SIEMENS

**Information and Training**
Automation and Drives

## SIMATIC S7

# Programming 1

## Course ST-7PRO1

AL: N    ECCN: N



Name: ______________________________

Course: from ____________ to ____________

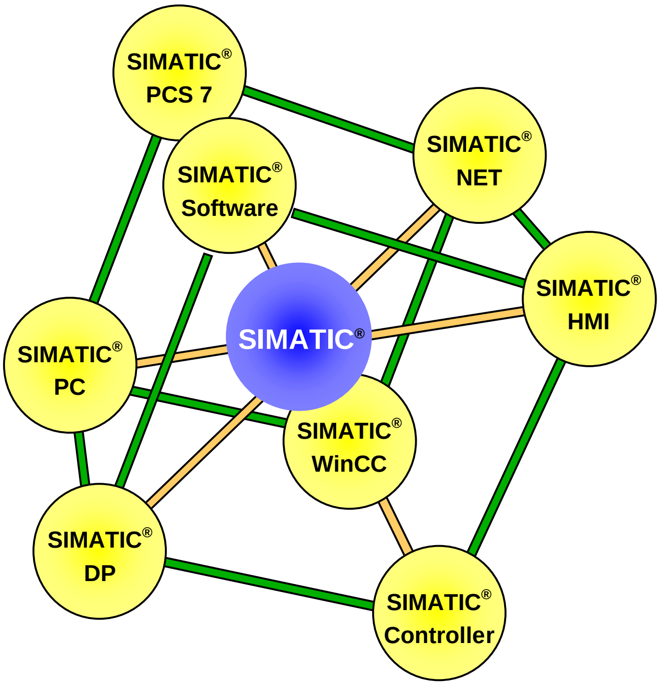
Instructor:


Infoline Tel: 01805 23 56 11
Fax: 01805 23 56 12
Internet: http://www.sitrain.com

ID-No.:
Version A5.4 (for STEP7 Version 5.2)

SIEMENS

# The SIMATIC® S7 System Family




SIMATIC® S7


Date: 12.03.03
File: PRO1_01e.1

**SITRAIN** Training for Automation and Drives

**Contents** Page

# Objectives

**Upon completion of this chapter the participant will ...**

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

... be familiar with the S7-200™ and S7-300/400™ automation systems

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

... understand the concept of "Totally Integrated Automation"

... be familiar with the SIMATIC® programming devices and the PC requirements for working with STEP7

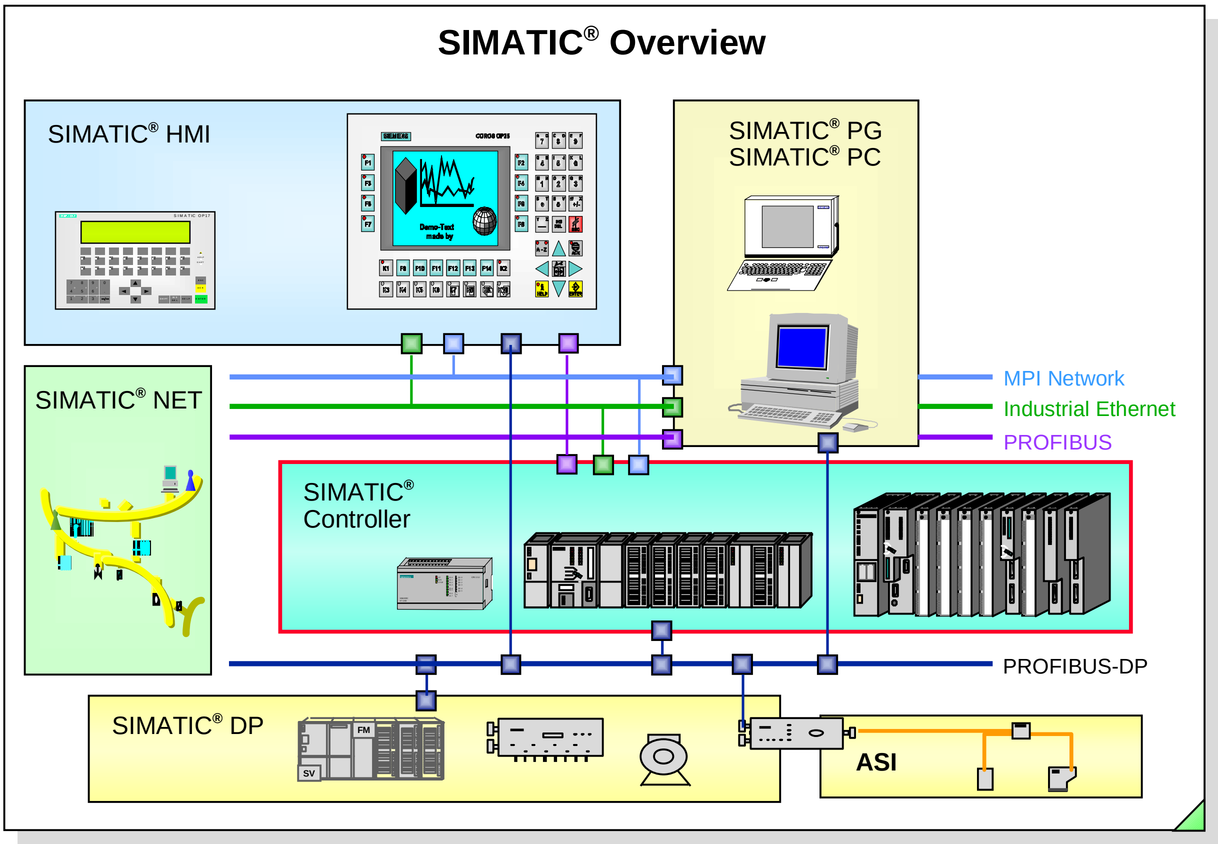
... be familiar with the tools of the STEP7 basic programming package

SIMATIC® S7


Date: 12.03.03
File: PRO1_01e.2

SITRAIN Training for Automation and Drives

**SIEMENS**




SIMATIC® S7


Date: 12.03.03
File: PRO1_01e.3

**SITRAIN** Training for Automation and Drives

**Introduction**

In the past, control tasks were solved with individual isolated **P**rogrammable **L**ogic **C**ontrols (PLCs) controlling a machine or process. Today in order for companies to remain competitive, it is not enough to automate only individual processing stations or machines in isolation. The demand for more flexibility with higher productivity can then be fulfilled when the individual machines are integrated in the entire system.

**Totally Integrated Automation**

Totally Integrated Automation (TIA) provides a common software environment that integrates all components, in spite of the diversification of applied technology, into one uniform system. This brings together everything you need to program, configure, operate, handle data, communicate, and maintain your control solutions.

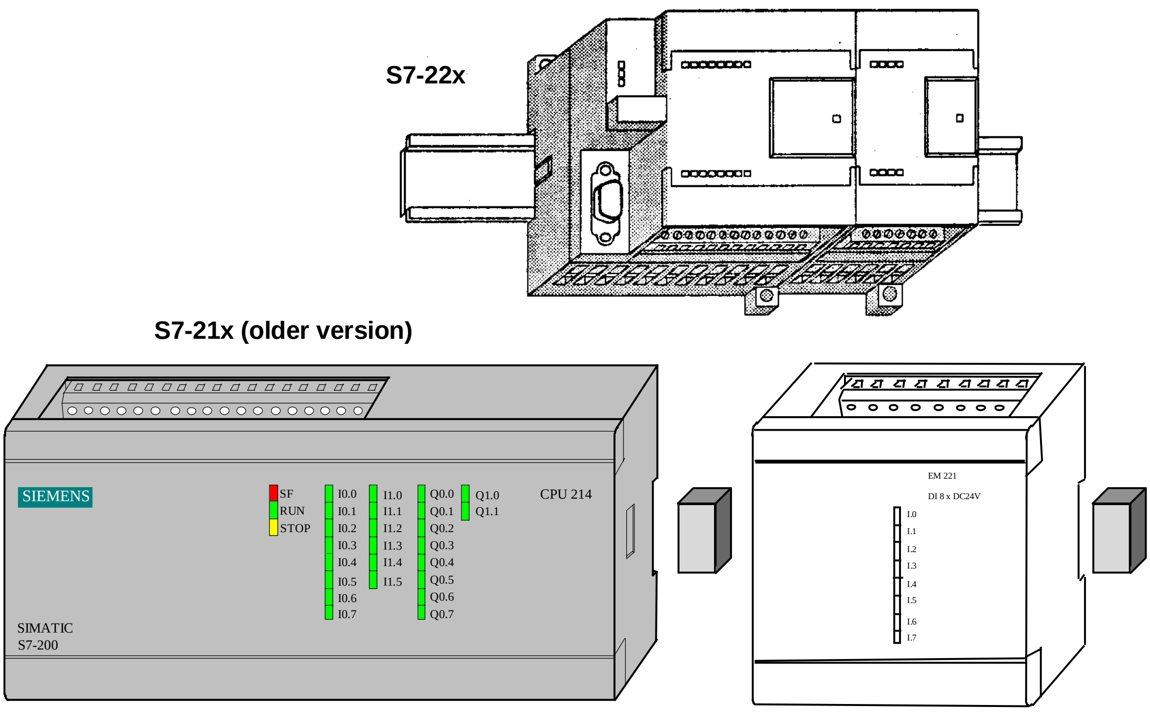
Step 7 SIMATIC Manager, running on Siemens PGs or PCs, provides an integrated set of tools for all system components that allows easy creation, testing, start-up, operation and maintenance of your control solutions. While you are configuring and programming, the Siemens software puts all of your data in a *central database* to which all of the tools have access.

**Central Database**

A common database of all components of Totally Integrated Automation means that data only have to be entered once and are then available for the entire project. The total integration of the entire automation environment is made possible with the help of:

- One common software environment (Step 7 SIMATIC Manager) that integrates all components and tasks into one uniform easy to use system.
- Common data management
- Standard open busses such as Ethernet, PROFIBUS, MPI, AS-interface connect all components to each other, from the management level to the field.

## S7-200 ™




SIMATIC® S7


Date: 12.03.03
File: PRO1_01e.4

**SITRAIN** Training for Automation and Drives

**Features**

- Modular small control system for the lowest performance range.
- Performance-graded range of CPUs (up to 8KB memory, 8-40 integrated I/O points onboard the CPU).
- Each CPU available in either 24 VDC or 120 VAC - 230 VAC supply voltage versions.
- Expandable design with up to seven expansion modules depending on CPU (none with CPU 210 or CPU 221).
- Extensive module selection. Note: Combined use of CPUs and modules of the S7-21x series with those of the S7-22x series is NOT possible!
- CPU connected to modules by flexible integrated ribbon cables (S7-22x series) and by bus connectors (S7-21x series).
- Network-capable with
  - RS 485 communication interface (Not CPU 210)
  - PROFIBUS slave (CPU 215 or CPU 222 or greater)
- Central PG/PC connection with access to all modules.
- No slot restrictions
- Uses its own S7 Micro/WIN32 software, therefore, STEP 7™ not required.
- “Total Package” (brick) with power supply, CPU and integrated I/O all in one.
- Password protection of user program - 3 levels.

## S7-200™: Modules S7-21x series


EM EM CP




**Expansion Modules (EM)**

- Digital input modules:
  - 24 VDC
  - 120/230 VAC
- Digital output modules:
  - 24 VDC
  - Relay
- Analog input modules:
  - Voltage
  - Current
  - Resistance
  - Thermocouple
- Analog output modules:
  - Voltage
  - Current

CPUs of the S7-21x series can only be extended with expansion modules of the S7-21x series. **Note:** Combined use of the S7-21x series with the S7-22x series is NOT possible!

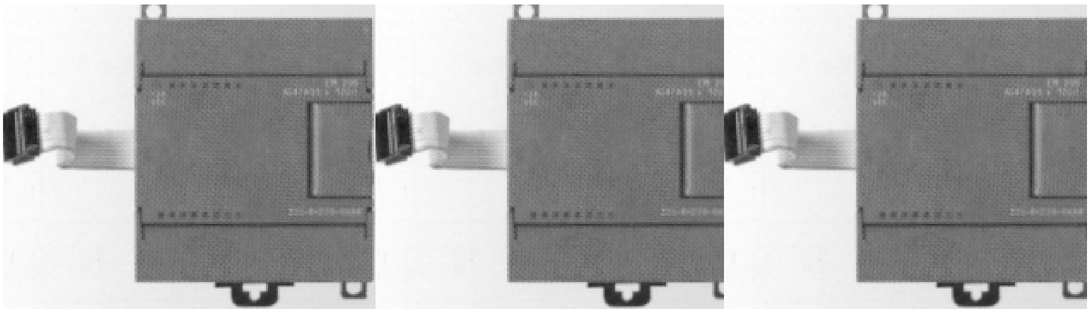
**Communications Processors (CP)**

You can use the CP 242-2 to significantly increase the number of inputs/outputs of the SIMATIC S7-200™ (S7-21x series). The CP acts as a master to an actuator and sensor interface (AS-Interface). As a result, 31 AS-Interface slaves can control up to 248 binary elements.

**Accessories**

Bus connector (S7-21x series only)

## S7-200™: Modules S7-22x series






**Expansion Modules (EM)**

- Digital input modules:
  - 24 VDC
  - 120/230 VAC
- Digital output modules:
  - 24 VDC
  - Relay
- Analog input modules:
  - Voltage
  - Current
  - Resistance
  - Thermocouple
- Analog output modules:
  - Voltage
  - Current

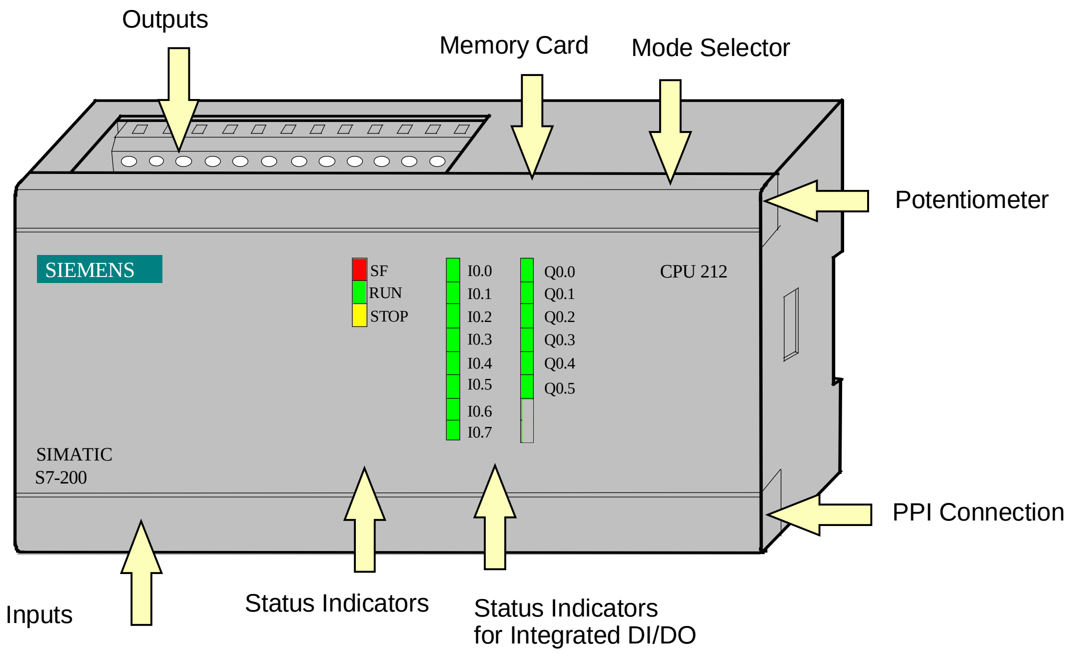
CPUs of the S7-22x series (CPU 222/224/226) can only be expanded using expansion modules of the S7-22x series. **Note:** Combined use of the S7-21x series with the S7-22x series is NOT possible!

**Communications Processors (CP)**

On the S7-22x series you can use the CP 243-2 to connect the S7-200™ as master to an AS-Interface. The newer CP243-2 supports up to 62 AS-Interface slaves being connected (max. 31 analog slaves). Up to three (CPU 224, CPU 226) CP243-2 processors can be operated simultaneously on these S7-200 CPUs.

The EM277 PROFIBUS-DP module allows connection of S7-22x series CPUs (6ES7-22x-xxx21-xxxx and later) to PROFIBUS-DP (as a slave) and MPI. Simultaneous operation is possible as a MPI slave and PROFIBUS-DP slave. PROFIBUS-DP data transfer rates of up to 12 Mbits/second are supported.

## S7-200™: CPU Design S7-21x Series




SIMATIC® S7


Date: 12.03.03
File: PRO1_01e.7

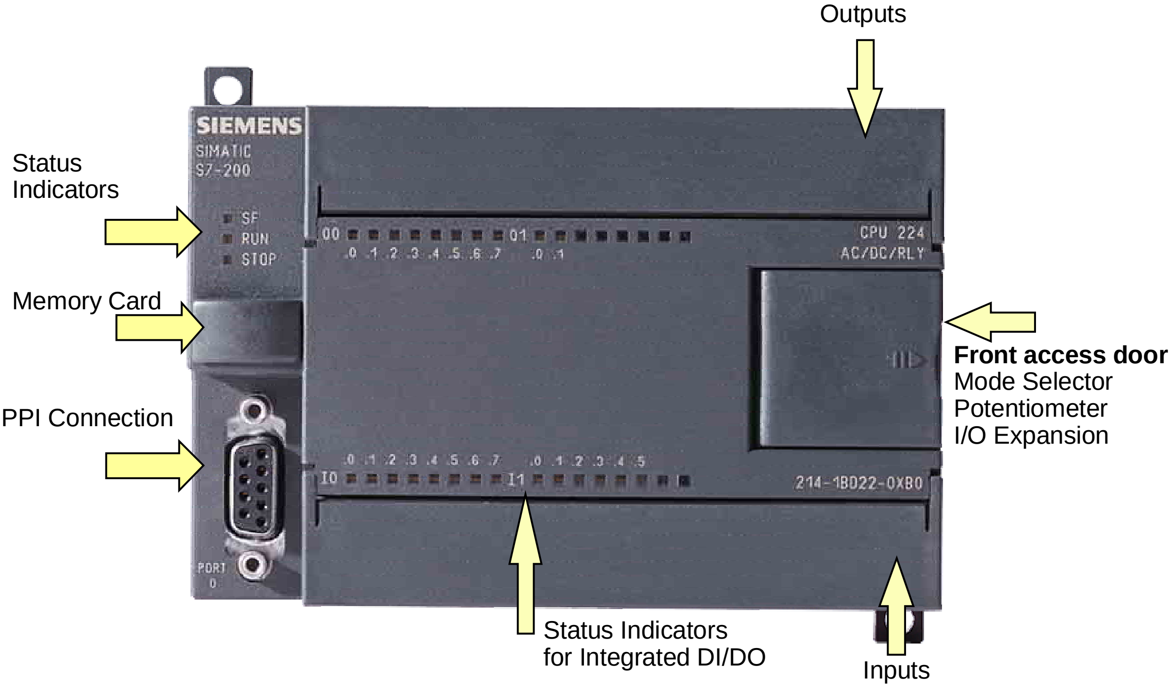
SITRAIN Training for Automation and Drives

| | | | |
|---|---|---|---|
| **Mode Selector** | For manual mode selection: | | |
| | STOP | = | Stop mode, the program is not executed. |
| | TERM | = | Program execution, read/write access possible from PG. |
| | RUN | = | Program execution, read-only access possible from PG. |
| **Status Indicators (LEDs)** | SF | = | Group error; internal CPU fault; red |
| | RUN | = | Run mode; green |
| | STOP | = | Stop mode; yellow |
| | DP | = | PROFIBUS-DP ( only CPU 215) |

**Memory Card** Slot for memory card. A memory card saves the program contents in the event of a power outage without the need for a battery.

**PPI Connection** The programming device, text display, or another CPU is connected here.

# S7-200™: CPU Design S7-22x Series




SIMATIC® S7


Date: 12.03.03
File: PRO1_01e.8

**SITRAIN** Training for Automation and Drives

| | | | |
|---|---|---|---|
| **Mode Selector** | For manual mode selection: | | |
| | STOP | = | Stop mode, the program is not executed. |
| | TERM | = | Program execution, read/write access possible from PG. |
| | RUN | = | Program execution, read-only access possible from PG. |
| **Status Indicators (LEDs)** | SF | = | Group error; internal CPU fault; red |
| | RUN | = | Run mode; green |
| | STOP | = | Stop mode; yellow |
| | DP | = | PROFIBUS-DP ( only CPU 215) |

**Memory Card**

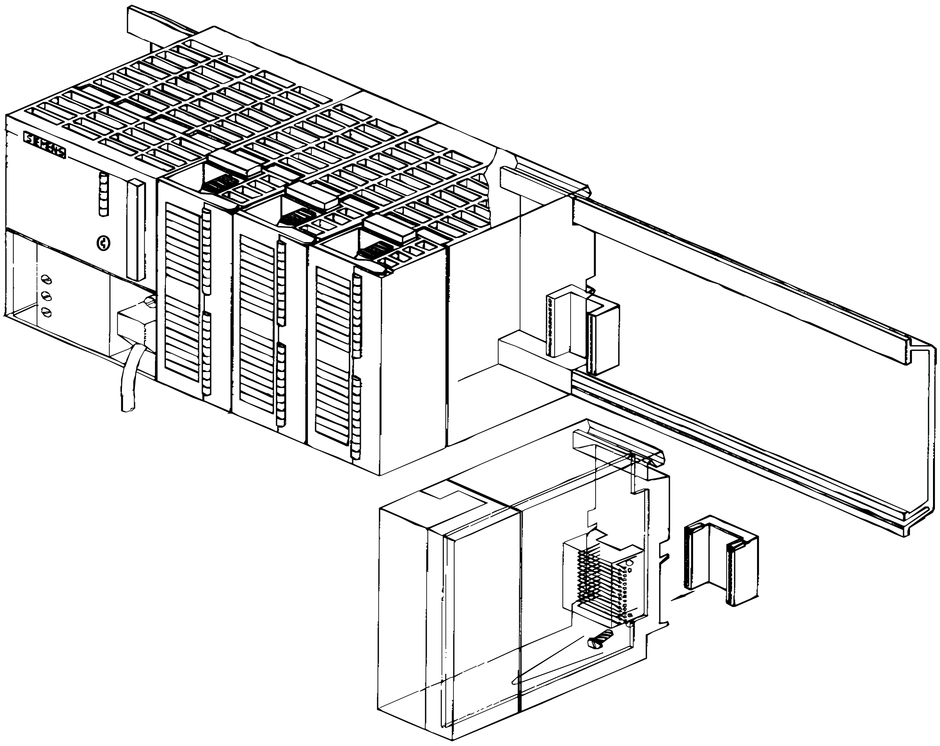
Slot for memory card. A memory card saves the program contents in the event of a power outage without the need for a battery.

**PPI Connection**

The programming device, text display, or another CPU is connected here, except for CPU 210 that is programmed in a programming system (PDS210) and then transferred to each CPU via a memory submodule.

Higher end CPUs contain two ports allowing the programming device and text display to be connected at the same time.

## S7-300™



SIMATIC® S7

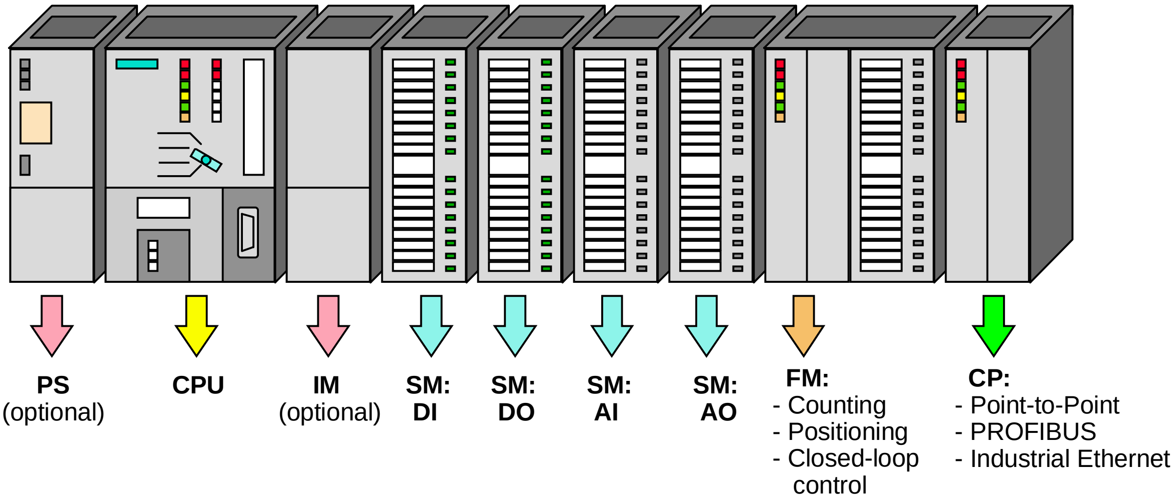

Date: 12.03.03
File: PRO1_01e.9

SITRAIN Training for Automation and Drives

**Features**

- Modular small control system for the lower performance range
- Performance-graded range of CPUs
- Extensive selection of modules
- Expandable design with up to 32 modules
- Backplane bus integrated in the modules
- Can be networked with - Multipoint interface (MPI),
  - PROFIBUS or
  - Industrial Ethernet.
- Central PG/PC connection with access to all modules
- No slot restrictions
- Configuration and parameter setting with the help of the "HWConfig" tool.

## S7-300™: Modules




SIMATIC® S7


Date: 12.03.03
File: PRO1_01e.10

SITRAIN Training for Automation and Drives

| | | |
|---|---|---|
| **Signal Modules (SM)** | • Digital input modules: | 24 VDC, 120/230 VAC |
| | • Digital output modules: | 24 VDC, Relay |
| | • Analog input modules: | Voltage, current, resistance, thermocouple |
| | • Analog output modules: | Voltage, current |

**Interface Modules (IM)**
The IM360/IM361 and IM365 make multi-tier configurations possible. The interface modules loop the bus from one tier to the next.

**Dummy Modules (DM)**
The DM 370 dummy module reserves a slot for a signal module whose parameters have not yet been assigned. A dummy module can also be used, for example, to reserve a slot for installation of an interface module at a later date.

**Function Modules (FM)**
Perform "special functions":
- Counting
- Positioning
- Closed-loop control.

**Communication Processors (CP)**
Provide the following networking facilities:
- Point-to-Point connections
- PROFIBUS
- Industrial Ethernet.

**Accessories**
Bus connectors and front connectors

## S7-300™: CPU Design


SIMATIC® S7


Date: 12.03.03
File: PRO1_01e.11

SITRAIN Training for Automation and Drives

| | | | |
|---|---|---|---|
| **Mode Selector** | MRES | = | Memory reset function (Module Reset) |
| | STOP | = | Stop mode, the program is not executed. |
| | RUN | = | Program execution, read-only access possible from PG. |
| | RUN-P | = | Program execution, read/write access possible from PG. |
| **Status Indicators (LEDs)** | SF | = | Group error; internal CPU fault or fault in module with diagnostics capability. |
| | BATF | = | Battery fault; battery empty or non-existent. |
| | DC5V | = | Internal 5 VDC voltage indicator. |
| | FRCE | = | FORCE; indicates that at least one input or output is forced. |
| | RUN | = | Flashes when the CPU is starting up, then a steady light in Run mode. |
| | STOP | = | Shows a steady light in Stop mode.<br>Flashes slowly for a memory reset request,<br>Flashes quickly when a memory reset is being carried out,<br>Flashes slowly when a memory reset is necessary because a memory card has been inserted. |

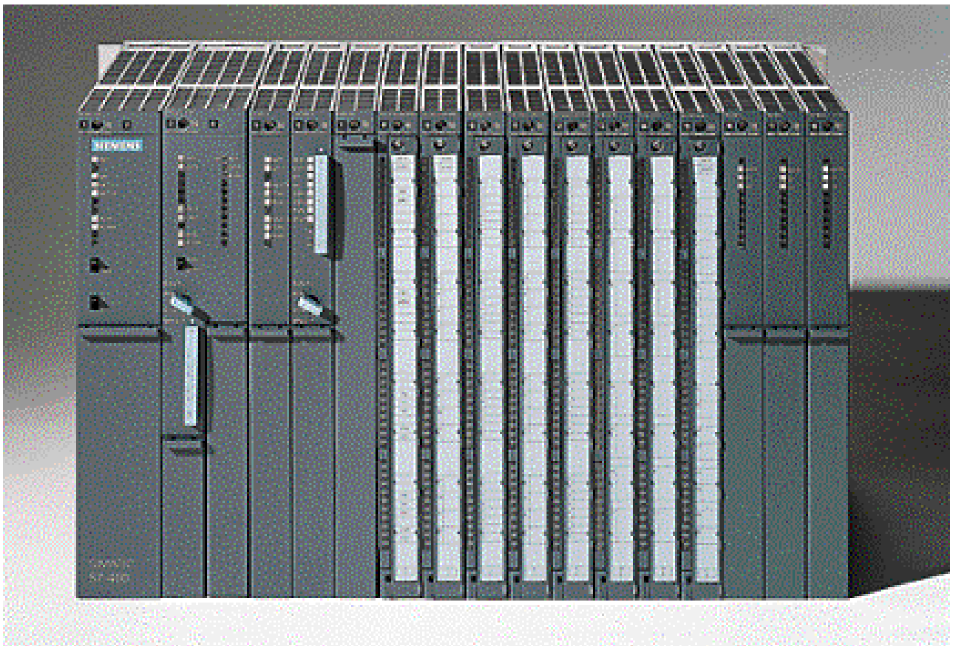
**Memory Card** A slot is provided for a memory card. The memory card saves the program contents in the event of a power outage without the need for a battery.

**Battery Compartment** There is a receptacle for a lithium battery under the cover. The battery provides backup power to save the contents of the RAM in the event of a power outage.

**MPI Connection** Connection for a programming device or other device with an MPI interface.

**DP Interface** Interface for direct connection of distributed I/Os to the CPU.

# S7-400™

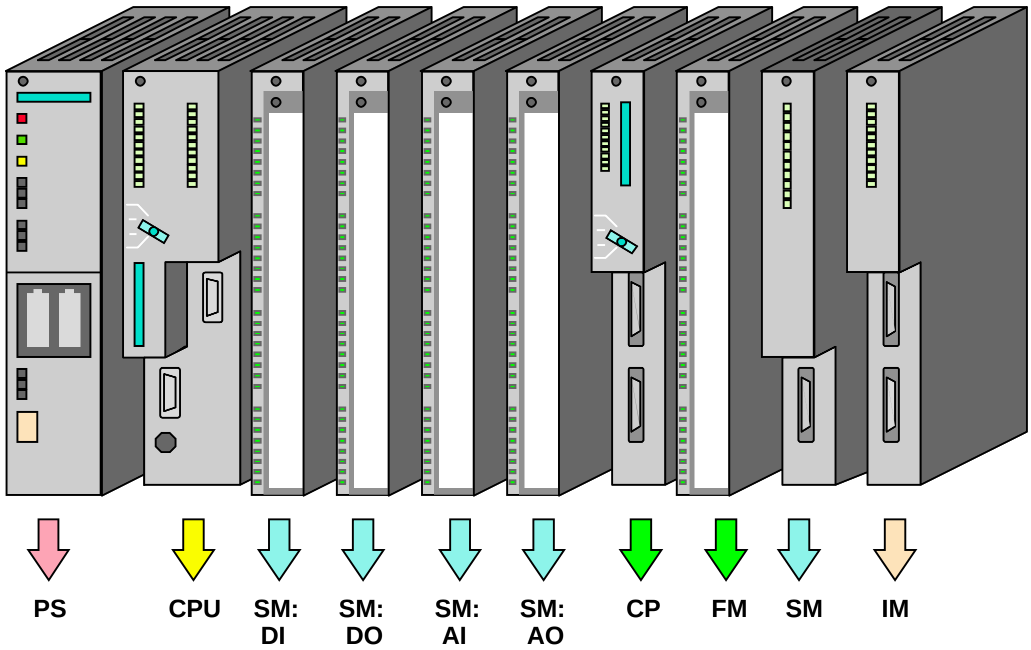




**Features**

- The power PLC for the mid to upper performance range
- Performance-graded range of CPUs
- Extensive selection of modules
- Expandable design to over 300 modules
- Backplane bus integrated in the modules
- Can be networked with - Multipoint interface (MPI),
  - PROFIBUS or
  - Industrial Ethernet
- Central PG/PC connection with access to all modules
- No slot restrictions
- Configuration and parameter setting with the help of the "HWConfig" tool
- Multicomputing (up to four CPUs can be used in the central rack )

## S7-400™: Modules




SIMATIC® S7


Date: 12.03.03
File: PRO1_01e.13

SITRAIN Training for Automation and Drives

**Signal Modules (SM)**

- Digital input modules: 24 VDC, 120/230 VAC
- Digital output modules: 24 VDC, Relay
- Analog input modules: Voltage, current, resistance, thermocouple
- Analog output modules: Voltage, current.

**Interface Modules (IM)**

The IM460, IM461, IM463, IM467 interface modules provide the connection between various racks:

- UR1 (Universal Rack) with up to 18 modules
- UR2 (Universal Rack) with up to 9 modules
- ER1 (Expansion Rack) with up to 18 modules
- ER2 (Expansion Rack) with up to 9 modules.

**Function Modules (FM)**

Perform "special functions":

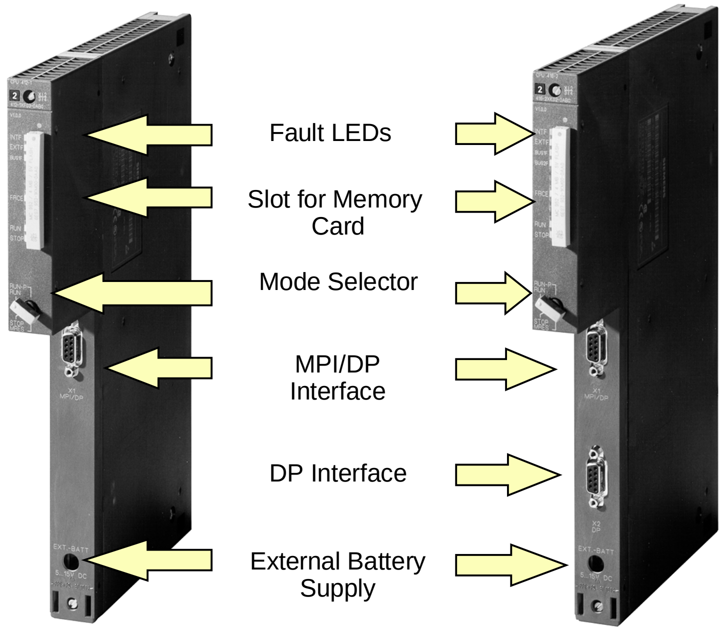
- Counting
- Positioning
- Closed-loop control.

**Communication Processors (CP)**

Provide the following networking facilities:

- Point-to-Point connections
- PROFIBUS
- Industrial Ethernet.

## S7-400™: CPU Design







**Fault LEDs** LEDs for the CPU's statuses and faults, both internal and external

**Slot for Memory Cards** With the S7-400™ CPUs you can, depending on your requirements, insert RAM or flash EPROM cards as external load memory :

- RAM cards are available with a capacity of: 64KByte, 256KByte, 1MByte, 2MByte.
  The CPU battery backs up the contents.
- Flash EPROM cards are available with a capacity of: 64KByte, 256KByte, 1MByte, 2MByte, 4MByte, 8MByte, 16MByte.
  The contents are backed-up on the integrated EEPROMs.

**Mode Selector**

MRES = Memory reset function (**M**odule **RES**et)
STOP = STOP mode, that is, no program execution and output disabled ("OD" mode = **O**utput **D**isabled).
RUN = Program execution, read-only access possible from PG.
RUN-P = Program execution, read/write access possible from PG.

**MPI / DP Interface** MPI / DP interface (parameter-assignable in HW-Config) for

- establishing the online connection to the programming device
- connecting to distributed peripherals (DP)
- data exchange with other stations (S7 Communication)

**DP Interface** For connecting to distributed peripherals (DP, only for CPUs with 2 interfaces)

**EXT-BATT** Additional external battery socket for a 5 VDC to 15 VDC source to back up the RAM when the power supply is being replaced.

## Programming Devices


SIMATIC® S7


Date: 12.03.03
File: PRO1_01e.15

SITRAIN Training for Automation and Drives

**Field PG**

An industry-standard programming device. The Field PG is powerful and easy to use, especially for maintenance and service. It is also appropriate for programming and configuration - the ideal tool for shop floor applications.

Features:
- Dimensions in Notebook format
- 2 hour battery operation
- AT-compatible
- TFT color display
- Equipped with all necessary SIMATIC interface ports

**Power PG**

A portable programming device, ideal for all applications in an automation project. It is also an extremely powerful, industry-standard PC.

Features:
- High-level system performance
- Excellent expansion facilities
- TFT color display
- Highly rugged design
- Equipped with all the necessary SIMATIC interface ports

**Note**

A hand-held programming device is also available for programming the S7-200™ in STL (Statement List). This programming device is the PG702 = approximately 230 grams, 144 x 72 x 27mm, 2 lines x 20 character LC display.

# PG/PC Requirements for Installing STEP 7

| | | | | |
|---|---|---|---|---|
| **Operating system**: | Windows (all, except Win 3.1 and 3.11) | | | |
| | 95/98 | ME | NT | 2000/XP |
| **Processor** | >= 80486 | >= P150 | >= Pentium | >= P233 |
| **RAM** | >= 32 MB | >= 64 MB | >= 32 MB | >= 128 MB |

**Memory on the Hard Drive:** depending on the installation, between 200 MB and 380 MB plus 128 to 256 MB minus the working memory for Windows Swap File

**Mouse**: yes

**Interfaces**: CP5611 (PCI) or
CP5511 / CP5512 (PCMCIA) or
PC adapter
Programming interface for Memory Card (optional)

SIMATIC® S7


Date: 12.03.03
File: PRO1_01e.16

SITRAIN Training for Automation and Drives

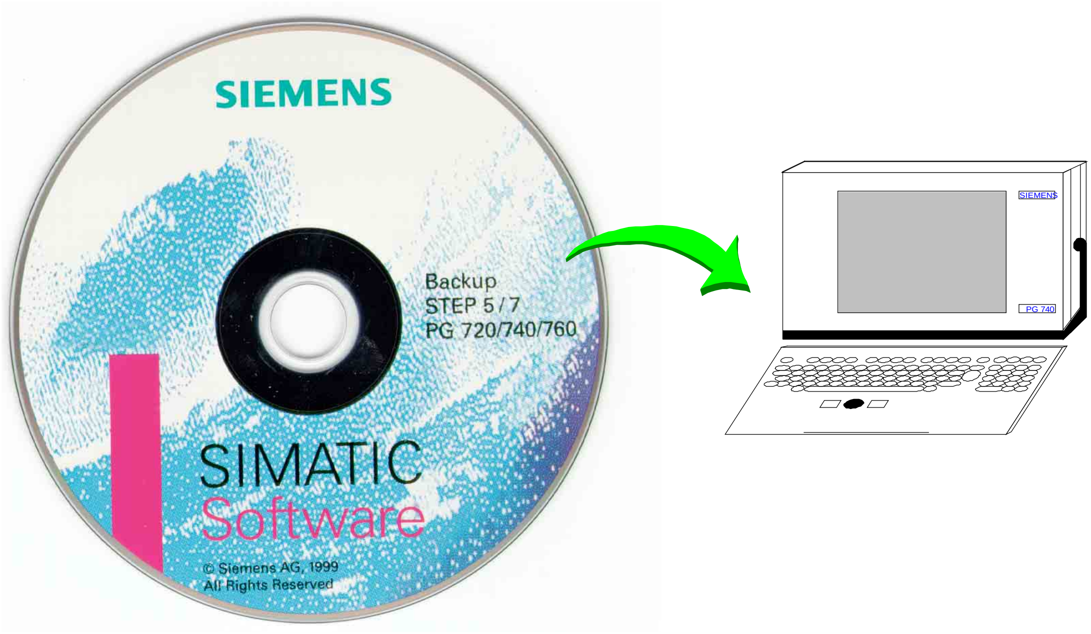
**Requirements**

SIMATIC PGs provide the optimum basis for using the STEP 7 software.

You can, however, also use a PC that fulfills the above-listed requirements. So that you can make the necessary online connection between the automation system (PLC) and the PC, the PC must be equipped with one of the interfaces listed in the slide above.

If user programs are to be loaded on memory cards, the PC must also be equipped with the appropriate programming interface.

# Installing STEP 7 Software




SIMATIC® S7


Date: 12.03.03
File: PRO1_01e.17

SITRAIN Training for Automation and Drives

**Installation**

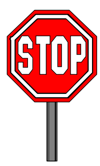
1. Activate "Setup.exe" by selecting "Add/Remove Programs" in the "Winxx->Control Panel".
2. Choose Options.
3. Choose Language.
4. Insert authorization disk when prompted.
5. Re-boot when prompted.

**Software Protection**

The STEP 7 software is copy-protected and can only be used on one programming device at a time.

When you have installed the software, you cannot start using it until you have transferred the authorization to the hard disk drive from the authorization disk.

**Note**

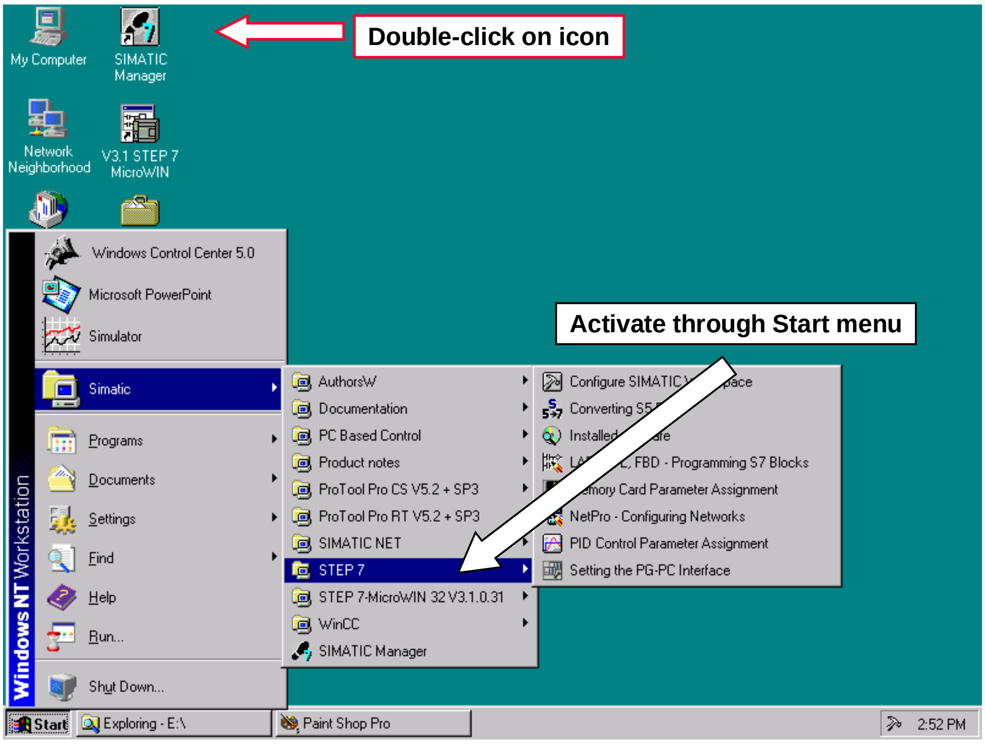



- Be sure to read the notes in the README.TXT file on the authorization disk. If you do not observe these instructions, you risk losing your authorization.
- As of STEP 7 V5.0, the STEP 7 basic package can also be operated without authorization until a new authorization has been received. This, however, does not apply to option packages, such as S7 Graph, if they have been installed .
- As of STEP 7 V4.0, the software is only available on CD-ROM
- Software Service Packs can also be downloaded from the Internet.

**Service Packs Free-of-Charge**

Software Service Packs can be downloaded from the Internet via http://www.siemens.com/automation/service&support.

## Result of Installation




SIMATIC® S7



Date: 12.03.03
File: PRO1_01e.18

SITRAIN Training for Automation and Drives

**Introduction**

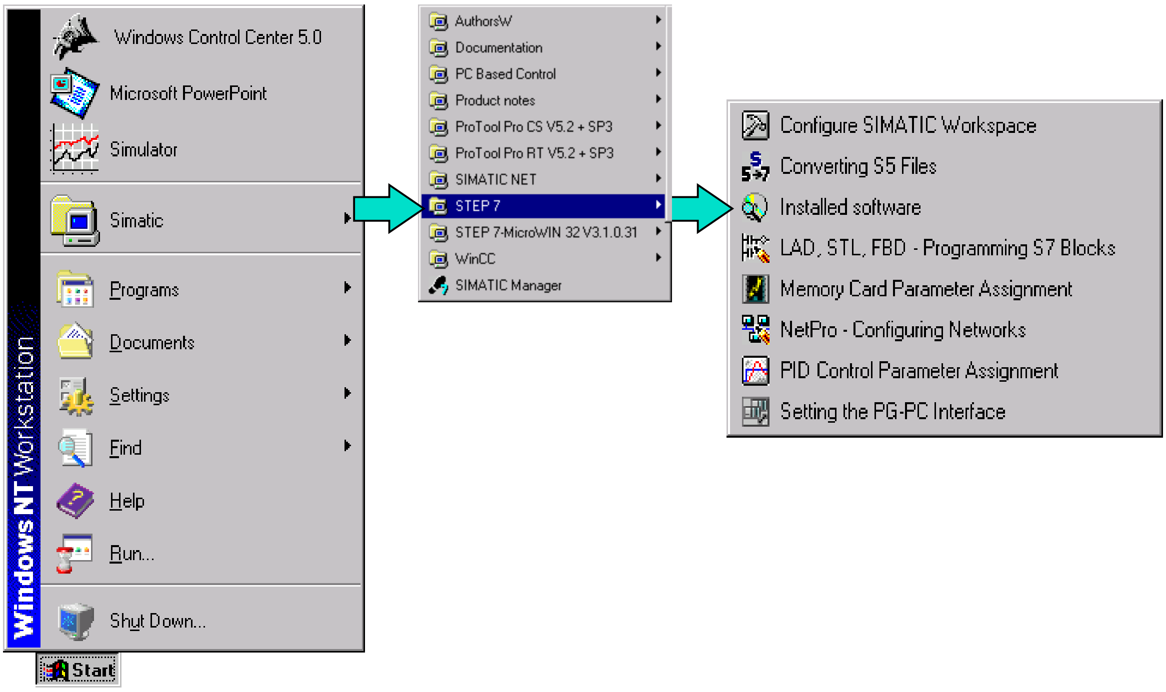
The main tool in STEP 7 is the SIMATIC® Manager. There are two ways in which to activate it:

1. through *Task bar -> Start -> SIMATIC® -> SIMATIC® Manager*

2. through the icon "SIMATIC® Manager".

SIEMENS

## STEP 7 Tools




SIMATIC® S7


Date: 12.03.03
File: PRO1_01e.19

SITRAIN Training for Automation and Drives

**SIMATIC® Manager** The SIMATIC® Manager manages the STEP 7 projects. It is the main program and also appears on the WINDOWS desktop.

**Notes** "STEP 7 - Readme" provides detailed information about the version, installation procedure, etc.

**LAD, STL, FBD** Tool for writing STEP 7 user programs in the "Ladder Diagram", "Statement List" or "Function Block Diagram" programming languages.

**Memory Card Parameter Assignment.** You can save your user programs on EPROM cards by either using the programming device or an external prommer.
Different drivers are required, depending on the application.

**Configuring Networks** Network configuration is discussed in the chapter on "Communication".

**Setting the PG-PC Interface** This tool is used for setting the local node address, the transmission speed and the highest node address in the MPI network.

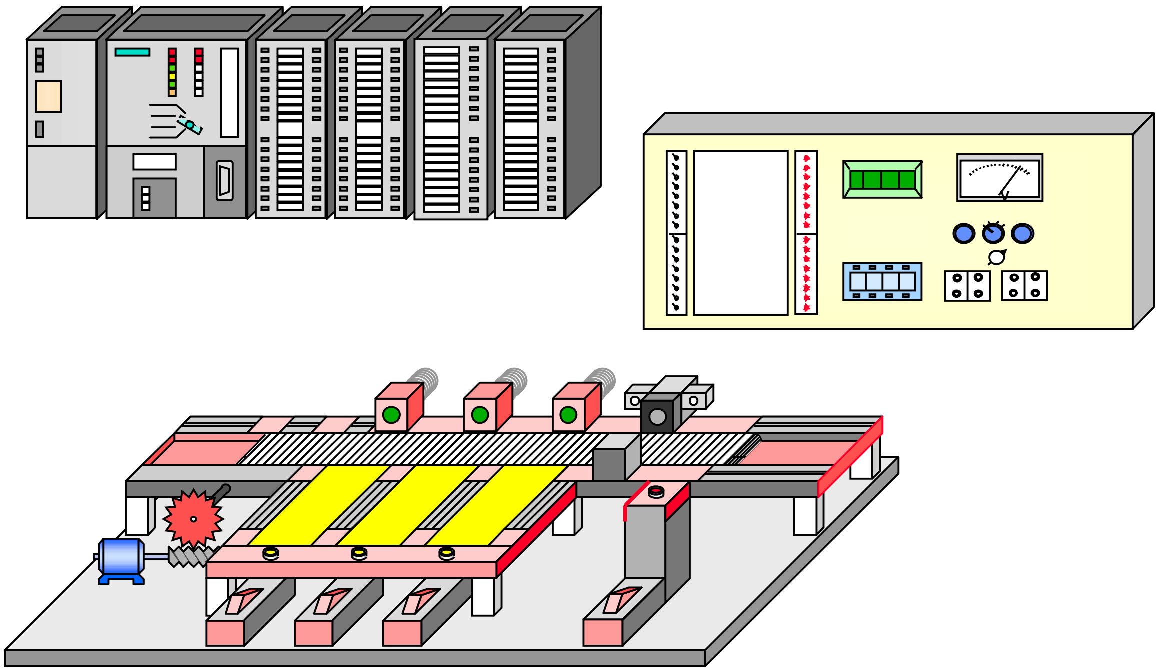
**PID Control Parameter Assignment** The basic STEP 7 software package also includes blocks for solving PID (closed-loop) control problems. You choose "PID Control Parameter Assignment" to start the program for assigning parameters to the closed-loop control blocks.

**Converting S5 Files** STEP5 programs can be converted into the corresponding STEP 7 programs with the help of the S5/S7 converter.

**Configure SIMATIC Workspace** This option provides facilities for configuring multi-user systems.

**Converting TI Files** SIMATIC TI programs can be converted into the corresponding STEP 7 program with the help of the TI/S7 converter.

## Training Units



SIMATIC® S7


Date: 12.03.03
File: PRO1_02e.1

SITRAIN Training for Automation and Drives

**Contents** Page

## Setup of a Training Area with S7-300™



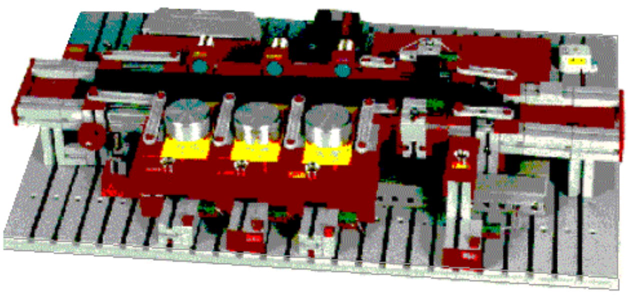
Date: 12.03.03
File: PRO1_02e.2


**Contents of the Training Kit**

The training kit consists of the following components:

- An S7-300™ programmable logic controller with CPU 314 or CPU 315-DP
- Digital input and output modules, analog modules
- Simulator with digital and analog sections
- Conveyor model

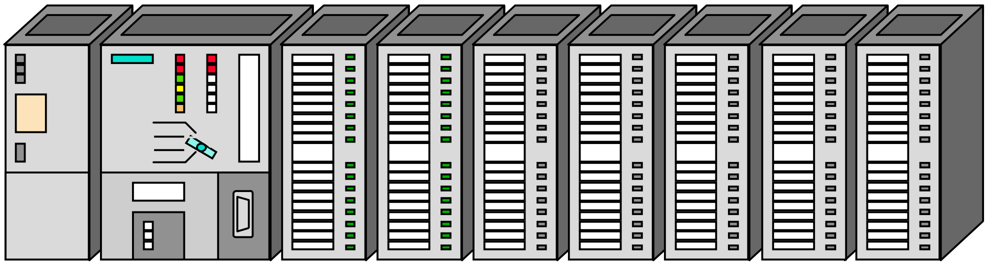
Note:
It is quite possible that your training area is not equipped with the conveyor model shown in the slide above, but rather with the conveyor model pictured below.



SIEMENS

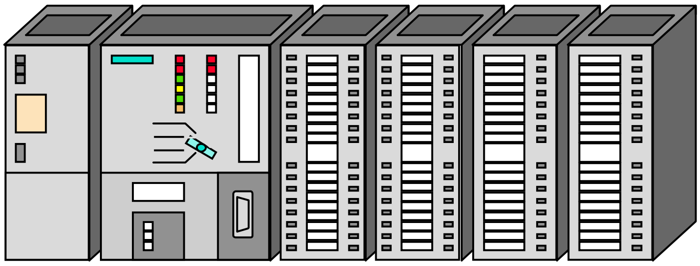
## Configuration of the S7-300™ Training Unit

**Version A**
(16 channel
I/O modules)



| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Module | --> | PS | CPU | DI 16 | DI 16 | DO 16 | DO 16 | DI 16 | DO 16 | AI/AO4 |
| Slot No. | --> | 1 | 2 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| I/O Address | --> | | | 0 | 4 | 8 | 12 | 16 | 20 | 352 |

**Version B**
(32 channel
I/O modules)



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 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 |

SIMATIC® S7


Date: 12.03.03
File: PRO1_02e.3

**SITRAIN** Training for
Automation and Drives

| | | | |
|---|---|---|---|
| **Configuration of Version A** | The programmable controller is configured with the following modules: | | |
| | Slot 1: | Power Supply 24V/5A | |
| | Slot 2: | CPU 314 or CPU 315-2 DP | |
| | Slot 4: | Digital input 16x24V | Inputs from the simulator |
| | Slot 5: | Digital input 16x24V | Thumbwheel buttons |
| | Slot 6: | Digital output 16x24V 0.5A | Outputs from the simulator |
| | Slot 7: | Digital output 16x24V 0.5A | Digital display |
| | Slot 8: | Digital input 16x24V | Conveyor model inputs |
| | Slot 9: | Digital output 16x24V 0.5A | Conveyor model outputs |
| | Slot 10: | Analog module 4 AI/4 AO | Adjustable from the simulator |

| | | | |
|---|---|---|---|
| **Configuration of Version B** | The programmable controller is configured with the following modules: | | |
| | Slot 1: | Power Supply 24V/5A | |
| | Slot 2: | CPU 314 or CPU 315-2 DP | |
| | Slot 4: | Digital input 32x24V | Inputs from the simulator and thumbwheel buttons |
| | Slot 5: | Digital output 32x24V/0.5A | Outputs from the simulator and digital display |
| | Slot 6: | Digital input and output module 8X24V/ 8x24V 0.5A | Conveyor model |
| | Slot 7: | Analog input 2 AI | Analog section from simulator |

**Addresses**

Fixed slot addressing is used for the S7-300™ (CPU 312-314). The module addresses are shown in the slide.

The starting addresses of the modules can be set by parameter assignment on the CPU 315-2DP and for S7-400™.

## Setup of a Training Area with S7-400™

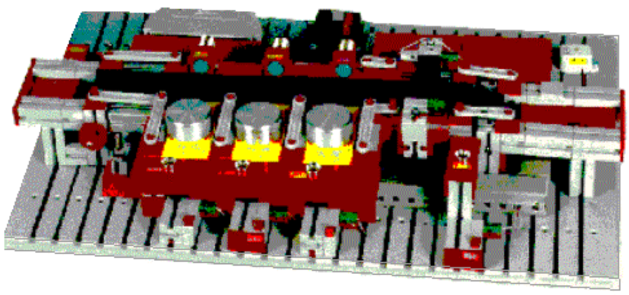


**Contents of the Training Kit**

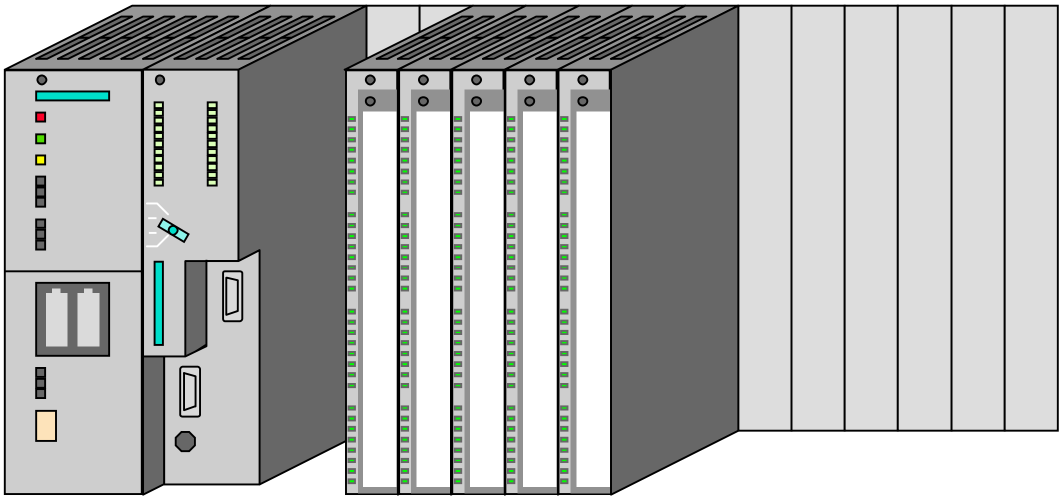
The training kit consists of the following components:

- An S7-400™ programmable logic controller with CPU 412 or CPU 413-2DP
- Digital input and output modules, analog modules
- Simulator with digital and analog sections
- Conveyor model

Note:
It is quite possible that your training area is not equipped with the conveyor model shown in the slide above, but rather with the conveyor model pictured below.



## Configuration of the S7-400™ Training Unit




SIMATIC® S7

Date: 12.03.03
File: PRO1_02e.5
**SITRAIN** Training for Automation and Drives

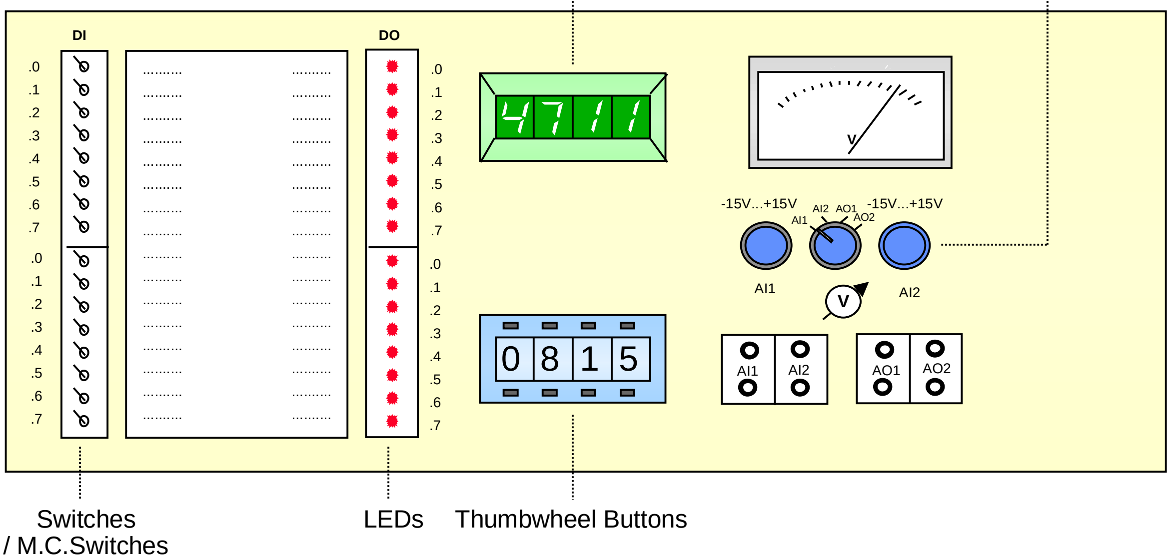
**Design** You can see the design of the S7-400™ training unit in the slide above.

**Configuration** The UR 1 mounting rack is configured with the following modules:

| | | |
|---|---|---|
| Slot 1: | Power supply 24V and 5V/20A | |
| Slot 2: | - " - | |
| Slot 3: | - " - | |
| Slot 4: | CPU 412 or other | |
| Slot 5: | vacant (when the CPU only has a single width) | |
| Slot 6: | vacant | |
| Slot 7: | vacant | |
| Slot 8: | Digital input 32x24V | (from Simulator) |
| Slot 9: | Digital input 32x24V | (from Conveyor Model) |
| Slot 10: | Digital output 32x24V 0.5A | (to Simulator) |
| Slot 11: | Digital output 32x24V 0.5A | (to Conveyor Model) |
| Slot 12: | Analog input 8X13 Bit | (from Poti on the Simulator) |
| Slot 13: | vacant | |
| Slot 14: | vacant | |
| Slot 15: | vacant | |
| Slot 16: | vacant | |
| Slot 17: | vacant | |
| Slot 18: | vacant | |

**Addressing** You have the default addresses, as shown in the slide above, as long as no configuration or parameter settings have been made.

## The Simulator







**Design**

Two cables connect the simulator to the S7-300™ or S7-400™ training unit. The simulator is divided into the following three sections:

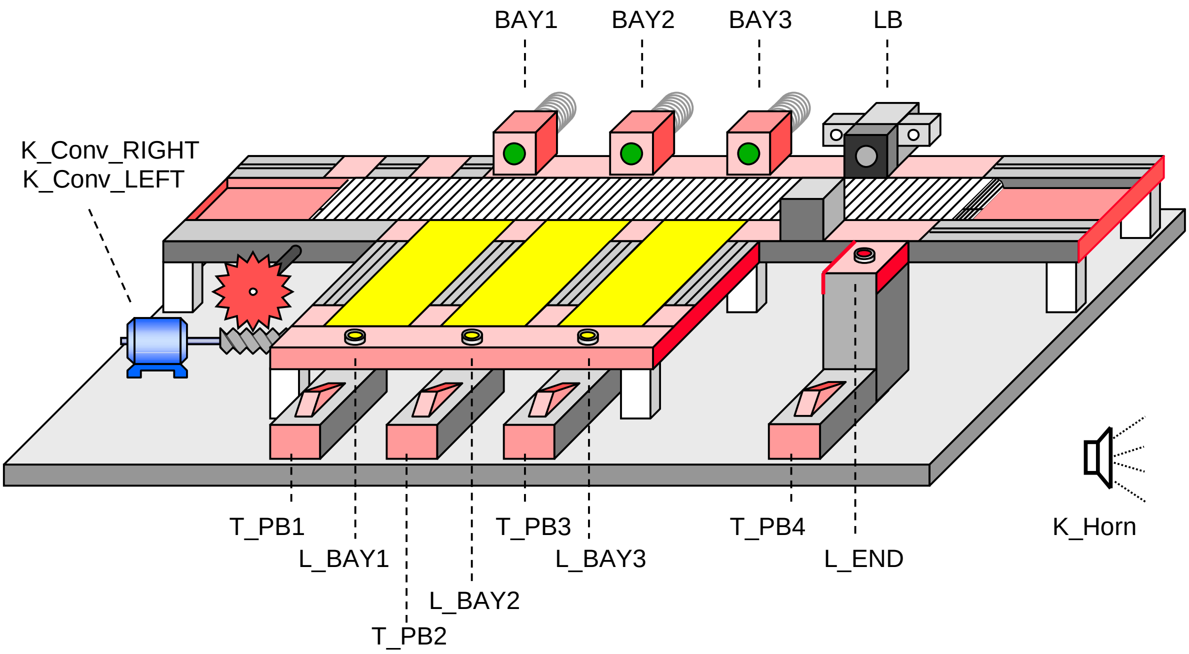
- Binary section with 16 switches/momentary-contact switches and 16 LED's
- Digital section with 4 thumbwheel buttons and a digital display. The thumbwheel and digital (BCD) display use BCD values,
- Analog section with a voltmeter for displaying the values at analog channels 0 and 1 or the analog outputs 0 and 1. You use the selector switch to choose the voltage value you want to monitor. There are two separate potentiometers for setting the values for the analog inputs.

**Addressing**

You use the following addresses to address the inputs and outputs in your user program:

| Sensor / Actuator | Version A (DI16, DO16) | Version B (DI32, DO32) | S7-400 (Default addresses) |
|---|---|---|---|
| Switch / M.C.Sw. | IW 0 | IW 0 | IW 28 |
| LEDs | QW 8 | QW 4 | QW 36 |
| Thumb. buttons | IW 4 | IW 2 | IW 30 |
| Digital display | QW 12 | QW 6 | QW 38 |
| Analog channels | PIW 352/354 | PIW 304/306 | PIW 1216/1230 |

## The Conveyor Model




SIMATIC® S7


Date: 12.03.03
File: PRO1_02e.7

SITRAIN Training for Automation and Drives

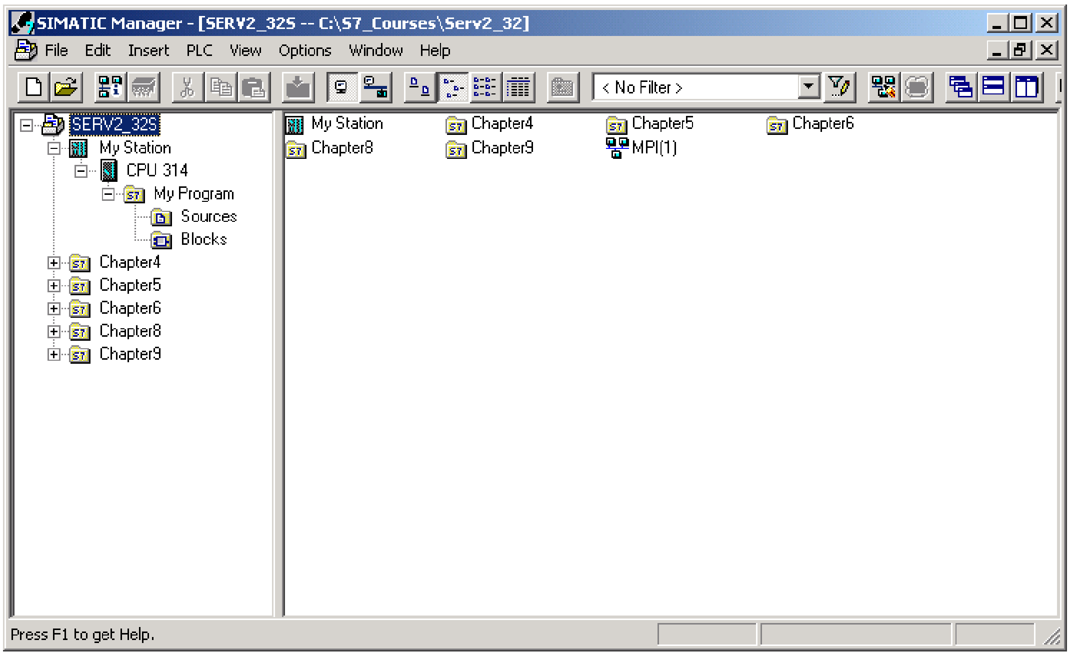
**Design** The slide shows a diagram of the conveyor model with its sensors and actuators.

**Addresses**

| S7-300™ Ver. A (DI16, DO16) | S7-300™ Ver. B (DI32, DO32) | S7-400™ (Default Ad-dresses) | Sensor / Actuator | Symbol |
|---|---|---|---|---|
| I 16.0 | I 8.0 | I 32.0 | Light Barrier at Conv. End | LB |
| I 16.1 | I 8.1 | I 32.1 | Push Button at Bay1, M.C. | T_PB1 |
| I 16.2 | I 8.2 | I 32.2 | Push Button at Bay2, M.C. | T_PB2 |
| I 16.3 | I 8.3 | I 32.3 | Push Button at Bay3, M.C. | T_PB3 |
| I 16.4 | I 8.4 | I 32.4 | Push Button at Conv.End,MC. | T_PB4 |
| I 16.5 | I 8.5 | I 32.5 | Proximity Sensor at Bay 1 | BAY1 |
| I 16.6 | I 8.6 | I 32.6 | Proximity Sensor at Bay 2 | BAY2 |
| I 16.7 | I 8.7 | I 32.7 | Proximity Sensor at Bay 3 | BAY3 |
| Q 20.1 | Q 8.1 | Q 40.1 | Indicator Light at Bay 1 | L_BAY1 |
| Q 20.2 | Q 8.2 | Q 40.2 | Indicator Light at Bay 2 | L_BAY2 |
| Q 20.3 | Q 8.3 | Q 40.3 | Indicator Light at Bay 3 | L_BAY3 |
| Q 20.4 | Q 8.4 | Q 40.4 | Indicator Light at Conv. End | L_END |
| Q 20.5 | Q 8.5 | Q 40.5 | Run Conveyor Right | K_RT |
| Q 20.6 | Q 8.6 | Q 40.6 | Run Conveyor Left | K_LT |
| Q 20.7 | Q 8.7 | Q 40.7 | Horn | K_Horn |

SIEMENS

# The SIMATIC® Manager




SIMATIC S7



Date: 12.03.03
File: PRO1_03E.1

SITRAIN Training for Automation and Drives

## Contents

Page

# Objectives

**Upon completion of this chapter the participant will ...**

... understand the project structure in the SIMATIC® Manager

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

... be familiar with the STEP 7 standard libraries

... be familiar with the STEP7 help system

... be able to create a new project with the SIMATIC® Manager

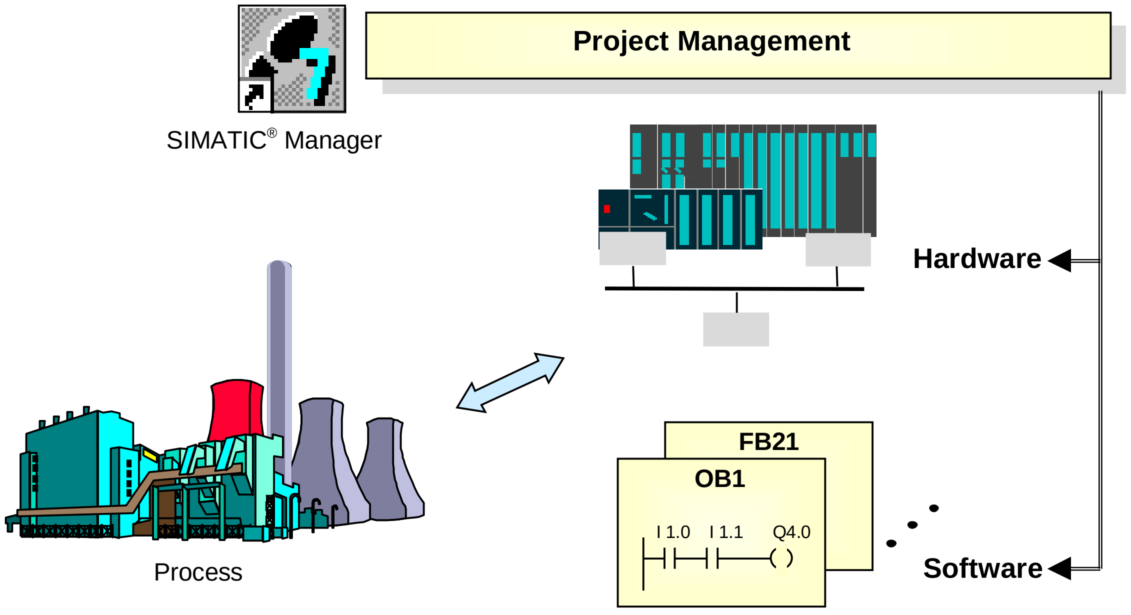
... be able to copy a block with the SIMATIC® Manager

SIMATIC S7


Date: 12.03.03
File: PRO1_03E.2

SITRAIN Training for Automation and Drives

## From Process to Project




SIMATIC S7


Date: 12.03.03
File: PRO1_03E.3

SITRAIN Training for Automation and Drives

**Process**

When you take a closer look at a process you want to automate, you will find that it is made up of a multitude of smaller sections and sub-processes, which are all interlinked and dependent on one another.

The first task is therefore to break down the automation process as a whole into separate sub-tasks.

**Hardware and Software**

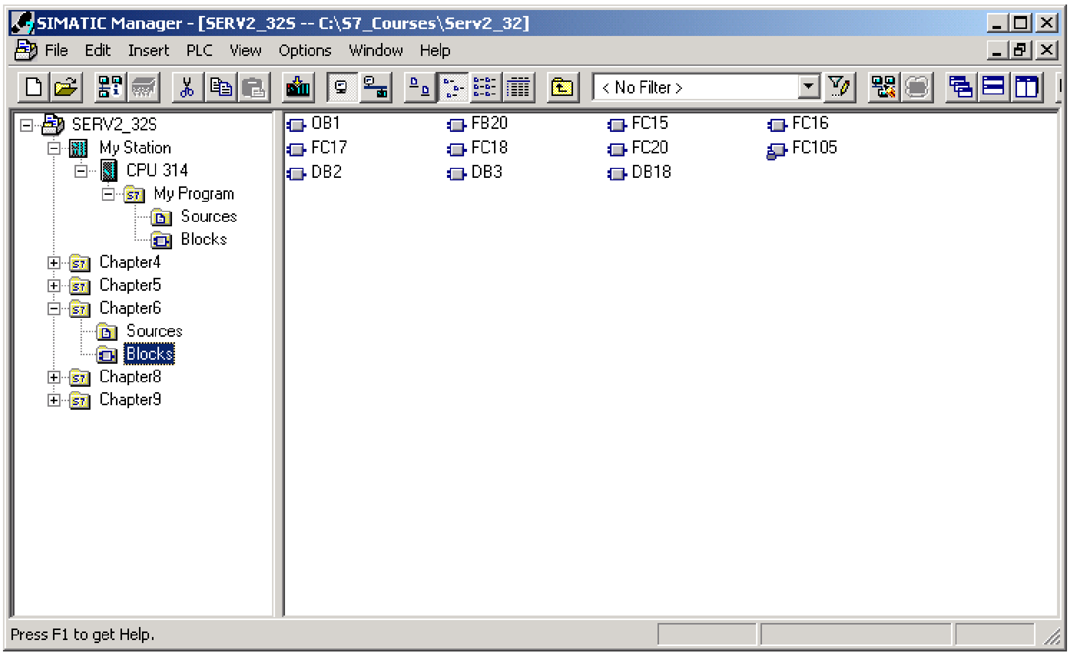
Each sub-task defines certain hardware and software requirements which the the automation system must fulfill:

- Hardware:
  - Number and type of inputs and outputs
  - Number and type of modules
  - Number of racks
  - Capacity and type of CPU
  - HMI systems
  - Networking systems
- Software:
  - Program structure
  - Data management for the automation process
  - Configuration data
  - Communication data
  - Program and project documentation.

**Project**

In SIMATIC® S7 all the hardware and software requirements of an automation process are managed within a project.
A project includes the necessary hardware (+ configuration), network (+ configuration), all the programs, and the entire data management for an automation solution.

## STEP 7 Project Structure




SIMATIC S7


Date: 12.03.03
File: PRO1_03E.4

SITRAIN Training for Automation and Drives

**Multiproject** The multiproject folder is the folder for all projects and libraries of an automation solution that contains one or more STEP 7 projects and optionally, also libraries. That way, the complete solution can be divided into individual, easy to follow projects. The projects within the multiproject can contain objects with cross-project interrelations (such as cross-project S7 connections).

**Project Structure** Data is stored in a project in the form of objects. The objects in a project are arranged in a tree structure (project hierarchy). The tree structure displayed in the project window is similar to that in the Windows Explorer. Only the icons for the objects are different.

**Project Hierarchy**

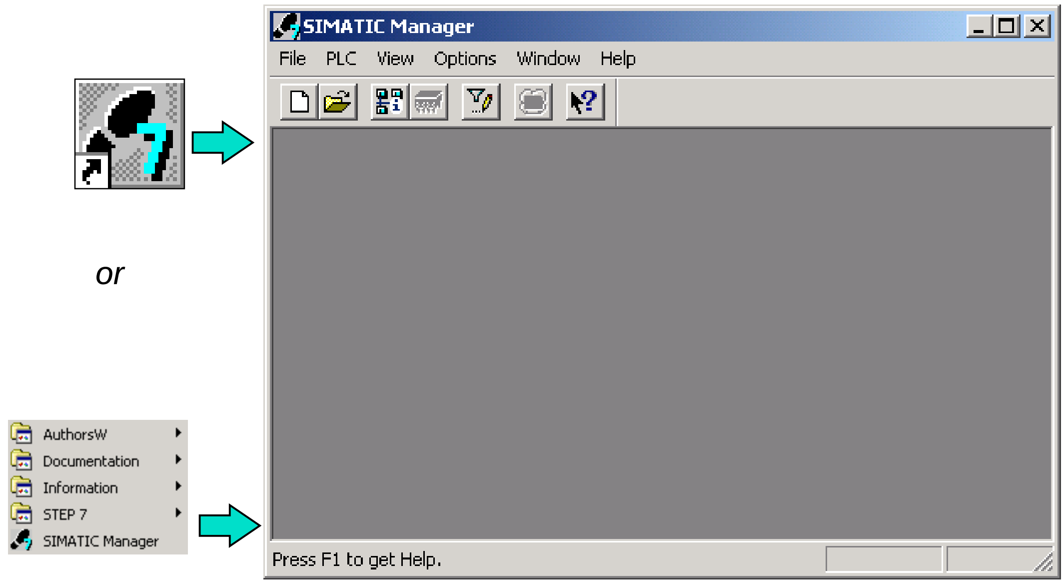
1st. Level: The first level contains the project icon. Each project represents a database where all the relevant project data are stored.

2nd. Level:

- Stations (such as the S7-300™ station) are where information about the hardware configuration and parameter assignment data of modules is stored.
  Stations are the starting point for configuring the hardware.
- S7 Program folders are the starting point for writing programs. All the software for a parameter-assignable module from the S7 range is stored in an S7 Program folder. This folder contains further folders for the program blocks and sources.
- Subnets (MPI,Profibus, Industrial Ethernet) are part of an overall network.

3rd. and subsequent levels: Depends on the object type of the next-higher level.

## Starting the SIMATIC® Manager




SIMATIC S7


Date: 12.03.03
File: PRO1_03E.5

SITRAIN Training for Automation and Drives

**Introduction**

The SIMATIC® Manager is a graphic user-interface for online/offline editing of S7 objects (projects, user program files, blocks, hardware stations and tools).
With the SIMATIC® Manager you can:

- manage projects and libraries,
- activate STEP 7 tools,
- access the PLC online,
- edit memory cards.

**Starting the SIMATIC® Manager**

There is a "SIMATIC® Manager" icon on the Windows desktop and a "SIMATIC® Manager" program item under SIMATIC® in the Start menu.
You activate the program just like all other Windows applications when you double-click the icon  or use the Start menu

*START -> SIMATIC® ->* 

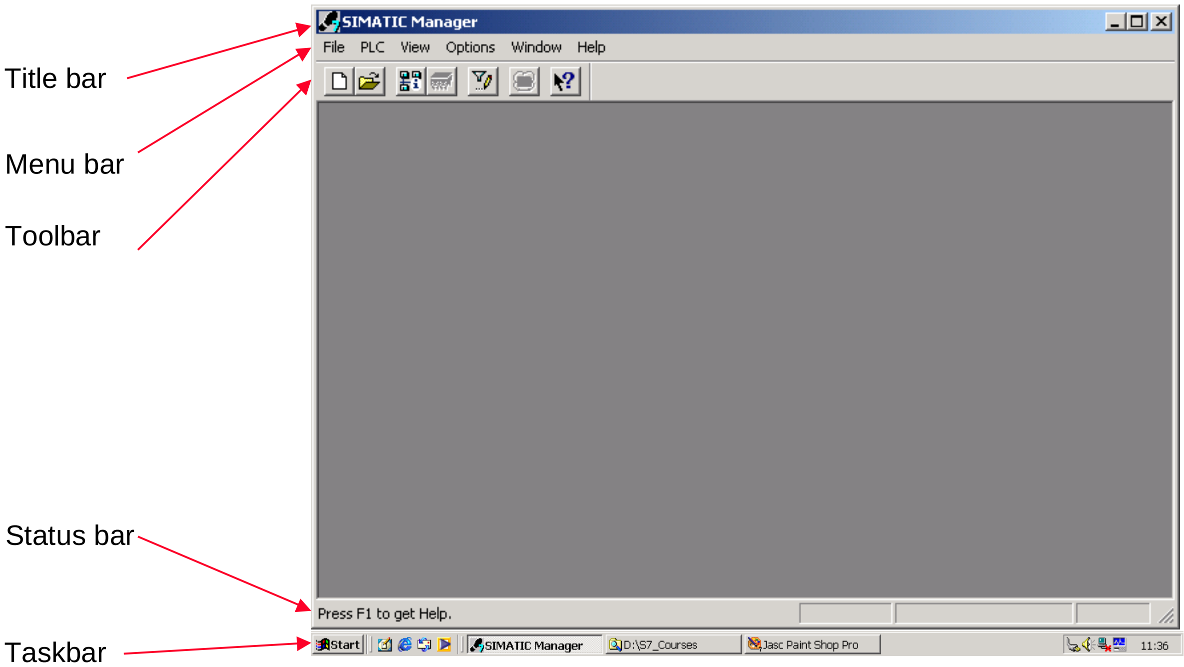

**User Interface**

After installation, the main tool is available as an icon on the Windows desktop. The SIMATIC® Manager manages the S7 objects such as projects and user programs.
When you open an object, the associated tool for editing starts. A double-click on a program block starts the Program Editor, where a block can be edited (object-oriented start)

**Note**

You can always obtain online help for the current window when you press the F1 function key.

## SIMATIC® Manager Menus and Toolbars




SIMATIC S7


Date: 12.03.03
File: PRO1_03E.6

SITRAIN Training for Automation and Drives

**Title Bar** The title bar contains the window title and the buttons for controlling the window.

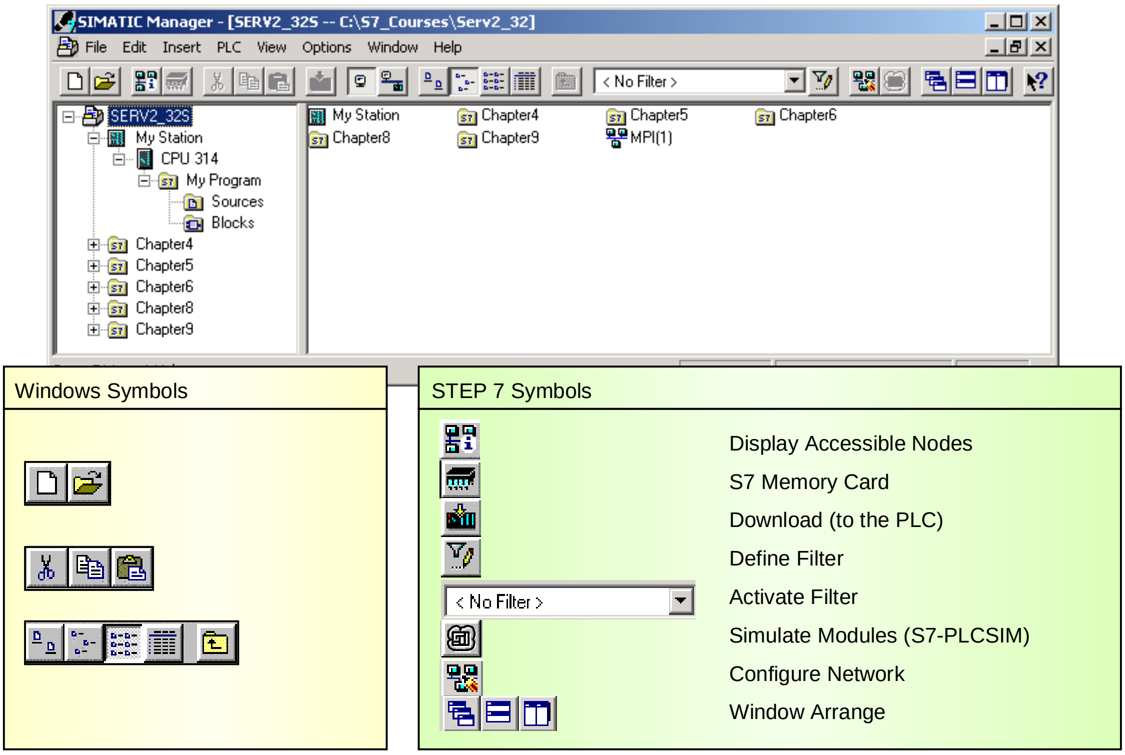
**Menu Bar** Contains all the menus available for the current window.

**Toolbar** Contains the most frequently used tasks as symbols. These symbols are self-explanatory.

**Status Bar** Displays the current status and further information.

**Taskbar** The taskbar contains all open applications and windows as buttons.
You can position the taskbar on either side of the screen by using the left mouse button and dragging it to its new position.

## The Toolbar in the SIMATIC® Manager



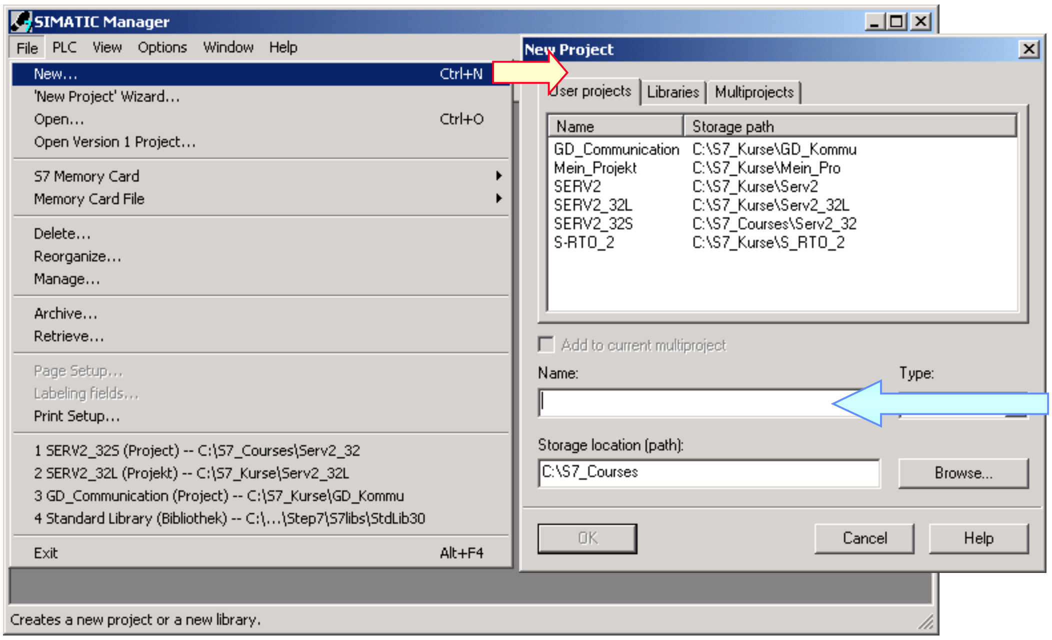

SIMATIC S7


Date: 12.03.03
File: PRO1_03E.7

SITRAIN Training for Automation and Drives

- New (File Menu)
- Open (File Menu)
- Display Accessible Nodes (PLC Menu)
- S7 Memory Card (File Menu)
- Cut (Edit Menu)
- Copy (Edit Menu)
- Paste (Edit Menu)
- Download (PLC Menu)
- Online (View Menu)
- Offline (View Menu)
- Large Icons (View Menu)
- Small Icons (View Menu)
- List (View Menu)
- Details (View Menu)
- Filter Command (View Menu)
- Up One Level
- Simulate Module (Options Menu)
- Arrange, Cascade (Window Menu)
- Arrange, Horizontally (Window Menu)
- Arrange, Vertically (Window Menu)
- Help Symbol

## Creating an S7 Project




Enter project name here and use "OK" to confirm

SIMATIC S7


Date: 12.03.03
File: PRO1_03E.8

SITRAIN Training for Automation and Drives

**Creating a Project**

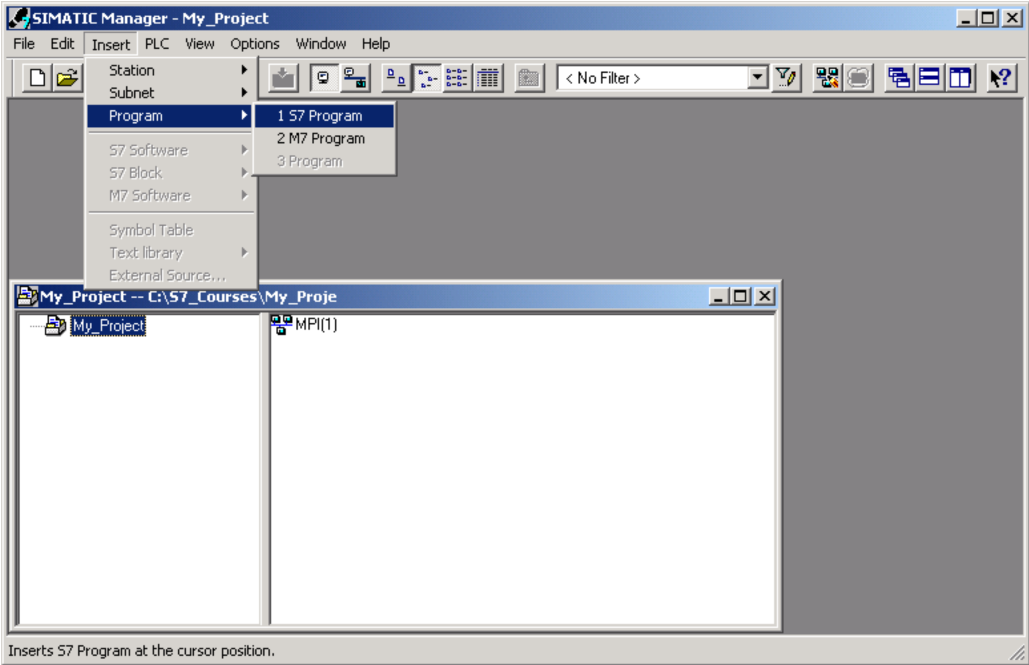
Select the menu options *File -> New* or the symbol in the toolbar to open the "New" dialog box for creating a new project or a new library.

Enter the project name in the "Name" box and click the "OK" button to confirm.

**Notes**

1. The "Storage location (path)" displays the path that was preset in the SIMATIC® Manager under *Options -> Customize*.
2. As of STEP 7 V3.2, the 'New Project' Wizard helps you create a new project.

## Inserting an S7 Program




SIMATIC S7


Date: 12.03.03
File: PRO1_03E.9

**SITRAIN** Training for Automation and Drives

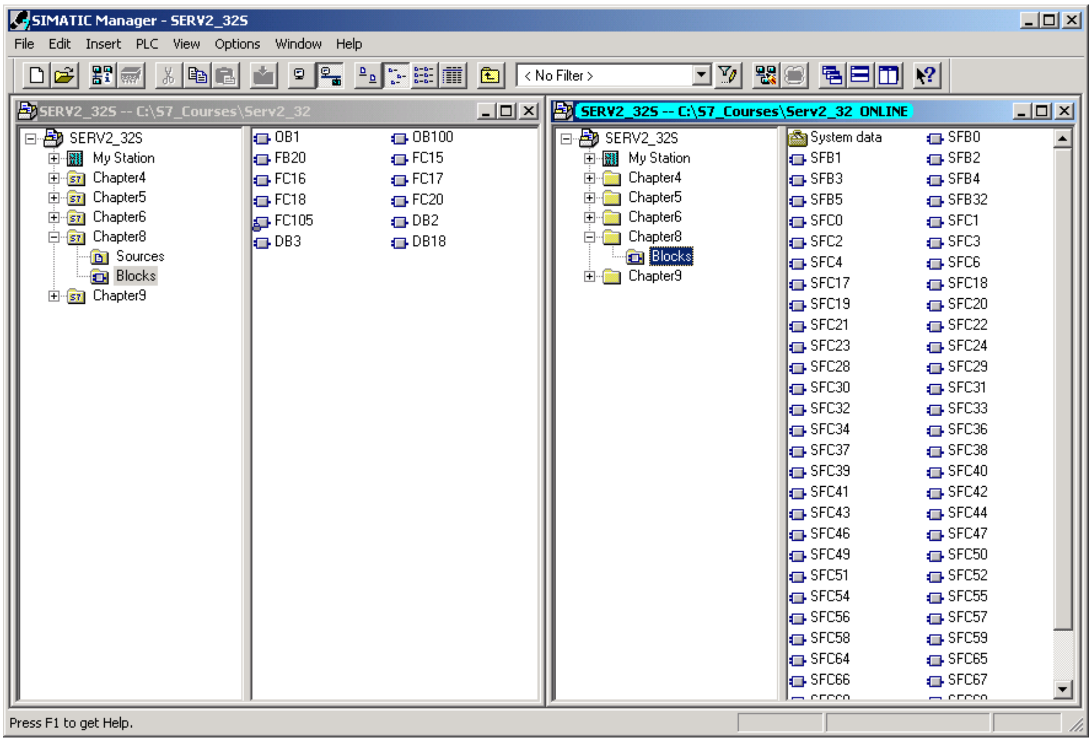
**Inserting a Program** Select the *Insert -> Program -> S7 Program* menu to insert a new program into the current project.

When you insert an object, the system automatically gives it a relevant name, such as "S7 Program(1)".
You can then change this name if you like.

**Note** You use the method described above to create a hardware-independent program.

Programs assigned to particular hardware are described in the "Hardware Configuration" chapter.

## Offline / Online View in the SIMATIC® Manager




SIMATIC S7


Date: 12.03.03
File: PRO1_03E.10

SITRAIN Training for Automation and Drives

**Offline View**

In the project window of the SIMATIC® Manager, the offline view displays the project structure stored on the hard disk of the programming device.
The "S7 Program" folder contains the "Sources" and "Blocks" objects.
The "Blocks" folder contains the system data created with the HWConfig tool and the blocks created with the LAD/STL/FBD Editor.

**Online View**

The online view shows the offline project structure in the left window and in the right window it shows the blocks stored online in the selected CPU.
As a result, the "S7 Program" seen in the online view only contains the "Blocks" folder which contains the following objects:

- System data blocks (SDB)
- User blocks (OB, FC, FB)
- System blocks (SFC, SFB).

