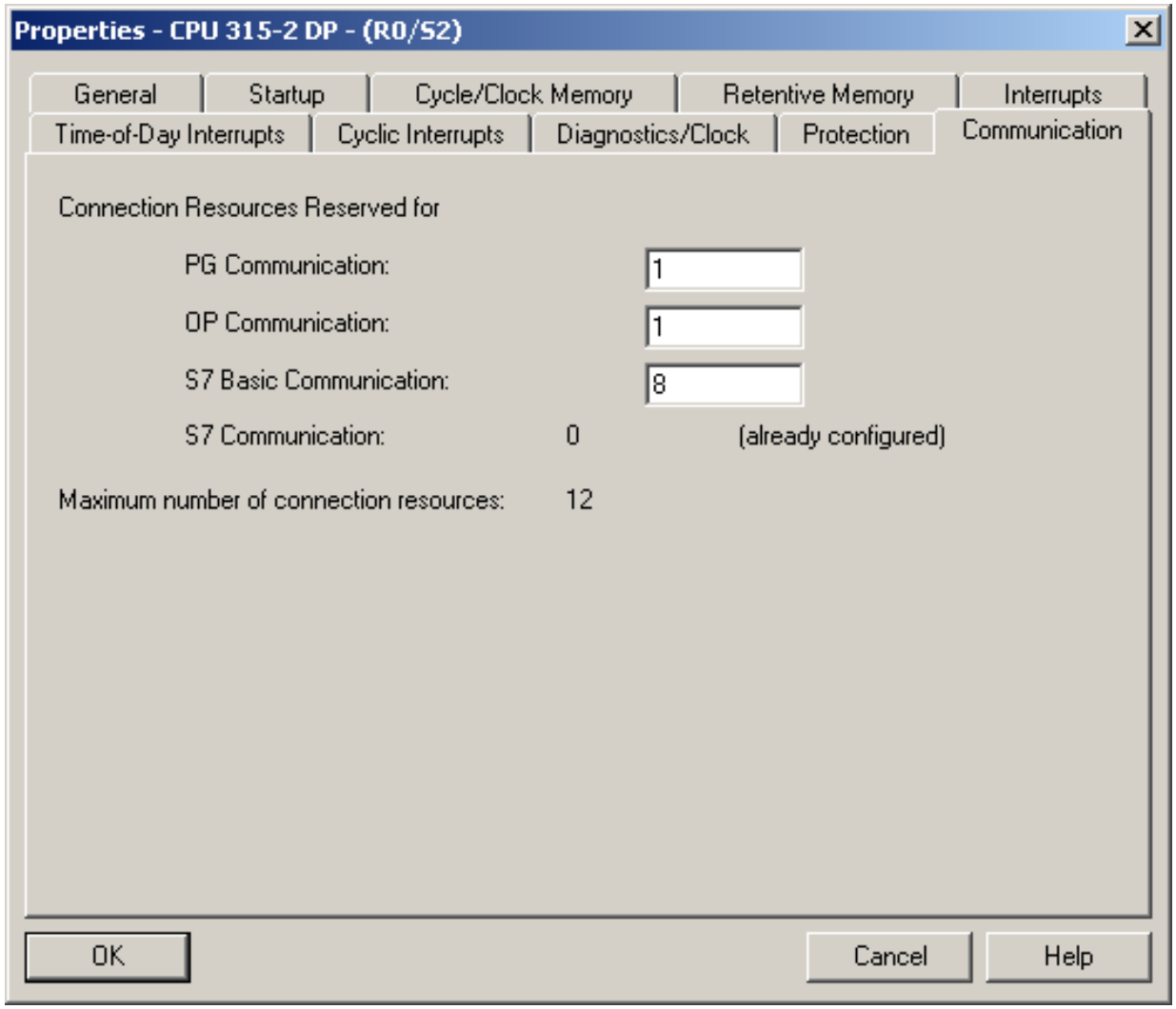
CPU Properties: Protection



CPU Properties: Diagnostics/Clock



CPU Properties: Communication

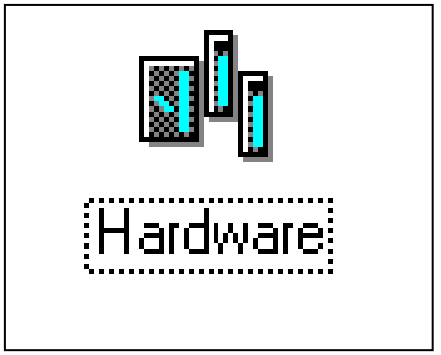


Further Processing of Hardware Stations in the CA01 Electronic Catalog

CA01 Electronic Catalog

Export function

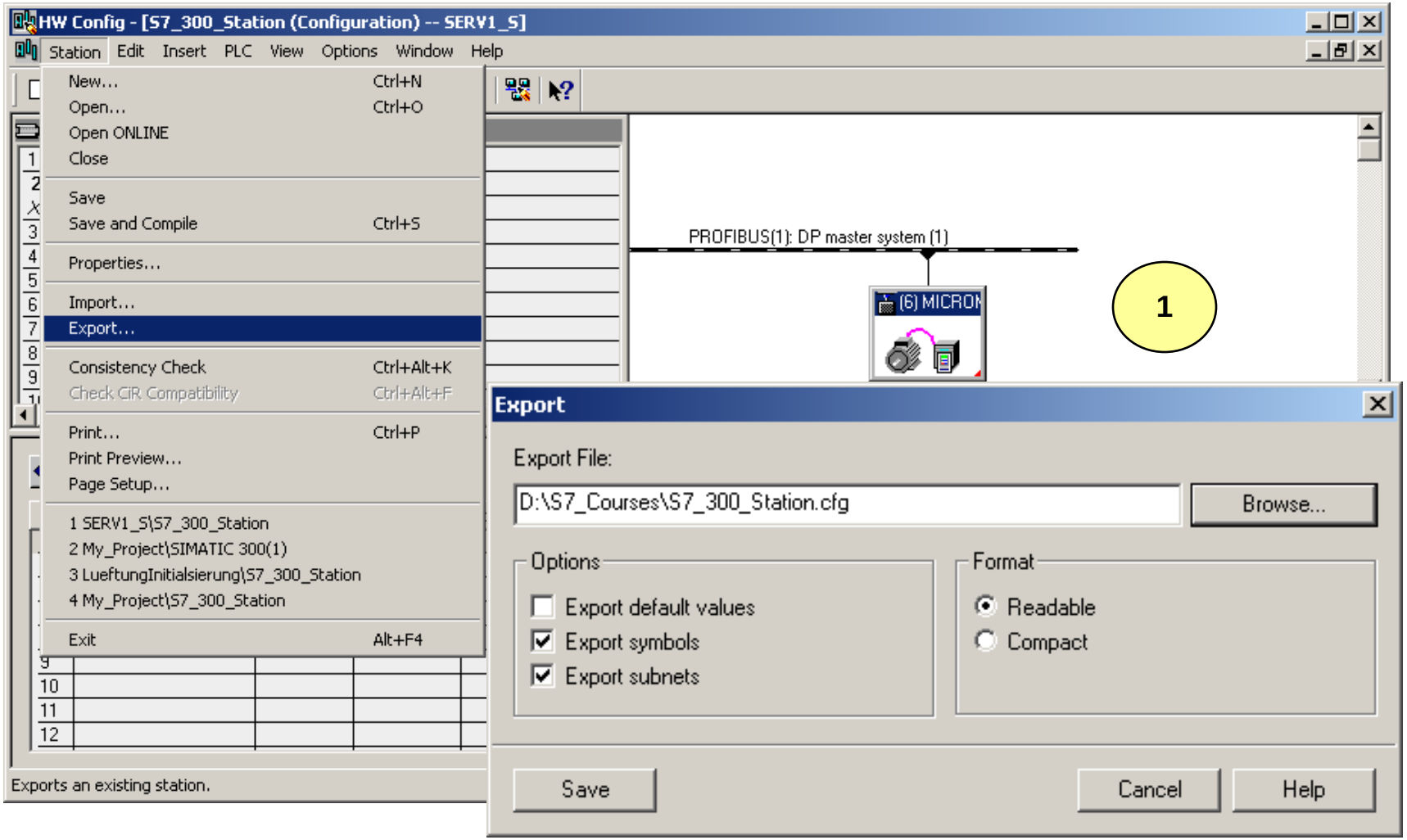
Import function



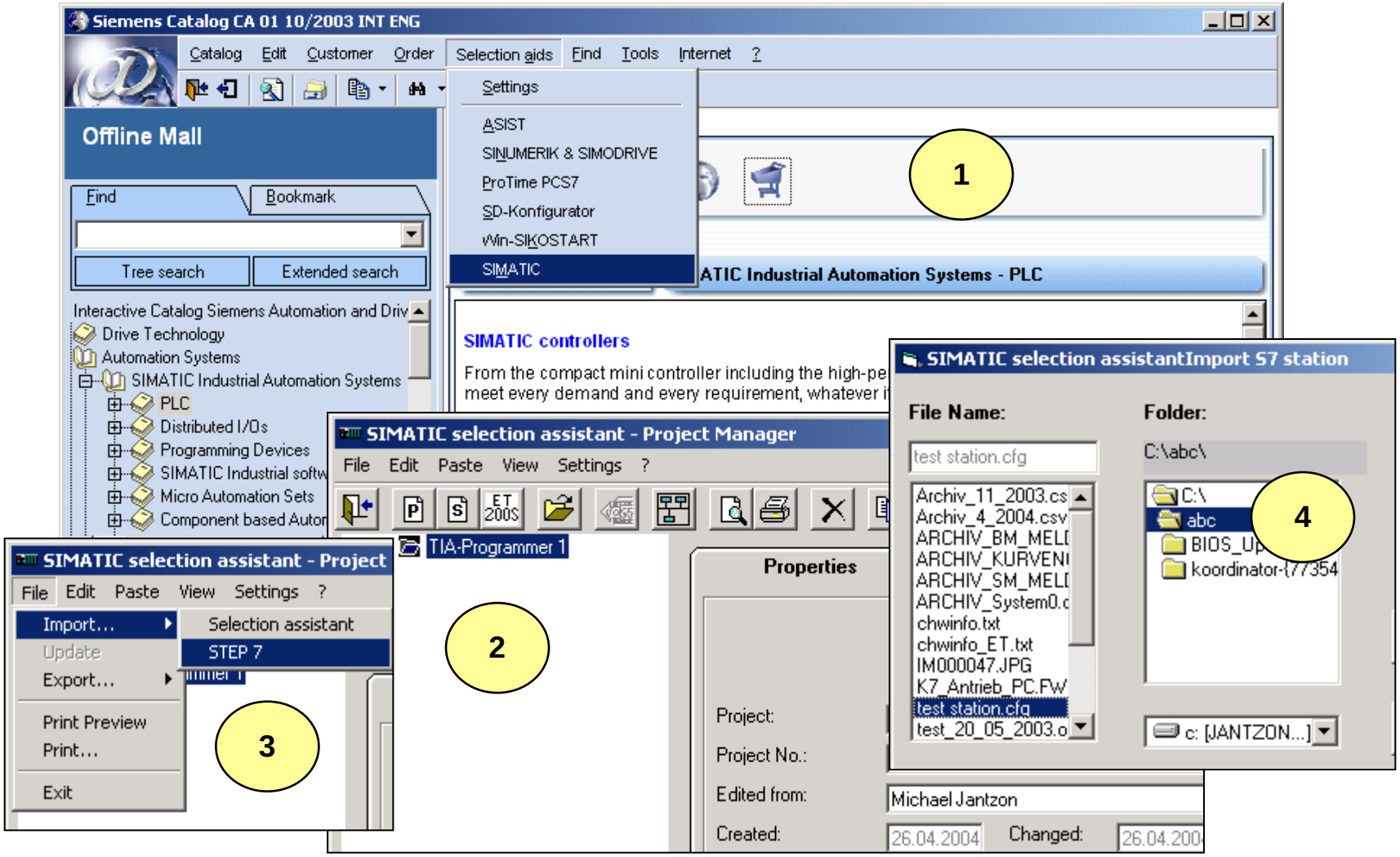
S7 Hardware-Station

Order lists
E-mail dispatch
Further processing with text editors

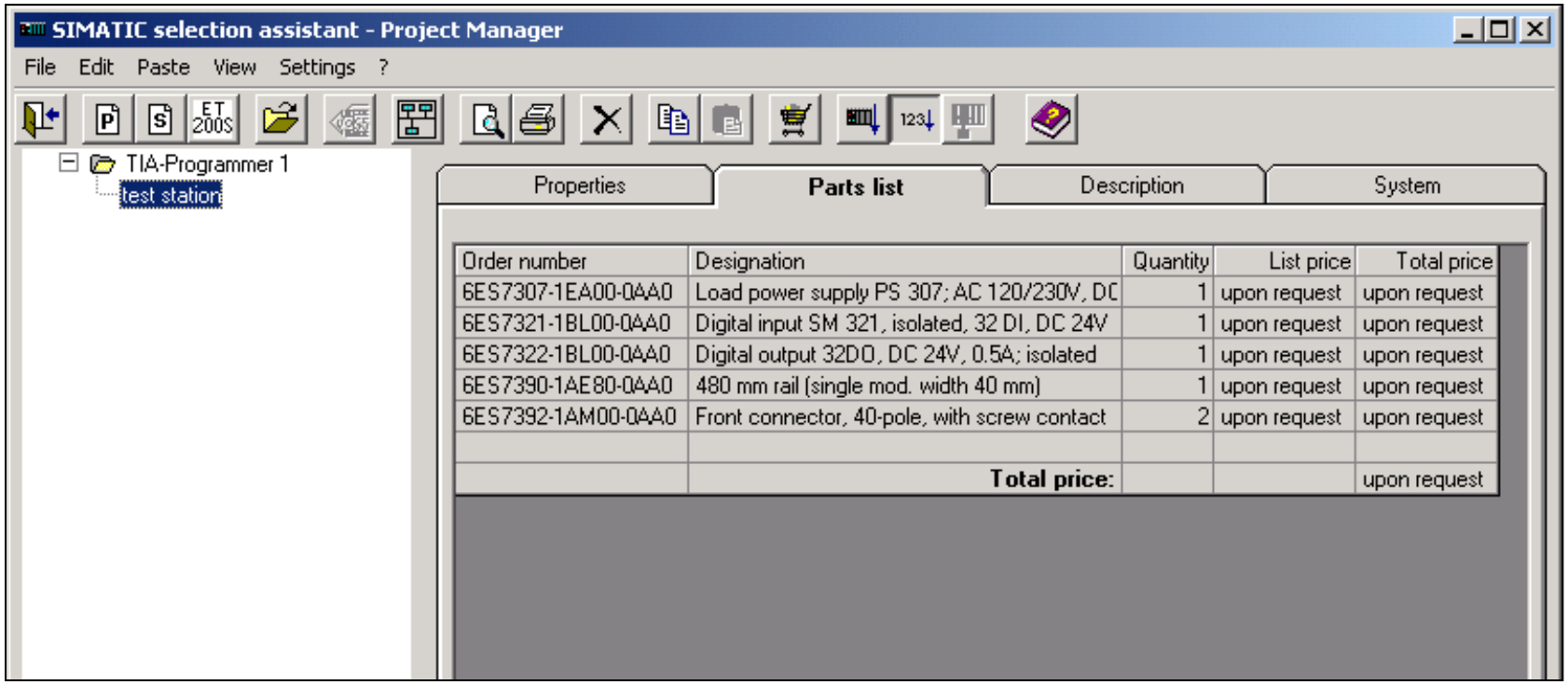
Exporting a Hardware Station



CA01: Selection Assistant for SIMATIC

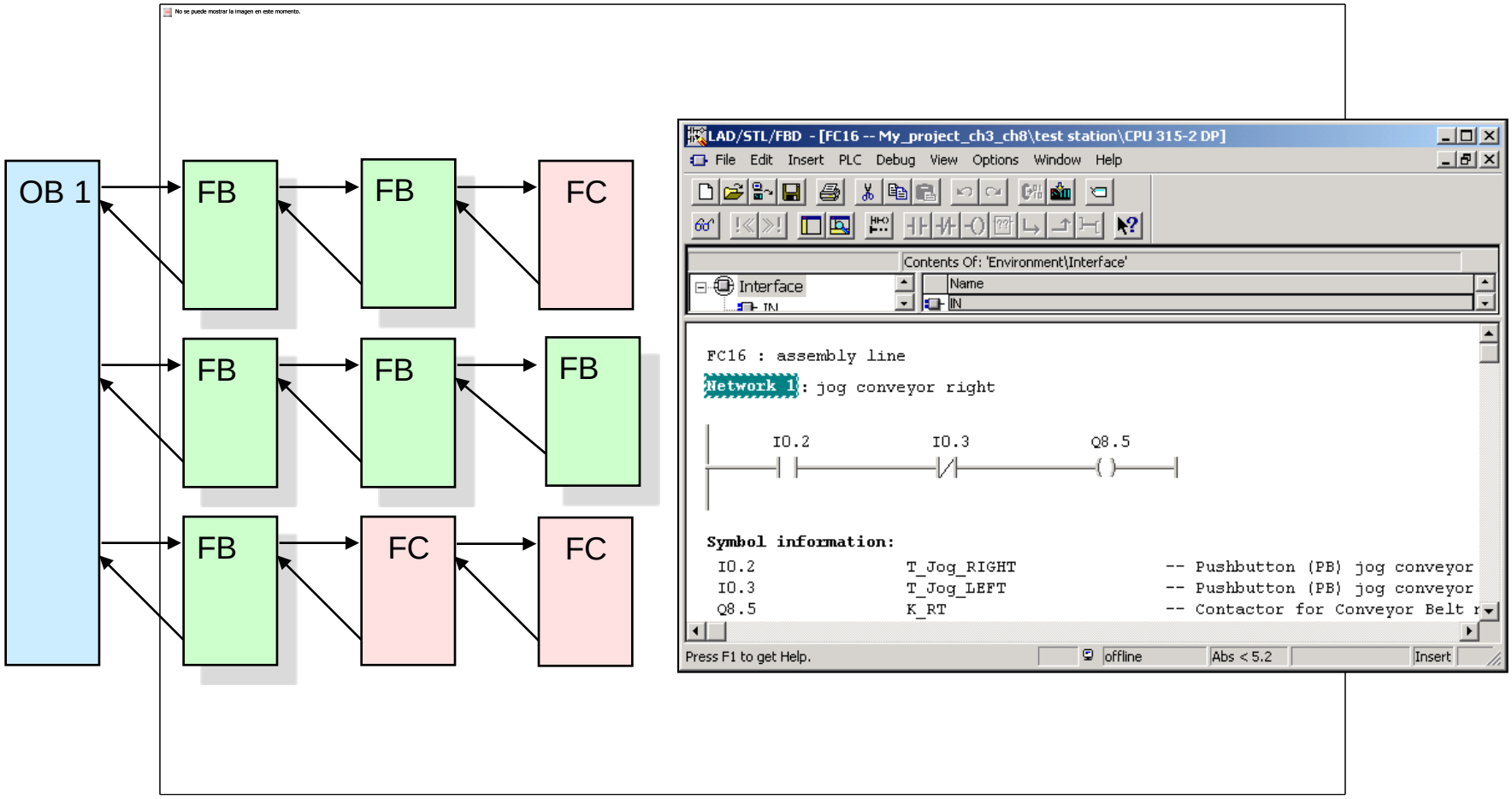


Parts List and Description



Order number	Designation	Quantity	List price	Total price
6ES7307-1EA00-0AA0	Load power supply PS 307; AC 120/230V, DC	1	upon request	upon request
6ES7321-1BL00-0AA0	Digital input SM 321, isolated, 32 DI, DC 24V	1	upon request	upon request
6ES7322-1BL00-0AA0	Digital output 32DO, DC 24V, 0.5A; isolated	1	upon request	upon request
6ES7390-1AE80-0AA0	480 mm rail (single mod. width 40 mm)	1	upon request	upon request
6ES7392-1AM00-0AA0	Front connector, 40-pole, with screw contact	2	upon request	upon request
	Total price:			upon request

Block Architecture and the LAD / STL / FBD Editor



Objectives

Upon completion of this chapter the participant will ...

... know the different types of S7 blocks

... understand the principle of "structured programming"

... know the meaning of the process images (PII, PIQ)

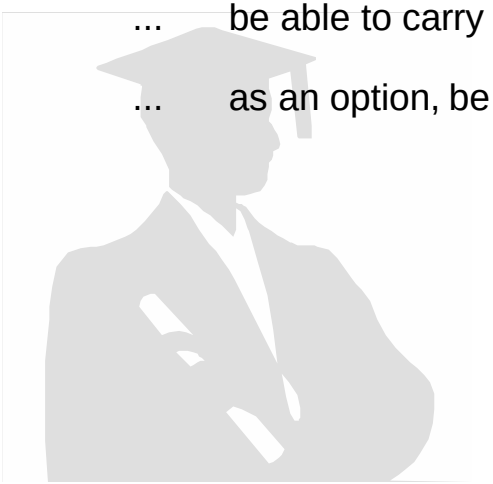
... be able to explain the principle of cyclic program execution

... know and be able to select the LAD, FBD and STL programming languages

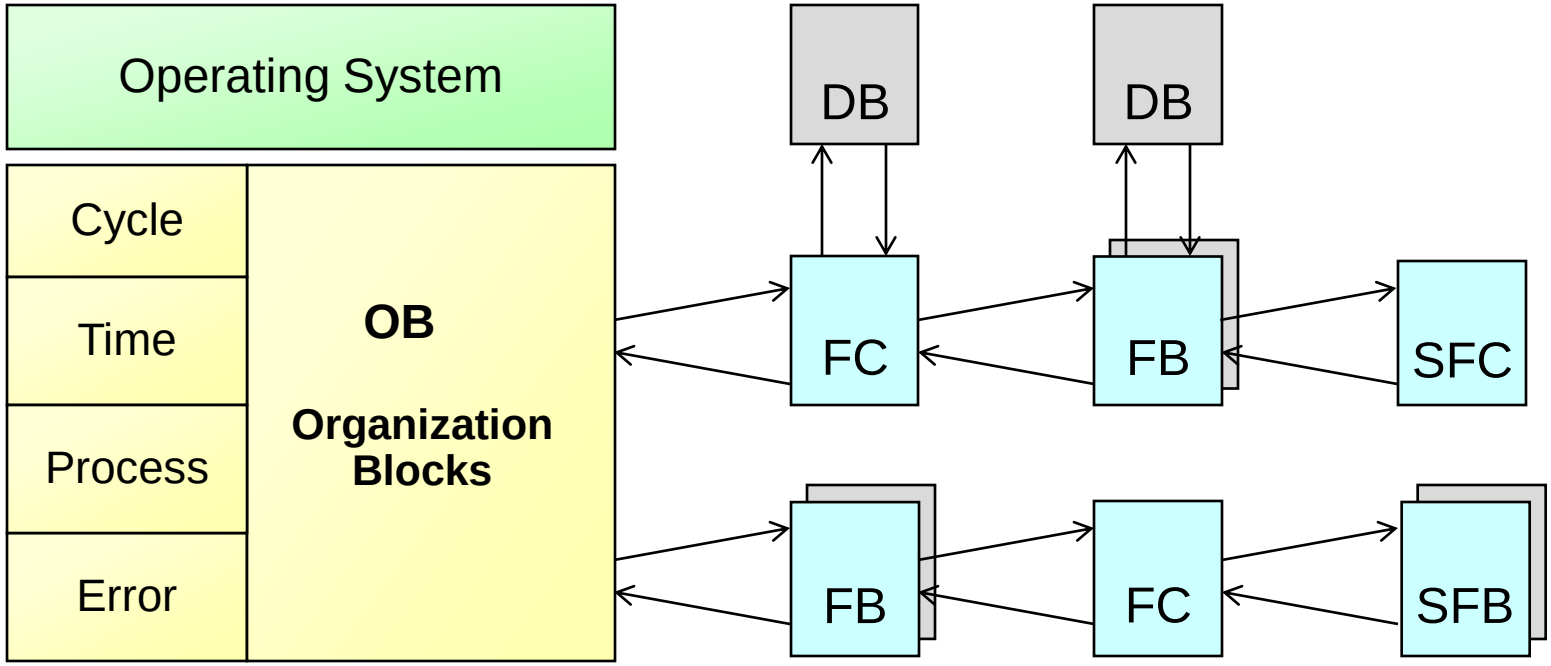
... be able to edit, save and download a block with the LAD/STL/FBD Editor

... be able to carry out a simple program debugging with the "Monitor Block" test function

... as an option, be able to make customizations to the LAD/FBD/STL Editor

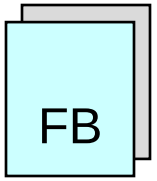


Types of Program Blocks



Legend:

OB = Organization Block
FB = Function Block
FC = Function
SFB = System Function Block
SFC = System Function
DB = Data Block



FB with
Instance DB

Maximum nesting depth:

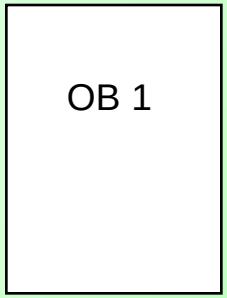
S7-300: 8 (16 for CPU 318)

S7-400: 24

(for each priority class,
2 to 4 additional levels for Error OBs)

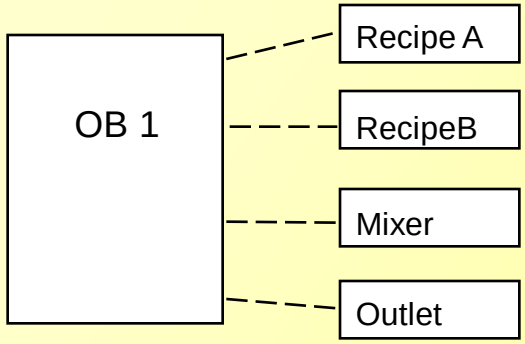
Program Structuring Possibilities

Linear Program



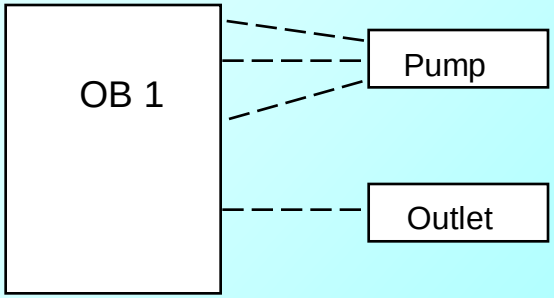
All instructions are found in one block (usually in Organization Block OB 1)

Program Partitioned into Areas



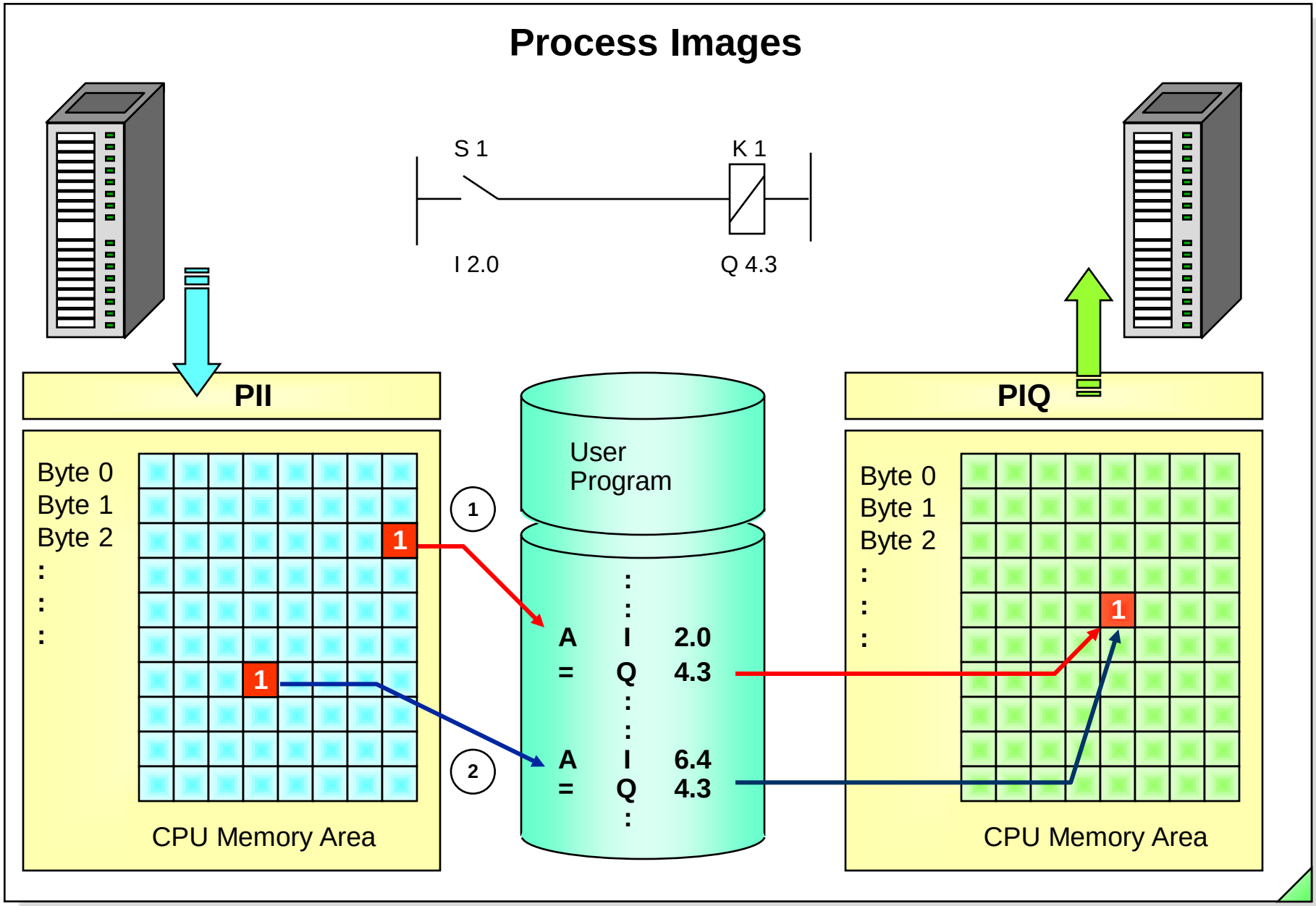
The instructions for the individual functions are found in individual blocks. OB 1 calls the individual blocks one after the other.

Structured Program



Reusable functions are loaded into individual blocks. OB 1 (or other blocks) call these blocks and pass on the pertinent data.

Process Images



Cyclic Program Execution

Before October 1998

- Startup program: Call and execution of OB 100 (once, after Power ON, for example)
- Transfer PIQ to the digital output modules

Start of the cycle monitoring time

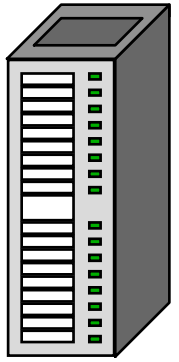
Reading the input states from the digital input modules and saving the states in the process image (PII)

Call and execution of OB1

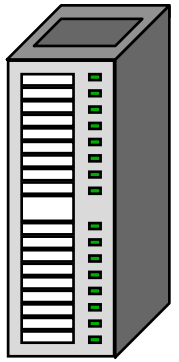
(possible interruption by call of other OBs for events such as time-of-day interrupt, hardware interrupts etc.)

Writing the process-image output table (PIQ) in the digital output modules

CPU Cycle



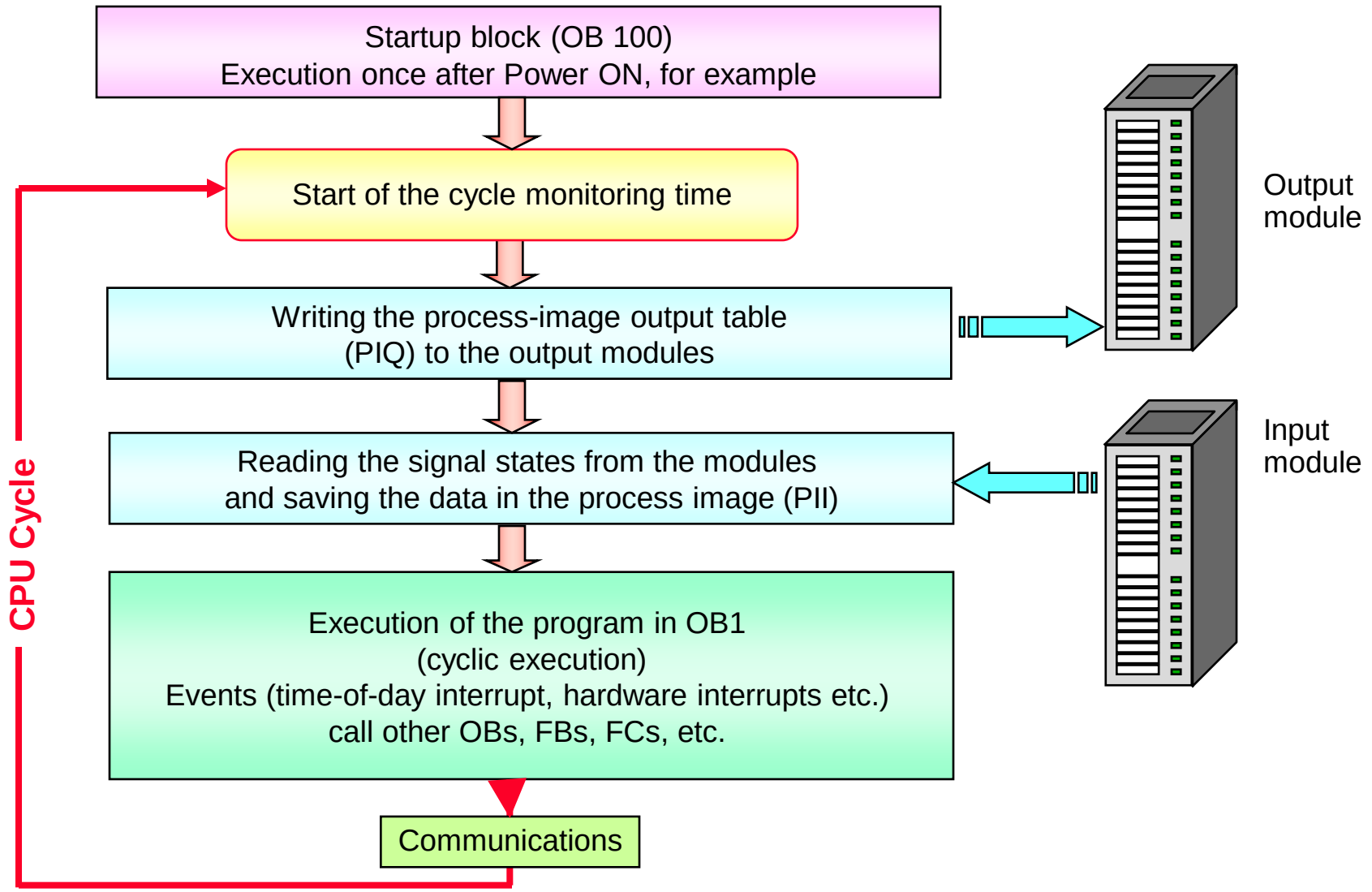
digital input module



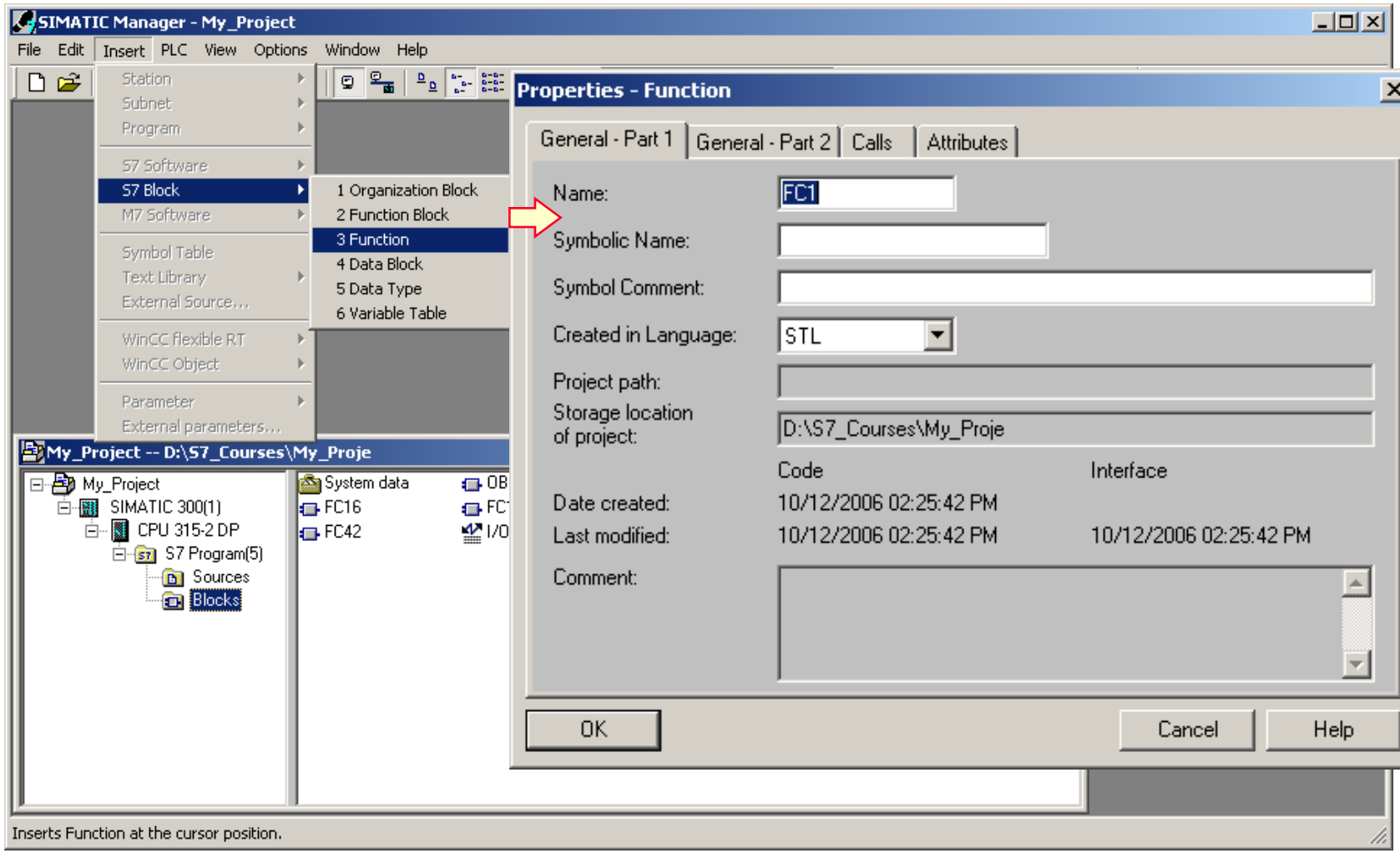
digital output module

Cyclic Program Execution

After October 1998



Inserting an S7 Block

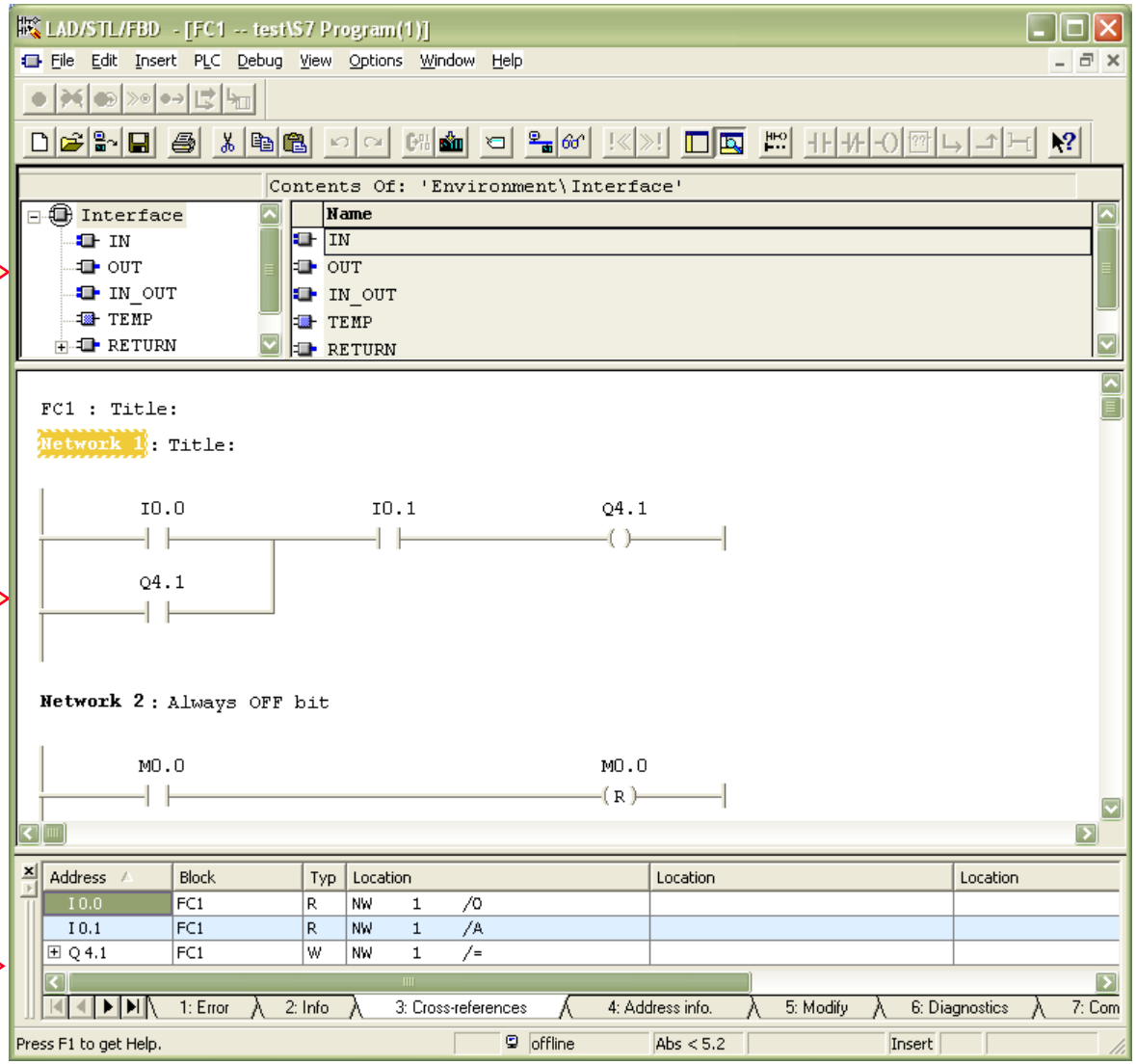


The LAD / STL / FBD Editor

Declaration Table

Code Section

Detail Window

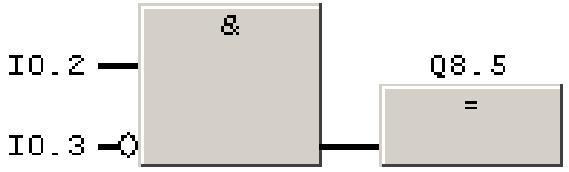


The STEP 7 Programming Languages

Function Block Diagram

FC16 : Conveyor Control

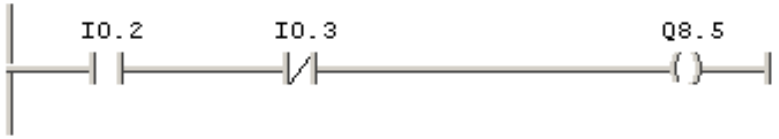
Network 1: Jog Conveyor RIGHT



Ladder Diagram

FC16 : Conveyor Control

Network 1: Jog Conveyor RIGHT



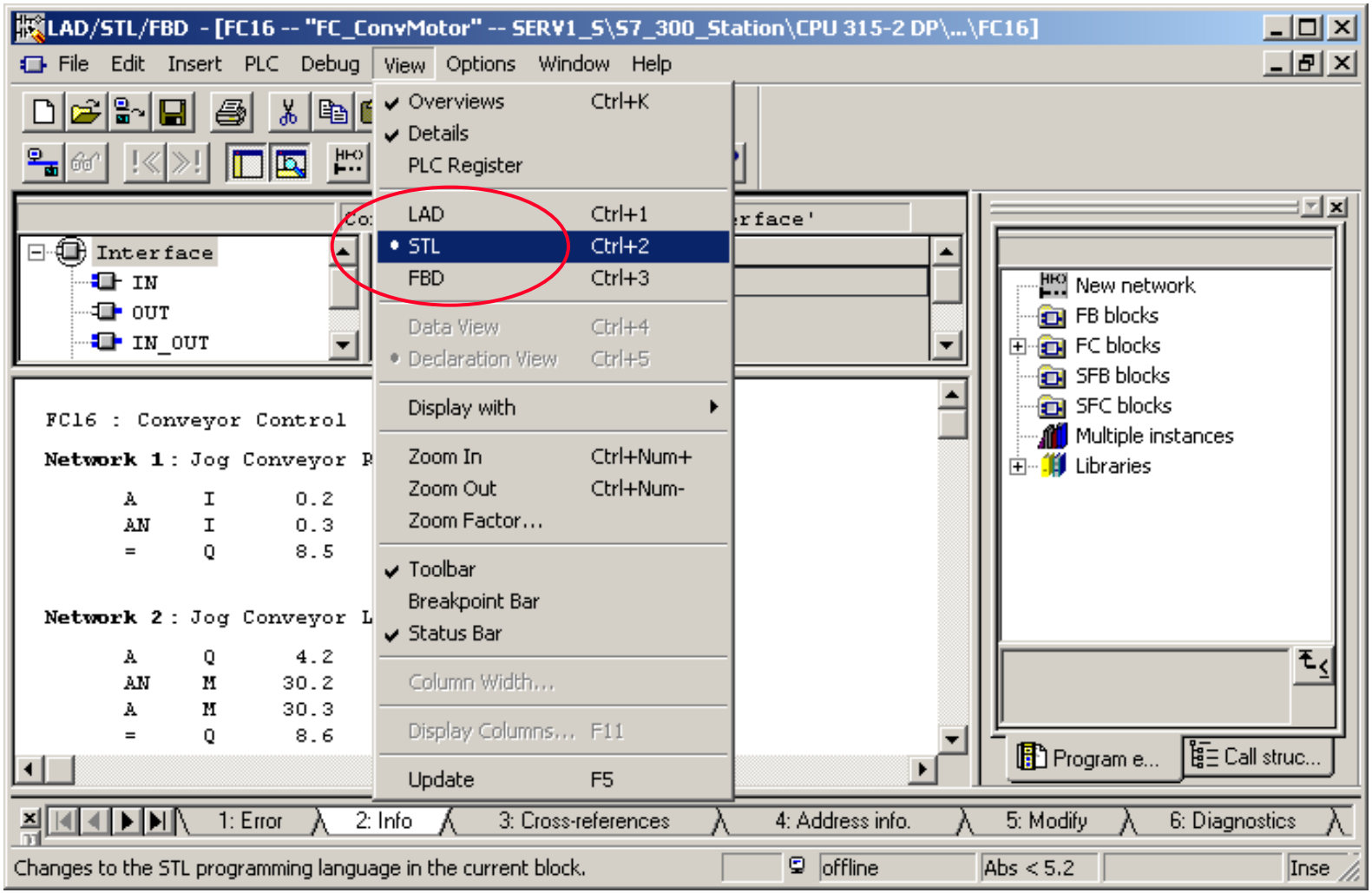
Statement List

FC16 : Conveyor Control

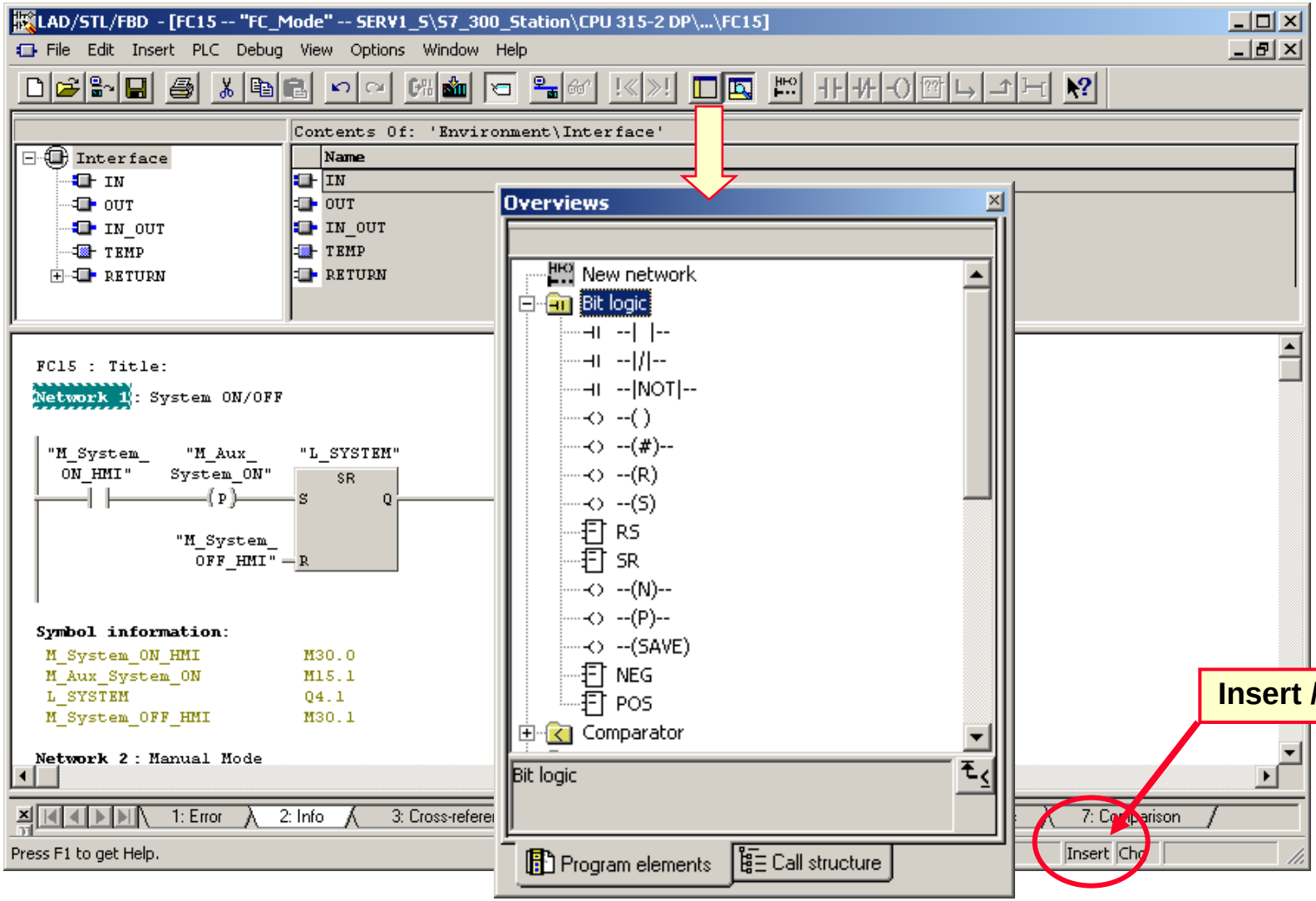
Network 1: Jog Conveyor RIGHT

```
A     I     0.2
AN    I     0.3
=     Q     8.5
```

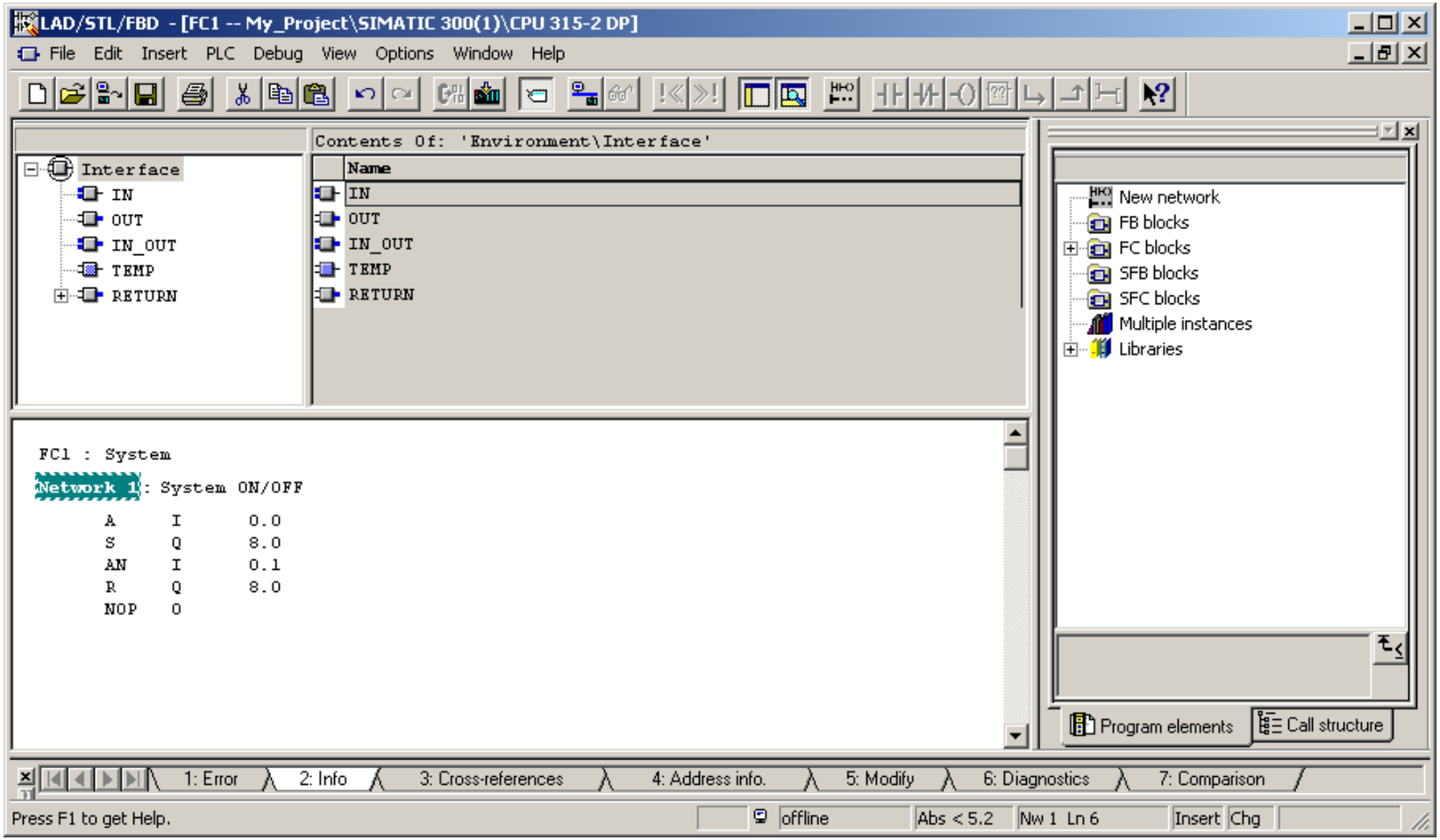
Selecting the Programming Language



Programming in LAD and FBD

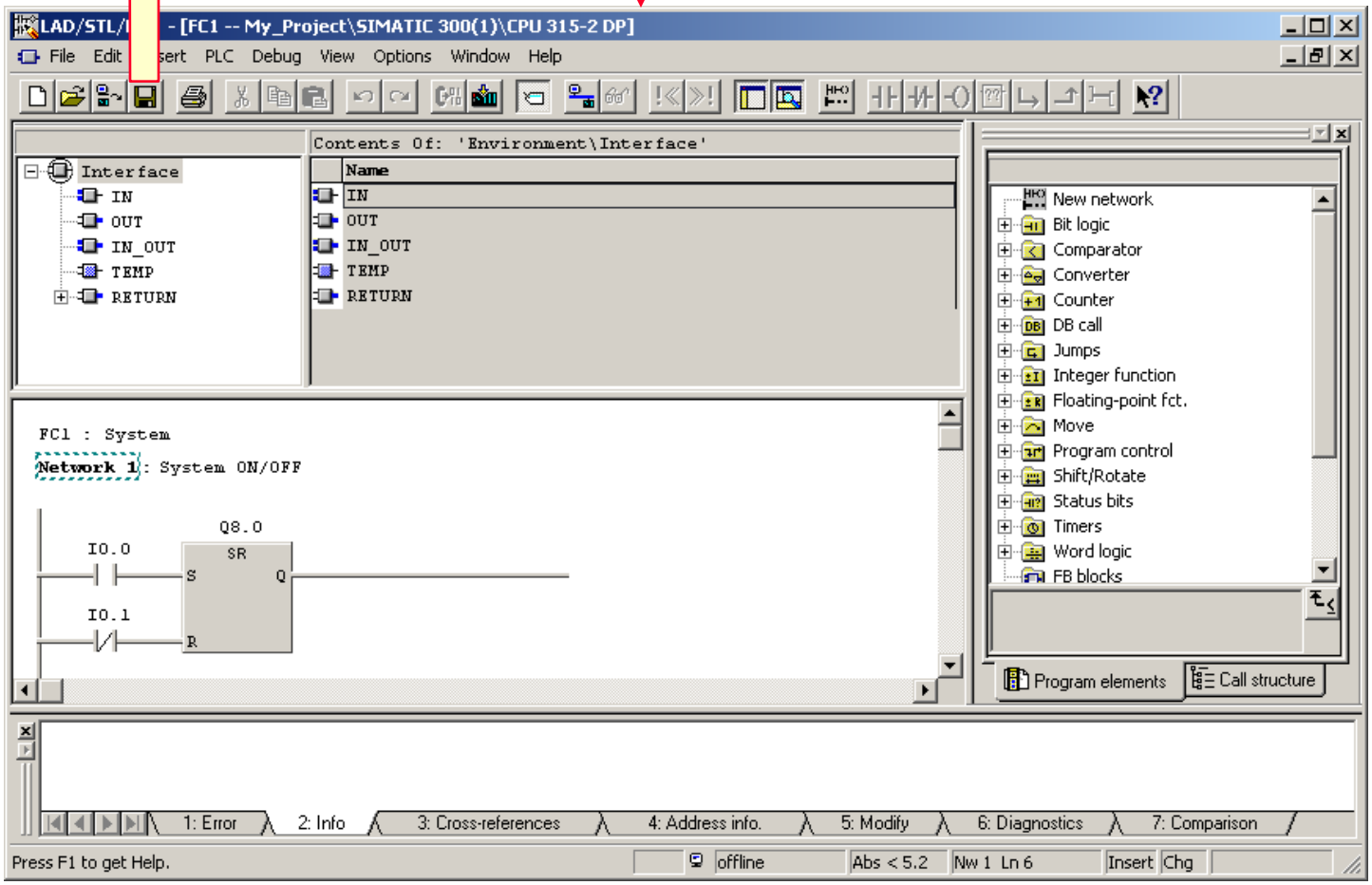


Programming in STL

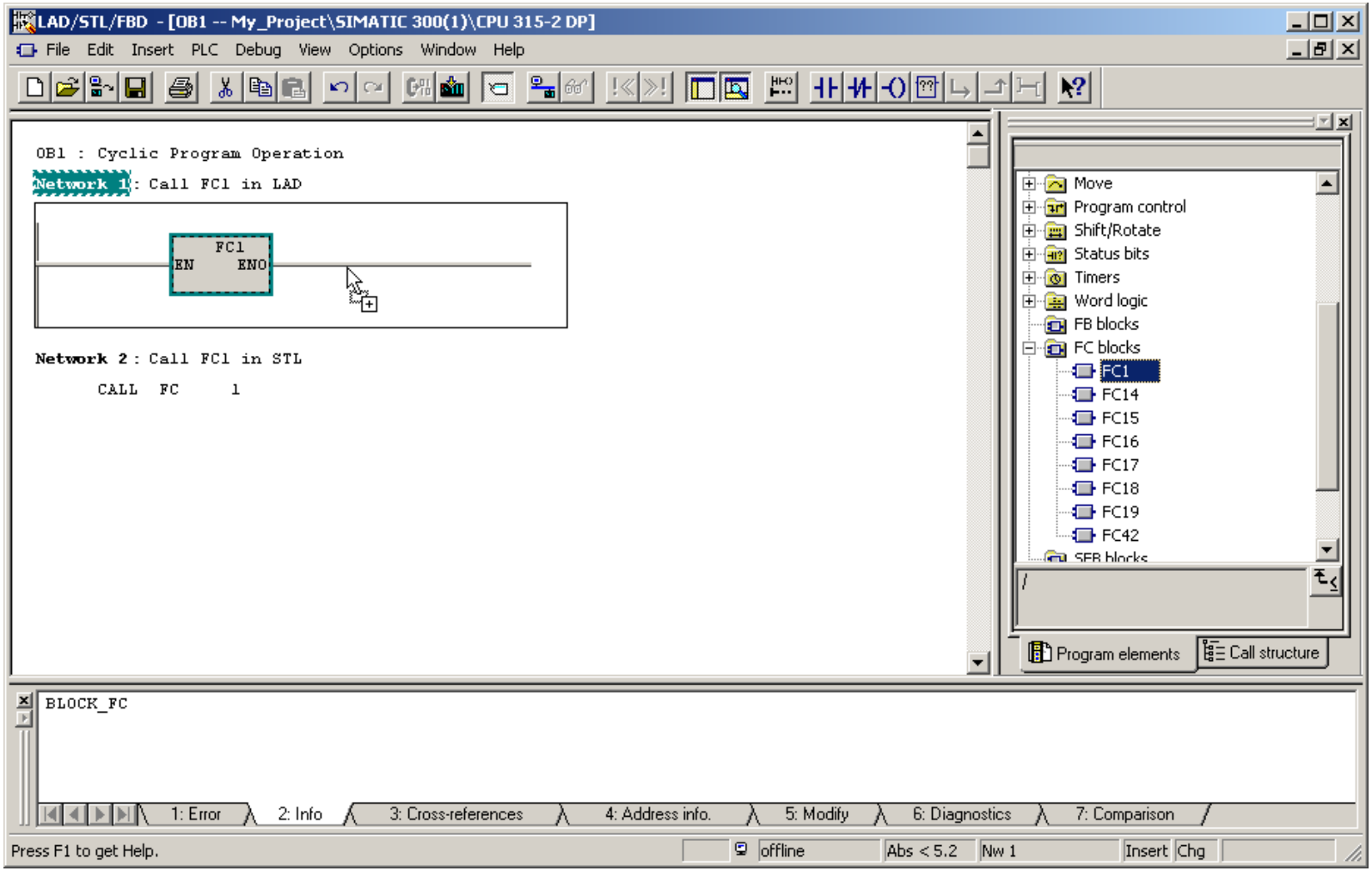


Saving a Block

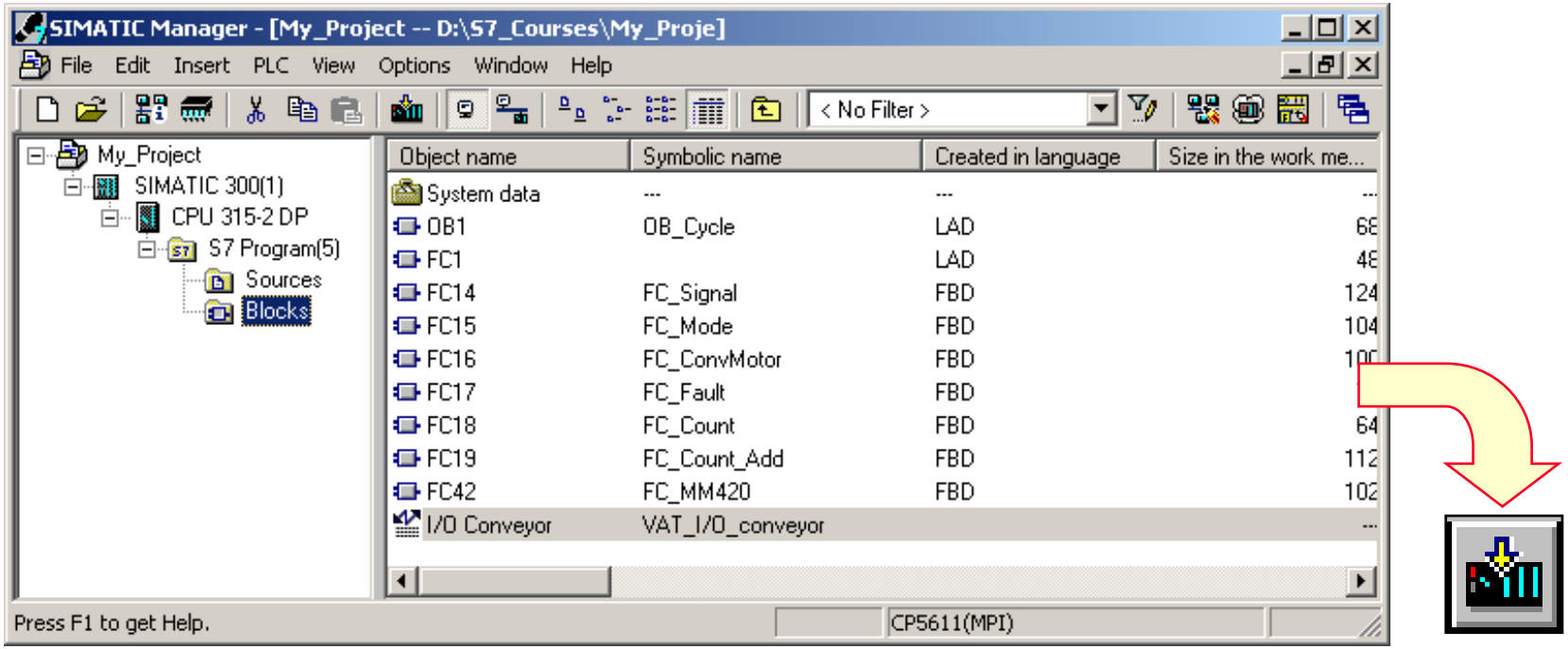
Current project directory with block name



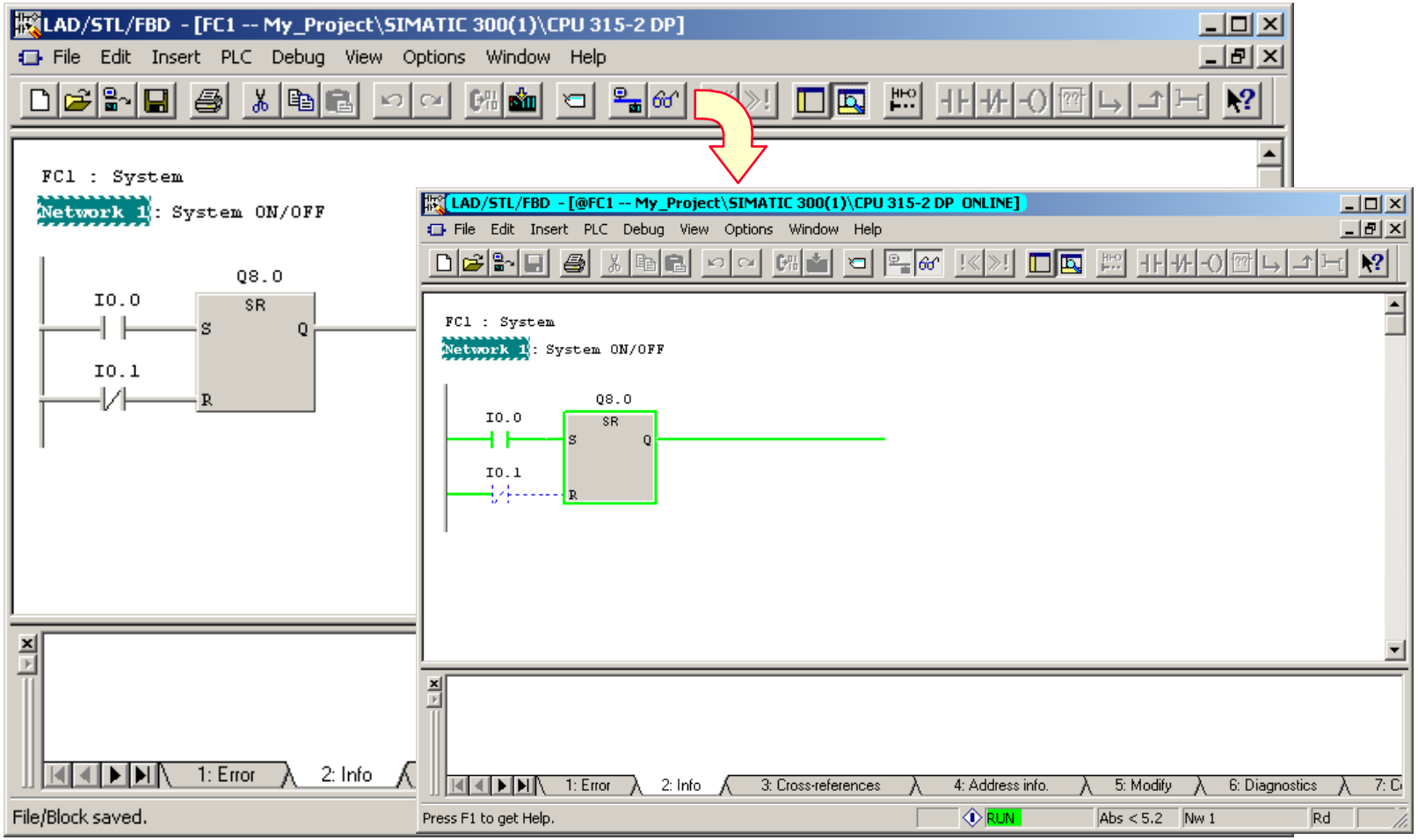
Calling a Block in OB1



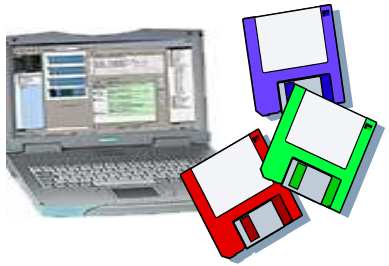
Downloading Blocks into the PLC



Simple Program Debugging

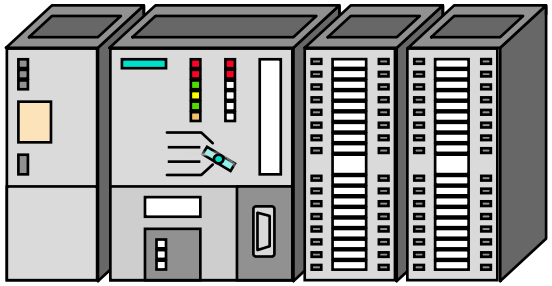


Downloading and Saving Modified Blocks



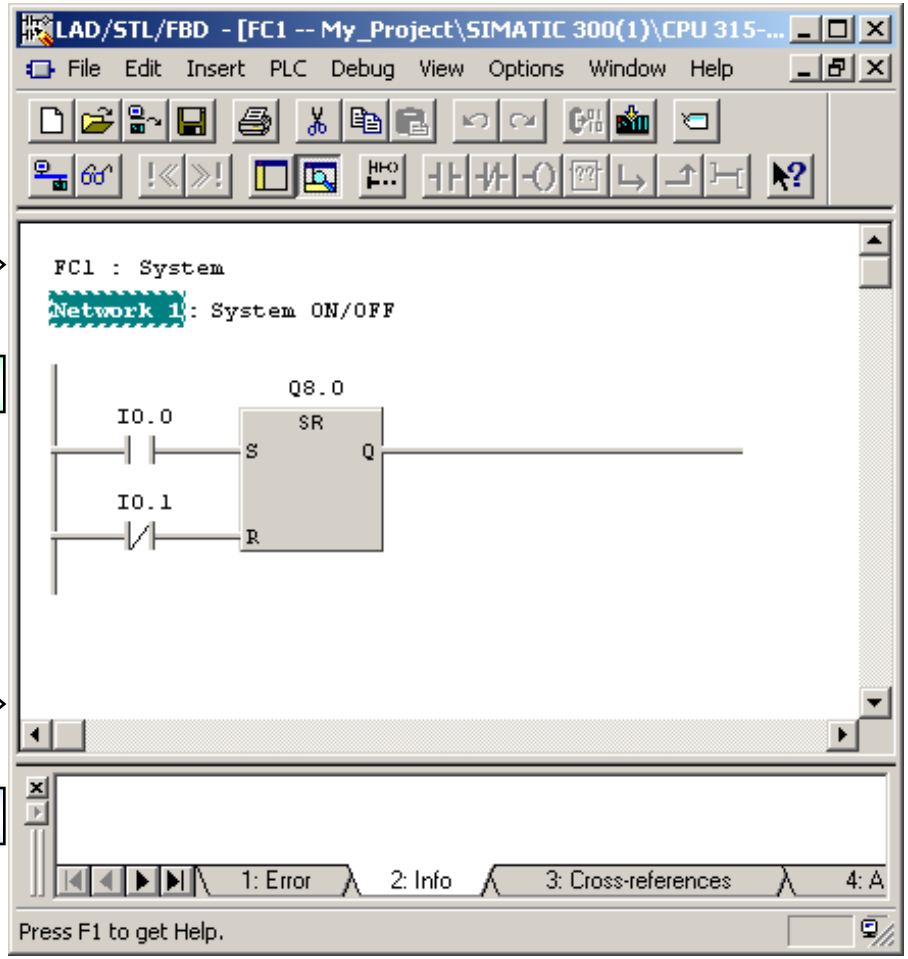
Open offline

Save

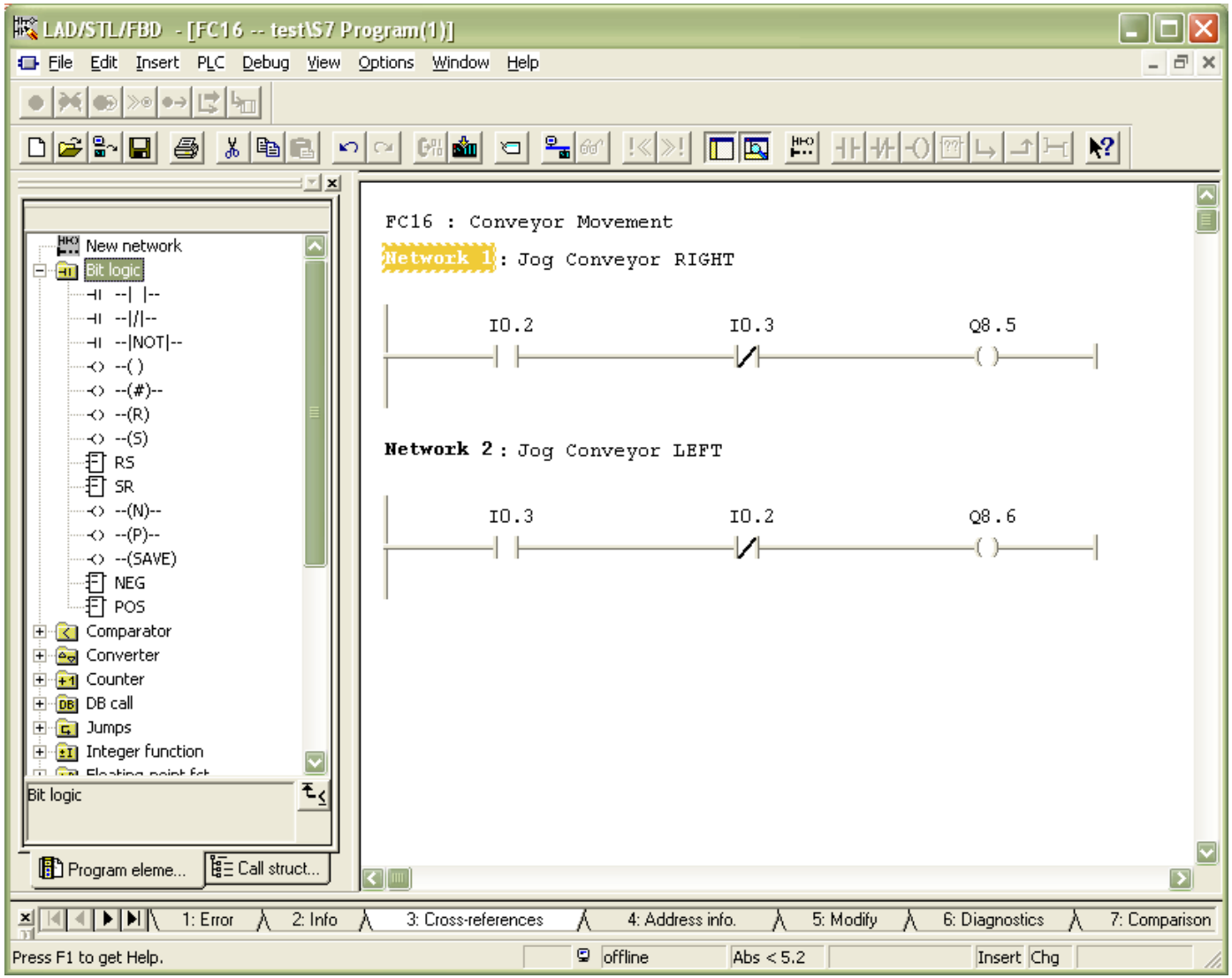


Open online

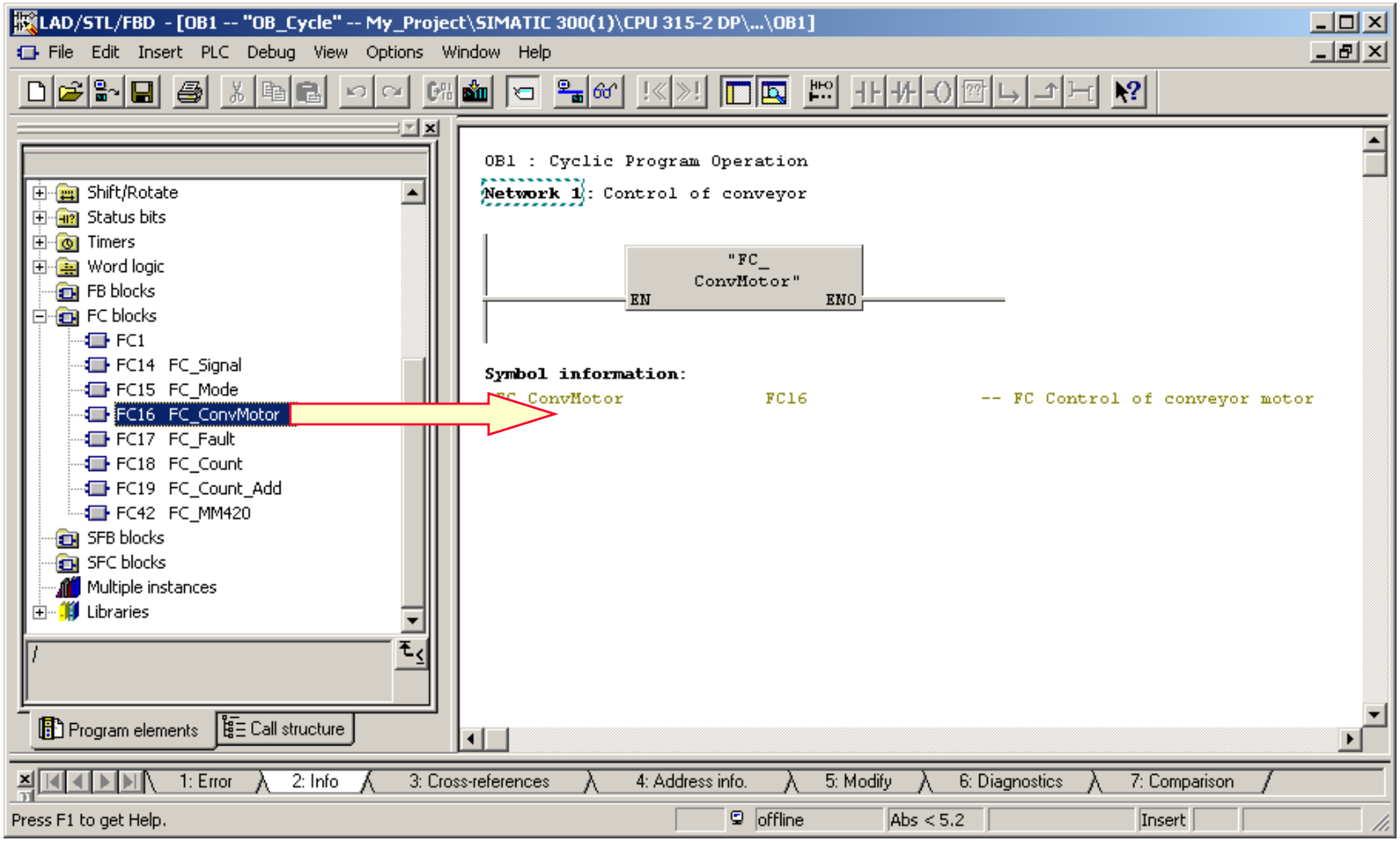
Download



Exercise 1: Jogging the Conveyor Motor (FC 16)

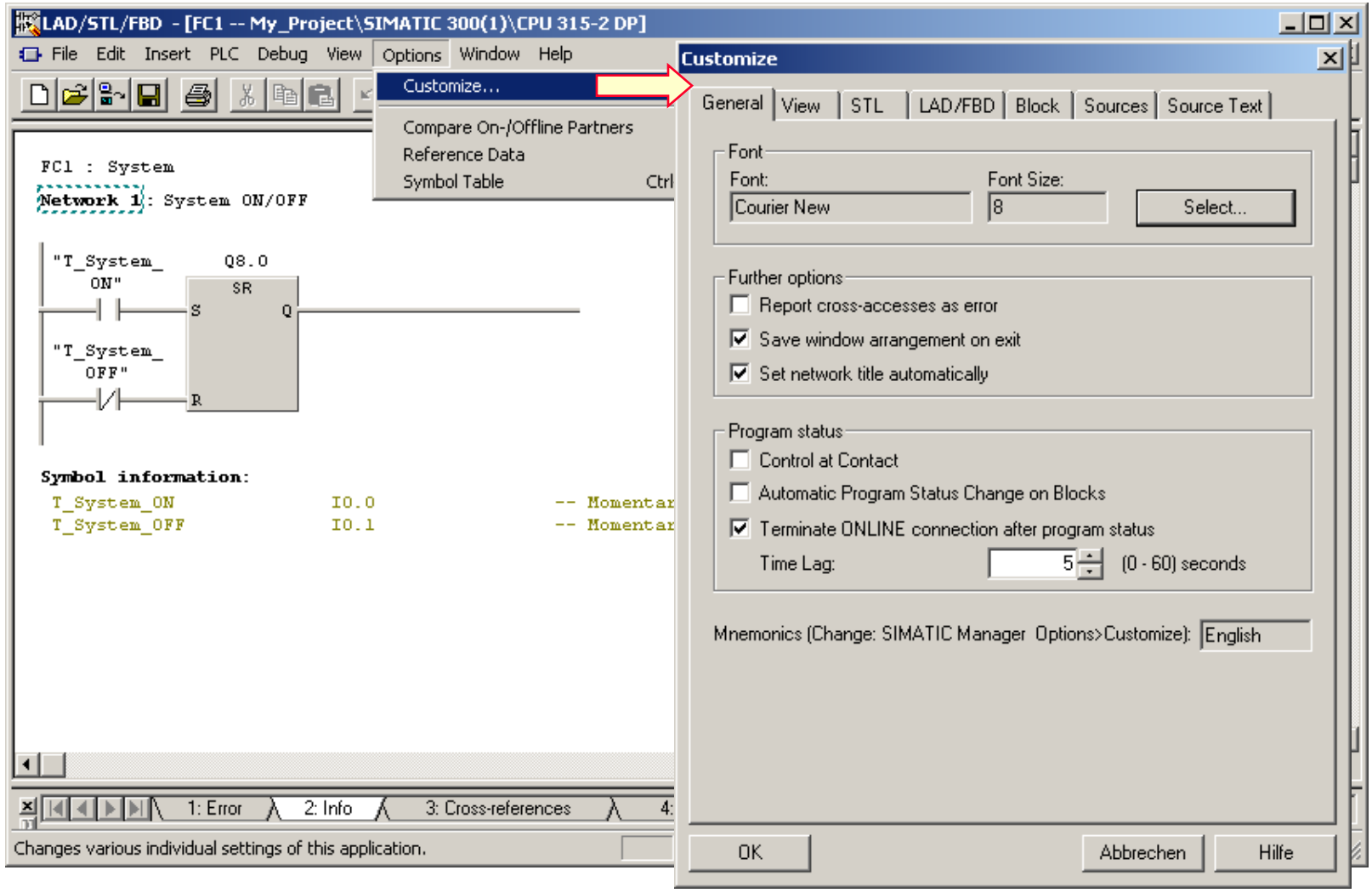


Exercise 2: Calling FC 16 in OB 1

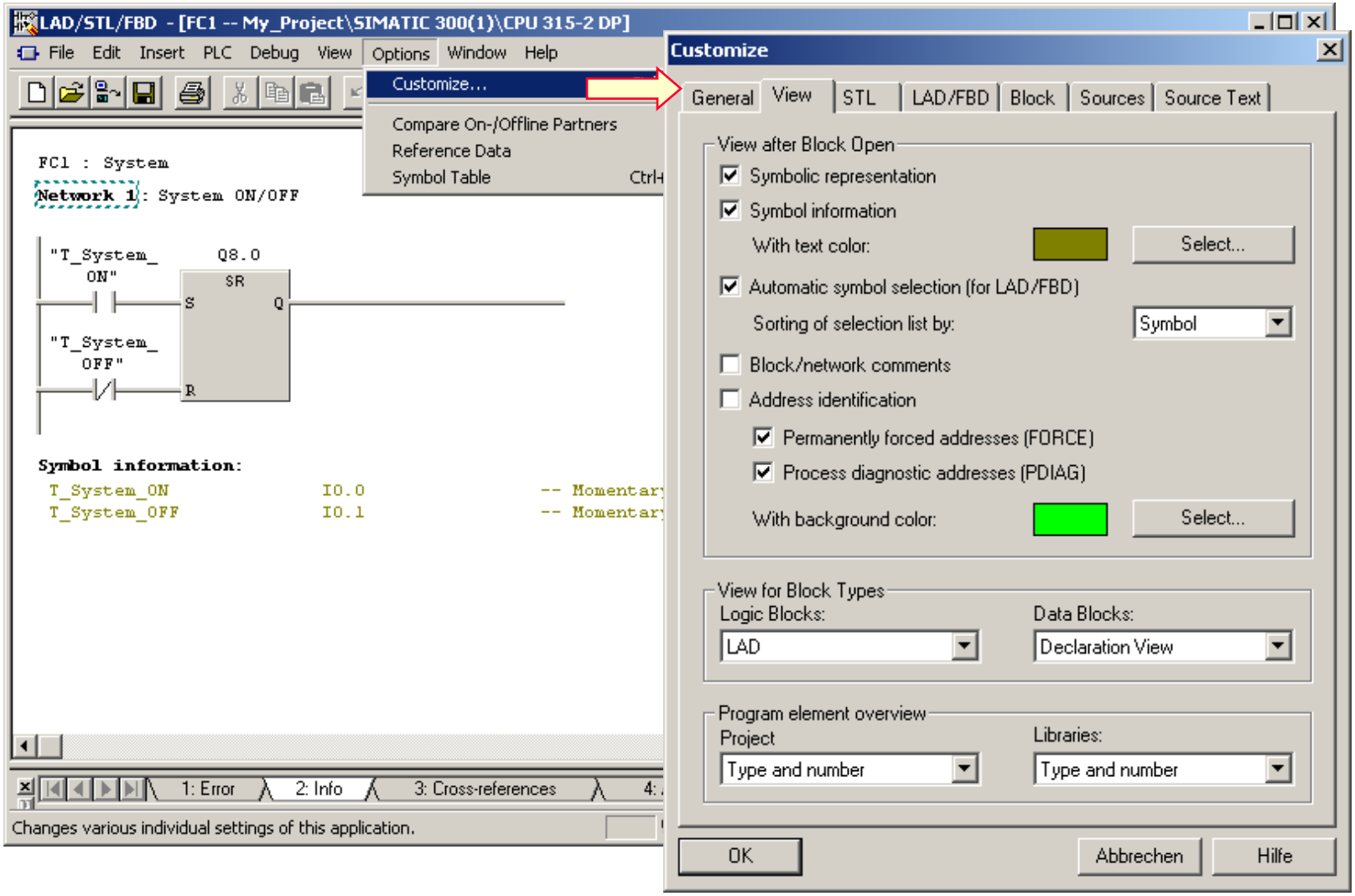


If You Want to Know More

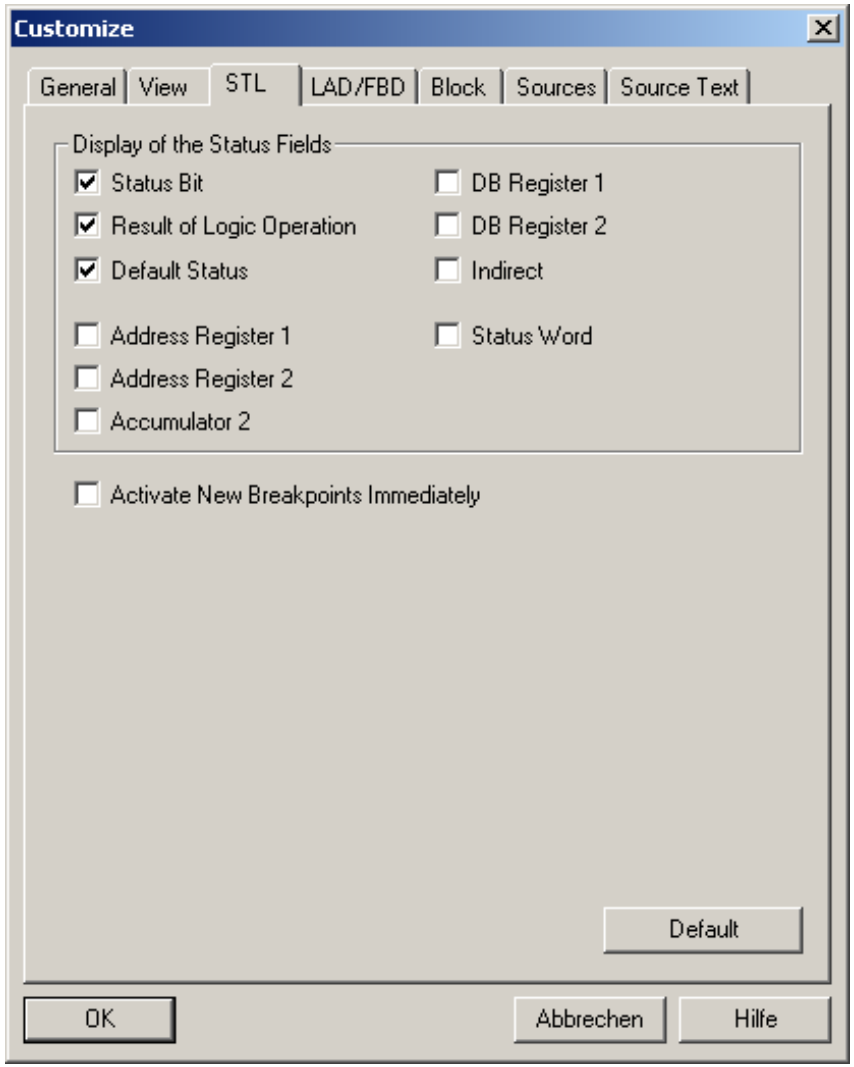
Editor Customization: "General" Tab



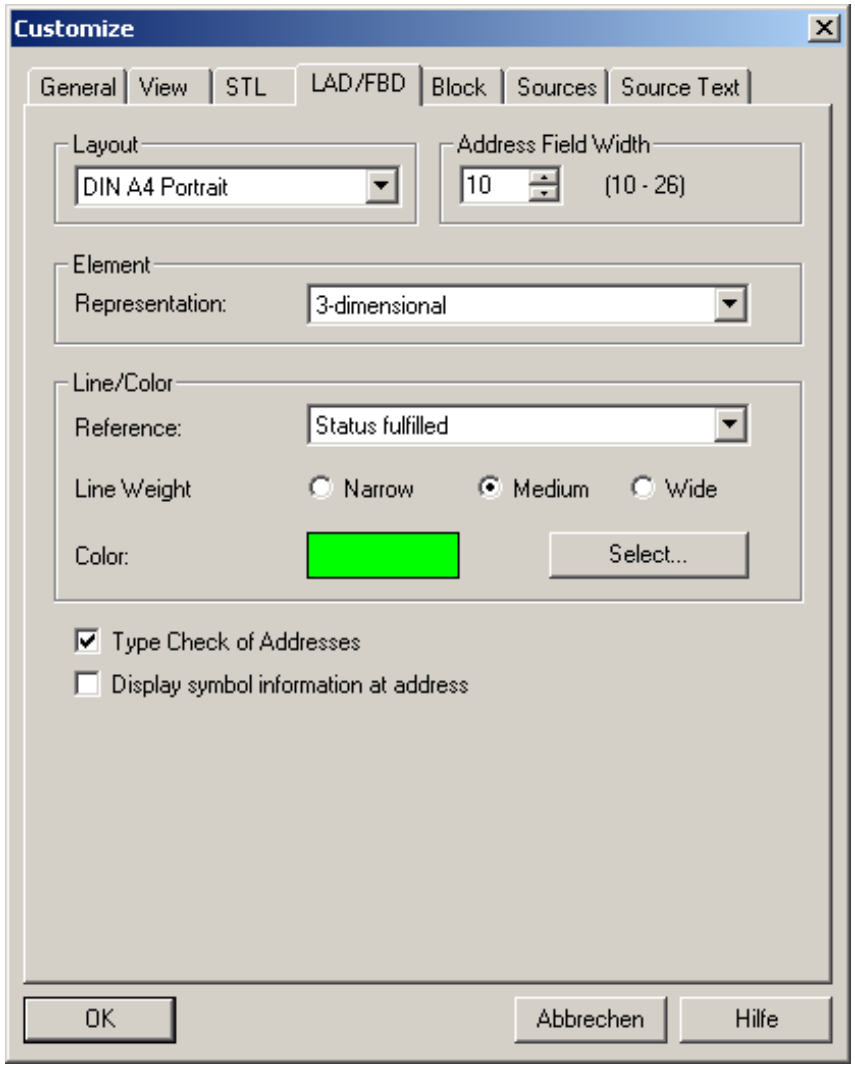
Editor Customization: "View " Tab



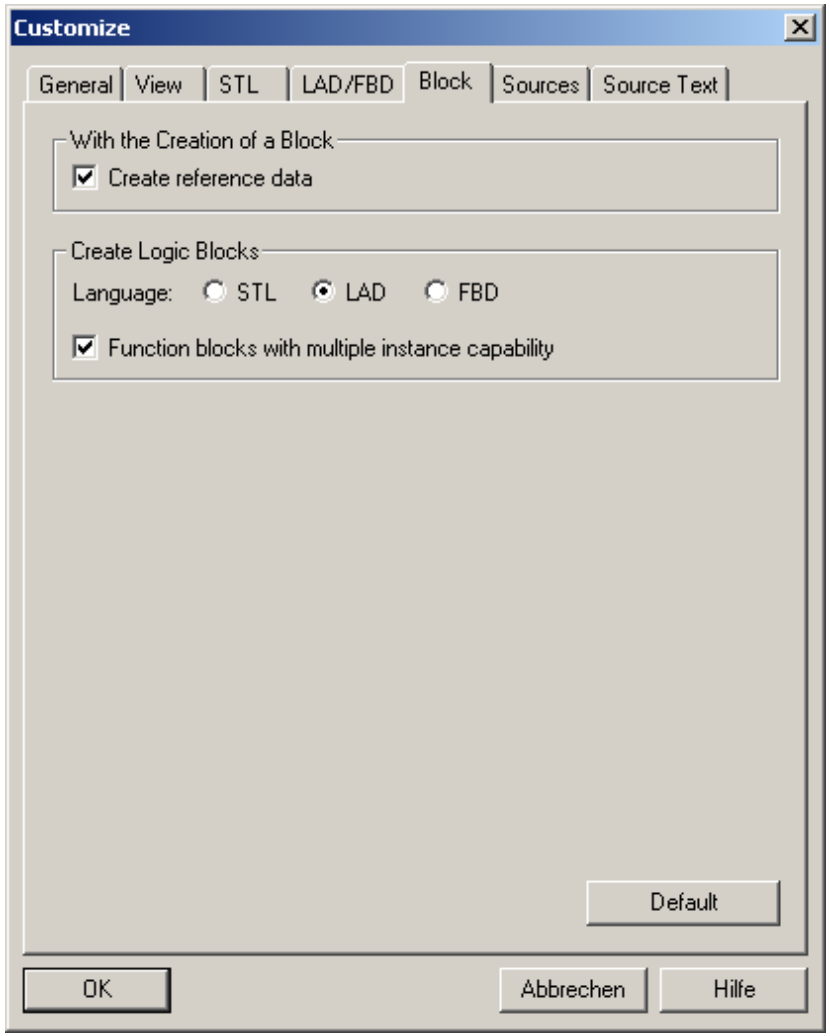
Editor Customization: "STL" Tab

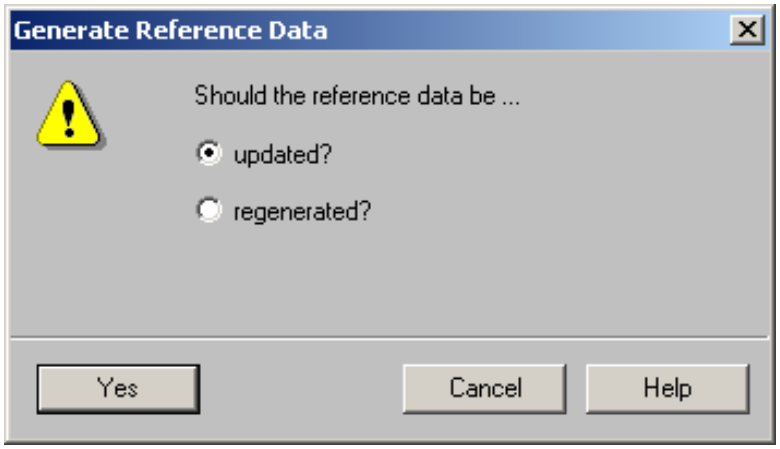


Editor Customization: "LAD/FBD" Tab

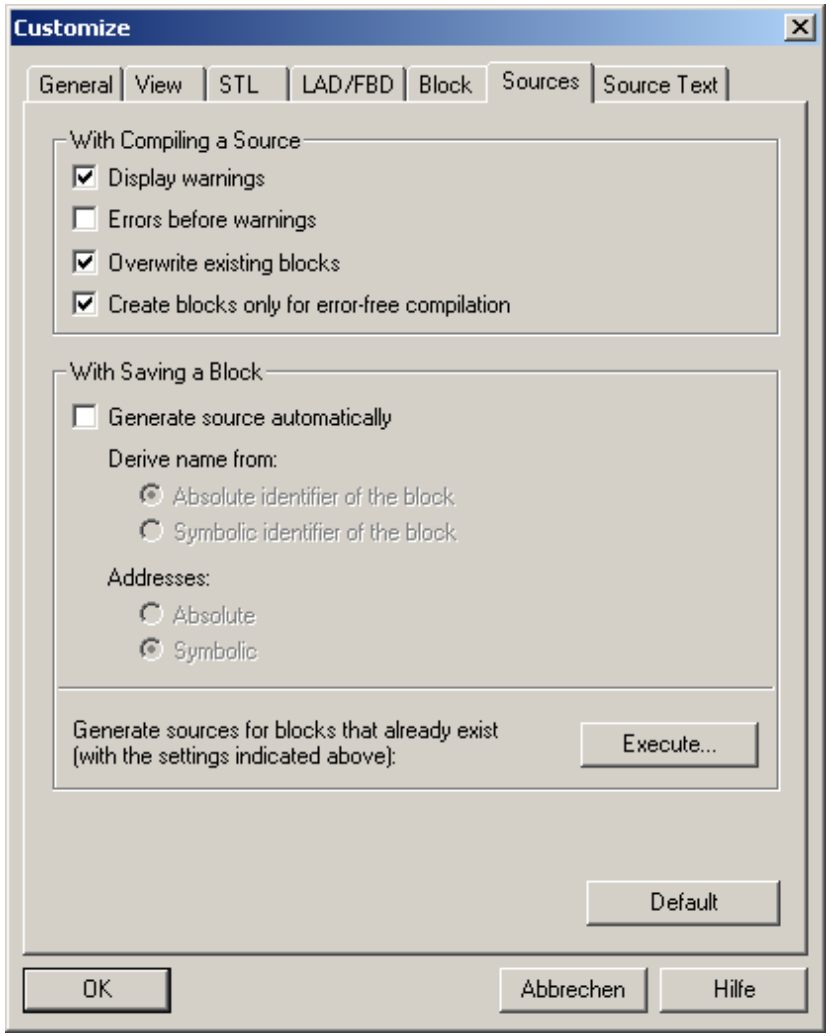


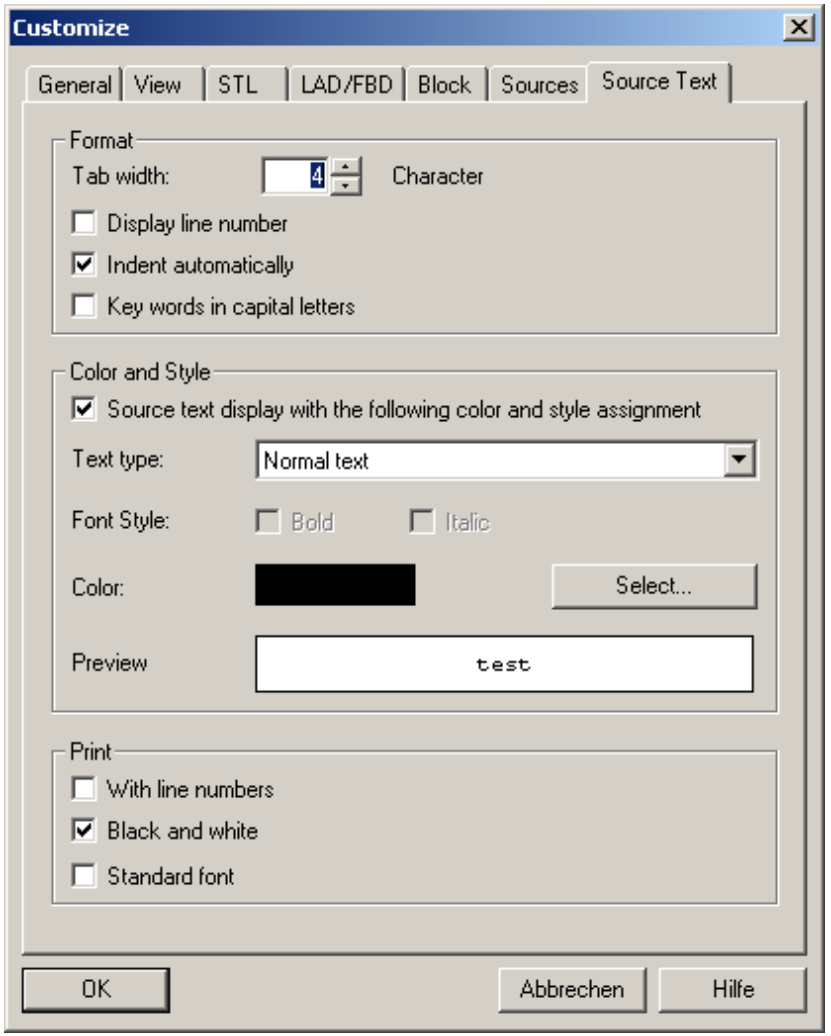
Editor Customization: "Block" Tab



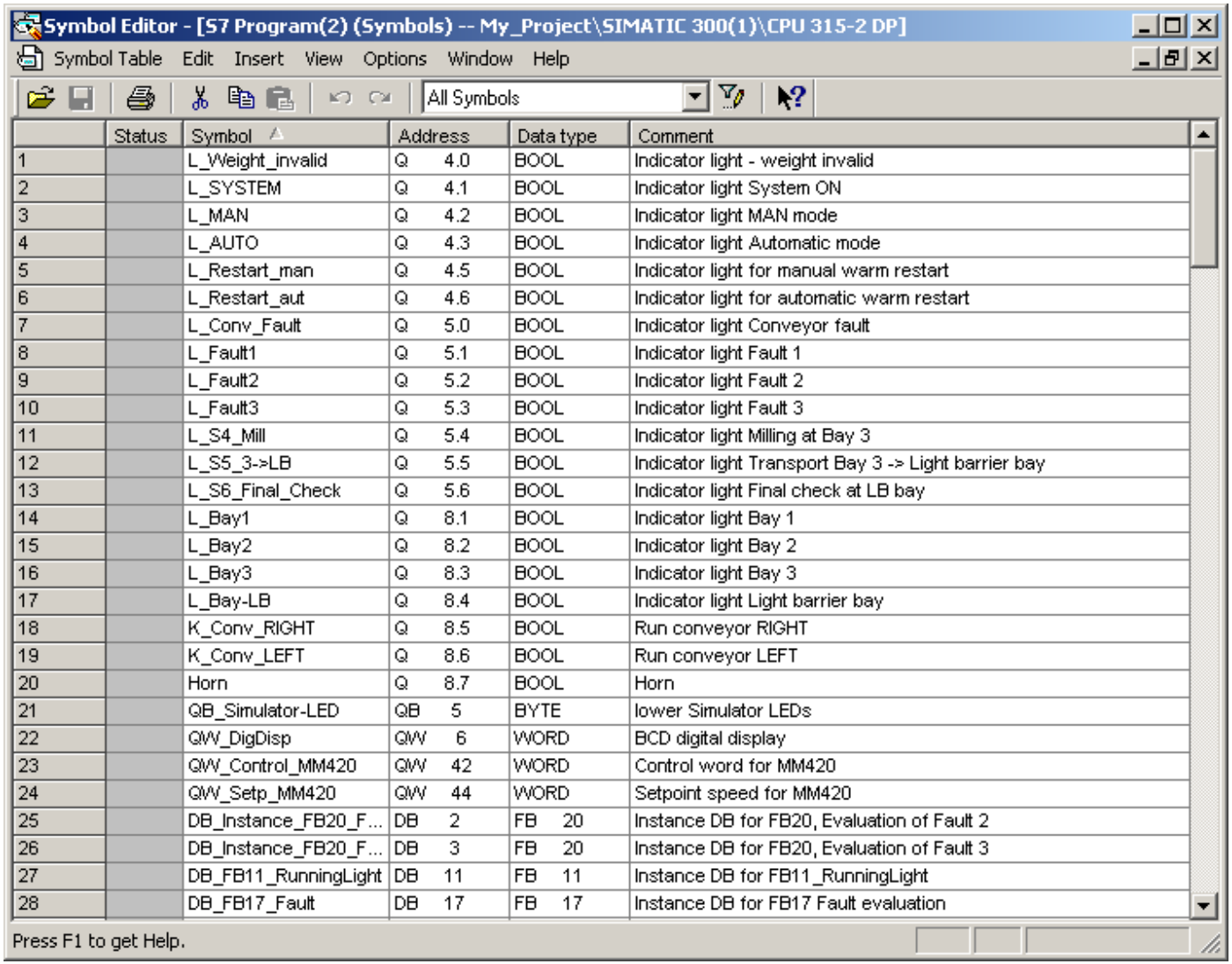


Editor Customization: "Sources/Source Text" Tab





Symbolic Addressing



	Status	Symbol	Address	Data type	Comment
1		L_Weight_invalid	Q 4.0	BOOL	Indicator light - weight invalid
2		L_SYSTEM	Q 4.1	BOOL	Indicator light System ON
3		L_MAN	Q 4.2	BOOL	Indicator light MAN mode
4		L_AUTO	Q 4.3	BOOL	Indicator light Automatic mode
5		L_Restart_man	Q 4.5	BOOL	Indicator light for manual warm restart
6		L_Restart_aut	Q 4.6	BOOL	Indicator light for automatic warm restart
7		L_Conv_Fault	Q 5.0	BOOL	Indicator light Conveyor fault
8		L_Fault1	Q 5.1	BOOL	Indicator light Fault 1
9		L_Fault2	Q 5.2	BOOL	Indicator light Fault 2
10		L_Fault3	Q 5.3	BOOL	Indicator light Fault 3
11		L_S4_Mill	Q 5.4	BOOL	Indicator light Milling at Bay 3
12		L_S5_3->LB	Q 5.5	BOOL	Indicator light Transport Bay 3 -> Light barrier bay
13		L_S6_Final_Check	Q 5.6	BOOL	Indicator light Final check at LB bay
14		L_Bay1	Q 8.1	BOOL	Indicator light Bay 1
15		L_Bay2	Q 8.2	BOOL	Indicator light Bay 2
16		L_Bay3	Q 8.3	BOOL	Indicator light Bay 3
17		L_Bay-LB	Q 8.4	BOOL	Indicator light Light barrier bay
18		K_Conv_RIGHT	Q 8.5	BOOL	Run conveyor RIGHT
19		K_Conv_LEFT	Q 8.6	BOOL	Run conveyor LEFT
20		Horn	Q 8.7	BOOL	Horn
21		QB_Simulator-LED	QB 5	BYTE	lower Simulator LEDs
22		QW_DigDisp	QW 6	WORD	BCD digital display
23		QW_Control_MM420	QW 42	WORD	Control word for MM420
24		QW_Setp_MM420	QW 44	WORD	Setpoint speed for MM420
25		DB_Instance_FB20_F...	DB 2	FB 20	Instance DB for FB20, Evaluation of Fault 2
26		DB_Instance_FB20_F...	DB 3	FB 20	Instance DB for FB20, Evaluation of Fault 3
27		DB_FB11_RunningLight	DB 11	FB 11	Instance DB for FB11_RunningLight
28		DB_FB17_Fault	DB 17	FB 17	Instance DB for FB17 Fault evaluation

Objectives

Upon completion of the chapter the participant will ...