**Changing Views**

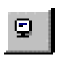
Changing between offline and online view takes place:

- through the *View -> Offline* or *View -> Online* menu items

or

- with the corresponding symbol in the toolbar:

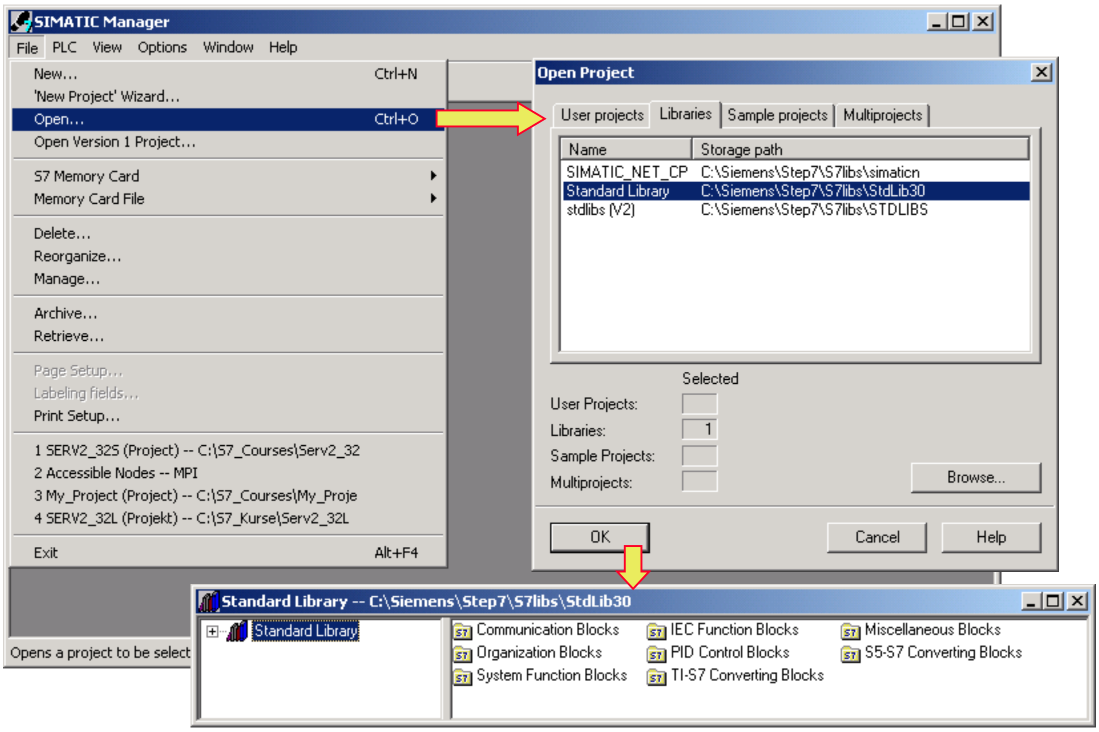
 Online

 Offline

**Note**

You can arrange the "ONLINE" and "OFFLINE" views next to each other or under each other when you use the *Window -> Arrange* option in the menu bar.

## Standard Library




SIMATIC S7


Date: 12.03.03
File: PRO1_03E.11

SITRAIN Training for Automation and Drives

**Introduction** Libraries are used to store reusable blocks.The blocks can be copied into a library from existing projects or they can be created directly in the library independent of projects.

**Standard Library** A Standard Library is installed when you install STEP 7. You can access this standard library from the SIMATIC® Manager (*File -> Open -> Libraries*) or from the Block Editor (*Overviews -> Libraries*). The library contains the following S7 programs:

**Communication Blocks:** FCs (functions) for communication between the CPU and the distributed I/O via communication processors.

**Organization Blocks:** Organization blocks (OBs).

**S5-S7 Converting Blocks:** Blocks that emulate STEP 5 standard function blocks and that are necessary for converting STEP 5 programs.

**TI-S7 Converting Blocks:** Generally usable standard functions such as analog value scaling.

**IEC Function Blocks:** Blocks for IEC functions (IEC: International Electrotechnical Commission), such as for processing time and date information, for string processing and for selecting maximum and minimum.

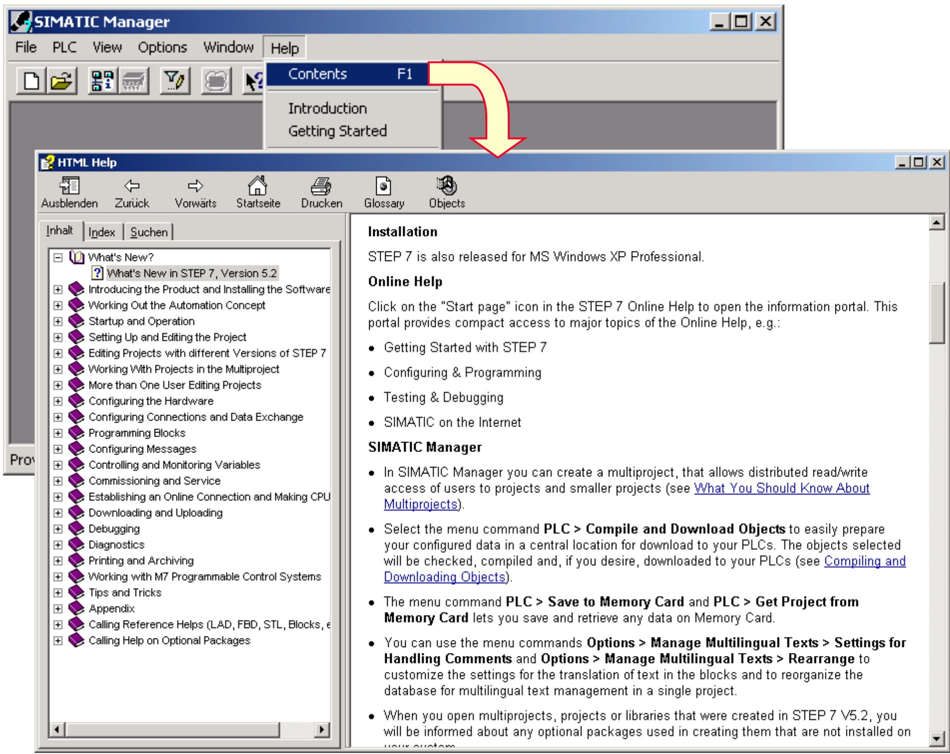
**PID Control Blocks:** Function blocks (FBs) for PID closed-loop controls.

**System Function Blocks:** System functions (SFCs) and System function blocks (SFBs).

**Miscellaneous Blocks:** FCs and Fs for switching between daylight savings time and and standard time (summer and winter times).

**Note** Additional libraries are added when optional software is installed.

## STEP 7 Help System




SIMATIC S7


Date: 12.03.03
File: PRO1_03E.12

SITRAIN Training for Automation and Drives

**Obtaining Help**

There are various ways of obtaining help:

1. You use the *Help - > Contents* menu to activate the general help.
2. You use the F1 function key or the  symbol in the toolbar to start the context-sensitive help.

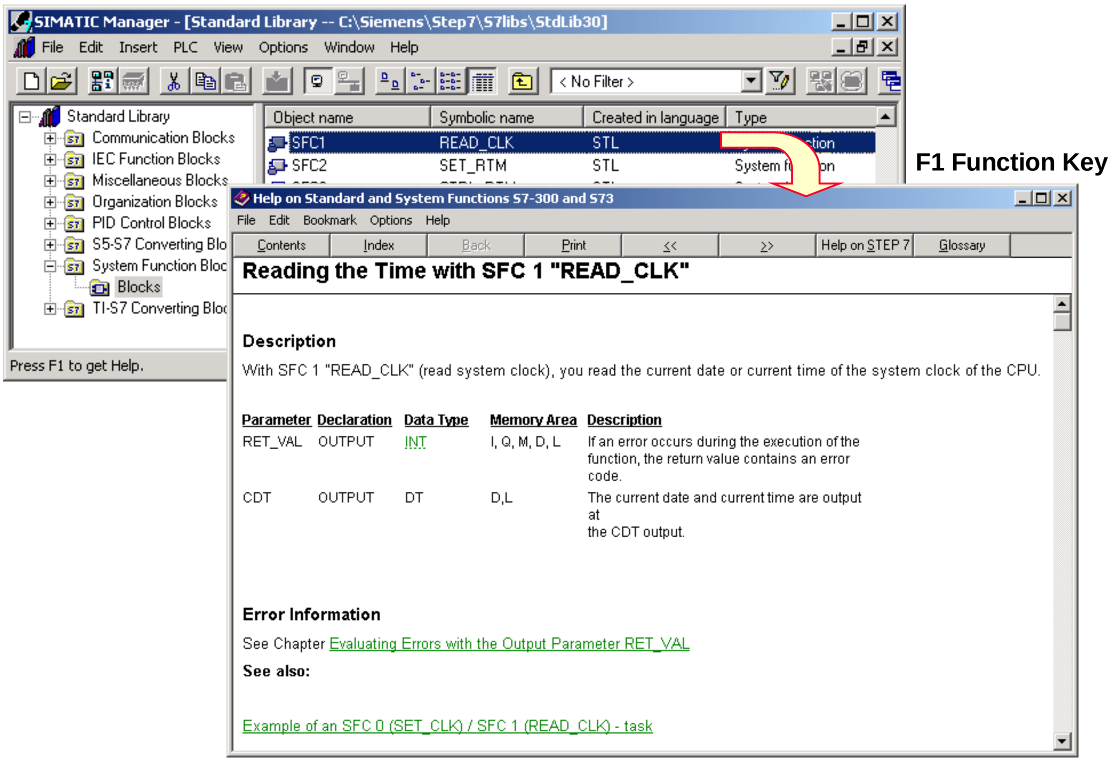
**Tabs**

- "Contents" - Displays a list of help topics under general headings.
- "Index" - Allows you to access the help information by displaying a list of available terms in alphabetical order.
- "Search" - Enables you to look for certain words or expressions in the help topics.

**Hot words**

Certain words are highlighted in green and are underlined with a broken line in the help texts (called "Hot words"). A mouse click on these "Hot words" leads to a further help text with detailed information.

## Context-Sensitive Help in STEP 7




SIMATIC S7


Date: 12.03.03
File: PRO1_03E.13

SITRAIN Training for Automation and Drives

**Context-Sensitive Help ...**

With the context-sensitive help, you can find information directly connected to the selected object. As the slide shows, the object can be a system function or it can be an STL instruction in a block, for example.

**... Activating**

You can activate the context-sensitive help from any tool by selecting the relevant object and then pressing the F1 function key.

Use the "Help on STEP 7" button to jump from the context-sensitive help to the general help.

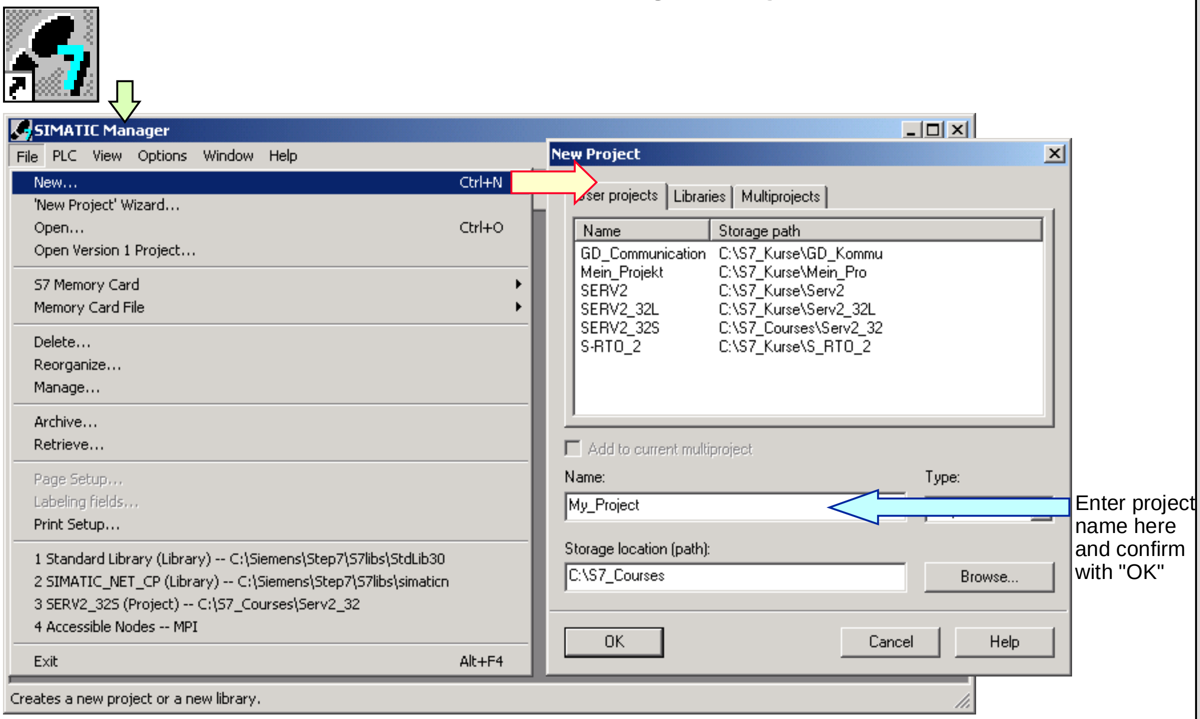
**Note**

You can find additional information on STEP 7 in the electronic manuals. Choose the following menu options to open the electronic manuals:

*Start -> Simatic -> Documentation*

SIEMENS

## Exercise: Creating a Project




SIMATIC S7


Date: 12.03.03
File: PRO1_03E.14

SITRAIN Training for Automation and Drives

**Task**

Delete an old project and create a new project called "My_Project".

**What To Do**

1. Start the SIMATIC® Manager
2. Delete the old project "My_Project" (if it exists):
   *File -> Delete -> User projects -> select "My_Project" in the list -> confirm*
3. Create the new project "My_Project"
   *File -> New... -> User projects -> enter "My_Project" in the Name box -> confirm*

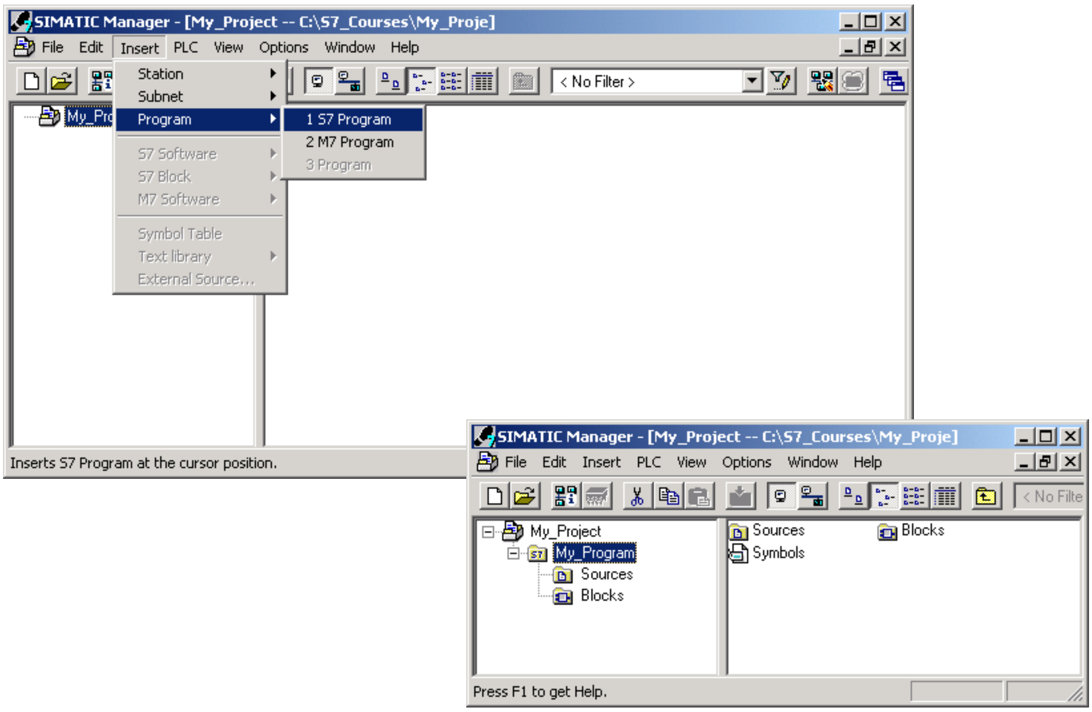
**Notes**

A project represents all the components of an automated system. As a result, a project can contain one or more hardware stations (programmable logic controllers) that are networked using a bus system. Data can then be exchanged between CPUs or communication cards.

In every station, you can install several intelligent modules (function modules or with the S7-400™ up to 4 CPUs as well). These modules usually have their own program folder assigned to them.

You can also create hardware-independent Step 7 program folders. This allows applications to be programmed before the hardware is known. You can later copy hardware-independent S7 programs or individual components (such as individual blocks) to the hardware-dependent S7 program folder or download them to the CPU without a problem.

## Exercise: Inserting an S7 Program




SIMATIC S7


Date: 12.03.03
File: PRO1_03E.15

**SITRAIN** Training for Automation and Drives

**Task**

Insert the S7 Program called "My_Program" into your project "My_Project" .

**What To Do**

1. Insert an S7 Program
   *Select the project called "My_Project" -> Insert -> Program -> S7Program*
2. Change the default S7 Program name "S7 Program (1)" to "My_Program":
   *Click the S7 Program twice (not a double-click!) -> write "My_Program" over the old name*

**Result**

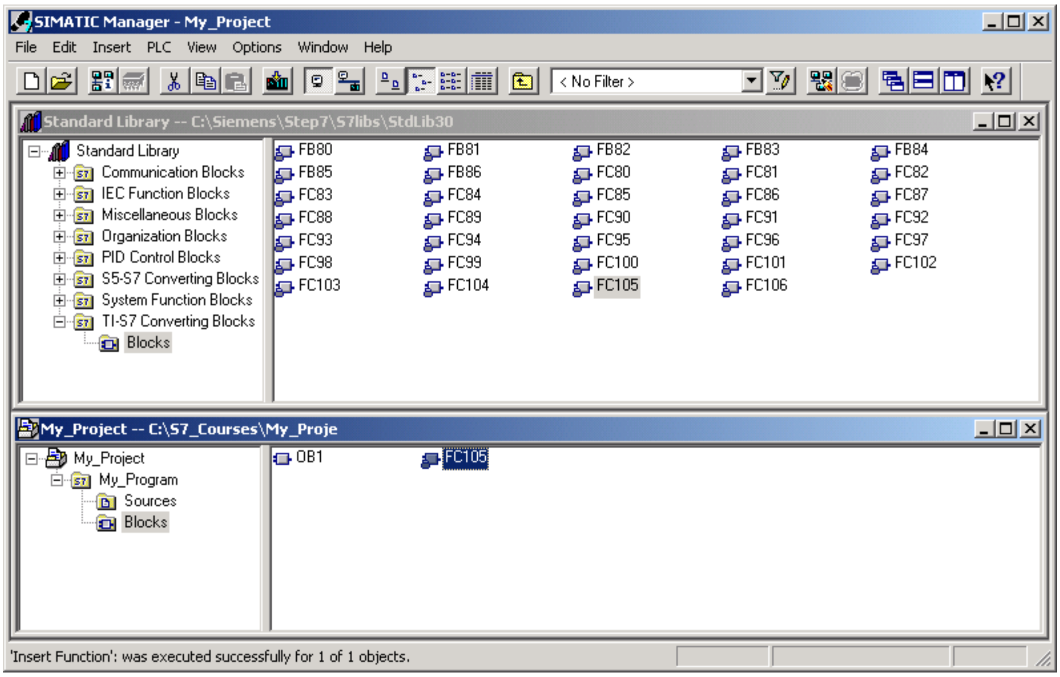
You created the hardware-independent S7 Program called "My_Program" in the project called "My_Project". The OB1 block that is now in your "Blocks" folder was automatically inserted. It has no instructions as of yet.

**Notes**

An S7 Program contains the following objects

- *Blocks*: where logic code is stored (OBs, FCs, FBs and DBs) that can later be downloaded to the CPU
- *Sources:* for storing source programs that are created with text editors, for example, for STL, S7-SCL or S7-HiGraph
- *Symbols*: where you declare symbols (names) for global S7 addresses such as inputs, outputs, bit memories, timers, counters

# Exercise: Copying a Block from the Standard Library




SIMATIC S7



Date: 12.03.03
File: PRO1_03E.16

SITRAIN Training for Automation and Drives

**Task**

For later use, copy the FC105 block from the STEP7 "Standard Library" into the Blocks folder of the S7 Program "My_Program" in the project "My_Project".

**What To Do**

1. Open the "Standard Library" in the SIMATIC® Manager:
   *File > Open... -> select the "Libraries" tab -> choose "Standard Library" in the list -> confirm*
2. In the project "Standard Library" open the Blocks folder of the S7 Program "TI-S7-Converting Blocks"
3. Display your project called "My_Project" and the "Standard Library" at the same time in two windows in the SIMATIC® Manager
   *Window > Arrange > Horizontally*
4. Using drag & drop, copy the FC 105 block from the "Standard Library" into your program folder "My_Program"
5. Close the library.

**Result**

The FC 105 block is stored in the Blocks folder of your S7 Program called "My_Program" in addition to the still empty OB 1.

**Notes**

Libraries are used for storing blocks which contain standardized functions. You can copy the blocks from the library into any project you wish. If the name (number) of the block you are copying already exists, you can rename the library block (number) when you insert the block into your program folder.

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

| Memory Reset Function | Manually | From the PG | After Inserting a Memory Card |
| --- | --- | --- | --- |
| **Request Memory Reset** | - Mode selector in “STOP” position<br>- Hold mode selector in “MRES” position until the “STOP” LED flashes twice (slowly)<br>- Release mode selector (returns automatically to the “STOP” position) | - Mode selector in “RUN-P” position<br>- Menu options: *PLC -> Operating Mode -> Stop*<br>- Menu options: *PLC -> Clear/Reset* | - Mode selector in “STOP” position<br>- Insert Memory Card (slow flashing of "STOP“ LED indicates request for memory reset) |
| **Perform Memory Reset** | - Switch the mode selector quickly to the "MRES" position again and release (fast flashing of "STOP“ LED indicates memory reset being performed) | - Confirm Memory Reset by clicking the “OK” button (fast flashing of "STOP“ LED indicates memory reset being performed) | - Switch the mode selector quickly to the "MRES" position and release (fast flashing of "STOP" LED indicates memory reset being performed) |
| **Perform Warm Restart** | - Switch mode selector to "RUN" or "RUN-P" position | - Menu options: *PLC -> Operating Mode -> Warm Restart* | - Switch mode selector to "RUN" or "RUN-P" position |

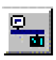
SIMATIC S7

Date: 12.03.03
File: PRO1_03E.17
**SITRAIN** Training for Automation and Drives

**Task**

You are to perform a CPU memory reset and check whether the memory reset was successful.

**What To Do**

- Carry out the memory reset according to the steps in the slide above
- Check the success of the memory reset. The memory reset was successful when only system blocks (SDBs, SFCs, SFBs) are left in the CPU
  *in the SIMATIC® Manager, select the S7 Program folder "My_Program" -> switch to the Online view using* 

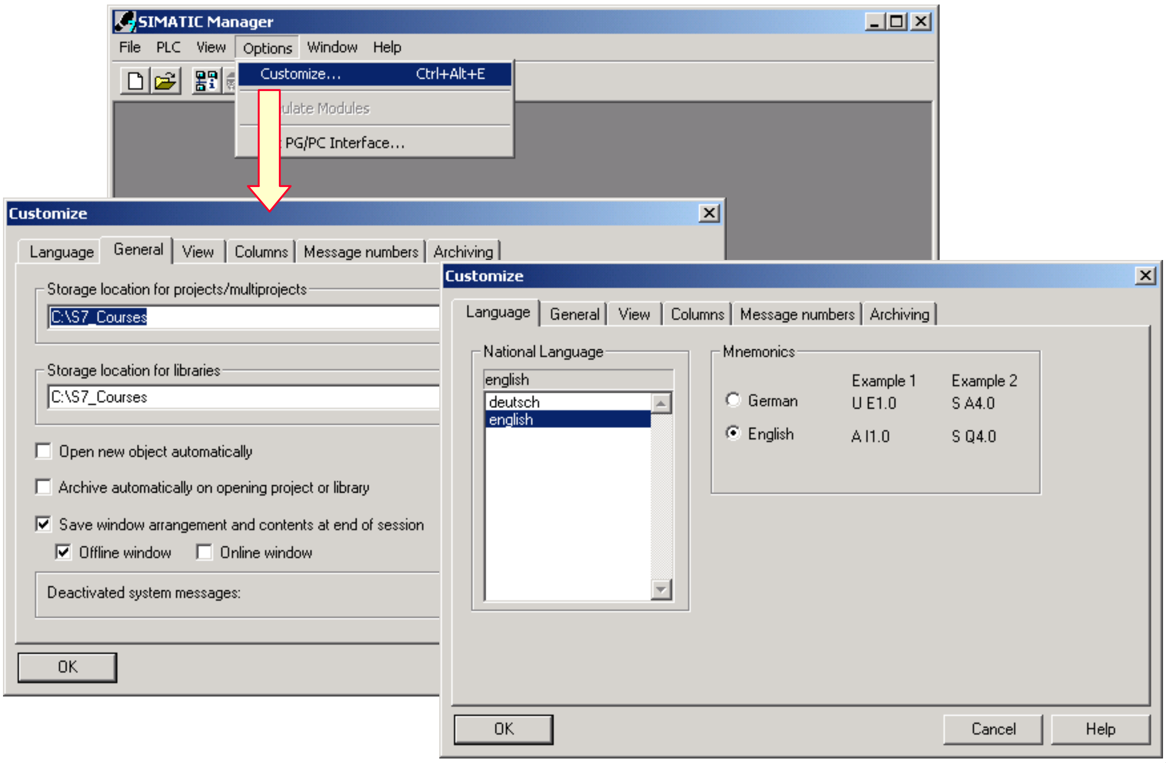
**Notes**

When the CPU memory is reset, all user data in the CPU are deleted.
To make sure that no “old” blocks are left in the CPU, a memory reset of the CPU should be performed. The following takes place during a memory reset:

- All user data are deleted
  (with the exception of the MPI parameter assignments).
- Hardware test and initialization
- If an Eprom memory card is installed, the CPU copies the EPROM contents back into the internal RAM after the memory reset.
- If no memory card is installed, the preset MPI address is retained. If, however, a memory card is installed, the MPI address stored on it is loaded.
- The contents of the diagnostic buffer (which can be displayed with the PG) are retained.

**SIEMENS**

## SIMATIC® Manager Customizing Options




SIMATIC S7


Date: 12.03.03
File: PRO1_03E.18

**SITRAIN** Training for Automation and Drives

**"Language" Tab**

- Language: You can select the language you want to use for the SIMATIC® Manager, menus, dialog boxes, help, etc. (Only the languages that have been installed appear in the list.)
- Mnemonics: You can select the mnemonics you want to use for programming the S7 blocks.

**"General" Tab**

Basic settings for editing projects and libraries:

- Storage location for projects/multiprojects is where you specify the directory in which you want to store your user projects.
- Storage location for libraries is where you specifiy the directory in which you want to store your user libraries.
- Further options for inserting objects, opening projects and for window arrangement will be dealt with later.
- Deactivated system messages
  By pressing the button “Activate“ you can reactivate all system messages that were switched-off in a window when the option “Always display this message….“ was chosen.

**"View" Tab**

This is where you specify how project objects are to be displayed on the screen.

**"Columns" Tab**

This is where you specify which columns are to be displayed when the Detail view is switched-on (see “Help“).

**"Message numbers" Tab**

This is where you specify the type of message number assignment for future new projects. The default setting “No default setting“ should only be changed if ProTool, WinCC or CPU messages are used.

**"Archive" Tab**

The archiving of projects will be discussed in the Chapter “Documenting, Saving, Archiving“.

# Hardware Configuration


SIMATIC® S7


Date: 12.03.03
File: PRO1_04E.1

**SITRAIN** Training for Automation and Drives

**Contents** Page

SIEMENS

# Objectives

**Upon completion of this chapter the participant will ...**

... be able to insert a hardware station into a project

... 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

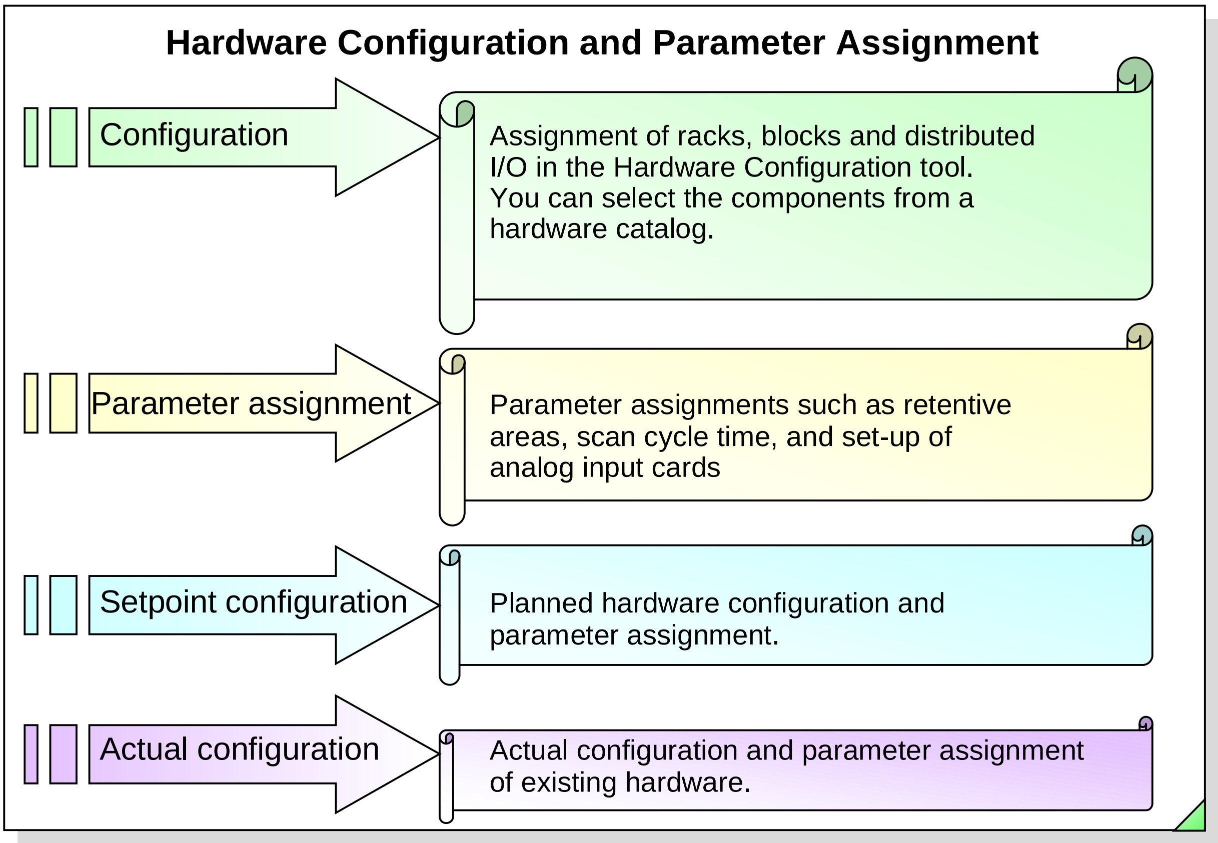
... be familiar with the addressing of S7-300™ input and output modules

SIMATIC® S7



Date: 12.03.03
File: PRO1_04E.2

SITRAIN Training for Automation and Drives




SIMATIC® S7


Date: 12.03.03
File: PRO1_04E.3

SITRAIN Training for Automation and Drives

**HW Configuration**

The modules are supplied from the factory with preset parameters. If these default settings are OK, a hardware configuration is not necessary.

A configuration is necessary:

- if you want to modify preset parameters or addresses of a module (such as to enable the hardware interrupt of a module)
- if you want to configure communication connections
- with stations that have distributed peripherals (PROFIBUS-DP)
- with S7-400™ stations that have several CPUs (multicomputing) or expansion racks
- with fault-tolerant programmable logic controllers (option package).

**Setpoint Configuration**

When you configure a system, a setpoint configuration is created. It contains a hardware station with the planned modules and the associated parameters. The PLC system is assembled according to the setpoint configuration. During commissioning, the setpoint configuration is downloaded to the CPU.

**Actual Configuration**

In an assembled system, the actual existing configuration and parameter assignment of the modules can be uploaded from the CPU. This creates a new HW station in the project.

A configuration upload is necessary, for example, if the project structure does not exist locally at the PG. After the actual configuration is read out, you can set parameters and add part numbers.

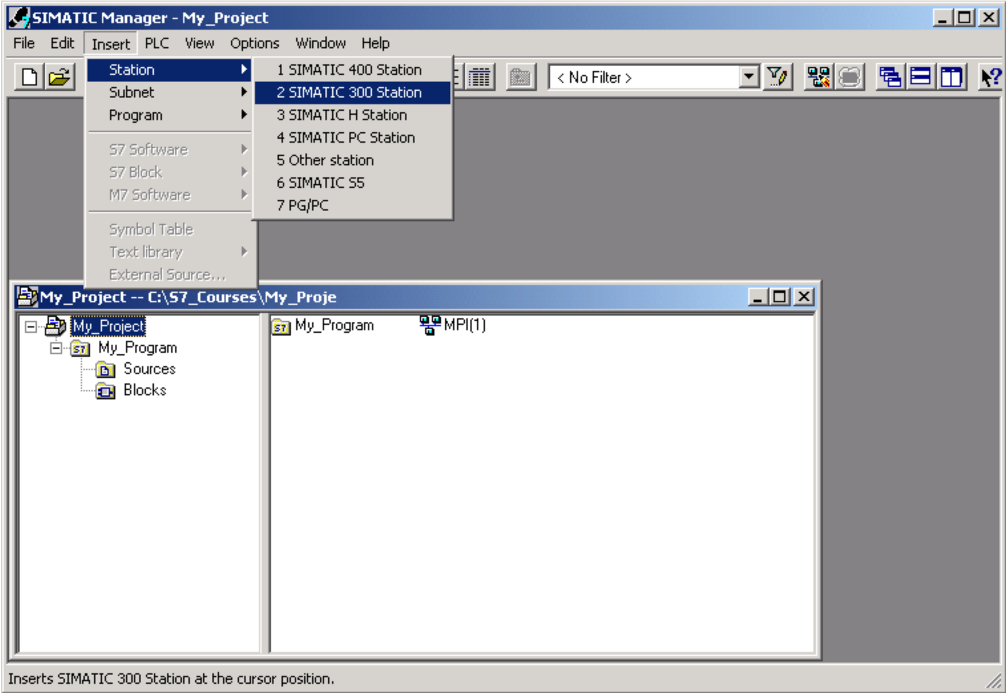
**Notes**

With the S7-400™, the CPU can be assigned parameters in such a way, that when there are differences between the setpoint configuration and the actual configuration, the CPU startup is interrupted.

To call the HW Config tool, there must be a hardware station in the SIMATIC® Manager.

**SIEMENS**

# Inserting a Station




SIMATIC® S7



Date: 12.03.03
File: PRO1_04E.4

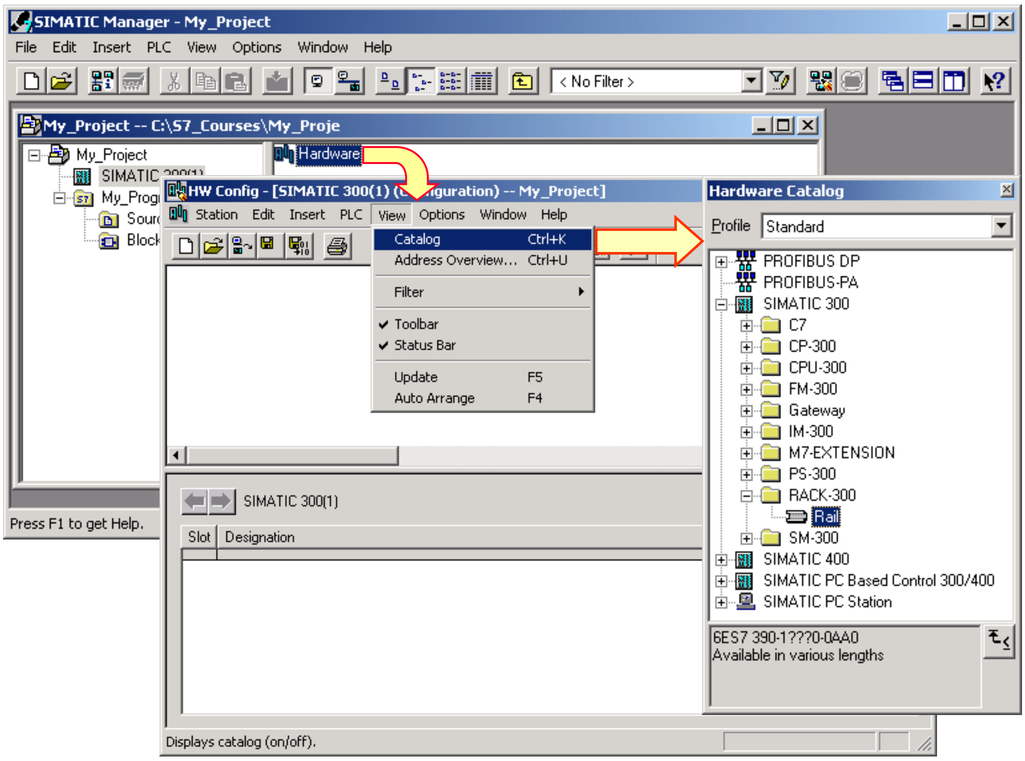
**SITRAIN** Training for Automation and Drives

## Insert Station

You insert a new station in the current project by selecting the menu options *Insert -> Station -> SIMATIC® 300 Station or SIMATIC® 400 Station*.

You can then change the name that is automatically given to this station - "SIMATIC® 300 (1)" - to one of your choice.

## Starting the HW Configuration Editor




SIMATIC® S7


Date: 12.03.03
File: PRO1_04E.5

SITRAIN Training for Automation and Drives

**HW Config** This tool helps you configure, assign parameters to and diagnose the hardware.

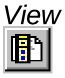
**Starting HW Config** To start the HW Config tool:

- select a hardware station in the SIMATIC® Manager and choose the *Edit --> Open Object* menu or
- double-click the hardware object. 

**"Hardware Configuration"** This is a window in the "HW Config" application you use for inserting components from the "Hardware Catalog" window.

The title bar of this window contains the name of the project and the station name.

**"Hardware Catalog"** To open the catalog:

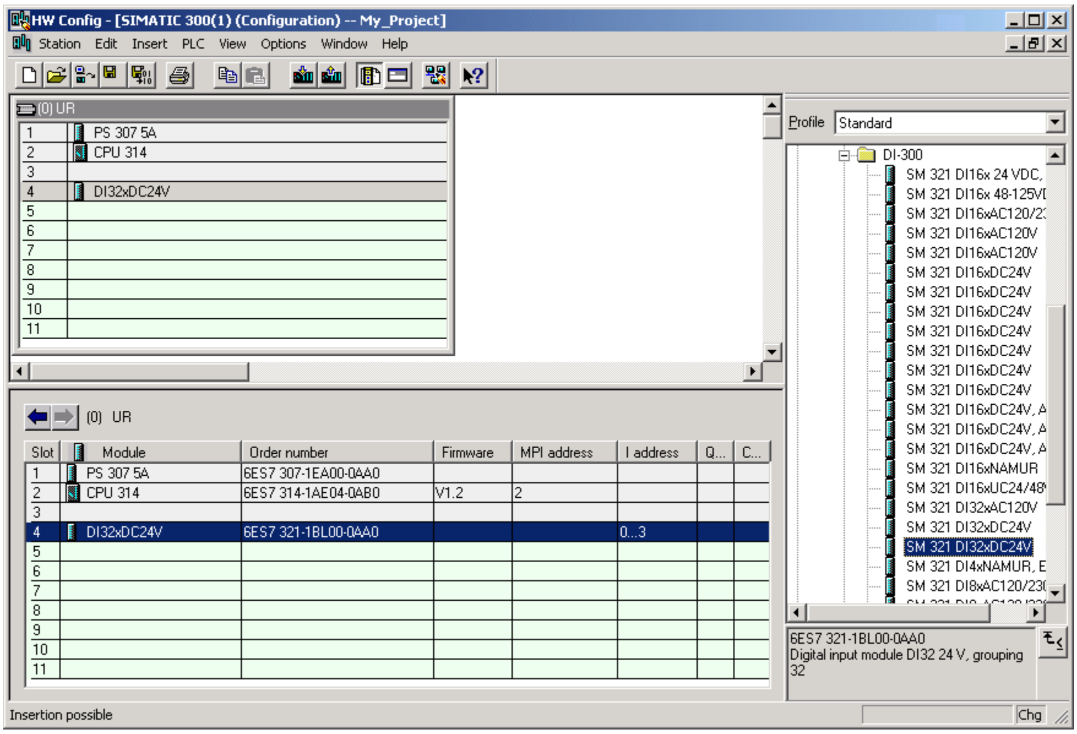
- select the *View -> Catalog* menu or
- click the  icon in the toolbar.

If "Standard" is selected as the catalog profile, all racks, modules and interface modules are available in the "Hardware Catalog" window.

You can create your own catalog profiles containing frequently used elements by selecting the menu options *Options -> Edit Catalog Profiles*.

You can add Profibus Slaves that do not exist in the catalog later on. To add slaves, you use GSE files that are provided by the manufacturer of the slave device. The GSE file contains a description of the device. To include the slave in the hardware catalog, use the *Options -> Install New GSE Files* menu and then *Options -> Update Catalog.* You will find the new devices in the catalog under Profibus, additional field devices.

# Generating a Hardware Setpoint Configuration




SIMATIC® S7


Date: 12.03.03
File: PRO1_04E.6

SITRAIN Training for Automation and Drives

**Generating a Setpoint Configuration**

This means specifying how the modules are to be arranged in the rack. This configuration, specified by you, is referred to as the setpoint configuration.

**Rack**

For example, you open a SIMATIC® 300 station in the Hardware Catalog. Opening the "RACK-300" folder shows the icon for a DIN rail. You can insert this in the "Hardware Configuration" window by double-clicking on it (or using drag & drop).

Two rack component lists then appear in the two-part window: a plain list in the top part and a detailed view with order numbers, MPI addresses and I/O addresses in the bottom part.

**Power Supply**

If a load current power supply is required double click or use drag & drop to insert the appropriate "PS-300" module from the catalog in slot no.1 in the list.

**CPU**

You select the CPU from the "CPU-300" folder, for example, and insert it in slot no. 2.

**Slot No. 3**

Slot no. 3 is reserved as the logical address for an interface module (for multi-tier configurations).

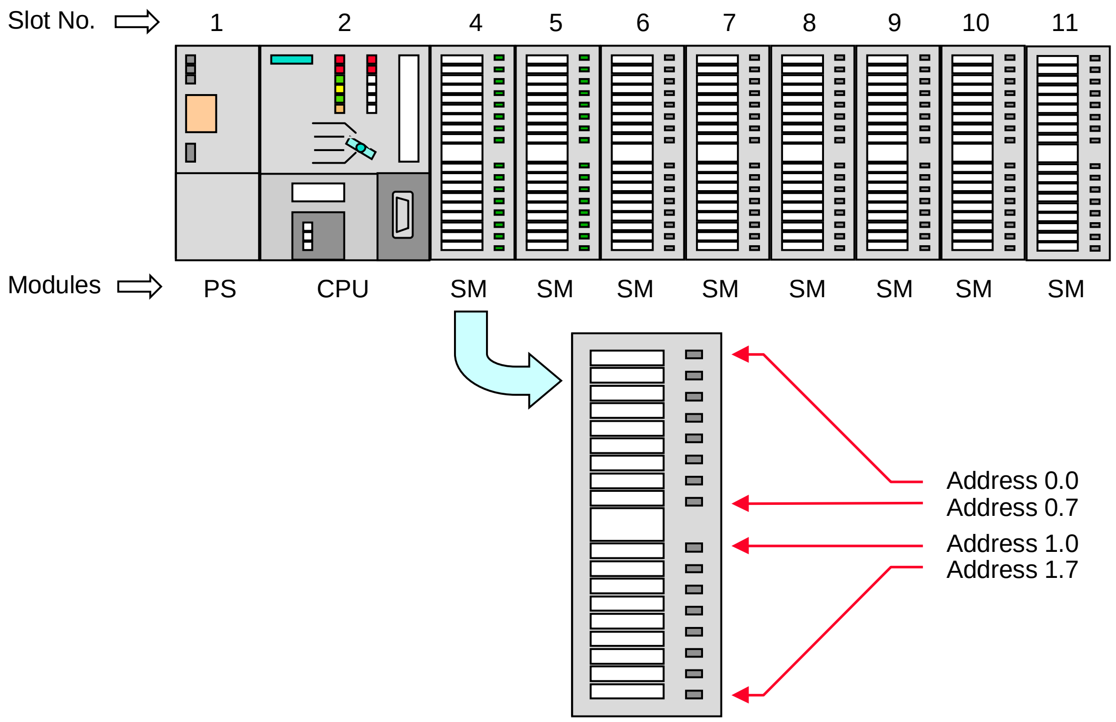
If this position is to be reserved in the actual configuration for the later installation of an IM, you must insert a dummy module DM370 (DUMMY).

**"Inserting" Modules**

From slot no. 4 onwards, you can insert a choice of up to 8 signal modules (SM), communications processors (CP) or function modules (FM) from the HardwareCatalog using drag & drop or with a double-click.

The slots on which the selected module can be inserted are automatically highlighted in green.

## Addressing S7-300™ Modules




SIMATIC® S7


Date: 12.03.03
File: PRO1_04E.7

SITRAIN Training for Automation and Drives

**Slot Numbers** The slot numbers in the rack of an S7-300™ simplify addressing in the S7-300™ environment. The position of the module in the rack determine the first address on a module.

**Slot 1** Power supply. This is the first slot by default.
A power supply module is not absolutely essential. An S7-300™ can also be supplied with 24V directly.

**Slot 2** Slot for the CPU.

**Slot 3** Logically reserved for an interface module (IM) for multi-tier configurations using expansion racks. Even if no IM is installed, it must be included for addressing purposes.
You can physically reserve the slot (such as for installing an IM at a later date) if you insert a DM370 dummy module.

**Slots 4-11** Slot 4 is the first slot that can be used for I/O modules, communications processors (CP) or function modules (FM).
Addressing examples:

- A DI module in slot 4 begins with the byte address 0 .
- The top LED of a DO module in slot 6 is called Q8.0 .

**Note** Four byte addresses are reserved for each slot. When 16-channel DI/DO modules are used, two byte addresses are lost in every slot!

## DI/DO Addressing in Multi-Tier Configurations

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Rack 3 | PS | | IM (Receive) | 96.0 to 99.7 | 100.0 to 103.7 | 104.0 to 107.7 | 108.0 to 111.7 | 112.0 to 115.7 | 116.0 to 119.7 | 120.0 to 123.7 | 124.0 to 127.7 |
| Rack 2 | PS | | IM (Receive) | 64.0 to 67.7 | 68.0 to 70.7 | 72.0 to 75.7 | 76.0 to 79.7 | 80.0 to 83.7 | 84.0 to 87.7 | 88.0 to 91.7 | 92.0 to 95.7 |
| Rack 1 | PS | | IM (Receive) | 32.0 to 35.7 | 36.0 to 39.7 | 40.0 to 43.7 | 44.0 to 47.7 | 48.0 to 51.7 | 52.0 to 55.7 | 56.0 to 59.7 | 60.0 to 63.7 |
| Rack 0 | PS | CPU | IM (Send) | 0.0 to 3.7 | 4.0 to 7.7 | 8.0 to 11.7 | 12.0 to 15.7 | 16.0 to 19.7 | 20.0 to 23.7 | 24.0 to 27.7 | 28.0 to 31.7 |
| Slot | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |

SIMATIC® S7


Date: 12.03.03
File: PRO1_04E.8

**SITRAIN** Training for Automation and Drives

**Multi-Tier Configurations**

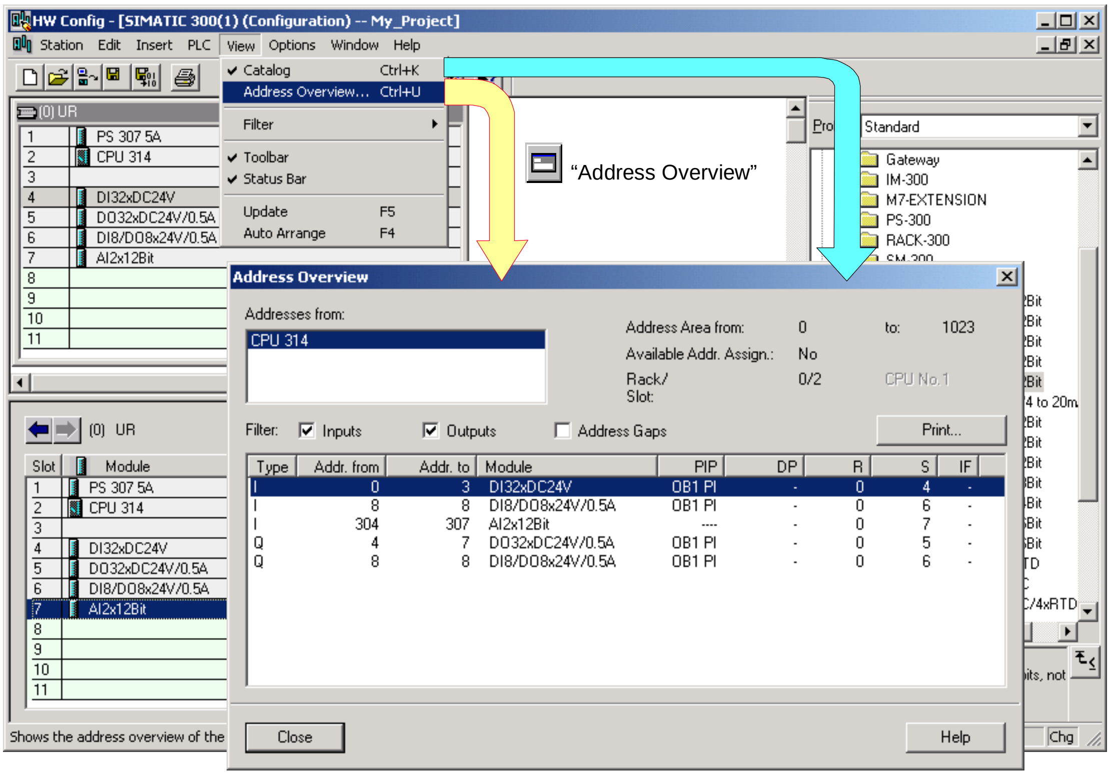
The slots also have fixed addresses in a multi-tier configuration.

Examples:

- Q7.7 is the last bit of a 32-channel DO module plugged into slot 5 of rack 0.
- IB105 is the second byte of a DI module in slot 6 of rack 3.
- QW60 is the first two bytes of a DO module in slot 11 of rack 1.
- ID80 is all four bytes of a 32-channel DI module in slot 8 in rack 2.

SIEMENS

# Module Address Overview




SIMATIC® S7


Date: 12.03.03
File: PRO1_04E.9

SITRAIN Training for Automation and Drives

**Address Overview**

Views the I/O addresses of the station configured.
Select: *View -> Address Overview …*

Abbreviations:

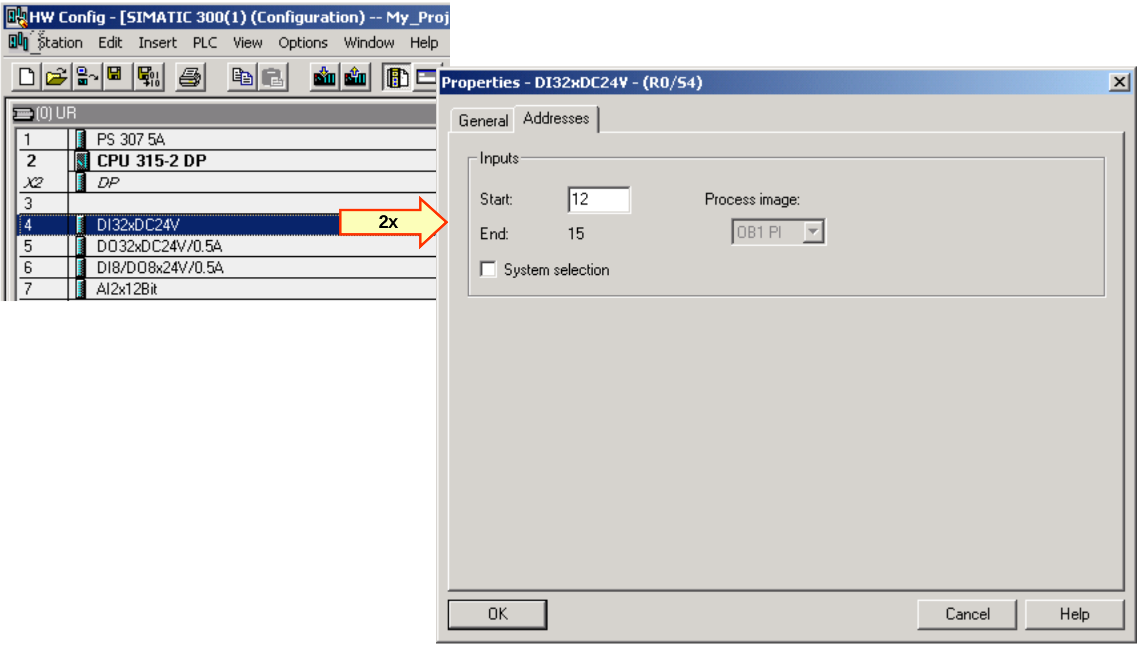
**R** **R**ack number

**S** **S**lot number of the relevant module

**DP** Relevant only when **D**istributed **P**eripherals (I/O) are used

**IF** **I**nter**f**ace module ID when programming the M7 system (in C++).

## Variable Addressing







**Slot dependent Addressing**

The modules are assigned fixed slot-dependent addresses with the S7-300 (CPUs without DP interface) and S7-400™ (without hardware configuration).

**Variable Addressing**

With the S7-300™ (CPUs with integrated DP interface) and with the S7-400™, you can assign parameters to the starting addresses of the modules.

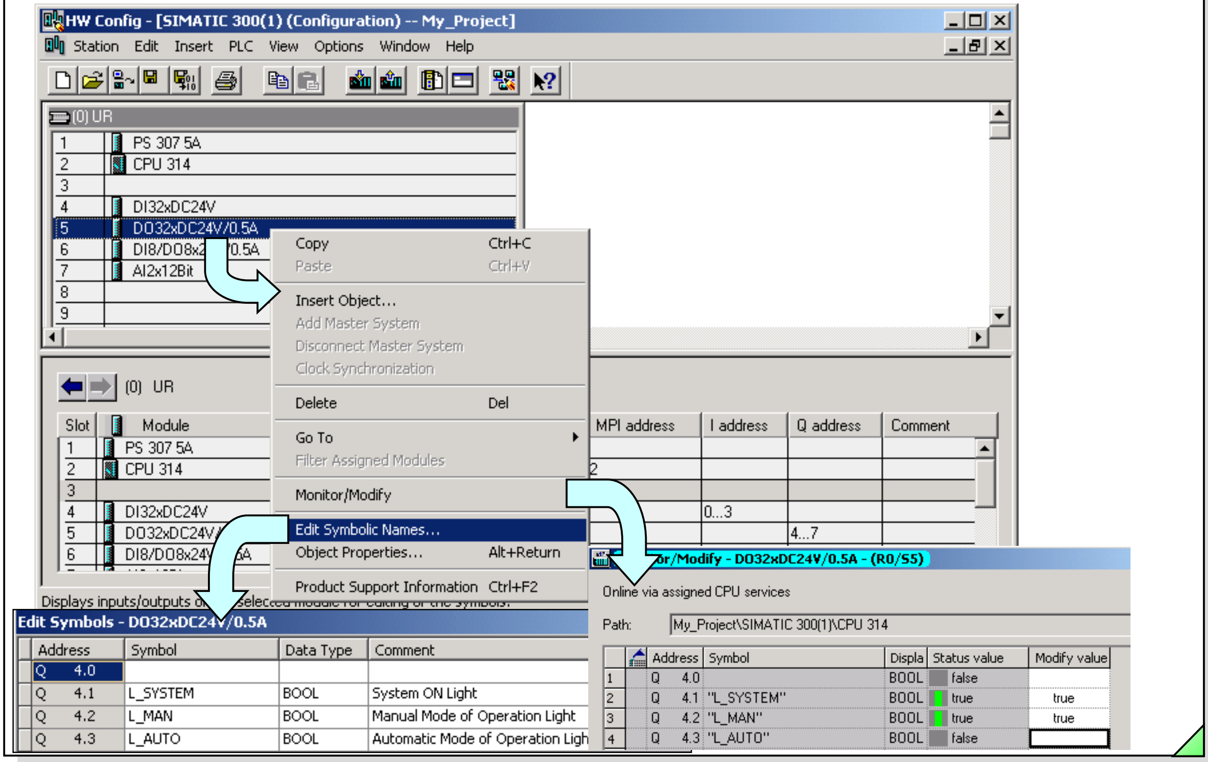
**What to Do**

When you double-click a digital or an analog module, the parameter assignment screen is opened. After you choose the "Addresses" tab, you can cancel "System selection". You can now define the starting address in the "Start" box. If the address is already used, an error message is triggered.
Part process images can be defined only in the S7-400™. That way, specific inputs and outputs (such as time-critical signals) can be combined into one group. A system function triggers the updating of a part process image in the user program.

**Note**

After a CPU memory reset, the parameters, and therefore also the addresses are lost. This means that the slot-dependent addresses of the S7-300™ or the default addresses of the S7-400™ are valid once more.

## HW Config: Edit Symbolic Names, Monitor/Modify Variables




SIMATIC® S7


Date: 12.03.03
File: PRO1_04E.11

SITRAIN Training for Automation and Drives

**Edit Symbolic Names** You can directly access the symbol table from the "HW Config" tool. This allows you to assign symbolic names to the inputs and outputs during hardware configuration or at a later date when you can make suppliments or changes. You open the symbol table with a right mouse click on the module. Select *Edit Symbolic Names* in the follow-up box. A section of the symbol table with the relevant addresses is then opened.

**Monitor/Modify Variables** You can monitor or modify the addresses of the configured modules directly from the HW Config tool. The signals of the input modules can be "checked" and the signals of the output modules can be "controlled" using the Monitor/Modify (Variables) function.

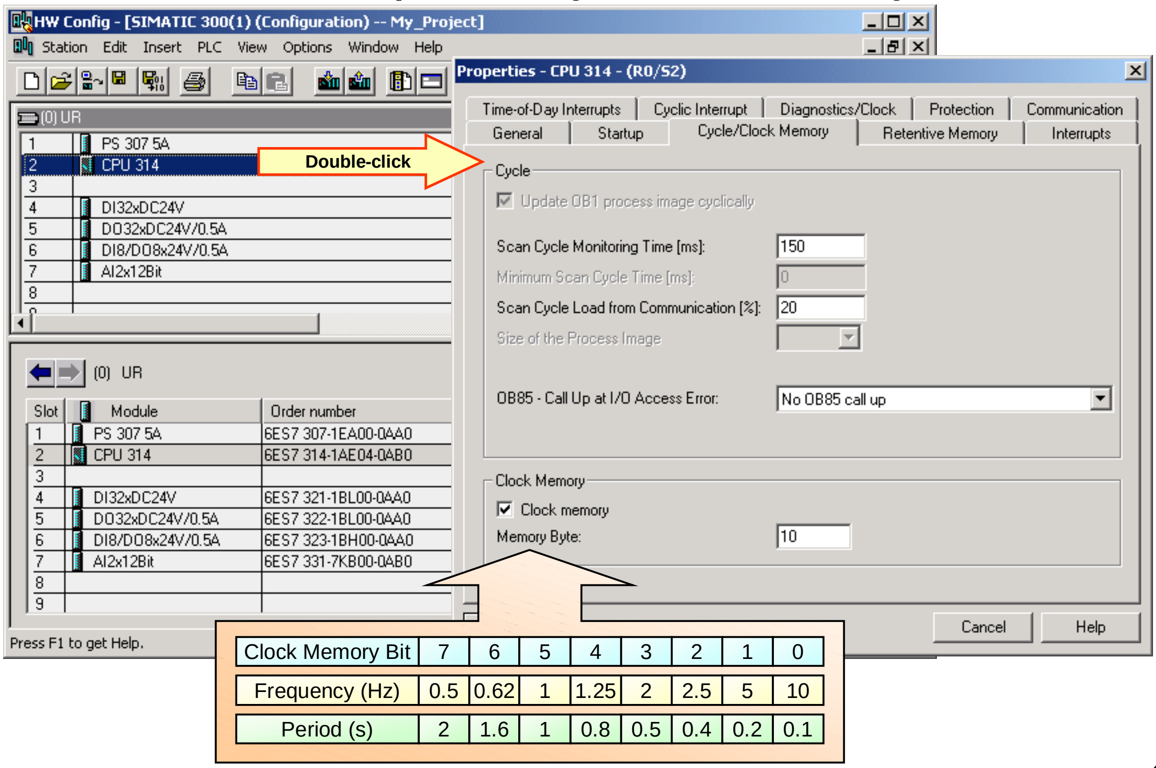
**Product Support Information** Directly from the Internet, you can fetch information on modules or components from the Product Support pages. As well, it is also possible to update HW Config by incorporating individual components such as new CPUs or new DP devices into the current STEP 7 version.

*Requirements*:

The PG/PC has an Internet connection, a browser for displaying Internet pages, and the function is enabled in the HW Config Settings along with the specification of the Internet address.

**Note** "Symbolic Addressing" and the editing of symbol tables is dealt with in depth in the "Symbols" chapter. The Monitor / Modify Variables function is dealt with in the "Troubleshooting" chapter.

## CPU Propertires: 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 |

SIMATIC® S7


Date: 12.03.03
File: PRO1_04E.12

SITRAIN Training for Automation and Drives

**Cycle**

- "Scan cycle monitoring time (ms):"
  - If this time is exceeded, the CPU goes into the STOP mode. Possible causes why this time is exceeded: communications processes, frequently from interrupt events, errors in the CPU program.
  - If you have programmed an error OB 80, the scan cycle time is doubled. After that, the CPU also goes into the STOP mode.
- "Cycle load from communication (%):"
  - Communication (such as data transmission to another CPU through MPI or test functions the PG/PC triggered) is restricted to the specified percentage of the current scan cycle time.
  - Restricting the cycle load can slow down communication between the CPU and PG.
  - Example: Restricting communication to 20% results in a maximum communication load of 20ms for a scan cycle time of 100ms.

**Size of the Process Image**

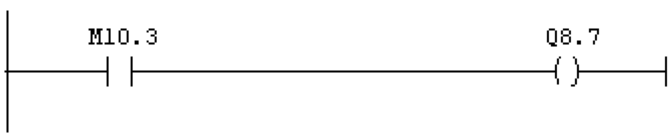
With the CPU 318-2 and several S7-400™ CPUs, you can specify the size of the process image (in bytes). The process image area always begins with input and output byte 0.

**Clock Memory**

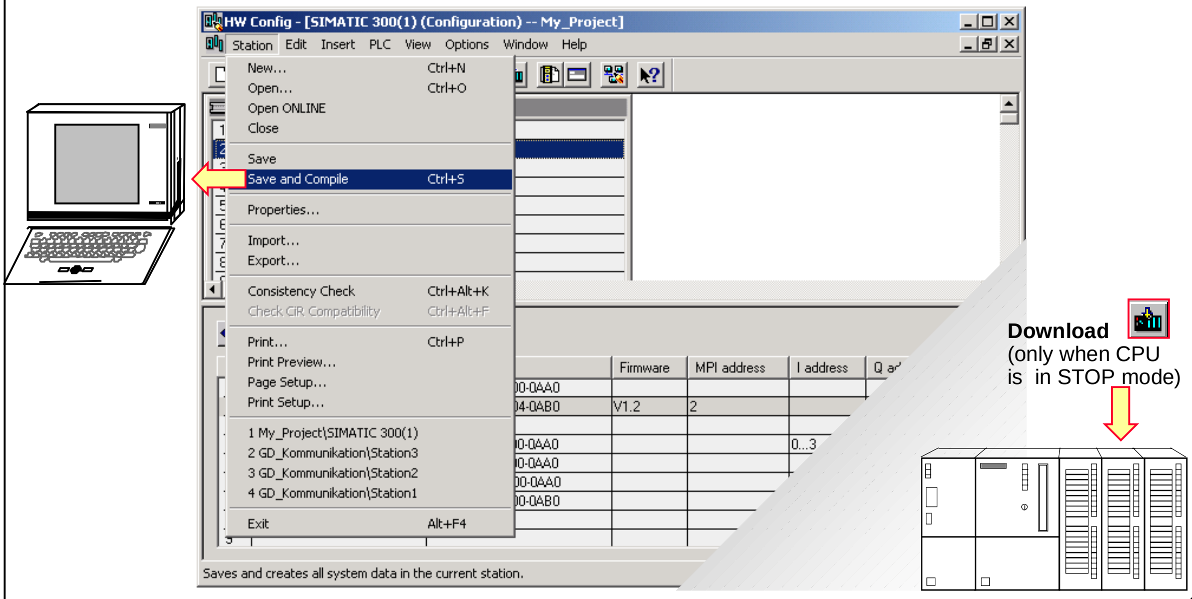
Clock memories are bit memories that change their binary value periodically (pulse-to-pause ratio 1:1).

Each bit in the clock memory byte is assigned a particular period/frequency.

Example of a flashing light with a flashing frequency of 2Hz:



# Saving the HW Setpoint Configuration and Downloading it in Module

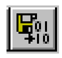



SIMATIC® S7


Date: 12.03.03
File: PRO1_04E.13

SITRAIN Training for Automation and Drives

**Save** You select the *Station->Save* menu to save the current configuration in the current project (without generating system data blocks).

**Save and Compile** When you select the *Station->Save and Compile* menu or click the  icon in the toolbar, the configuration and parameter assignment data are also saved in system data blocks.

**Consistency Check** You select the *Station -> Consistency Check* menu to check whether it is possible to generate configuration data from the entries made.

**Download in Module** You select the *PLC -> Download* menu or click the  icon in the toolbar to download the selected configuration to the PLC.

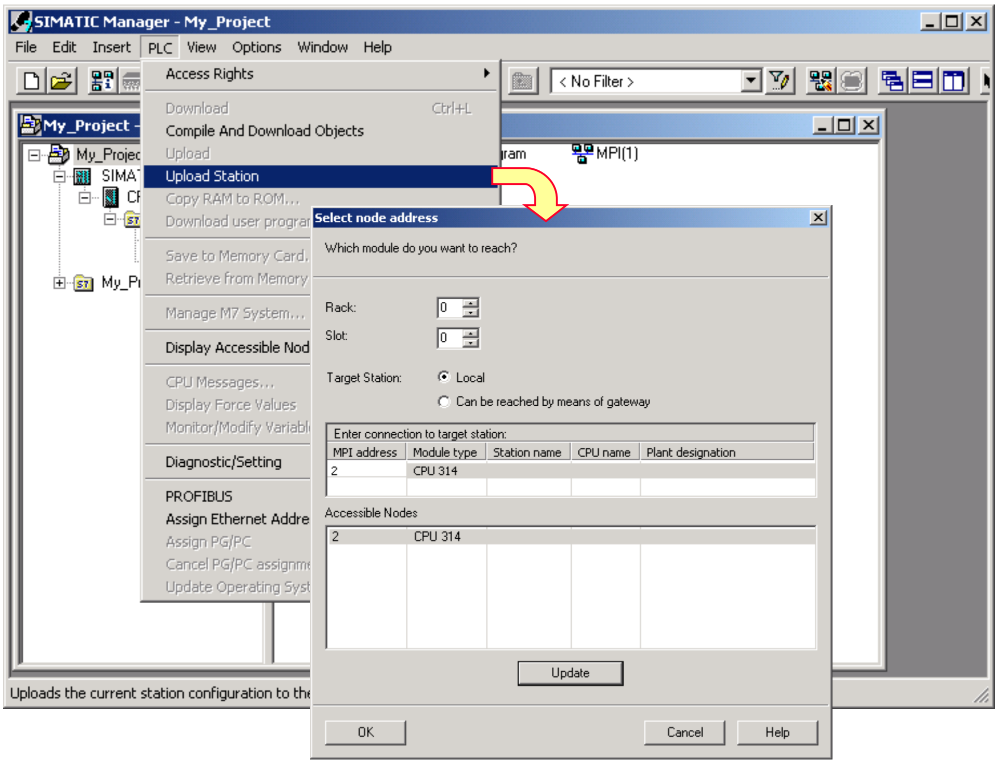
The PLC must be in "STOP" mode!

**System Data Blocks** The system data blocks (SDBs) are generated and modified when you configure the hardware and compile the hardware configuration. SDBs contain configuration data and module parameters. When a system data block is downloaded, it is stored in the CPU's work memory.
This makes it easier to replace modules, because the parameter assignment data is downloaded to the new module from the system data blocks on startup.

In the programming device, the system data blocks are saved under: Project \ Station \ CPU \ S7_program \ Blocks \ System data.
You double-click the System data briefcase  to see the list of system data blocks.

If you use a memory card as Flash EPROM, you should save the SDBs there as well. That way, the configuration is not lost if you operate without battery backup and there is a power failure.

## Uploading the Actual HW Configuration to the PG/PC




SIMATIC® S7


Date: 12.03.03
File: PRO1_04E.14

SITRAIN Training for Automation and Drives

**Introduction**

A configuration is necessary only in the following cases:

- if you want to modify the basic module settings
- for stations with distributed I/O
- for S7-400™ with several CPUs or with expansion racks.

You can read out the actual configuration from the CPU and look at the set parameters in an existing system.

**Actual Configuration**

During startup, the CPU generates an actual configuration. That is, the CPU saves the arrangement of the modules and allocates the addresses in accordance with a fixed algorithm. If no parameters have been assigned, the default parameters defined at the factory are used.

The system stores this actual configuration in system data blocks.

**Uploading to PG/PC**

There are two ways of uploading the actual configuration to the PG/PC:

1. In the SIMATIC® Manager:
   select the *PLC -> Upload Station* menu.
2. In the HW Config tool:
   select the *PLC -> Upload* menu or click the  icon.

**Storage on PG/PC**

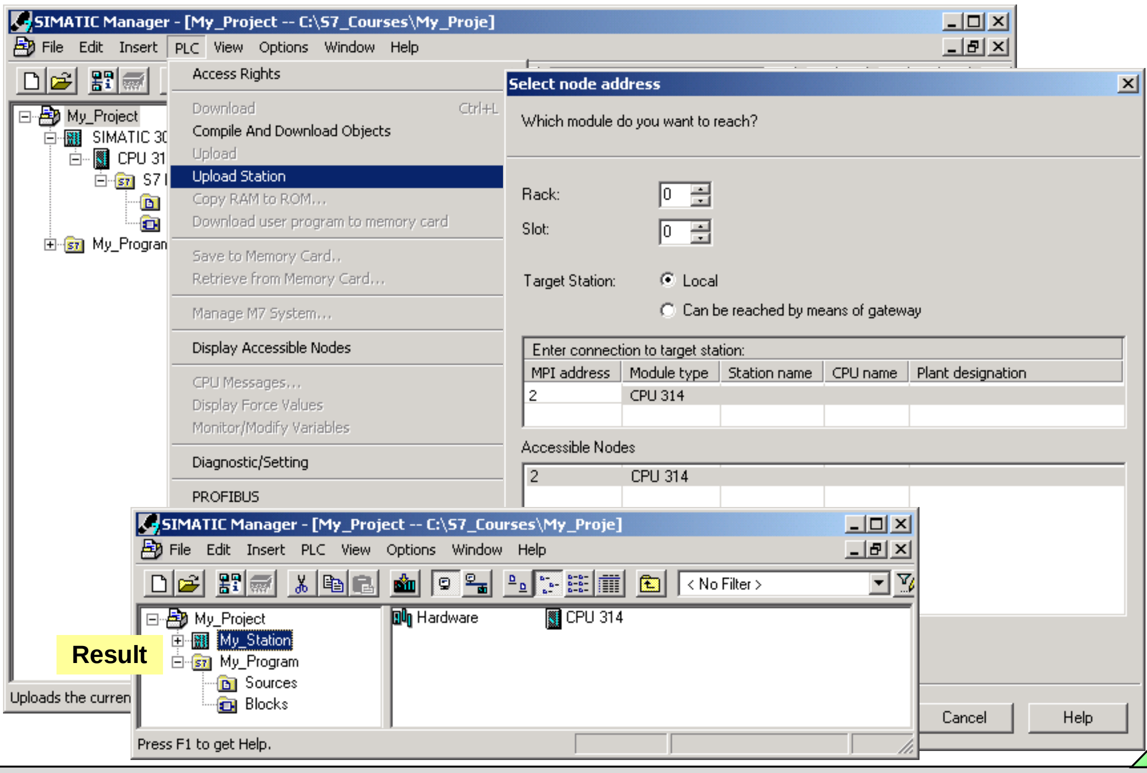
The actual configuration read from the hardware is inserted as a new station in the selected project on the PG/PC.

**Note**

When you read out the actual configuration, the order numbers of the modules cannot be completely identified. For this reason, you should check the configuration. If required, insert the exact module type of the existing modules. To do so, choose the module, and then select the *Options -> Specify Module* menu.

SIEMENS

# Exercise: Upload Actual Configuration to the PG/PC and Rename It








**Task**

To upload a PLC's hardware configuration. Since the project called "My_Project" does not yet have a HW Station, you are to read out the actual PLC configuration from your training area. Rename the newly created hardware station in the project "My_Station".

**What To Do**

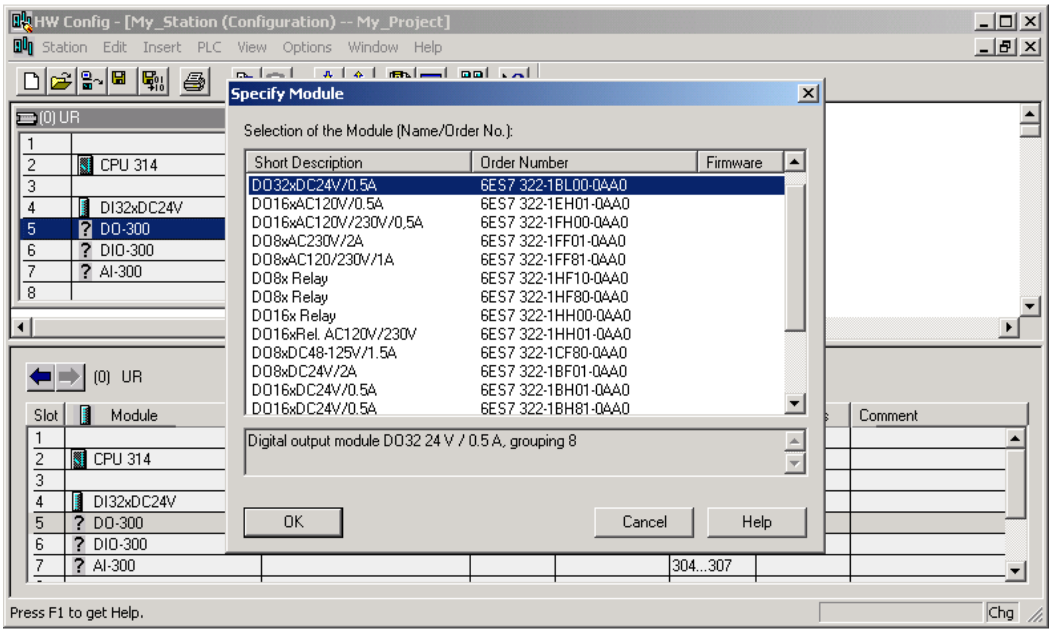
- Start the SIMATIC® Manager and open your project called "My_Project"
- Load the actual configuration from your training area into your project; in *SIMATIC® Manager -> highlight My_Project -> PLC menu -> Upload Station -> OK*

  Complete the follow-up dialog box as shown in the slide above. If no "Accessible Nodes" are visible, you must click "Update".
- Rename the newly created "SIMATIC® 300(1)" hardware station "My_Station"
  *Click twice on "SIMATIC® 300(1)" (not a double-click !) and type "My_Station"*

**Result**

In your project called "My_Project" you now have a hardware station called "My_Station" and the hardware-independent program called "My_Program" (see bottom picture of slide).

# Exercise: Adapting the ACTUAL Configuration




SIMATIC® S7



Date: 12.03.03
File: PRO1_04E.16

**SITRAIN** Training for Automation and Drives

**Task**

The ACTUAL configuration read out with "Upload Station" is incomplete because several module order numbers are missing. These numbers are necessary to clearly identify and assign parameters to the modules. You are to enter the order numbers of the modules of your training area (located on the bottom, outside module cover) in the uploaded "actual configuration".

**What To Do**

1. Start the *HW Config* tool
   *SIMATIC® Manager (Offline view) -> select HW Station called "My_Station" -> double-click "Hardware" icon*
2. Update the modules with correct order numbers
   *double-click each signal module -> in the dialog box "Specify Module", choose the correct part number for the modules on your training area -> confirm the follow-up "Properties" dialog box with OK (since the preset standard parameters do not have to be changed).*
3. Only if your training unit is an S7-400™:
   Specify the module addresses so that they correspond to those of an S7-300™ training unit with 32 channel modules (see slide)
   *double-click on Module -> specify the address in the Properties dialog box*
4. Save and compile the adapted ACTUAL configuration
   *Station -> Save and Compile*
5. Download the adapted ACTUAL configuration to the CPU
   *PLC -> Download*
6. Exit the HW Config tool

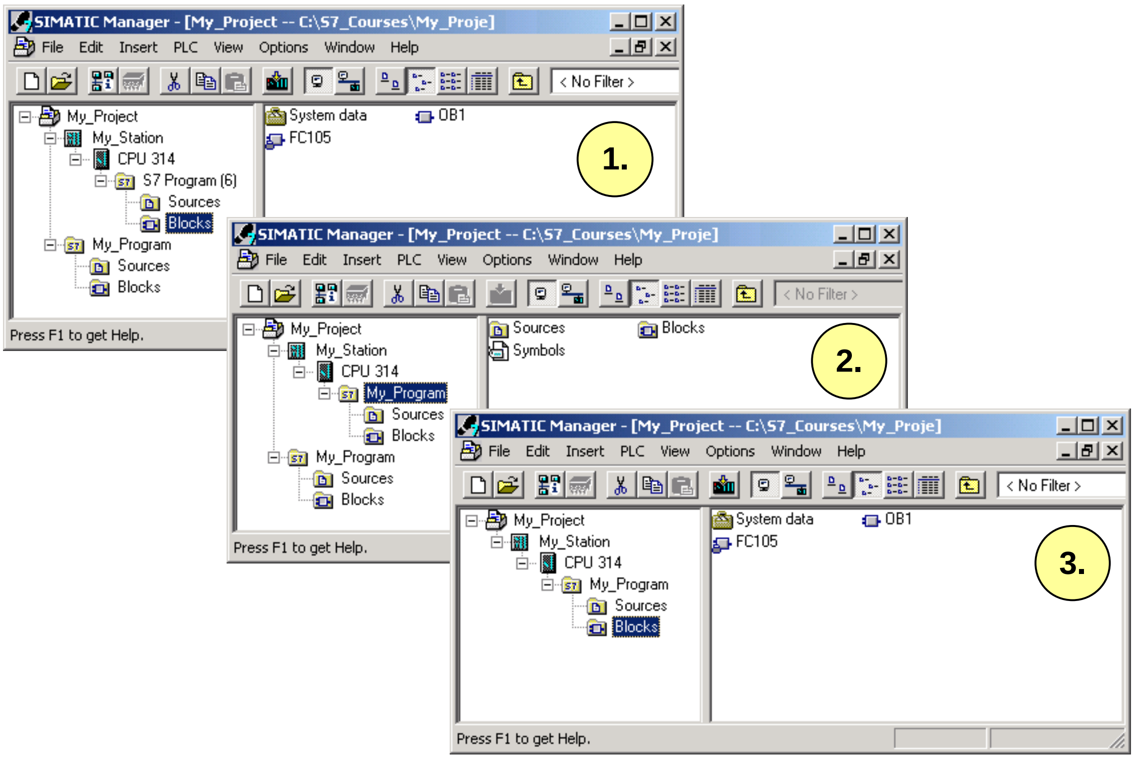
**Result**

The hardware station called "My_Station" in your project called "My_Project" corresponds to the main rack of your training unit.

**Note**

If the training unit has a subnet (Profibus), this portion of the configuration will be completed in the next chapter.

## Exercise: Copy Block from "My_Program"




SIMATIC® S7


Date: 12.03.03
File: PRO1_04E.17

**SITRAIN** Training for Automation and Drives

**Task**

The CPU S7-Program(x) created by the "*Upload Station"* is to be used as the storage location for your user blocks from now on. The blocks created in your hardware-independent program called "My_Program" are therefore to be copied into this new S7 program. You are then to delete your originally created hardware-independent S7 program which you called "My_Program". Then you are to rename the S7 program assigned to the CPU as "My_Program".

**What To Do**

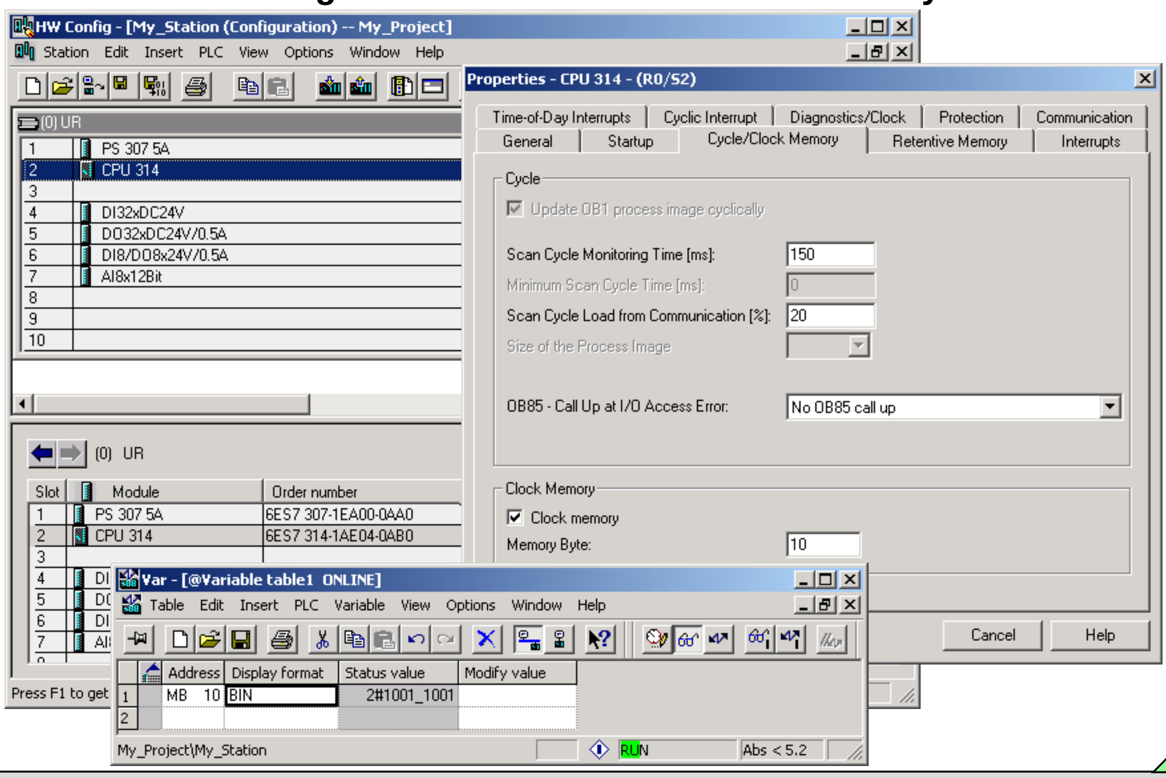
(also see the steps in the slide above)

1. Using drag & drop, copy all blocks from the Blocks folder of the S7 program called "My_Program" into the Blocks folder of the CPU assigned program "S7-Program(x)".
2. Rename the CPU assigned S7-Program(x) as "My_Program".
3. Delete your original hardware-independent program "My_Program".
   Verify that the structure of your project corresponds to picture three above.

**Result**

Your project contains the hardware station called "My_Station" with a CPU whose S7 program is called "My_Program". This project structure corresponds to that of your training area.

# Exercise: Assign Parameters to CPU Clock Memory and Test




SIMATIC® S7


Date: 12.03.03
File: PRO1_04E.18

**SITRAIN** Training for Automation and Drives

**Task**

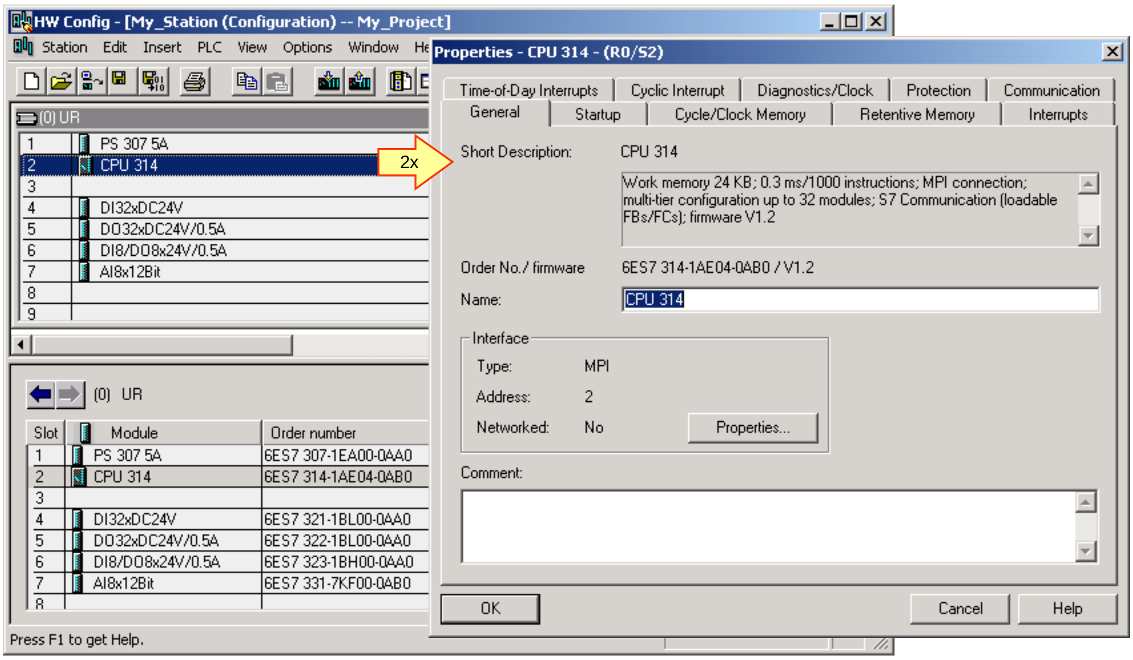
Assign parameters to the CPU clock memory byte. Choose memory byte MB 10.

Then check the success of your parameter assignment with the Monitor/Modify Variable function.

**What To Do:**

- Start the *HW Config* tool
  *SIMATIC® Manager (Offline view) -> select the HW Station called "My_Station" -> double-click the "Hardware" icon*
- In the HW Config editor, open the CPU's *Object Properties* window. Double click the CPU icon. Select the *Cycle / Clock Memory* tab and activate the Clock Memory by selecting it (click on the box).
  Enter 10 in the Memory Byte window and confirm.
- Save and compile the modified configuration
  *Station -> Save and Compile*
- Download the modified configuration into the CPU
  *PLC -> Download*
- Exit the HW Config tool
- Monitor MB10 in the "binary" display format to see the individual flashing frequencies
  *in the SIMATIC® Manager select "My_Program" -> PLC menu -> Monitor/Modify Variable -> enter MB 10 in the variable table address field -> right mouse click on "Display format" -> specify binary -> activate the function using the Monitor variable*  *button.*

# CPU Properties




SIMATIC® S7



Date: 12.03.03
File: PRO1_04E.19

SITRAIN Training for Automation and Drives

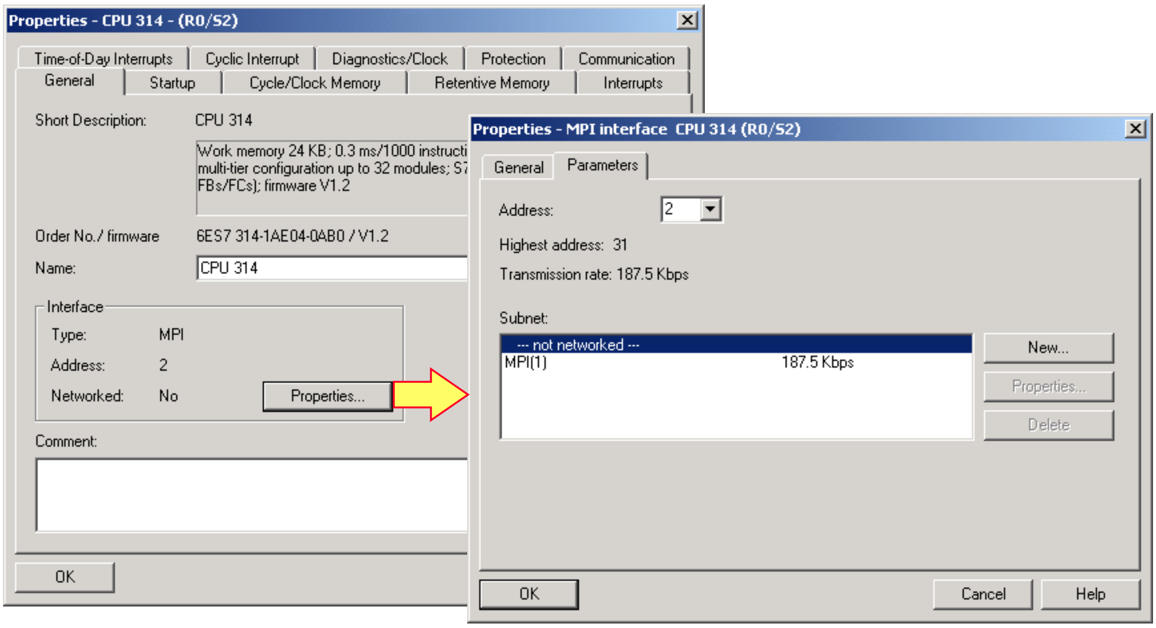
**Assigning Parameters**

You assign parameters to the modules to adapt them to the requirements of the process.

What to do:

1. Select a module in the station window.
2. Double-click the selected module to open the "Properties" dialog window.
3. This dialog window contains nine tabs in which you can assign parameters for the various CPU characteristics (see next pages).

## CPU Properties: General




SIMATIC® S7



Date: 12.03.03
File: PRO1_04E.20

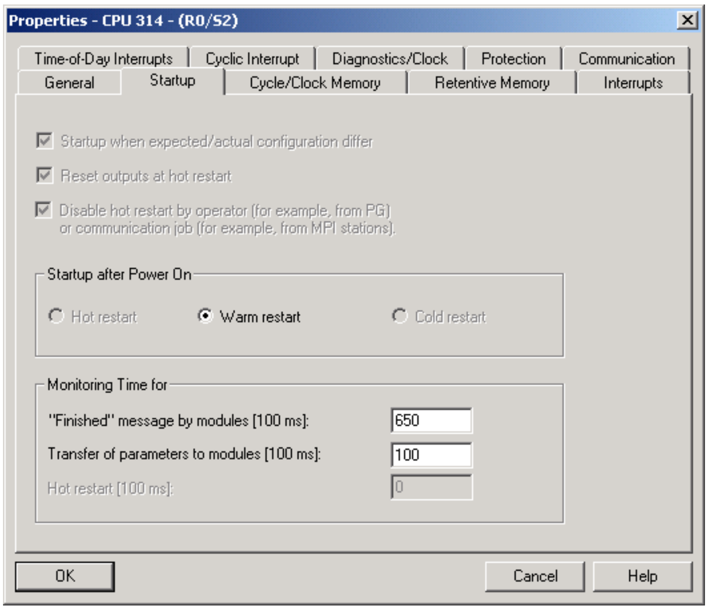
SITRAIN Training for Automation and Drives

**"General" Tab** The "General" tab page provides information about the type of module, its location and, in the case of programmable modules, the MPI address.

**MPI Address** If you want to network several PLCs using the MPI interface, you must assign a different MPI address to each CPU.

Click the "Properties" button to open the "Properties - MPI Node" dialog window, which contains the "General" and "Parameters" tabs.

# CPU Properties: Startup




SIMATIC® S7


Date: 12.03.03
File: PRO1_04E.21

SITRAIN Training for Automation and Drives

**Startup Characteristics**

The S7-300™ and S7-400™ CPUs have different startup characteristics. The special features of the S7-400™ will be dealt with in a later chapter.

**Startup if Setpoint Configuration Not Equal to Actual Configuration**

Only with CPUs with integrated DP interface (and S7-400™) can you use the "Startup when expected/actual configuration differ" checkbox to decide whether the CPU should start up if the setpoint configuration is not the same as the actual configuration (number and type of modules installed).
The other S7300™ CPUs go into RUN when the setpoint configuration is not the same as the actual configuration.

**Warm Restart**

The S7-300™ only recognizes the "Warm restart" startup. New S7-CPUs also recognize "Cold restart". All non-retentive addresses (PII, PIQ, non-retentive bit memories, timers, counters) are reset (overwritten by 0) and the cyclic program execution starts at the beginning.

**Cold Restart (only S7-400™)**

Cold restart behaves the same as Warm restart, except that ALL - even the retentive memory areas - are reset.

**Hot Restart (only S7-400™)**

All - even the non-retentive - memory areas retain their contents and program execution restarts where it stopped.

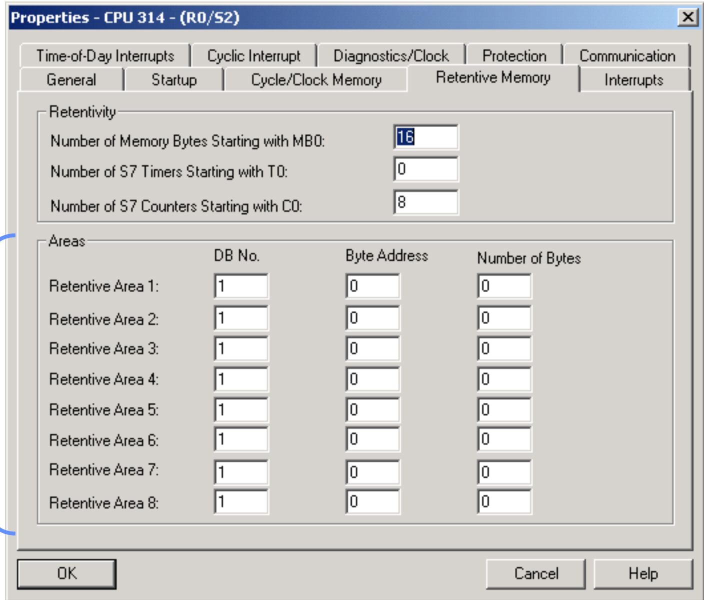
**Monitoring Times**

"Finished message by modules (x100ms):"
Maximum time for all modules to issue a Finished message after power ON. If the modules do not send a Finished message to the CPU within this time, the actual configuration is not equal to the setpoint configuration. For example, in a multi-tier configuration, all power supplies can be switched on within this time without paying attention to a particular sequence.

"Transfer of parameters to modules (x100ms):"
Maximum time for "distributing" the parameters to the parameter-assignable modules (timing begins after "Finished message by modules"). If, after the monitoring time has run out, all modules have not been assigned parameters, then the actual configuration is not equal to the setpoint configuration.

# CPU Properties: Retentive Memory

Relevant only if CPU has no backup battery




SIMATIC® S7


Date: 12.03.03
File: PRO1_04E.22

SITRAIN Training for Automation and Drives

**Retentive Memory**

The "Retentive Memory" tab page is used for specifying the memory areas to be retained after a power failure or during the transition from STOP to RUN.

A "complete restart" is performed in both cases on the S7-300™.

**Warm Restart with Backup Battery**

On warm restart, the blocks stored in the battery-backed RAM (OB, FC, FB, DB) as well as the bit memories, timers and counters defined as retentive are retained. Only the non-retentive bit memories, timers and counters are reset.

**Warm Restart without Backup Battery**

If the RAM is not battery-backed, the information in it is lost. Only the bit memories, timers and counters defined as retentive and the retentive data block areas are saved in the non-volatile RAM area.

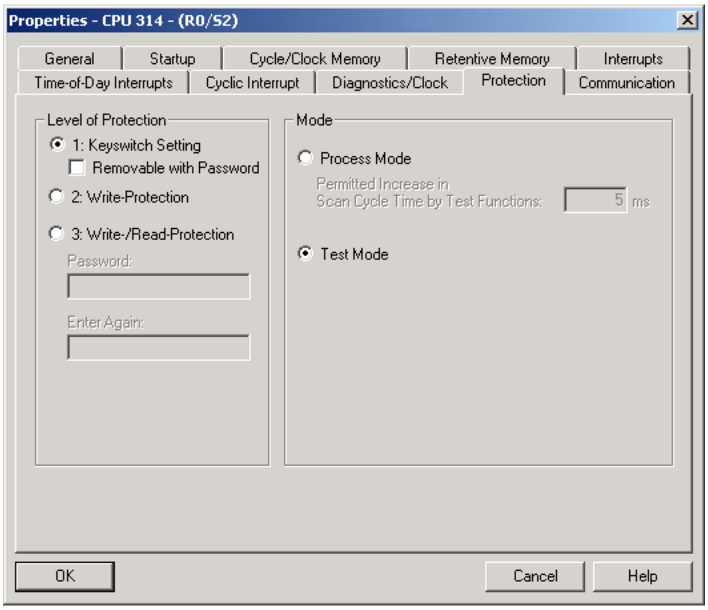
After a warm restart (without battery backup), the program must be downloaded again:

- from the memory card (if inserted) or
- from the PG/PC (if no memory card exists).

**Note**

For CPUs delivered after 10/2002, a backup battery is no longer necessary. All retentive data are saved on the MMC card in case of a power failure.

## CPU Properties: Protection




SIMATIC® S7


Date: 12.03.03
File: PRO1_04E.23

**SITRAIN** Training for Automation and Drives

**Default Setting**

Default setting (protection level 1; no password assigned):
The keyswitch position on the CPU determines the protection level:
- Keyswitch in RUN-P or STOP position: no restrictions
- Keyswitch in RUN position: read-only access possible!

**Password**

If a protection level was assigned with a password (only valid until a memory reset), a "person who knows the password" has reading and writing access.
"The person who doesn‘t know the password" has the following restrictions:
- protection level 1: corresponds to the default setting
- protection level 2: read-only access possible, irregardless of the keyswitch setting
- protection level 3: neither reading nor writing access possible, regardless of the keyswitch setting.

**Characteristics of a Password-Protected Module in Operation**

Example: if you want to execute the "Modify Variable" function, you must enter the password for a module that has been assigned the protection level 2 parameter.

**Access Rights**

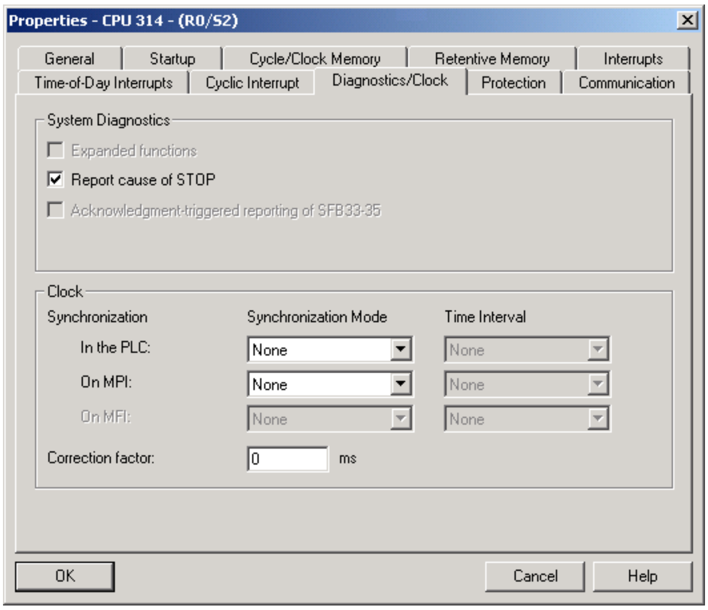
You can also enter the password for a protected module in the SIMATIC® Manager:
1. Select the protected module or its S7 program
2. Enter the password when you select the *PLC -> Access Rights* menu. The access rights, after a password has been entered, is only valid until the last S7 application is completed.

**Mode**

The cycle load for test functions depends on which of the following modes you select.
In Process Mode, test functions such as "Monitor" or "Monitor/Modify Variable" are restricted so that the scan cycle time can not be exceeded. Testing with breakpoints and single-step (program execution) cannot be performed.

In Test Mode, all test functions through the PG/PC can be used without restrictions, even if the scan cycle time is greatly increased.

# CPU Properties: Diagnostics / Clock








**System Diagnostics**

System diagnostics record, evaluate, and report errors in the programmable controller. Examples of errors include an error in te CPU program, a module failure, and wire break for sensors and actuators. If the "Report cause of STOP" checkbox is deactivated (not checked), no message is sent to the PG/PC or OP when the CPU goes into Stop mode ("CPU  Messages").

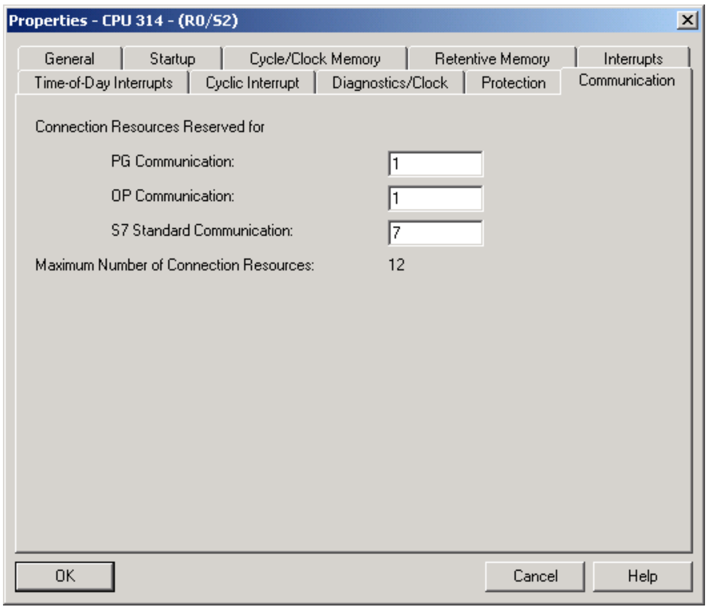
The cause of the stop is still entered in the diagnostic buffer.

**Clock**

For synchronizing the clocks in networked devices. The CPU's clock can be synchronized in the programmable controller (internally), on the MPI (externally), or on the MFI (externally, using a second interface).

- The Synchronization type specifies whether the CPU clock should be used to synchronize the clocks of other CPUs. (The setting options depend on the CPU used.)
- The Time interval selects the time intervals within which the synchronization is to be carried out.
- The **correction factor** compensates for a loss or gain in the clock time within a 24 hour period.  Positive or negative millisecond values can be specified.
  **Example:** If the clock is 3 seconds fast after 24 hours, this inaccuracy can be corrected with the "-3000ms" factor.

## CPU Properties: Communication




SIMATIC® S7


Date: 12.03.03
File: PRO1_04E.25

SITRAIN Training for Automation and Drives

**Communication**

The communications tab allocates the CPU’s connection resources for data exchange over the subnets (MPI, PROFIBUS, etc). S7 functions (mainly integrated in the CPU’s operating system) control communication between CPUs, HMIs, and programming devices.

Every communication connection occupies a connection resource on the S7-CPU. Depending on the technical specifications, a specific number of possible connections are available to every S7-CPU.

When communication services log on, the connection resources are occupied in the sequence of the log on.

So that the occupation of these resources is not dependent only on the sequence of the log on of the various communication services, you can also reserve communication resources for the following services:

- PG/PC Communication and OP Communication
- S7 Standard Communication

At least one connection resource each is reserved for the PG/PC and OP communication. Smaller values are not possible.

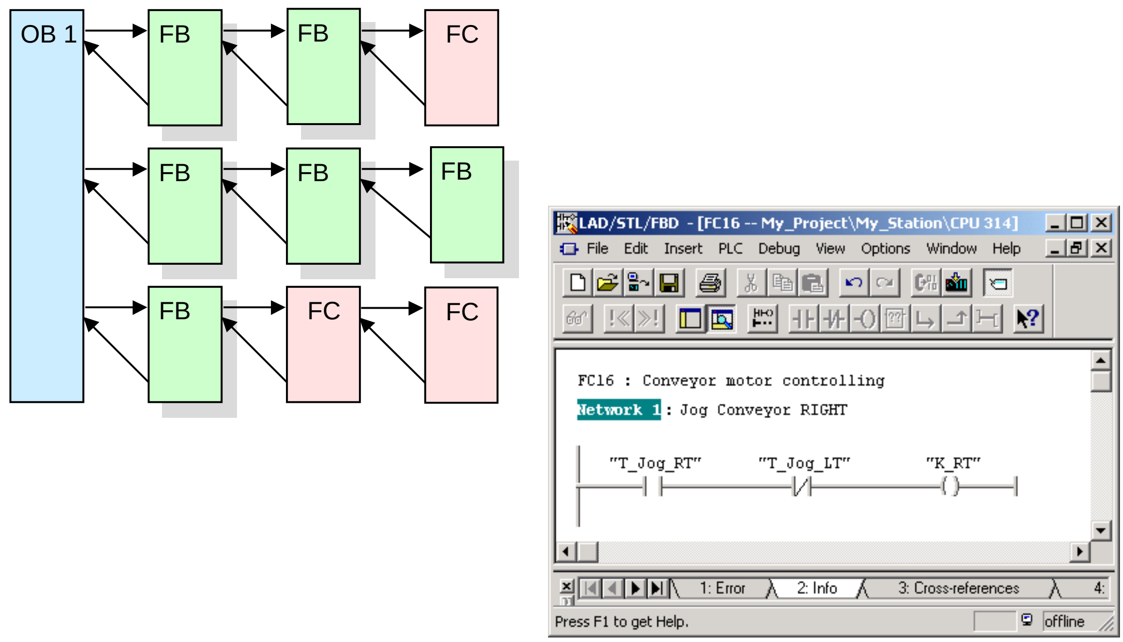
Other communication services, such as S7 Communication with PUT/GET functions, cannot occupy these connection resources even if the services make their connection first. Instead, still available connection resources are occupied that were not specifically reserved for a service.

**Note**

The "Interrupts", "Time-Of-Day Interrupts" and "Cyclic Interrupt" tabs are discussed in the "Organization Blocks" chapter.

SIEMENS

## Block Architecture and Block Editor




SIMATIC® S7


Date: 12.03.03
File: PRO1_05E.1

SITRAIN Training for Automation and Drives

## Contents

Page

# Objectives

**Upon completion of this chapter the participant will ...**

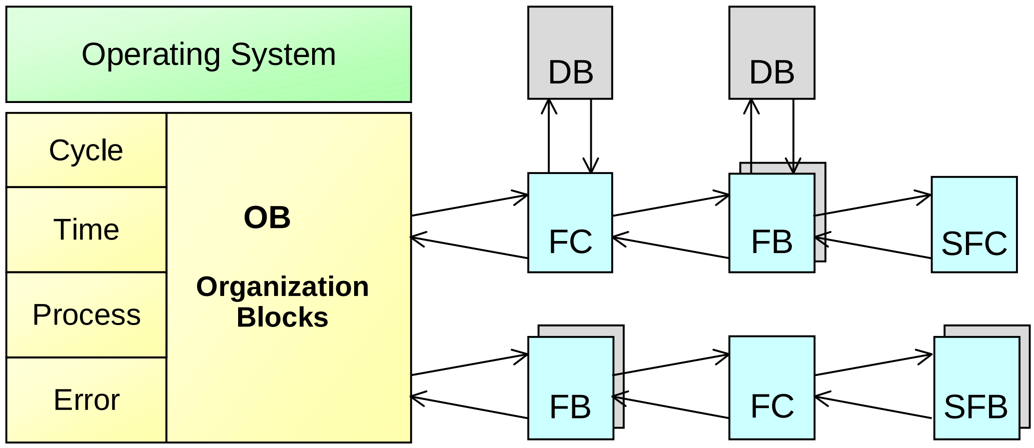
- ... know the different types of S7 blocks
- ... understand the principle of "structured programming"
- ... know the meaning of the process images (PII, PIQ)
- ... be able to explain the principle of cyclic program execution
- ... know and be able to select the STEP7 programming languages - LAD, FBD and STL
- ... be able to edit, save and download an S7 logic block into the CPU using the LAD/STL/FBD Editor
- ... be able to carry out a simple program debugging with the "Monitor Block" test function
- ... will be able to make customizations to the LAD/STL/FBD Editor

SIMATIC® S7


Date: 12.03.03
File: PRO1_05E.2

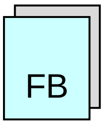
**SITRAIN** Training for Automation and Drives

## 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

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

SIMATIC® S7


Date: 12.03.03
File: PRO1_05E.3

SITRAIN Training for Automation and Drives

**Blocks**

The programmable logic controller provides various types of blocks in which the user program and the related data can be stored. Depending on the requirements of the process, the program can be structured in different blocks.

**Organization Block OB**

Organization blocks (OBs) form the interface between the operating system and the user program. The entire program can be stored in OB1 that is cyclically called by the operating system (linear program) or the program can be divided and stored in several blocks (structured program).

**Function FC, SFC**

A function (FC) contains a partial functionality of the program. It is possible to program functions so that they can be assigned parameters. As a result, functions are also suited for programming recurring, complex partial functionalities such as calculations.

System functions (SFC) are parameter-assignable functions integrated in the CPU's operating system. Both their number and their functionality are fixed. More information can be found in the Online Help.

**Function Block FB, SFB**

Basically, function blocks offer the same possibilites as functions. In addition, function blocks have their own memory area in the form of instance data blocks. As a result, function blocks are suited for programming frequently recurring, complex functionalities such as closed-loop control tasks.

System function blocks (SFB) are parameter-assignable functions integrated in the CPU's operating system. Both their number and their functionality are fixed. More information can be found in the Online Help.

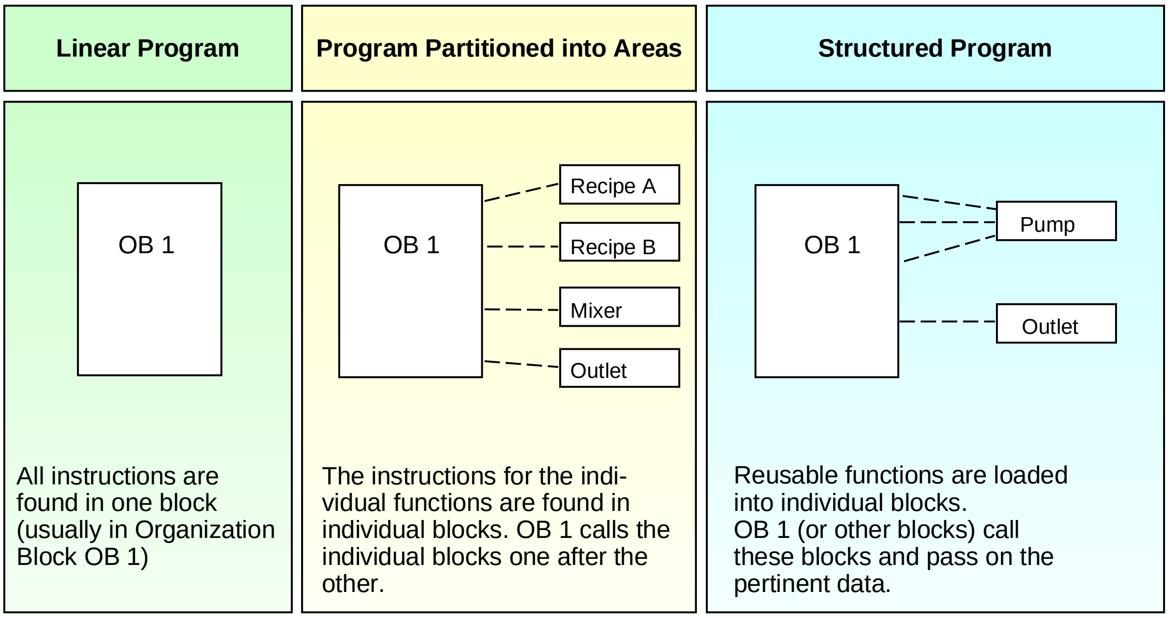
**Data Blocks DB**

Data blocks (DB) are data areas of the user program in which user data are managed in a structured manner.

**Permissible Operations**

You can use the entire operation set in all blocks (FB, FC and OB).

## Program Structure




SIMATIC® S7

Date: 12.03.03
File: PRO1_05E.4
SITRAIN Training for Automation and Drives

**Linear Program**

The entire program is found in one continuous program block.

This model resembles a hard-wired relay control, that was replaced by a programmable logic controller. The CPU processes the individual instructions one after the other.

**Partitioned Program**

The program is divided into blocks, whereby every block only contains the program for solving a partial task. Further partitioning through networks is possible within a block. You can generate network templates for networks of the same type.

The OB 1 organization block contains instructions that call the other blocks in a defined sequence.

**Structured Program**

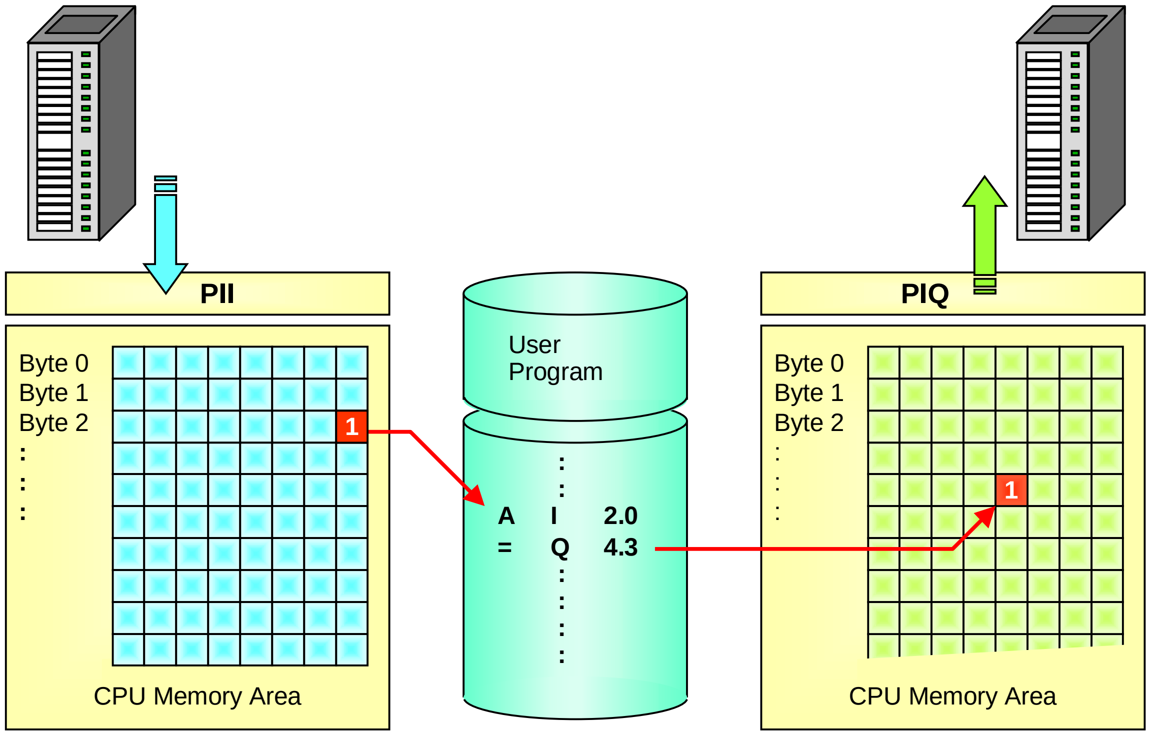
A structured program is divided into blocks. The code in OB1 is kept to a minimum with calls to other blocks containing code. The blocks are parameter assignable. These blocks can be written to pass parameters so they can be used universally.

When a parameter assignable block is called, the programming editor lists the local variable names of the blocks. Parameter values are assigned in the calling block and passed to the function or function block.

Example:

- A "pump block" contains instructions for the control of a pump.
- The program blocks, which are responsible for the control of special pumps, call the "pump block" and give it information about which pump is to be controlled with which parameters.
- When the "pump block" has completed the execution of its instructions, the program returns to the calling block (such as OB 1), which continues processing the calling block's instructions.

## Process Images




SIMATIC® S7


Date: 12.03.03
File: PRO1_05E.5

SITRAIN Training for Automation and Drives

**Introduction** The CPU checks the status of the inputs and outputs in every cycle. There are specific memory areas in which the module's binary data are stored: PII and PIQ. The program accesses these registers during processing.

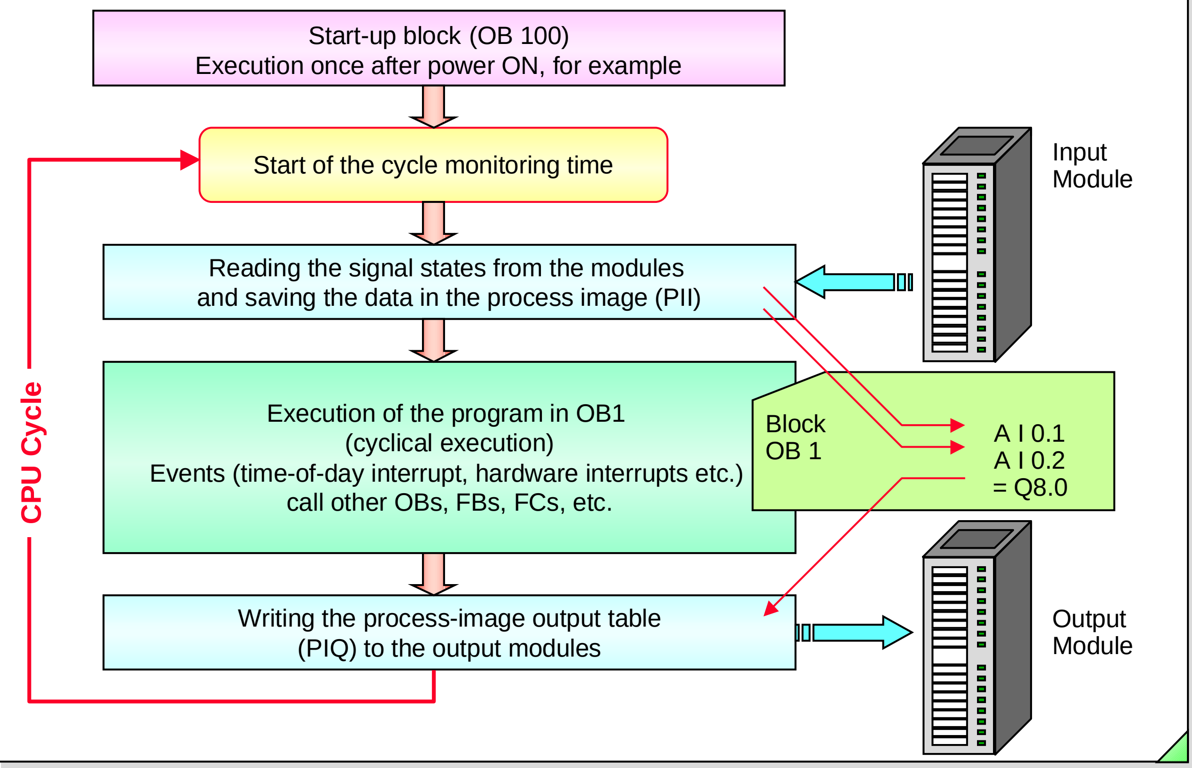
**PII** The **P**rocess-**I**mage **I**nput table is found in the CPU's memory area. The signal state of all inputs is stored there.

**PIQ** The **P**rocess-**I**mage Output (**Q**) table contains the output values that result from the program execution. These output values are sent to the actual outputs (Q) at the end of the cycle.

**User Program** When you check inputs in the user program with, for example, A **I** 2.0, the last state from the PII is evaluated. This guarantees that the same signal state is always delivered throughout one cycle.

**Note** Outputs can be assigned as well as checked in the program. Even if an output is assigned a state in several locations in the program, only the state that was assigned last is transferred to the appropriate output module.

## Cyclic Program Execution




SIMATIC® S7


Date: 12.03.03
File: PRO1_05E.6

SITRAIN Training for Automation and Drives

**Starting**

The CPU carries out a complete restart (with OB100) when switching on or when switching from STOP --> RUN. During a complete restart, the operating system:

- deletes the non-retentive bit memories, timers and counters.
- deletes the interrupt stack and block stack.
- resets all stored hardware interrupts and diagnostic interrupts.
- starts the scan cycle monitoring time.

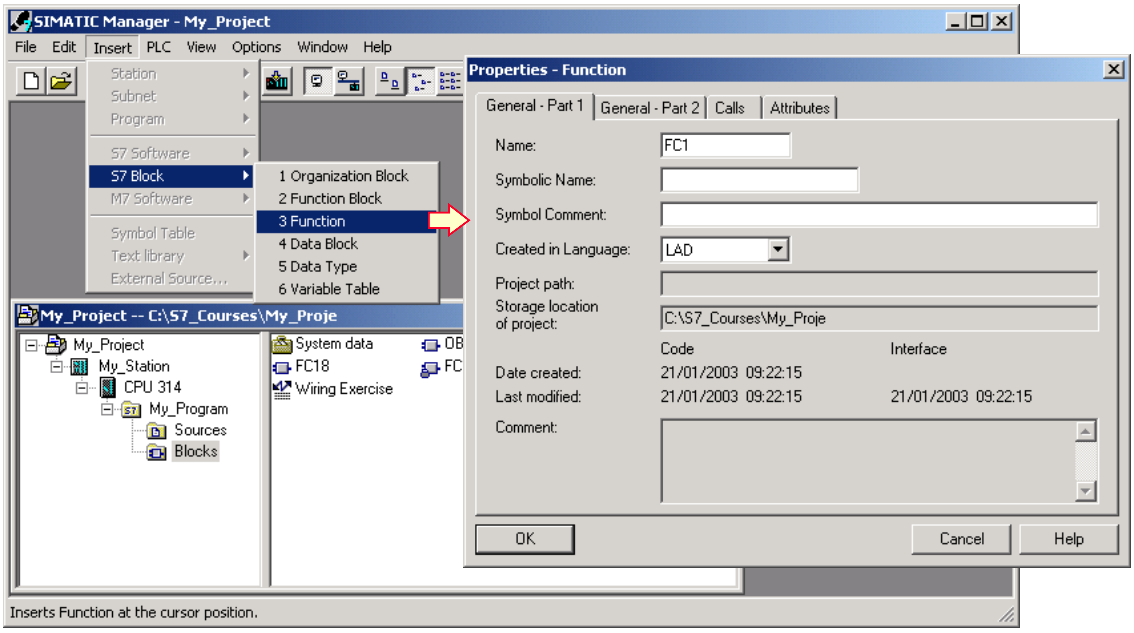
**Scan Cycle**

The cyclical operation of the CPU consists of three main sections, as shown in the diagram above. The CPU:

- checks the status of the input signals and updates the process-image input table.
- executes the user program with the respective instructions.
- writes the values from the process-image output table into the output modules.

SIEMENS

## Inserting an S7 Block




SIMATIC® S7



Date: 12.03.03
File: PRO1_05E.7

SITRAIN Training for Automation and Drives

**Inserting a Block**

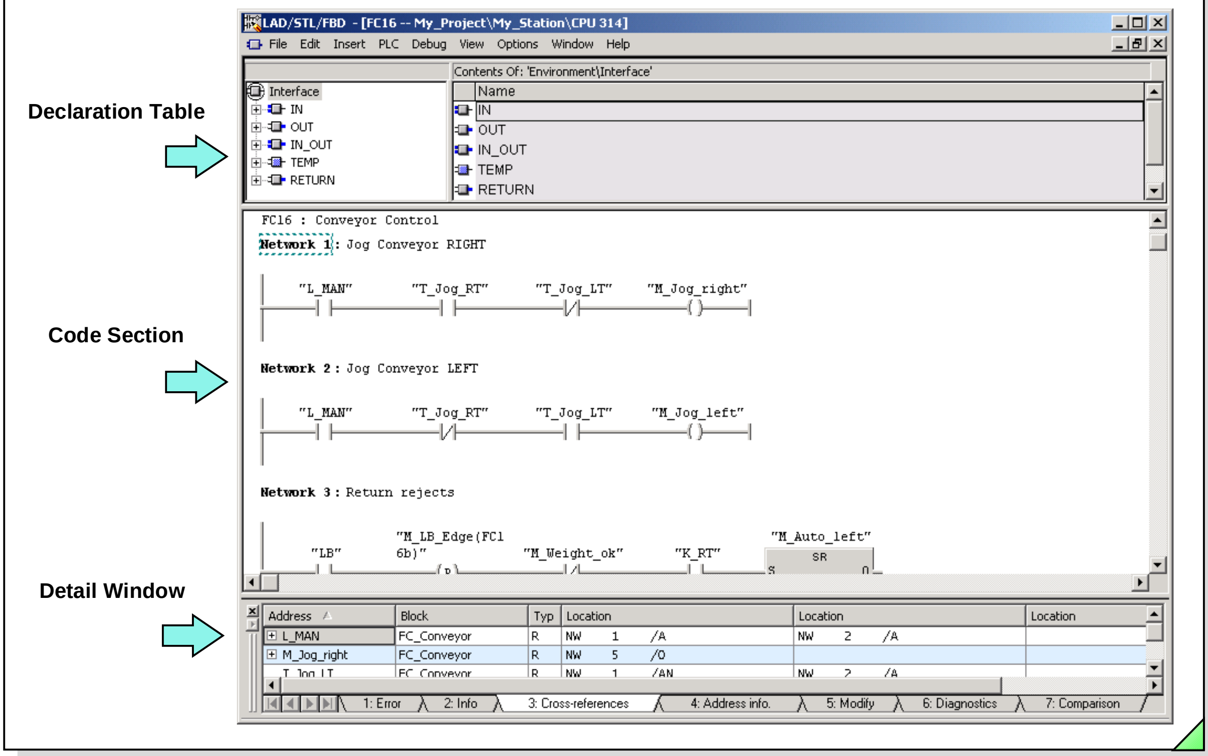
With the appropriate "Blocks" folder highlighted, from a specific "S7 Program", select the *Insert -> S7 Block* menu option to display a list of block types:

- Organization blocks (OB) are called by the operating system. These blocks form the interface between operating system and user program.
- Functions (FC) and function blocks (FB) contain the actual user program. They enable a complex program to be divided into small, easy-to-follow units.
- Data blocks contain user data.

After choosing the type of block you want, the "Properties" dialog box opens so that you can enter the block number and choose a programming language (LAD, STL or FBD).
There are other settings you can make, depending on the type of block, but these will be discussed later.

When you have made your settings and confirmed them by clicking the "OK" button, the new block is inserted in the current program.

## The LAD/STL/FBD Editor




SIMATIC® S7


Date: 12.03.03
File: PRO1_05E.8

SITRAIN Training for Automation and Drives

**Starting the Editor** The easiest way to start the LAD/STL/FBD Editor is by double-clicking on an S7 block in the SIMATIC® Manager.

The Editor has the following components:

**Declaration Table** The declaration table belongs to the block. This table is used for declaring variables and parameters for the block.
The declaration table is discussed in detail in the "Functions and Function Blocks" chapter.

**Code Section** The code section contains the program itself, divided into separate networks if required.
A syntax check is made during instruction input (in STL) and in labelling program elements or operation symbols.

**Detail Window** The detail window provides the following functions and information:

*1: Error:*
lists the syntax errors found in the course of a context check or a compilation procedure

*2: Info:*
gives additional information such as "expected data type of an address"

*3: Cross references*
a list of addresses used in the network and where they are used in the entire program

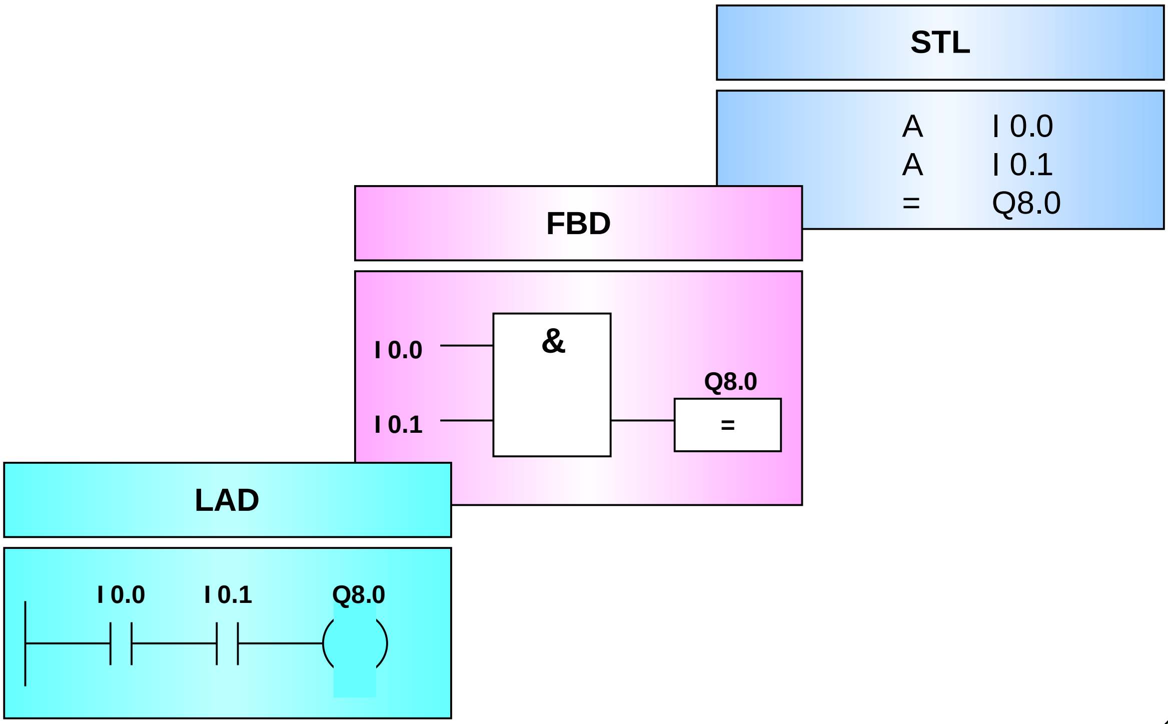
*4: Address info*
enables you to monitor the addresses used in the network

*5: Modify*
enables you to modify the addresses used in the network

*6: Diagnostics*
display of existing data for process diagnostics (only if configured)

*7: Comparison*
Navigation with the function "Compare blocks"

## The STEP7 Programming Languages




SIMATIC® S7



Date: 12.03.03
File: PRO1_05E.9

SITRAIN Training for Automation and Drives

**Introduction**

There are several programming languages in STEP 7 that can be used depending on preference and knowledge. By adhering to specific rules, the program can be created in Statement List and later converted into another programming language.

**LAD**

Ladder Diagram is very similar to a circuit diagram. Symbols such as contacts and coils are used. This programming language often appeals to those who have a drafting or electrical background.

**STL**

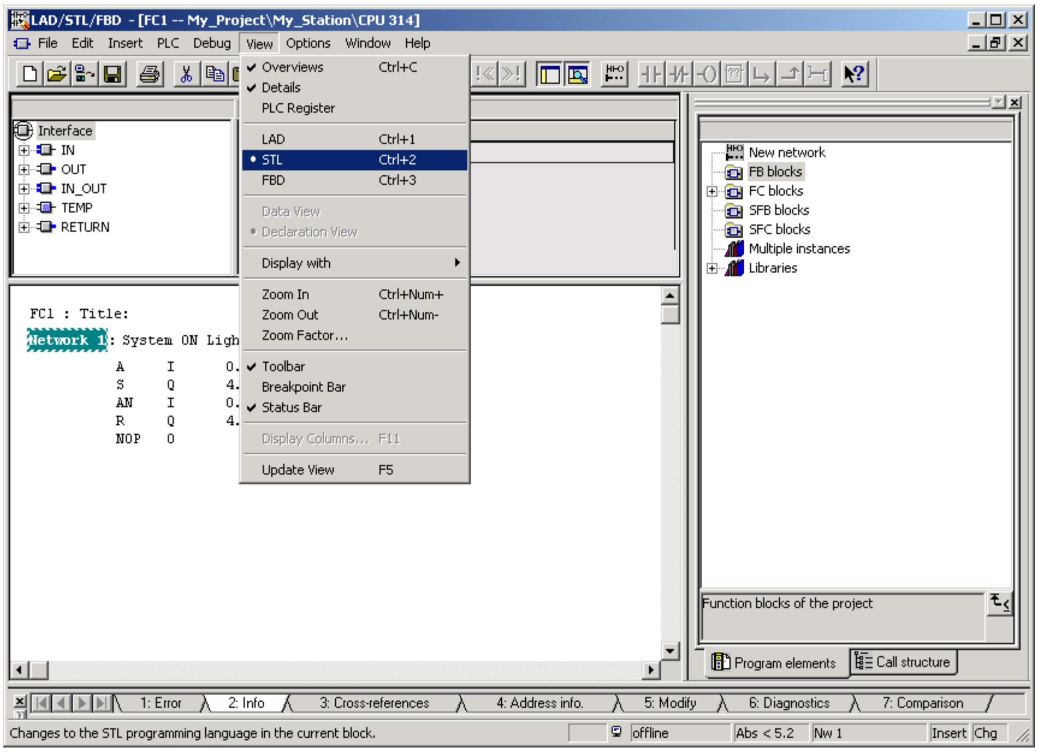
The Statement List consists of STEP 7 instructions. You can program fairly freely with STL. This programming language is preferred by programmers who are already familiar with other programming languages.

**FBD**

The Function Block Diagram uses “boxes” for the individual functions. The character in the box indicates the function (such as & --> AND Logic Operation). This programming language has the advantage that even a “non-programmer” can work with it. Function Block Diagram is available as of Version 3.0 of the STEP7 Software.

SIEMENS

## Selecting the Programming Language




SIMATIC® S7


Date: 12.03.03
File: PRO1_05E.10

**SITRAIN** Training for Automation and Drives

**View** You choose the *View* menu to switch from one STEP 7 programming language to another:

- LAD (Ladder Diagram)
- FBD (Function Block Diagram)
- STL (Statement List).

**Switching the Program Language** You can switch the programming language as you wish when you create as well as later on.

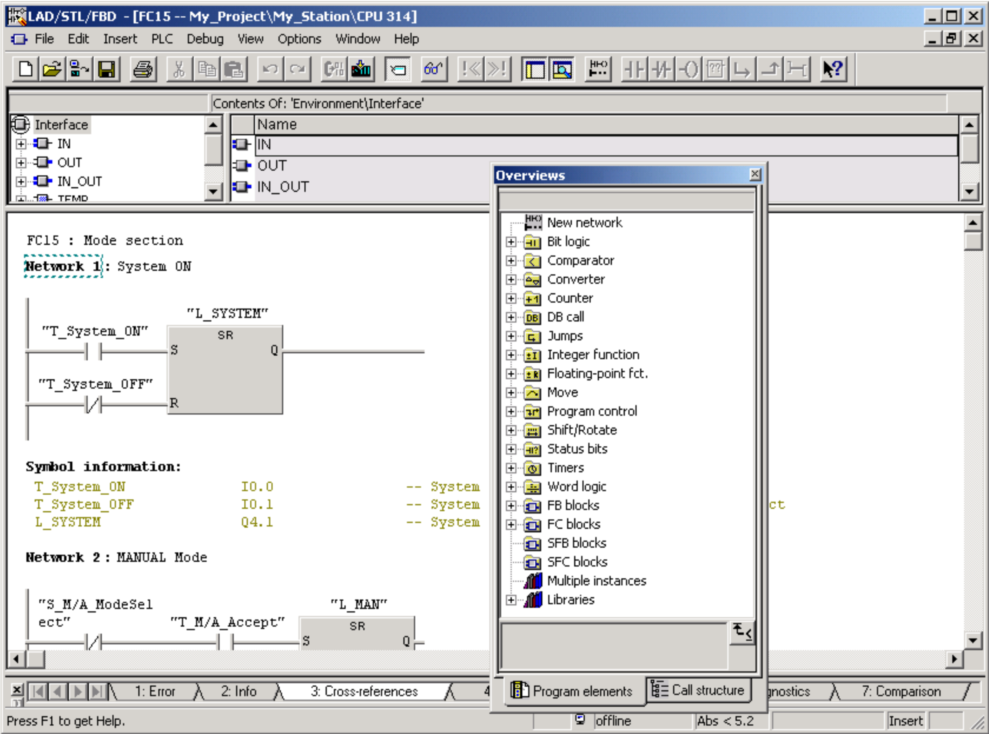
**LAD/FBD => STL** You can convert program sections that have been written in the graphical programming languages (LAD/FBD) into STL. You should, however, be aware that the result of this conversion is not always the most efficient solution in Statement List.

**STL => LAD/FBD** It is not always possible to convert program sections written in STL into LAD or FBD. The sections of the program that cannot be converted are left in STL.

No sections of the program are lost on conversion.

**SIEMENS**

# Programming in LAD/FBD



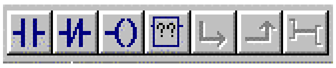

SIMATIC® S7


Date: 12.03.03
File: PRO1_05E.11

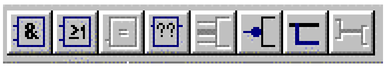
**SITRAIN** Training for Automation and Drives

**Elements** Frequently used LAD and FBD elements appear as icons in the toolbar. You click them with the mouse to insert them at the selected position in the program.

Toolbar icons in FBD:



Toolbar icons in LAD:



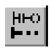
**Overviews** By clicking the "Overviews" symbol, a new window is opened with the following contents:

**Program Elements**:

with all program elements and operation symbols.
(The contents of this window depends on the programming language - LAD/STL/FBD selected)

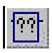
**Call Structure**:

shows the program structure and/or the block nesting, which block is called from where.

**Networks** When you click the "New Network" icon  in the toolbar, a new network is added after the current network. You can also right mouse click and choose "insert network".

**Note** If you want to insert a new network before Network 1, you must select the block name ("FC1: Operating Mode Section" in the example above) before you click the "New Network" icon.

**Empty Box**

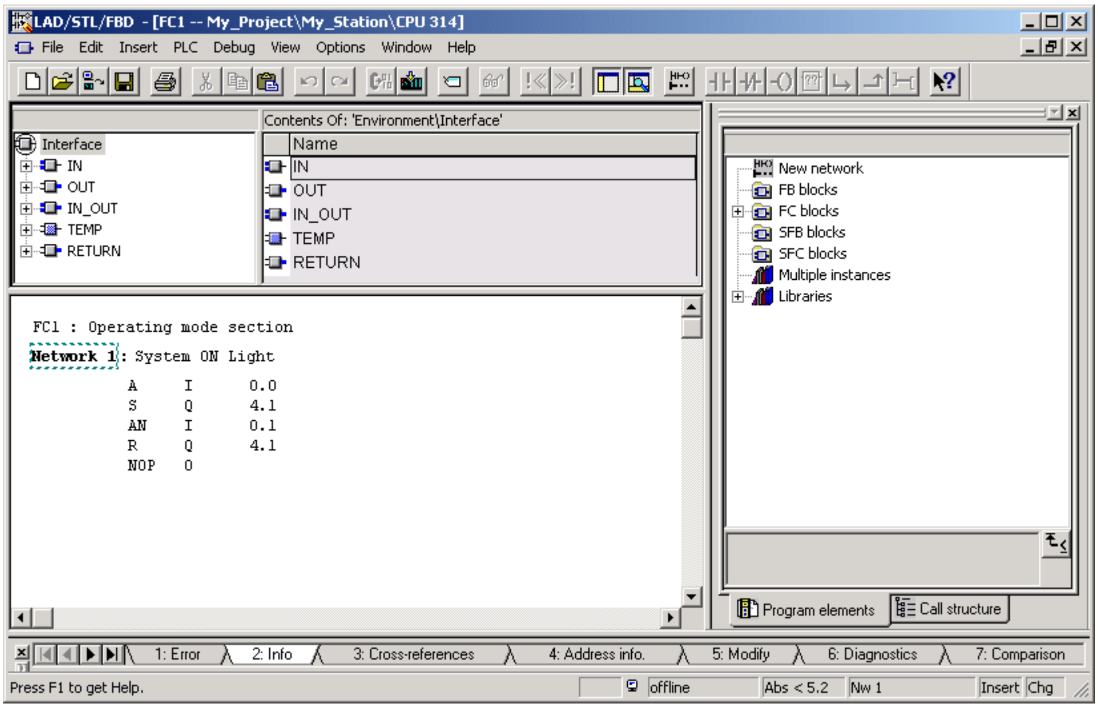


You can use the Empty Box to insert LAD or FBD elements more quickly. You can insert elements directly without having to select them from the Program Elements browser. After you have selected the position in the network where you want to insert an element, click the "Empty Box" icon in the toolbar. When you enter the first letters of an element name, a list appears (beginning with these letters), and you can make a selection.

**Insert / Overwrite** You use the "Insert" key to toggle between the "Cp" (overwrite) and "Insert" modes. The current setting appears in the status bar.

SIEMENS

## Programming in STL




SIMATIC® S7


Date: 12.03.03
File: PRO1_05E.12

SITRAIN Training for Automation and Drives

**Statements**

The user needs to know the statements for writing a program in STL. You can obtain information about the syntax and functionality from the online help: *Help -> Help on STL.*

The following information is available:

- "Statement List Instructions", a description of all the statements that exist in this programming language
- "Working with Statement List", a description of
  - Statement List View and General Syntax
  - Entering and Viewing Constant Data
  - Types of Blocks
  - Switch Contacts and Signal States

**Overviews**

When you are using the STL Editor, the "Overviews" window contains only a list of the existing blocks which can be called from the current block and the libraries.

**Networks**

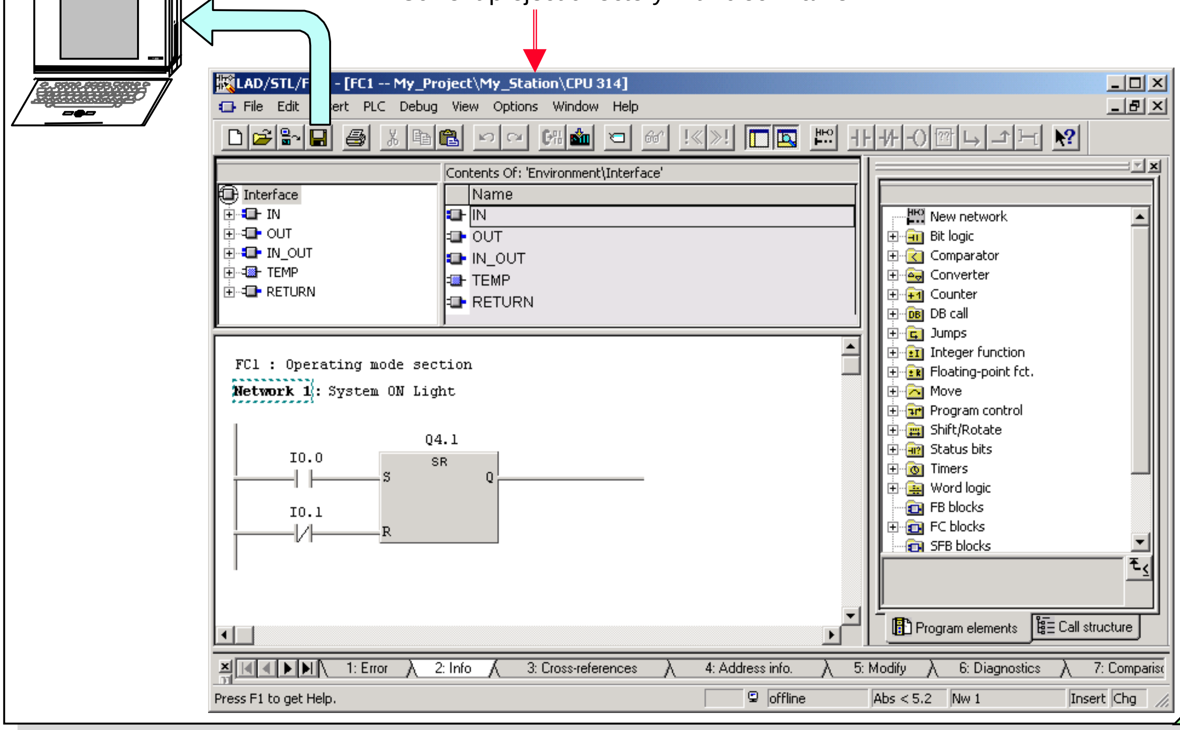
Networks are inserted in the same way as in the LAD/FBD Editor (see previous page).

**Insert/Overwrite**

You use the "Insert" key to toggle between the "Cp" (overwrite) and "Insert" modes. The current setting appears in the status bar.

## Saving a Block

Current project directory with block name




SIMATIC® S7


Date: 12.03.03
File: PRO1_05E.13

SITRAIN Training for Automation and Drives

**Saving a Block**

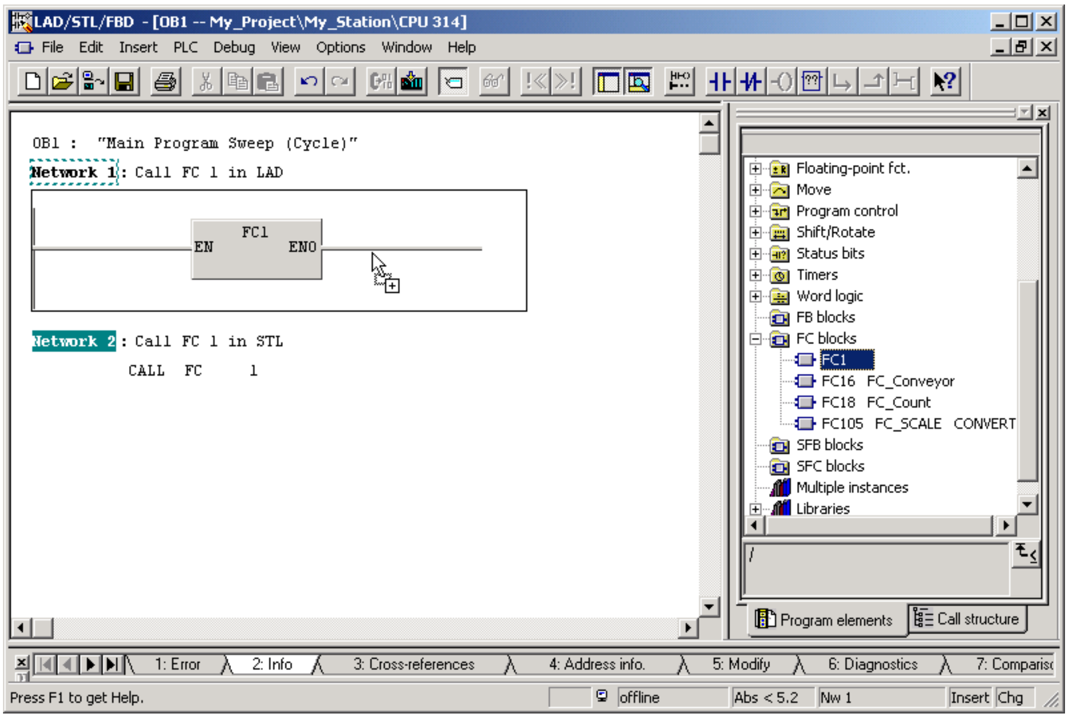
When you have finished editing a block, you can save it on the hard disk of the programming device:

- by selecting the *File -> Save* menu option or
- by clicking the "Save" icon in the toolbar.

Note

If more than one block is opened with the Editor, only the block that is visible in the active window is saved with the action "Save" .

## Calling a Block in OB1




SIMATIC® S7

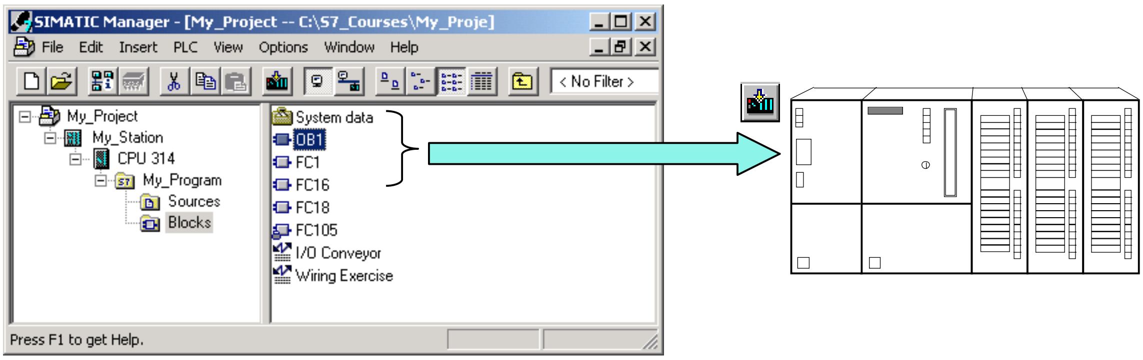

Date: 12.03.03
File: PRO1_05E.14

**SITRAIN** Training for Automation and Drives

**Cyclic Execution**

To integrate a newly created block in the cyclic program execution of the CPU, the block must be called in OB1.
The simplest way of inserting the block call in the graphic programming languages LAD and FBD is through the browser (see picture above). In the STL programming language, the instruction for calling a block is CALL.

# Downloading Blocks into the PLC




SIMATIC® S7


Date: 12.03.03
File: PRO1_05E.15

**SITRAIN** Training for Automation and Drives

**Downloading**

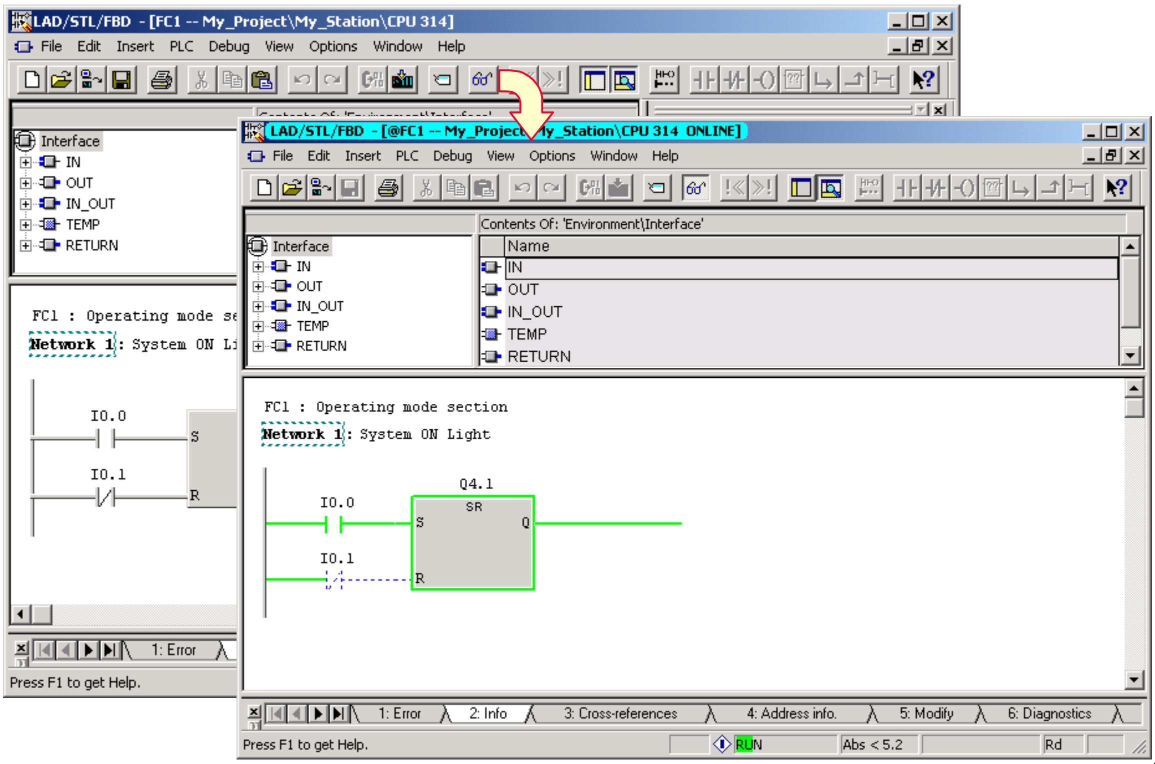
From the SIMATIC® Manager, download the blocks into the PLC by:

- clicking the  icon or
- selecting the *PLC -> Download* menu option.

Before you do this, you must select the blocks you want to download:

- All blocks: Select the "Blocks" folder in the left pane of the project window.
- Several blocks: Hold down the CTRL key and select the blocks you want.
- One block: Select the block.

## Simple Program Debugging






**Requirements**

Before you can activate the monitoring mode, you must open the block you want to monitor either **offline or online** with the LAD/STL/FBD Editor.

Note: In order to test a block in the offline mode, the block must first be downloaded into the PLC.

**Activation / Deactivation**

There are two ways of activating/deactivating the "Monitor" test function:

- click the "glasses" icon 
- select the *Debug -> Monitor* menu option.

**View**

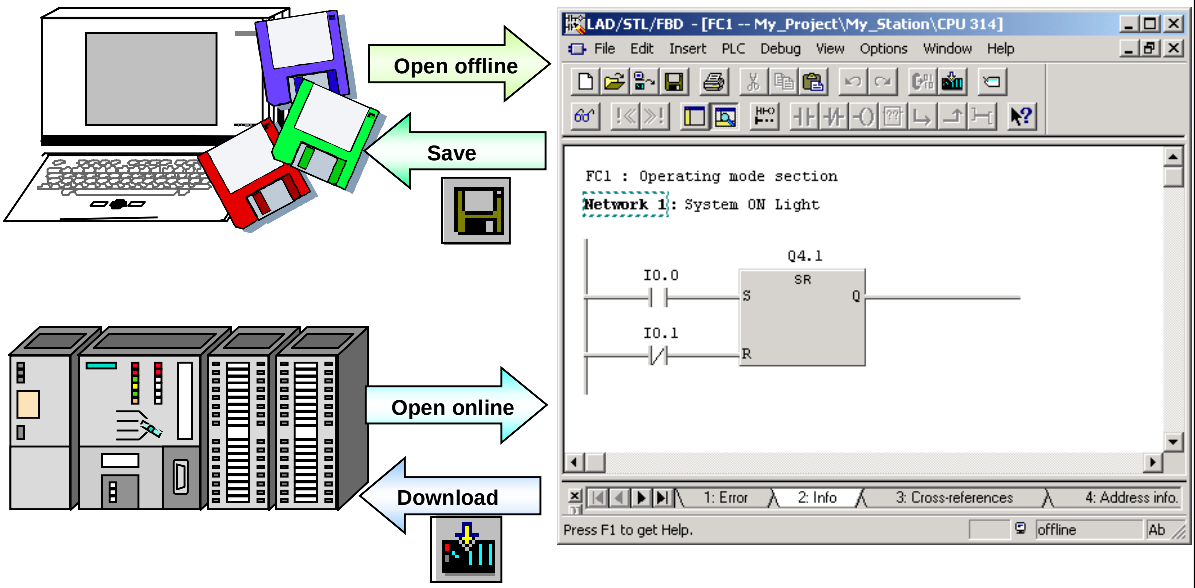
The program status is displayed in different ways depending on the programming language selected (LAD/STL/FBD).

When the monitoring function is activated, you cannot change the programming language in which the block is viewed (LAD/FBD/STL).

**Note**

You will find more information about testing programs in the "Troubleshooting" chapter.

## Downloading and Saving Modified Blocks




SIMATIC® S7


Date: 12.03.03
File: PRO1_05E.17

SITRAIN Training for Automation and Drives

**Making Corrections to Blocks**

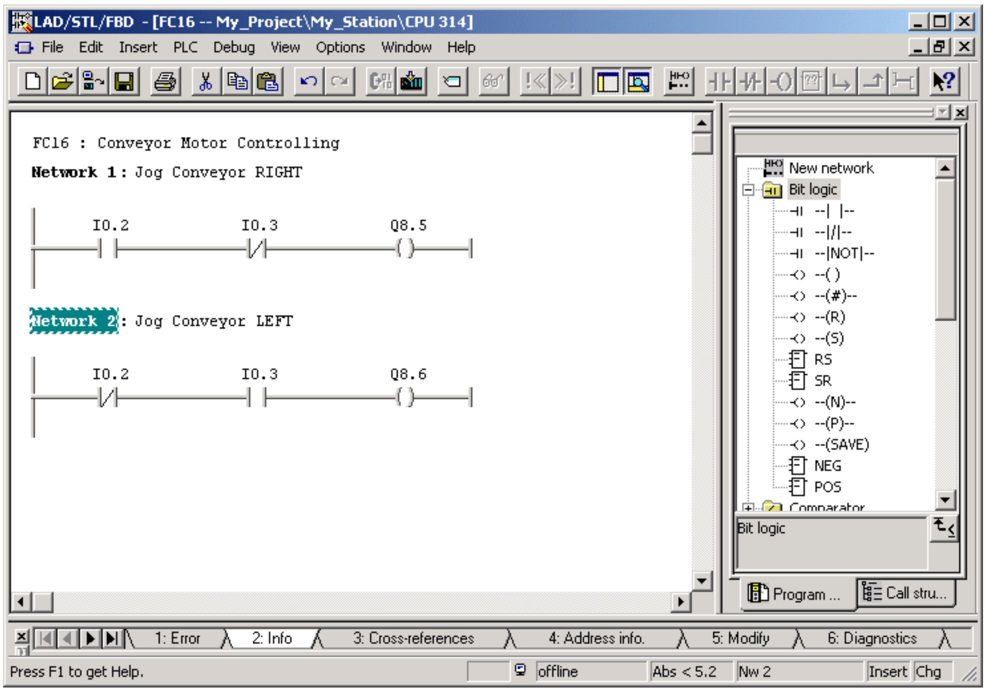
You can make corrections to blocks that have been opened either *online* or *offline*; however, not in the test mode.

- You normally download the modified block to the PLC, test the block, make further corrections if necessary and finally save it on the hard disk when it has been fully debugged.
- If you do not want to test the program immediately, you can first save the changes on the hard disk. The old version of the block is then erased.
- If you make corrections to a number of blocks and don't want to overwrite the original version of the program yet, you can download the changed blocks to the CPU first without saving them on the hard disk of the PG/PC. You can save them on the hard disk of the PG/PC when you have tested the whole program successfully.

**Insert/Cp (Overwrite)**

The insert mode is set by default for LAD or FBD. By pressing the "Insert (Ins)" key, you activate the Cp (overwrite) mode. After that, you can, for example, modify the type of timer (such as change ON delay to OFF delay), without rewiring the inputs and outputs.

## Exercise: Jog Motor (FC 16)



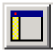
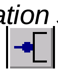
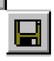




**Task**

Using the simulator momentary contact switch I 0.2, you should be able to jog the conveyor motor to the RIGHT (Q 20.5 or Q 8.5). Using the simulator momentary contact switch I 0.3 you should be able to jog the conveyor motor to the LEFT (Q 20.6 or Q 8.6). If both momentary contact switches are pressed simultaneously, then the conveyor motor cannot be set in either direction (Lock-out!).

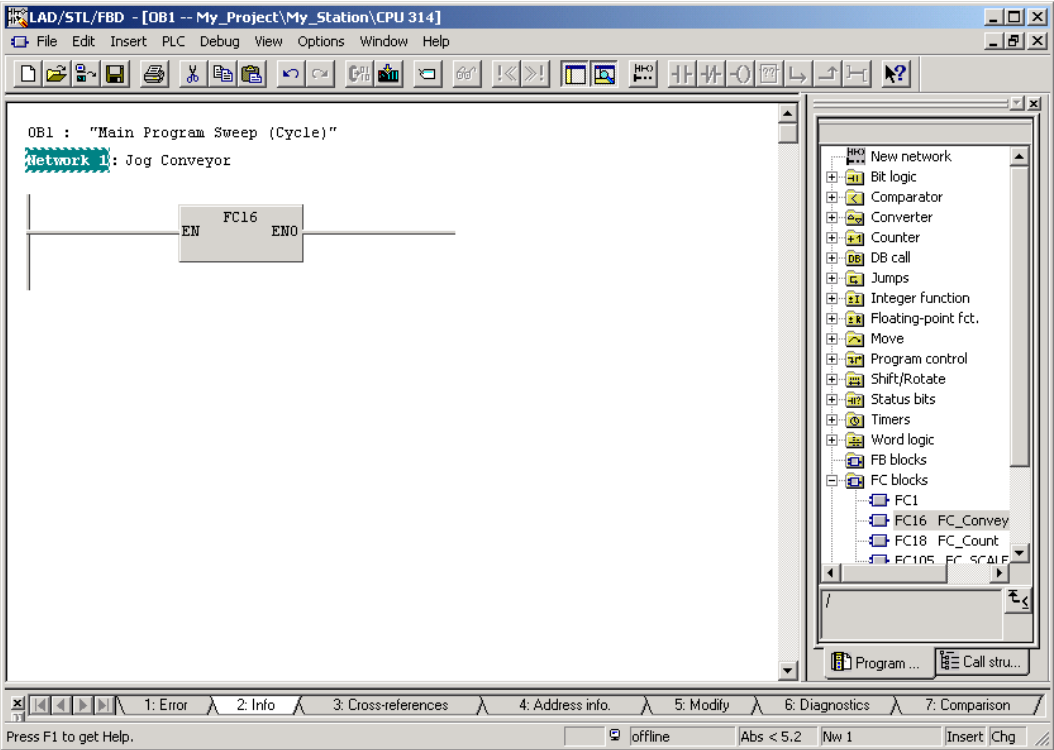
**What To Do**

1. Insert a new FC16 block in the SIMATIC® Manager,.
   *Select Blocks folder -> Insert -> S7 Block -> Function ->*
   *in the Properties box choose FBD as the programming language*
2. Start the LAD/STL/FBD Editor by double-clicking FC 16
3. Open the Program Elements browser in Overviews with 
4. Edit Network 1 of FC16 (see slide)
   *using drag & drop, copy the required logic symbol from the Program Elements browser to the chosen spot in the code section of the block -> label the addresses at the logic operation symbol -> to negate the address scan, select the addresses and then* 
5. Add a new network using  and program Network 2 analog
6. Save the block offline using 
7. Download the block into the CPU using 

**Switching the Programming Language**

Also observe your block in the LAD/STL/FBD programming languages.
From the *LAD/STL/FBD Editor -> View ->* choose either *LAD,STL, or FBD*

## Exercise: Calling FC 16 in OB 1



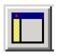
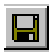

SIMATIC® S7


Date: 12.03.03
File: PRO1_05E.19

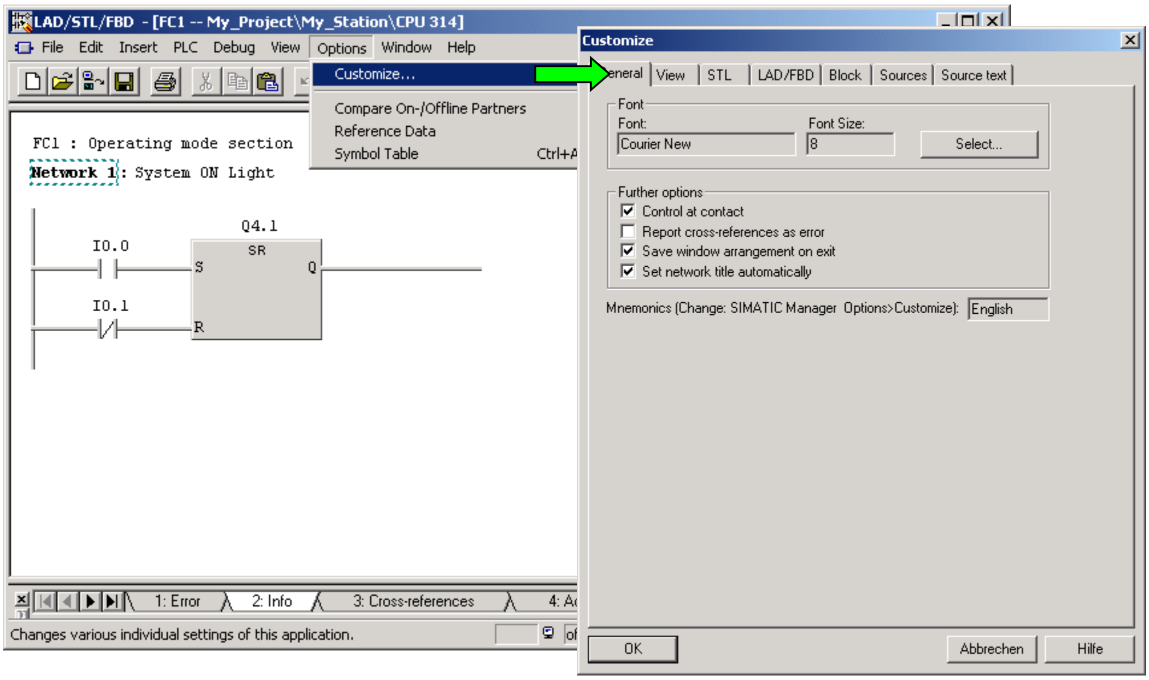
SITRAIN Training for Automation and Drives

**Task** In OB 1, program the call of FC 16 so that it is cyclically executed.

**What To Do**

1. Open the OB 1 block with the LAD/STL/FBD Editor
2. In the "View" menu, select the FBD programming language
3. Open the "Program Elements" browser in Overviews using 
4. In the browser, open the "FC Blocks" folder and, using drag & drop, drag the FC 16 onto Network 1 of OB 1.
5. Save the block offline using 
6. Download the block into the CPU using 
7. Open the FC 16 block once more with the LAD/STL/FBD Editor
8. Test the FC 16 function using 

## Editor Customization: "General" Tab








**Font** Here you select using "Select" the font and the size of the text to be used for programming blocks.

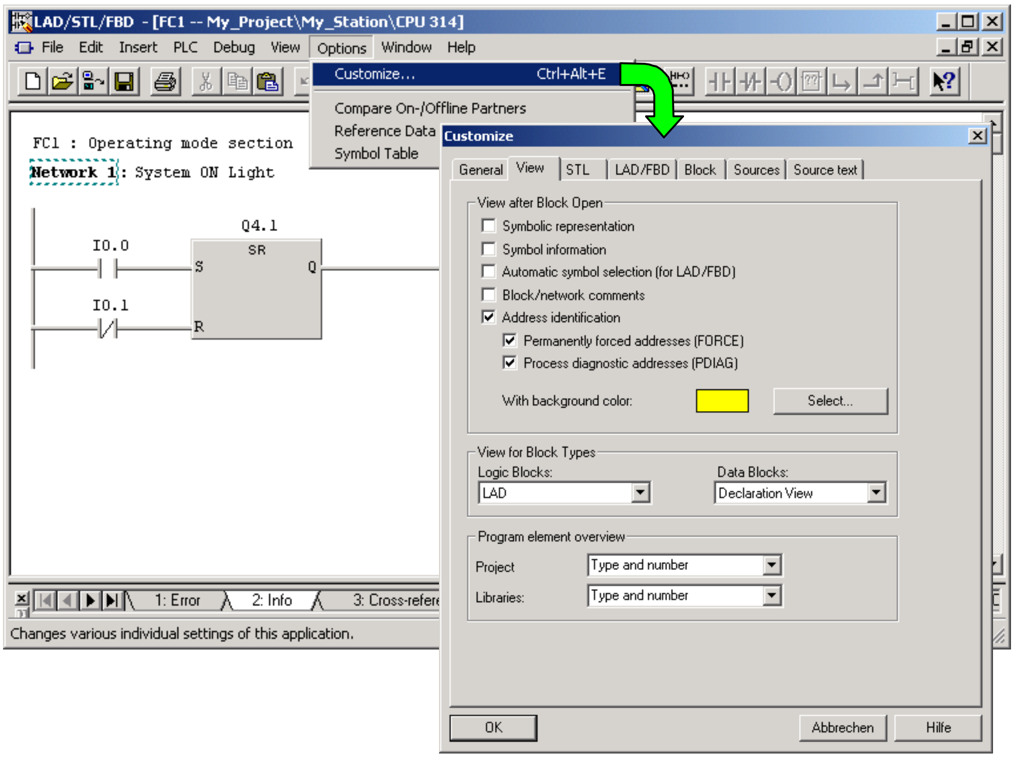
**Control at Contact** Inputs and bit memories that were given the attribute CC (**C**ontrol at **C**ontact) in the symbol table, can be controlled directly from the Program Editor using buttons (on the contact).

**Report Cross References as Error** Here you can specify that global accesses to instance data blocks, that were entered as such in the symbol table, be reported as errors.

**Save Window Arrangement on Exit** The contents and the arrangement of possibly still open windows are saved when you exit. The next time you start, they are reestablished.

**Set Network Title Automatically** Here you can specify that the symbol comment of the first output, bit memory, timer or counter address that is assigned a state in a network ("=", "S" and "R"), be automatically used as network title.

## Editor Customization: "View" Tab






**View after Block Open**

After opening with the Editor, you can display blocks as follows:

- with symbolic or absolute addressing
- with or without symbol information
- with or without symbol selection (only in LAD and FBD)
- with or without block and network comments
- in the language in which they were written (language in which the block was last saved) or in a preset language (LAD/STL/FBD).

**View for Block Types:**

**Logic Blocks**

You use the "STL", "LAD", "FBD" options to select the language in which you want to write a new block.

Multi-instance function blocks are discussed in an advanced programming course.

**Data Blocks**

You can display data blocks in the following views:

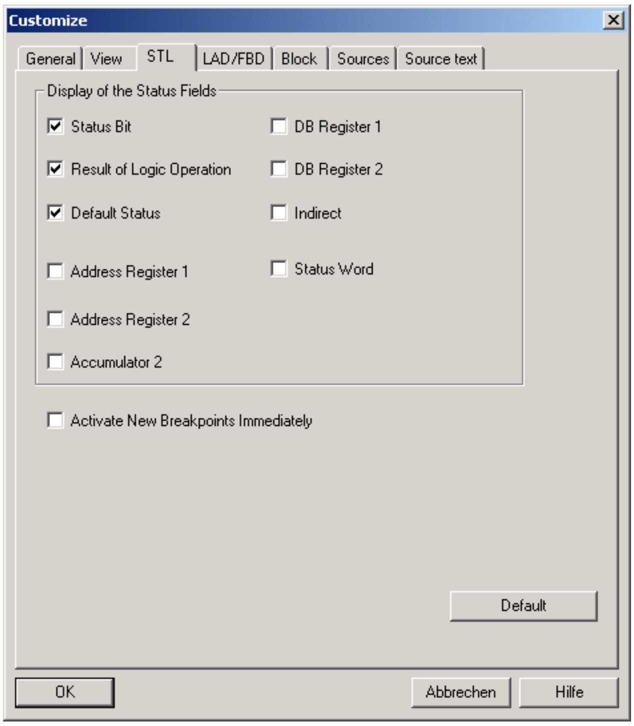
- declaration view or
- data view .

**Program Elements - Overview**

Here you can specify how logic blocks are to be sorted in the "Overviews" browser - according to type and number or according to family name (entry in a block's Properties dialog).

SIEMENS

## Editor Customization: "STL" Tab




SIMATIC® S7


Date: 12.03.03
File: PRO1_05E.22

SITRAIN Training for Automation and Drives

**Status Fields**

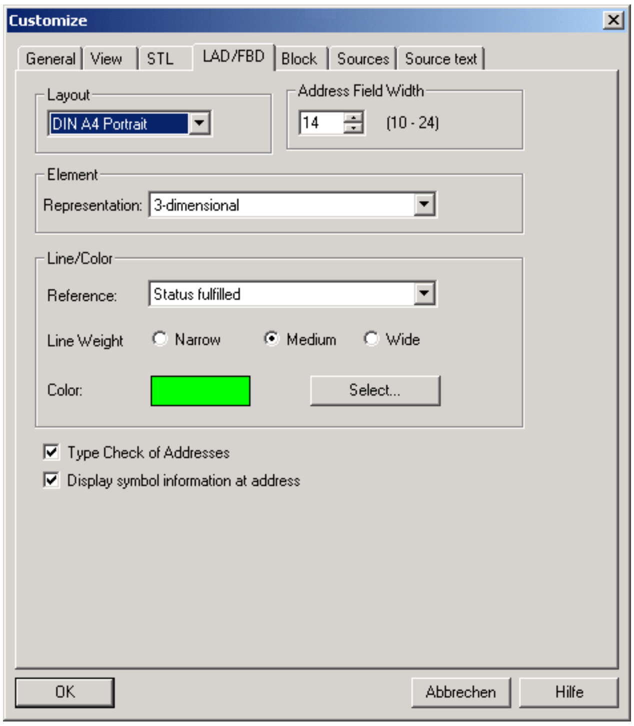
When you monitor the status of a block in STL, only the status fields you activate in this dialog box will be displayed.
The following options are available:

| | |
|---|---|
| • Status Bit | The status bit is displayed. |
| • Result of Logic Operation | The result of logic operation (RLO) is displayed. |
| • Default Status | A timer word, counter word or the contents of Accumulator 1 are displayed - depending on the operation used. |
| • Address Registers *) | The address registers are used with indirect addressing. |
| • Accumulator 2 | The contents of Accumulator 2 are displayed. |
| • DB Registers *) | The contents of the relevant data block register are displayed. |
| • Indirect *) | This display is possible only with memory-indirect addressing. |
| • Status Word | The status word is displayed. |
| • Default | The “Default” button selects the standard system setting for the Status field. The status bit, RLO and standard status are displayed. |
| • Activate New Breakpoints Immediately | This option is relevant only for the “Breakpoint” test function. |

**Note*)**

The topics of “Indirect Addressing” , “DB Registers” and the structure of the status word are discussed in an advanced programming course.

## Editor Customization: "LAD/FBD" Tab




SIMATIC® S7


Date: 12.03.03
File: PRO1_05E.23

SITRAIN Training for Automation and Drives

**Layout** Here you select the print format:

- DIN A4 Portrait
- DIN A4 Landscape
- maximum size.

**Address Field Width** You can set the limit for the maximum number of characters in an address name to a number between 10 and 24. This number changes the width of the program element in LAD and FBD. With symbolic representation, a line break takes place according to the Address Field Width.

**Element** The program elements can be displayed in different ways:

- 2-dimensional (without shadow)
- 3-dimensional (with shadow).

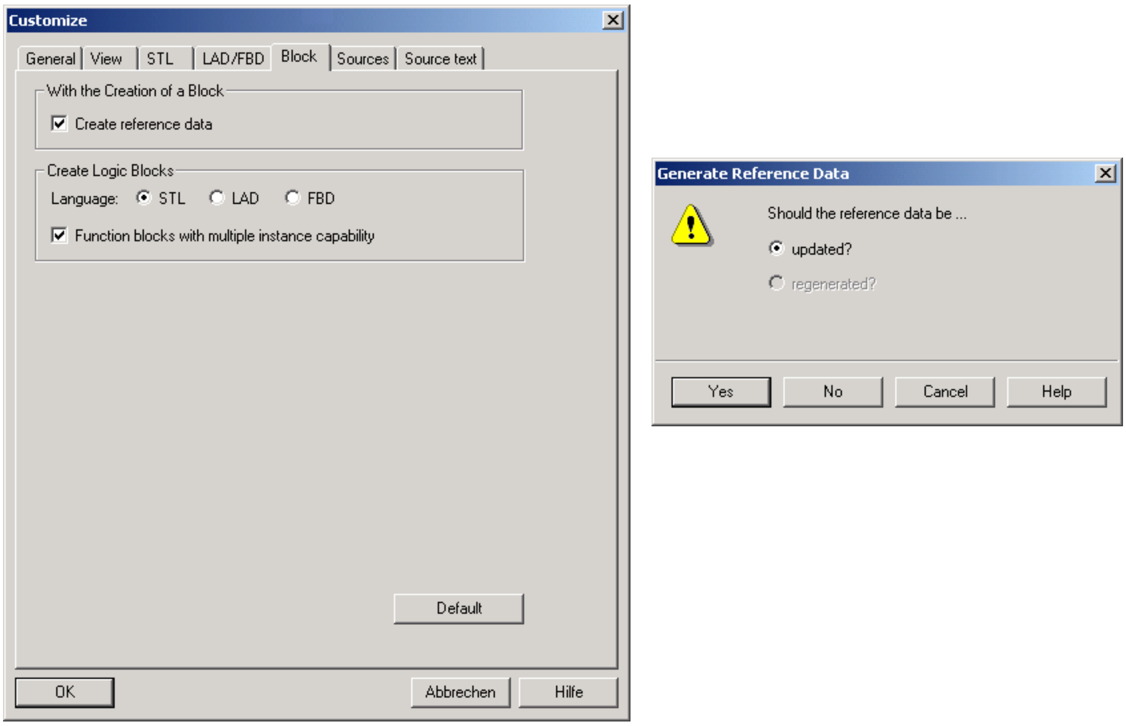
**Line/Color** You use this box to choose how you want the following to be displayed

- Selected Element (color)
- Contacts (line)
- Status Fulfilled (color and line)
- Status Not Fulfilled (color and line)

**Type Check** When you edit a block, the type of address entered in bit logic instructions is always checked.
You can deactivate the Type Check of Addresses: for comparisons, mathematical operations etc. ( for experienced users only! ).

**Symbol Information at Address** If you activate this function, the symbol information is not overlaid at the lower edge of the networks, rather is overlaid directly at the address

# Editor Customization: "Block" Tab




SIMATIC® S7



Date: 12.03.03
File: PRO1_05E.24

SITRAIN Training for Automation and Drives

**Create Reference Data**

When you modify blocks and save them, the reference data is automatically updated, if the "Create Reference Data" option in the "Create Block" tab is checked.

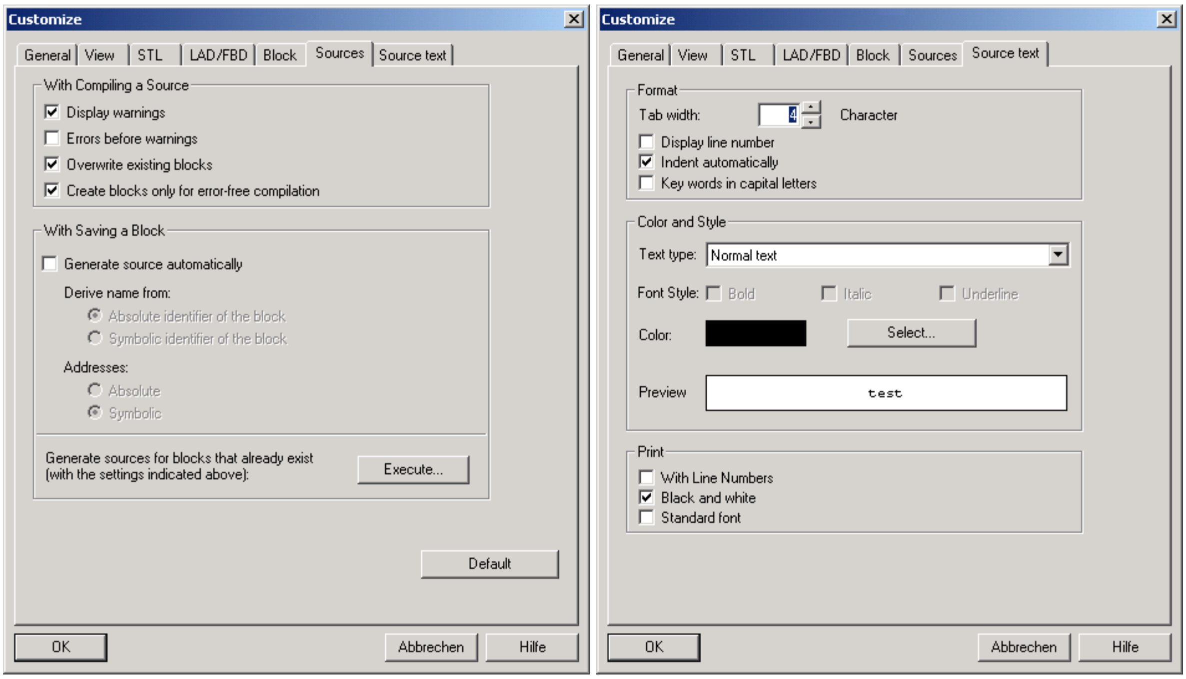
If this option is not checked, the reference data is not updated at first. But the next time you open: *Options -> Reference Data -> Display*, you must decide whether you want to update the reference data and for which blocks.

Note: The topic "Reference Data" is discussed in detail in the "Troubleshooting" chapter.

**Create Logic Blocks**

Here you specify the default language (LAD/STL/FBD) for a new block.

## Editor Cutomization: "Sources/Source Text" Tabs




SIMATIC® S7



Date: 12.03.03
File: PRO1_05E.25

**SITRAIN** Training for Automation and Drives

**Sources**

It is possible to enter a program or parts of a program as an STL source (ASCII sources). The source file can contain the code for one, several, or all blocks. The STL source can then be compiled into executable S7 blocks.

Creating a program using a source has the following advantages:

- You can create and edit your source with any ASCII editor and then inport it into STEP 7. The source can then be compiled into individual, executable S7 blocks with STEP 7.
- You can program several blocks in a source, whereby you can use the advantages of the ASCII editors (such as find and replace one operation in all blocks).
- You can save sources even with syntax errors. This is not possible when creating logic blocks with the incremental LAD/STL/FBD Editor because of the integrated syntax check. As a result, blocks can be programmed with symbolic addresses before a symbol table has even been created. This is an advantage when the wiring of the PLC has not yet been determined, for example.

The source is created in the syntax of the "Statement List (STL)" programming language. The structuring within the sources as well as within the blocks themselves (declaration table, networks etc.) takes place using keywords.

**Compiling Sources**

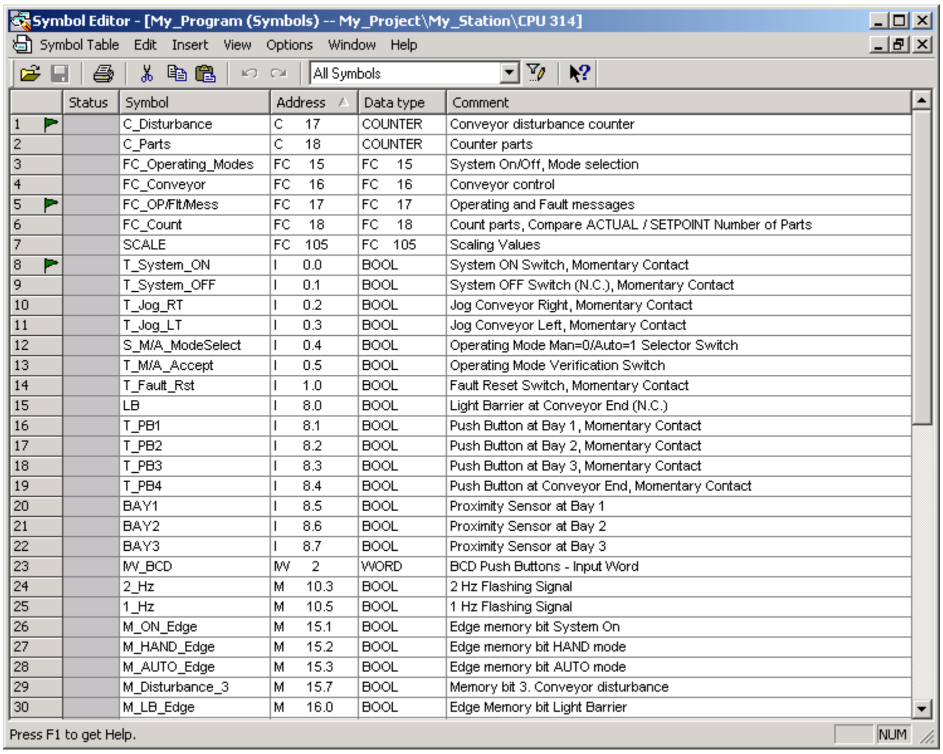
In the "Sources" tab, you can select with which options executable S7 blocks are to be generated from an STL or ASCII source. The individual options are explained completely in the STEP 7 Online Help.

**Source Text**

Here you select options of how the text is to be displayed in the source files. The individual options are explained completely in the STEP 7 Online Help.

SIEMENS

# Symbols




| | Status | Symbol | Address | | Data type | Comment |
|---|---|---|---|---|---|---|
| 1 | | C_Disturbance | C | 17 | COUNTER | Conveyor disturbance counter |
| 2 | | C_Parts | C | 18 | COUNTER | Counter parts |
| 3 | | FC_Operating_Modes | FC | 15 | FC 15 | System On/Off, Mode selection |
| 4 | | FC_Conveyor | FC | 16 | FC 16 | Conveyor control |
| 5 | | FC_OP/Flt/Mess | FC | 17 | FC 17 | Operating and Fault messages |
| 6 | | FC_Count | FC | 18 | FC 18 | Count parts, Compare ACTUAL / SETPOINT Number of Parts |
| 7 | | SCALE | FC | 105 | FC 105 | Scaling Values |
| 8 | | T_System_ON | I | 0.0 | BOOL | System ON Switch, Momentary Contact |
| 9 | | T_System_OFF | I | 0.1 | BOOL | System OFF Switch (N.C.), Momentary Contact |
| 10 | | T_Jog_RT | I | 0.2 | BOOL | Jog Conveyor Right, Momentary Contact |
| 11 | | T_Jog_LT | I | 0.3 | BOOL | Jog Conveyor Left, Momentary Contact |
| 12 | | S_M/A_ModeSelect | I | 0.4 | BOOL | Operating Mode Man=0/Auto=1 Selector Switch |
| 13 | | T_M/A_Accept | I | 0.5 | BOOL | Operating Mode Verification Switch |
| 14 | | T_Fault_Rst | I | 1.0 | BOOL | Fault Reset Switch, Momentary Contact |
| 15 | | LB | I | 8.0 | BOOL | Light Barrier at Conveyor End (N.C.) |
| 16 | | T_PB1 | I | 8.1 | BOOL | Push Button at Bay 1, Momentary Contact |
| 17 | | T_PB2 | I | 8.2 | BOOL | Push Button at Bay 2, Momentary Contact |
| 18 | | T_PB3 | I | 8.3 | BOOL | Push Button at Bay 3, Momentary Contact |
| 19 | | T_PB4 | I | 8.4 | BOOL | Push Button at Conveyor End, Momentary Contact |
| 20 | | BAY1 | I | 8.5 | BOOL | Proximity Sensor at Bay 1 |
| 21 | | BAY2 | I | 8.6 | BOOL | Proximity Sensor at Bay 2 |
| 22 | | BAY3 | I | 8.7 | BOOL | Proximity Sensor at Bay 3 |
| 23 | | IW_BCD | IW | 2 | WORD | BCD Push Buttons - Input Word |
| 24 | | 2_Hz | M | 10.3 | BOOL | 2 Hz Flashing Signal |
| 25 | | 1_Hz | M | 10.5 | BOOL | 1 Hz Flashing Signal |
| 26 | | M_ON_Edge | M | 15.1 | BOOL | Edge memory bit System On |
| 27 | | M_HAND_Edge | M | 15.2 | BOOL | Edge memory bit HAND mode |
| 28 | | M_AUTO_Edge | M | 15.3 | BOOL | Edge memory bit AUTO mode |
| 29 | | M_Disturbance_3 | M | 15.7 | BOOL | Memory bit 3. Conveyor disturbance |
| 30 | | M_LB_Edge | M | 16.0 | BOOL | Edge Memory bit Light Barrier |



SIMATIC® S7



Date: 12.03.03
File: PRO1_06E.1

SITRAIN Training for Automation and Drives

## Contents

Page

## Objectives

**Upon completion of this chapter the participant will ...**

- ... know the difference between absolute and symbolic addressing
- ... know the difference between local and global symbols
- ... know the difference between leading symbols and leading absolute addresses
- ... be able to edit a global symbol table
- ... also be able to edit global symbols from the LAD/STL/FBD Editor
- ... be able to import and export a symbol table

SIMATIC® S7


Date: 12.03.03
File: PRO1_06E.2

SITRAIN Training for Automation and Drives

SIEMENS

## Absolute and Symbolic Addressing

```
A      I 0.0
=      Q4.1
A      I 0.4
=      Q8.5
Call   FC18
```

```
A      "T_System_ON"
=      "L_SYSTEM"
A      "S_M/A_ModeSelect"
=      "K_RT"
Call   "FC_Count"
```

| Symbol | Address | Data Type | Comment |
|---|---|---|---|
| K_RT | Q8.5 | BOOL | Run Conveyor Right |
| FC_Count | FC18 | FC18 | Count Transported Parts |
| T_System_ON | I 0.0 | BOOL | System ON Switch, Momentary Contact |
| L_SYSTEM | Q4.1 | BOOL | System ON Light |
| S_M/A_ModeSelect | I 0.4 | BOOL | Operating Mode Man=0/Auto=1 Selector Switch |

(max. 24 characters) (max. 80 characters)

SIMATIC® S7


Date: 12.03.03
File: PRO1_06E.3

SITRAIN Training for Automation and Drives

**Absolute Addressing**

In absolute addressing, you specify the address (such as input I 1.0) directly. In this case you don't need a symbol table, but the program is harder to read.

**Symbolic Addressing**

In symbolic addressing, you use symbols (such as MOTOR_ON) instead of the absolute addresses.
You store the symbols for inputs, outputs, timers, counters, bit memories and blocks in the symbol table.

**Note**

When you enter symbol names, you don't have to include quotation marks. The Program Editor adds these for you.

## Symbolic Addressing - Overview

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

SIMATIC® S7


Date: 12.03.03
File: PRO1_06E.4

**SITRAIN** Training for Automation and Drives

**Global Symbols** Global symbols are declared in the symbol table and can be used in all blocks of a program.

The name in the symbol table must be unique, that is, a symbolic name must appear only once in the table.

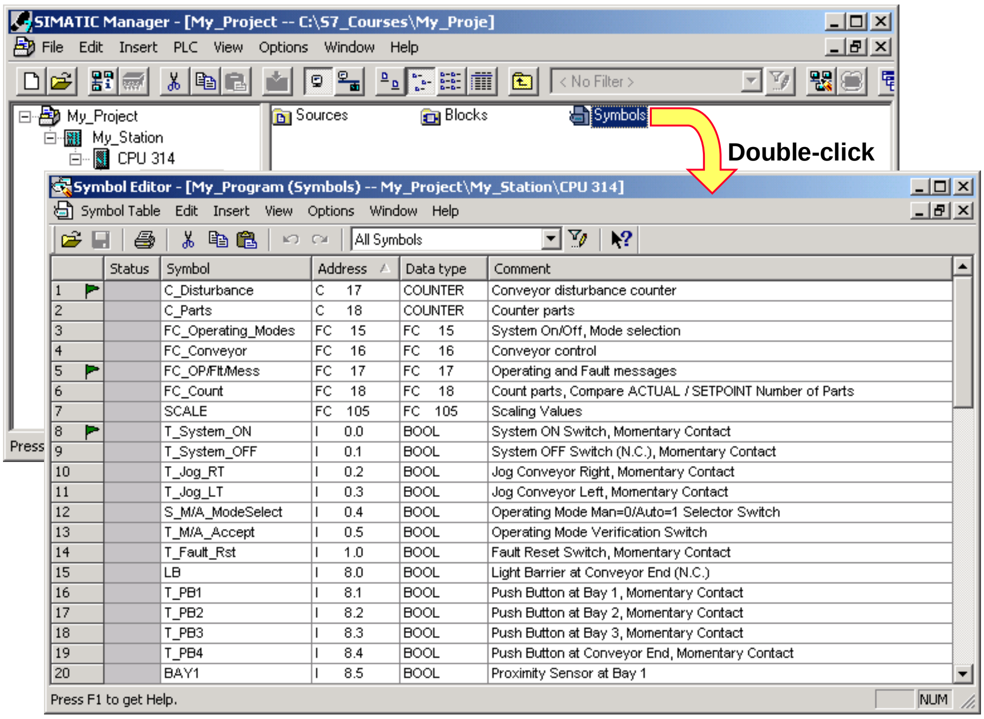
**Local Symbols** Local symbols are declared in the declaration part of a block. They can be used only within that block.