... know the difference between absolute and symbolic addressing

... know the difference between local and global symbols

... be able to edit a global symbol table

... be able to use the Block Editor to edit global symbols

Absolute and Symbolic Addressing

Absolute Representation

```
A       I0.0
=       Q8.0
A       I0.4
=       Q20.5
Call    FC18
```

Symbolic Representation

```
A       "SYSTEM_ON"
=       "SYSTEM_ON"
A       "M_FORW"
=       "MOTOR_FOR"
Call    "COUNT"
```

Symbol	Address	Data Type	Comment
MOTOR_FOR	Q20.5	BOOL	Motor moves forward
COUNT	FC18	FC18	Count bottles
SYSTEM_ON_SW	I0.0	BOOL	Switch system on
SYSTEM_ON_LT	Q8.0	BOOL	Indicator: System is “On”
M_FORW	I0.4	BOOL	Pushbutton: Motor forward

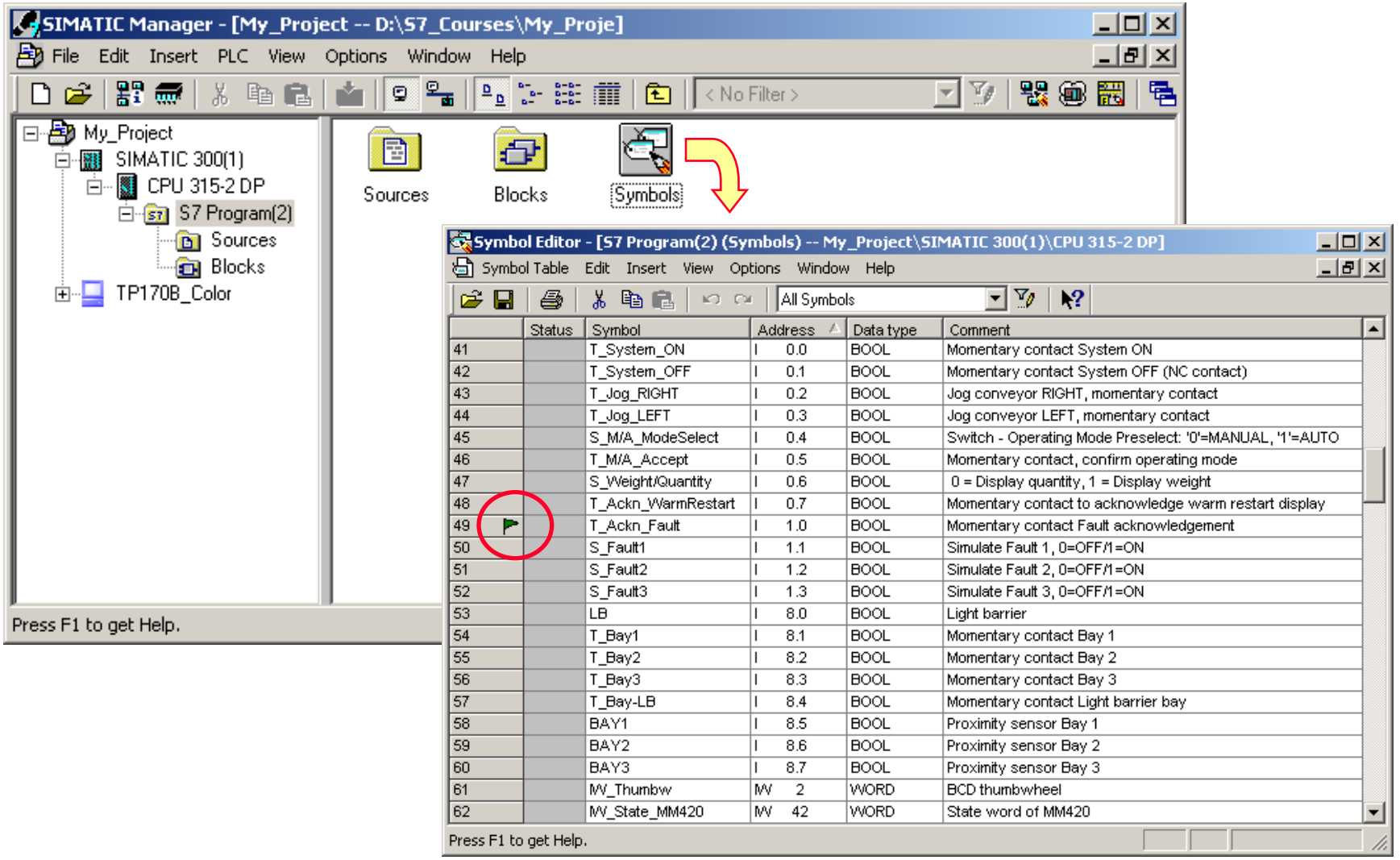
(max. 24 characters)

(max. 80 characters)

Symbolic Addressing

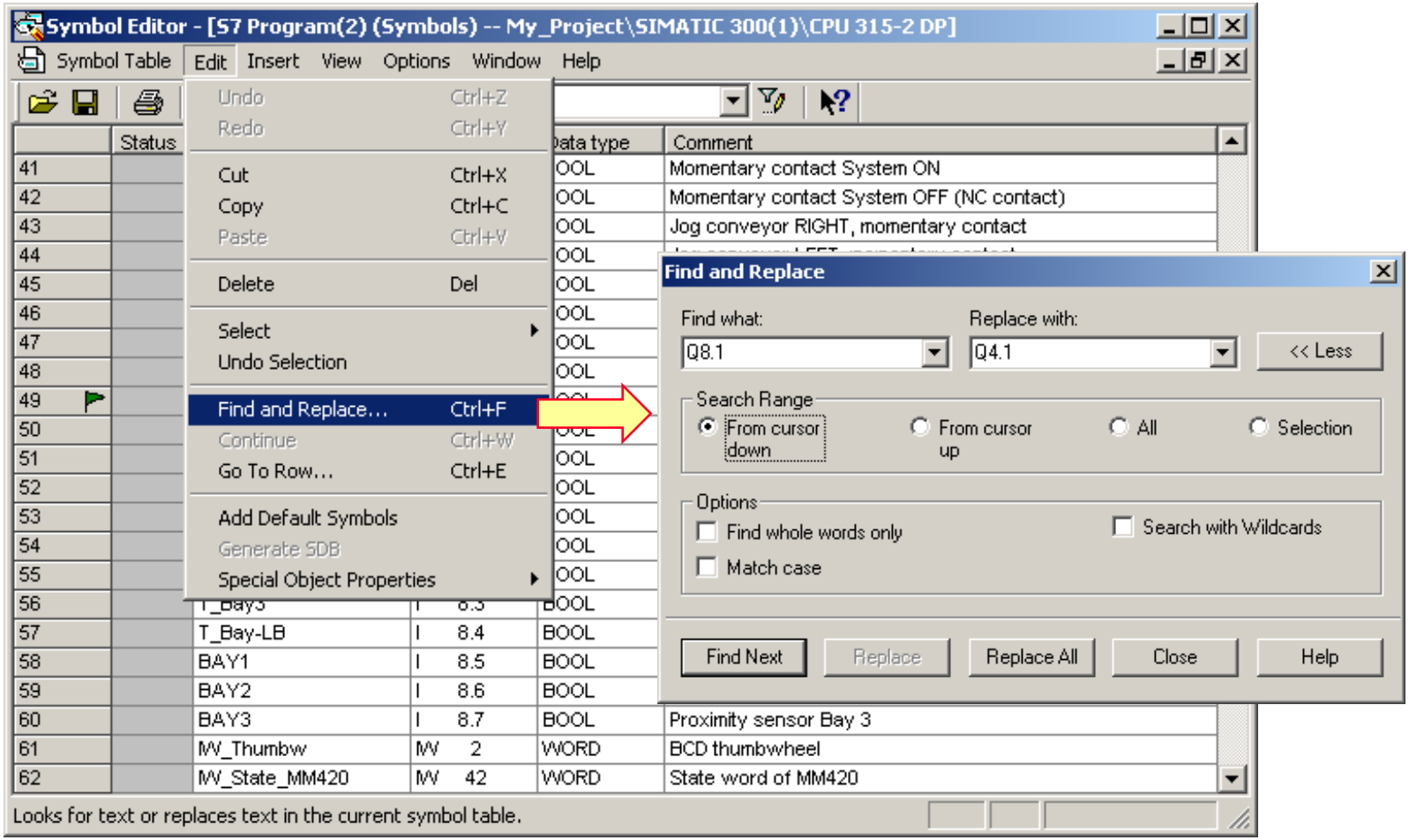
Where are symbols used?	Where are they stored?	With what are they created?
Global Data: - Inputs - Outputs - Bit mem., timers, counters - Peripheral I/O	Symbol Table	Symbol Editor
Local Block Data: - Block parameters - local / temporary data	Declaration part of the block	Program Editor
Jump Labels	Code section of the block	Program Editor
Block Names: - OB - FB - FC - DB - VAT - UDT	Symbol Table	Symbol Editor
DB Components	Declaration part of the DB	Program Editor

The Symbol Table

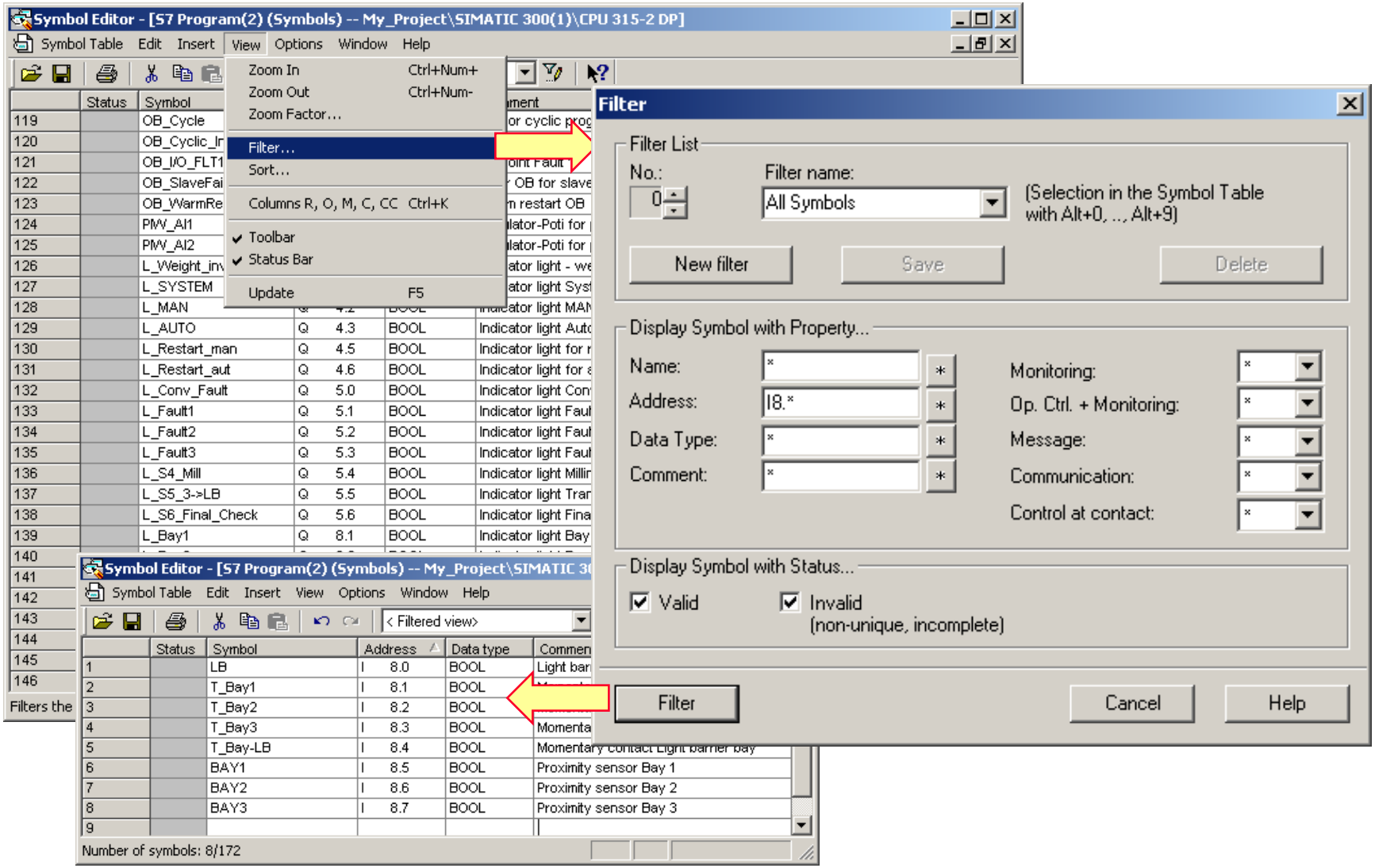


	Status	Symbol	Address	Data type	Comment
41		T_System_ON	I 0.0	BOOL	Momentary contact System ON
42		T_System_OFF	I 0.1	BOOL	Momentary contact System OFF (NC contact)
43		T_Jog_RIGHT	I 0.2	BOOL	Jog conveyor RIGHT, momentary contact
44		T_Jog_LEFT	I 0.3	BOOL	Jog conveyor LEFT, momentary contact
45		S_M/A_ModeSelect	I 0.4	BOOL	Switch - Operating Mode Preselect: '0'=MANUAL, '1'=AUTO
46		T_M/A_Accept	I 0.5	BOOL	Momentary contact, confirm operating mode
47		S_Weight/Quantity	I 0.6	BOOL	0 = Display quantity, 1 = Display weight
48		T_Ackn_WarmRestart	I 0.7	BOOL	Momentary contact to acknowledge warm restart display
49		T_Ackn_Fault	I 1.0	BOOL	Momentary contact Fault acknowledgement
50		S_Fault1	I 1.1	BOOL	Simulate Fault 1, 0=OFF/1=ON
51		S_Fault2	I 1.2	BOOL	Simulate Fault 2, 0=OFF/1=ON
52		S_Fault3	I 1.3	BOOL	Simulate Fault 3, 0=OFF/1=ON
53		LB	I 8.0	BOOL	Light barrier
54		T_Bay1	I 8.1	BOOL	Momentary contact Bay 1
55		T_Bay2	I 8.2	BOOL	Momentary contact Bay 2
56		T_Bay3	I 8.3	BOOL	Momentary contact Bay 3
57		T_Bay-LB	I 8.4	BOOL	Momentary contact Light barrier bay
58		BAY1	I 8.5	BOOL	Proximity sensor Bay 1
59		BAY2	I 8.6	BOOL	Proximity sensor Bay 2
60		BAY3	I 8.7	BOOL	Proximity sensor Bay 3
61		MW_Thumbw	MW 2	WORD	BCD thumbwheel
62		MW_State_MM420	MW 42	WORD	State word of MM420

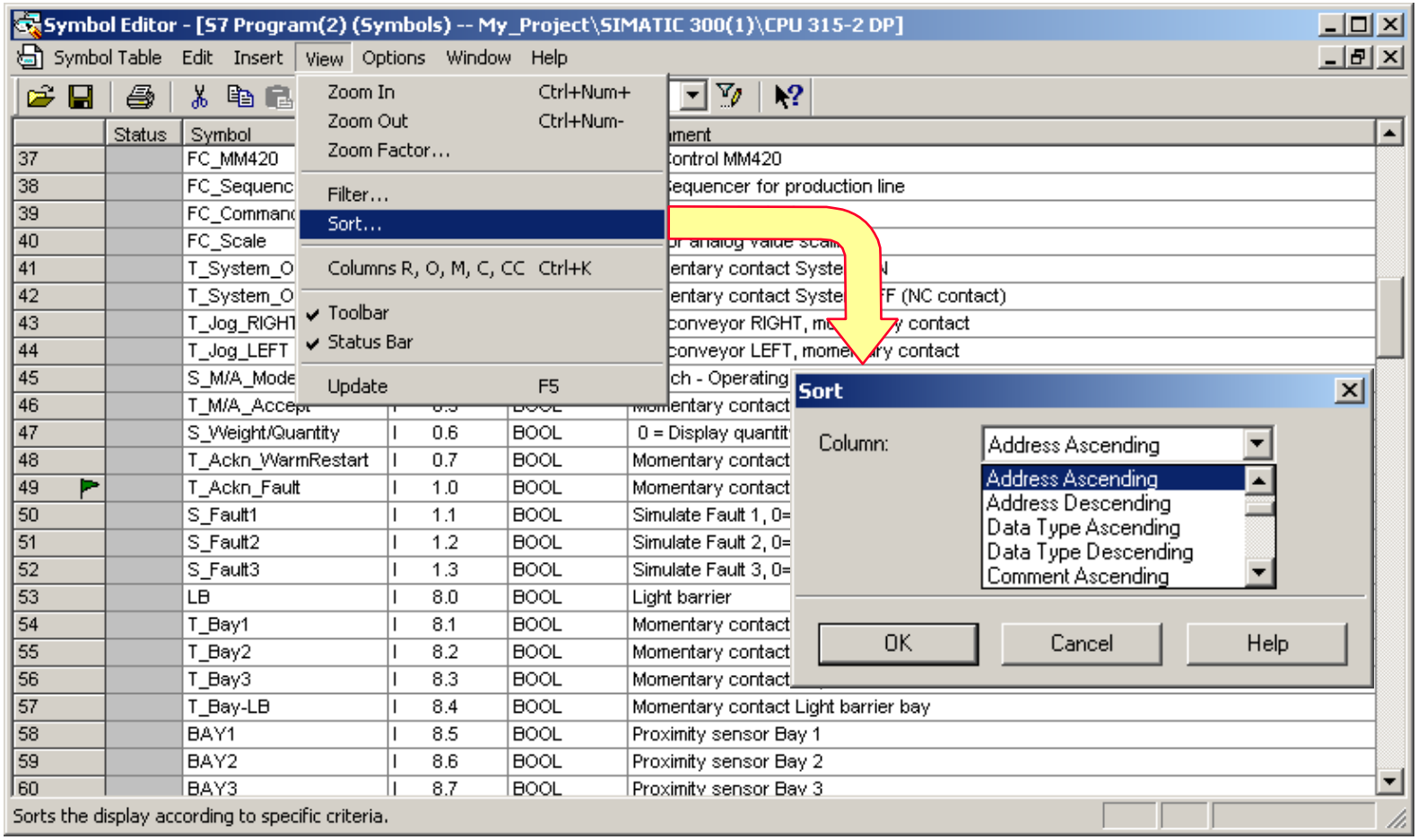
Edit: Find and Replace



View: Filter

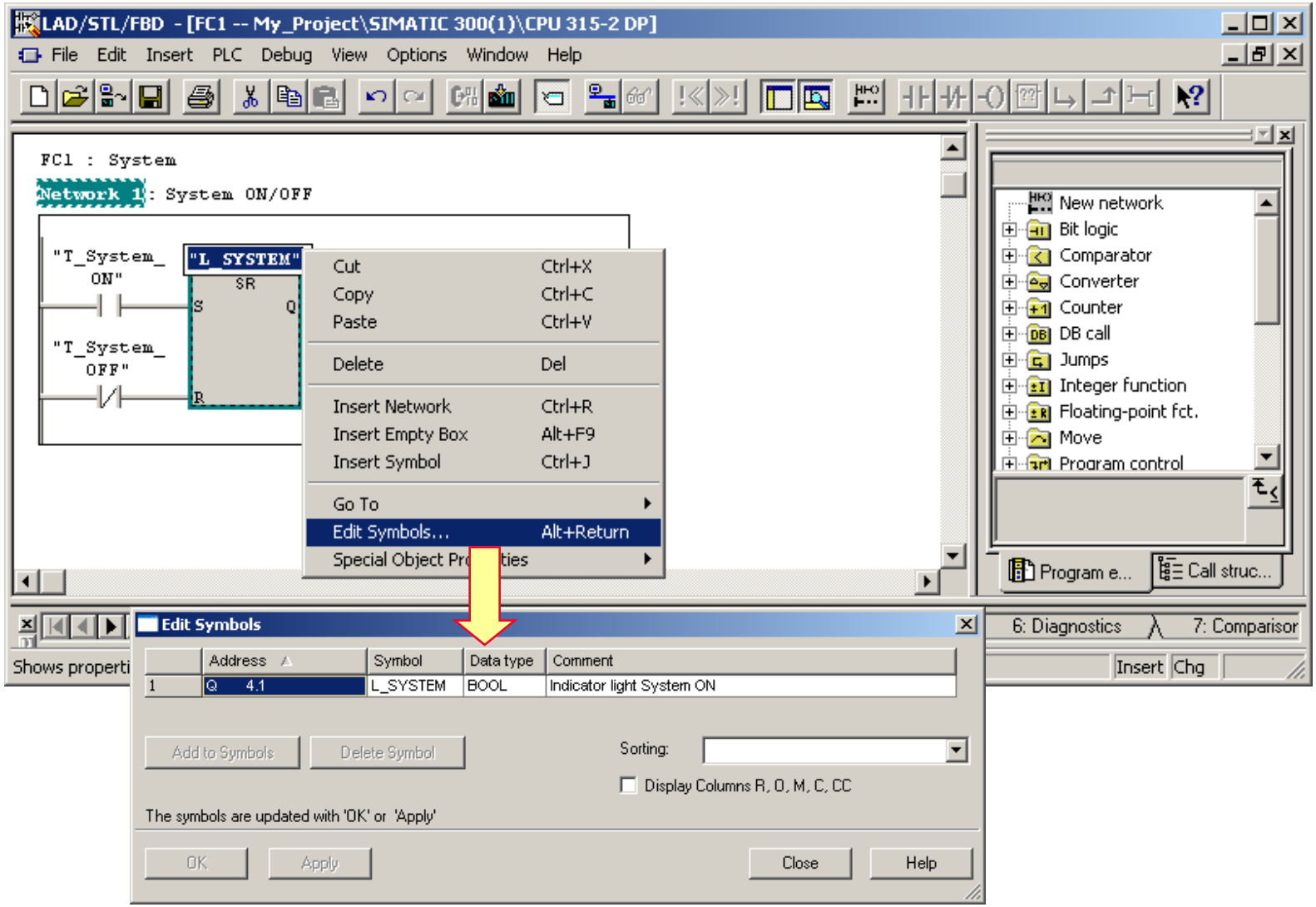


View: Sort

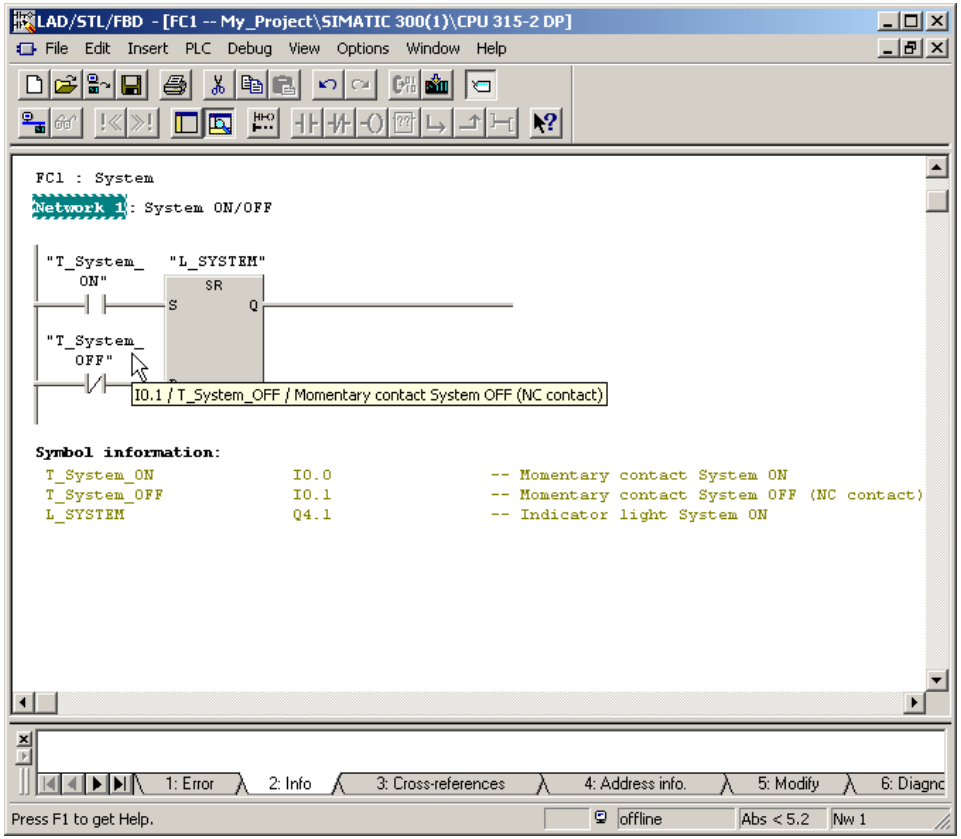


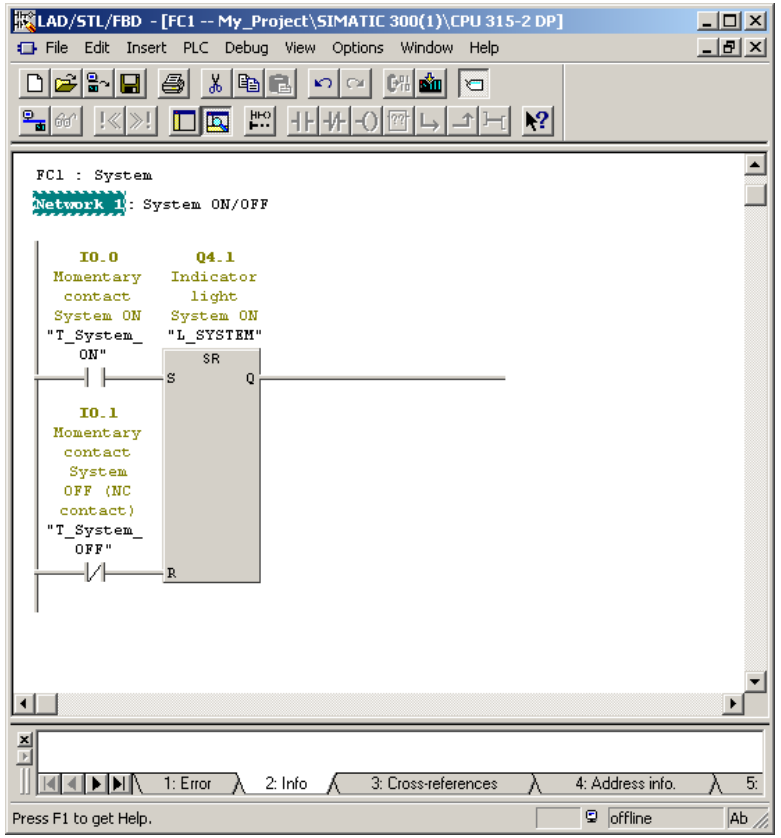
Date: 2/4/2012
File: PRO1_04E.8

Editing Symbols in the LAD/STL/FBD Editor

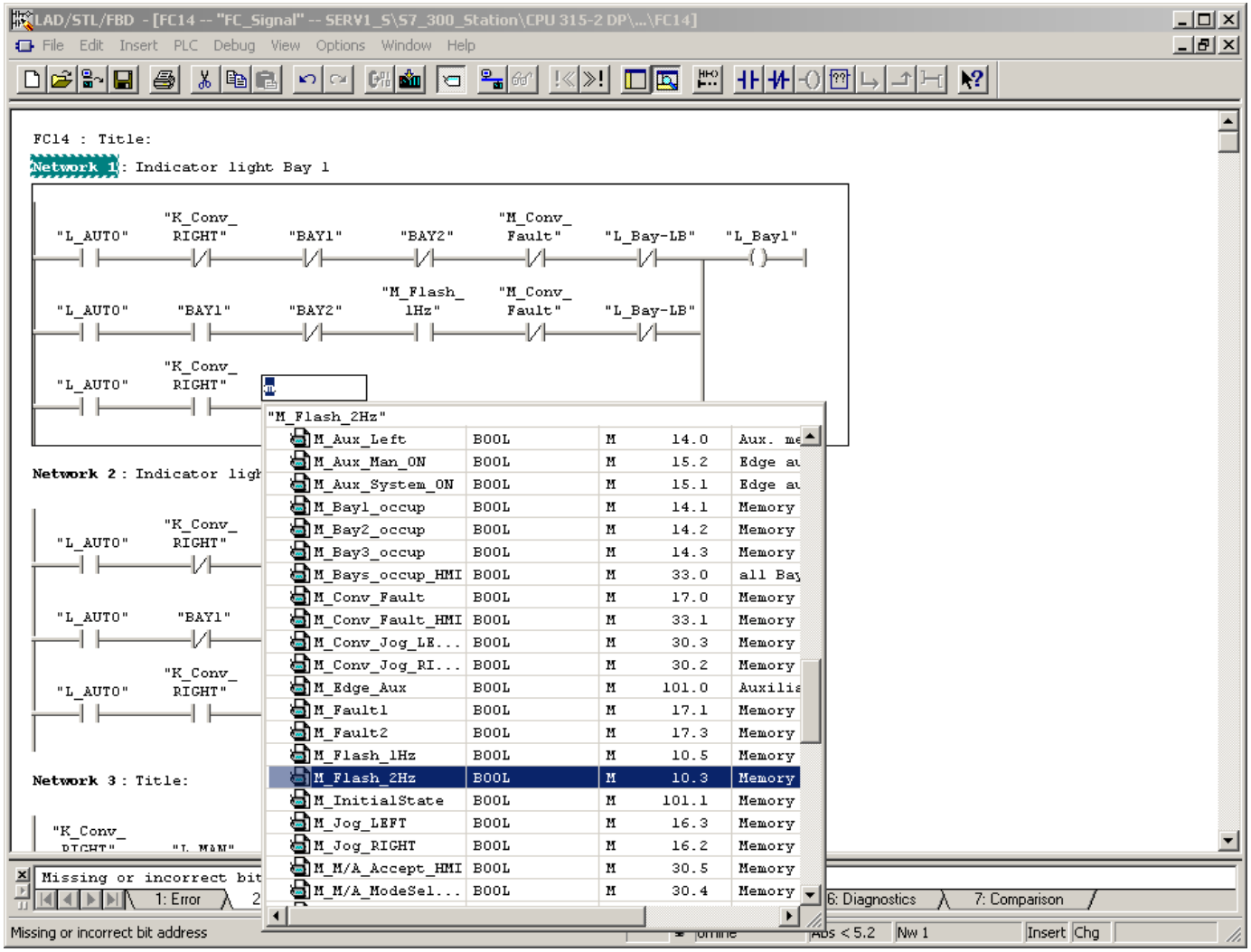


Symbol Information in the LAD/STL/FBD Editor

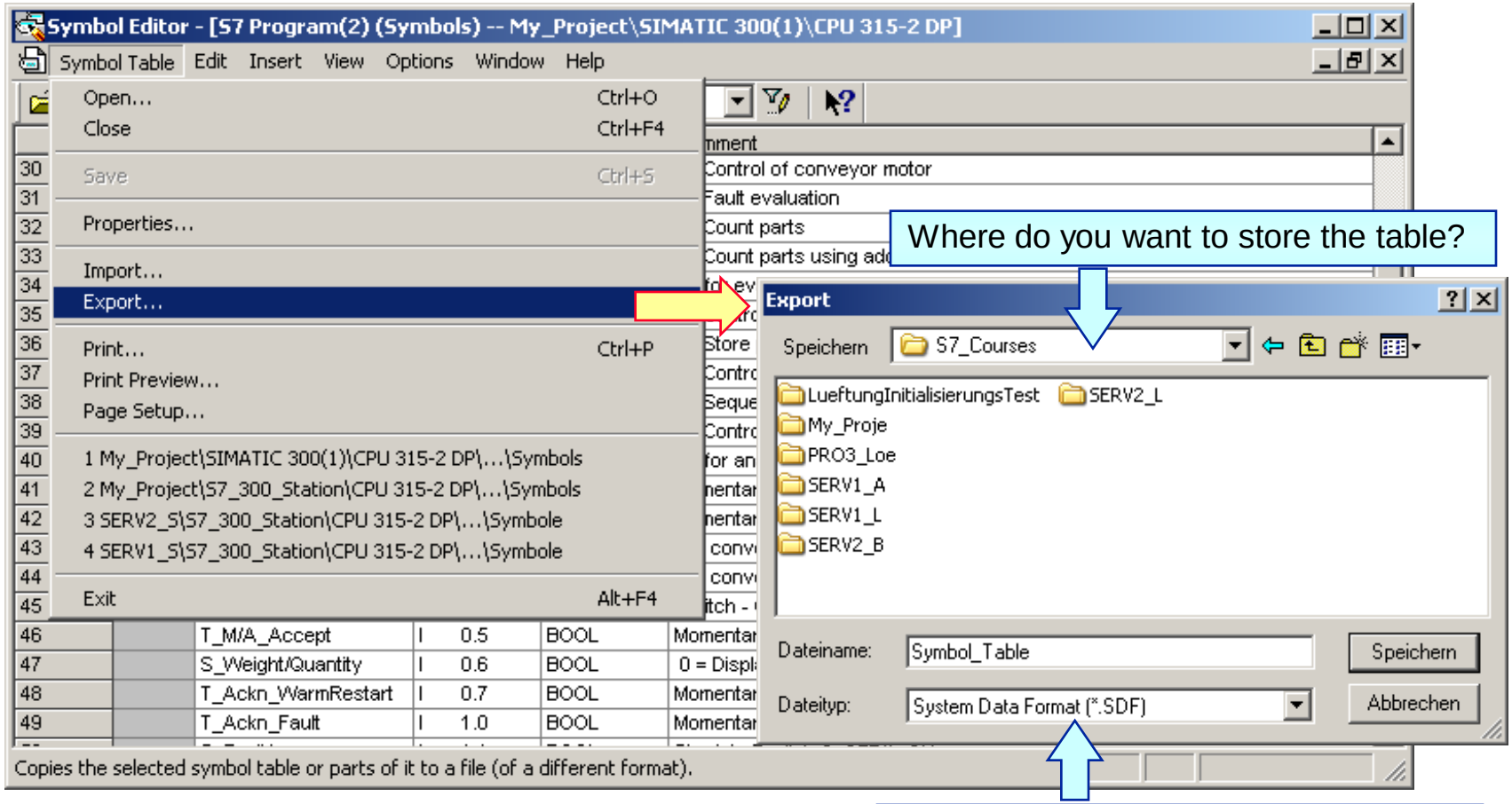




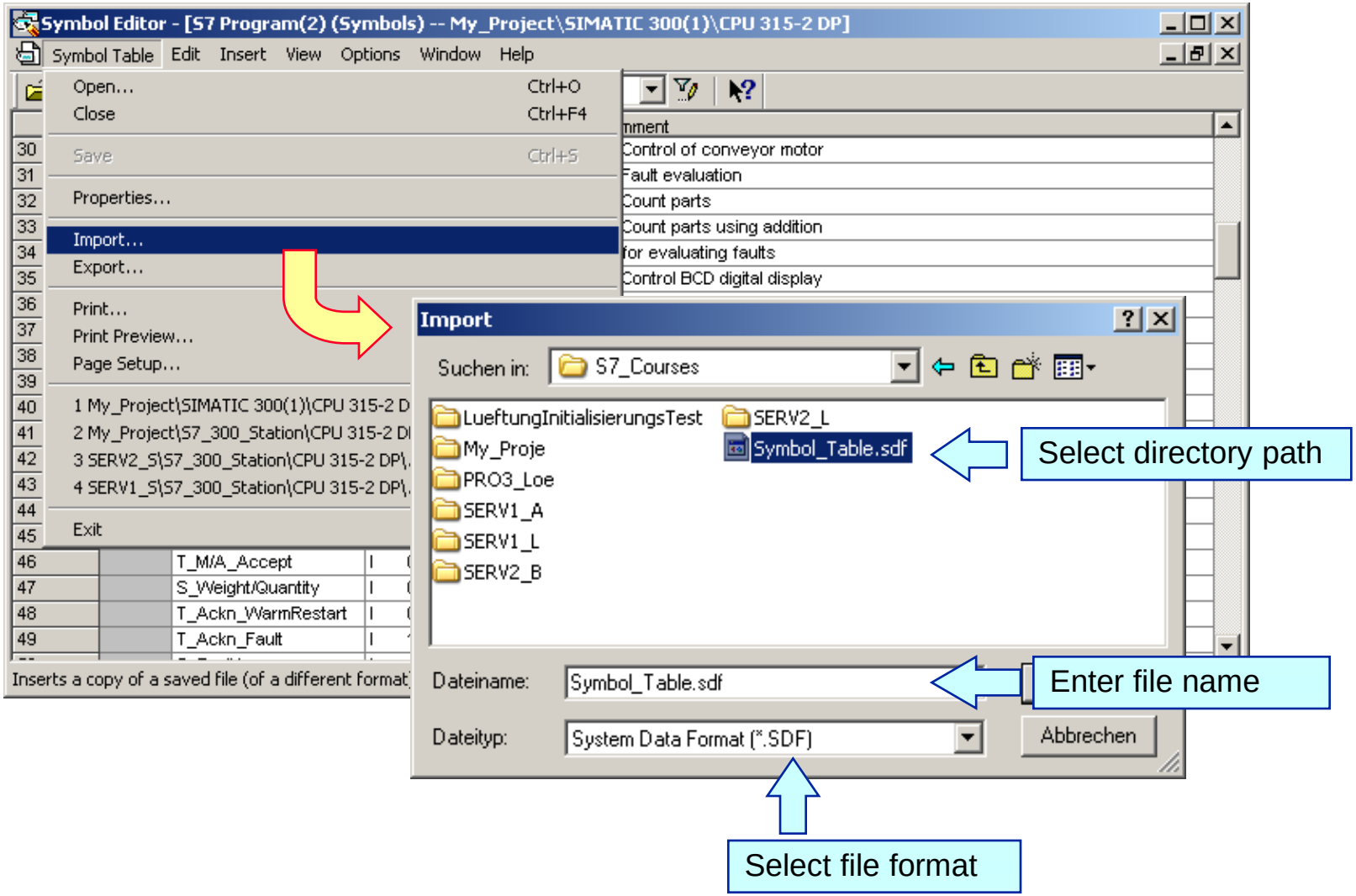
Symbol Selection in LAD/FBD



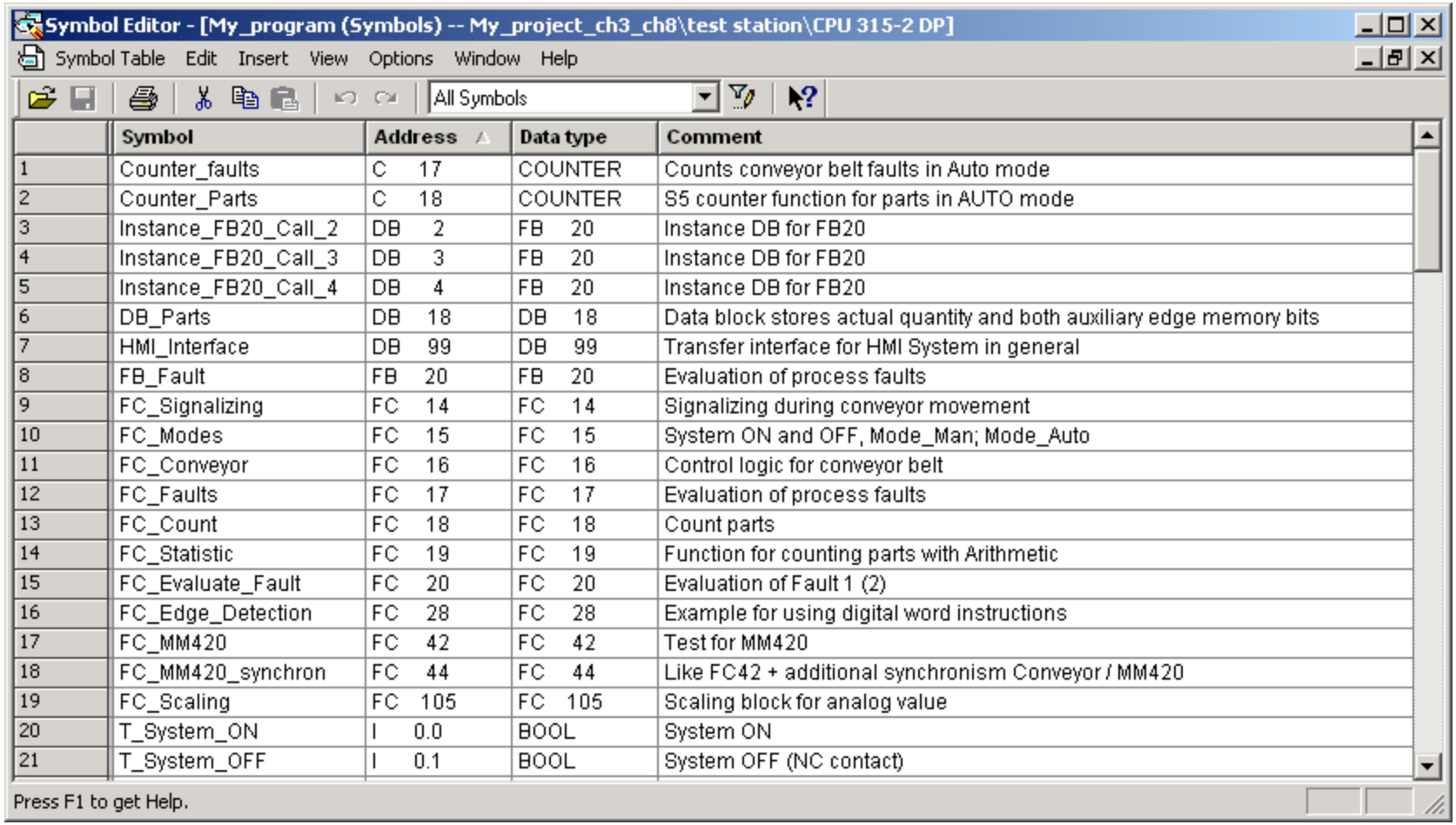
Symbol Table: Export



Symbol Table: Import

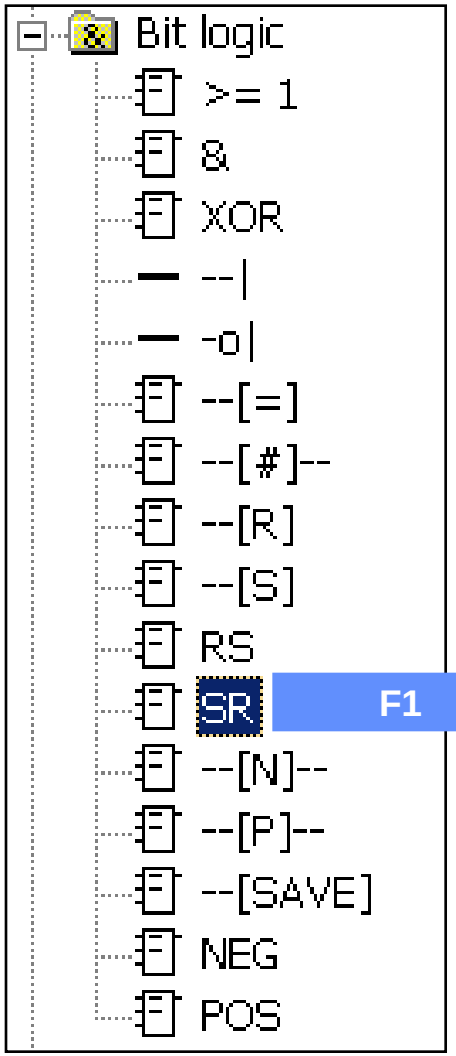


Exercise 1: Importing a Symbol Table



	Symbol	Address	Data type	Comment
1	Counter_faults	C 17	COUNTER	Counts conveyor belt faults in Auto mode
2	Counter_Parts	C 18	COUNTER	S5 counter function for parts in AUTO mode
3	Instance_FB20_Call_2	DB 2	FB 20	Instance DB for FB20
4	Instance_FB20_Call_3	DB 3	FB 20	Instance DB for FB20
5	Instance_FB20_Call_4	DB 4	FB 20	Instance DB for FB20
6	DB_Parts	DB 18	DB 18	Data block stores actual quantity and both auxiliary edge memory bits
7	HMI_Interface	DB 99	DB 99	Transfer interface for HMI System in general
8	FB_Fault	FB 20	FB 20	Evaluation of process faults
9	FC_Signalizing	FC 14	FC 14	Signalizing during conveyor movement
10	FC_Modes	FC 15	FC 15	System ON and OFF, Mode_Man; Mode_Auto
11	FC_Conveyor	FC 16	FC 16	Control logic for conveyor belt
12	FC_Faults	FC 17	FC 17	Evaluation of process faults
13	FC_Count	FC 18	FC 18	Count parts
14	FC_Statistic	FC 19	FC 19	Function for counting parts with Arithmetic
15	FC_Evaluate_Fault	FC 20	FC 20	Evaluation of Fault 1 (2)
16	FC_Edge_Detection	FC 28	FC 28	Example for using digital word instructions
17	FC_MM420	FC 42	FC 42	Test for MM420
18	FC_MM420_synchron	FC 44	FC 44	Like FC42 + additional synchronism Conveyor / MM420
19	FC_Scaling	FC 105	FC 105	Scaling block for analog value
20	T_System_ON	I 0.0	BOOL	System ON
21	T_System_OFF	I 0.1	BOOL	System OFF (NC contact)

Binary Operations



F1

Help on Function Block Diagram

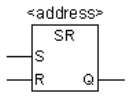
File Edit Bookmark Options Help

Contents | Index | Back | Print | << | >>

Help on STEP 7 | Glossary

SR : Set_Reset Flip Flop

Symbol



Parameter	Data Type	Memory Area	Description
<address>	BOOL	I, Q, M, D, L	The address specifies which bit will be set or reset.
S	BOOL	I, Q, M, D, L, T, C	Set instruction enabled
R	BOOL	I, Q, M, D, L, T, C	Reset instruction enabled
Q	BOOL	I, Q, M, D, L	Signal state of <address>

Description

The **Set_Reset Flip Flop** instruction executes Set (S) or Reset (R) instructions only when the RLO is 1. An RLO of 0 has no effect on these instructions, the address specified in the instruction remains unchanged.

Set_Reset Flip Flop is set when the signal state at input S is 1 and the signal state at input R is 0. If input S is 0 and input R is 1, the flip flop is reset. If the RLO at both inputs is 1 the flip flop is reset.

The **Set_Reset Flip Flop** instruction is affected by the Master Control Relay (MCR). For more detailed information about how the MCR functions, refer to MCR on/off.

Status Word

	BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
writes	-	-	-	-	-	X	X	X	1

Objectives

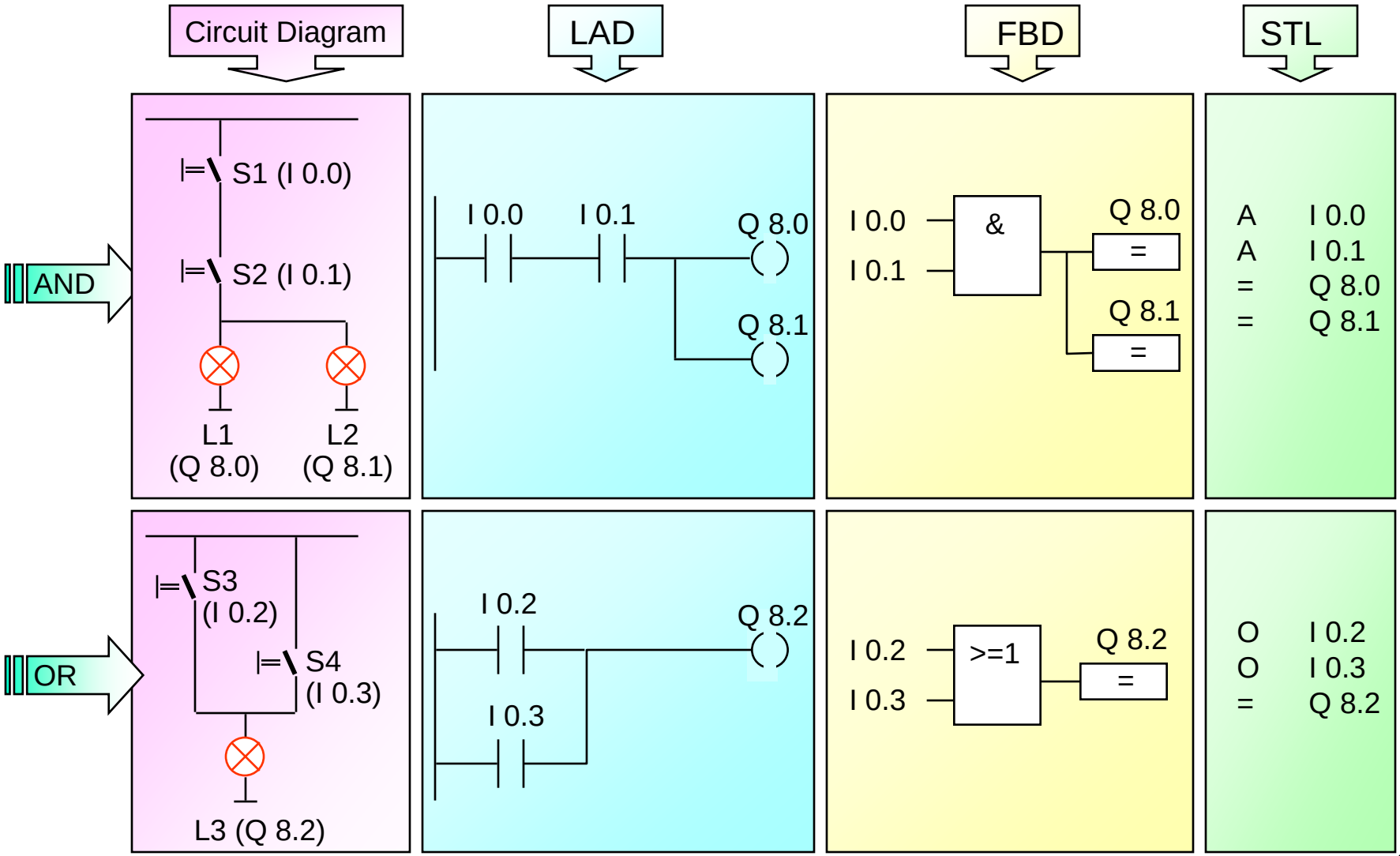
Upon completion of the chapter the participant will …

... understand the difference between ‘real‘ connected NC contacts and NO contacts, and programmed symbols

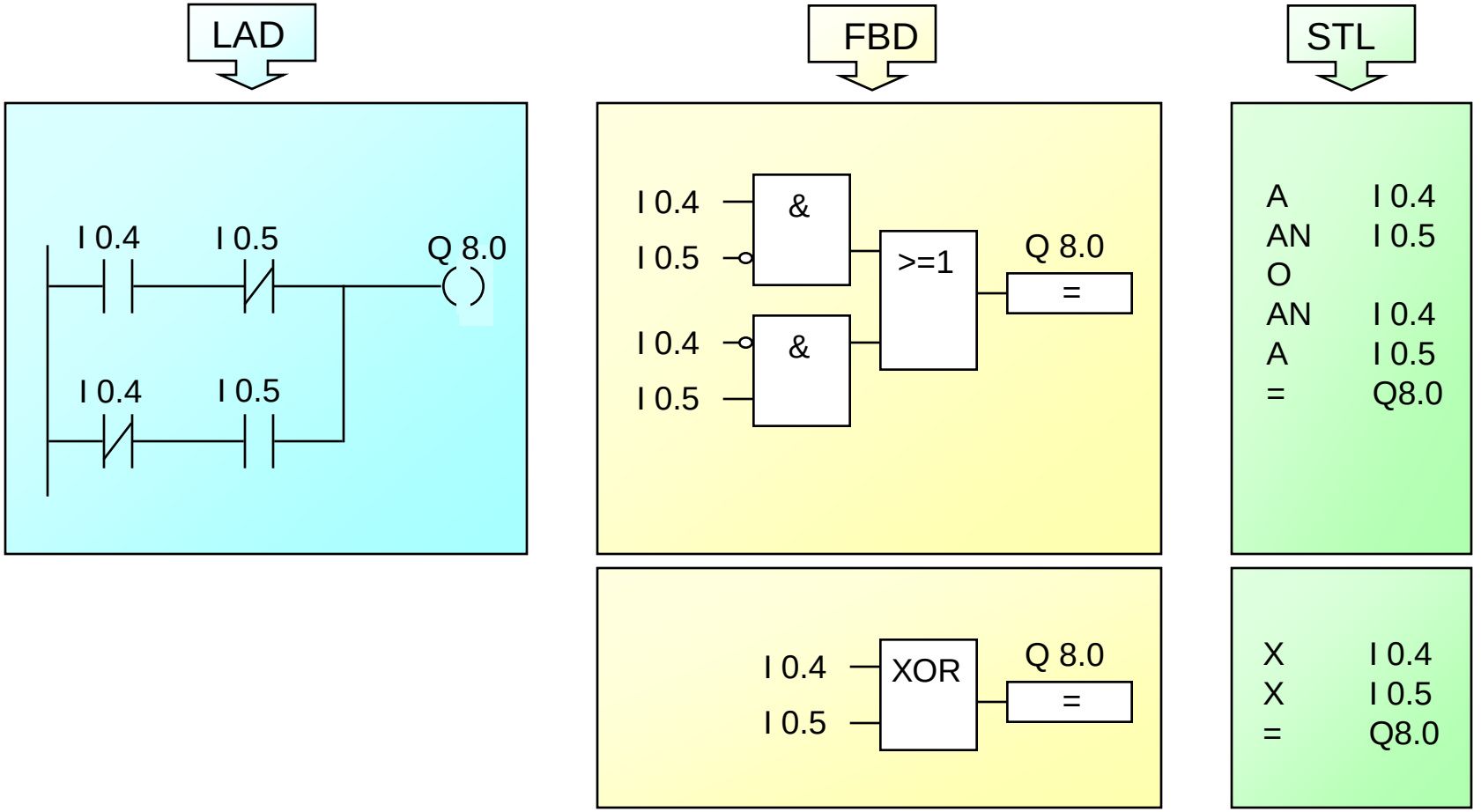
... be able to explain the terms Result of Logic Operation (RLO), Status (STAT) and First Check

... be able to program basic binary logic operations

Binary Logic Operations: AND, OR



Binary Logic Operations: Exclusive OR (XOR)

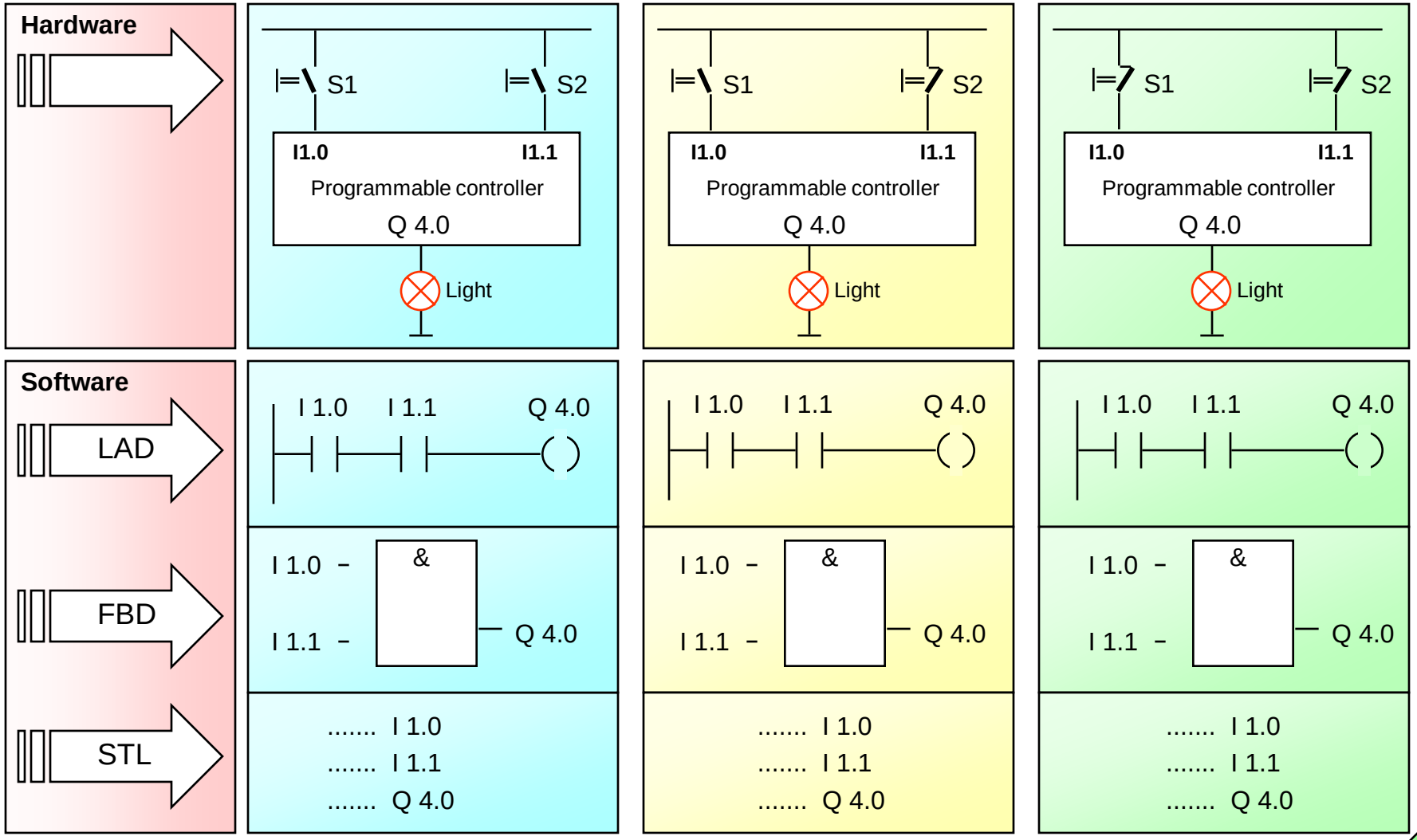


Sensors and Symbols

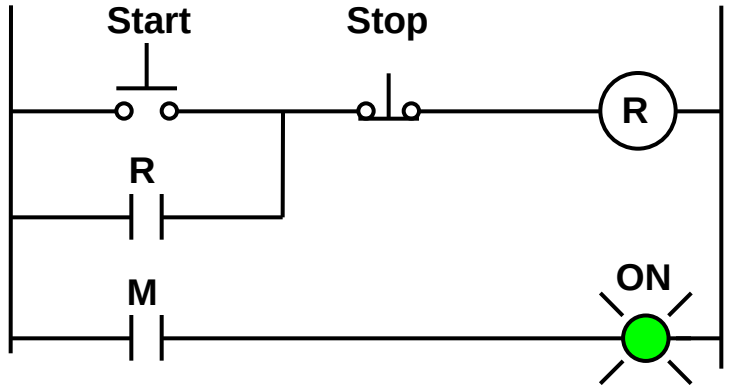
Process			Interpretation in the PLC Program				
The sensor is a ...	The sensor is ...	Voltage present at input?	Signal state at input	Check for signal state “1”		Check for signal state “0”	
				Symbol / Instruction	Result of check	Symbol / Instruction	Result of check
NO contact	activated	yes	1	LAD: “NO contact” FBD: & STL: A I x.y	“Yes” 1	LAD: “NC contact” FBD: & STL: AN I x.y	“No” 0
	not activated	no	0		“No” 0		“Yes” 1
NC contact	activated	no	0		“No” 0		“Yes” 1
	not activated	yes	1		“Yes” 1		“No” 0

Exercise 1: Sensors and Symbols

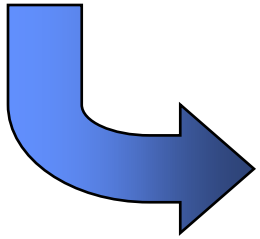
Task: In all three examples, the light should be on when S1 is activated and S2 is not activated!

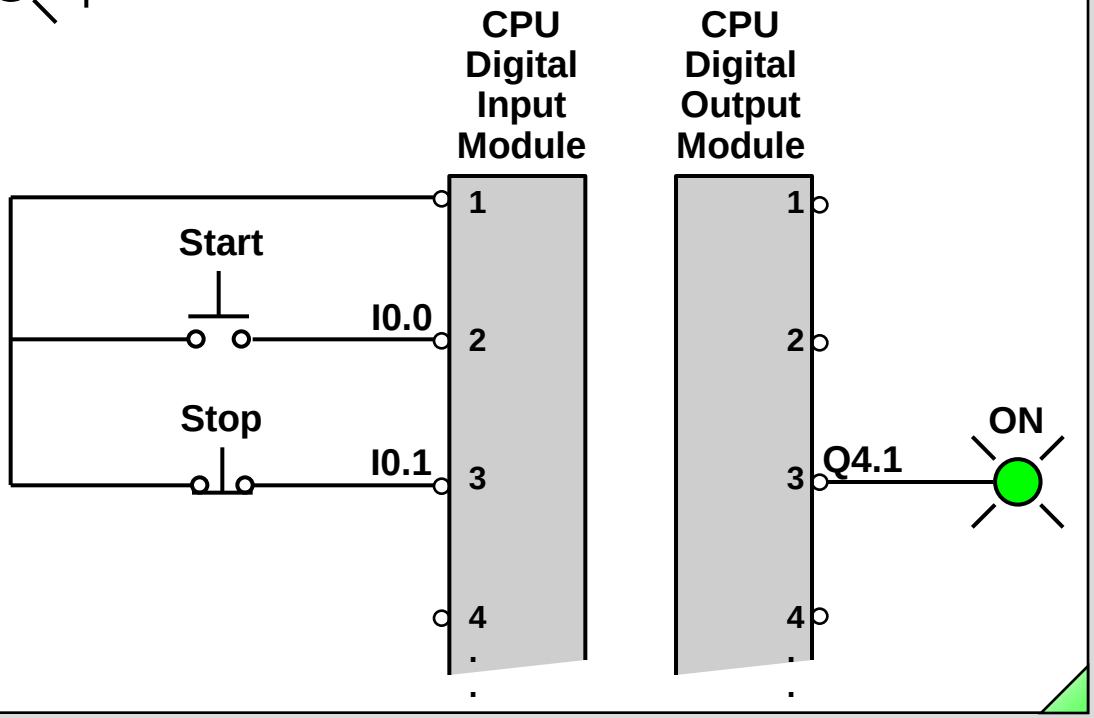


Exercise 2: Add Conveyor "Master" Control Logic

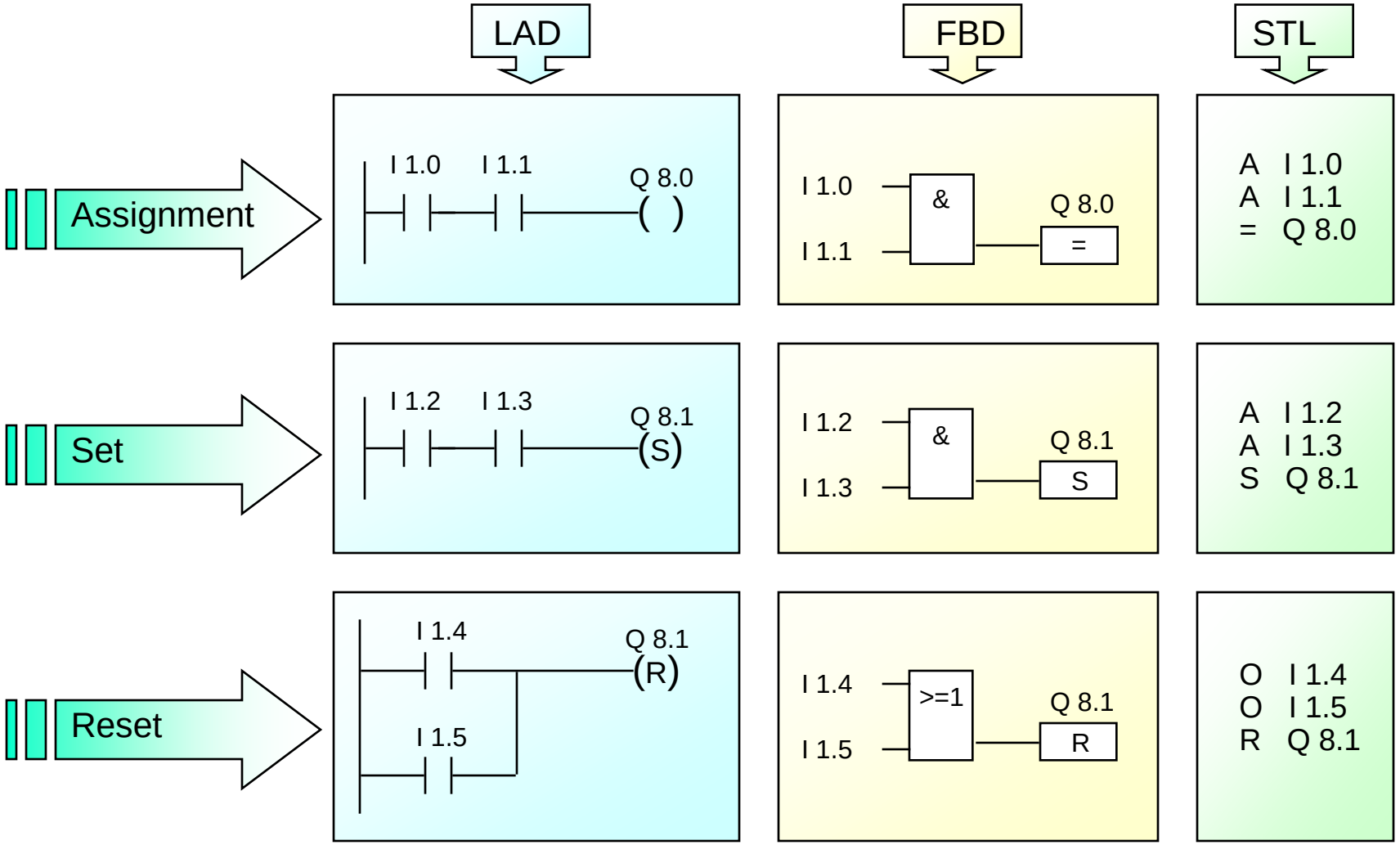


Hard Wire Circuit

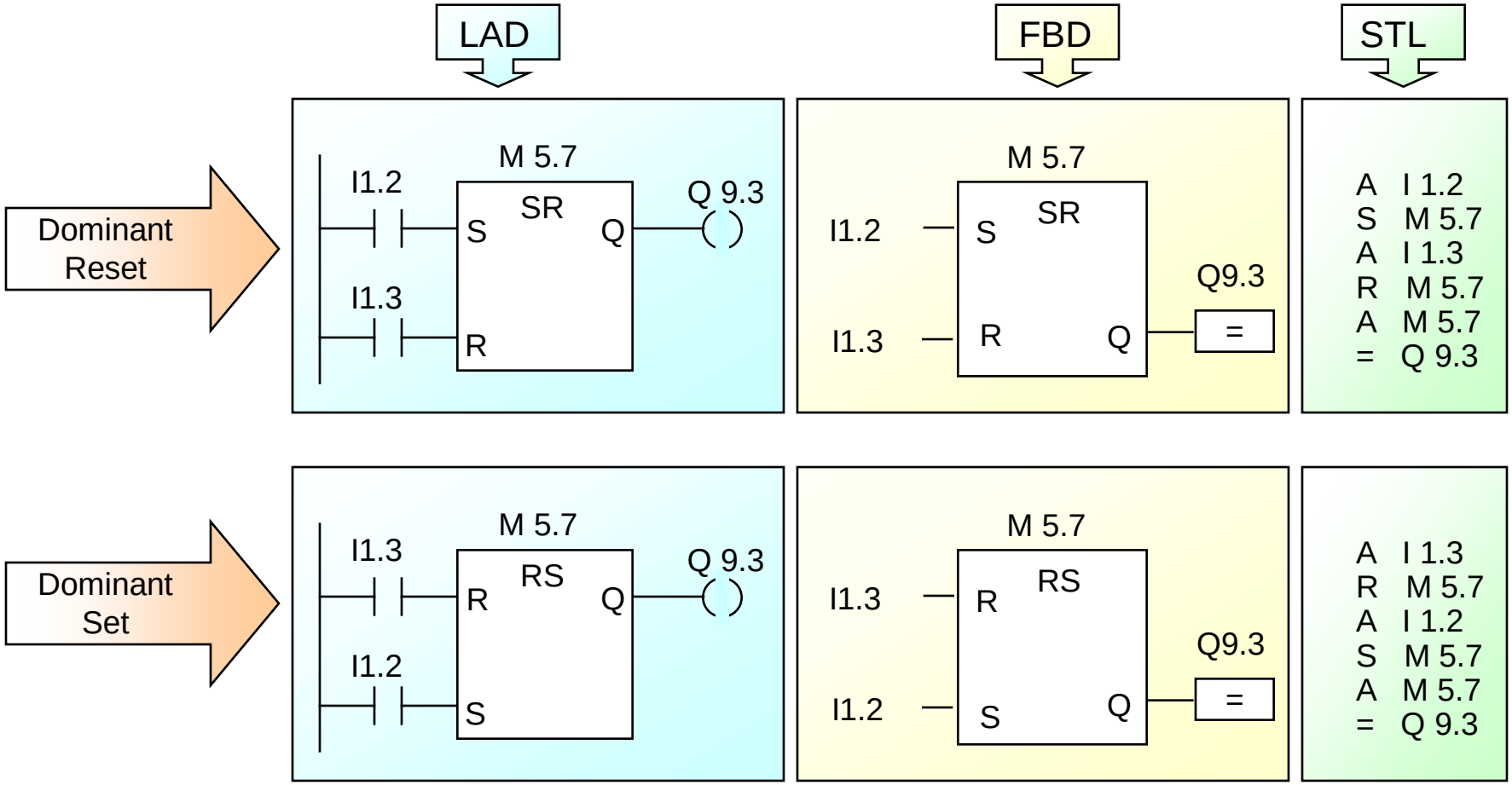




Assignment, Setting, Resetting

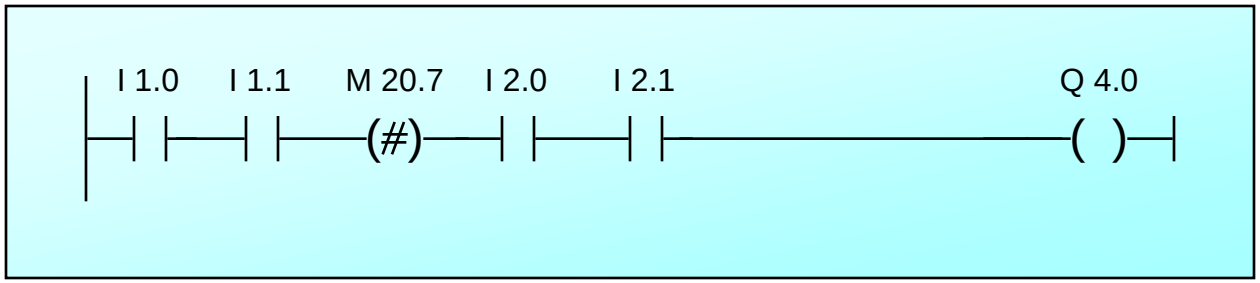


Setting and Resetting a Flip Flop



Midline Output Coil

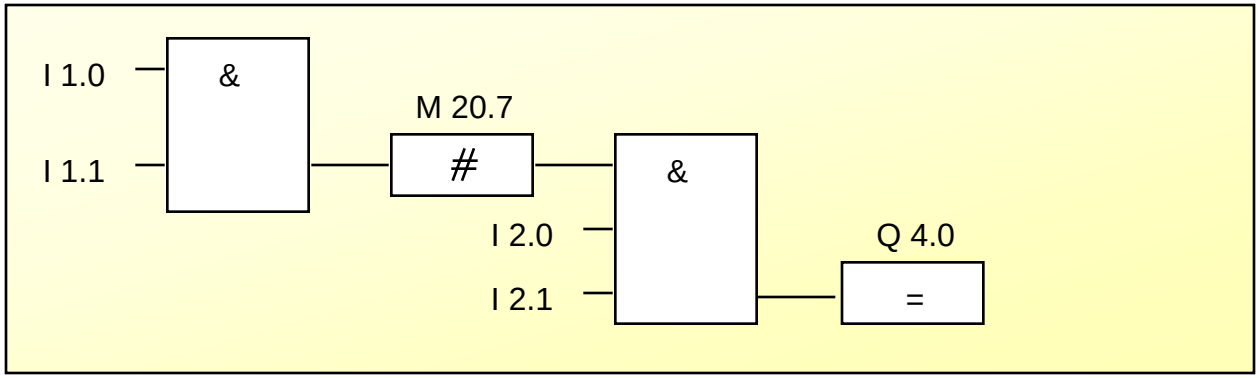
LAD



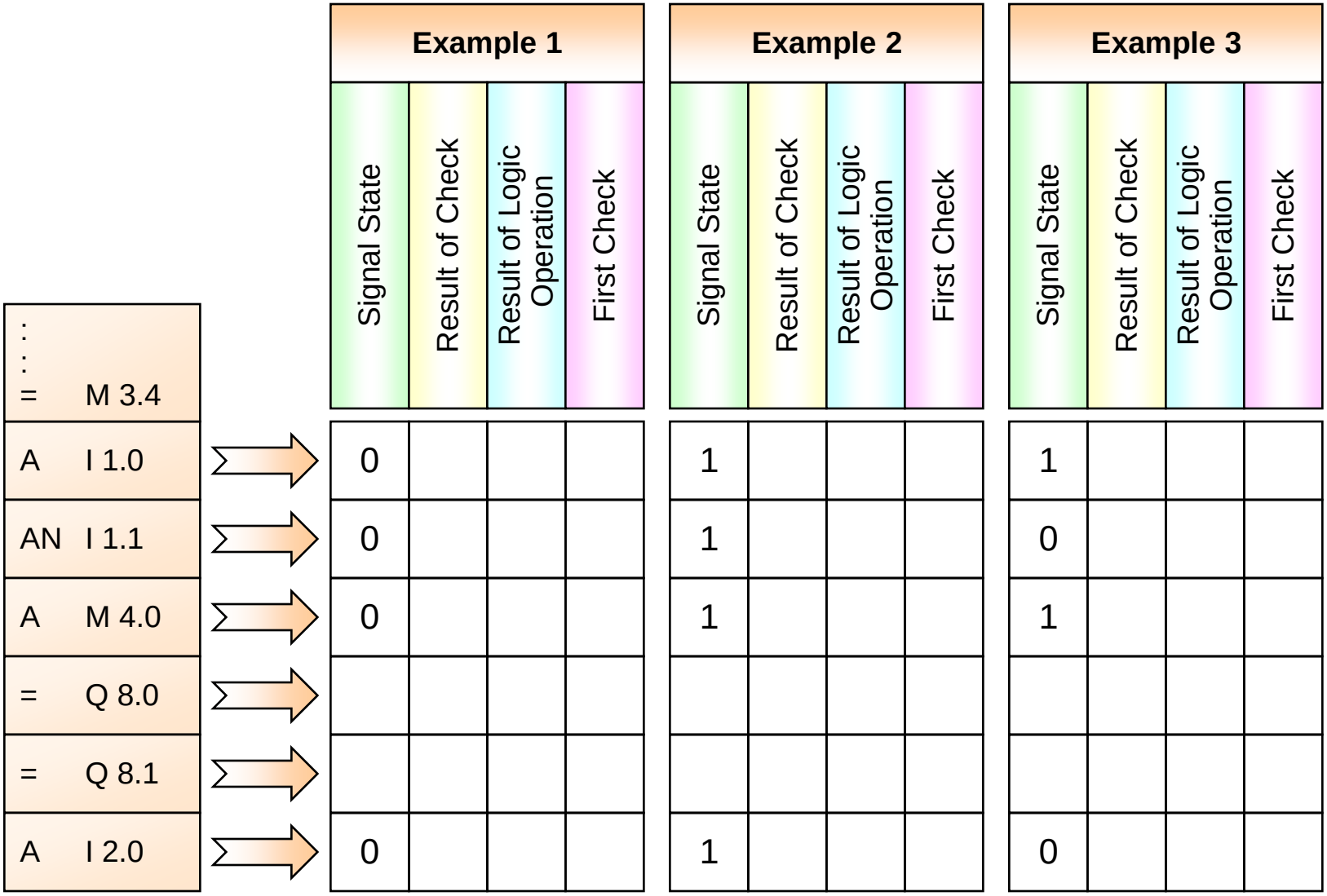
STL

```
A   I  1.0
A   I  1.1
=   M  20.7
A   M  20.7
A   I  2.0
A   I  2.1
=   Q  4.0
```

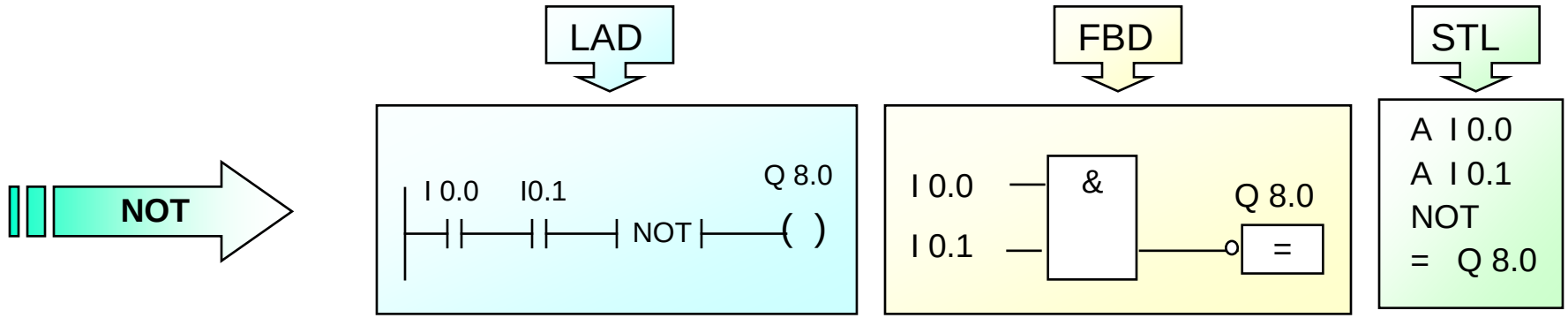
FBD



Result of Logic Operation, First Check, and Examples



Instructions that Affect the RLO



CLR — not available — not available

SET — not available — not available

Examples:

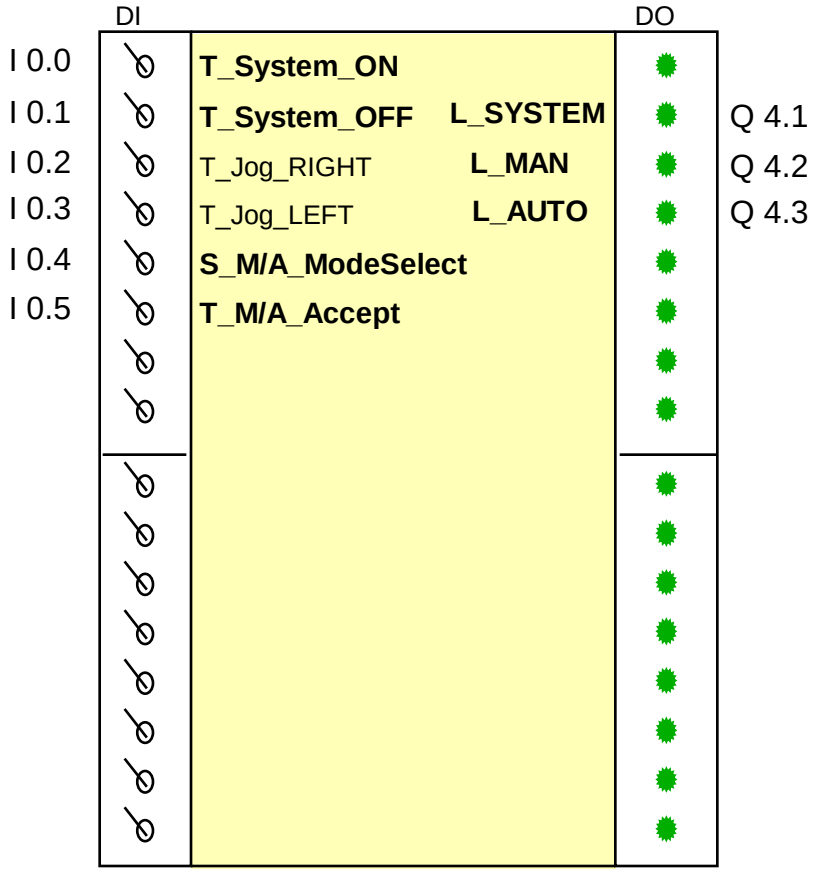
STAT 0 – Bit memory

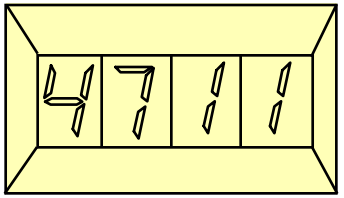
```
CLR
=  M 0.0
```

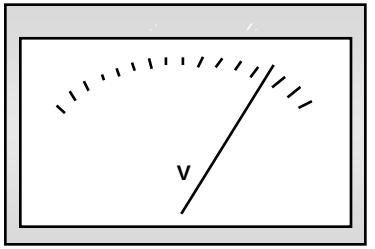
STAT 1 – Bit memory

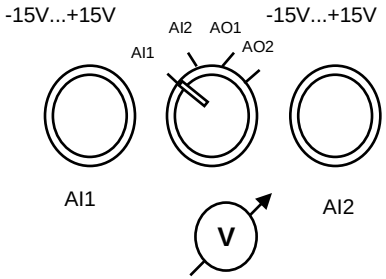
```
SET
=  M 0.1
```

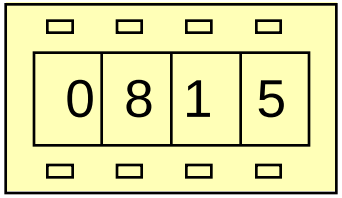
Exercise 3: Programming the Operating Mode Section (FC 15)

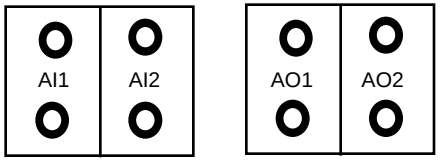






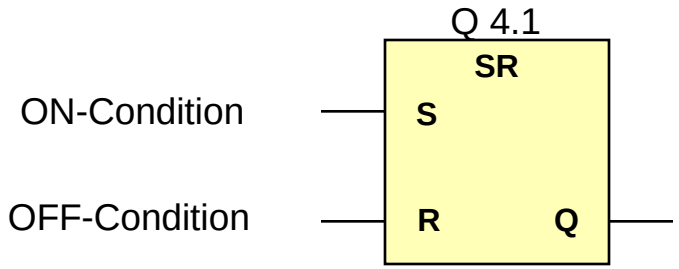




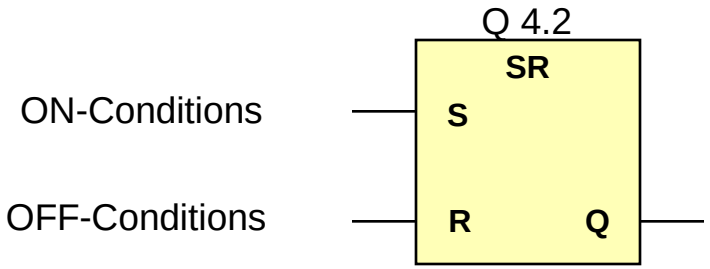


Exercise 3: "FC_Mode" (FC 15), Solution Hints

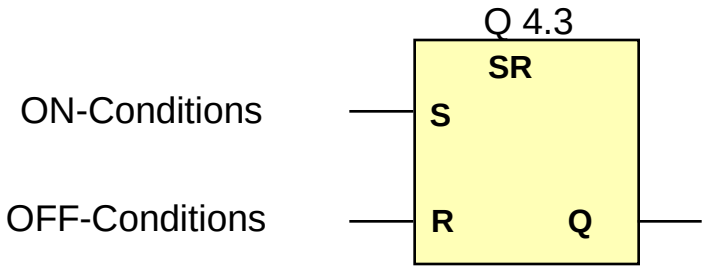
Network 1: Indicator Light System ON



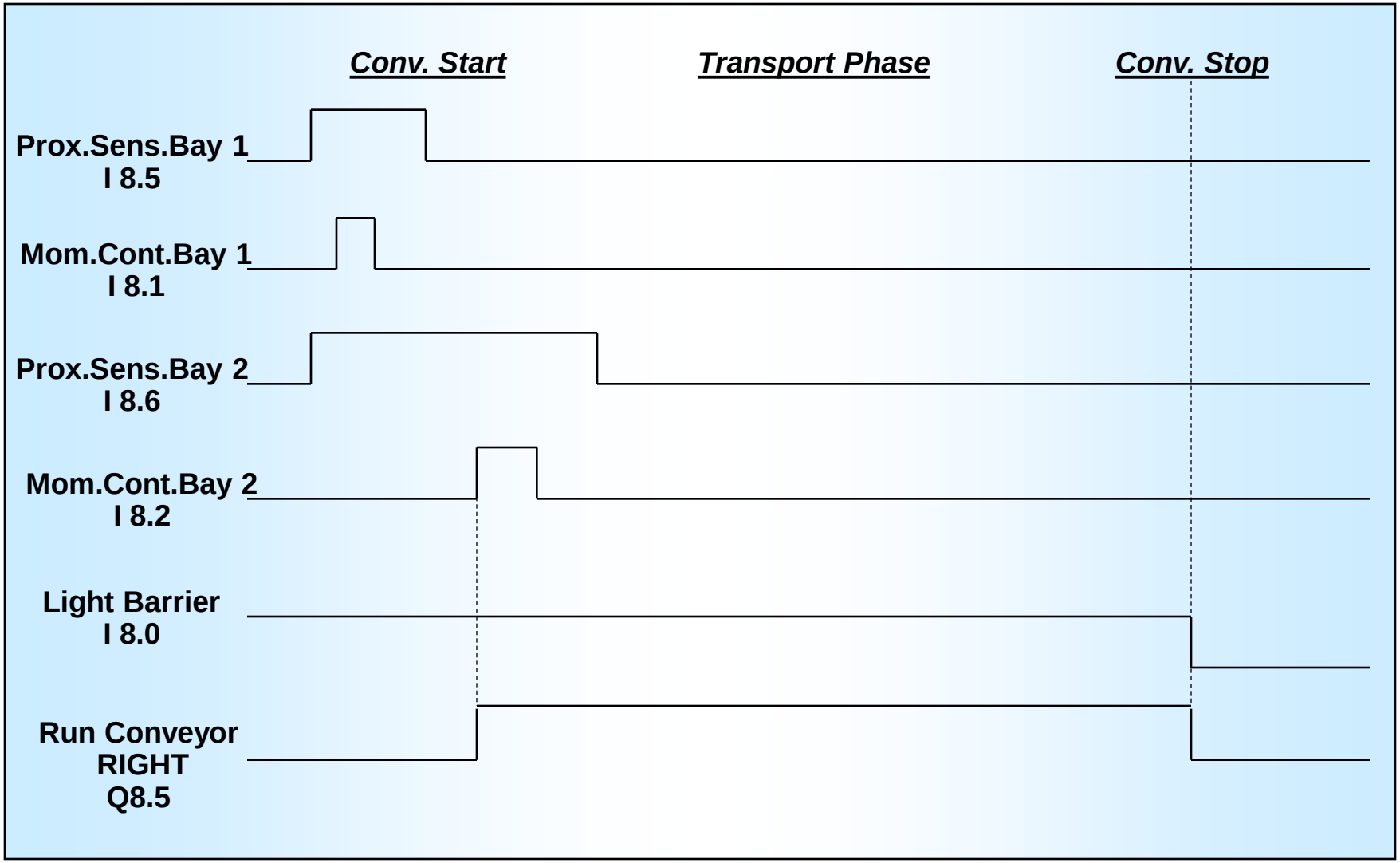
Network 2: Indicator Light MAN Mode



Network 3: Indicator Light AUTO Mode

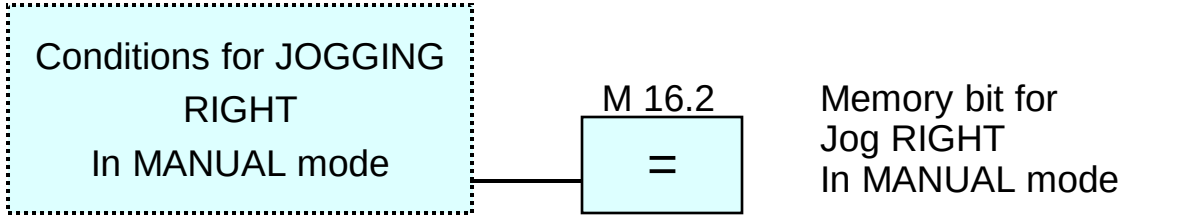


Exercise 4: Conveyor in AUTO Mode (Expanding FC 16)

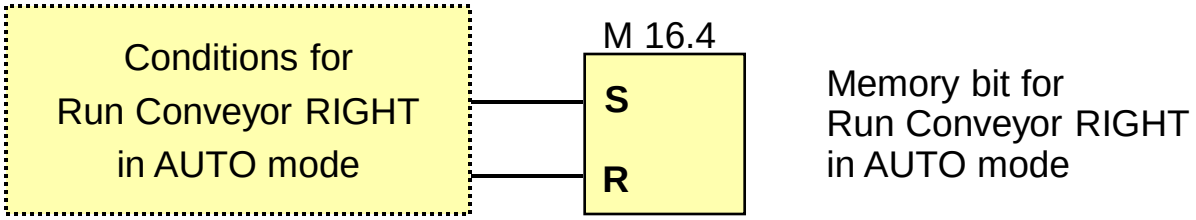


Exercise 4: "FC_ConvMotor" (FC 16), Solution Hints

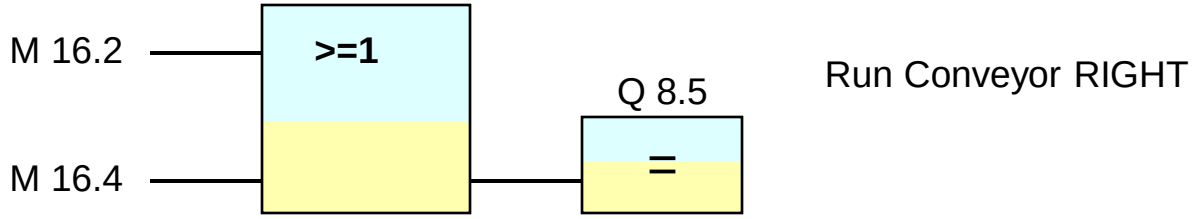
Network x — Jog RIGHT in MANUAL mode



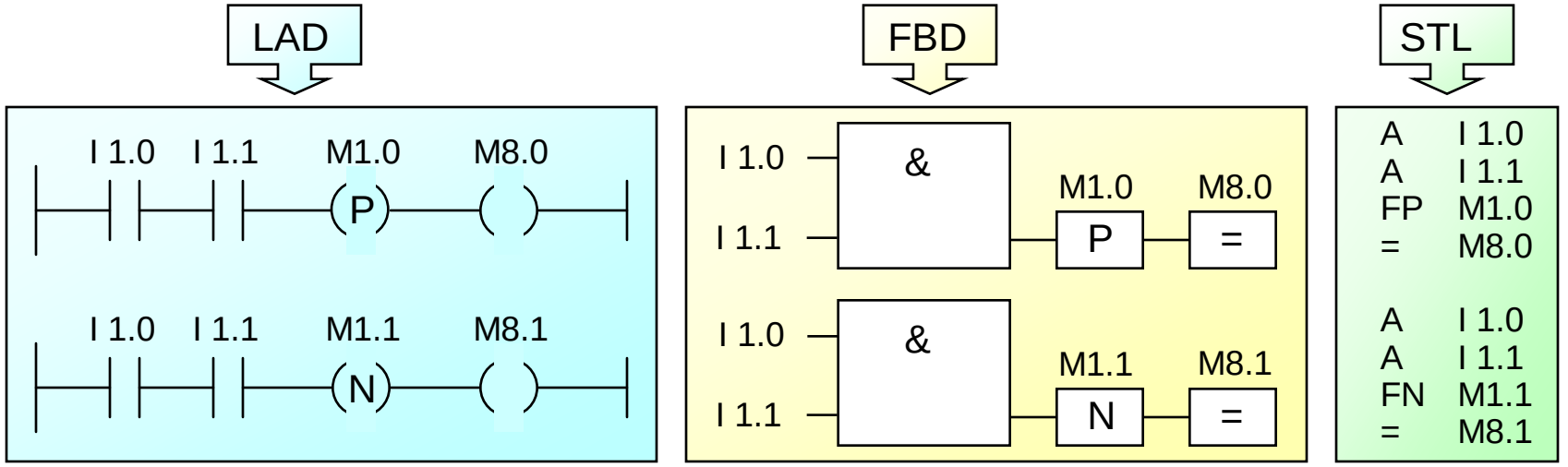
Network y — Run Conveyor RIGHT in AUTO mode

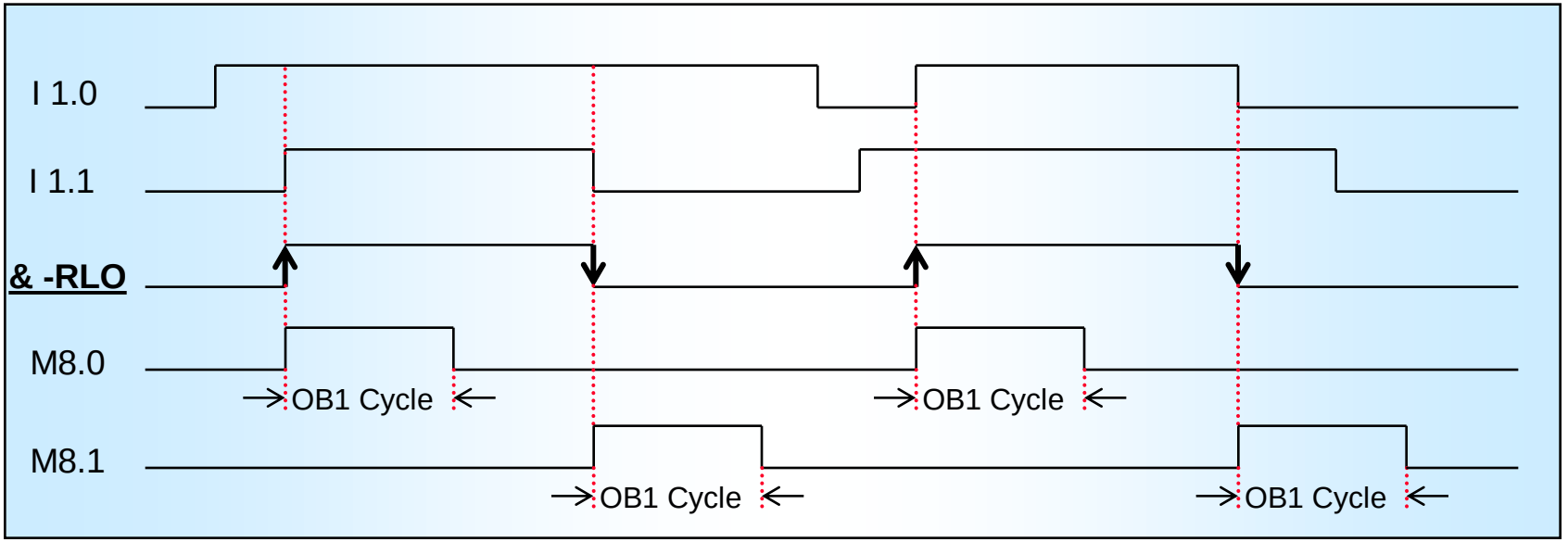


Network z — Control Run Conveyor RIGHT

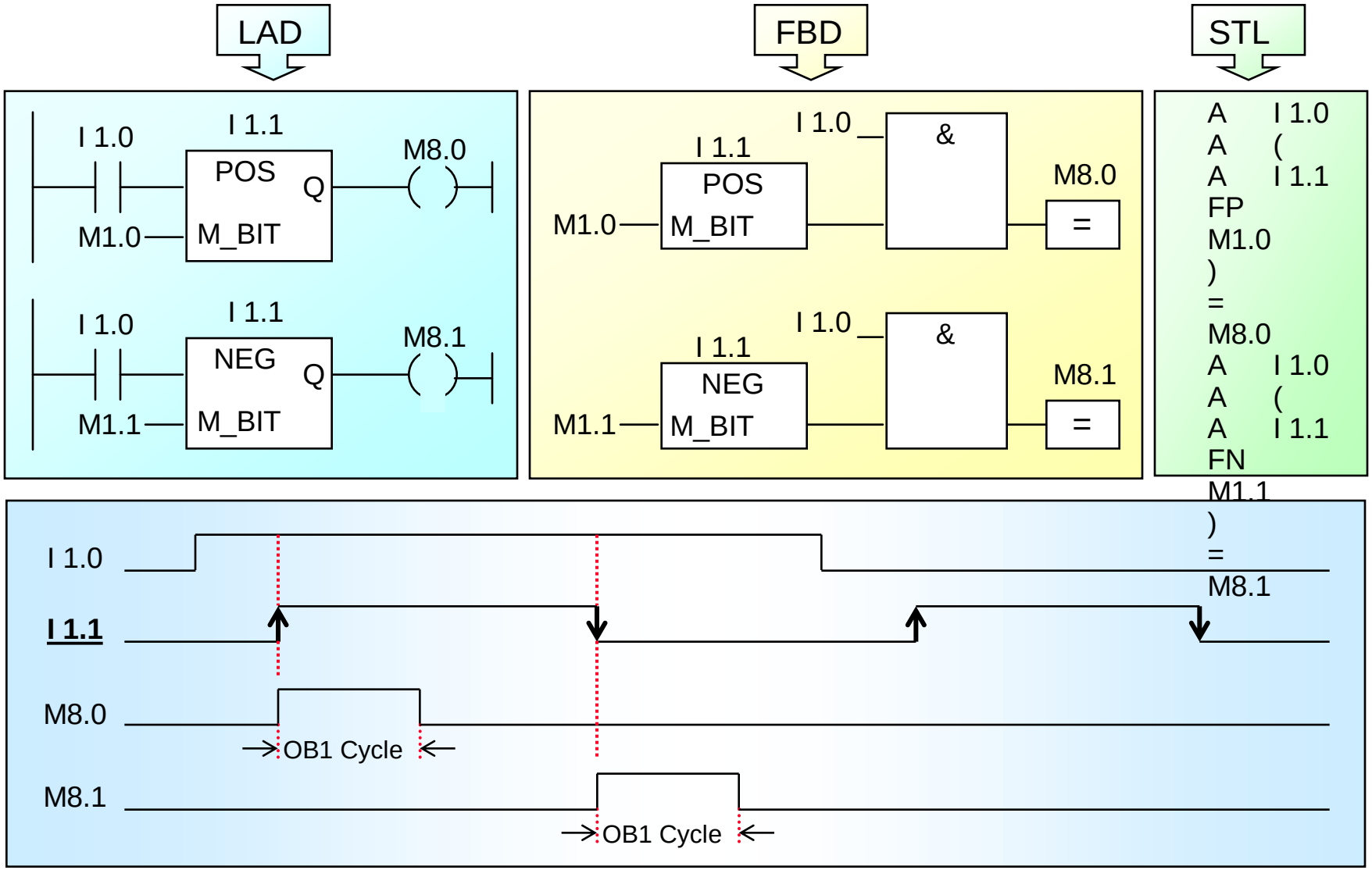


RLO – Edge Detection

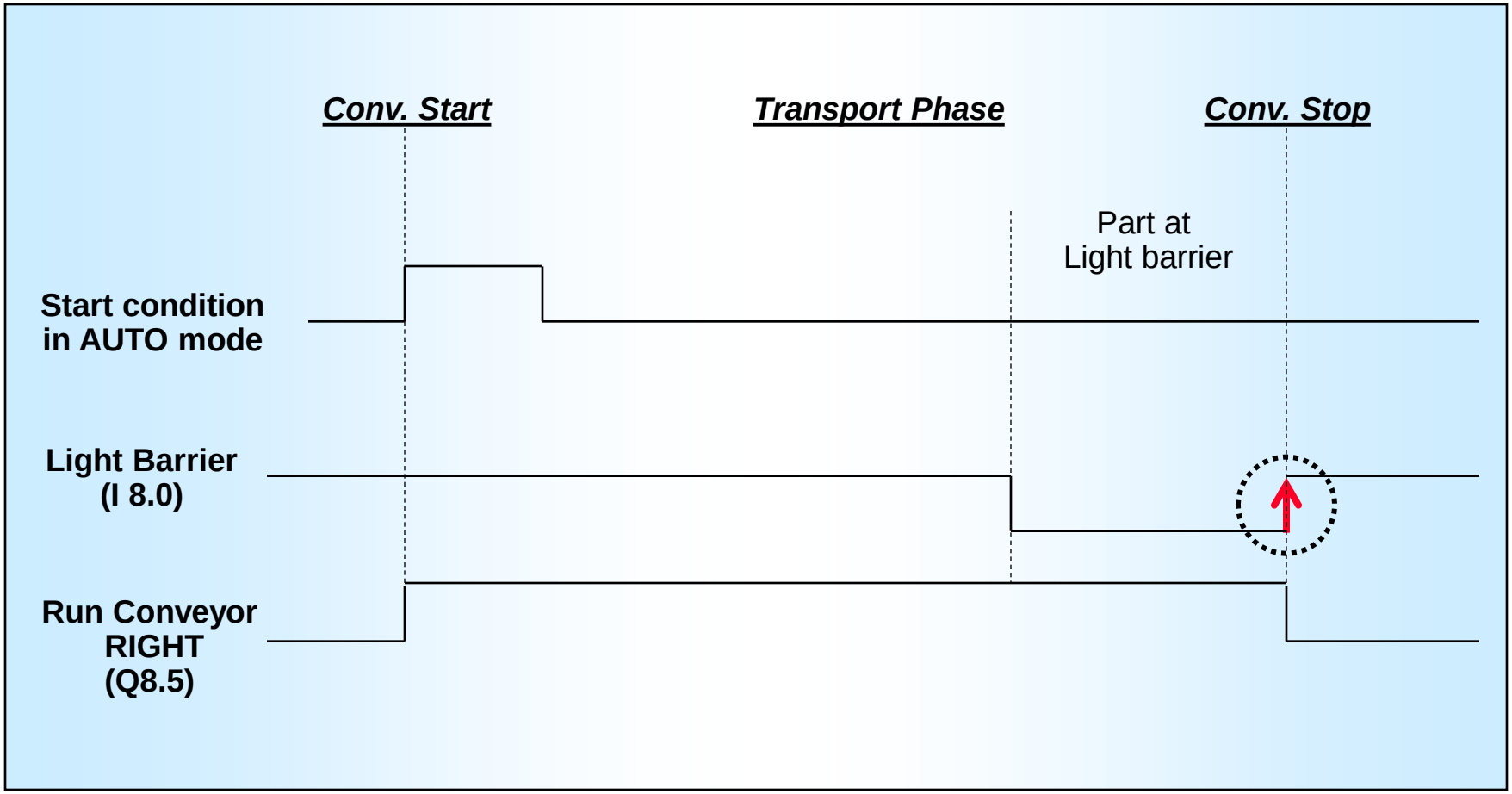




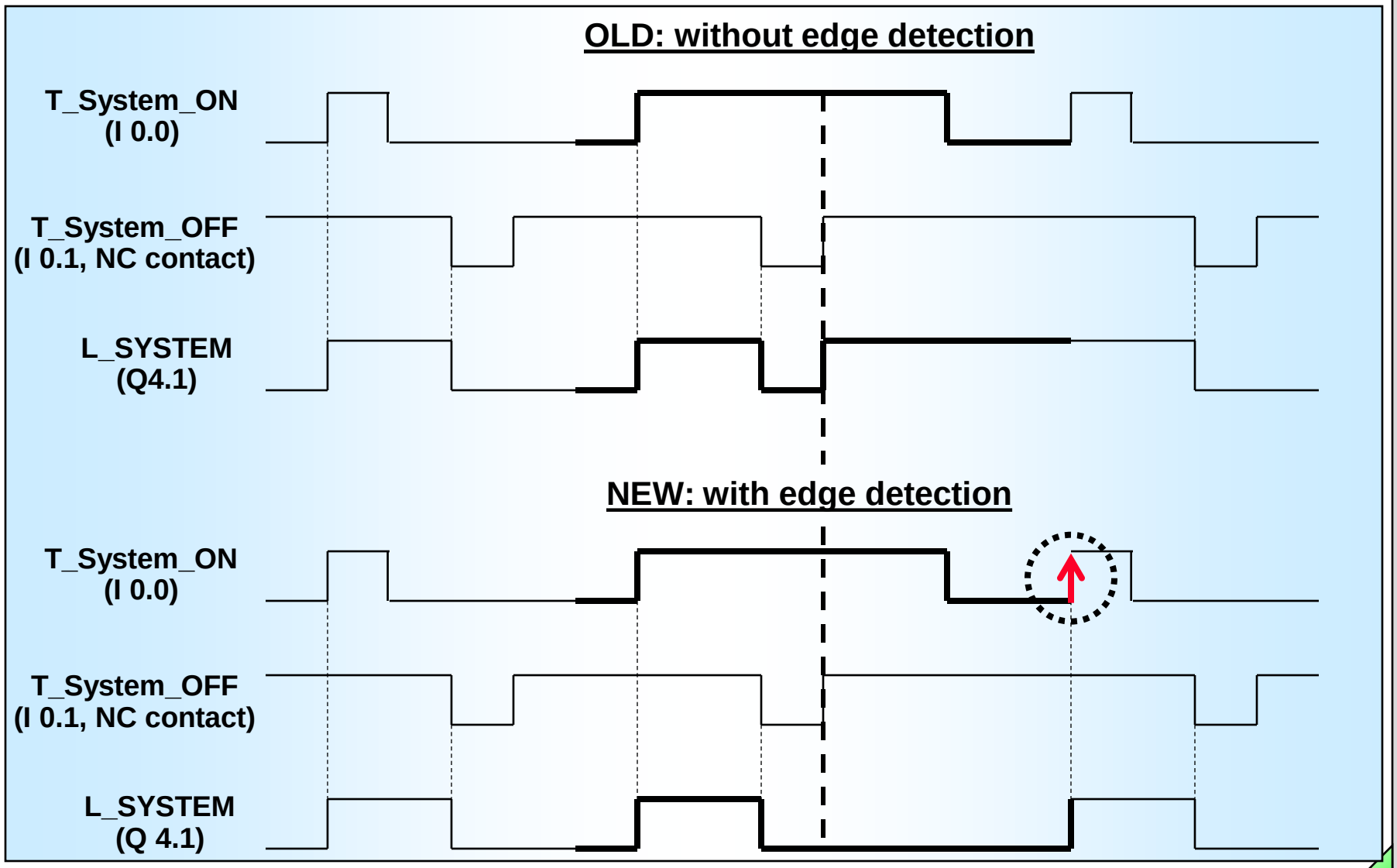
Signal – Edge Detection



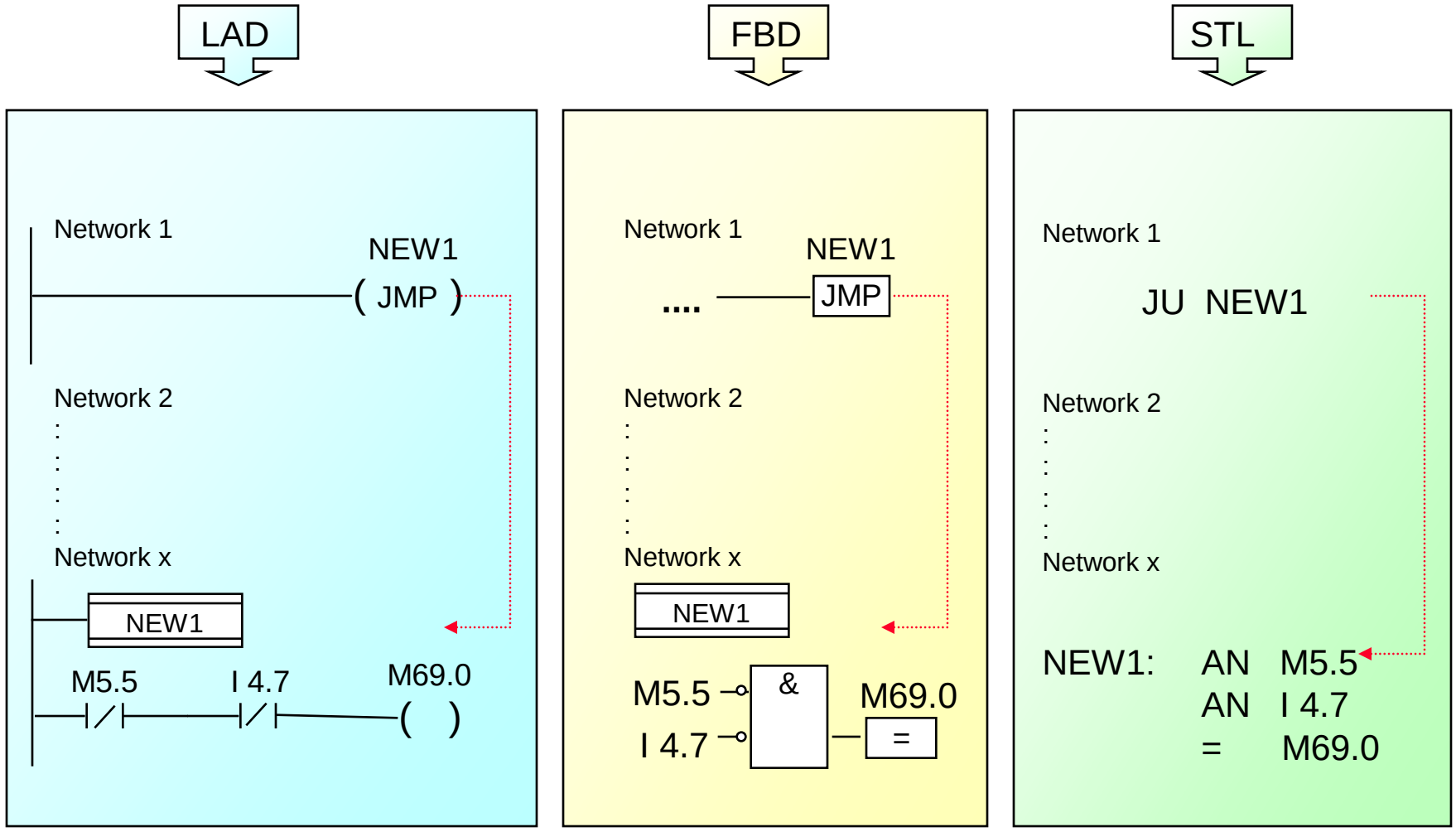
Exercise 5: Integrating the Edge Detection in FC 16



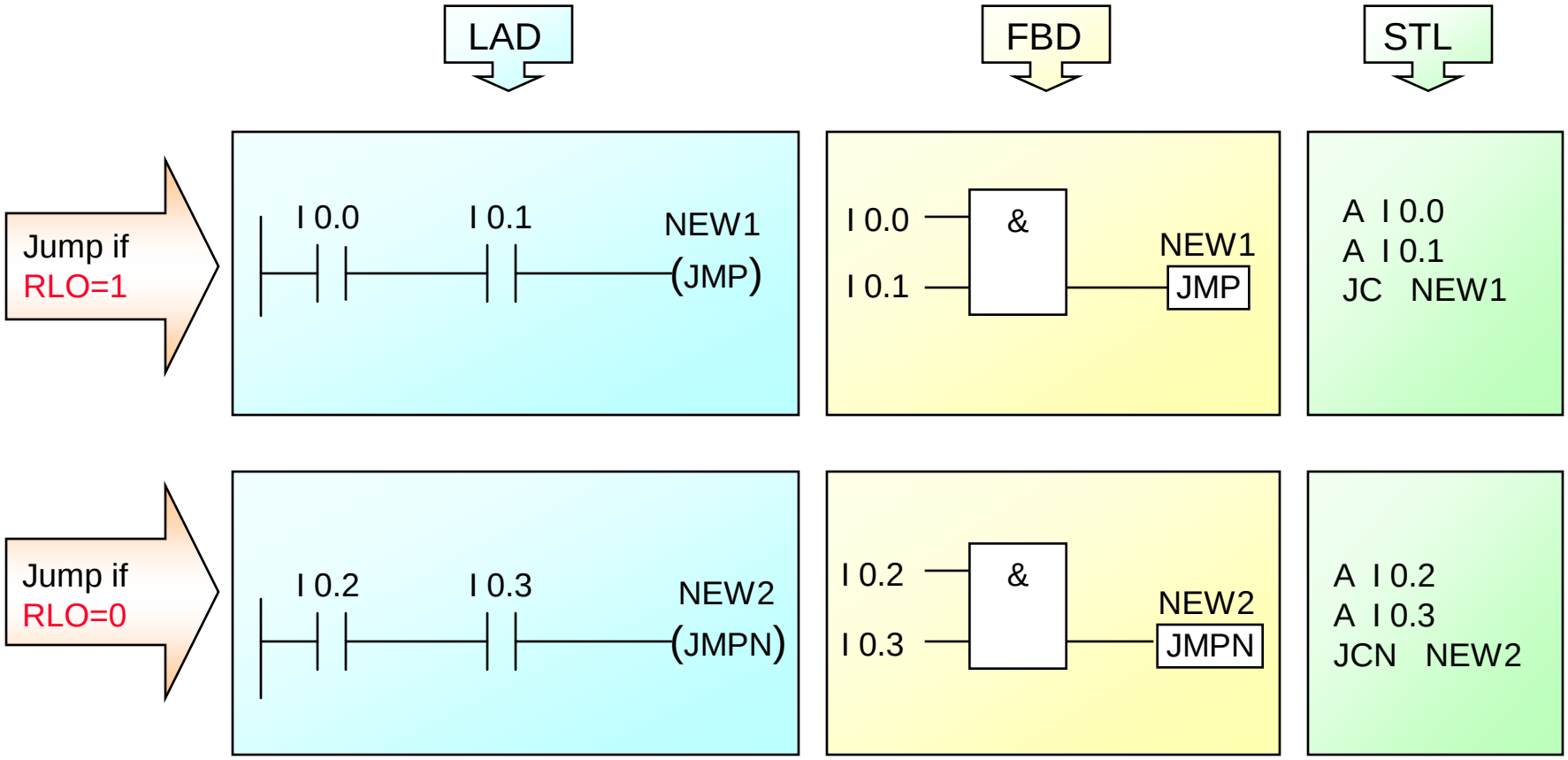
Exercise 6: Optimizing the Operating Mode Section (FC 15)



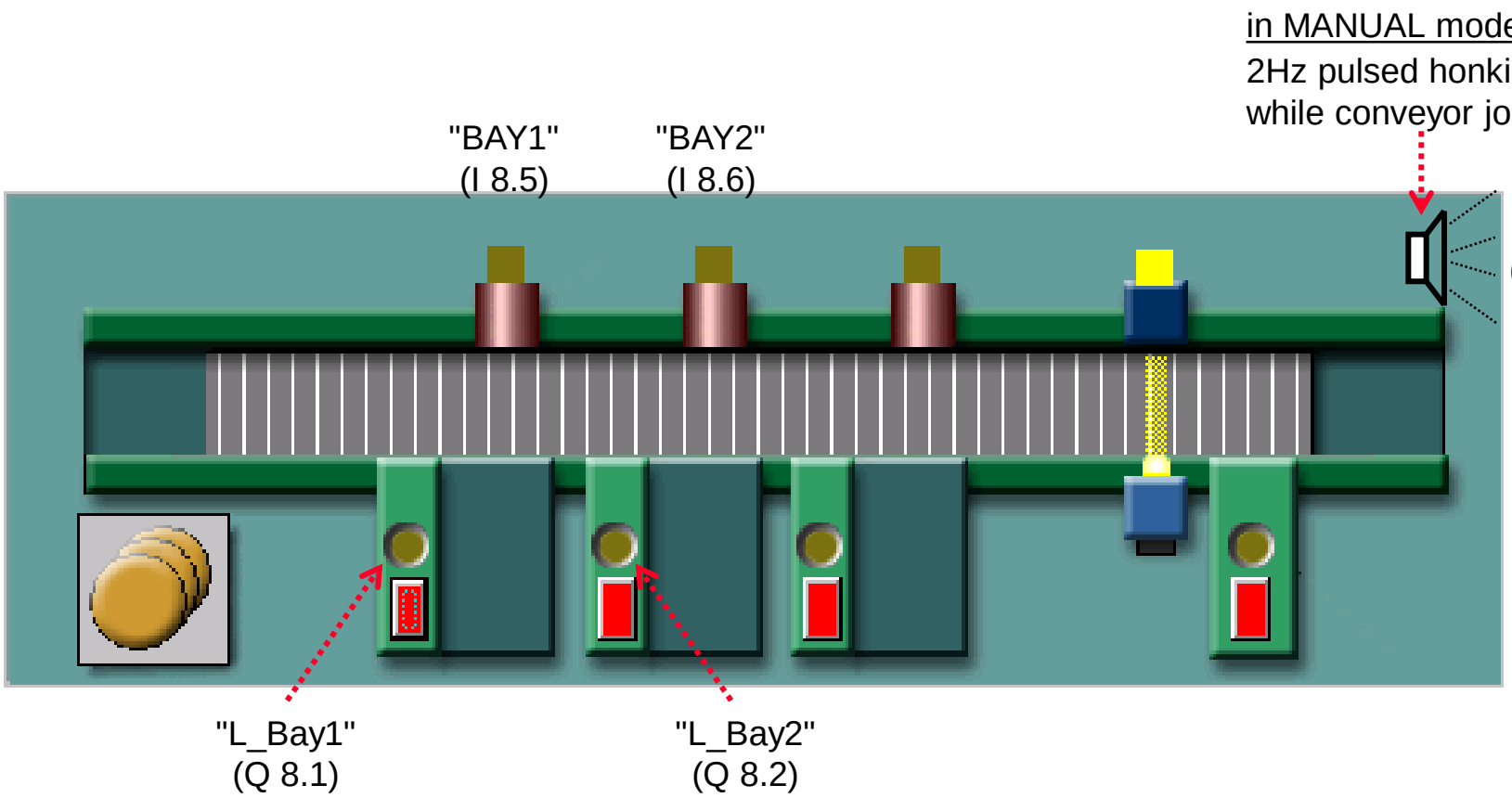
Unconditional Jump (Regardless of the RLO)



Conditional Jump (Dependent on RLO)



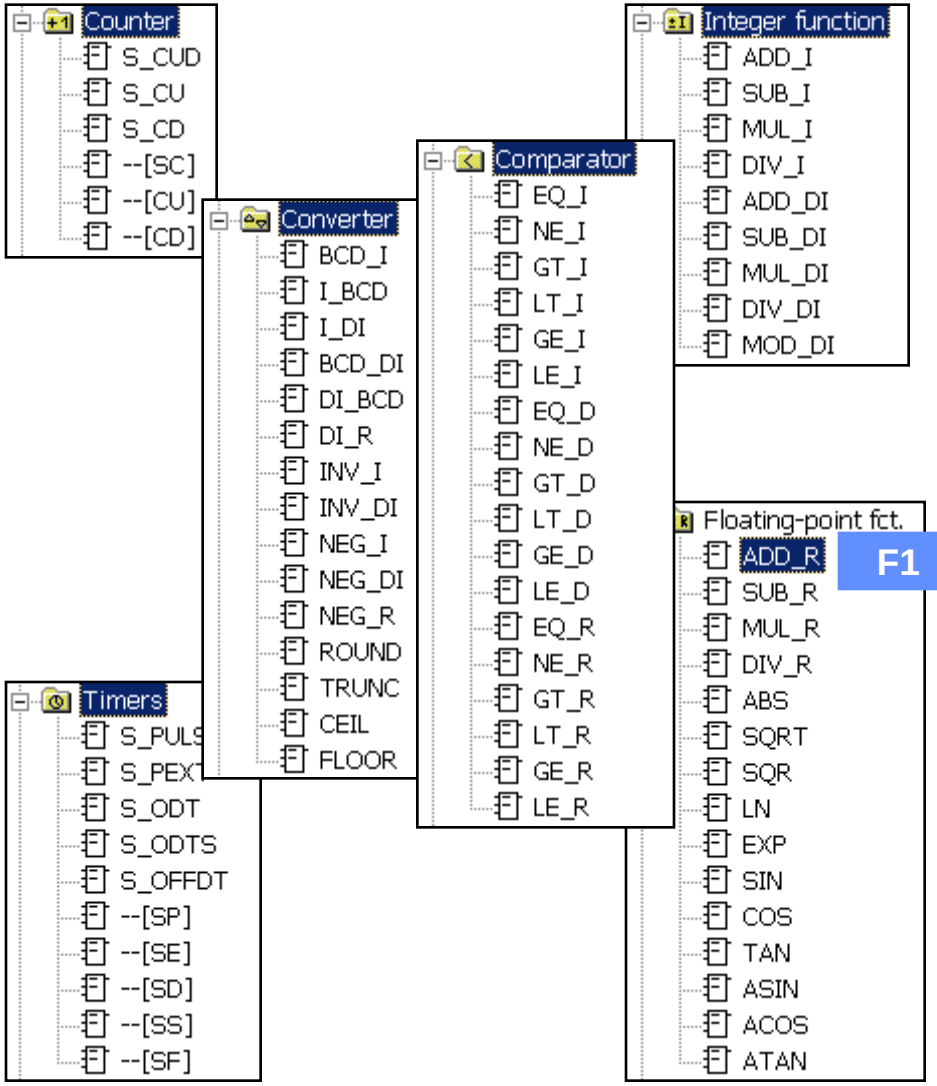
Exercise 7: Putting Indicator Lights + Horn (FC 14) into Operation



in AUTO mode:

- Constant light at both bays when a new part can be placed (both bays empty)
- 1Hz flashing light at the bay where the part is placed and the conveyor has not yet been started
- 2Hz flashing light at both bays during conveyor movement

Digital Operations



Help on Function Block Diagram

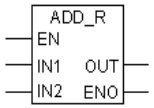
File Edit Bookmark Options Help

Contents | Index | Back | Print | << | >>

Help on STEP 7 | Glossary

ADD_R : Add Real

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, D, L, T, C	Enable input
IN1	REAL	I, Q, M, D, L or constant	First number to be added
IN2	REAL	I, Q, M, D, L or constant	Second number to be added
OUT	REAL	I, Q, M, D, L	Result of addition
ENO	BOOL	I, Q, M, D, L	Enable output

Description

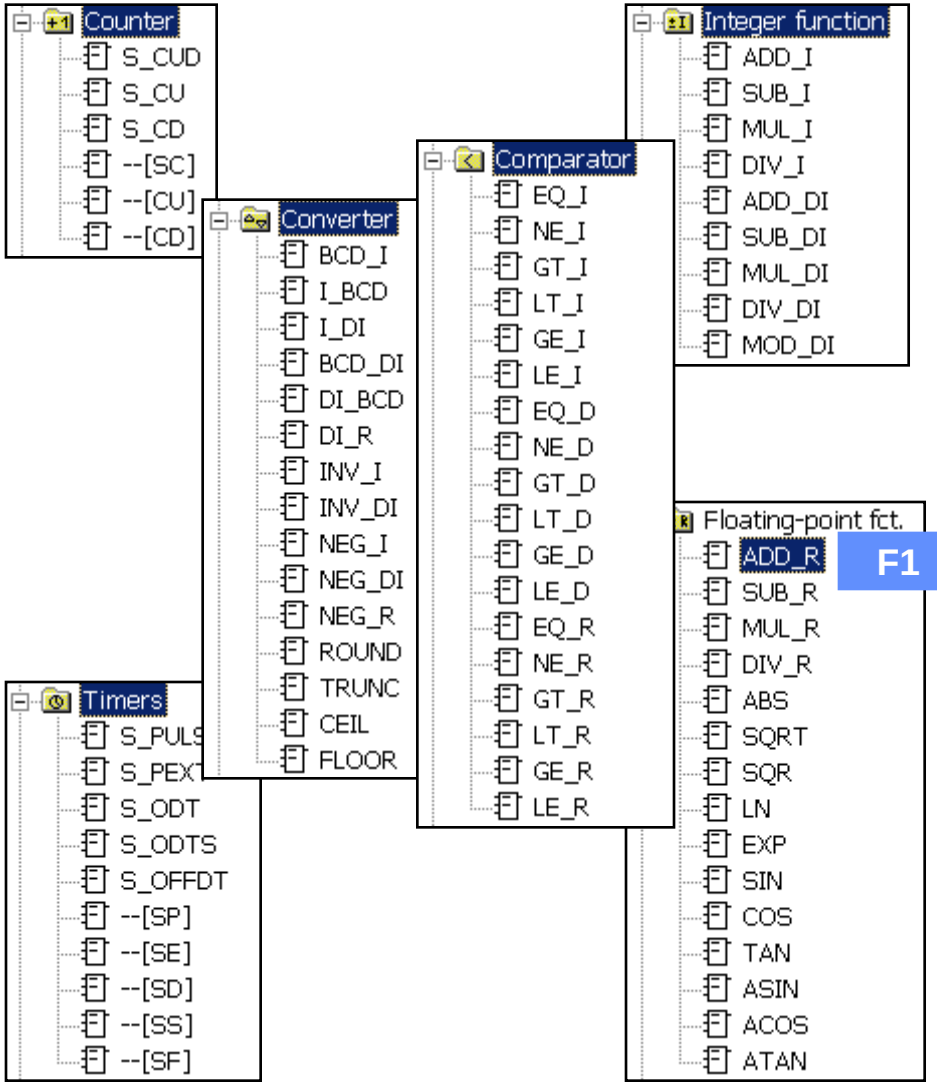
A signal state of 1 at the Enable input (EN) activates the **Add Real** instruction. This instruction adds inputs IN1 and IN2. The result can be scanned at output OUT. If either of the inputs or the result is not a floating-point number, the OV bit and OS bit are set to 1 and ENO is set to 0.