The same symbolic name can be used again in the declaration part of another block.

**Notes** The LAD/STL/FBD Editor always displays symbols declared in the global symbol table in quotation marks. Local address symbols (local variables and parameters) are always displayed with a # (hash or pound mark).

You don't have to include the quotation marks or the hash mark when you enter symbolic addresses. The program editor automatically adds these for you.

# The Symbol Table




| | Status | Symbol | Address | | Data type | | Comment |
|---|---|---|---|---|---|---|---|
| 1 | | C_Disturbance | C | 17 | COUNTER | | Conveyor disturbance counter |
| 2 | | C_Parts | C | 18 | COUNTER | | Counter parts |
| 3 | | FC_Operating_Modes | FC | 15 | FC | 15 | System On/Off, Mode selection |
| 4 | | FC_Conveyor | FC | 16 | FC | 16 | Conveyor control |
| 5 | | FC_OP/Flt/Mess | FC | 17 | FC | 17 | Operating and Fault messages |
| 6 | | FC_Count | FC | 18 | FC | 18 | Count parts, Compare ACTUAL / SETPOINT Number of Parts |
| 7 | | SCALE | FC | 105 | FC | 105 | Scaling Values |
| 8 | | T_System_ON | I | 0.0 | BOOL | | System ON Switch, Momentary Contact |
| 9 | | T_System_OFF | I | 0.1 | BOOL | | System OFF Switch (N.C.), Momentary Contact |
| 10 | | T_Jog_RT | I | 0.2 | BOOL | | Jog Conveyor Right, Momentary Contact |
| 11 | | T_Jog_LT | I | 0.3 | BOOL | | Jog Conveyor Left, Momentary Contact |
| 12 | | S_M/A_ModeSelect | I | 0.4 | BOOL | | Operating Mode Man=0/Auto=1 Selector Switch |
| 13 | | T_M/A_Accept | I | 0.5 | BOOL | | Operating Mode Verification Switch |
| 14 | | T_Fault_Rst | I | 1.0 | BOOL | | Fault Reset Switch, Momentary Contact |
| 15 | | LB | I | 8.0 | BOOL | | Light Barrier at Conveyor End (N.C.) |
| 16 | | T_PB1 | I | 8.1 | BOOL | | Push Button at Bay 1, Momentary Contact |
| 17 | | T_PB2 | I | 8.2 | BOOL | | Push Button at Bay 2, Momentary Contact |
| 18 | | T_PB3 | I | 8.3 | BOOL | | Push Button at Bay 3, Momentary Contact |
| 19 | | T_PB4 | I | 8.4 | BOOL | | Push Button at Conveyor End, Momentary Contact |
| 20 | | BAY1 | I | 8.5 | BOOL | | Proximity Sensor at Bay 1 |

SIMATIC® S7


Date: 12.03.03
File: PRO1_06E.5

**SITRAIN** Training for Automation and Drives

**Opening the Symbol Table**

Every "S7 program" has its own symbol table. You can open a symbol table from the SIMATIC® Manager with a double-click on the "Symbols" icon.

You can also open the symbol table from the LAD/STL/FBD Editor using the *Options -> Symbol Table* menu option.

**Table Structure**

In the symbol table, a line is created for every variable. You can then enter the symbol name, the address, the data type and a comment for the variable in the columns. A blank line is automatically added at the end of the table for defining a new symbol.

**"Status" Column**

Invalid symbol definitions are marked as follows in this column:

**=** The symbol name or address is identical to another entry in the symbol table.

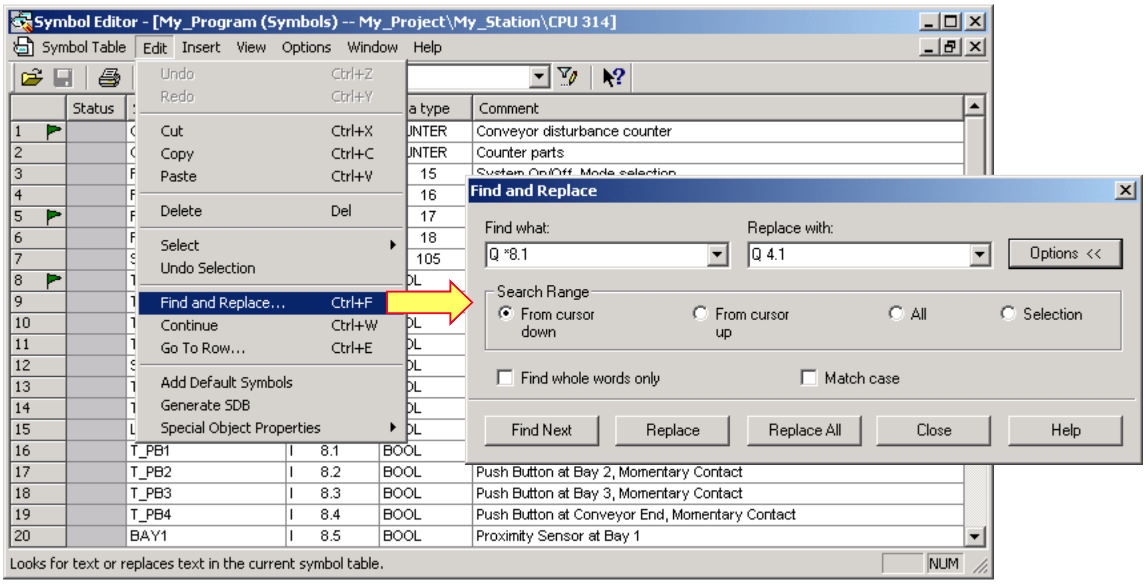
**X** The symbol is incomplete (the symbol name and/or the address is missing).

**Note**

As soon as a symbol table has been created, it is available to all other tools (such as LAD/STL/FBD Editor, HW-Config, and Monitor/Modify Variables).

SIEMENS

## Edit: Find and Replace




SIMATIC® S7

Date: 12.03.03
File: PRO1_06E.6
SITRAIN Training for Automation and Drives

**Find and Replace**

A number of options are available for finding and replacing text in the current window:

- Find what:
  Enter the text you are looking for.
- Replace with:
  Enter the replacement text.
- From cursor down:
  Searches downwards to the last line in the symbol table.
- From cursor up:
  Searches upwards to the first line in the symbol table.
- Match case:
  Only searches for the specified text with identical use of uppercase and lowercase letters.
- Find whole words only:
  Searches for the specified text as a separate word, not as part of a longer word.
- All:
  Searches through the whole symbol table, starting from the cursor position.
- Selection:
  Searches only the selected symbol lines.

**Note**

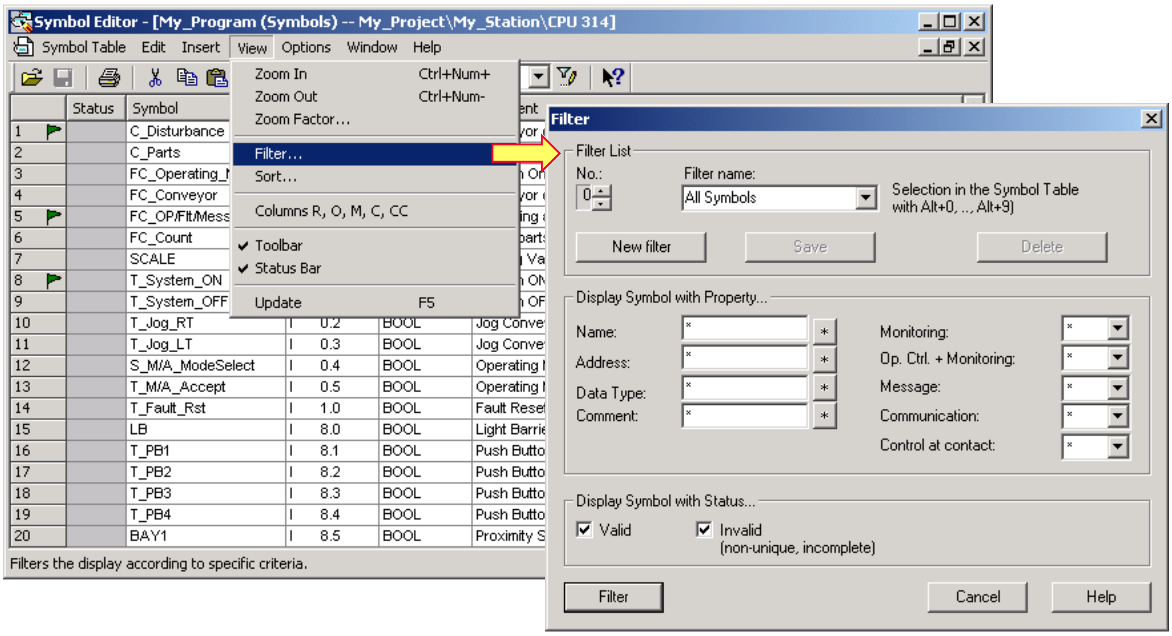
When looking for addresses, you must insert a wildcard after the address identifier, otherwise the address cannot be found. There are two wildcards that can be used:

- Asterisk (*) for no character, or one or more unspecified characters;
- Question mark (?) for one unspecified character.

Example: replace all outputs with address 8. with address 4.:

| Find what: | Replace with: |
| --- | --- |
| Q*8.* | Q 4. |

## View: Filter




SIMATIC® S7


Date: 12.03.03
File: PRO1_06E.7

SITRAIN Training for Automation and Drives

**Filter**

Only the symbols which meet the active filter criteria ("symbol properties") are displayed in the current window.

You can apply several criteria at once. The specified filter criteria are linked with one another.

**Symbol Properties**

You can select various filters and link them according to the following properties: Name, Address, Data Type, Comment, Operator Control & Monitoring, Communication, Message and Monitoring.

Permissible wildcards are * and ?.

**Examples**

Name: M*

Only the names that begin with "M" and that contain <u>any number</u> of additional characters are displayed in the symbol table.

Name: SENSOR_?
Only the names that begin with "SENSOR_" and that contain <u>one</u> other character are displayed in the symbol table.

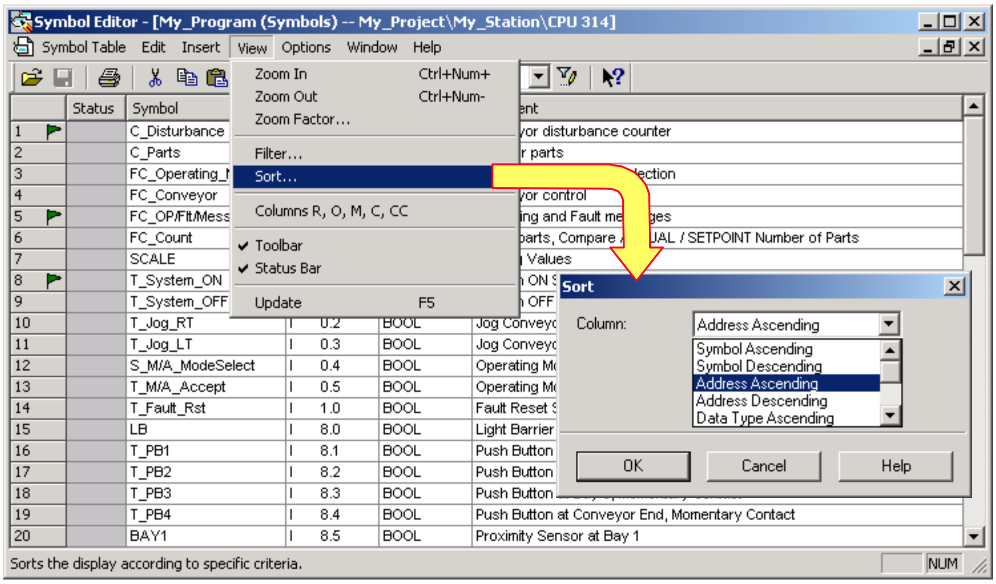
Address: I*.*
Only the inputs are displayed.

**Valid, Invalid**

The symbols must be unique, that is, a symbol or an address must exist only once in the symbol table.

If a symbol or an address appears more than once, the lines in which it appears are displayed in "**Bold**". If your symbol table is long, and you want to find such ambiguous symbols or addresses more quickly, you can display only these lines of the symbol table by selecting the menu option *View -> Filter* and the attribute "Invalid".

## View: Sort




SIMATIC® S7


Date: 12.03.03
File: PRO1_06E.8

SITRAIN Training for Automation and Drives

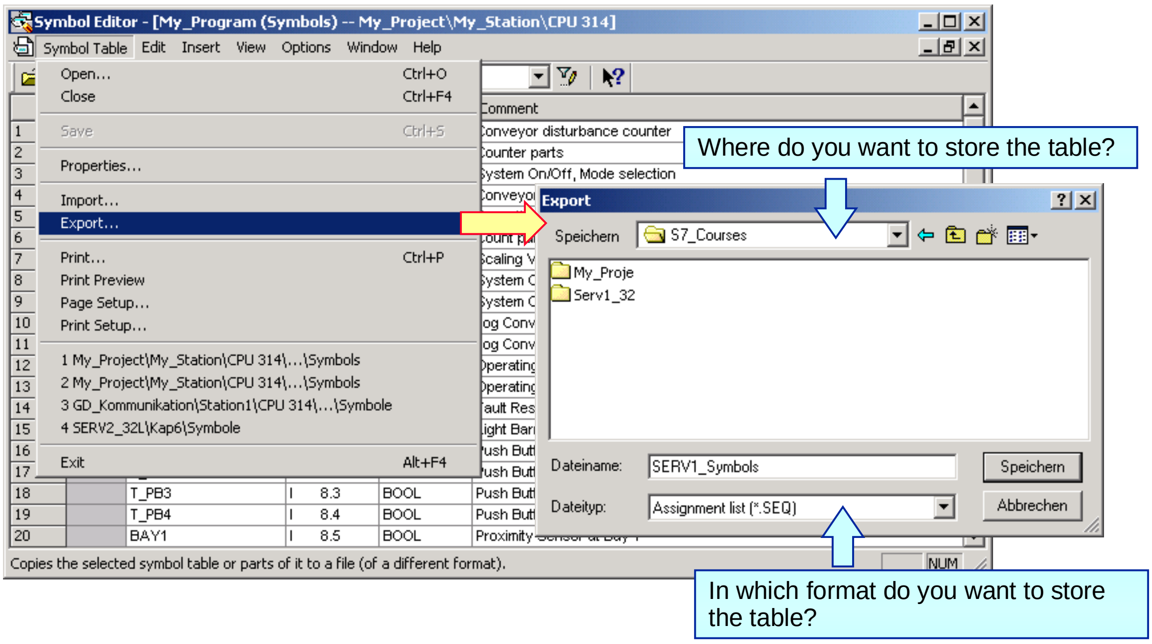
**Sort**

The entries in the symbol table can be displayed in alphabetical order. You use the *View -> Sort* menu option to specify the column to be used as the point of reference for sorting in the current window.

There is an alternative way to sort:

1. Click the column heading "Symbol, Address, Data type or Comment" for sorting in ascending order in this column. Answer "Yes" when prompted.
2. Click the column heading once more for sorting in descending order in this column. Answer "Yes" when prompted.

## Symbol Table: Export




SIMATIC® S7



Date: 12.03.03
File: PRO1_06E.9

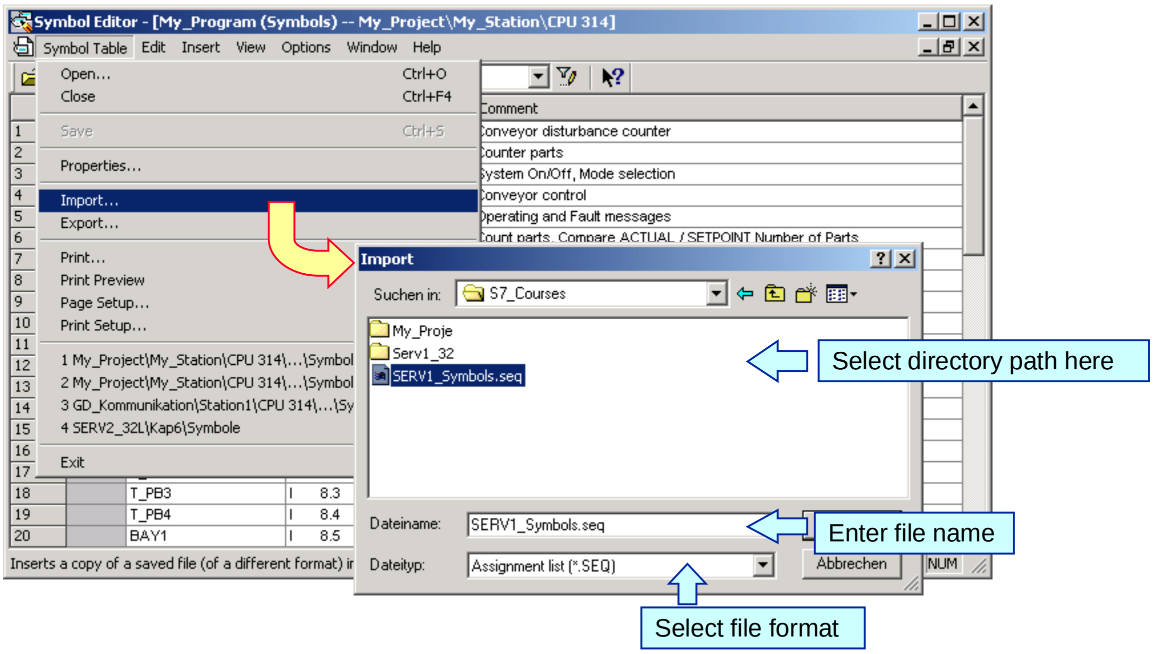
SITRAIN Training for Automation and Drives

**General**

The *Symbol Table -> Export* menu option enables you to store symbol tables in different file formats so that you can work on them with other programs. You can select the following file formats:

- ASCII Format (*.ASC)
  - Notepad
  - Word
- Data Interchange Format (*.DIF)
  - EXCEL
- System Data Format (*.SDF)
  - ACCESS
- Assignment List (*.SEQ)
  - STEP 5 assignment list

## Symbol Table: Import




SIMATIC® S7


Date: 12.03.03
File: PRO1_06E.10

SITRAIN Training for Automation and Drives

**General**

The *Symbol Table -> Import* menu option enables you to import symbol tables that were created with other user programs.

What to do:

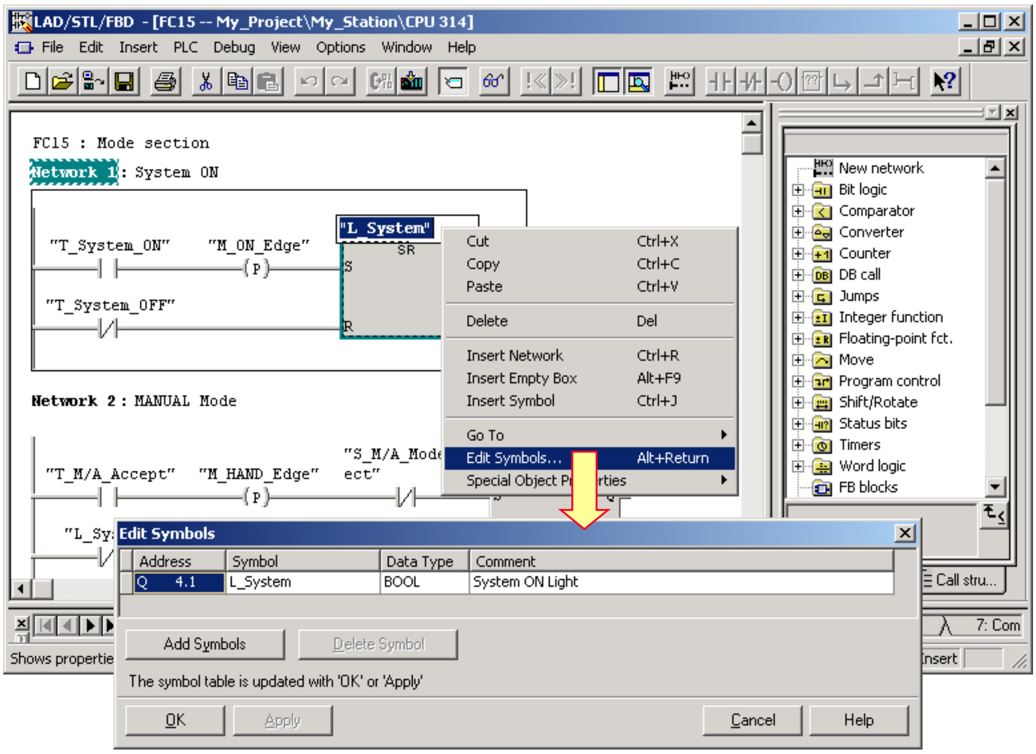
1. Activate the *Symbol Table -> Import* menu option.
2. Select the file format in the "Import" dialog window.
   You will find the same formats as for Export.
3. Select the directory path in the "Look in:" list box.
4. Enter the file name in the "File Name:"  box
5. Click the "Open" button.

**File Types**

You can import the following file formats:

- ASCII Format (*.ASC)
  - Notepad
  - Word
- Data Interchange Format (*.DIF)
  - EXCEL
- System Data Format (*.SDF)
  - ACCESS
- Assignment List (*.SEQ)
  - STEP 5 assignment list

# Editing Symbols in the LAD/STL/FBD Editor




SIMATIC® S7


Date: 12.03.03
File: PRO1_06E.11

SITRAIN Training for Automation and Drives

**Edit Symbols**

Edit Symbols enables you to assign symbolic names to absolute addresses at a later time. These assigned names are automatically entered into the symbol table.

**What to do**

There are two ways to get to the “Edit Symbols“ from the LAD,STL,FBD editor:

- highlight an address field and choose Edit -> Symbols… or
- right mouse click the address field and select Edit Symbols...

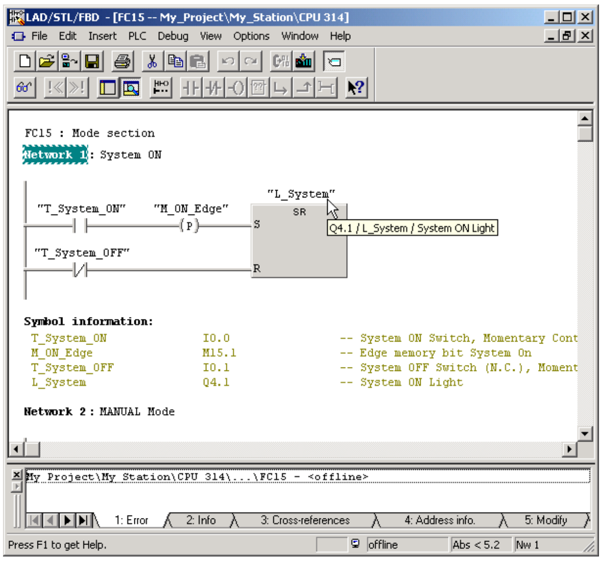
Enter the Symbol name, Data Type and Comment you want to assign to that address and then OK.

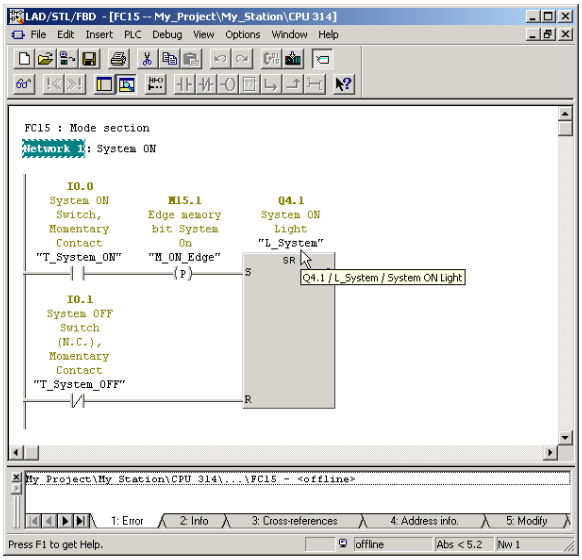
**Note**

If you assign a name that is already in the symbol table, it will be displayed in a different color. Duplicate names cannot be used in the symbol table.

**SIEMENS**

# Symbol Information in the LAD/STL/FBD Editor




SIMATIC® S7


Date: 12.03.03
File: PRO1_06E.12

**SITRAIN** Training for Automation and Drives

**Addressing**

In the LAD/STL/FBD Editor you can choose to display the addresses in one of the following two ways when you select the *View -> Display with -> Symbolic Representation* menu option:

- Symbolic Addressing or
- Absolute Addressing.

You can display the symbolic and absolute address assignments used in the network along with their comments by selecting the *View -> Display with -> Symbol Information* menu option.

The assignments are found under the network in LAD/FBD and in STL they are found in the statement line.

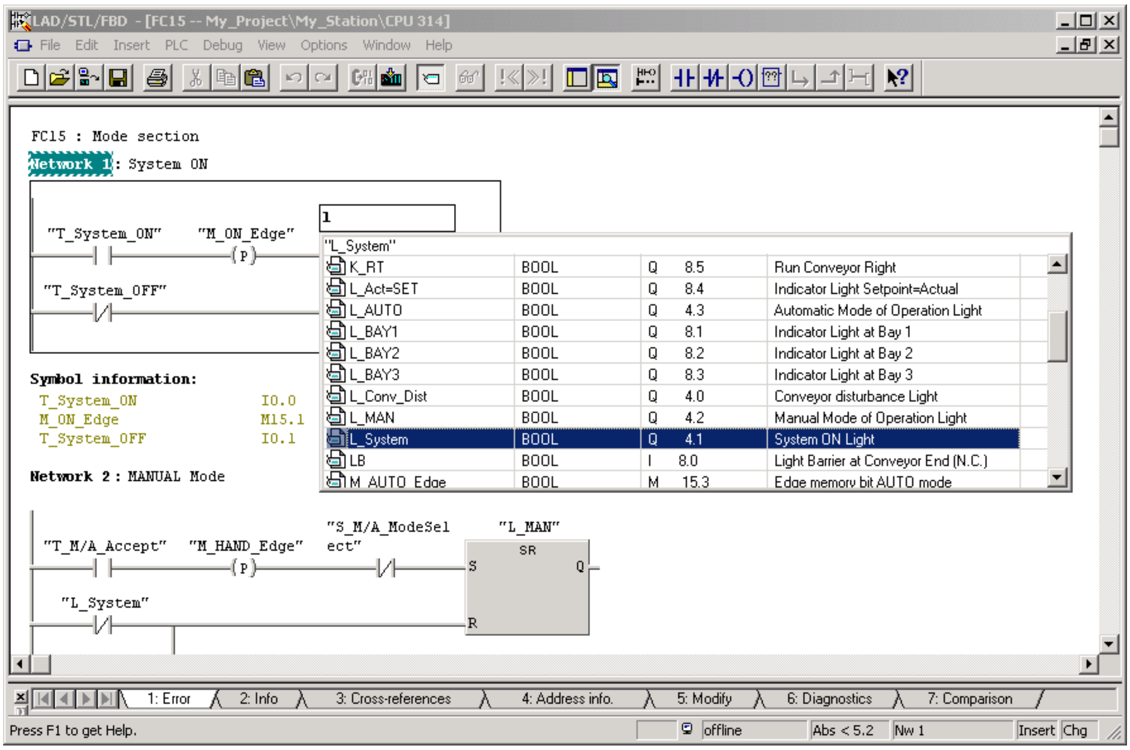
**Symbol Information at the Address Yes/No**

In the LAD/STL/FBD Editor settings you can select whether the symbol information for the addresses is to be displayed directly at the address (see right picture) or at the lower edge of the network (see left picture).

**Note**

If you position the mouse pointer on an address, a "Tooltip" appears with the symbol information for this address.

## Symbol Selection in the LAD/STL/FBD Editor




SIMATIC® S7


Date: 12.03.03
File: PRO1_06E.13

**SITRAIN** Training for Automation and Drives

**Introduction**

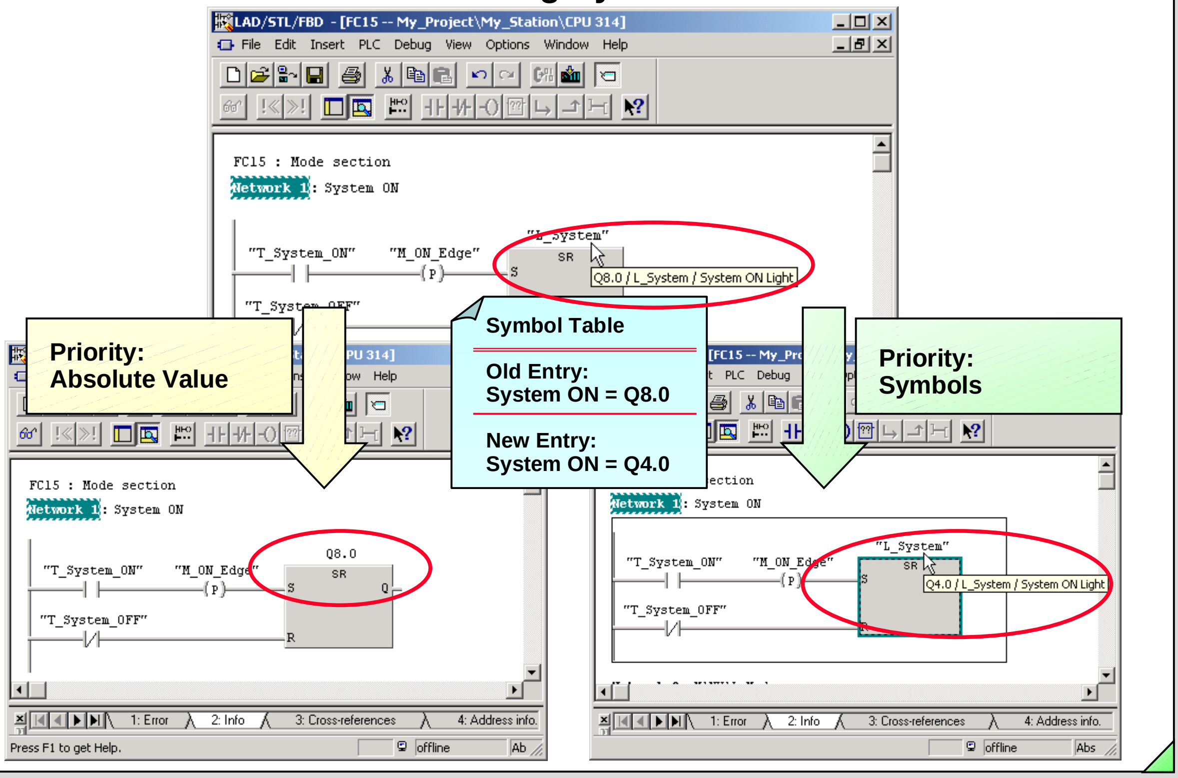
You can use the *View -> Display with -> Symbol Selection* menu option to simplify writing a symbolic program.

When you label the adddress and you enter the first letter of a symbol name, a section of the symbol table pops up that starts with this letter.

All valid addresses for this block are displayed. These can be all global variables (even those declared in data blocks), local variables (temporary and static) and the parameters of the affected block.

SIEMENS

## "Leading Symbols"




SIMATIC® S7


Date: 12.03.03
File: PRO1_06E.14

SITRAIN Training for Automation and Drives

**Introduction**

If you want to change the assignments in the symbol table of an already existing program, you can also decide whether the absolute address or the symbolic address has priority.

**Selection**

In the SIMATIC® Manager, right mouse click the "Blocks" object of an S7 program. Select the *Object Properties* menu option and then the "Blocks" tab. You can choose between "Absolute Value" or "Symbol" in the "Address priority" field.

**Priority: Absolute Value**

With this setting, the absolute address of an operand does not change if you change the address assignment in the symbol table later on.
In the example above, the output Q8.0 (symbol name "System On") was changed to output Q4.0 in the symbol table. With the "Priority: Absolute Value" setting, the program continues to use the output Q8.0.

**Priority: Symbol**

With this setting, the absolute address of the operand is changed to the new entry in the symbol table.
In the example above, the output Q8.0 (symbol name "System On") was changed to output Q4.0 in the symbol table. With the "Priority: Symbols" setting, the address is changed from Q8.0 to Q4.0 throughout the entire program.
The changed address also keeps its symbol name. That way you can change the absolute addresses in an existing symbolic user program.

# Exercise: Creating a Symbol Table for the Conveyor Model

| | Version A 16 channel Modules | Version B 32 channel Modules | |
|---|---|---|---|
| L_BAY1 | Q 20.1 | Q 8.1 | Indicator Light at Bay 1 |
| L_BAY2 | Q 20.2 | Q 8.2 | Indicator Light at Bay 2 |
| L_BAY3 | Q 20.3 | Q 8.3 | Indicator Light at Bay 3 |
| L_END | Q 20.4 | Q 8.4 | Indicator Light at Conveyor End |
| K_RT | Q 20.5 | Q 8.5 | Run Conveyor Right |
| K_LT | Q 20.6 | Q 8.6 | Run Conveyor Left |
| K_Horn | Q 20.7 | Q 8.7 | Horn |
| QW_DISPLAY | QW 12 | QW 6 | BCD - Output Display Word |
| LB | I 16.0 | I 8.0 | Light Barrier at Conveyor End (N.C.) |
| T_PB1 | I 16.1 | I 8.1 | Push Button at Bay 1, Momentary Contact |
| T_PB2 | I 16.2 | I 8.2 | Push Button at Bay 2, Momentary Contact |
| T_PB3 | I 16.3 | I 8.3 | Push Button at Bay 3, Momentary Contact |
| T_PB4 | I 16.4 | I 8.4 | Push Button at Conveyor End, Momentary Contact |
| BAY1 | I 16.5 | I 8.5 | Proximity Sensor at Bay 1 |
| BAY2 | I 16.6 | I 8.6 | Proximity Sensor at Bay 2 |
| BAY3 | I 16.7 | I 8.7 | Proximity Sensor at Bay 3 |
| IW_BCD | IW 4 | IW 2 | BCD Push Buttons - Input Word |

SIMATIC® S7


Date: 12.03.03
File: PRO1_06E.15

**SITRAIN** Training for Automation and Drives

**Task**

Create a symbol table for the sensors and actuators of the conveyor model.

**What To Do**

1. Open the FC16 block with the LAD/STL/FBD Editor
2. With the right mouse button, click one after the other the inputs I 0.2 and I 0.3, so as to be able to declare symbols for these inputs using "Edit Symbols"
3. In the SIMATIC® Manager, select the S7 Program "My_Program"
4. Start the Symbol Editor by double-clicking the symbol table icon
5. Edit the symbol table according to your training area (see slide)
6. Save your symbol table

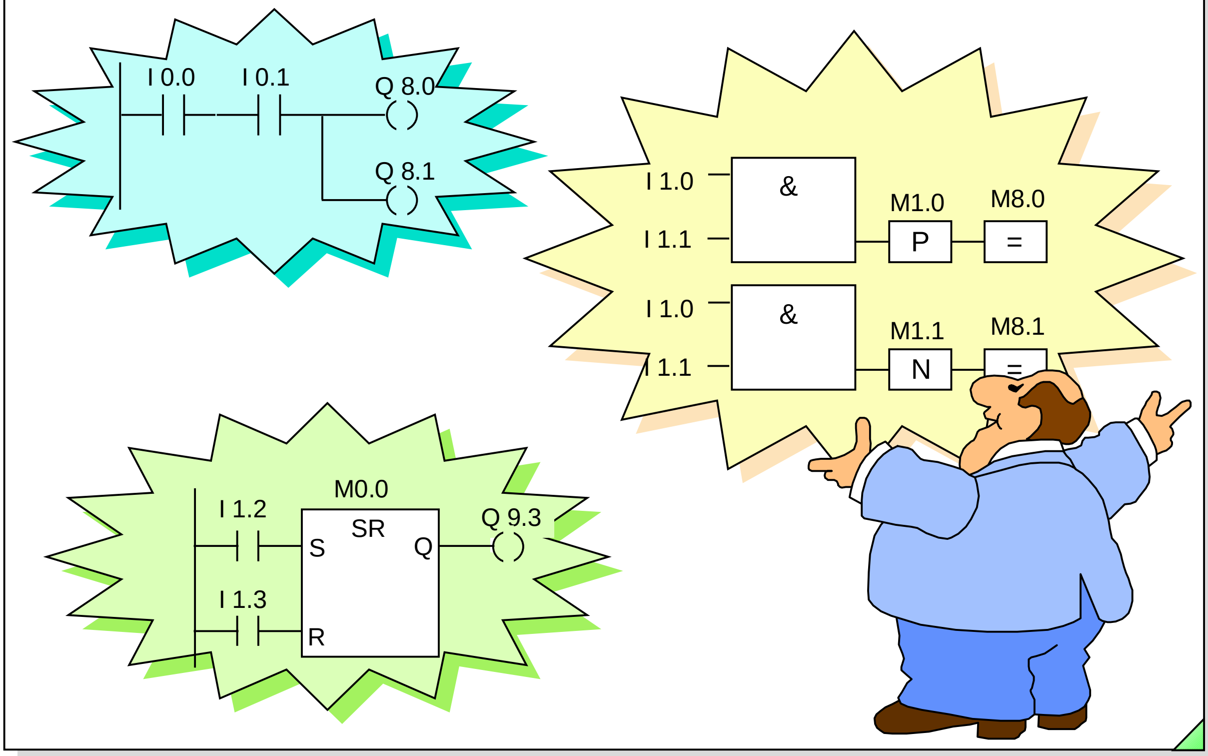
**Result**

All addresses that had a symbolic name assigned to them in the symbol table can be addressed symbolically or absolutely during program creation with the LAD/FBD/STL Editor. As well, you can display the comments from the symbol table as "Symbol Information".

**Note**

A symbol for the FC 105 block is already defined in the symbol table. The entry was automatically made in the symbol table when the block was copied from the library in a previous chapter. The automatic entry of symbols for copied blocks is called "hidden import".

# Binary Operations




SIMATIC® S7


Date: 12.03.03
File: PRO1_07E.1

SITRAIN Training for Automation and Drives

## Contents

Page

# Objectives

**Upon completion of this 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

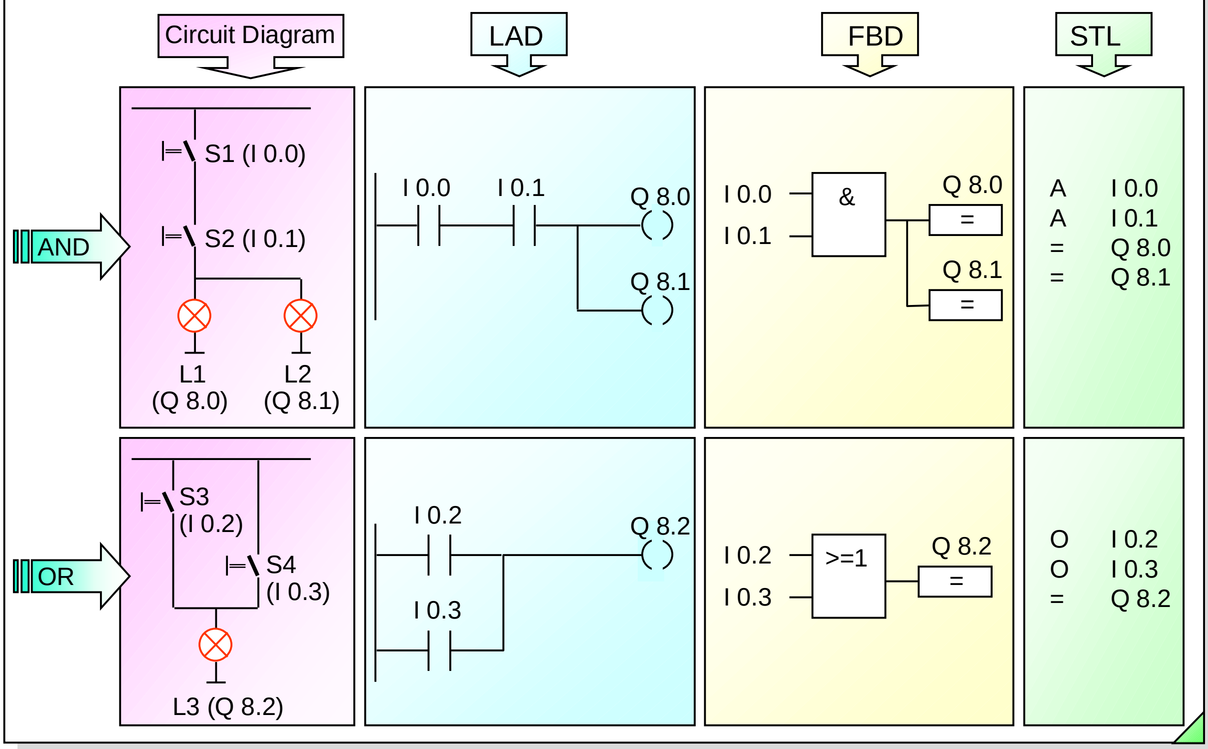
... be able to program basic binary logic operations

... be able to use and program edge detections for problem solving



Date: 12.03.03
File: PRO1_07E.2

## Binary Logic Operations: AND, OR




SIMATIC® S7



Date: 12.03.03
File: PRO1_07E.3

SITRAIN Training for Automation and Drives

## Logic Tables

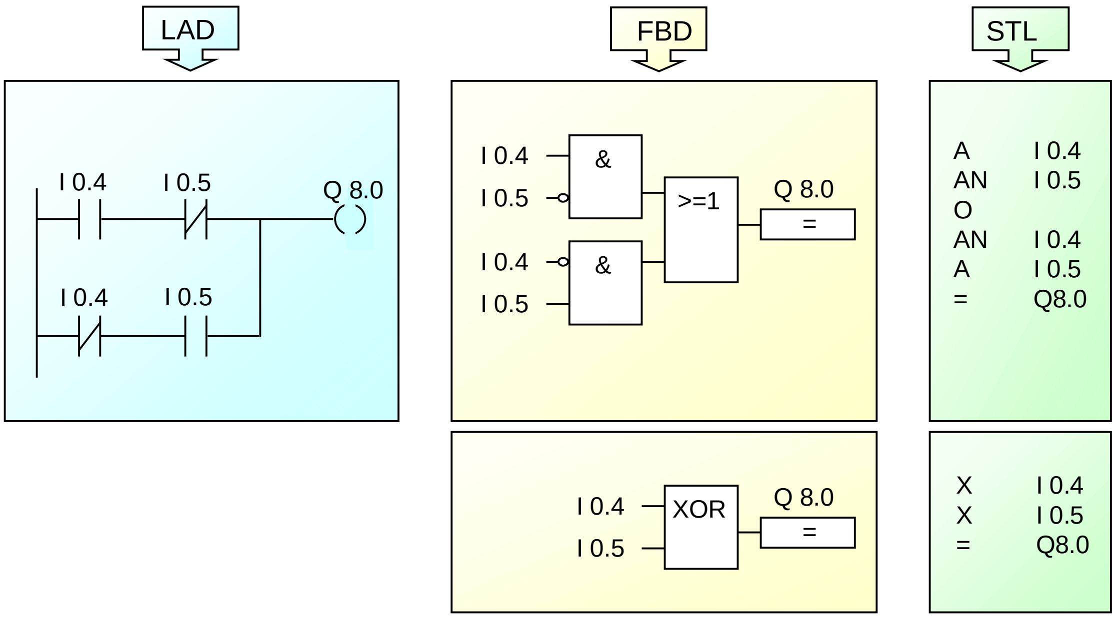
AND

| I 0.0 | I 0.1 | Q 8.0 / Q8.1 |
|---|---|---|
| 0 | 0 | |
| 0 | 1 | |
| 1 | 0 | |
| 1 | 1 | |

OR

| I 0.2 | I 0.3 | Q 8.2 |
|---|---|---|
| 0 | 0 | |
| 0 | 1 | |
| 1 | 0 | |
| 1 | 1 | |

## Binary Logic Operations: Exclusive OR (XOR)




SIMATIC® S7


Date: 12.03.03
File: PRO1_07E.4

SITRAIN Training for Automation and Drives

**Logic Table**

| XOR | I 0.4 | I 0.5 | Q 8.0 |
|---|---|---|---|
| | 0 | 0 | |
| | 0 | 1 | |
| | 1 | 0 | |
| | 1 | 1 | |

**Rule** The following rule is valid for the logic operation of two addresses after XOR: the output has signal state "1", when one and only one of the two checks is fulfilled.

**Careful!** This rule cannot be generalized to "one and only one of n" ! for the logic operation of several addresses after XOR !!

As of the third XOR instruction, the old RLO is gated with the new result of check after XOR.

SIEMENS

## Normally Open and Normally Closed Contacts, Sensors and Symbols

| Process | | | Interpretation in PLC program | | | | |
|---|---|---|---|---|---|---|---|
| The sensor is a ... | The sensor is ... | Voltage present at input? | Signal state at input | Check for signal state "1" – Symbol / Instruction | Check for signal state "1" – Result of check | Check for signal state "0" – Symbol / Instruction | Check for signal state "0" – Result of check |
| NO contact | activated | Yes | 1 | *LAD:* "NO contact" | "Yes" 1 | *LAD:* "NC contact" | "No" 0 |
| | not activated | No | 0 | *FBD:* & | "No" 0 | *FBD:* & | "Yes" 1 |
| NC contact | activated | No | 0 | | "No" 0 | | "Yes" 1 |
| | not activated | Yes | 1 | *STL:* A I x.y | "Yes" 1 | *STL:* AN I x.y | "No" 0 |

SIMATIC® S7


Date: 12.03.03
File: PRO1_07E.5

SITRAIN Training for Automation and Drives

**Process**

The use of normally open or normally closed contacts for the sensors in a controlled process depends on the safety regulations for that process.

Normally closed contacts are always used for limit switches and safety switches, so that dangerous conditions do not arise if a wire break occurs in the sensor circuit.

Normally closed contacts are also used for switching off machinery for the same reason.

**Symbols**

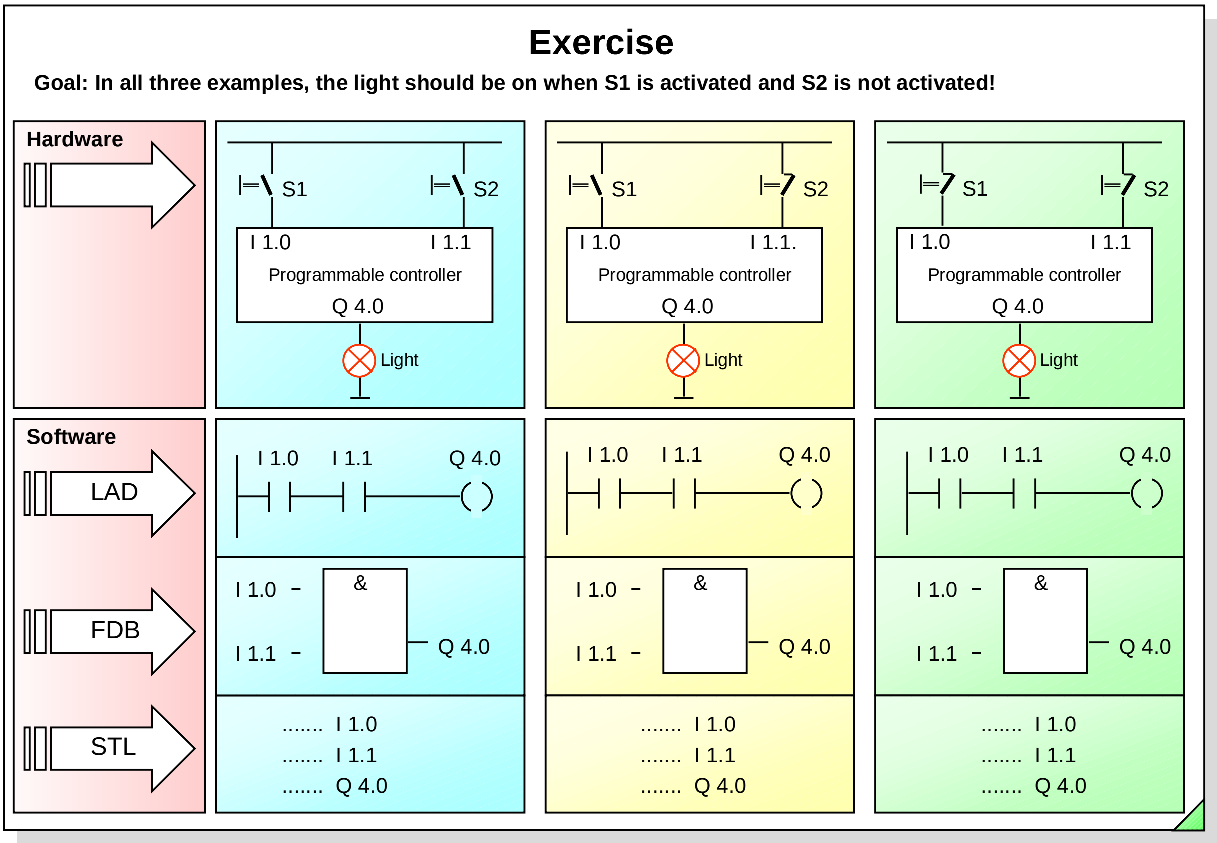
In LAD, a symbol with the name "NO contact" is used for checking for signal state "1" and a symbol with the name "NC contact" to check for signal state "0". It makes no difference whether the process signal "1" is supplied by an activated NO contact or a non-activated NC contact.

**Example**

If an NC contact in the machine is not activated, the signal in the process image table will be "1". You use the NO contact symbol in LAD to check for a signal state of "1".

General:

The "NC contact" symbol delivers the result of check "1" when the checked address state or status is "0".







**Exercise** Complete the programs above to obtain the following functionality: When switch S1 is activated and switch S2 is not activated, the light should be ON in all three cases.

**Note !** The terms "NO contact" and "NC contact" have different meanings depending on whether they are used in the process hardware context or as symbols in the software.

## Result of Logic Operation, First Check, and Examples

| | Example 1 | | | | Example 2 | | | | Example 3 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Signal State | Result of Check | Result of Logic Operation | First Check | Signal State | Result of Check | Result of Logic Operation | First Check | Signal State | Result of Check | Result of Logic Operation | First Check |
| :<br>:<br>= M 3.4 | | | | | | | | | | | | |
| A I 1.0 | 0 | | | | 1 | | | | 1 | | | |
| AN I 1.1 | 0 | | | | 1 | | | | 0 | | | |
| A M 4.0 | 0 | | | | 1 | | | | 1 | | | |
| = Q 8.0 | | | | | | | | | | | | |
| = Q 8.1 | | | | | | | | | | | | |
| A I 2.0 | 0 | | | | 1 | | | | 0 | | | |

SIMATIC® S7


Date: 12.03.03
File: PRO1_07E.7

SITRAIN Training for Automation and Drives

**Signal State** A logic operation is made up of a series of instructions to check the states of signals (inputs (**I**), outputs (**Q**), bit memories (**M**), timers (**T**), counters (**C**) or data bits (**D**) ) and instructions to set Q,M,T,C or D.

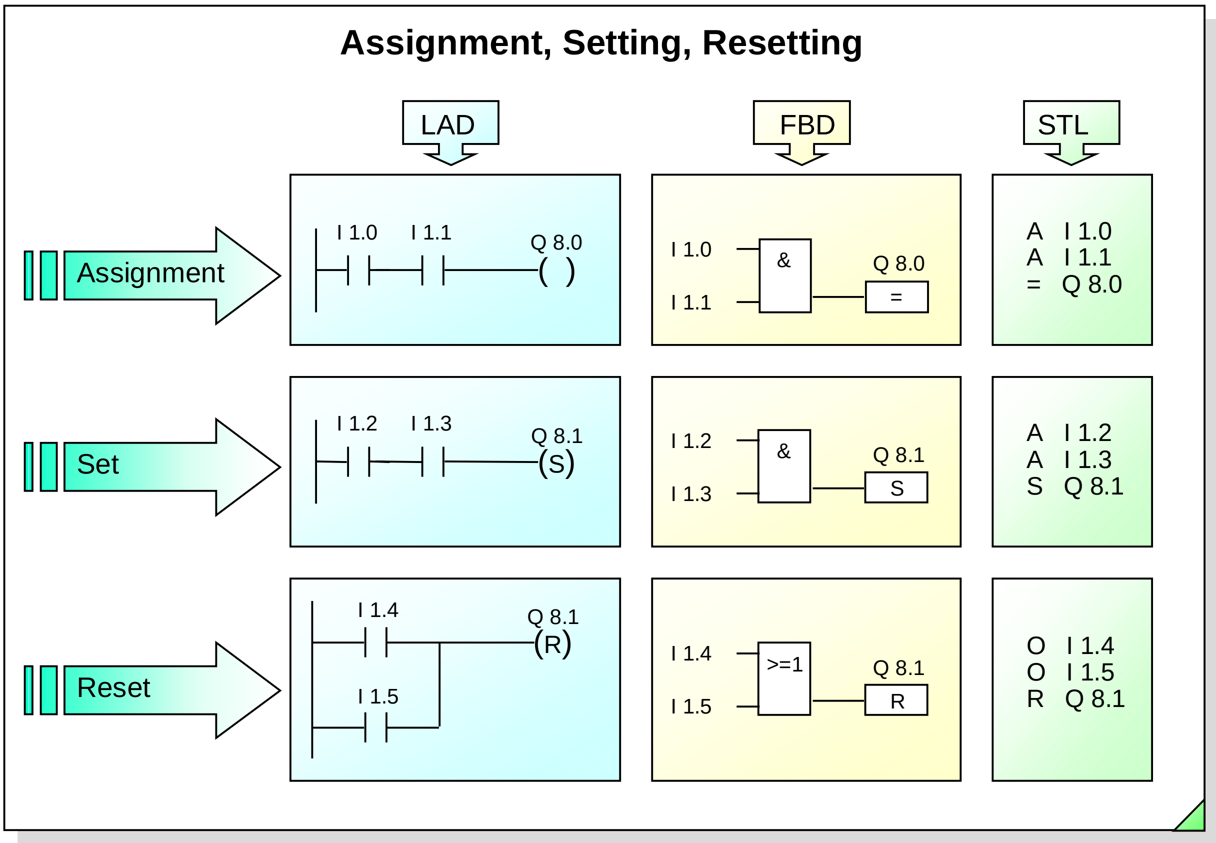
**Result of Check** When the program is executed, the result of check is obtained. If the check condition is fulfilled, the result of check is “1”. If the check condition is not fulfilled, the result of check is “0”.

**First Check** The first check that follows an RLO limiting operation (such as S, R, CU, = …) or the first check in a logic string is called a First Check (FC) since the result of this check - regardless of the last RLO - is accepted as the new RLO.

**Result of Logic Operation** When the next check instructions are executed, the result of logic operation is gated with the result of check and a new RLO is obtained.

When the last check instruction in a logic operation has been executed, the RLO remains the same. A number of instructions using the same RLO can follow.

**Note** The result of the first check is stored without being subjected to a logic operation. Therefore, it makes no difference whether you program the first check with an AND or an OR instruction in STL. To convert your program to one of the other programming languages, you should, however, always program using the correct instruction.




SIMATIC® S7

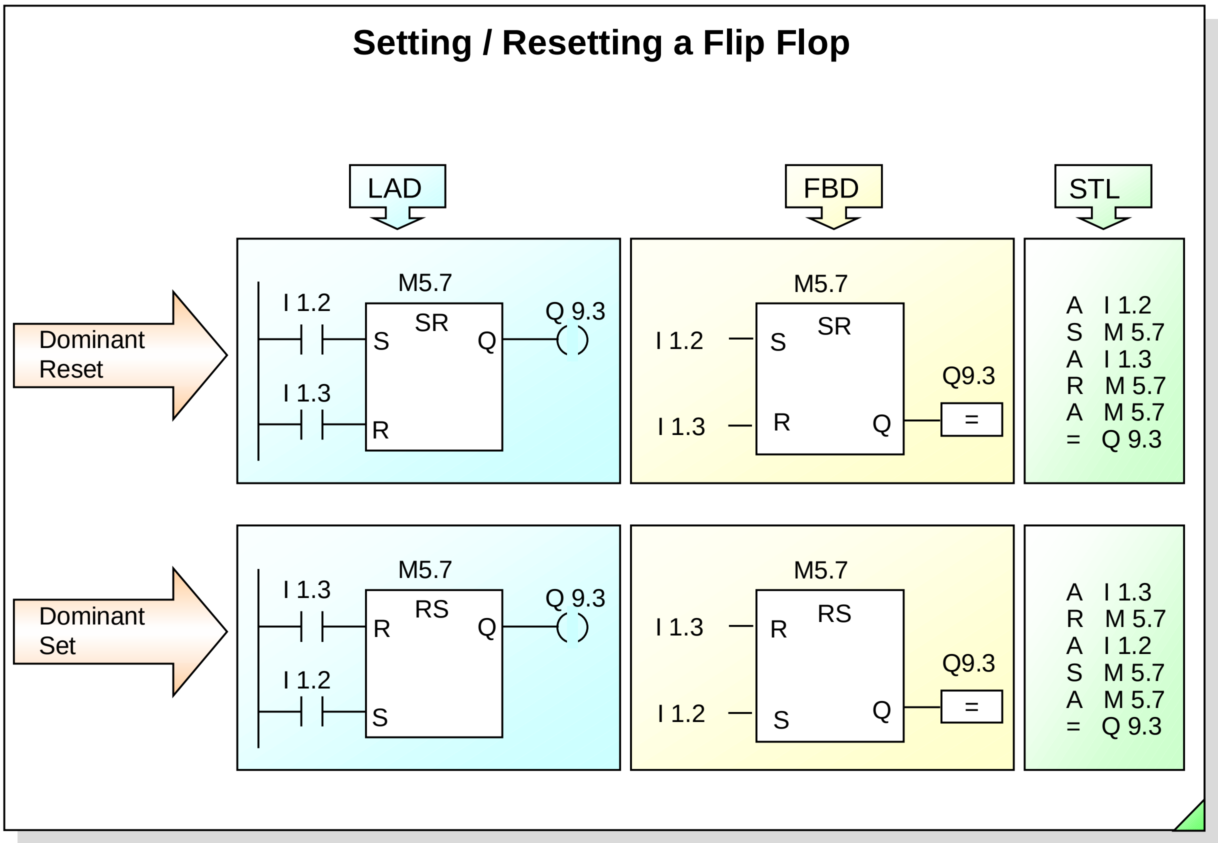

Date: 12.03.03
File: PRO1_07E.8

SITRAIN Training for Automation and Drives

**Assignment**
An assignment passes the RLO on to the specified address (Q, M, D). When the RLO changes, the signal state of that address also changes.

**Set**
If RLO= "1", the specified address is set to signal state "1" and remains set until another instruction resets the address.

**Reset**
If RLO= "1", the specified address is reset to signal state "0" and remains in this state until another instruction sets the address again.




SIMATIC® S7

Date: 12.03.03
File: PRO1_07E.9
SITRAIN Training for Automation and Drives

**Flip Flop**

A flip flop has a Set input and a Reset input. The memory bit is set or reset, depending on which input has an RLO=1.

If there is an RLO=1 at both inputs at the same time, the priority must be determined.

**Priority**

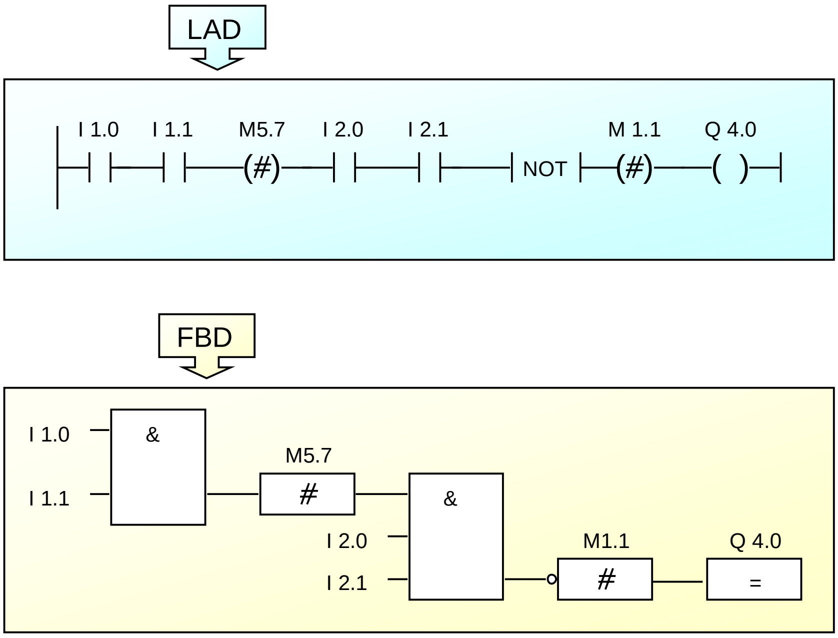
In LAD and FBD there are different symbols for Dominant Set and Dominant Reset memory functions.
In STL, the instruction that was programmed last has priority.

**Note**

If an output is set with a set instruction, the output is reset on a complete restart of the CPU.

If M 5.7 in the example above has been declared retentive, it will remain in the set state after a complete restart of the CPU, and the reset output Q 9.3 will be assigned the set state again.

## Midline Output Coil




STL

```
A    I  1.0
A    I  1.1
=    M  5.7
A    M  5.7
A    I  2.0
A    I  2.1
NOT
=    M  1.1
A    M  1.1
=    Q  4.0
```

SIMATIC® S7


Date: 12.03.03
File: PRO1_07E.10

SITRAIN Training for Automation and Drives

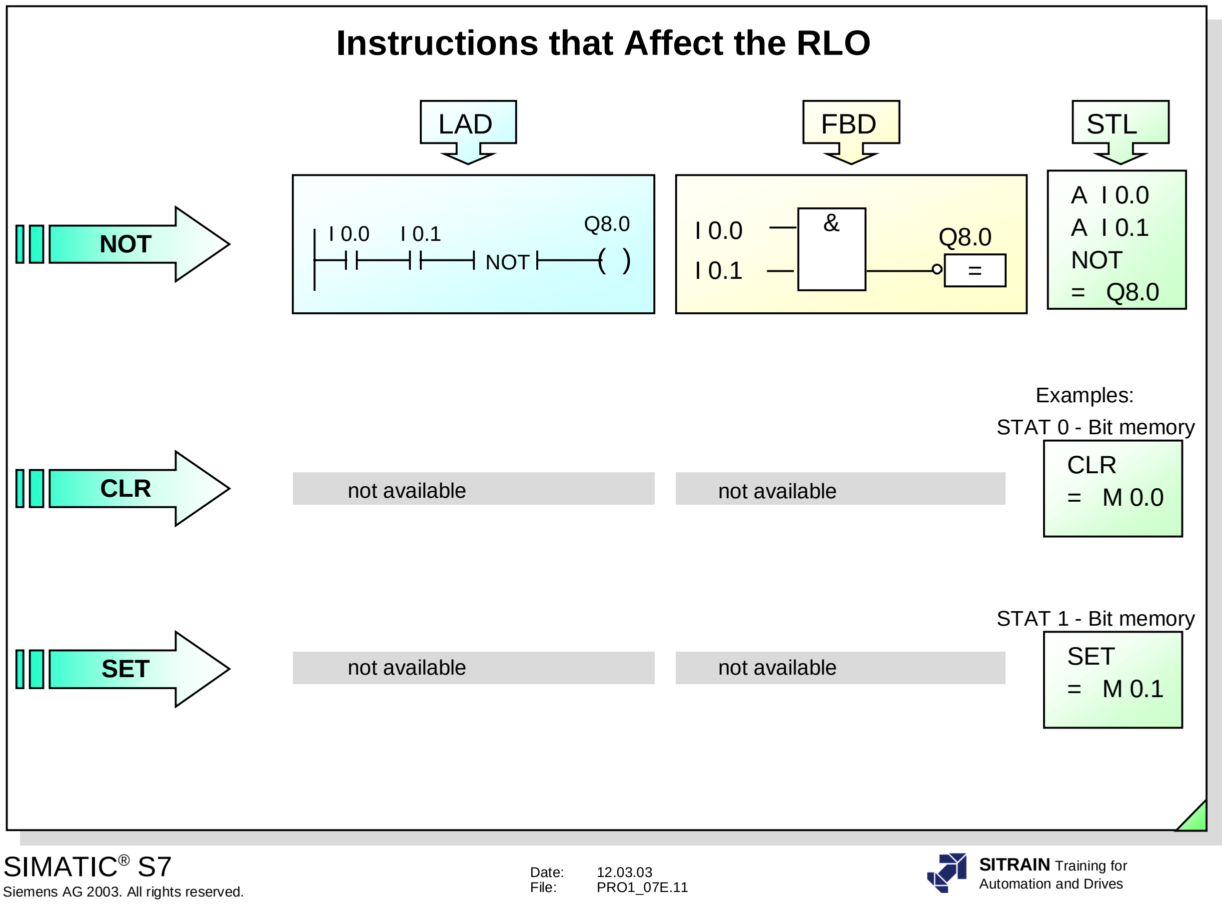
**Midline Output Coil** The midline output coil exists only in the LAD and FBD graphic languages. It is an intermediate assignment element with assignment function that assigns the current RLO at a specified address (M5.7 in the slide). The midline output coil provides this same address in the same network for subsequent gating.

In the STL language, this is equivalent to

= M 5.7
A M 5.7

In the LAD language, when connected in series with other elements, the "midline output coil" instruction is inserted in the same way as a contact.

SIEMENS

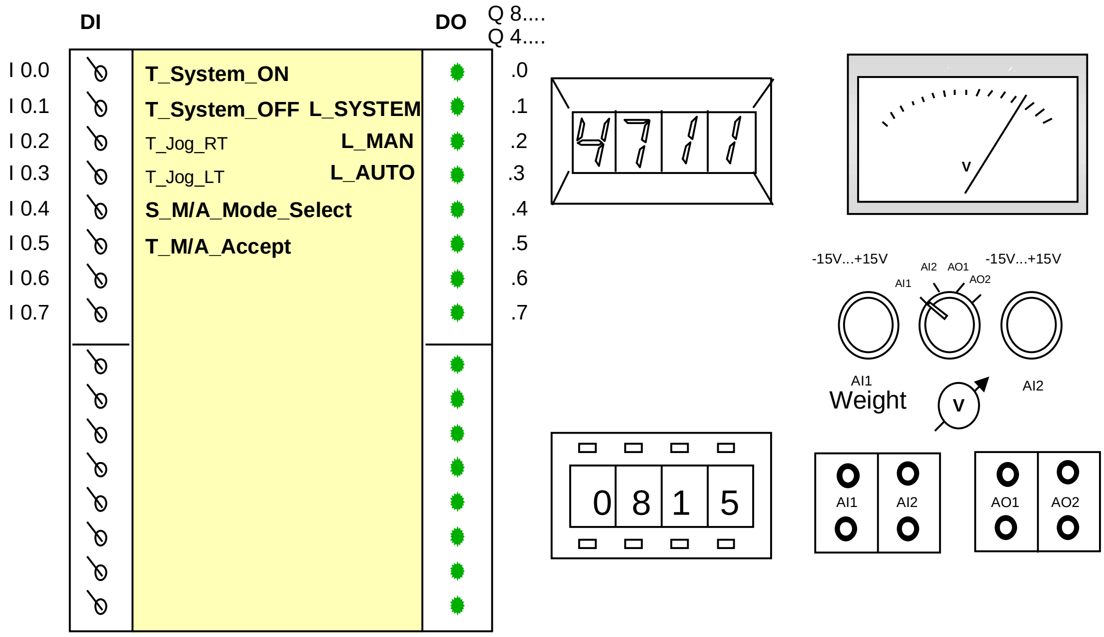



**NOT** The NOT instruction inverts the RLO.

**CLR** The CLEAR instruction sets the RLO to "0" without pre-conditions (available only in STL at present !).
The CLR instruction completes the RLO, thus the next scan becomes a first check.

**SET** The SET instruction sets the RLO to "1" without pre-conditions (available only in STL at present !).
The SET instruction completes the RLO, thus the next scan becomes a first check.

## Exercise: Mode Section of the Distribution Conveyor






**Task**

You are to program a "mode" section in FC 15 for the distribution conveyor and integrate the message MANUAL mode (Q 8.2 or Q 4.2) as a lock-out in the FC 16 block.

**Function of the mode section in the FC 15:**

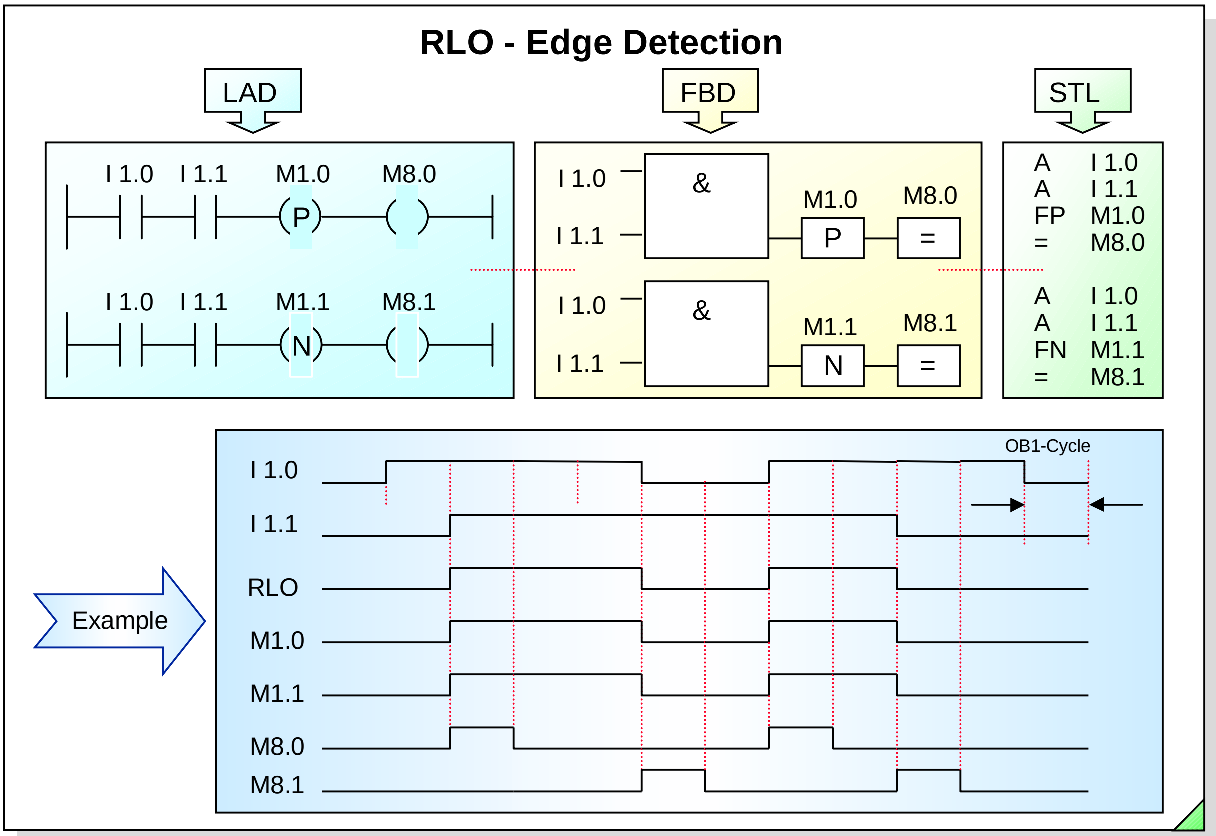
- Start with I 0.1 in the closed position to simulate a NC stop switch.
- The system "L_System" (LED Q8.1 or Q4.1) is turned "on" using I 0.0, the simulator momentary contact switch (T_System_ON). It is turned "off" using I 0.1(NC contact) (T_System_OFF), the simulator momentary contact switch
- You can preselect "MANUAL" mode (LED Q 8.2 or Q 4.2) or "AUTO" mode (LED Q 8.3 or Q 4.3) through switch I 0.4 (S_M/A_ModeSelect)as follows:
  - I 0.4 switched off (= ´0´): "MANUAL" mode preselected,
  - I 0.4 switched on (= ´1´): "AUTO" mode preselected.
- The operating mode that you preselected through switch I 0.4 has to be acknowledged through momentary contact switch I 0.5 (T_M/A_Accept).
- The operating modes are switched off when you change the preselection of the operating mode (I 0.4) or when the system is switched off (Q 8.1 or Q 4.1 = ´0´).

**Integrating the MANUAL mode (Q8.2 or Q 4.2):**

- The "Jog Conveyor Motor" programmed in FC 16 is now only to be possible when the "MANUAL" mode is switched on. Program the relevant lock-out in FC 16.

**What To Do**

- Insert the new block FC 15 in the S7 Program "My_Program" and program it according to the task.
- Program the call of the FC 15 block in OB1.
- Program the required lock-out in FC 16.
- Download all blocks into the CPU and test your program







**RLO Edge Detection** An "RLO edge" detection is when the result of a logic operation changes from "0" to "1" or from "1" to "0".

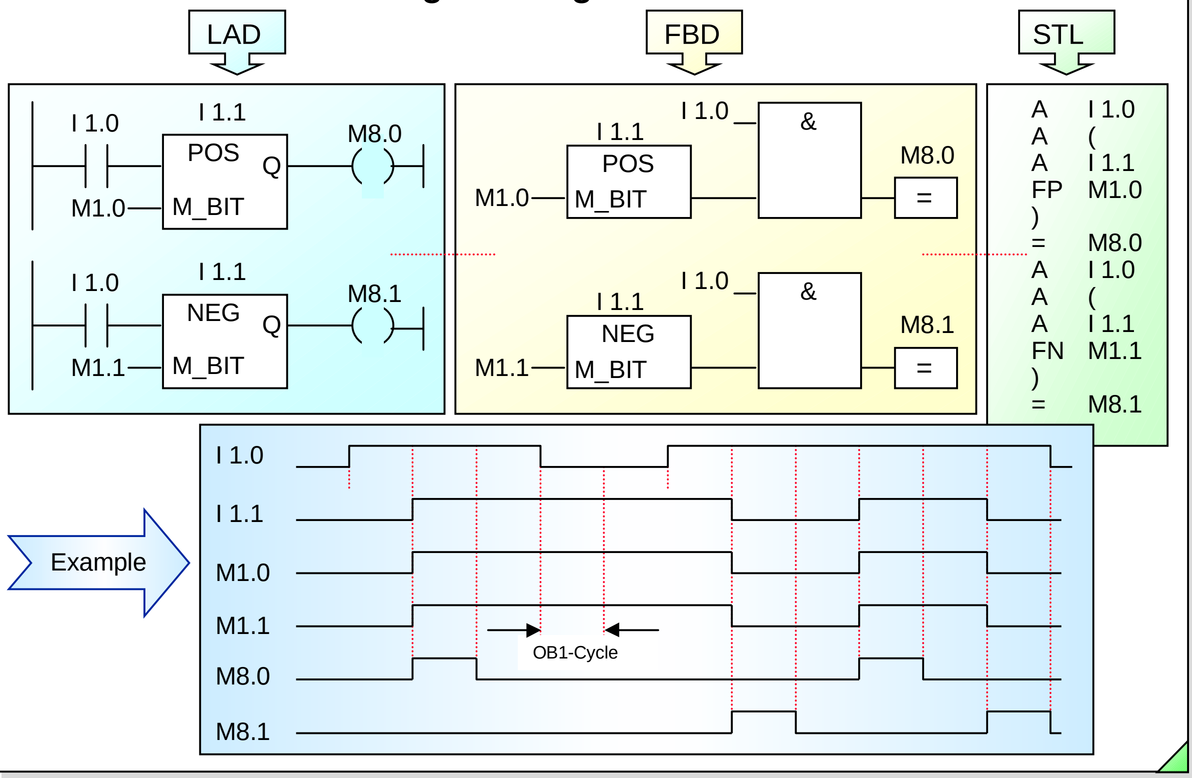
**Positive Edge** (Positive RLO Edge Detection) detects a signal change in the address (M1.0) from "0" to "1", and displays it as RLO = "1" after the instruction (such as at M 8.0) for one cycle.

To enable the system to detect the edge change, the RLO must be saved in an FP bit memory (such as M 1.0), or a data bit.

**Negative Edge** (Negative RLO Edge Detection) detects a signal change in the address (M1.1) from "1" to "0" and displays it as RLO = "1" after the instruction (such as at M 8.1) for one cycle.

To enable the system to detect the edge change, the RLO must be saved in an FN bit memory (such as M 1.1), or a data bit.

## Signal - Edge Detection




SIMATIC® S7

Date: 12.03.03
File: PRO1_07E.14
SITRAIN Training for Automation and Drives

**Signal Edge** A "signal edge" is when a signal changes its state.

**Example** Input I 1.0 acts as a static enable. Input I 1.1 is to be monitored dynamically and every signal change is to be detected.

**Positive Edge** When the signal state at I 1.1 changes from "0" to "1", the "POS" check instruction results in signal state "1" at output Q for one cycle, provided input I 1.0 also has signal state "1" (as in the example above).

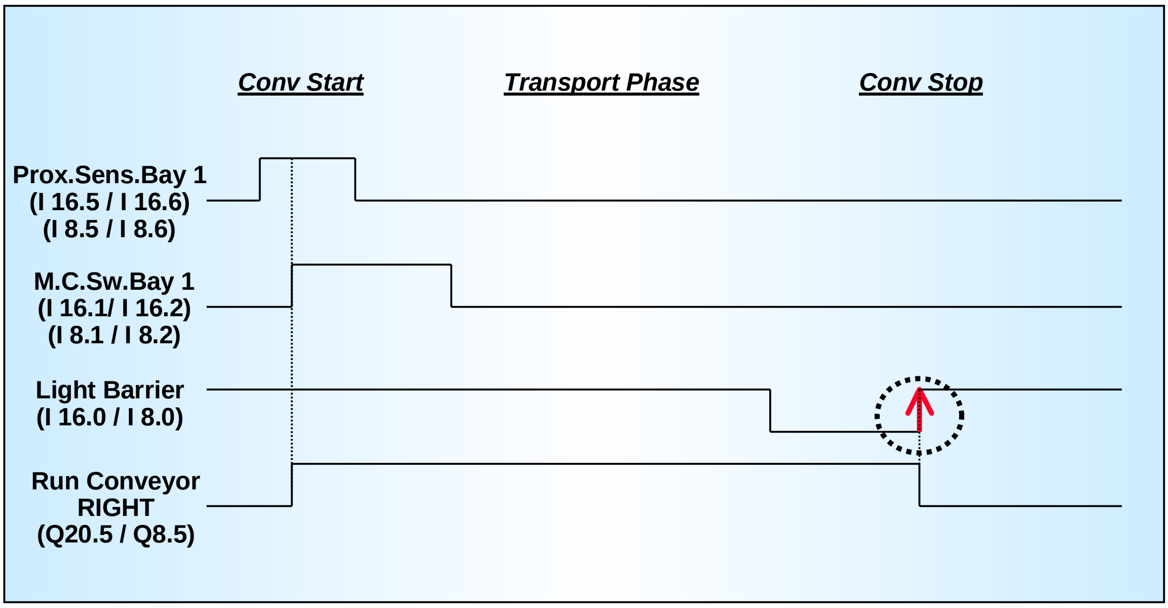
To enable the system to detect the edge change, the signal state of I 1.1 must also be saved in an M_BIT (bit memory or data bit) (such as M 1.0).

**Negative Edge** When the signal state at I 1.1 changes from "1" to "0", the "NEG" check instruction results in signal state "1" at output Q for one cycle, provided input I 1.0 has signal state "1" (as in the example above).

To enable the system to detect the edge change, the signal state of I 1.1 must also be saved in an M_BIT (bit memory or data bit) (such as M 1.1).

SIEMENS

## Exercise: Conveyor Operation in AUTO Mode




SIMATIC® S7


Date: 12.03.03
File: PRO1_07E.15

SITRAIN Training for Automation and Drives

**FC 16 Up Till Now:** When the MANual mode is switched on (Q 8.2 / Q 4.2 = ´1´), you can jog the conveyor motor to the RIGHT or to the LEFT using the simulator momentary contact pushbutton.

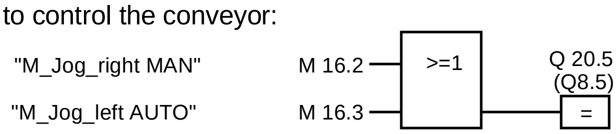
**Task:** You are to expand the function of FC 16 to include controlling of the conveyor motor as follows (also see the function diagram in the slide):

- With AUTO mode switched on (Q 8.3 / Q 4.3 = ´1´), the conveyor motor starts to the RIGHT (Run Conveyor RIGHT), as soon as a part has been placed in front of Bay 1 or Bay 2 and the associated momentary contact pushbutton has been pressed.
- The conveyor motor stops as soon as the part reaches the Conveyor's End or has **passed through** the light barrier (-> edge detection necessary, see slide) or the AUTO mode is switched off.

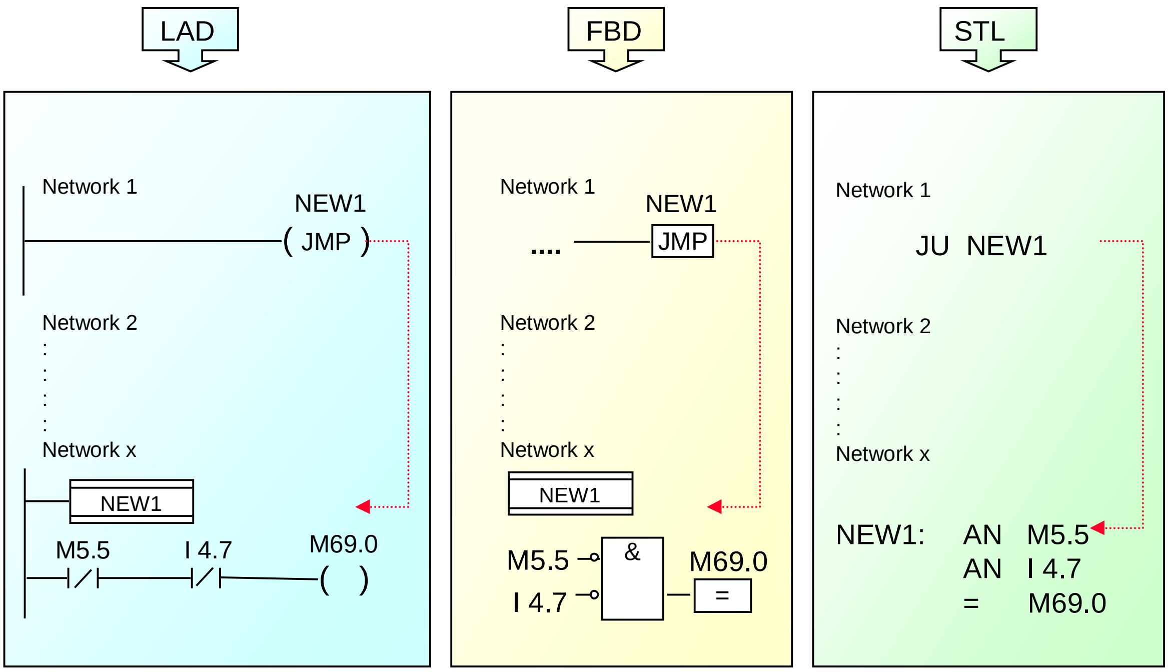
**What To Do:**

1. Program the AUTO conveyor operation in FC 16. The jogging of the conveyor motor when the MANUAL mode is switched on is already programmed.
2. Download the modified FC 16 block into the CPU
3. Check whether you program fulfills the desired function!

**Solution Hints** The Run Conveyor RIGHT (Q 20.5 / Q 8.5) must be controlled under two conditions: in the MANUAL mode while jogging RIGHT OR in the AUTO mode. Program a memory bit for each of the two conditions or transfer the results of the logic operations into bit memories so that you can then use these in a new network to control the conveyor:



## Unconditional Jump (Independent of RLO)






Date: 12.03.03
File: PRO1_07E.16

SITRAIN Training for Automation and Drives

**Jump Instruction**

In LAD/FBD, the label (NEW1) is entered as an identifier above the coil symbol or assignment symbol. In STL it comes after the Jump (JU) instruction.

The label can have up to four characters, the first of which must be a letter or the "_" character.

The label marks the point where execution of the program is to continue. Any instructions or networks between the jump instruction and the label are not executed.

Jumps can be made both forwards and backwards.
The jump instruction and the jump destination must both be in the same block (max. jump length = 64kbyte). The label's name can only be used once in a block.

Jump instructions can be used in FBs, FCs and OBs.

**Inserting a Label**

In LAD and FBD, you use the Program Elements browser to insert a label: *Program Elements -> Jumps -> LABEL.*

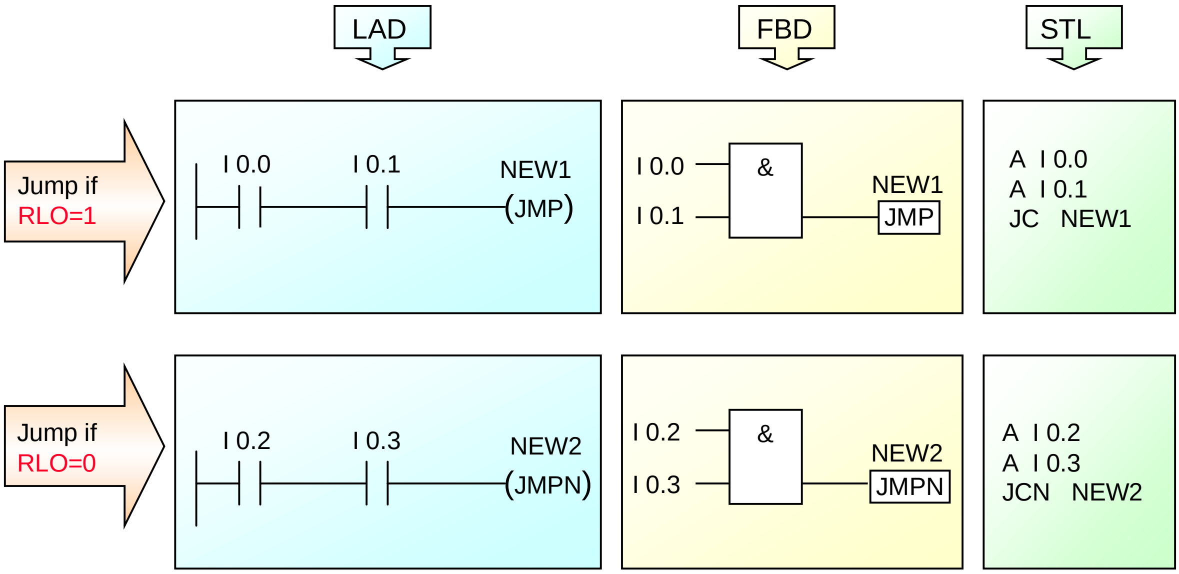
**Jump Label**

The label may be as many as four characters of which the first character must be a letter. Jump labels are followed with a mandatory colon ":" and must precede the program statement in a line.
Example: NEXT: A I 0.0

**JMP**

An unconditional jump instruction causes a program jump to a label **regardless of the RLO**.

## Conditional Jump (Dependent on RLO)




SIMATIC® S7

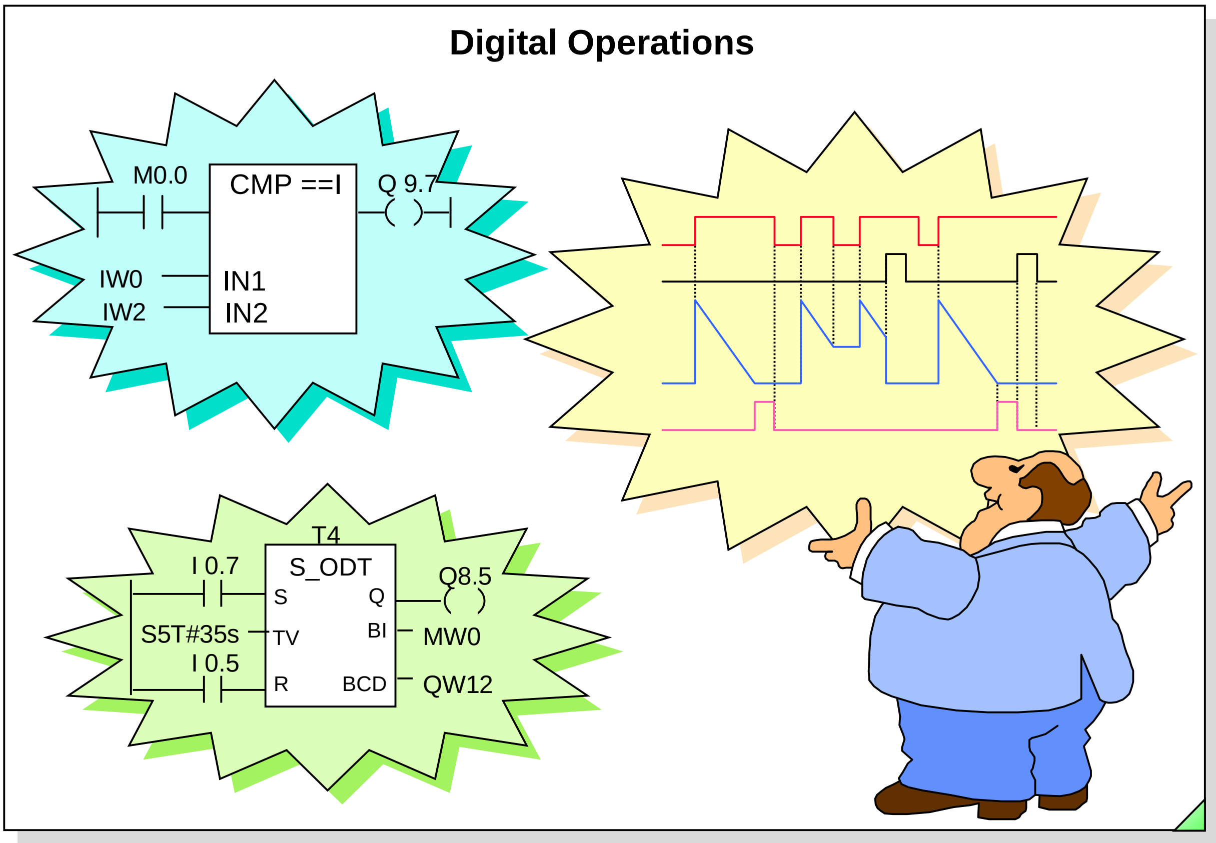

Date: 12.03.03
File: PRO1_07E.17

SITRAIN Training for Automation and Drives

**JC** The "JC" conditional jump is only executed if the RLO is "1". If the RLO is "0", the jump is not executed, the RLO is set to "1" and program execution continues with the next instruction.

**JCN** The "JCN" conditional jump is only executed if the RLO is "0". If the RLO is "1", the jump is not executed and program execution continues with the next instruction.

**Note** STL provides additional jump operations, which are discussed in another programming course.




SIMATIC® S7


Date: 12.03.03
File: PRO1_08E.1

**SITRAIN** Training for Automation and Drives

## Contents

Page

# Objectives

**Upon completion of this chapter the participant will ...**

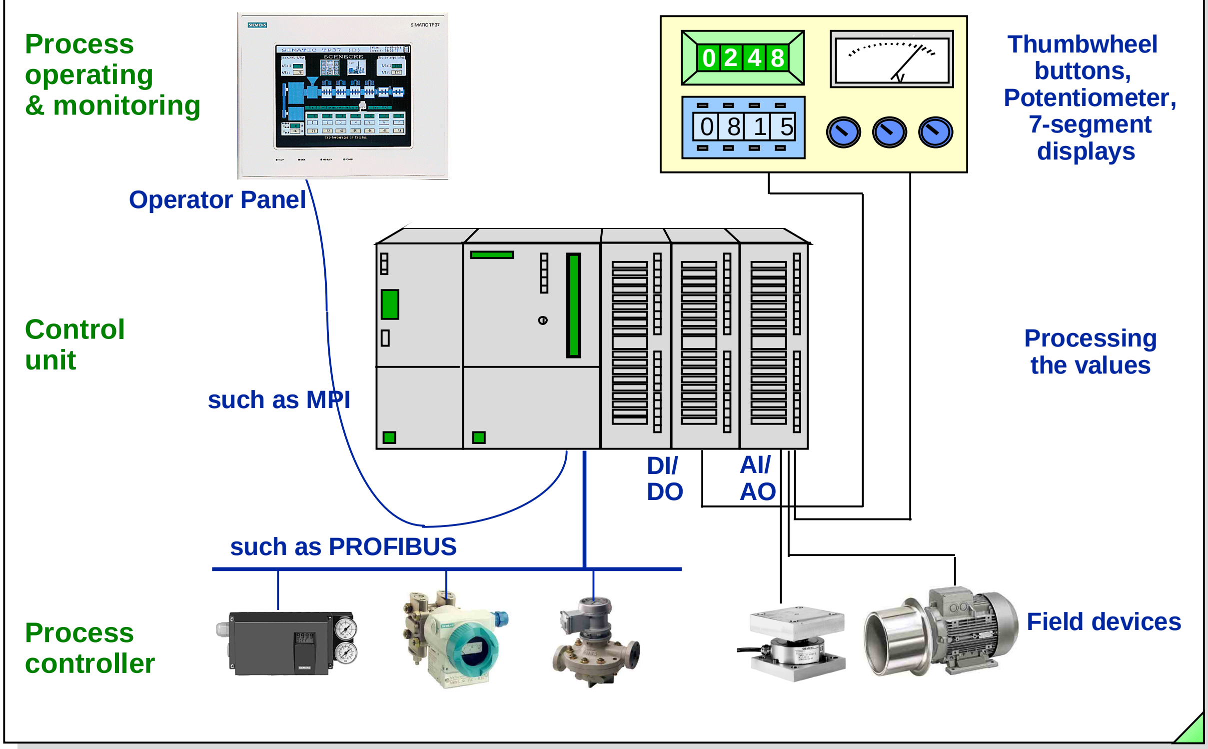
- ... be familiar with the INT, DINT, REAL data types and the BCD display
- ... be able to apply the selectable display formats in the "Monitor / Modify Variable" test function
- ... understand the "Load" and "Transfer" instructions
- ... be able to apply and program S5 counter functions for problem solving
- ... be able to apply and program S5 timer functions for problem solving
- ... be able to apply and program the conversion operations INT <-> BCD for problem solving
- ... be able to apply and program comparison operations for problem solving
- ... be able to apply and program basic mathematical functions for problem solving

SIMATIC® S7


Date: 12.03.03
File: PRO1_08E.2

**SITRAIN** Training for Automation and Drives

## Acquisition, Processing and Outputting Data




SIMATIC® S7


Date: 12.03.03
File: PRO1_08E.3

SITRAIN Training for Automation and Drives

**Binary/Digital Processing**

True logic control systems are recognizable in the fact that they exclusively process binary data.

The performance of today's control computer, as well as tasks in the data processing, quality control areas, among others, has increased the importance of digital data processing using PLCs.

Digital process variables can be found in all areas of open-loop control - such as in connected devices for process operating and monitoring or in the control of field devices.

**Operating and Monitoring**

The goal of process monitoring is to provide the operator with up-to-the-minute information about the working machine or system quickly, concisely and clearly as well as the opportunity to intervene and control and influence the process.