See also Evaluating the Bits of the Status Word with Floating Point Instructions.

Status Word

	BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
writes	X	X	X	X	X	0	X	X	1

Digital Operations



Help on Function Block Diagram

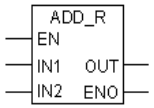
File Edit Bookmark Options Help

Contents | Index | Back | Print | << | >>

Help on STEP 7 | Glossary

ADD_R : Add Real

Symbol



Parameter	Data Type	Memory Area	Description
EN	BOOL	I, Q, M, D, L, T, C	Enable input
IN1	REAL	I, Q, M, D, L or constant	First number to be added
IN2	REAL	I, Q, M, D, L or constant	Second number to be added
OUT	REAL	I, Q, M, D, L	Result of addition
ENO	BOOL	I, Q, M, D, L	Enable output

Description

A signal state of 1 at the Enable input (EN) activates the **Add Real** instruction. This instruction adds inputs IN1 and IN2. The result can be scanned at output OUT. If either of the inputs or the result is not a floating-point number, the OV bit and OS bit are set to 1 and ENO is set to 0.

See also Evaluating the Bits of the Status Word with Floating Point Instructions.

Status Word

	BR	CC1	CC0	OV	OS	OR	STA	RLO	FC
writes	X	X	X	X	X	0	X	X	1

Objectives

Upon completion of the chapter the participant will

... be familiar with the INT, DINT, REAL data types and the BCD display

... be able to apply the selectable display formats in the "Monitor / Modify Variable" test function

... understand the "Load" and "Transfer" instructions

... be able to apply and program the S5 counter and timer functions

... understand conversion operations for number formats

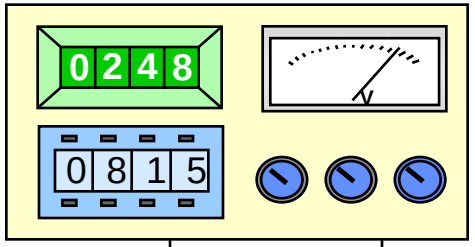
... be able to apply and program the comparison operations

... understand basic mathematical functions

Acquiring, Processing and Outputting Data

Operating + monitoring processes

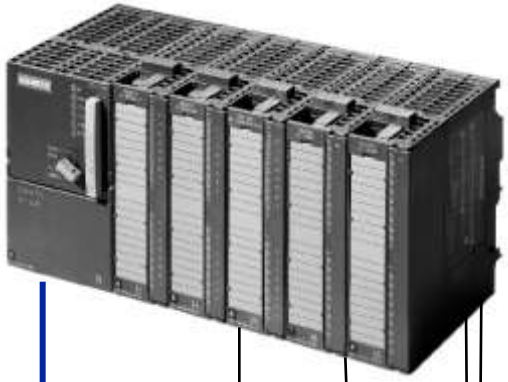
Operator Panel



Thumbwheel buttons, Potentiometer, 7-segment display

Control unit

such as MPI



Processing the values

DI/ DO

AI/ AO

such as PROFIBUS

Process controller

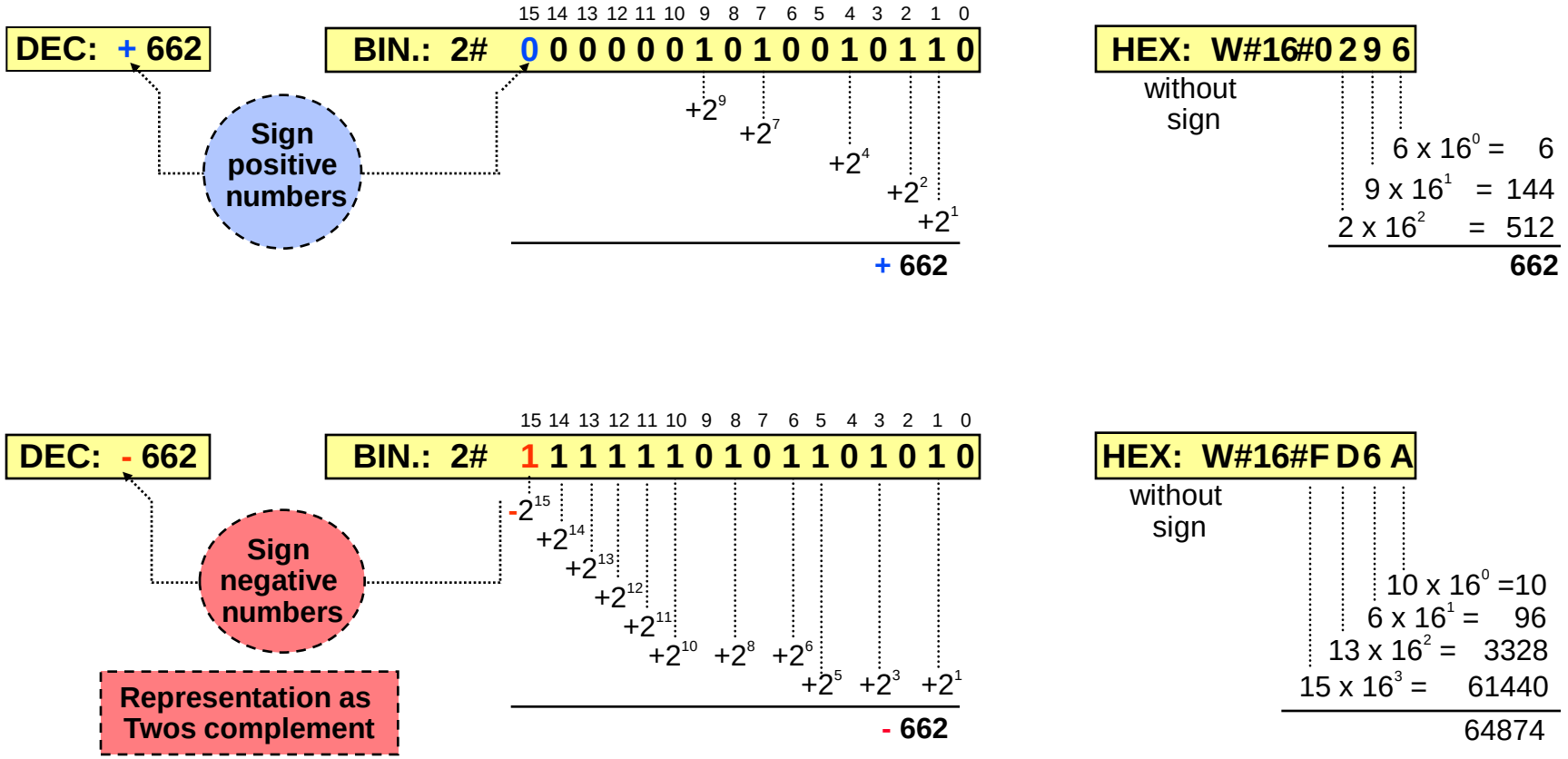
Field devices

Integer (INT, 16-Bit Integer) Data Type

Value range -32,768 to +32,767
(without sign: 0 to 65,535)

Arithmetic operations: such as + I, * I, <I, ==I

Display Formats:

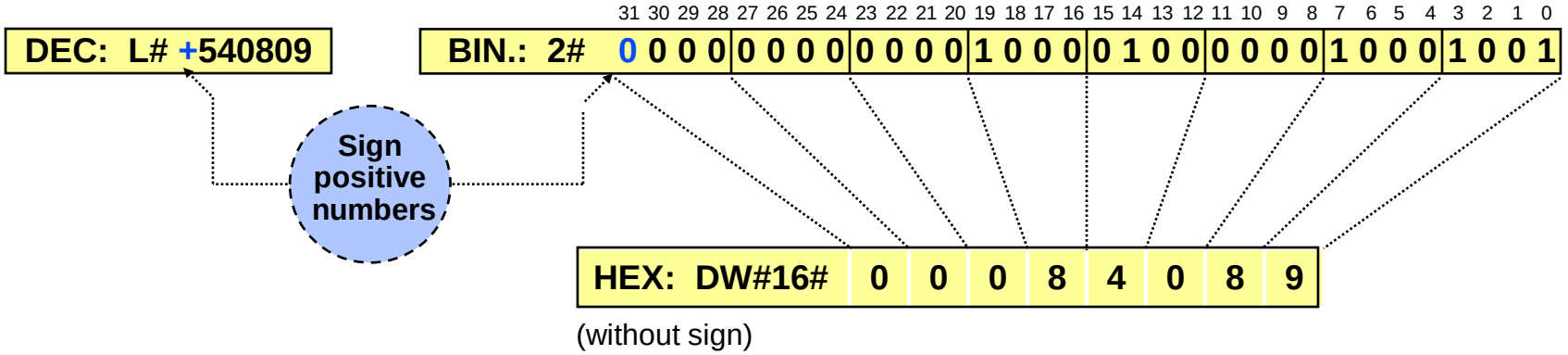


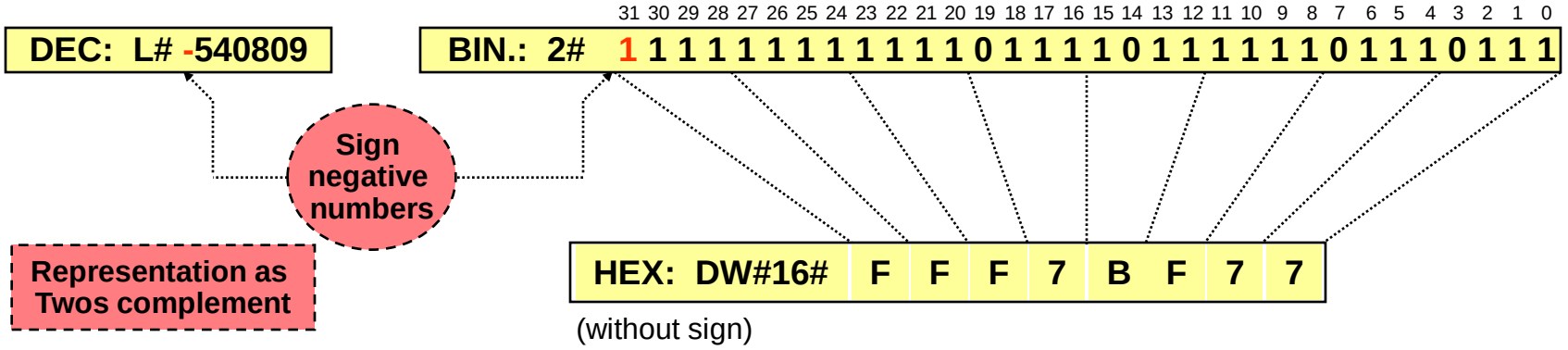
Double Integer (DINT, 32-Bit Integer) Data Type

Value range L# -2,147,483,648 to L#+2,147,483,647
(without sign: 0 to 4,294,967,295)

Operations: such as **+ D, * D, <D, ==D**

Display Formats:





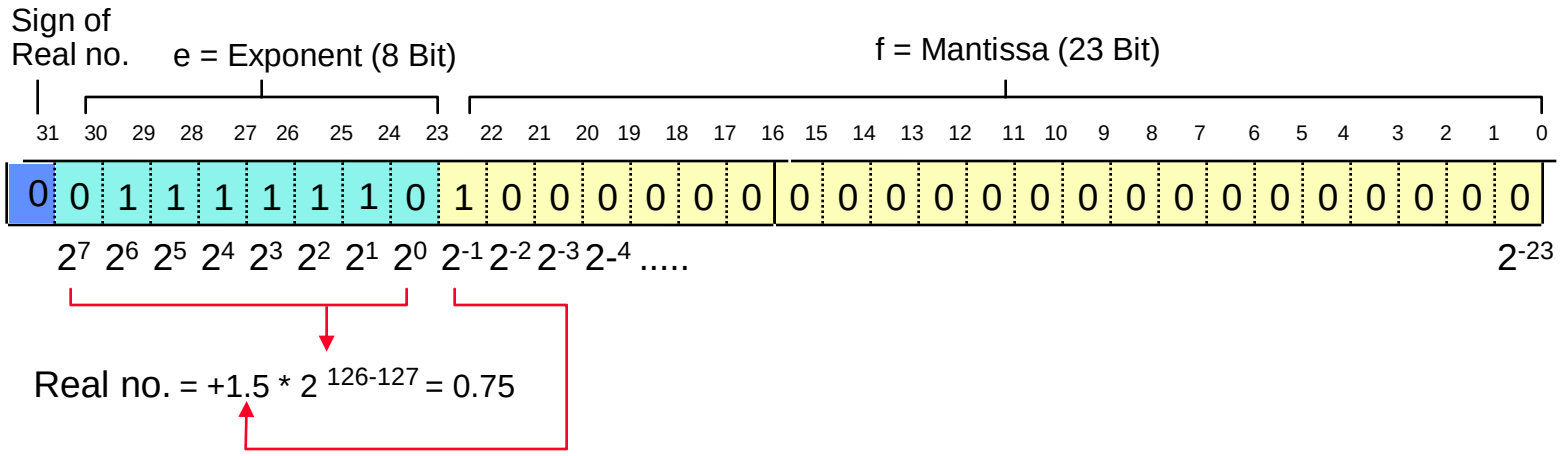
REAL (Floating-point Number, 32 Bit) Data Type

Value range **-3.402823•10^{+38} to -1.175495•10^{-38} , 0.0, +1.175495•10^{-38} to +3.402823•10^{+38}**

Operations: such as **+ R, * R, <R, ==R**
sin, acos, ln, exp, SQR

General format of a Real number = (Sign) • (1 + f) • (2^{e-127})

Example: 0.75

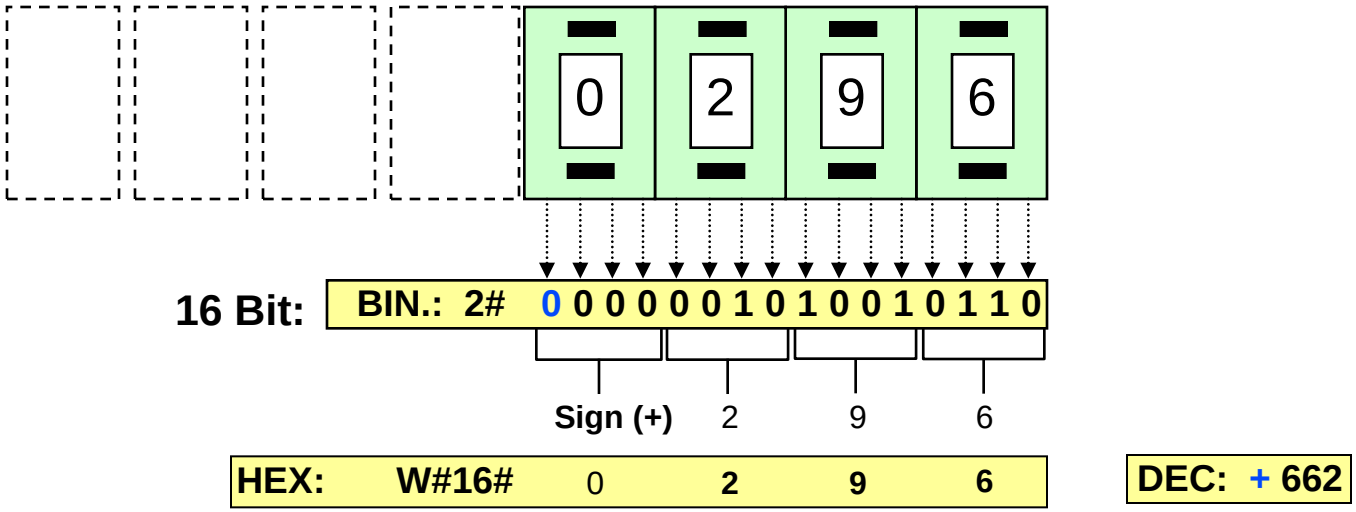


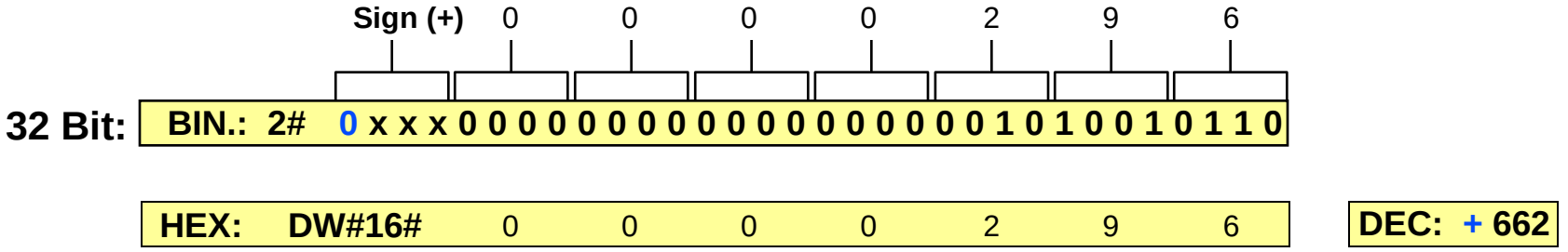
The BCD Code for Inputting and Outputting Integers

Value range

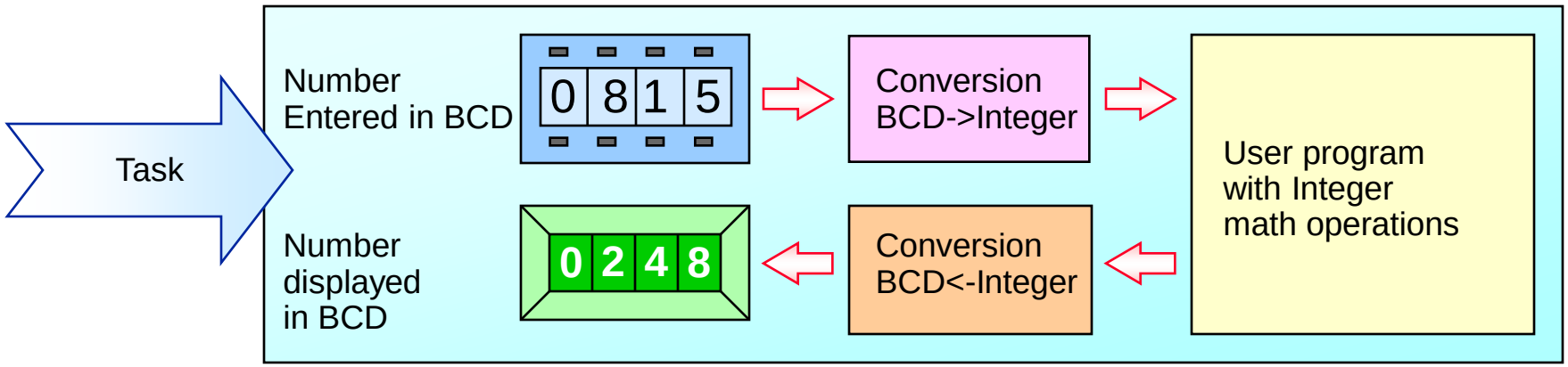
16 Bit: **- 999 to + 999**
32 Bit: **-9999999 to + 9999999**

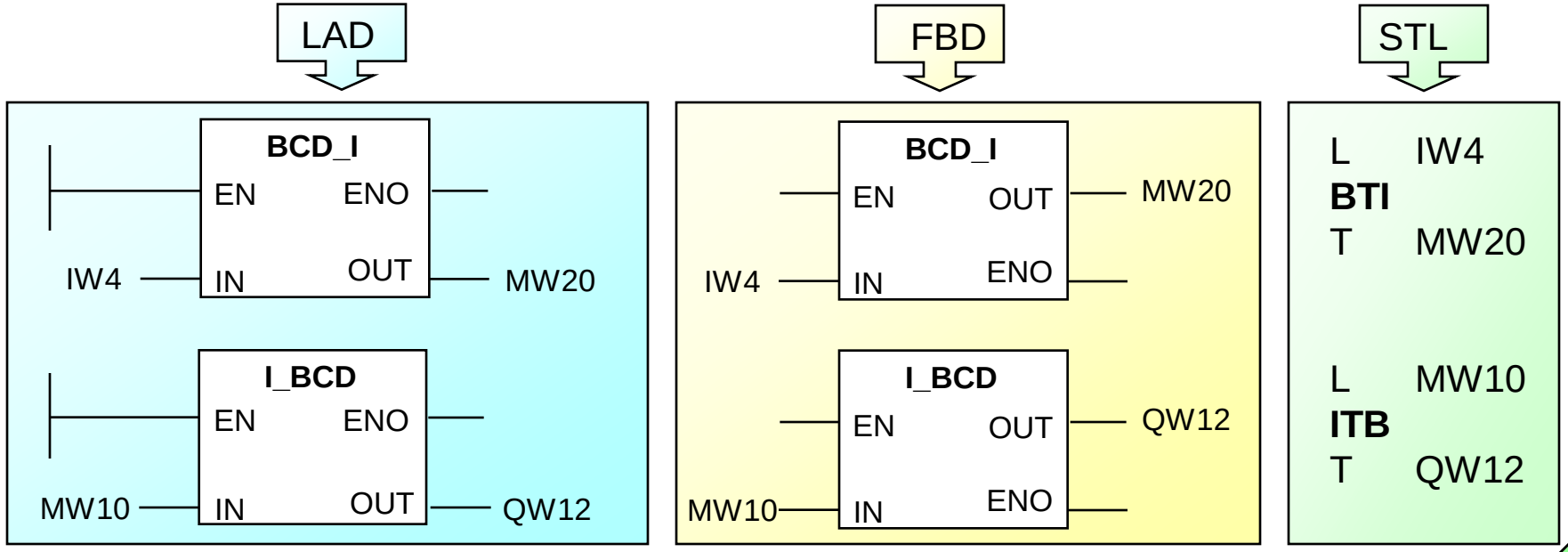
Conversion operations: BTI, BTD, ITB, DTB (no arithmetic!)



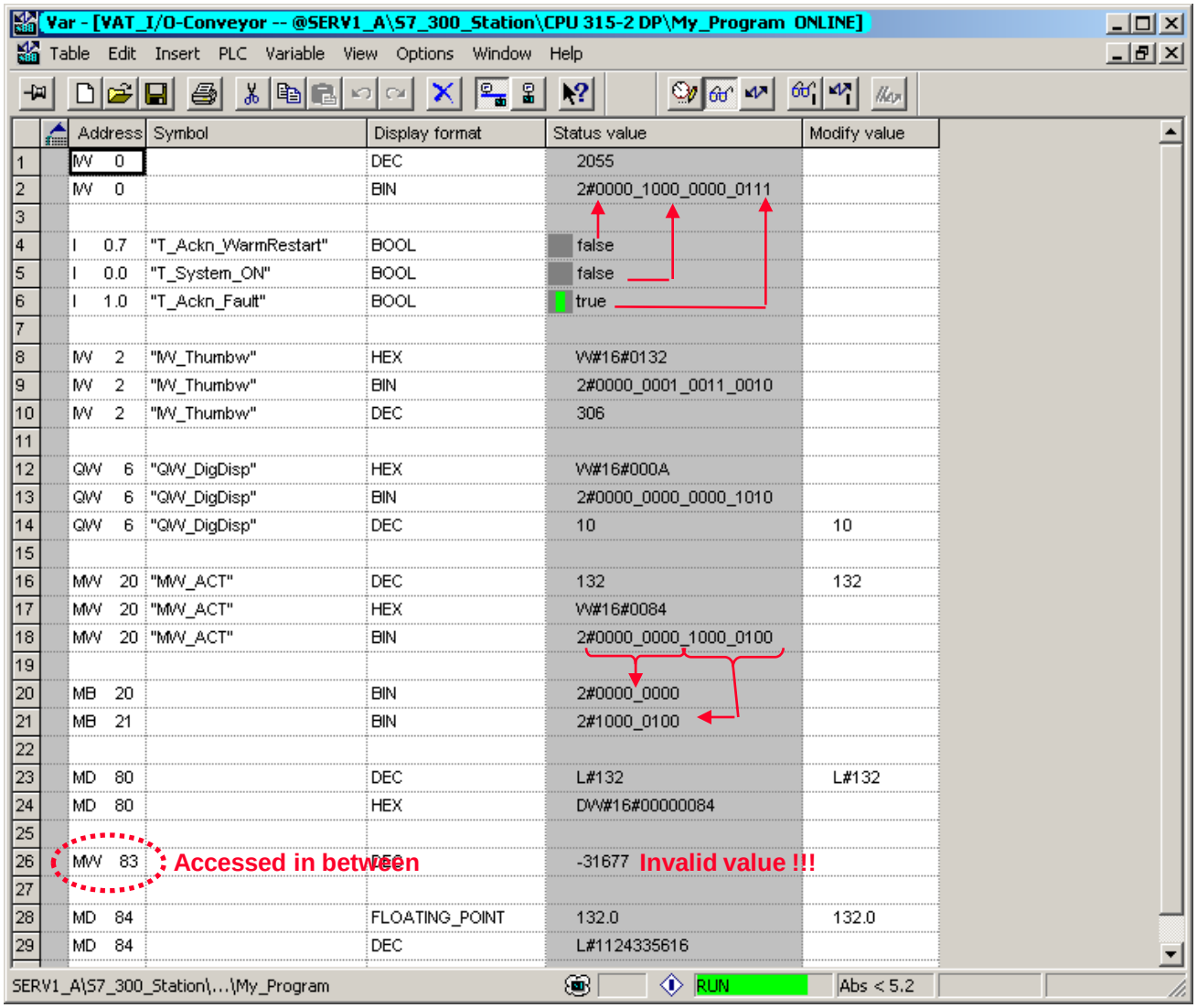


Conversion Operations BCD ←→ Integer





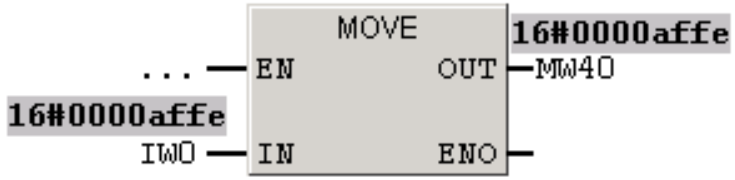
Number Formats



Loading and Transferring Data

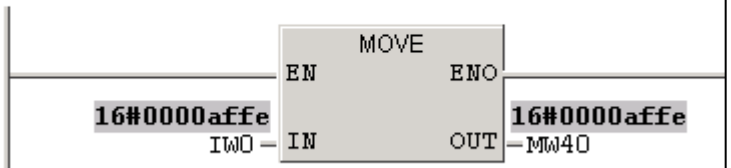
FC13 : move operations

Network 1: function block diagram



FC13 : move operations

Network 1: ladder diagram



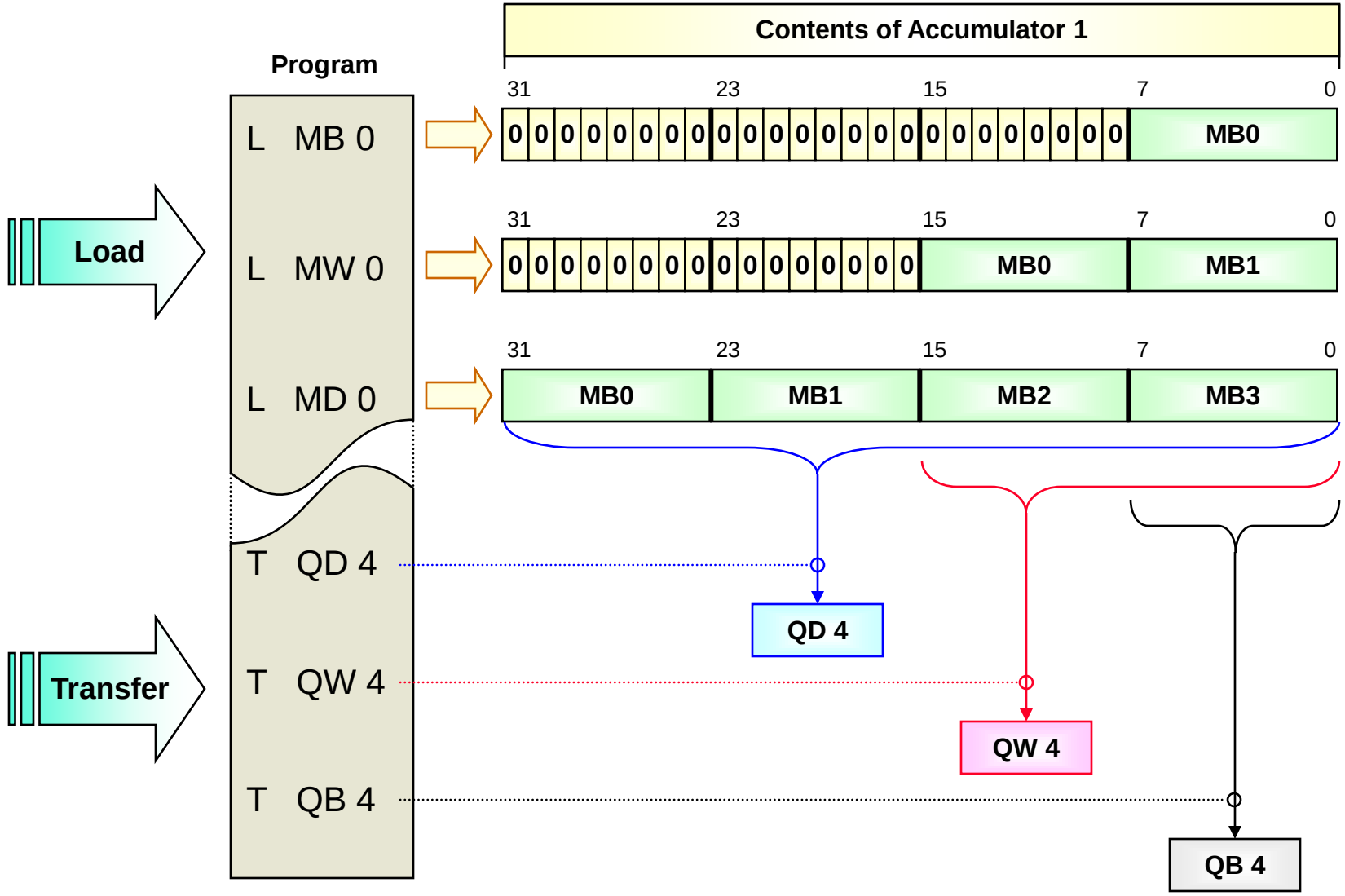
FC13 : move operations

Network 1: statement list

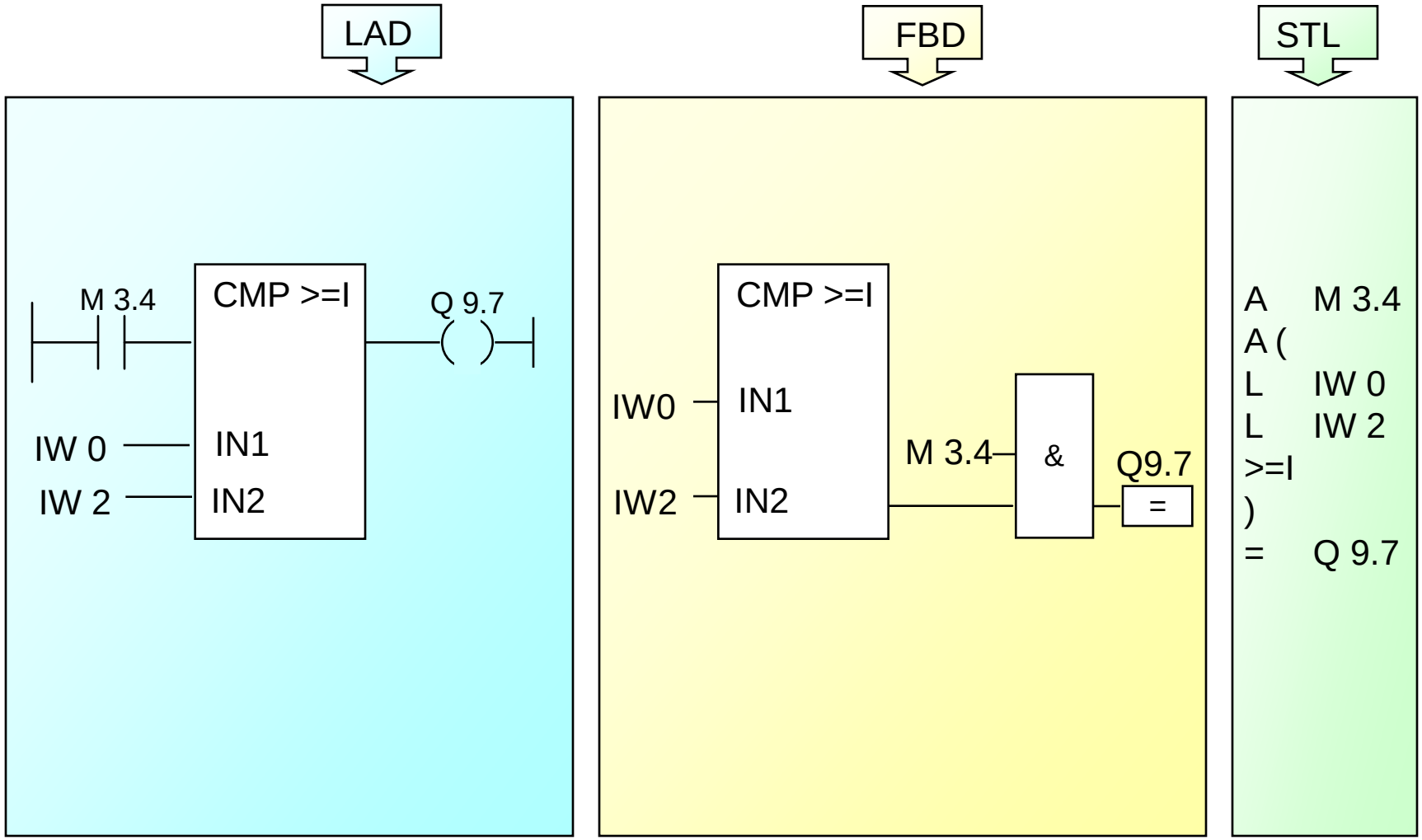
```
L     DW#16#CAFECAFE
T     MD     42
L     W#16#ADAC
L     MW     46
L     B#16#FF

TAK
```

RLO	STA	STANDARD	ACCU 2
		Accumulator 1	
0	1	cafecafe	0
0	1	cafecafe	0
0	1	adac	cafecafe
0	1	0	adac
0	1	ff	0
0	1	0	ff

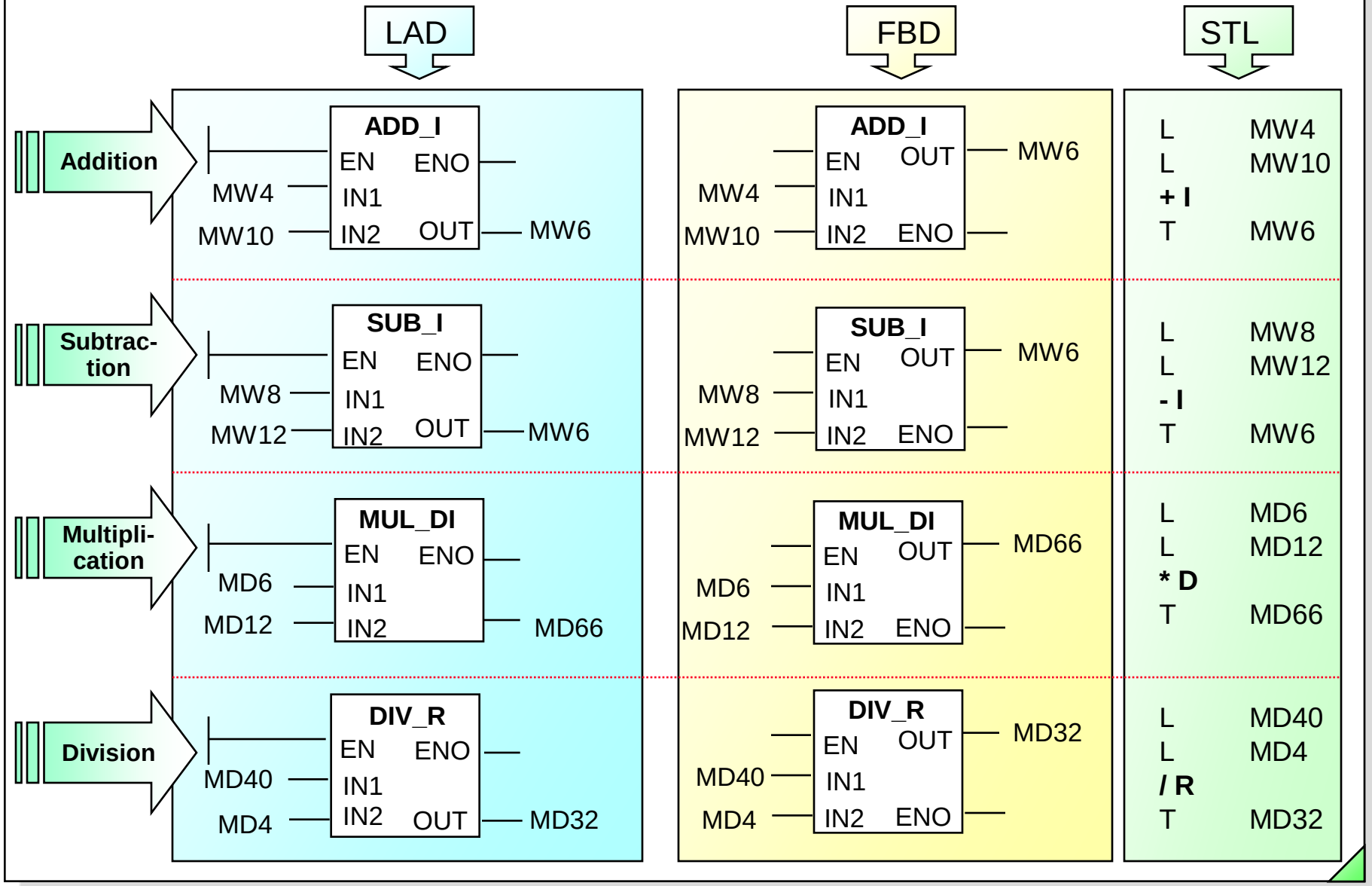
Data Storage in Accumulator 1



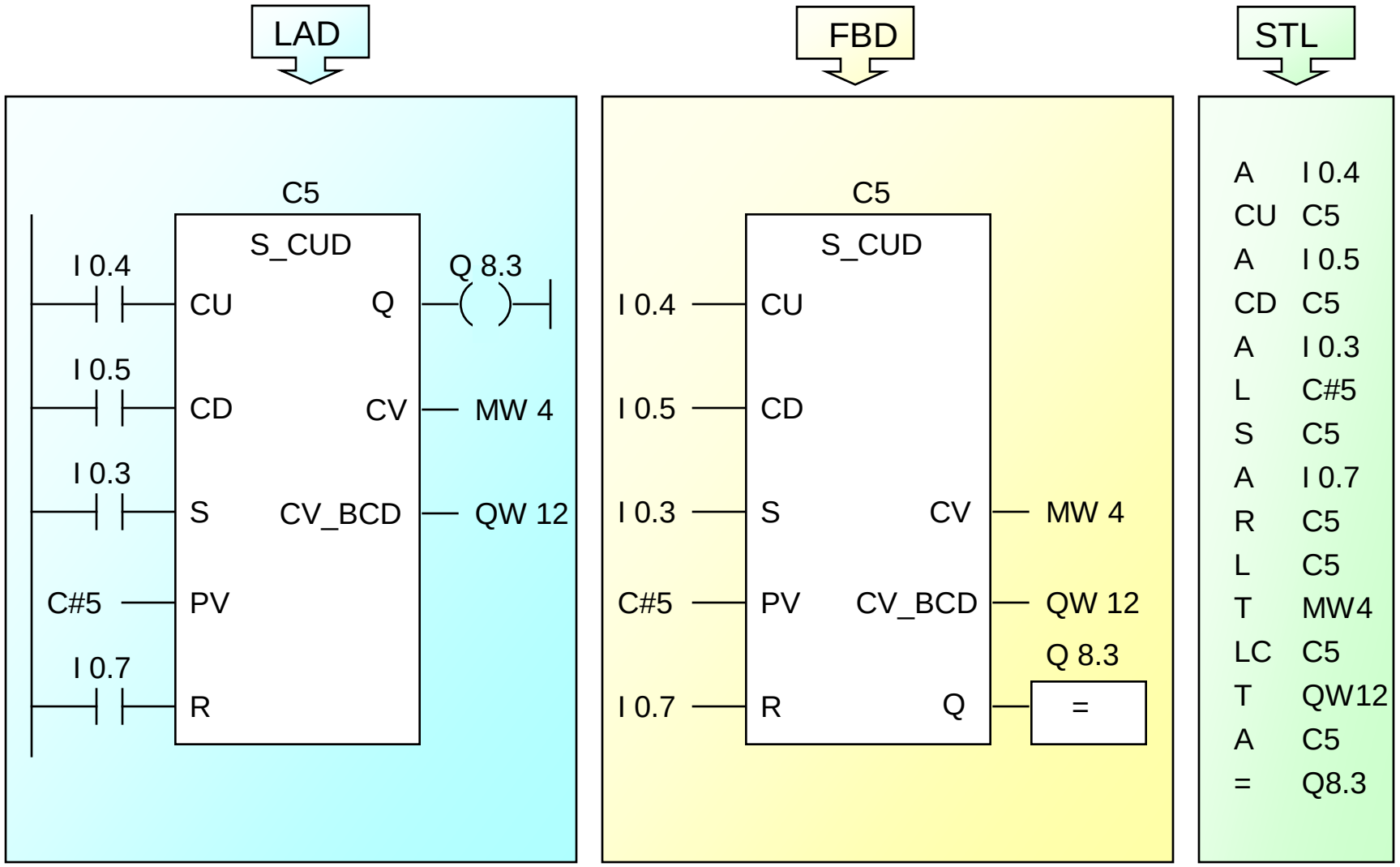
Comparison Operations



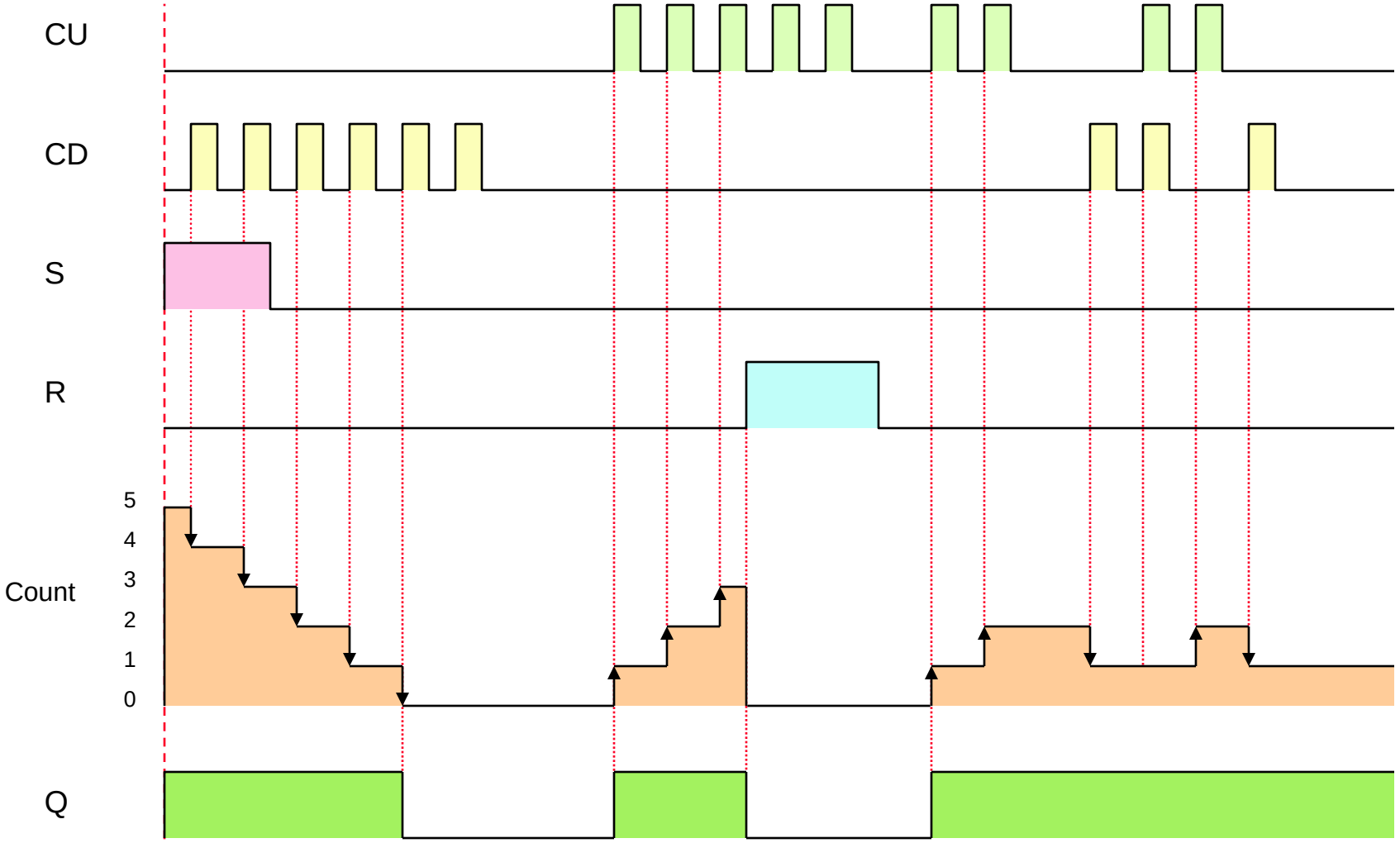
Basic Mathematical Functions



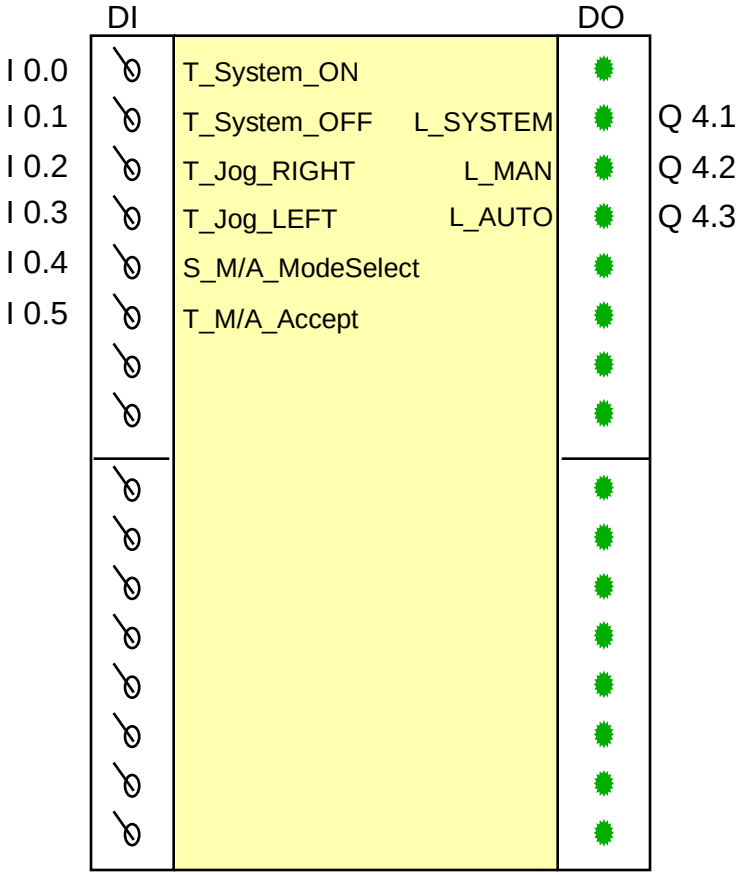
S5 Counter Functions

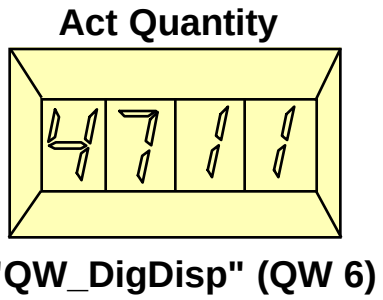


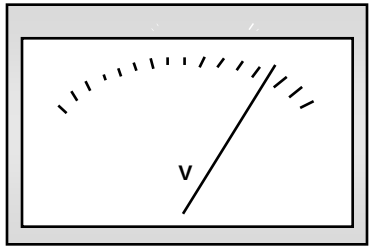
Counters: Function Diagram

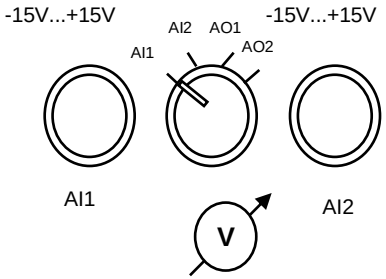


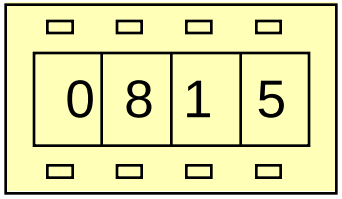
Exercise 1: Counting the Transported Parts (FC 18, C 18)

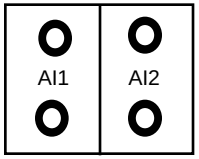


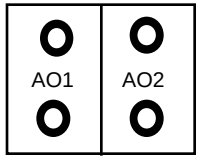




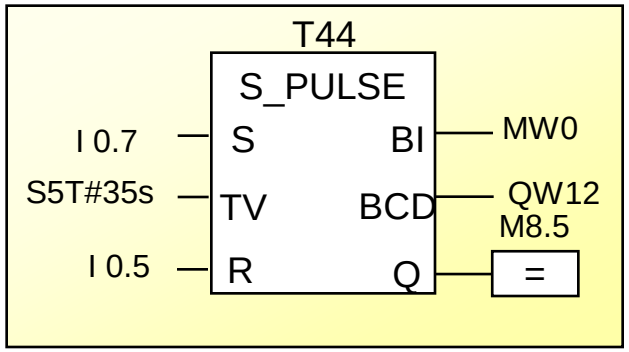




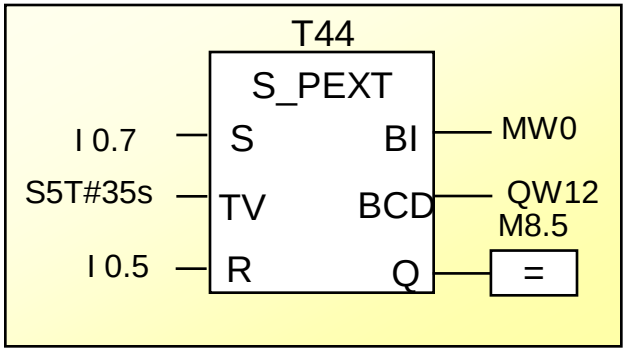




S5 Timer Functions

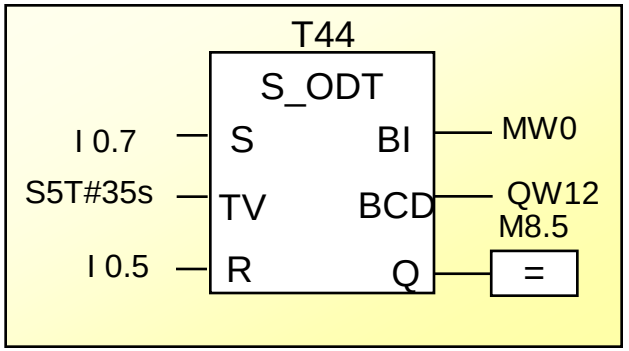


Pulse Timer (SP)

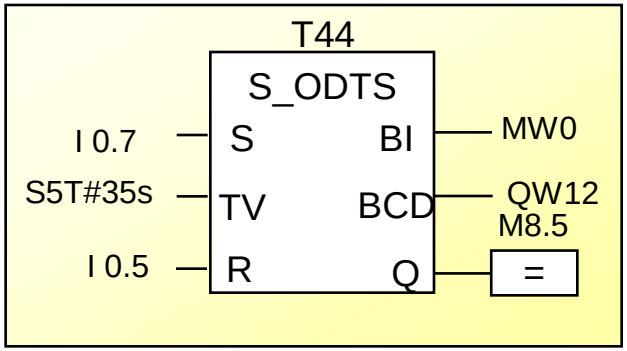


Extended Pulse (SE)

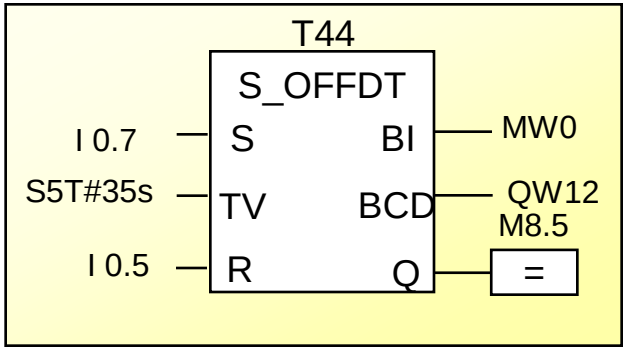
ON Delay (SD)



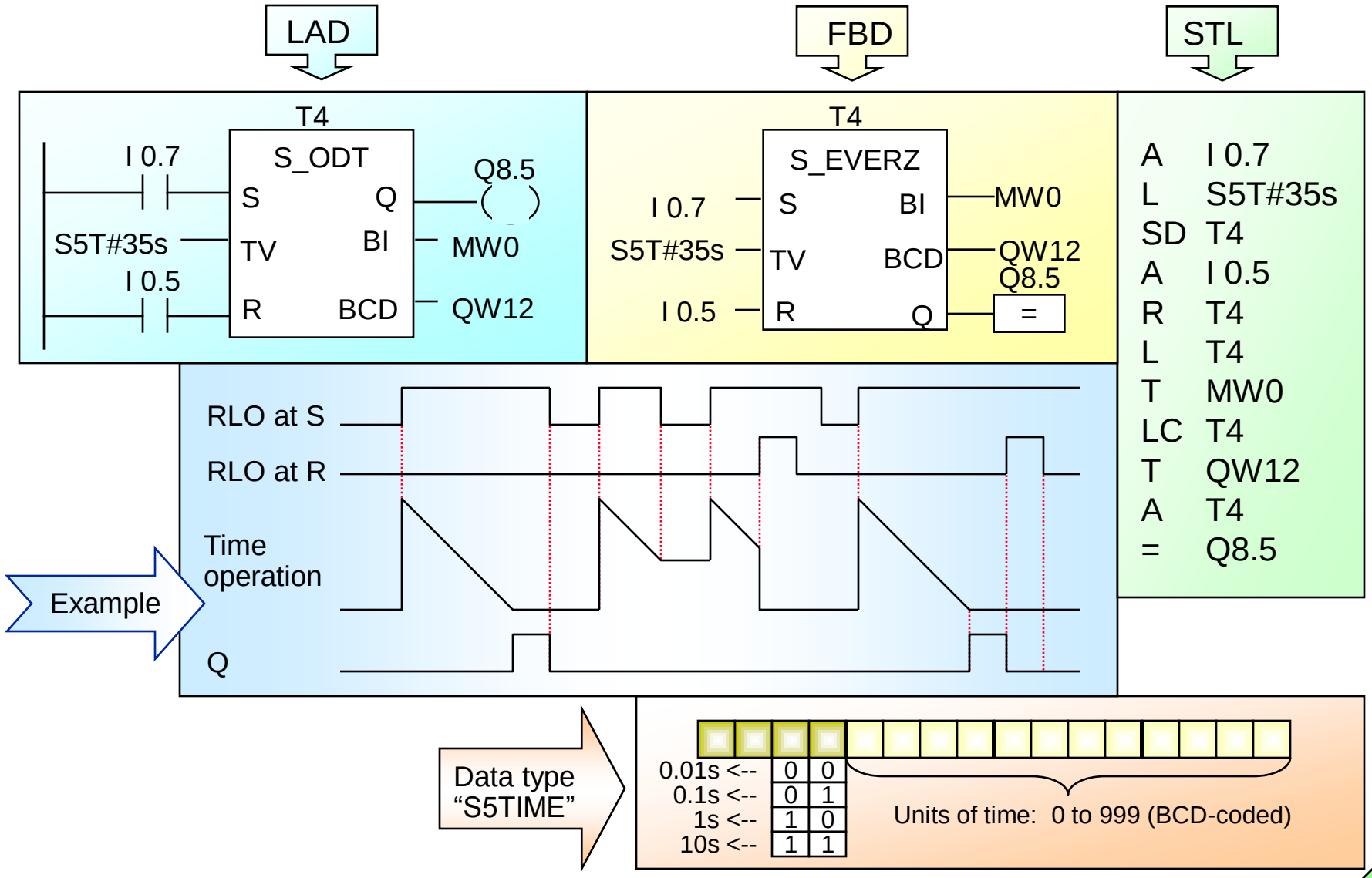
Stored ON Delay (SS)



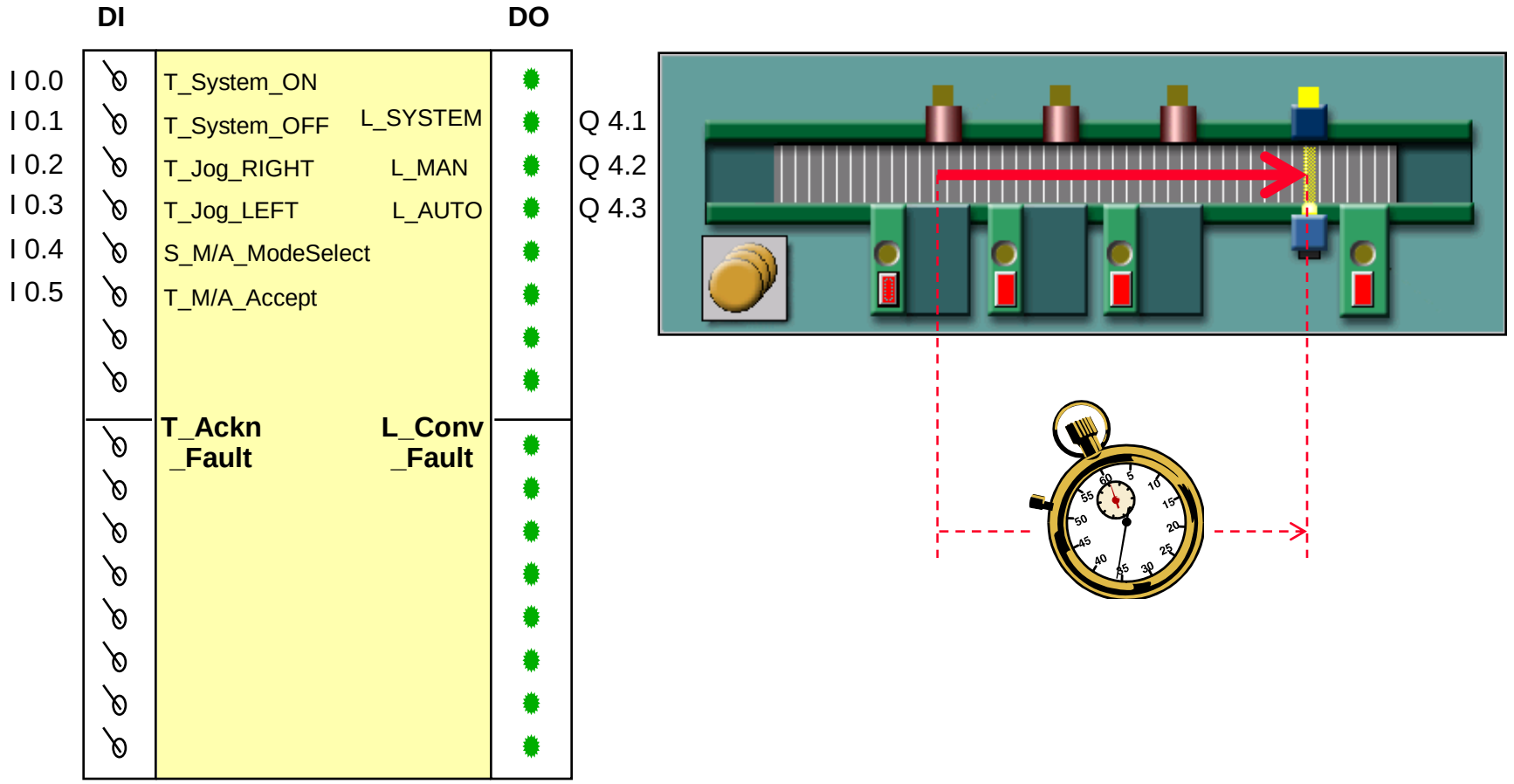
OFF Delay (SF)



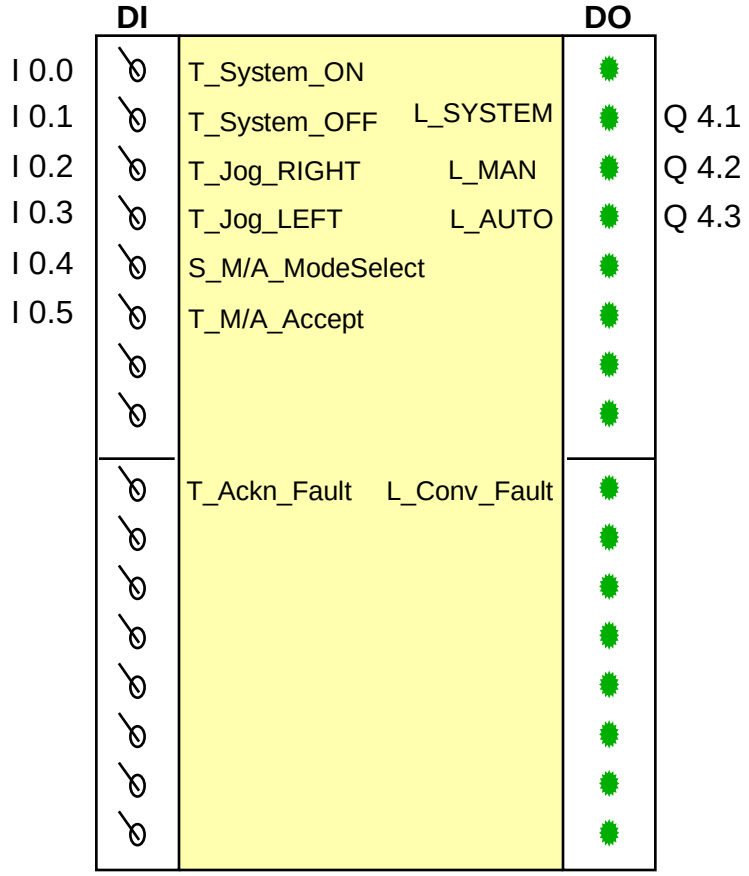
Timers: ON Delay (SD)



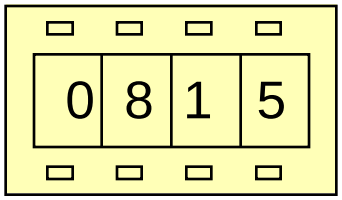
Exercise 2: Monitoring the Transport Functions (FC 17)



Exercise 3: Counting the Transported Parts (FC 19)

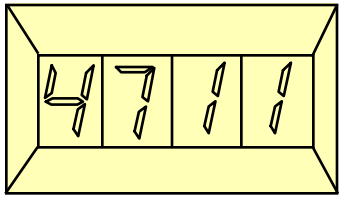


SETPOINT Quantity

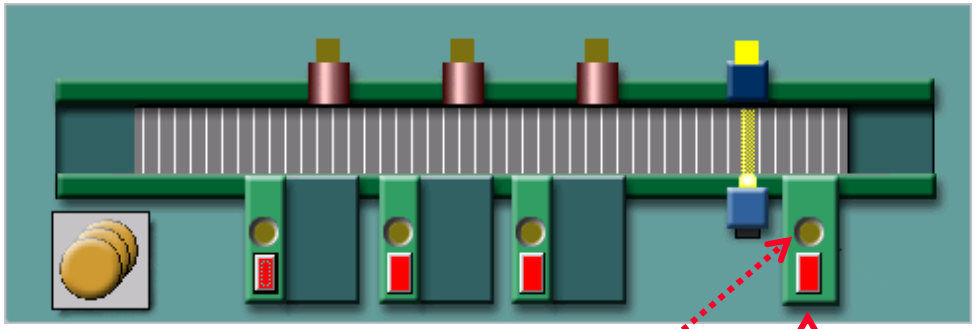


"IW_Thumbw" (IW 2)

ACTUAL Quantity



"QW_DigDisp" (QW 6)

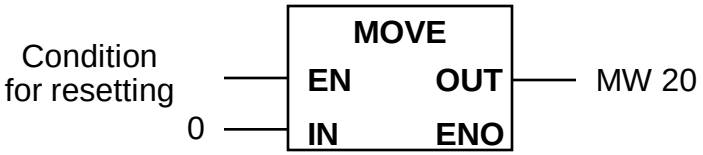


ACT=SET
"L_Bay-LB" (Q 8.4)

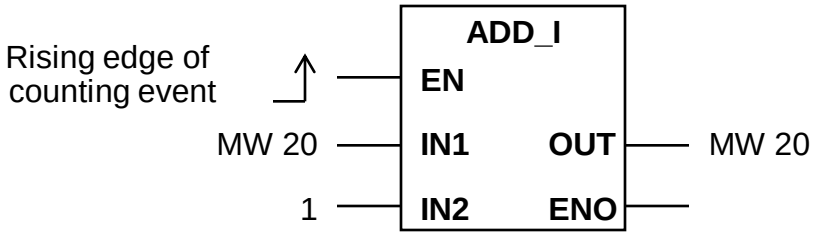
Acknowledgement
"T_Bay-LB" (I 8.4)

Exercise 3: Counting the Transported Parts (FC 19), Solution Hints

NW 1: Set "Counter" to 0



NW 2: Counting via memory word, ACT quantity



NW 3: Display ACT quantity on the BCD digital display

Convert ACT quantity (MW 20) from Integer to BCD -> Result on BCD digital display (QW 6)

NW 4: Read in SETP quantity and convert from BCD to Integer

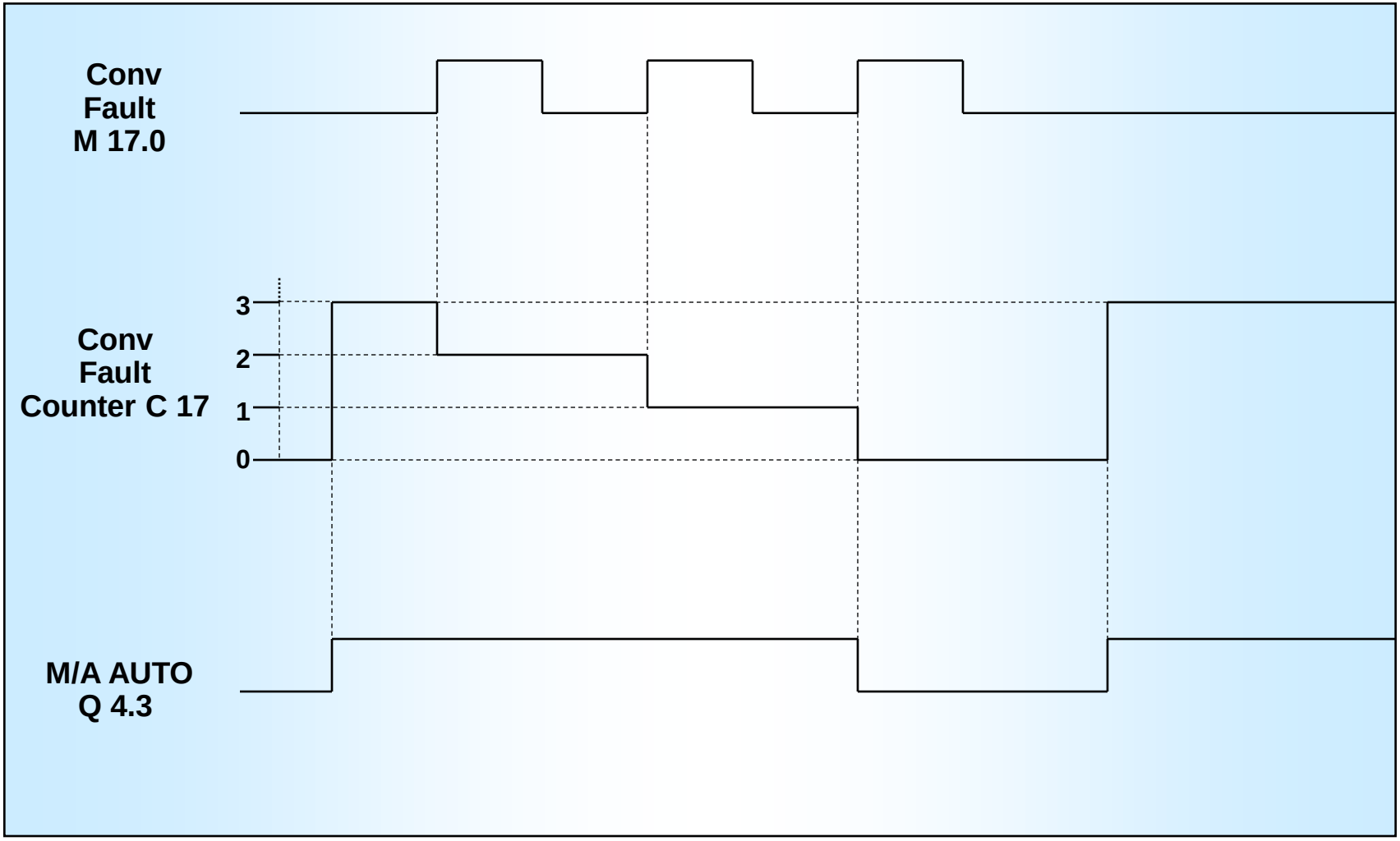
Convert value from thumbwheel button (IW 2) from BCD to Integer -> Store SETP quantity (Integer) in MW 22

NW 5: ACT-SETP comparison, Conveyor indicator light Q 8.4

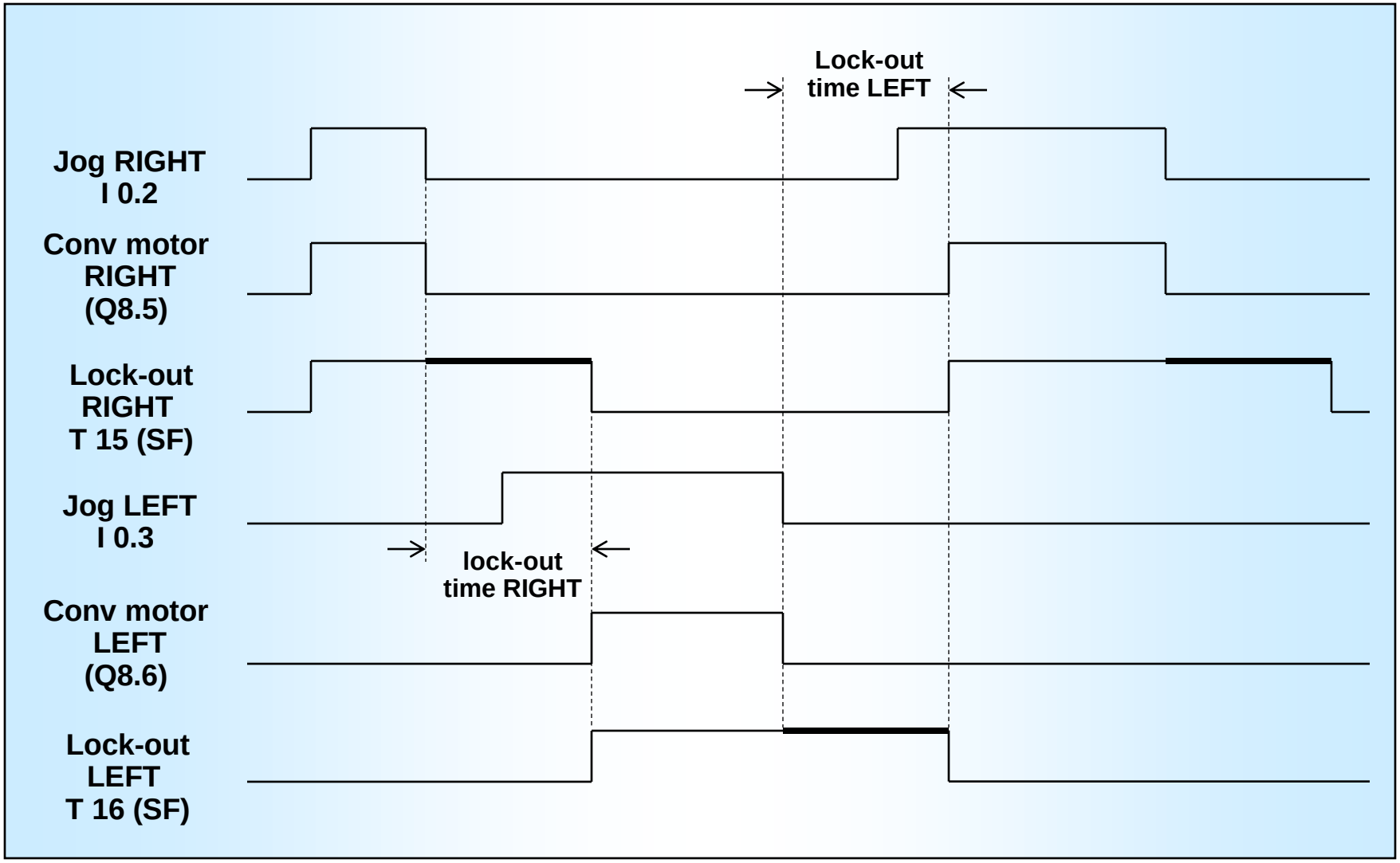
Compare ACT (MW 20) to SETP (MW 22) -> Place comparison result on indicator light

If You Want to Know More

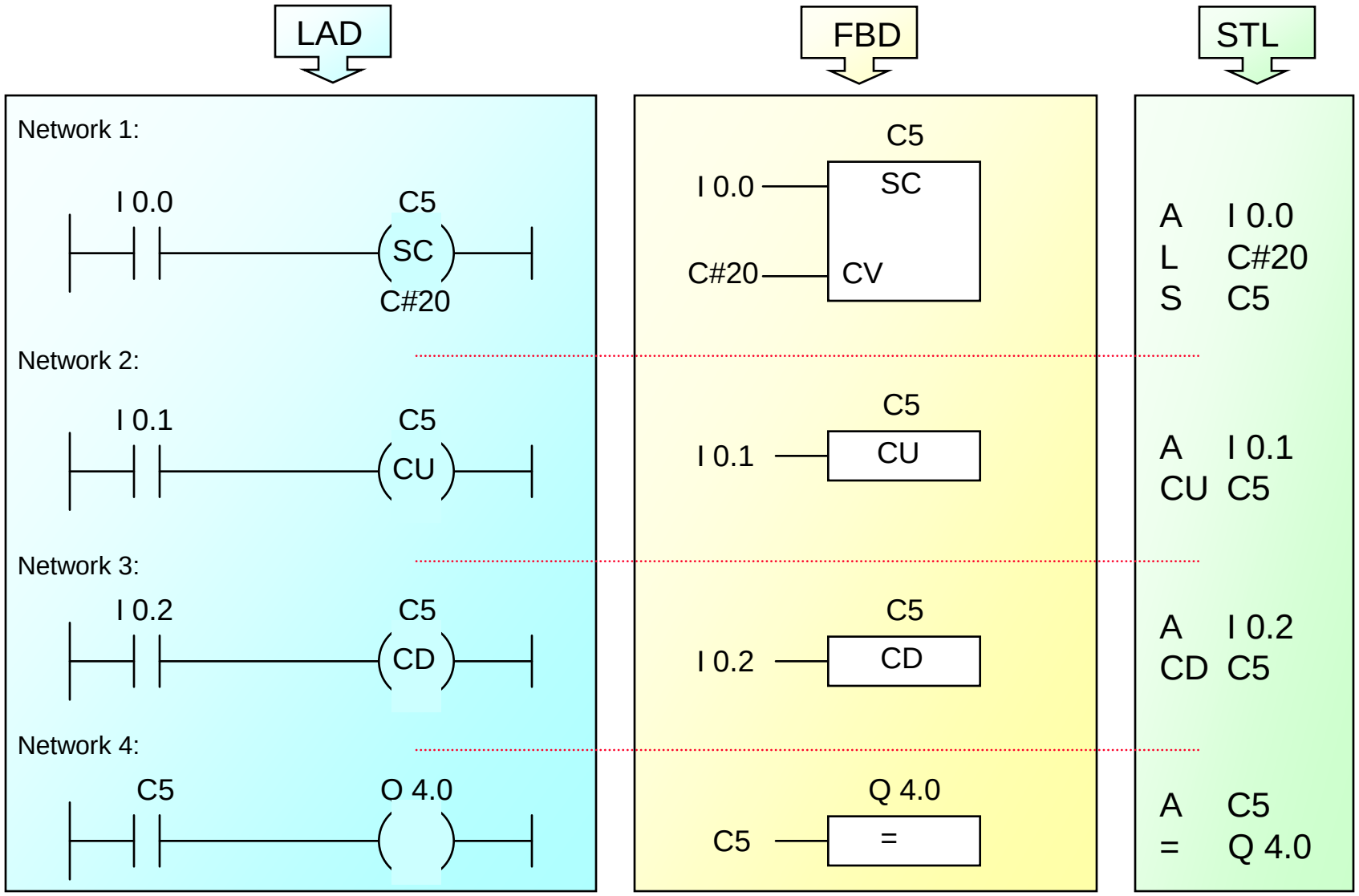
Additional Exercise 4: Counting the Conveyor Belt Faults (FC17)



Additional Ex 5: Timely Lock-out of the Conveyor Motor Jogging

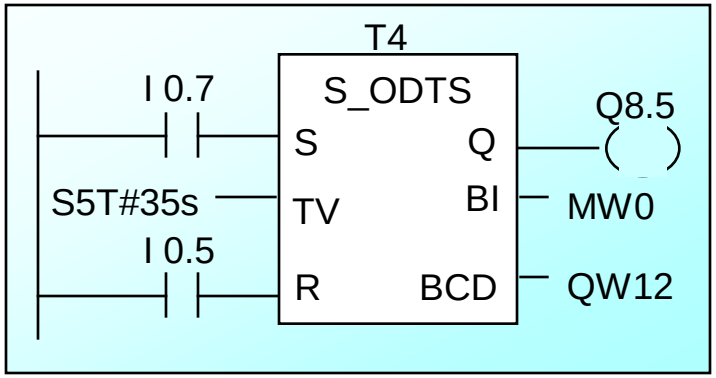


Counters: Bit Instructions

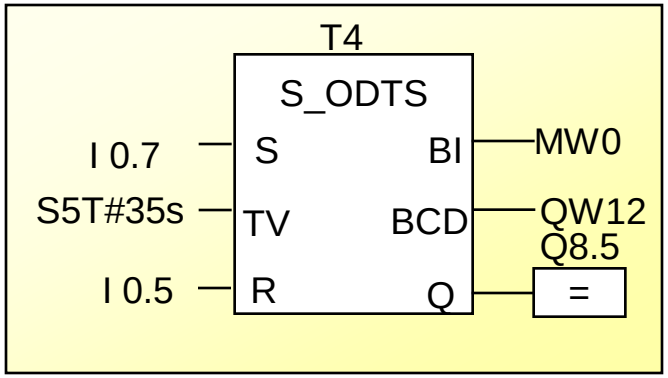


Timers: Stored ON Delay (SS)

LAD



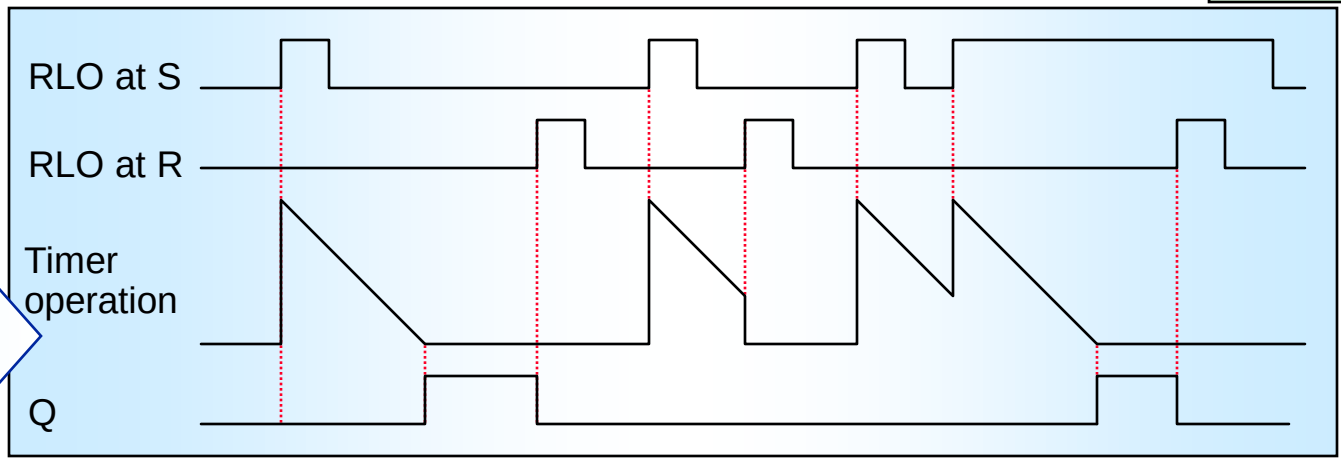
FBD



STL

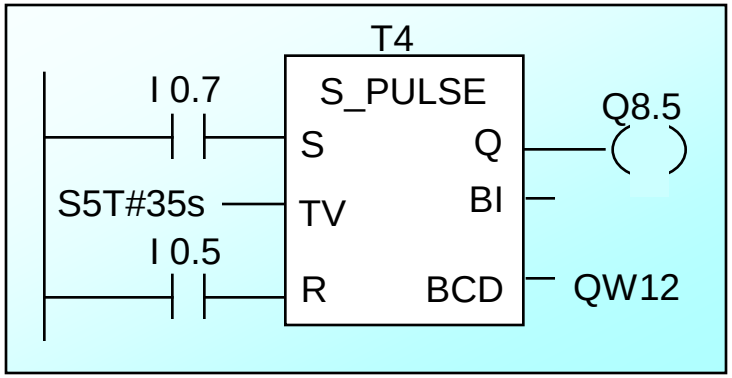
```
A    I 0.7
L    S5T#35s
SS   T4
A    I 0.5
R    T4
L    T4
T    MW0
LC   T4
T    QW12
A    T4
=    Q8.5
```

Example

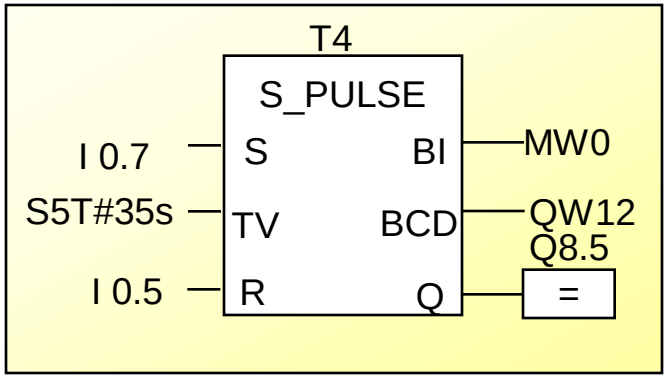


Timers: Pulse (SP)

LAD



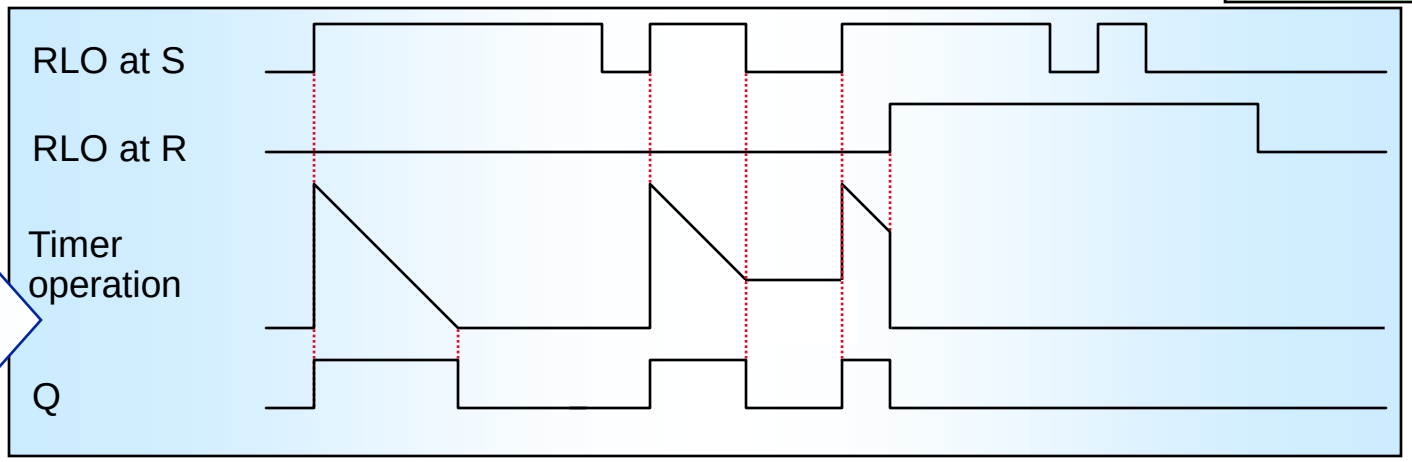
FBD



STL

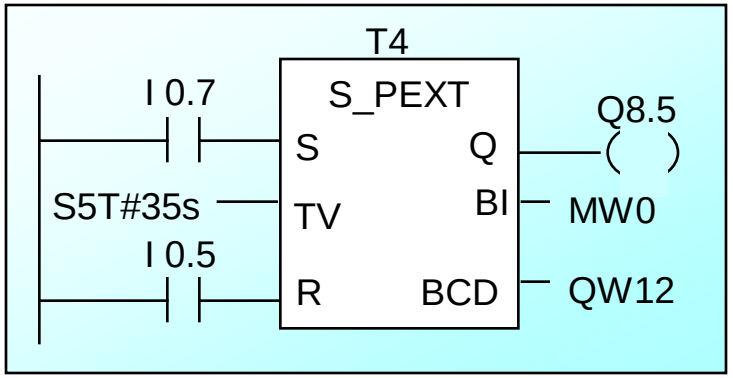
```
A    I 0.7
L    S5T#35s
SP   T4
A    I 0.5
R    T4
L    T4
T    MW0
LC   T4
T    QW12
A    T4
=    Q8.5
```

Example

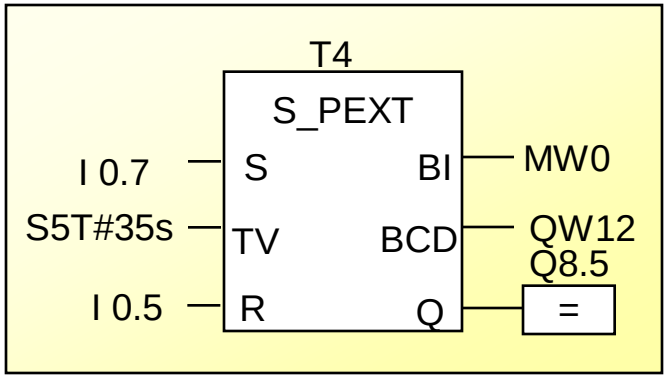


Timers: Extended Pulse (SE)

LAD



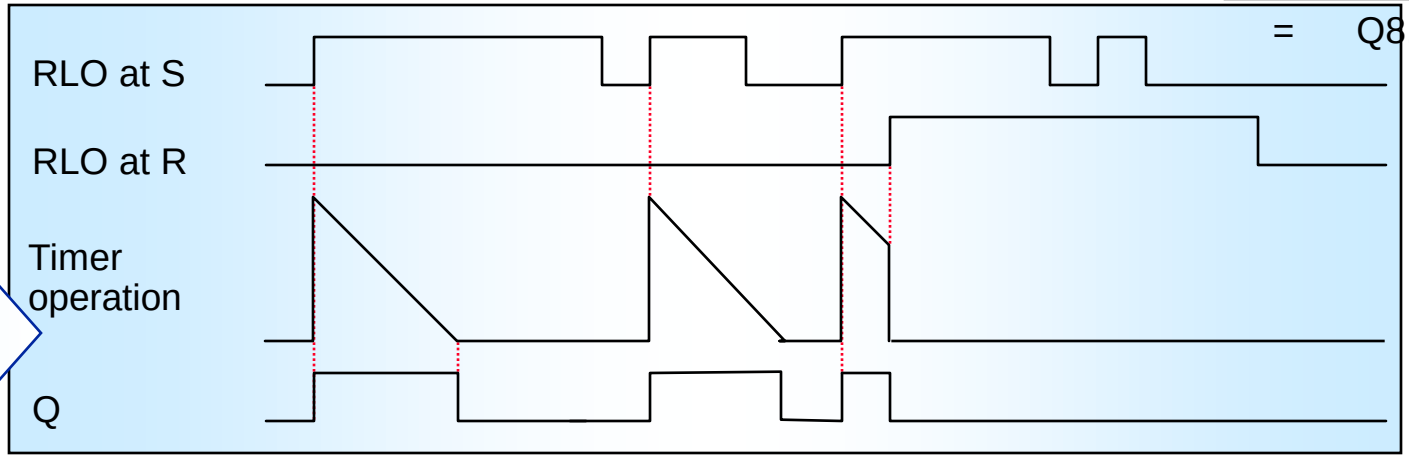
FBD



STL

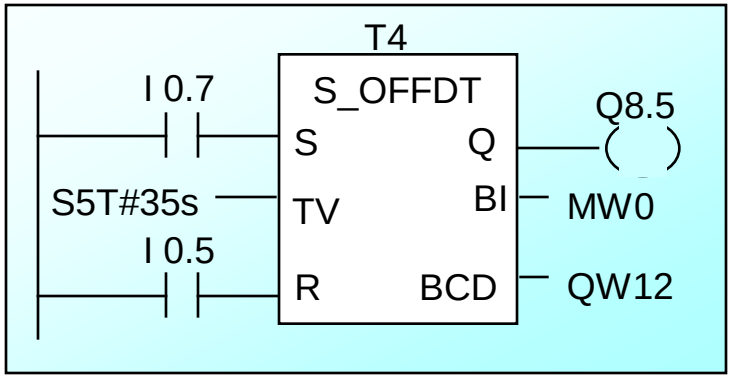
```
A     I 0.7
L
S5T#35s
SE    T4
A     I 0.5
R     T4
L     T4
T     MW0
LC    T4
T     QW12
A     T4
=     Q8.5
```

Example

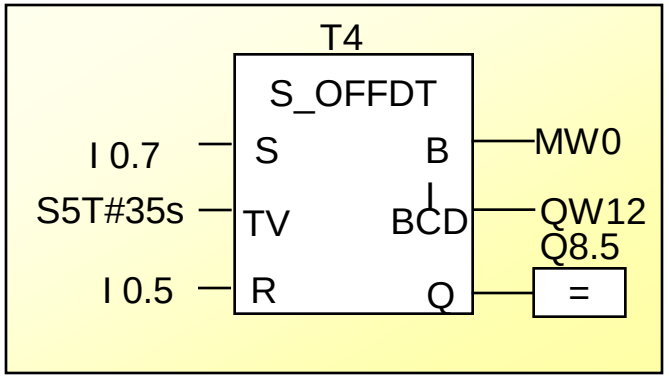


Timers: OFF Delay (SF)

LAD



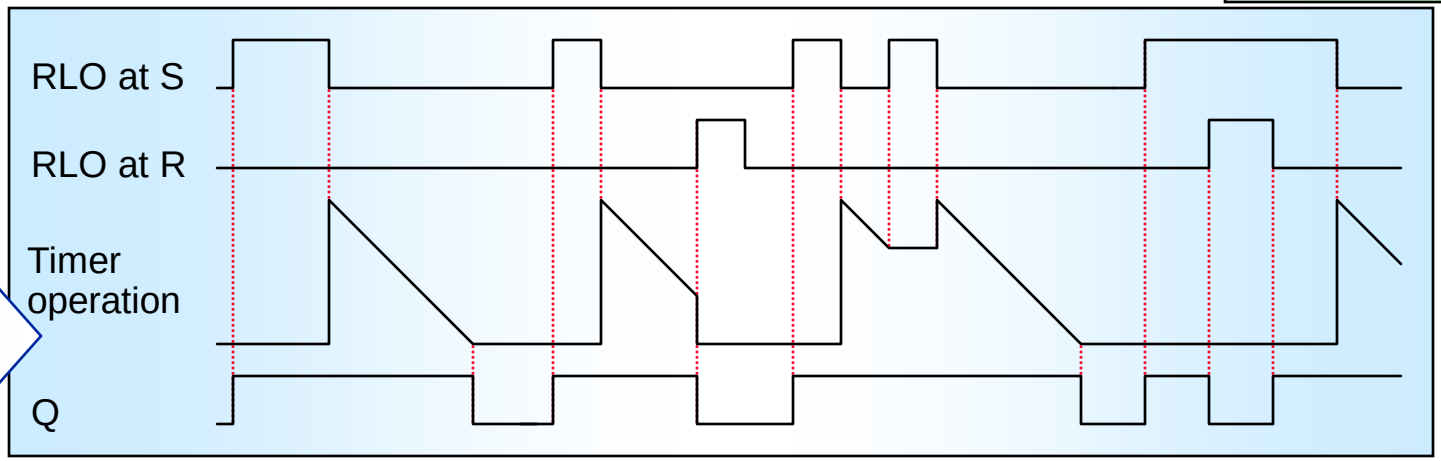
FBD



STL

```
A    I 0.7
L    S5T#35s
SF   T4
A    I 0.5
R    T4
L    T4
T    MW0
LC   T4
T    QW12
A    T4
=    Q8.5
```

Example



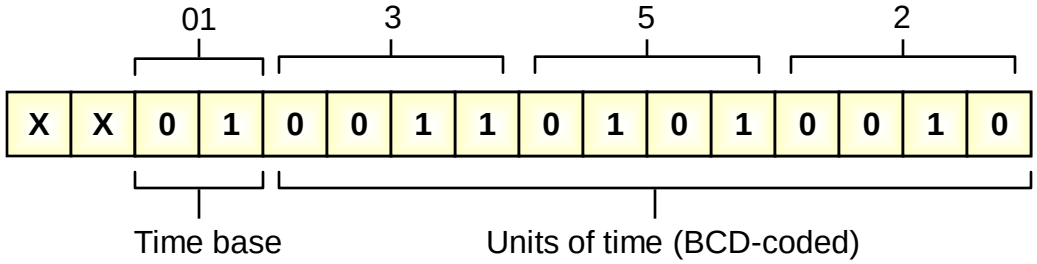
Time Formats for Traditional S5-Timers in STEP 7

Time specifications as constants

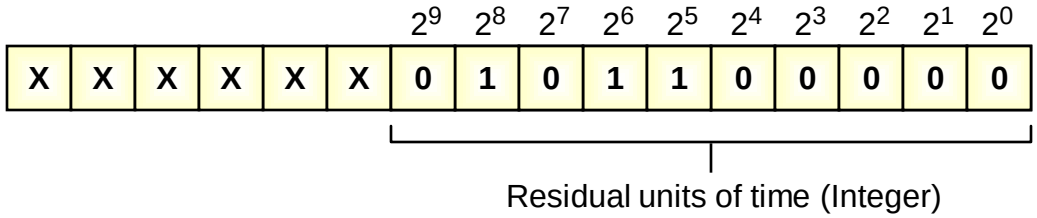
S5T#35s200ms

(Time base: 01 (100ms), Number of units of time: 352)

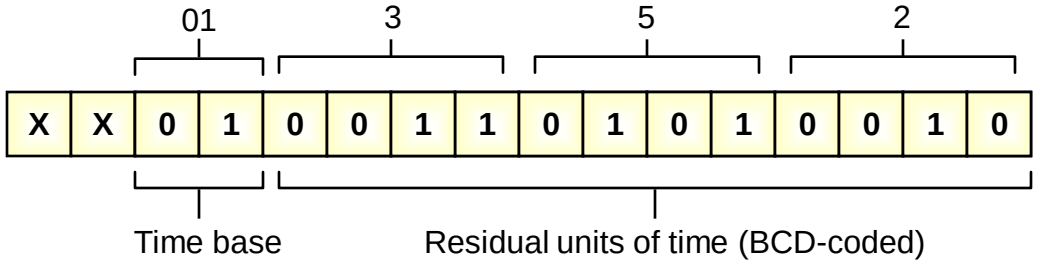
Time specifications per variable



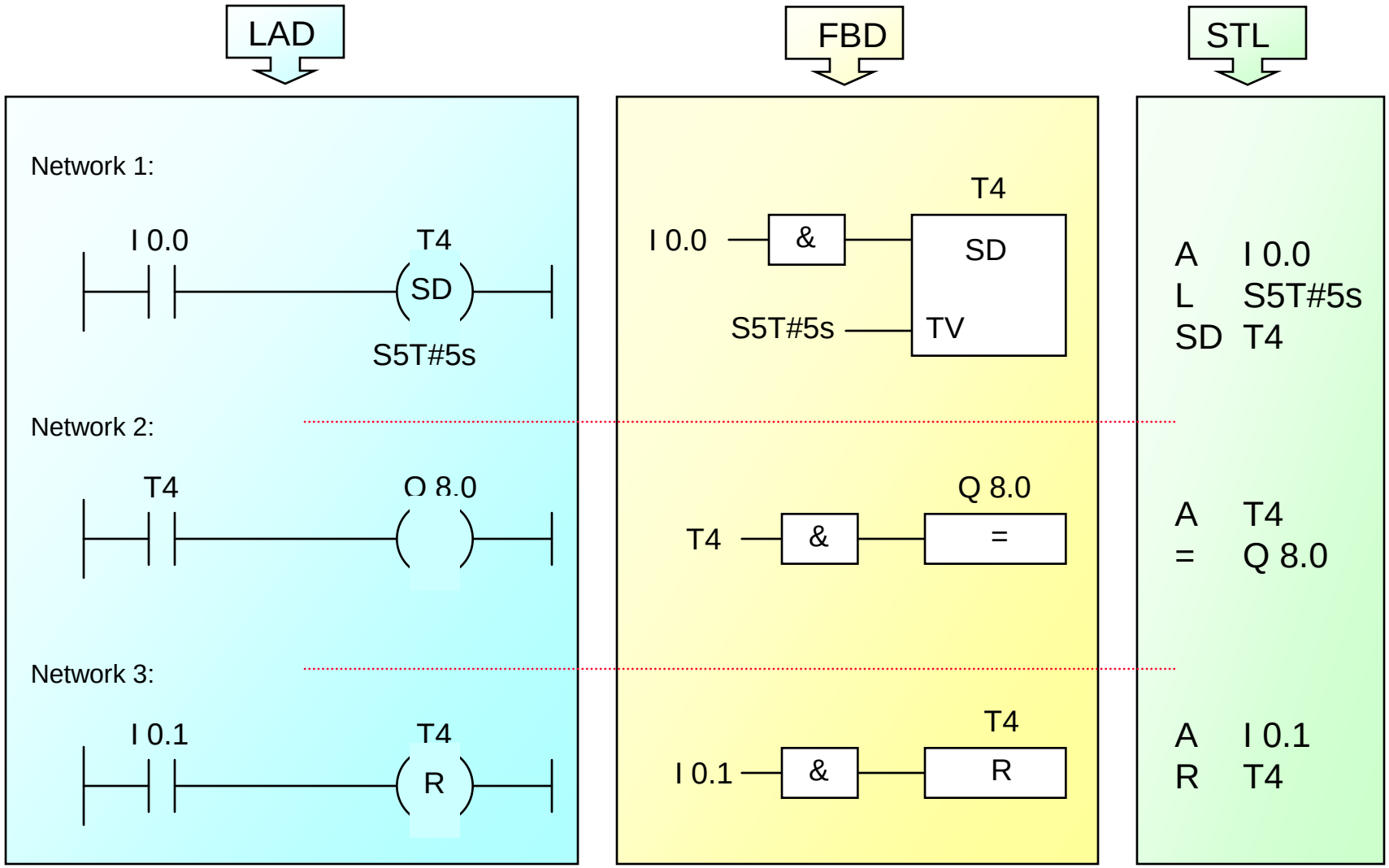
Accu 1 contents after “L T...” exec.



Accu 1 cont. after “LC T...” exec.