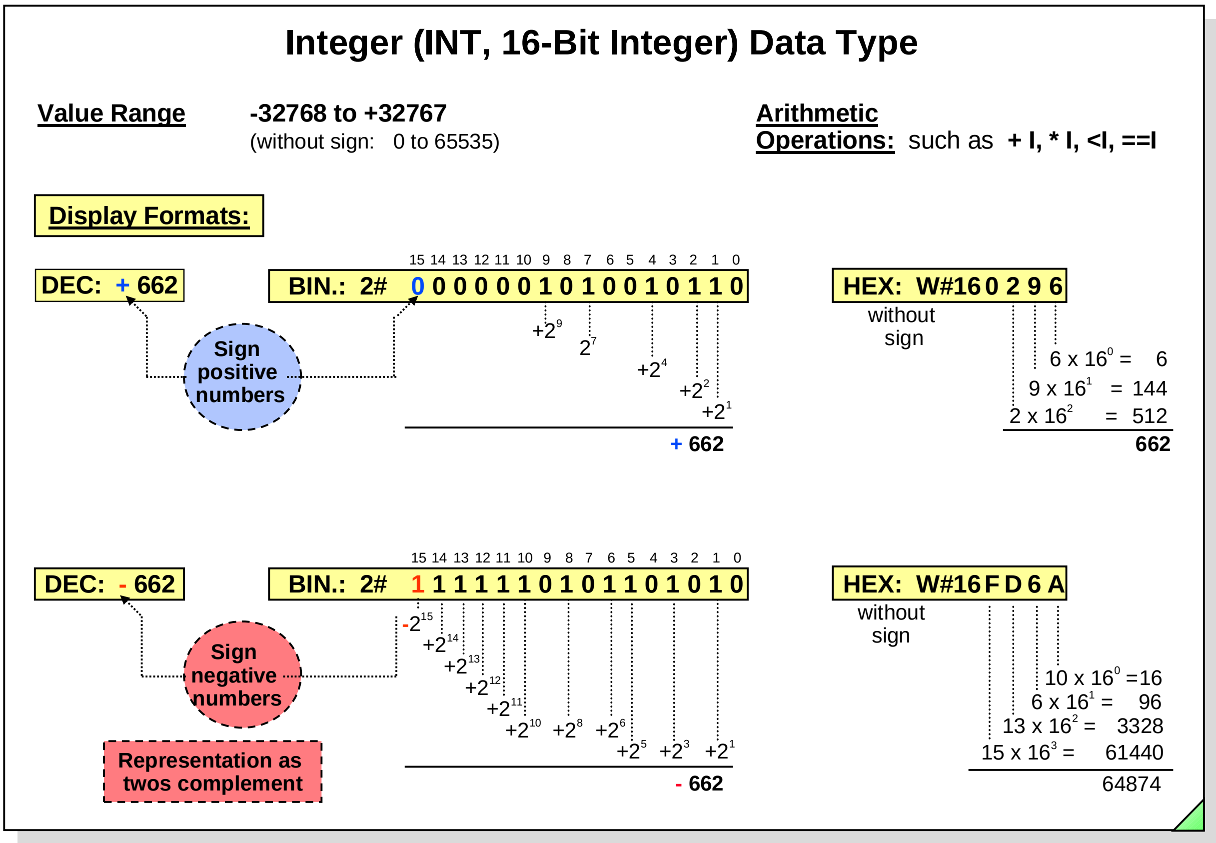
While in the past mostly simple, that is, "dumb" input and output devices, such as 7-segment displays and thumbwheel buttons were used to display and enter digital values, today "intelligent" operating and monitoring devices are frequently connected to a PLC.

**Field Devices**

Today as well, field devices that acquire process data or that control the process are supplied directly with digital variables through field bus systems. The connection of field devices, such as drives or weighing systems, using analog input and output modules is becoming more and more a thing of the past.

**Formats**

Depending on the type of device connected, different number formats for the coding of data are used to transmit data between device and PLC, as well as for storing and processing data in the PLC.




SIMATIC® S7



Date: 12.03.03
File: PRO1_08E.4

SITRAIN Training for Automation and Drives

**Integer Data Type** (16-Bit Integer)

An *Integer* data type value is a whole number value, that is, a value without a decimal point.

SIMATIC® S7 stores *Integer* data type values with sign in 16 bit code. This results in the value range shown in the slide above. As well, SIMATIC® S7 provides arithmetic operations for processing Integer values.

**Decimal**

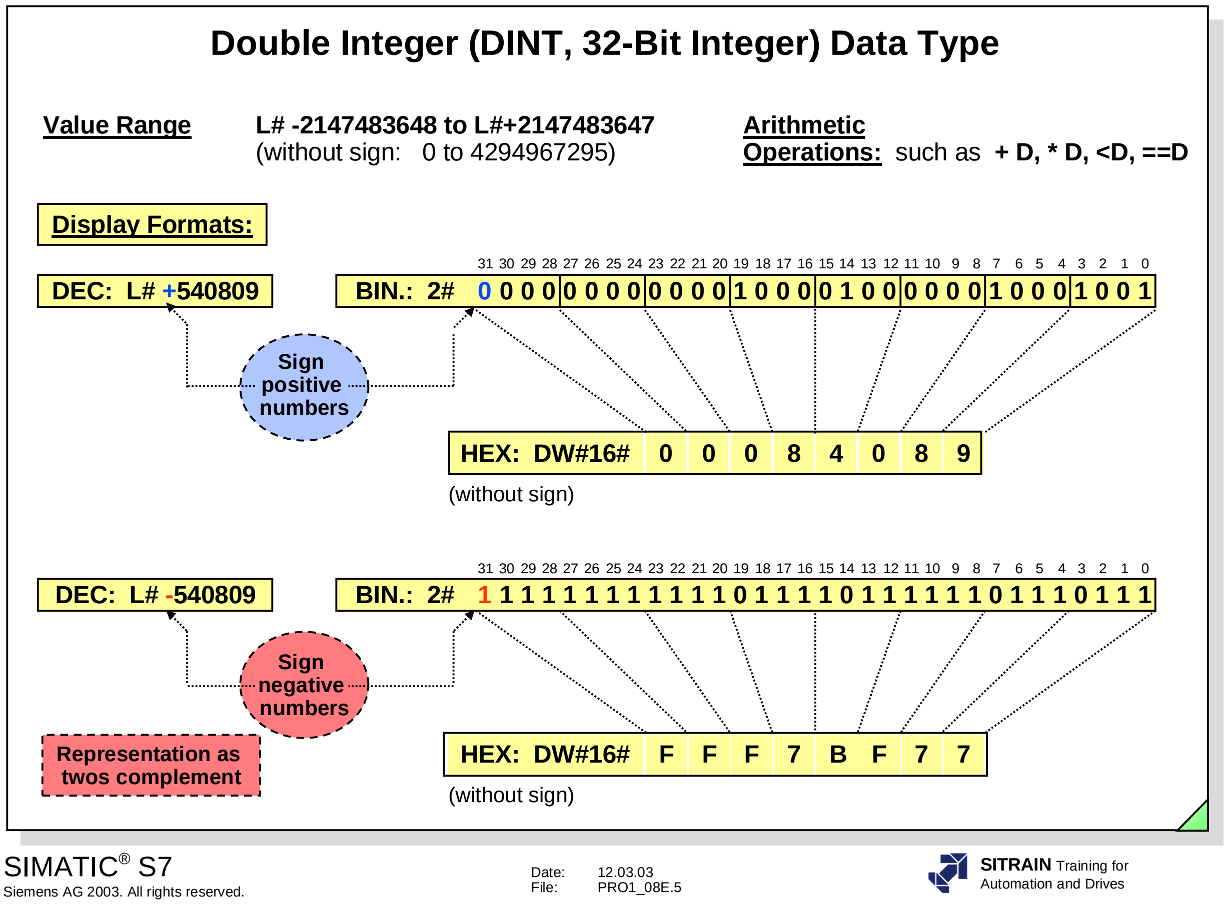
STEP7 uses the *Decimal* (not BCD!) display format to specify the constants of the *Integer* data type. That is, with sign and without explicit format description. The use of constant Integer values in the *Binary* and *Hexadecimal* display formats is possible in principle, but because of the poor legibility, they are more or less not suitable. For this reason, the syntax of STEP7 provides the specification of Integer values only in the decimal display format.

**Binary**

In a digital computer system, all values are stored in a binary-coded form. Only the digits 0 and 1 are available in the binary number system. Base 2 of this numbers system results from the number of available digits. Accordingly, the value of every position of a binary number results from a power of Base 2. This is also expressed in the format specification **2#**.... .

Negative values are represented as binary numbers in twos complement. In this representation, the most significant bit (bit no. 15 for the Integer data type) has the value - $2^{15}$. Since this value is greater than the sum of all residual values, this bit also has the sign information. That is, if this bit = 0, then the value is positive; if the bit is = 1, then the value is negative. The conversion of a binary number into a decimal number is made by adding the values of the positions that have a 1 (see slide).

Specifying constants in the binary display format is not only used for specifying Integer values, but more often to specify bit patterns (such as in digital logic operations) in which the Integer value represented by the bit pattern is of no interest. The number of specifyable bits is variable from 1 to 32. Missing bits are filled with leading zero digits.




**Double Integer**
(32-Bit Integer)

SIMATIC® S7 stores *Double Integer* data type values with sign as 32 bit code. This results in the value range shown in the slide above. As well, SIMATIC® S7 provides arithmetic operations for processing DINT values.

**Decimal**

STEP7 uses a decimal number (not BCD!) to specify a constant of the *Double Integer* data type. That is, with sign and the format **L#** for "long" (double word, 32 bit).

When a value smaller than -32768 or greater than 32767 is specified, the format **L#** is automatically added. For negative numbers smaller than -32768, the user must specify the format as **L#** - (for example: L# -32769). This is imperative if the value is to be further processed arithmetically as a double integer. Otherwise you would work with false values (value + sign!)!

**Hexadecimal**

The hexadecimal numbers system provides 16 different digits (0 to 9 and A to F). This results in Base 16 of this numbers system. Accordingly, the value of every position of a hexadecimal number results from a power of Base 16.

Hexadecimal numbers are specified with the format **W#** for the dimension (W = word = 16 bit) or **DW#** (DW = double word = 32 bit) and **16#** for identifying the basic numbering system. The number of specifyable bits is variable from 1 to 8. Missing bits are filled with leading zero digits.

The digits A to F correspond to the decimal values 10 to 15. The value 15 is the last value that can be binary-coded - without sign - with 4 bits. This connection results in the simple conversion of a binary number in a hexadecimal number and vice versa. Four binary bits make up one digit of a hexadecimal number.

Constants in the hexadecimal format are therefore not used for specifying integer values. They are used instead of binary numbers for specifying bit patterns in which the integer value represented by the bit pattern is of no interest.

SIEMENS

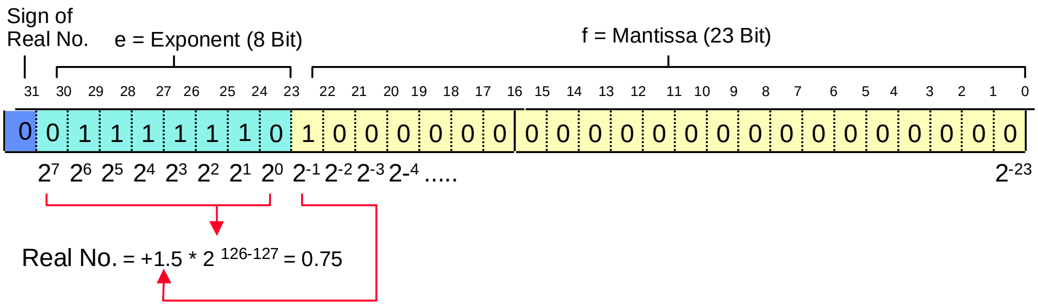
# REAL (Floating-point Number, 32 Bit) Data Type

**Value Range** $-1.175495 \cdot 10^{-38}$ to $3.402823 \cdot 10^{+38}$

**Arithmetic 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**: 7.50000e-001 (7.5 * $10^{-1}$ = 0.75)




SIMATIC® S7


Date: 12.03.03
File: PRO1_08E.6

SITRAIN Training for Automation and Drives

**Real**

The previously described INT and DINT data types are used to store whole number values with sign. Accordingly, only operations that supply a whole number value as the result can be performed with these data types.

In cases where analog process variables such as voltage, current, and temperature have to be processed, it becomes necessary to use *Real* values (real numbers, "decimal numbers"). In order to be able to represent such values, binary digits have to be defined whose value is less than 1 (power of base 2 with negative exponent).

**Real Format**

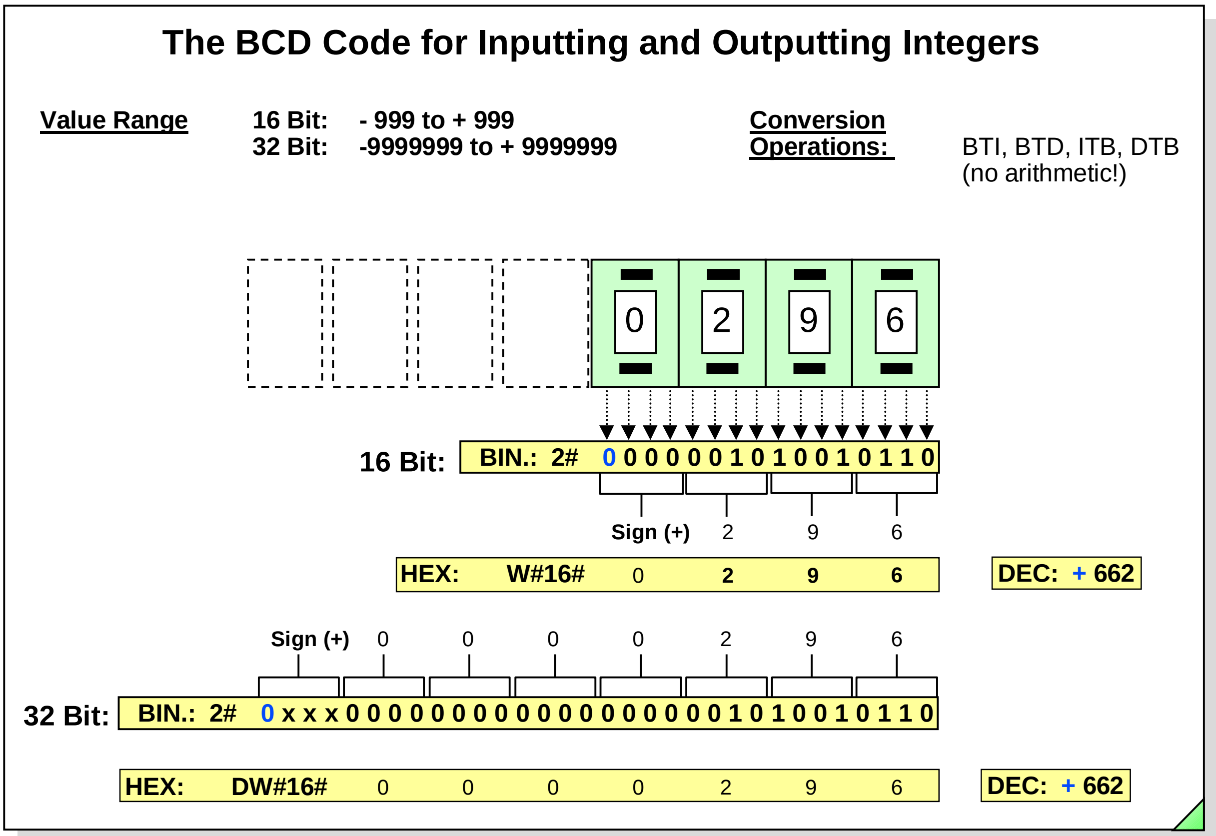
In order to be able to form the greatest possible value range within a defined memory capacity (for SIMATIC® S7: double word, 32 bit) (see slide), you must be able to select the decimal point position. Early on, IEEE defined a format for floating-point numbers. This format was laid down in IEC 61131 and was included in STEP 7. This format makes it easy to process a variable decimal point position.

In a binary coded floating-point number, a portion of the binary digits contain the mantissa and the rest contain the exponent and the sign of the floating-point number.

When you specify real values, you do so without specifying the format. After you enter a constant real value (for example: 0.75), the Editor automatically makes a conversion (for example: 7.5000e-001).

**Application**

Floating-point numbers are used for "analog value processing", among others. A great advantage of floating-point numbers is in the number of operations possible with such numbers. These include, in addition to the standard operations such as: +, -, * , / also instructions such as sin, cos, exp, ln, etc, that are used mainly in closed-loop control algorithms.




SIMATIC® S7


Date: 12.03.03
File: PRO1_08E.7

SITRAIN Training for Automation and Drives

**Origin**

In the past, the specification and visualization of whole numbers was done exclusively using simple, mechanical thumbwheel buttons and digital displays. These thumbwheel buttons and digital displays were connected to the PLC's digital input and output modules through parallel wiring. The structure could also be cascaded, without having to change the mechanical coding of a digit.

**BCD Code**

Each digit of a decimal number is encoded in four bit positions. Four bits are used because the highest decimal digit, 9, requires at least four bit positions in binary code.

| Decimal No. | BCD Code | Decimal No. | BCD Code |
|---|---|---|---|
| 0 | 0000 | 6 | 0110 |
| 1 | 0001 | 7 | 0111 |
| 2 | 0010 | 8 | 1000 |
| 3 | 0011 | 9 | 1001 |
| 4 | 0100 | 10 ... 15 | not allowed |
| 5 | 0101 | | |

**Negative Numbers**

So that negative numbers can also be specified using a BCD thumbwheel button, STEP 7 codes the sign in the most significant bit of the most significant digit (see slide). A sign bit = 0 indicates a positvie number. A sign bit = 1 indicates a negative number.

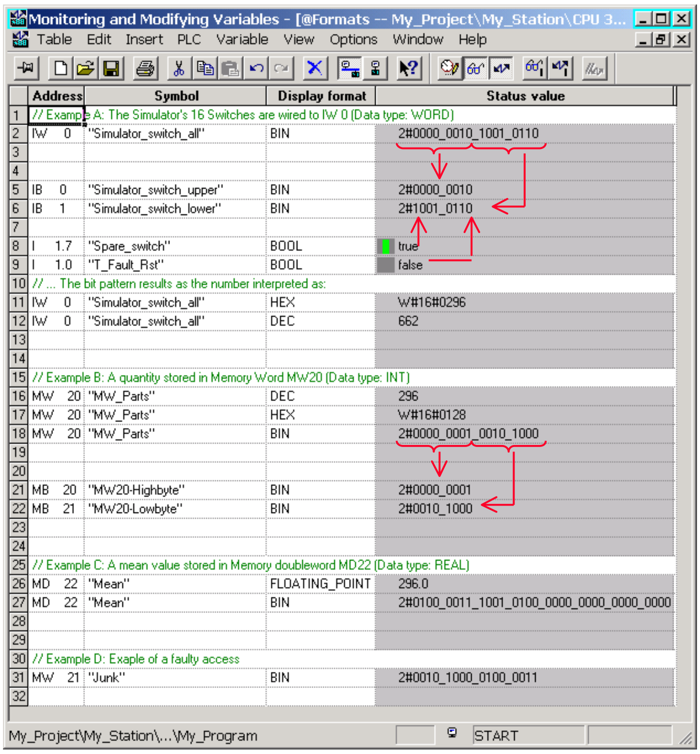
STEP 7 recognizes 16-bit-coded (sign + 3 digits) and 32-bit-coded (sign + 7 digits) BCD numbers.

**Data Formats**

There is no data format for specifying BCD-coded values in STEP 7. You can, however, specify the decimal number whose BCD code is to be given, as a HEX number. The binary code of the HEX number and that of the BCD-coded decimal number is identical.

As you can see in the slide, the DEC data format is not suitable for specifying BCD coded numbers!

# "Monitor / Modify Variables": Display Formats




| | Address | Symbol | Display format | Status value |
|---|---|---|---|---|
| 1 | // Example A: The Simulator's 16 Switches are wired to IW 0 (Data type: WORD) | | | |
| 2 | IW 0 | "Simulator_switch_all" | BIN | 2#0000_0010_1001_0110 |
| 3 | | | | |
| 4 | | | | |
| 5 | IB 0 | "Simulator_switch_upper" | BIN | 2#0000_0010 |
| 6 | IB 1 | "Simulator_switch_lower" | BIN | 2#1001_0110 |
| 7 | | | | |
| 8 | I 1.7 | "Spare_switch" | BOOL | true |
| 9 | I 1.0 | "T_Fault_Rst" | BOOL | false |
| 10 | // ... The bit pattern results as the number interpreted as: | | | |
| 11 | IW 0 | "Simulator_switch_all" | HEX | W#16#0296 |
| 12 | IW 0 | "Simulator_switch_all" | DEC | 662 |
| 13 | | | | |
| 14 | | | | |
| 15 | // Example B: A quantity stored in Memory Word MW20 (Data type: INT) | | | |
| 16 | MW 20 | "MW_Parts" | DEC | 296 |
| 17 | MW 20 | "MW_Parts" | HEX | W#16#0128 |
| 18 | MW 20 | "MW_Parts" | BIN | 2#0000_0001_0010_1000 |
| 19 | | | | |
| 20 | | | | |
| 21 | MB 20 | "MW20-Highbyte" | BIN | 2#0000_0001 |
| 22 | MB 21 | "MW20-Lowbyte" | BIN | 2#0010_1000 |
| 23 | | | | |
| 24 | | | | |
| 25 | // Example C: A mean value stored in Memory doubleword MD22 (Data type: REAL) | | | |
| 26 | MD 22 | "Mean" | FLOATING_POINT | 296.0 |
| 27 | MD 22 | "Mean" | BIN | 2#0100_0011_1001_0100_0000_0000_0000_0000 |
| 28 | | | | |
| 29 | | | | |
| 30 | // Example D: Exaple of a faulty access | | | |
| 31 | MW 21 | "Junk" | BIN | 2#0010_1000_0100_0011 |
| 32 | | | | |

SIMATIC® S7


Date: 12.03.03
File: PRO1_08E.8

SITRAIN Training for Automation and Drives

**Display Formats**

Different display formats can be selected in both the "Monitor / Modify Variables" and the "Monitor (Block)" test function when displaying variables or register contents in STL.

Every variable can be monitored with several display format options. Depending on the variable's data type, it becomes apparent that monitoring with the appropriate display format makes more sense.

| | |
|---|---|
| **BOOL:** | Display of a single bit (only possible for a variable of the BOOL data type) |
| **BIN**: | Display of the individual bits of a variable (makes sense for variables of the BYTE, WORD, DWORD data types) |
| **HEX**: | Display the contents of a variable as hexadecimal number (BCD) (makes sense for variables of the BYTE, WORD, DWORD data types) |
| **DEC**: | Display the contents of a variable as decimal number (not BCD!) with sign (makes sense for variables of the INT, DINT data types) |
| **FLOATING_ POINT** | Display of the contents of a variable as floating-point number (makes sense for variables of the REAL data type) |

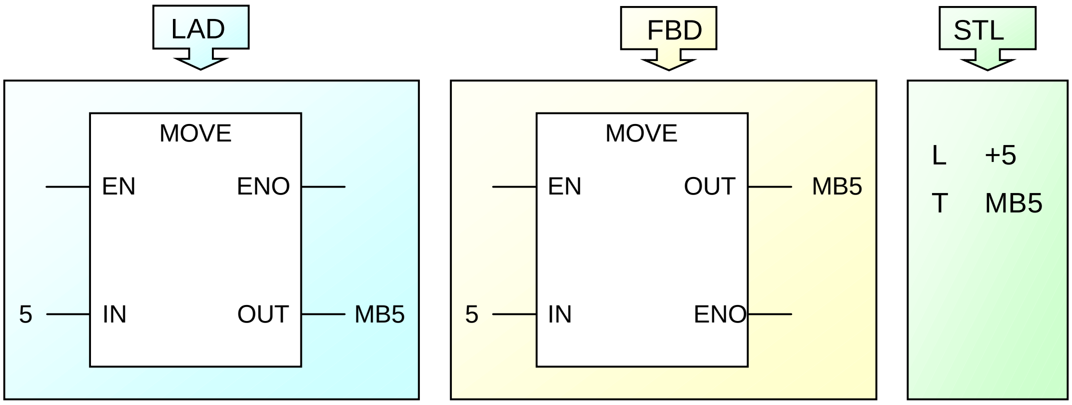
**Addressing**

The memory of S7 controllers is byte-oriented. As a result, the memory word MW 20 contains the memory bytes MB 20 (highbyte) and MB 21 (lowbyte). The memory double word MD 22 contains the memory bytes MB 22, 23, 24 and 25 (see examples in the slide).

When there is an absolute access of variables (such as with "L MW 20"), you must make sure that the dimension of the access (MB..., MW... or MD...) as well as the address (is always the address of the high byte) is correct. If you make an unintentional access "in between", an invalid value will be loaded!

The example in the slide shows that when the MW 21 was loaded, part of the variable "MW_Parts" (MW 20) and the variable "Mean" (MD 22) was loaded. Such errors can be avoided with a symbolic addressing of variables.

## Loading and Transferring Data (1)




Examples of Load

| | | | |
|---|---|---|---|
| L | +5 | // | 16-bit constant (Integer) |
| L | L#523123 | // | 32-bit constant (Double Integer) |
| L | B#16#EF | // | byte in hexadecimal form |
| L | 2#0010 0110 1110 0011 | // | 16-bit binary value |
| L | 3.14 | // | 32-bit constant (Real) |

SIMATIC® S7

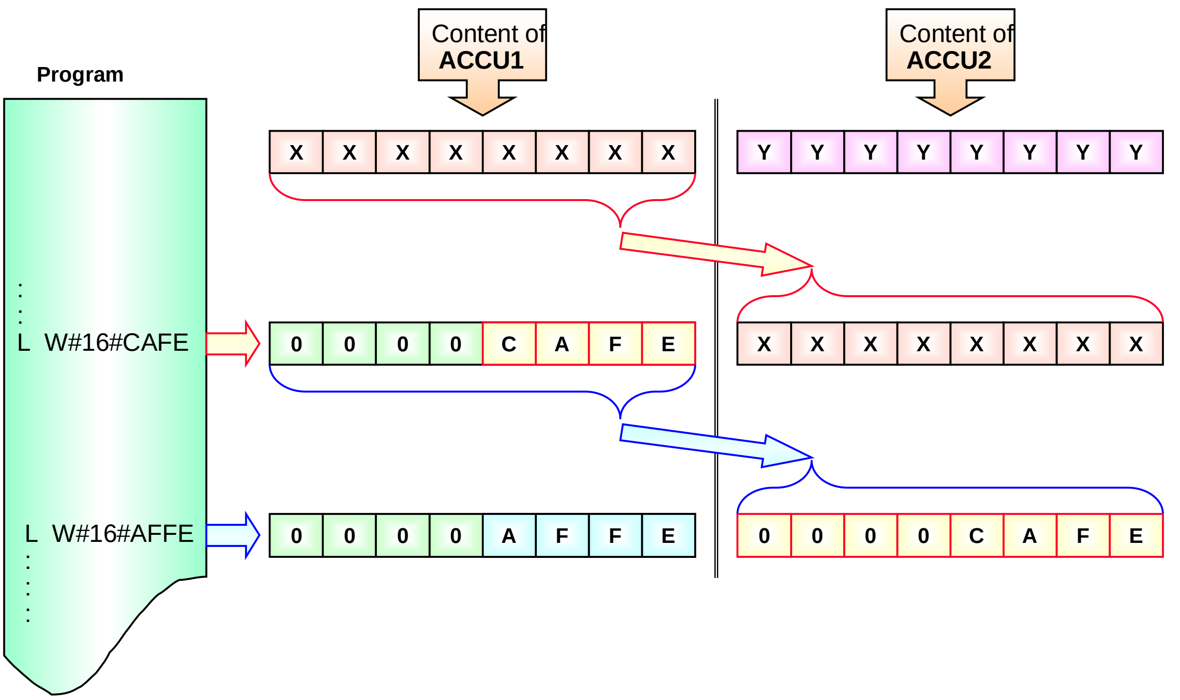

Date: 12.03.03
File: PRO1_08E.9

SITRAIN Training for Automation and Drives

**MOVE (LAD/FBD)**
If the EN input is active, the value at input "IN" is copied to the address at output "OUT".
"ENO" has the same signal state as "EN".

**L and T (STL)**
Load and transfer instructions are executed regardless of the RLO. Data is exchanged through the accumulator.
The load instruction writes the value from the source address right-justified into accumulator 1 and pads the remaining bits (32 bits in all) with "0"s.
The transfer instruction copies some or all of the contents of the accumulator 1 to the specified destination (see next page).

## Loading and Transferring Data (2)




SIMATIC® S7


Date: 12.03.03
File: PRO1_08E.10

SITRAIN Training for Automation and Drives

**ACCU1**

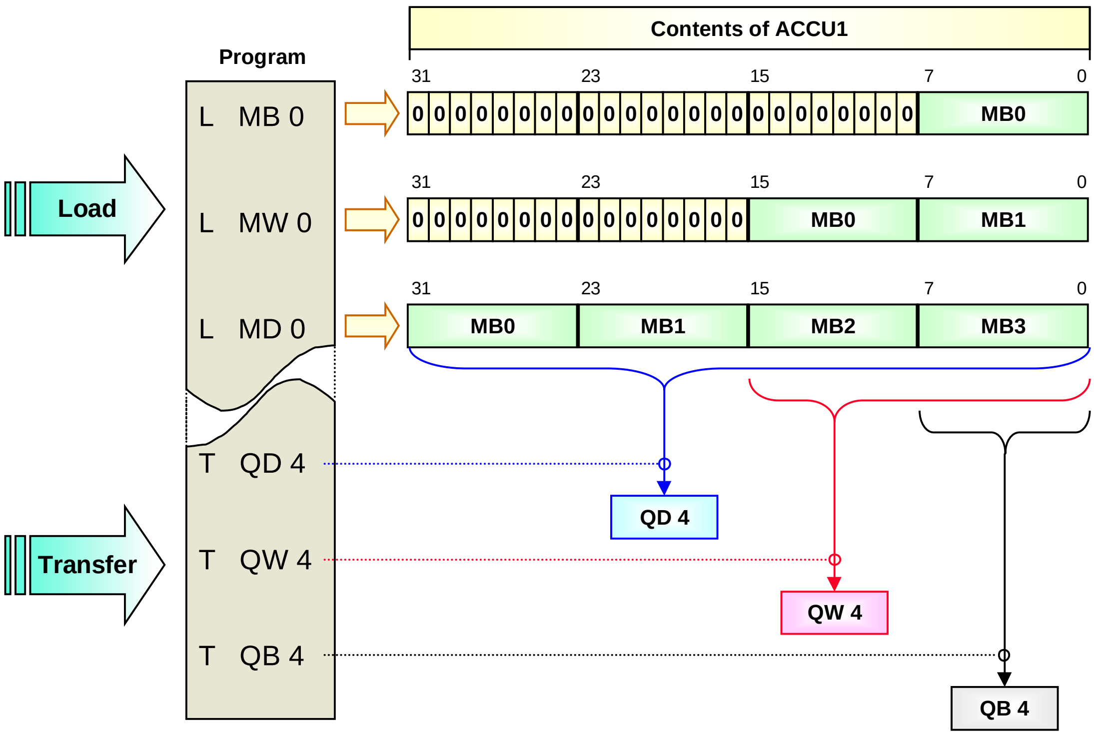
ACCU 1 is the central register in the CPU. When a load instruction is executed, the value to be loaded is written into ACCU 1. For a transfer instruction, the value to be transferred is read from ACCU 1. Results of the mathematical functions, shift and rotate operations, for example, are also entered in ACCU 1.

**ACCU2**

When a load instruction is executed, the old contents of ACCU 1 are first shifted to ACCU 2 and ACCU 1 is cleared (reset to "0") before the new value is written into ACCU 1.

ACCU 2 is also used for comparison operations, digital logic operations, mathematical and shift operations. These operations will be discussed in detail later on.

# Loading and Transferring Data (3)




SIMATIC® S7


Date: 12.03.03
File: PRO1_08E.11

SITRAIN Training for Automation and Drives

**General** Accumulators are auxiliary memories in the CPU for data exchange between various addresses used in comparison and mathematical operations. The S7-300™ has two accumulators which are 32 bits each and the S7-400™ has four accumulators also with 32 bits each.

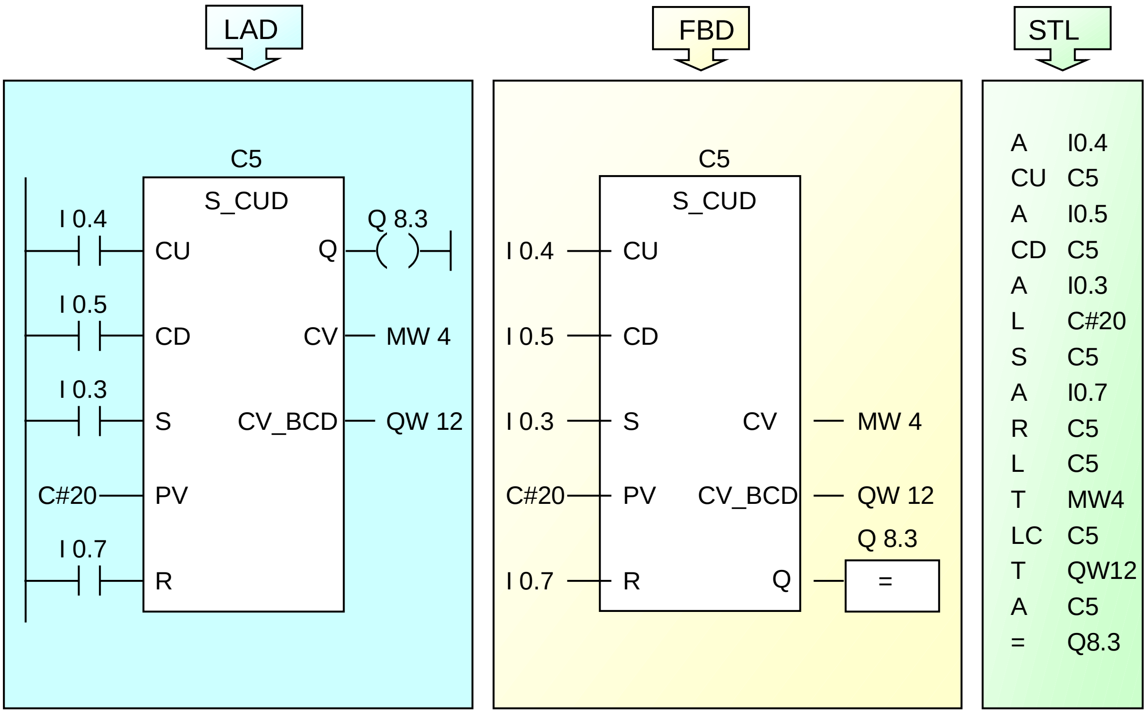
**Load** The load instruction loads the contents of the specified byte, word or double word into ACCU 1.

**Transfer** When a transfer instruction is executed, the contents of ACCU 1 are retained. Therefore, the same information can be transferred to different destinations. If only one byte is transferred, the eight bits farthest to the right are used (see diagram).

**RLO** In LAD and FBD, you can use the Enable input (EN) of the MOVE box to make load and transfer operations dependent on the RLO.

In STL, load and transfer operations are always executed, regardless of the RLO. You can implement an RLO-dependent load and transfer by using conditional jumps to skip the load / transfer instructions.

## S5 Counters in STEP 7




SIMATIC® S7


Date: 12.03.03
File: PRO1_08E.12

SITRAIN Training for Automation and Drives

**Counter Value** A 16-bit word is reserved for each counter in the system data memory. This word is used for storing the counter's value (0 to 999) in binary code.

**Count Up** When the RLO at the "CU" input changes from "0" to "1", the counter's current value is incremented by 1 (upper limit = 999).

**Count Down** When the RLO at the "CD" input changes from "0" to "1", the counter's current value is decremented by 1 (lower limit = 0).

**Set Counter** When the RLO at the "S" input changes from "0" to "1", the counter is set to the value at the "PV" input.

**Reset Counter** When the RLO at the Reset changes from "0" to "1", the counter's value is set to zero. If the reset condition is fulfilled (stays "high"), the counter cannot be set and counting in either direction is not possible.

**PV** The preset value (0 to 999) is specified in BCD format at the "PV" input as:

- as a constant (C#...)
- a BCD format through a data interface.

**CV / CV_BCD** The counter value can be loaded as a binary number (CV) or BCD number (CV_BCD) into accumulator 1 and then transferred to other addresses.

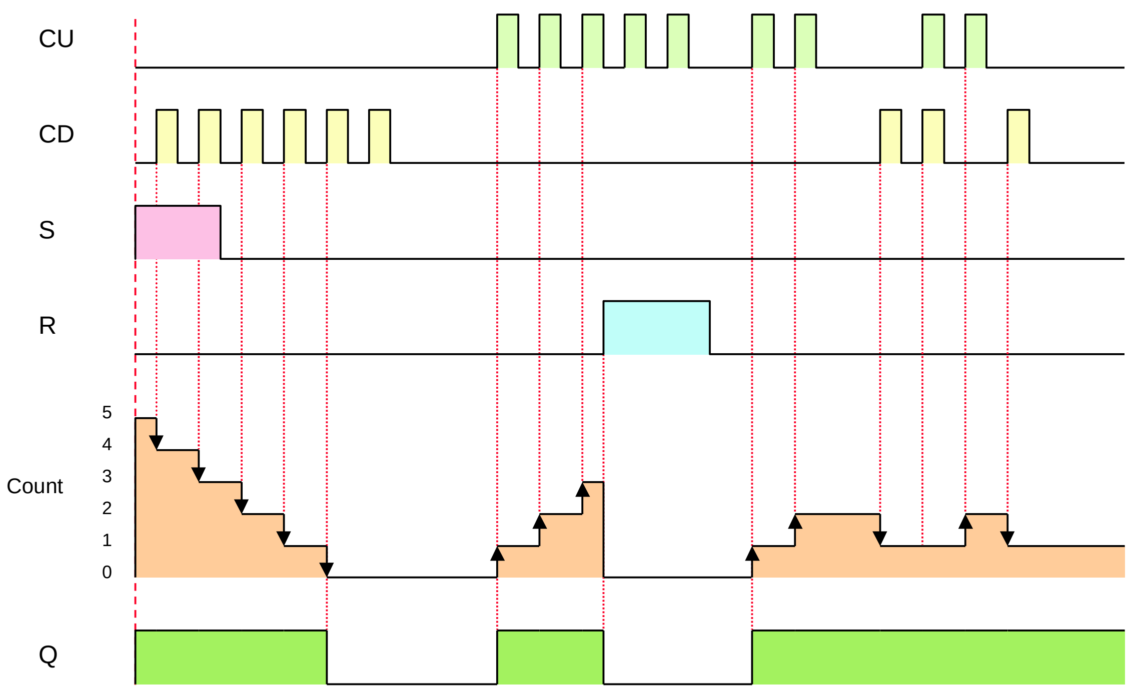
**Q** The signal state of the counter can be checked at output "Q":

- Count = 0 -> output Q = 0
- Count >< 0 -> output Q = 1

**Types of Counters**

- S_CU = Up counter (counts up only)
- S_CD = Down counter (counts down only)
- S_CUD = Up/Down counter.

## Counters: Function Diagram




SIMATIC® S7

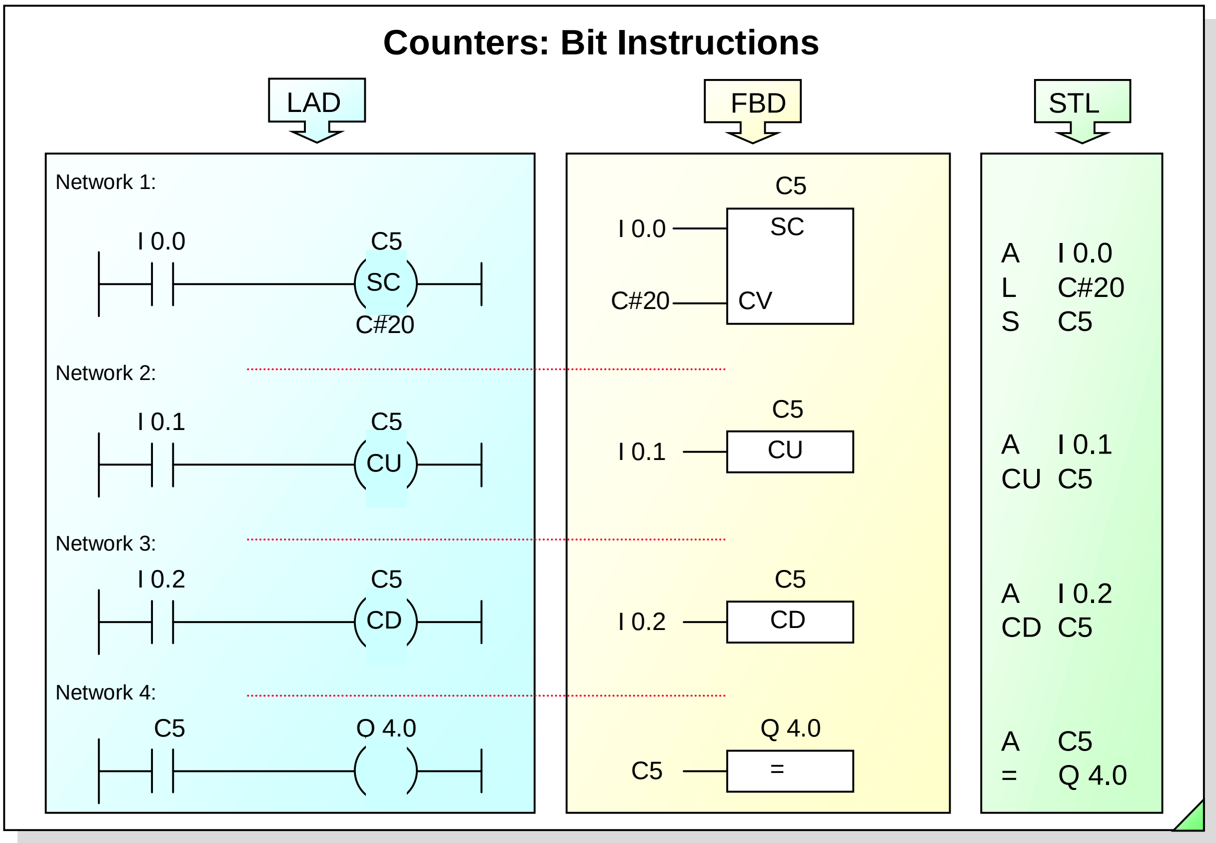

Date: 12.03.03
File: PRO1_08E.13

SITRAIN Training for Automation and Drives

**Notes**

When the counter reaches its maximum value (999), the next count up signal does not affect the counter. Likewise, when the counter reaches its minimum value (0), the next count down signal does not affect the counter. The counters do not count above 999 of lower than zero.

If an up count and a down count signal occur at the same time, the count remains the same.




SIMATIC® S7


Date: 12.03.03
File: PRO1_08E.14

SITRAIN Training for Automation and Drives

**Bit Instructions**

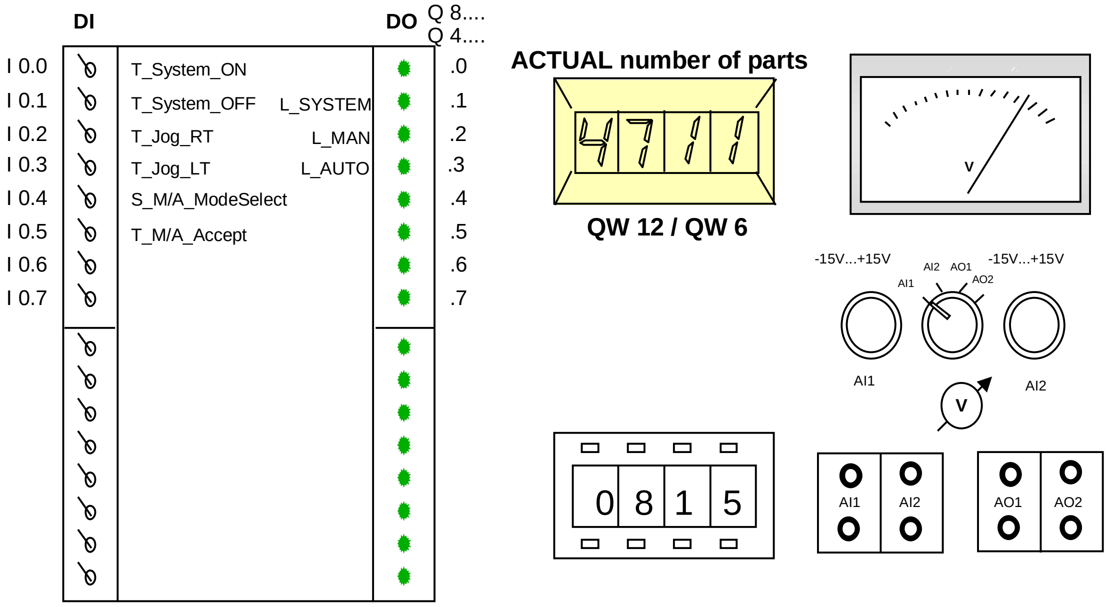
All counter functions can also operate with simple bit instructions. The similarities and differences between this method and the counter functions discussed so far are as follows:

- Similarities:
  - Setting conditions at the "SC" input
  - Specification of the counter value
  - RLO change at the "CU" input
  - RLO change at the "CD" input
- Differences:
  - It is not possible to check the current counter value since there are no Binary (CV) or BCD (CV_BCD) outputs.
  - There is no binary output Q in the graphical representation.

**Note**

IEC-compliant counters can also be implemented in STEP 7.
The use of system function blocks for implementing IEC counters is dealt with in an advanced programming course.

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




SIMATIC® S7


Date: 12.03.03
File: PRO1_08E.15

SITRAIN Training for Automation and Drives

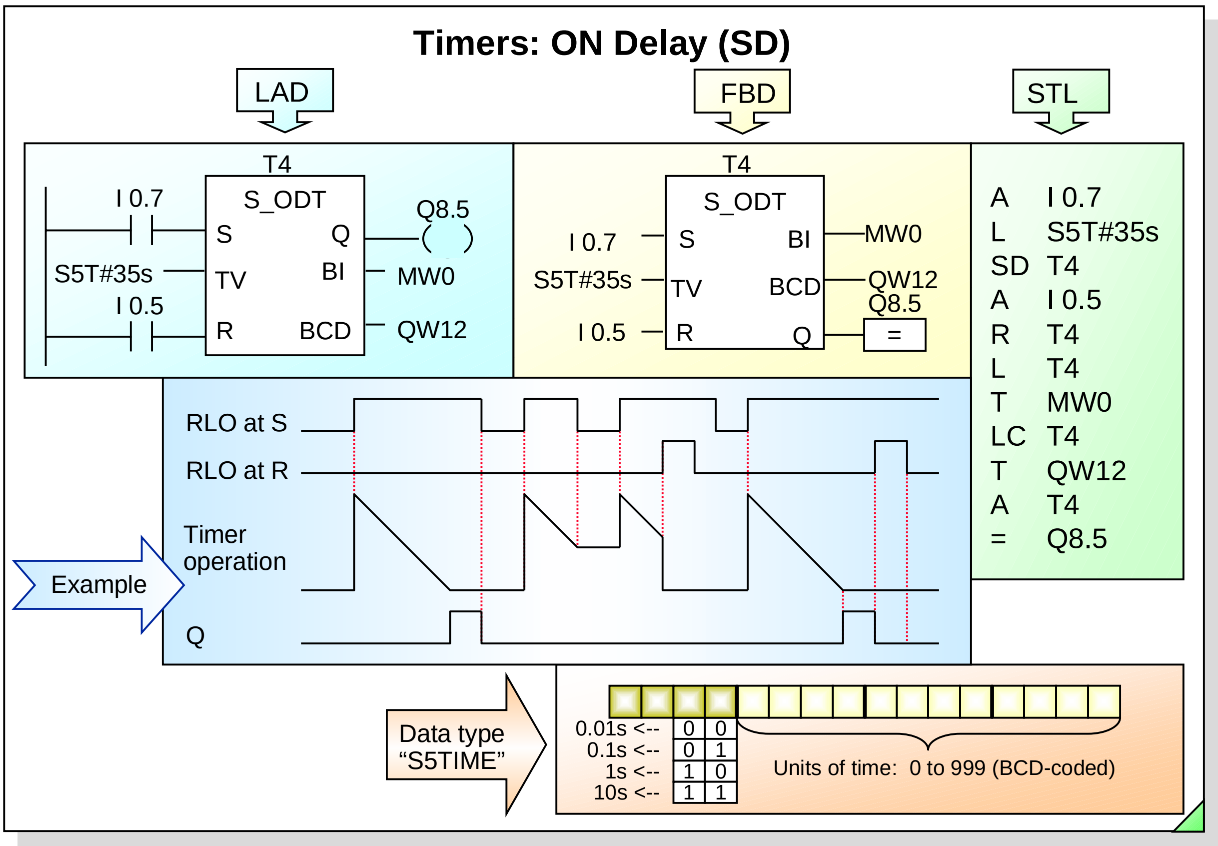
**Function Up Till Now** In the AUTO mode, parts are transported from Bay 1 or Bay 2 to the Conveyor's End passing through the light barrier. The transportation function starts as soon as a part is placed on Bay 1 or 2 and the associated momentary contact switch at that bay is pressed and it ends as soon as the part has passed through the light barrier.

**Task:**

- The parts transported in the AUTO mode are to be counted as soon as they have passed through the "LB" light barrier ("LB"  0->1).
- The number of transported parts (ACTUAL number of parts) is to be displayed on the BCD digital display.
- The counter is to be reset when the system is switched off (Q 8.1 / 4.1 = ´0´).

**What To Do**:

- Program the counting of the transported parts in function FC 18. Use the S5 counter (C 18) in FC 18 for this.
- Program the call of FC 18 in OB 1




SIMATIC® S7


Date: 12.03.03
File: PRO1_08E.16

SITRAIN Training for Automation and Drives

**Start** The timer starts when the RLO at the Start input "S" changes from "0" to "1". The timer starts with the time value specified at the Time Value "TV" for as long as the signal state at input "S" =1.

**Reset** When the RLO at the Reset input "R" changes from "0" to "1", the current time value and the time base are deleted and the output "Q" is reset.

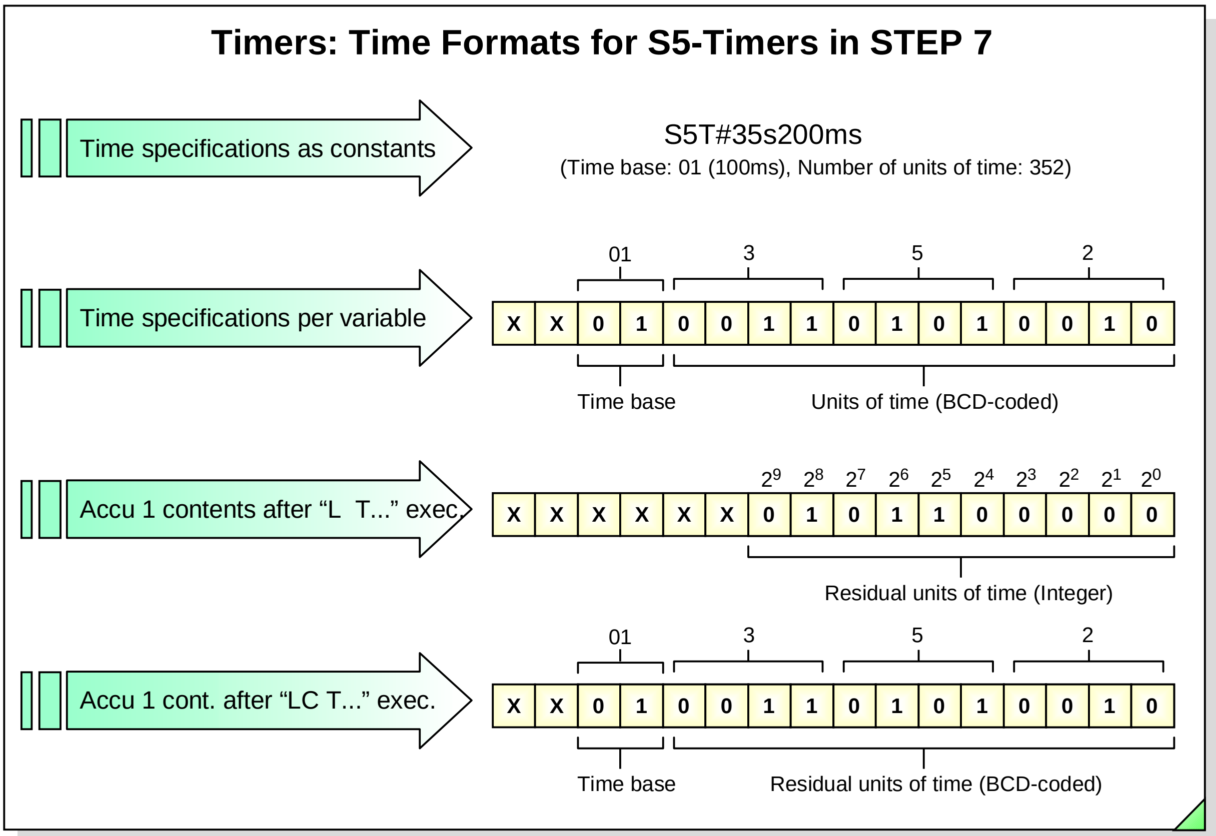
**Digital Outputs** The current time value can be read as a binary number at the "BI" output and as a BCD number at the "BCD" output.

The current time value is the initial value of "TV" minus the value for the time that has elapsed since the timer was started.

**Binary Output** The signal at the "Q" output changes to "1" when the timer has expired without error and input "S" has signal state "1".

If the signal state at the "S" input changes from "1" to "0" before the timer has expired, the timer stops running and output "Q" has a signal state "0".

**Note** In STEP 7, you can also implement IEC conforming timers using SFBs. The use of system function blocks is dealt with in an advanced programming course.




SIMATIC® S7


Date: 12.03.03
File: PRO1_08E.17

SITRAIN Training for Automation and Drives

**Time Specification**

Time values can be fixed and are specified as time constants. The permissible range in which the time values are found ranges from S5T#10ms to S5T#2h46m30s0ms.

Variable times can be specified using variables (such as memory words or data words) containing the S5TIME data type. The user must make sure that the appropriate time base and the number of units of time, as shown in the slide, are stored in the variable in his program.

**Time Base**

The time base defines the interval at which the number of units of time is to be decremented by one unit when the timer runs. Bits 12 and 13 of the variable must contain the time base as a binary-coded number:

| | | |
|---|---|---|
| Time base 0 (bit 13 = 0, bit 12 = 0) | = | 10 ms |
| Time base 1 (bit 13 = 0, bit 12 = 1) | = | 100ms |
| Time base 2 (bit 13 = 1, bit 12 = 0) | = | 1s |
| Time base 3 (bit 13 = 1, bit 12 = 1) | = | 10s |

**Units of Time**

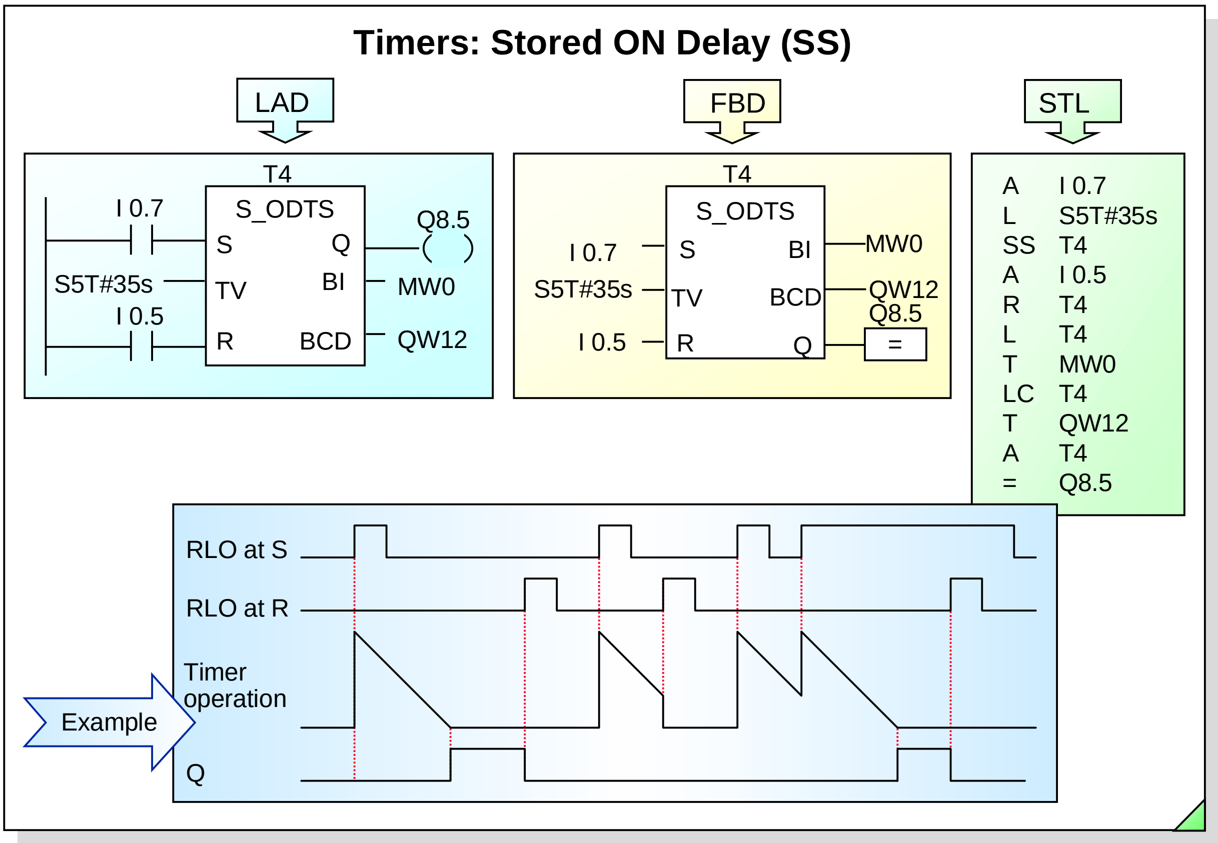
The number of units of time must be specified as a BCD-coded number. When the number of units of time are multiplied by the time base, this results in the desired time value. The range from 1 to 999 is possible. When there is a time specification using a constant (S5T#...), the system automatically uses the smallest possible time base and the number of units of time.

**L / BI**

At output "BI" or with the instruction "L T..." , the residual time value (number of units of time) of the timer is queried as an integer without time base.

**LC / BCD**

At output "BCD" or with the instruction "LC T..." , the residual time value (number of units of time) of the timer is queried as a BCD-coded number with the time base in Bit 12 and 13.




SIMATIC® S7


Date: 12.03.03
File: PRO1_08E.18

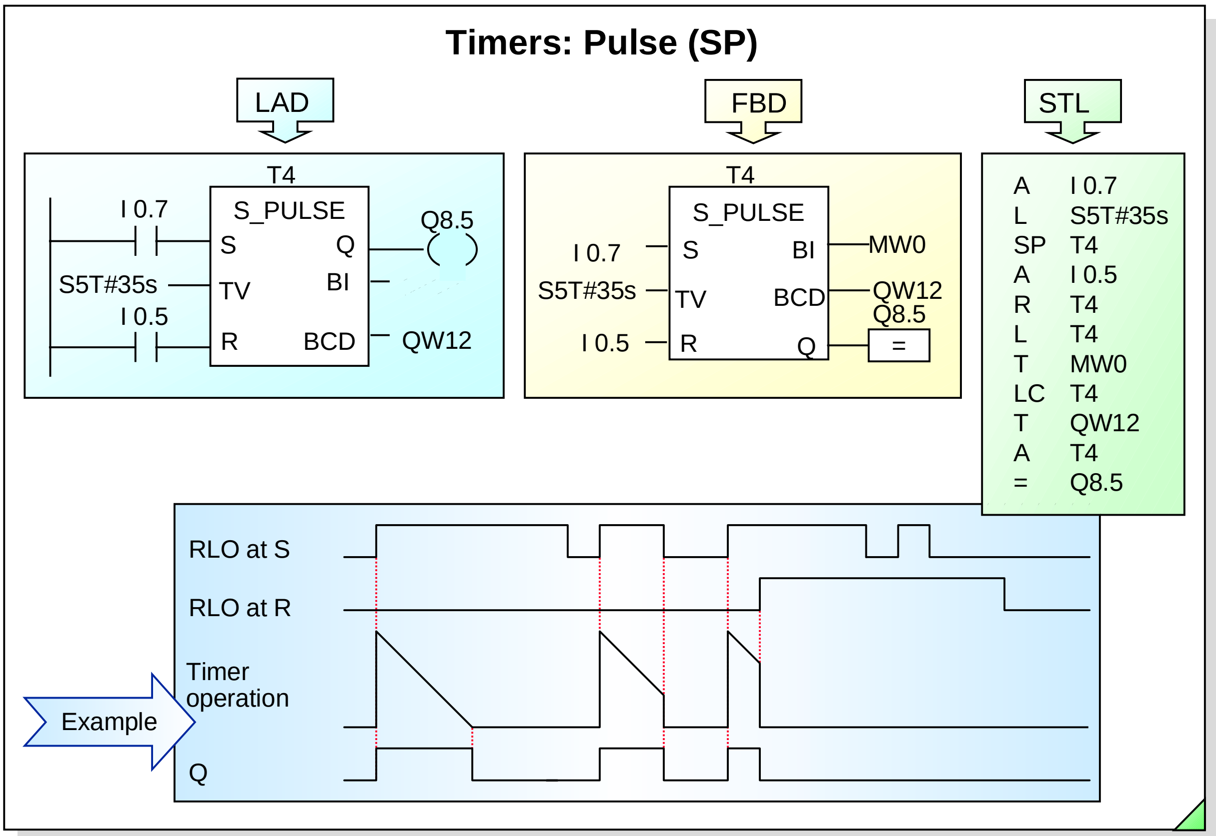
SITRAIN Training for Automation and Drives

**Start** The stored-on-delay timer starts when the RLO at the "S" input changes from "0" to "1". The timer runs starting with the time value specified at input "TV" and continues to run even if the signal at input "S" changes back to "0" during that time.

If the signal at the start input changes from "0" to "1" again while the timer is still timing down, the timer starts again from the beginning.

**Reset** When the RLO at reset input "R" changes from "0" to "1", the current time value and the time base are deleted and output "Q" is reset.

**Binary Output** The signal state at output "Q" changes to "1" when the timer has expired without error, regardless of whether the signal state at input "S" is still "1".




SIMATIC® S7


Date: 12.03.03
File: PRO1_08E.19

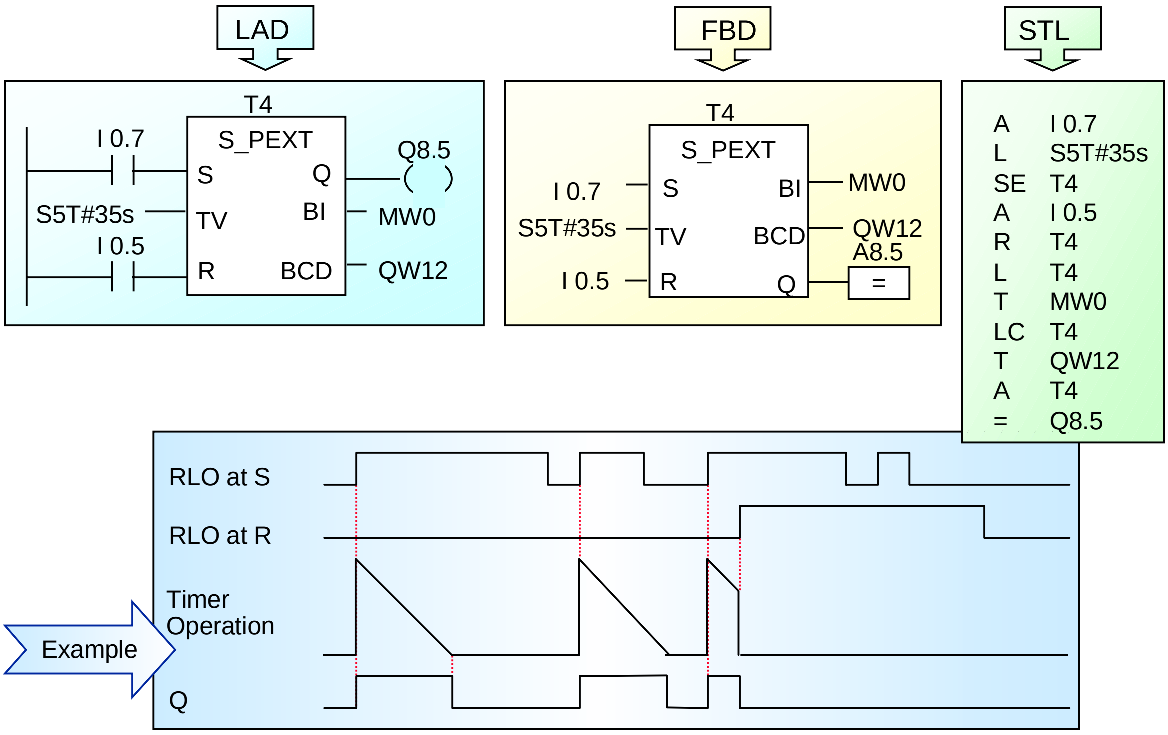
SITRAIN Training for Automation and Drives

**Start** The pulse timer starts when the RLO at the "S" input changes from "0" to "1". Output "Q" is also set to "1".

**Reset** Output "Q" is reset when:

- the timer has expired, or
- the start "S" signal changes from "1" to "0", or
- the reset input "R" has a signal state of "1".

## Timers: Extended Pulse (SE)






Date: 12.03.03
File: PRO1_08E.20

SITRAIN Training for Automation and Drives

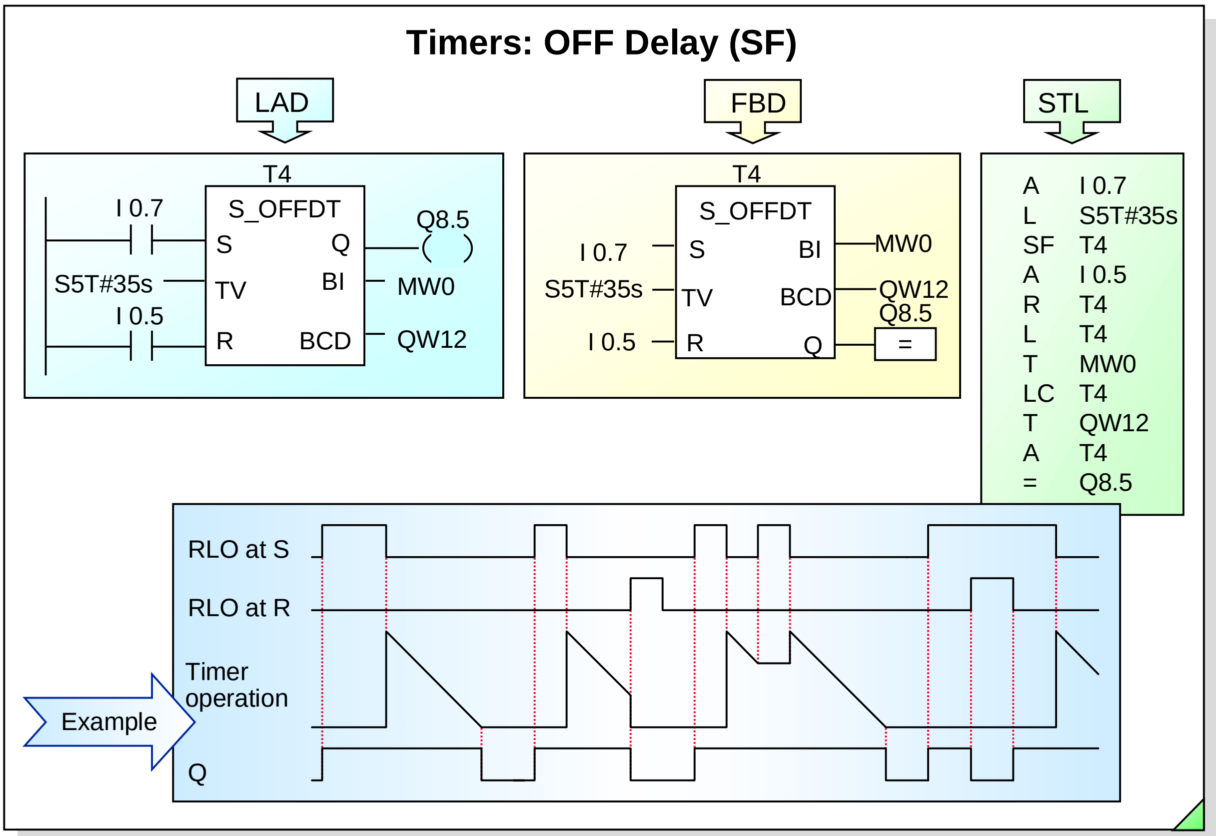
**Start**

The extended pulse timer starts when the RLO at the "S" input changes from "0" to "1". Output "Q" is also set to "1".
The signal state at output "Q" remains at "1" even if the signal at the "S" input changes back to "0".

If the signal at the start input changes from "0" to "1" again while the timer is running, the timer is restarted.

**Reset**

Output "Q" is reset when:
- the timer has expired, or
- the reset input "R" has a signal state of "1".




SIMATIC® S7



Date: 12.03.03
File: PRO1_08E.21

SITRAIN Training for Automation and Drives

**Start**

The off-delay timer starts when the RLO at the "S" input changes from "1" to "0". When the timer has expired, the signal state at output "Q" changes to "0".

If the signal state at the "S" input changes from "0" to "1" while the timer is running, the timer stops. The next time the signal state at the "S" input changes from "1" to "0", it starts again from the beginning.

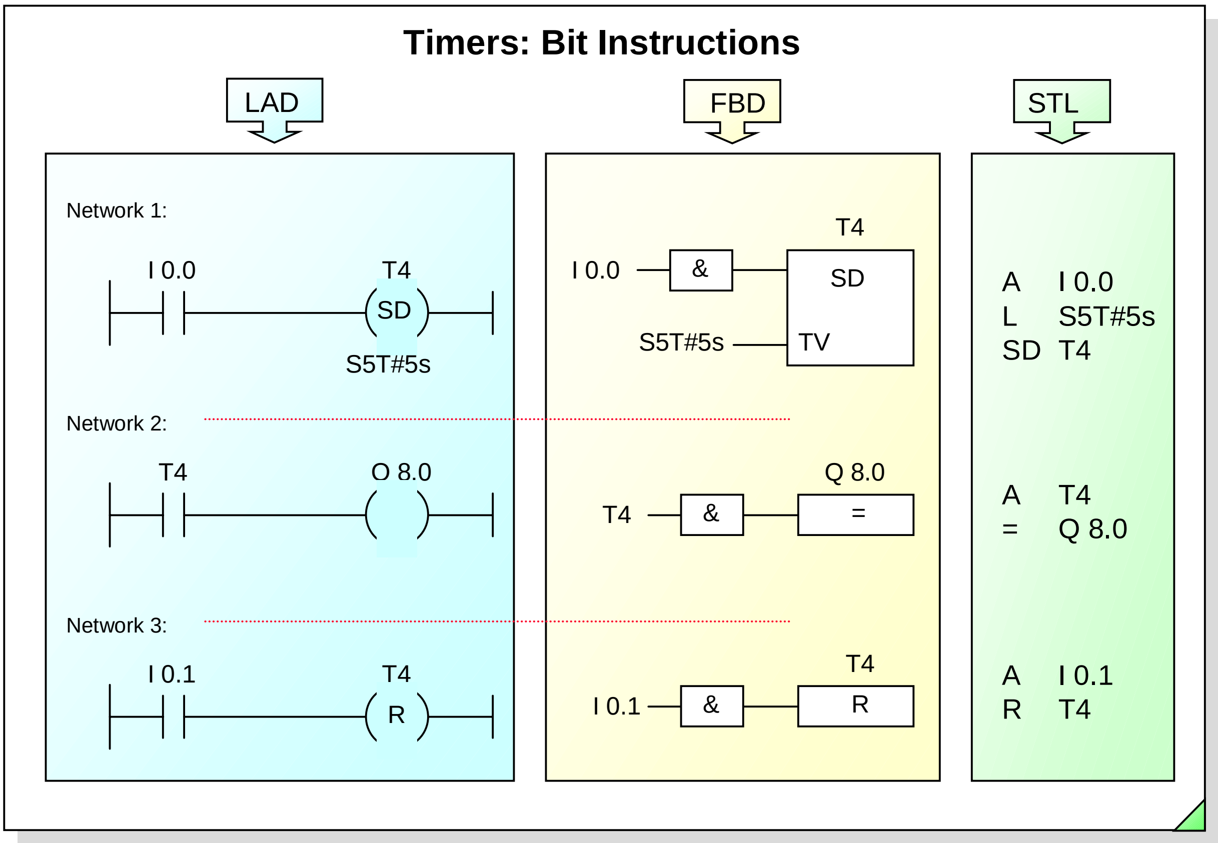
**Reset**

When the RLO at reset input "R" is "1", the current time value and the time base are deleted and output "Q" is reset.

If both inputs (S and R) have signal states of "1", output "Q" is not set until the dominant reset is deactivated.

**Binary Output**

Output "Q" is activated when the RLO at the "S" input changes from "0" to "1". If input "S" is deactivated, output "Q" continues to have signal state of "1" until the programmed time has expired.




SIMATIC® S7


Date: 12.03.03
File: PRO1_08E.22

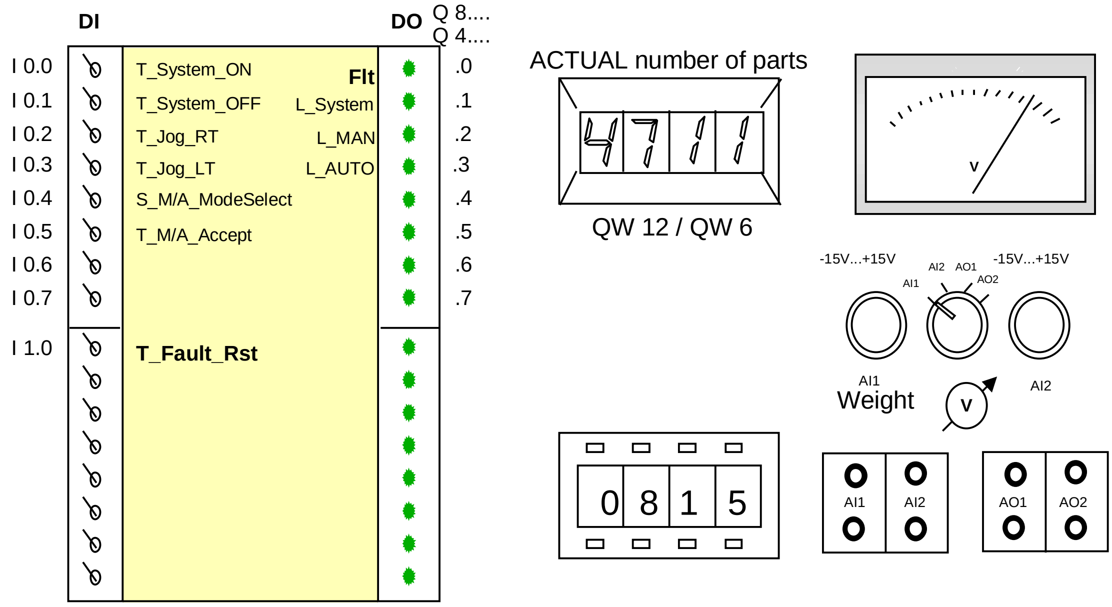
SITRAIN Training for Automation and Drives

**Bit Instructions**

All timer functions can also be started with simple bit instructions. The similarities and differences between this method and the timer functions discussed so far are as follows:

- Similarities:
  - Start conditions at the "S" input
  - Specification of the time value
  - Reset conditions at the "R" input
  - Signal response at output "Q"
- Differences (for LAD and FBD):
  - It is not possible to check the current time value (there are no "BI" and "BCD" outputs).

## Exercise: Monitoring of the Transport Functions (FC 17)




SIMATIC® S7


Date: 12.03.03
File: PRO1_08E.23

SITRAIN Training for Automation and Drives

**FC 16 Up Till Now:** In AUTO mode, parts are transported from Bay 1 or Bay 2 to the Conveyor's End. The transportation function starts as soon as a part is placed on Bay 1 or Bay 2 and the associated momentary contact switch at that bay is pressed and it ends as soon as the part has passed through the light barrier.

**Task:** You are to create a program that will monitor for a fault condition while in the AUTO mode and stop the conveyor if a fault occurs. Monitoring is to function as follows:

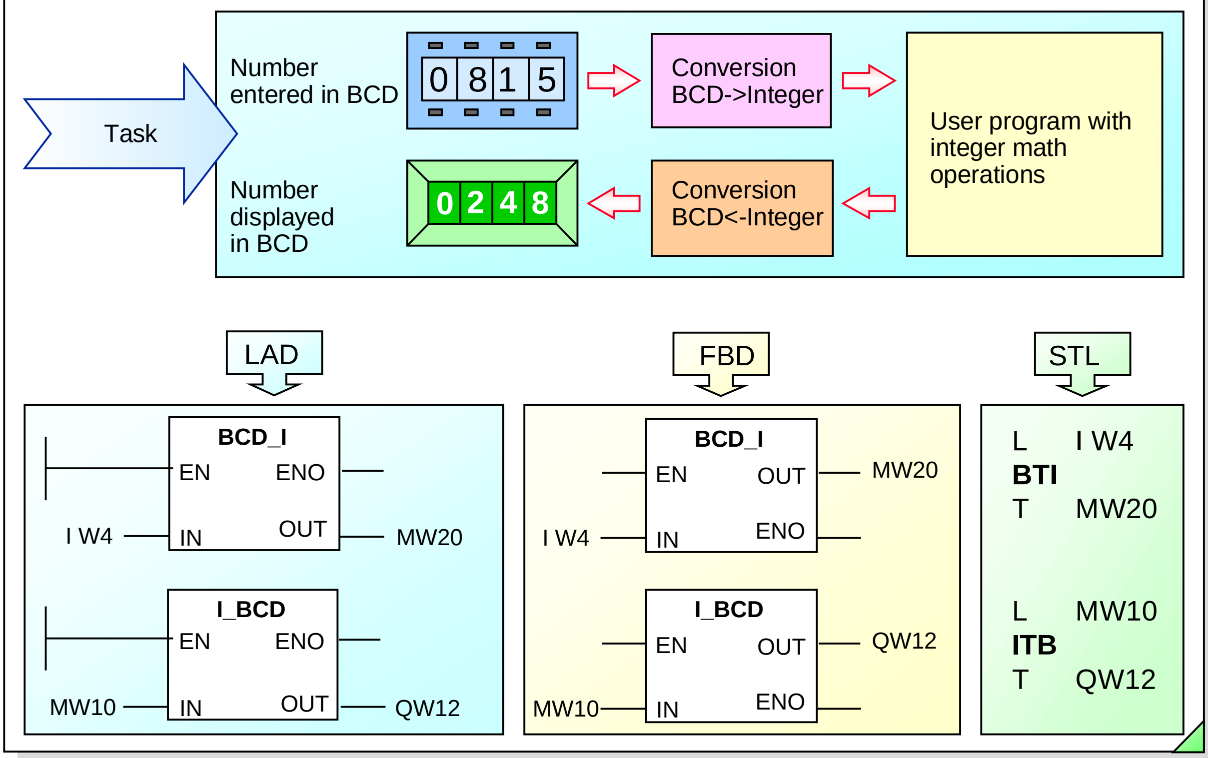
- If a part does not pass through the Light Barrier within 6 seconds of its start time, generate a fault and turn the conveyor motor OFF (REMINDER: conveyor motor is controlled in FC 16)
- A fault is displayed with a 2 Hz flashing light (bit no. 3 of the CPU clock memory byte MB 10) at the simulator LED Q 8.0 / Q 4.0
- A fault must be acknowledged through the simulator momentary contact switch "T_Fault_Rst" **I** 1.0
- The conveyor can only be started as described above after the fault has been acknowledged. (lock-out in FC 16)

**What To Do**:

- Program the described monitoring function in FC 17
  - Use the S5 timer T 17 as ON delay (SD) for monitoring
  - Set the M 17.0 bit memory when a fault occurs so that you can then process it in FC 16.
- Program the call of FC 17 in OB 1
- Modify FC 16 to include the necessary lock-out or switching off of the conveyor motor when a fault occurs.

SIEMENS

## Conversion Operations BCD <-> Integer




SIMATIC® S7



Date: 12.03.03
File: PRO1_08E.24

SITRAIN Training for Automation and Drives

**Example**

A user program is to perform mathematical functions using values entered with thumbwheel buttons and display the result on a digital display. Mathematical functions cannot be performed in BCD format, so the format must be changed.

**Conversion Instructions**

The instruction set of the S7-300™/400™ supports a multitude of conversion operations. The instructions all have the same format.

**EN, ENO**

If RLO is =1 at Enable input EN, the conversion is performed. Enable output ENO always has the same signal state as EN. If this is not the case, it is clearly indicated in the corresponding instructions.

**IN**

When EN=1, the value at IN is read into the conversion instruction.

**OUT**

The result of the conversion is stored at the address at the OUT output.

**BCD_I / BTI**

(Convert BCD to integer) reads the contents of the IN parameter as a three-digit BCD number (+/- 999) and converts it to an integer value (16 bits).

**I_BCD / ITB**

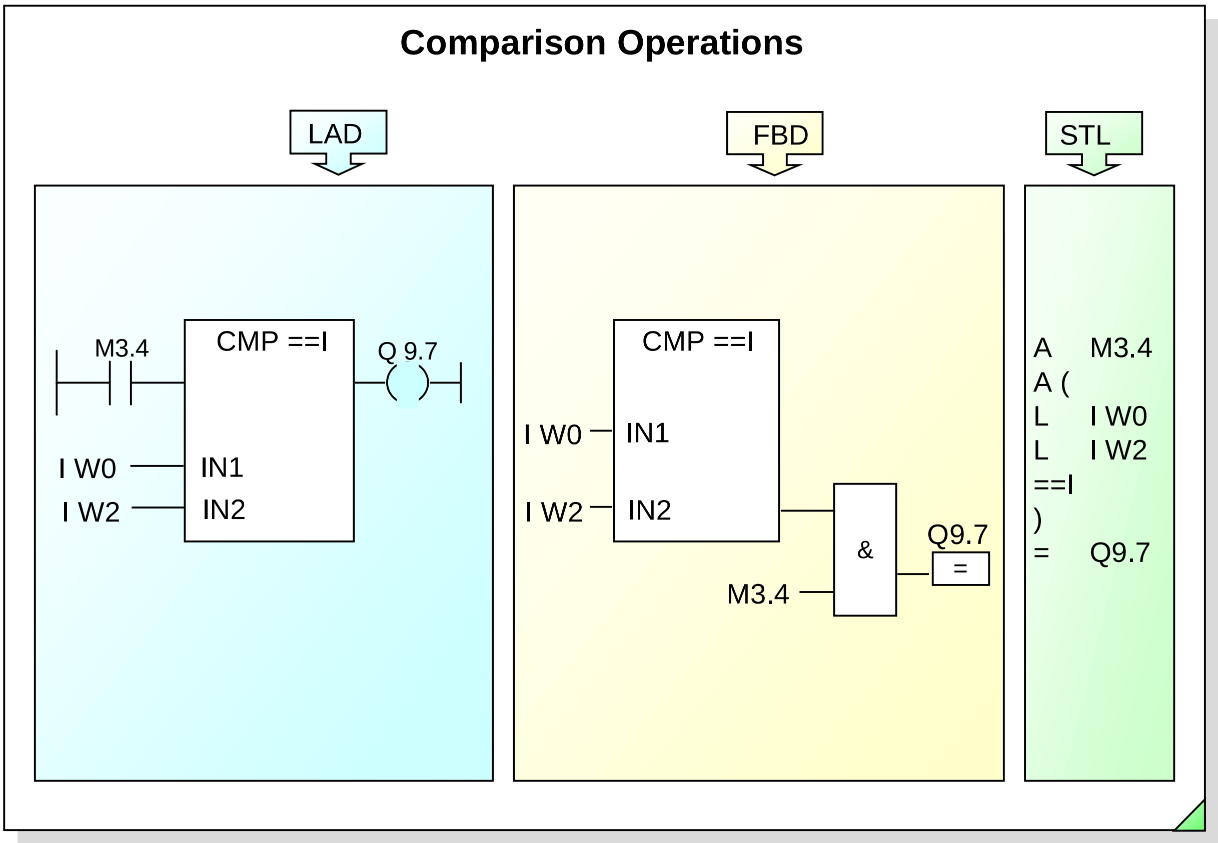
(Convert integer to BCD) reads the contents of the IN parameter as an integer value (16 bits) and converts this value to a three-digit BCD number (+/- 999). If an overflow occurs, ENO = 0.

**BCD_DI / BTD**

Converts a BCD number (+/- 9999999) to a double integer (32 bits).

**DI_BCD / DTB**

Converts a double integer to a seven-digit BCD number (+/- 9999999). If an overflow occurs, ENO = 0.




SIMATIC® S7


Date: 12.03.03
File: PRO1_08E.25

SITRAIN Training for Automation and Drives

**CMP**

You can use comparison instructions to compare the following pairs of numerical values:

**I** Compare integers (on the basis of 16 bit fixed-point number)

**D** Compare integers (on the basis of 32 bit fixed-point number)

**R** Compare floating-point numbers (on 32 bit real number basis = IEEE floating-point numbers).

If the result of the comparison is "true", then the RLO of the operation is "1", otherwise it is "0".

The values at inputs IN1 and IN2 are compared for conformity with the specified condition:

**==** IN1 is equal to IN2

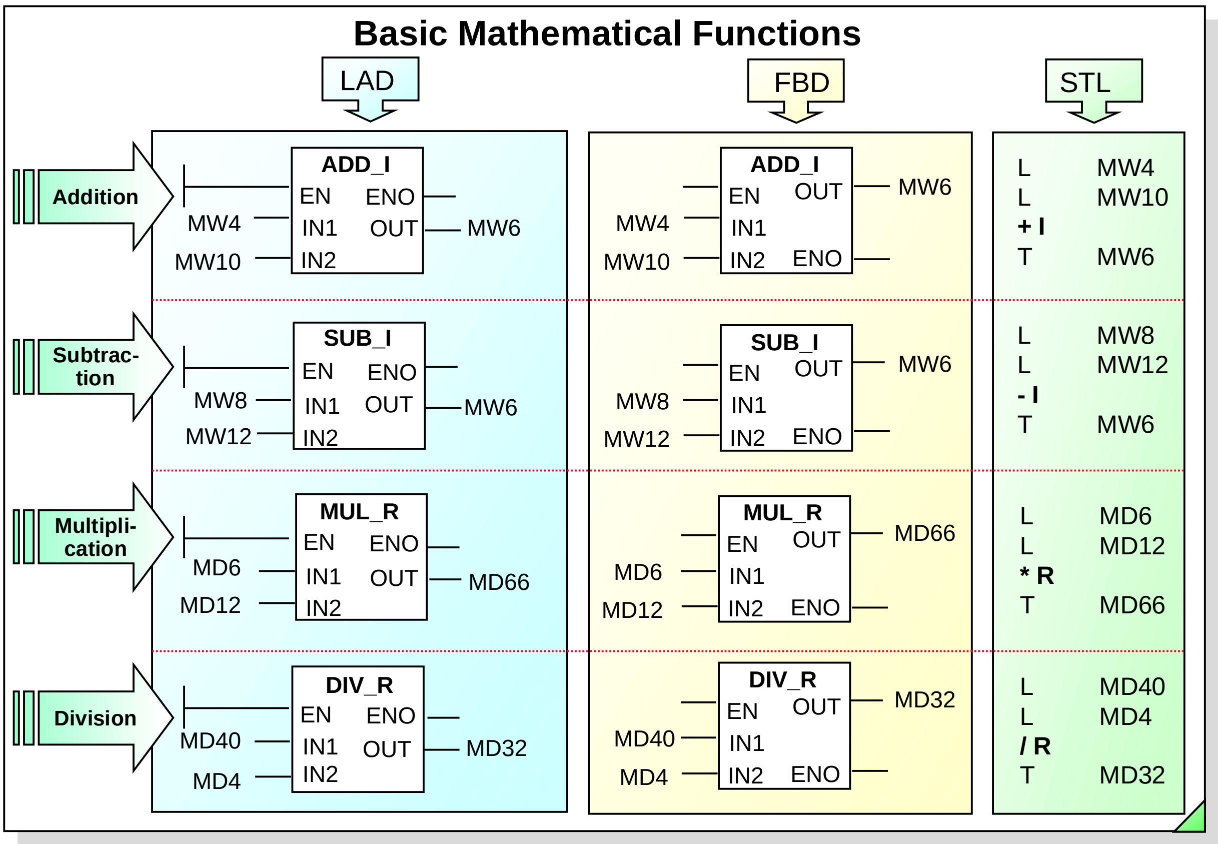
**<>** IN1 is not equal to IN2

**>** IN1 is greater than IN2

**<** IN1 is less than IN2

**>=** IN1 is greater than or equal to IN2

**<=** IN1 is less than or equal to IN2.




SIMATIC® S7 Date: 12.03.03
 File: PRO1_08E.26

SITRAIN Training for Automation and Drives

**General** The instruction set of the S7-300™/400™ supports a multitude of mathematical functions. All the instructions have the same format.

**EN** The instruction is executed if RLO is =1 at Enable input EN.

**ENO** If the result is outside the permissible range for the data type concerned, overflow bits OV= "Overflow" and OS= "Stored Overflow" are set and Enable output ENO=0. This prevents subsequent operations dependent on ENO from being executed.

**IN1, IN2** The value at IN1 is read in as the first address and the value at IN2 as the second.

**OUT** The result of the mathematical operation is stored at the address at output OUT.

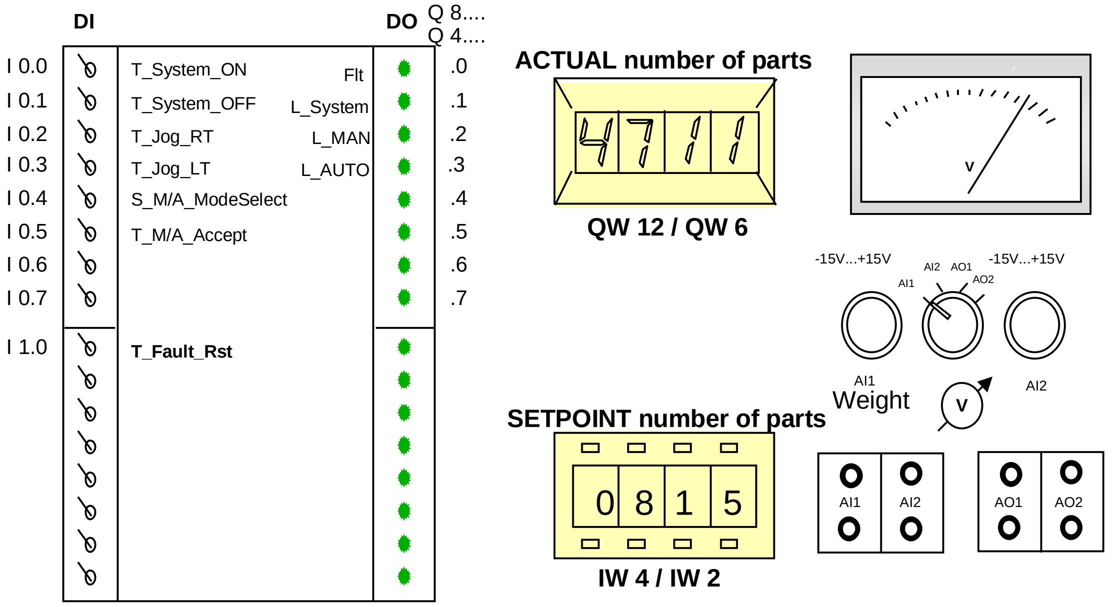
**Instructions**

| | | |
|---|---|---|
| Addition: | ADD_I | Add integer |
| | ADD_DI | Add double integer |
| | ADD_R | Add real number |
| Subtraction: | SUB_I | Subtract integer |
| | SUB_DI | Subtract double integer |
| | SUB_R | Subtract real number |
| Multiplication: | MUL_I | Multiply integer |
| | MUL_DI | Multiply double integer |
| | MUL_R | Multiply real number |
| Division: | DIV_I | Divide integer |
| | DIV_DI | Divide double integer |
| | DIV_R | Divide real number |

**Note** The advanced mathematical functions (ABS, SQR, SQRT, LN, EXP, SIN, COS, TAN, ASIN, ACOS, ATAN) are discussed in an advanced programming course.

SIEMENS

## Exercise: Counting the Transported Parts (FC 18, MW 20)




SIMATIC® S7


Date: 12.03.03
File: PRO1_08E.27

SITRAIN Training for Automation and Drives

**FC 18 Up Till Now:**

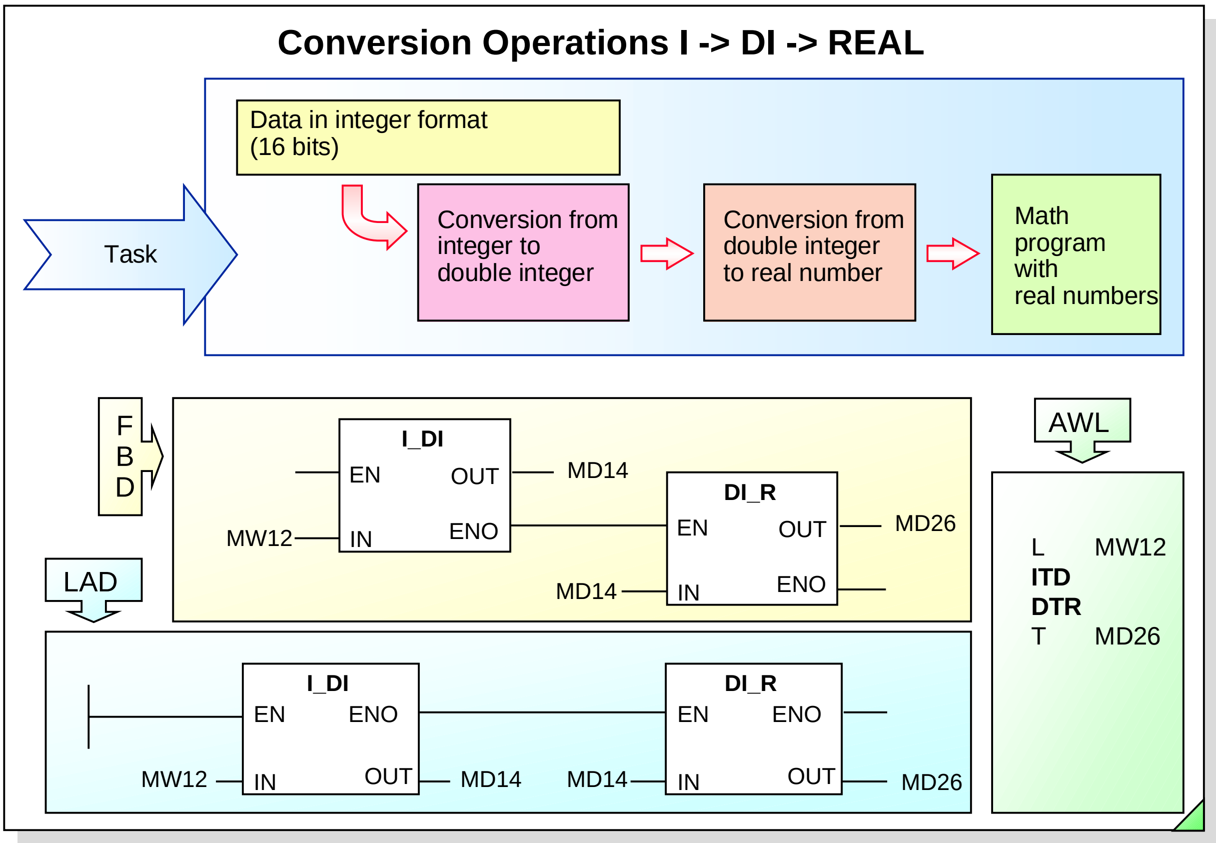
- The parts transported in AUTO mode are to be counted (with S5 counter C 18), as soon as they have passed the light barrier "LB" ("LB" 0->1).
- The number of transported parts (ACTUAL number of parts) is displayed on the BCD digital display.
- The counter is reset when the system is switched off (Q 8.1 / 4.1 = ´0´)

**Task:**

- Counting the transported parts is no longer to be done with the S5 counter C 18, but rather by addition using MW 20 to store the sum. Use the M 18.0 memory bit as the auxiliary memory marker for the necessary edge detection.
- The *SETPOINT number* of how many parts are to be transported can be set through the BCD thumbwheel button. When the given *SETPOINT number of parts* is *reached,* it is displayed on the red LED (Q 20.4 / Q 8.4) at the Conveyor End.
- As long as the message *SETPOINT number of parts reached* (red LED) exists, no other transport function can be started (lock-out in FC 16).
- The message can be acknowledged with the momentary contact switch at the Conveyor End. The acknowledgement resets the *ACTUAL number of parts* (MW 20) to 0, just as it was for switching off the system.

**What To Do**:

- Insert the OB 121 organization block into your program and download it to the CPU. A program in OB 121 is not necessary. Downloading the "empty" OB 121 prevents the CPU from going into the STOP state during setting of the *SETPOINT number of parts,* which is caused by a "rebound" from the BCD thumbwheel button (more information on this in the Chapter OBs).
- Amend your current FC 18 for counting the parts to the new task
- Modify the FC 16 for controlling the conveyor motor according to the task.




SIMATIC® S7

Date: 12.03.03
File: PRO1_08E.28
SITRAIN Training for Automation and Drives

**Example** A user program that works with integers also needs to perform division, which is likely to result in values less than 1. Since these values can be represented only as real numbers, conversion to real numbers is necessary. To do this, the integer must first be converted to a double integer.

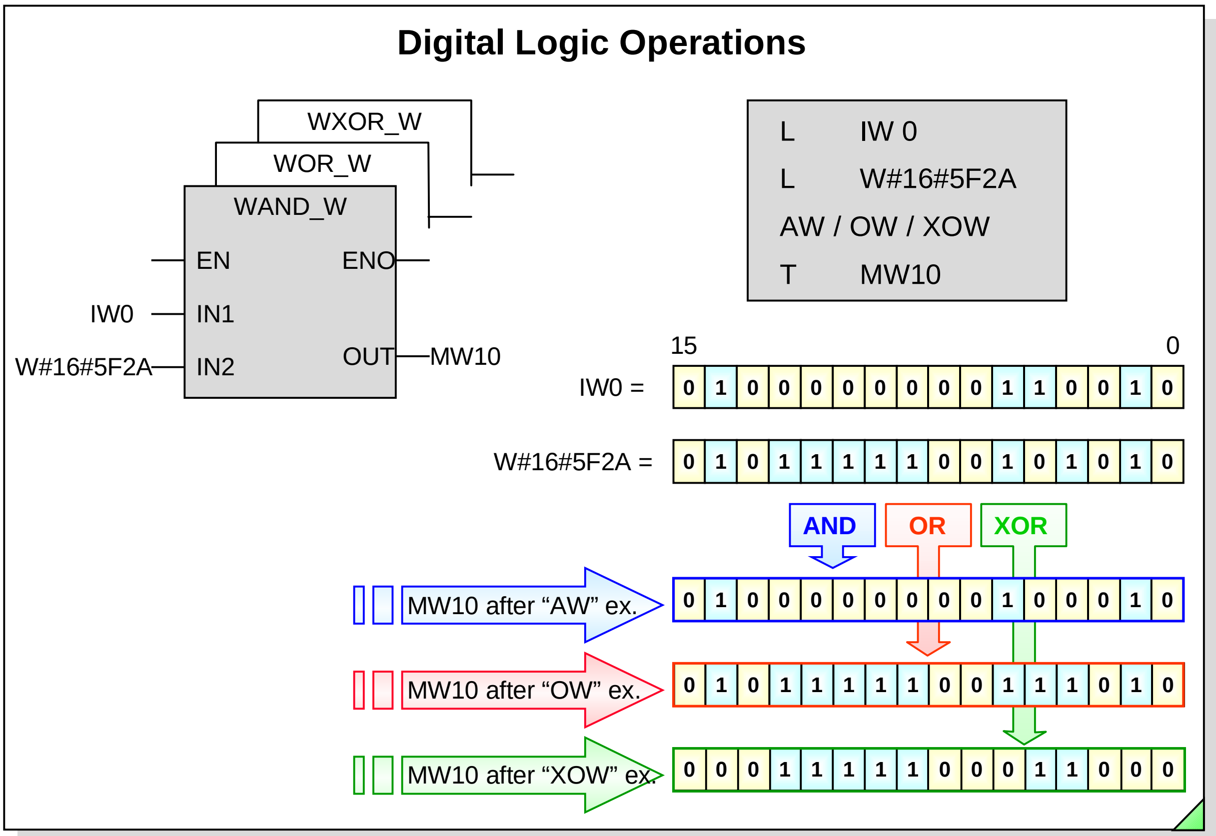
**I_DI / ITD** Converts an integer to a double integer.

**DI_R / DTR** Converts a double integer to a real number.

**Note** Other conversion instructions, such as:

- INV_I / INVI
- NEG_I / NEGI
- TRUNC / TRUNC
- ROUND / RND
- CEIL / RND+
- FLOOR / RND-
- INV_DI / INVD
- NEG_DI / NEGD
- NEG_R / NEGR
- CAW, CAD

are discussed in an advanced programming course.




SIMATIC® S7



Date: 12.03.03
File: PRO1_08E.29

SITRAIN Training for Automation and Drives

**WAND_W**

The "AND Word" operation gates the two digital values at inputs IN1 and IN2 bit by bit in accordance with the AND truth table. The result of the AND operation is stored at the address at output OUT.
The instruction is executed when EN = 1.

Example: Masking out the 4th decade of the thumbwheel buttons :

| | |
|---|---|
| IW4= | = 0100 0100 1100 0100 |
| W#16#0FFF | = 0000 1111 1111 1111 |
| MW30 | = 0000 0100 1100 0100 |

**WOR_W**

The "OR Word" operation gates the two digital values at inputs IN1 and IN2 bit by bit in accordance with the OR truth table. The result of the OR operation is stored at the address at output OUT.
The instruction is executed when EN = 1.

Example: Setting bit 0 in MW32 :

| | |
|---|---|
| MW32 | = 0100 0010 0110 1010 |
| W#16#0001 | = 0000 0000 0000 0001 |
| MW32 | = 0100 0010 0110 1011 |

**WXOR_W**

The "Exclusive OR Word" operation gates the two digital values at inputs IN1 and IN2 bit by bit in accordance with the XOR truth table. The result of the OR operation is stored at the address at output OUT. The result of the XOR operation is stored at the address at output OUT.
The instruction is executed when EN=1.

Example: detecting signal changes in IW0 :

| | |
|---|---|
| IW0 | = 0100 0100 1100 1010 |
| MW28 | = 0110 0010 1011 1001 |
| MW24 | = 0010 0110 0111 0011 |

# Data Storage in Data Blocks

SIMATIC® S7


Date: 12.03.03
File: PRO1_09E.1

SITRAIN Training for Automation and Drives

## Contents

Page

## Objectives

**Upon completion of this chapter the participant will ...**

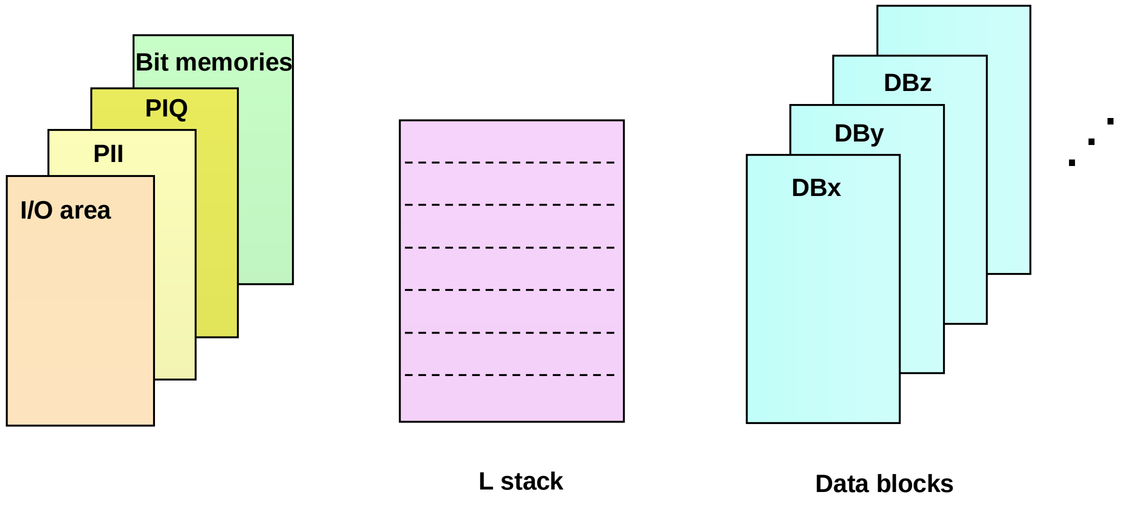
... understand the purpose of global data blocks

... be familiar with elementary and complex data types

... be able to edit, save and download into the CPU a data block with elementary variables

... be familiar with and be able to apply the possibilities for addressing data block variables

## Storage Areas for Data






**Overview**

In addition to program blocks, a user program also consists of data containing information about process states, signals, etc. This data is then processed according to the instructions in the user program.

Data is stored in variables of the user program, which are uniquely identified by:

- Storage location (address: such as P, PII, PIQ, bit memory, L stack, DB)
- Data type (elementary or complex data type, parameter type)

Depending on the accessibility, a distinction is also made between:

- Global variables, which are declared in the global symbol table or in global data blocks
- Local variables, which are declared in the declaration part of OBs, FBs and FCs.

Variables can have a permanent storage location in the process image, bit memory area or in a data block. They can also be created dynamically in the L stack when a block is being executed.

**Local Data Stack**

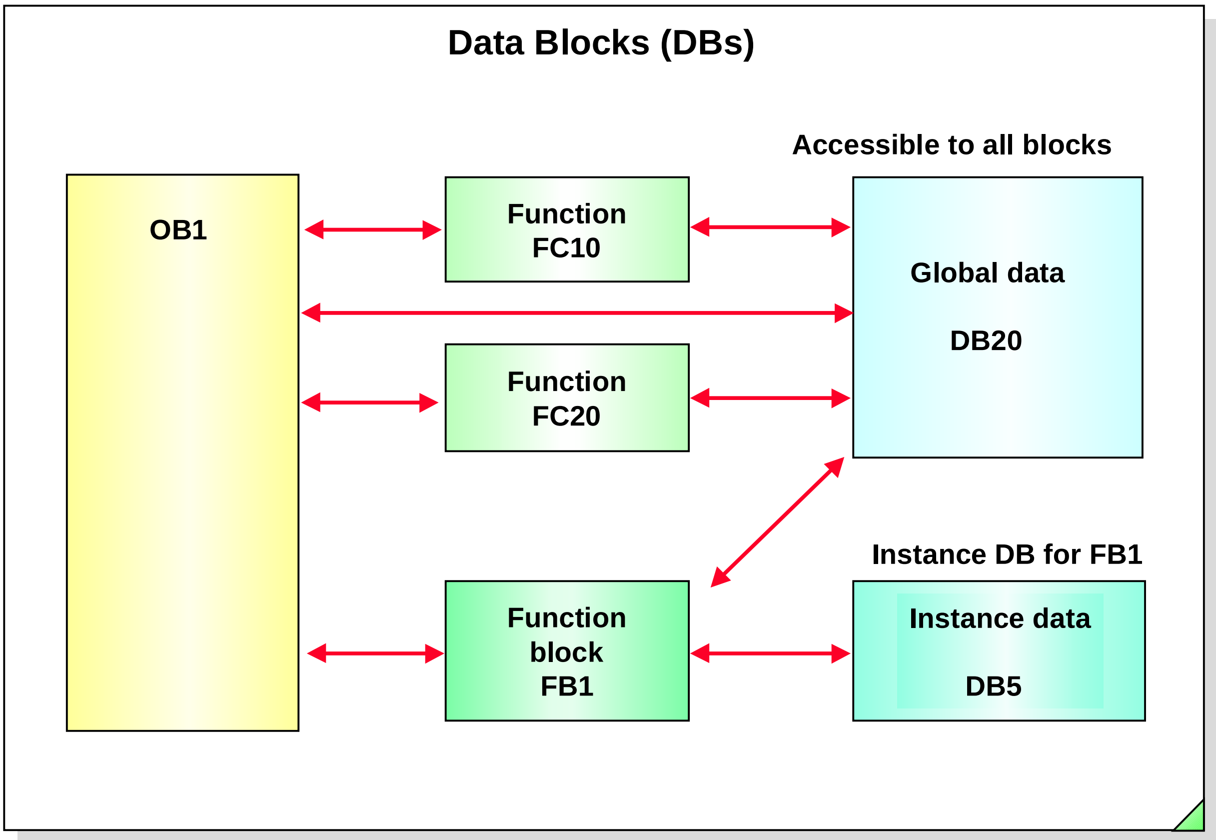
The local data stack (L stack) is an area for storing:

- temporary variables of a logic block, including OB start information
- actual addresses in the parameter passing of FC calls
- intermediate logic results in LAD programs

This topic is dealt with in the chapter "Functions and Function Blocks".

**Data Blocks**

Logic blocks of the user program use data blocks for storing values. Unlike the temporary data, the data in data blocks is not overwritten when execution of the logic block is completed or when the DB is closed.




SIMATIC® S7


Date: 12.03.03
File: PRO1_09E.4

SITRAIN Training for Automation and Drives

**Overview**

Data blocks are used for storing user data. Like logic blocks, data blocks take up space in the user memory. Data blocks contain variable data (such as numeric values) that is used in the user program.

The user program can access the data in a data block with bit, byte, word or doubleword operations. Symbolic or absolute addresses can be used.

**Uses**

You can use data blocks in different ways, depending on their contents. You differentiate between:

- Global data blocks: These contain information that all the logic blocks (that would include OB1) in the user program can access.
- Instance data blocks: These are always assigned to a particular FB. The data in each DB should be used only by the assigned FB.
  Instance data blocks are dealt with in more detail in the "Functions and Function Blocks" chapter.

**Creating DBs**

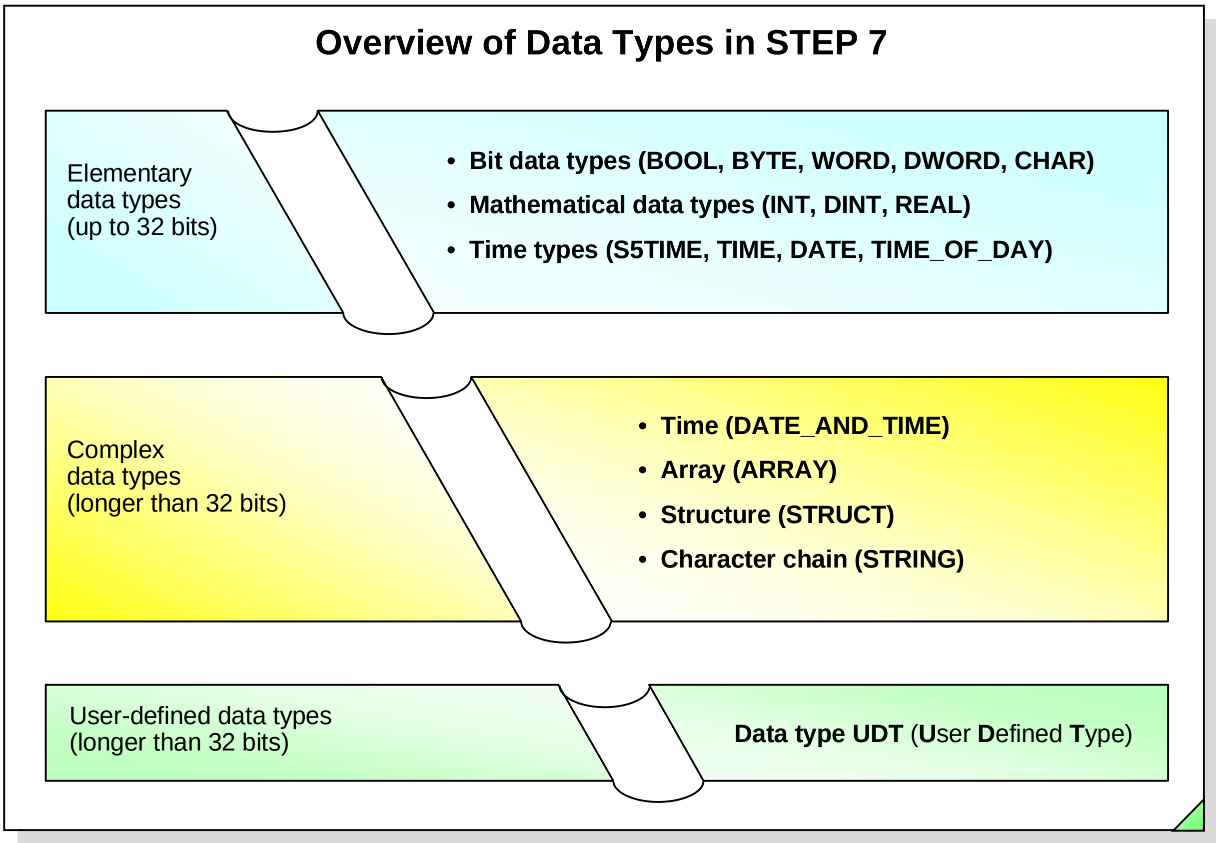
You can create global DBs with either the Program Editor or with a "user-defined data type" (UDT) that you have already created.

Instance data blocks are created when an FB block is called.

**Registers**

The CPU has two data block registers, the DB and DI registers. Therefore, you can have two data blocks open at the same time.

This topic is dealt with in an advanced programming course.




SIMATIC® S7



Date: 12.03.03
File: PRO1_09E.5

SITRAIN Training for Automation and Drives

**Overview**

Data types determine the properties of data, that is, how the contents of one or more associated addresses are to be represented and what the permissible range of values is.

The data type also determines which operations can be used.

**Elementary Data Types**

Elementary data types are predefined in accordance with IEC 61131-3. The data type determines the amount of memory space required. For example, the word data type takes up 16 bits in the user memory.

Elementary data types are never more than 32 bits long and can be loaded into the accumulators of the S7 processor in full and processed with elementary STEP 7 instructions.

**Complex Data Types**

Complex data types can only be used in conjunction with variables declared in global data blocks. Complex data types cannot be completely loaded into the accumulators with load instructions. You use standard blocks from the library ("IEC" S7 Program) to process complex data types.

**User-Defined Data Types**

A user-defined data type can be used for data blocks or as a data type in a variable declaration table.

You use the Data Block Editor to create UDTs.

The structure of a UDT can contain groups of elementary and/or complex data types.

## Elementary Data Types in STEP 7

| Keyword | Length (in bits) | Example of a constant of this type |
|---|---|---|
| BOOL | 1 | 1 or 0 |
| BYTE | 8 | B#16#A9 |
| WORD | 16 | W#16#12AF |
| DWORD | 32 | DW#16#ADAC1EF5 |
| CHAR | 8 | ' w ' |
| S5TIME | 16 | S5T#5s_200ms |
| INT | 16 | 123 |
| DINT | 32 | 65539 |
| REAL | 32 | 1.2 or 34.5E-12 |
| TIME | 32 | T#2D_1H_3M_45S_12MS |
| DATE | 16 | D#1993-01-20 |
| TIME_OF_DAY | 32 | TOD#12:23:45.12 |

SIMATIC® S7


Date: 12.03.03
File: PRO1_09E.6

**SITRAIN** Training for Automation and Drives

**BOOL, BYTE, WORD DWORD, CHAR**

Variables of the BOOL data type consist of one bit. Variables of BYTE, WORD, and DWORD data types are sequences of 8, 16 and 32 bits respectively. The individual bits are not evaluted in these data types.

Special forms of these data types are the BCD numbers and the count value used in conjunction with the count function. The CHAR data type represents a character in ASCII code.

**S5TIME**

Variables of the S5TIME data type are required for specifying time values in timer functions. The format is *S5T#,* followed by the time. You specify the time in hours, minutes, seconds or milliseconds. You can enter the timer values with an underline (1h_4m) or without an underline (1h4m).
Functions FC 33 and FC40 from the library convert S5TIME to TIME format and TIME to S5TIME format.

**INT, DINT, REAL**

Variables of these data types represent numbers that can be used in mathematical operations.

**TIME**

A variable of data type TIME takes up a doubleword. This variable is used, for example, for specifying timer values in IEC timer functions. The contents of the variable are interpreted as a DINT number in milliseconds and can be either positive or negative (for example: T#1s=L#1 000, T#24d20h31m23s647msw = L#214748647).

**DATE**

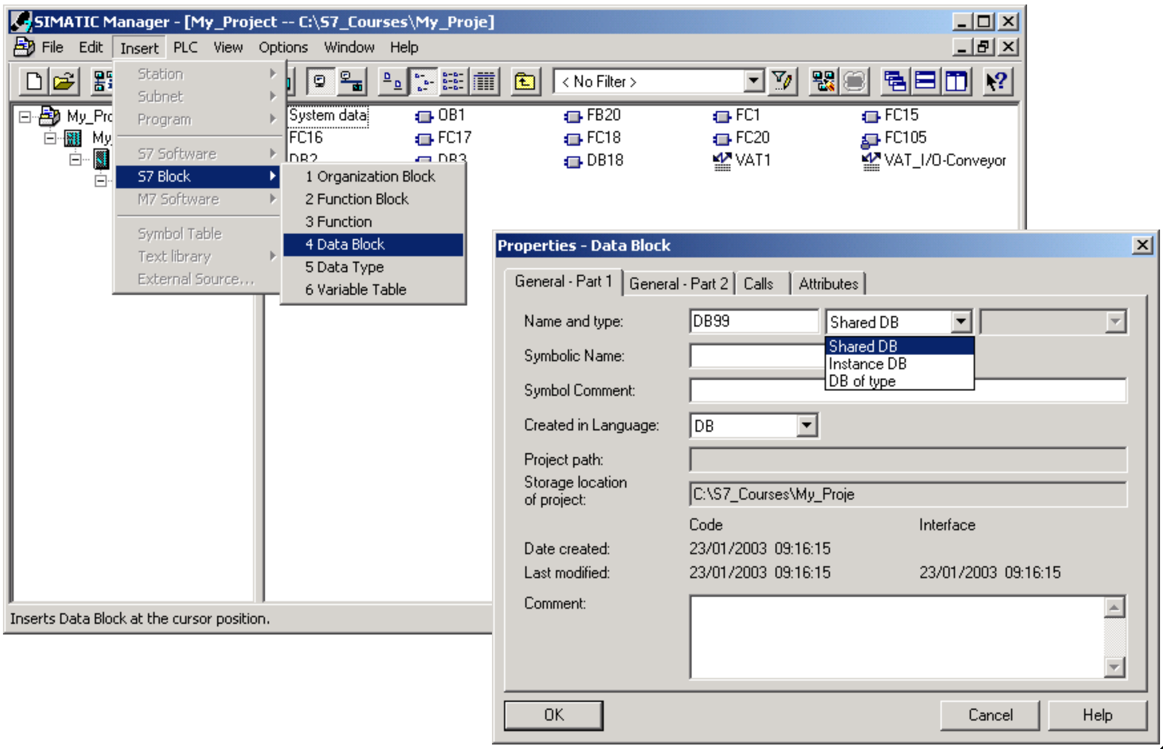
A variable of data type DATE is stored in a word in the form of an unsigned integer. The contents of the variable represent the number of days since 01.01.1990 (for example: D#2168-12-31 = W#16#FF62).

**TIME_OF_DAY**

A variable of data type TIME_OF_DAY takes up a doubleword. This variable contains the number of milliseconds since the beginning of the day (0:00 o'clock) in the form of an unsigned integer. (for example: TOD#23:59:59.999 = DW#16#05265B77).

SIEMENS

## Creating a New Data Block




SIMATIC® S7


Date: 12.03.03
File: PRO1_09E.7

**SITRAIN** Training for Automation and Drives

**Creating a DB**

You can insert a new data block - as shown in the slide - in the SIMATIC® Manager by first selecting the Blocks folder of the S7 program and then following the menu options shown.

You can also create a new data block in the LAD/STL/FBD Editor using the following menu options:
*File -> New -> select Project and Project Name -> select Blocks folder of the S7 program -> Object Name: DB 99*

**Shared DB**

Shared data blocks are used to store global data. That is, for storing general data that can be accessed by every logic block (OB, FC, FB).

The user has to edit the global data blocks himself. He does so by declaring the necessary variables for saving data in the data block.

**Instance DB**

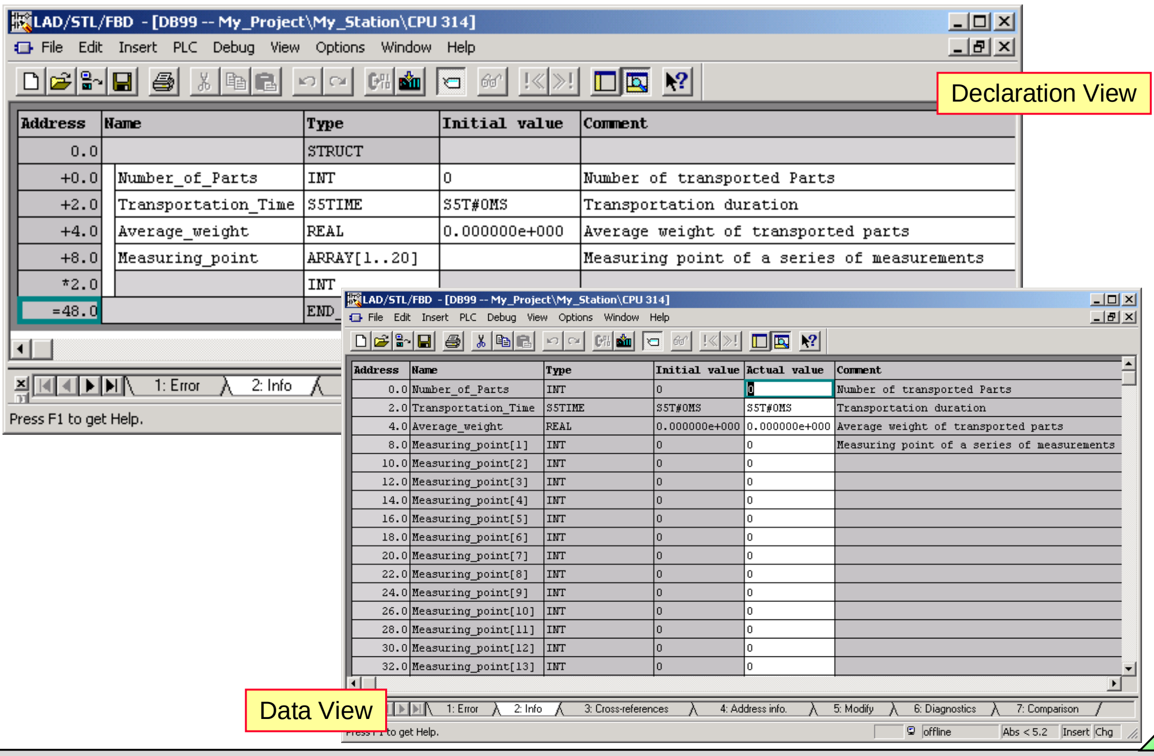
Instance data blocks are used as the "private memory area" or as the "memory" for a function block (FB). The parameters and the static variables of an FB are managed in its instance DB.

Instance data blocks are generally not edited by the user, rather they are generated by the Editor (see the Functions and Function Blocks chapter).

**DB of Type**

Data blocks can also be generated according to a <u>U</u>ser <u>d</u>efined Data <u>T</u>ype (UDT) by the Editor. A UDT, that the user must first edit like a data block, is used as a template for this. The UDT can thus also be used as a template for creating additional data blocks and/or for generally declaring variables and block parameters.

# Entering, Saving, Downloading and Monitoring a Data Block







**Declaration View** (Entering Variables)

Data blocks are edited in the "declaration view", that is, the user declares the variables needed for storing the data here. The variables are created in a table, organized in lines and columns.

**Columns**

The columns have the following meaning:

- Address - is entered by the Program Editor. It is the first byte address occupied by the variable
- Name - symbolic name of the variable
- Type - data type (INT, REAL, ....., you select this with the right mouse button)
- Initial value - used for setting a default value for a variable the first time the data block is created and/or edited. If you do not make an entry here, the Editor automatically enters the value 0.
- Comment - is used to document the variable (optional)

**Save**

You save the data block on the hard disk of the programming device using the "Diskette" icon.

**Download**

You have to download data blocks to the CPU, just as you do with logic blocks.

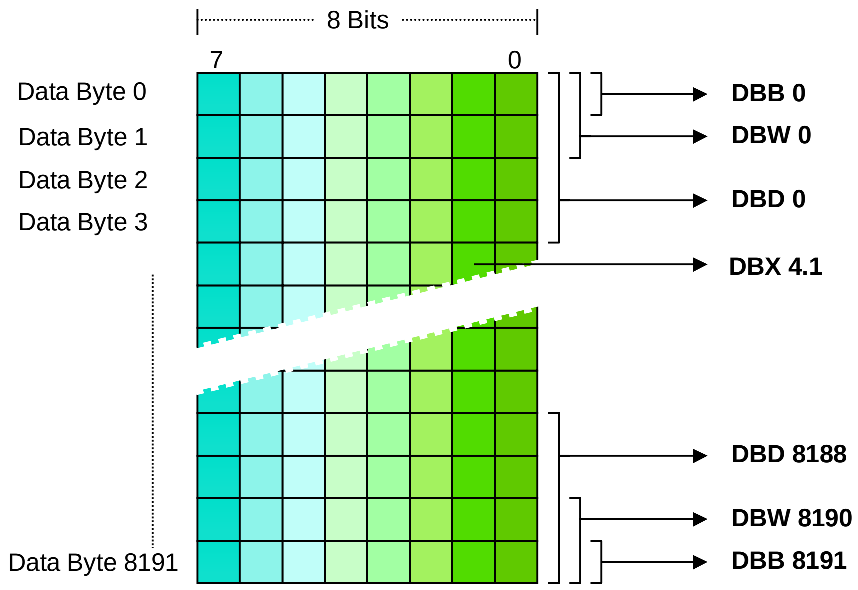
**Data View** (Monitor DB)

You can monitor online the current variable values in the data block (values of the variables in the CPU). To do so, you must first go into the "View" menu and switch to the "Data View". You can then activate the function using the "Glasses" icon.

**Initialize DB**

When you initialize a data block you overwrite the variable's current values with the initial values. This is also necessary when initial values that have been changed later on have to be accepted as actual values.
*View -> Data View -> Edit -> Initialize Data Block*

## Addressing Data Elements




SIMATIC® S7


Date: 12.03.03
File: PRO1_09E.9

SITRAIN Training for Automation and Drives

**General**

You address the data elements of a data block byte-by-byte, just as you do bit memories.
You can load and transfer data bytes, data words or data doublewords. When using data words, you specify the first byte address (such as L DBW 2) with the operation and two bytes are loaded beginning with this first byte of this address. With doublewords, four bytes are loaded beginning with the first byte address that you enter.

**Number, Length**

The number of data blocks available depends on the CPU you use.
The maximum block length is 8KByte for the S7-300™ and 64KByte for the S7-400™.

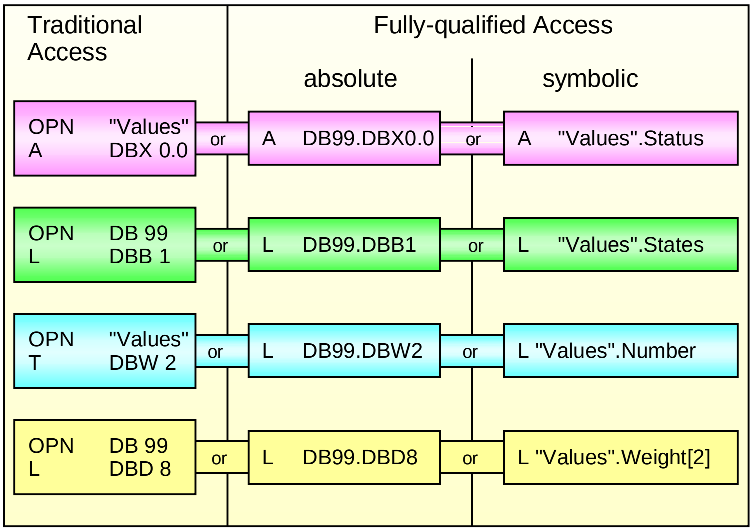
**Note**

If you access non-existent data elements or data blocks an Area Length error System Fault will occur. The CPU goes into the Stop mode if you did not program an error OB.

## Accessing Data Elements

DB 99 "Values"

| Add | Name | Type |
|---|---|---|
| 0.0 | Status | BOOL |
| 1.0 | States | BYTE |
| 2.0 | Number | INT |
| | | |
| 4.0 | Weight[1] | REAL |
| | | |
| | | |
| | | |
| 8.0 | Weight[2] | REAL |
| | | |
| | | |
| | | |




SIMATIC® S7


Date: 12.03.03
File: PRO1_09E.10

**SITRAIN** Training for Automation and Drives

**Traditional Access**

In the traditional (typical in the S5 world) data block access, data blocks have to be opened explicitly before the actual access. This can take place absolutely or symbolically with the OPN *DB 99* or *OPN "Values"* instruction (see example). If another data block was open, the data block that was open first is automatically closed. Then, the individual data elements can be accessed bit-by-bit (DBX...), byte-by-byte (DBB...), word-by-word (DBW...) or doubleword-by-doubleword (DBD...) without a data block having to be specified each time.

*Disadvantages:*

- When accessing data elements, you have to make sure that the correct data block is open.
- Access can be absolute only. The programmer must make sure that he "reaches" the correct value in the data block. If DBW3 in the example were loaded, then neither the value of the *Number* nor *Weight[1]* variables would be loaded, but an invalid value.
- Absolute accesses hamper correction possibilities and make the program difficult to read.

**Fully-qualified Access...**

A fully-qualified access is the opening of a data block which closes any previously opened DBs. A fully-qualified access can be made absolutely and symbolically.

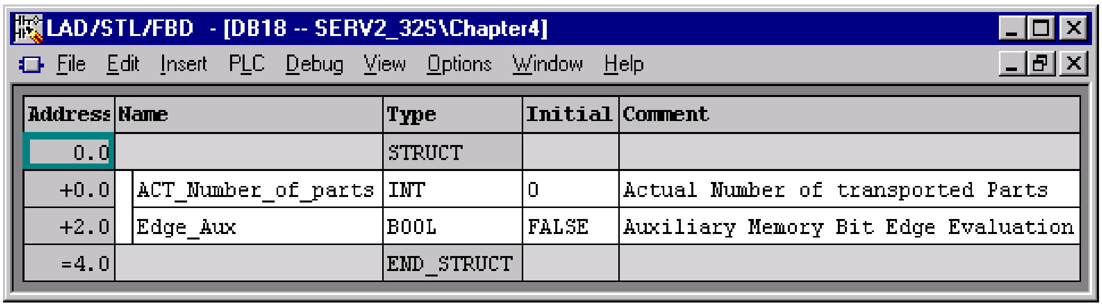
**...absolute**

An absolute access is the opening of the data block and access of the data element in combination with an instruction. Disadvantages are similar to those of the traditional access.

**...symbolic**

A symbolic access of a variable in a data block is possible only if the data block and its elements are both accessed symbolically. The Editor does allow the "mixing" of absolute and symbolic addresses during editing, however, it switches over to completely symbolic after the entry has been confirmed.

# Exercise: Counting the Transported Parts (FC 18)




| Address | Name | Type | Initial | Comment |
|---|---|---|---|---|
| 0.0 | | STRUCT | | |
| +0.0 | ACT_Number_of_parts | INT | 0 | Actual Number of transported Parts |
| +2.0 | Edge_Aux | BOOL | FALSE | Auxiliary Memory Bit Edge Evaluation |
| =4.0 | | END_STRUCT | | |

SIMATIC® S7


Date: 12.03.03
File: PRO1_09E.11

SITRAIN Training for Automation and Drives

**Function till Now : in FC 18**

- The parts transported in AUTO mode are counted (through addition in MW 20, auxiliary memory bit for edge evaluation M 18.0), as soon as they reach the Conveyor End or have passed through the light barrier.
- The *ACTUAL number of parts* of transported parts are shown on the BCD digital display.
- You can set how many parts are to be transported - *SETPOINT-number of parts* - using the BCD thumbwheel button. If the preset *SETPOINT-number of parts* is reached, it is indicated on the red LED (Q 20.4 / Q 8.4) at the Conveyor's End.
- As long as the message "SETPOINT number of parts reached" (red LED) exists, no other transport function can be started (lock-out in FC 16).
- The ACTUAL number of parts (MW 20) is set to 0 when the system is switched off or when the message "SETPOINT number of parts reached" is acknowledged through the pushbutton at the Conveyor End.

**Task:**

- The functionality programmed in FC 18 is to remain unchanged. However, instead of using MW 20 for counting by addition, use the variable ACT_Number_of_parts (INT) that is to be declared in DB 18.

  Instead of using the auxiliary memory bit M 18.0 for the edge evaluation necessary for counting, use the variable *Edge_Aux* (BOOL), that is also to be declared in DB 18.

**What to Do**:

1. Edit the DB 18 data block (see slide) with the variables *ACT_Number_of_parts* (INT) and *Edge_Aux* (BOOL) and download them into the CPU.
2. Give DB 18 the symbolic name "DB_Parts" in the global symbol table
3. Make the changes to FC 18 as described in the Task. Use fully qualified, symbolic accesses!

## Complex Data Types

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

SIMATIC® S7


Date: 12.03.03
File: PRO1_09E.12

SITRAIN Training for Automation and Drives

**Complex Data Types**

Complex data types (arrays and structures) consist of groups of elementary or complex data types.

They enable you to create data types with which you can structure large quantities of data and process it symbolically.

Complex data types (longer than 32 bits), cannot be processed with STEP 7 instructions all at once. Only one element at a time can be processed.

Complex data types are predefined. The data type DATE_AND_TIME has a length of 64 bits. The lengths of the data types ARRAY, STRUCT and STRING are defined by the user.

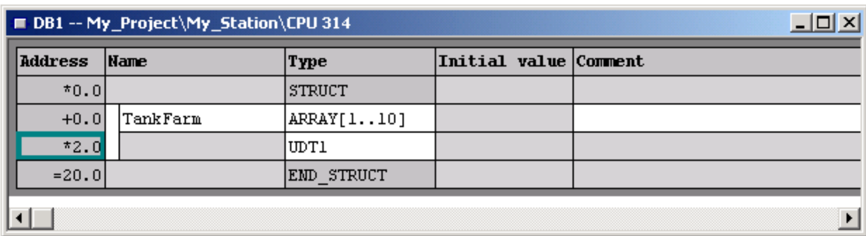
Variables for complex data types can be declared only in global data blocks and as parameters or local variables of logic blocks.

**User-Defined Data Type**

User-defined data types represent a self-defined structure. This structure is stored in UDT blocks (UDT1 to UDT65535) and can be used as a "template" in another variable‘s data type.

You can save typing time when you input a data block if you need the same structure several times.

Example: You need the same structure 10 times in a data block. First, you create a UDT, define the structure and save it (for example as UDT1). Then you create a global DB and define a variable(ex "Tank_Farm") whose type is ARRAY [1..10]. On the next row you reference the UDT you defined earlier.


DB1 -- My_Project\My_Station\CPU 314

| Address | Name | Type | Initial value | Comment |
|---|---|---|---|---|
| *0.0 | | STRUCT | | |
| +0.0 | TankFarm | ARRAY[1..10] | | |
| *2.0 | | UDT1 | | |
| =20.0 | | END_STRUCT | | |

This creates 10 data ranges in one data block. Each data range has the structure defined in UDT 1.

## 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 | | |

SIMATIC® S7



Date: 12.03.03
File: PRO1_09E.13

SITRAIN Training for Automation and Drives

**Array** An array consists of several elements of the same data type. In the slide above, you can see the "Measuring_point" array with 10 elements of the REAL data type.
Later, various measured values are to be stored in this array.

**Define Array in DB** The keyword for an array is "ARRAY[n..m]". The first element (n) and the last element (m) are specified in the square brackets. In the example, [1..10] means 10 elements, whereby the first element is addressed with the index [1] and the last with the index [10]. Instead of [1..10] you could, for example, define [0..9]. The first element would index [0] and the last element [9].

**Initial Values** A single value entered provides the value for the first element only. Values, separated by commas, provide the values in sequence. The formulea *x*(initial value) inserts the initial value *x* times in sequence.

**Data View** To see the actual values stored in the individual elements, you select the menu option *View -> Data View* to switch to another display. In "Data View", you will find the values currently stored in the column "Actual Value".

DB2 -- My_Project\My_Station\CPU 314

| Address | Name | Type | Initial value | Actual value | Comment |
|---|---|---|---|---|---|
| 0.0 | Measuring_point[1] | REAL | 0.000000e+000 | 1.000000e+002 | |
| 4.0 | Measuring_point[2] | REAL | 0.000000e+000 | 1.023000e+002 | |
| 8.0 | Measuring_point[3] | REAL | 0.000000e+000 | 1.035000e+002 | |
| 12.0 | Measuring_point[4] | REAL | 0.000000e+000 | 1.067000e+002 | |
| 16.0 | Measuring_point[5] | REAL | 0.000000e+000 | 1.052000e+002 | |
| 20.0 | Measuring_point[6] | REAL | 0.000000e+000 | 1.050000e+002 | |
| 24.0 | Measuring_point[7] | REAL | 0.000000e+000 | 1.055000e+002 | |
| 28.0 | Measuring_point[8] | REAL | 0.000000e+000 | 1.055000e+002 | |
| 32.0 | Measuring_point[9] | REAL | 0.000000e+000 | 1.079000e+002 | |
| 36.0 | Measuring_point[10] | REAL | 0.000000e+000 | 1.079900e+002 | |

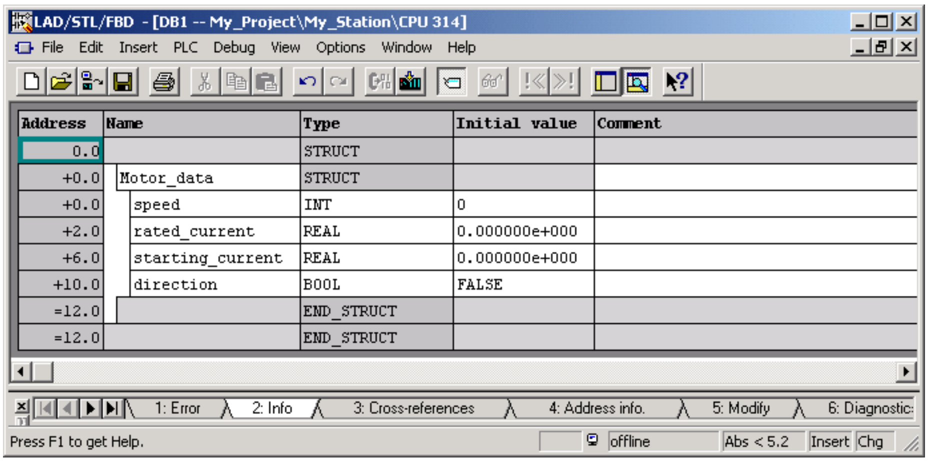
# Example of a Structure

**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):**




| Address | Name | Type | Initial value | Comment |
|---|---|---|---|---|
| 0.0 | | STRUCT | | |
| +0.0 | Motor_data | STRUCT | | |
| +0.0 | speed | INT | 0 | |
| +2.0 | rated_current | REAL | 0.000000e+000 | |
| +6.0 | starting_current | REAL | 0.000000e+000 | |
| +10.0 | direction | BOOL | FALSE | |
| =12.0 | | END_STRUCT | | |
| =12.0 | | END_STRUCT | | |

SIMATIC® S7


Date: 12.03.03
File: PRO1_09E.14

**SITRAIN** Training for Automation and Drives

**Structure**

The slide shows an example of a structure namee "Motor_data". The structure consists of several elements of different data types. The individual elements of a structure can be elementary or complex data types.

The access to the individual elements of a structure contains the structure name. This makes the program easier to read.

In order to be able to access the elements symbolically, the data block must be given a symbol name, for example, "Drive_1".

Example: accessing elements of a structure using the load command "L"

L "Drive_1".Motor_data.rated_current
L "Drive_1".Motor_data. speed

The format is: *block symbolic name, dot, structure name, dot, element name*.

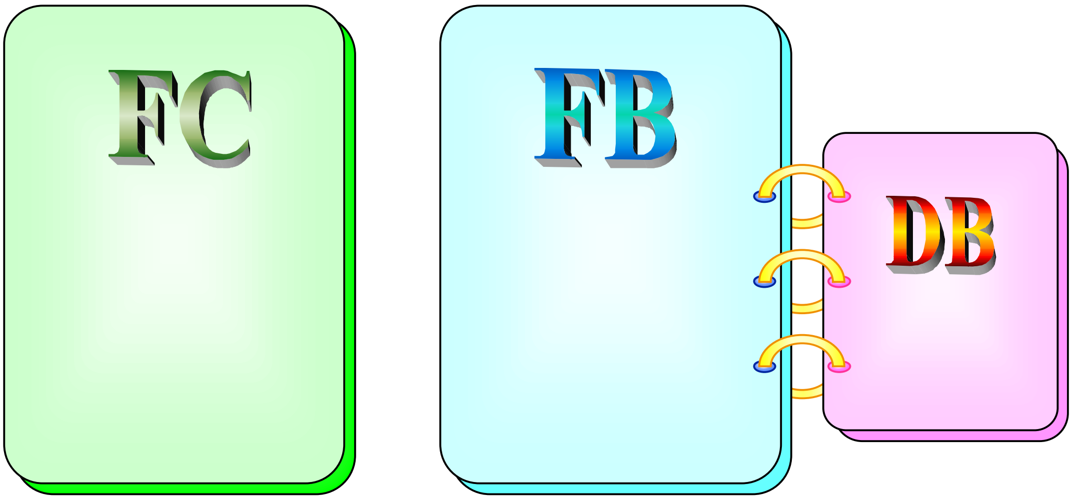
Note that the symbolic block name ("Drive_1") is enclosed in quotations, indicating the name is from the global symbol editor. The structure name and element names are not enclosed in quotations, because they are symbols defined in the data block and are not listed in the global symbol editor.

**Define Structure in DB**

The keyword for a structure is "STRUCT". The end of a structure is indicated by "END_STRUCT". A name is defined for the structure (in the example: "Motor_data").

SIEMENS

# Functions and Function Blocks




SIMATIC® S7



Date: 12.03.03
File: PRO1_10E.1

SITRAIN Training for Automation and Drives

**Contents** Page

## Objectives

**Upon completion of this chapter the participant will ...**

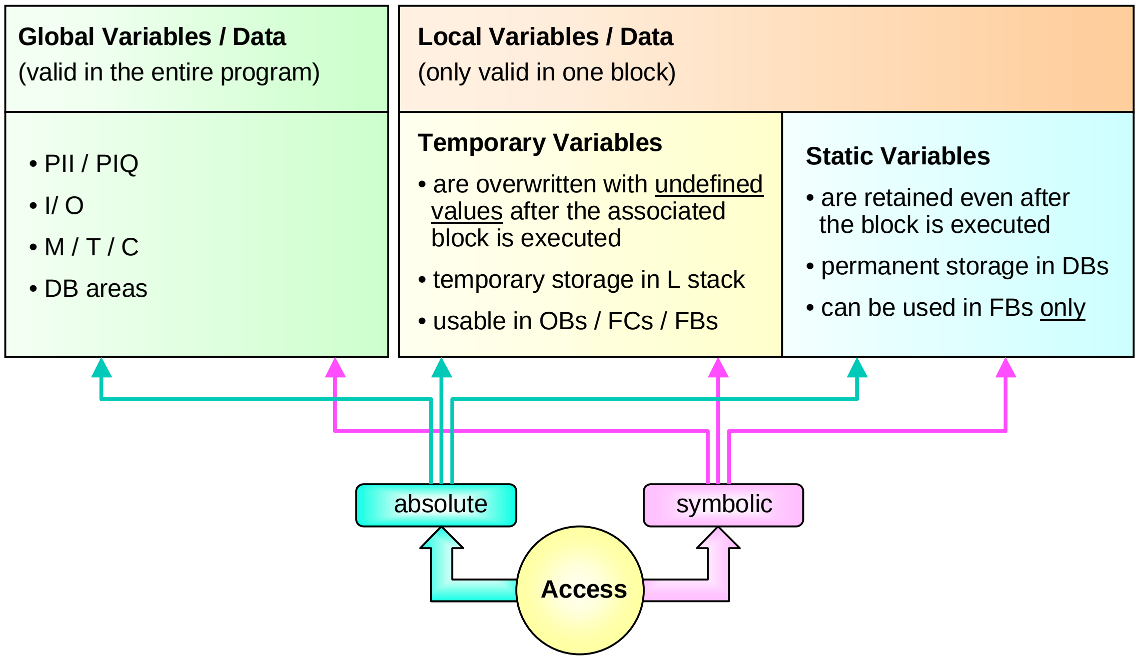
- ... be familiar with the purpose of temporary variables
- ... be able to declare temporary variables and use them in the program
- ... be familiar with the purpose of parameter-assignable blocks
- ... be able to program parameter-assignable functions and their calls
- ... know the difference between functions (FCs) and function blocks (FBs)
- ... be familiar with the instance model and the multi-instance model
- ... 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 their calls

SIMATIC® S7


Date: 12.03.03
File: PRO1_10E.2

SITRAIN Training for Automation and Drives

## Introduction




SIMATIC® S7



Date: 12.03.03
File: PRO1_10E.3

**SITRAIN** Training for Automation and Drives

**General**

Up until now, the inputs and outputs in our programs were coded directly with their actual address assignments. This type of programming is fixed to these address locations and is therefore not well-suited for repetitive processes.

Blocks that are not parameterized are best used for custom machinery where there is no repetition in the process.

For frequently reoccurring functions it is better to make reusable, parameter-assignable blocks (Functions, FCs and Function Blocks, FBs). These blocks use symbolic input and output parameters (local symbol names), which are supplied with actual operands when the block is called.

You have to assign these operands when you make a call to a Function or a Function Block. The program logic of the FC/FB remains unchanged and you can therefore reuse the logic several times.

**Local Variables**

Up until now, you used global variables (bit memories and data blocks) to save production data, for example. In this chapter you will find out more about data storage in local variables.
Local variables can be read only by the block in which they were originally created. Because of this, local variables cannot be used as data interfaces between different program blocks.

There are two types of local variables: Temporary and Static.

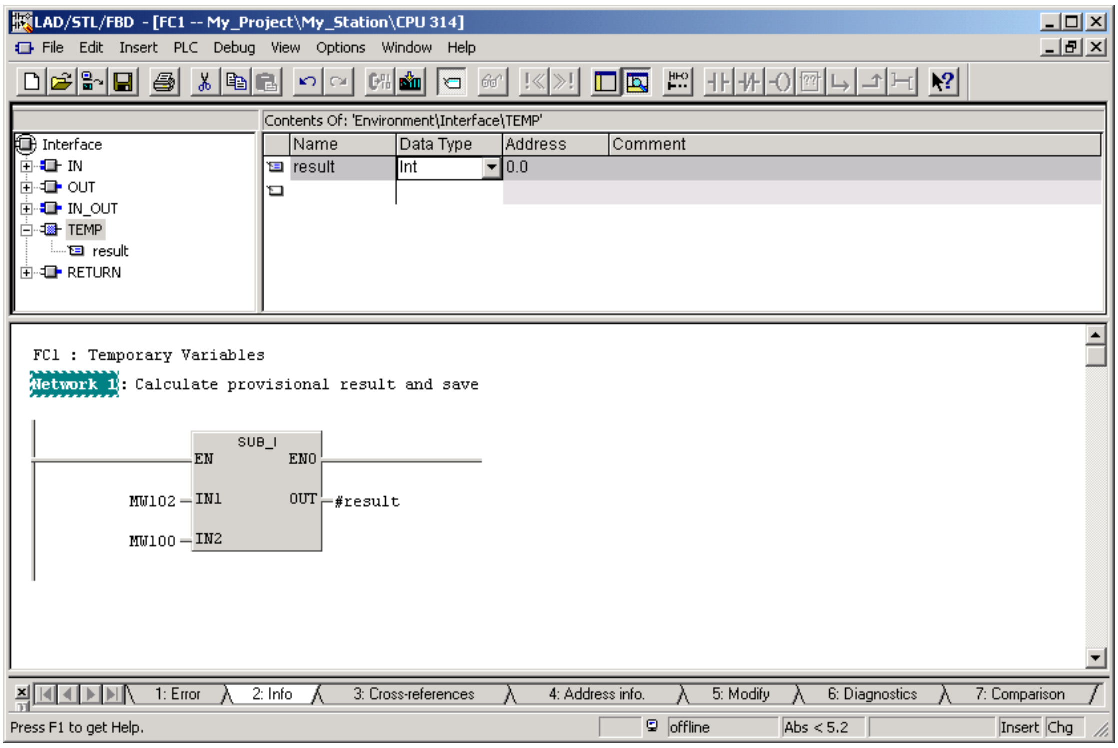
**Temporary Variables**

Temporary variables are variables that are stored only while the block is being executed. They can be declared in all blocks (OB, FC, FB).

**Static Variables**

If the data are to be retained even after the block is executed, the data must be stored in static variables.
Static variables can only be declared in function blocks. The instance DB assigned to the FB is used as the storage location for these static variables.

## Temporary Variables




SIMATIC® S7


Date: 12.03.03
File: PRO1_10E.4

**SITRAIN** Training for Automation and Drives

**General**

Temporary variables can be used in all blocks (OB, FC, FB). They are used to temporarily store information while the block is being executed. The data are lost when the block is exited.

The data are stored in the L stack (local data stack). The L stack is a separate memory area in the CPU.

**Declaration**

Before a temporary variable can be used in a block, it must be declared in the block's declaration table. In the "temp" row you enter a variable name and the associated data type. You cannot predefine an initial value for temporary variables.

After you complete a "temp" entry row, by pressing the "Return" key, a new "temp" line is added after it. The L stack absolute address is assigned by the system and displayed in the "Address" column for the line that was completed.

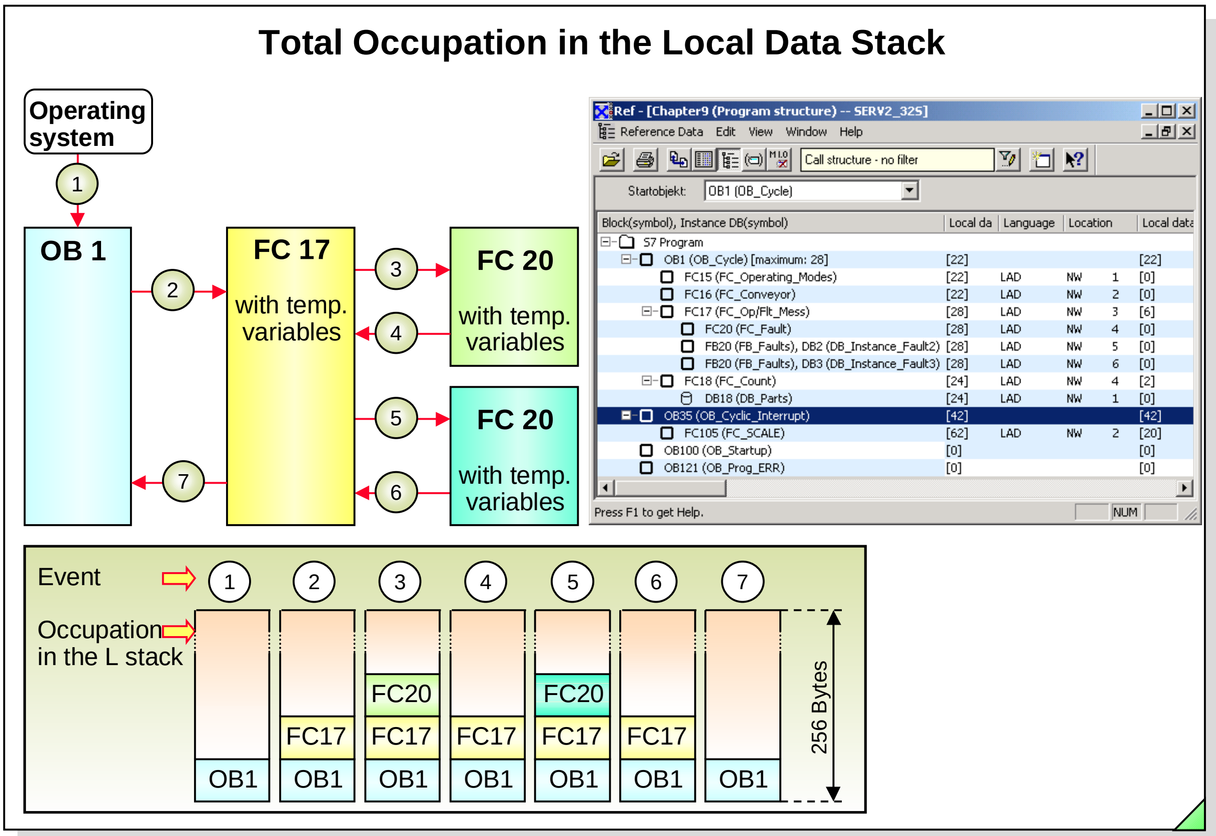
**Access**

In Network 1, you see an example of the symbolic access to a temporary variable.
The result of the subtraction is stored in the temporary variable #result.
You can also enter an absolute access (T LW0). You should, however, try to avoid this since the program is difficult to read.

**Note**
#

Unlike the global symbols from the symbol table that are displayed in the program logic with quotation marks ("symbol name"), a local symbol has a # in front of it ( #result). The # character is automatically inserted in front of the symbol name by the Editor when no " " are used and the symbol name is located in the block's declaration table. The editor checks the block's declaration table before it checks the Global Symbol Table.

SIEMENS




SIMATIC® S7


Date: 12.03.03
File: PRO1_10E.5

SITRAIN Training for Automation and Drives

**Total Occupation in Local Data Stack**

For every program processing level (such as OB1 with all the blocks nested in it), a specific area of the L stack is reserved. This area is limited in size. For example, the CPU 314 has 256 bytes available in its L stack. That means that for all the local variables from OB1 and all the nested blocks which are called in OB1, there are 256 bytes available.

You can display the number of bytes an entire program requires in the local data stack with the "Reference Data" tool. You will become familiar with this tool in the chapter "Troubleshooting". The total occupation of the local data stack and the number of bytes required per call path is displayed on the screen.

**Activate Reference Data**

In the SIMATIC® Manager you highlight the block folder and then select the menu options:
*Options -> Reference Data -> Display -> Program Structure.*

**Note**

If the maximum number of local data is exceeded during program execution in the CPU, the CPU goes into the Stop mode. "STOP caused by error when allocating local data" is entered as the cause of error in the diagnostics buffer.

## Local Data Stack Size

Entire size:
**1.5 Kbyte**
(CPU 313..316)

| Execution | | For S7-300™: Priority class | For S7-300™: L stack size |
|---|---|---|---|
| Startup (one-time execution) | | 27 | 256 bytes |
| Cyclic execution | | 1 | |
| Time-controlled execution | Time-of-Day Interrupt | 2 | 256 bytes |
| | Time-Delay Interrupt | 3 | 256 bytes |
| | Cyclic Interrupt | 12 | 256 bytes |
| Event-driven execution | Hardware Interrupt | 16 | 256 bytes |
| | Error handling in startup | 28 | 256 bytes |
| | Error handling in scan cycle | 26 | |

SIMATIC® S7

Date: 12.03.03
File: PRO1_10E.6
SITRAIN Training for Automation and Drives

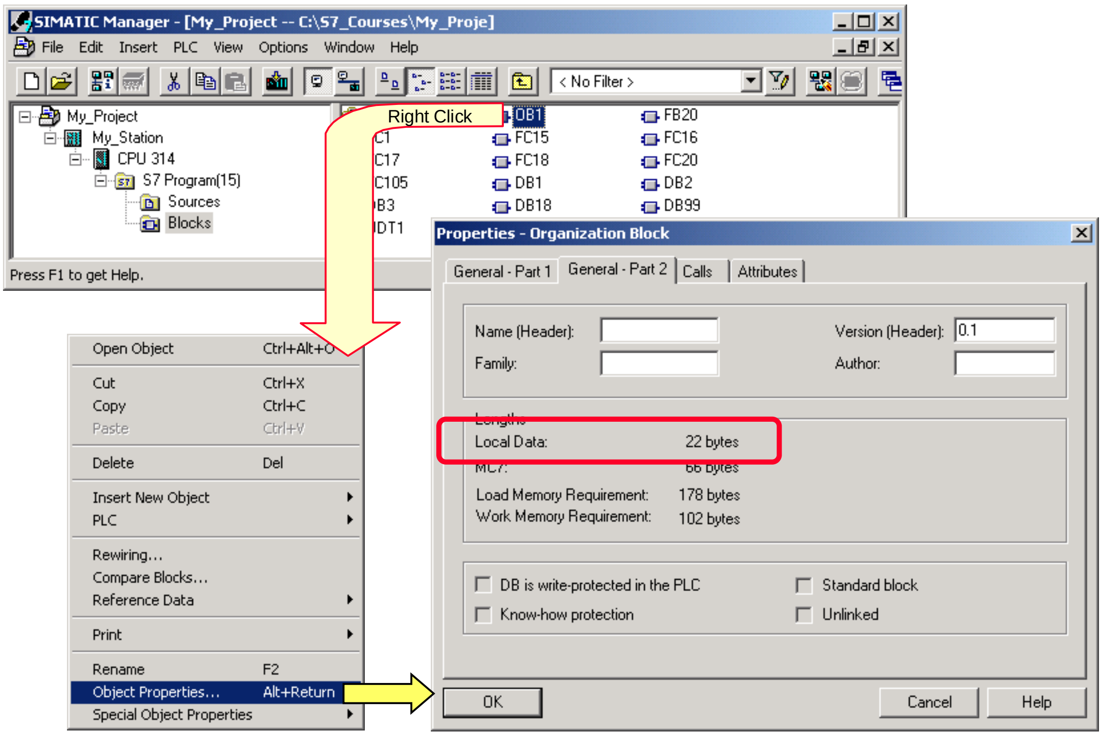
**Local Data Stack** The local data stack (L stack) is a memory area that contains the temporary variables (replacement for scratchpad memories in SIMATIC® S5) of the blocks.

**Local Data Stack Size** When the operating system calls an OB, an L stack area of 256 bytes is opened up while the OB and the blocks called in it are executed.
Every priority class is assigned 256 bytes.
The L stack of the 313..316 CPUs has a total of 1536 bytes (1.5kByte).

**Priority Classes** There are a total of eight priority classes with the S7-300™. However, no more than 6 priority classes can be active at the same time. If, for example, OB 100 is active (with priority class 27), then OB1 (priority class 1) can never be active. Furthermore, the error OBs 80 to 87 for asynchronous errors can only then have priority class 28, if the fault occurs in the startup program. In other words, when they interrupt OB100. More information can be found in the "Organization Blocks" chapter.

**S7-400™** With the S7-400™ CPUs, you can decide what the size of the local data stack is for the individual priority classes (Tool: HW Config.).
You can deselect the priority classes which you do not need. That way, you can make more local data available to the other priority classes.

## Byte Requirement of a Block in the Local Data Stack






**Displaying the Byte Requirement**

You can see the exact number of bytes a block requires in the local data stack by going into the block properties.

**Activate**

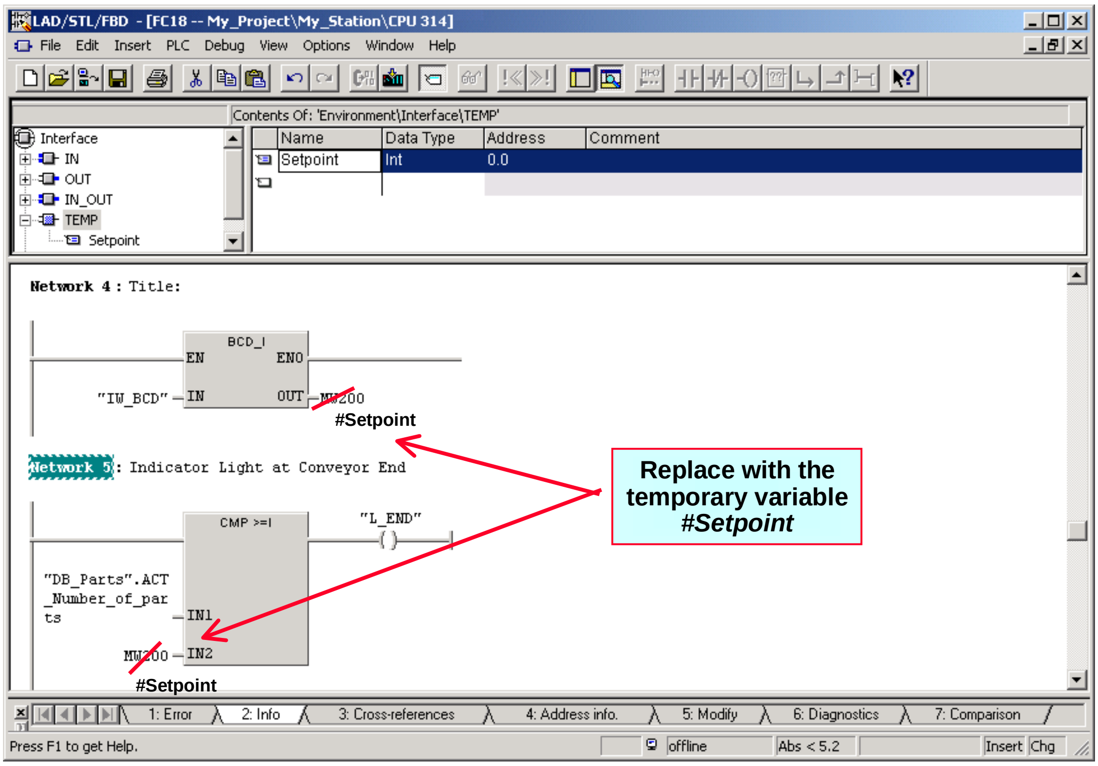
1. In the SIMATIC® Manager, select the block with the right mouse button and then -> *Object Properties.* or
2. In the SIMATIC ® Manager, select the block with the left mouse button and then the menu option *Edit -> Object Properties.*

**Notes**

The sum of local data for an execution level (OB) is a maximum of 256 bytes with the S7-300™. Every OB itself always takes up 20 or 22 bytes. This means that a maximum of 234 bytes can be used in an FC or FB.

If more than 256 bytes of local data are defined in a block, the block cannot be downloaded into the CPU. The transmission is interrupted with the error message "The block could not be copied". Within this error message is a "Details" button. If you click on it, a message box appears with the explanation "Incorrect local data length".

## Exercise: Use of Temporary Variables




SIMATIC® S7


Date: 12.03.03
File: PRO1_10E.8

SITRAIN Training for Automation and Drives

**FC 18 Up Till Now:**

- The parts transported in AUTO mode are counted (by addition in the variable "DB_Parts".ACTUAL_quantity), as soon as they reach the Conveyor End or have passed through the light barrier.
- The number of transported parts (*ACTUAL quantity*) is displayed on the BCD digital display.
- You can set the *SETPOINT quantity,* of how many parts are to be transported, using the BCD thumbwheel button. When the given *SETPOINT quantity* is reached, it is displayed on "L_ACTUAL=SETPOINT" (red LED Q 20.4 / Q 8.4) at the Conveyor End.
- The number of transported parts (variable "DB_Parts".ACTUAL_Quantity) is reset through the momentary contact switch at the Conveyor End when the system is switched off (Q 8.1 / 4.1 = ´0´) or when the message "L_ACTUAL=SETPOINT" (red LED) is acknowledged.
- As long as the message "L_ACTUAL=SETPOINT" (red LED) exists, no other transport function can be started (lock-out in FC 16).

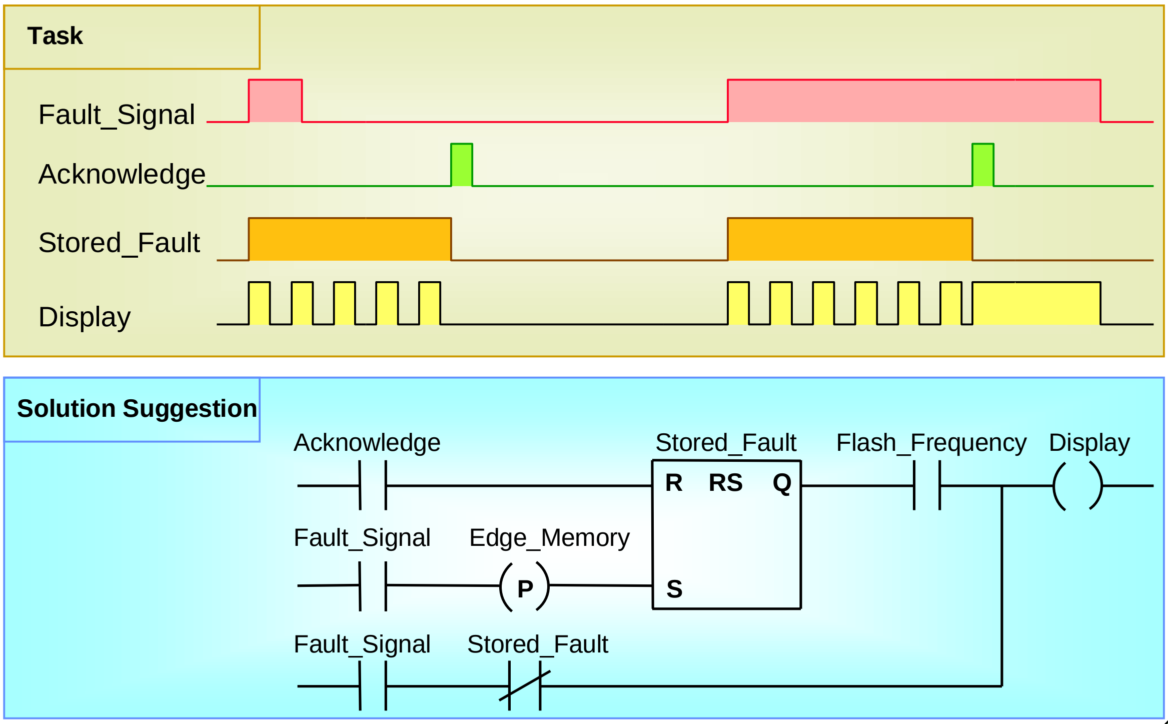
**Task:**

The functionality programmed in FC 18 is to remain unchanged. However, use the local, temporary variable *Setpoint* for the intermediate storage of the *SETPOINT quantity* converted from BCD to INT.

**What To Do**:

- In FC 18, declare the temporary variable *Setpoint* as an INT type.
- Replace the auxiliary memory marker used for intermediate storage with the newly declared *Setpoint* variable.
- Download and monitor the program.

## Example: Fault Signal Displayed with Output LED




SIMATIC® S7



Date: 12.03.03
File: PRO1_10E.9

SITRAIN Training for Automation and Drives

**Example**

Based on the above problem, we would now like to show a practical example for reusable logic. This solution will show how to create a parameter-assignable block and call it from our program.

**Description**

An LED on the operator console will display faults that occur. When the fault occurs, the LED will flash at 2Hz. If the Acknowledge pushbutton is pressed after the fault has cleared, the LED will stop flashing and turn off. If the fault still exists when it is acknowledged, the LED will stop flashing and remain on steady until the fault is corrected. When the fault is removed after it has been acknowledged the LED will turn off.

**Program**

An RLO edge detection of the fault signal is also carried out, since the memory (Stored_Fault) would otherwise immediately be set again when an existing problem is acknowledged.

If the Stored_Fault is set (message has not yet been acknowledged), the upper branch causes the LED to flash. The flashing LED is generated using the memory bit M10.3 gated with the Stored_Fault. Memory byte MB 10 was defined as a clock memory when parameter assignment was made in the CPU using the Hardware Configuration.

The lower branch is used to cause a steady Display for a fault that is acknowledged but not yet corrected.

**Call**

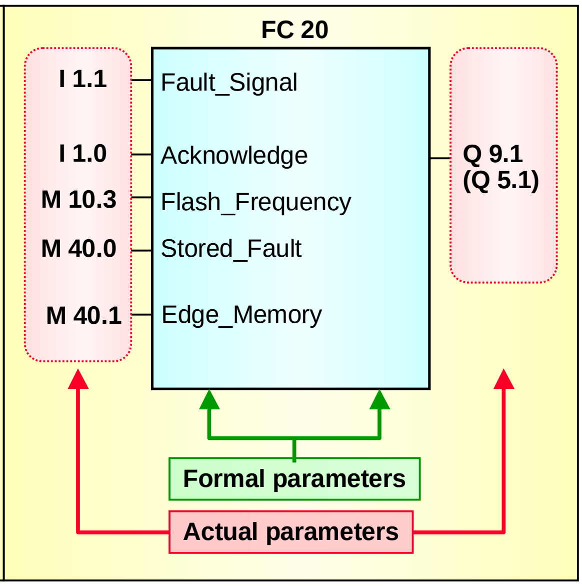
In our program we will be calling this fault routine multiple times. The function described above will be used to detect different faults. Notice this kind of programming can save us some work. Instead of typing the entire set of logic twice, we have only to write the subroutine once. Then we can call the subroutine as many times as we wish.

SIEMENS

# Parameter-assignable Blocks

| Solution with *non-parameter-assignable* block | Solution with *parameter-assignable* block: STL solution for FC 20 |
|---|---|
| A( | A( |
| A I 1.0 | A #Acknowledge |
| R M 40.0 | R #Stored_Fault |
| A I 1.1 | A #Fault_Signal |
| FP M 40.1 | FP #Edge_Memory |
| S M 40.0 | S #Stored_Fault |
| A M 40.0 | A #Stored_Fault |
| ) | ) |
| A M 10.3 | A #Flash_Freq. |
| O | O |
| A I 1.1 | A #Fault_Signal |
| AN M 40.0 | AN #Stored_Fault |
| = Q 9.1 | = #Display |

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




SIMATIC® S7


Date: 12.03.03
File: PRO1_10E.10

SITRAIN Training for Automation and Drives

**Application**

You can program parameter-assignable blocks for frequently recurring program functions. This has the following advantages:

- the program only has to be created once, which significantly reduces programming time.
- the block is only stored in the user memory once, which significantly reduces the amount of memory used.
- the block or the functionality implemented with the block can be called as often as you like, each time with a different address. For this, the formal parameters (input, output, or in/out parameters) are supplied with different actual operands every time they are called.

**Program Execution**

Statement List Language is shown in the above example as it is easier to follow the program execution. The STL program code above performs the same fault logic as the previous example. Statement List programming language will be discussed further in a more advanced class.

When the block shown above is executed and the statement *"A #Acknowledge"* is evaluated, the parameter Acknowledge is replaced by the actual parameter given during the call. If the input I 1.0 is given as the actual parameter for the parameter *Acknowledge,* then the statement "A I 1.0" is evaluated instead of the logic you see in the FC20 program block, "A #Acknowledge" .

**Parameter-assignability**

You can program FC or FB blocks as parameter-assignable. You cannot program organization blocks as parameter-assignable since they are called directly by the operating system. As no block call takes place in the user program, it is not possible to pass actual operands.

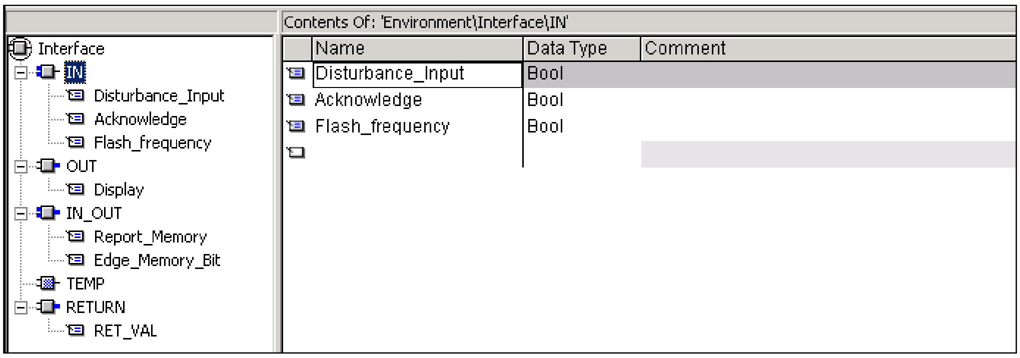
**Our Example**

Even if the fault subroutine is required twice in the system, you only have to program FC 20 once as parameter-assignable. The FC 20 is then called twice for the two different faults and is assigned a different actual address each time.

# Declaring the Formal Parameters in FC 20

**Formal parameters**

| Type of parameter | Declaration | Use | Graphic Display |
|---|---|---|---|
| Input parameter | in | Read only | To the left of the block |
| Output parameter | out | Write only | To the right of the block |
| In/out parameter | In_out | Read / write | To the left of the block |




SIMATIC® S7


Date: 12.03.03
File: PRO1_10E.11

**SITRAIN** Training for Automation and Drives

**Formal Parameters**

Before you can create the program for the parameter-assignable block, you have to define the formal parameters in the declaration table.

**Type of Parameter**

In the table in the slide, you can see the three different parameter types that are available for use in the block. It is up to the programmer to select the declaration type for each formal paramater. The ‘in‘ declaration type should be assigned only to declaration types that will be read for instructions in the subroutine. Use the ‘out‘ declaration type for paramaters that will be written to within the function. Please make sure that formal parameters that have a reading access (queried with the operations A, O, L) and a writing access (assigned with the operations S, R, T) are declared as ‘in/out‘ parameters.

**Interface**

The interface of a block forms the IN, OUT, and IN_OUT parameters. The RETURN parameter is a defined, additional OUT parameter that has a specific name according to IEC 61131-3. This parameter only exists in FCs in the interface.
The TEMP variables are - even though they are listed under “Interface“ - not components of the block interface, since they do not become visible when the block is called or that no actual parameters have to be passed for the declared TEMP variables in the calling block.
To declare the parameters and TEMP variables, the type of parameter or TEMP must be selected in “Interface“ (see lower picture). Then, in the table appearing to the right, the names can be edited with the associated data types and comments.

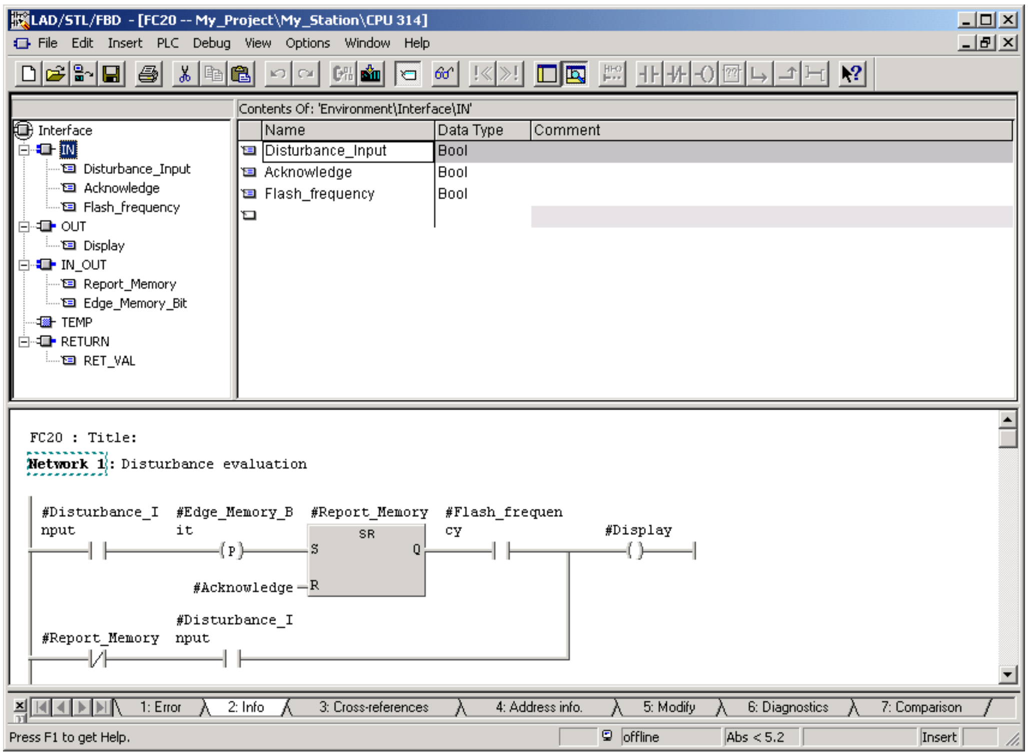
**Example FC20**

In the lower section of the slide, you can see the declaration table and/or the interface of the FC 20 block “Disturbance Input“ (see previous page). Notice that since the formal parameters #Report_Memory and #Edge_Memory_Bit are used for both reading and writing instructions in connection with the operation FP, you have to declare these as in/out parameters.

**Attention!**

The declared formal parameters (IN, OUT and IN_OUT, not TEMP) of a block are its interface to the “outside“ That is, they are “visible“ or relevant to other blocks, that call this block. If the interface of a block is changed by deleting or adding formal parameters later on, then the calls have to be updated. All blocks that call this block must be updated.

## Editing a Parameter-assignable Block




SIMATIC® S7


Date: 12.03.03
File: PRO1_10E.12

SITRAIN Training for Automation and Drives

**Notes**

It doesn‘t matter whether the names of the formal parameters are written with capital or small letters. The "#" character in front of the name is automatically inserted by the PG/PC. The character is used to indicate that the parameter is a local variable that was defined in the variable declaration table of this block.

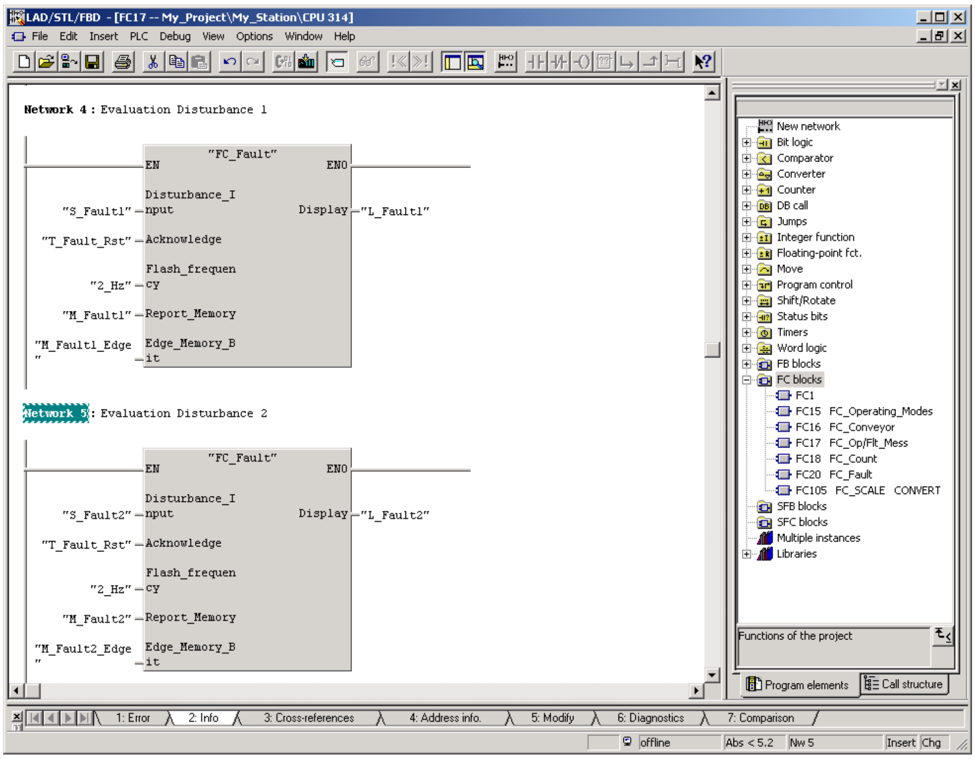
It is possible, that when you write the program in LAD / FBD, that the name is not completely displayed in one line. This depends on how you have customized the settings in the Program Editor (*Options* -> *Customize* -> "*LAD*/*FBD*" tab -> *A*ddress Field Width).

**Symbols**

1. If you use a symbolic name when you edit a block, the Editor first of all searches through the declaration table of the block.
   If the symbolic name is there, the symbol with a # in front of it is accepted in the program as a local variable. Capital and small letters may be corrected to match the way a symbol is entered in the declaration table.
2. If a symbol cannot be found as a local variable, the Editor searches through the program‘s symbol table for the global symbol.
   If the symbol is found there, the symbol is placed in quotation marks and is accepted in the program as a global variable.
3. If you specified the same symbolic name in the symbol table as well as in the local declaration table, the Editor will always insert the local variable.

   If, however, you want to work with the global symbol, you must place the symbol name in quotation marks when you make the entry.

## Calling a Parameter-assignable Block





Date: 12.03.03
File: PRO1_10E.13

SITRAIN Training for Automation and Drives

**Programming a Block Call**

The call of a parameter-assignable block can be programmed by copying the symbol of the desired block into the code section of the calling block. The drag & drop method works nicely for this purpose. This symbol is found in the "FC Blocks" or "FB Blocks" folder of the Program Elements Catalog in the LAD/FBD/STL Editor. A field with question marks then automatically appears for each formal parameter of the called block in which the actual parameters are to be entered.

**Note**

When a parameter-assignable function (FC) is called, an actual parameter must be passed for every formal parameter.

*Exception:*

In the graphic programming languages LAD and FBD, the assignment of the EN and ENO parameters, which are automatically added by the Editor, is optional. Here we are not dealing with formal parameters, rather the possibility of calling a block conditionally.

**Parameter Assignment**

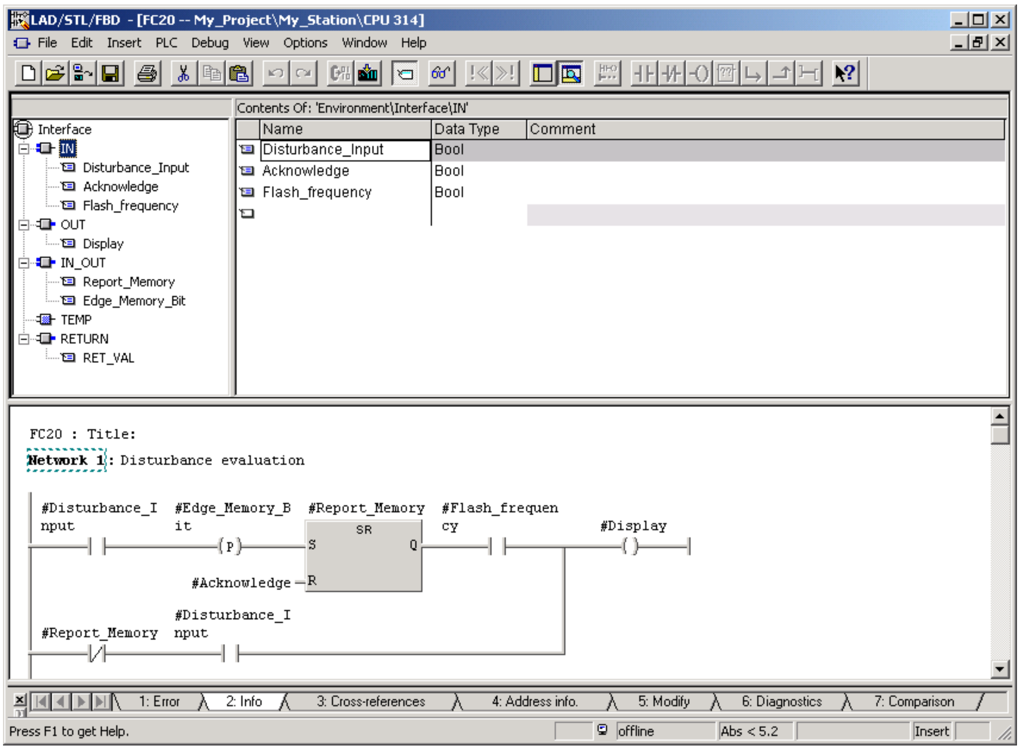
All global and local addresses whose data type corresponds to the formal parameters of the called block can be passed as actual parameters.

The actual parameters can be passed with an absolute address or with a symbolic name just as in the global symbol table or in the declaration table of the calling block.

**Passing On of Parameters**

Basically, a *passing on of parameters* is also possible. That is, formal parameters of the calling block are passed on as actual parameters to the called block. For complex data type parameters, this is however only possible with limitations. It is dealt with in greater detail in another more advanced course.

# Exercise: Editing a Parameter-assignable FC Block




SIMATIC® S7



Date: 12.03.03
File: PRO1_10E.14

SITRAIN Training for Automation and Drives

**Function of the Fault Evaluation**

When a fault occurs an output light will begin flashing at a 2 Hertz rate. When the fault is acknowledged one of two things will occur. If the fault is no longer valid the output will turn off or if it is still valid the output will change to a steady light.

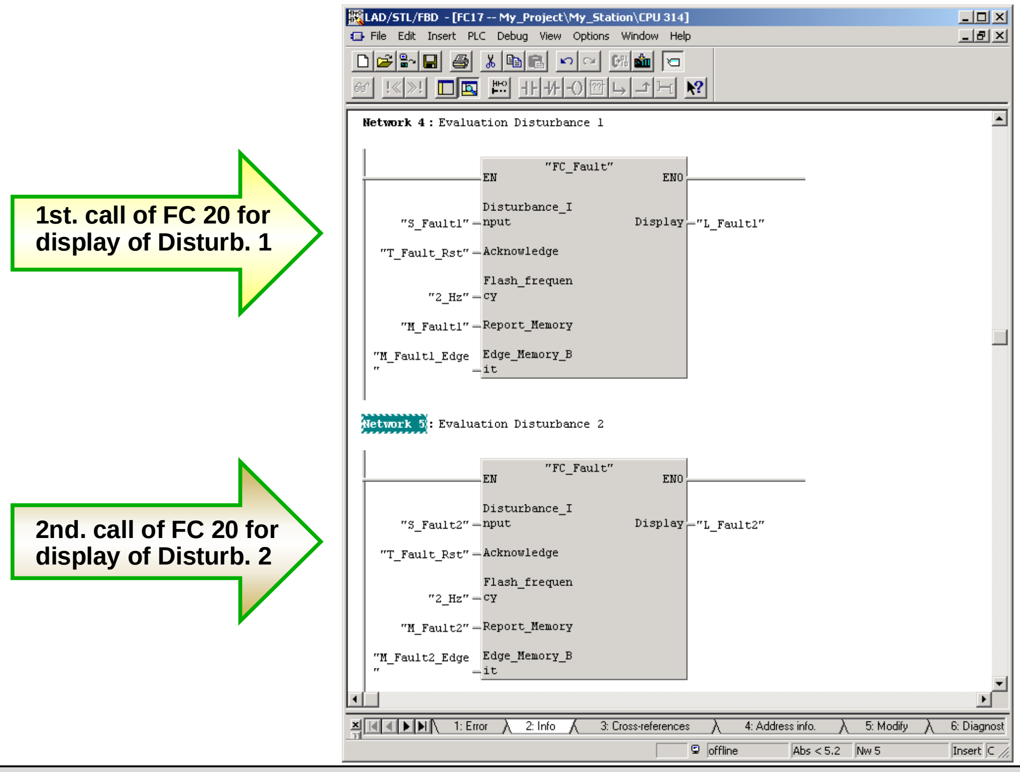
**Task**

Create a fault evaluation program in the parameter-assignable block FC 20. In the slide above, the declaration table and logic code are provided for FC 20.

**What To Do**

- Insert the FC 20 block.
- Declare the formal parameters as shown in the slide.
- Create the program in FC 20 using the declared formal parameters.
- Save the block and download it to the CPU.

## Exercise: Calling a Parameter-assignable FC Block




SIMATIC® S7


Date: 12.03.03
File: PRO1_10E.15

**SITRAIN** Training for Automation and Drives

**Task**

Two process disturbances (two switches on the simulator) are to be evaluated or displayed through an LED on the simulator. Accordingly, program two calls of FC 20 and assign parameters to it using the actual parameters shown in the slide.

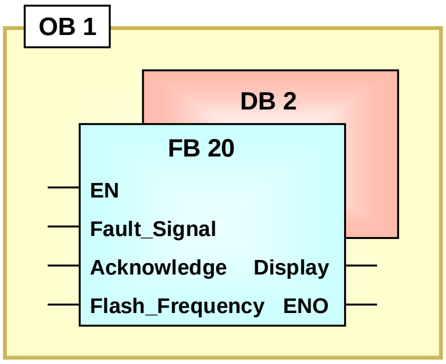
**What To Do**

- Program the FC 20 - calls in two new networks in the block *Disturbance Evaluation* FC 17
- Save the modified FC 17 and download it to the CPU
- Monitor and test the program

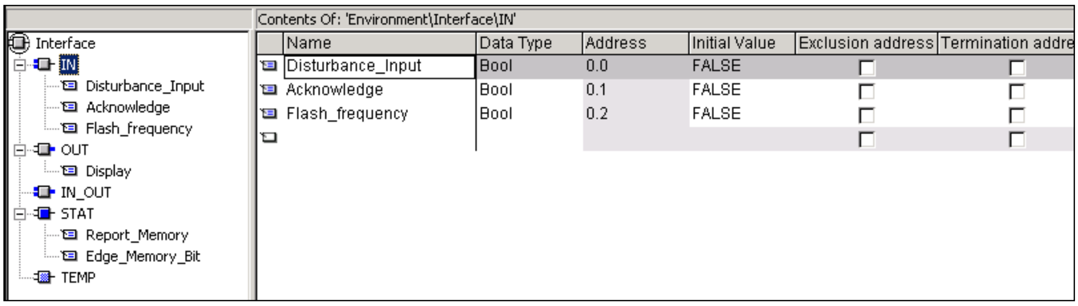
**Note**

You have parameterized the memory byte MB 10 as a clock memory byte in the CPU properties using the HW-Config tool. Should you have performed a memory reset in the meantime, you have to reload the *system data* you generated with *HW Config* into the CPU in order to make the bit memory M10.3 flash.

# Function Blocks (FBs)




Declaration table of the function block




SIMATIC® S7


Date: 12.03.03
File: PRO1_10E.16

**SITRAIN** Training for Automation and Drives

**Special Features of FBs**

Unlike functions (FCs), function blocks (FBs) have a (recall) memory. That means that a local data block is assigned to the function block, the so-called **instance data block**. When you call an FB, you also have to specify the number of the instance DB, which is automatically opened.

An instance DB is used to save **static variables**. These local variables can only be used in the FB, in whose declaration table they are declared. When the block is exited, they are retained.

**Exclusion Address and Termination Address**

By activating this option, you can assign properties to the FB parameters and static variables that are only relevant in connection with a process diagnosis.

**Parameters**

When the function block is called, the values of the actual parameter are stored in the instance data block.

If no actual parameters are assigned to a formal parameter in a block call, then the last value stored in the instance DB for this parameter is used in the program execution.

You can specify different actual parameters with every FB call.
When the function block is exited, the data in the data block is retained.