Timers: Bit Instructions



Conversion Operations I → DI → REAL

Data in integer format
(16-bit integer)

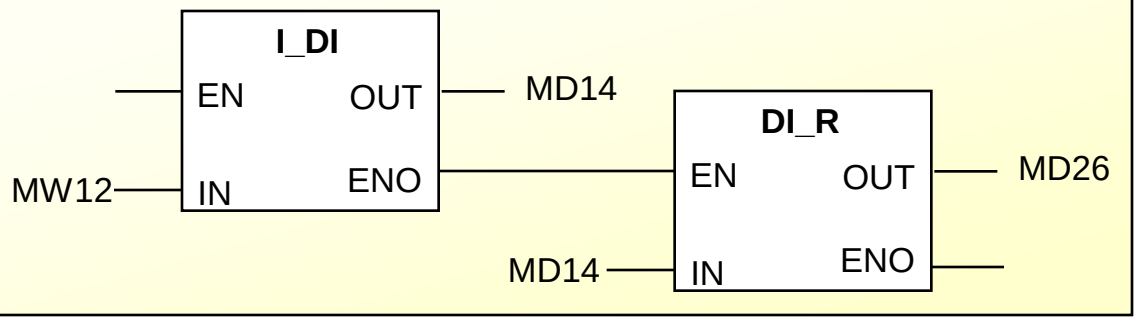
Task

Conversion from integer to double integer

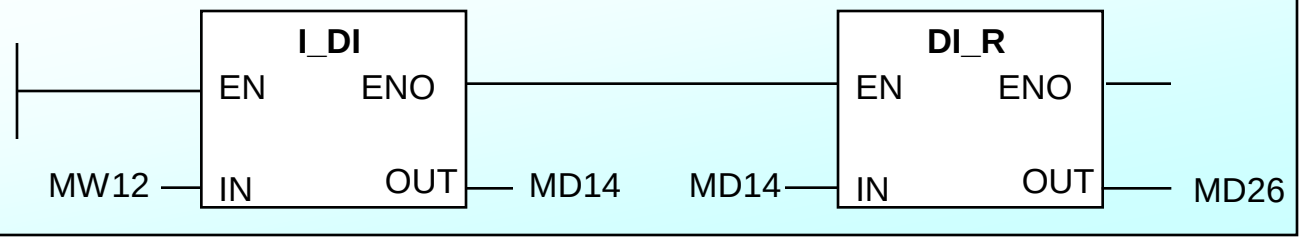
Conversion from double integer to real number

Math program with real numbers

FBD



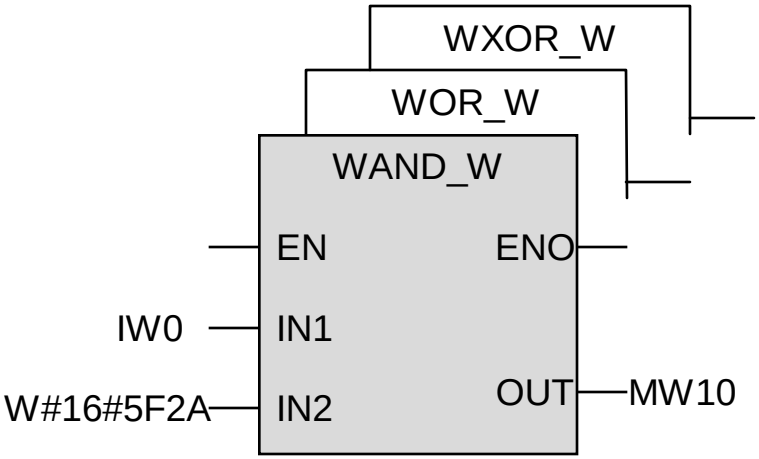
LAD



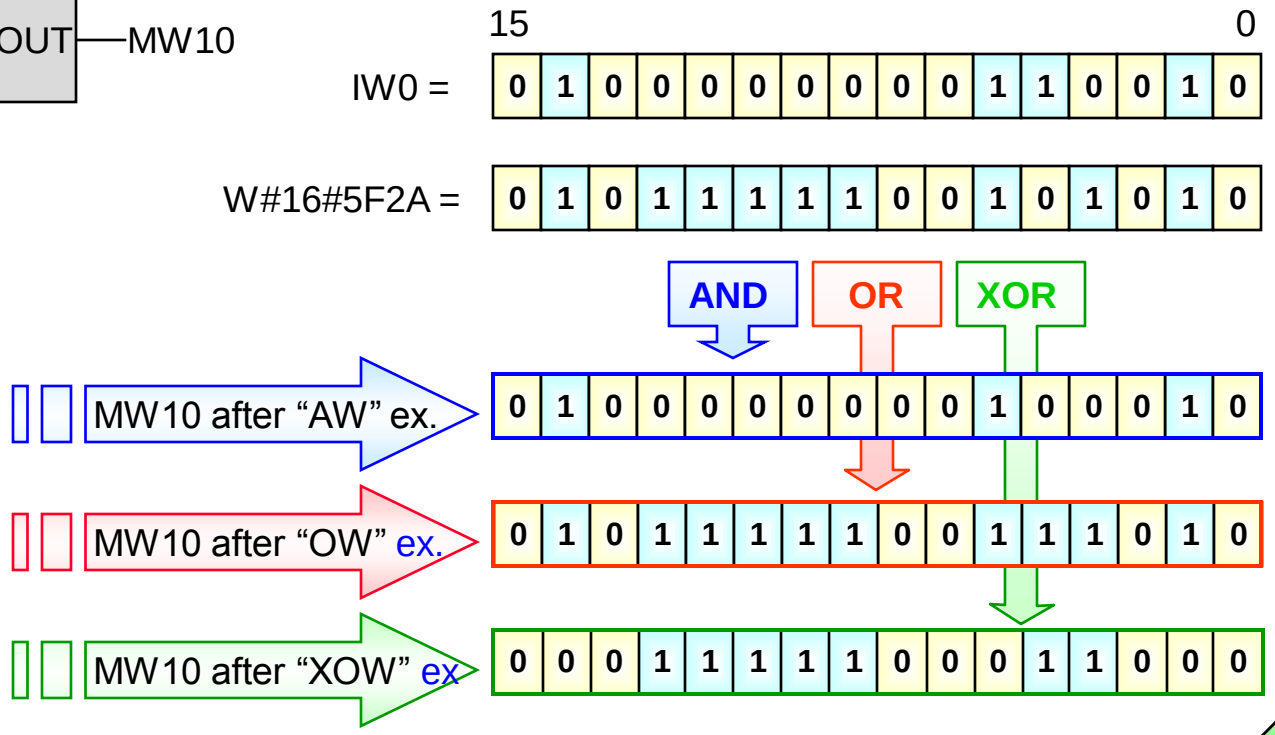
STL

```
L    MW12
ITD
DTR
T    MD26
```

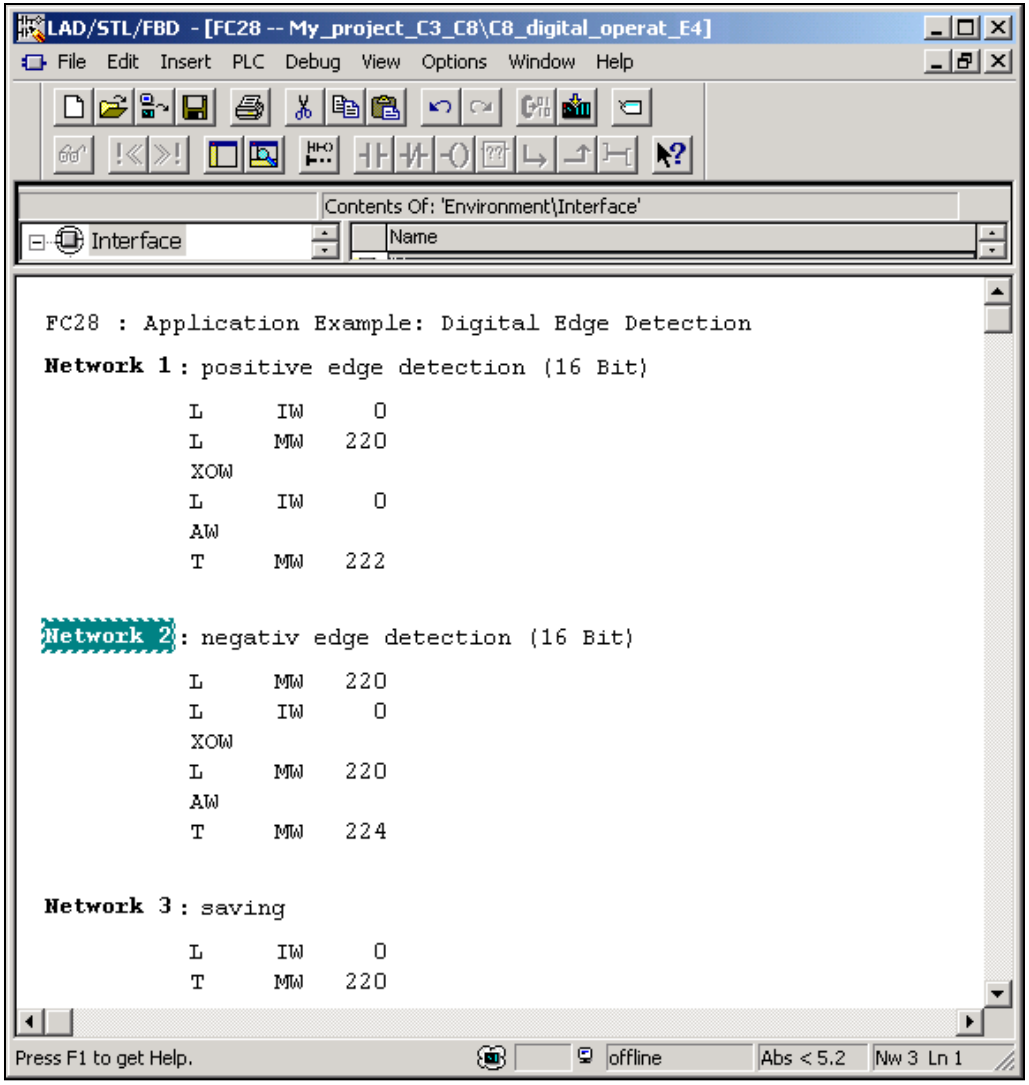
Digital Logic Operations



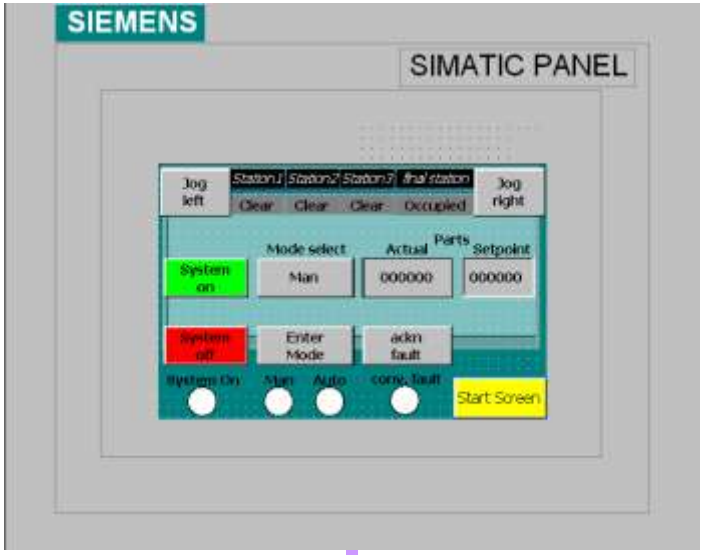

```
L     IW 0
L     W#16#5F2A
AW / OW / XOW
T     MW10
```

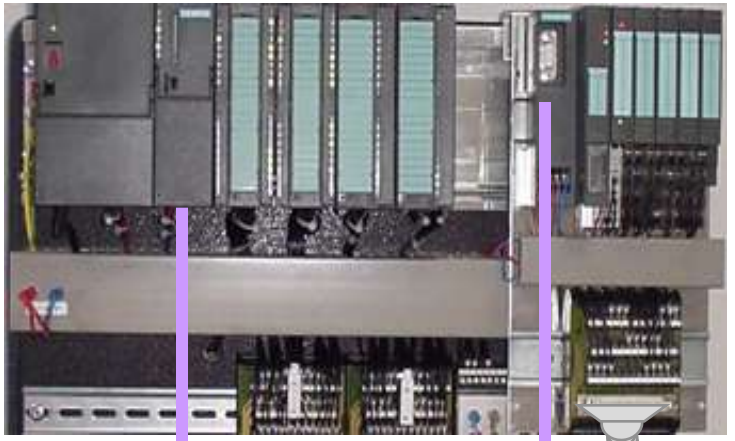


Application Example: Digital Edge Detection

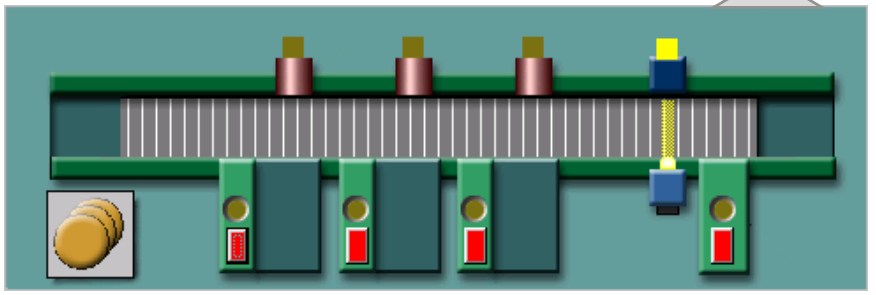


Introduction to PROFIBUS DP and the HMI Operator Panel (1)





PROFIBUS DP



Objectives

Upon completion of the chapter the participant will ...

… be familiar with the use of PROFIBUS DP

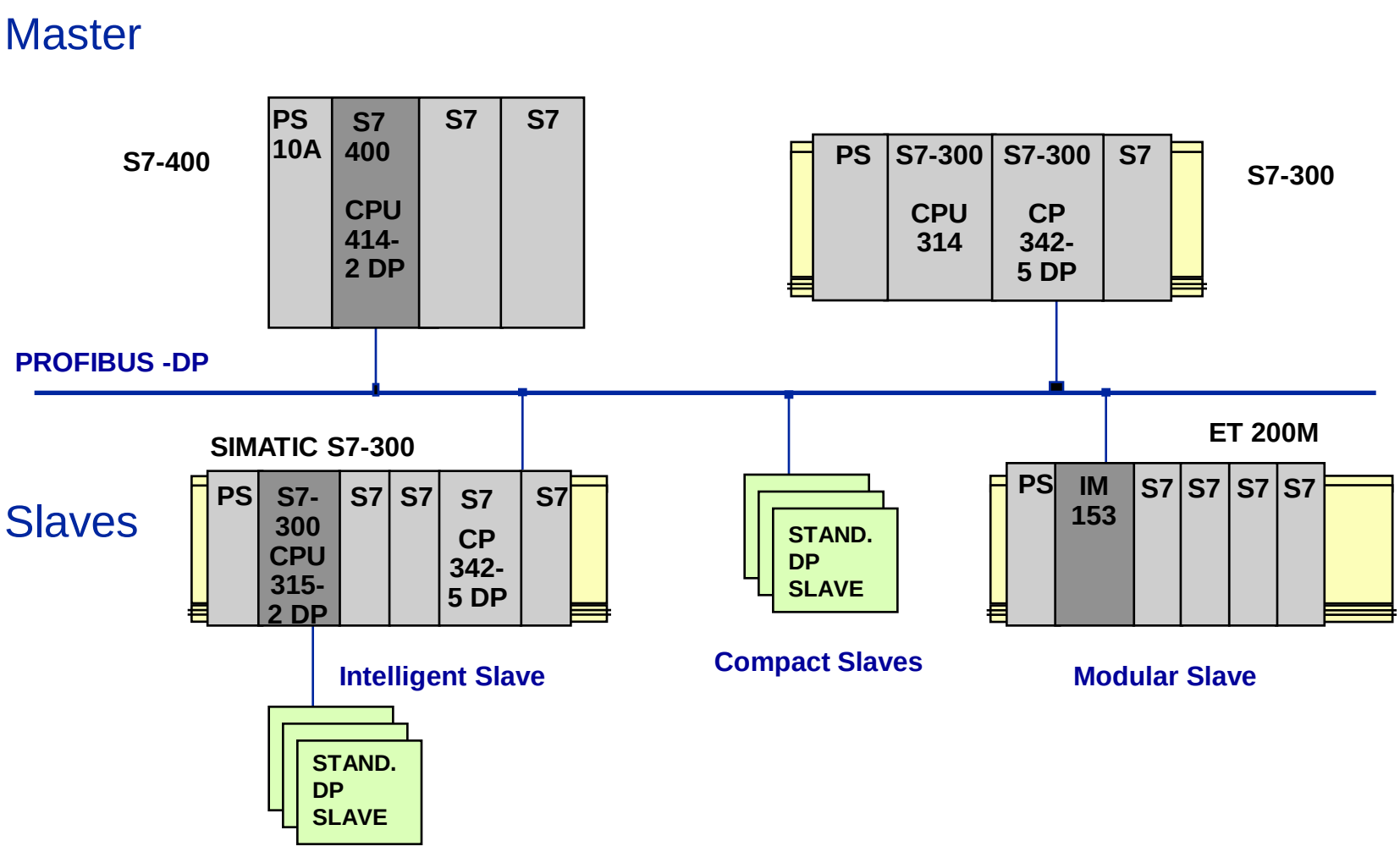
... be familiar with the WinCC flexible software

... be able to set the interface of the HMI operator panel

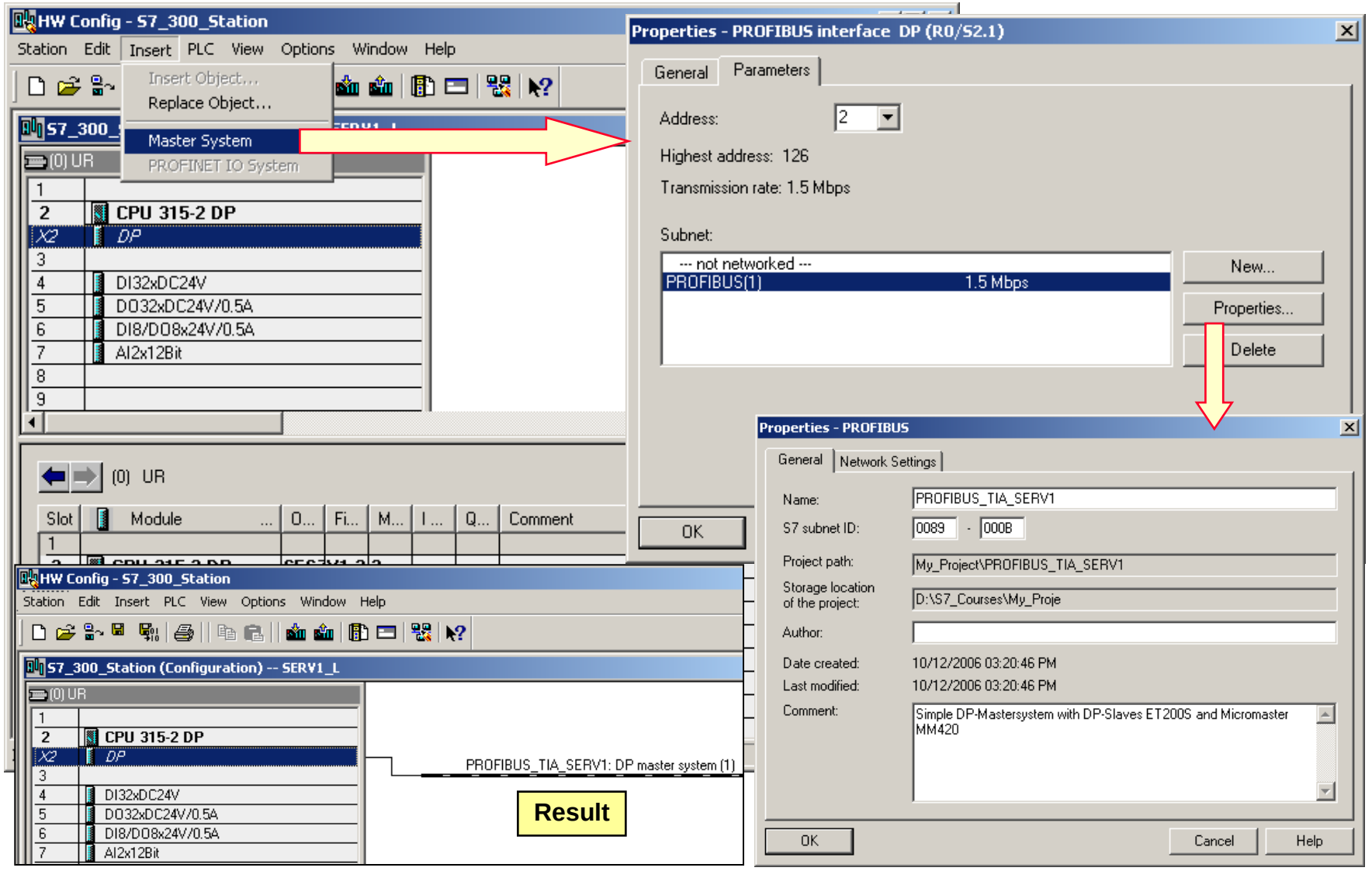
... be able to download a project to the HMI operator panel

... be familiar with the principle of interfacing the HMI operator panel via tags

Structure of a PROFIBUS DP System

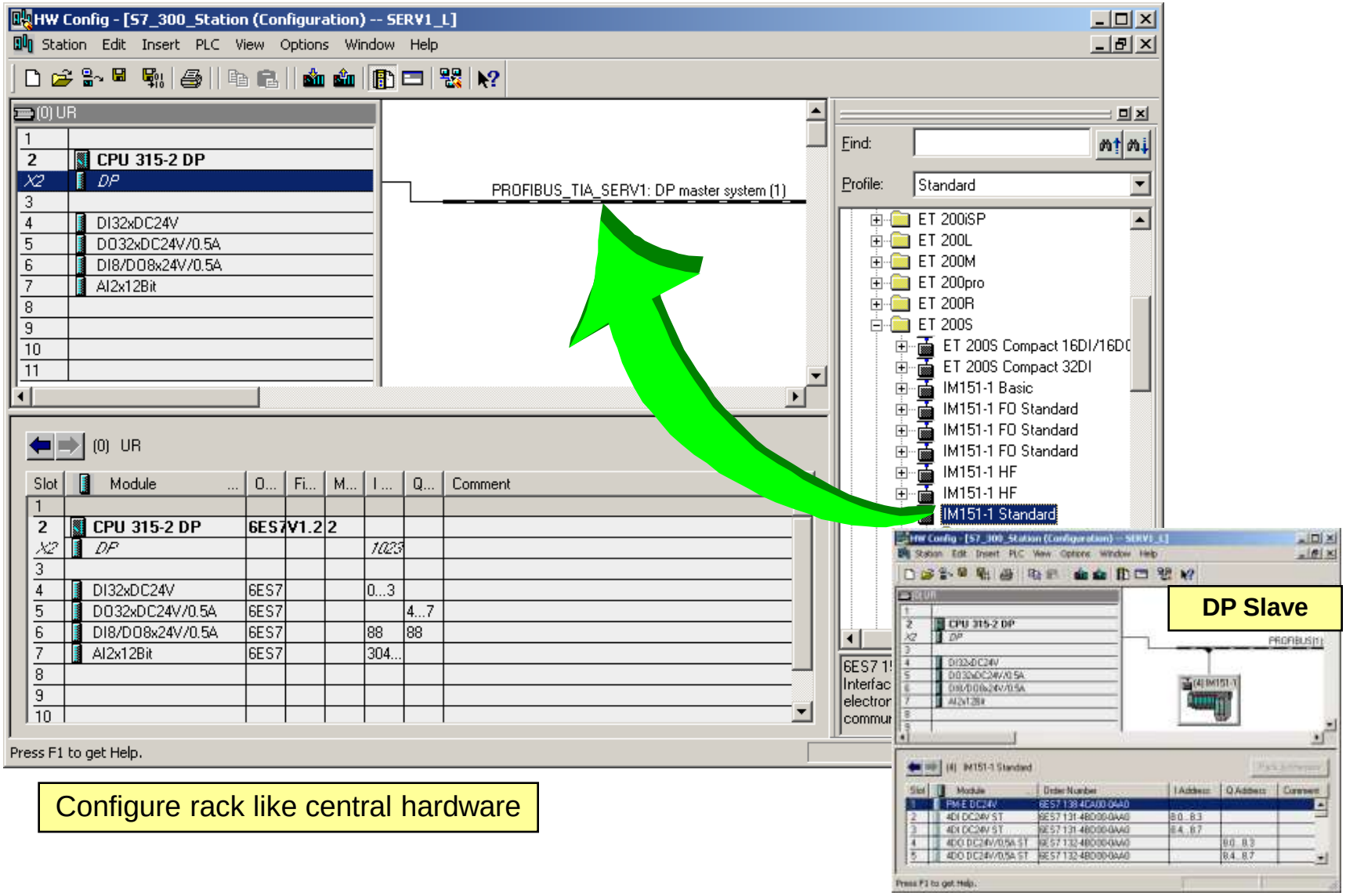


Configuring a PROFIBUS DP Master System



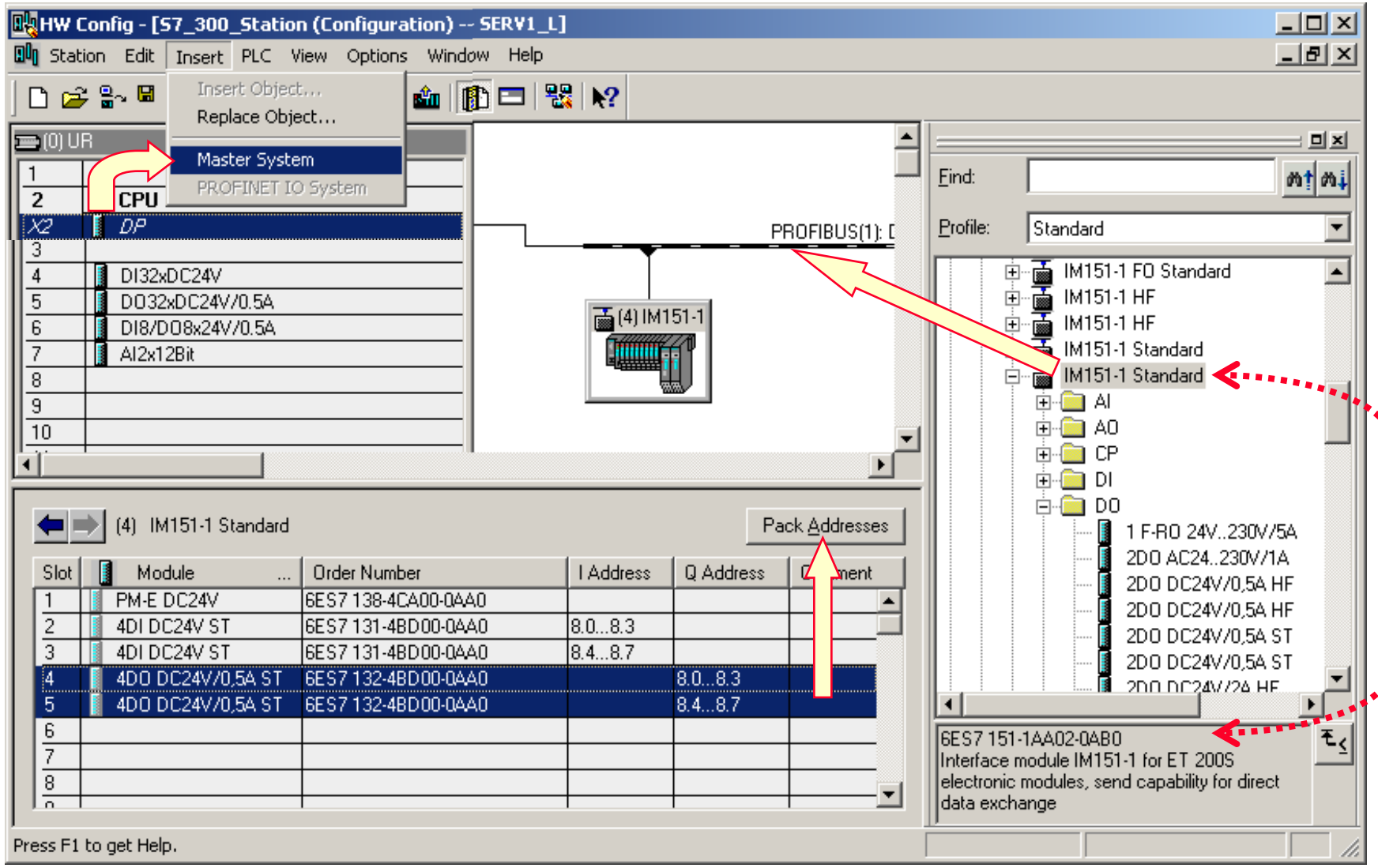
Date: 2/4/2012
File: SERV1_11E.4

Configuring Compact and Modular DP Slaves



Configure rack like central hardware

Exercise 1: Configuring the DP System (ET 200S)



Control and Monitoring with SIMATIC HMI

Process
Visualization System
SIMATIC WinCC

SIMATIC Panels

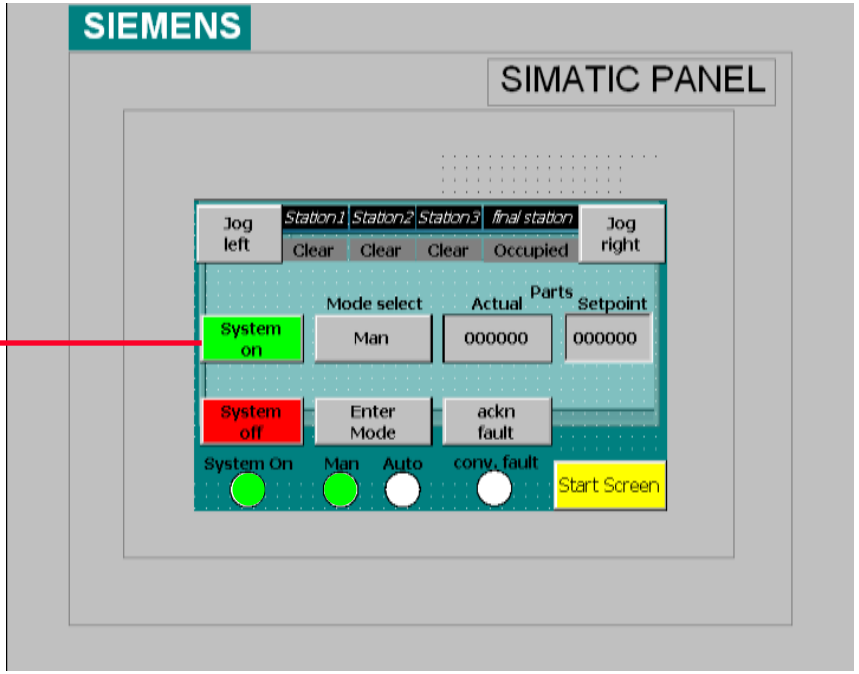
Communication between SIMATIC S7 and the HMI System

- **MPI**
- **PROFIBUS DP**
- **Industrial Ethernet**

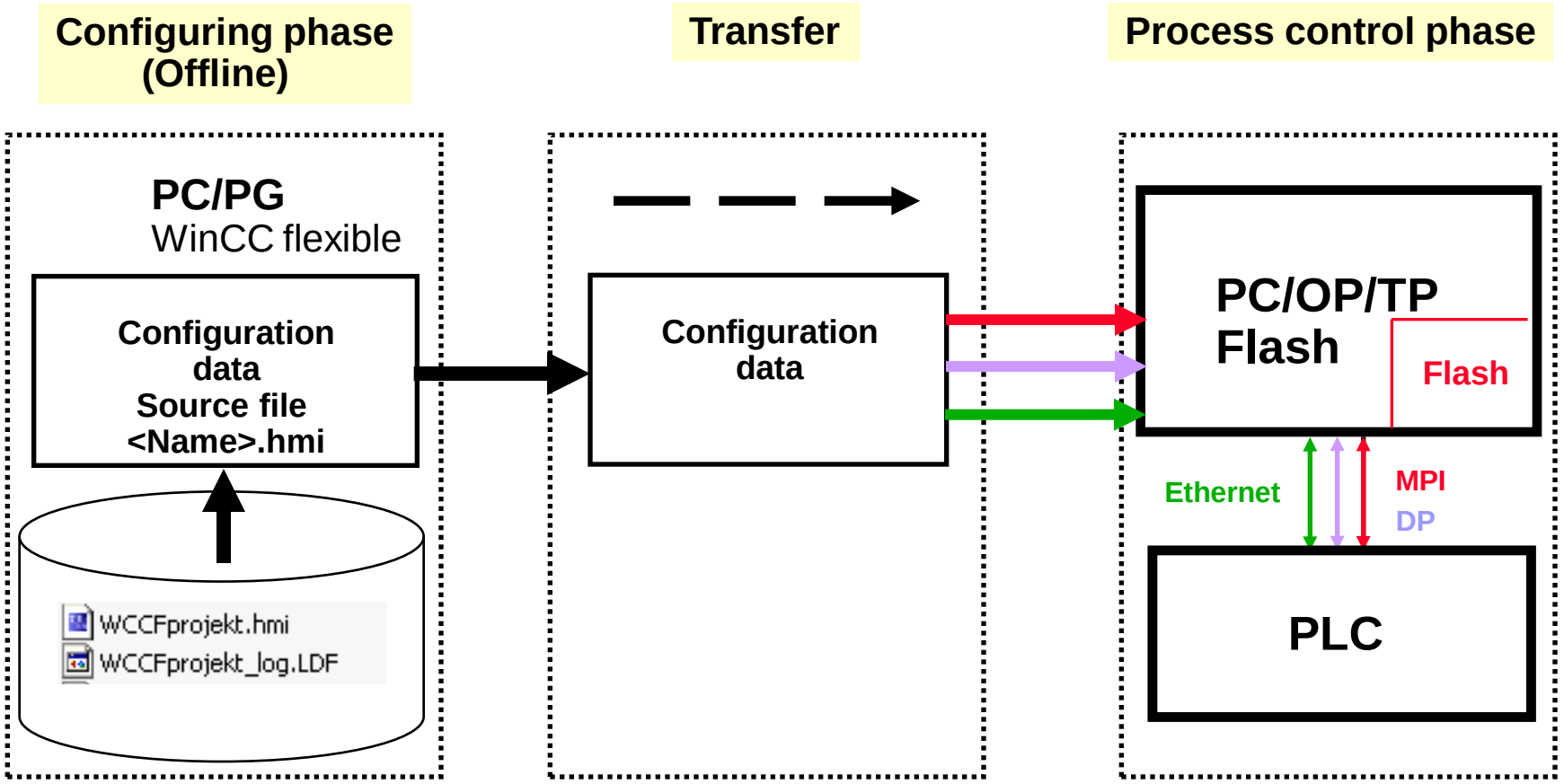
Update

Press key “set bit“

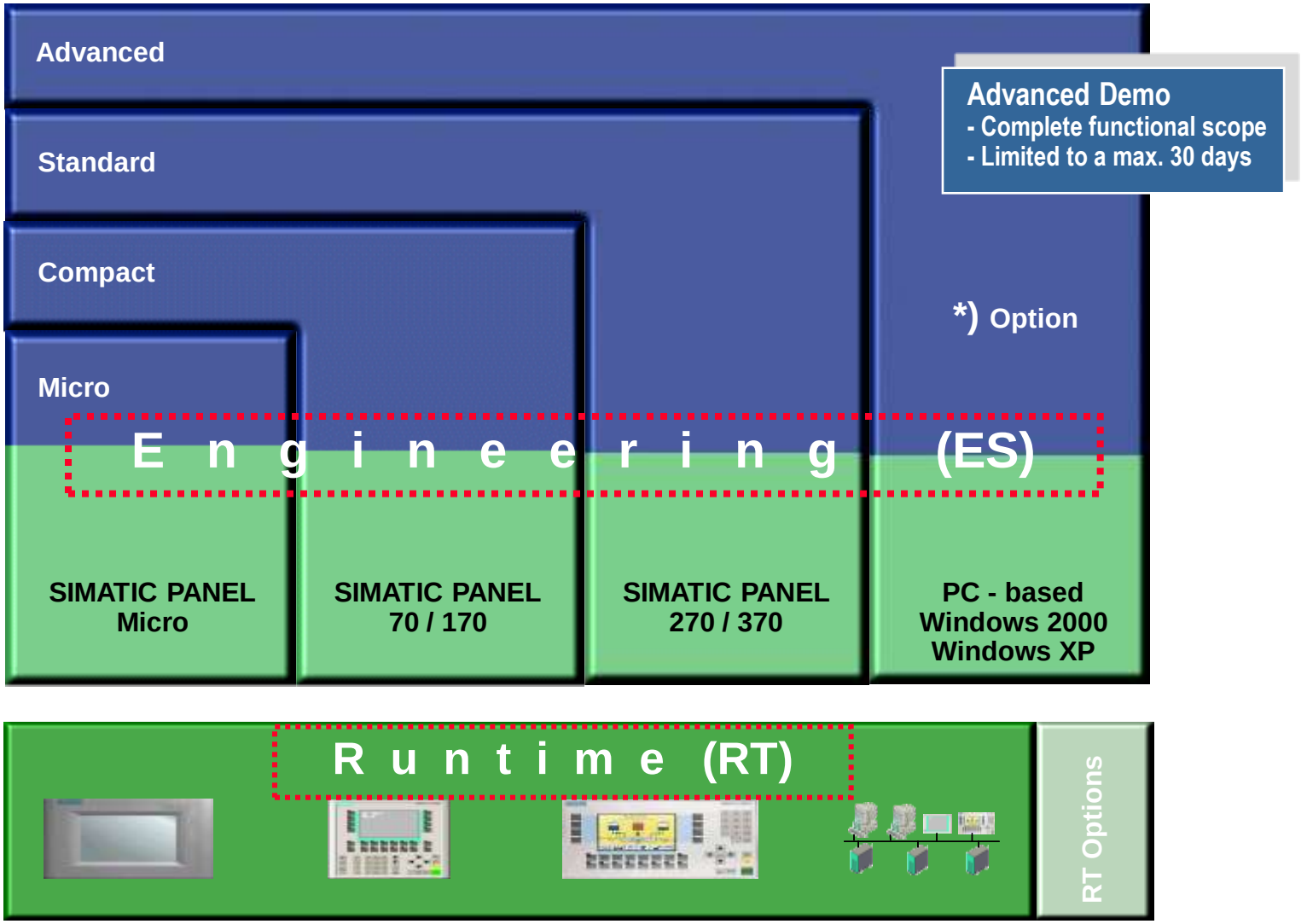
Let go of key “reset bit“



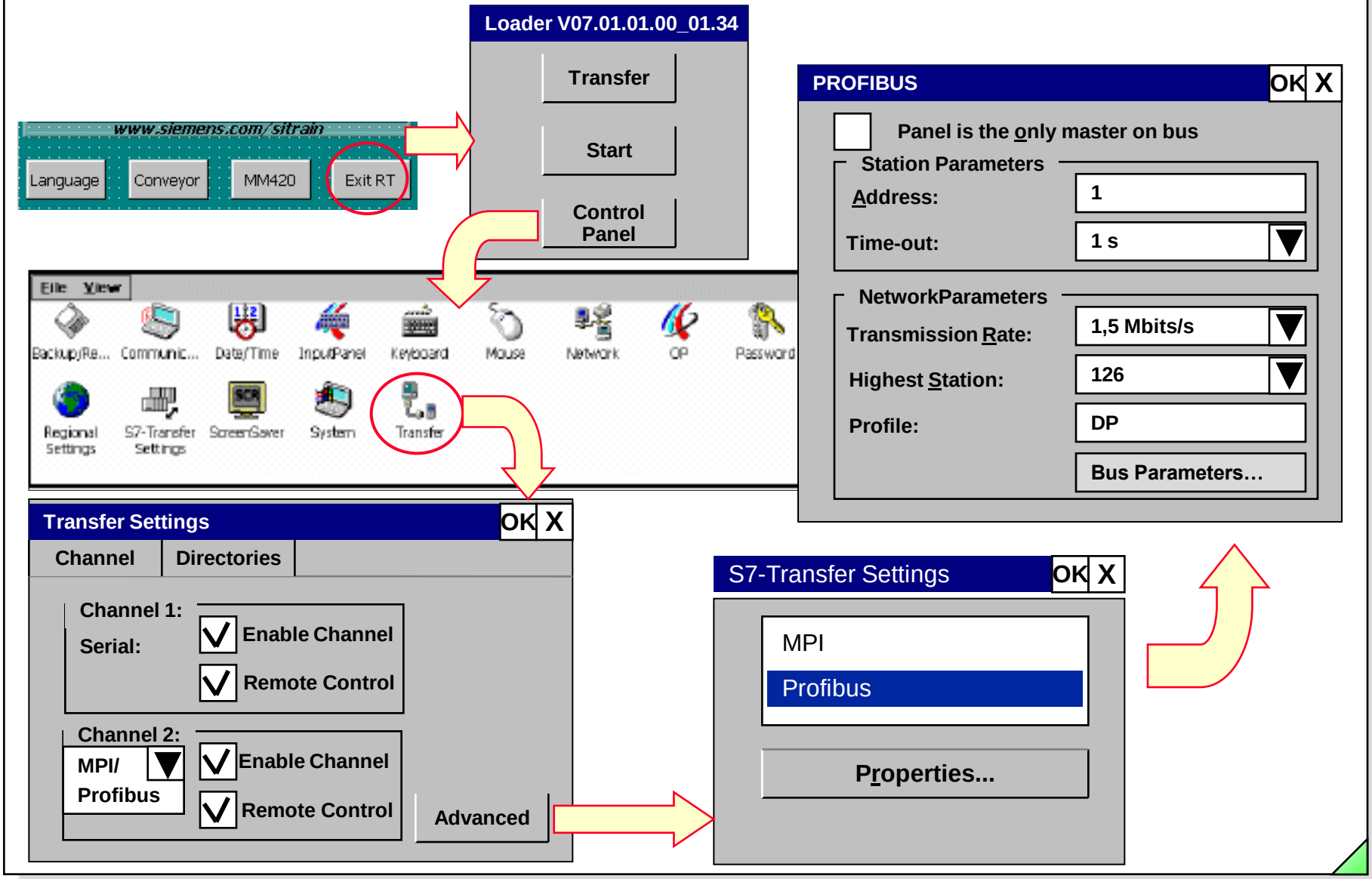
Configuring Phase, Downloading Phase and Process Control Phase



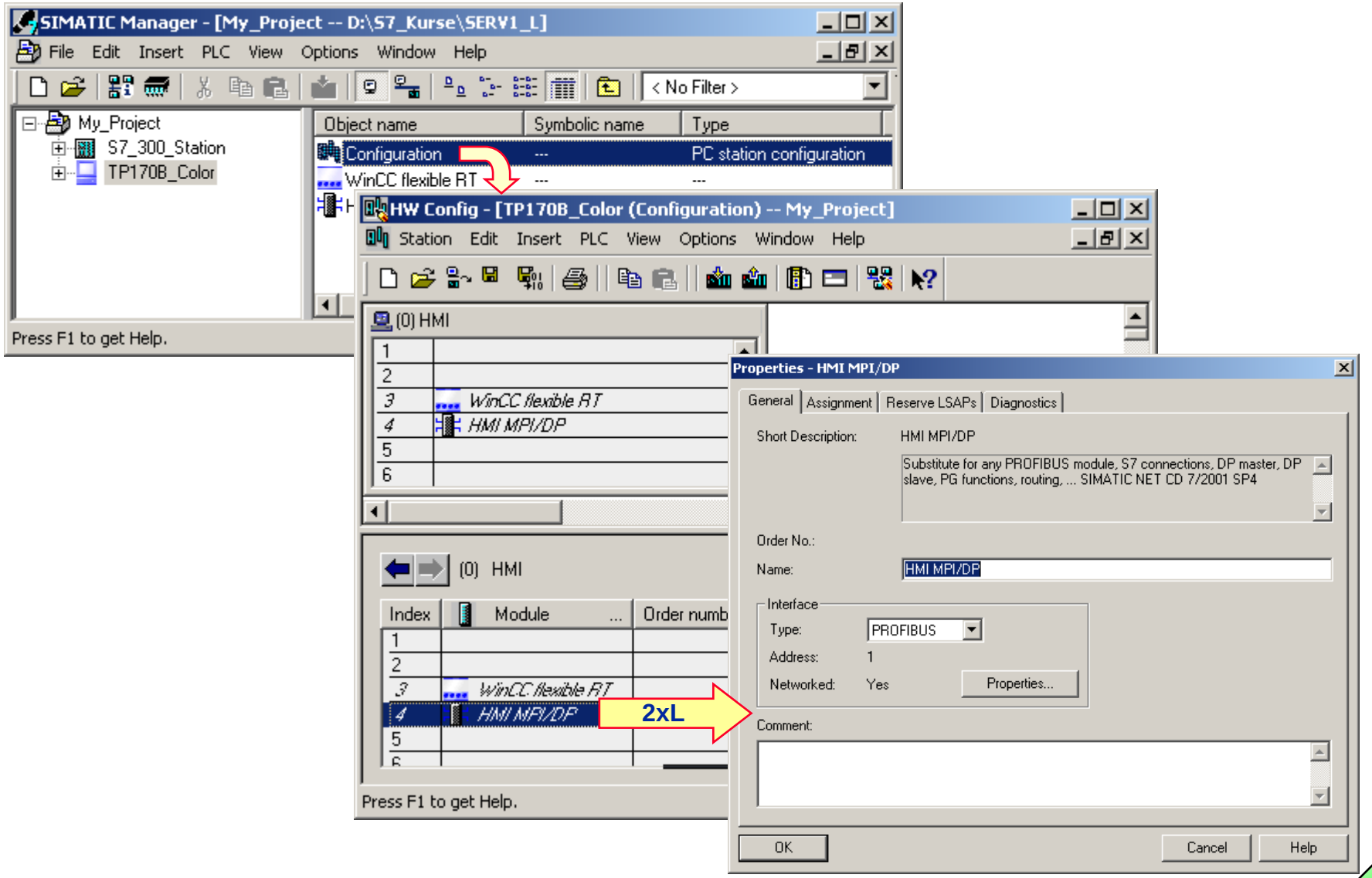
WinCC flexible



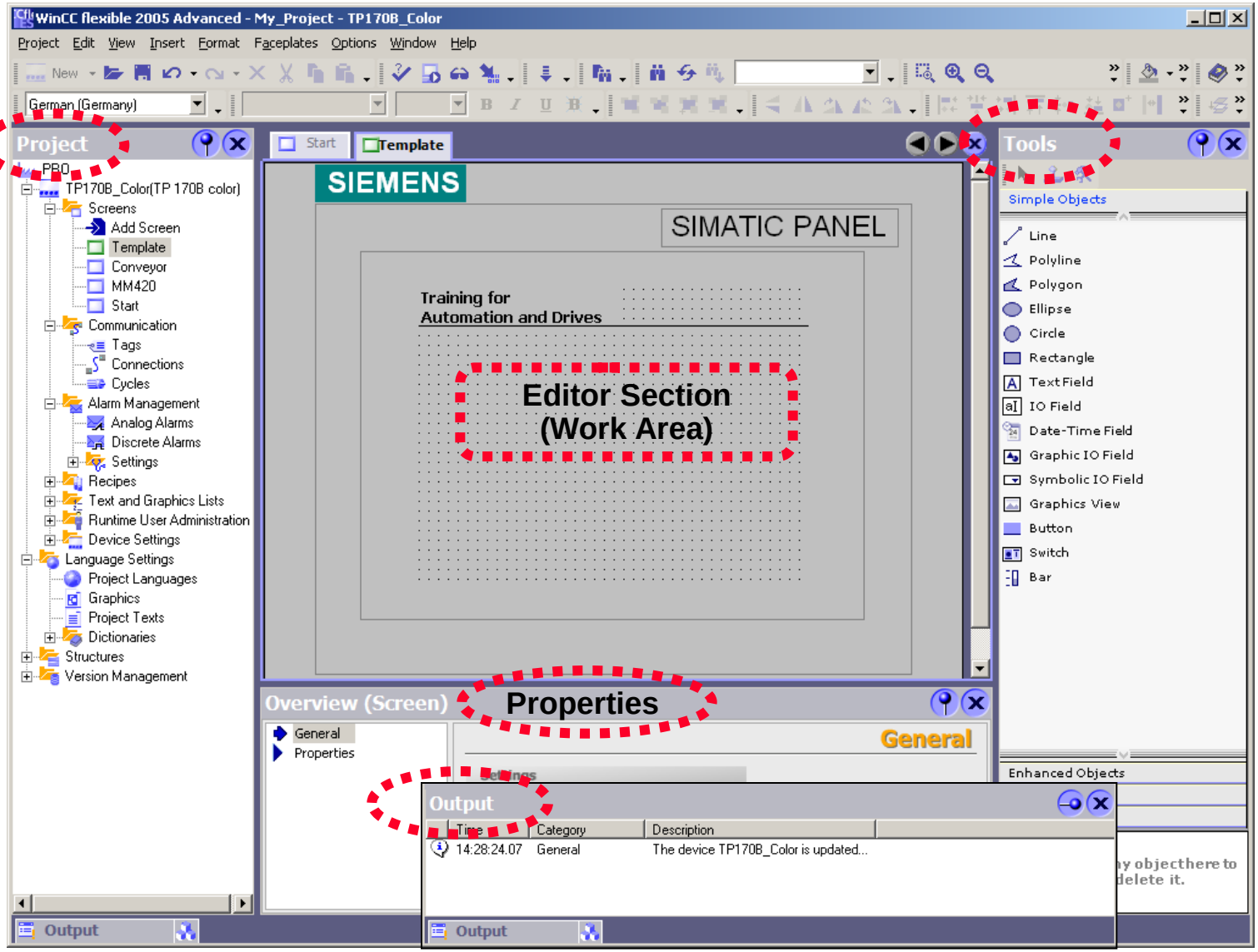
HMI Operator Panel: Setting the Interface for the Project Transfer



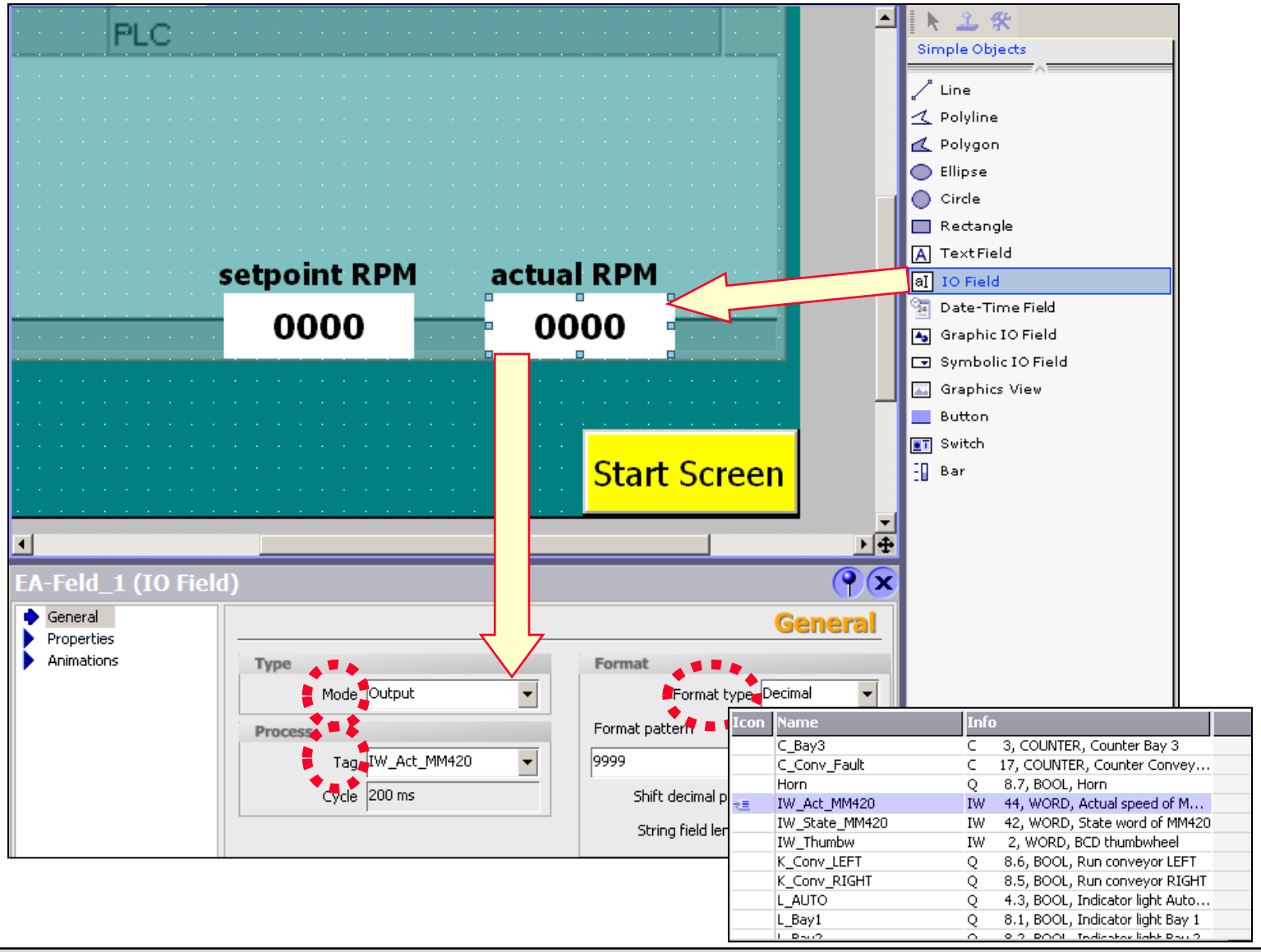
HW Config: Setting the Runtime Interface



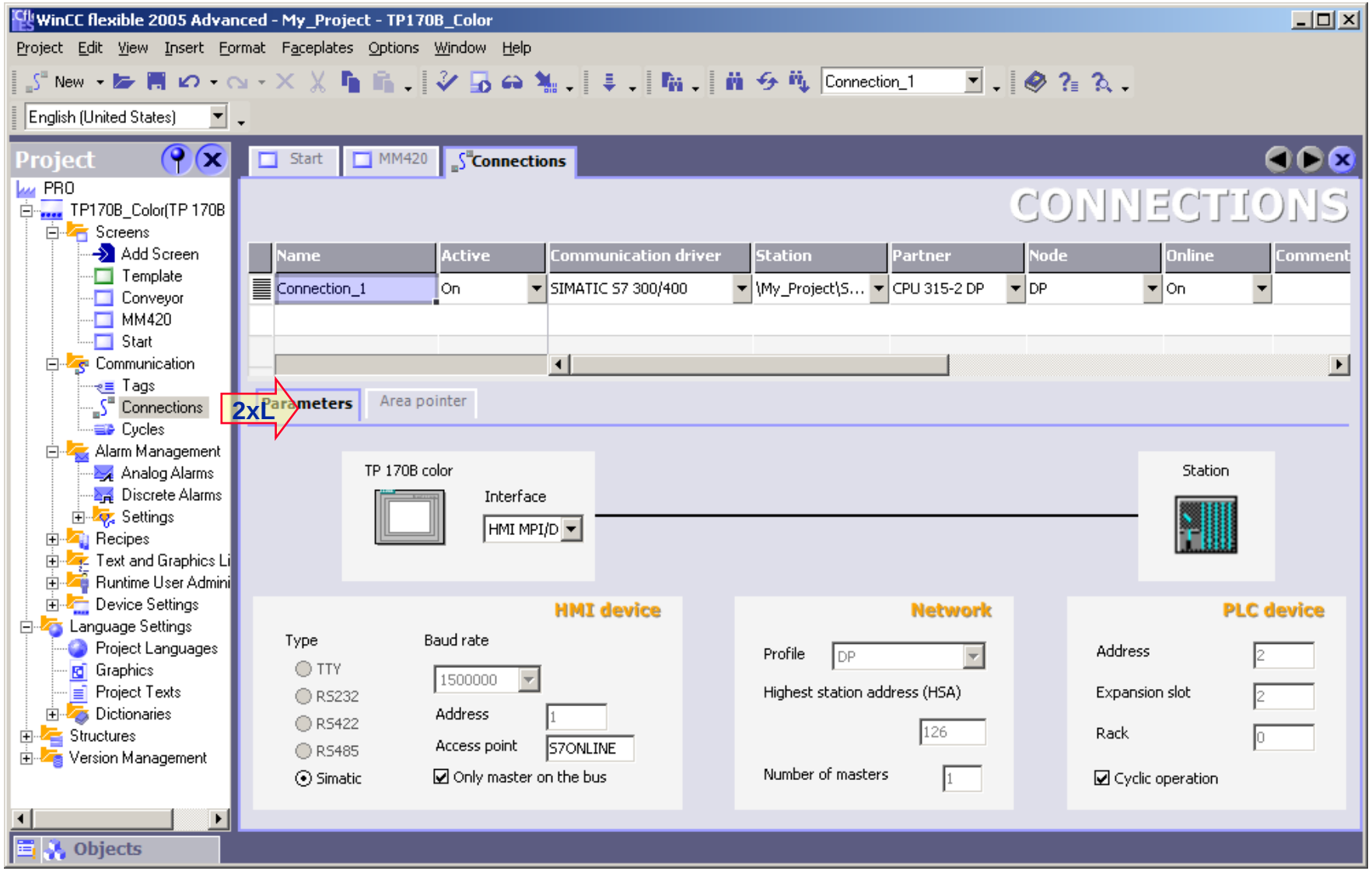
WinCC flexible – Configuration Tool



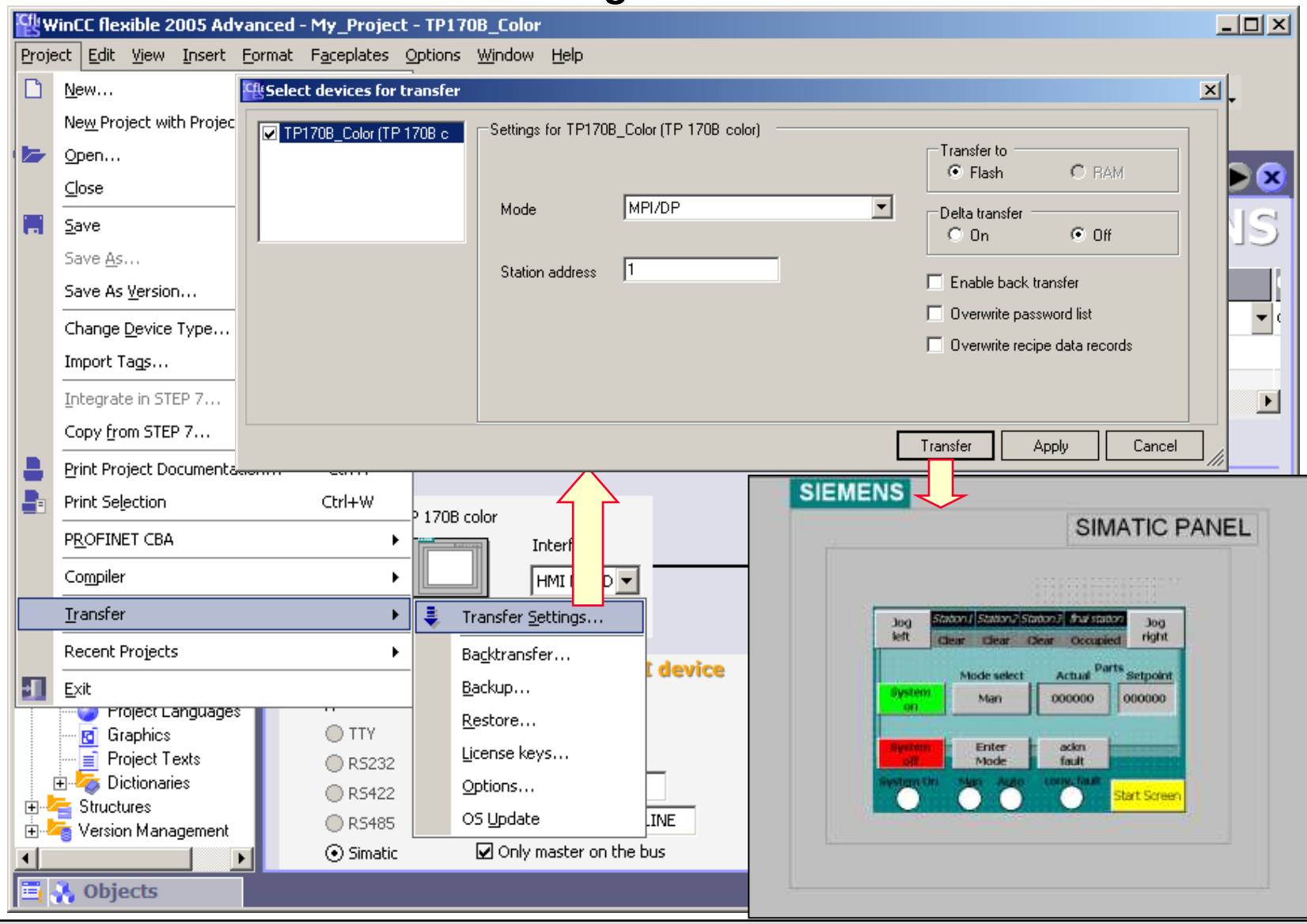
Input and Output Fields



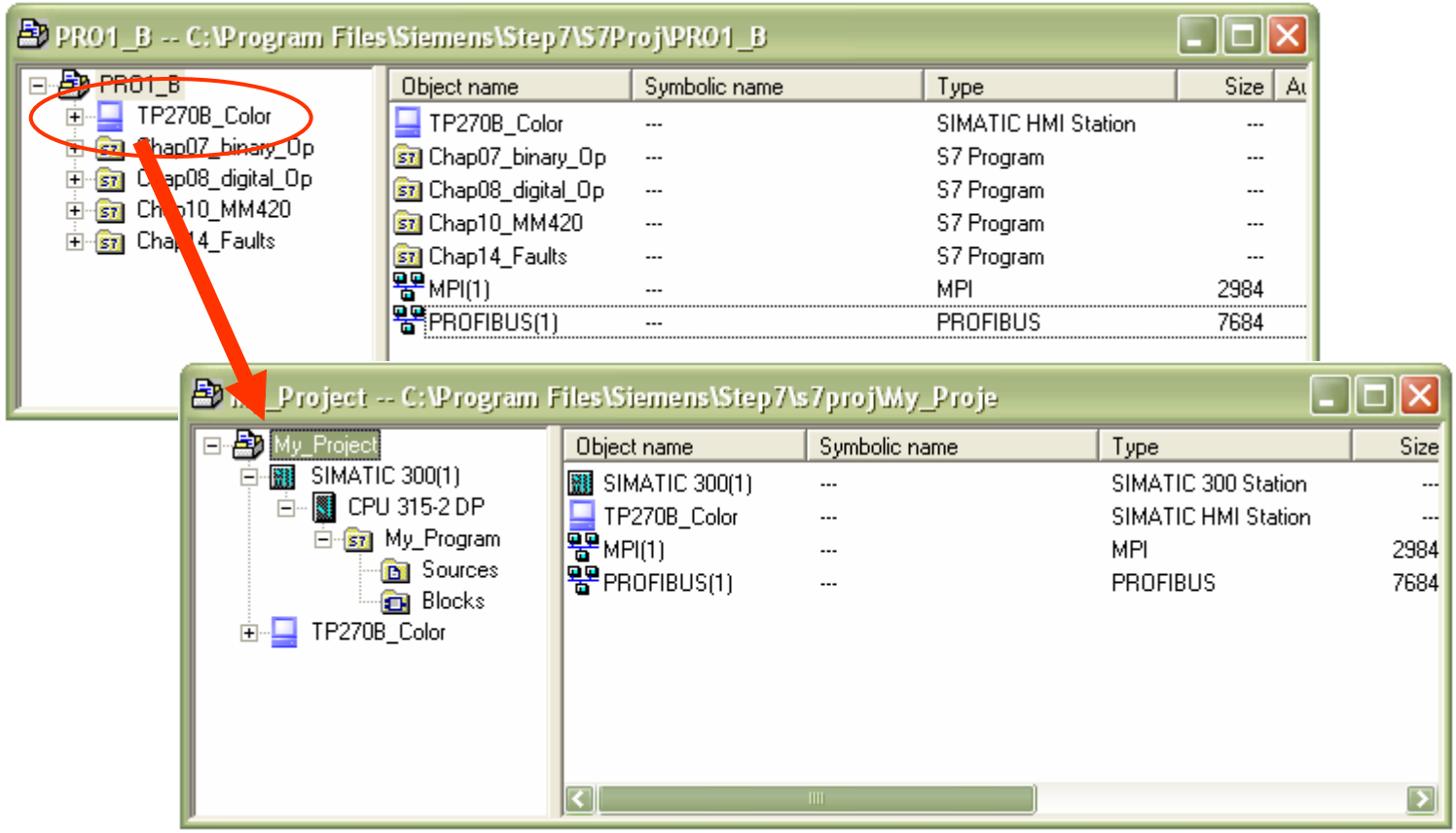
Defining the Connection to the Controller



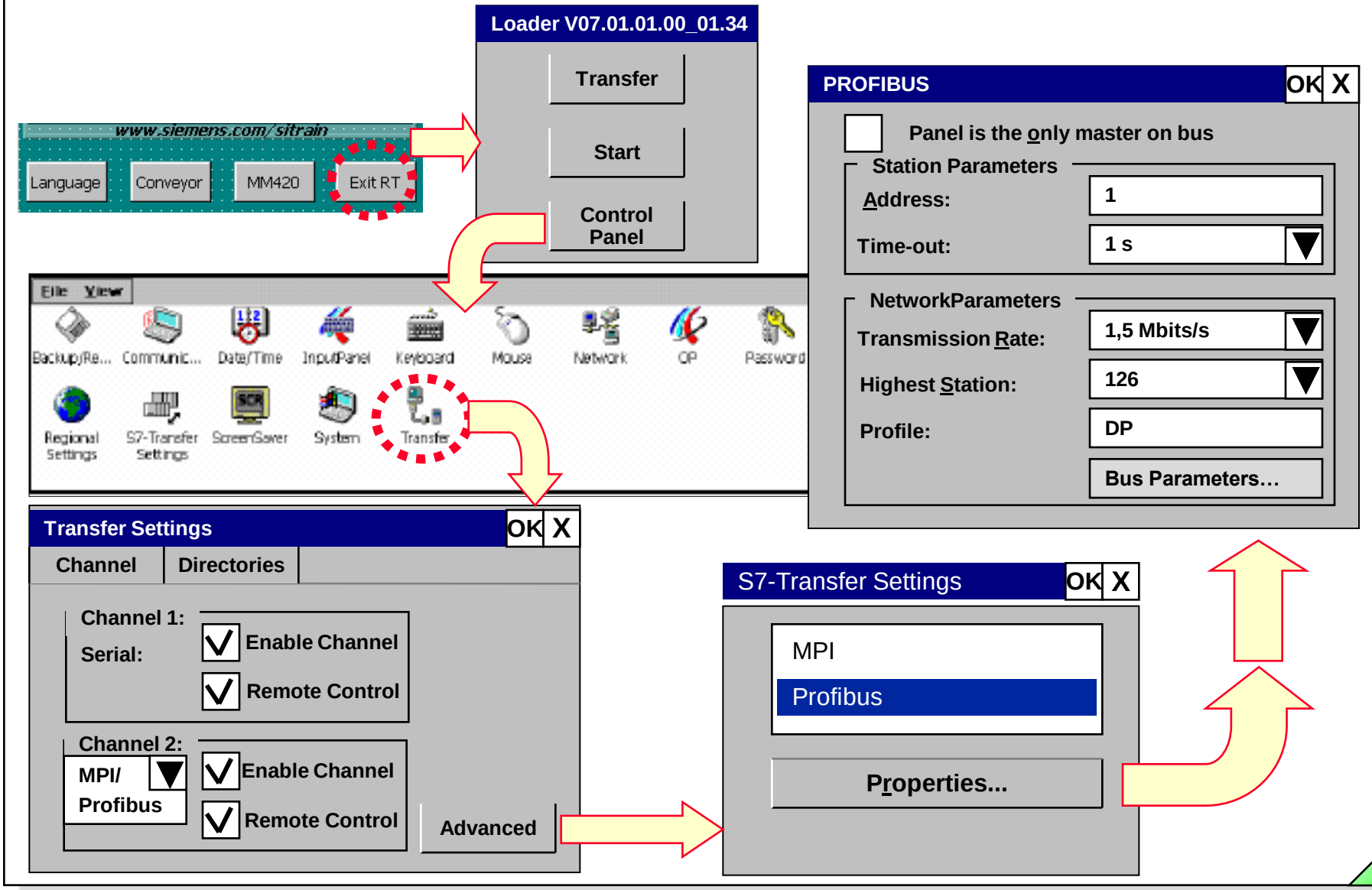
Transfer Settings in WinCC flexible



Exercise 2: Copying the Touchpanel Project



Exercise 3: Setting the HMI Operator Panel Interface for Transfer

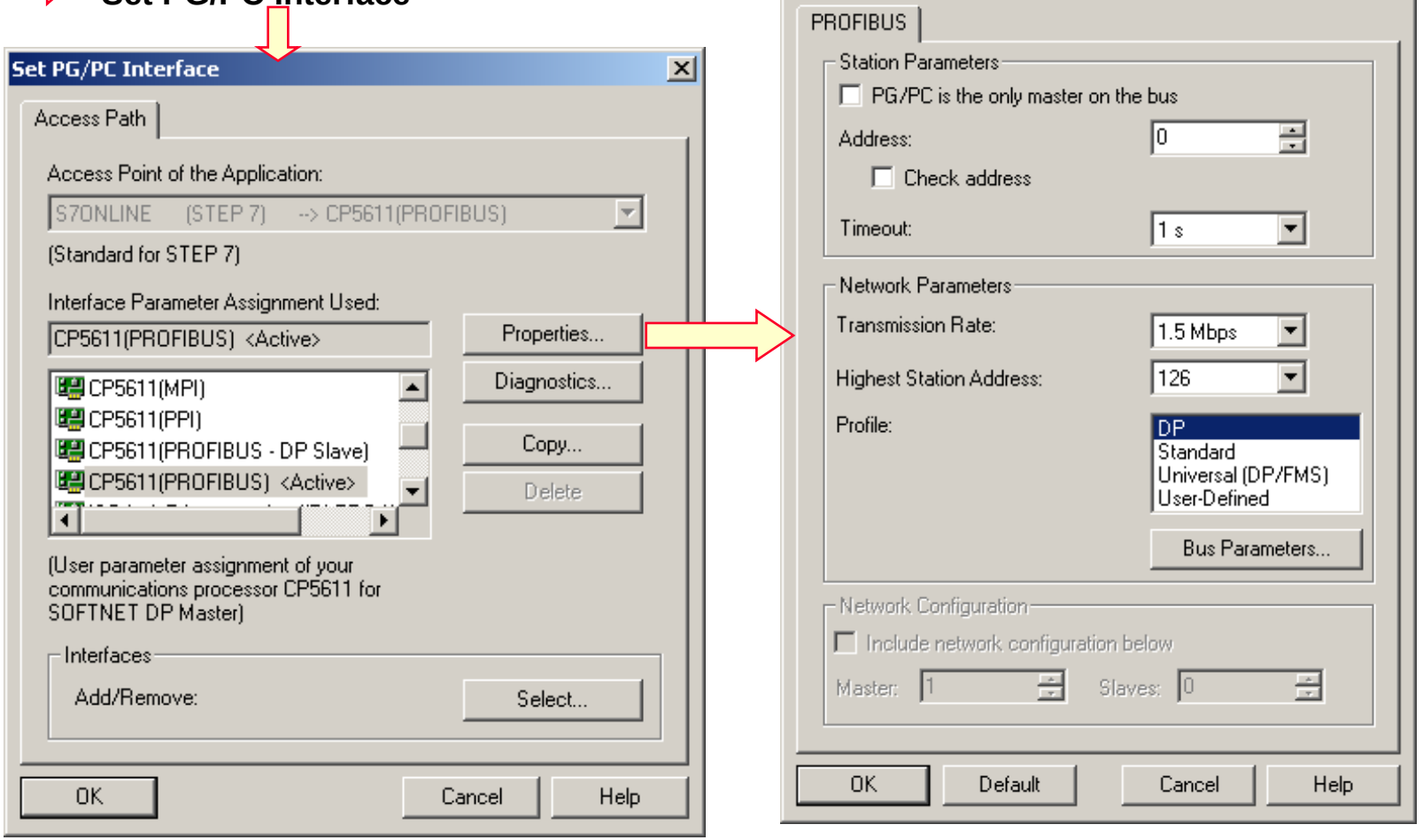


Exercise 4: Switching the PG Interface to PROFIBUS DP

SIMATIC Manager

Options

Set PG/PC Interface

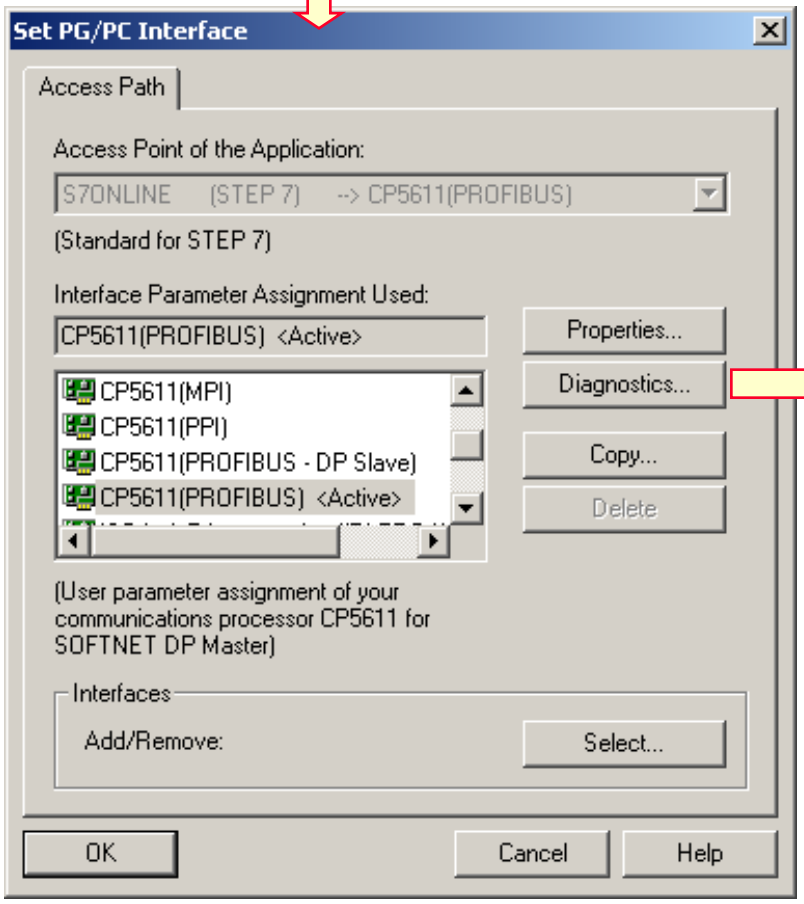


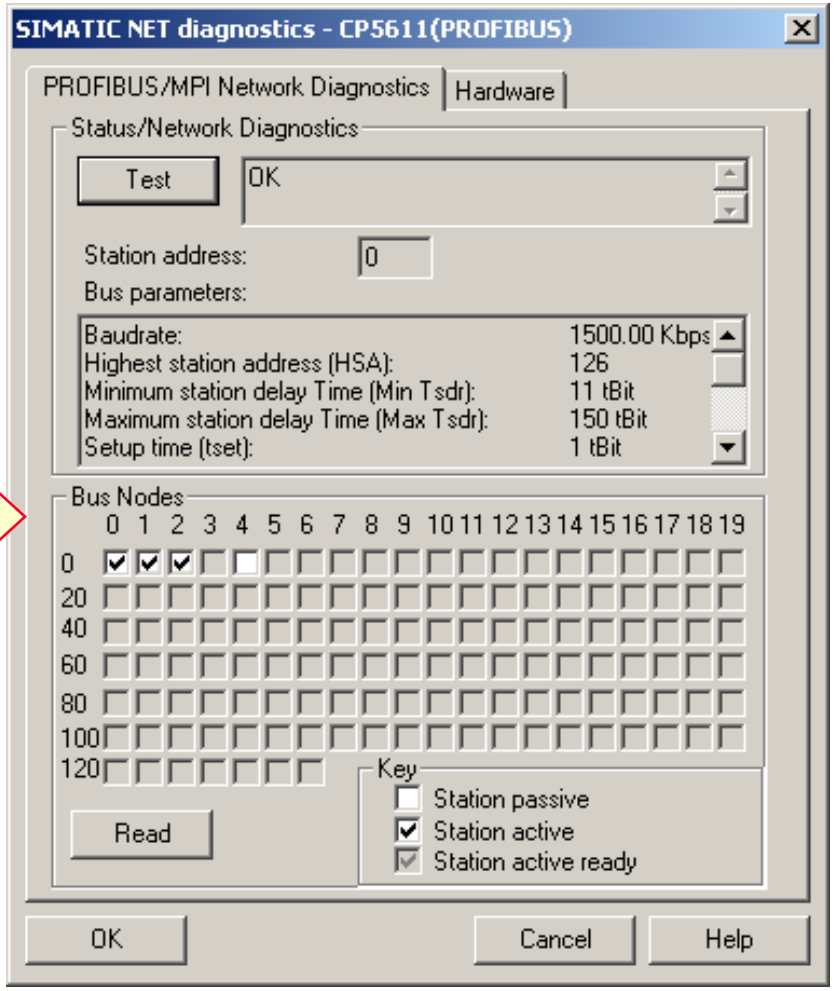
Exercise 5: Checking the PROFIBUS DP

SIMATIC Manager

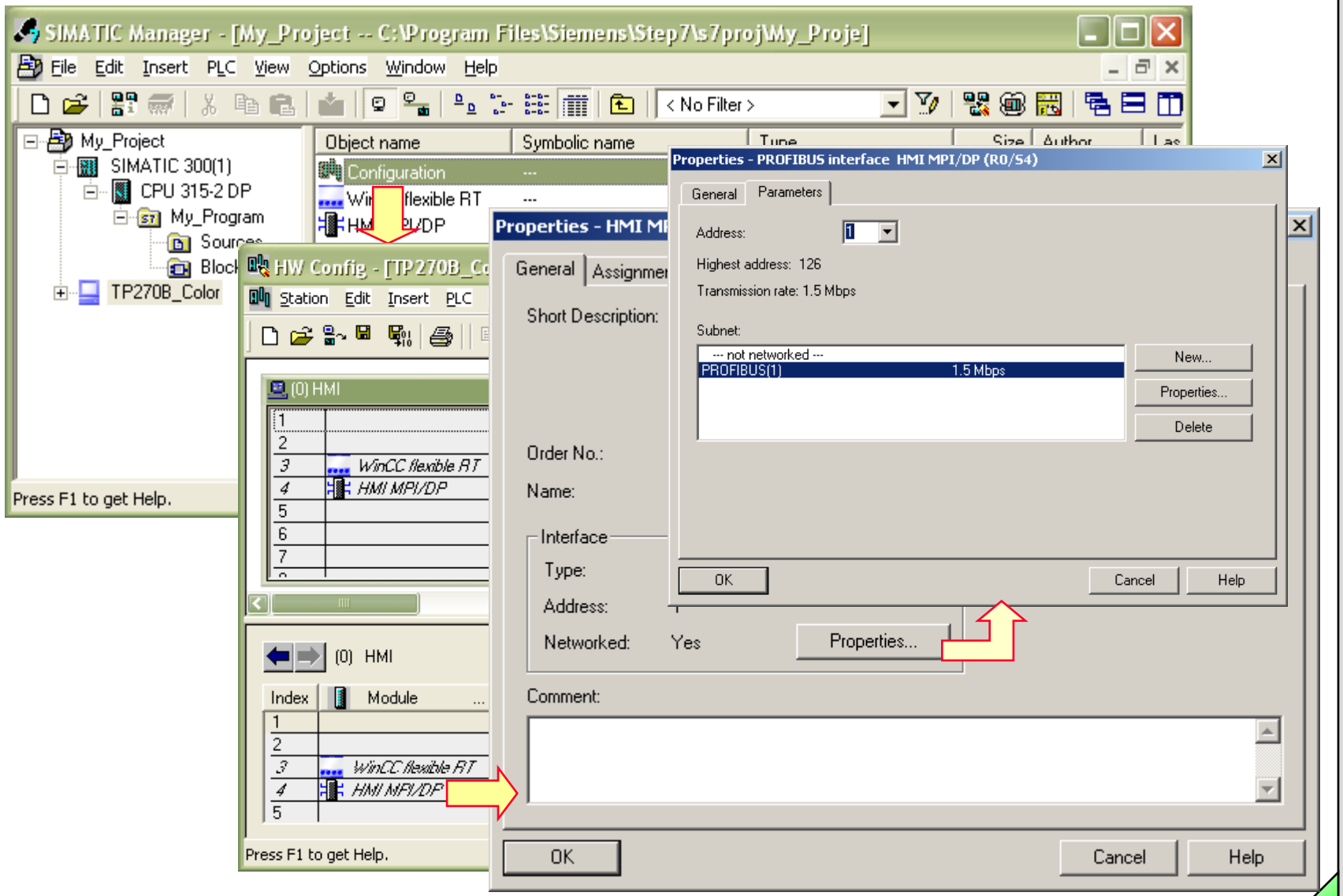
Options

Set PG/PC Interface

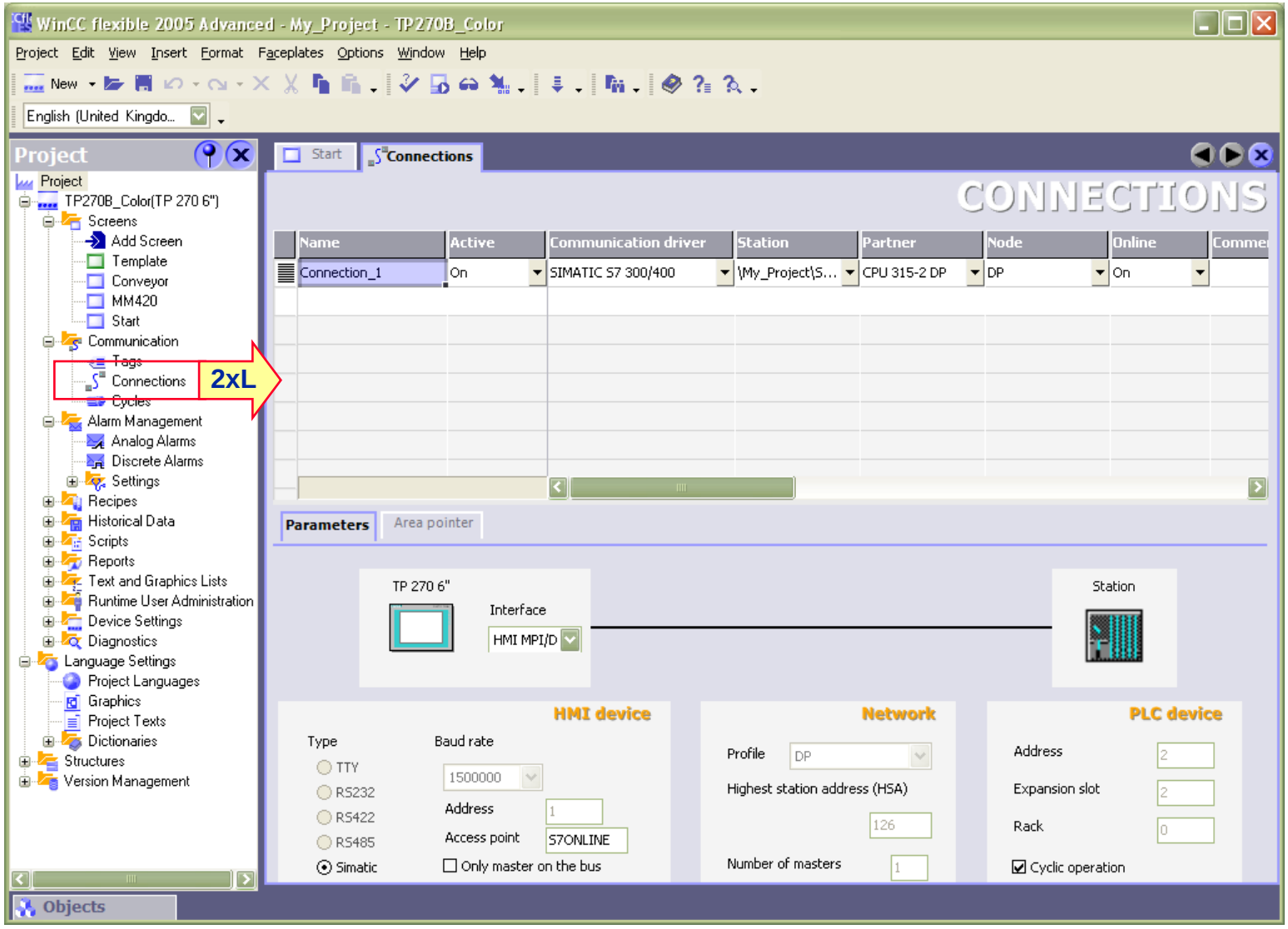




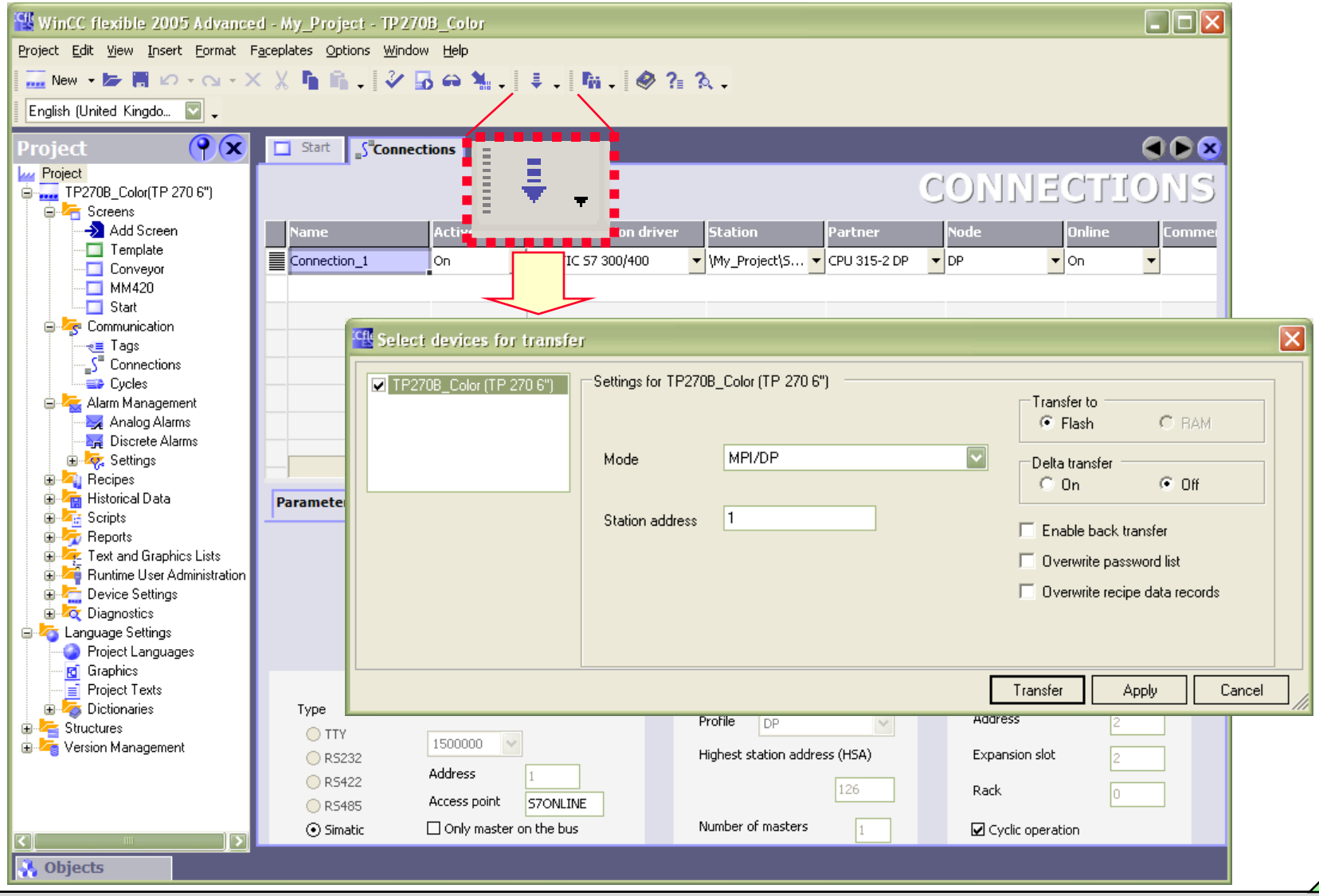
Exercise 6: Checking the RT Interface of the WinCC flexible Station



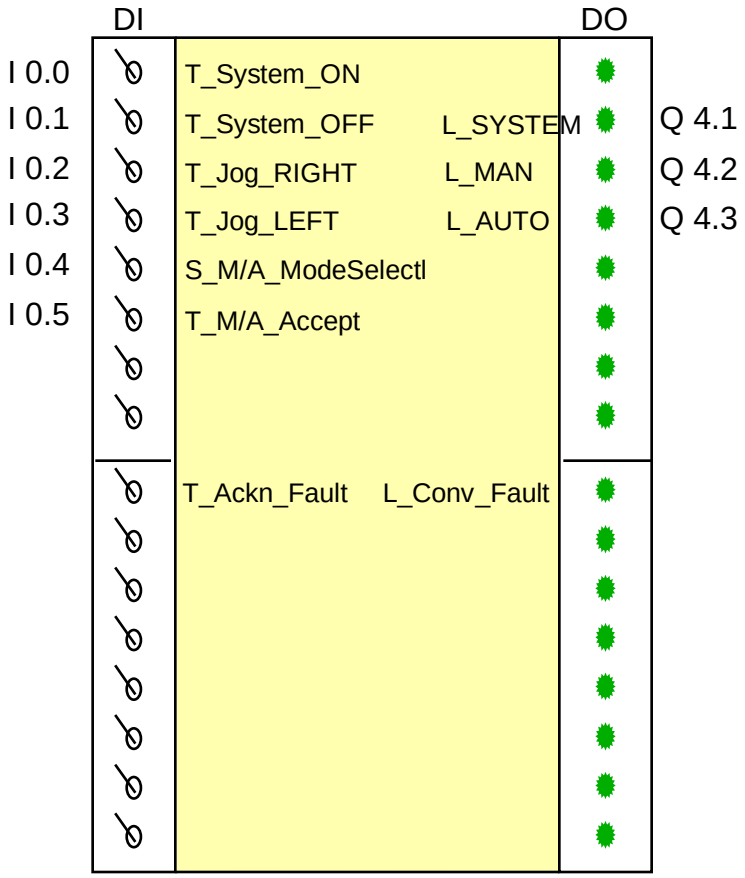
Exercise 7: Checking the Connection to the Controller



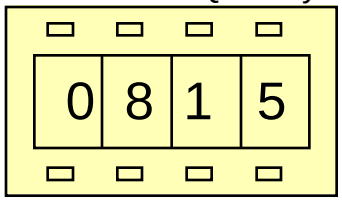
Exercise 8: Transferring the WinCC flexible Project into the HMI



Exercise 9: Testing the WinCC flexible Project

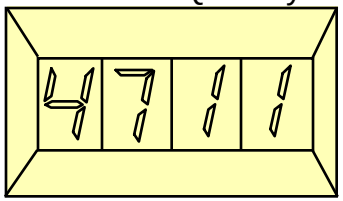


SETPOINT Quantity

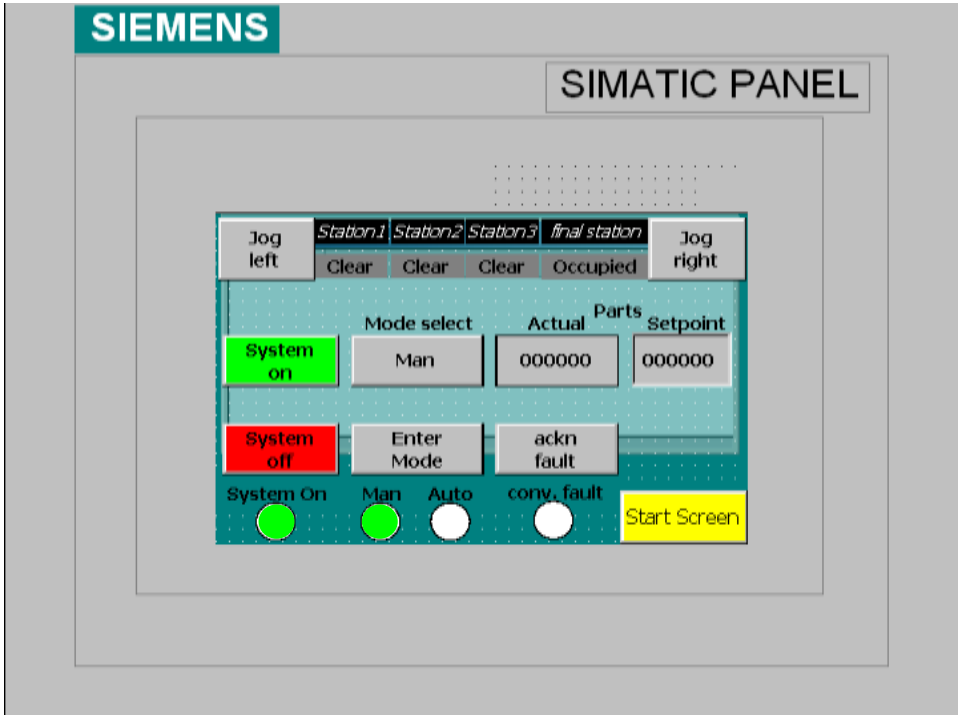


"IW_Thumbw" (IW 2)

ACTUAL Quantity

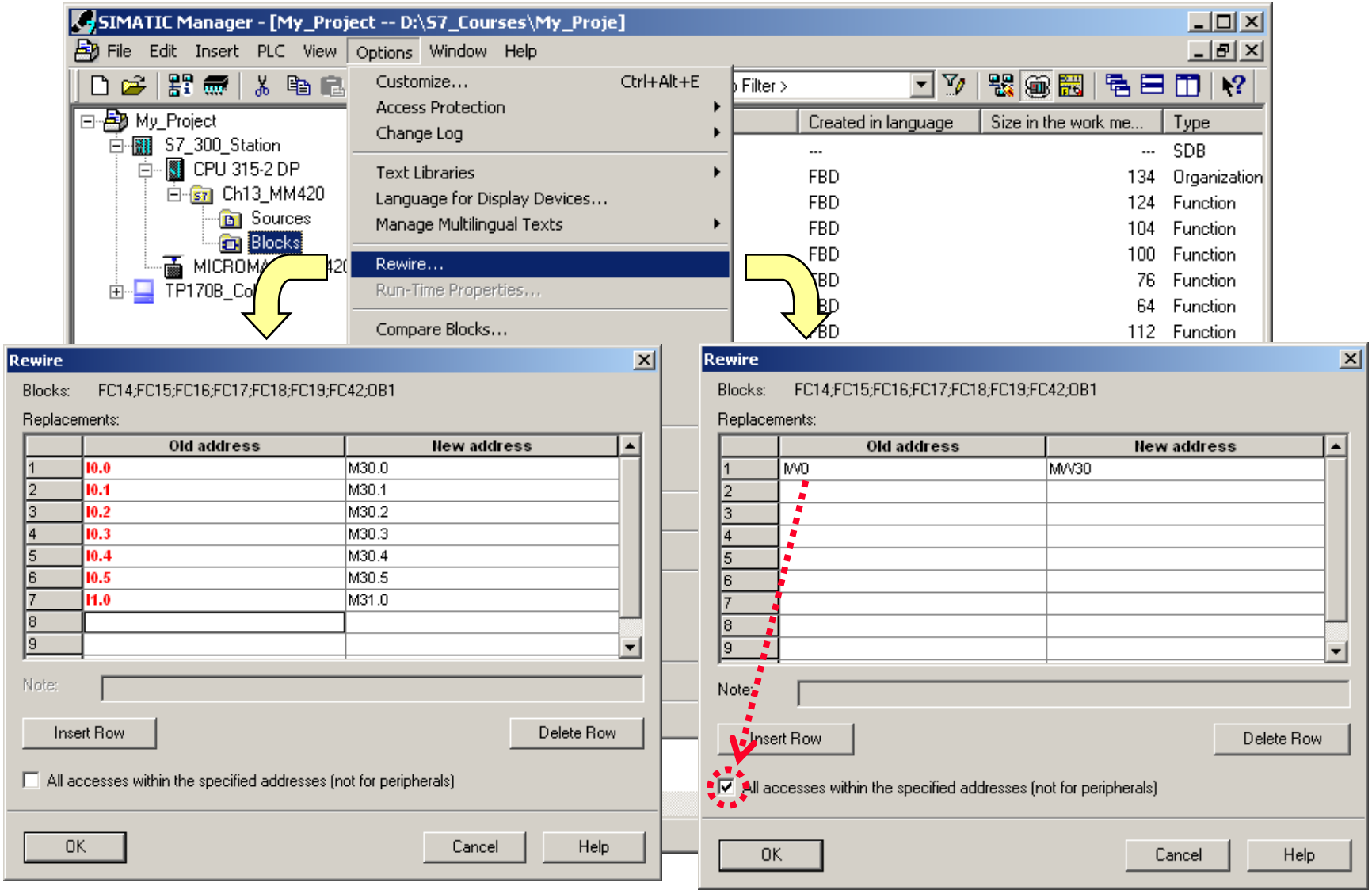


"QW_DigDisp" (QW 6)

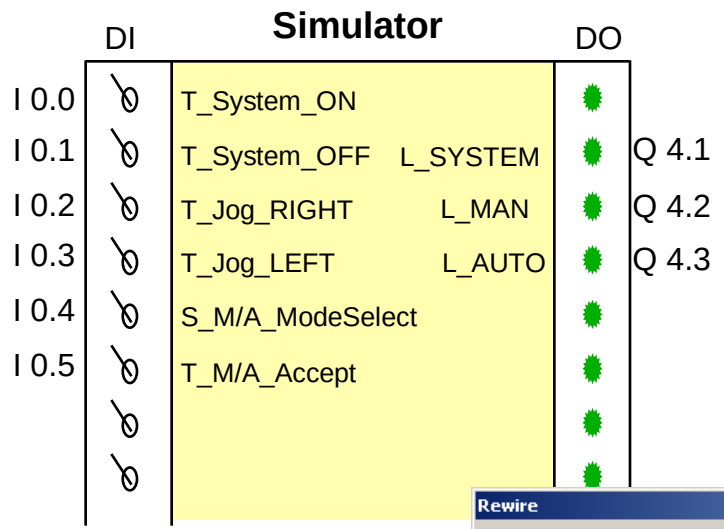


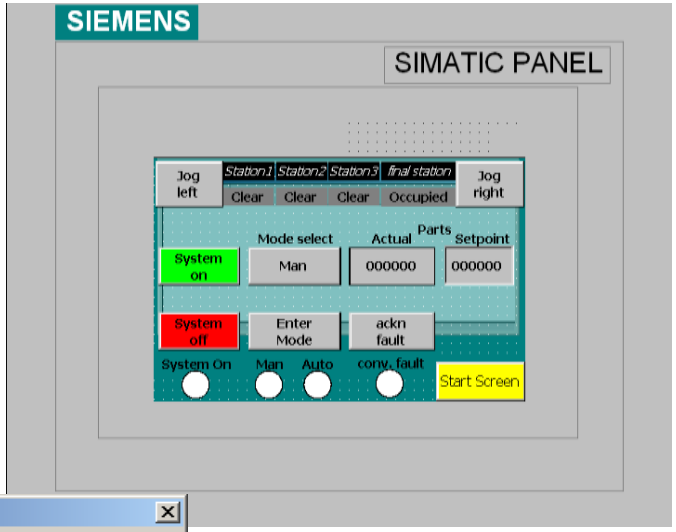
Date: 2/4/2012
File: SERV1_11E.24

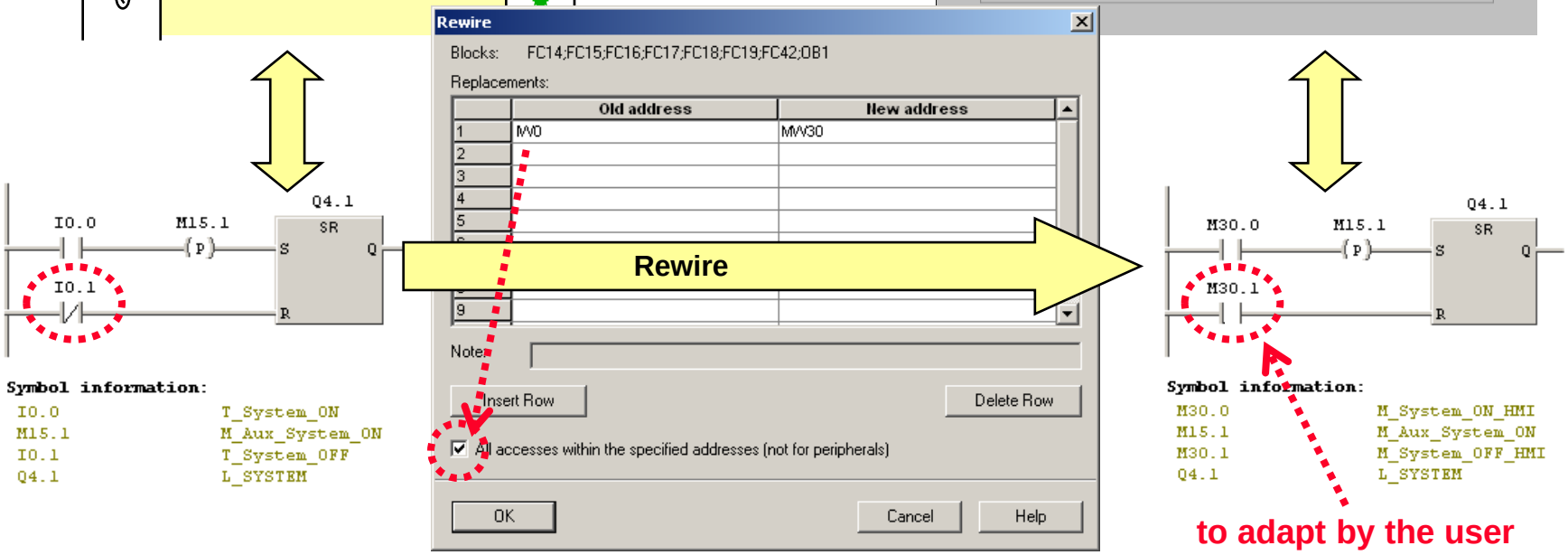
Rewiring using Absolute Addresses



Exercise 10: Adapting the S7 Program using "Rewiring"

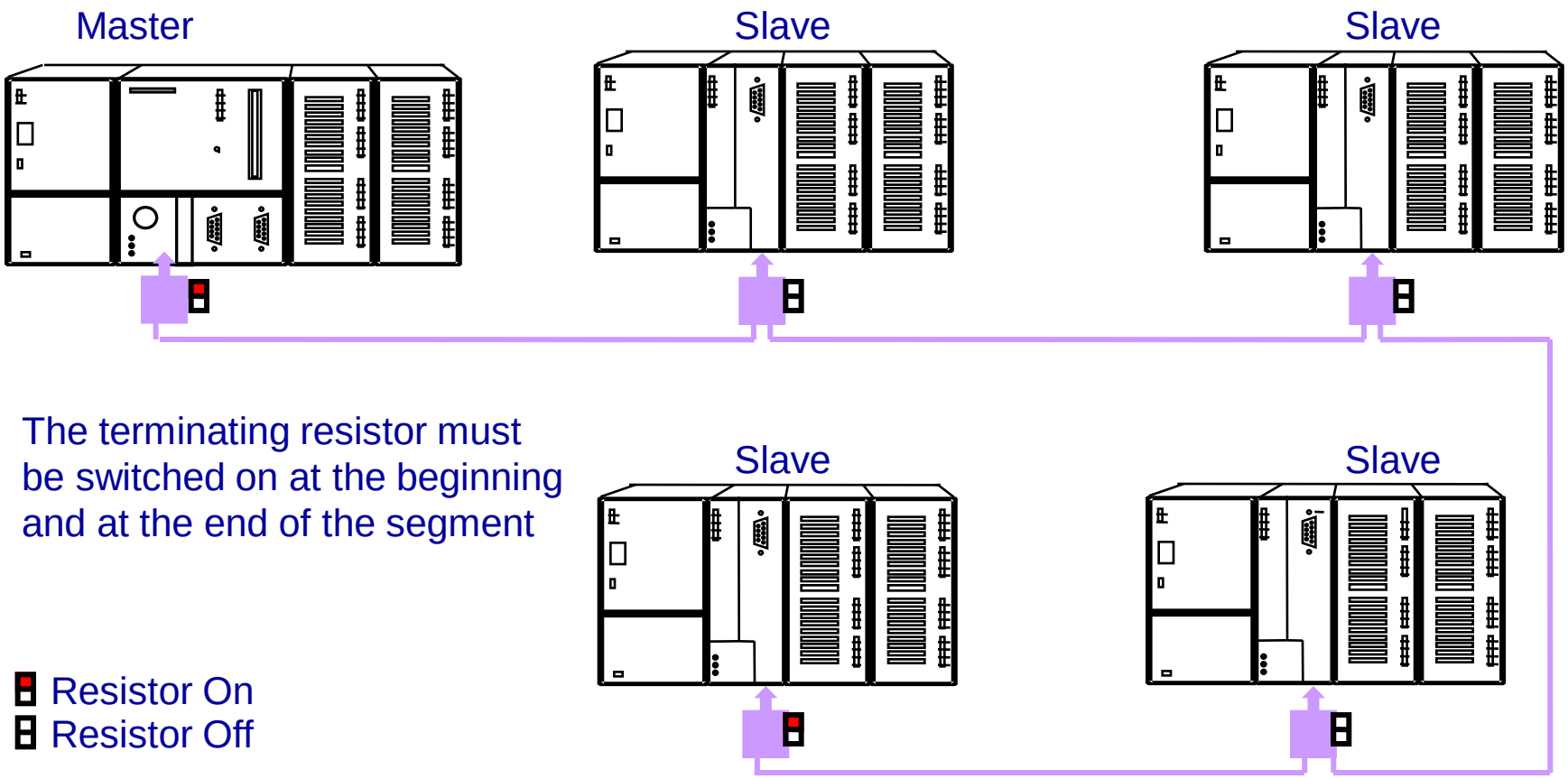






If You Want to Know More

PROFIBUS – DP Terminating Resistor

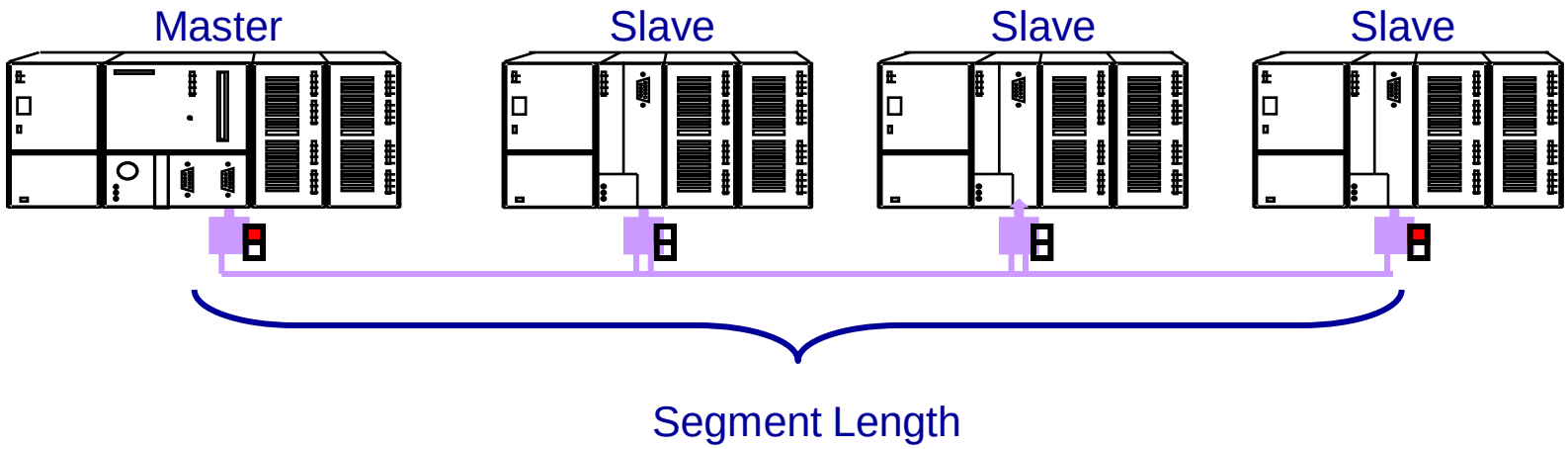


The terminating resistor must be switched on at the beginning and at the end of the segment

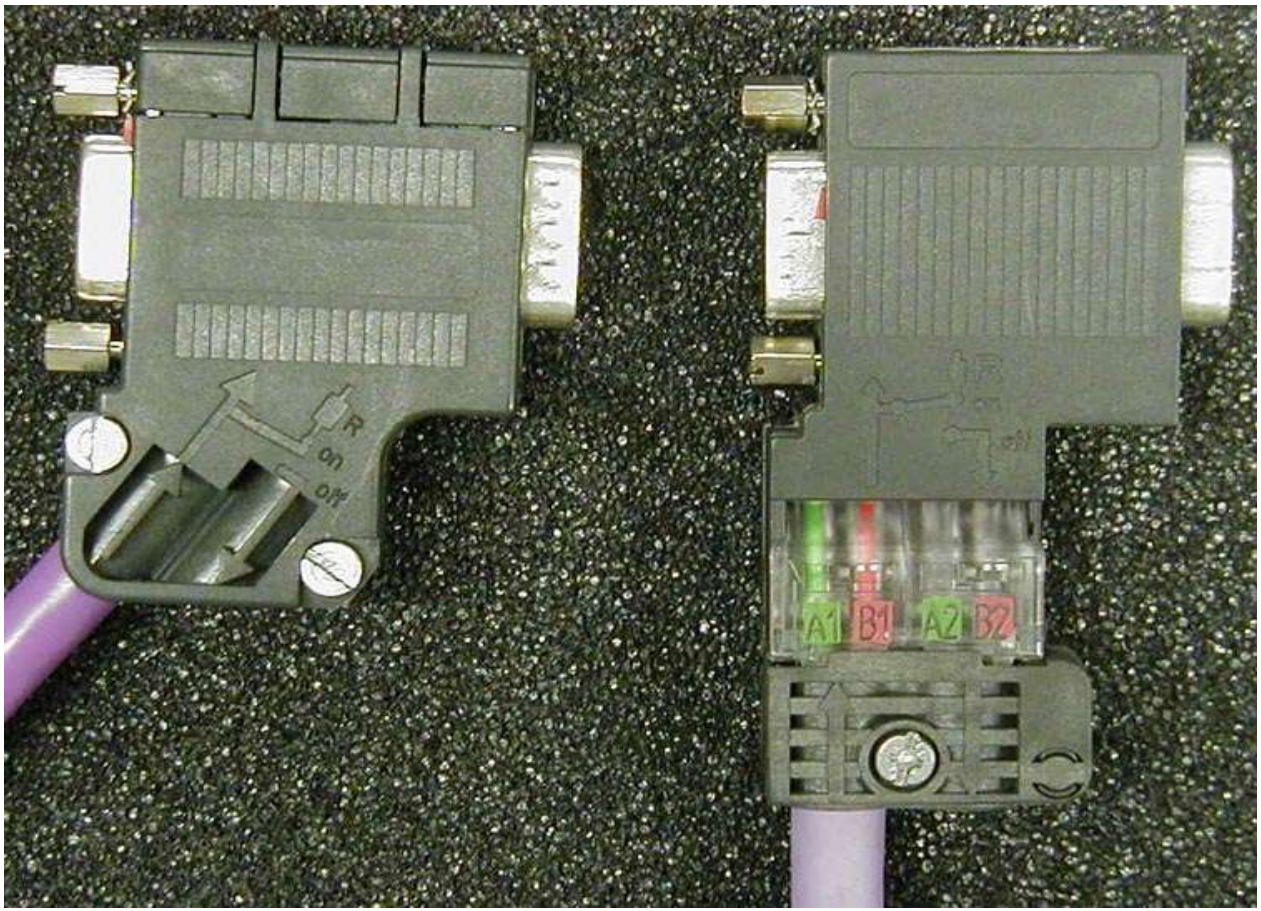
Resistor On
Resistor Off

Segment Length Depending on the Baud Rate

Baud Rate		Segment Length
9.6 to 187.5	kBaud	1,000 meters
500	kBaud	400 meters
1.5	kBaud	200 meters
3 to 12	MBaud	100 meters



PROFIBUS Connector



Standard connector
with screw terminal

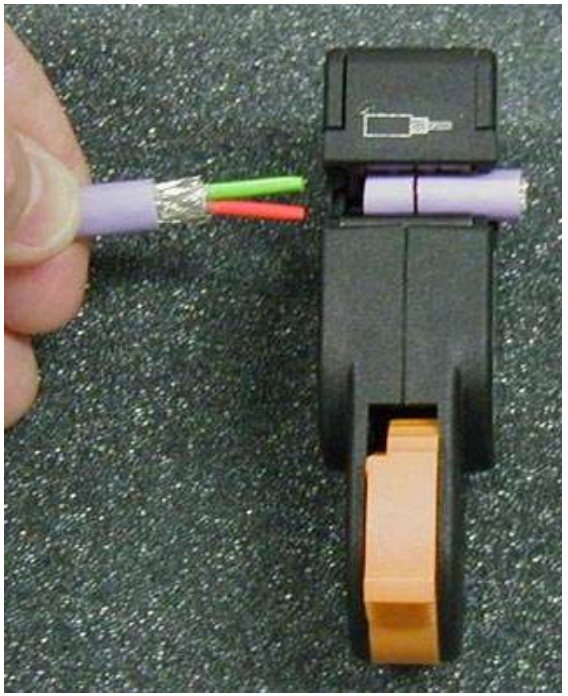
Plug for
Fast Connect

Preparing a Fast Connect Cable

1. Measure

2. Strip

3. Remove



Data Blocks

Objectives

Upon completion of this chapter the participant will ...

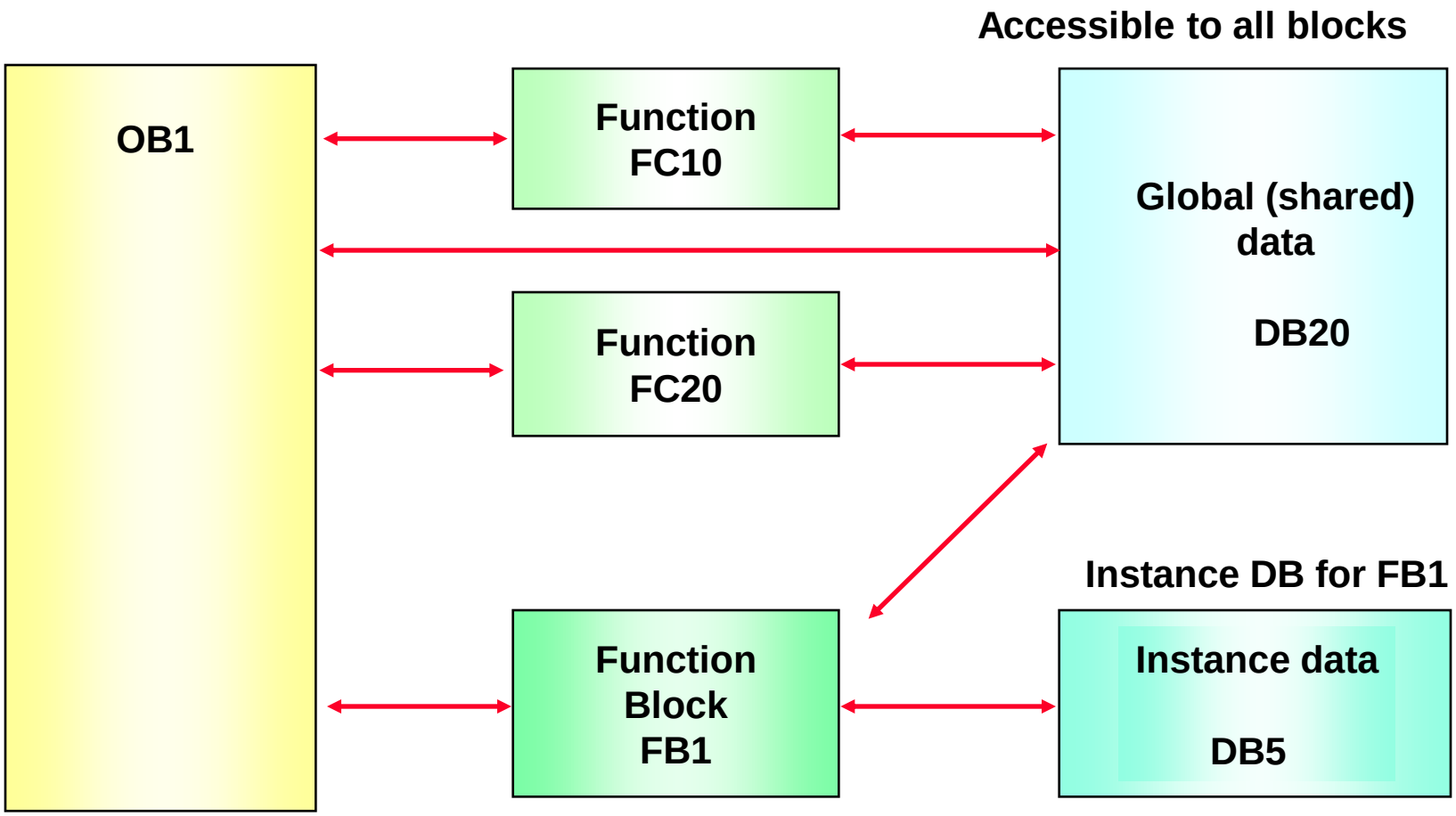
... understand the purpose of global data blocks

... be familiar with elementary and complex data types

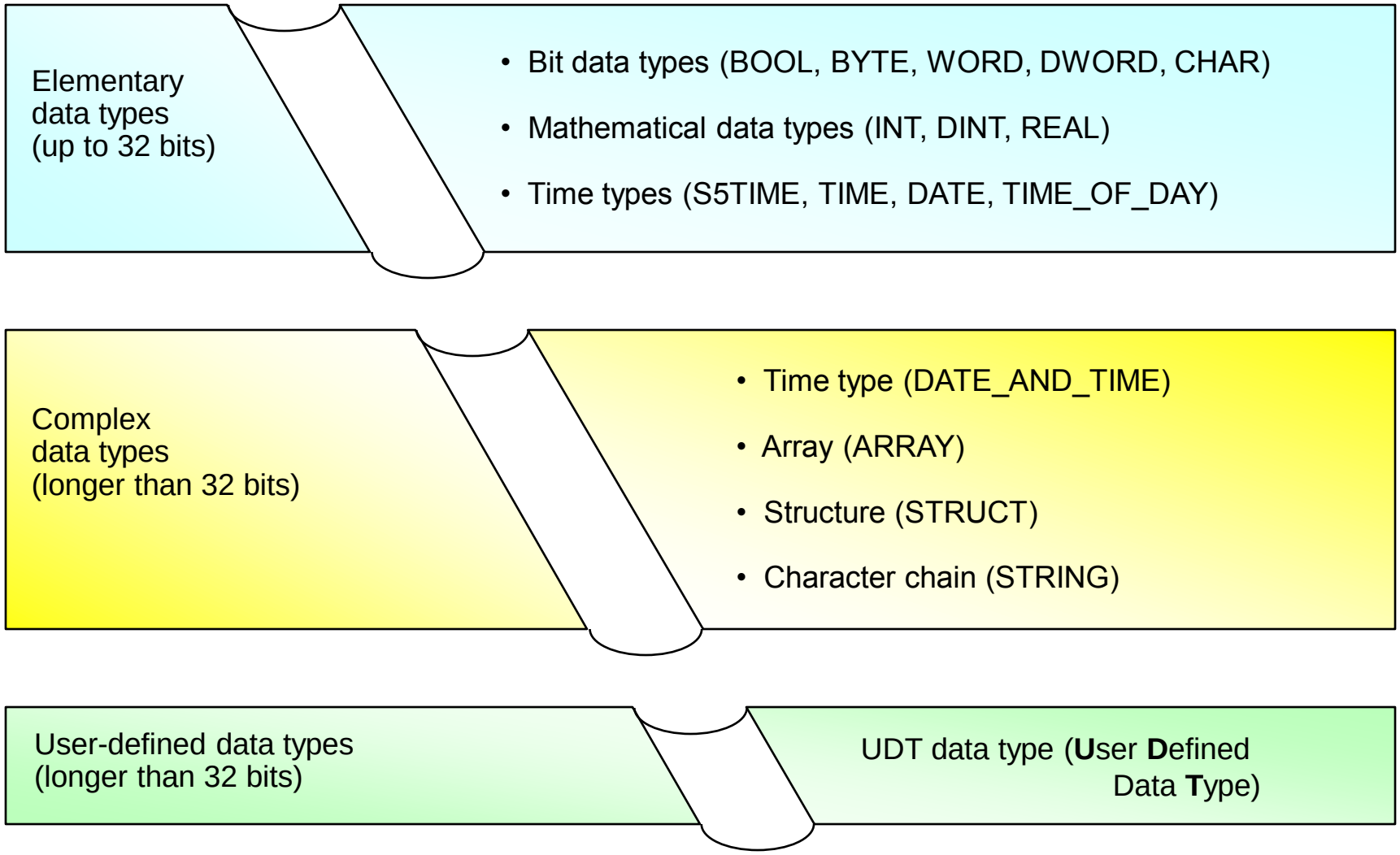
... be able to monitor a data block

... be familiar with the possibilities for addressing data block variables

Data Blocks (DBs)



Overview of Data Types in STEP 7



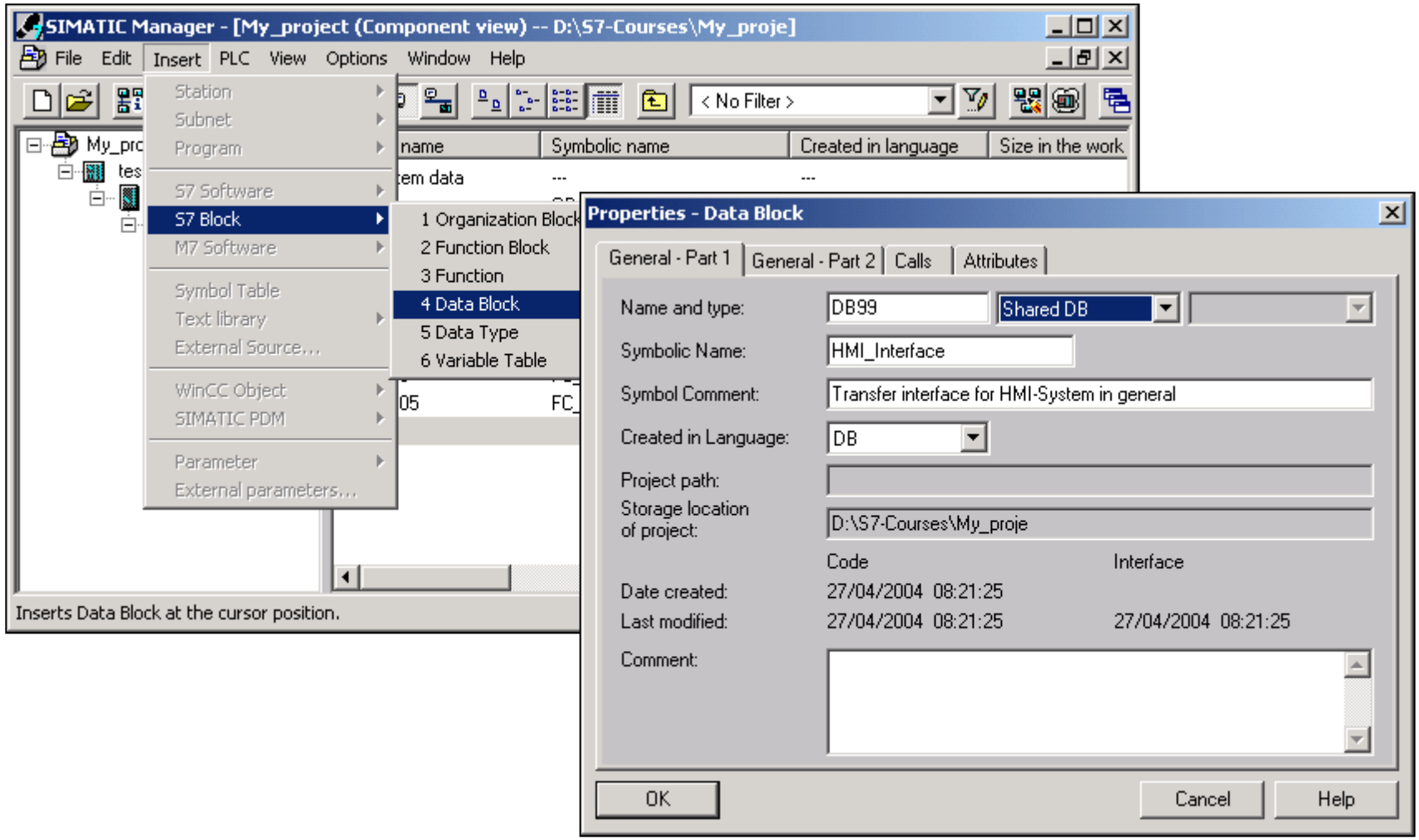
Elementary Data Types in STEP 7

Keyword	Length (in bits)	Constants	Variables
BOOL	1	1 or 0	I 1.0
BYTE	8	B#16#A9	MB70
WORD	16	W#16#12AF	MW72
DWORD	32	DW#16#ADAC1EF5	QD40
CHAR	8	' w '	DBB4
S5TIME	16	S5T#5s_200ms	MW30
INT	16	123	#Value
DINT	32	L#65539	MD80
REAL	32	1.2 or 34.5E-12	DBD60

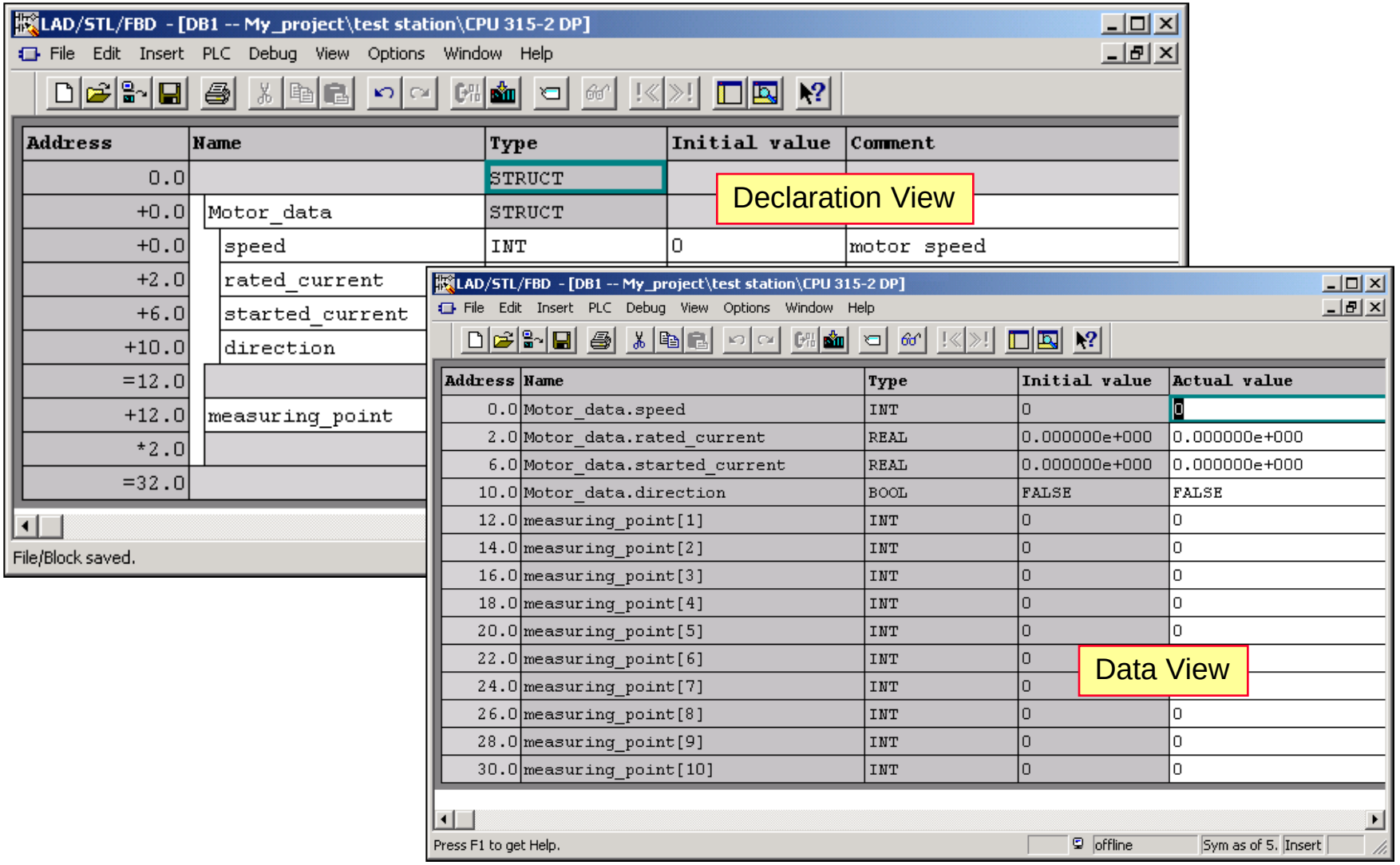
Complex Data Types

Keyword	Length (in bits)	Example
DATE_AND_TIME	64	DT#01-08-24-12:14:55:234-1
STRING (character string with max. 254 characters)	8 * (number of characters +2)	´This is a string´ ´SIEMENS´
ARRAY (Group of elements of the same data type)	user-defined	Measured values: ARRAY[1..20] INT
STRUCT (Group of elements of different data types)	user-defined	Motor: STRUCT Speed : INT Current : REAL END_STRUCT
UDT (**U**ser **D**efined Data **T**ype = "Template" consisting of elementary or complex data types	user-defined	UDT as block: STRUCT Speed : INT Current : REAL UDT1 END_STRUCT UDT as array element: Drive: ARRAY[1..4]

Creating a Data Block



Entering, Saving, Downloading and Monitoring a Data Block



Initial Value, Actual Value, Initialization, Retentivity

Load Memory

DB
Initial values
+ Actual values
(with data types)

Work Memory

DB
Actual values

CPU with STEP 7 Program

DB

Initial values

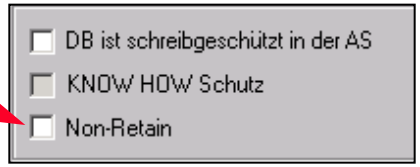
+ Actual values

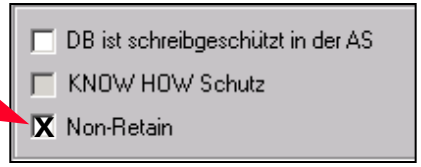
Upload in PG

Initialize data block

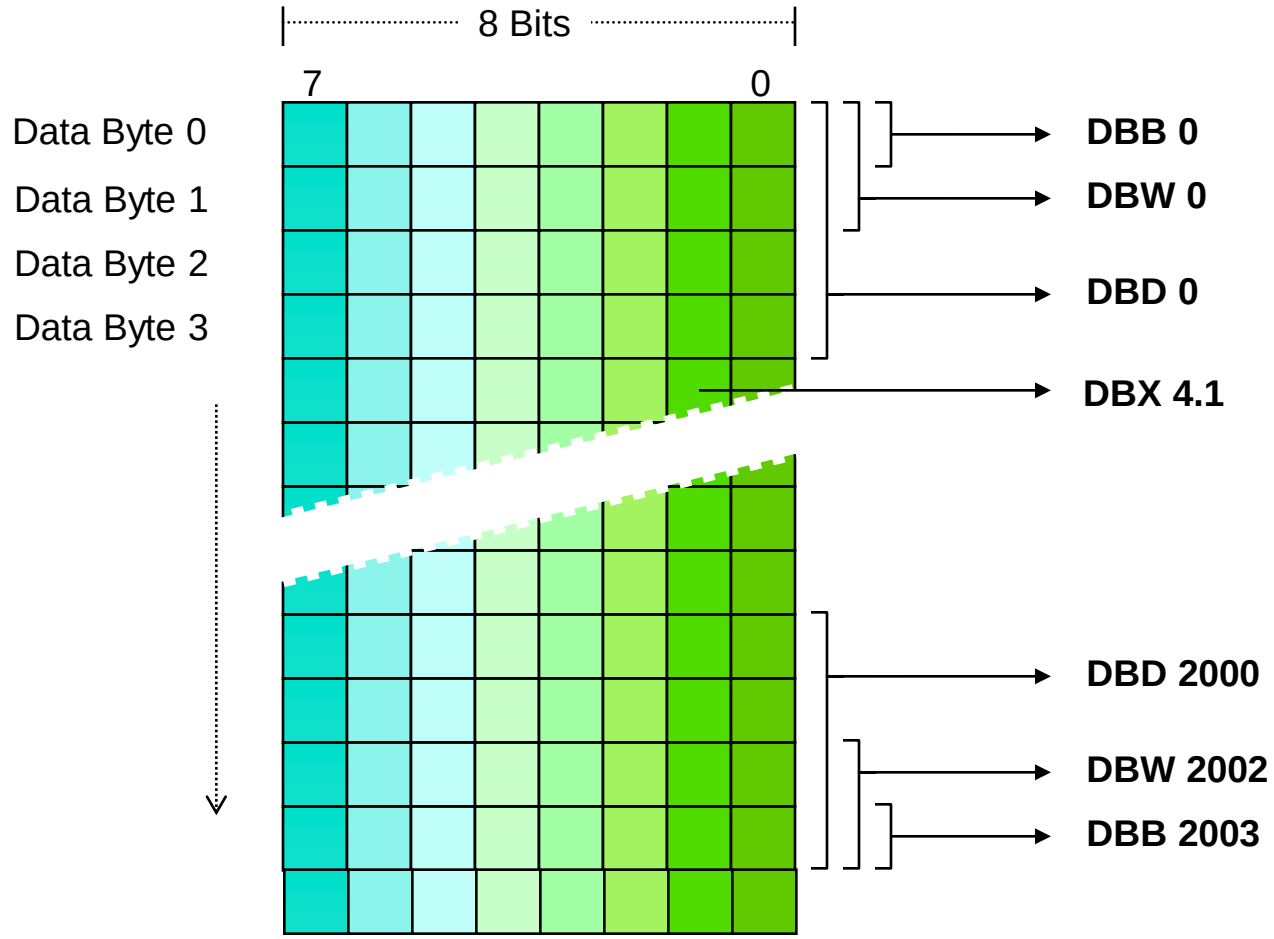
DB is retentive until the next transmission PG → CPU

With every CPU startup (i.e. with every STOP-RUN transition) the DB data are reset to the initial loading state





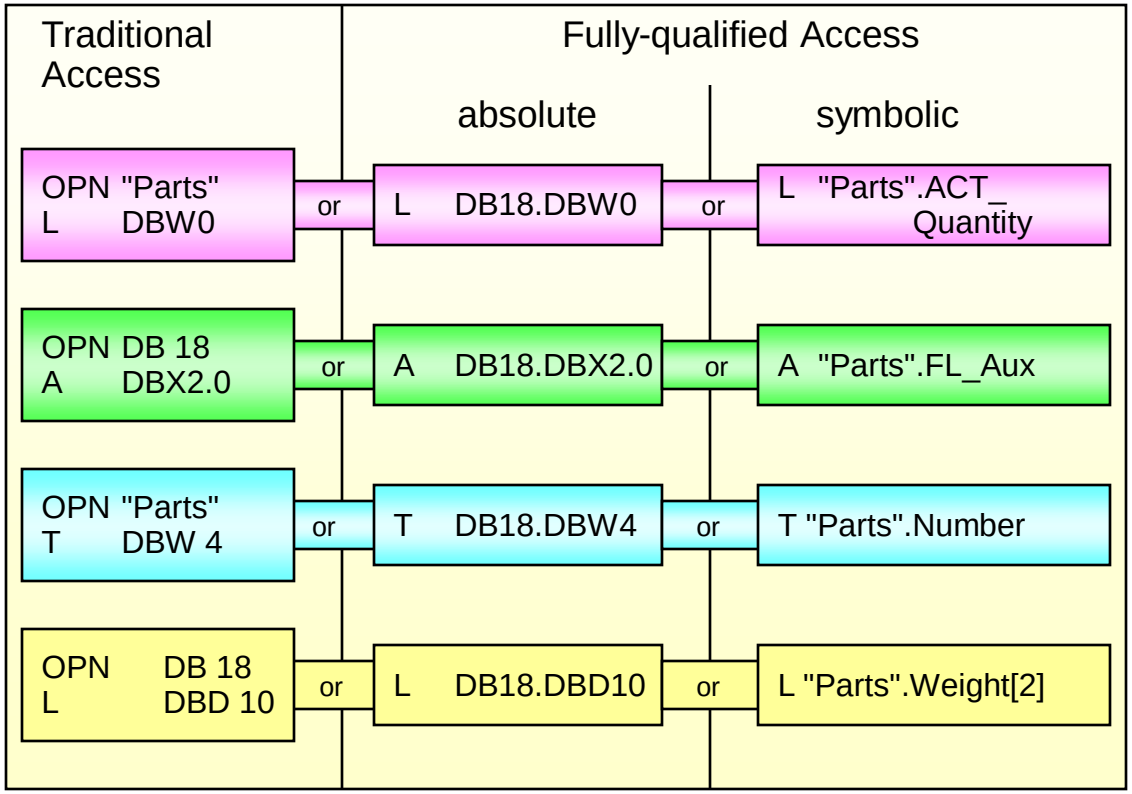
Addressing Data Elements



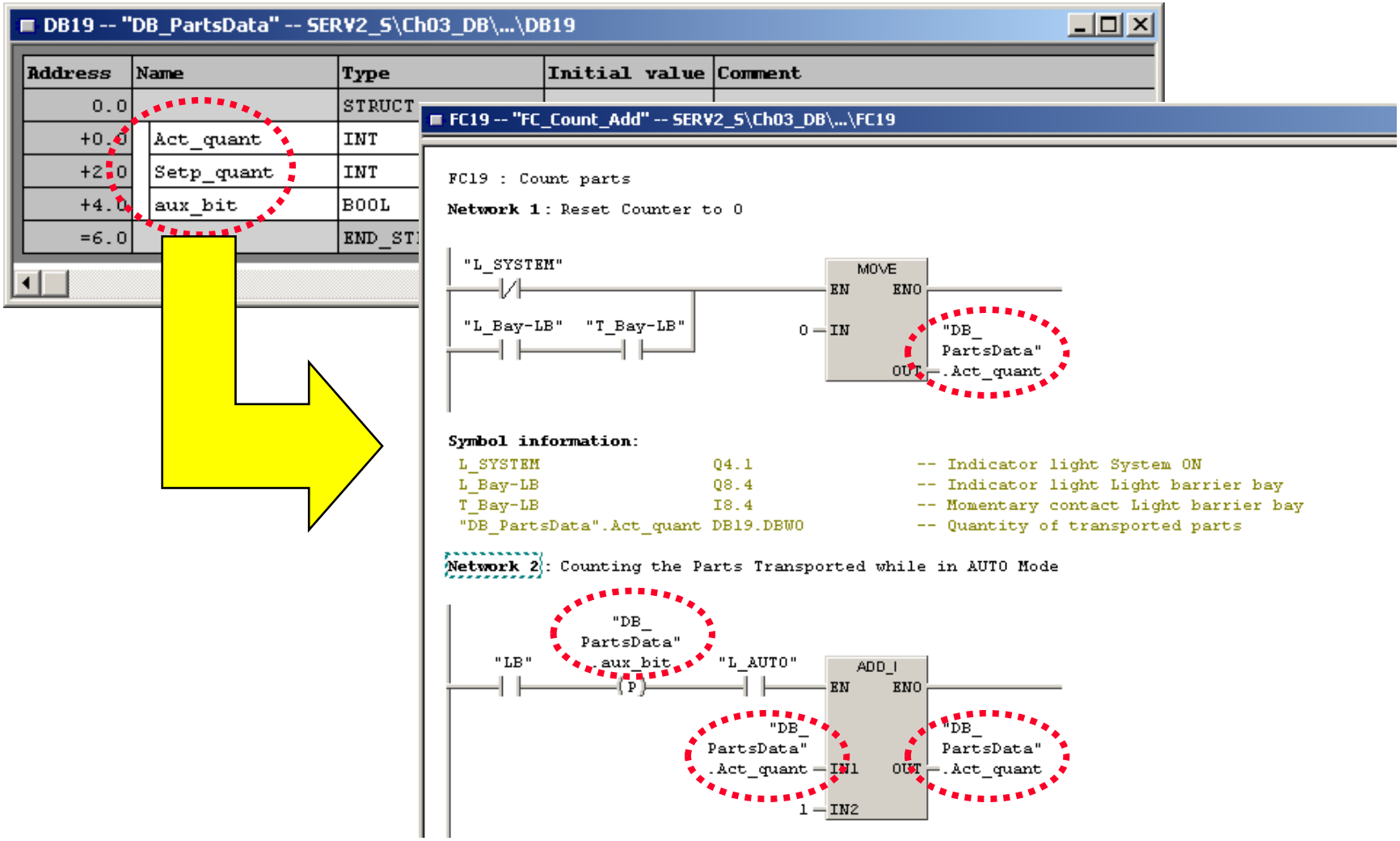
Accessing Data Elements

DB 18 "Parts"

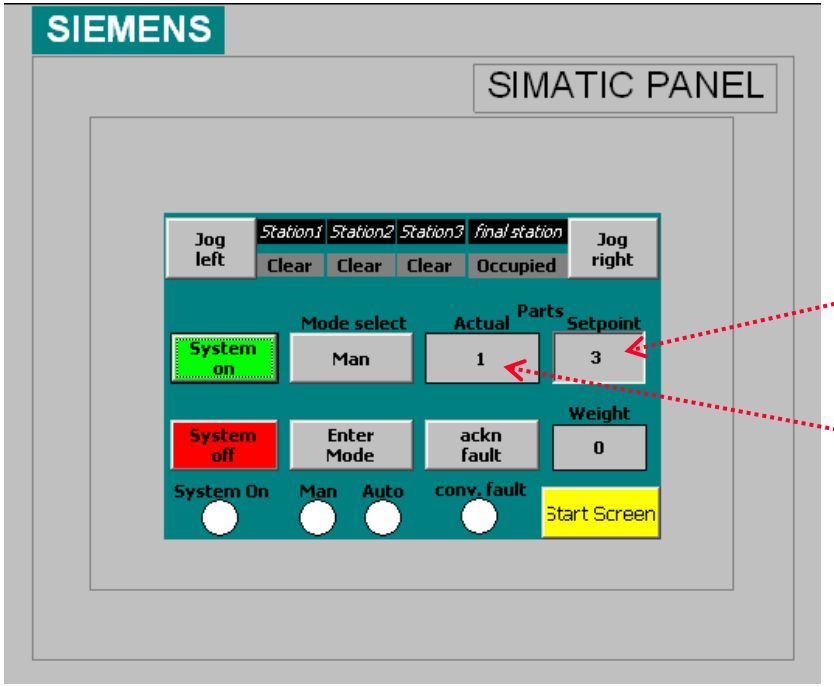
Addr.	Name	Type
0.0	Act_Quantity	INT
2.0	FL_Aux	BOOL
4.0	Number	INT
6.0	Weight[1]	REAL
10.0	Weight[2]	REAL

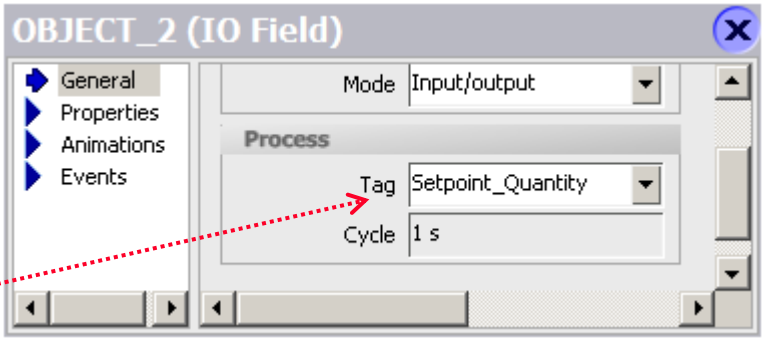


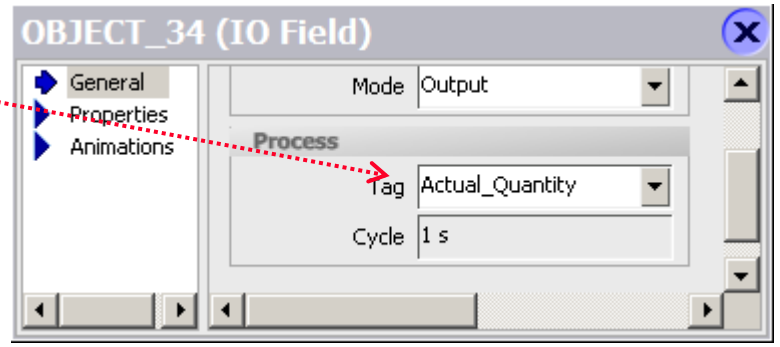
Exercise 1: Replacing Bit Memory Variables with DB Variables

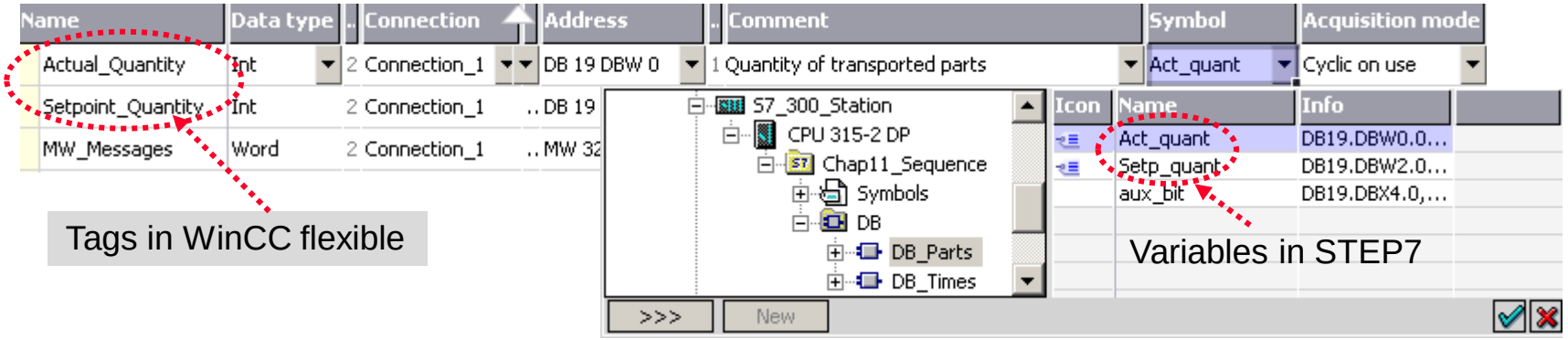


Exercise 2: Connecting WinCC flexible Tags to DB Variables









If You Want to Know More

Example of an ARRAY

Measuring_point

1. Measuring_point, data type Real
2. Measuring_point, data type Real
3. Measuring_point, data type Real
•
•
•
10. Measuring_point, data type Real

Array with the name "Measuring_point"
(several elements of the same data type)

Display in the Program Editor (Data Block DB 2):

DB2 -- My_Project\My_Station\CPU 314

Address	Name	Type	Initial value	Comment
*0.0		STRUCT		
+0.0	Measuring_point	ARRAY[1..10]		
*4.0		REAL		
=40.0		END_STRUCT		

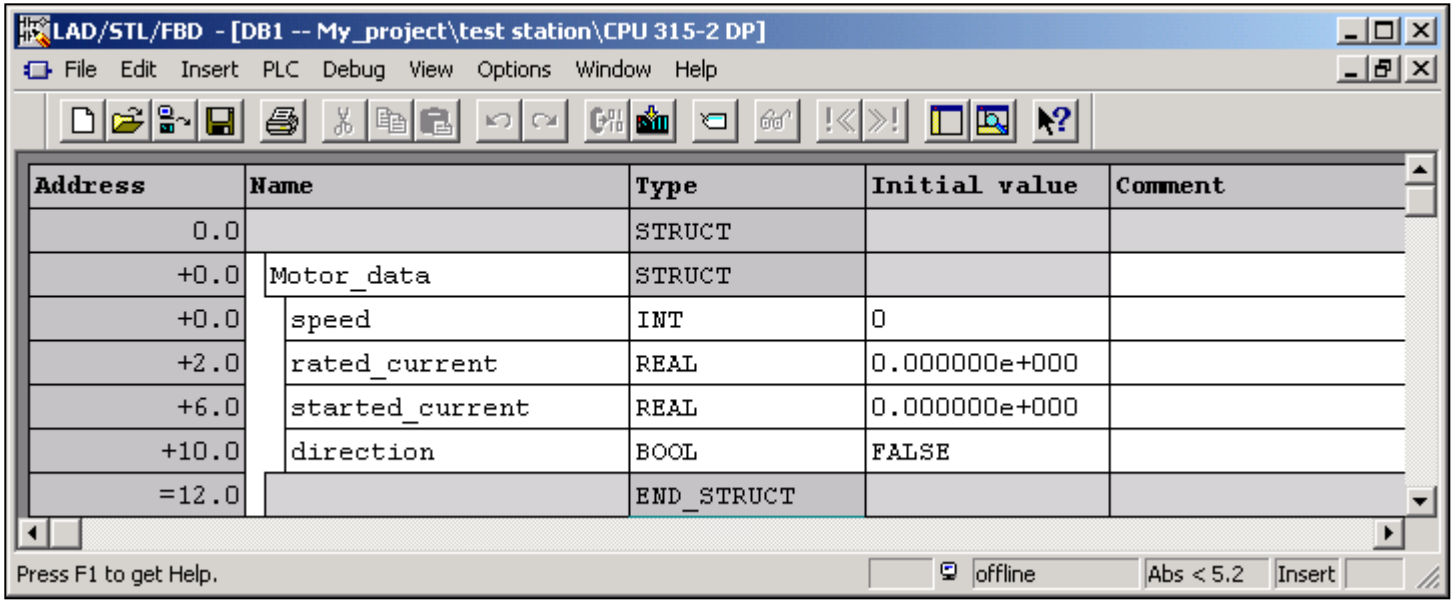
Example of a STRUCTURE

Motor_data

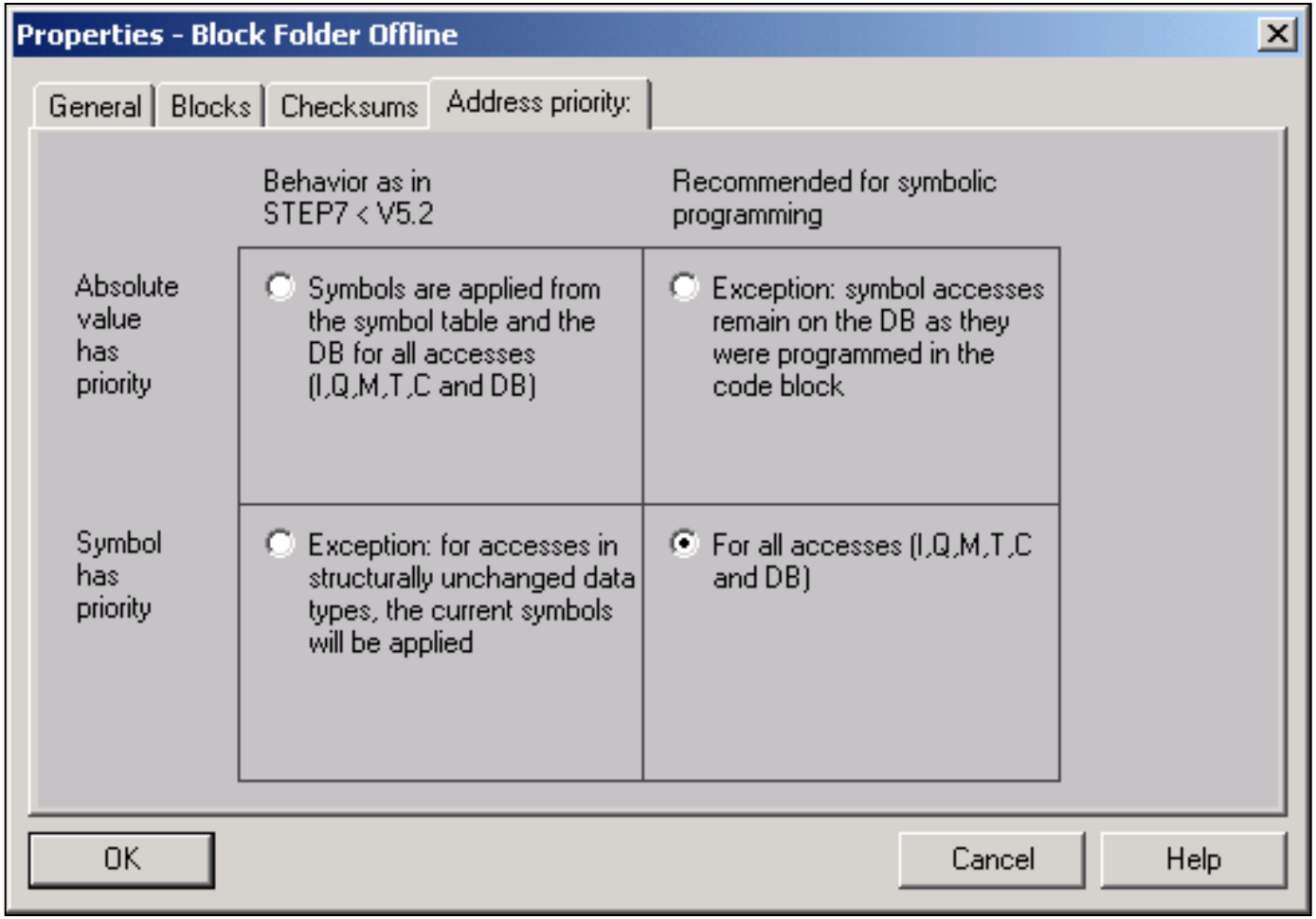
Speed, data type Integer
Rated_current, data type Real
Starting_current, data type Real
Direction, data type Bool

Structure with the name "Motor_data"
(several elements
with different data types)

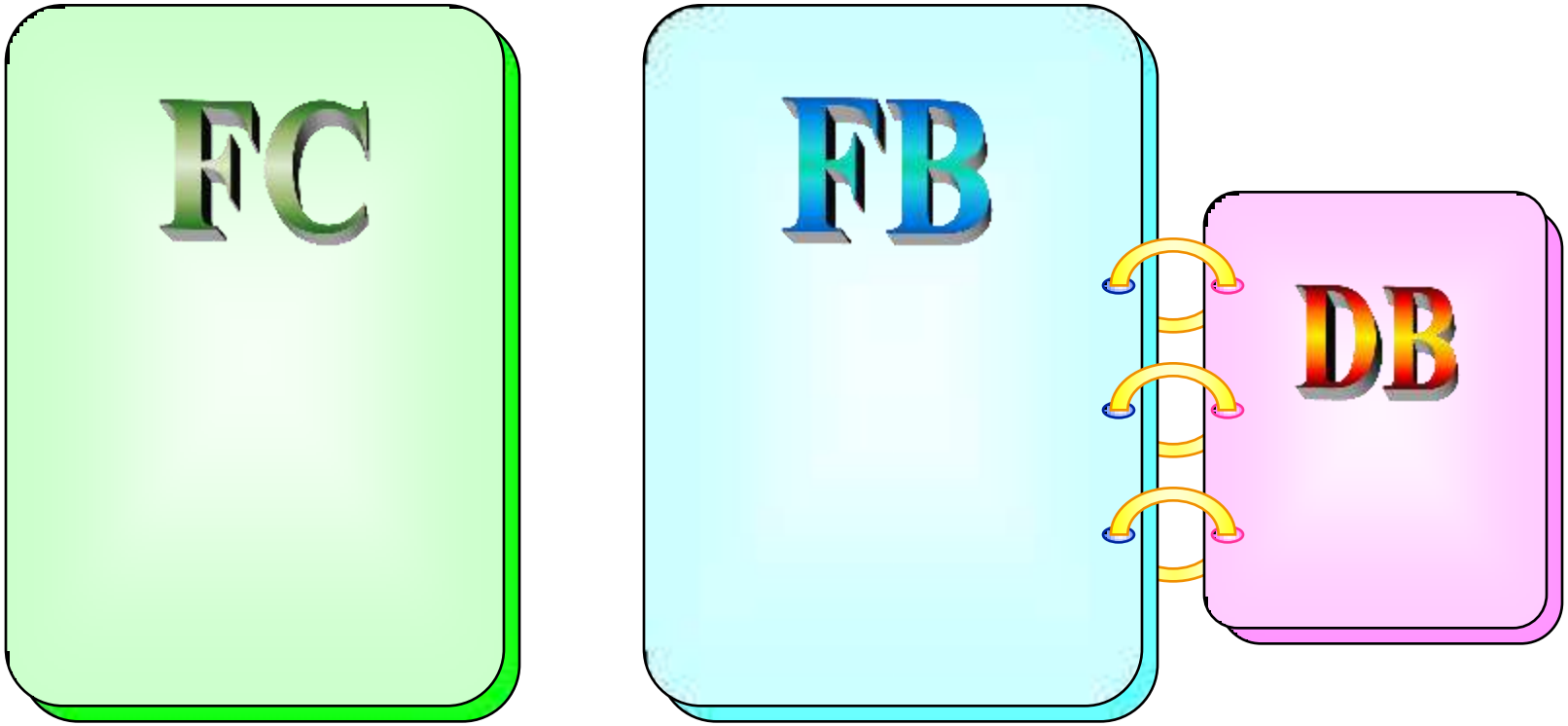
Display in the Program Editor (Data block DB 1):



Defining the Address Priority (Symbolic/Absolute)



Functions and Function Blocks



Objectives

Upon completion of this chapter the participant will ...

…	be familiar with the purpose of temporary variables
...	be able to program parameter-assignable functions and their calls
...	know the difference between functions (FCs) and function blocks (FBs)
...	be familiar with the purpose of static variables
...	be able to declare static variables and apply them in the program
...	be able to program parameter-assignable function blocks and call them

Variables Overview

Global Variables / Data
(valid in the entire program)

- PII / PIQ
- I / O Peripherals
- M / T / C
- DB areas

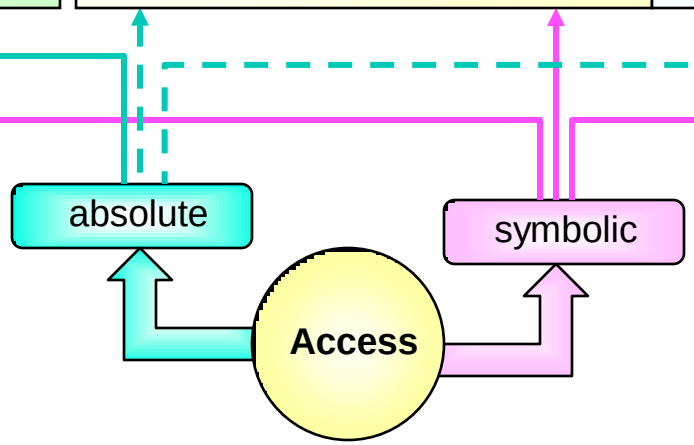
Local Variables / Data
(only valid in one block)

Temporary Variables

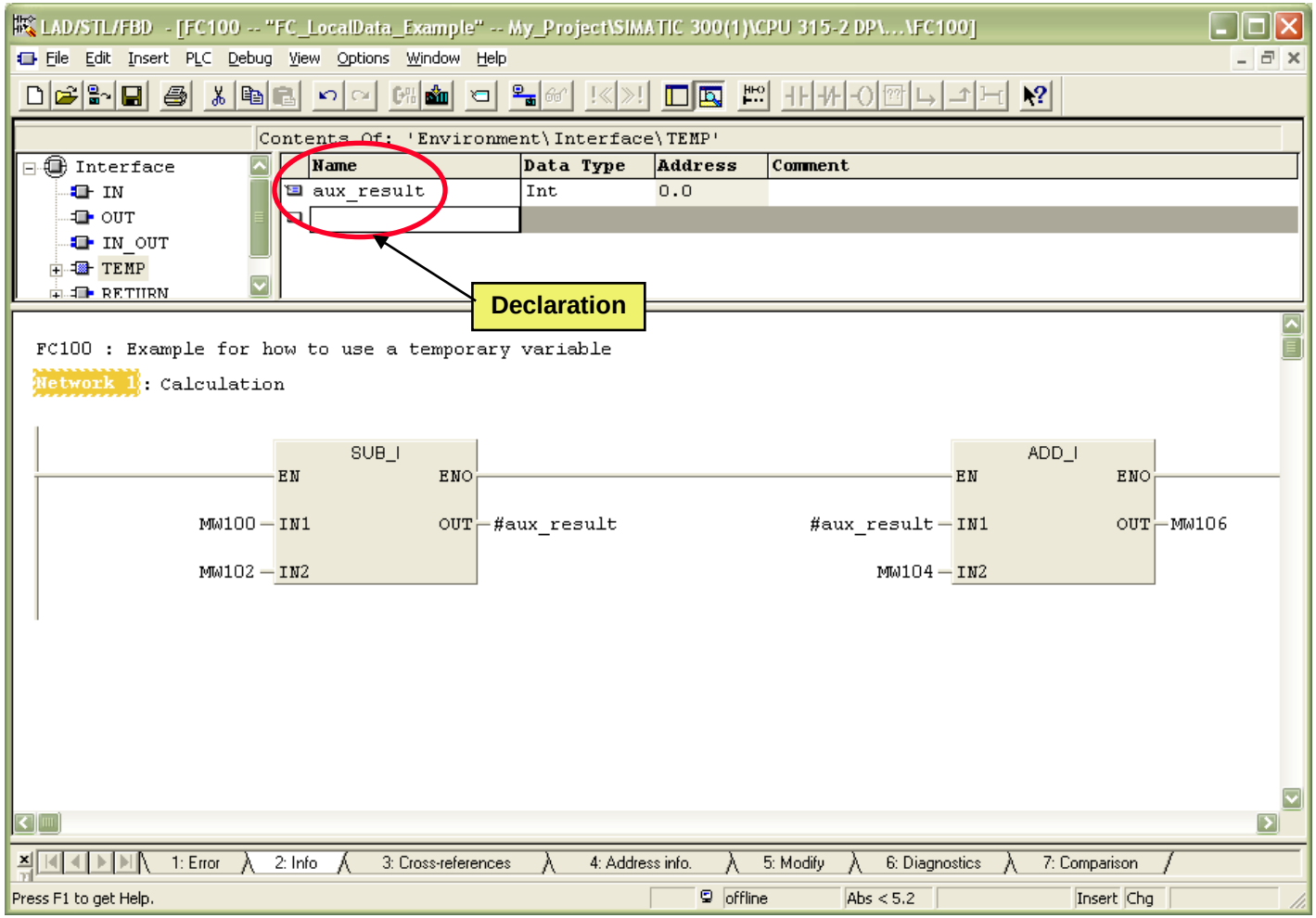
- are overwritten with undefined values after the associated block is executed
- temporary storage in L stack
- usable in OBs / FCs / FBs

Static Variables

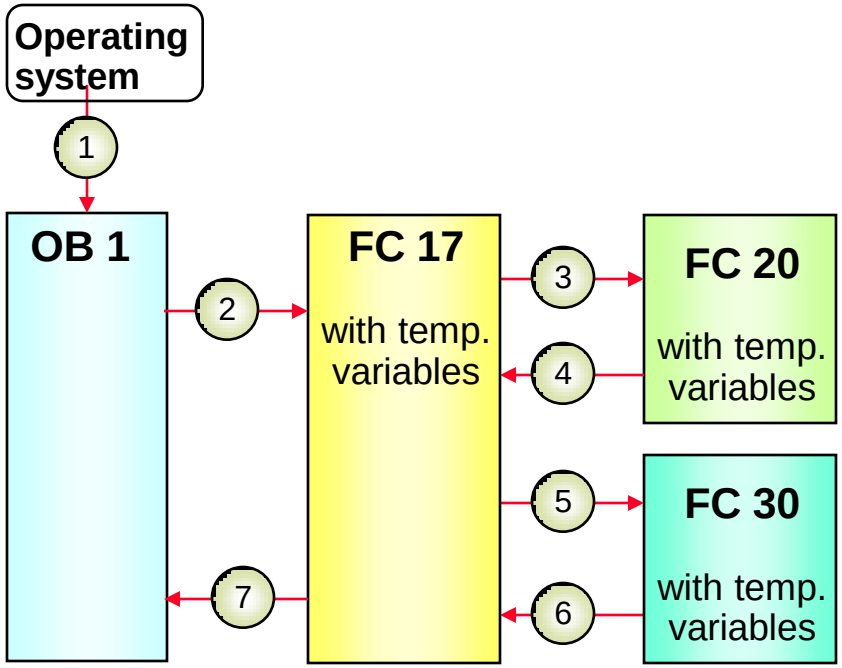
- are retained even after the block is executed
- permanent storage in DBs
- can be used in FBs only

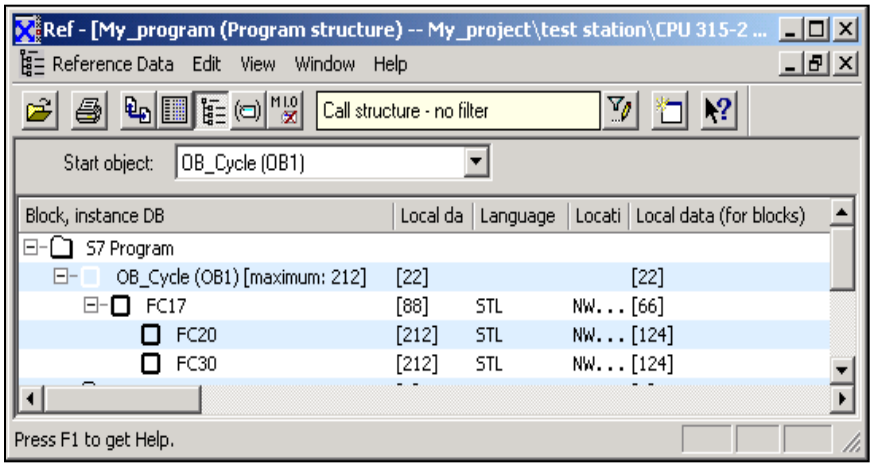


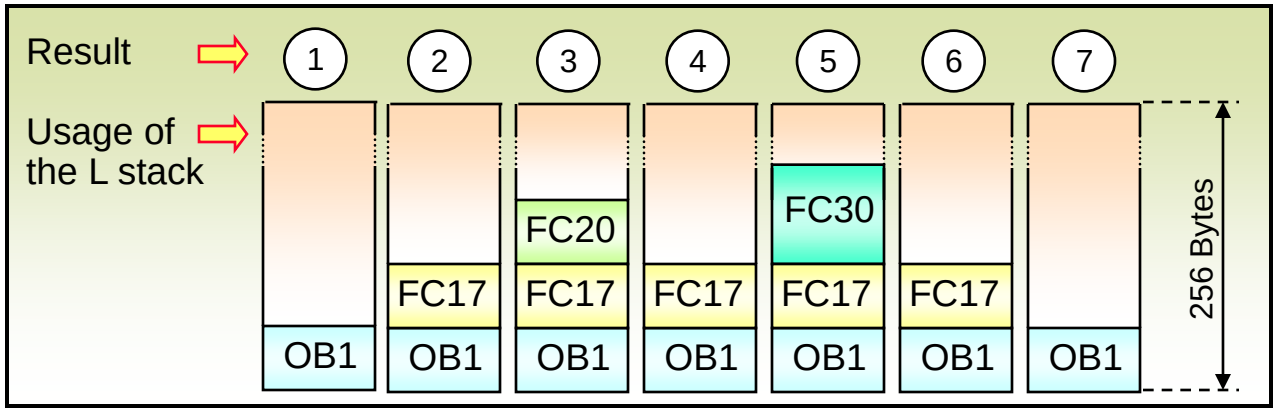
Temporary Variables



Total Usage of the Local Data Stack

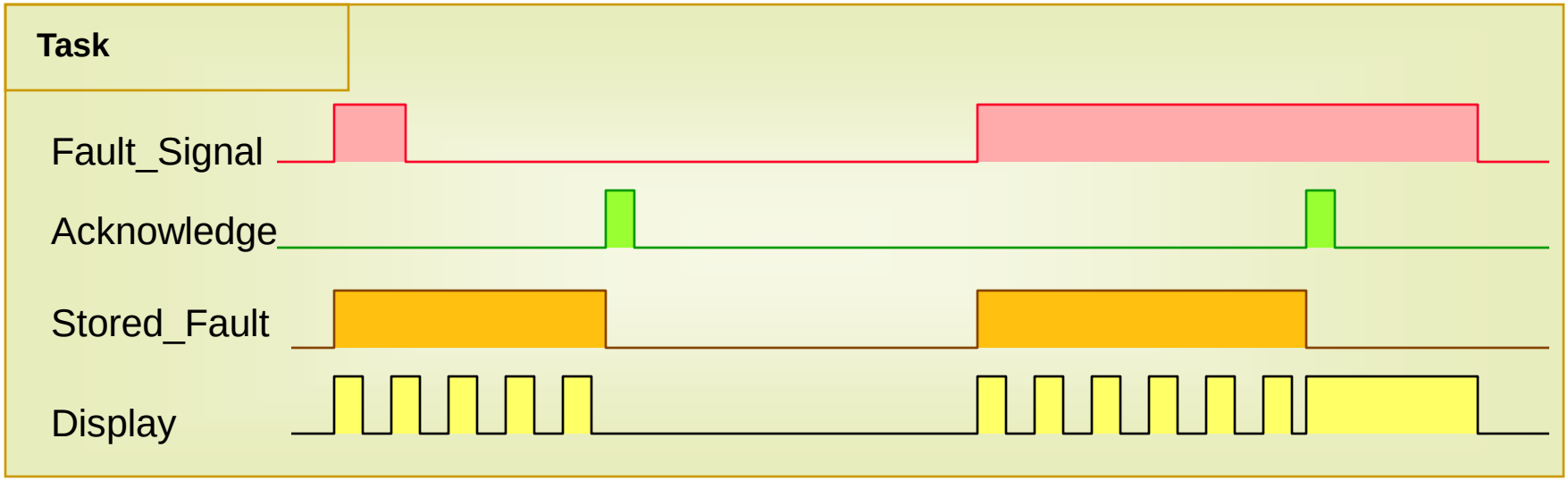




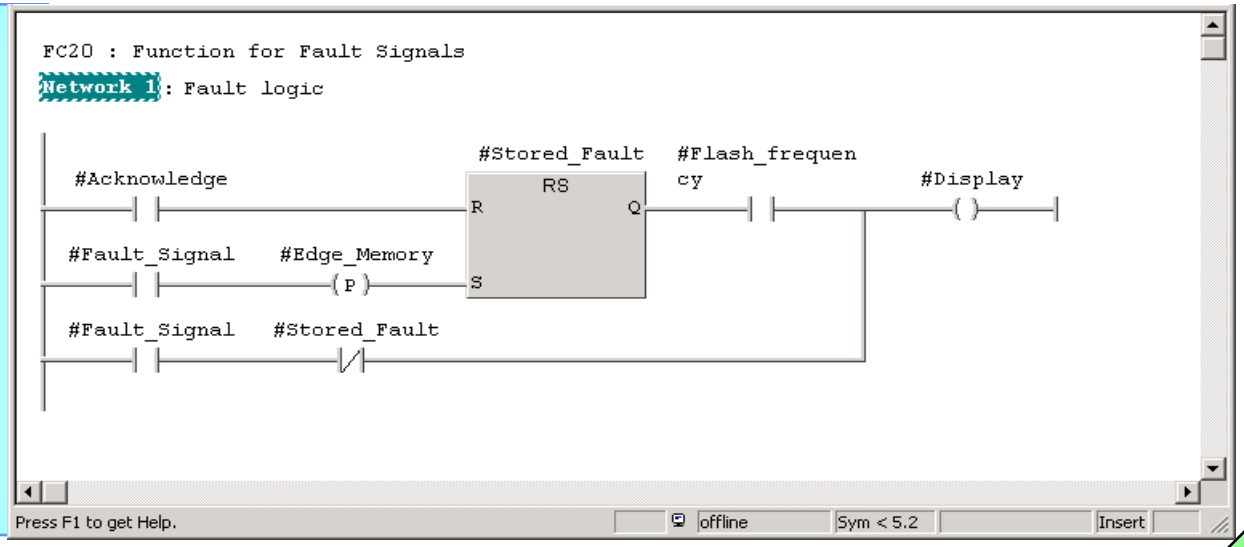


Example for a Fault Display

Task



Solution Suggestion



Parameter-assignable Blocks

Solution with *non-parameter-assignable* block

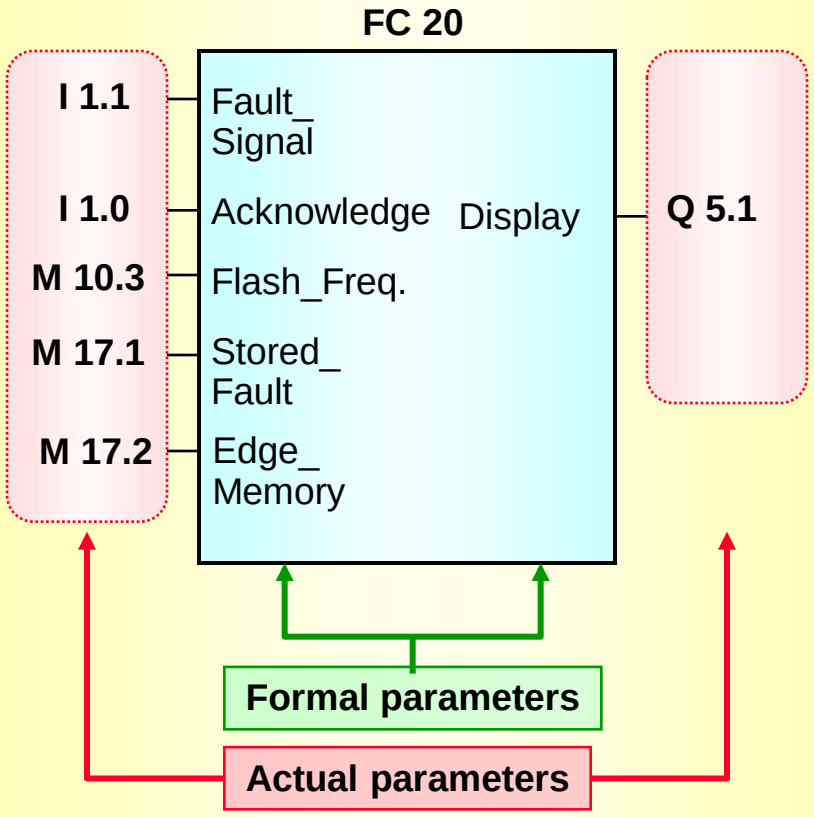
```
A    I   1.1
FP   M   17.2
S    M   17.1
A    I   1.0
R    M   17.1
A    M   17.1
A    M   10.3
O
AN   M   17.1
A    I   1.1
=    Q   5.1
```

Solution with *parameter-assignable* block

Program in FC 20

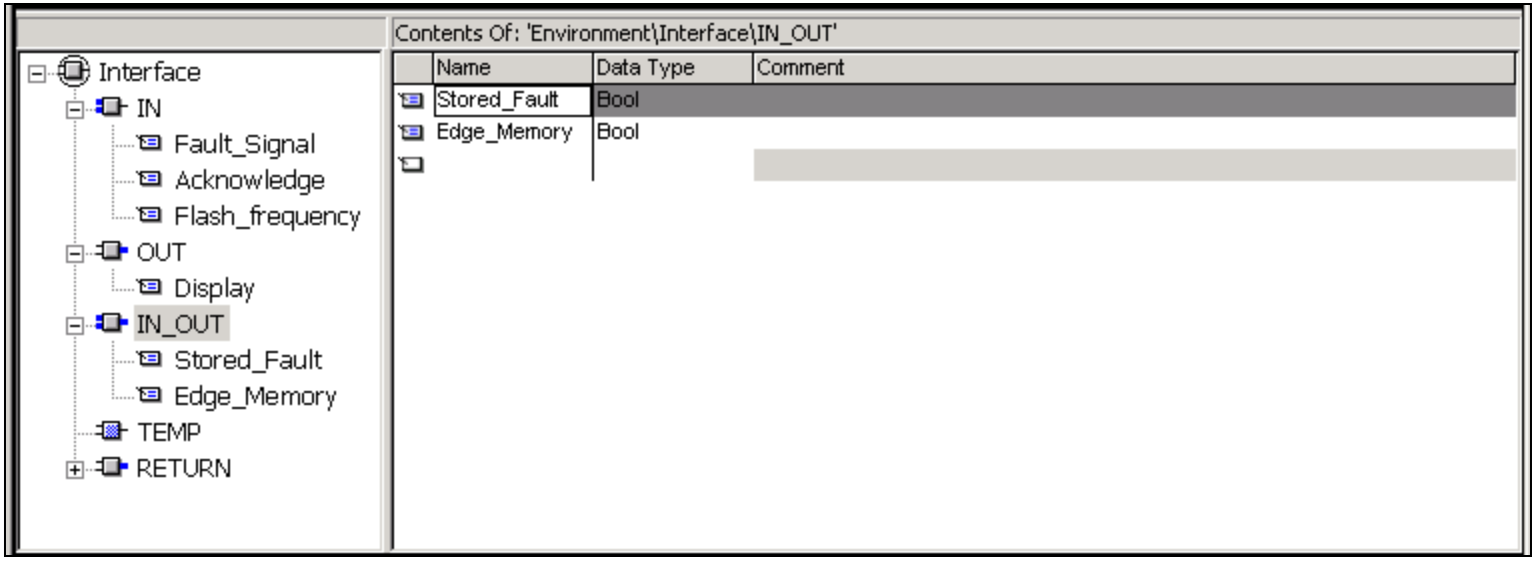
```
A    #Fault_Signal
FP   #Edge_Memory
S    #Stored_Fault
A    #Acknowledge
R    #Stored_Fault
A    #Stored_Fault
A    #Flash_Freq.
O
AN   #Stored_Fault
A    #Fault_Signal
=    #Display
```

Call of FC 20 (e.g. in OB 1)

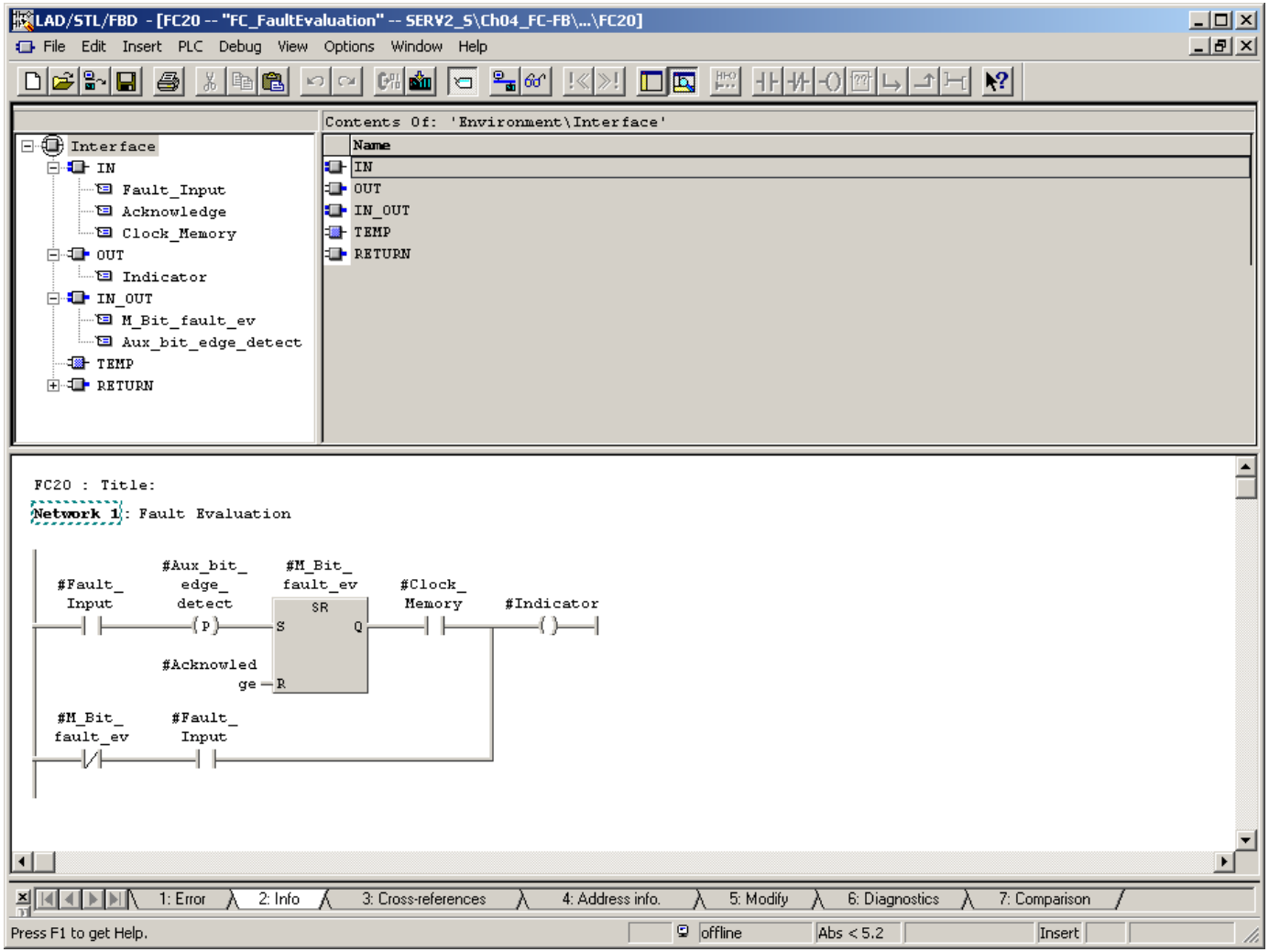


Declaring Formal Parameters

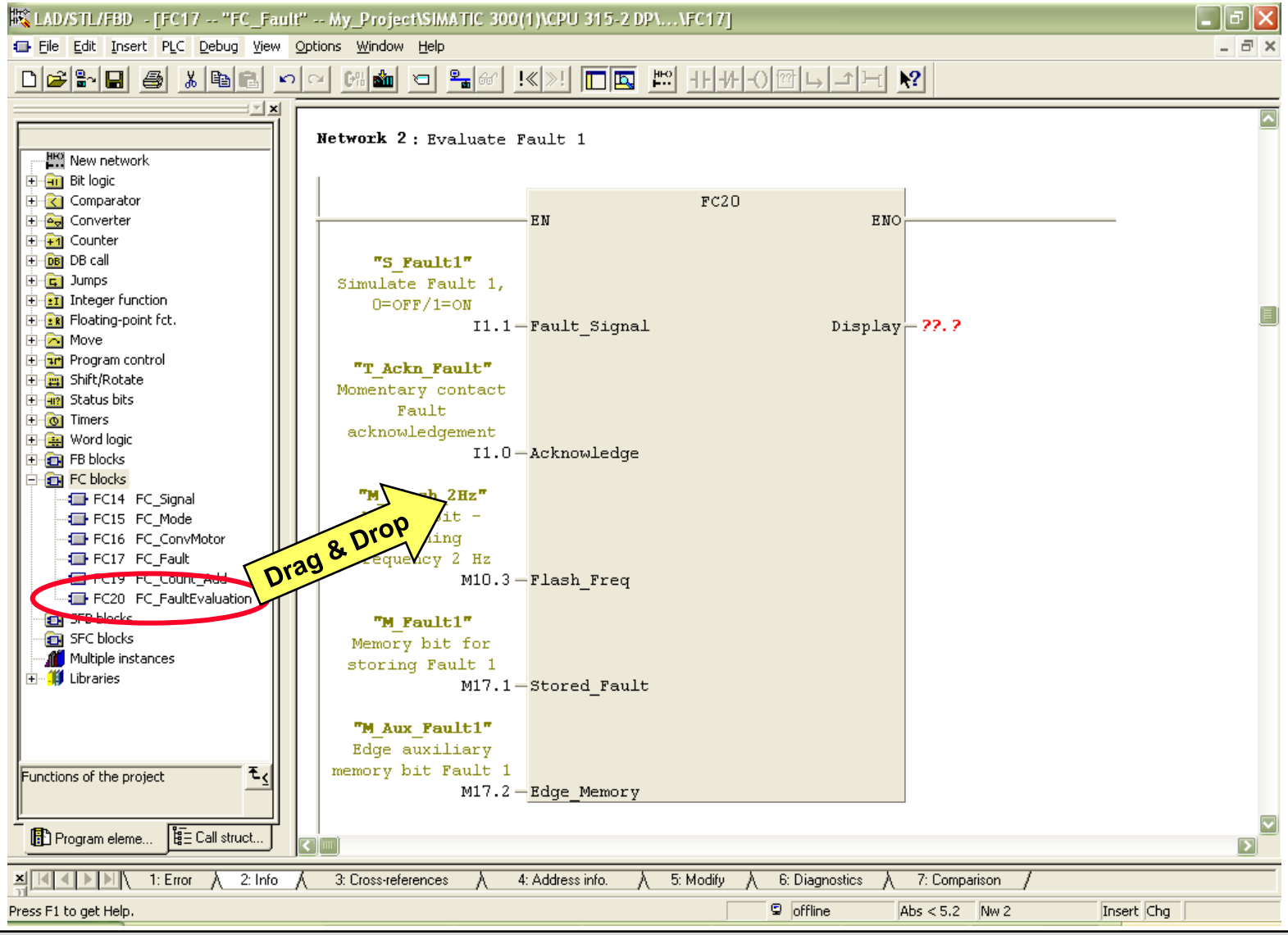
Type of Parameter	Declaration	Use	Graphic Display
Input parameter	in	Read only	To the left of the block box
Output parameter	out	Write only	To the right of the block box
In/Out parameter	In_out	Read / Write	To the left of the block box



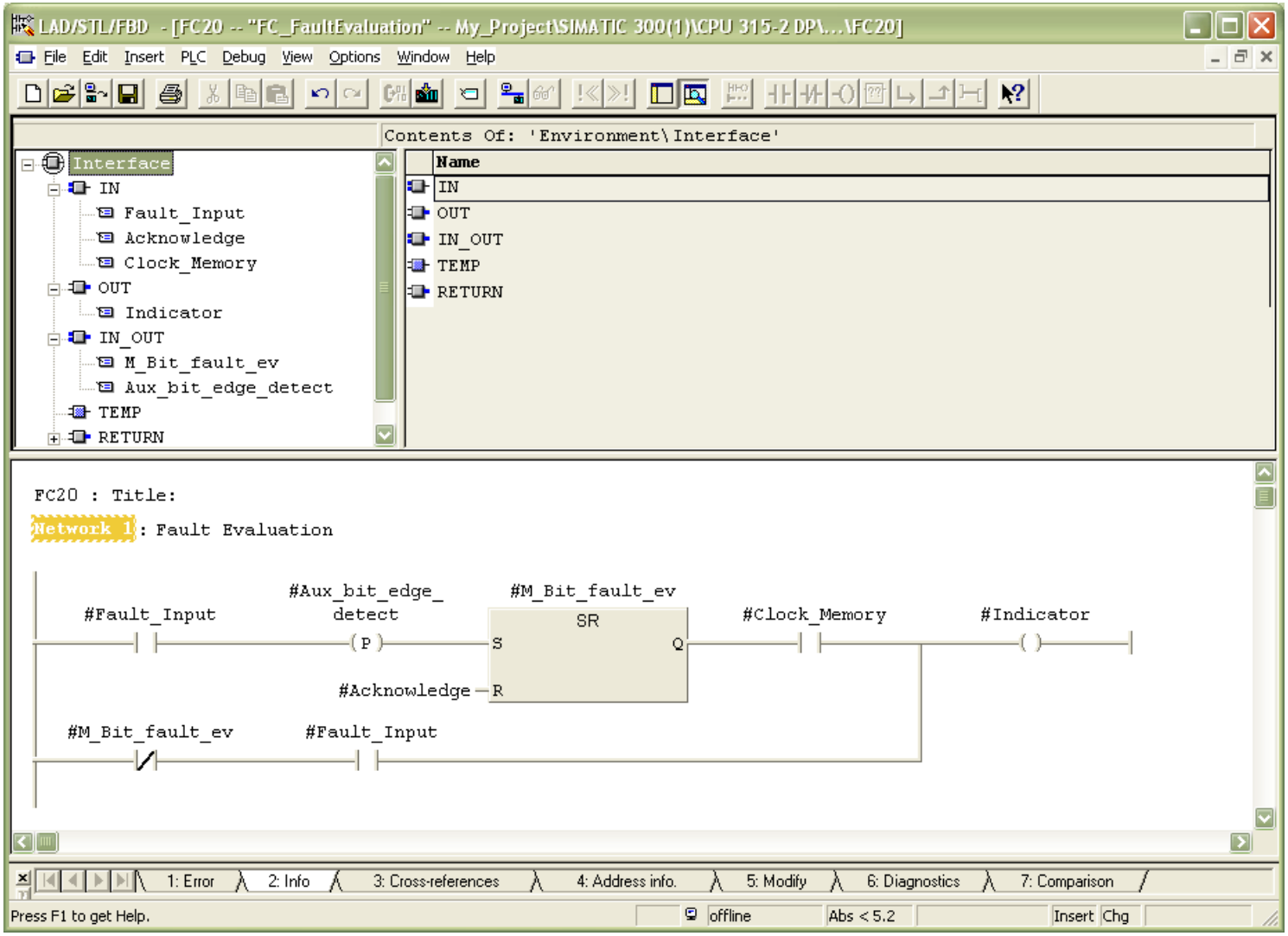
Editing a Parameter-assignable Block



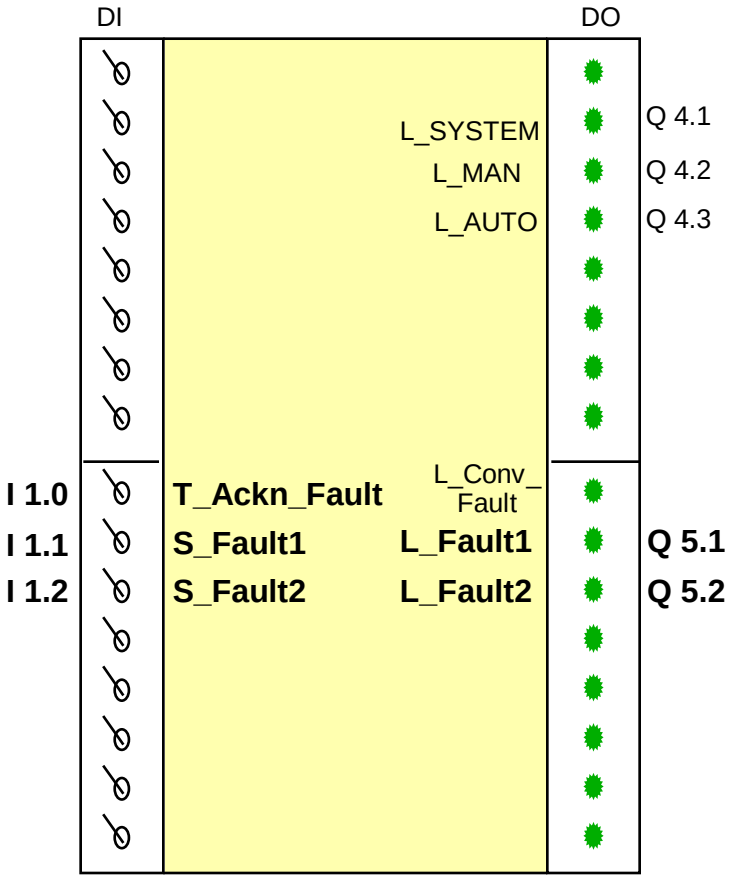
Calling a Parameter-assignable Block

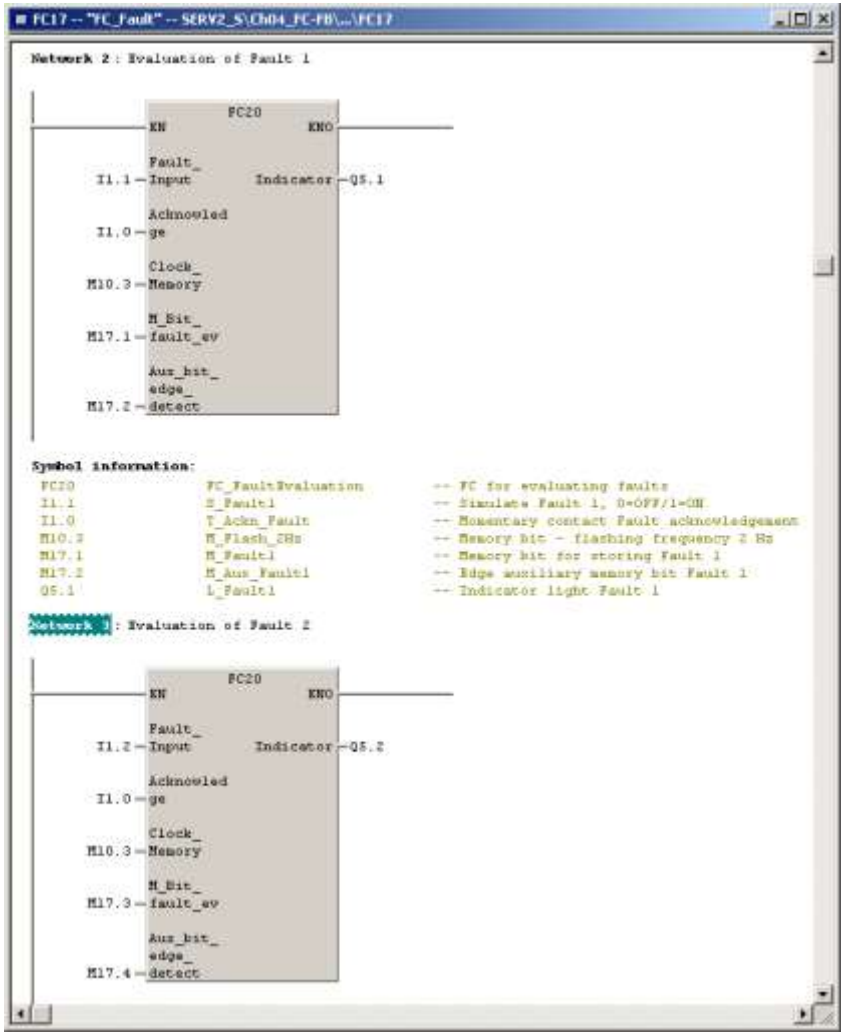


Exercise 1: Editing the Parameter-assignable Function FC 20

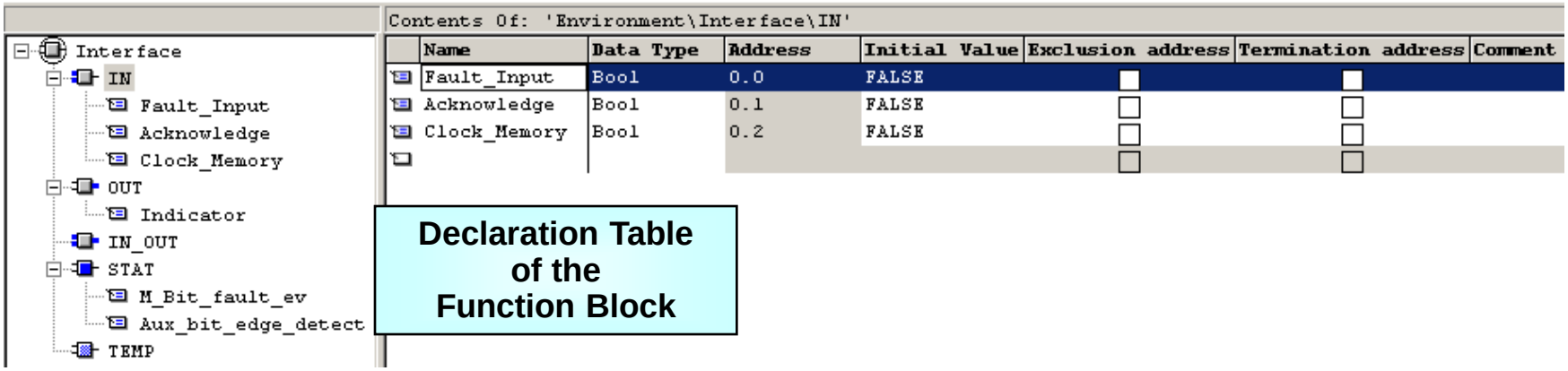


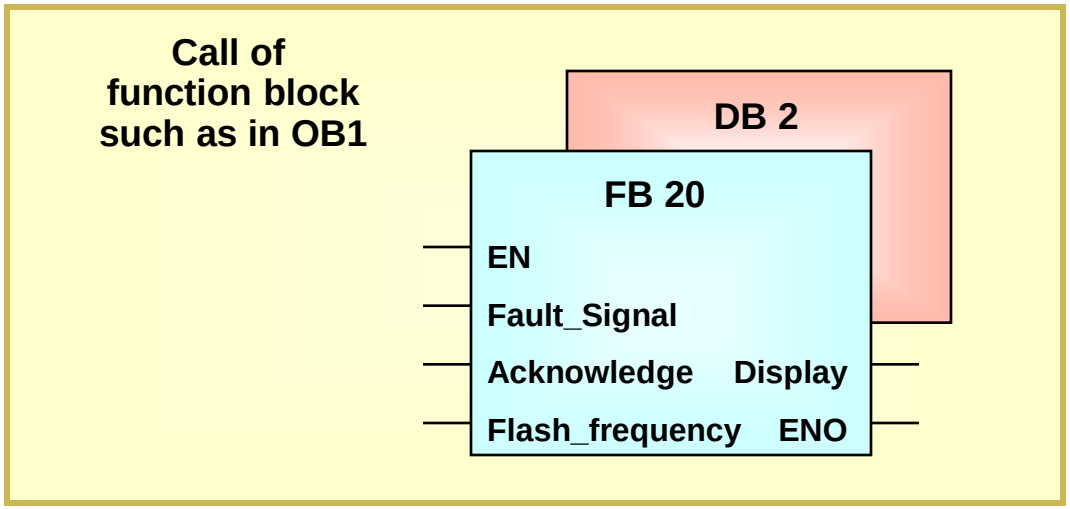
Exercise 2: Calling the Parameter-assignable Function FC 20



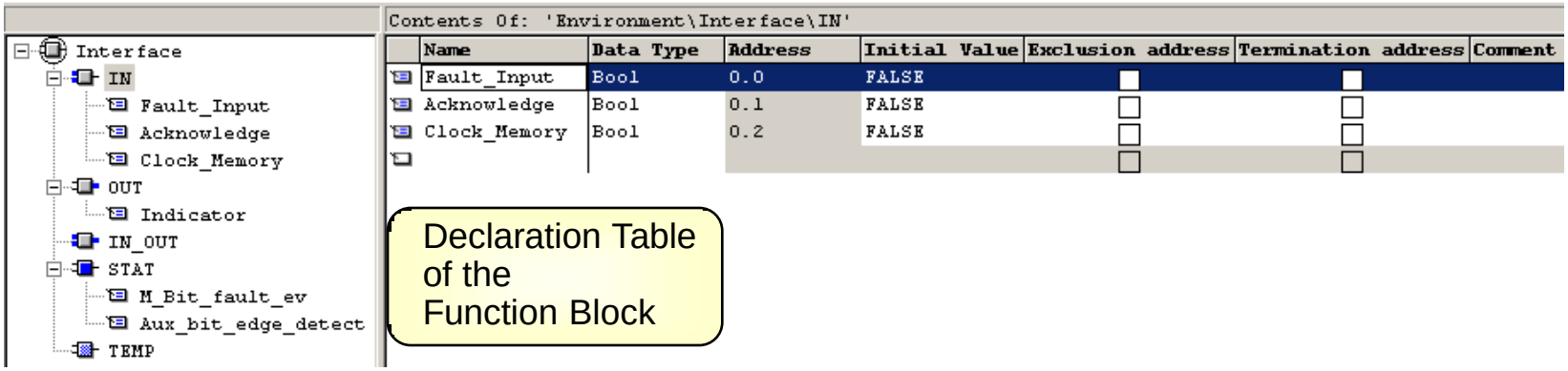


Function Blocks (FB)

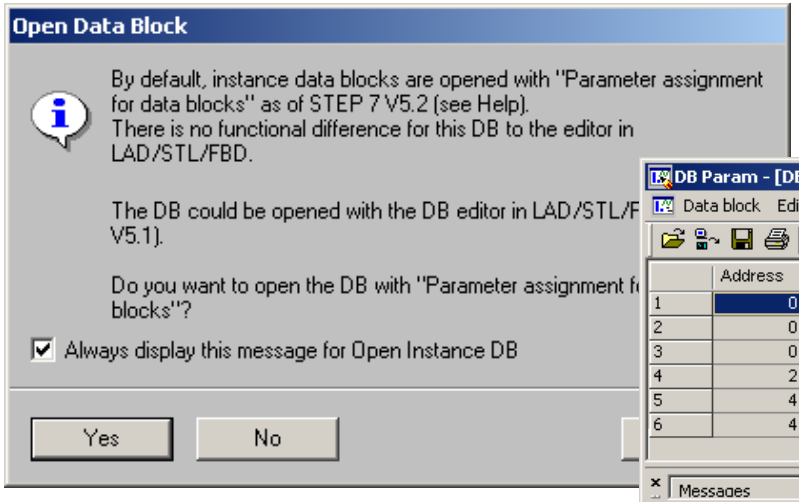




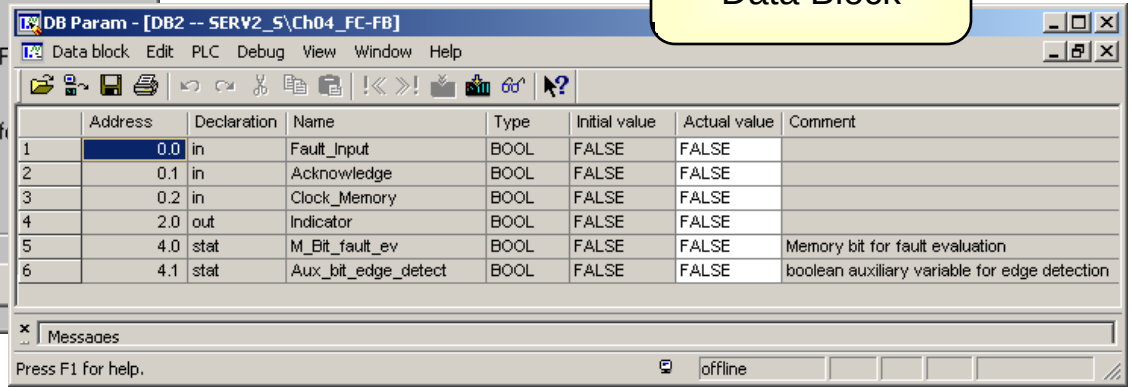
Function Block for Fault Display



Declaration Table of the Function Block



Instance Data Block

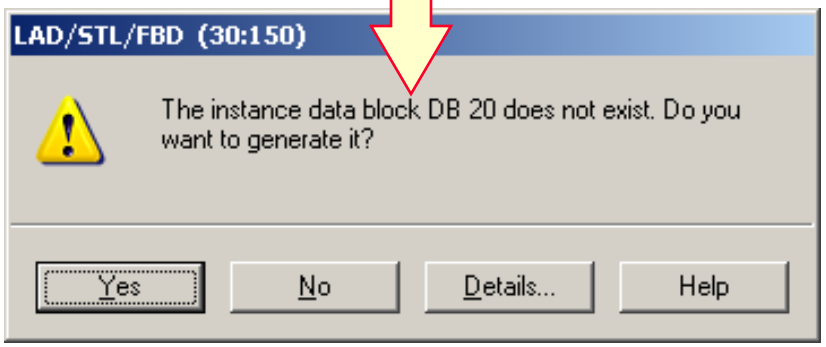


Generating Instance Data Blocks

1. Generate instance DB with FB call

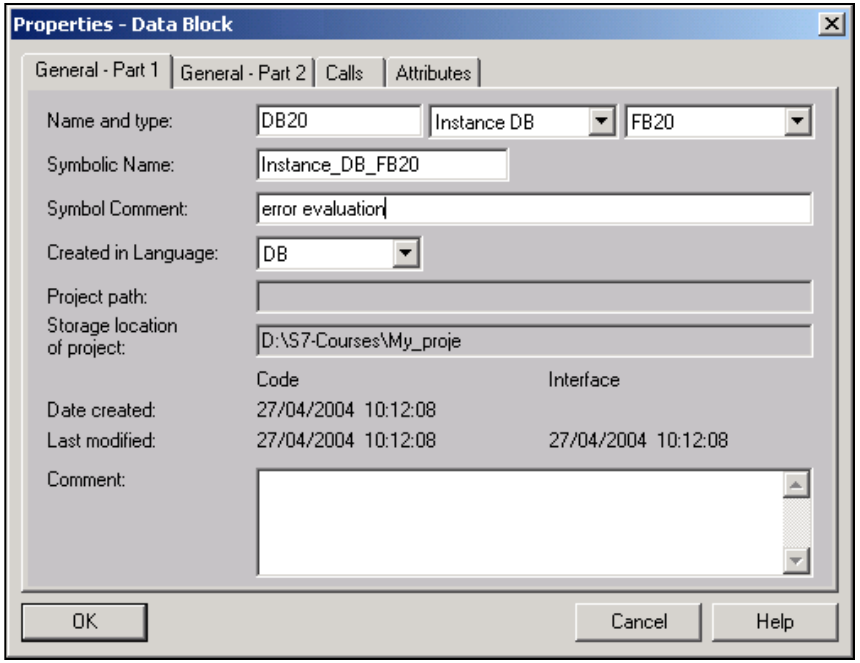
In the LAD / FBD / STL Editor

```
OB1 :   "Main Program Sweep (Cycle)"
Network 1: Fault Indicator
call fb20, db20
```

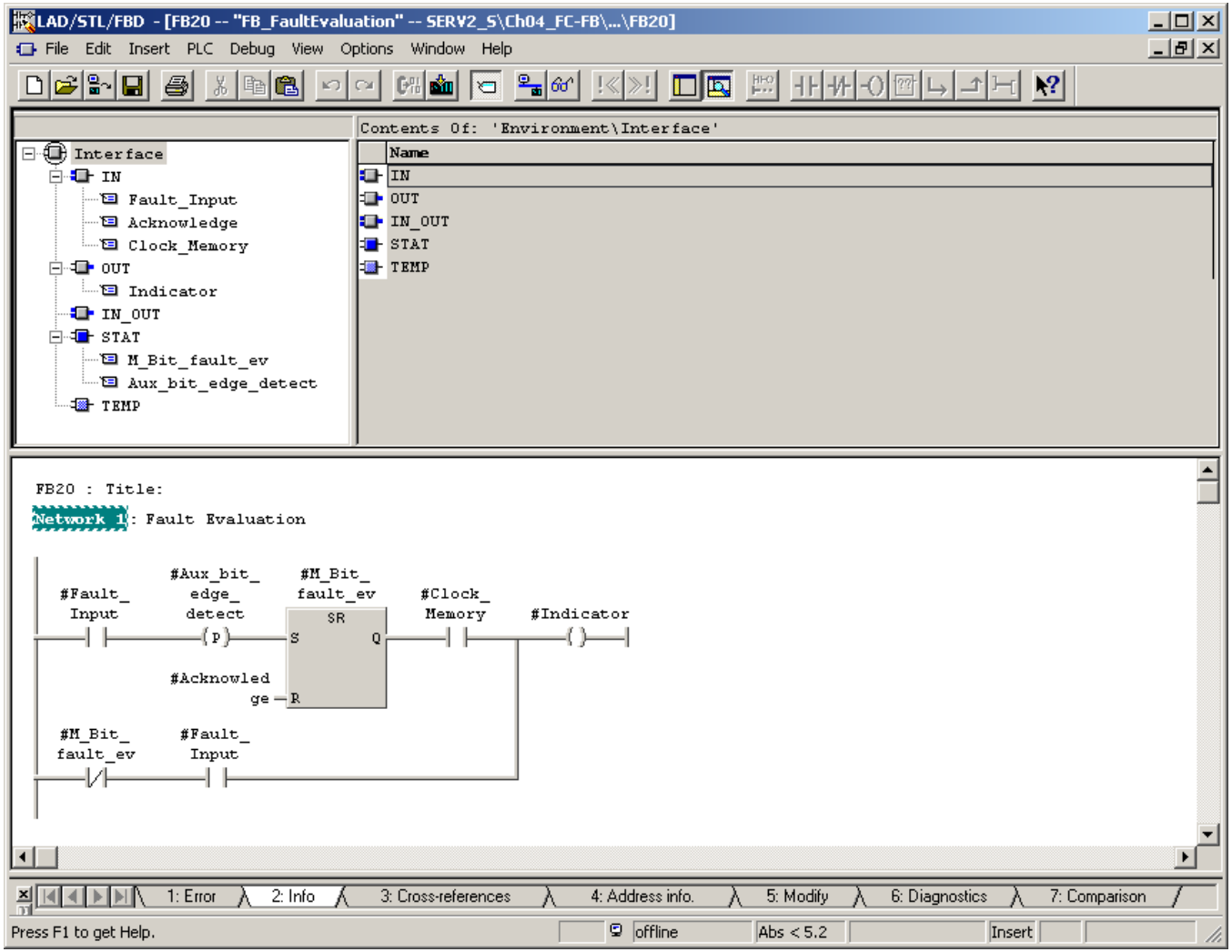


2. Create new instance DB

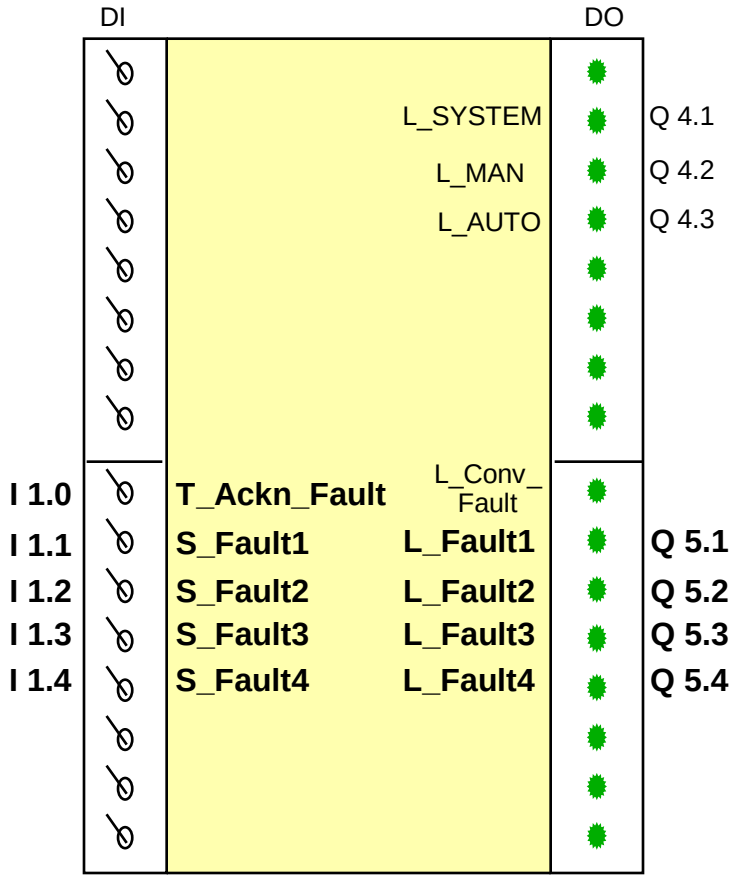
In the SIMATIC Manager

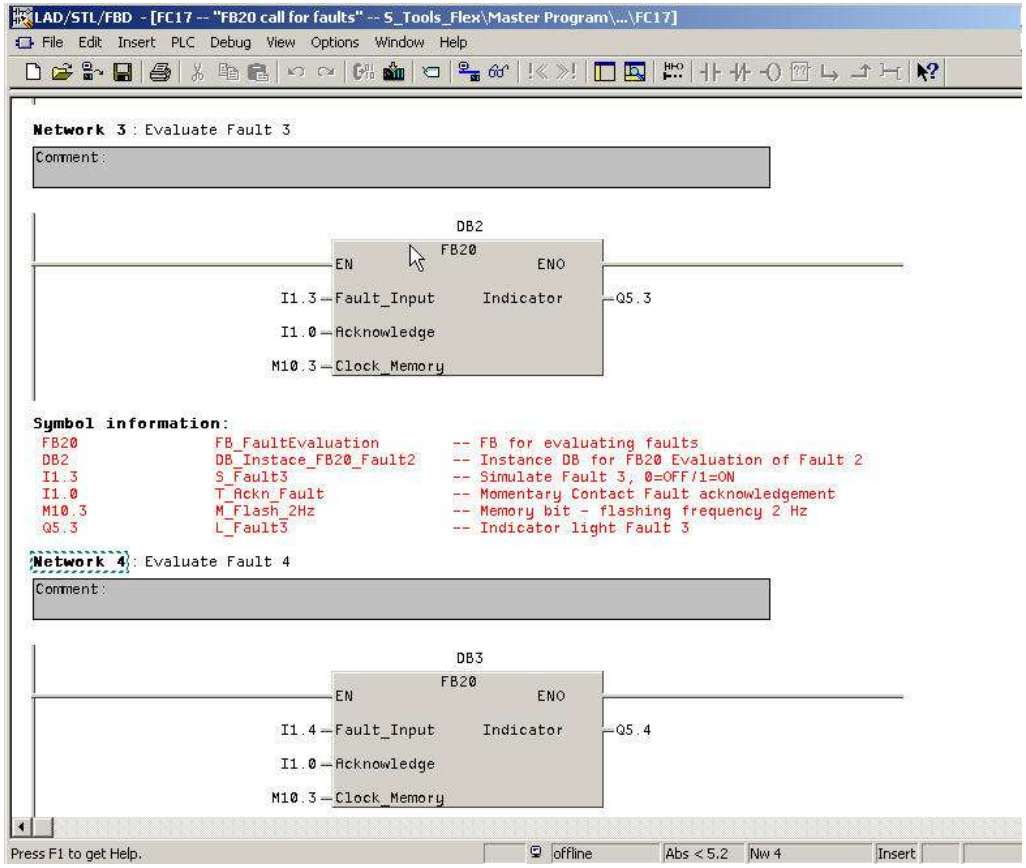


Exercise 3: Editing the Parameter-assignable Function Block FB 20

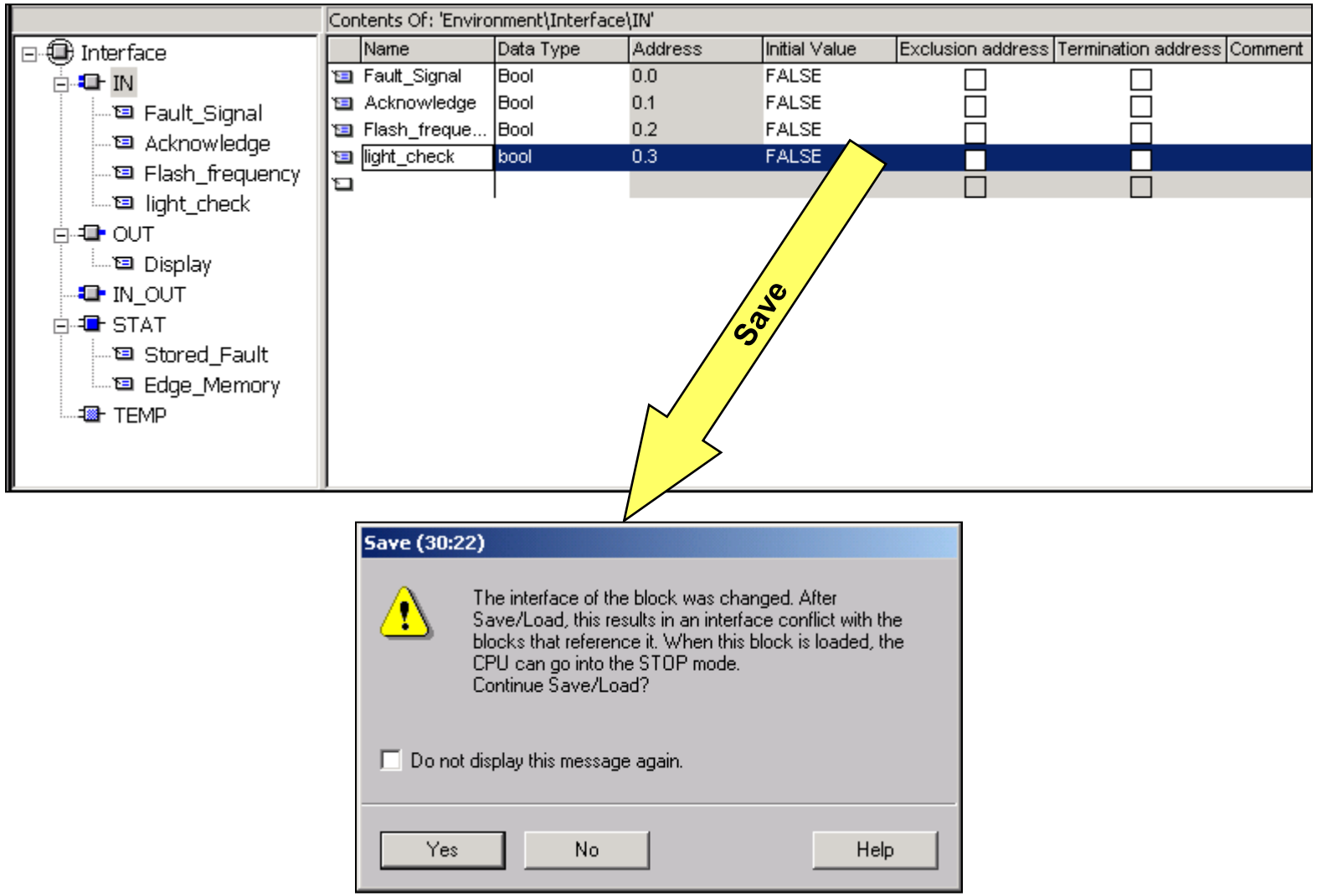


Exercise 4: Calling the Parameter-assignable Function Block FB 20

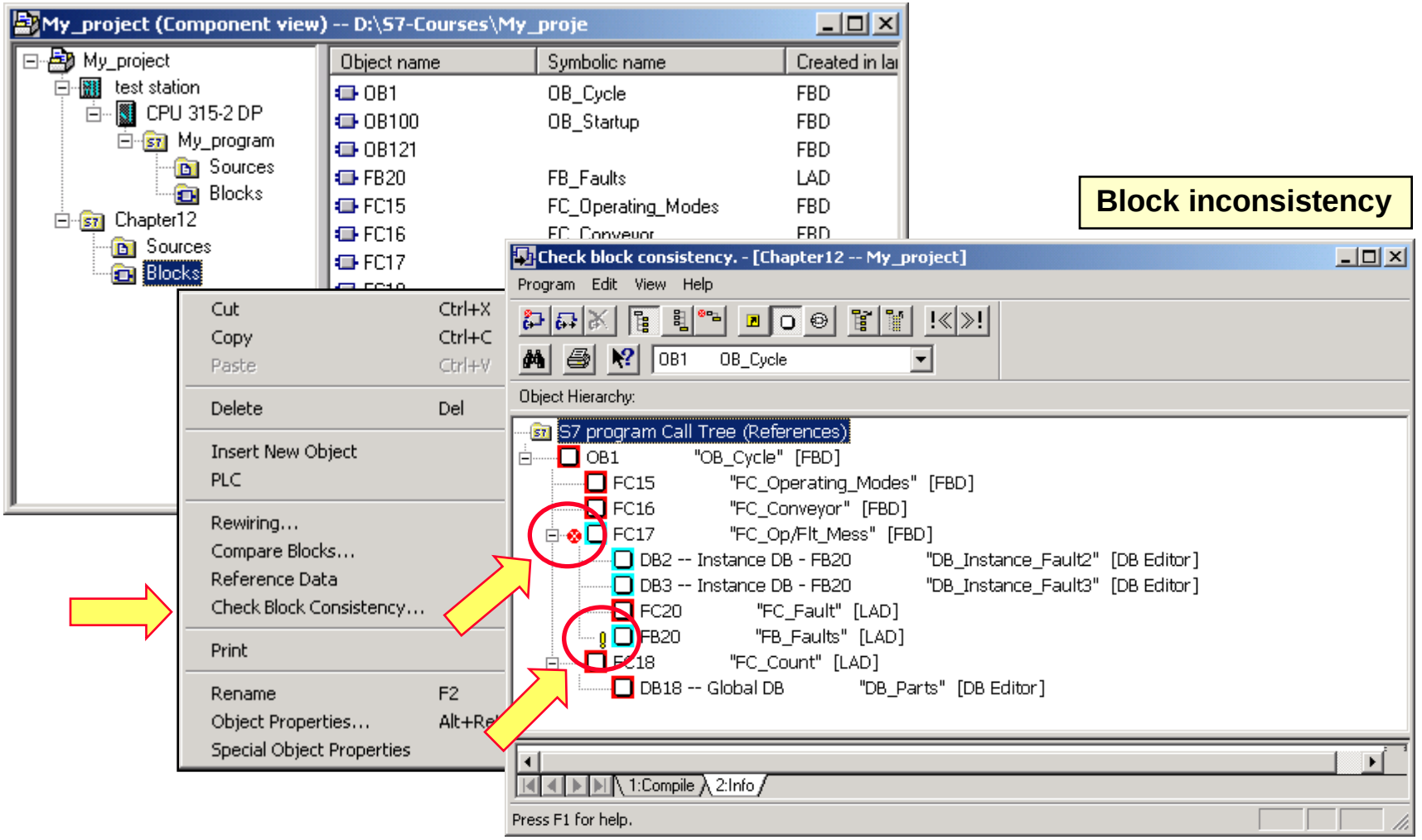




Inserting / Deleting Block Parameters Later On

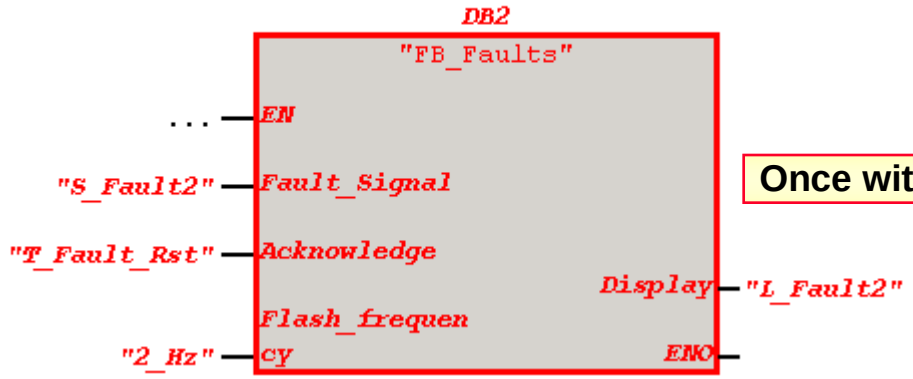


Checking the Block Consistency

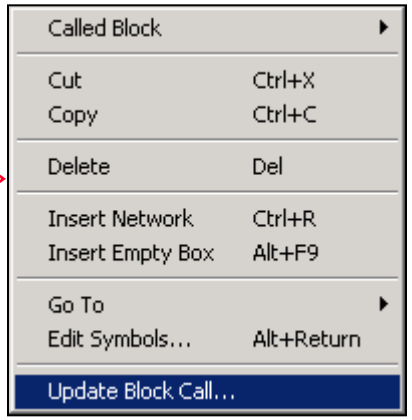


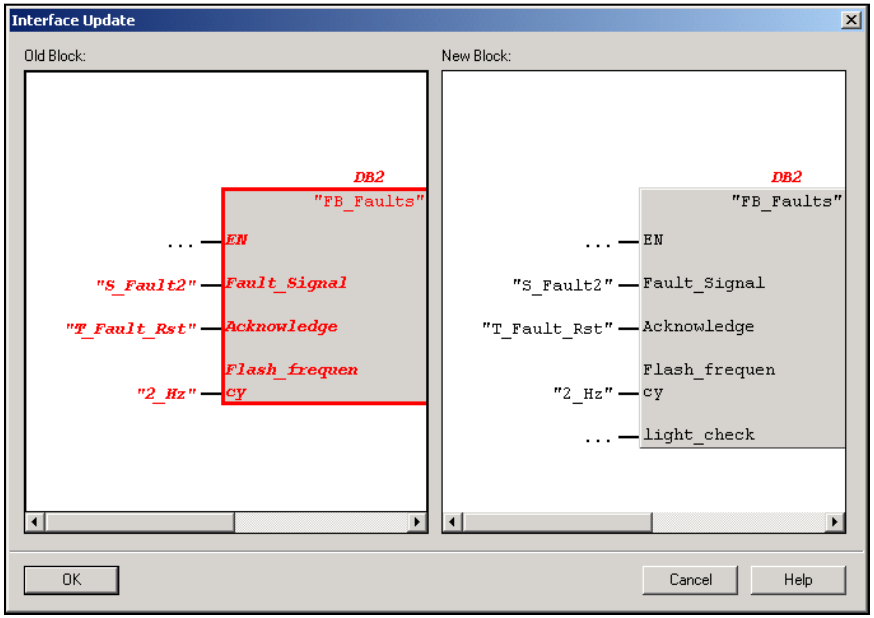
Corrections when Calling Modified Blocks

Network 3 : Evaluation of disturbance 2

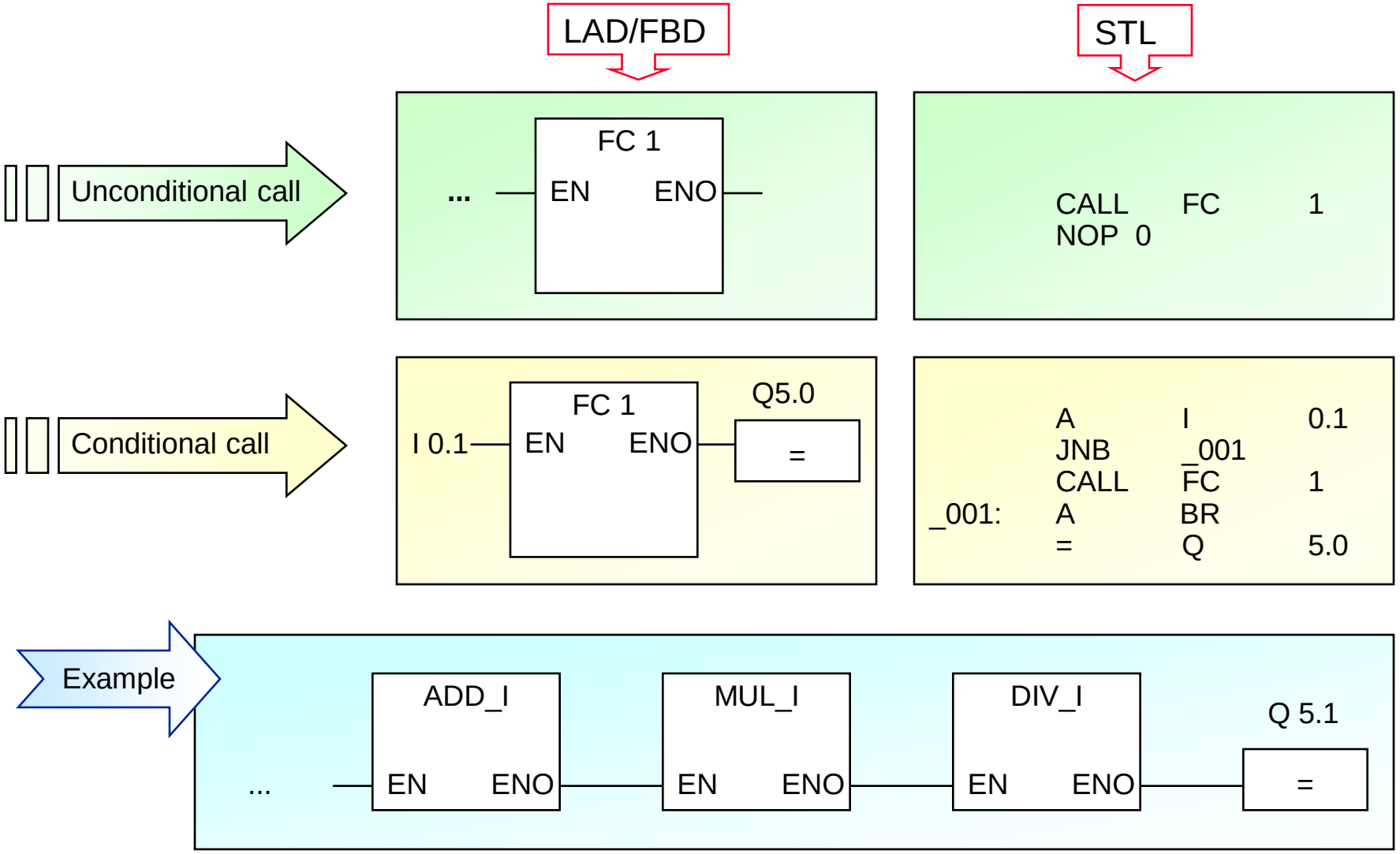


Once with the right





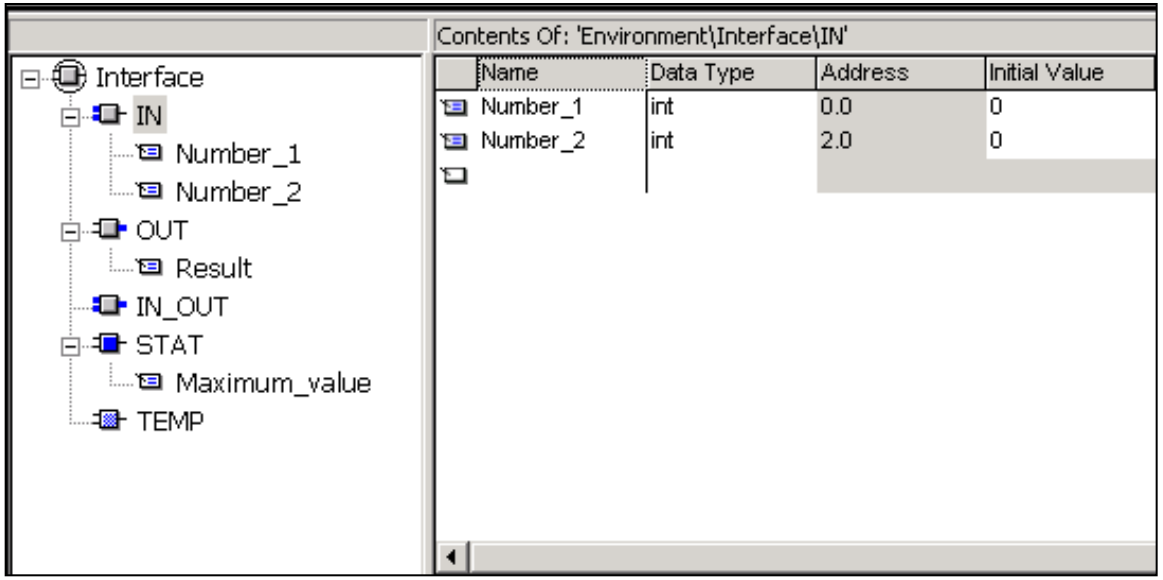
Using the EN and ENO Output



Summary: Block Calls

Language	FC		FB	
	without parameters	with parameters	w/o parm., w/o stat var.	with parm., and/or stat var
STL	• CALL FC1 • UC FC1 • CC FC1	• CALL FC2 Par1: ... Par2: ... Par3: ...	• UC FB1 • CC FB1	• CALL FB2, DB3 Par1: ... Par2: ... Par3: ...
LAD	FC1 (CALL) FC1 EN ENO	FC2 EN ENO Par1 Par2 Par3	not available	DB3 FB2 EN ENO Par1 Par2 Par3
FBD	FC1 CALL FC1 EN ENO	FC2 EN Par3 Par1 Par2 ENO	not available	DB3 FB2 EN Par3 Par1 ENO Par2

Exercise 5: Recognizing Operand Types



Statement	Operand Type					
	Global	Local	Absolute	Symbolic	Static	Parameter
L #Number_1						
L #Number_2						
T #Maximum_value						
L „Number_1"						
T MW 40						
T #Number_2						

If You Want to Know More

Comparison of Functions and Function Blocks

Program execution

```
CALL  FC   10
On_1  :=   I 0.1
On_2  :=   I 0.2
Off   :=   Q 8.0
```

FC10

```
in     On_1   BOOL
in     On_2   BOOL
out    Off    BOOL
...
...
A      #On_1
A      #On_2
=      #Off
...
```

```
CALL  FB5, DB5
     Start     :=I 0.0
     Stop      :=I 0.1
     Motor_on
               :=Q8.0
     Speed     :=QW12
```

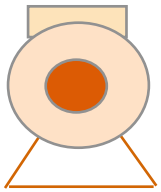
DB 5

```
0.0  Start     BOOL
0.1  Stop      BOOL
2.0  Motor_on  BOOL
4.0  Speed     INT
```

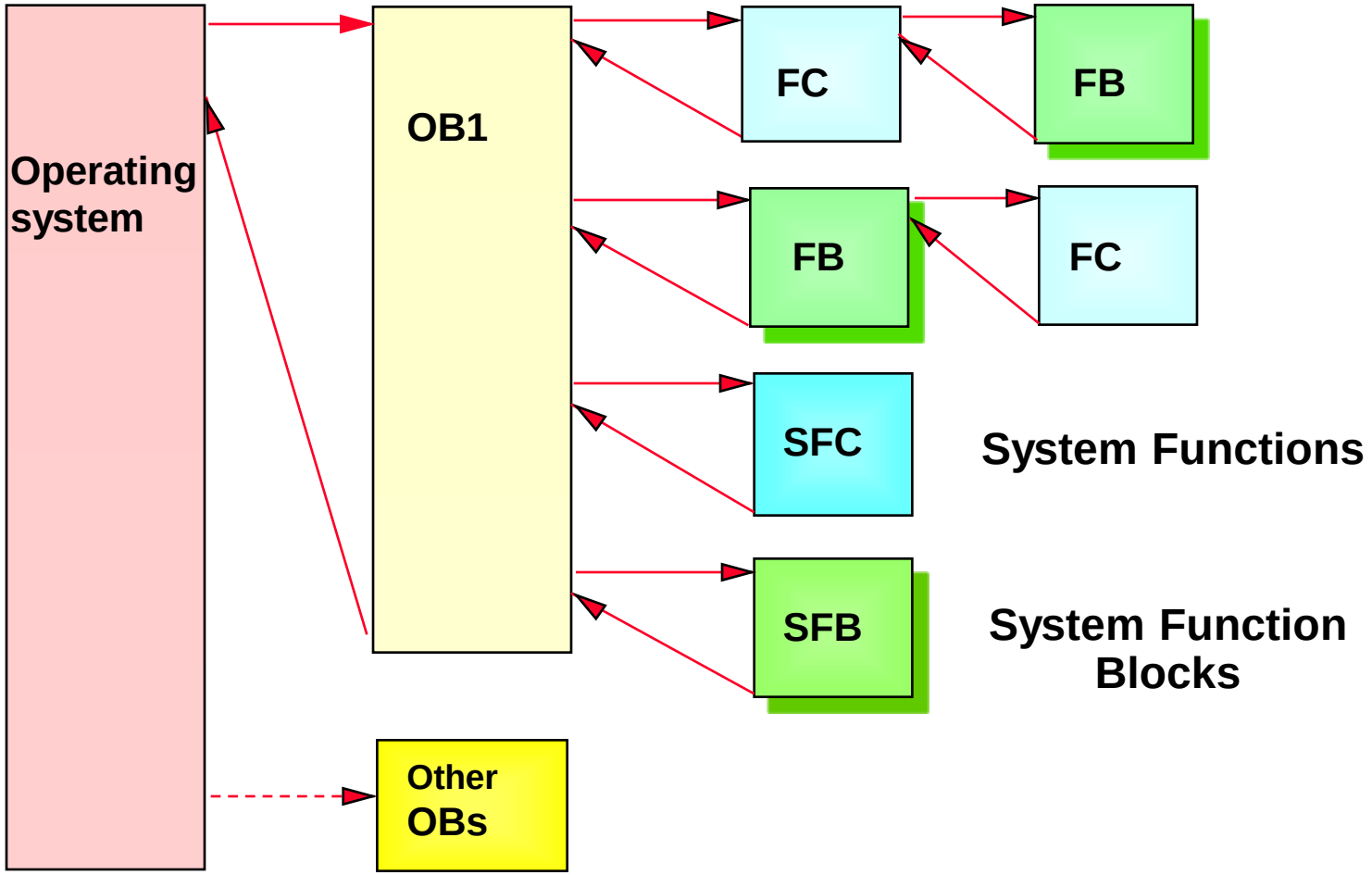
FB 5

```
in     Start     BOOL
in     Stop      BOOL
out    Motor_on  BOOL
out    Speed     INT
stat   ...
Temp...
...
A      #Start
AN     #Stop
=      #Motor_on
...
```

Drive



Organization Blocks (1)



Objectives

Upon completion of the chapter the participant will ...