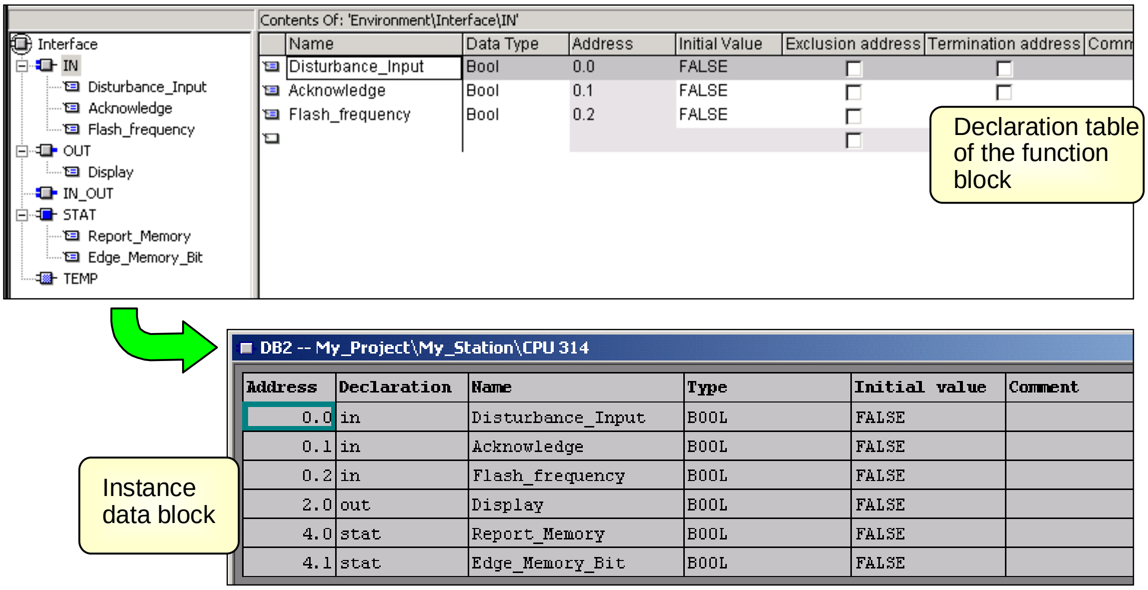
**Static Variables**

Static local variables store block specific data that are not accessed exterior to the function block. In other words the variable is not passed in or out of the block as a formal parameter.

**FB Advantages**

- When you write a program for an FC, you must search for empty bit memory address areas or data areas and you must maintain them yourself. The static variables of an FB, on the other hand, are maintained by the STEP 7 software.
- When you use static variables you avoid the risk of assigning bit memory address areas or data areas twice.
- Instead of the formal parameters "Report memory" and "Edge memory bit" of the FC20, you use the static variables "Stored_Fault" and "Edge_Memory" in the FB. This makes the block call simpler since the two formal parameters are dropped.

# Function Block for Message Display




| Name | Data Type | Address | Initial Value | Exclusion address | Termination address | Comm |
|---|---|---|---|---|---|---|
| Disturbance_Input | Bool | 0.0 | FALSE | ☐ | ☐ | |
| Acknowledge | Bool | 0.1 | FALSE | ☐ | ☐ | |
| Flash_frequency | Bool | 0.2 | FALSE | ☐ | | |
| | | | | ☐ | | |

| Address | Declaration | Name | Type | Initial value | Comment |
|---|---|---|---|---|---|
| 0.0 | in | Disturbance_Input | BOOL | FALSE | |
| 0.1 | in | Acknowledge | BOOL | FALSE | |
| 0.2 | in | Flash_frequency | BOOL | FALSE | |
| 2.0 | out | Display | BOOL | FALSE | |
| 4.0 | stat | Report_Memory | BOOL | FALSE | |
| 4.1 | stat | Edge_Memory_Bit | BOOL | FALSE | |

SIMATIC® S7


Date: 12.03.03
File: PRO1_10E.17

SITRAIN Training for Automation and Drives

**Message Display**

In an earlier exercise you created a parameter-assignable FC 20 block for displaying a message (indicating a problem).

Instead of bit memories, that were used in the FC20 to save the message signal and its RLO edge detection, you can use so-called static variables in an FB. They are stored in the instance DB referencing the FB.

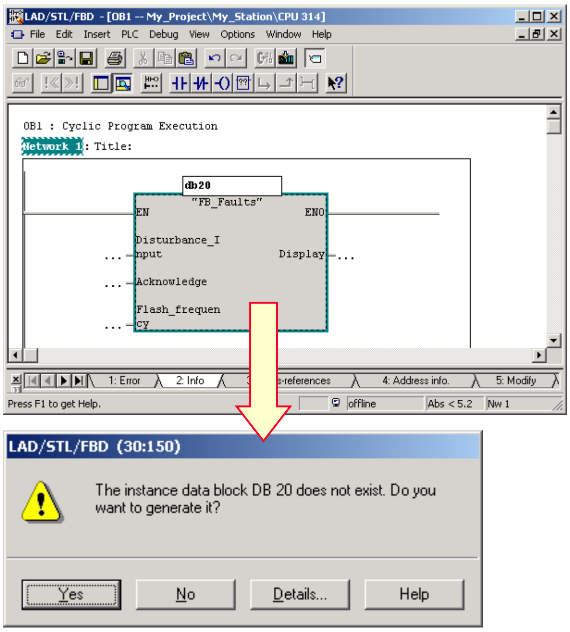
**Instance DB Structure**

When a DB is generated and references an FB, STEP7 creates the data structure of the data block using the structure specified in the local declaration table for the function block. After you save the DB, the data block is created and can then be used as an instance DB.

# Generating Instance Data Blocks

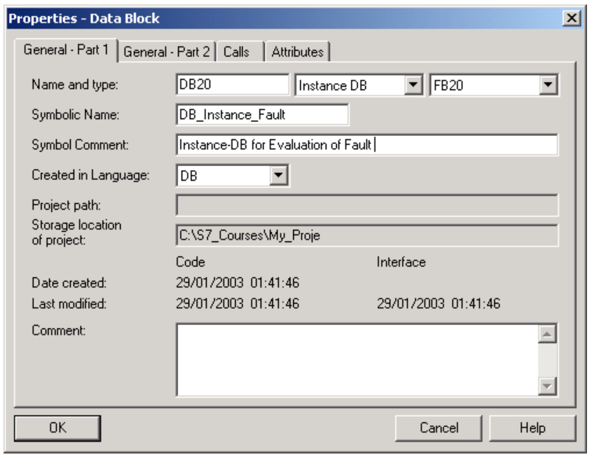
**1. Generate instance DB with FB call**

In the LAD/STL/FBD Editor




**2. Create new instance DB**

In the SIMATIC Manager




SIMATIC® S7


Date: 12.03.03
File: PRO1_10E.18

SITRAIN Training for Automation and Drives

## Generating an Instance DB

There are two ways of generating a new instance DB:

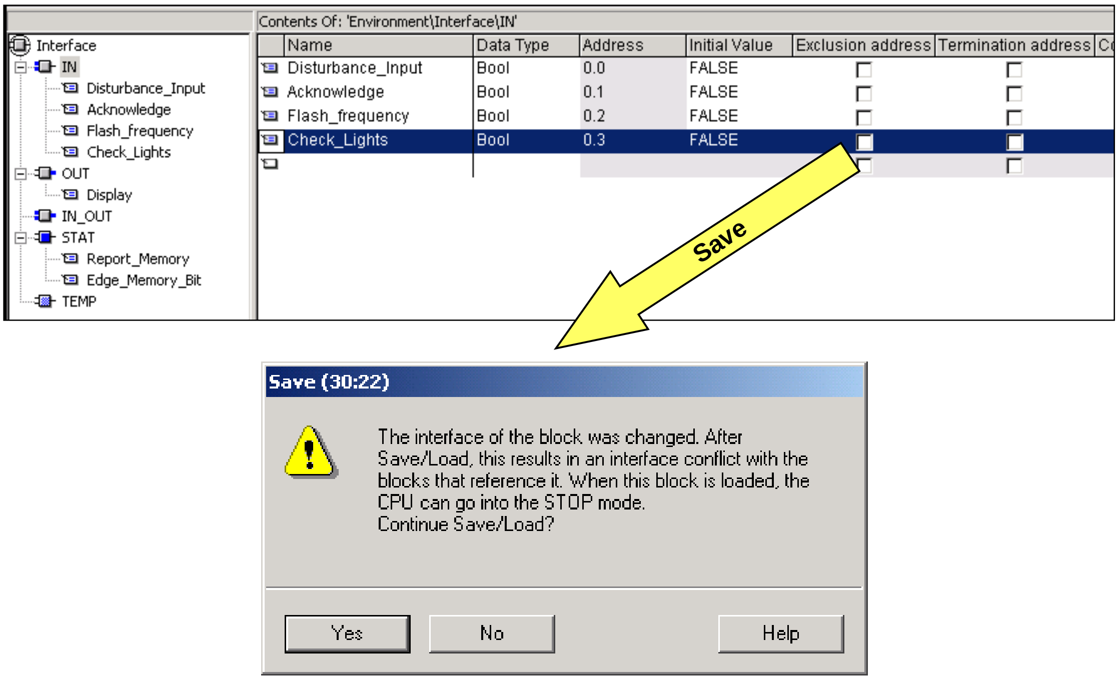
- When you call an FB, you specify with which instance DB the FB is to work. The following message then pops up:
  "The instance data block DB x does not exist. Do you want to generate it?".
- When you create a new DB, you select the option "Data block referencing a function block".

## Notes

One instance DB can only reference one FB. However, one FB can be referenced by a different instance DB every time it is called.

If you modify the FB (by adding parameters or static variables), you must then also generate the instance DB again.

## Inserting/Deleting Block Parameters Later On




SIMATIC® S7


Date: 12.03.03
File: PRO1_10E.19

**SITRAIN** Training for Automation and Drives

**Problem**

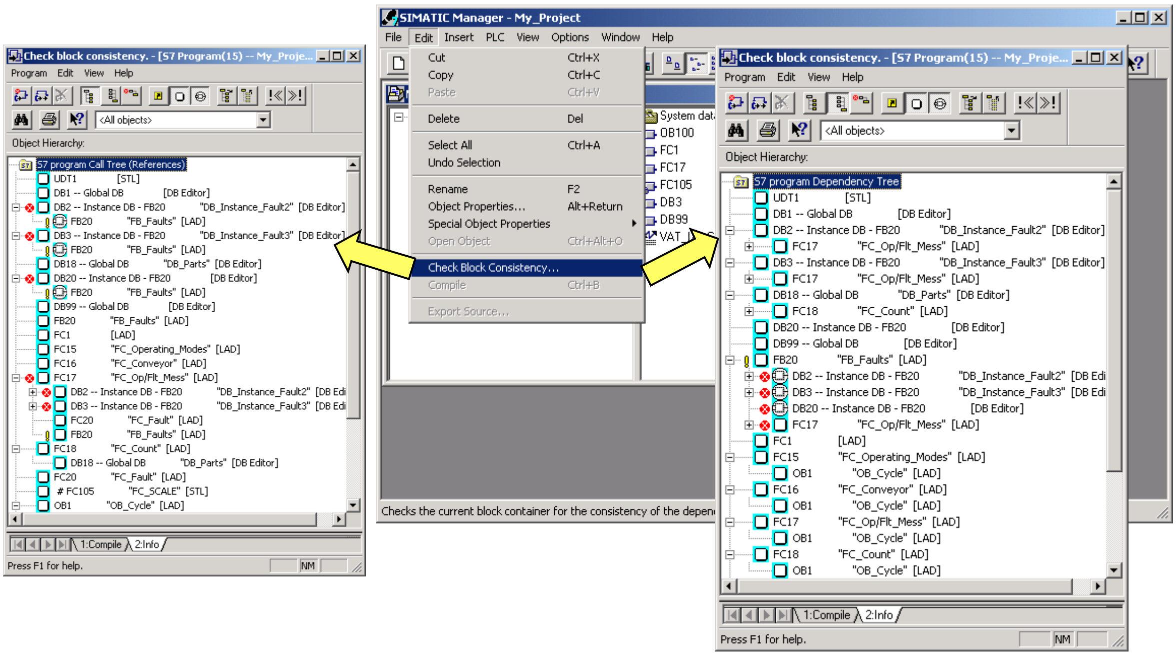
If you have to adjust or supplement the interfaces or the code of individual blocks during or after program creation, it can lead to time stamp conflicts. Time stamp conflicts can, in turn, lead to block inconsistencies between calling and called blocks or reference blocks and thus to a great degree of correction.

If block parameters are added or deleted later on to a block already called in the program, you also have to update the calls of the block in other blocks. If this is neglected, the CPU either goes into STOP or the block function can no longer be guaranteed since additionally declared formal parameters are not supplied with actual parameters when called.

In the example, the additional input parameter "Check_lights" was inserted which has to be supplied later on with an actual parameter in all block calls.

When you save a block whose interface was modified by adding or deleting formal parameters, a message pops up warning you of possible problems.

## Checking the Block Consistency







**Area of Use**

The function *Check block consistency -> Compile* clears up a large portion of all time stamp conflicts and block inconsistencies.

Interface conflicts occur when the interface of a parameter-assignable block is modified, after the block's calls have already been programmed in other blocks. Block inconsistencies also occur when, for example, addresses are accessed symbolically, and the assignment Symbol <-> Absolute address is changed later on in the global symbol table or in data blocks.

As you can see in the slide, blocks that have consistencies that could not be cleared up automatically (such as for interface conflicts) have symbols to indicate this (see online help). These blocks can then be opened and corrected by the user with the respective Editor using the right mouse button (please see next page).

**Tree View...**

The tree view shows the logic/interface dependencies or references of the blocks of the selected block folder. The tree view can be displayed either as Dependency Tree or as Reference Tree using *View -> Reference Tree / Dependency Tree*.

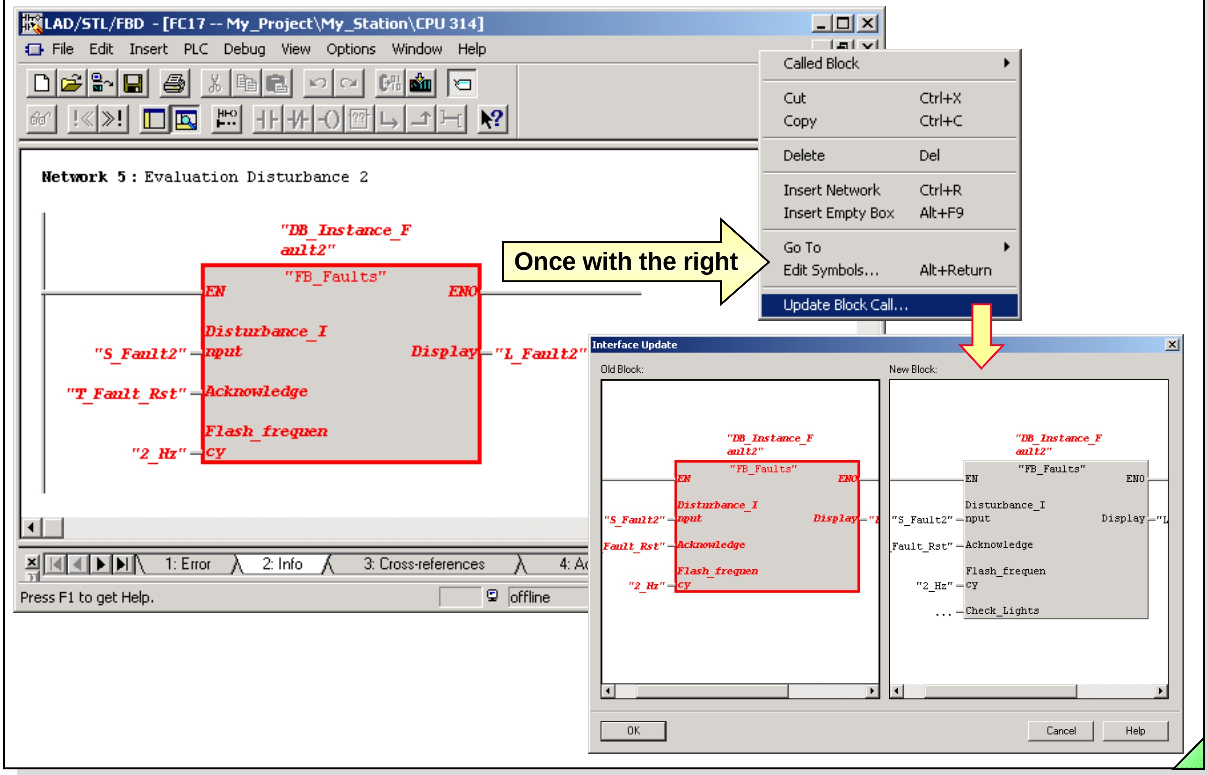
**...as Reference Tree**

The reference tree shows in levels from left to right the dependencies of all blocks or their nesting. Just as with the *Reference data Program structure,* the call paths are displayed from left to right starting from nesting depth 1. Thus, the reference tree gives an overview of the nesting depths in the individual program execution levels.

**...as Dependency Tree**

The dependency tree shows in levels from left to right the dependencies of all blocks or their nesting. In this case the displayed call paths don't start from nesting depth 1 but from the individual blocks. Accordingly, all blocks of the block folder are listed in the far left level. The following levels (to the right) show the dependencies or the blocks from which they are called. Just as with the *Reference data-Cross reference list,* the dependency tree supplies you with information about which other blocks call every block.

## Corrections when Calling Modified Blocks




SIMATIC® S7


Date: 12.03.03
File: PRO1_10E.21

SITRAIN Training for Automation and Drives

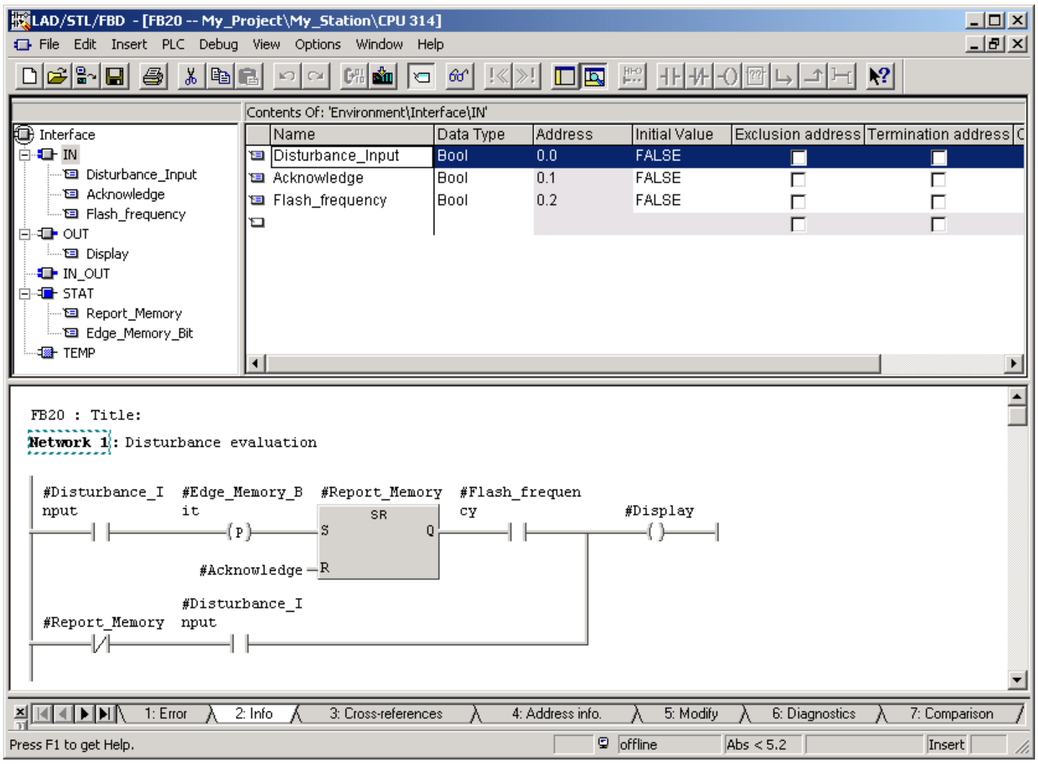
**Updating a Call**

Inconsistent calls of a block (in the slide FB 20) are marked in red in the opened, calling block (in the slide OB 1).

By clicking on the inconsistent call with the right mouse button, you can select the function *Update Block Call* in the follow-up dialog box. A window then appears in which the old (faulty) and the new block call (in the slide with the additional parameter "Check_Lights") is displayed. After confirming with OK, you can pass the missing actual parameter for the formal parameter "Check_Lights".

The Instance_DB is then regenerated for function blocks.

# Exercise: Editing a Function Block




SIMATIC® S7



Date: 12.03.03
File: PRO1_10E.22

**SITRAIN** Training for Automation and Drives

**Task**

An additional fault (simulator switch) is to be evaluated. The easiest way to do this would be to program another FC 20 call.

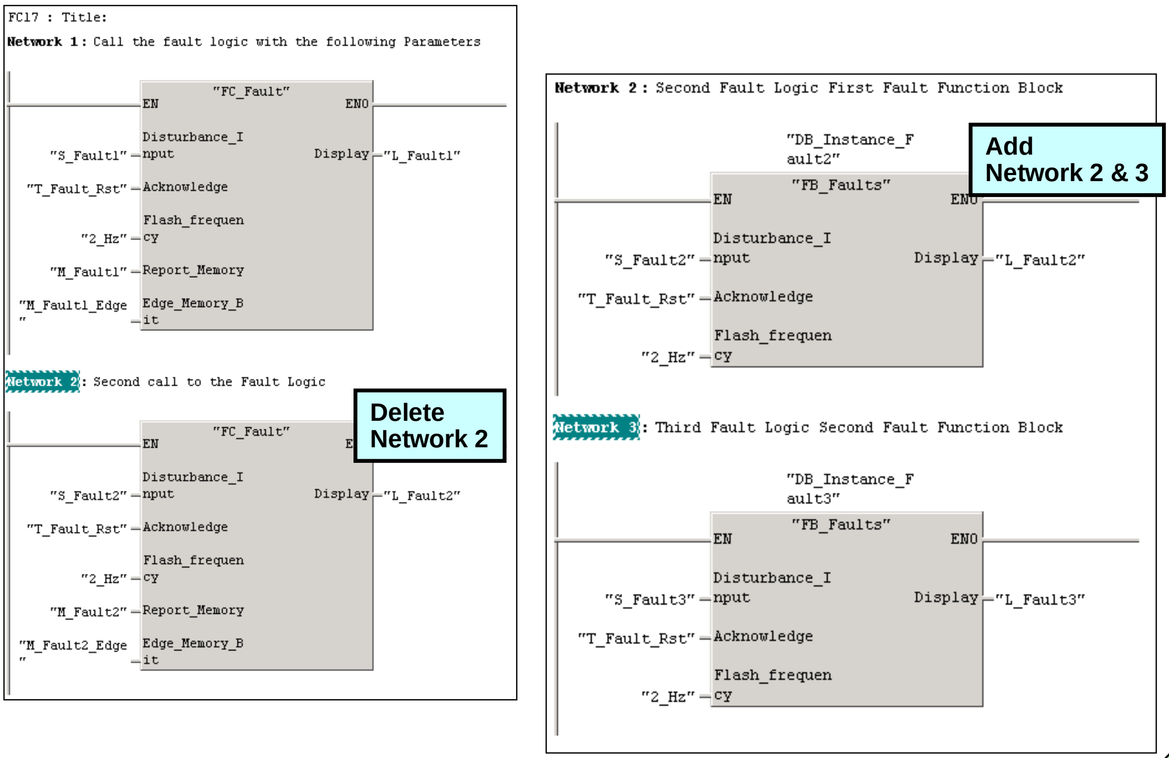
However, in order to make use of the given advantages of an FB solution, you are to program a parameter-assignable FB 20 for the evaluation of this third fault.

The static variables *Edge_Memory* and *Stored_Fault* are stored in the instance DB of the FB.
In the slide you can see the declaration table and the program code for FB 20.

**What To Do**

- Insert the FB 20 block into the S7 program called "My Program" .
- Declare the formal parameters and the static variables of the block as shown in the slide.
- Write the program code for FB 20. The easiest method of doing this is to copy and paste the necessary network from the already programmed FC 20 block.
- Save the new block and download it into the CPU.

## Exercise: Calling a Function Block and Testing It




SIMATIC® S7


Date: 12.03.03
File: PRO1_10E.23

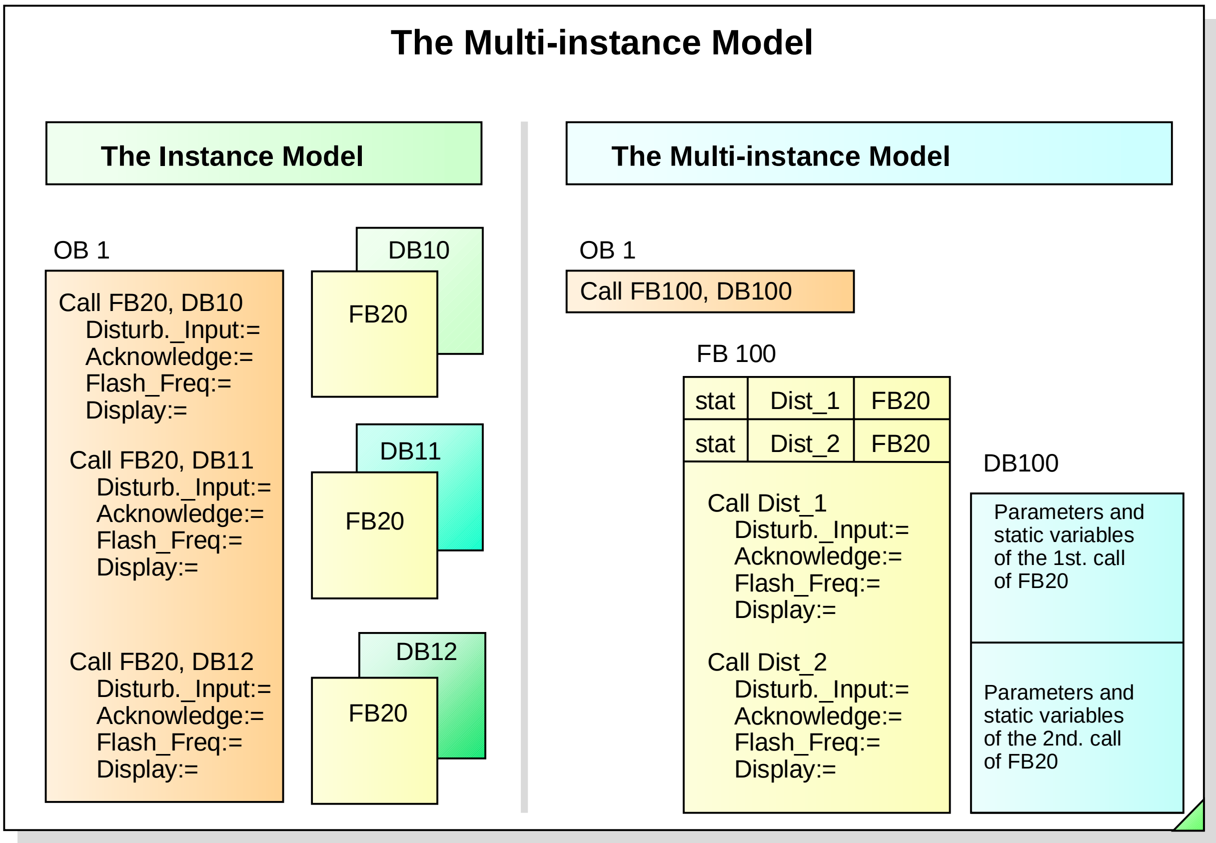
SITRAIN Training for Automation and Drives

**Task:** In FC 17 the second call to FC 20 will be deleted. The parameter-assigned block FB 20 will be called twice, once to replace the second call of FC 20 and evaluate Fault # 2 and the second time to evaluate Fault # 3.

Each time FB 20 is called assign a different instance data block.

**What To Do**

- In FC 17 delete the second call of FC 20, since the evaluation of Fault #2 is to be subsequently implemented with FB 20.
- Program both calls of FB 20 as shown in the slide in two new networks in FC 17. Let the Editor generate the instance DBs 2 and 3.
- Save the modified FC 17 offline only, for the time being.
- First download both generated instance DBs 2 and 3 from the SIMATIC® Manager into the CPU and then the modified FC 17.
- Test the functioning of your program.




SIMATIC® S7

Date: 12.03.03
File: PRO1_10E.24
SITRAIN Training for Automation and Drives

**The Multi-instance Model**

Up until now, you had to use a different instance data block for every call of a function block. The number of data blocks is limited however, and for that reason there is a method that allows you to use a common instance DB for several FB calls.

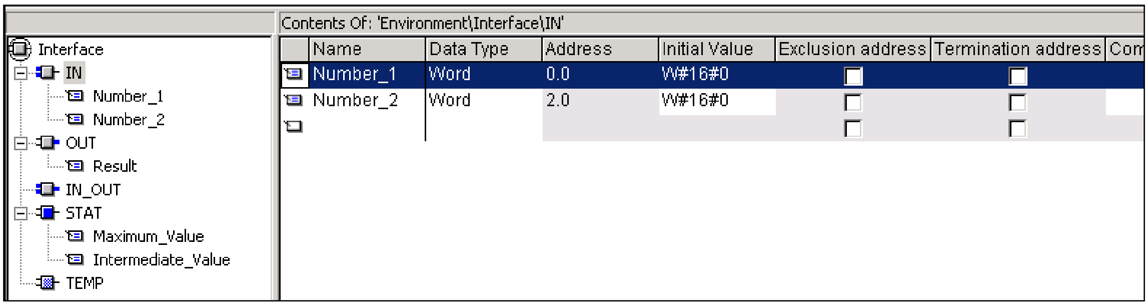
The multi-instance model now enables you to use a single DB for several calls. To do this, you need an additional FB to manage these instances.
For every FB call (FB 20), you define a static variable in the higher-level FB (FB 100). With the block call Call Dist_1, you do not then have to specify an instance DB.

The higher-level FB (FB 100) is called, for example, in OB1, the common instance DB (DB 100) is only generated once.

**Note**

Multi-instances are discussed in an advanced programming course.

# Exercise: Recognizing Types of Variables




TYPE OF VARIABLE

| Statement | Global | Local | Absolute | Symbolic | Temporary | Static | Parameter |
|---|---|---|---|---|---|---|---|
| L #Number_1 | | | | | | | |
| L #Number_2 | | | | | | | |
| T #Maximum_value | | | | | | | |
| L #Intermediate_result | | | | | | | |
| L “Number_1" | | | | | | | |
| T MW 40 | | | | | | | |
| T #Number_2 | | | | | | | |

SIMATIC® S7


Date: 12.03.03
File: PRO1_10E.25

SITRAIN Training for Automation and Drives

**Task** In the Statement section of the slide, you can see a program with various variables. In the table below it, assign the corresponding properties to the variables.

**What To Do** In the table, mark the associated data type with an X.

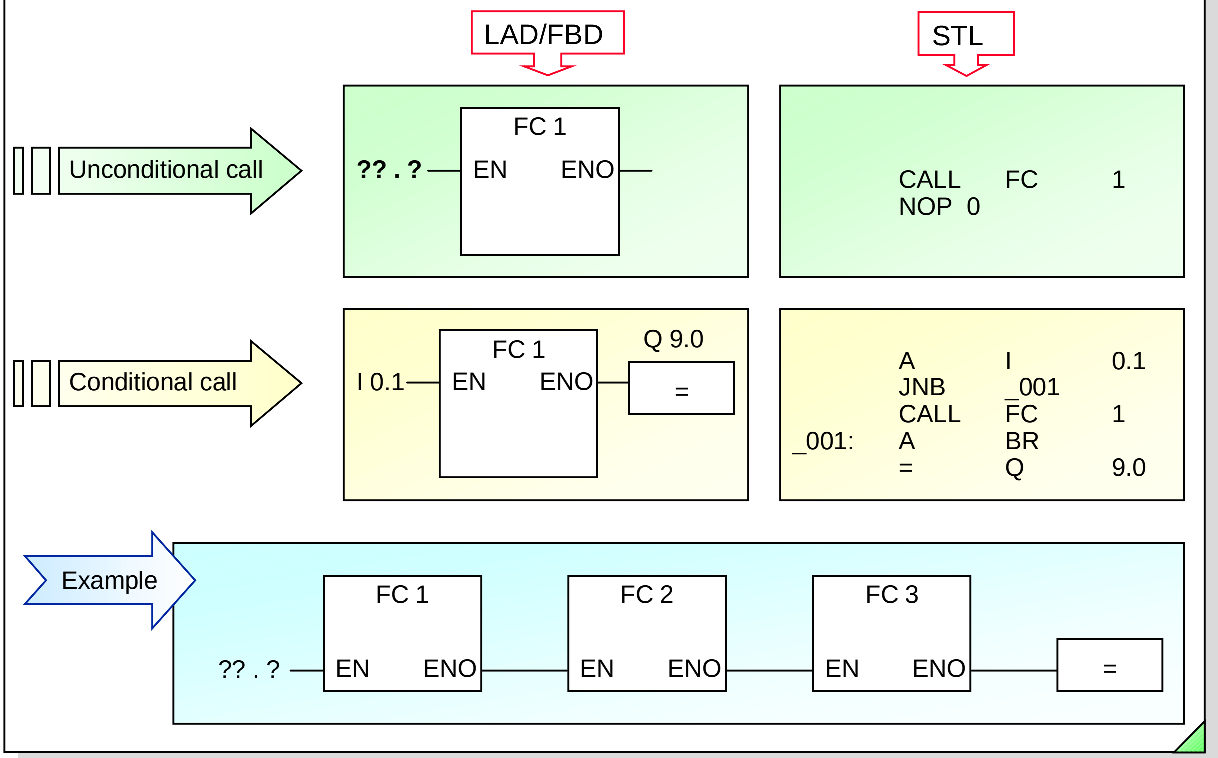
Answer the following question:
What is not correct in the statement T#Number_2 ?

.......................................................................................................................

.......................................................................................................................

.......................................................................................................................

## Using the EN/ENO Parameters with Block Calls




SIMATIC® S7


Date: 12.03.03
File: PRO1_10E.26

SITRAIN Training for Automation and Drives

**Standard FCs**

The following rules exist for the execution of standard FCs:

- If EN=0, the block is not executed and ENO is also =0.
- If EN=1, the block is exeucted and if it is executed without errors ENO is also =1.

  If an error occurs while the block is being executed, ENO becomes =0.

**User FCs**

It doesn't matter whether a user block was written in LAD, FBD or STL, when it is called in LAD/FBD, the parameters EN and ENO are added as well.

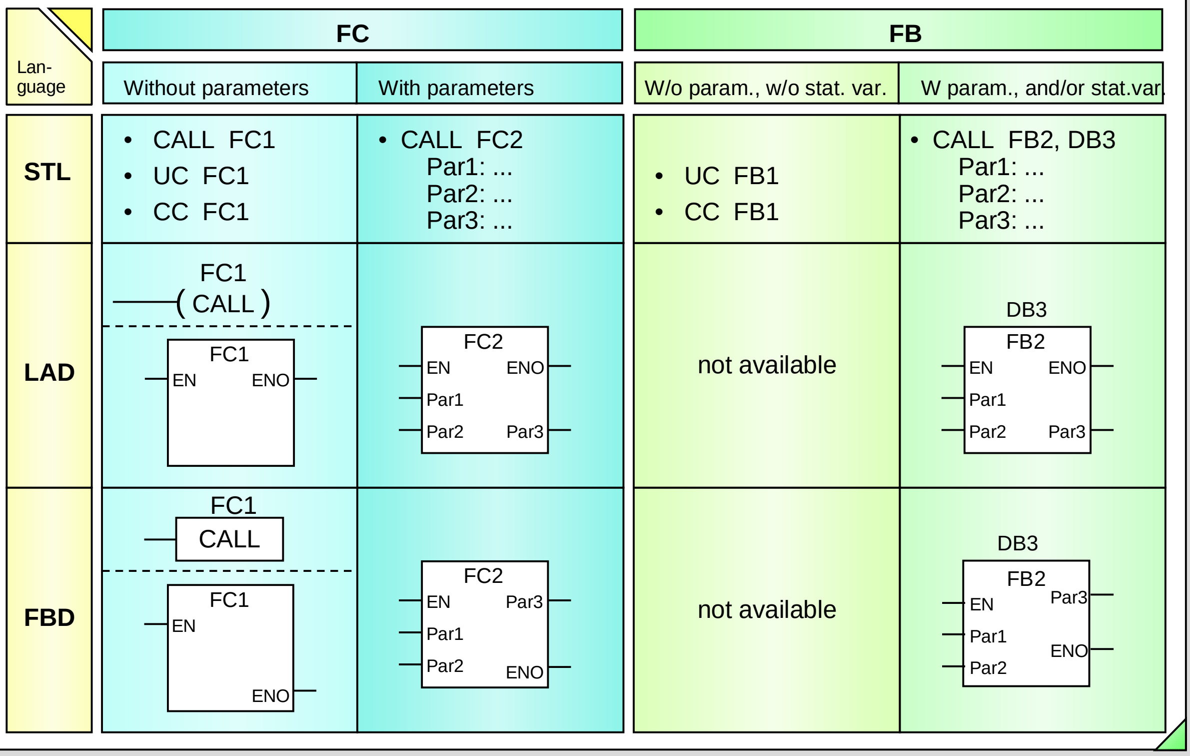
EN/ENO do not exist in STL. You can, however, emulate them.

You must program -irregardless of the programming language- an error evaluation.

**Interconnection**

In LAD/FBD, several boxes can be grouped together one after the other and linked logically with EN / ENO.

## Summary: Block Calls




SIMATIC® S7



Date: 12.03.03
File: PRO1_10E.27

SITRAIN Training for Automation and Drives

**CALL**

The "CALL" instructions is used for calling program blocks (FC, FB, SFC, SFB). In the graphic programming languages LAD and FBD, the block call can be made dependent on a condition (RLO) using the EN input of the CALL box. In the STL programming language the block call is absolute, that is, regardless of the RLO.

If you call an FB or SFB with "CALL", you must also specify the relevant instance DB. You can use either the absolute or the symbolic name for the block.

For example: "CALL FB2, DB2" or "CALL valve, level".

**UC**

The "UC" instruction is an unconditional call of a block of the type FC or FB. The UC operation is only allowed if the FC or FB called is not parameter-assignable. As well, no static variables may be declared in an FB called with UC.

**CC**

The "CC" instruction is a conditional call of a block of the type FC or FB. The CC operation is only allowed if the FC or FB called is not parameter-assignable. As well, no static variables may be declared in an FB called with CC.

**Note**

The UC and CC instructions do not convert to Ladder or Function Block Diagram representation. Also, the program editor will allow the UC or CC call of a parameter assigned block. When the program is downloaded an ”AREA ERROR when Reading” system fault will occur.

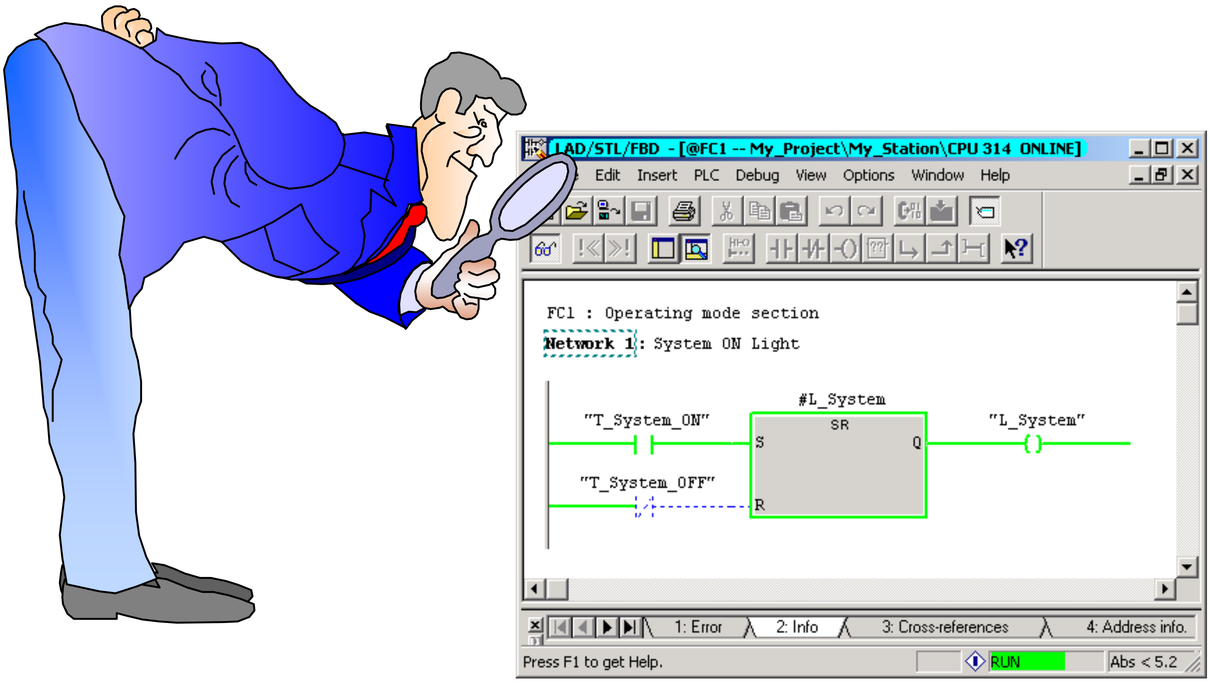
**Parameters**

The formal parameters declared in the declaration table of a block are the interface of the block. When a parameter-assignable FC is called, an actual parameter must be passed for every formal parameter. This parameter passing is not mandatory when an FB is called.

Static and temporary variables are not parameters and are accordingly not part of the block interface. As a result, no parameter passing for static or temporary variables can take place in block calls.

SIEMENS

# Troubleshooting



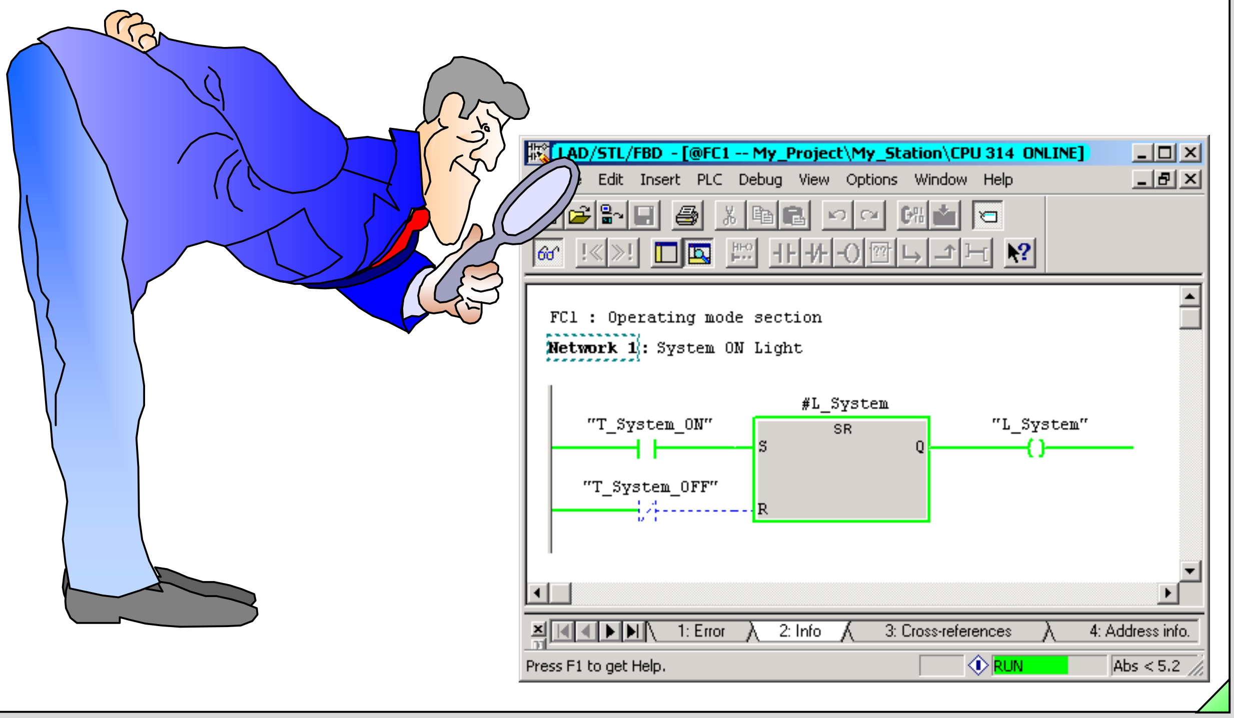

SIMATIC® S7


Date: 12.03.03
File: PRO1_11E.1

SITRAIN Training for Automation and Drives

**Contents** Page

# Troubleshooting




SIMATIC® S7


Date: 12.03.03
File: PRO1_11E.2

**SITRAIN** Training for Automation and Drives

**Contents (Contents)** Page

## Objectives

**Upon completion of this chapter the participant will ...**

... be able to classify occurring errors as "Errors recognized 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 Variable" function and use it for troubleshooting

... be able to interpret the displays of the "Monitor" 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

... understand the "Breakpoints" function

SIMATIC® S7

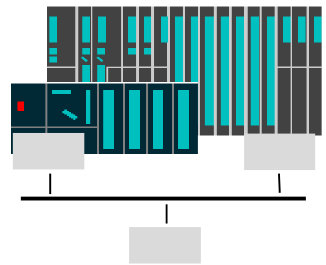

Date: 12.03.03
File: PRO1_11E.3

SITRAIN Training for Automation and Drives

# Categories of Errors

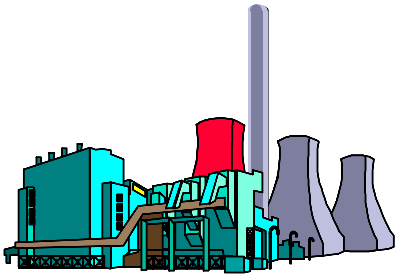
## Errors Detected by the System

- 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)



SIMATIC® S7


Date: 12.03.03
File: PRO1_11E.4

SITRAIN Training for Automation and Drives

**Monitoring Functions** Diagnosis is important in the operating phase of a system or machine. Diagnosis usually occurs when a problem (disturbance) leads to standstill or to the incorrect functioning of the system or machine.
Due to the costs associated with downtimes or faulty functions, the associated cause of the disturbance has to be found quickly and then eliminated.

**Categories of Errors** Errors that occur can be divided into two categories, depending on whether or not they are detected by the PLC:

- Errors that are detected by the PLC‘s operating system and normally cause the CPU to go into the Stop state.
- Functional errors, that is, the CPU processes the program as usual, but the desired function is either not executed at all or it is executed incorrectly.

  The search for these types of errors is much more difficult, since the cause of the error is initially hard to determine.

  There are two types of functional errors.

  – **Process Fault** (such as a wiring error)

    A fault that was triggered by the faulty functioning of components directly associated with the process control, such as cables to sensors/actuators or by a defect in the sensor/actuator itself.

  – **Logical Programming Errors** (such as a double assignment)

    A software error that was not detected during creation and startup of the user program and probably occurs only on extremely rare occassions.

## 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**
- **Set Breakpoints**
- **Block Compare**

SIMATIC® S7


Date: 12.03.03
File: PRO1_11E.5

**SITRAIN** Training for Automation and Drives

**Using Test Functions ...** There are various STEP 7 test functions for troubleshooting, depending on the type of error caused.

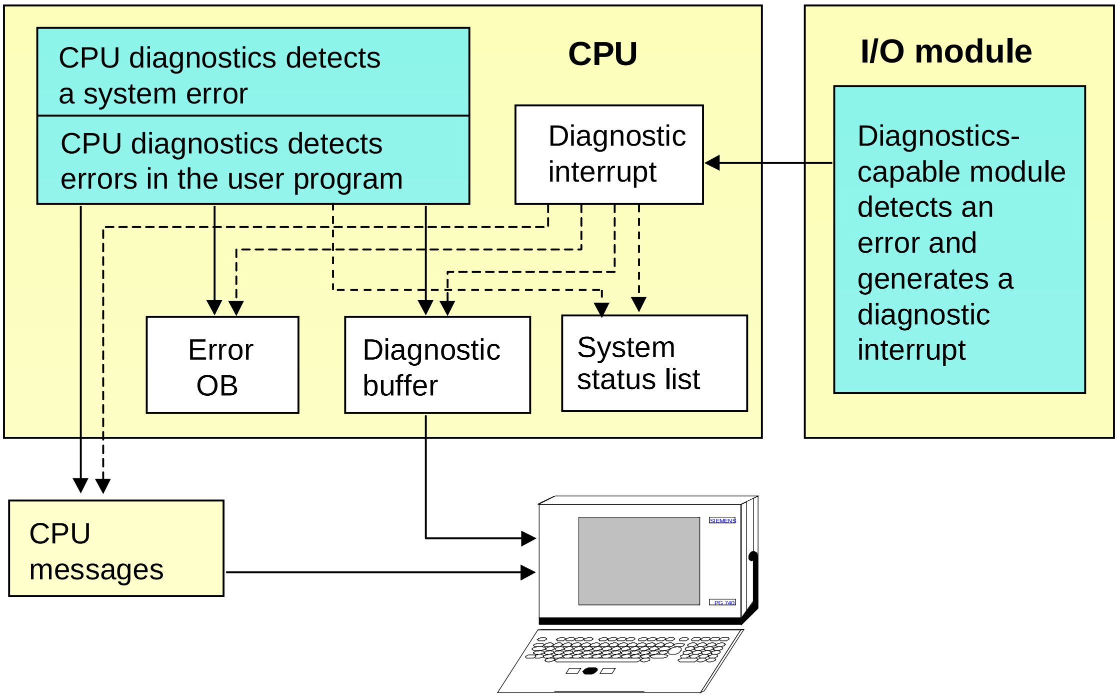
**…when CPU in STOP** For errors that are detected by the system, the test functions Diagnostic Buffer, I STACK, B STACK, L STACK and Hardware Diagnostics give detailed information on the cause of the error and the location of the interruption. By programming Error OBs (see the chapter on Organization Blocks), information on the error that occurred can be evaluated by program and the transition of the CPU into the STOP state can be prevented. If the CPU has stopped, the use of the test functions Monitor / Modify Variable and Monitor Blocks makes little sense since the CPU neither reads nor outputs process images while in the STOP state, and also no longer executes the program.

**…when CPU in RUN** Vice versa, it makes little sense, as a rule, to use test functions such as I STACK, B STACK or L STACK for troubleshooting when the CPU is in RUN, since program execution has not been interrupted and the system does not provide any information on the error that occurred. The Module Information test function merely provides general information on the CPU‘s operating mode or on errors that occurred in the past. Functional errors can be diagnosed as follows:

- **Process Fault** (such as a wiring error)
  - wiring test of the inputs: *Monitor Variable*
  - wiring test of the outputs: *Enable Peripheral Outputs (*only when CPU-STOP)
- **Logical Programming Errors** (such as a double assignment)
  - All test functions listed, with the exception of *Enable Peripheral Outputs* , can be used for searching for logical program errors.

**Force** Forced control of addresses regardless of the program logic.
**Breakpoints** For following the program execution in single steps.
Both test functions are used mostly during program design.

## System Diagnostics - Overview







**System Dignostics**

All the monitoring functions with regard to the correct functioning of the components of a PLC are grouped together under System Diagnostics.

All S7-CPUs have an intelligent diagnostics system. The acquisition of diagnostic data by the system diagnostics does not have to be programmed. It is integrated in the operating system of the CPU and in other diagnostics-capable modules and runs automatically.

The CPU stores (temporarily) errors that occur in the diagnostic buffer and thus enables a fast and targeted error diagnosis by service personnel, even for sporadically occurring errors.

**System Reaction**

The operating system takes the following actions when it detects an error or a STOP event, such as an orperating mode change (RUN -> STOP):

- A message on the cause and the effect of the occurring error is entered in the diagnostic buffer, complete with the date and time.

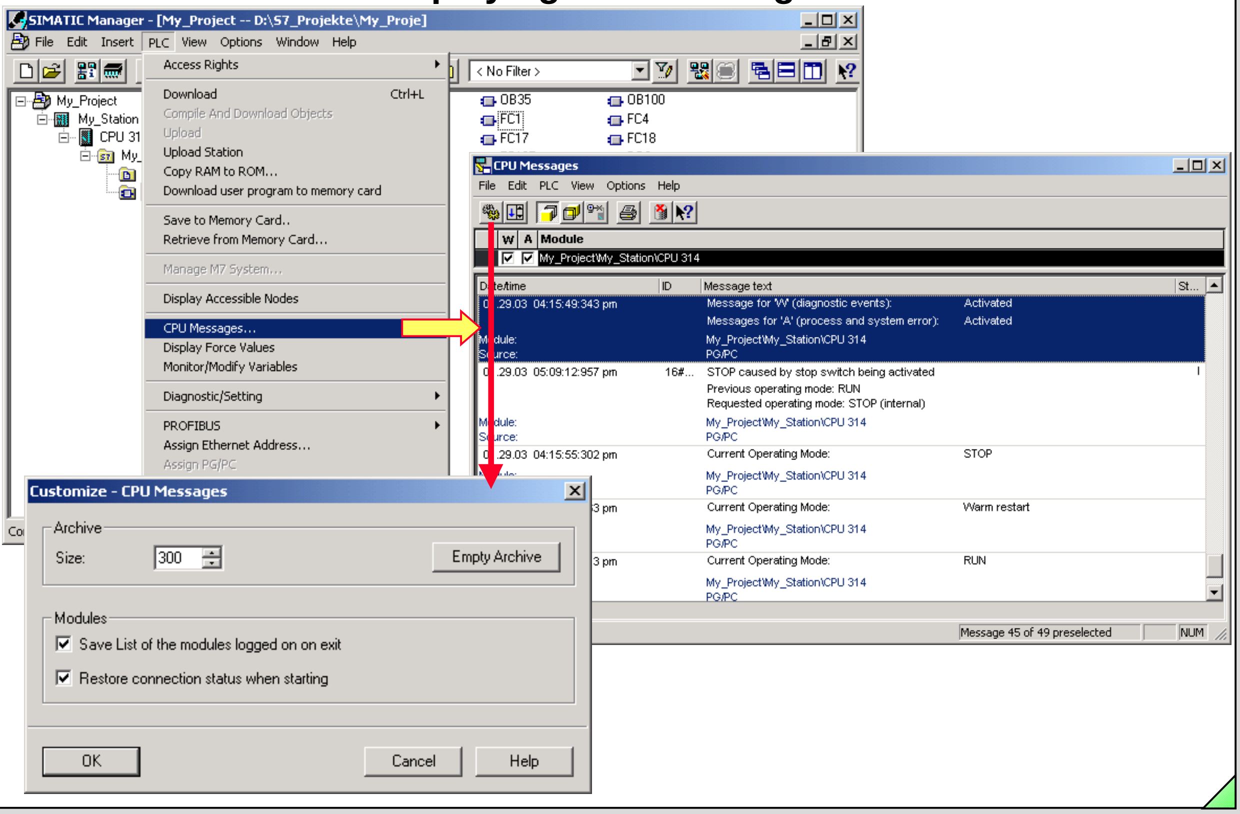
  The diagnostic buffer is a FIFO buffer on the CPU module for storing error events. The size of the diagnostic buffer depends on the CPU (such as the CPU 314 = 100 entries).

  In the FIFO buffer structure, the most recently entered message overwrites the oldest diagnostic buffer entry. The diagnostic buffer is also not deleted by a CPU Memory Reset.
- System status lists, that give information of the system status, are updated.
- The Error OB associated with this error is called. This gives the user the opportunity of carrying out his own error handling.

**CPU Messages**

If the CPU is to signal the cause of the STOP to all associated display devices (such as a PG or OP) during a transition to STOP, the "Report Cause of STOP" function has to be activated under the CPU Properties "Diagnostics/Clock" in the Hardware Configuration.

## Displaying CPU Messages




SIMATIC® S7


Date: 12.03.03
File: PRO1_11E.7

SITRAIN Training for Automation and Drives

**CPU Messages**

With this function you can immediately display an error message for sporadic errors in the system on a programming device or an HMI device. A message window pops up on the PG or OP as soon as the connected CPU diagnostics detects an error. User formatted messages can also be output programmed using System Function SFC 52, WR_USMSG.

**Module**

At the upper edge of the window all CPUs that were called in the SIMATIC® Manager with the menu option *PLC -> CPU Messages* are entered in a list. The list is divided into four columns:

1. In the first column, an icon displays whether a connection was interrupted by the external partner.
2. In the column "W", system diagnostic and user diagnostic messages are activated /deactivated.
3. In the column "A", interrupt messages are activated / deactivated.
   The "CPU Messages" application checks if the module in question even supports diagnostic and interrupt messages. If this is not the case, a message is output.
4. In the column "Module", the name of the module or the path of the S7 program is entered.

**Incoming Messages**

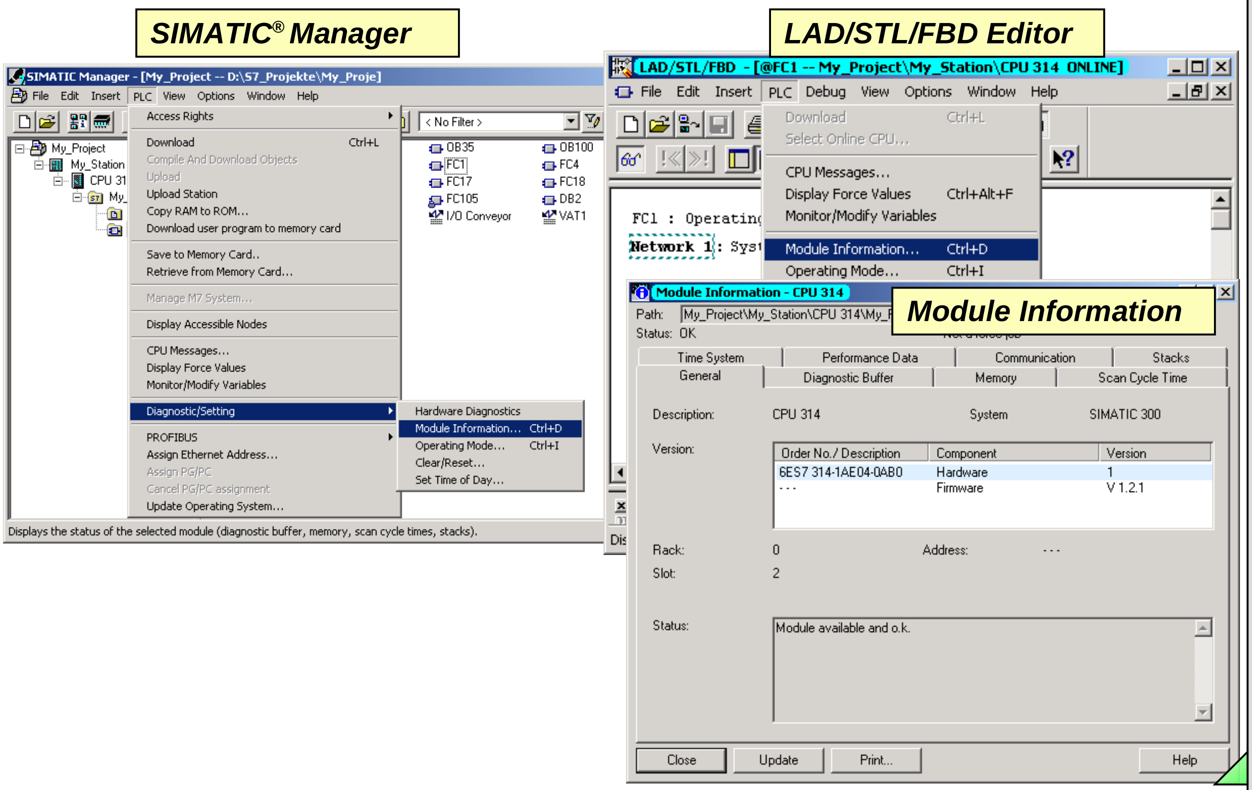
The following options can be selected through the "View" menu:

- Place on Top: As soon as a message is received, the "CPU Messages" window pops up on top, the message is displayed and at the same time it is entered in the message archive.
- Leave in the Background: The receiving of messages takes place in the background. Messages are displayed in the window, but the window remains in the background. The messages are archived and can be displayed as required.
- Ignore Message: Messages are neither displayed nor archived.

**Archive**

In the Settings you can select the size of the archive (40 to 2000 messages) or empty the archive, among other things.
*Options -> Customize -> Customize - CPU Messages*

## Calling the "Module Information" Tool




SIMATIC® S7


Date: 12.03.03
File: PRO1_11E.8

SITRAIN Training for Automation and Drives

**Overview**

The CPU information supports you in system diagnosis without you having to do any programming, and makes it possible to quickly detect errors, localize them and eliminate them.

The information that you need for troubleshooting is supplied by the function:

*PLC -> Diagnostic/Setting -> Module Information*

You can access this function from the SIMATIC® Manager or from other tools (such as the STL/LAD/FBD Editor).

**Module Information**

The Module Information function reads the most important data from the directly connected module. You will find additional information in the individual tabs:

General: Among other things, the module description, hardware and firmware versions

Diagnostic Buffer: It contains all diagnostic events in the order they occurred. All events are listed in plain text and in the order they occurred in the display.

Memory: Size and usage of the EPROM load memory, RAM load memory and the work memory.

Scan Cycle Time: Displays the monitoring time selected, the shortest, the longest and the current cycle time

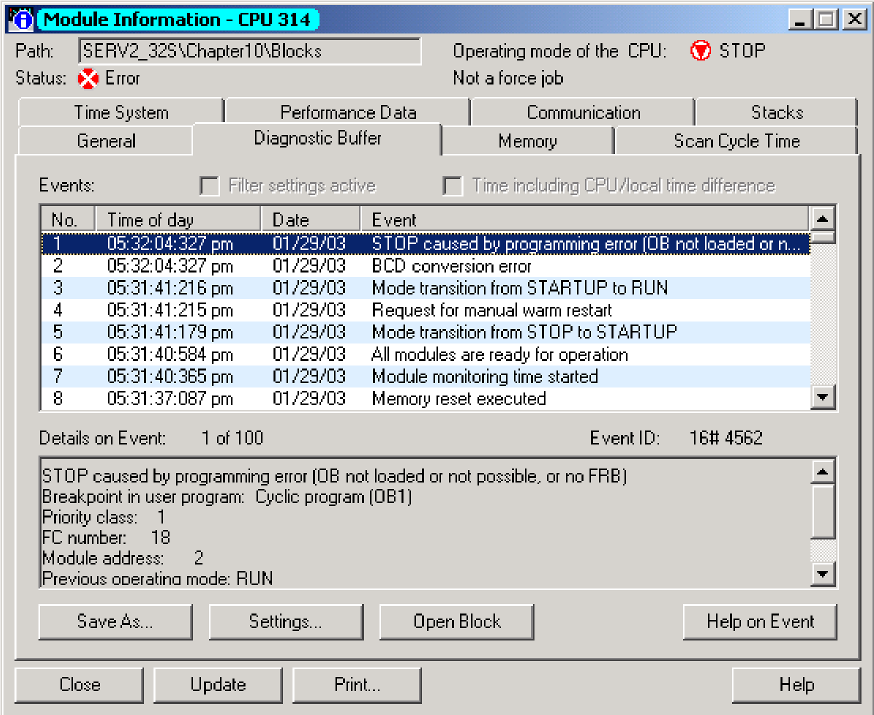
Time System: Displays the real-time clock and the integrated run-time meter

Performance Data: Displays the integrated system blocks and the available organization blocks as well as address areas (I,Q,M,T,C,L)

Communication: Displays the performance data of the communication interfaces and the connection overview

Stacks: Information on the contents of the I Stack, B Stack and L Stack. For this, the CPU must be in the STOP state or have reached a breakpoint.

## Module Information Tab: "Diagnostic Buffer"






**Diagnostic Buffer**

The diagnostic buffer is is a FIFO buffer in the CPU's battery-backed memory area. The diagnostic buffer contains all diagnostic events in the order in which they occurred. The diagnostic buffer cannot be deleted by a memory reset.

All events can be displayed on the programming device in plain text and in the sequence in which they occur.

**Details on Event**

When you select an event, the following additional information is supplied in the "Details on Event" box, such as:

- Event ID and event number
- Block type and number
- Additional information, depending on the event, such as the relative STL line address of the instruction that caused the event (Module address 80 in the example)

**Help on Event**

When you click on the Help on Event button, help on the event selected in the list is opened.

(In the example shown, a programming error has occurred for which the relevant error OB (OB121) is not programmed in the CPU.)

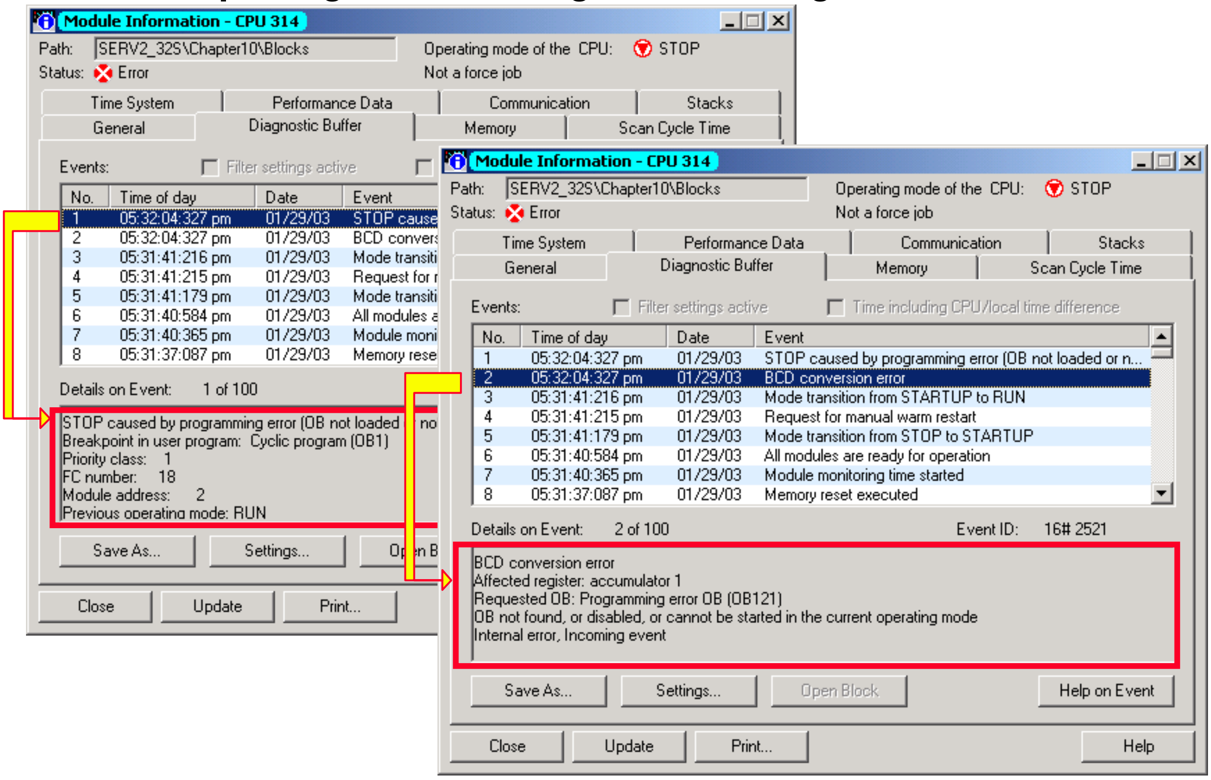
**Open Block**

When you click on the Open Block button, the block in which the interruption occurred can be opened online (in the CPU) (in the above example: "FC 18").

**Opening the Tool**

You open the diagnostic buffer by selecting the menu options *PLC -> Diagnostic/Setting -> Module Information --> Diagnostic Buffer* tab in the SIMATIC® Manager or Program Editor.

# Interpreting Error Messages in the Diagnostic Buffer







**General**

The last entry appears at the top of the list. The time shows you which error messages belong together (event no. 1 and 2 in the slide).

**Interpreting Errors**

In our example, a complete restart was performed before the error occurred (event no. 3 to 5). After the restart, the error occurred and caused entries no. 1 and 2.

Event No. 1: the CPU goes into the STOP mode because the associated error OB (OB 121) was not loaded into the controller.

The "Details" window shows the processing level, for example, OB 1 (Cycle) as well as the block and the address of the instruction that caused the error (FC 18, block address 80).

Event No. 2: the actual cause of error (BCD conversion error) is shown here. Under "Details" you see that an invalid BCD number was stored in Accumulator 1. Also, the error OB (OB 121) is shown that is called by the operating system when this error has occurred.

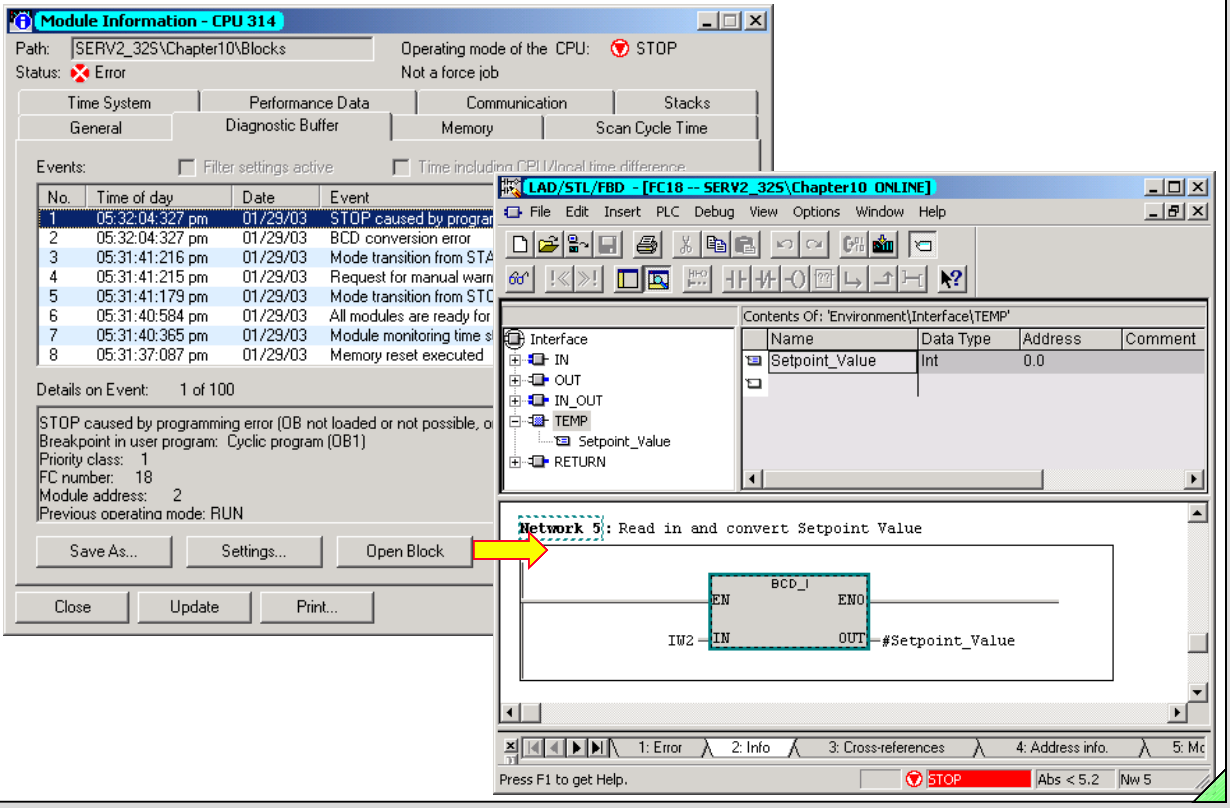
**Error OBs**

The following error OBs are available for error handling:

- OB 81: Power supply fault (backup battery failure)
- OB 82: Diagnostic interrupt (such as wire break or module ground fault)
- OB 84: CPU hardware fault (incorrect signal level at the MPI interface, only for S7-400™)
- OB 85: Program execution error (error in updating the process image)
- OB 86: Failure of distributed rack or DP-Slave
- OB 87: Communication error (incorrect frame ID)
- OB 121: Programming error (such as BCD conversion error, called block not available)
- OB 122: Access error (Load or transfer operations to non-existent or defective I/O)

Organization blocks are dealt with in more detail in their own chapter!

# Opening a Block Containing an Error




SIMATIC® S7


Date: 12.03.03
File: PRO1_11E.11

**SITRAIN** Training for Automation and Drives

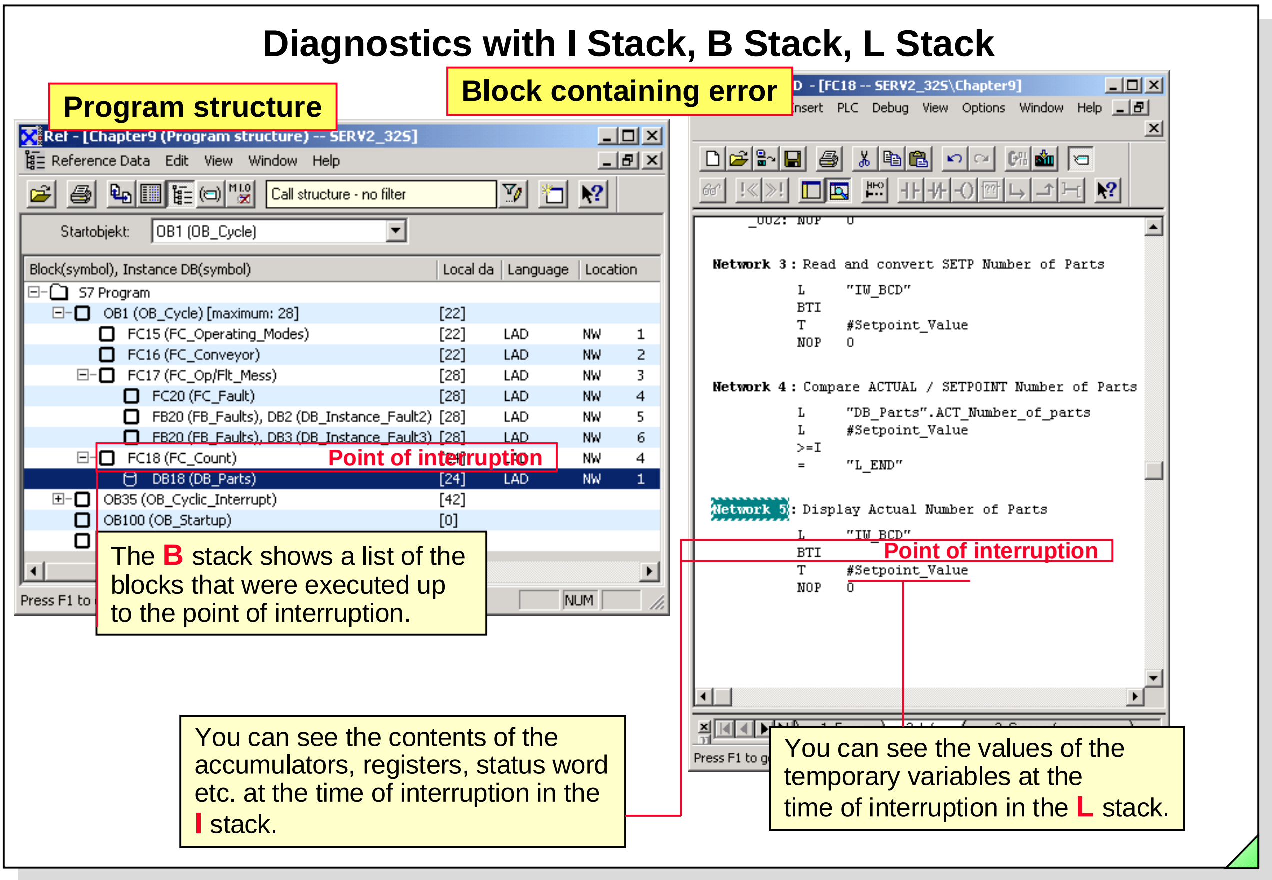
**Opening a Block**

For so-called synchronous errors, that is, for errors that were triggered by a faulty instruction in the user program, you can open the interrupted block by clicking on the "Open Block" button.

If the STL language is selected, the cursor is positioned directly in front of the instruction that caused the interruption. In LAD/FBD, the network causing the interruption is displayed.

In the example shown, the value read in from thumbwheel button IW 2 is converted from BCD to Integer. At the time of interruption, this was not a valid BCD number, so that the value in accumulator 1 could not be converted from BCD to Integer. In this case, the invalid BCD number can be found out by reading out the I stack (see following pages).

The error occurred in FC 18, Network 5.




SIMATIC® S7


Date: 12.03.03
File: PRO1_11E.12

SITRAIN Training for Automation and Drives

**Introduction**

For synchronous errors (OB 121, OB122) it may make sense, depending on the circumstances, to display further information about the cause of the error or its location using the stack contents (I stack, B stack, L stack). That way, you can determine for example, which values were stored in the accumulators at the time when the CPU went into STOP.

**B Stack**

Blocks are often called several times in a user program. This means that the information concerning the block number and the instruction causing the interruption does not clearly indicate in which call chain the error occurred.

The B stack contains a list of all those blocks called whose execution at the time of the transition to the STOP state were started but were not as yet completed.

**I Stack**

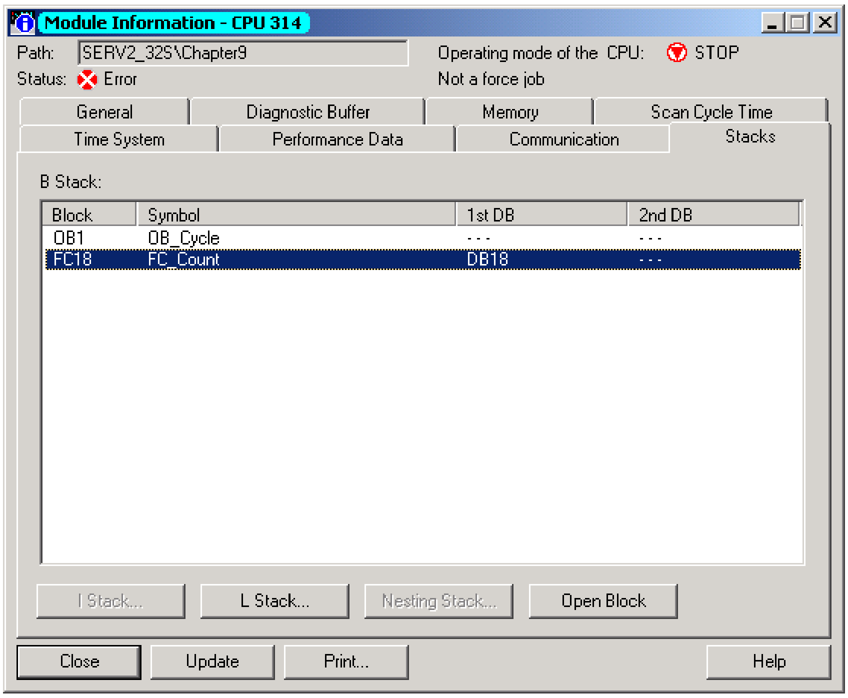
The I stack contains the contents of the registers at the time of interruption:

- contents of the accumulators and address registers
- which data blocks are open
- contents of the status word
- program execution level (such as OB 1 or OB 10)
- interrupted block specifying the network and the number of the instruction
- next block to be executed

**L Stack**

The L stack contains the values of the temporary variables of the blocks. You need some experience to evaluate this data however, since the contents are given in a "Hex Dump" rather than in "plain language".

# Contents of the B Stack




SIMATIC® S7



Date: 12.03.03
File: PRO1_11E.13

SITRAIN Training for Automation and Drives

**Stacks**

In order to display the stack information, the CPU must be in the STOP mode

- because of a program error
- because of a programmed transition to the STOP state (per SFC)
- because a breakpoint is reached

**B Stack**

The block stack (B stack) is a graphic representation of the call hierarchy, that is, the sequence and nesting of the called blocks up to the point of interruption.

The B stack also contains a listing of all interrupt and error OBs as well as the open DBs.

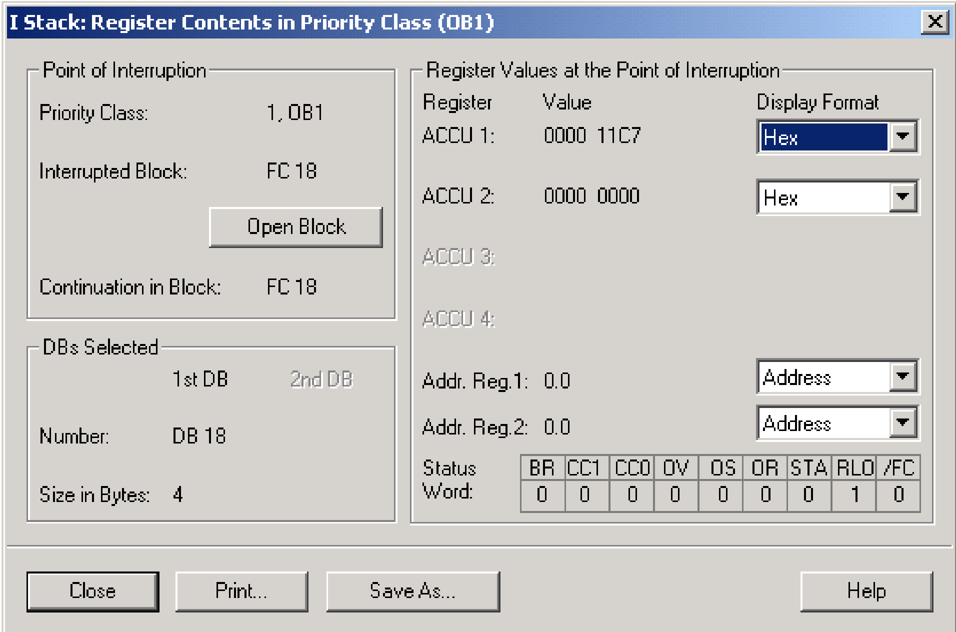
The program execution was interrupted in the block shown at the bottom of the list.

In the slide you can see that the interruption occurred in the block FC 18. When a parameter-assignable block is called several times, the information, after which block call the program execution was interrupted, is relevant for troubleshooting, since the cause of the interruption can lie in the passing of faulty actual parameters by the calling block.

**Open Block**


To open the block online, you select the block in the B stack list and then click on the "Open Block" button. You can then edit this block. The cursor is located in the line that follows the interrupt causing instruction or, in LAD and FBD, the network is marked in which the program execution was interrupted.

## Contents of the I Stack




SIMATIC® S7


Date: 12.03.03
File: PRO1_11E.14

SITRAIN Training for Automation and Drives

**I Stack** The interrupt stack (I Stack) always refers to an execution level. Before you can open the I Stack, the organization block concerned must be selected in the B Stack.

**Register** The contents of all relevant registers at the time of the interruption are displayed in the I Stack screen:

- Accumulators
  You can select the numbers format for displaying the accumulator contents in the "Display Format" list.
- Address Register
  You can select the numbers format for displaying the address register contents in the "Display Format" list.
- Status Word
  Bits 0 to 7 of the status word are displayed. They are identified with abbreviations according to their meaning.

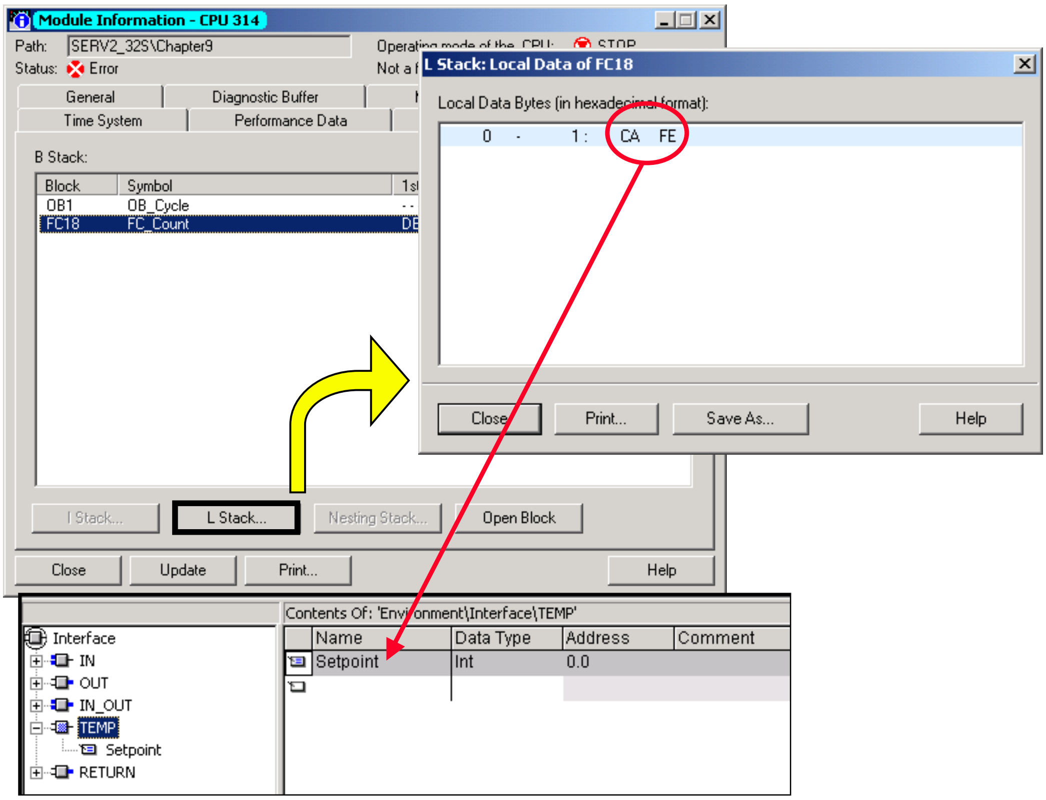
**Point of Interruption** The "Point of Interruption" field gives you information about:

- the interrupted block, with the option of opening it directly (the cursor is then located directly in front of the faulty instruction),
- the priority class of the OB, whose execution level was interrupted,
- open data blocks with their numbers and sizes.

**Error Example** For our example, you can see that the hexadecimal number 0000 11C7 is stored in accumulator 1. This is not a valid BCD number and for this reason a conversion error occurred during the conversion from BCD to Integer (BTI instruction).

This error occurred during the switching of the pushwheel button (mechanical problem). To remedy this, you could make the programmed conversion of the selected value dependent on the confirmation of the setting using a momentary-contact swtich.

## Contents of the L Stack




SIMATIC® S7


Date: 12.03.03
File: PRO1_11E.15

SITRAIN Training for Automation and Drives

**L Stack**

The current values of the temporary variables for blocks not ended at the time of interruption are contained in the L Stack.

The blocks not yet ended when the CPU switched to the STOP mode are listed in the block stack (B Stack). The local data displayed in the L Stack window refer to the block selected in the B Stack.

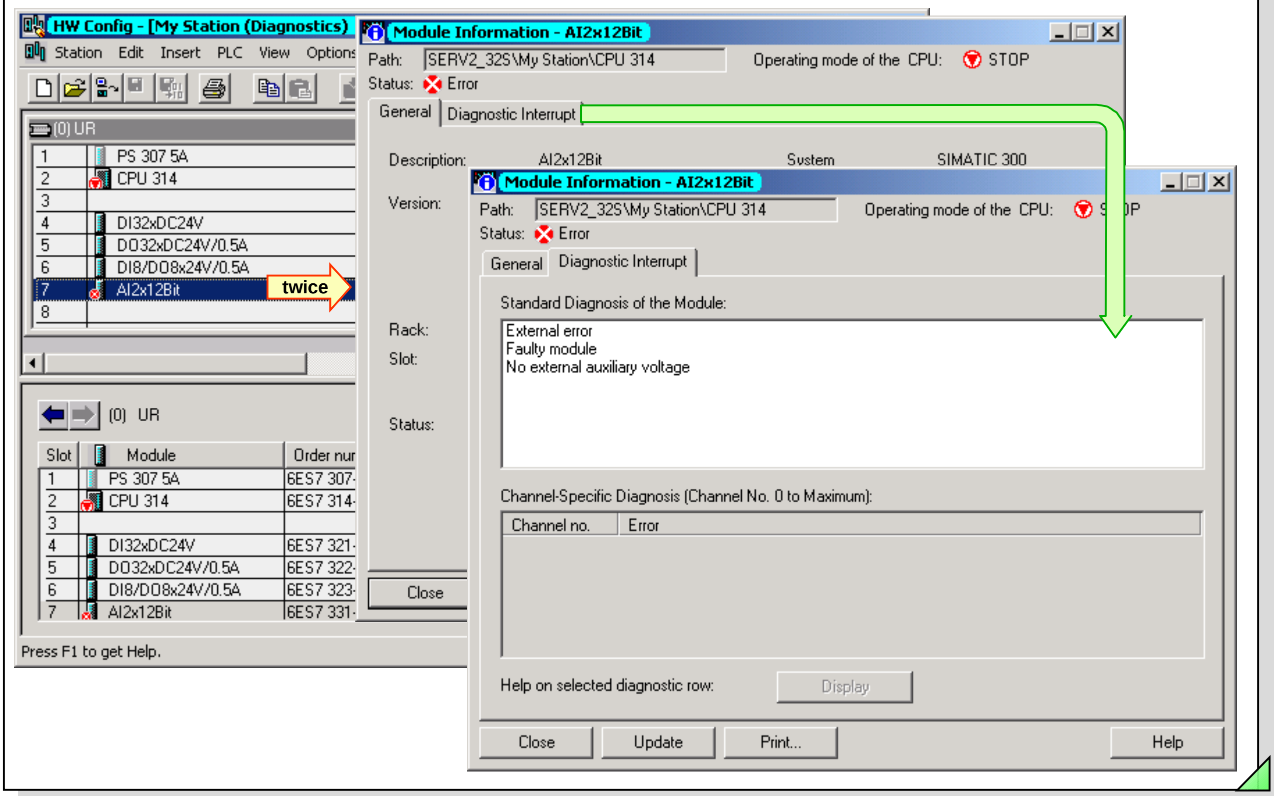
**Error Example**

In the example shown, the temporary variable #Setpoint is declared as integer in the FC 18 block. Accordingly, it occupies two bytes in the L Stack.

In the declaration table of the FC 18 block, the relative initial address of the variable in the L Stack is displayed in the column "Address".

The variable #Setpoint occupies the bytes 0 and 1 of the L Stack and has the value $\text{CAFE}_{\text{Hex}}$ = $-13570_{\text{Dec}}$ as content.

# Displaying the Hardware Diagnostics




SIMATIC® S7



Date: 12.03.03
File: PRO1_11E.16

SITRAIN Training for Automation and Drives

**Diagnosing Hardware** The function opens the station that can be accessed online and gives you information about the status or operating mode of the modules. You can see that there is diagnostic information for a module when you see the diagnostic symbols that indicate the status of the associated module or the operating mode of the CPU. When you double-click the symbol, a screen with further information pops up.
In the example shown, the analog input module (slot 7) has triggered a diagnostic interrupt. As a result, the CPU has gone into the STOP mode. Both modules have been given symbols accordingly. By double-clicking the CPU, you would see the diagnostic buffer. By double-clicking the analog module, you would be given the relevant diagnostic data. In the example, the external auxiliary voltage (supply voltage) of the analog module has failed.

**Opening the Tool** You can call the function as follows:

- in the SIMATIC® Manager
  - using *PLC -> Diagnostic/Setting -> Hardware Diagnostics*
  - in the *online view,* with a double-click on the *Hardware* icon of the station
- in HWConfig, by opening the station using *online*

**Customizing Settings** If you have selected the *Options -> Customize -> View* menu options in the SIMATIC® Manager and activated (checked) the *"Display Quick View when Diagnosing Hardware*" checkbox, only a list of faulty modules will be displayed instead of the full "Diagnosing Hardware" view.

**Note** You will find an actual application example on this test function in the Analog Value Processing chapter.

## Exercise: Finding STOP Errors and Eliminating Them

| Step | What to Do | Result |
|---|---|---|
| 1 | Perform a CPU memory reset and switch the CPU to the STOP mode | Your user program in the CPU is deleted |
| 2 | Copy the S7 program "ERROR _16" or "ERROR _32" from the project "Error" as a hardware-independent program into your project "My_Project" | The hardware-independent S7 program "Error_16" ("Error_32") exists in your project "My_Project" |
| 3 | Download the system data from the Blocks folder that is assigned to the CPU S7 program "My_Program" into the CPU | The hardware of the PLC is once again assigned parameters as before so that even the flashing clock memory is once again available |
| 4 | Doawnload all blocks from the S7 program "ERROR_16" ("ERROR_32") into the CPU and carry out a complete restart | The faulty program is downloaded and the CPU goes into the STOP mode after the complete restart |
| 5 | Find and eliminate the errors that lead to the Stop state. While doing so answer the questions below about the errors | The CPU remains in RUN |

SIMATIC® S7


Date: 12.03.03
File: PRO1_11E.17

SITRAIN Training for Automation and Drives

**Task**

The S7 program *Error_16* (*Error_32*) that you copied, corresponds exactly to the program that you developed so far in this course. It contains STOP errors that you are to find and eliminate in this exercise so that the CPU remains in RUN after a complete restart.

**What To Do**

Please note that every time you make a STOP error correction, you have to carry out a complete restart of the CPU. If the CPU again goes into the STOP state after a complete restart then another STOP error still exists.

- Proceed as indicated in the slide
- As you correct the errors, answer the following questions on the occurring errors:

  - 1st. Error: Interruption in FC 18:
    Even if it is not relevant for troubleshooting at this point:

    Which value does the L STACK show for the tempory variable *#Setpoint* at the time of interruption:

    Value of the variable *#Setpoint*:.........................................

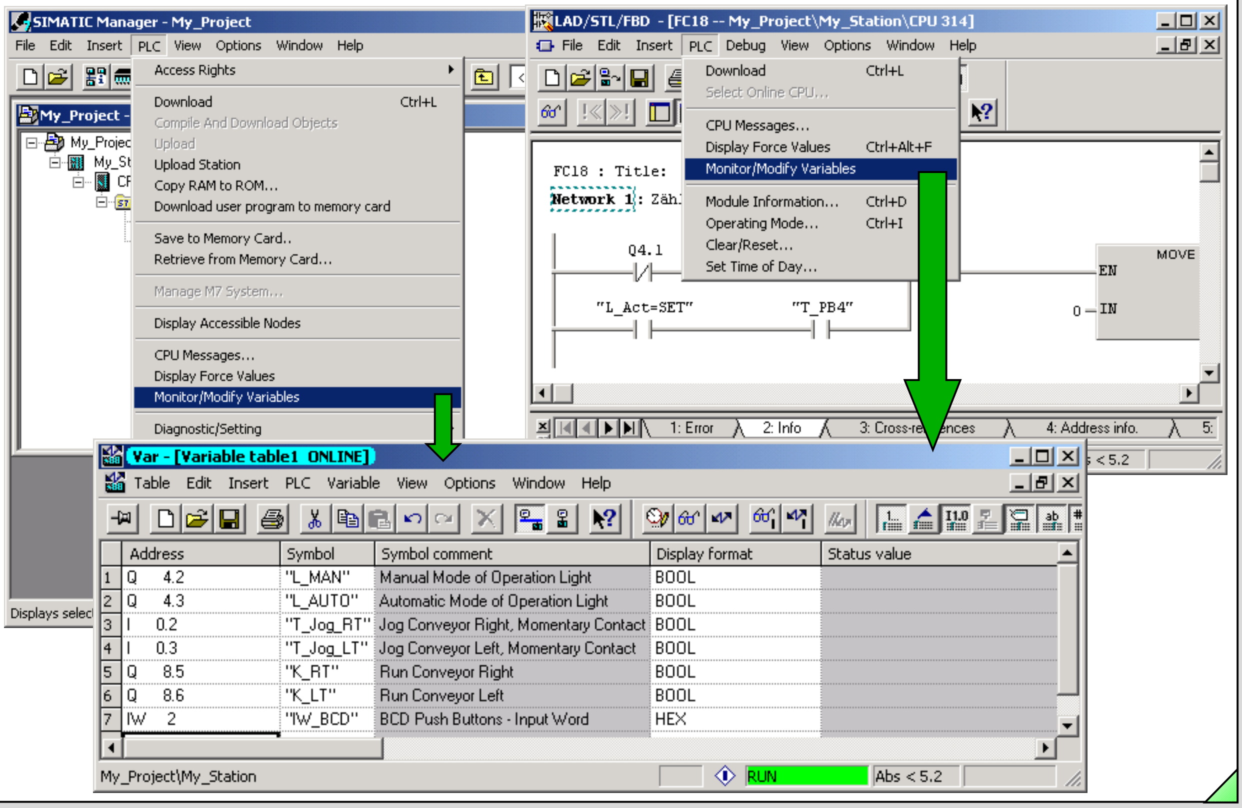
  - 3rd. (last) Error: Interruption in FC 20:
    From the B STACK, determine which block calls the FC 20 with the faulty actual address:

    calling block: .........................................................