... know the purpose of temporary variables

... know the organization blocks that are available

... understand the difference between "Warm Restart", "Hot Restart" and "Cold Restart"

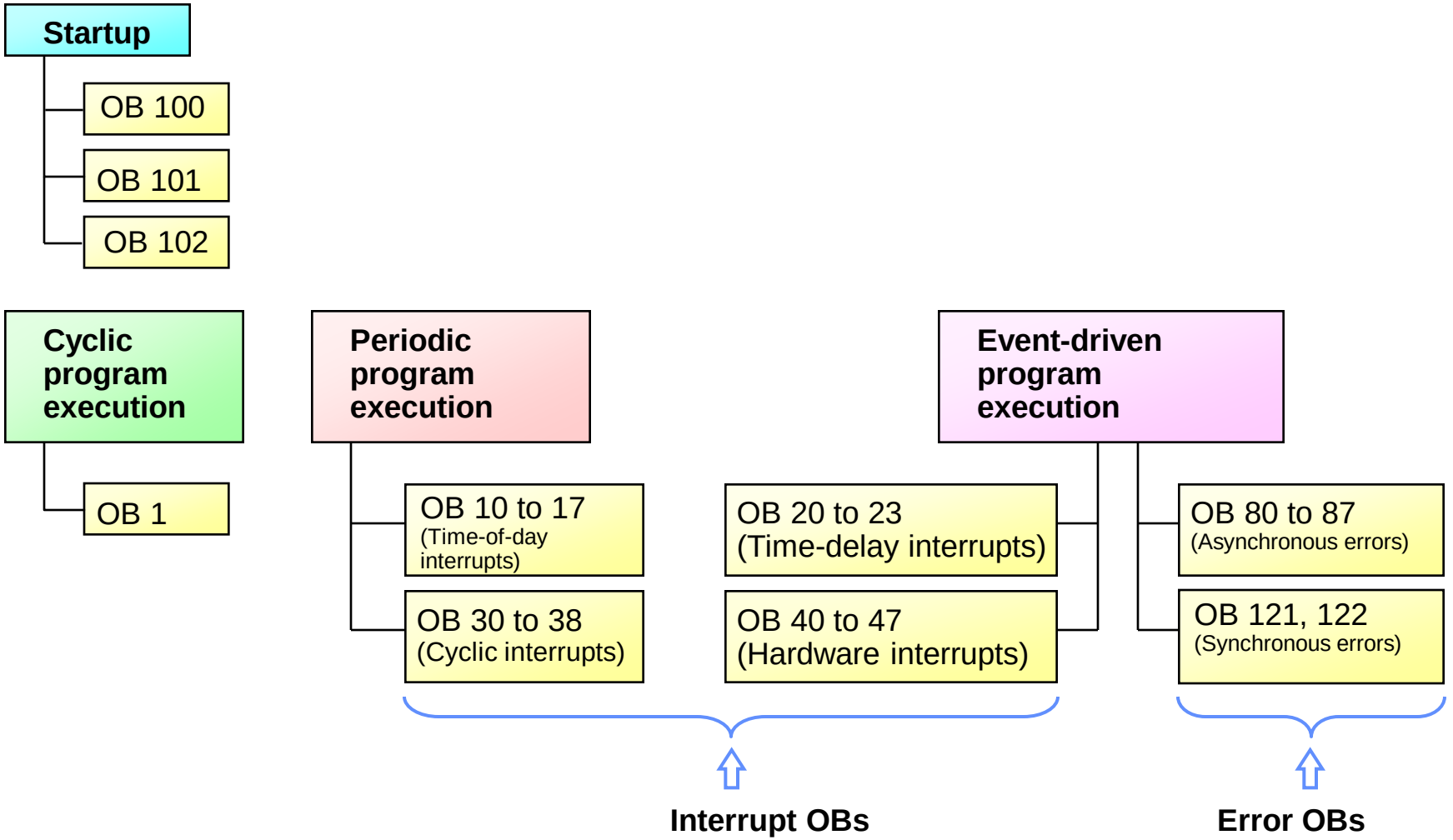
... be able to explain the principle of interrupt processing

... know the "Time-of-day Interrupt", "Cyclic Interrupt", "Hardware Interrupt" and "Diagnostic Interrupt"

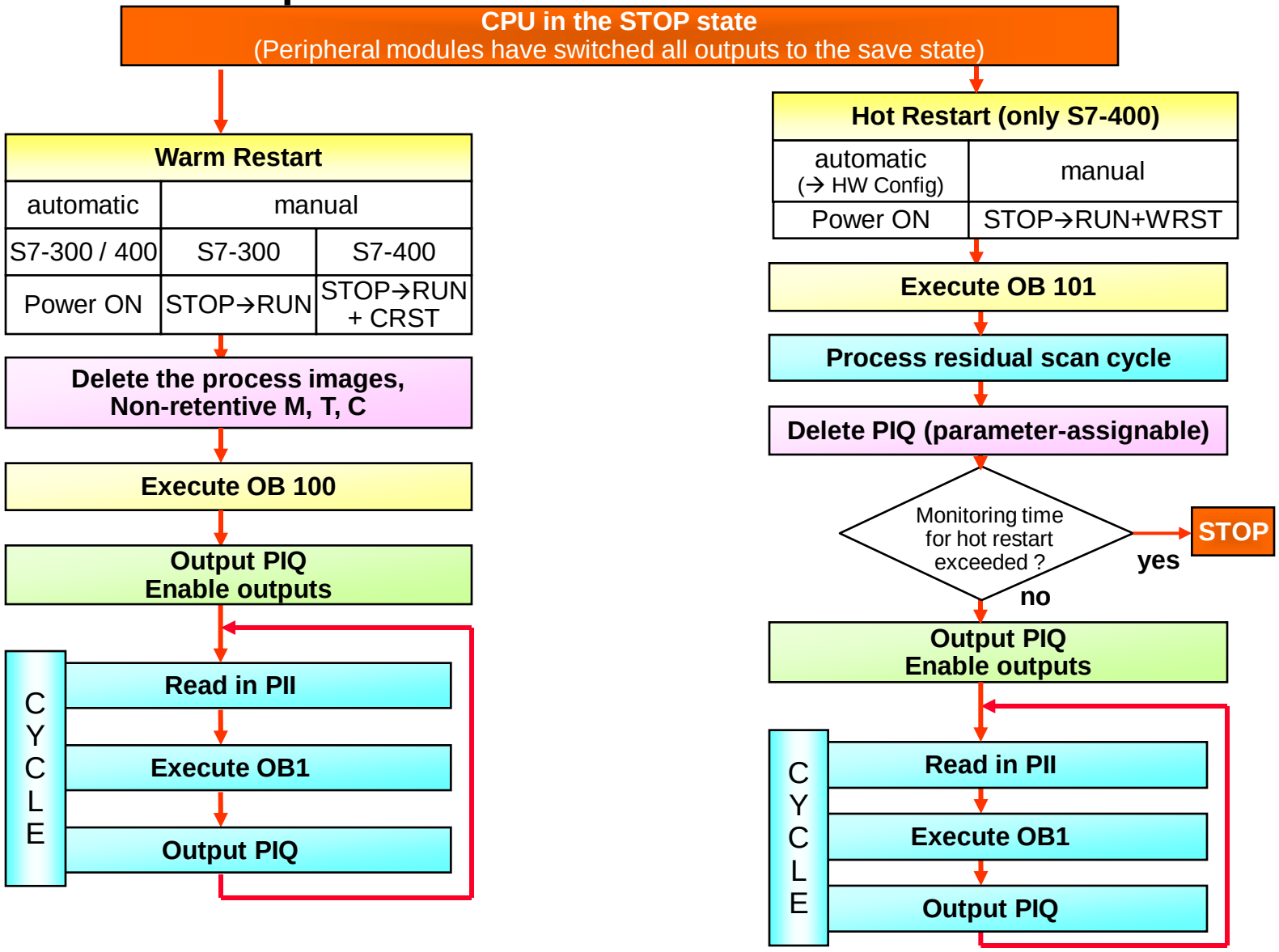
... know and be able to use the error OBs

... be able to interpret the OB start information

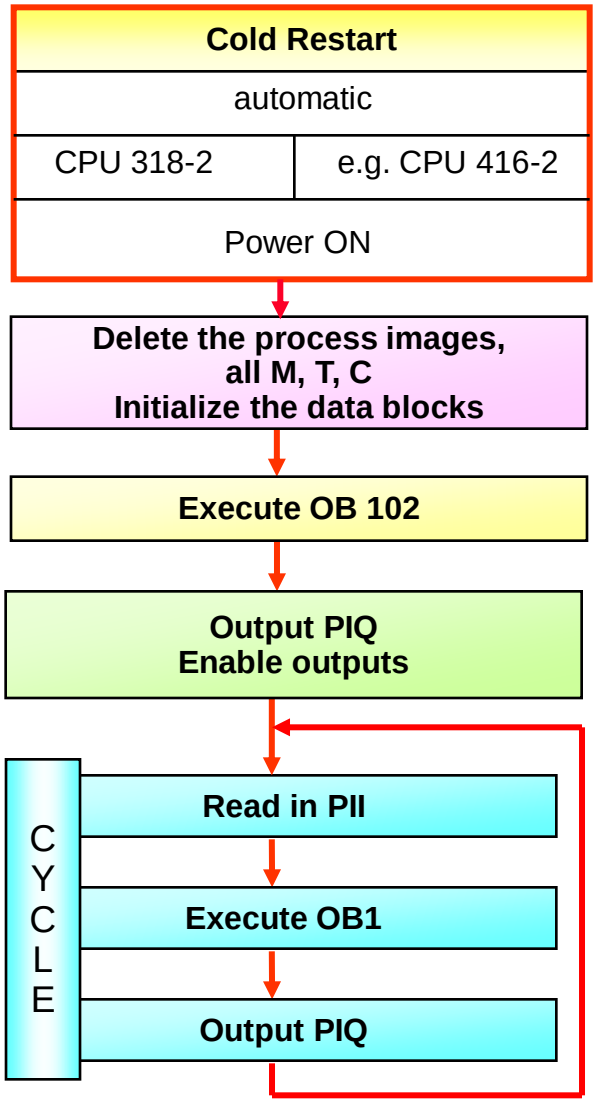
Overview of the Organization Blocks



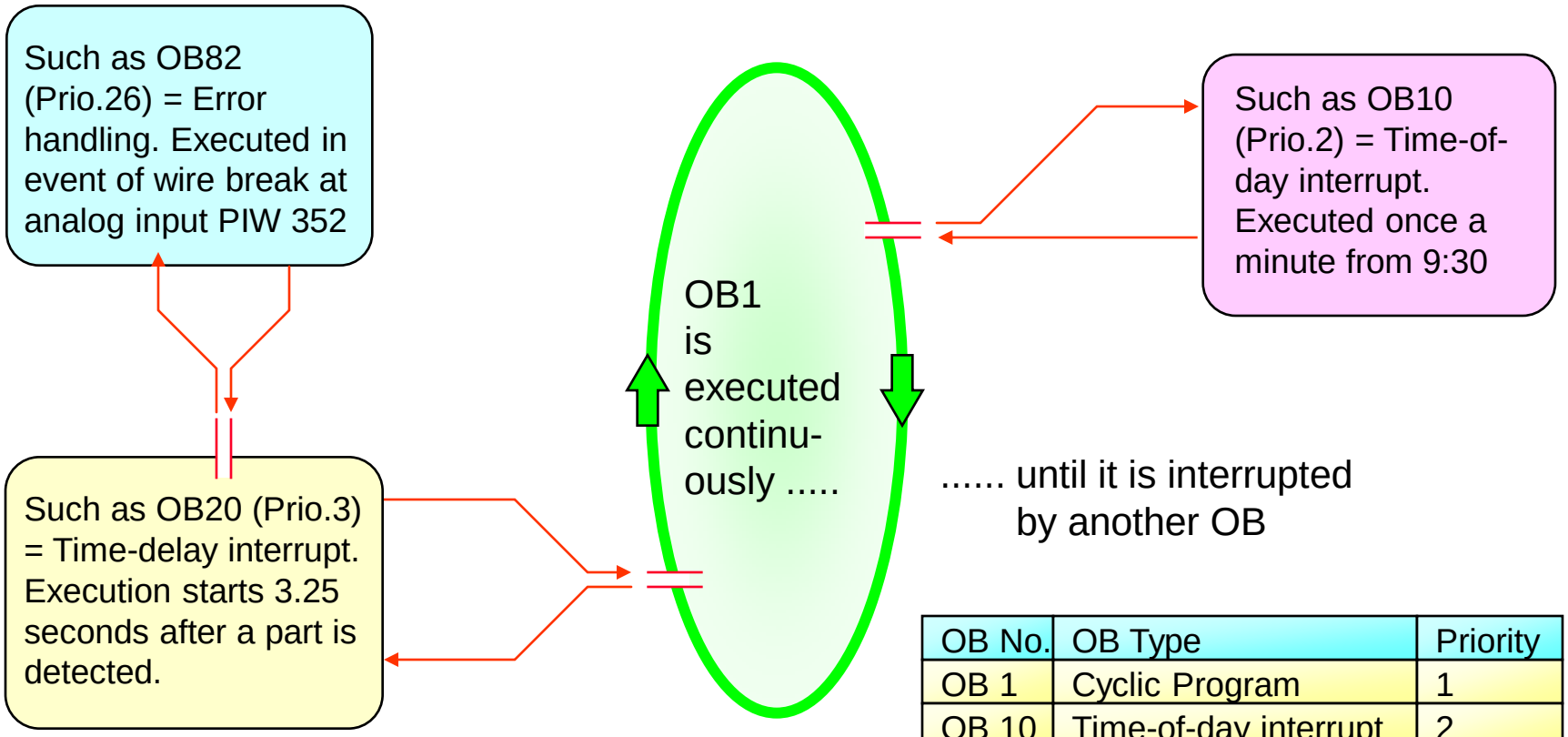
Startup OBs for Warm Restart and Hot Restart



Startup OB for Cold Restart



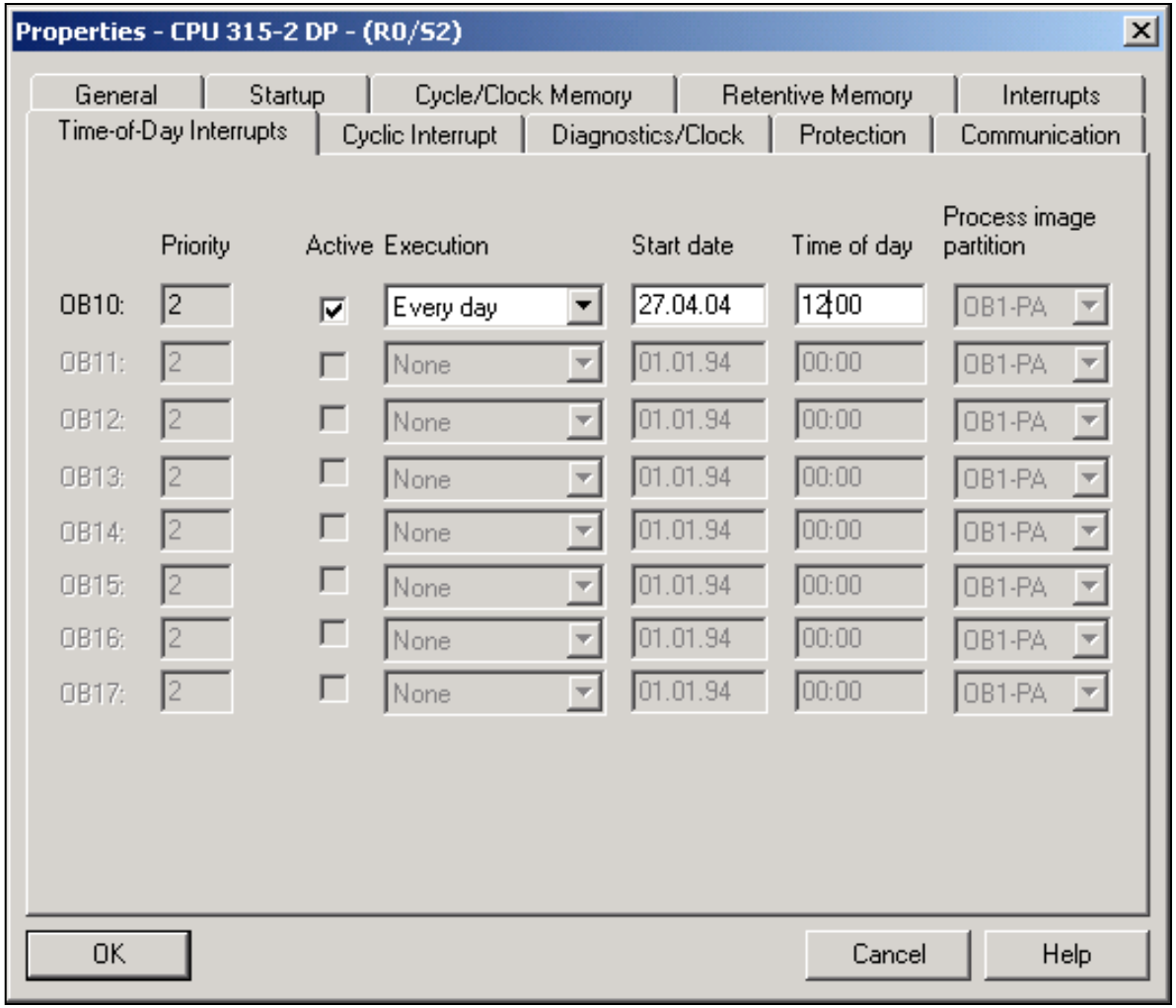
Interrupting the Cyclic Program



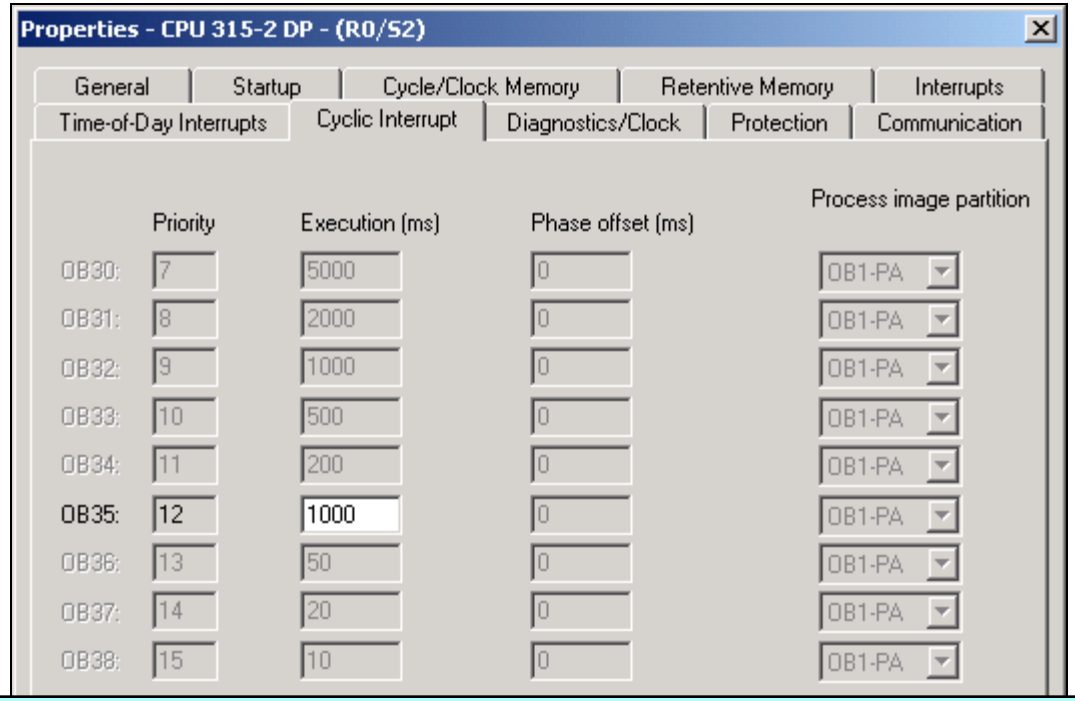
..... until it is interrupted by another OB

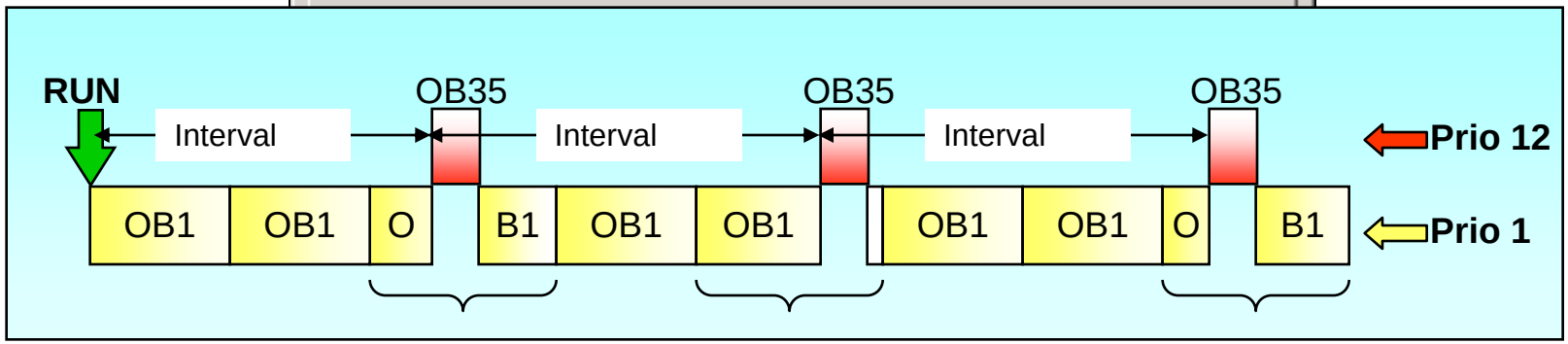
OB No.	OB Type	Priority
OB 1	Cyclic Program	1
OB 10	Time-of-day interrupt	2
OB 20	Time-delay interrupt	3
OB 35	Cyclic interrupt	12
OB 40	Hardware interrupt	16
OB 82	Error handling	25 / 28

Time-of-Day Interrupt (OB 10)



Cyclic Interrupt (OB 35)





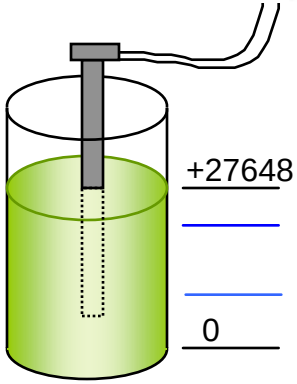
Hardware Interrupt (OB 40)

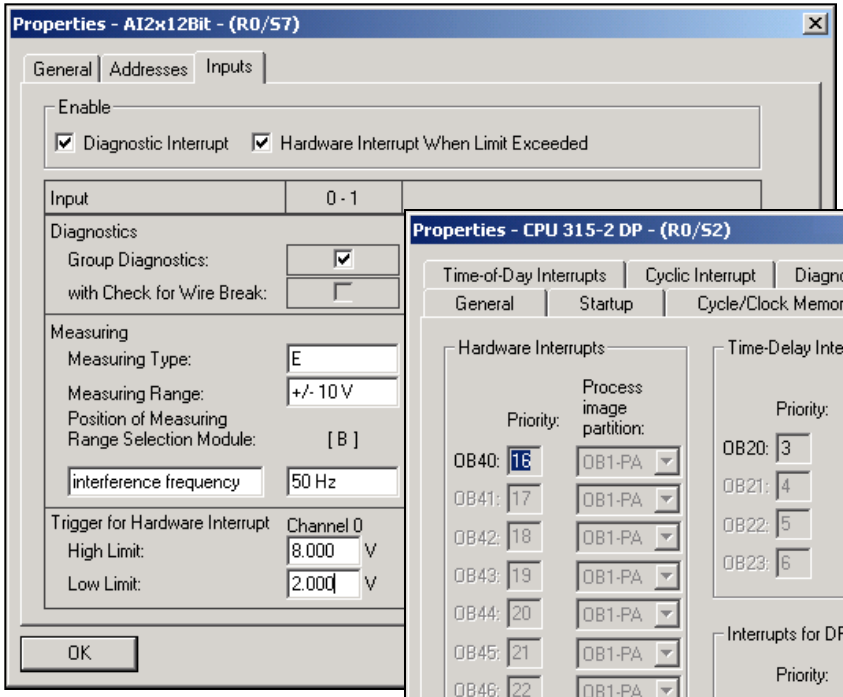
Hardware Configuration

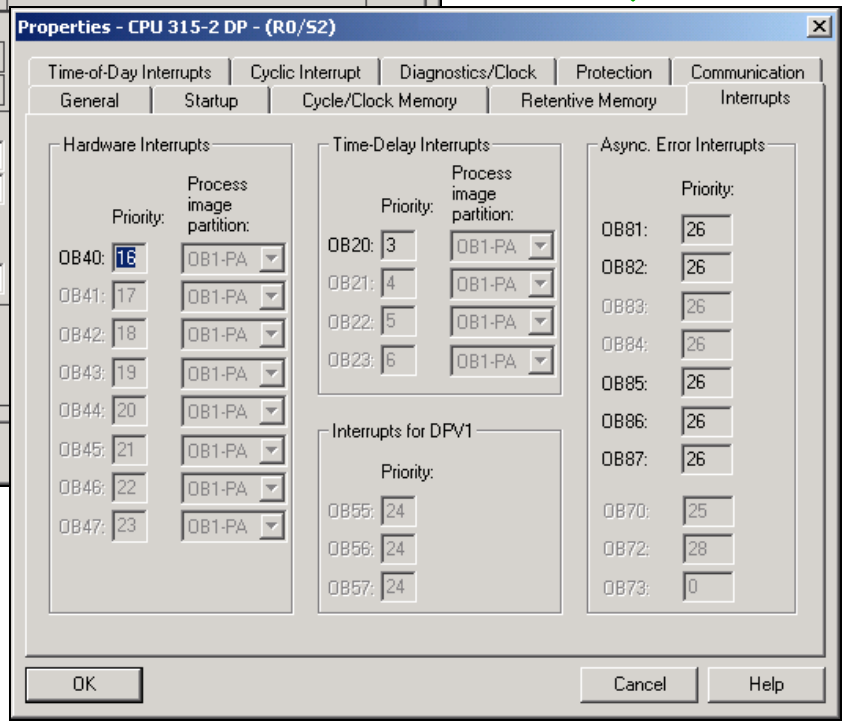
Analog input properties

CPU properties

Analog input module

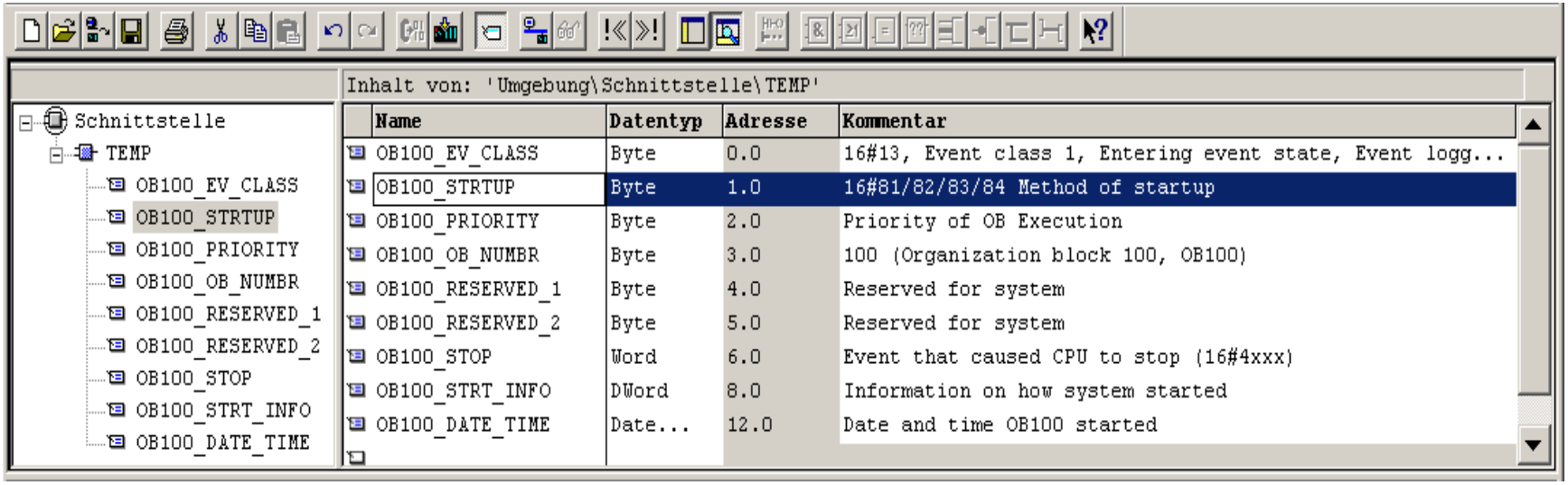




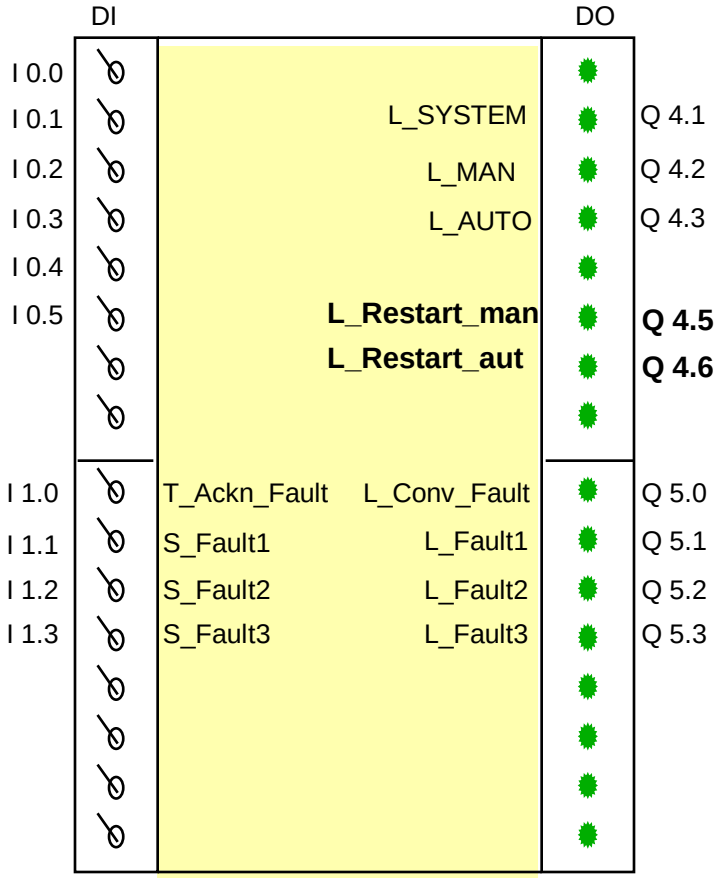


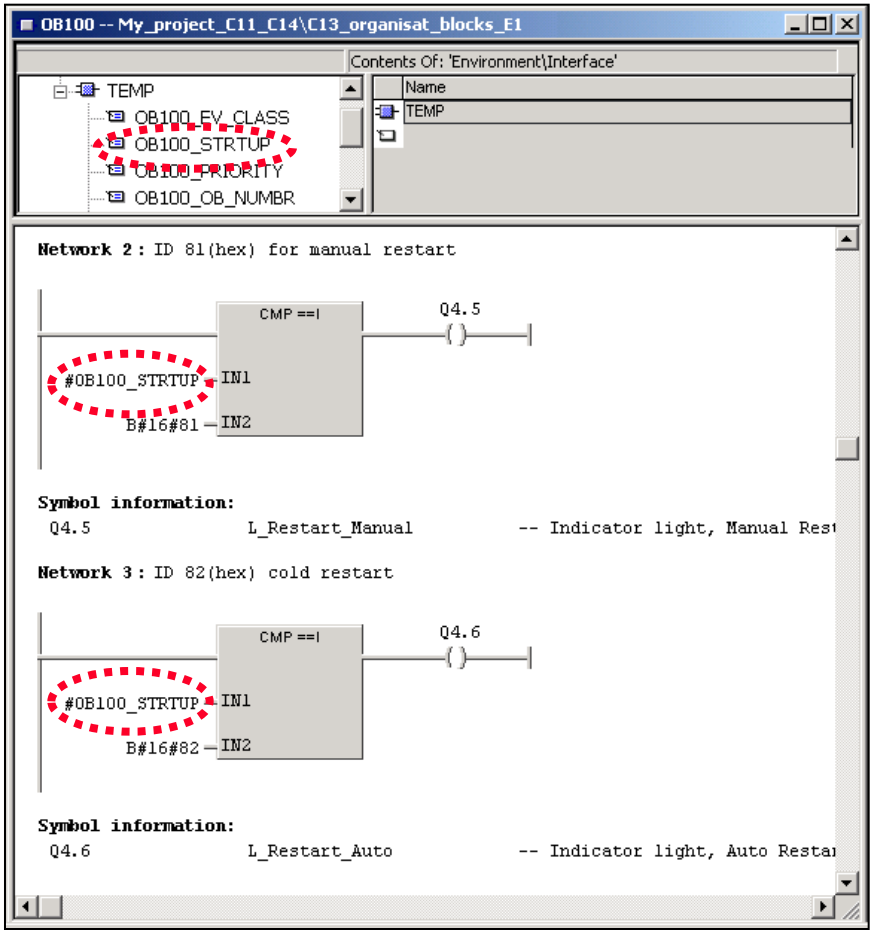
OB Start Information using OB100 as an Example

Address		
0 / 1	Start event	Start-up request
2 / 3	Priority	OB No.
4 / 5	Reserved	
6 / 7	Number of the event that caused the CPU to go into STOP	
8 / 9	Additional information on the current startup???	
10 / 11		
12 / 13	Year	Month
14 / 15	Day	Hours
16 / 17	Minutes	Seconds
18 / 19	1/10 seconds, 1/100 seconds	1 /1000 seconds, weekday



Ex. 1: Displaying the Startup Type (OB100) and Acknowledging It



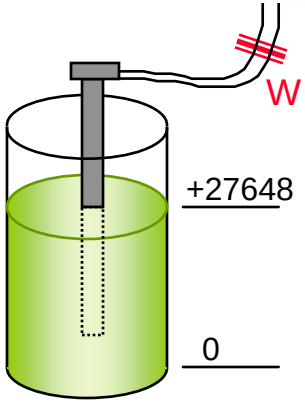


Asynchronous Errors

Type of error	Example	OB	Priority
Time error	Maximum scan cycle time exceeded	OB80	25
Power supply fault	Backup battery failure	OB81	
Diagnostic interrupt	Wirebreak at input of diagnostics-capable module	OB82	
Insert / remove interrupt	Removal of a signal module during operation of an S7-400™	OB83	
CPU hardware fault	Incorrect signal level at the MPI interface	OB84	
Program execution error	Error in updating the process image (module defective)	OB85	
Rack fault	Failure of an expansion device or a DP slave	OB86	
Communication error	Error in reading message frame	OB87	25 / 28

Diagnostic Interrupt (OB 82)

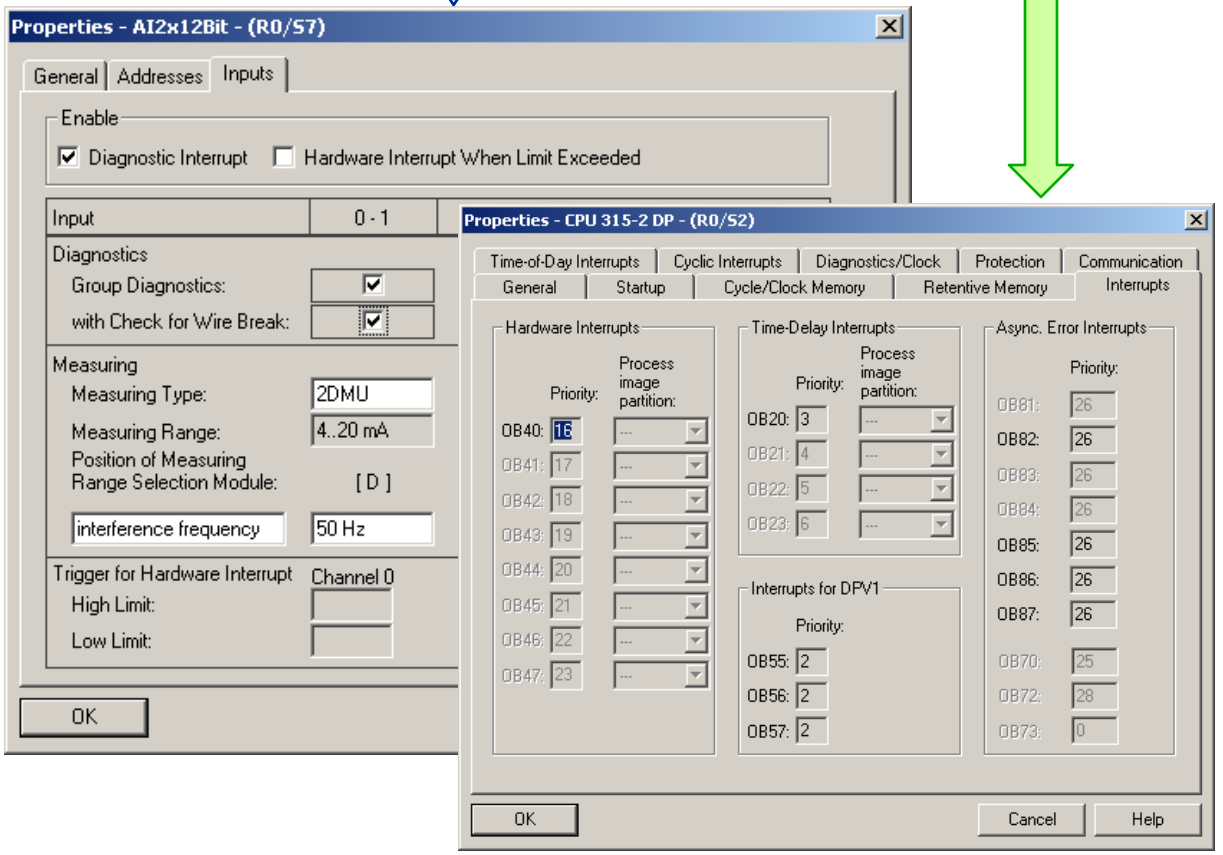
Analog input module



Hardware Configuration:

Analog input properties

CPU properties



Synchronous Errors

Type of Error	Example	OB	Priority
Programming error	A block that is not present in the CPU is called in the program	OB121	Same as that of the OB interrupted as a result of the error
Access error	A module that is either defective or not present is addressed in the program (such as direct access to a non-existent I/O module)	OB122	

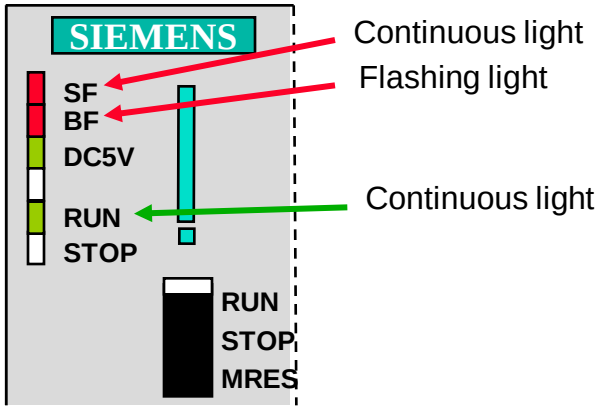
Exercise 2: Response when a Slave Fails

PROFIBUS slave failed

Asynchronous error

CPU response...

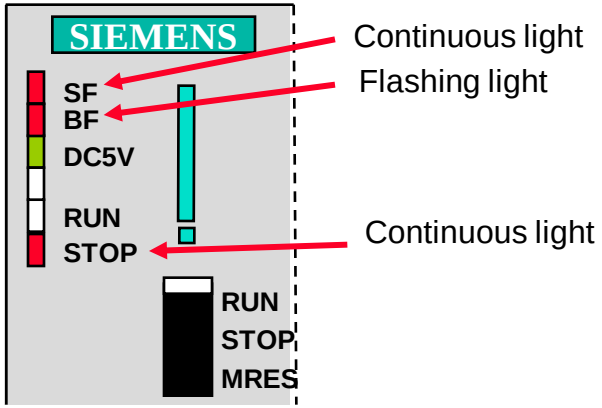
...with OB 86



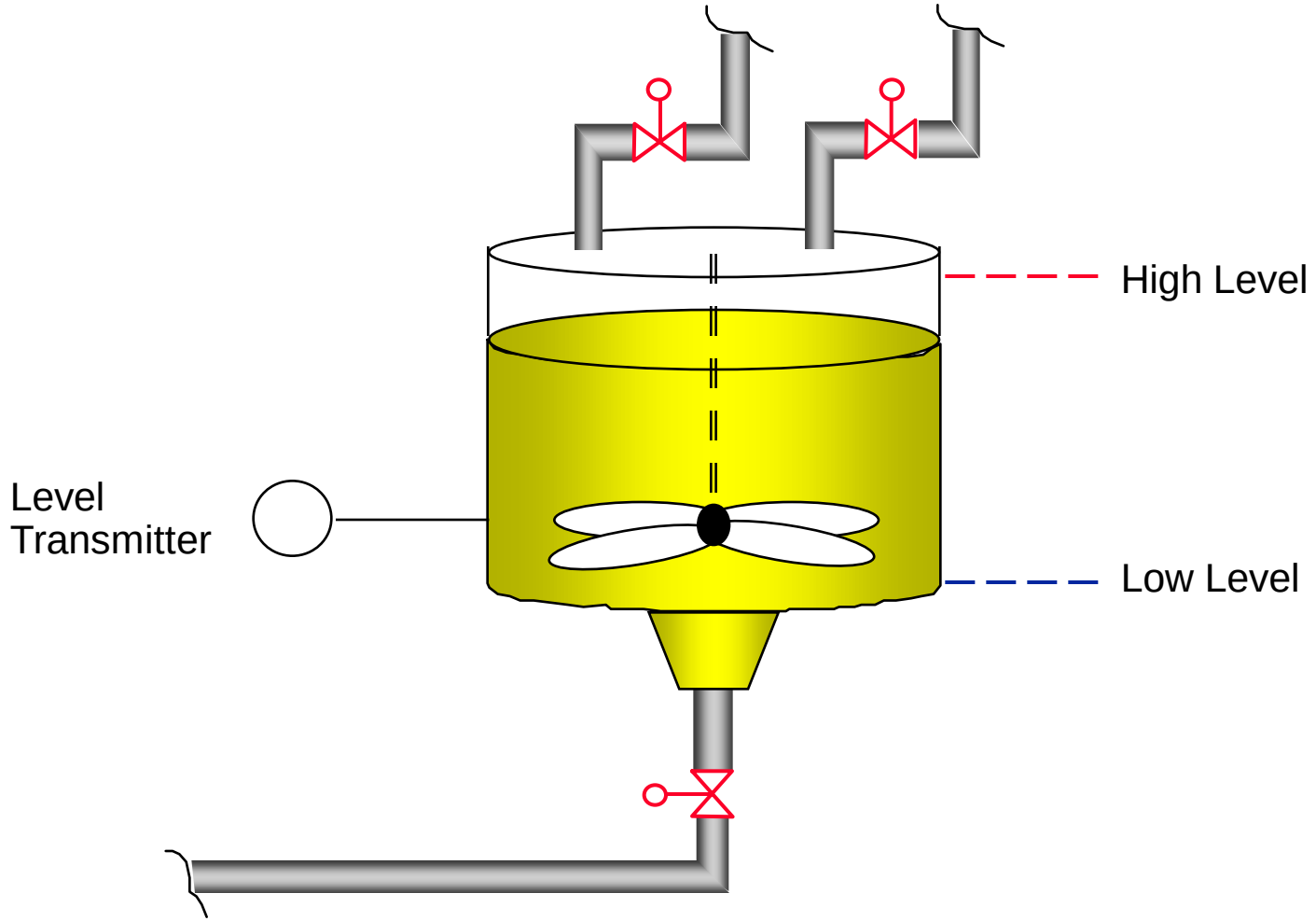
PROFIBUS slave failed

Asynchronous error

...without OB 86



Analog Value Processing



Objectives

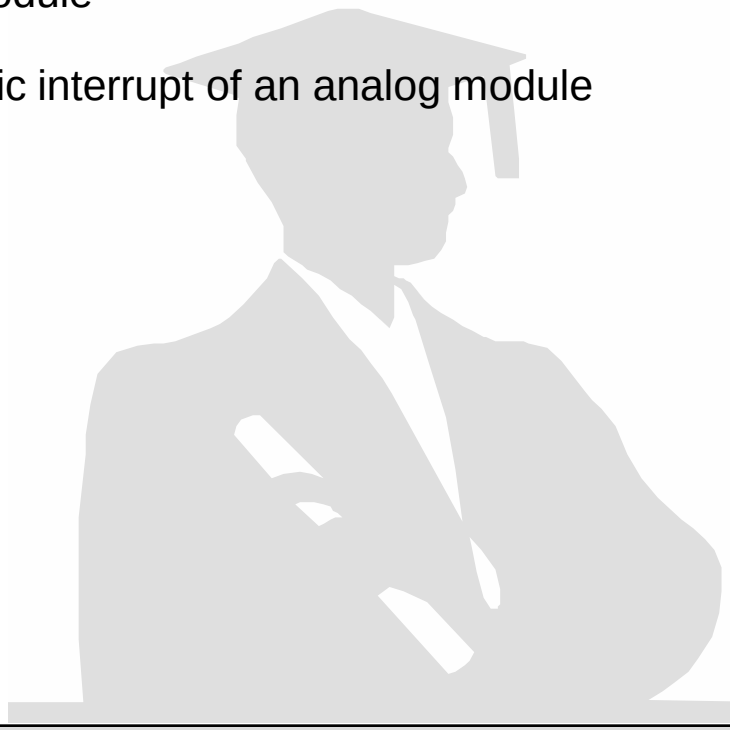
Upon completion of the module the participant will ...

... know the principle of analog value processing

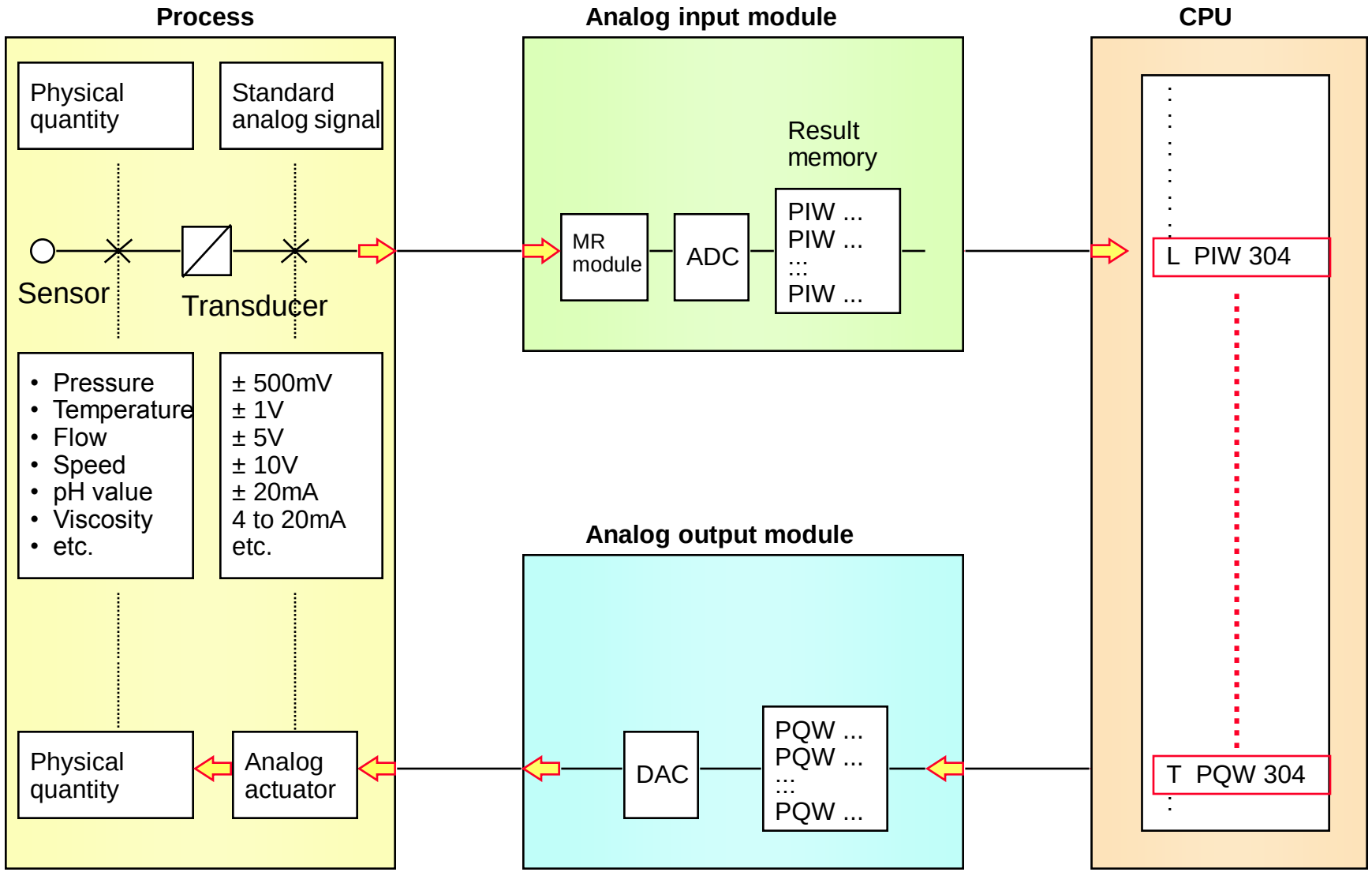
... be able to assign parameters to an analog module

... be able to address an analog module

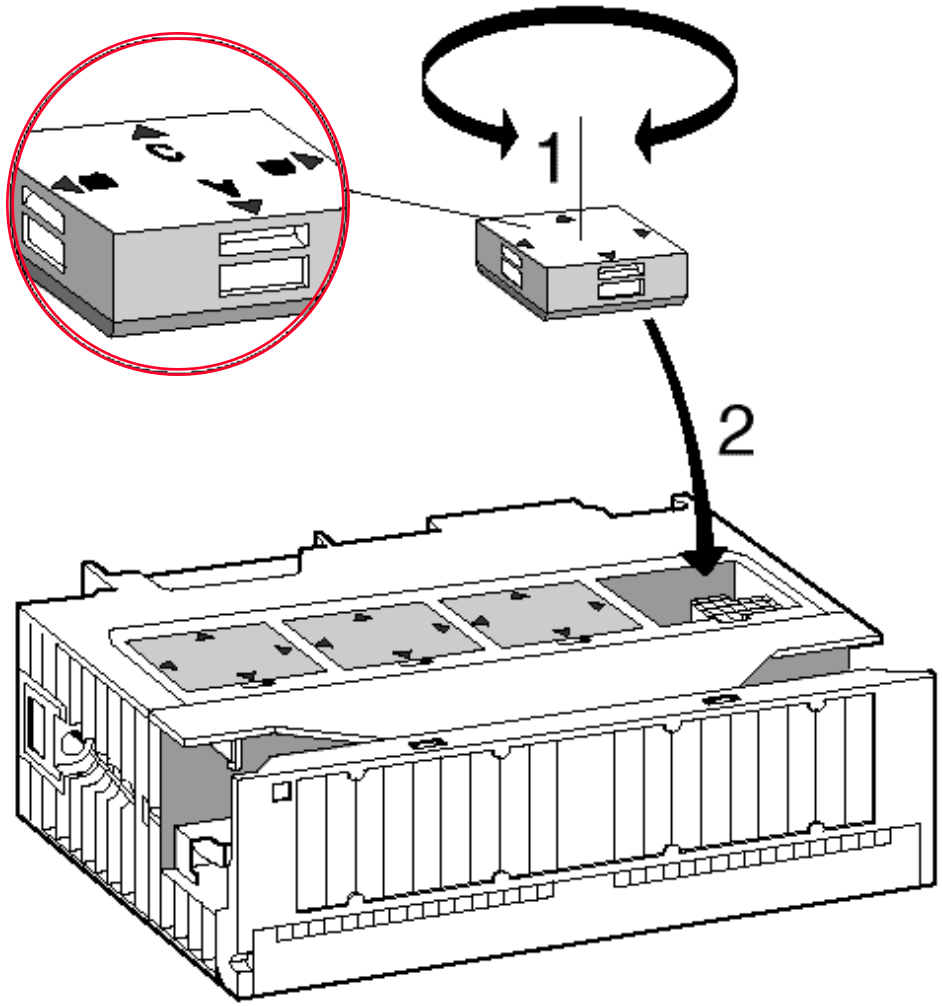
... be able to evaluate the diagnostic interrupt of an analog module



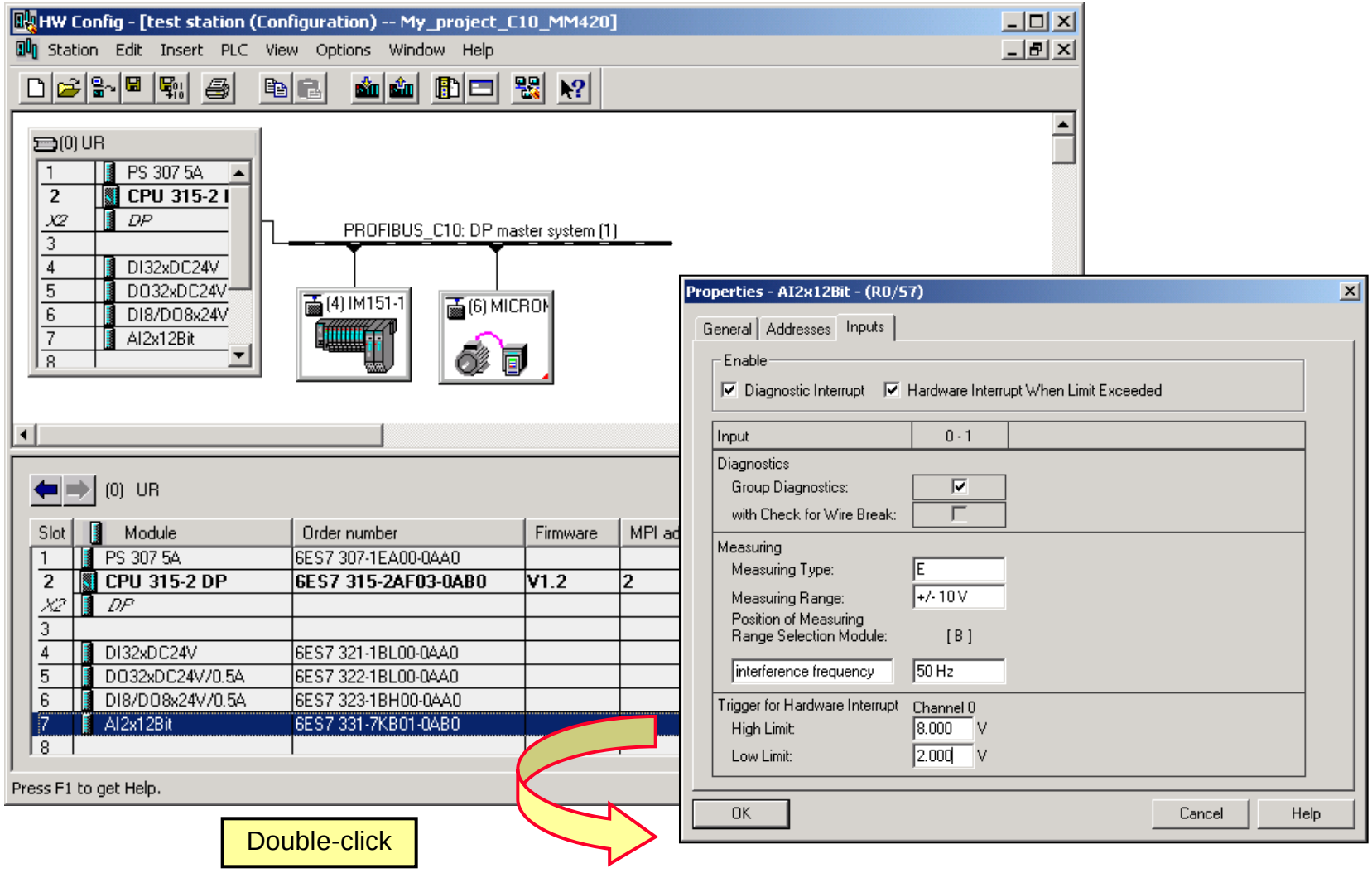
Using Analog Modules



Measuring Range Modules

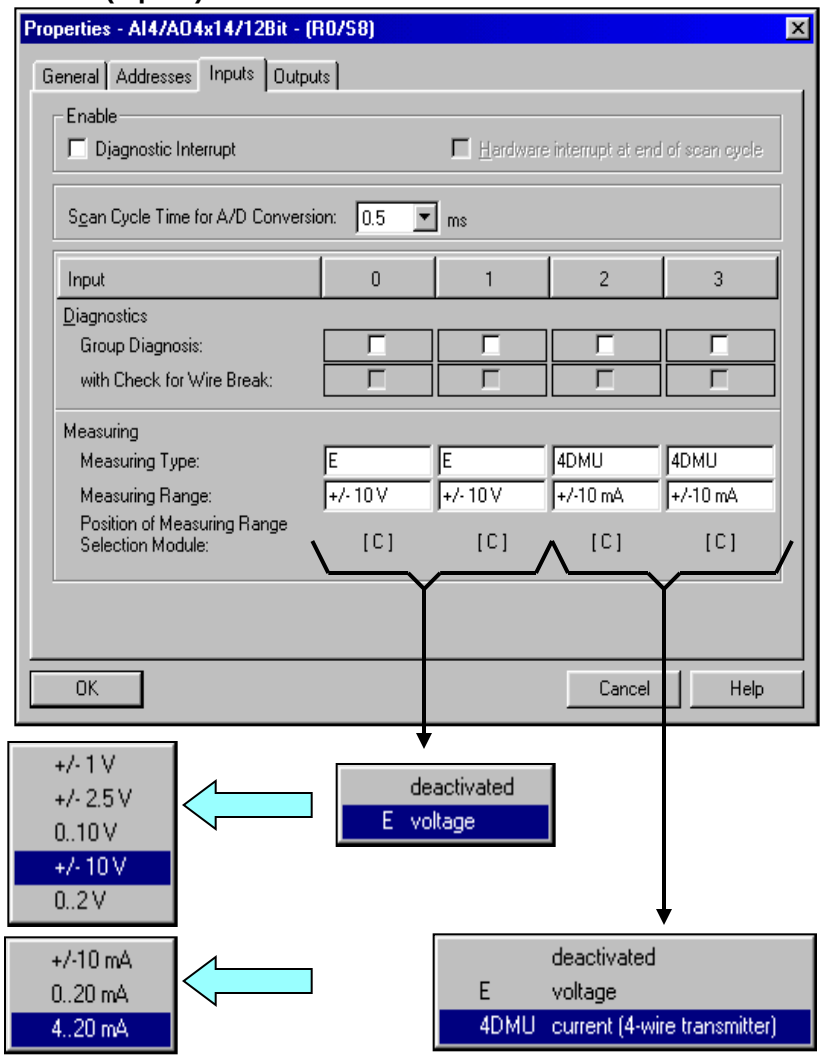


Analog Module Properties

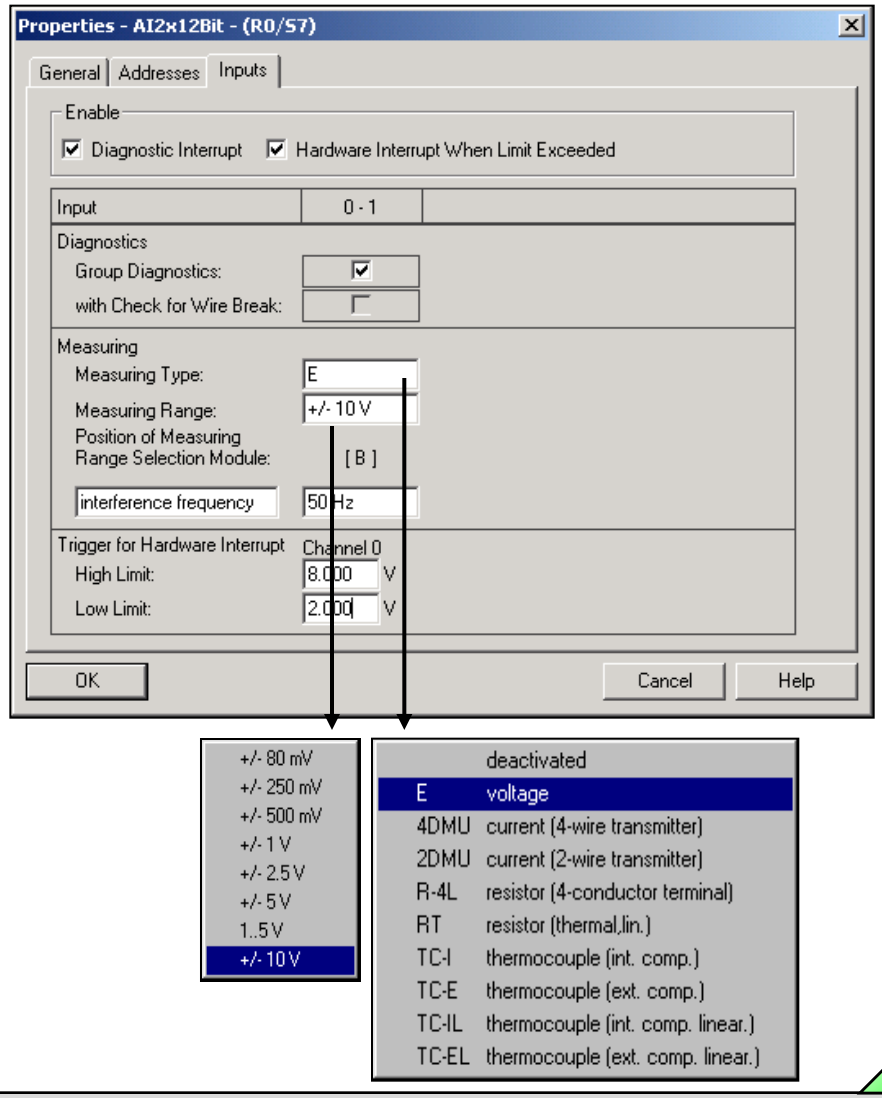


Analog Input Modules

SM335 (Inputs)



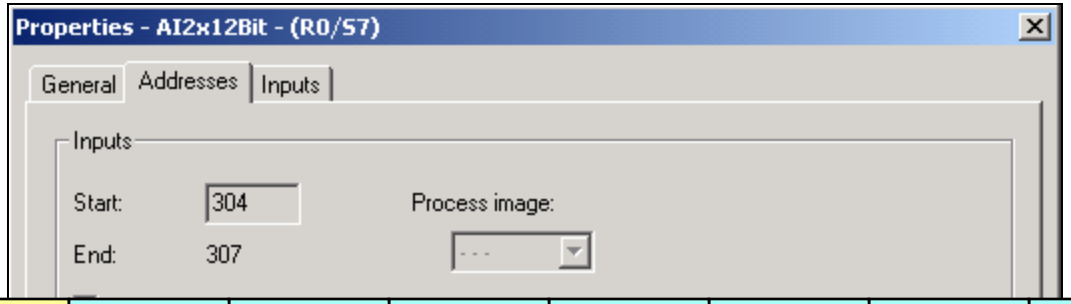
SM331



Analog Value Representation of Different Measuring Ranges

Range	Voltage such as:		Current such as:		Resistance such as:		Temperature e.g. Pt100 (Standard)	
	Meas.range ± 10V	Units	Meas.range 4 to 20mA	Units	Meas.range 0 ...300Ohm	Units	Meas.range -200...+850ºC	Units
Overflow	>= 11.759	32767	>= 22.815	32767	>=352.778	32767	>= 1000.1	32767
Overrange	11.7589 : 10.0004	32511 : 27649	22.810 : 20.0005	32511 : 27649	352.767 : 300.011	32511 : 27649	1000.0 : 850.1	10000 : 8501
Rated range	10.00 7.50 : -7.5 -10.00	27648 20736 : -20736 -27648	20.000 16.000 : : 4.000	27648 20736 : : 0	300.000 225.000 : : 0.000	27648 20736 : : 0	850.0 : : : -200.0	8500 : : : -2000
Underrange	- 10.0004 : - 11.759	- 27649 : - 32512	3.9995 : 1.1852	- 1 : - 4864	Negative values not possible	- 1 : - 4864	- 200.1 : - 243.0	- 2001 : - 2430
Underflow	<= - 11.76	- 32768	<= 1.1845	- 32768		- 32768	<= - 243.1	- 32768

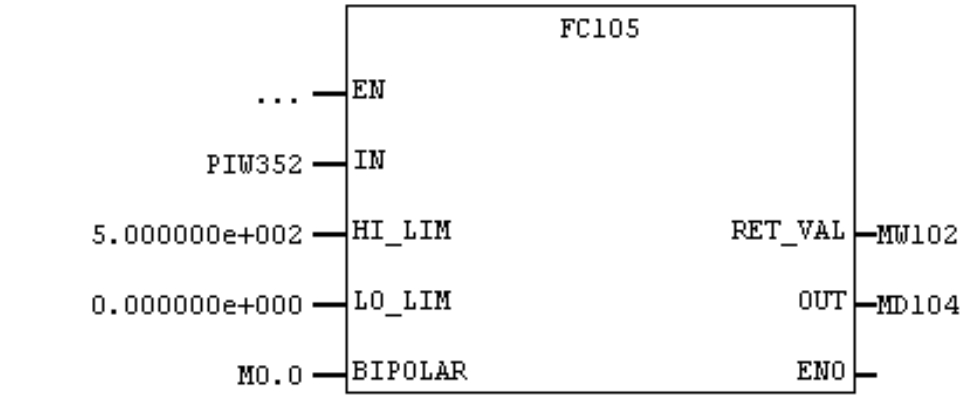
Analog Module Addresses for the S7-300™



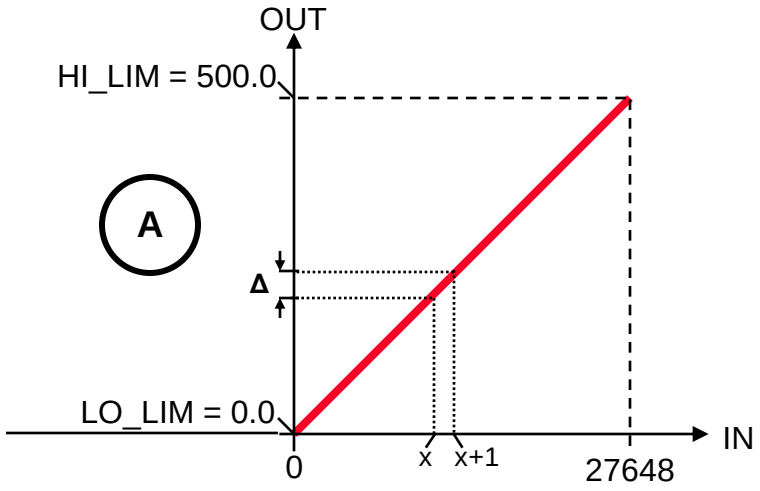
	Slot 2	Slot 3	Slot 4	Slot 5	Slot 6	Slot 7	Slot 8	Slot 9	Slot 10	Slot 11
Rack 3	Power Supply	IM (Receive)	640 to 654	656 to 670	672 to 686	688 to 702	704 to 718	720 to 734	736 to 750	752 to 766
Rack 2	Power Supply	IM (Receive)	512 to 526	528 to 542	544 to 558	560 to 574	576 to 590	592 to 606	608 to 622	624 to 638
Rack 1	Power Supply	IM (Receive)	384 to 398	400 to 414	416 to 430	432 to 446	448 to 462	464 to 478	480 to 494	496 to 510
R 0	Power Supply	CPU	IM (Send)	256 to 270	272 to 286	288 to 302	304 to 318	320 to 334	336 to 350	352 to 366	368 to 382

Scaling Analog Input Values

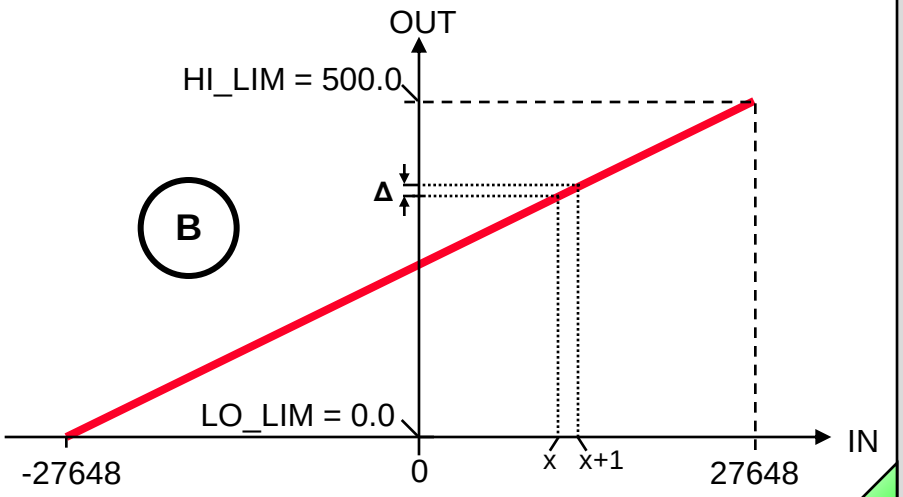
Network 5: Scale Analog Input Value



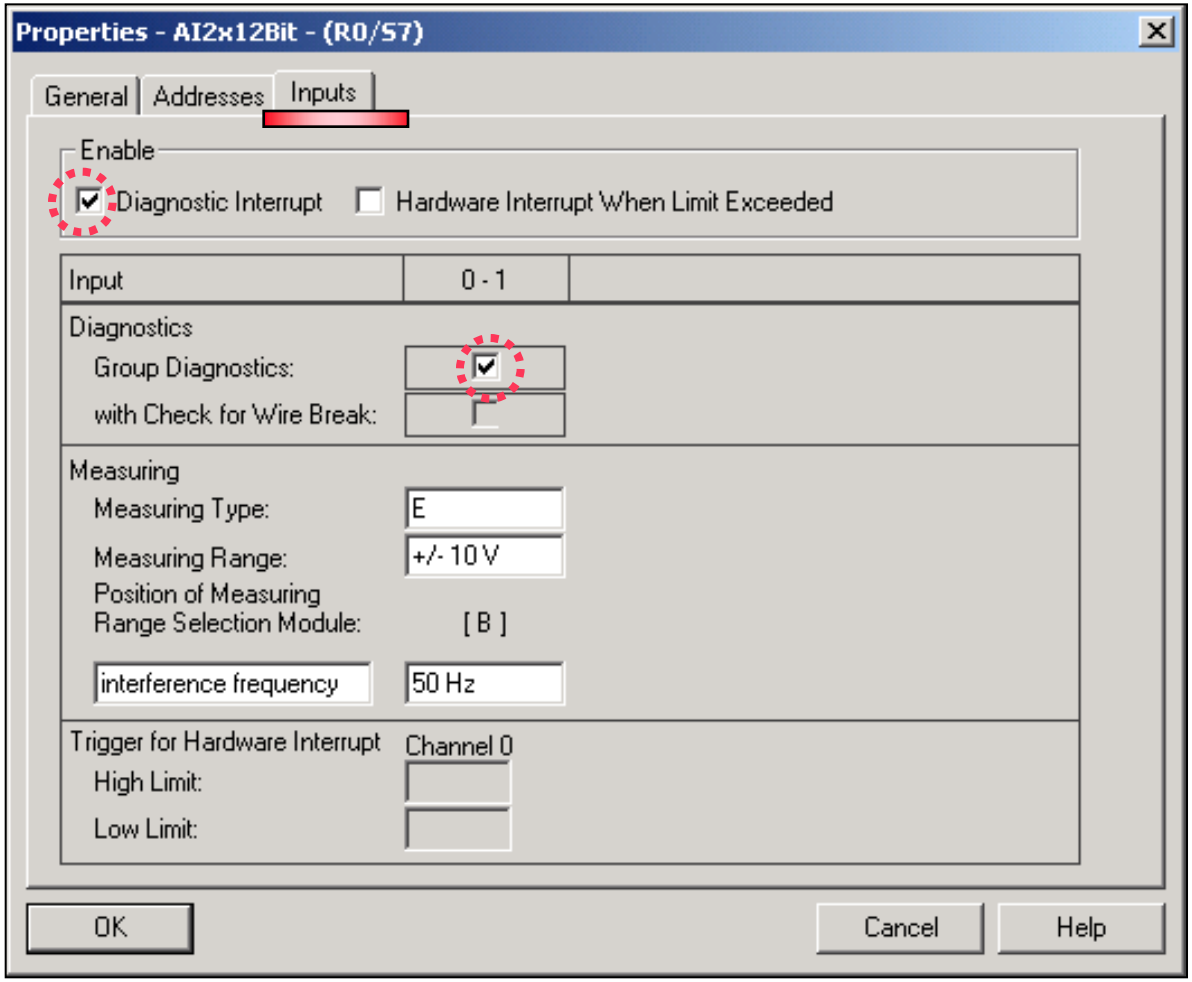
unipolar (M 0.0 = ´0´)
(Sensor supplies only positive voltage)



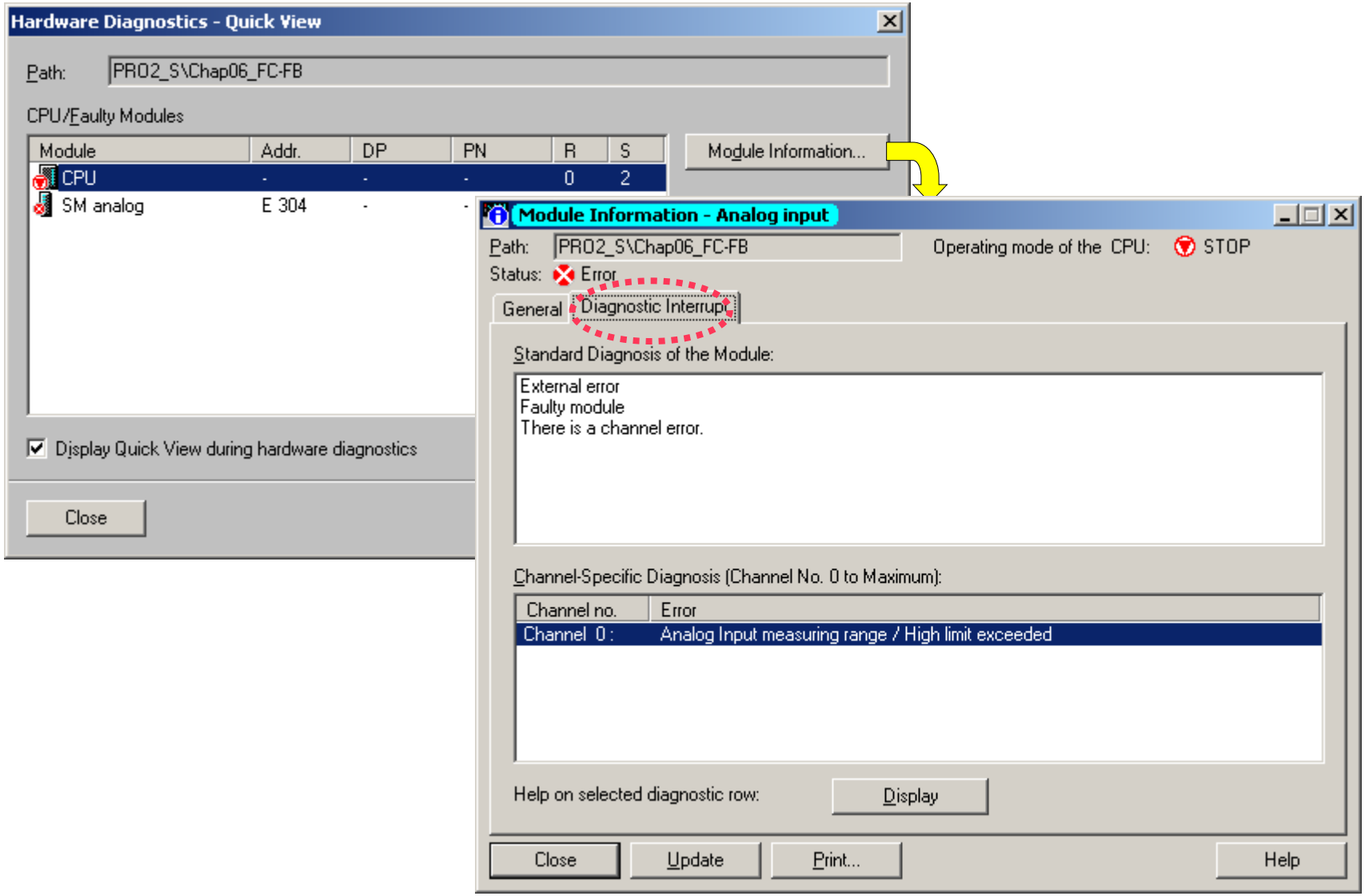
bipolar (M 0.0 = ´1´)
(Sensor also supplies negative voltage)



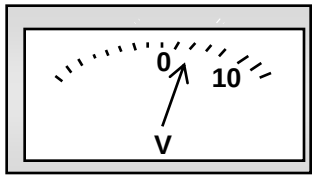
Exercise 1: Assigning Parameters to an Analog Module

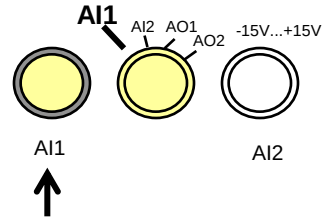


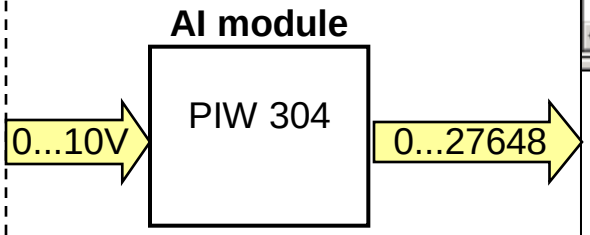
Exercise 2: Hardware Diagnosis with Diagnostic Interrupt

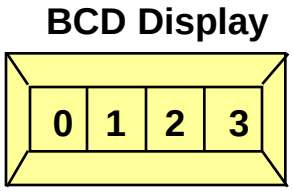


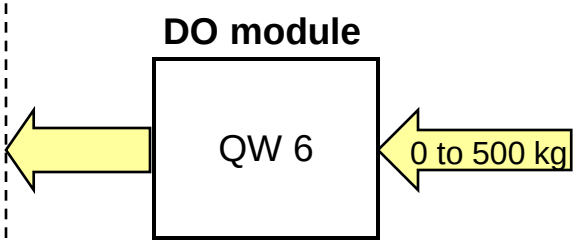
Exercise 3: Scaling an Analog Input (OB35)

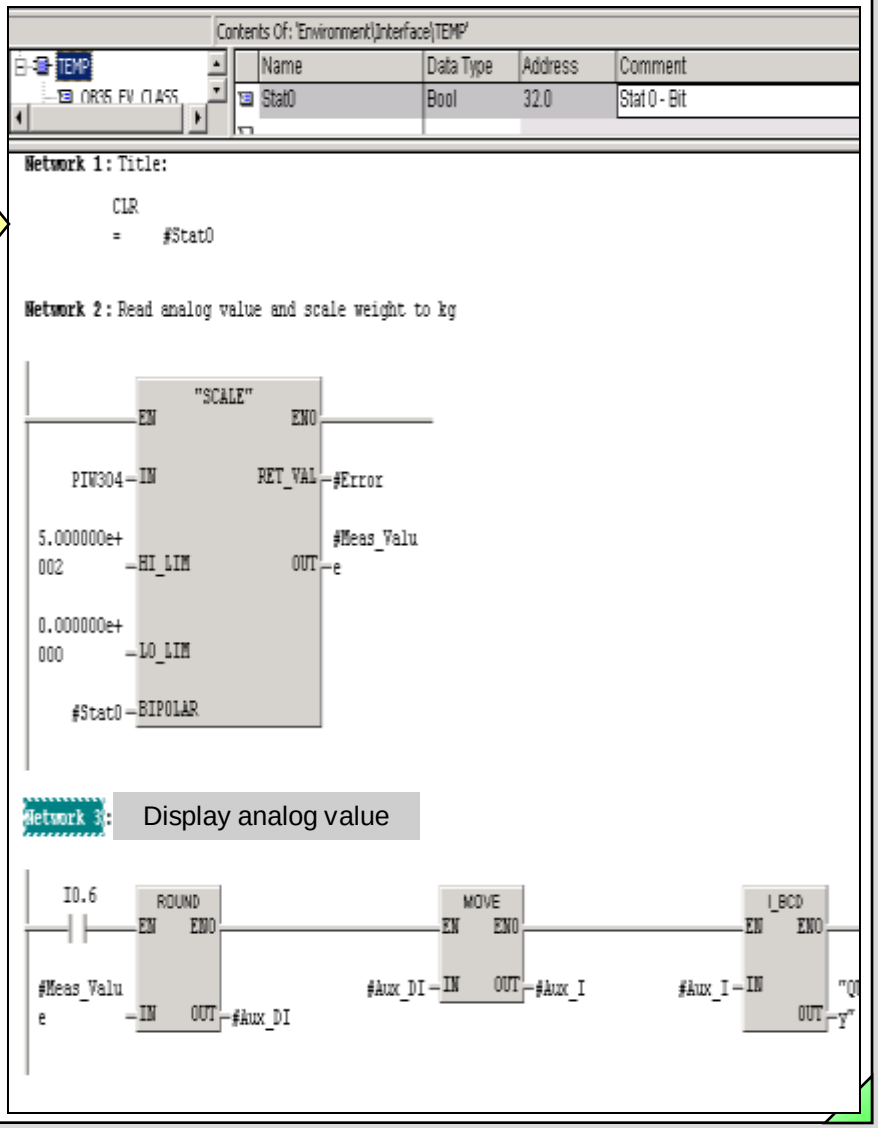






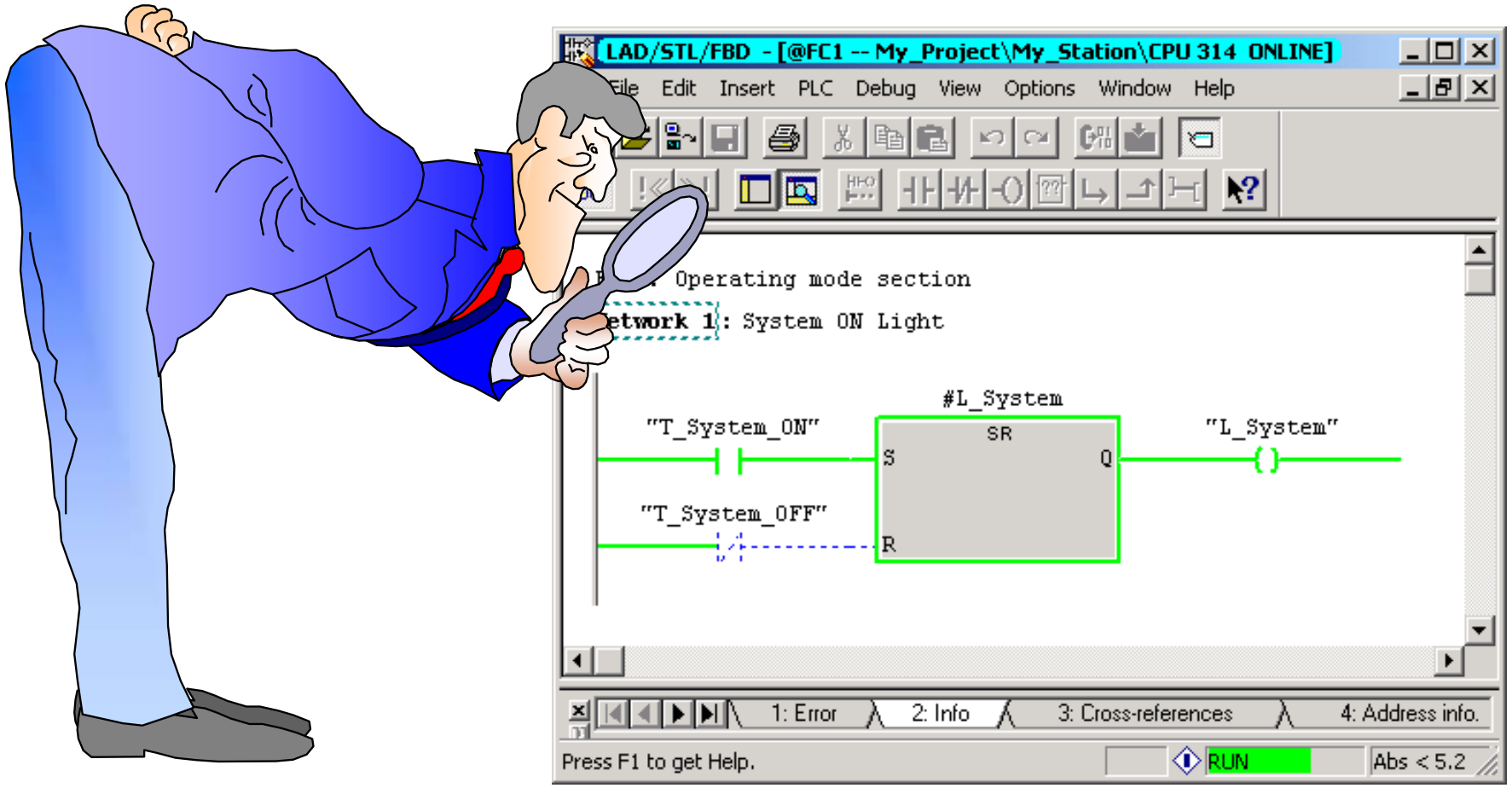




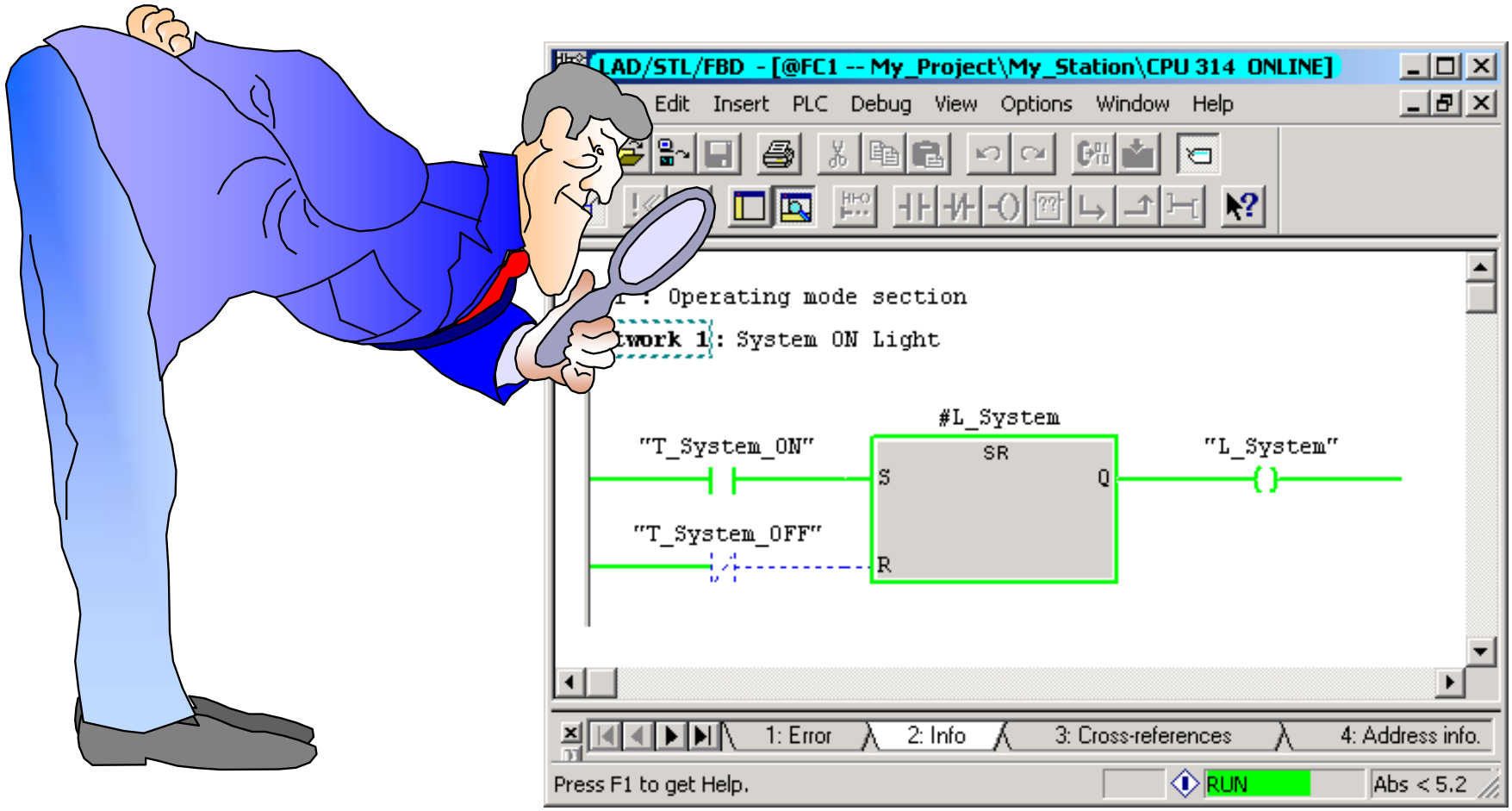


Date: 2/23/2012
File: PRO1_13E.12

Troubleshooting



Troubleshooting



Objectives

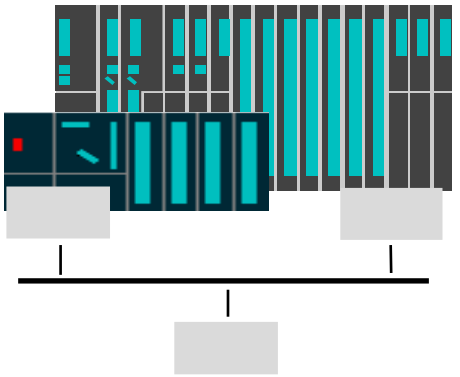
Upon completion of the chapter the participant will ...

- ... be able to classify occurring errors as "Errors detected by the system" and "Functional errors"
- ... be familiar with the "Displaying CPU Messages" function
- ... be able to read out the diagnostic buffer, interpret it and use it for troubleshooting
- ... be able to read out the I STACK, B STACK and L STACK and interpret them
- ... be able to read out the hardware diagnosis
- ... be able to apply the "Monitor/Modify Variables" test function
- ... be able to interpret the displays of the "Monitor" test function in the LAD/STL/FBD Editor and use them for troubleshooting
- ... be able to read out the reference data, interpret them and use them for troubleshooting
- ... understand the "Force" function

Categories of Errors

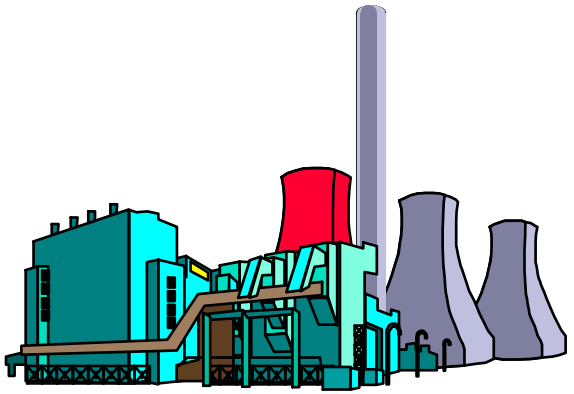
Errors Detected by the System

- Recording, evaluating and indicating errors within a PLC (as a rule: CPU STOP)
  - Module failure
  - Short-circuit in signal cables
  - Scan time overrun
  - Programming error (accessing a non-existent block)



Functional Errors

- Desired function is either not executed at all or is not correctly executed
  - Process fault (sensor/actuator, cable defective)
  - Logical programming error (not detected during creation and startup)



STEP7 - Debugging Tools, Overview

Error

Error detected by the System:
General Rule: CPU in STOP
(such as accessing a non-existent DB)
(Diagnostic interrupt of a signal module)

Debugging Tools:

- **Module Information**
  - Diagnostic buffer
  - I STACK
  - B STACK
  - L STACK
- **Hardware Diagnostics**

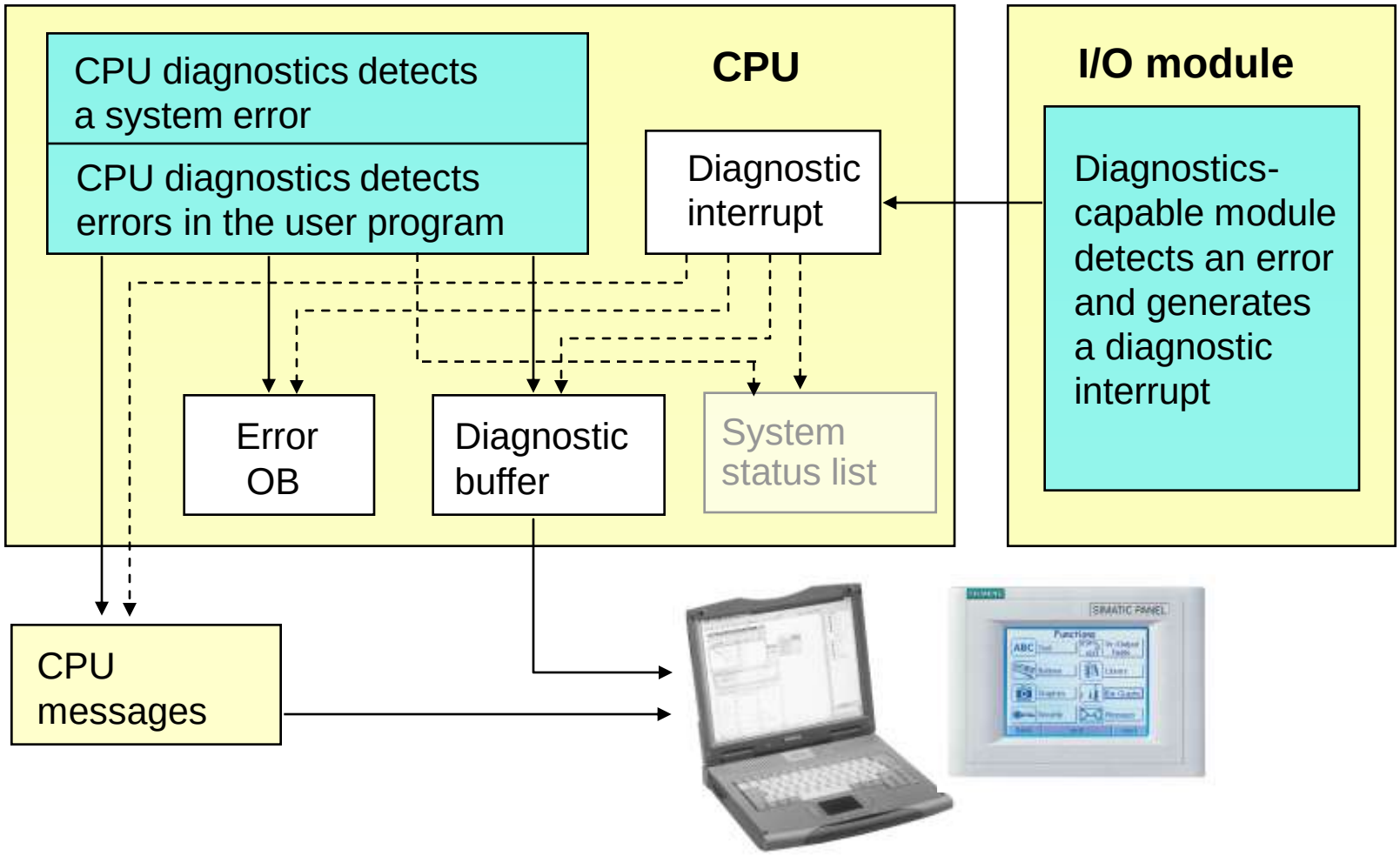
Functional fault:
General Rule: CPU in RUN
(process fault, such as a wire break)
(logical programming error, like a double assignment)

Debugging Tools:

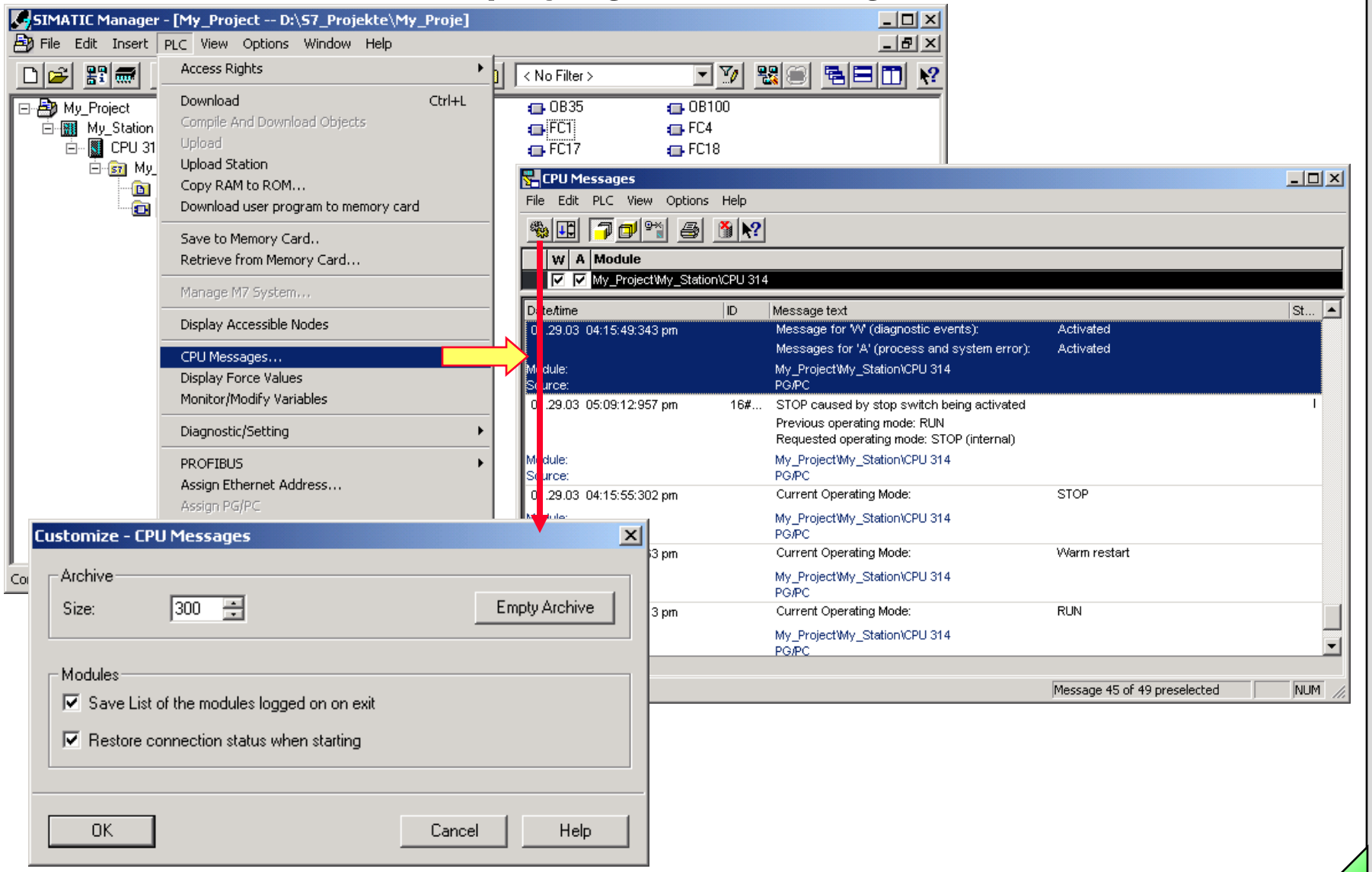
- **Enable Peripheral Outputs (modify outputs)**
- **Monitor / Modify Variable**
- **Monitor Blocks (Block Status)**
- **Reference Data**
  - Cross References
  - Assignment of I/Q/M/T/C
  - Program Structure
  - Addresses without Symbols
  - Unused Symbols

Regardless of the cause of the error you could use:
- **Force**
- **Enable Peripheral Outputs**
- **Set Breakpoints**

System Diagnostics - Overview

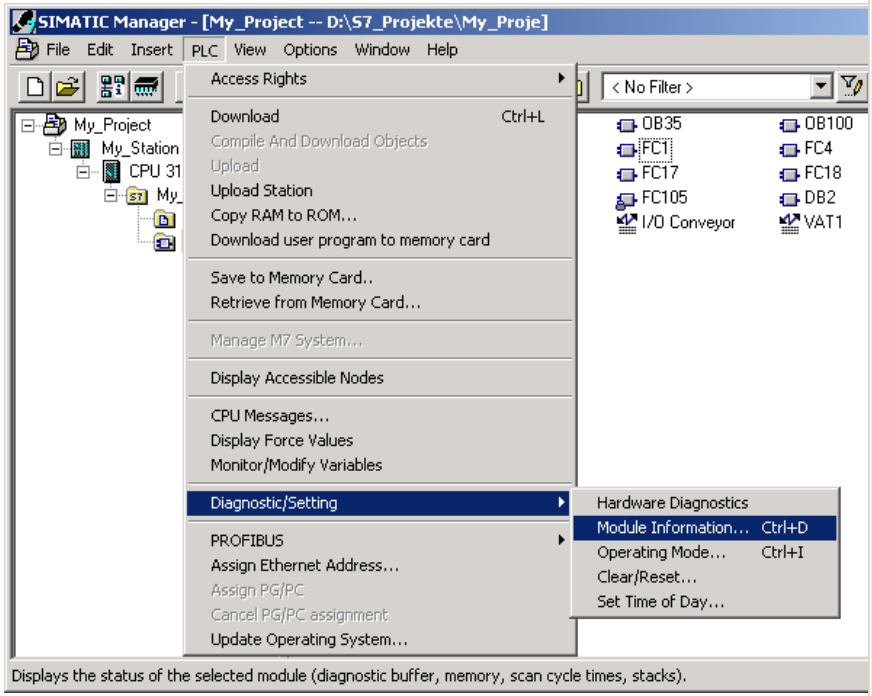


Displaying CPU Messages

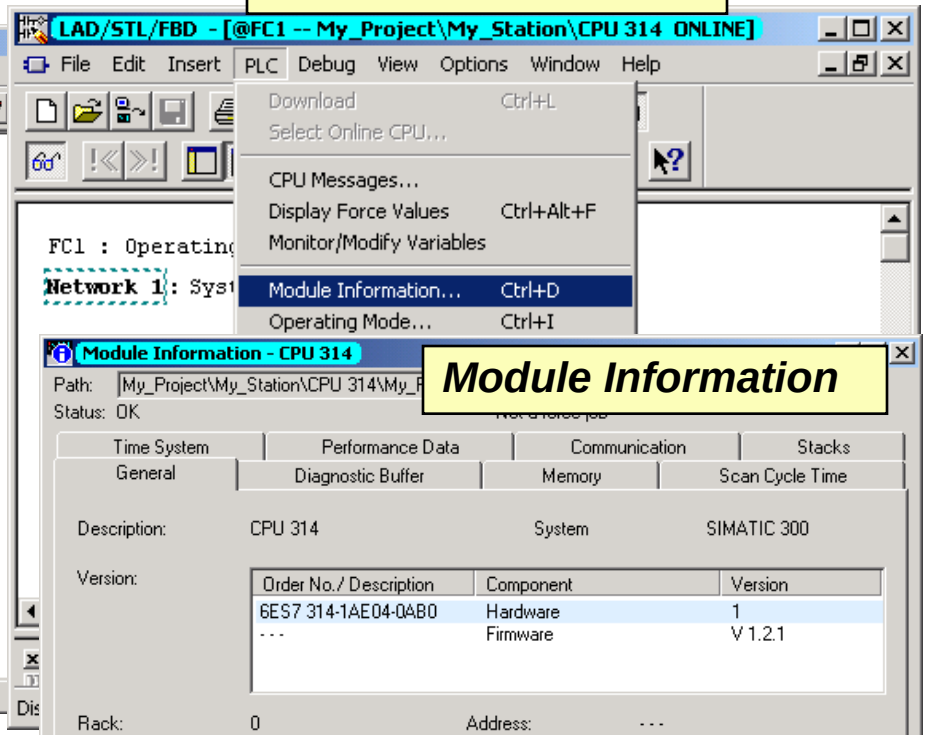


Calling the "Module Information" Tool

SIMATIC® Manager

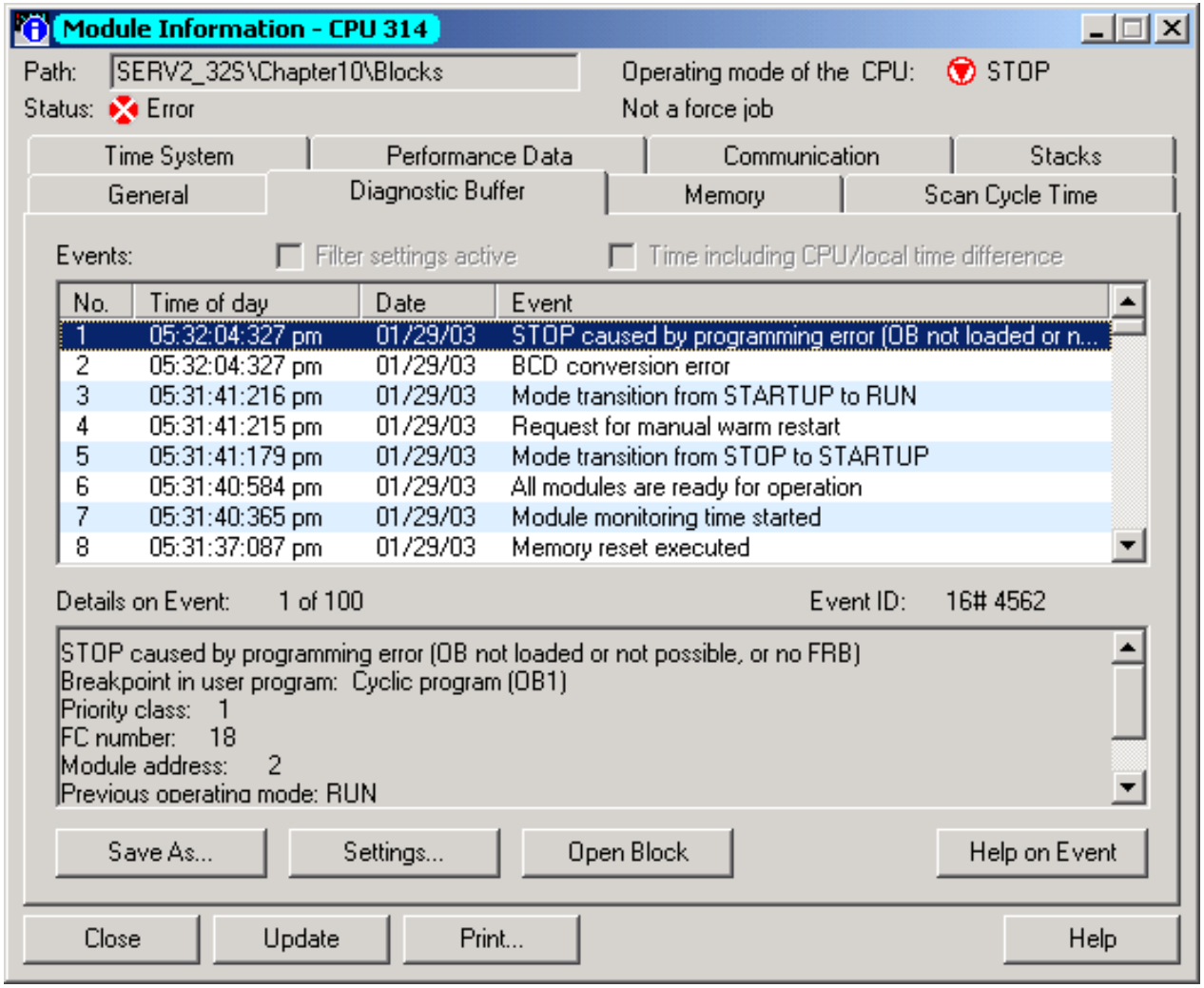


LAD/STL/FBD Editor

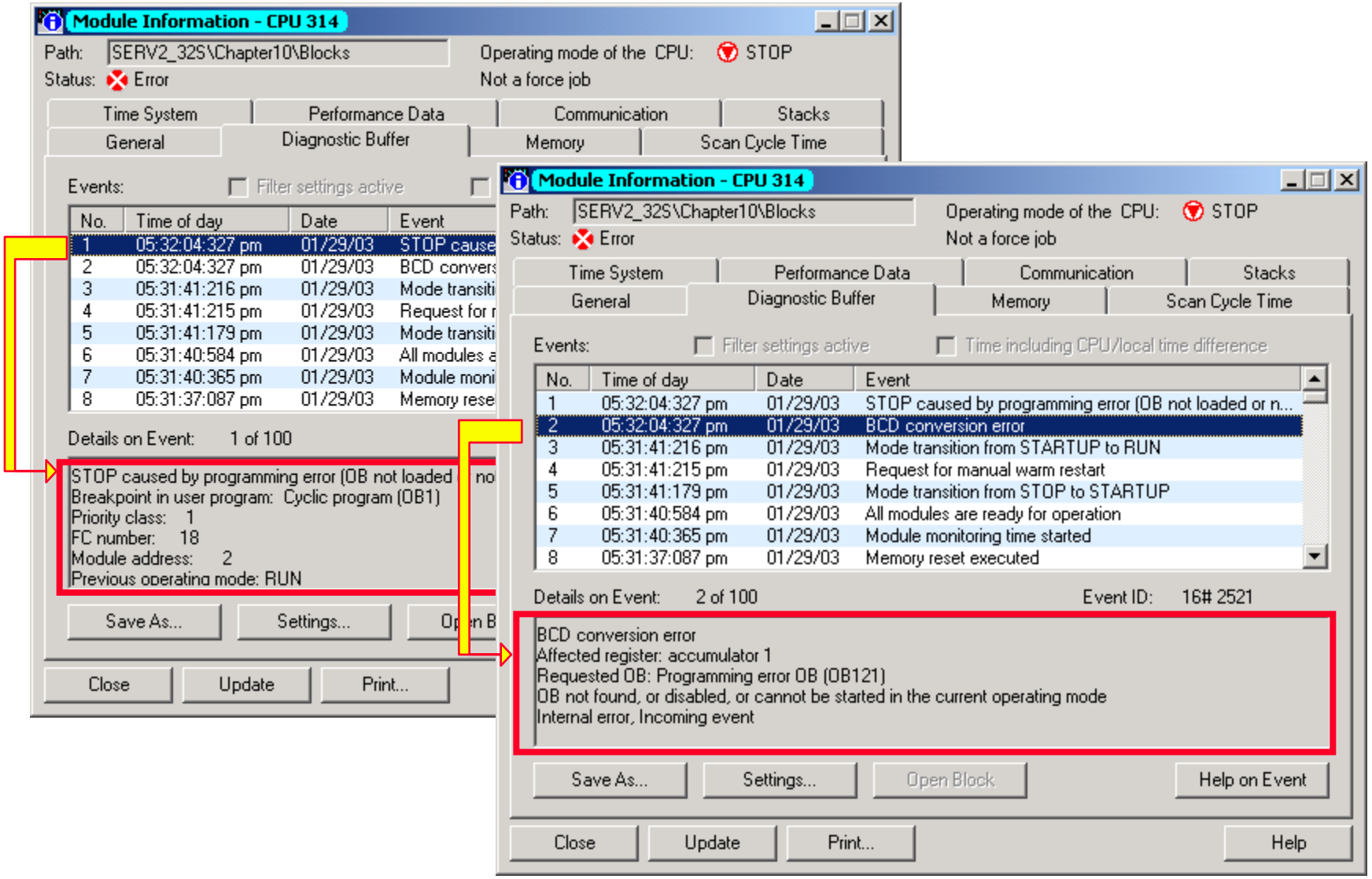


Module Information

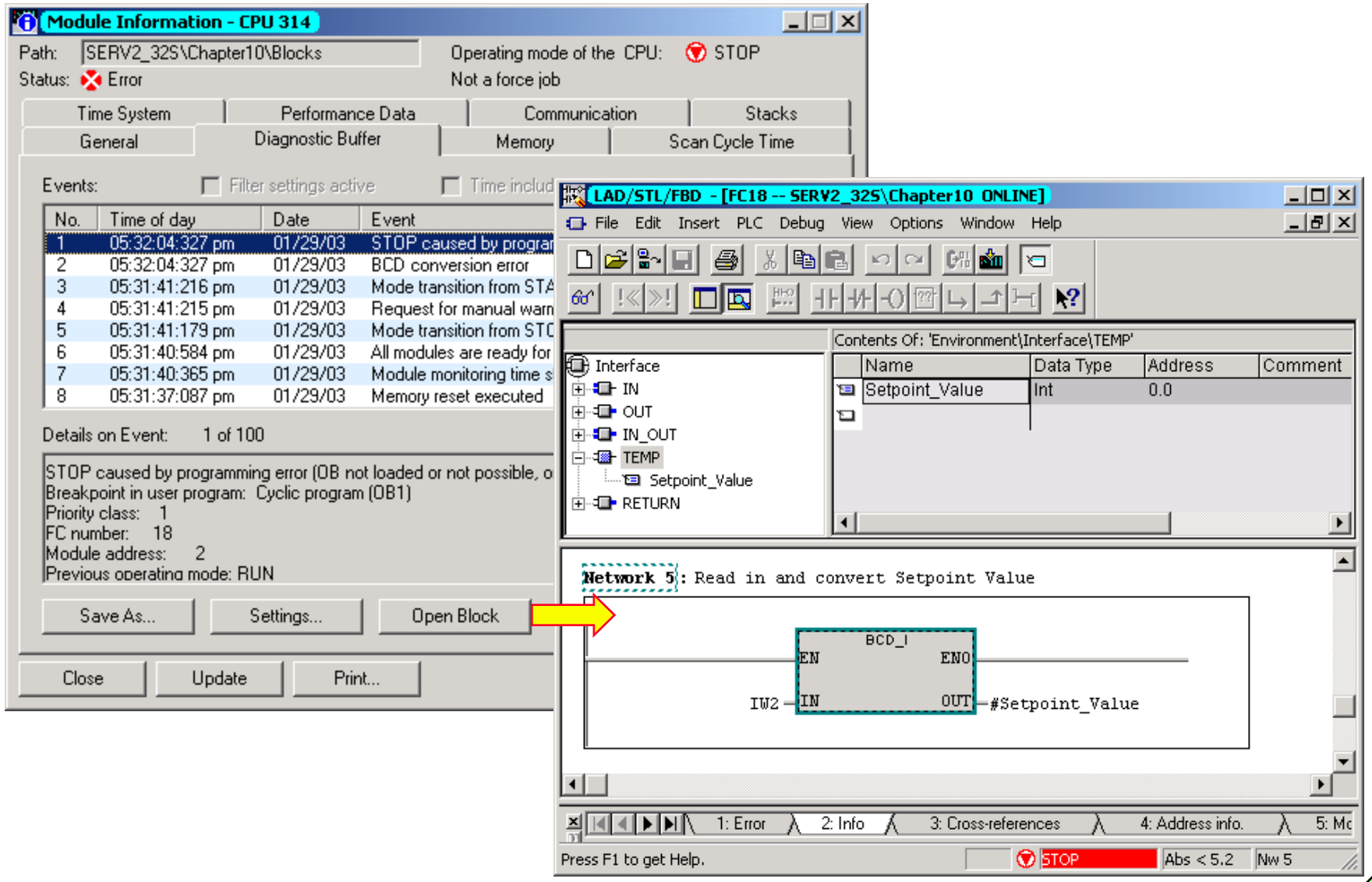
Module Information Tab: "Diagnostic Buffer"



Interpreting Error Messages in the Diagnostic Buffer



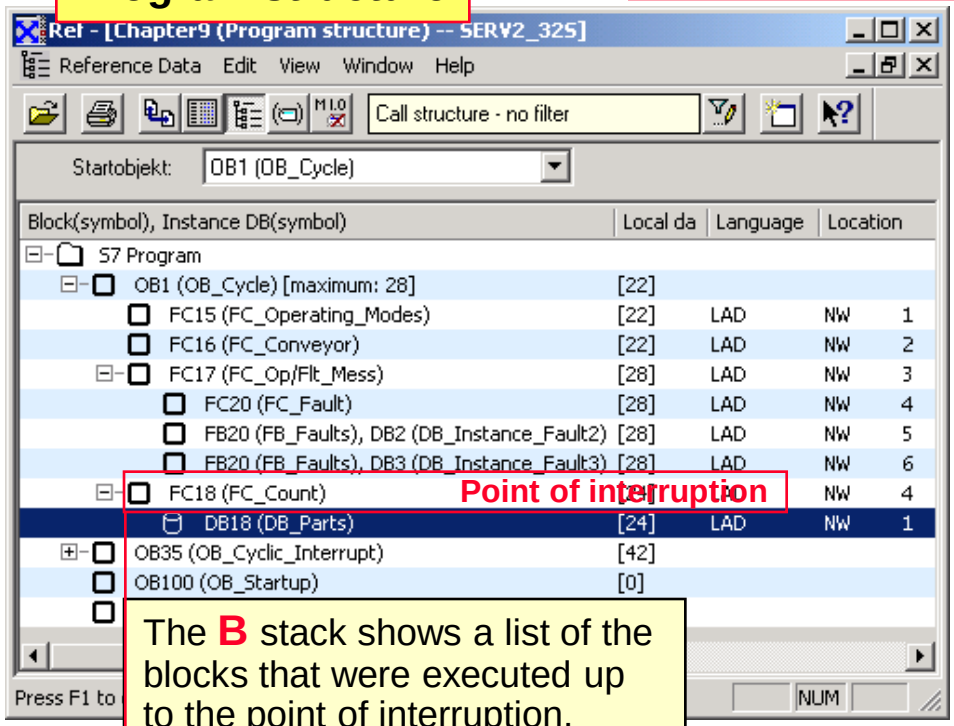
Opening a Block Containing an Error



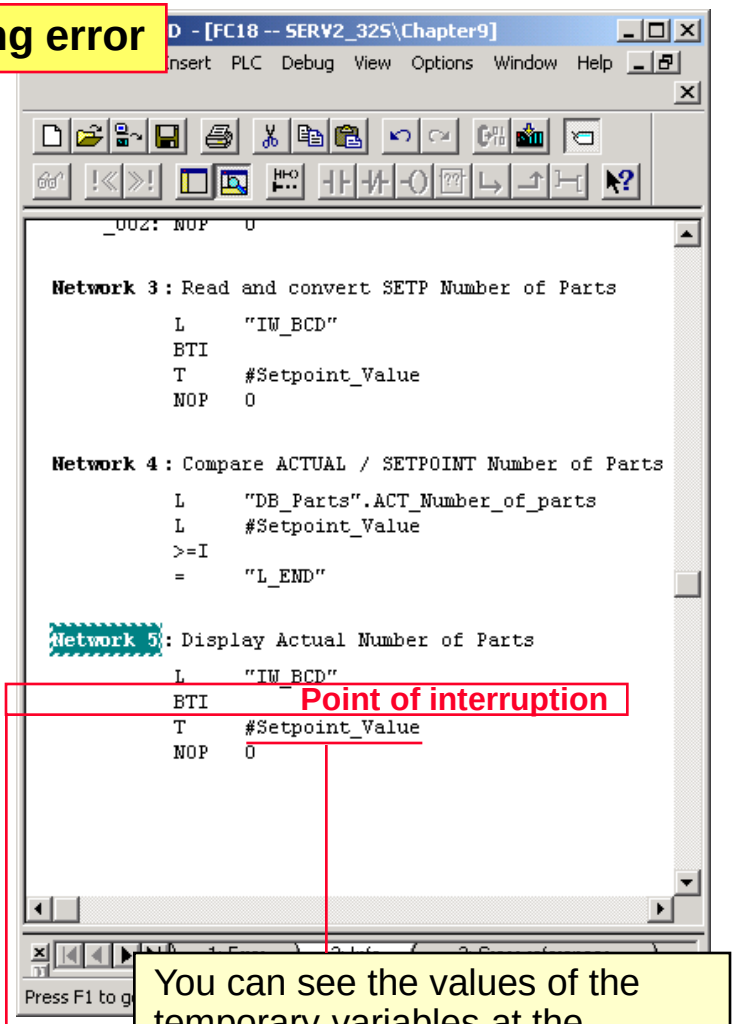
Diagnostics with I Stack, B Stack, L Stack

Program structure

Block containing error



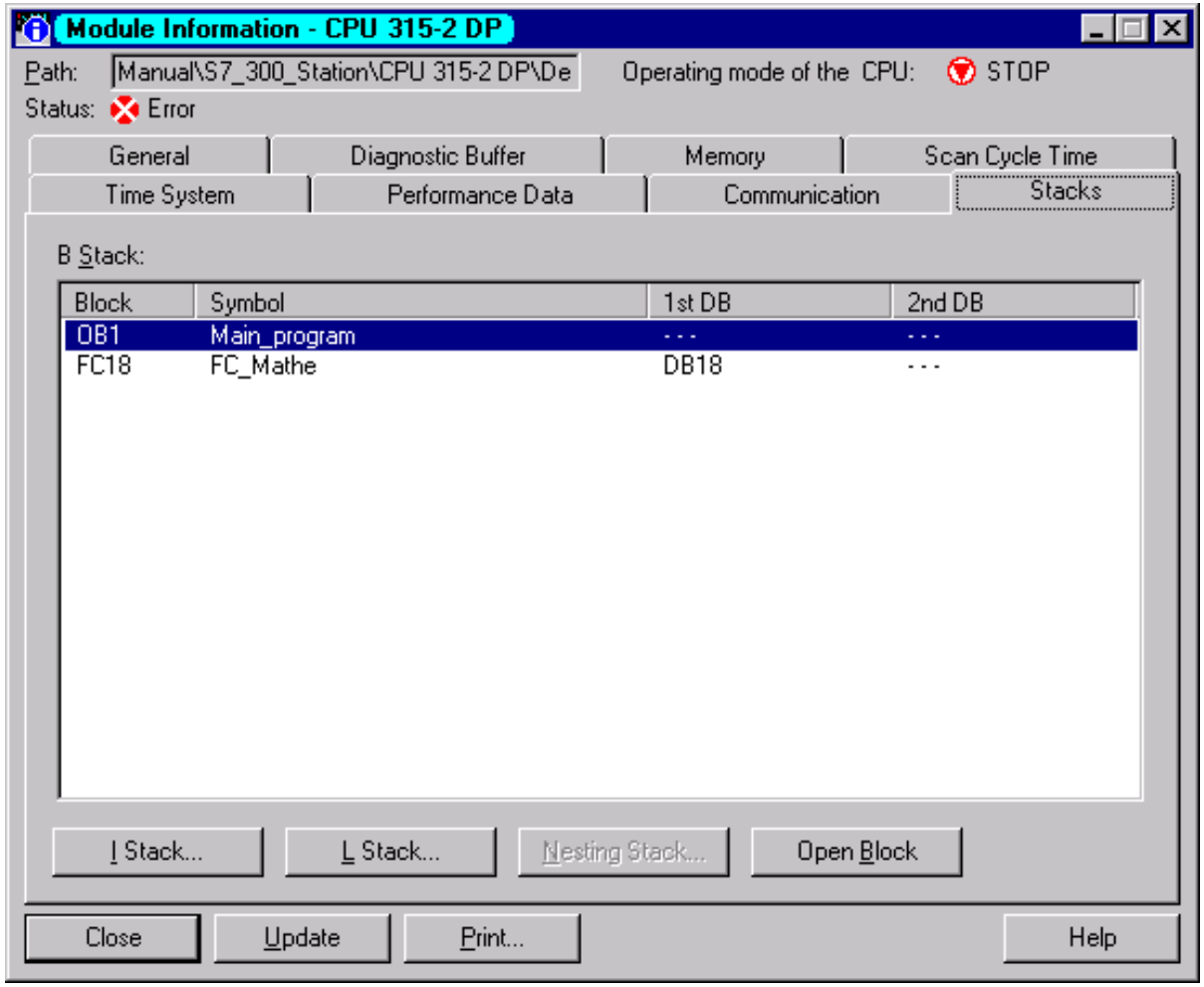
The **B** stack shows a list of the blocks that were executed up to the point of interruption.



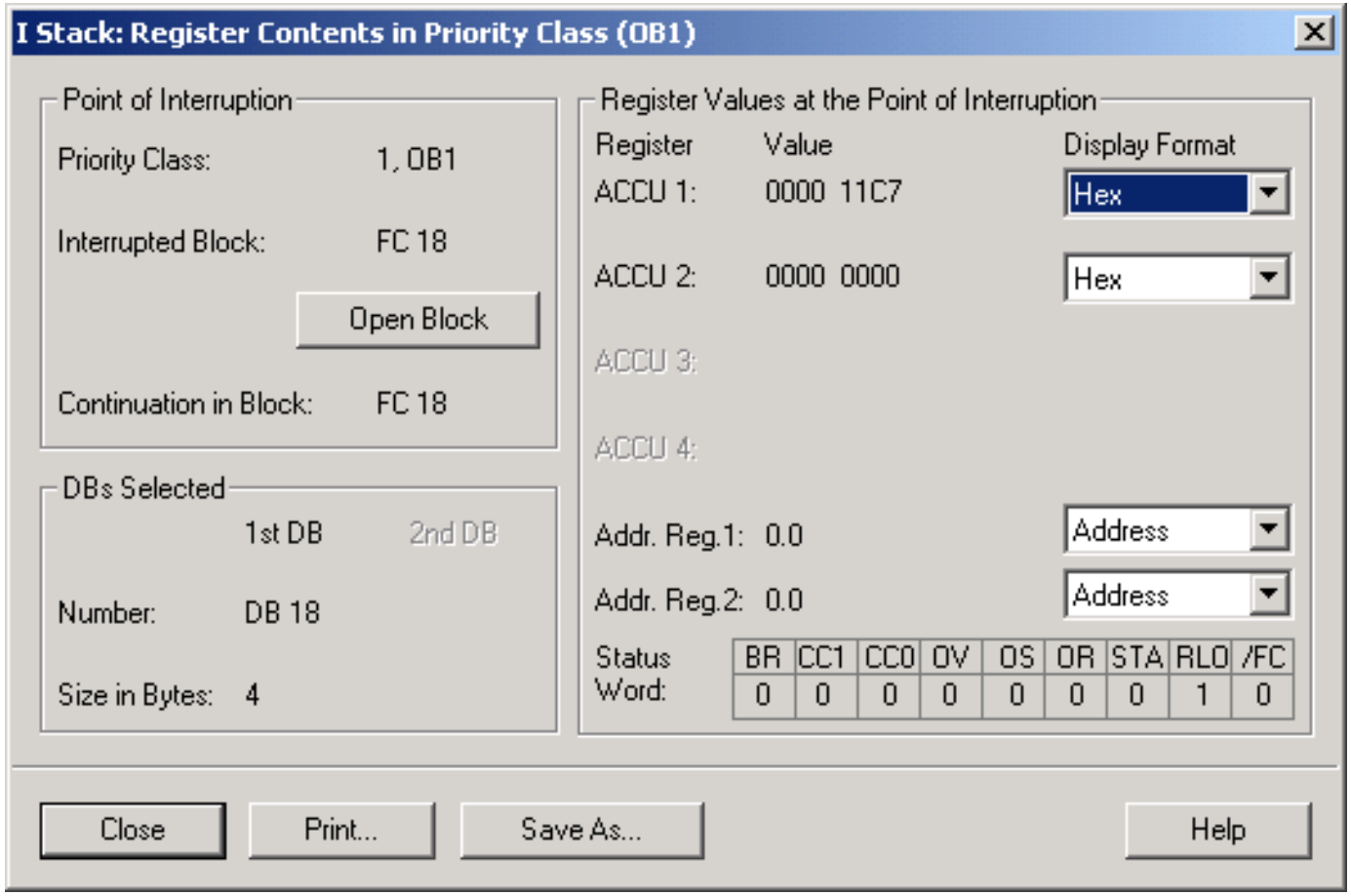
You can see the contents of the accumulators, registers, status word etc. at the time of interruption in the **I** stack.

You can see the values of the temporary variables at the time of interruption in the **L** stack.

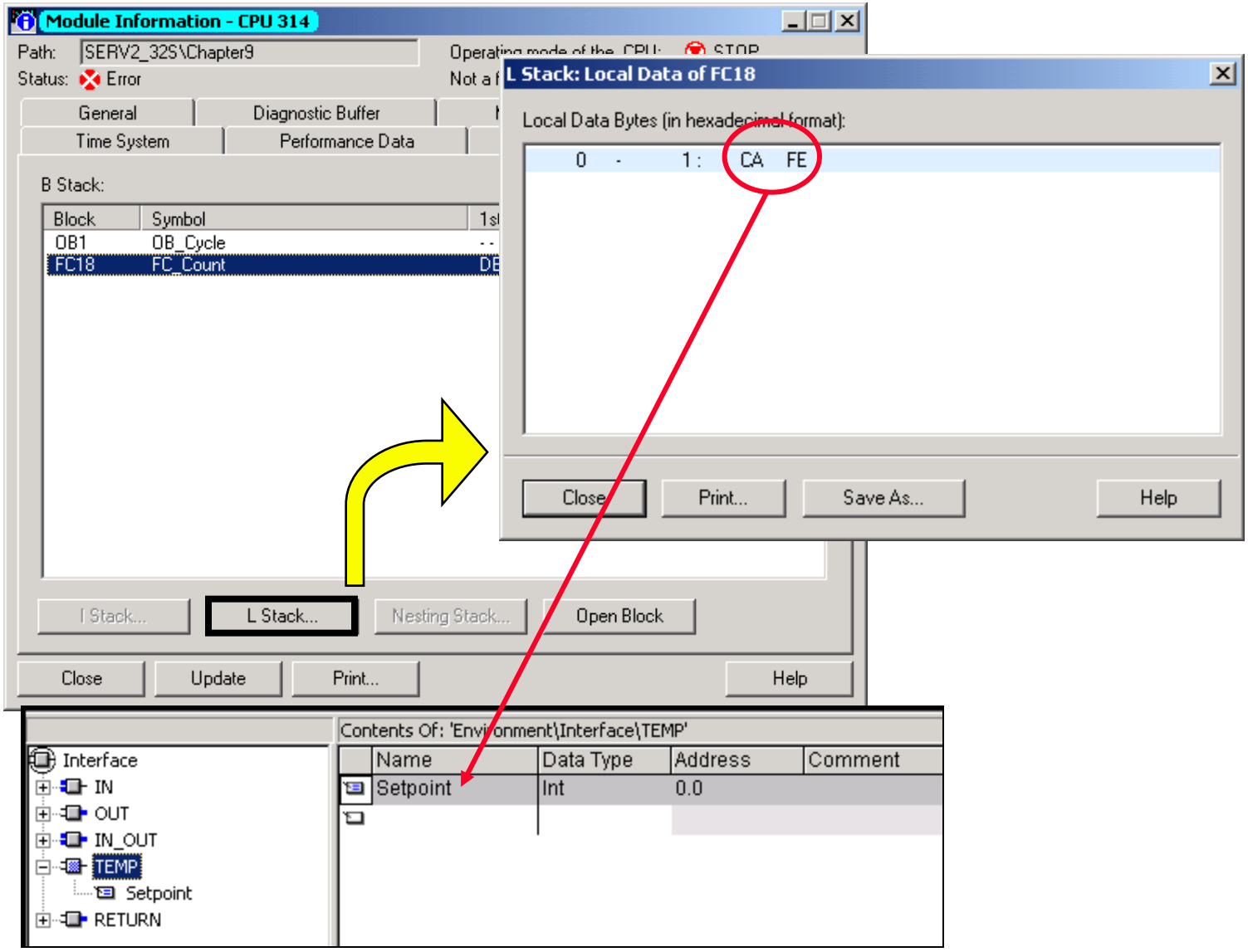
Contents of the B Stack



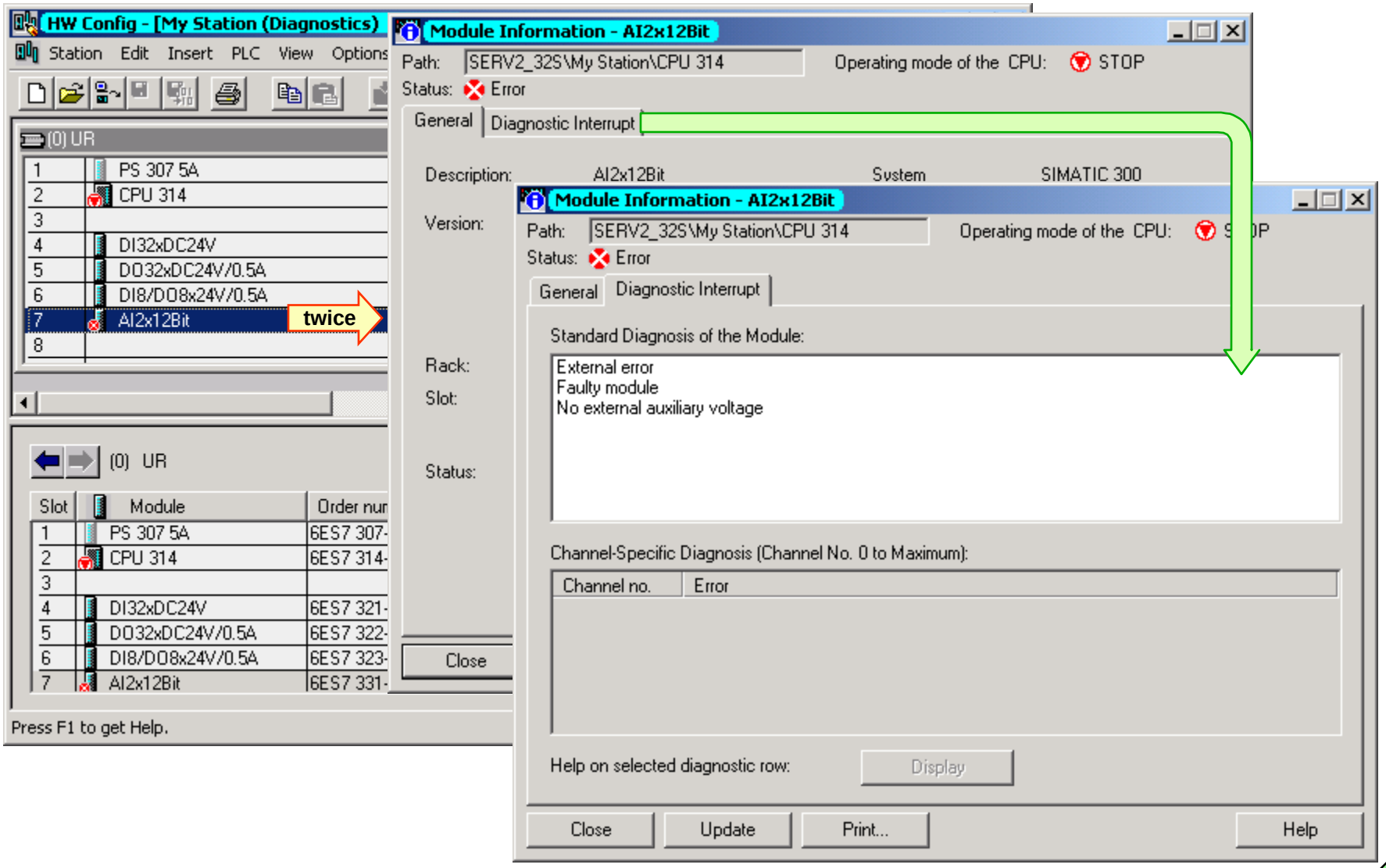
Contents of the I Stack



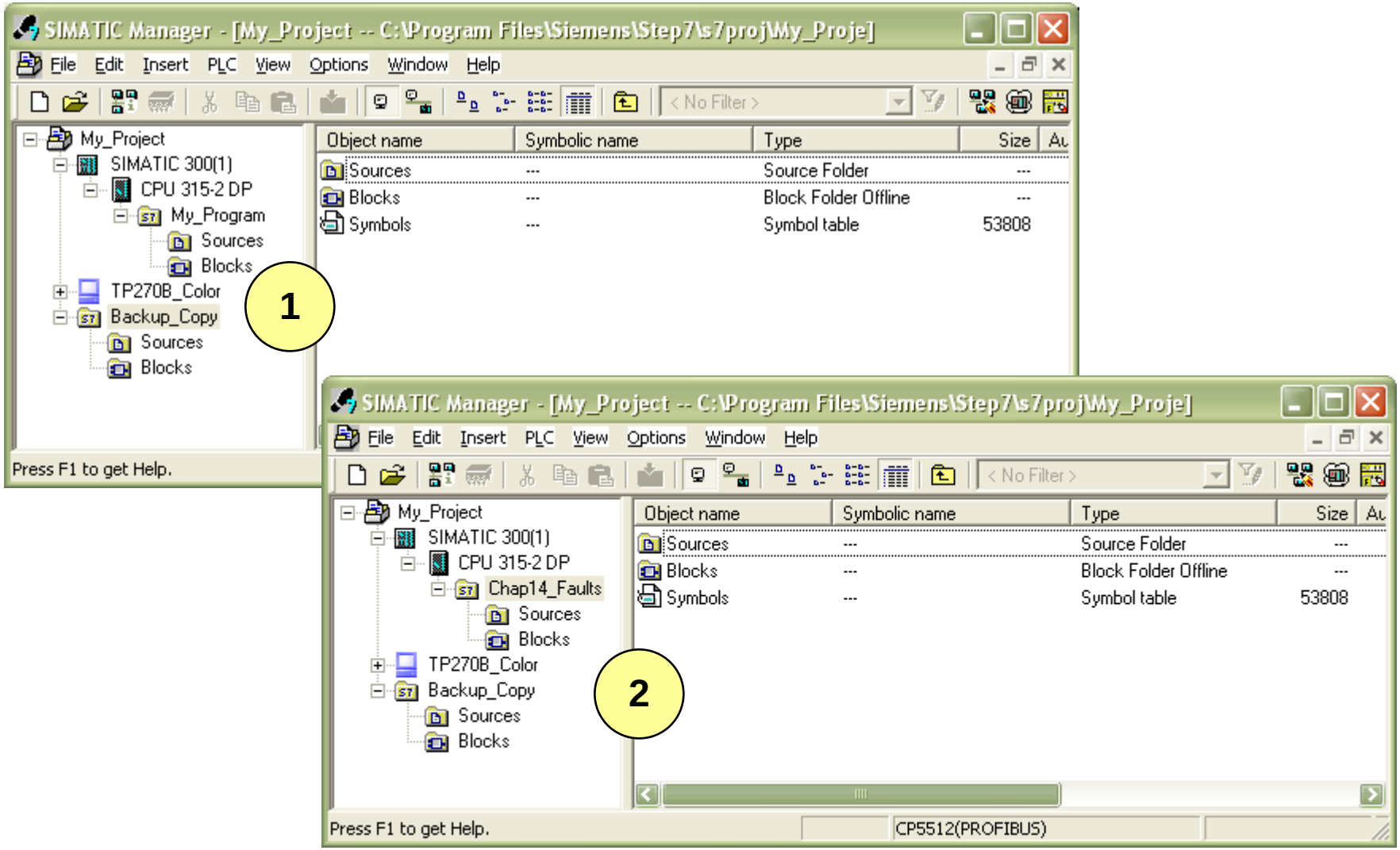
Contents of the L Stack



Displaying the Hardware Diagnostics



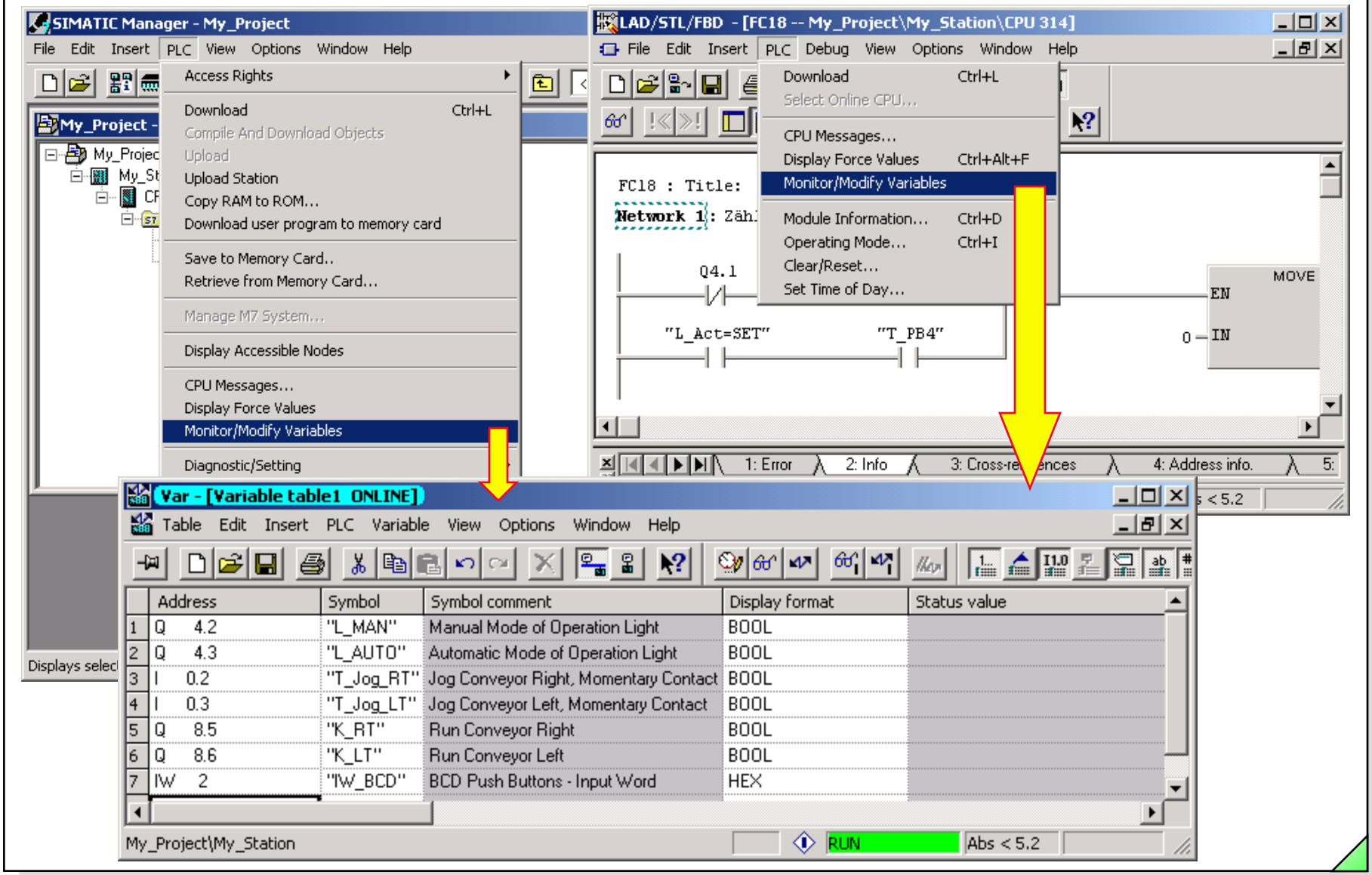
Ex 1: Copying a Faulty Program and Preparing for Troubleshooting



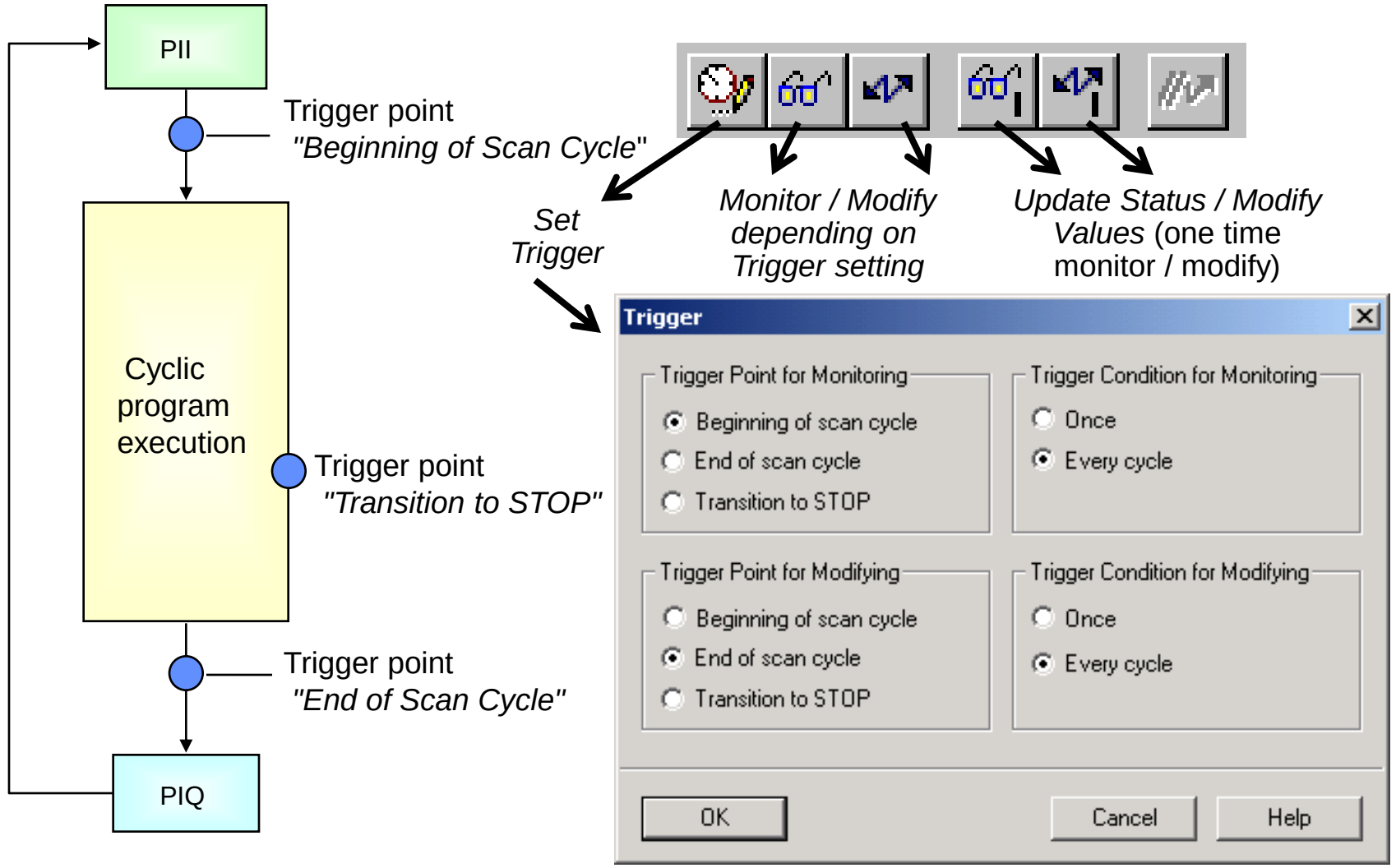
Exercise 2: STOP Troubleshooting

Please Note: That after every STOP error correction, you must carry out a CPU warm restart. If, after the warm restart, the CPU once again goes into the STOP state, there still exists a further STOP error.

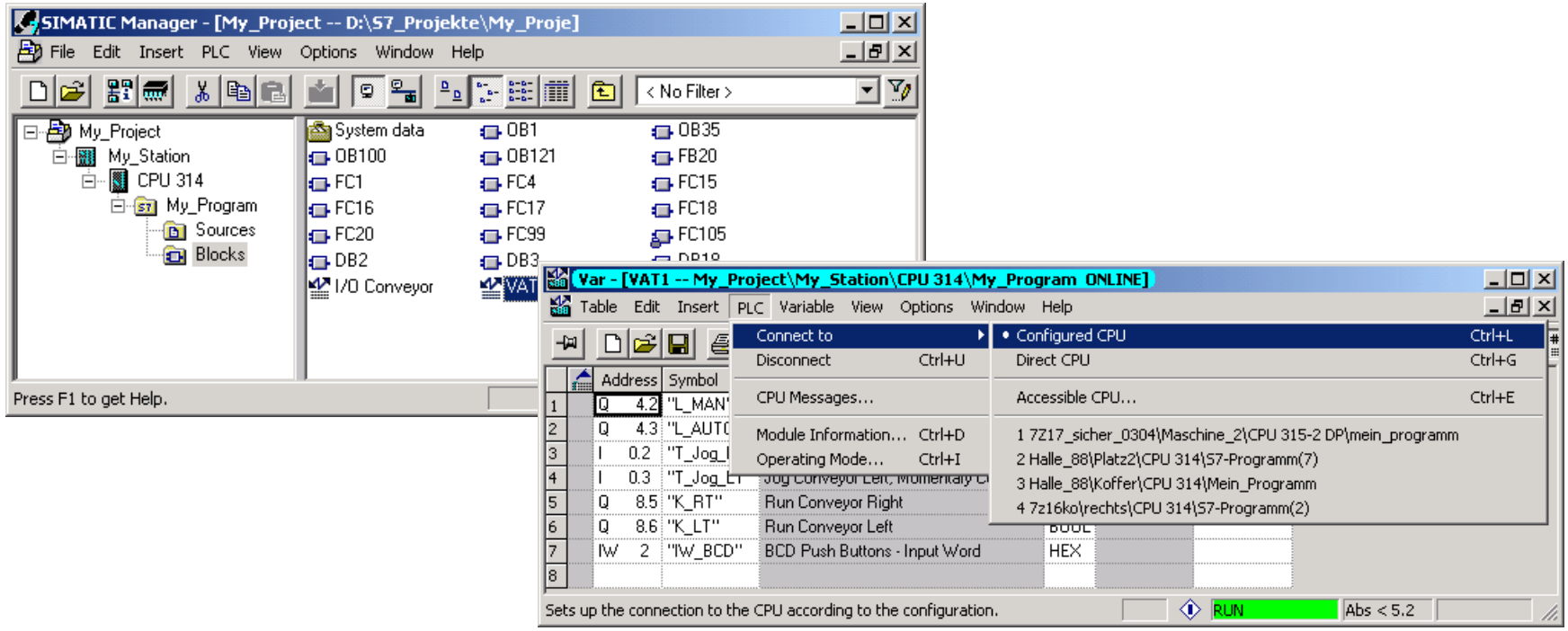
Calling the "Monitor/Modify Variables" Tool

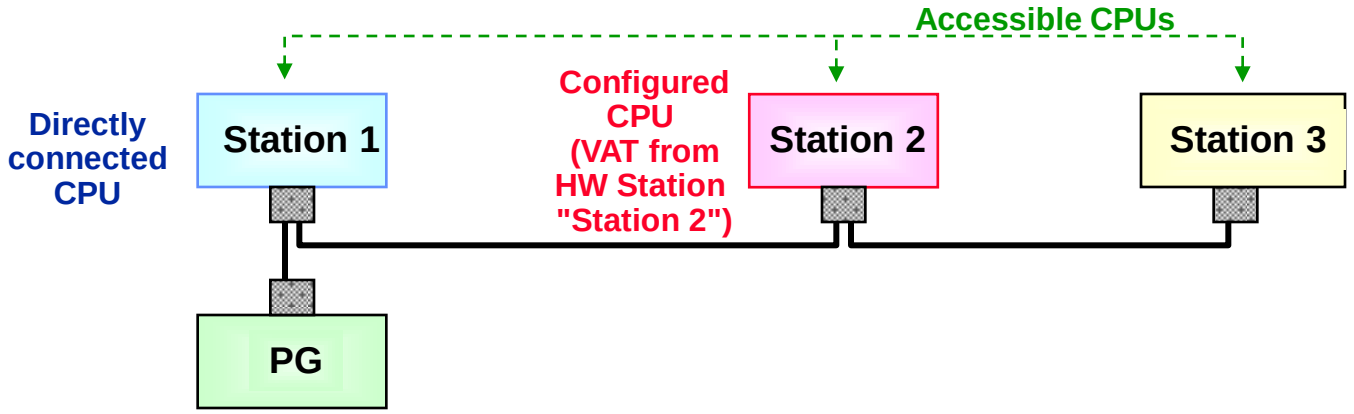


Establishing Trigger Points for "Monitor/Modify Variable"

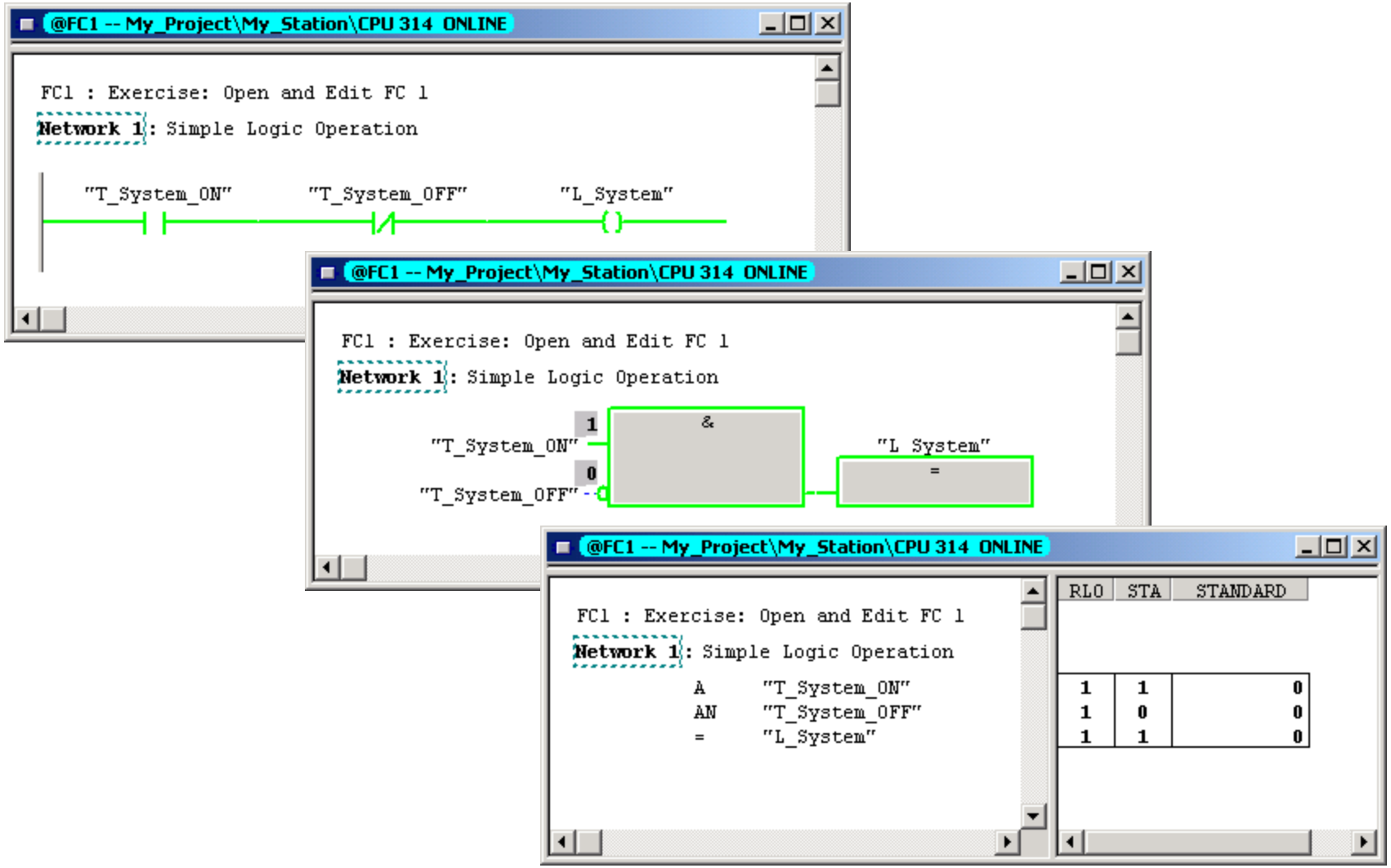


Opening a Variable Table and Establishing a Connection to the CPU





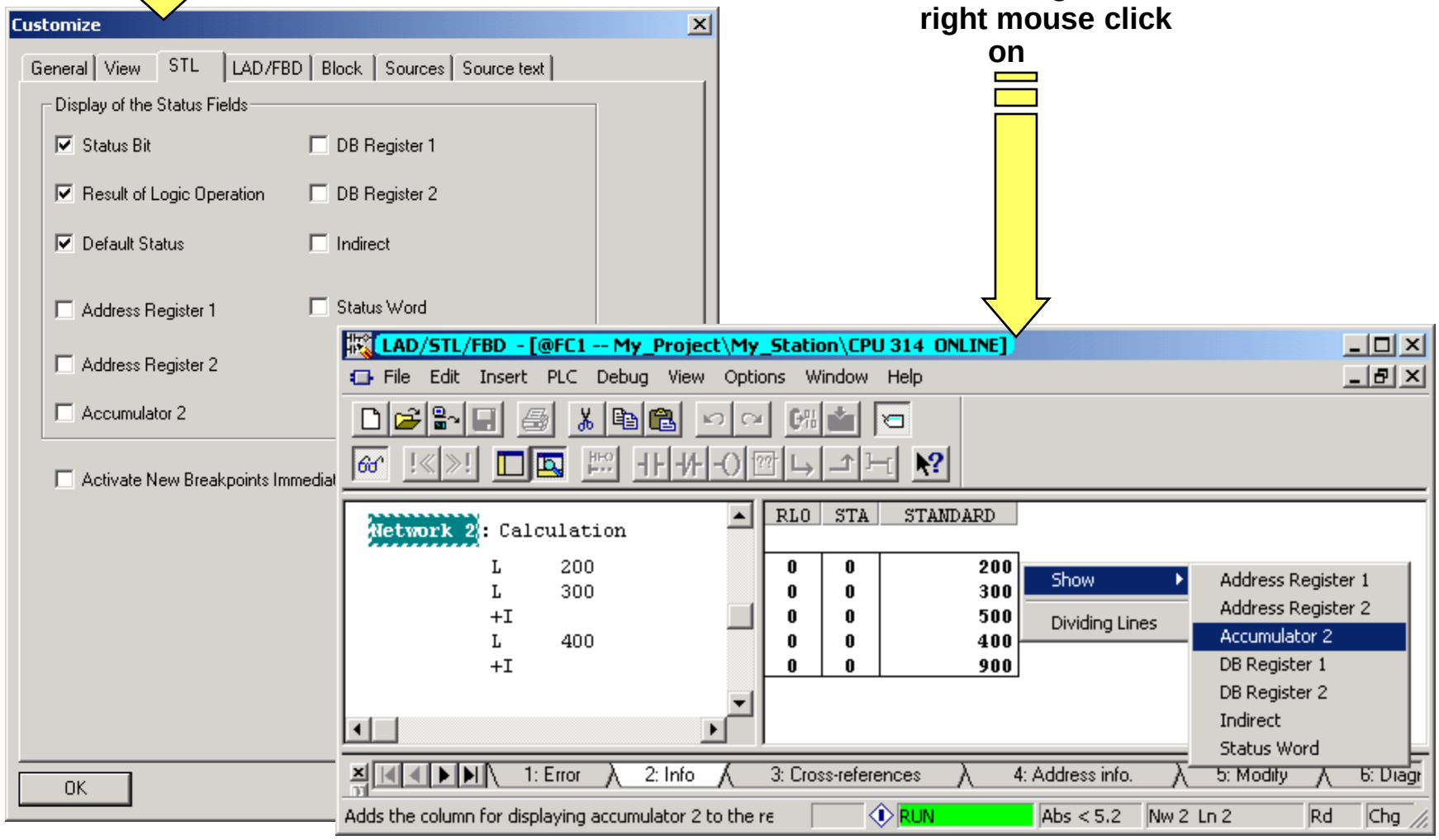
Testing (Debugging) Blocks using "Monitor" (Block Status)



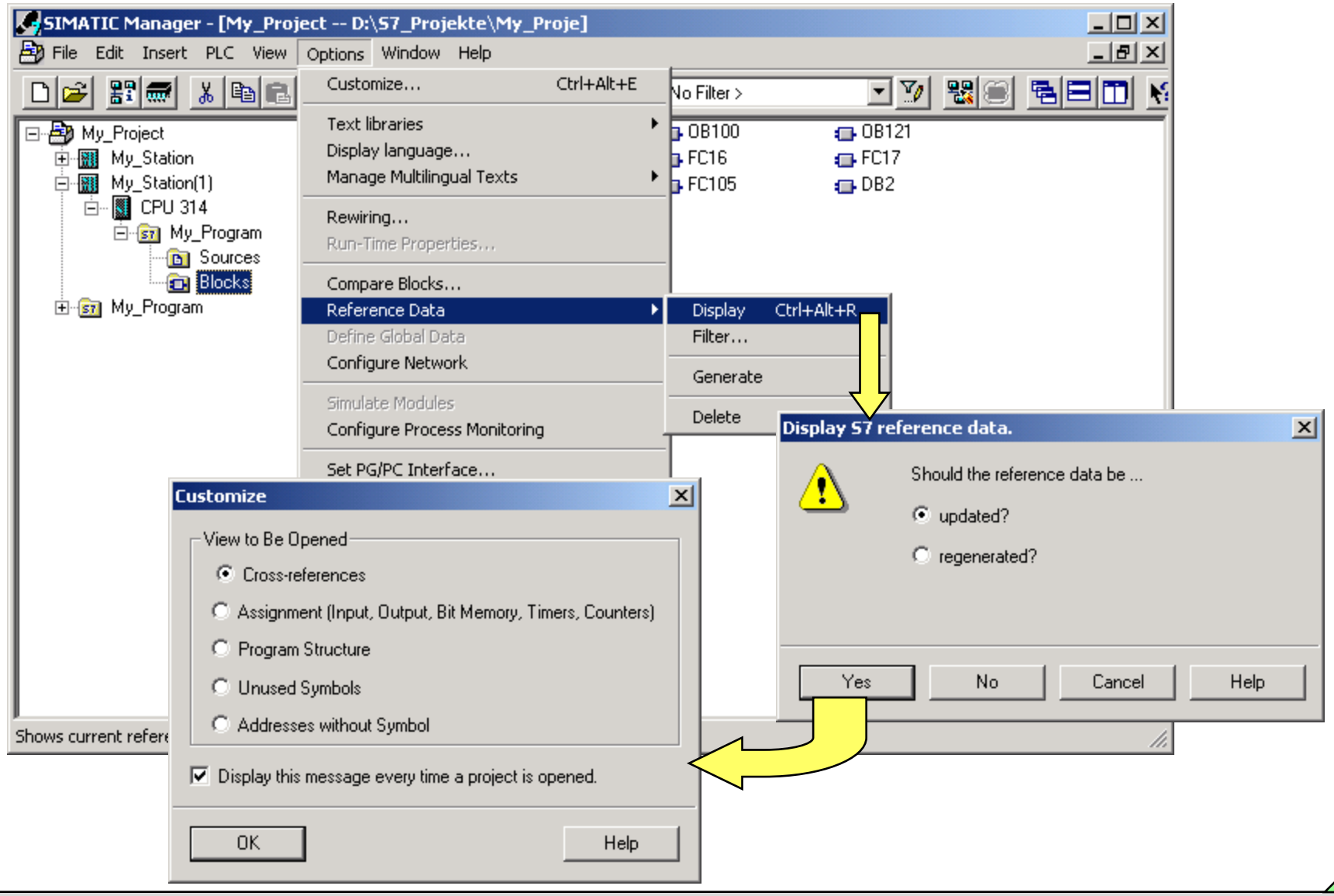
Information Displayed with "Block Debug > Monitor"

LAD/STL/FBD Editor -> *Options ->Customize*

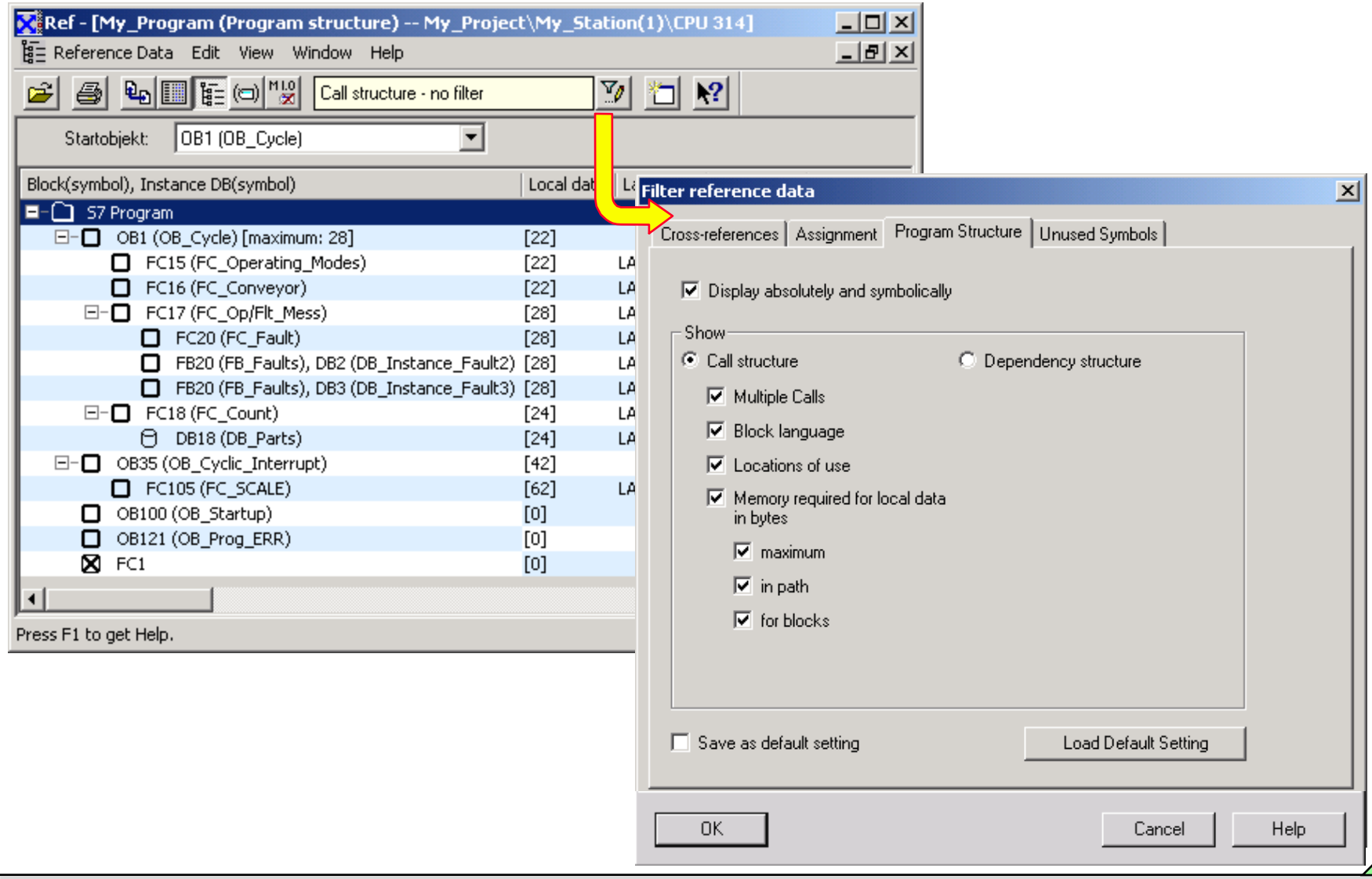
... or
in running status:
right mouse click
on



Displaying Reference Data



Displaying the Program Structure



Displaying Cross References

Ref - [My_Program (Cross-references) -- My_Project\My_Station(1)\CPU 314]

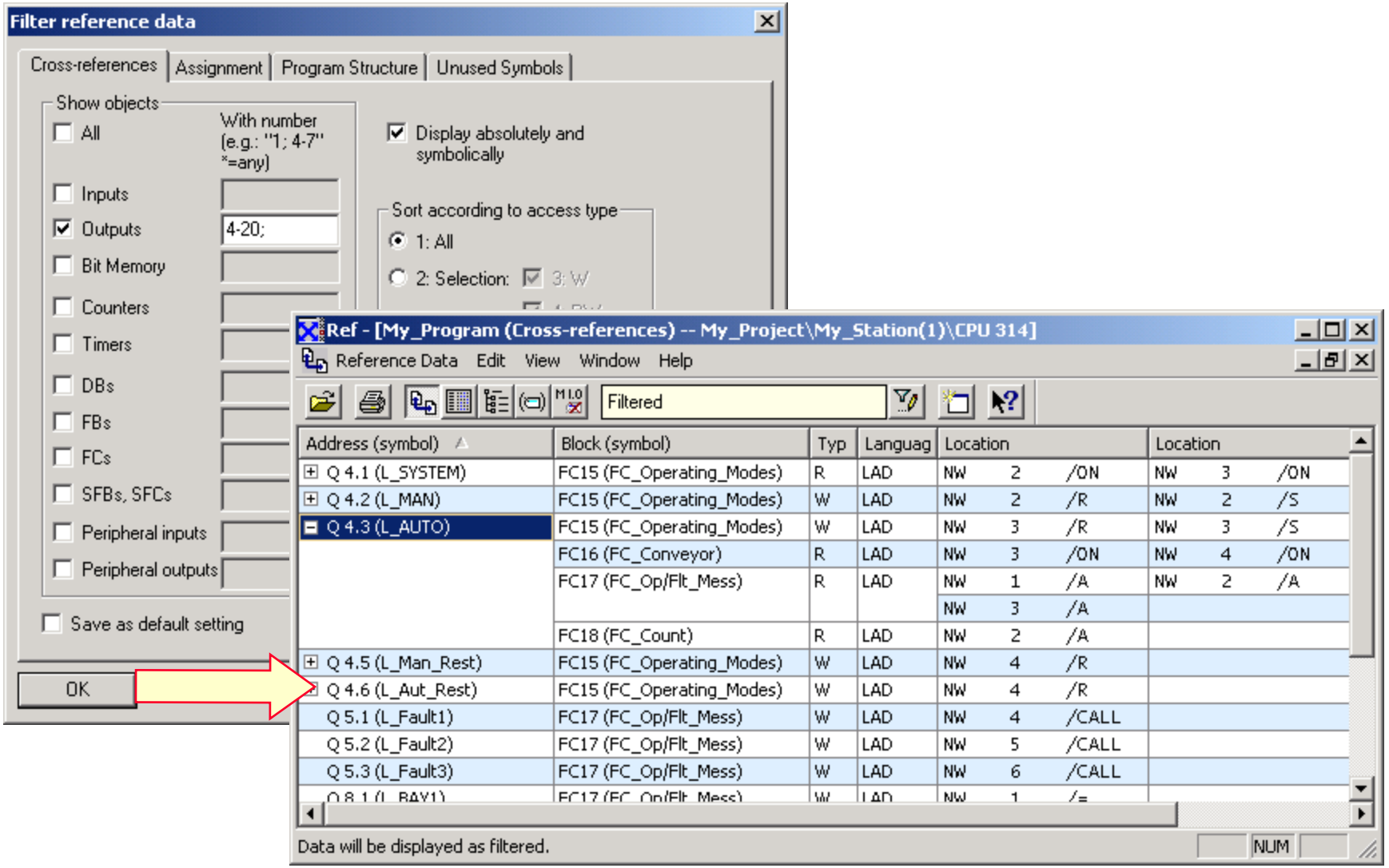
Reference Data Edit View Window Help

Filtered

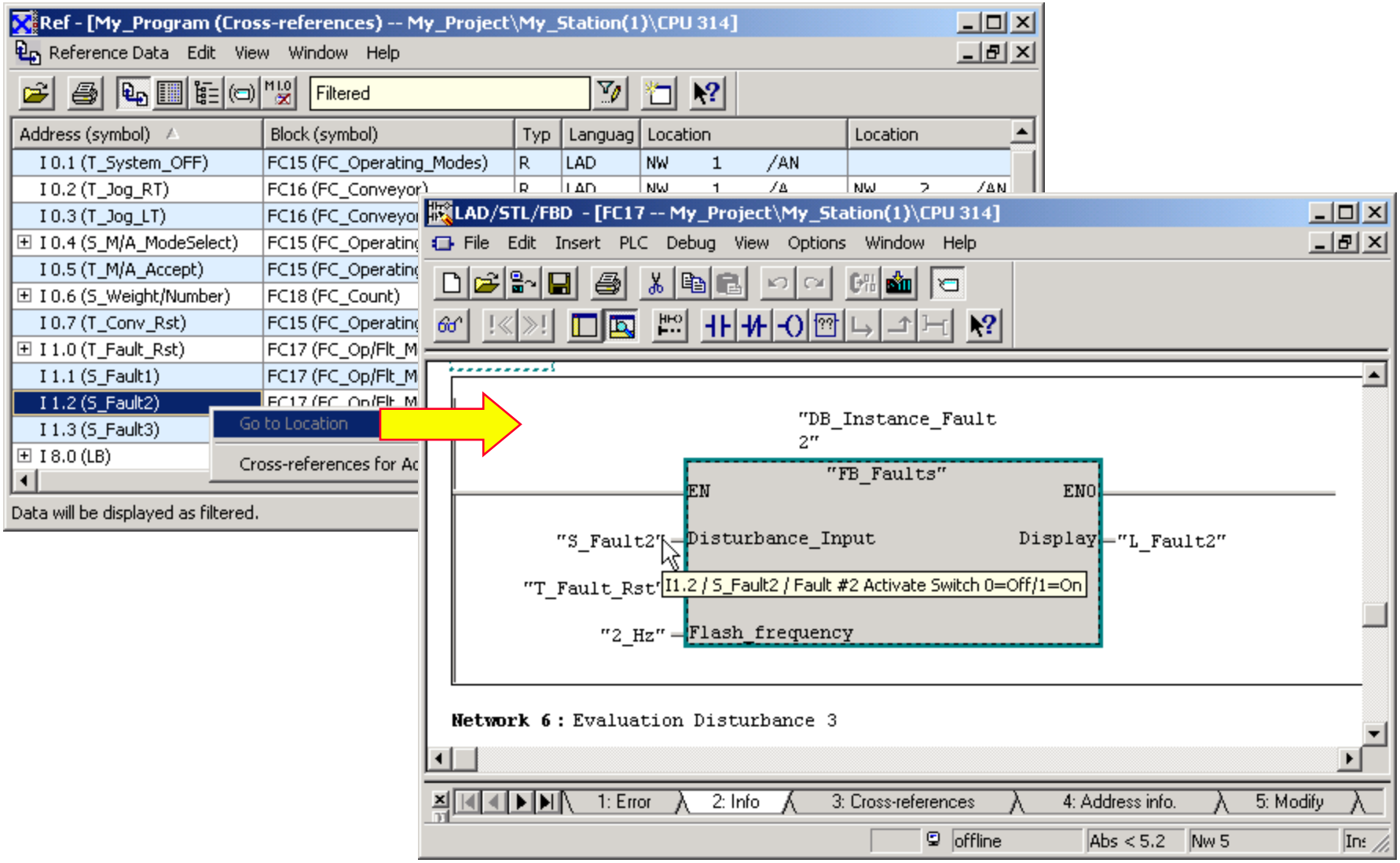
Address (symbol)	Block (symbol)	Typ	Languag	Location			Location		
I 0.0 (T_System_ON)	FC15 (FC_Operating_Modes)	R	LAD	NW	1	/A			
I 0.1 (T_System_OFF)	FC15 (FC_Operating_Modes)	R	LAD	NW	1	/AN			
I 0.2 (T_Jog_RT)	FC16 (FC_Conveyor)	R	LAD	NW	1	/A	NW	2	/AN
I 0.3 (T_Jog_LT)	FC16 (FC_Conveyor)	R	LAD	NW	1	/AN	NW	2	/A
⊟ I 0.4 (S_M/A_ModeSelect)	FC15 (FC_Operating_Modes)	R	LAD	NW	2	/AN	NW	2	/O
				NW	3	/A	NW	3	/ON
I 0.5 (T_M/A_Accept)	FC15 (FC_Operating_Modes)	R	LAD	NW	2	/A	NW	3	/A
⊟ I 0.6 (S_Weight/Number)	FC18 (FC_Count)	R	LAD	NW	5	/AN			
	OB35 (OB_Cyclic_Interrupt)	R	LAD	NW	4	/A			
I 0.7 (T_Conv_Rst)	FC15 (FC_Operating_Modes)	R	LAD	NW	4	/A			
⊞ I 1.0 (T_Fault_Rst)	FC17 (FC_Op/Flt_Mess)	R	LAD	NW	4	/A	NW	5	/A
I 1.1 (S_Fault1)	FC17 (FC_Op/Flt_Mess)	R	LAD	NW	4	/A			
I 1.2 (S_Fault2)	FC17 (FC_Op/Flt_Mess)	R	LAD	NW	5	/A			
I 1.3 (S_Fault3)	FC17 (FC_Op/Flt_Mess)	R	LAD	NW	6	/A			
⊞ I 8.0 (LB)	FC16 (FC_Conveyor)	R	LAD	NW	3	/A	NW	4	/A
I 8.3 (T_PB3)	FC17 (FC_Op/Flt_Mess)	R	LAD	NW	3	/A			
I 8.4 (T_PB4)	FC18 (FC_Count)	R	LAD	NW	1	/A			
⊞ I 8.5 (BAY1)	FC16 (FC_Conveyor)	R	LAD	NW	4	/A	NW	4	/AN
⊞ I 8.6 (BAY2)	FC16 (FC_Conveyor)	R	LAD	NW	4	/A	NW	4	/AN
⊞ I 8.7 (BAY3)	FC16 (FC_Conveyor)	R	LAD	NW	3	/O			
IW 2 (IW_BCD)	FC18 (FC_Count)	R	LAD	NW	3	/L			
⊞ M 10.3 (2_Hz)	FC17 (FC_Op/Flt_Mess)	R	LAD	NW	1	/A	NW	2	/A
M 10.5 (1_Hz)	FC17 (FC_Op/Flt_Mess)	R	LAD	NW	1	/A	NW	2	/A
M 16.0 (M_LB_Edge(FC16a))	FC16 (FC_Conveyor)	W	LAD	NW	4	/FP			
⊞ M 16.1 (M_Jog_left)	FC16 (FC_Conveyor)	R	LAD	NW	6	/O			

Press F1 to get Help. NUM

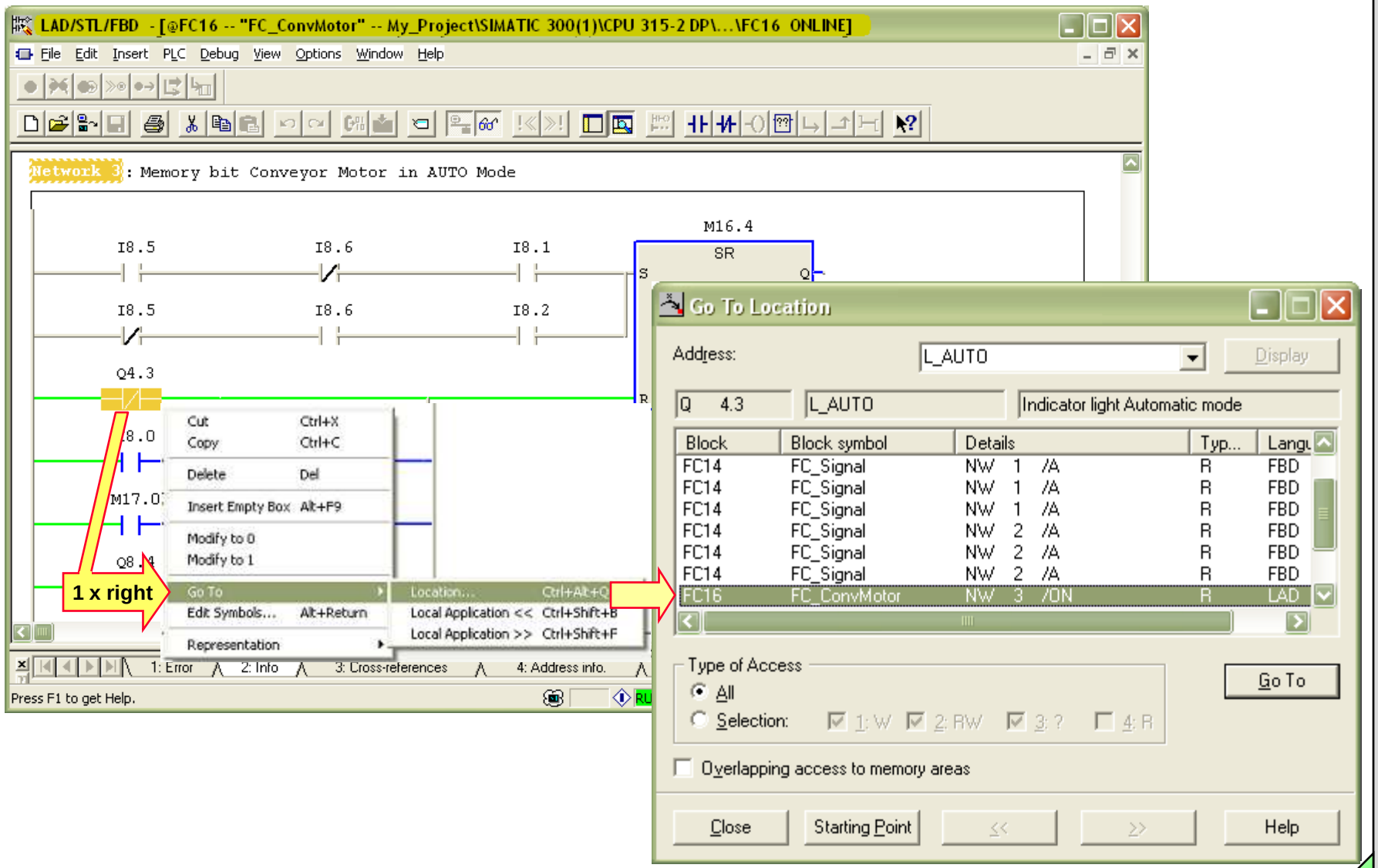
Filtering Cross References



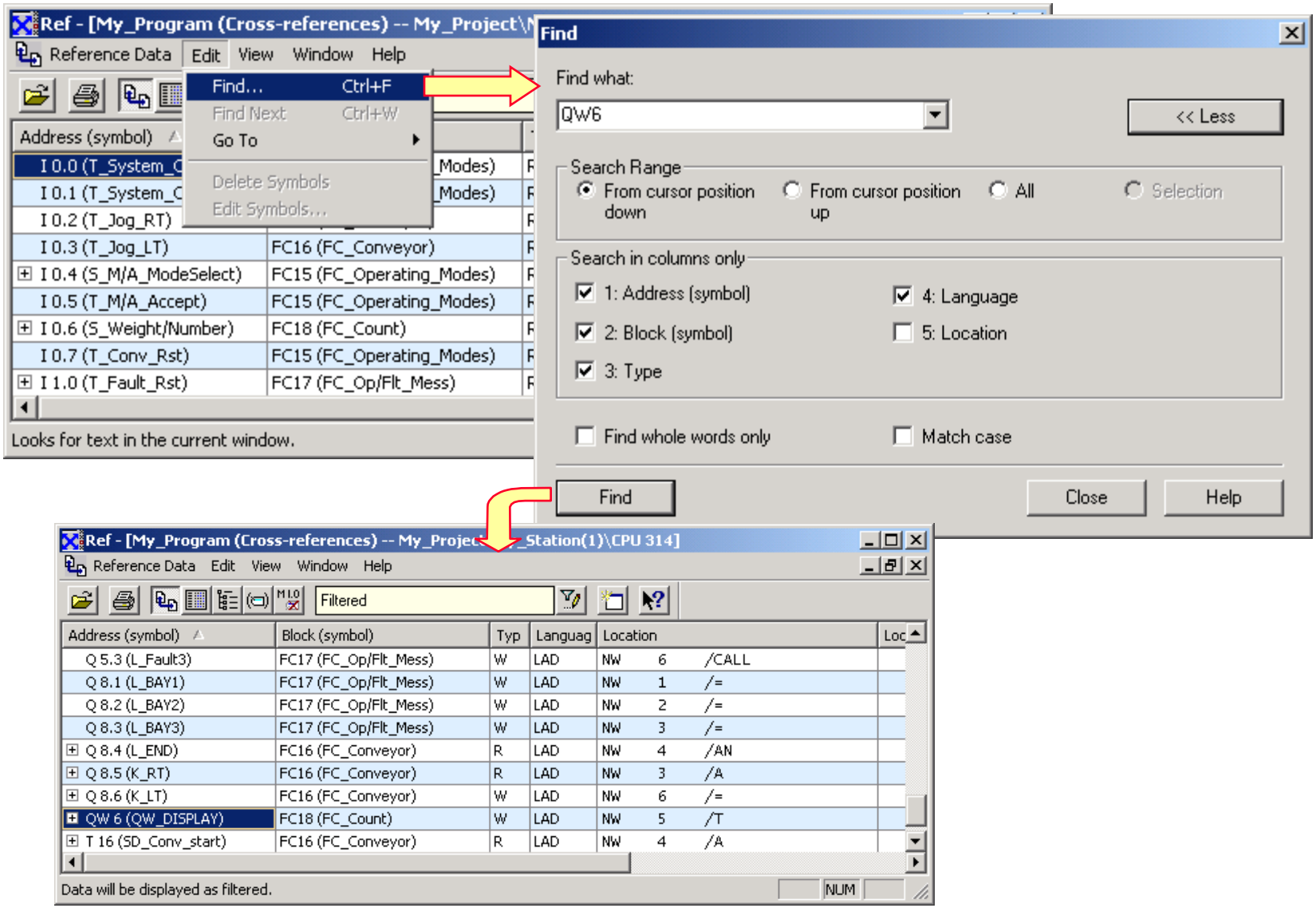
Block Correction using Cross References



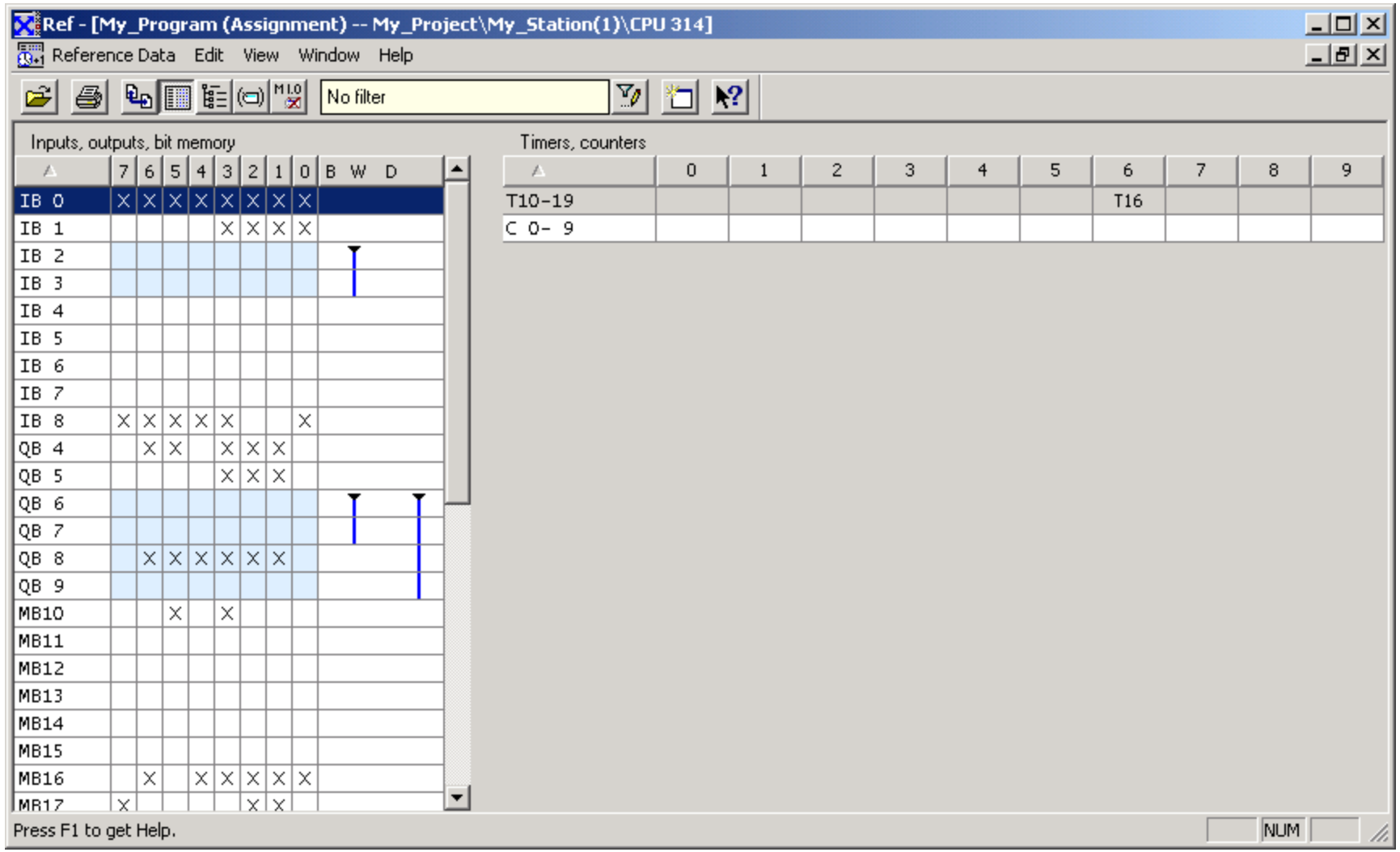
Go To Location



"Find" In Reference Data



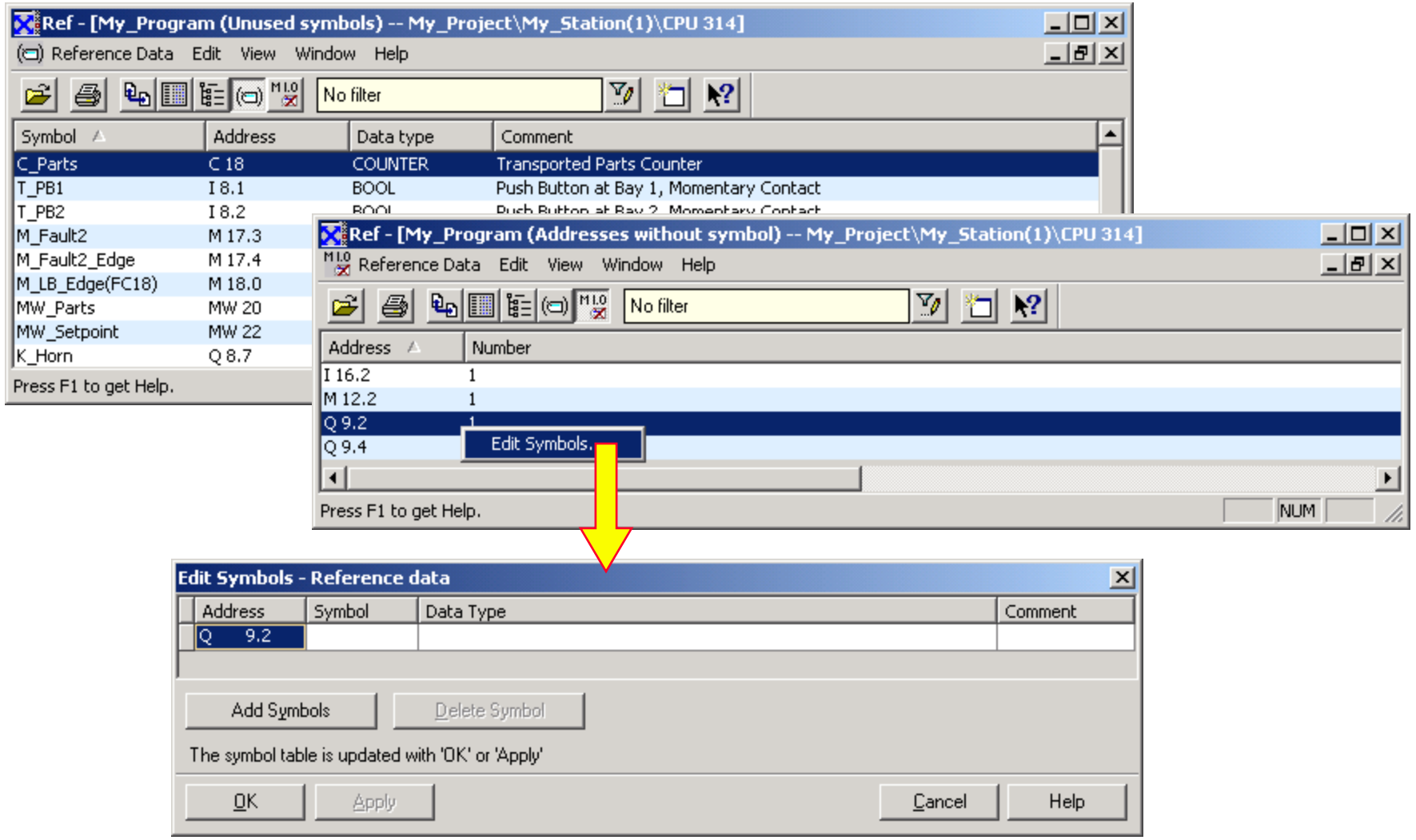
Assignment of I, Q, M, T, C



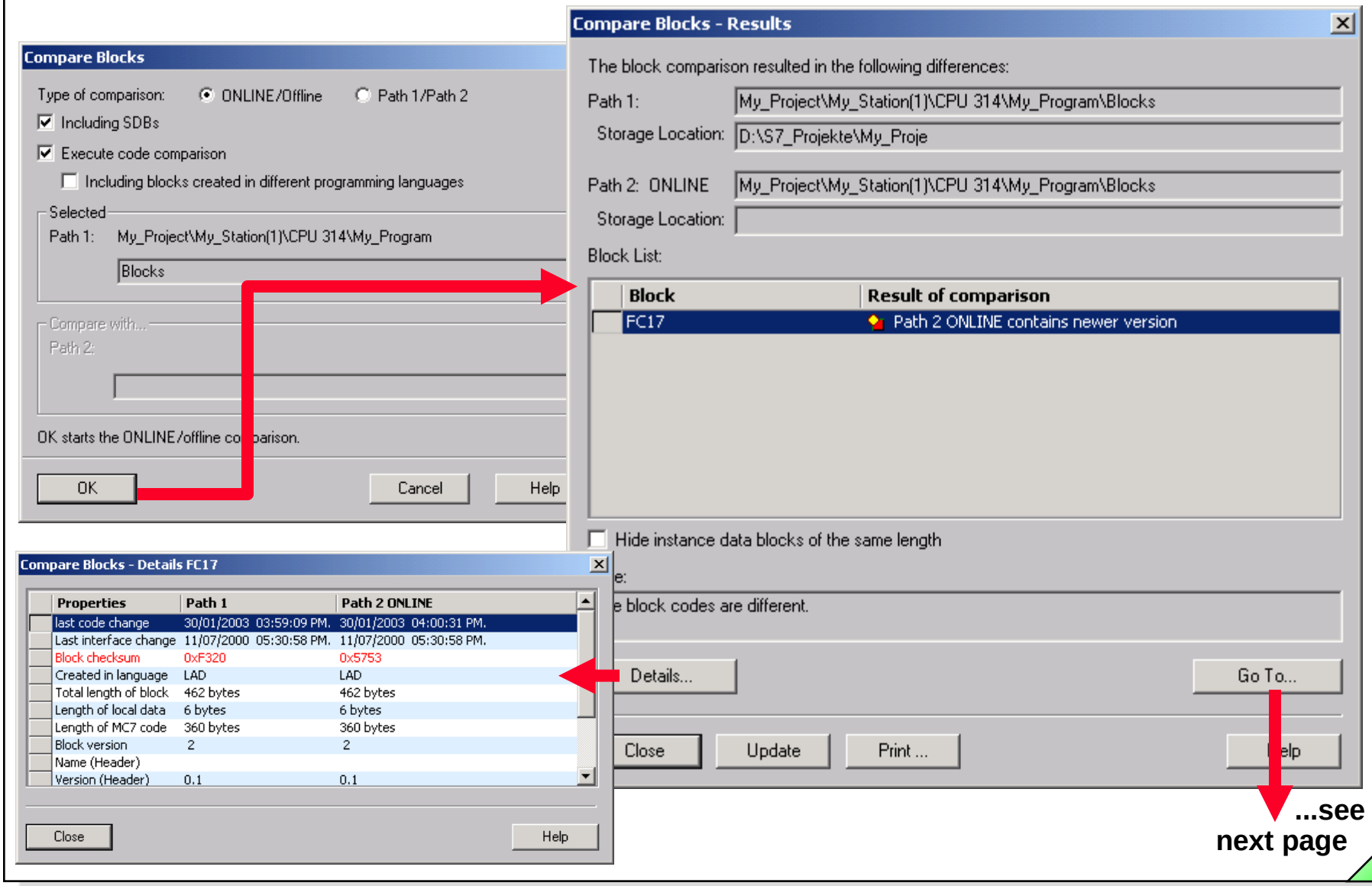
Inputs, outputs, bit memory	7	6	5	4	3	2	1	0	B W D
IB 0	X	X	X	X	X	X	X	X	
IB 1					X	X	X	X	
IB 2									
IB 3									
IB 4									
IB 5									
IB 6									
IB 7									
IB 8	X	X	X	X	X			X	
QB 4		X	X		X	X	X		
QB 5					X	X	X		
QB 6									
QB 7									
QB 8		X	X	X	X	X	X		
QB 9									
MB10			X		X				
MB11									
MB12									
MB13									
MB14									
MB15									
MB16		X		X	X	X	X	X	
MB17	X					X	X		

Timers, counters	0	1	2	3	4	5	6	7	8	9
T10-19							T16			
C 0- 9										

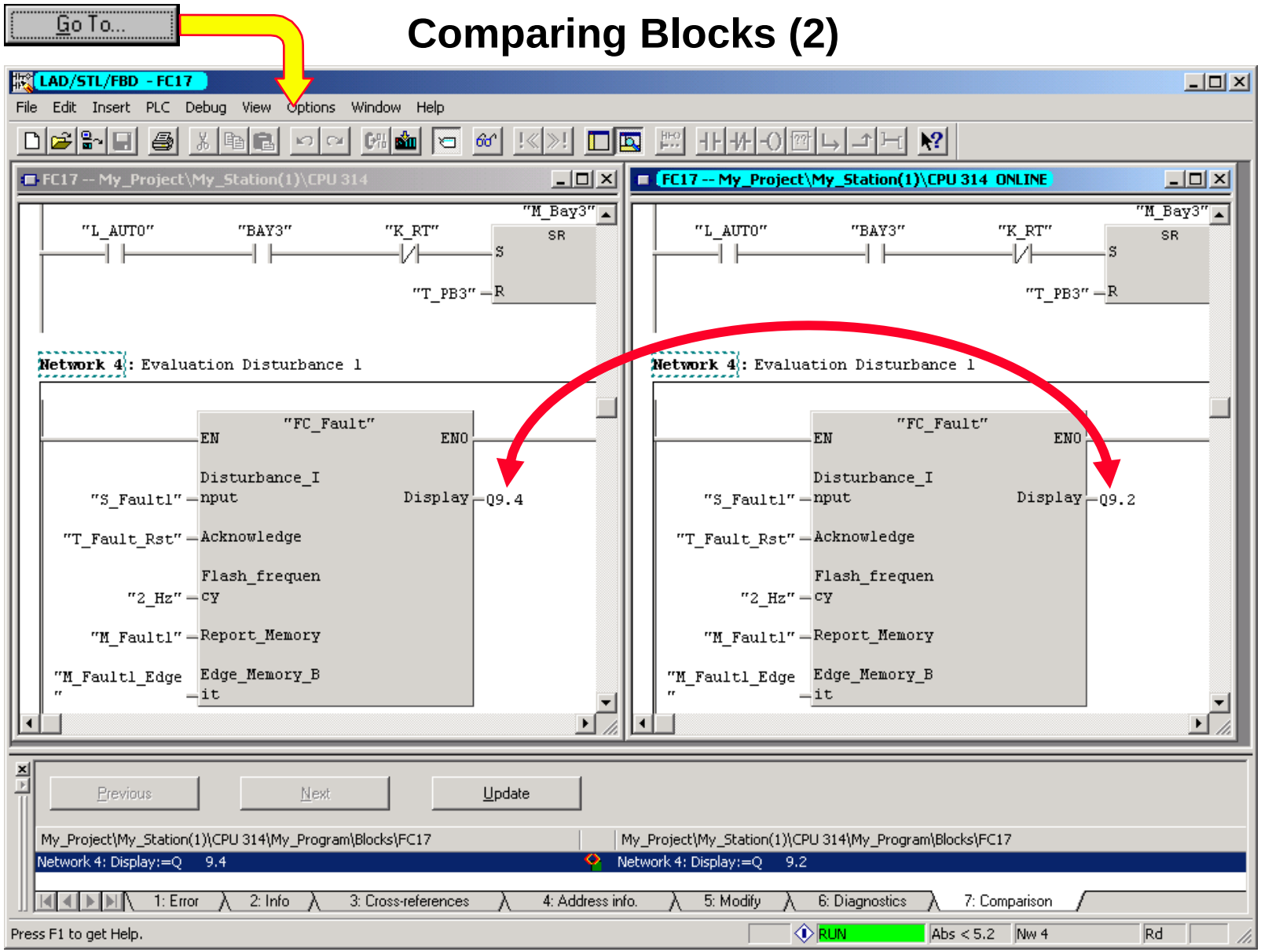
Unused Symbols / Addresses without Symbols



Comparing Blocks (1)

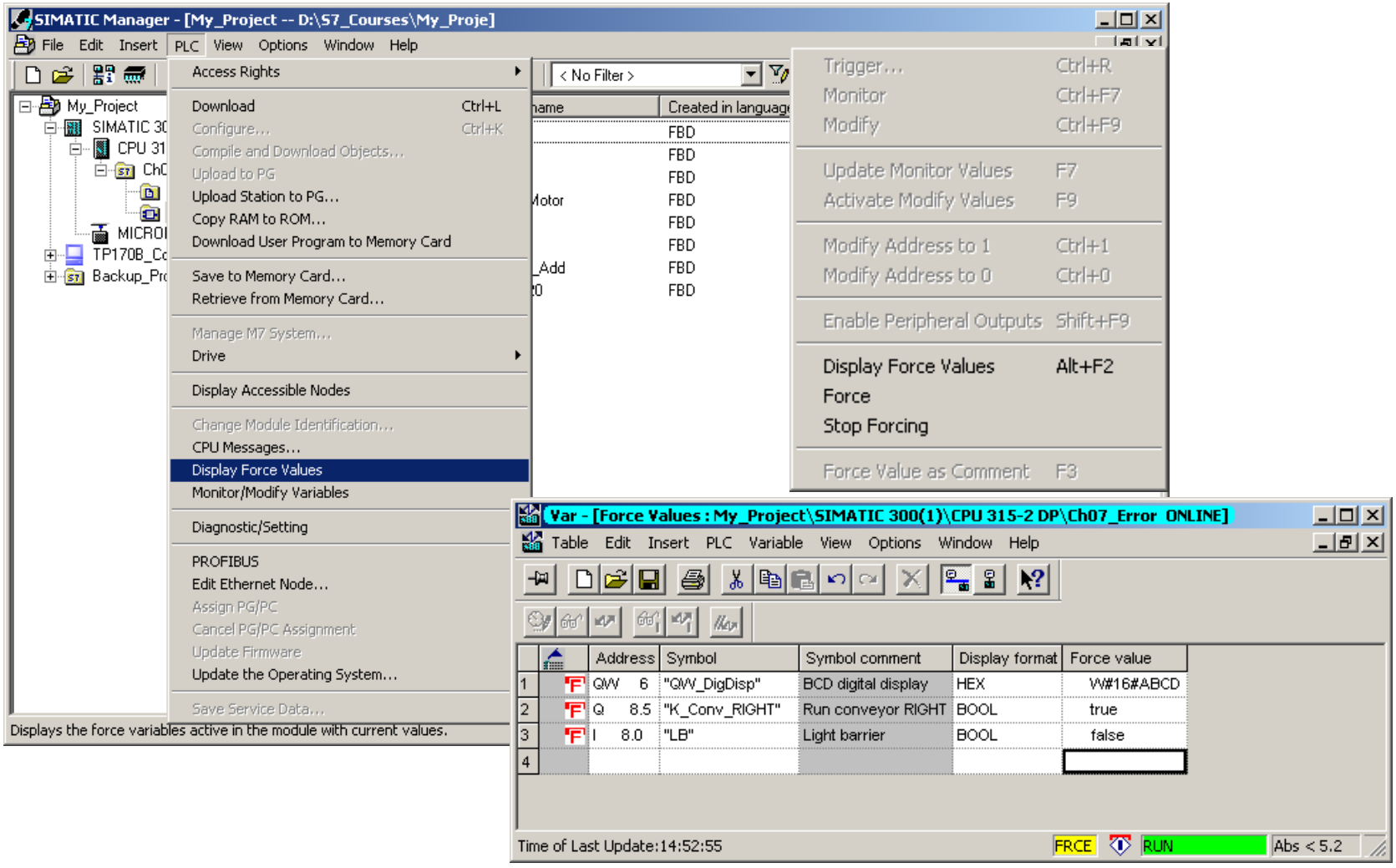


Comparing Blocks (2)

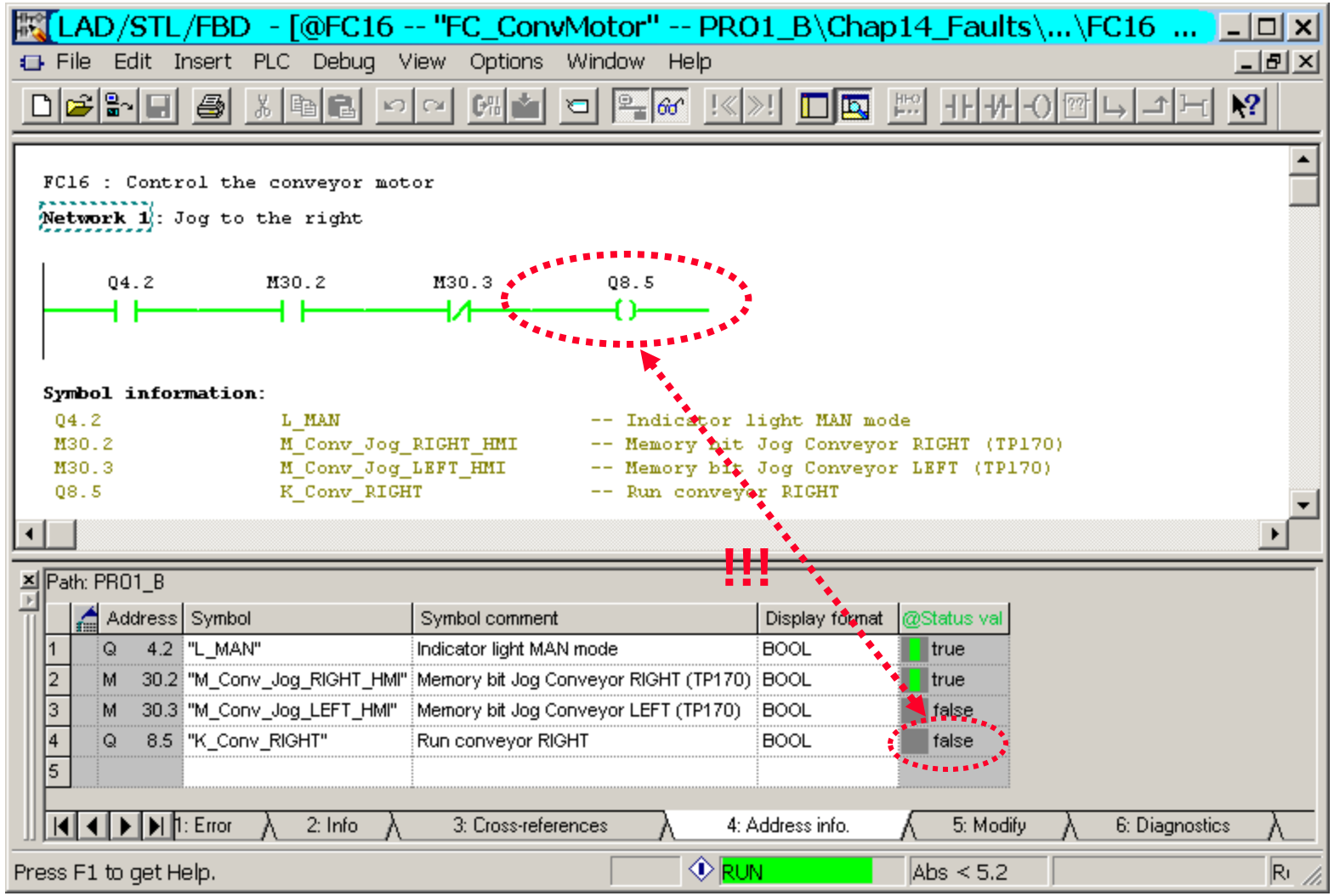


Date: 2/23/2012
File: PRO1_14E.34

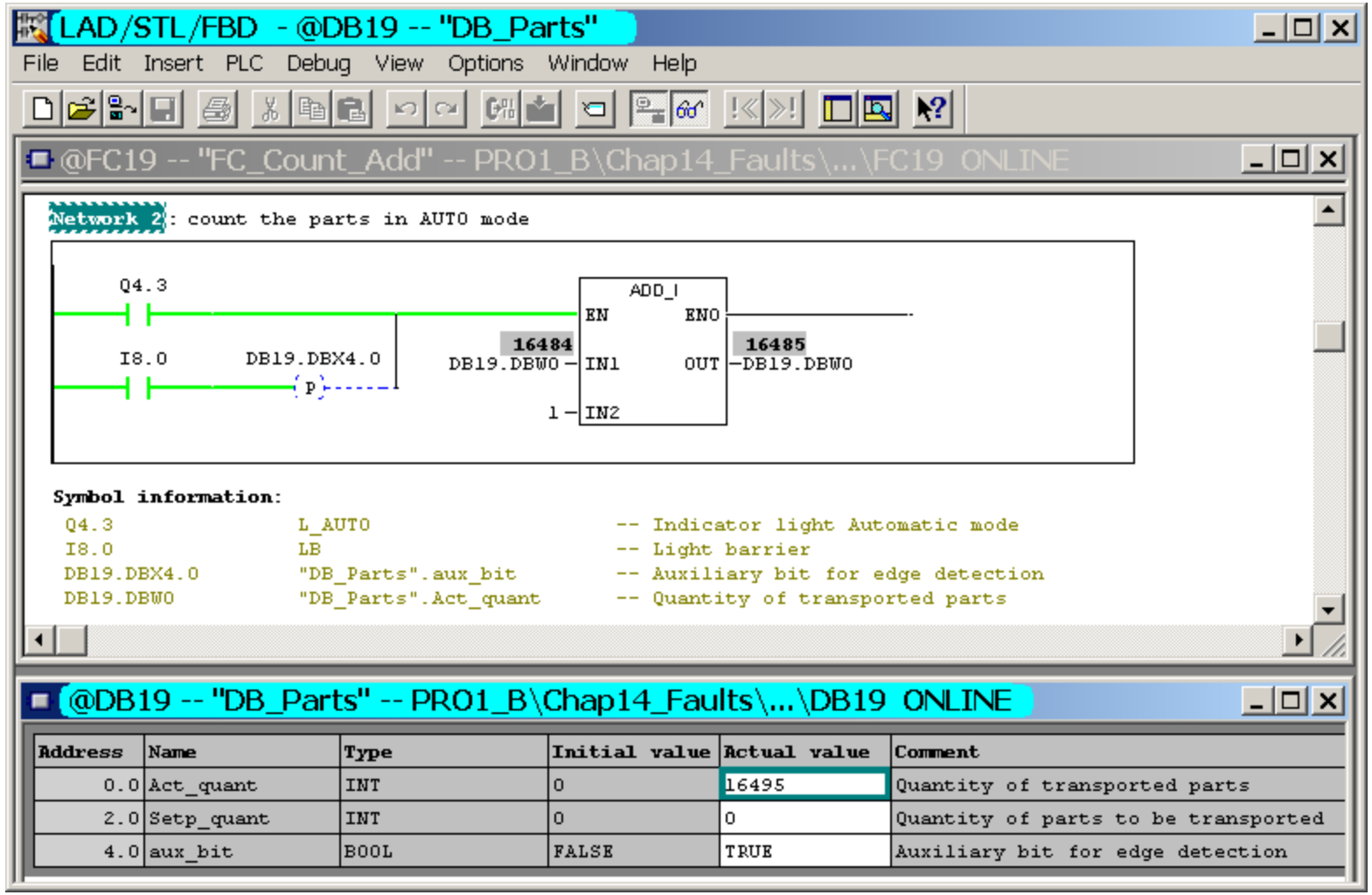
"Force" Variables



Exercise 3: Testing the Motor Jog



Exercise 4: Testing the Quantity Display

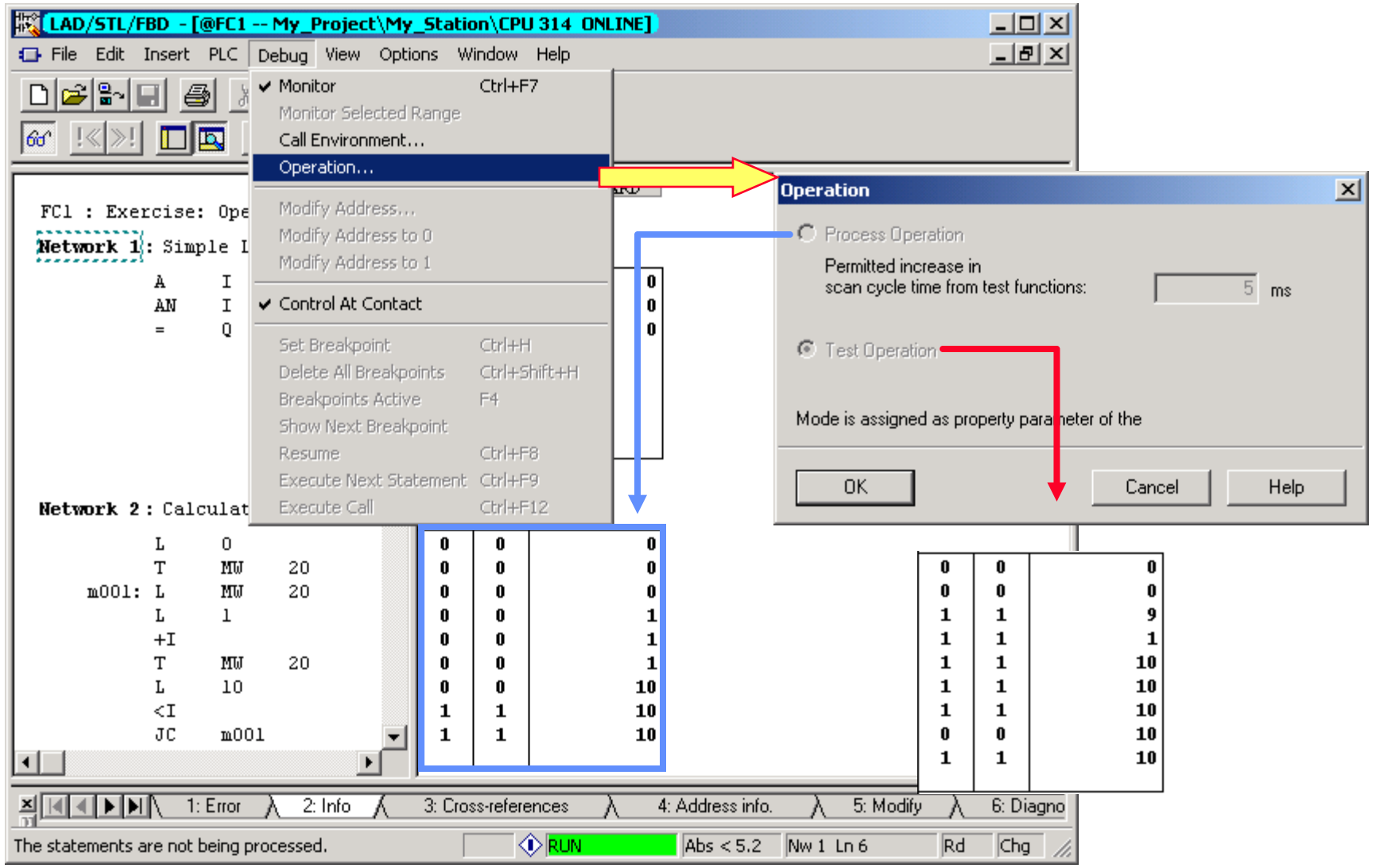


Address	Name	Type	Initial value	Actual value	Comment
0.0	Act_quant	INT	0	16495	Quantity of transported parts
2.0	Setp_quant	INT	0	0	Quantity of parts to be transported
4.0	aux_bit	BOOL	FALSE	TRUE	Auxiliary bit for edge detection

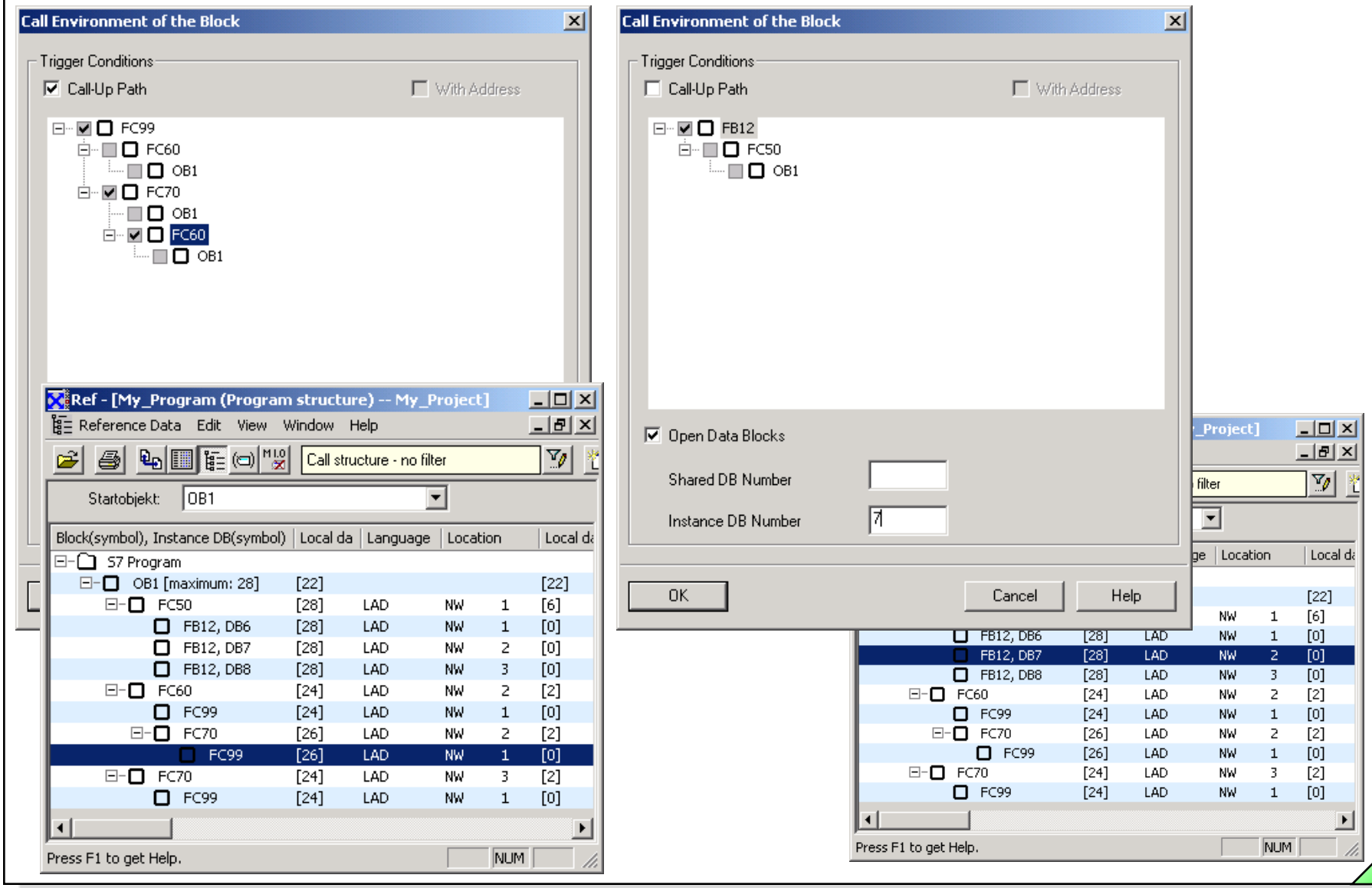
If You Want to Know More

Date: 2/23/2012
File: PRO1_14E.38

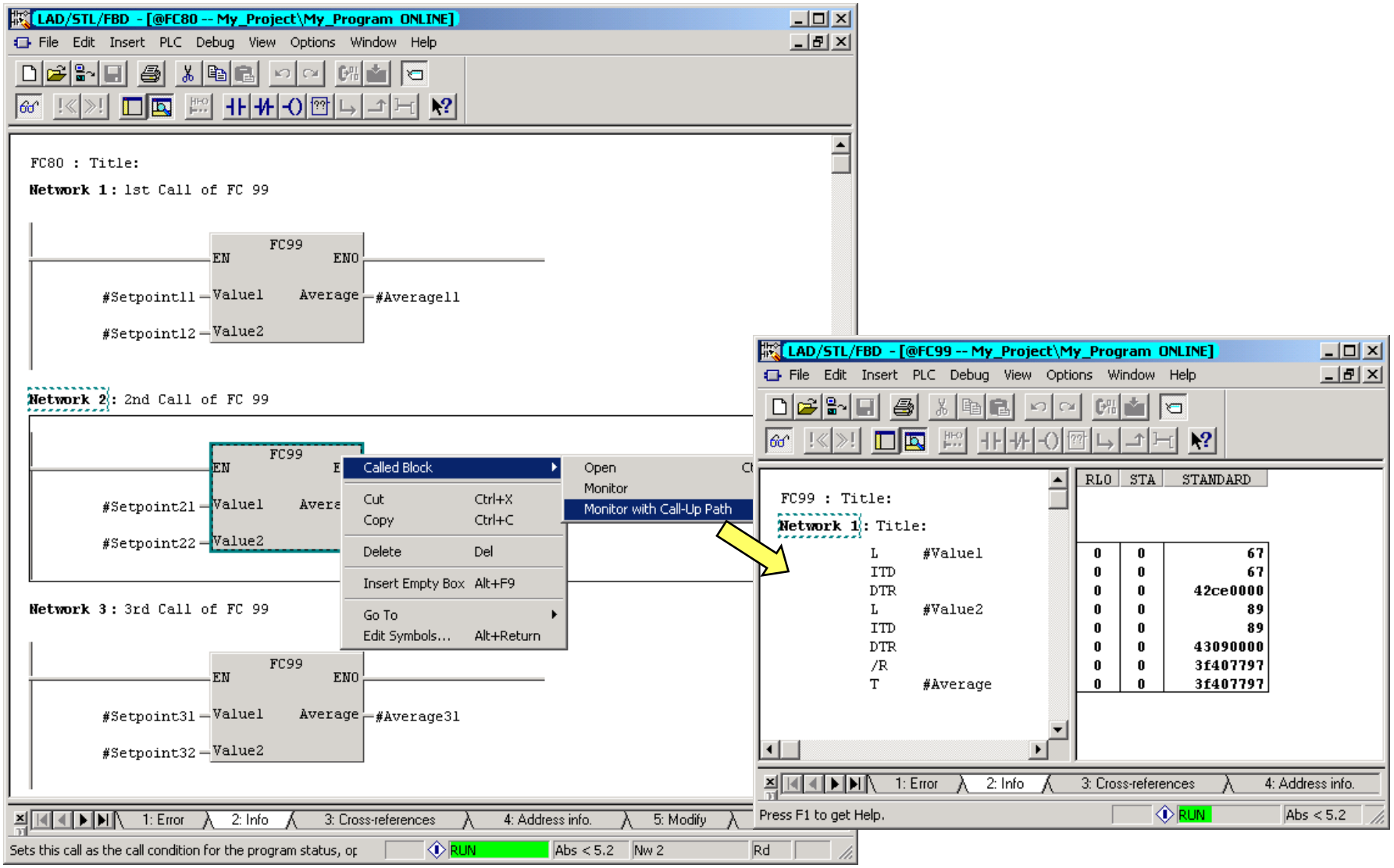
Selecting Process and Test Operation



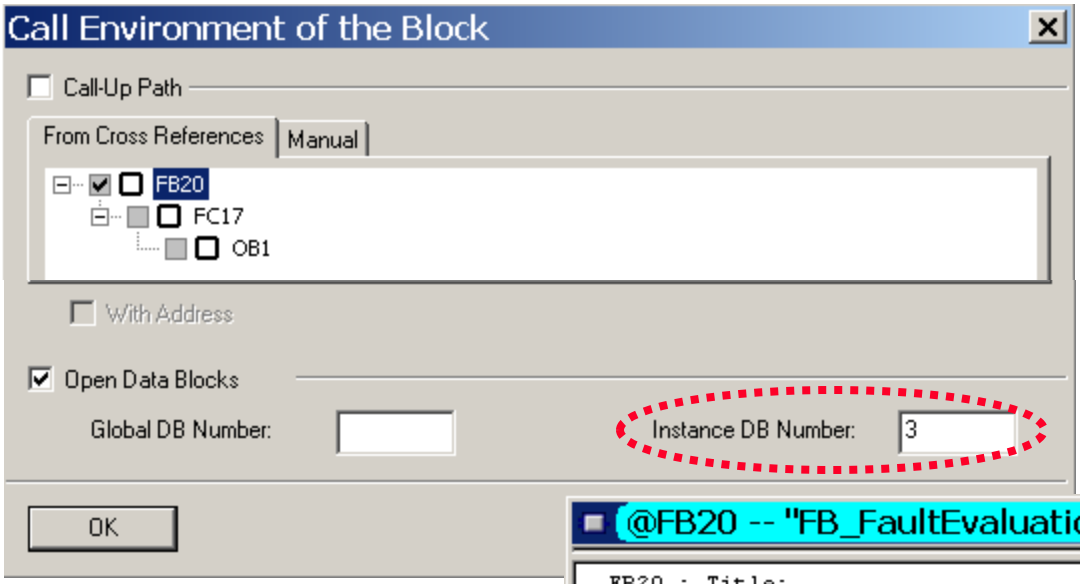
Trigger Conditions for Block Monitoring (1)

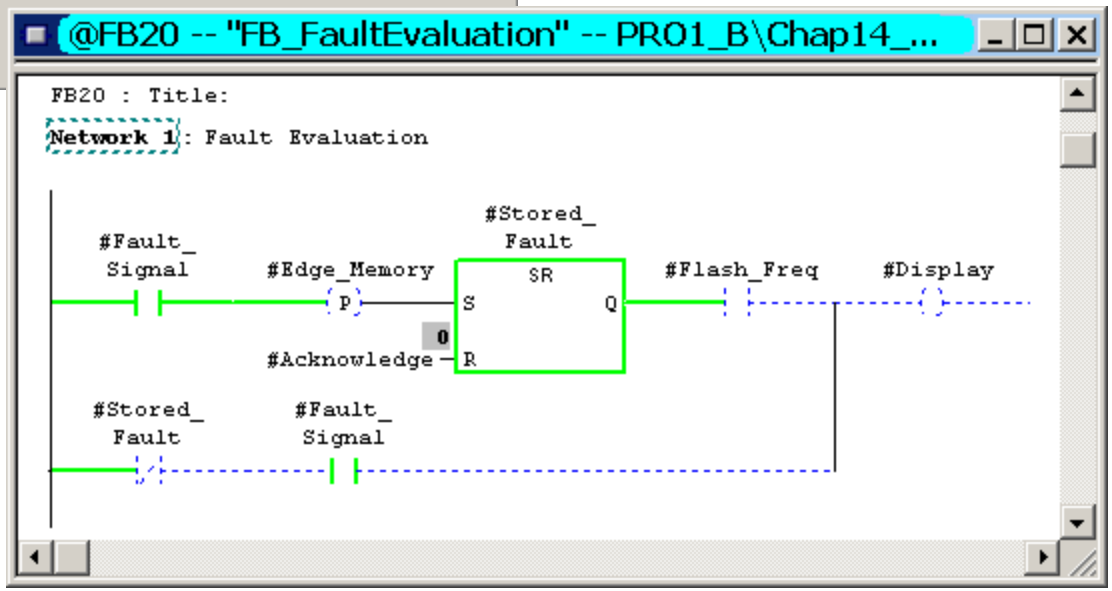


Trigger Conditions for Block Monitoring (2)

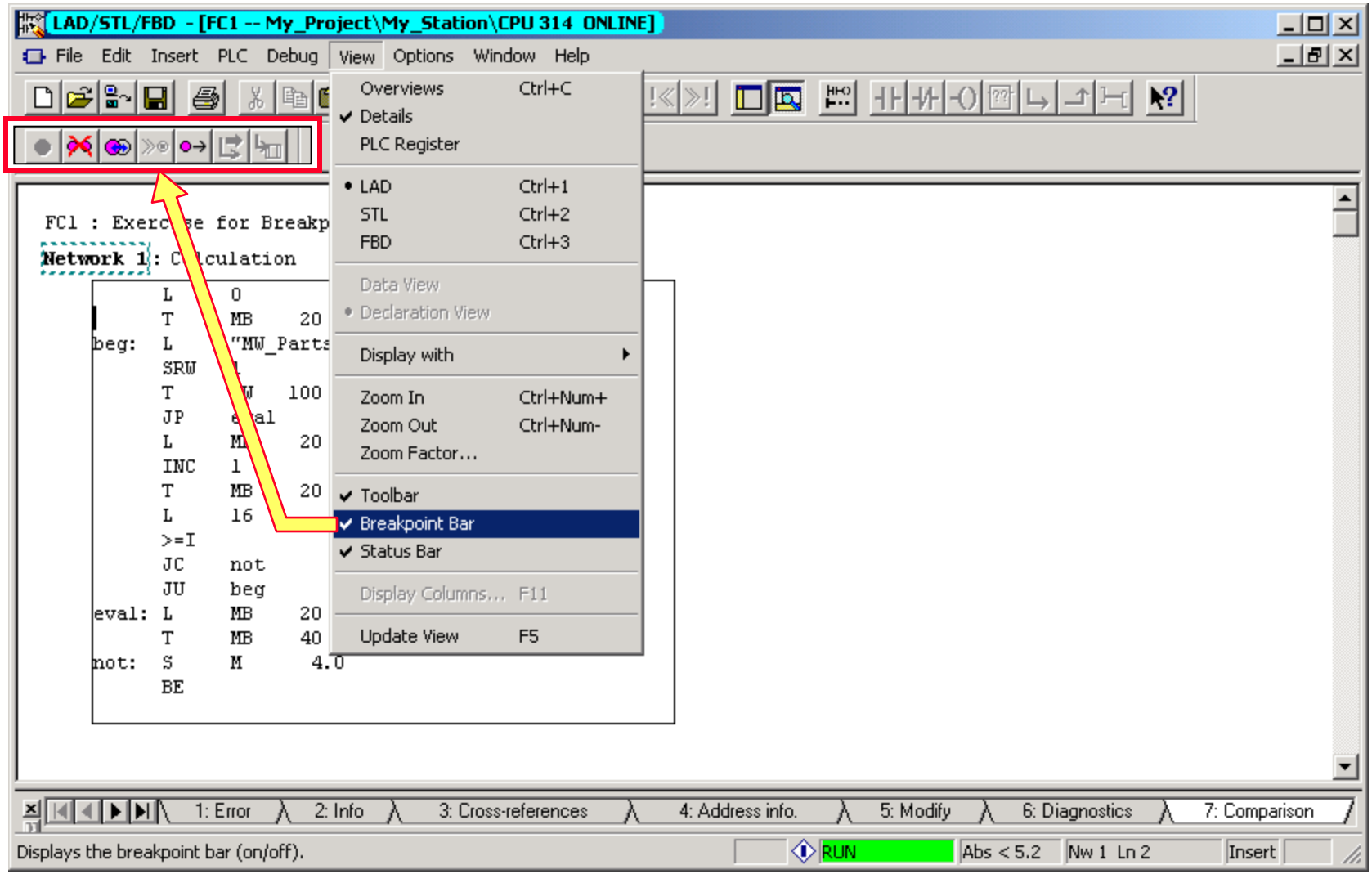


Exercise 5: Testing the Evaluation of Fault 4

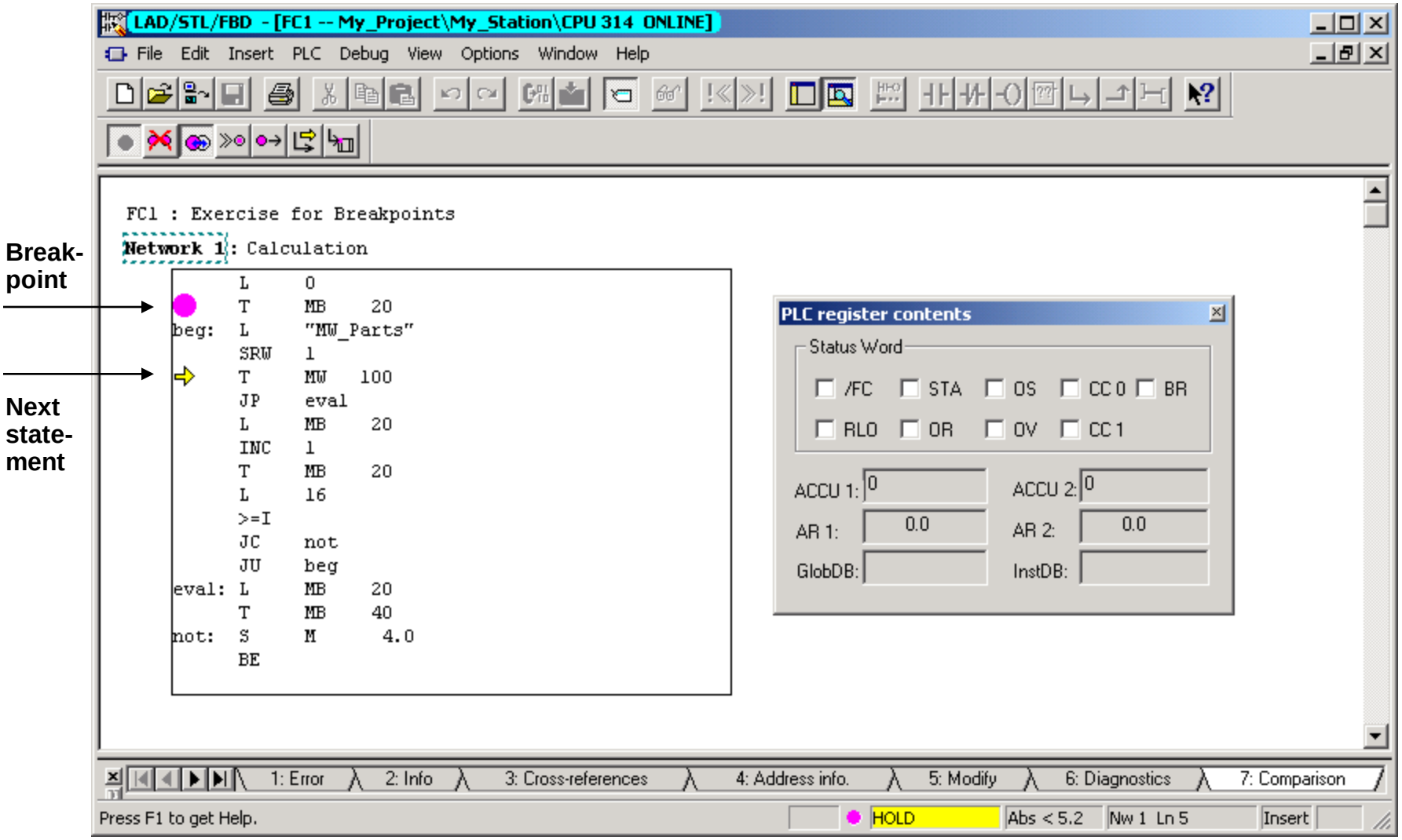




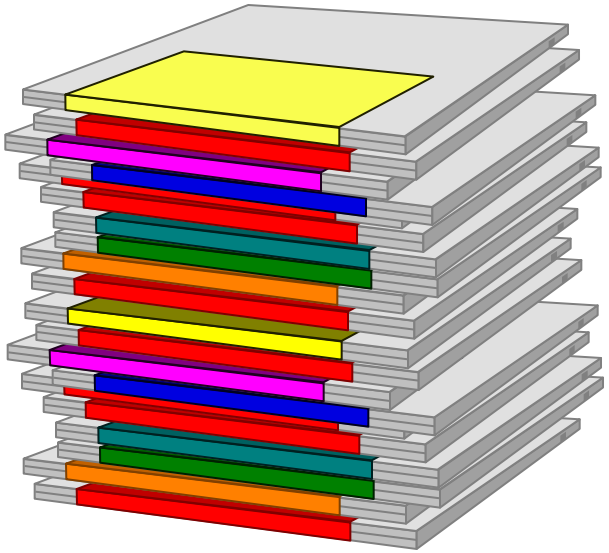
Testing the Program Execution using Breakpoints (Part 1)

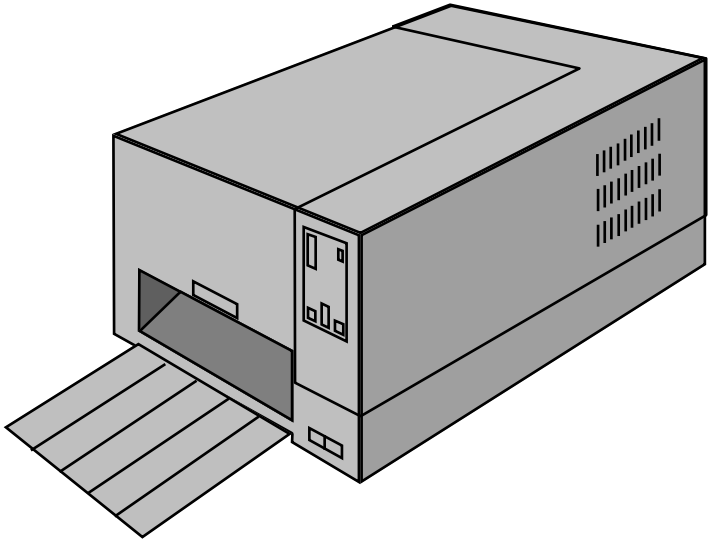


Testing the Program Execution using Breakpoints (Part 2)



Documenting, Saving, Archiving





Objectives

Upon completion of the chapter the participant will ...

... be familiar with the documentation possibilities of blocks

... understand the "Managing multilingual project" function

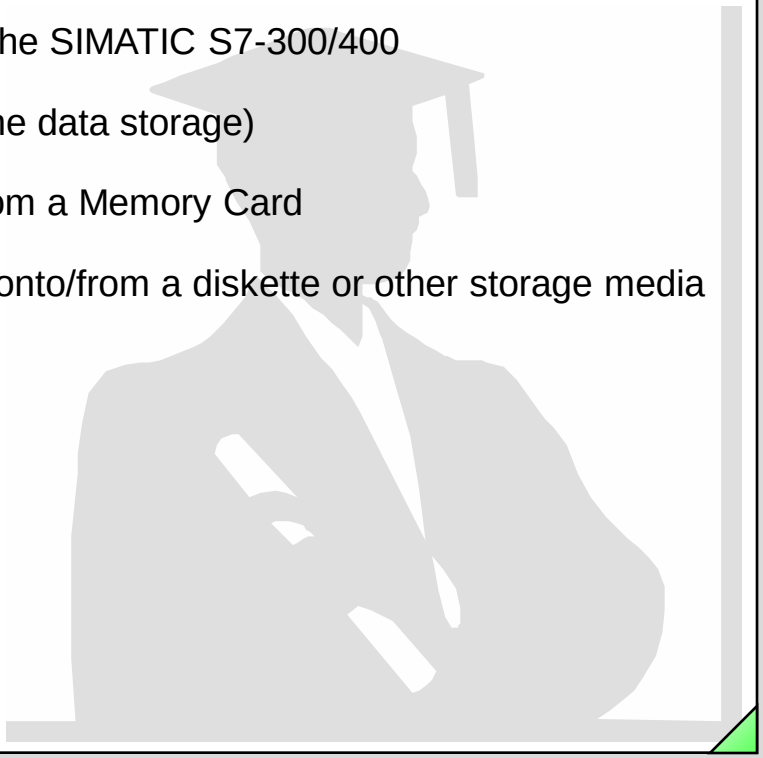
... be able to print out programs

... understand the memory concept of the SIMATIC S7-300/400

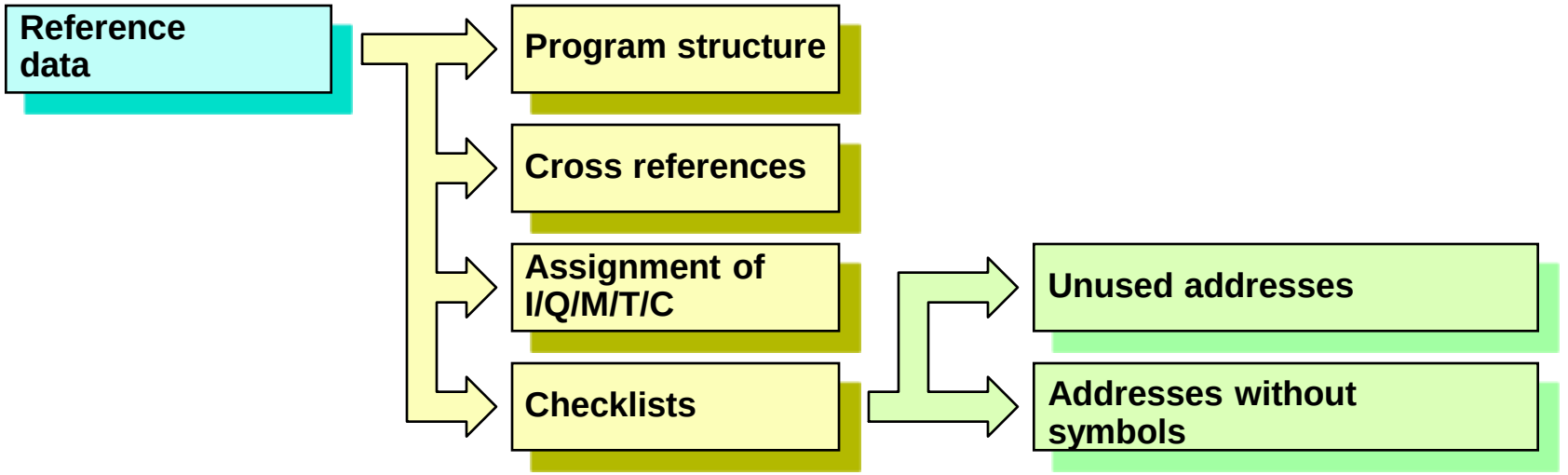
... be able to make a "PLC Copy" (online data storage)

... be able to load/read a program to/from a Memory Card

... be able to achrive/retrieve a project onto/from a diskette or other storage media

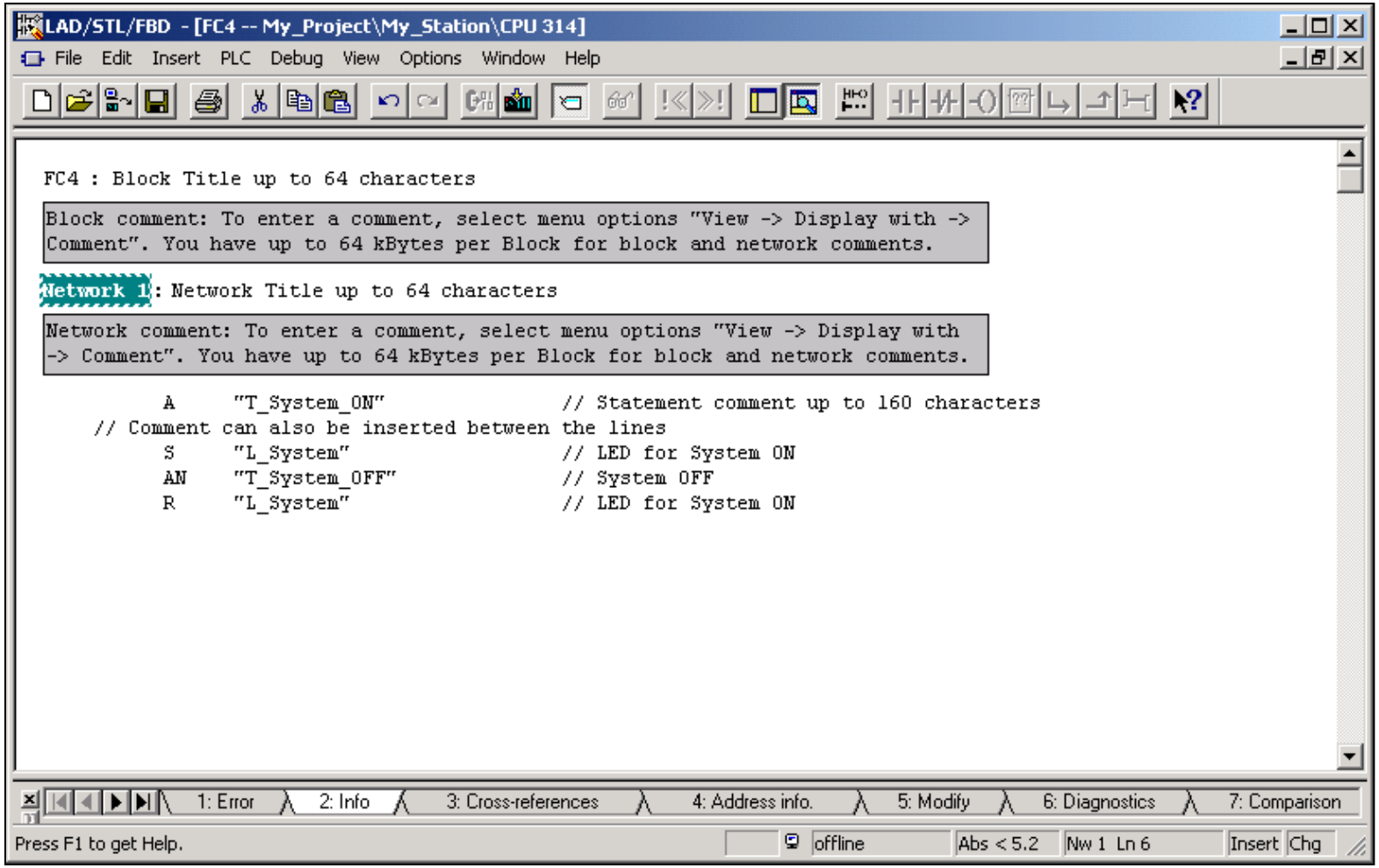


Other Documentation Tools

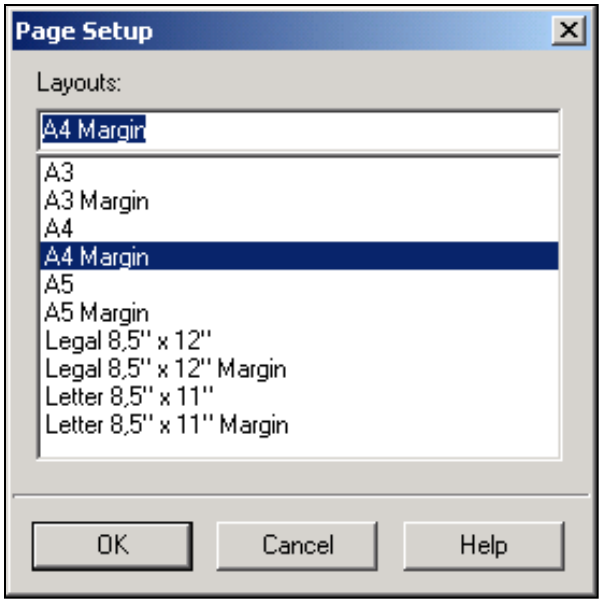


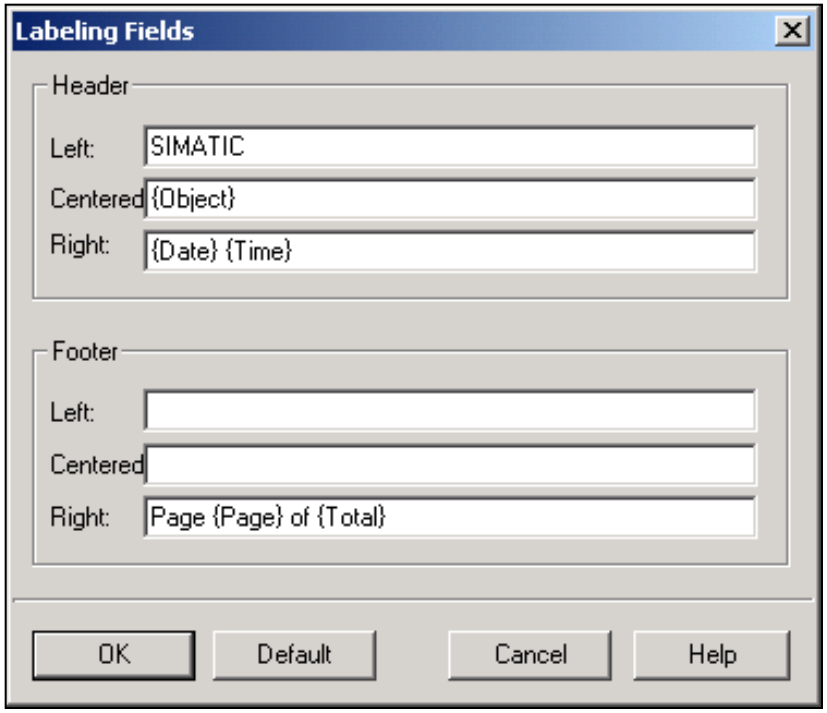
- Symbol table
- Configuration
- Network configuration

Block Documentation



Page Setup





Print Preview

SIMATIC My Project\My Station\CPU 314\...\FC20 - <offline> 01/22/2003 9:20:24

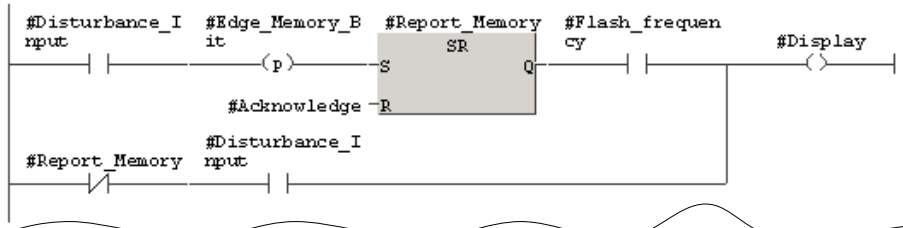
FC20 - <offline>

"FC_Fault" Evaluation of Faults no memory
Name: **Family:**
Author: **Version:** 0.1
Block version: 2
Time stamp Code: 10.01.2002 09:17:22
Interface: 12.07.2000 14:24:45
Lengths (block/logic/data): 00152 00048 00000

Name	Data Type	Address	Initial Value	Comment
IN		0.0		
Disturbance_Input	Bool	0.0	FALSE	
Acknowledge	Bool	0.1	FALSE	
Flash_frequency	Bool	0.2	FALSE	
OUT		2.0		
Display	Bool	2.0	FALSE	
IN_OUT		4.0		
Report_Memory	Bool	4.0	FALSE	
Edge_Memory_Bit	Bool	4.1	FALSE	
TEMP		0.0		
RETURN		0.0		
RET_VAL		0.0		

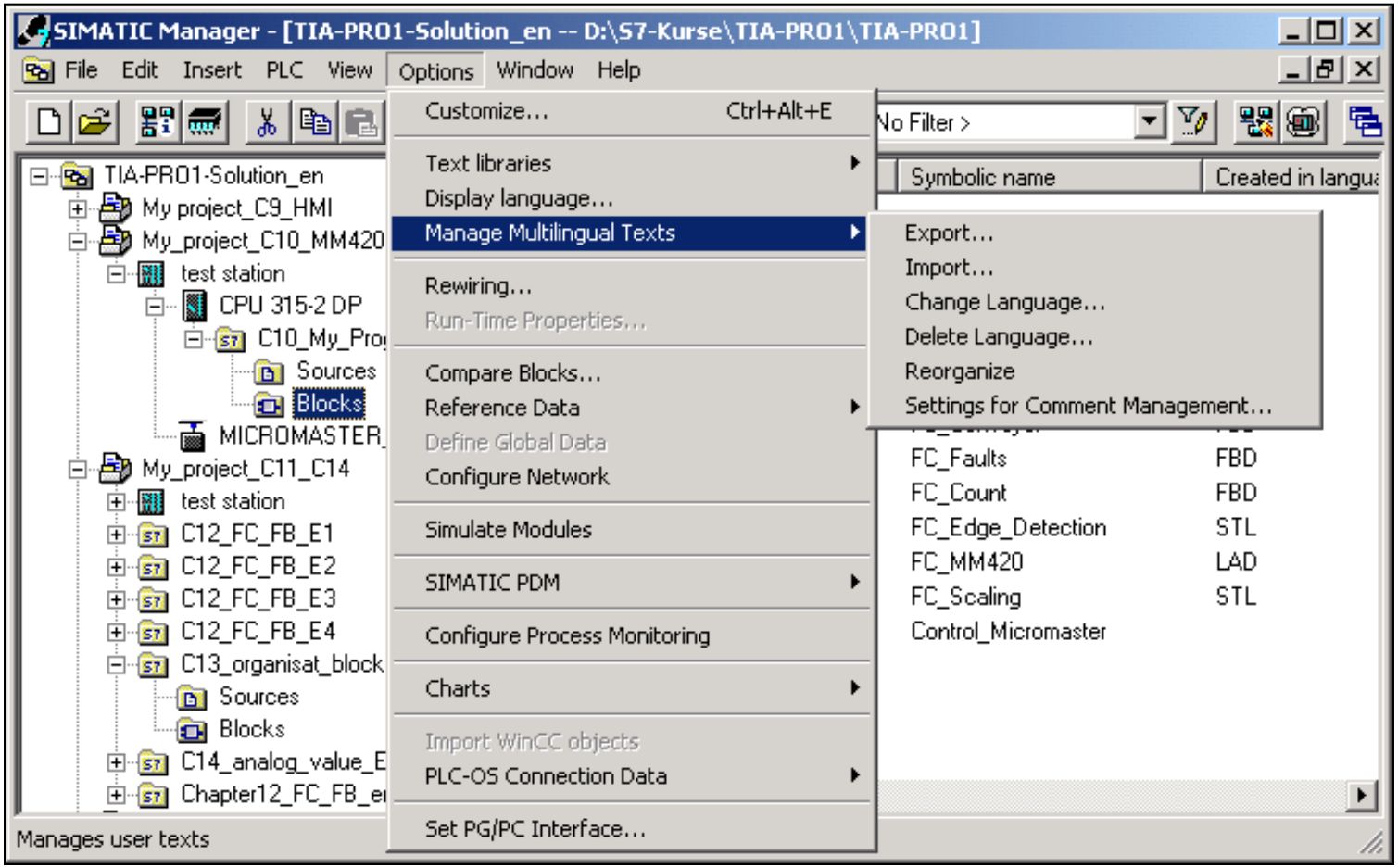
Block: FC20

Network: 1 Disturbance evaluation



Page 1 of 1

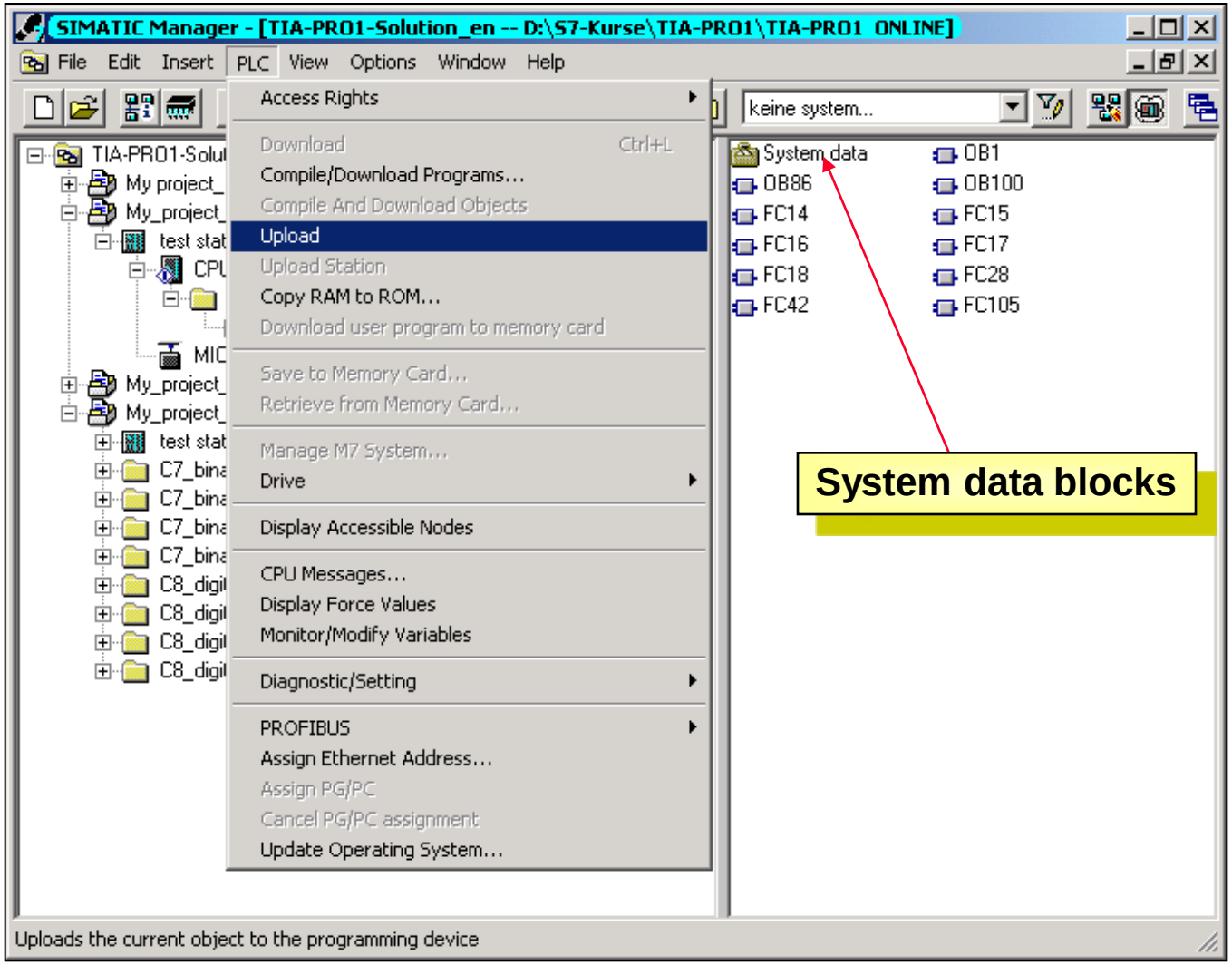
Managing Multilingual Project Documentation



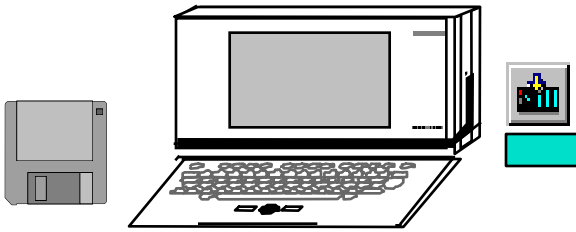
Overview: Saving User Data

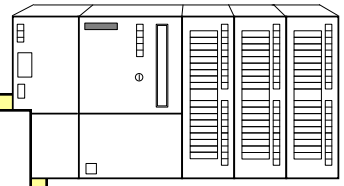
Uploading program from the CPU to PG	1. Create a new S7 program in the SIMATIC Manager 2. Switch to the online view 3. Open the new S7 program and select Blocks folder 4. SIMATIC Manager → PLC → Upload
Upload station in PG	• SIMATIC Manager → PLC → Upload Station
Load program from PG to Memory Card... **... inserted in PG/PC** **or** **... Inserted in CPU**	1. Open two windows in the SIMATIC Manager: "Blocks folder of the S7 program" and "S7 Memory Card" 2a. Use drag & drop to copy blocks to the "S7 Memory Card" 2b. SIMATIC Manager → PLC → Download user program to Memory Card
Copy program from CPU to Memory Card (only S7-300)	• SIMATIC Manager → PLC → Copy RAM to ROM...
Project archiving on Memory Card/MMC (S7-400, S7-300 after 10.02)	1. Select the CPU on whose Memory Card the project data are to be saved 2. SIMATIC Manager → PLC → Save Project on Memory Card
Project archiving on diskette	1. SIMATIC Manager → File → Archive 2. Select project to be archived 3. Specify name and storage path of the archive file and start function 4. In the Windows Explorer, copy archive file onto diskette

Uploading a Program from the CPU to the PG



Memory Concept of the S7-300 until Oct. 2002

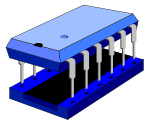




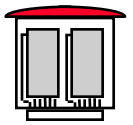
Load memory:

Blocks:
- Logic blocks (OB,FC,FB)
- Data blocks (DB)

Additional info



RAM



Flash EPROM

Working memory:
- OB,FC,FB
- DB

n. reten.	reten.

System memory:
- PII,PIQ
- M, T, C

n. reten.	reten.

Retentive memory:
- Retentive M, T, C
- Retent. Data blocks

With Power ON without battery backup

With Power OFF without battery backup

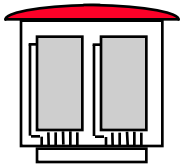
Comments

Symbols

Blocks:
- Logic blocks (OB,FC,FB)
- Data blocks (DB)

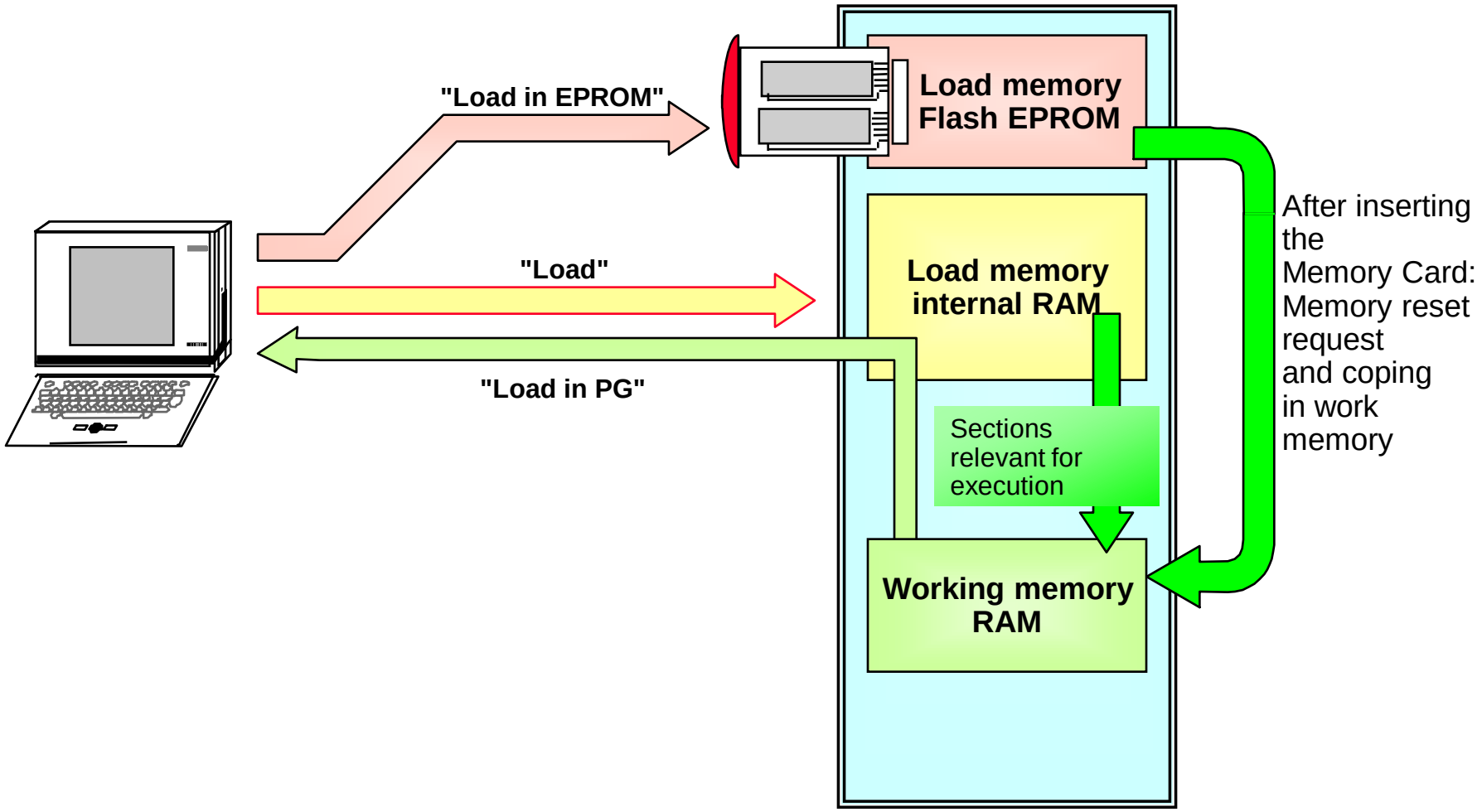
Blocks:
- Logic blocks (OB,FC,FB)
- Data blocks (DB)

Additional info

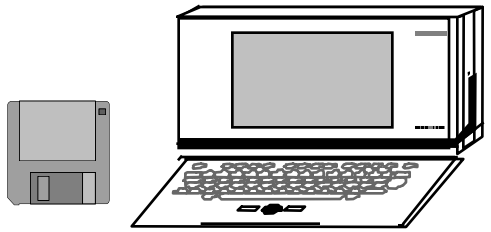


Flash EPROM Memory Card in PG (inserted later in the CPU)

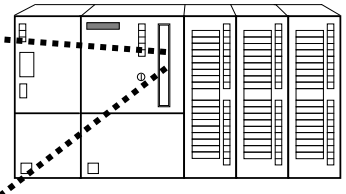
Loading Blocks into / out of a Flash EPROM Memory Card



Memory Concept of the S7-300 after Oct. 2002



Micro Memory Card (CPU load memory)
Blocks:
- Logic blocks (OB,FC,FB)
- Data blocks (DB)
- System data



Load

Working memory:
Sections of the
- logic blocks
- data blocks relevant for execution

System memory:
- PII, PIQ
- Local data
- M, T, C

retentive
n. retent.

Power OFF

Power OFF

Warm restart after Power ON

Comments

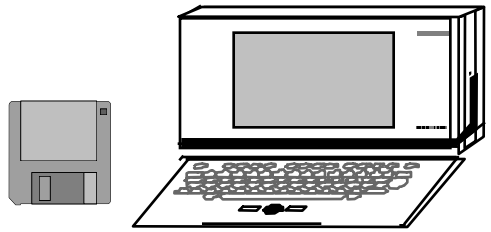
Symbols

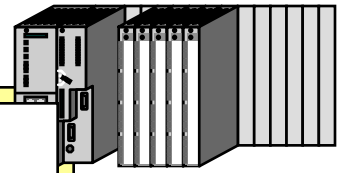
Blocks:
- Logic blocks (OB,FC,FB)
- Data blocks (DB)

Blocks:
- Logic blocks (OB,FC,FB)
- Data blocks (DB)
- complete project, if desired

Micro Memory Card in the PG (inserted later in the CPU)

Memory Concept of the S7-400





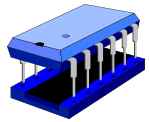
Load memory:

Blocks:
- Logic blocks (OB,FC,FB)
- Data blocks (DB)

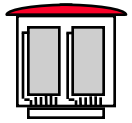
Additional info

Working memory:
- OB,FC,FB
- DB

System memory:
- PII,PIQ
- M, T, C



RAM



Flash EPROM

Backup via battery

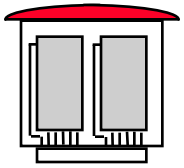
Comments

Symbols

Blocks:
- Logic blocks (OB,FC,FB)
- Data blocks (DB)

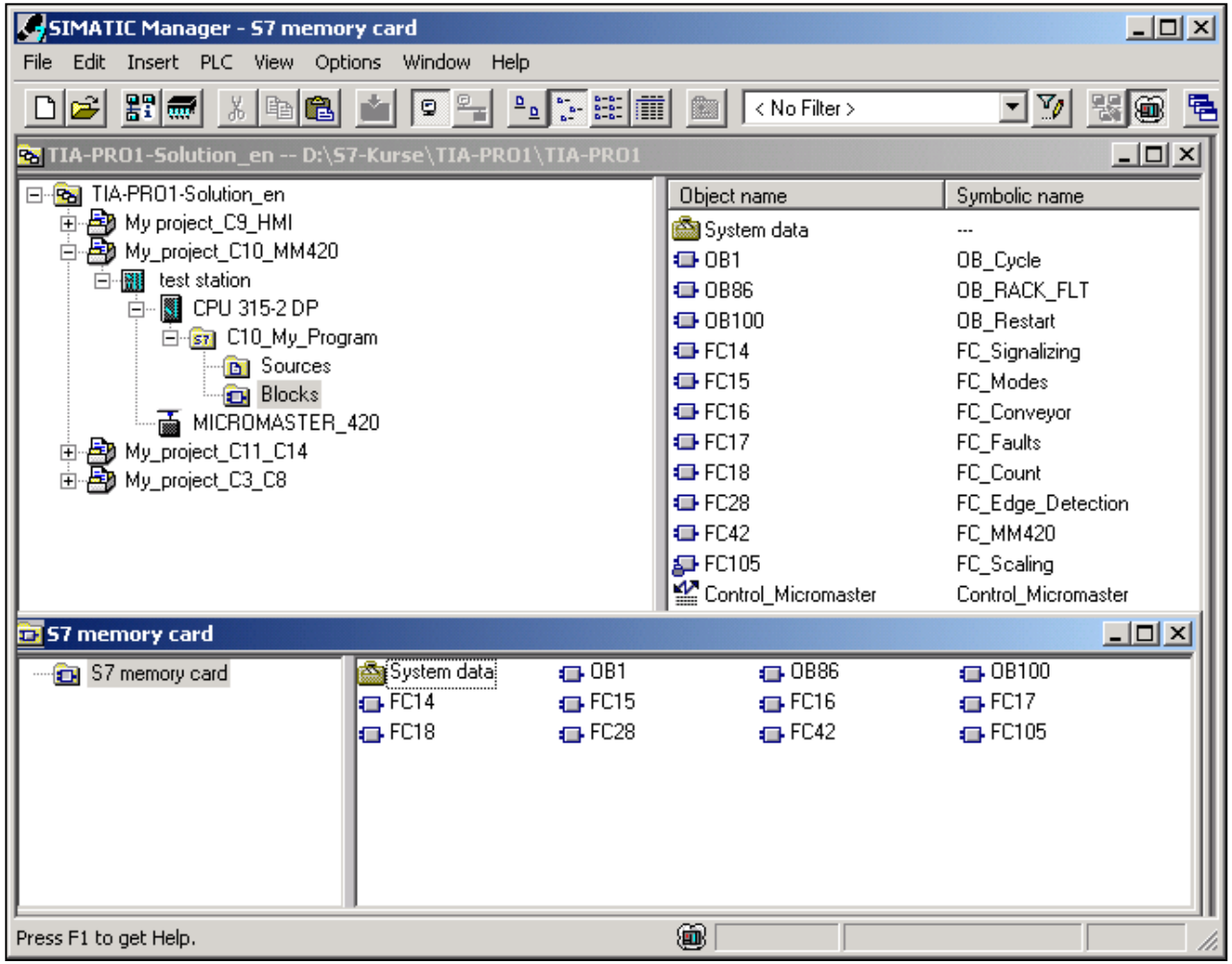
Blocks:
- Logic blocks (OB,FC,FB)
- Data blocks (DB)

Additonal info

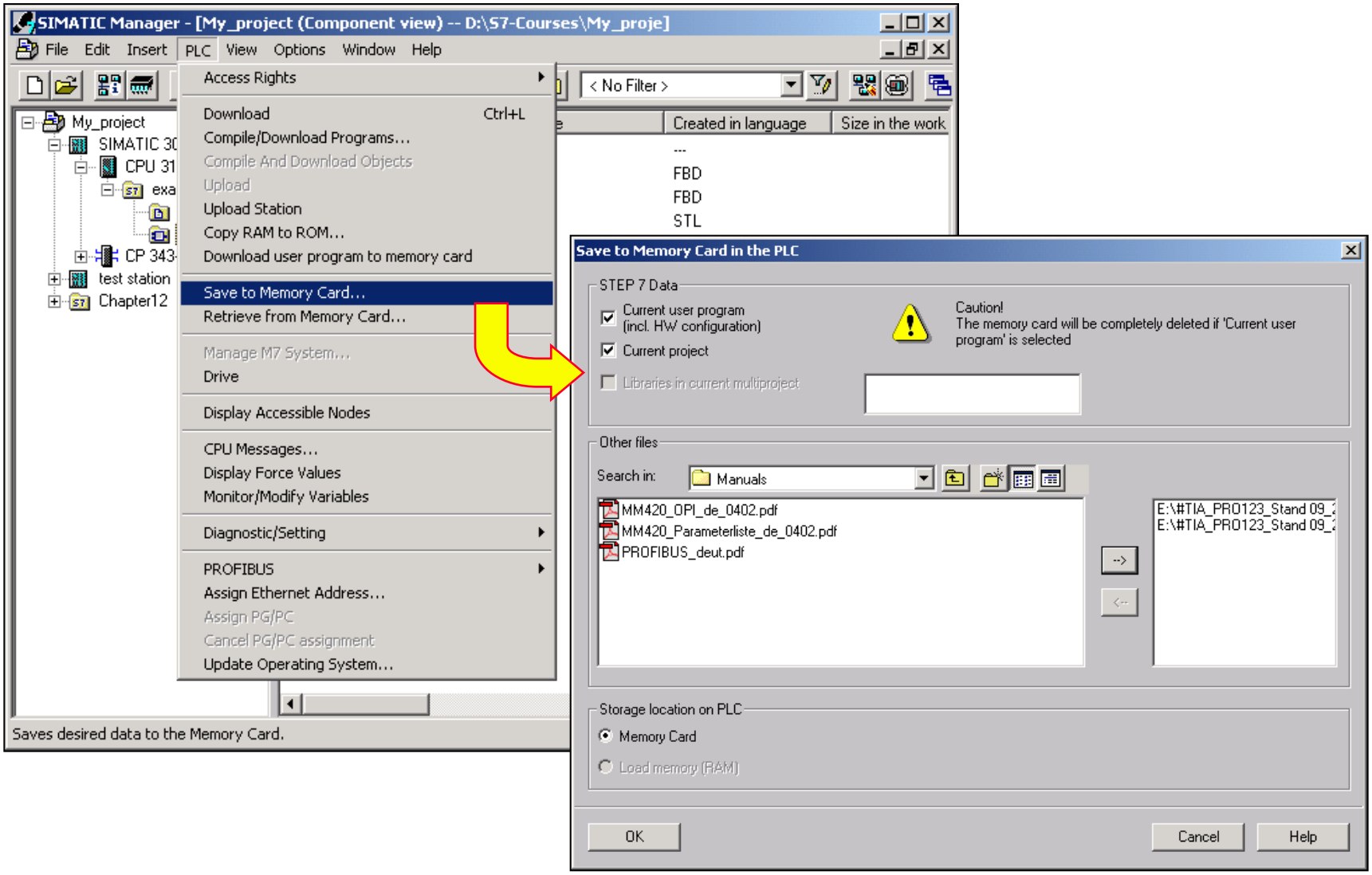


Flash EPROM Memory Card in PG (inserted later in the CPU)

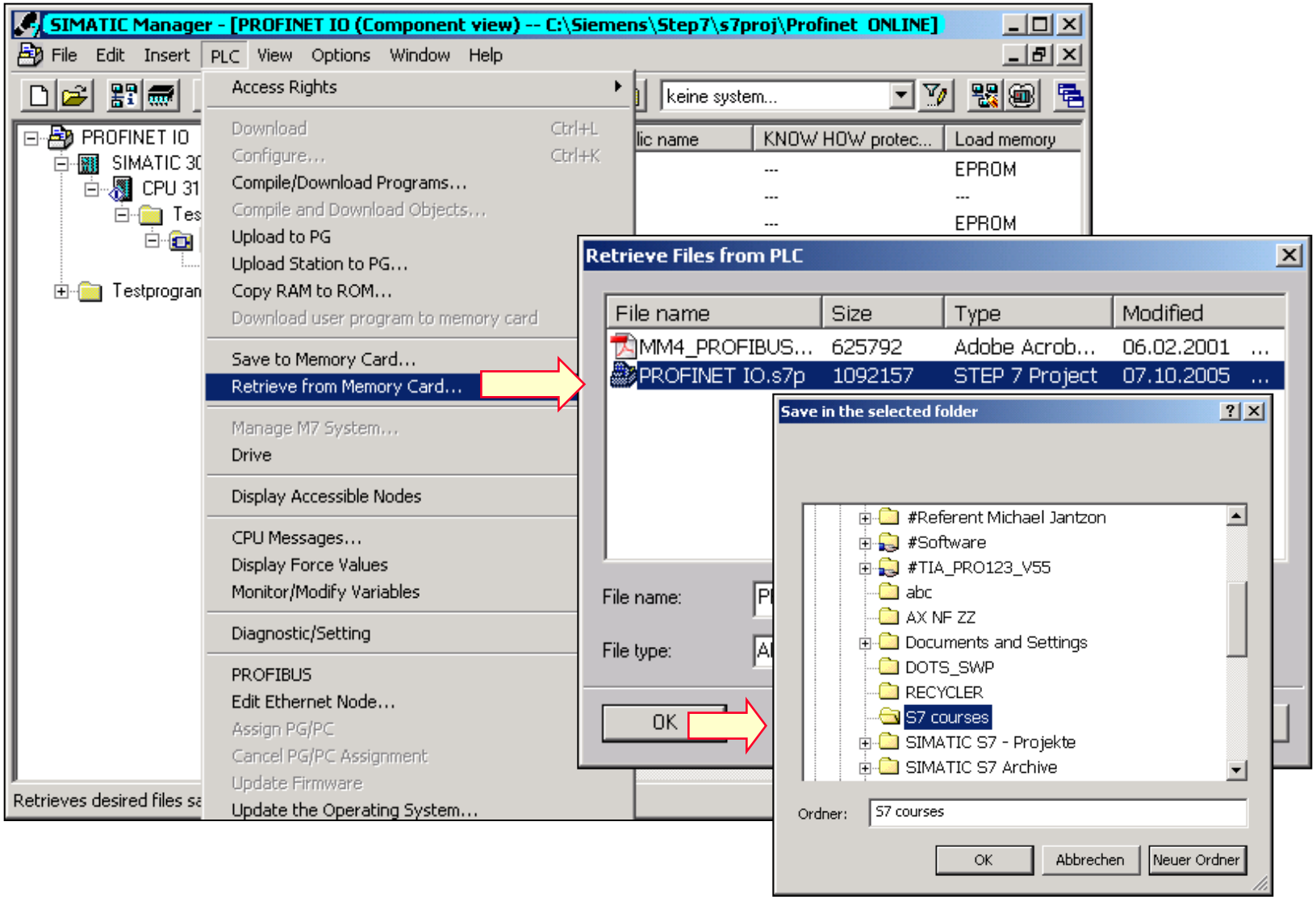
Copying a Program onto a Memory Card



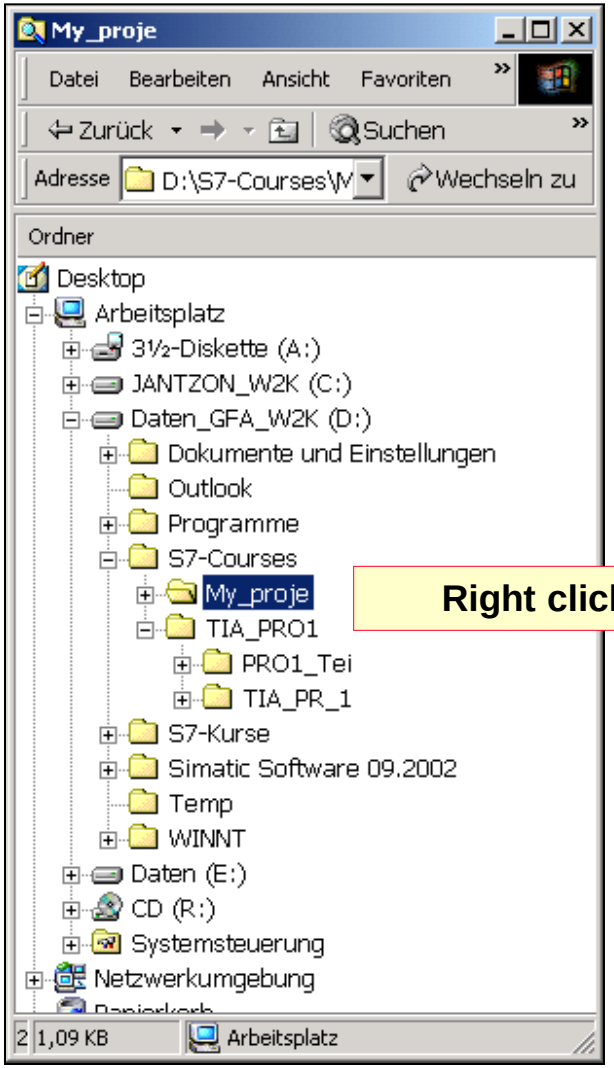
Saving a Project on a Memory Card



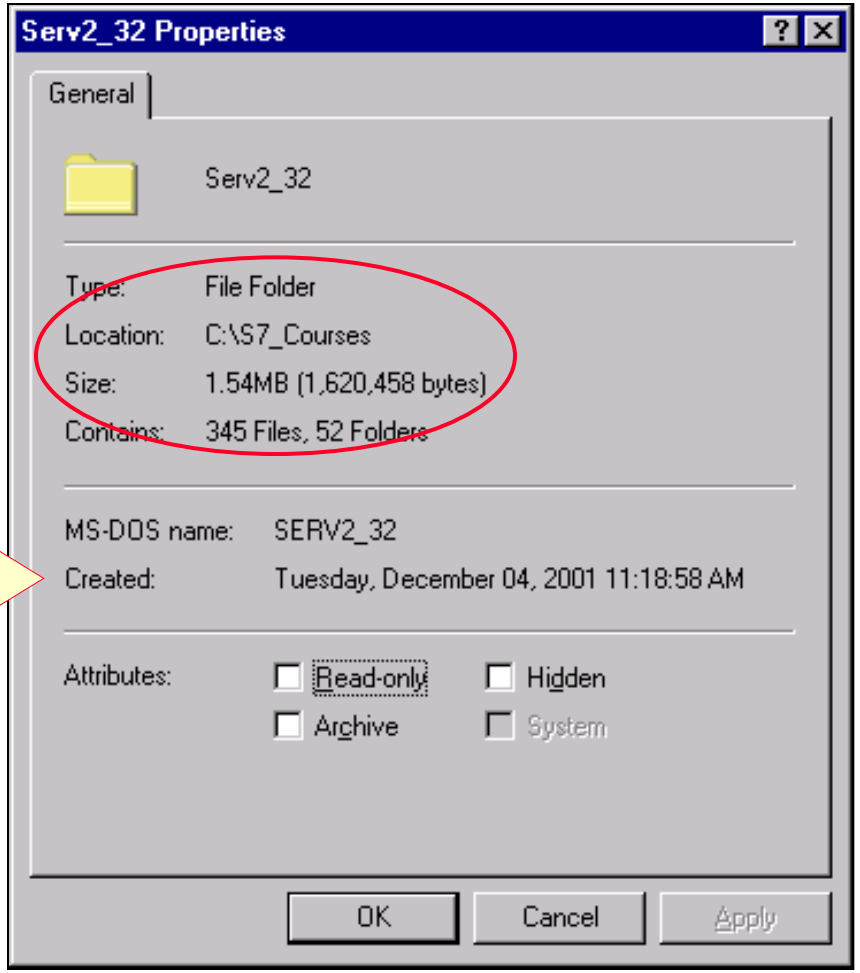
Loading a Project from the Micro Memory Card (MMC)



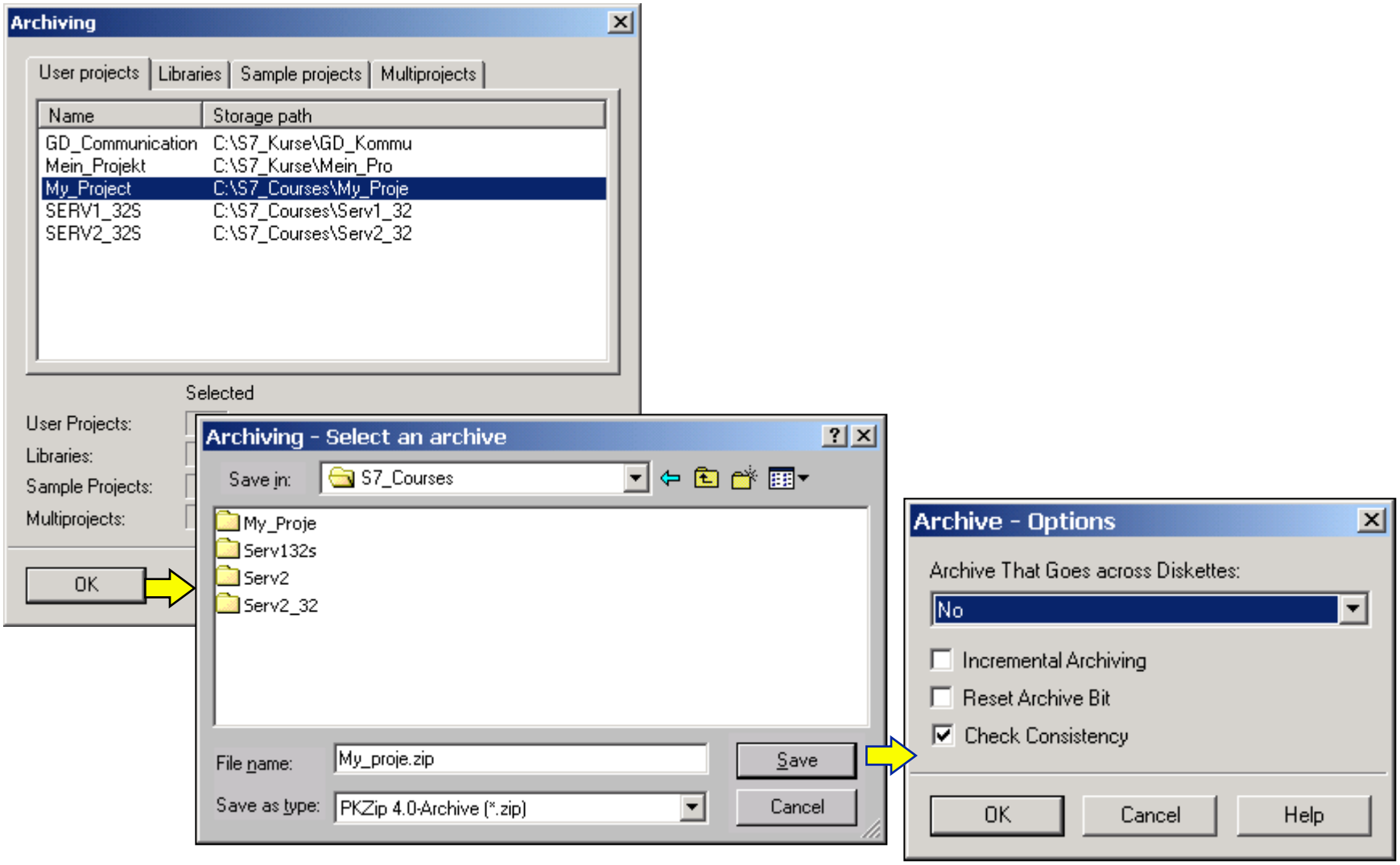
Determining the Size of a Project



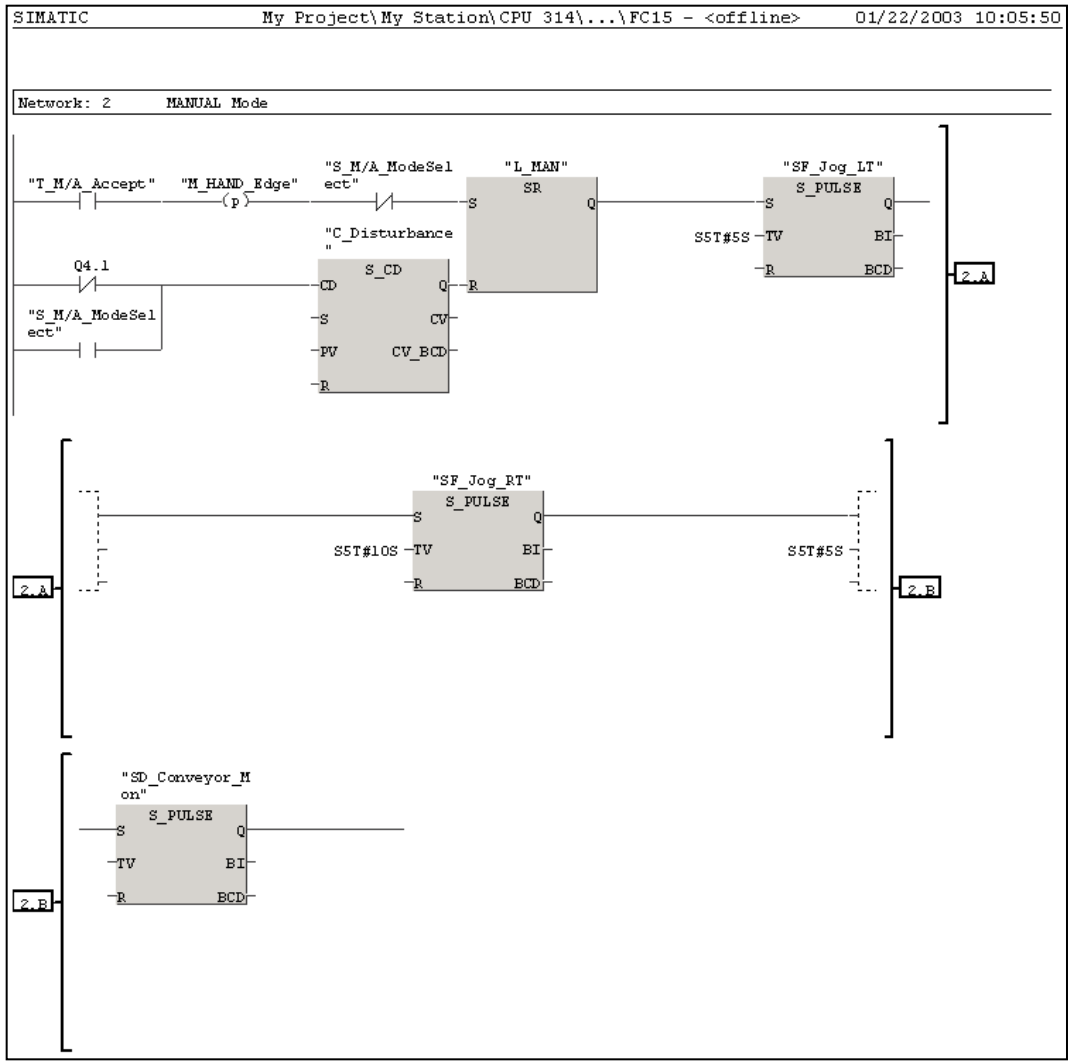
Right click



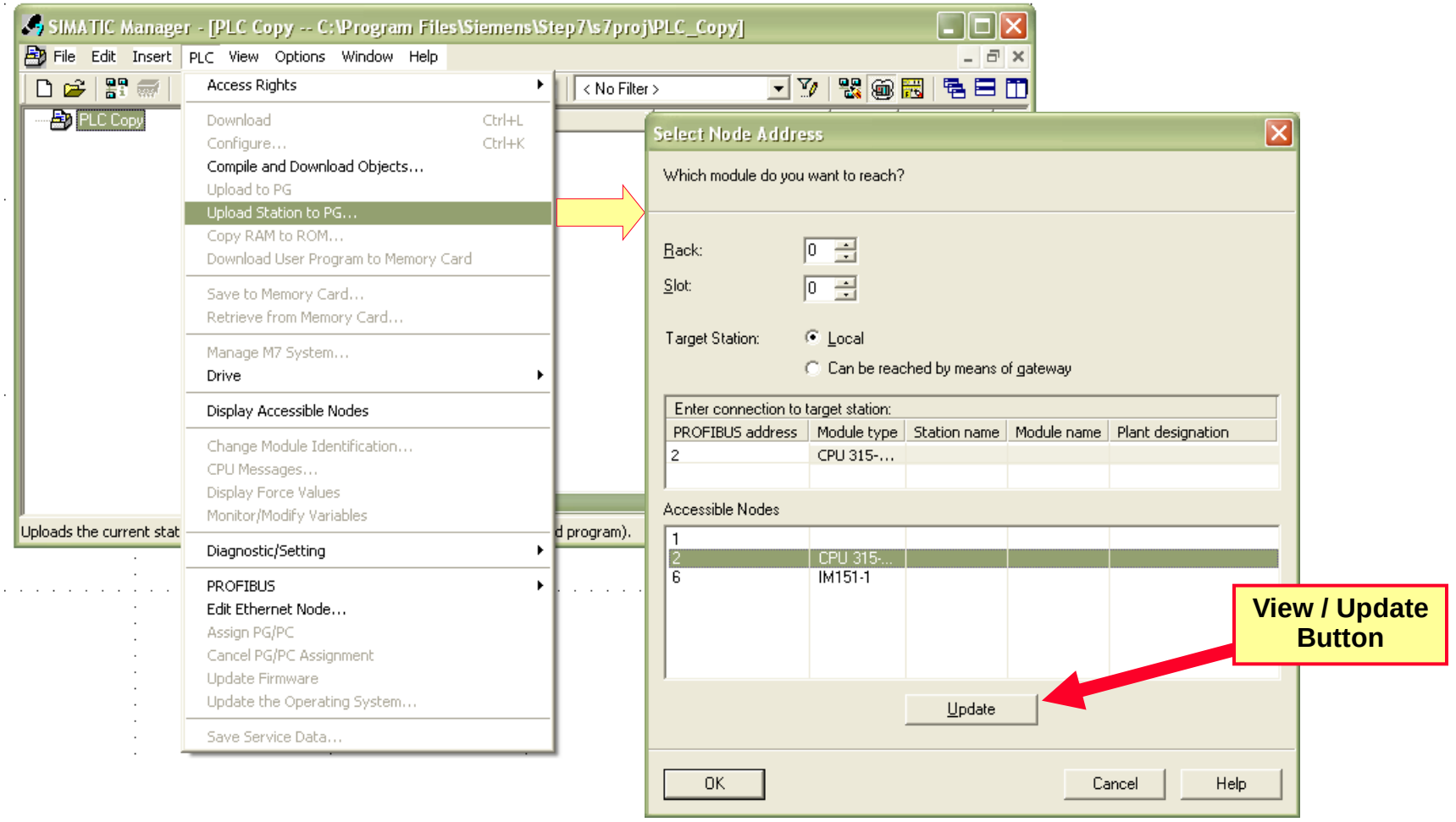
Project Archiving



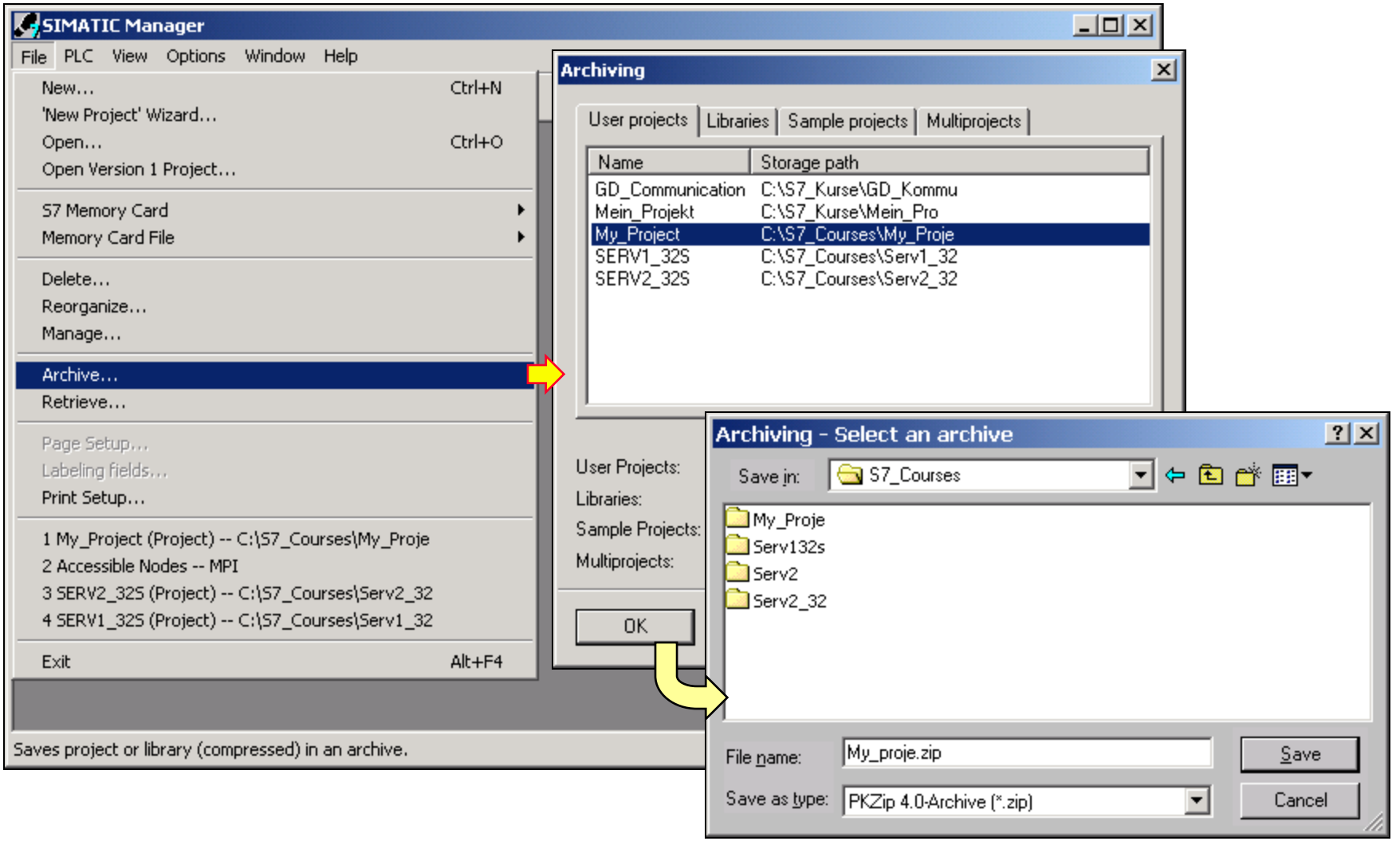
Exercise 1: Documenting a Block and Printing It Out



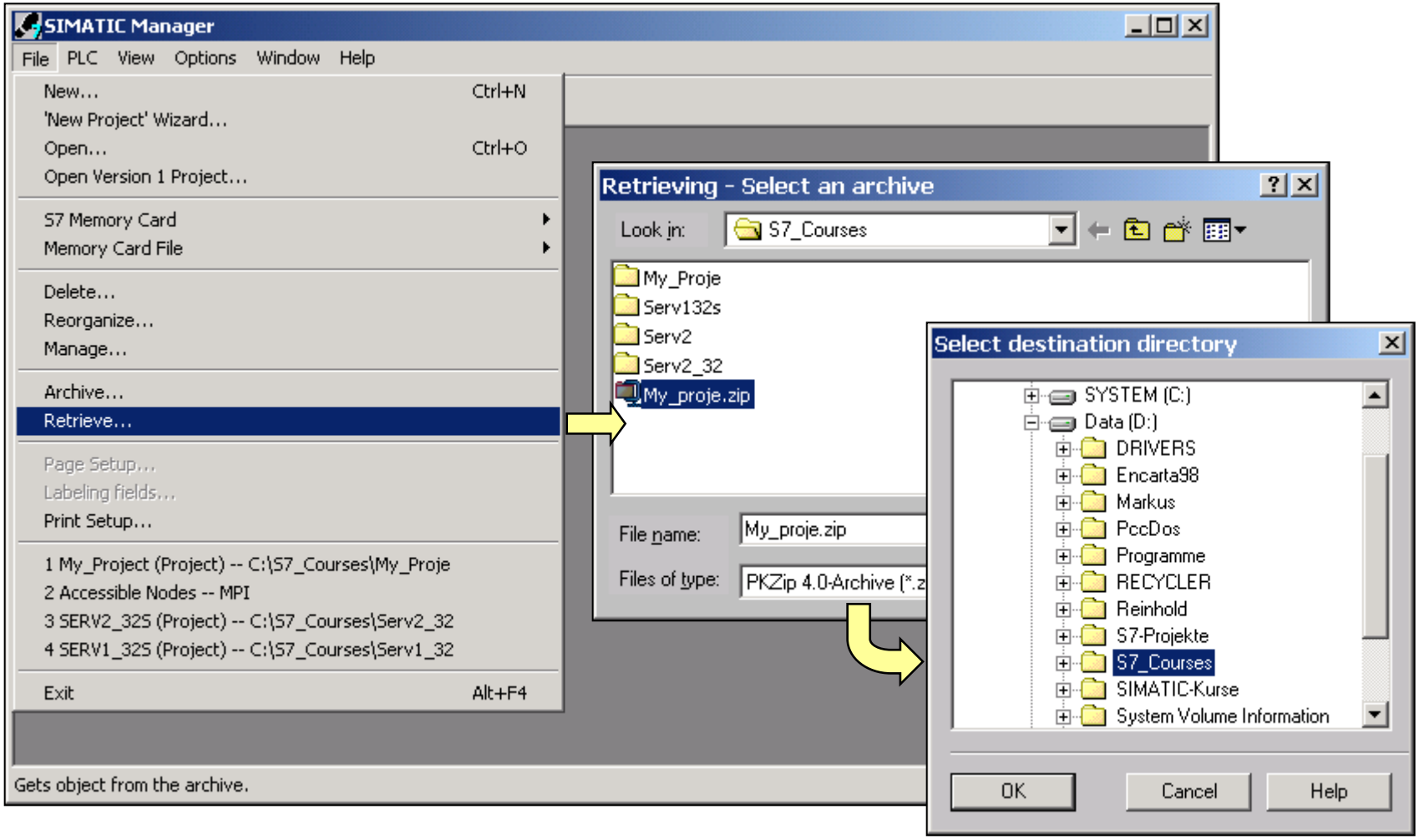
Ex.2: Saving the Program and Hardware Configuration (PLC Copy)



Exercise 3: Archiving a Project

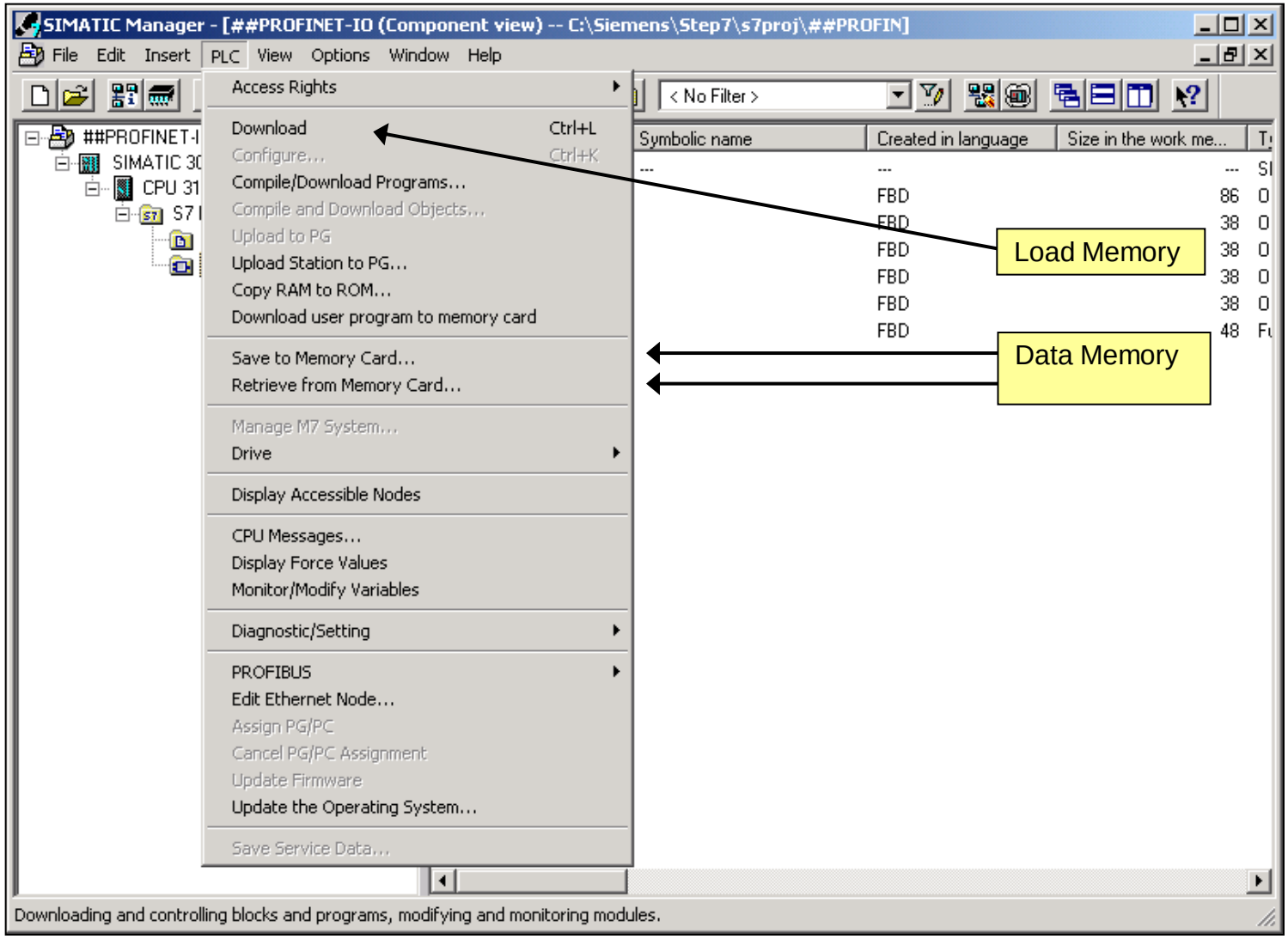


Exercise 4: Deleting a Project and Retrieving It



If You Want to Know More

MMC - as Additional Data Memory on the CPU

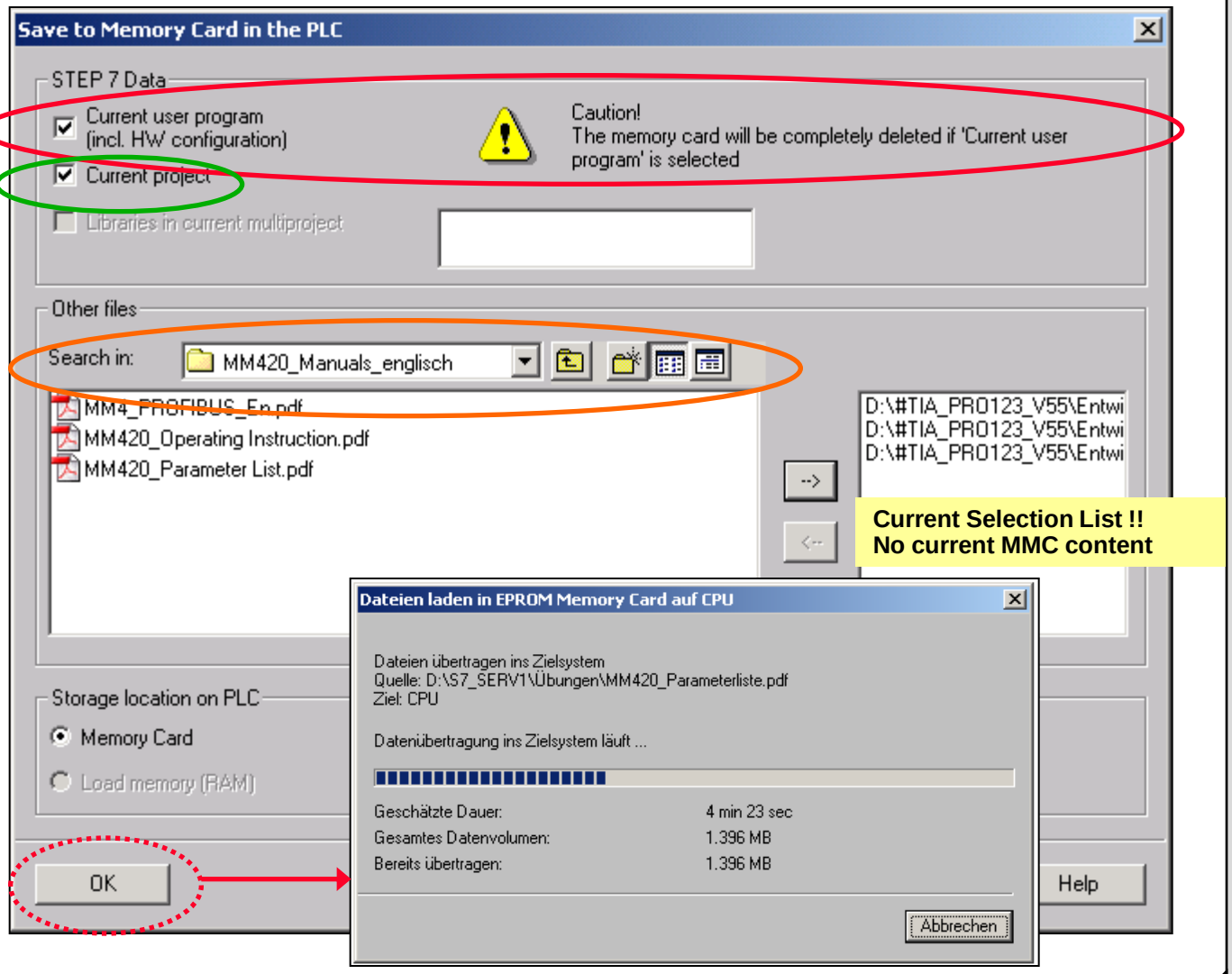


PLC: Storing Data on the Memory Card

Corresponds to "Load user program on MC"

Currently open project is compressed and copied

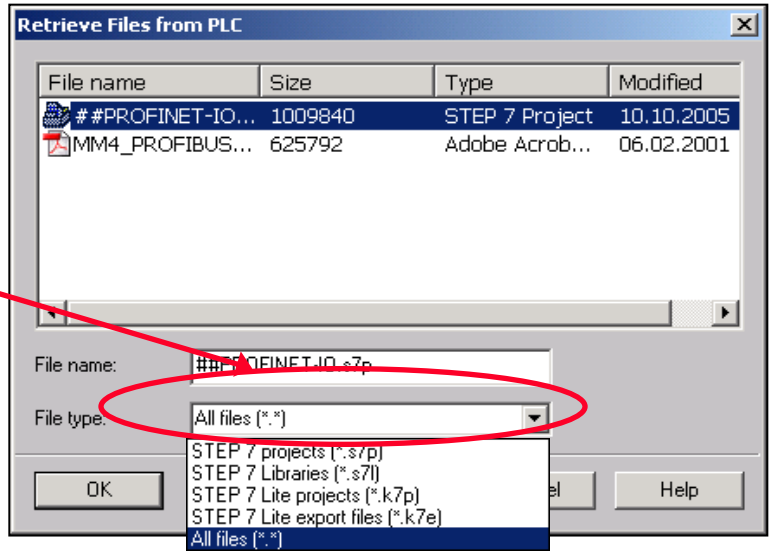
All individual files except archives (e.g. *.zip) and directories



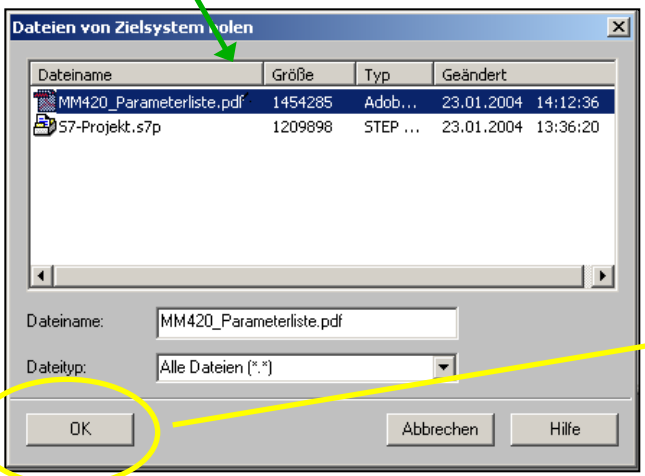
Starts transmission

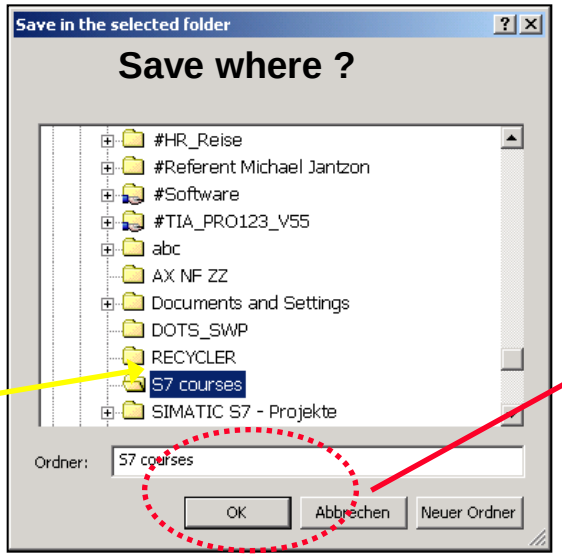
PLC: Getting Data from the Memory Card

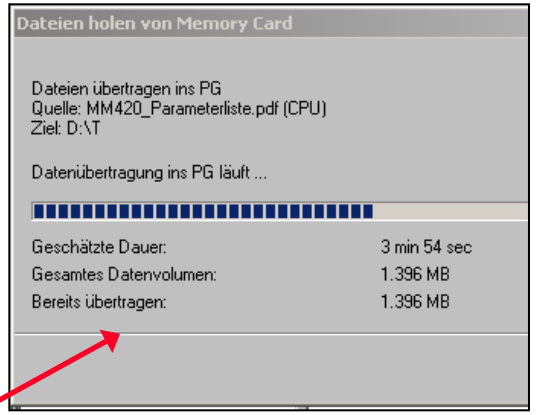
Contents are displayed
Attention! File type by default only “S7 Projects“; if necessary, set to “all files“



Select + highlight particular files





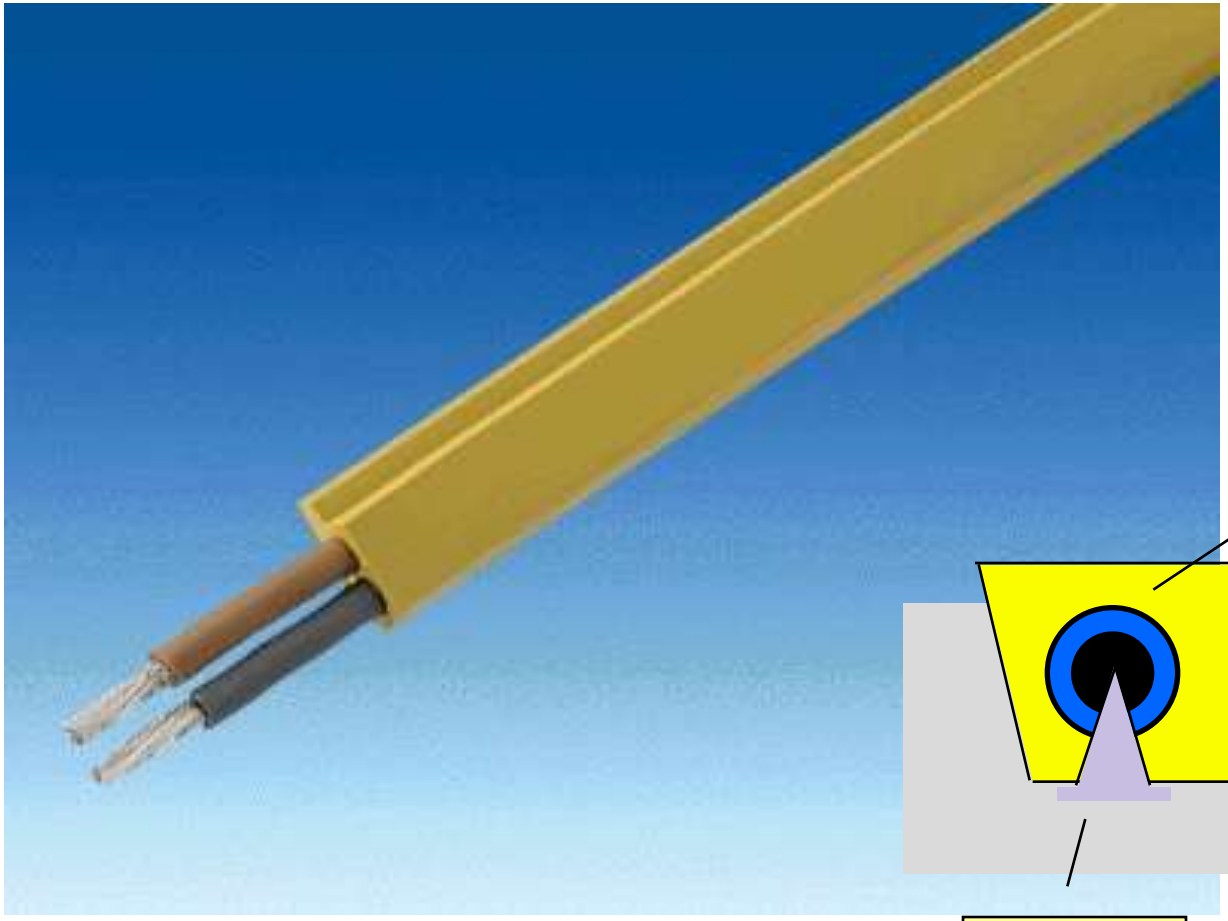


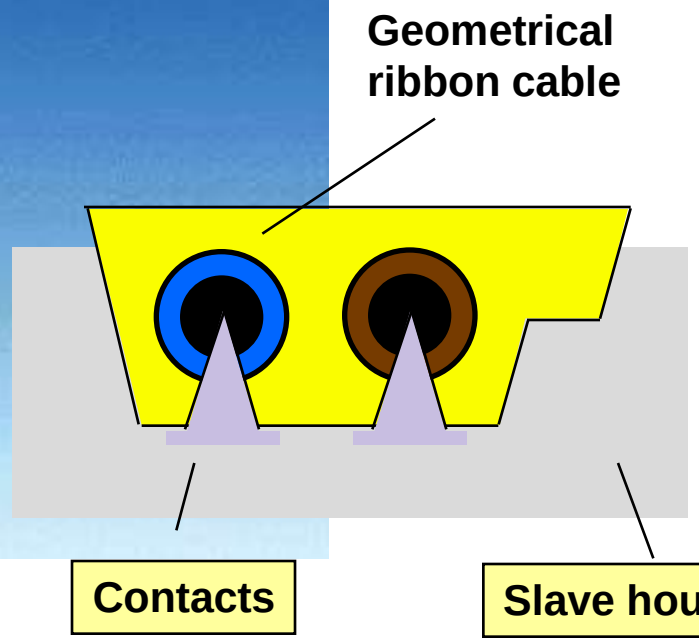
Starts transmission

AS-i Communications System

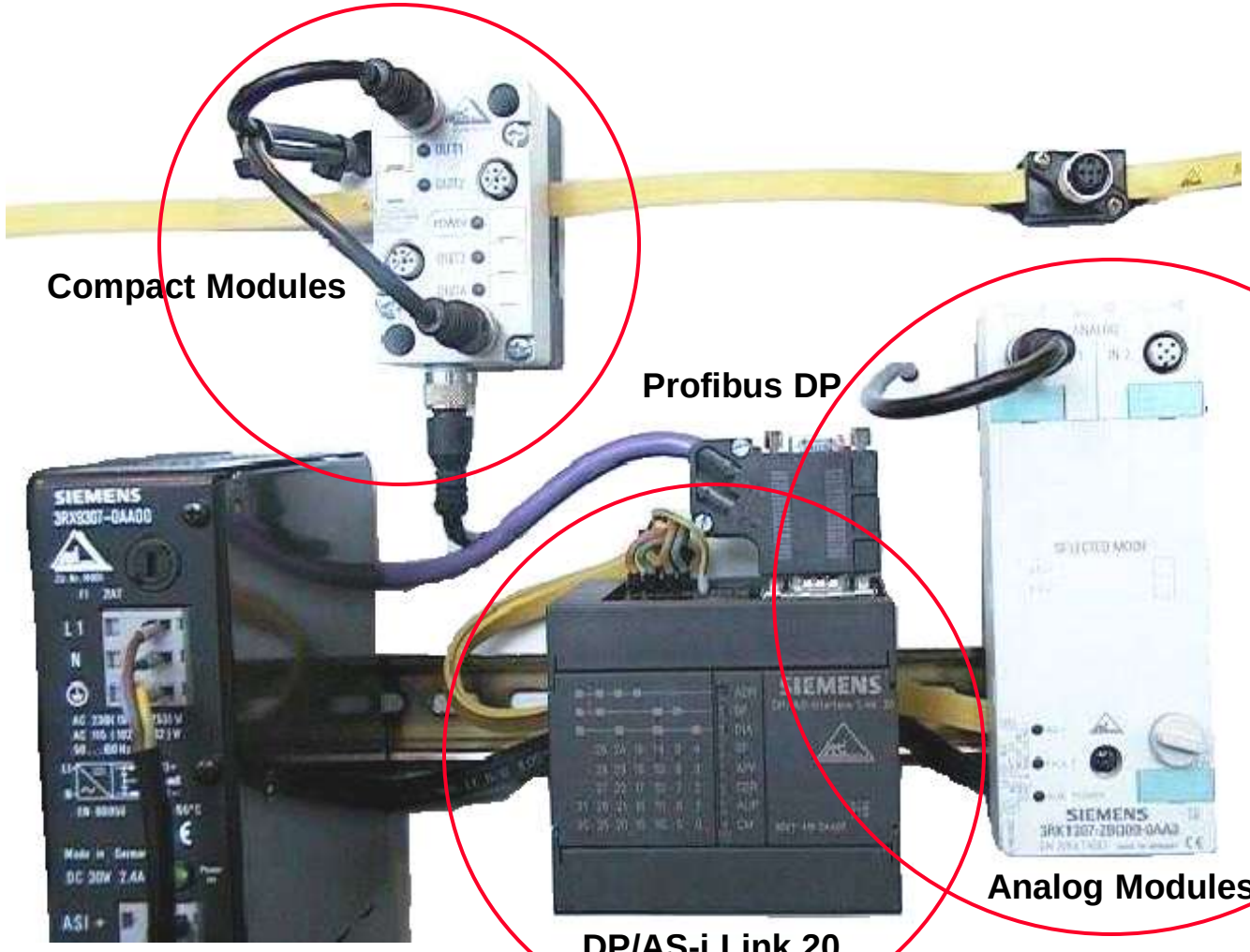
Area of Use and Basic Properties

AS-i Cable

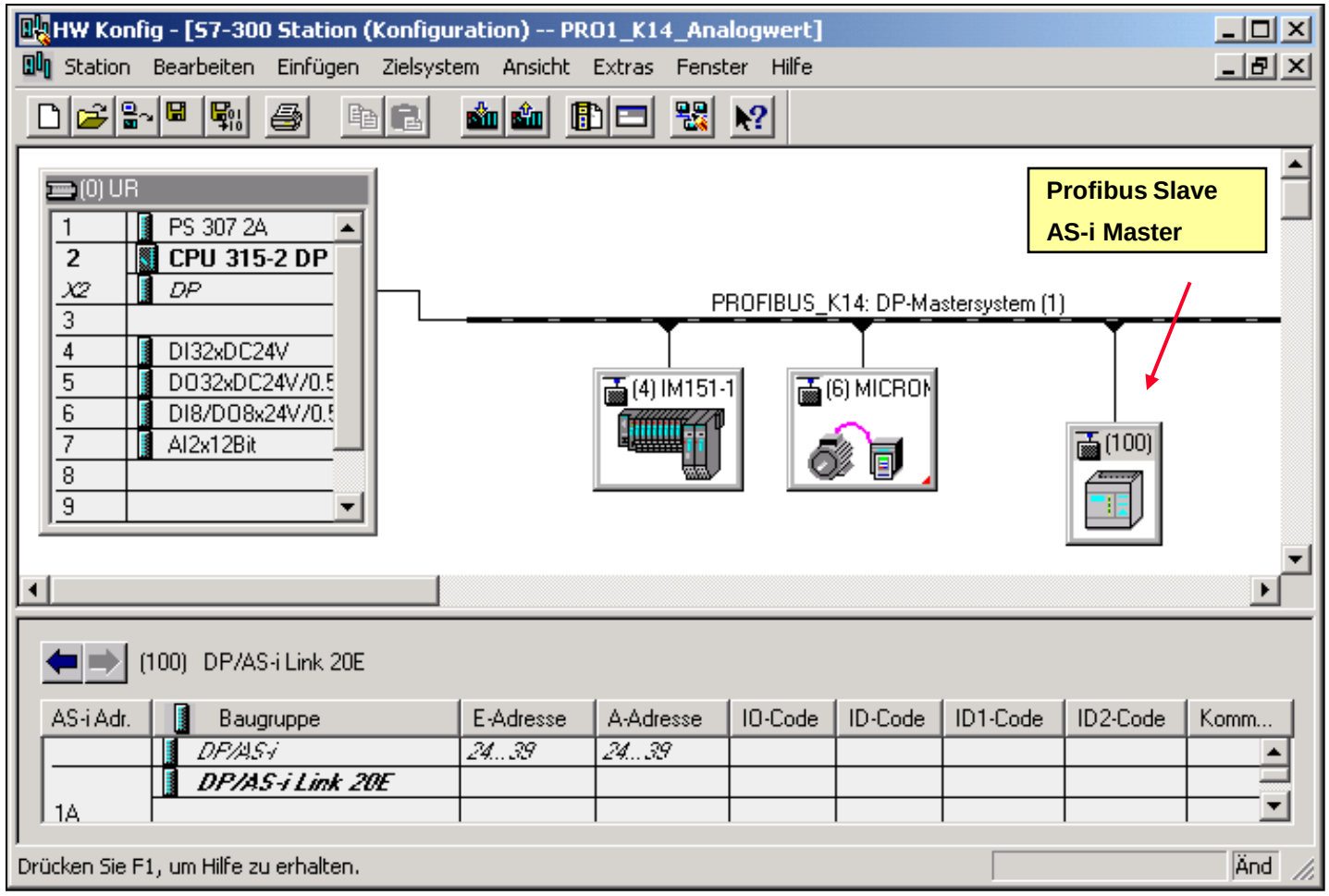




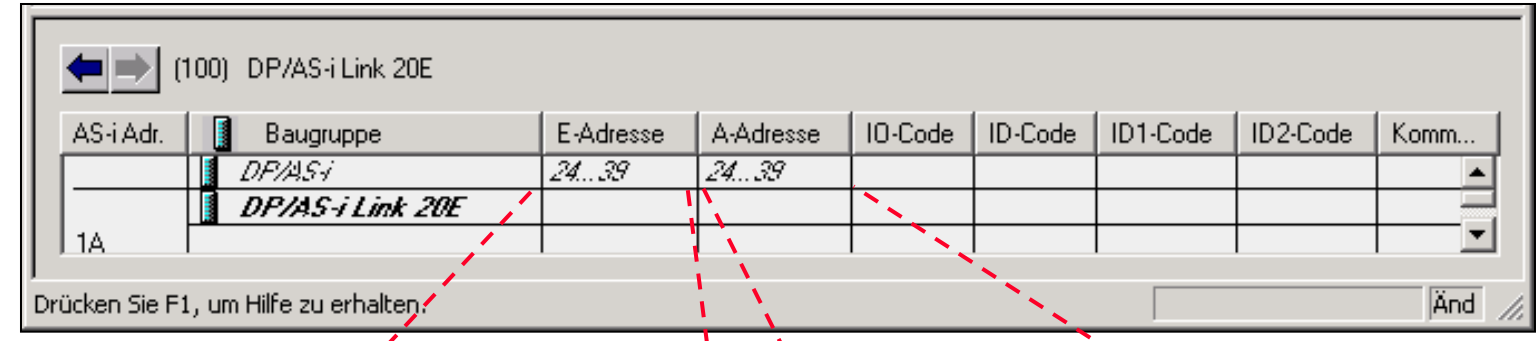
AS-i Modules



Hardware Configuration



Addressing AS-i slaves



26.7	26.6	26.5	26.4

	7 6 5 4	3 2 1 0
24		**slave 1**
25	slave 2	**slave 3**
26	**slave 4**	slave 5
27	slave 6	slave 7
28	slave 8	slave 9
29	slave 10	slave 11
30	slave 12	slave 13
31	slave 14	slave 15
32	slave 16	slave 17
33	slave 18	slave 19
34	slave 20	slave 21
35	slave 22	slave 23
36	slave 24	slave 25
37	slave 26	slave 27
38	slave 28	slave 29
39	slave 30	slave 31

	7 6 5 4	3 2 1 0
24		slave 1
25	**slave 2**	slave 3
26	slave 4	**slave 5**
27	slave 6	slave 7
28	slave 8	slave 9
29	slave 10	slave 11
30	slave 12	slave 13
31	slave 14	slave 15
32	slave 16	slave 17
33	slave 18	slave 19
34	slave 20	slave 21
35	slave 22	slave 23
36	slave 24	slave 25
37	slave 26	slave 27
38	slave 28	slave 29
39	slave 30	slave 31

26.3	26.2	26.1	26.0