**Note**

In addition to the errors (STOP errors) recognized by the system, the program also contains functional errors (RUN errors), so that the correct program function still doesn‘t exist even after the STOP errors have been eliminated. The RUN errrors will be eliminated in the following exercises.

## Calling the "Monitor/Modify Variables" Tool





**Area of Use**

The Variable Address Table (VAT) is used to view (monitor) and change (modify) a CPU memory area's address value in a defined format. The "Monitor/Modify Variables" function is started from the SIMATIC® Manager or from the LAD/STL/FBD Editor.

**Design of the Variable Table**

The variables you choose are entered in a VAriable Table (VAT). With the exception of block-local, temporary variables, you can monitor and/or modify all variables or addresses.

You can select the columns of the variable table displayed in the *View* menu. The columns have the following meanings:

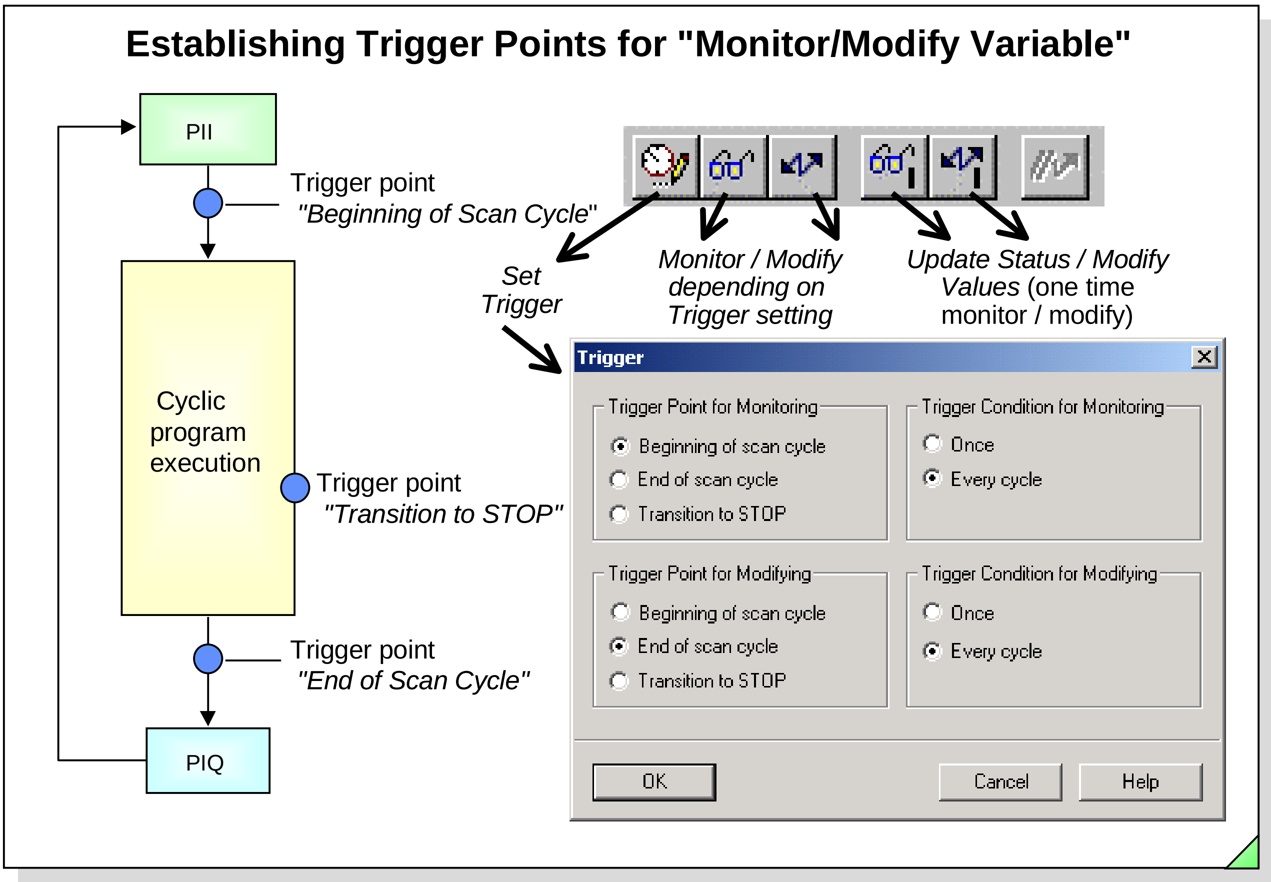
- **Address**: absolute address of the variable.
- **Symbol:** symbolic name of the variable
- **Symbol comment:** comment on the variable displayed
- **Display format**: a data format you can choose per mouse click (such as binary or decimal), in which the contents of the variable is displayed
- **Status value**: value of the variable in the selected status format
- **Modify value**: value to be assigned to the variable

**Saving the Variable Table**

You can use *Table -> Save* or *Table -> Save as...* to save a variable table.
The first time a variable table is saved, the *Save As...* dialog window opens. The *Save as ...* window allows the user to select the Blocks folder that the variable table will be stored in. This Storage path does not default to the Project/Program which the user may have currently open.
You can give the variable table any name you choose. The name is inserted as the symbolic name in the symbol table.
You can reuse saved variable tables for monitoring and modifying, making it unnecessary to re-enter the variables.

**Note**

To check the input and output wiring (regardless of the user program), you can also call the *Monitor/Modify Variables* tool (VAT) directly from the *HWConfig* tool (see the *HWConfig* chapter).




SIMATIC® S7


Date: 12.03.03
File: PRO1_11E.19

SITRAIN Training for Automation and Drives

**Trigger Points**

You can establish the trigger points for *Monitoring* and *Modifying* using *Variable -> Set Trigger...* or by using the icon shown here on the left.

The "Trigger Point for Monitoring" specifies when the values of the variables being monitored on the screen are to be updated. The "Trigger Point for Modifying" specifies when the given modify values are to be assigned to the variables being modified.

**Trigger Condition**

The "Trigger Condition for Monitoring" specifies whether the values are to be updated on the screen once when the trigger point is reached or every cycle (when the trigger point is reached).

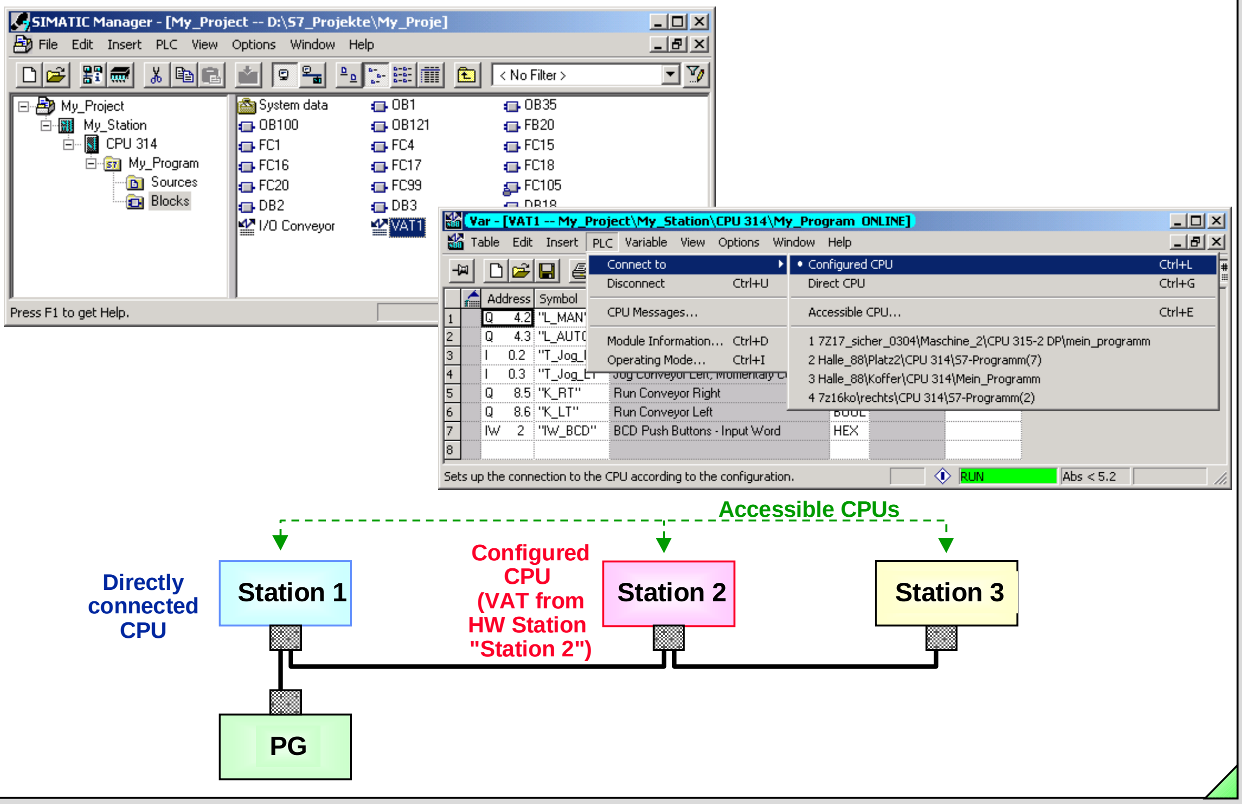
The "Trigger Condition for Modifying" specifies whether the given modify values are to be assigned to the variables being modified once or every cycle (every time the trigger point is reached).

**Area of Use**

The following tests, among others, can be implemented with the appropriate selection of trigger points and conditions:

- **Wiring test of the inputs*:*** (also possible in HW Config)
  *> Monitor Variable*
  Trigger point: *Beginning of Scan Cycle,* Trigger condition: *Every Cycle*
- **Simulate input states** (user specified, independent of process):
  *> Modify Variable*
  Trigger point: *Beginning of Scan cycle,* Trigger condition: *Every Cycle*
- **Differentiation between hardware / software errors**
  (an actuator that should be activated in the process is not controlled)
  *> Monitor Variable,* in order to monitor the relevant output
  Trigger point: *End of Scan Cycle,* Trigger condition: *Every Cycle*
  (output state = ´1´ > program logic OK > process error (hardware)
  (output state = ´0´ > program logic error (such as double assignment)
- **Control Outpus** (independent of the program logic)
  *> Modify Variable*
  Tigger point: *End of Scan Cycle*, Trigger condition: *Every Cycle*

# Opening a Variable Table and Establishing a Connection to the CPU







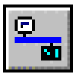
**Area of Use**

You can double-click to reopen variable tables that are stored in a Blocks folder of an S7 program at a later time for test purposes.

However, before you can activate the "Monitor" and "Modify" functions, an online connection to a CPU has to be established.

Using the menu commands *PLC -> Connect to . . .* provides three connection choices: Configured CPU, Direct CPU and Accessible CPU. There are toolbar icons for connecting to the Configured CPU or Direct CPU.

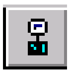
**Configured CPU**



If the variable table is stored in the Blocks folder of a hardware-dependent S7 program (assigned to a CPU in the project view), a connection to the station with the MPI address is established. This station was also assigned to the higher-level CPU with the help of the HW Config tool.

If the variable table is stored in the Blocks folder of a hardware-independent S7 program folder (assigned directly to the project root in the project view), the MPI address of the hardware-independent S7 program can be established using the Properties dialog of the program folder while in the online view of the SIMATIC® Manager.

**Direct CPU**



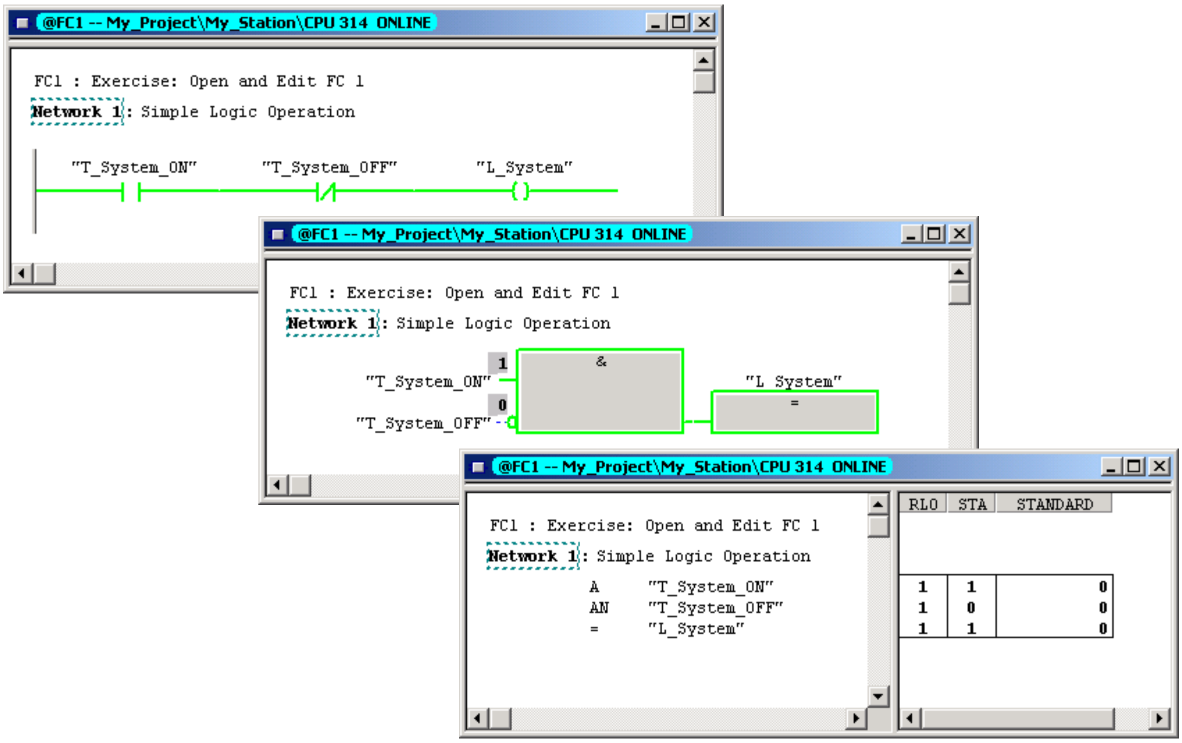
This menu command establishes an online connection between the active variable table and the CPU to which the MPI cable from the PG/PC to the programmable controller is connected. The PG can determine to which PLC it is connected through the MPI cable.

**Accessible CPU**

This menu command establishes an online connection between the active variable table and a CPU that you select. If the user program is already linked to a CPU, this menu command can be used to change the CPU.

Select a CPU to which you want to establish an online connection in the dialog box. You can choose between configured and accessible CPUs.

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




SIMATIC® S7


Date: 12.03.03
File: PRO1_11E.21

SITRAIN Training for Automation and Drives

**Area of Use**

The *Monitor Block* test function is used to be able to follow the program execution within a block. For this, the states or contents of the addresses used in the block at the time of program execution are displayed on the screen.

**Monitor**

You can activate the "Monitor" ("Block Status") test mode for the block which is currently open in the LAD/STL/FBD Editor by clicking the *Glasses icon* or by selecting *Debug -> Monitor.*

At the beginning of the test function, it is insignificant whether the block to be monitored is opened *online* or *offline* in the Editor. Should, however, the block opened offline not concur with the block saved online in the CPU, you first either have to open the block saved online or load the block opened offline into the CPU and then monitor it .

In the test mode, the states of the addresses and LAD / FBD elements are displayed in different colors. You define these by selecting the menu option *Options -> Customize*:

Examples:

- Status fulfilled -> "Element is displayed in green"
- Status not fulfilled -> "Element is displayed in blue"

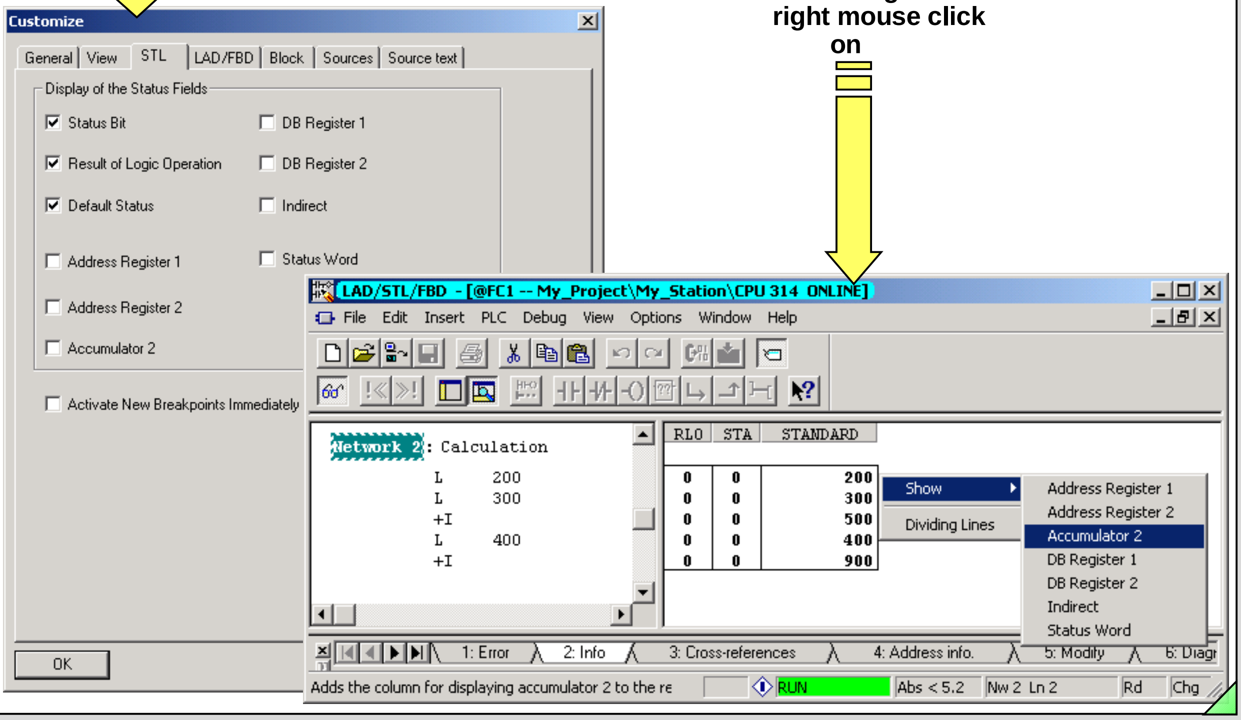
**Notes**

The Status display is only active when the CPU is in RUN mode and the instructions to be monitored are being processed!

# Information Displayed with "Block Debug > Monitor"

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

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




SIMATIC® S7


Date: 12.03.03
File: PRO1_11E.22

SITRAIN Training for Automation and Drives

**Selecting Information** When you monitor blocks in the STL language, you can select which information is to be displayed on the screen. By default, the RLO, STATUS and STANDARD (Accumulator 1) information is displayed.

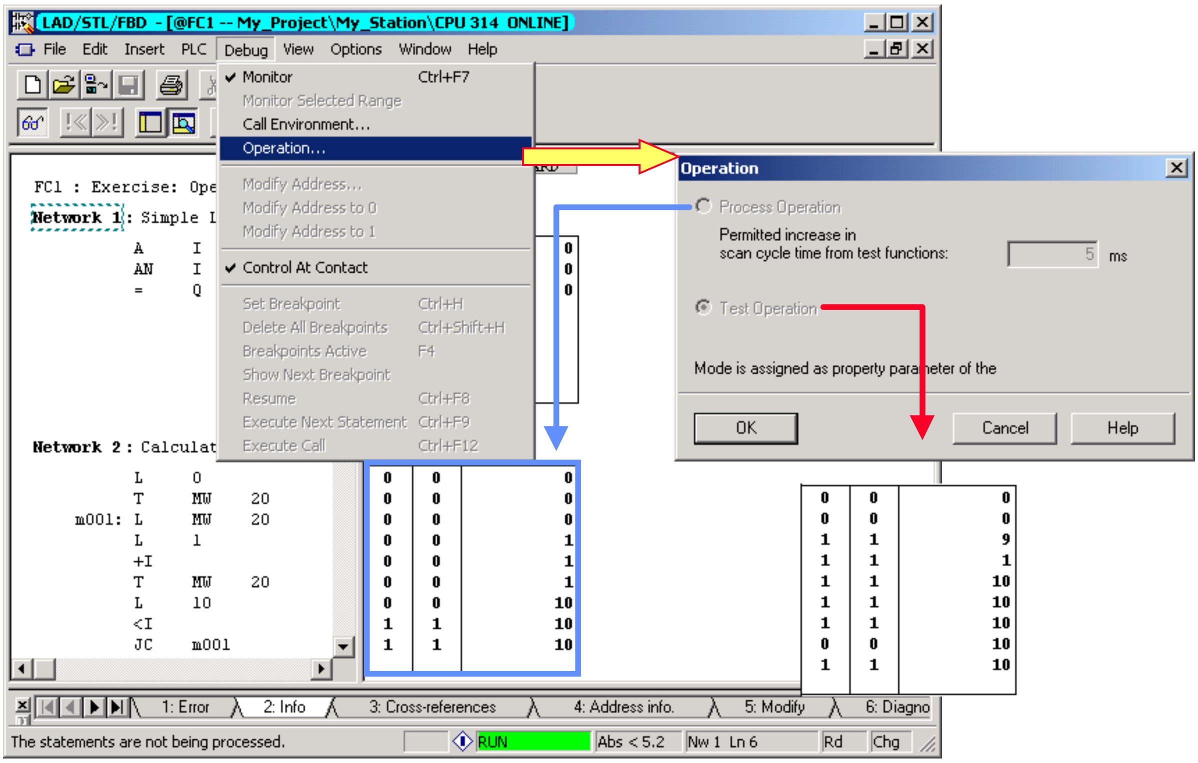
- You can select which information is to be preset or displayed by default using the menu options *Options* > *Customize* > *STL*.
- While the test function is running, you can, at any time, show or hide information that is to be displayed in the table by using the right mouse button.

**Displayable Information**

- **RLO:** Result of logic operation
- **STAT:** Status of the (binary) address
- **Default** (Accumulator 1): Contents of Accumulator 1
- **Accumulator 2:** Contents of Accumulator 2
- **AR1:** Address register 1, only meaningful with register indirect addressing
- **AR2:** Address register 2, only meaningful with register indirect addressing
- **DB Register 1**: Number of the global or 1st. DB, that is just open
- **DB Register 2:** Number of the local or 2nd. DB or instance DB, that is just open
- **Indirect:** Contents of MD..., DBD... or LD..., which is used in memory indirect addressing (such as the instruction L IW [MD 100] ).
- **Status Word:** States of the Status Bits (OV, OS, BR, .........)

**Display Format** You can select the data format (decimal, hexadecimal, .....), in which the register contents are to be displayed using the right mouse button. For this, click on the Register column with the right mouse button and select the data format.

## Selecting Process and Test Operation




SIMATIC® S7


Date: 12.03.03
File: PRO1_11E.23

SITRAIN Training for Automation and Drives

**Test Modes**

There are two modes of operation for the test mode, which differ in their effect on the scan cycle time of the user program:

- **Test Operation** (in the LAD/STL/FBD Editor) **Test Mode** (in HW Config)
- **Process Operation** (in the LAD/STL/FBD Editor) **Process Mode** (in HW Config)

**Setting**

For most CPUs, you make the setting for process or test operation when you assign the CPU parameters using *HW Config*:
*CPU -> Protection -> Process Mode /Test Mode*

For the older CPUs, the setting can be made in the STL/LAD/FBD Editor directly using *Debug -> Operation -> Process operation/Test operation*.

**Test Operation**

In test operation, all test functions can be performed without restrictions. The status of programmed loops is determined every time they are executed.

The scan cycle time can be increased considerably due to the updating of the test function Program Status and thus trigger a CPU STOP because of scan time overrun.

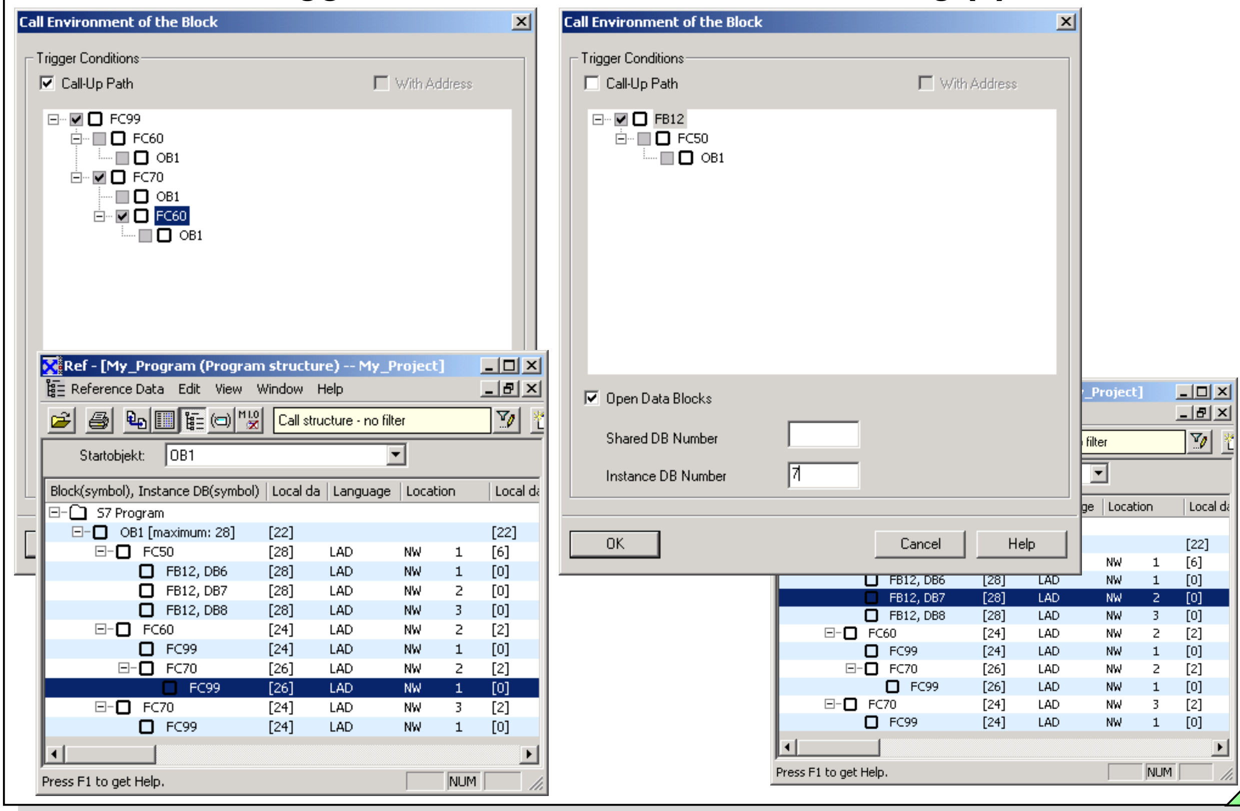
**Process Operation**

In process operation, the test functions are restricted in such a way that the permissible scan cycle time increase that you select in the CPU parameter assignment is not exceeded. As well, the status (Accumulator 1, Accumulator 2 etc.) of programmed loops is only displayed for the first execution in process operation.

The following expanded test functions can also not be selected in process operation:

- Breakpoint (STL Editor: *Debug -> Set Breakpoint*, etc.):
  stops the program execution at a specific location (breakpoint) and then continues with the execution (e.g. instruction by instruction, in order to test the individual executions of program loops) on the user's initiative.
- Trigger Points for monitor block:
  Monitoring a particular block call of blocks that are called several times in the user program.

# Trigger Conditions for Block Monitoring (1)






Date: 12.03.03
File: PRO1_11E.24

SITRAIN Training for Automation and Drives

**Area of Use**

In order to monitor a parameter-assignable block (FCs, FBs) that is called several times in the user program and is executed each time with a different actual parameter, you can use the menu option *Debug -> Call Environment.* With Call Environment, you can specify which call or which execution of the block is to be monitored.

If the block to be monitored is already open in the LAD/FBD/STL Editor, you have two ways of triggering a specific call or of defining a *call environment* for monitoring the block: Call-up Path and Open Data Block.

**Call-Up Path**

By using *Call-up Path* (triggering using B STACK), you can specify the path through the program structure up to the block that is to be monitored. The *Call-up Path* can have up to three nesting depths or up to three blocks. Thus, the *3rd. block* calls the block to be monitored (*status block*), the *2nd. block* calls the 3rd. etc.

*Example* (in the slide on the left hand side):

FC 99 (*status block*) is to be monitored when it is called from FC 70 (*3rd. Block*) AND FC 70 is called from FC 60 (*2nd. Block*).

**Open Data Blocks**

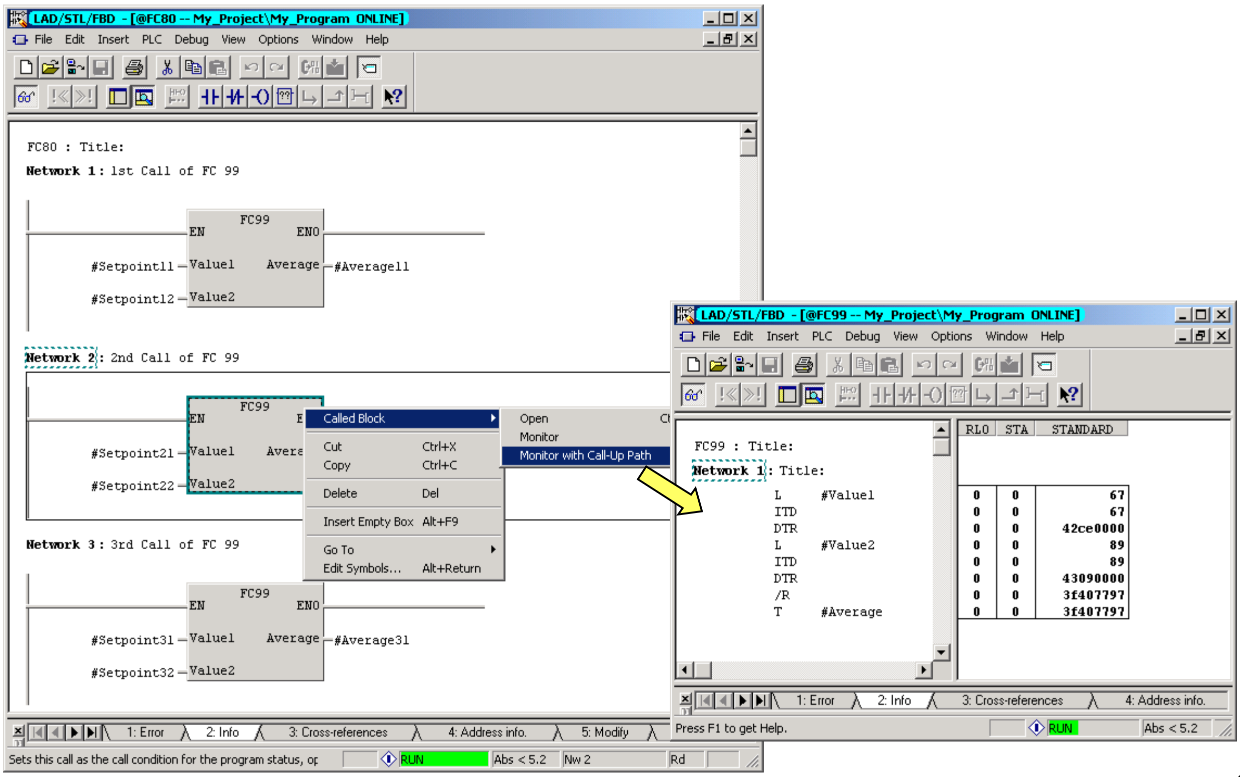
With the trigger condition *Open Data Blocks,* you can specify that a block is to be monitored when at it's call a particular data block is valid or open. For this you can trigger using the *Global DB Number* (DB Register 1) or the *Instance DB Number* (DB Register 2).
*Example* (in the slide on the right hand side):

FB 12 (status block, not visible in the slide) is to be monitored when it is called with *Instance data block DB 7*.

**Notes**

- *Test Mode (Operation)* must be selected
- The block call that is the first to fulfill the given trigger condition is always the block call that is "monitored".

# Trigger Conditions for Block Monitoring (2)




Date: 12.03.03
File: PRO1_11E.25


**Area of Use**

In the example shown, FC 99 is called three times in FC 80. That is, the call-up path for all three FC 99 calls is the same. Accordingly, it is not possible to specifically call one of the FC 99 calls by defining a call-up path as trigger condition. A triggering using *Open Data Blocks* is also not possible since neither a global nor an instance DB is relevant with the FC 99 calls shown.

**What To Do:**

For the specific monitoring of an FC call, proceed as follows:

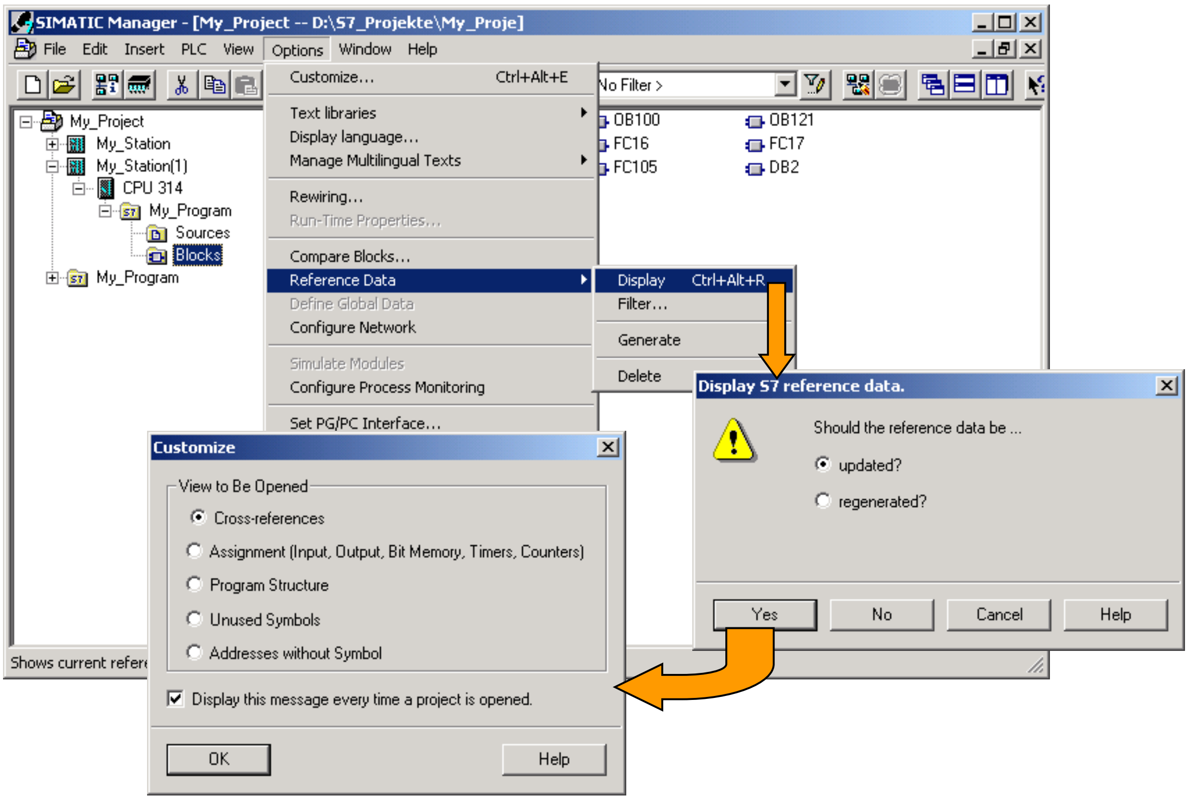
- Open *online* the block (in the example FC 80) that calls the block to be monitored (in the example FC 99) (not the block that is to be monitored ! )
- With the right mouse button, click on the call of the block to be monitored (Call Box in LAD/FBD or Call Line in STL)
- In the dialog box that appears, select *Called Block > Monitor with Call-Up Path*

The block to be monitored (in the example the FC 99) is then opened *online* and the test function *Monitor Block* is activated.

**Notes**

The function described is possible on all S7-400™s. For the S7-300™, it will only be available starting from the compact versions (delivery after October 2001).

## Displaying Reference Data




SIMATIC® S7


Date: 12.03.03
File: PRO1_11E.26

SITRAIN Training for Automation and Drives

**Area of Use**

For extensive programs, it is particularly necessary when troubleshooting to have an overview of where which address is scanned or assigned, which inputs and outputs are actually used, or how the entire user program is generally structured with regards to the call hierarchy.

The "Reference Data" tool gives you an overview of the structure of the user program as well as the addresses used. The reference data are generated from the user program saved *offline*.

For functional errors, that can be traced back to logical program errors for example (such as double assignment), you will find the "Program Status" tool together with the "Reference Data" tool useful.

If, for example, a logic operation is not fulfilled because a memory bit is not set, you can use the "Reference data" to determine where this memory bit is assigned.

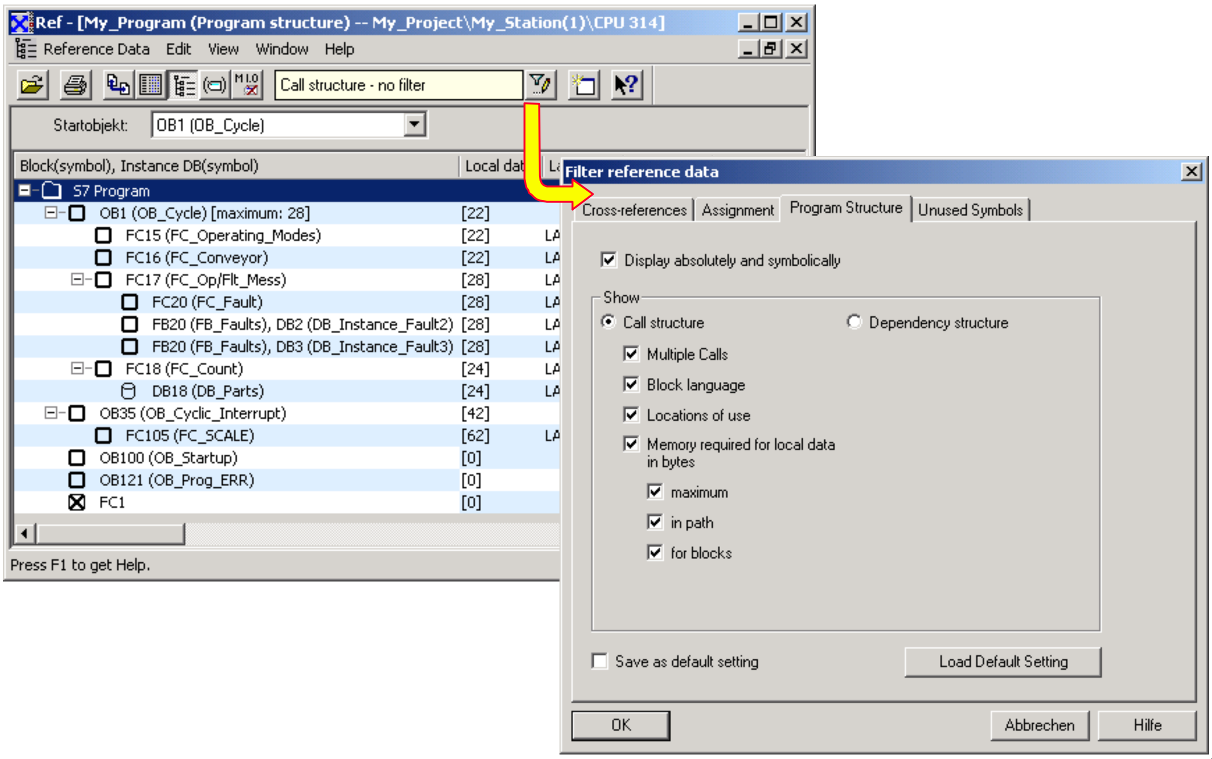
**Reference Data**
**…Generate**
**…Display**

You can trigger the generating and displaying of reference data in the SIMATIC® Manager (when the "Blocks" folder is selected offline) or in the LAD/STL/FBD Editor using *Options -> Reference Data -> Display* or *>Filter and Display.*

**…Filter**

The reference data consist of various lists (see *Customize* in the slide) that are displayed as filtered data (individually), (regardless of whether the item *Display* or *Filter and Display* was selected in the *Options* menu). When you select *Display Reference Data,* you can choose in the Customize dialog which list is to be displayed first. Then you can choose any of the different lists.

# Displaying the Program Structure




SIMATIC® S7


Date: 12.03.03
File: PRO1_11E.27

SITRAIN Training for Automation and Drives

**Program Structure** The program structure describes the call hierarchy of the blocks in an S7 user program.

**Filter** Depending on the settings of the filter, the program paths are displayed in a Tree structure or as "Parent/child structure" (in each case the calling and the called block are displayed).

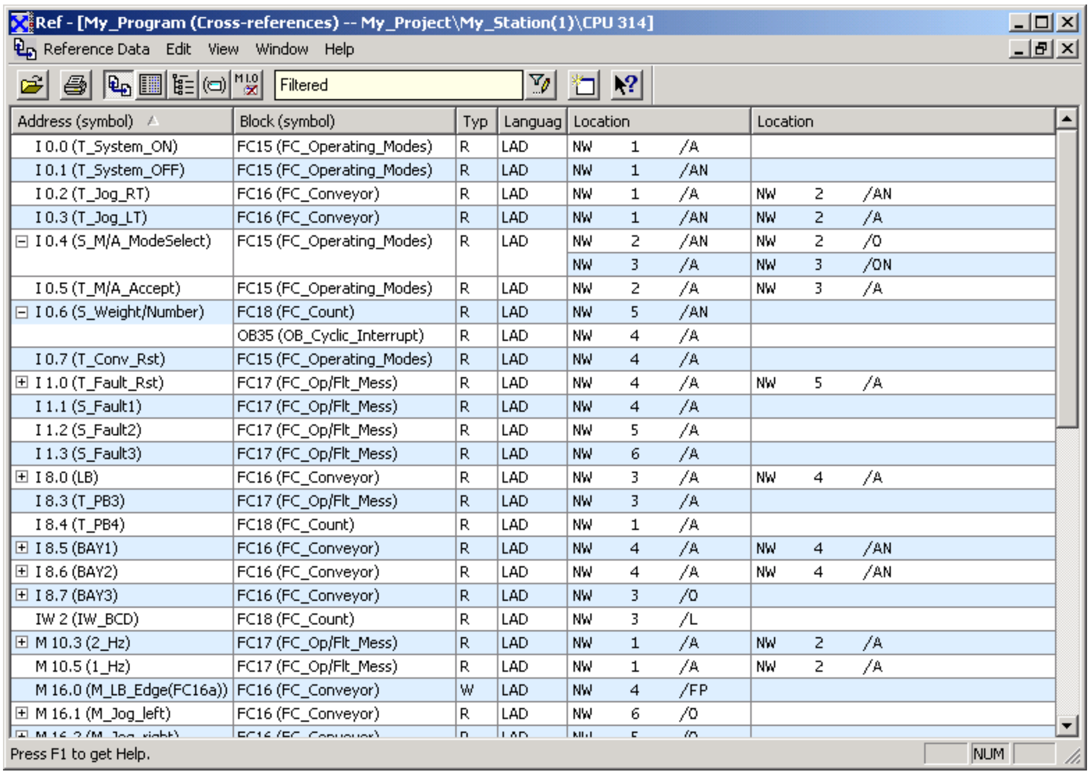
**Symbols** The following symbols are possible only in the tree structure display:

< maximum : nnn >
- the maximum memory requirement (in bytes) of the local data is given in the root of the tree structure.

[ nnn ]
- per path, the maximum memory requirement (in bytes) of the local data is stated at the last block of every program path.

| Symbol | Meaning |
|---|---|
|  | Block called normally (CALL FB10) |
|  | Block called conditionally (CC FB10) |
|  | Block called unconditionally (UC FB10) |
|  | Data block (CALL DB10, L DB10.DBW0) |
|  | Recursion |
|  | Recursion and called conditionally |
|  | Recursion and called unconditionally |
|  | Block not called |

# Displaying Cross References



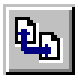

| Address (symbol) | Block (symbol) | Typ | Languag | Location | Location |
|---|---|---|---|---|---|
| I 0.0 (T_System_ON) | FC15 (FC_Operating_Modes) | R | LAD | NW 1 /A | |
| I 0.1 (T_System_OFF) | FC15 (FC_Operating_Modes) | R | LAD | NW 1 /AN | |
| I 0.2 (T_Jog_RT) | FC16 (FC_Conveyor) | R | LAD | 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 | |

SIMATIC® S7


Date: 12.03.03
File: PRO1_11E.28

**SITRAIN** Training for Automation and Drives

**Area of Use**



The Cross References (list) gives you information about how which addresses are used in which blocks (with which instruction). Thus, you can find out, for example, where in the entire user program a memory bit is (double) assigned. You open the cross references with *View -> Cross References* or by clicking the icon you see here on the left.

You can display the cross references for all inputs, outputs, bit memories, timers, counters, blocks (except OBs), peripheral inputs and outputs.

**Cross Reference of Individual Addresses**

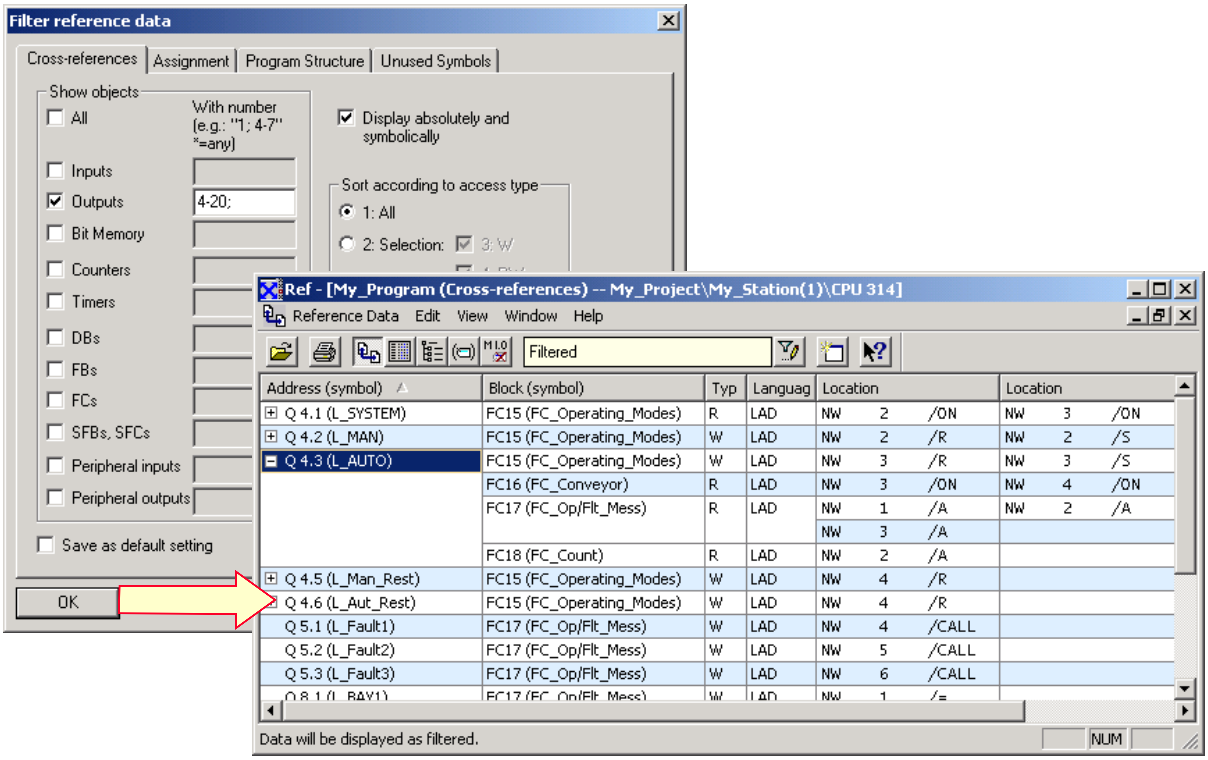
When you select an address in the cross reference list, you can open a new window using the right mouse button and *View -> Cross Reference for Address.* This window contains only the cross references for this one address.

**Structure**

The cross references list is structured as a table. This list has the following columns:

- **Address:** absolute address of the operand
- **Symbol:** symbolic name of the address
- **Block:** block in which the address is used
- **Type:** read-only (R) or write-only (W) access
- **Language:** programming language in which the block was created
- **Details:** instruction which uses the address

## Filtering Cross References






Date: 12.03.03
File: PRO1_11E.29

SITRAIN Training for Automation and Drives

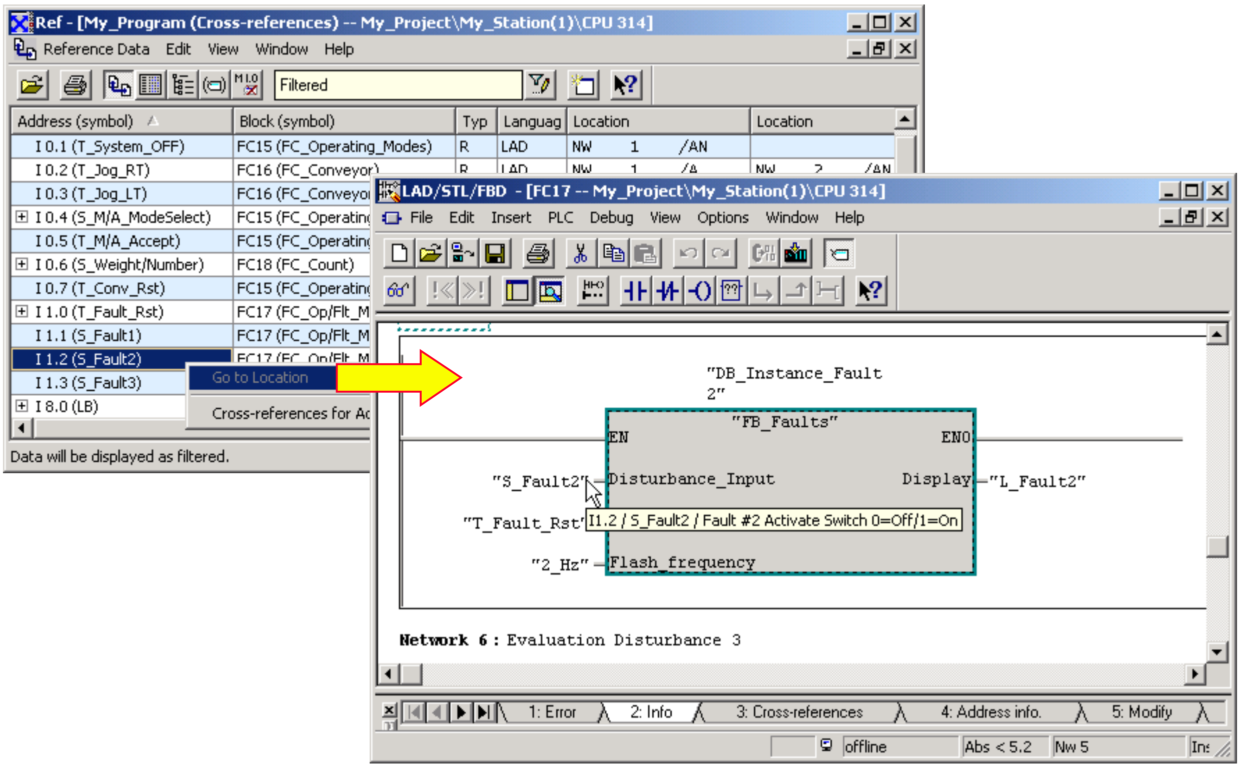
**Filtering Cross References**

You can also use the Filter function to display individual addresses or address areas separately. You access the *Filter reference data* dialog through the *View* menu.

The entries in the Filter Dialog have the following meaning:

- **Objects**
  You determine which address type is to be listed by activating the appropriate check box.
- **…with Number**
  The filter area specifies the address area to be displayed. You can also specify several partial areas.
  The filter area entry "10-50; 70; 100-130" means that the address 70 and the address area from 10 to 50 and from 100 to 130 is to be displayed.
- **Display absolutely and symbolically**
  When this option is activated, the addresses, just like in the slide, are displayed with a symbol. When the option is deactivated they are shown absolutely.
- **Access Type**
  In the default setting, all access types are displayed. You also have the option of choosing "Selection", in which case you then have to choose the access type by clicking on the check box(es), (for example - W - for write-only access) or "Only multiple assignments with operation = ".
- **Default Setting**
  If the settings are to be accepted for the next time the "Display Reference Data" application is started, you have to activate the "Save as default setting" check box. The basic setting or the default setting you saved is restored with the "Load Default Setting" button.

## Block Correction using Cross References







**Handling**

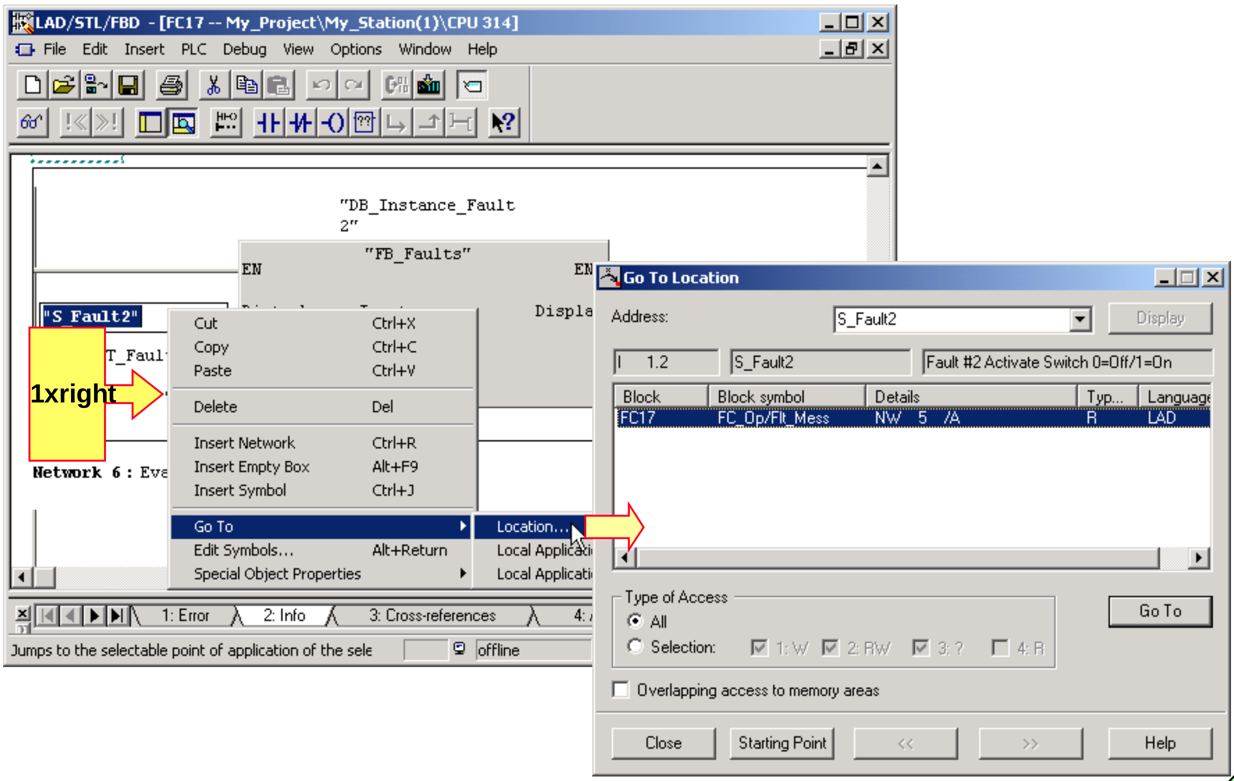
When you double-click an address in the cross references list, you start the LAD/FBD/STL Editor and open the block where the selected address is used. The cursor is located in the network (LAD/FBD) or in the line (STL) in which the address is used.

**Note**

The reference data are generated only from the blocks that are stored in the offline data management!

For that reason, you must make sure that the blocks stored *online* and *offline* are identifcal for troubleshooting. You can check this in the SIMATIC® Manager using *Options -> Compare Blocks.*

## Go To Location




SIMATIC® S7


Date: 12.03.03
File: PRO1_11E.31

SITRAIN Training for Automation and Drives

**Function and Area of Use**

In troubleshooting, it is often necessary only to determine where one address is used or assigned in the program. In this case, it makes more sense to call the "*Go To -> Location*" function instead of the cross references list. The Go to Location is called directly from the LAD/FBD/STL Editor and gives you an excerpt from the cross references list for the specific address.
When you select the option *Overlapping Access to Memory Areas,* word-by-word accesses to an address are also displayed, for example.

**Handling**

Using the right mouse button, click on the address. The *Go to Location* dialog window appears. Its entries have the same meaning as those in the cross references.

If an entry is of particular interest or an indicated program location is to be shown, you use *Go To -> Location* to open the indicated block with the Editor.

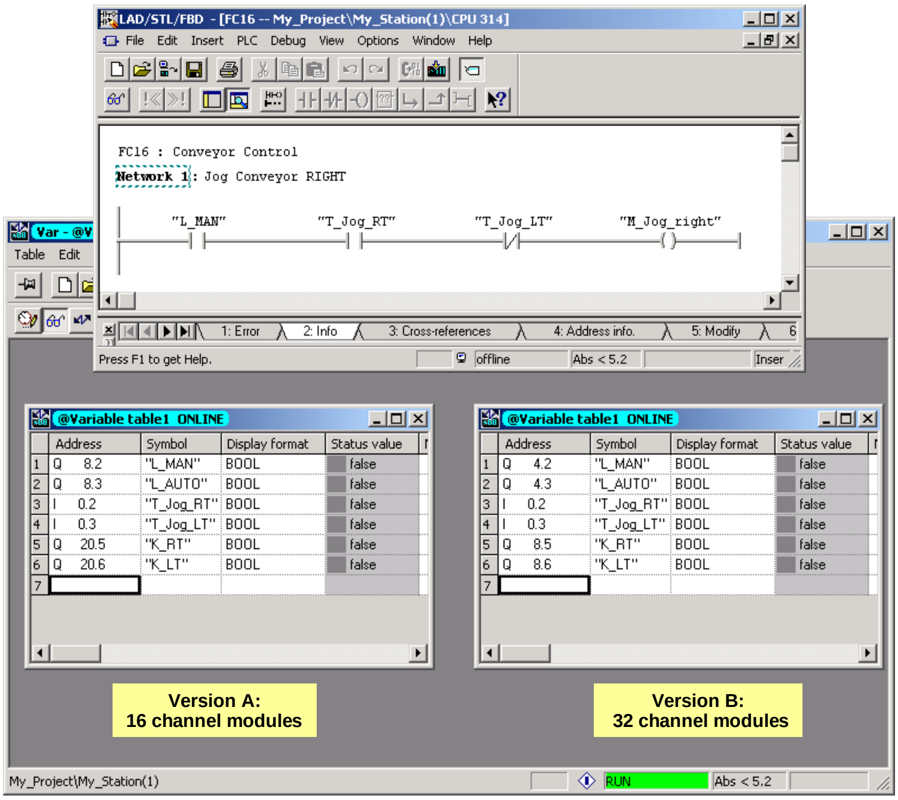
In the above example, the program location at which the input I 0.2 is scanned (Access Type R) is of interest. After selecting the relevant line, you can use the *Go To* button to open FC 16, NW 1 directly.

When you click the *Starting Point* button, you return to the beginning.

**Type of Access**

By default, all accesses to the addresses are displayed. When you choose "Selection", you can display, for example, write-only accesses (assignment, set, reset).

## Exercise: Testing Motor Jog




SIMATIC® S7


Date: 12.03.03
File: PRO1_11E.32

SITRAIN Training for Automation and Drives

**Task**

The *Error_16* (*Error_32*) S7 program that you copied is to fulfill exactly the same functions as the program that you have programmed up until now. Since you eliminated the STOP errors in the last exercise, you are now to eliminate the logical errors in this one.

In the search for the first logical error you are to become familiar with using the two test functions *Monitor block* and *Monitor variable* in combination (see slide).

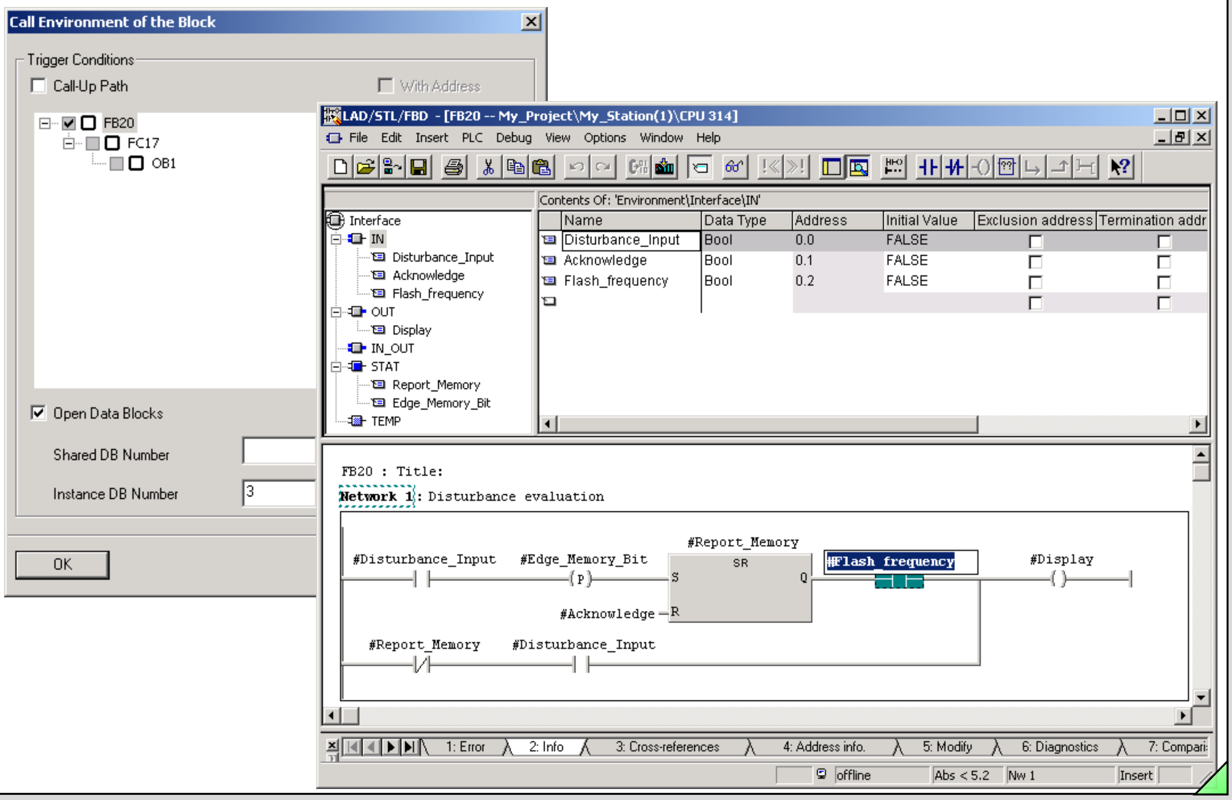
**Function Test No. 1:**

Check to see whether you can jog the conveyor motor to the right ("K_RT") using the pushbutton "T_Jog_RT" and to the left ("K_LT") using the pushbutton "T_Jog_LT" when you have switched on MANUAL mode ("L_MAN" = ´1´).

**What To Do:**

1. Enter a variable table with the addresses shown in the slide
2. Using *Variable -> Trigger,* specify *End of Scan Cycle* as the trigger point and *Every Cycle* as the trigger condition, so that you can monitor the output states relevant to the conveyor motor after the program has been executed. Activate the test function.
3. In the Editor - without exiting *Monitor Variable* - monitor FC 16, where the jogging of the conveyor motor is programmed.
4. Select the size of the windows for the two test functions so that you can arrange them as shown in the slide. The states of the outputs are visible during program execution in the *Monitor block* test function, while in the *Monitor variable* test function they are visible at the end of the cycle. The test functions show different states for the output "K_RT", which leads to the conclusion that the error function occurred because of a double assignment.
5. Find and correct the double assignment as follows:
   *right mouse button on K_RT -> Go to Location*

## Exercise: Testing the Evaluation of Disturbance 3







**Task**

In the search for the second logical error you are to use the *Monitor block* test function with triggering of a specific block call.

**Function Test No. 2**

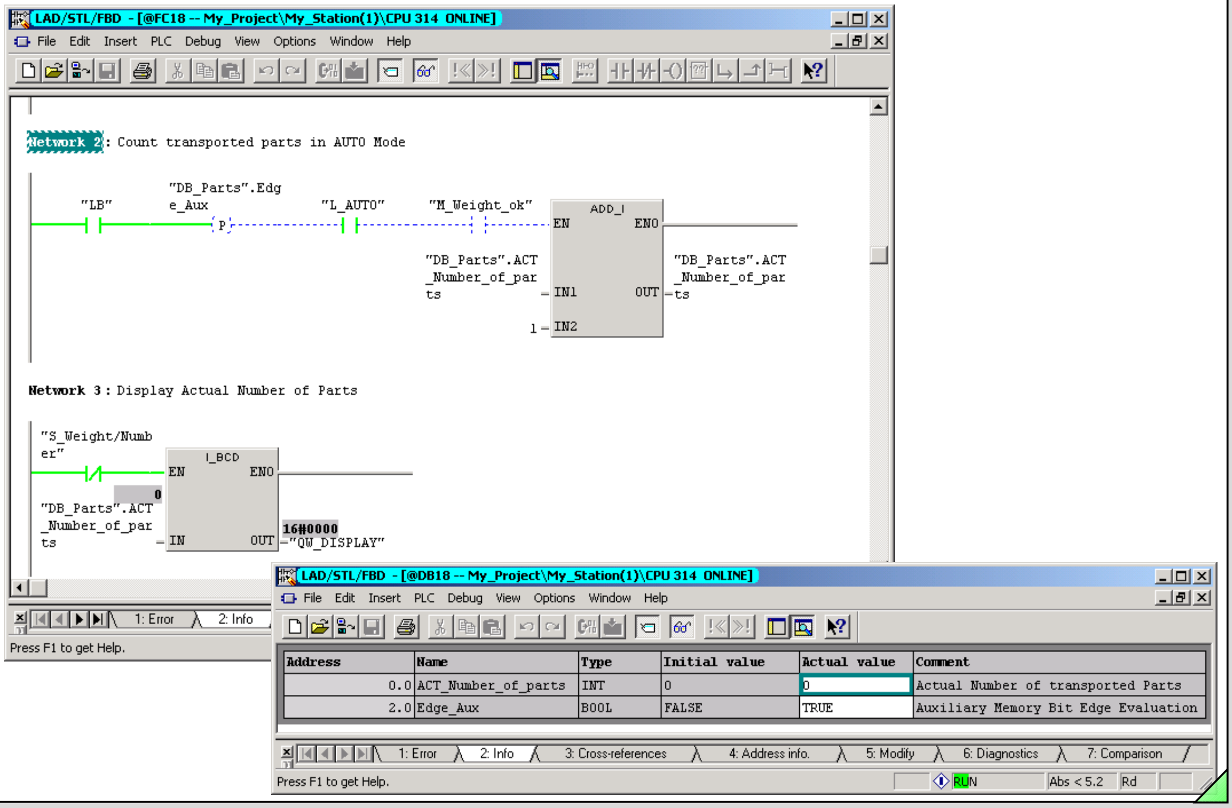
Check whether the occurrence of *Disturbance 3* (Switch I 1.3 on the simulator) is displayed with a flashing light on the simulator LED Q 9.3 / Q 5.3, and whether the flashing light switches to a constant light when you acknowledge using the simulator pushbutton I 1.0 .

**What To Do:**

1. After *Disturbance 3* occurs, no flashing light is displayed. To troubleshoot, monitor FB 20 in the Editor when it is called with instance DB 3 to evaluate Disturbance 3.
   *Debug -> Call Environment -> Trigger condition: Open Data Blocks, Instance DB Number: 3* (see slide upper left).

   (Note: *Mode-> Test Mode* must be set to be able to trigger!)
2. Monitoring FB 20 shows that no flashing light is displayed because the IN Parameter has #Flash_frequency constant ´0´ . Since it concerns a parameter, the error is not in the FB 20 itself, but in the actual parameters that are passed on to the FB 20 by the calling block.
3. The FB 20 is called in several locations in the program. Find out in which block the FB 20 with instance DB 3 is called to evaluate Disturbance 3.
   *SIMATIC® Manager Options -> Reference data -> Display -> Program structure*
4. Correct the FB 20 call in the calling block and retest the evaluation of the disturbance function.

# Exercise: Testing the Display of the Quantity




SIMATIC® S7


Date: 12.03.03
File: PRO1_11E.34

SITRAIN Training for Automation and Drives

**Task**

In the search for the third logical error you are to use the *Monitor data block* test function, among other things.

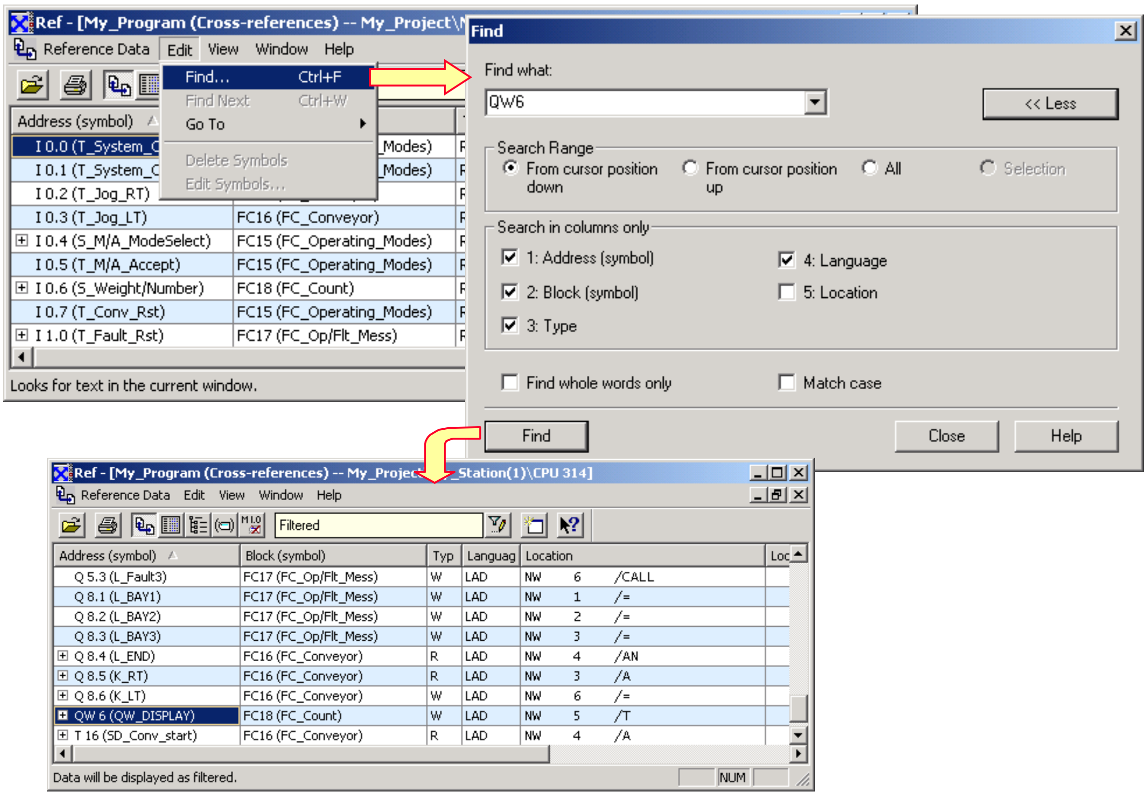
**Function Test No. 3**

Check whether the number of transported parts is recorded correctly and displayed on the BCD digital display.

**What To Do:**

1. The counting and displaying of the number of transported parts is programmed in FC 18. FC 18 calculates the current number through addition in the variable *"DB_Parts.ACTUAL_Quantity"* (DB18.DBW0). First of all, open DB 18 with the Editor and monitor online the contents of the variable "ACTUAL_Quantity" (DBW 0)
(Reminder: Before you can monitor a DB you must switch to *Data View* using the *View* menu option )
2. It appears that the FC 18 block does not deposit the correct number of parts in the variable "*DB_Parts.ACTUAL_Quantity*". Therefore, open the FC 18 with the Editor and test it with the *Monitor block* function.
3. Correct the FC 18 and retest the function.

# "Find" In Reference Data




SIMATIC® S7


Date: 12.03.03
File: PRO1_11E.35

**SITRAIN** Training for Automation and Drives

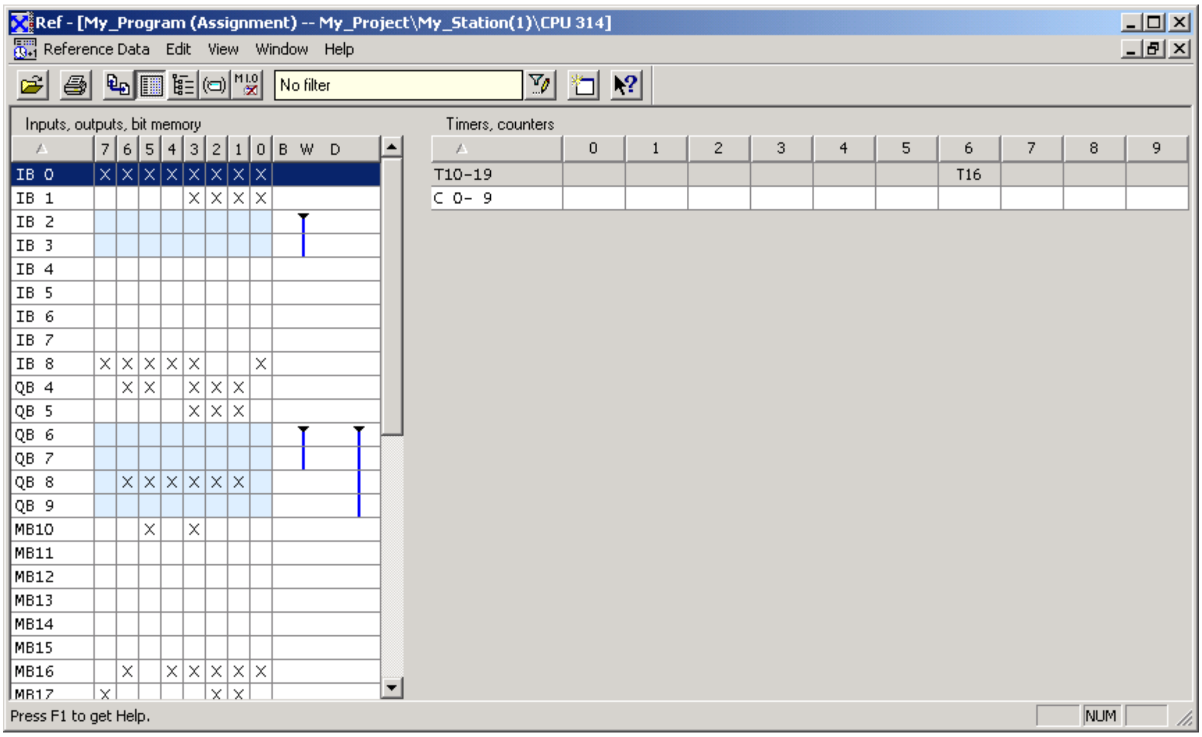
**Find** While reference data are displayed, you can start a search for addresses (character string) in the list displayed at the time.

**Note** The search function is a pure text search, that is, the entries must be "exact - including every dot, dash and space".

Additional settings are:

- search for address, symbol, block or language,
- the character string entered as search term is a whole word or part of a word,
- capital/small letters are to be taken into account or ignored,
- the search range and the direction of the search can be specified.

## Assignment of I, Q, M, T, C



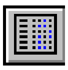

SIMATIC® S7



Date: 12.03.03
File: PRO1_11E.36

SITRAIN Training for Automation and Drives

**Assignment I/Q/M**



You open the Assignment of I/Q/M/T/C by selecting the *View -> Assignment* menu options or by clicking on the relevant icon.
This assignment list gives you an overview of which bit is used in which byte of the memory areas input (I), output (Q) and bit memory (M) and which S5 timer and S5 counter. The type of use (reading or writing) is not displayed.

The inputs (I), outputs (Q) and bit memories (M) are displayed byte-by-byte in lines.

The bits identified with an "x" or binary addresses (in the slide, for example, I 1.0, Q4.3, or M10.3) are used explicitly in the program.

The colored bytes identify byte, word or doubleword addresses (in the slide, for example, input byte IB0, the input word IW2 or the output doubleword QD6) that are used in the user program. The address dimension (byte, word or doubleword) comes from the vertical line in one of the columns "B" (Byte), "W" (Word) and "D" (Doubleword).

Bits that are both colored and have an "x" are used explicitly as a binary address in the user program and are used through a byte, word, or doubleword address.

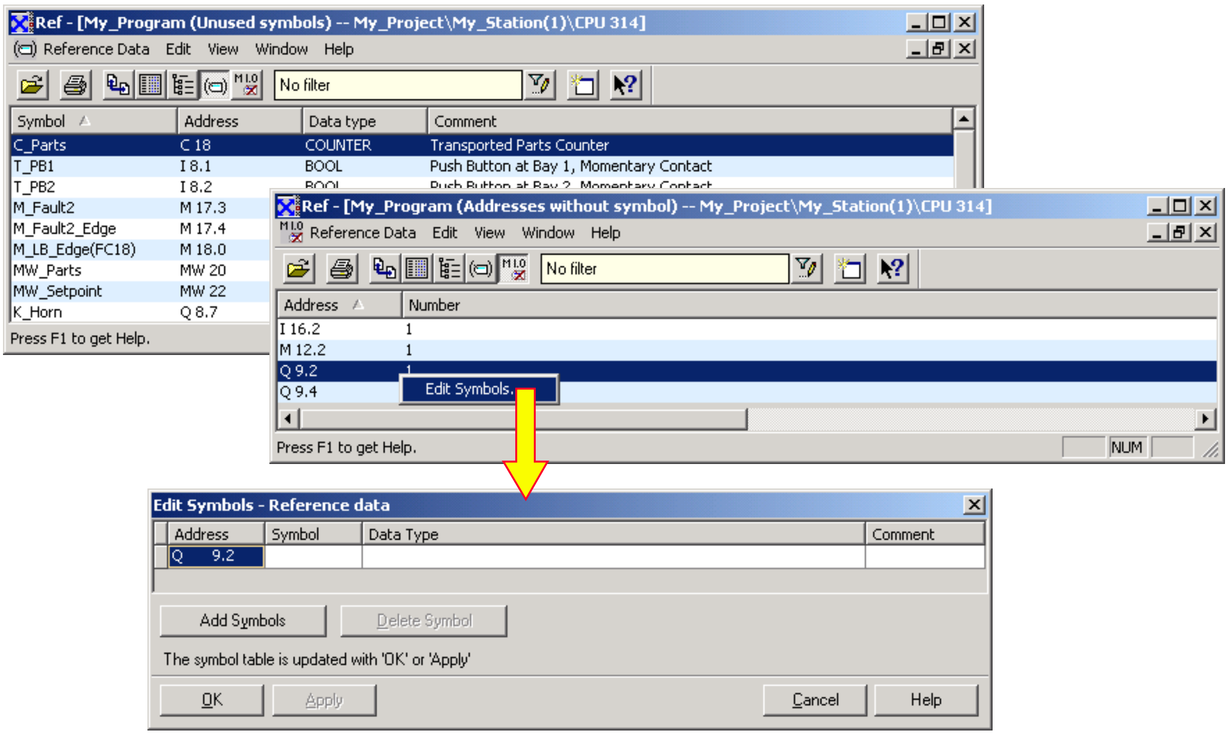
Example (see slide):

The output Q8.4 is used explicitly as a binary address ("x") as well as indirectly through the output doubleword QD6 in the user program (the output bytes QB6,7,8,9 are colored, vertical line in the "D" column for doubleword).

**Filter**

By selecting "Filter", you can choose the memory areas to be listed and restrict the individual address areas.
The same rules as for filtering in the Cross References list apply.

## Unused Symbols / Addresses without Symbols

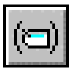



SIMATIC® S7


Date: 12.03.03
File: PRO1_11E.37

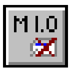
SITRAIN Training for Automation and Drives

**Unused Symbols**



When you select the *View -> Unused Symbols* menu option or click the relevant icon, a list of addresses appears. These addresses are defined in the symbol table but are not used in the S7 user program.
By clicking the right mouse button and then -> *Delete Symbols,* you can then remove those addresses or symbols from the symbol table.

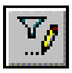
**Addresses without Symbols**




When you select the *View -> Addresses without Symbols* menu option or click the relevant icon, a list of addresses appears. These addresses have been used in the S7 user program but are not defined in the symbol table.

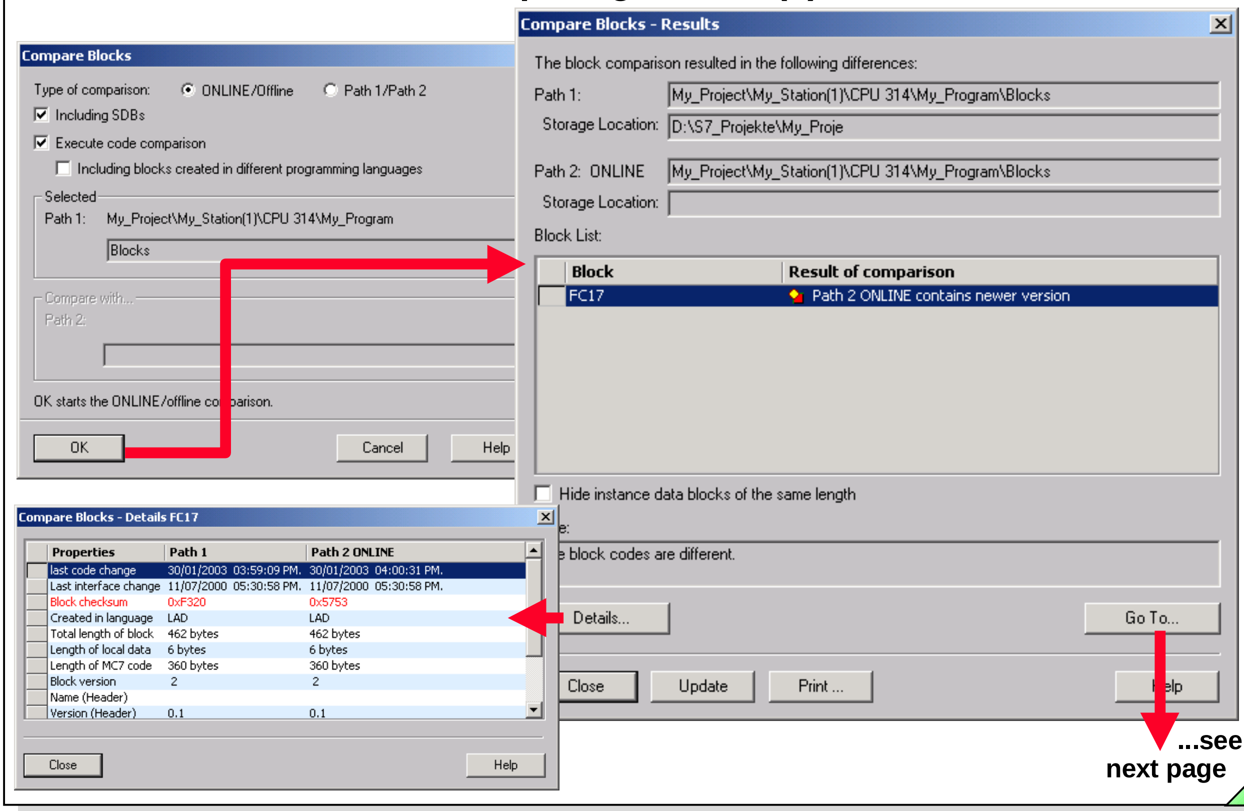
By clicking the right mouse button and then -> *Edit Symbols,* you can declare symbols for the affected addresses after the fact.

**Filter**



You use "Filter" to make selections of detailed information for the display of unused symbols.

# Comparing Blocks (1)




...see next page

SIMATIC® S7


Date: 12.03.03
File: PRO1_11E.38

SITRAIN Training for Automation and Drives

**Introduction**

You can compare blocks between online and offline data management or between two user programs on the hard disk of the PG.

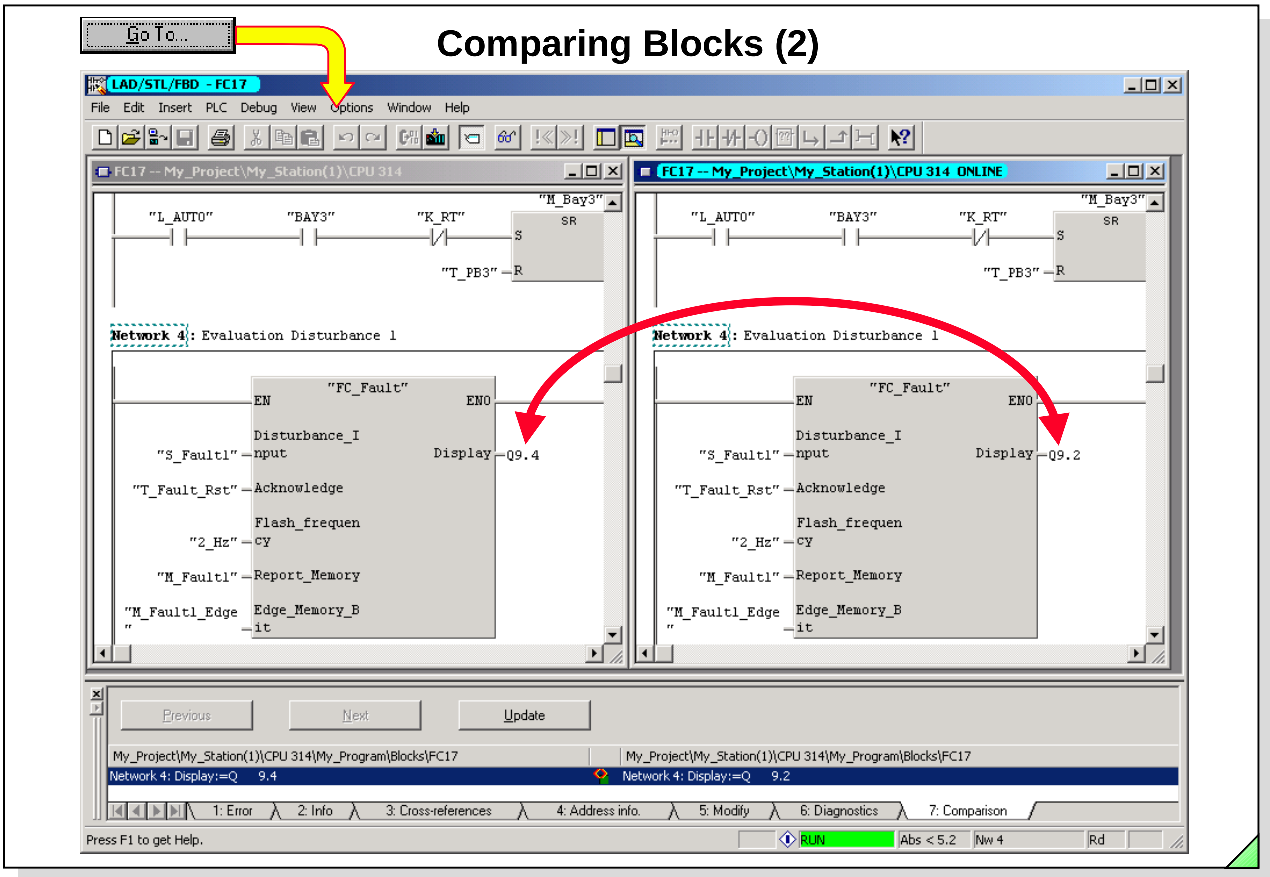
With this function, you can determine whether, for example, program corrections were made in the CPU later on and in which network the blocks differ.

**What To Do**

1. With the right mouse button, select the Blocks folder of an S7 program.
2. Select the *Compare Blocks* menu option.
3. Choose whether you want to compare online/offline or between two offline programs and acknowledge with the "OK" button.
4. In the follow-up screen, the blocks that differ are listed.
5. Select the line in which a difference was determined and then select the "Details" button.
6. In the "Compare Blocks - Details" window you can see when the block was modified and if the block length was changed.
7. After you select the "Go To..." button, the differing block is opened online and offline in two windows, for example, and the network, in which the first difference was determined is displayed.

**Note**

Program corrections can only be made in the offline window.




SIMATIC® S7


Date: 12.03.03
File: PRO1_11E.39

SITRAIN Training for Automation and Drives

**Differences**

After you select the "Go To..." button (see previous page), the Program Editor is opened with two windows side-by-side, in which the network with the first difference is displayed.

If LAD/STL/FBD Editor defaults were set to open blocks in Statement List (STL) the PG's cursor would be at the first command lines that are different.

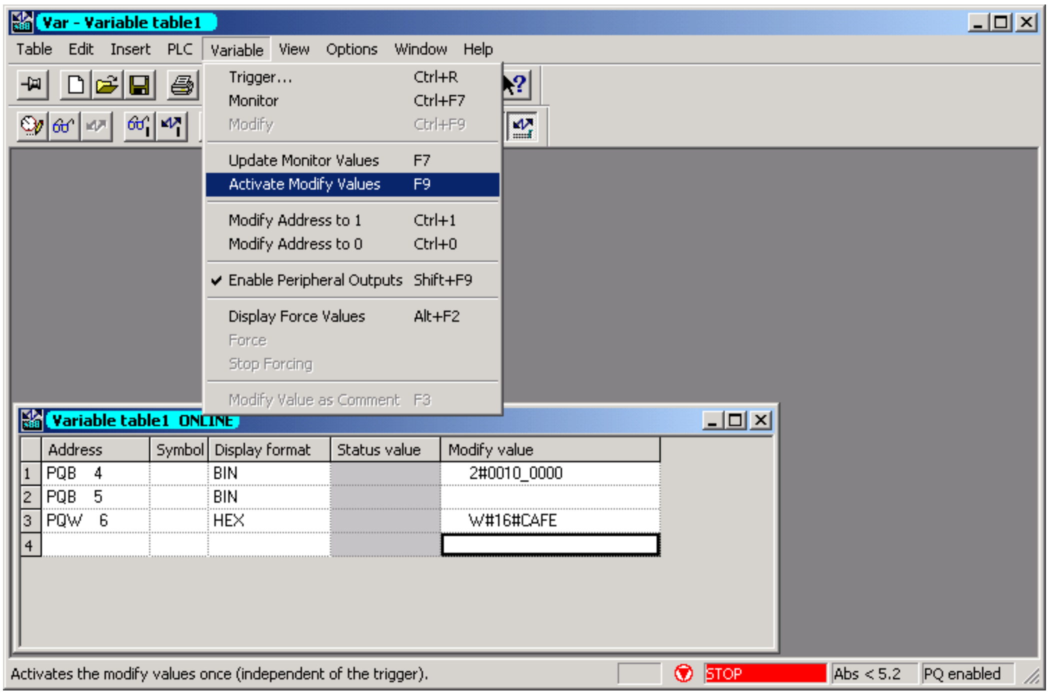
If the blocks displayed differ in several locations, you can switch between the different program locations using the "Previous" and "Next" buttons.

**Example**

In the example shown above, the FC 17 block which is stored offline (in the slide on the left), calls the FC 20 block. For the formal parameter "Display", output Q9.4 is given as the actual parameter, whereas in the program that is stored online it is the output Q9.2. That means, that after the block was downloaded into the CPU, a correction was only made to either the block saved offline or to the block saved online.

You can identify which of the two blocks was modified last by reading out the the time stamp in the "Compare Blocks - Results" screen.

# Modifying Outputs in the Stop State




SIMATIC® S7


Date: 12.03.03
File: PRO1_11E.40

**SITRAIN** Training for Automation and Drives

**Function and Area of Use**

When a CPU transitions to STOP, all digital outputs are switched off and analog outputs are switched to their defined parameter - either off, as is, or a predefined value. The "Enable Peripheral Outputs" function allows modification of outputs when the CPU is in the STOP state.

The "Enable Peripheral Outputs" function is used mainly to check peripheral output wiring. It can, however, also be used to continue to control actuators in the process, even though the CPU has gone into the STOP mode because an error has occurred.

**What To Do**

To enable the peripheral outputs, proceed as follows:

1. Open or edit a variable table (VAT) that contains the peripheral outputs that you want to test or modify
*(specify the peripheral outputs byte-by-byte, word-by-word or doubleword-by-doubleword; you cannot modify individual output bits!)*
2. Select the *PLC -> Connect to…* menu option to establish a connection to the CPU you want
3. Switch the CPU to the STOP state
4. Enter the appropriate values for the peripheral outputs you want to modify in the "Modify Value" column.

| | | | | |
|---|---|---|---|---|
| Examples: | PQB | 7 | Modify Value: | 2#01000011 |
| | PQW | 2 | | W#16#0027 |
| | PQD | 4 | | DW#16#0001 |

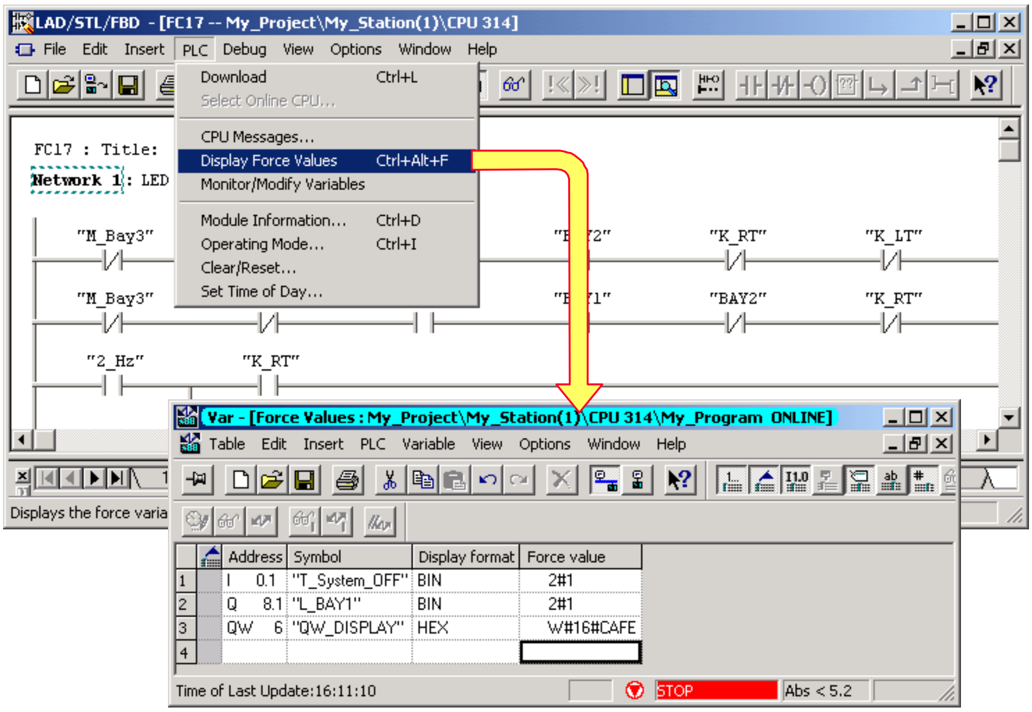
5. Use *Variable -> Enable Peripheral Outputs* to activate the modifying of the outputs
6. Use *Variable -> Activate Modify Values* to modify the peripheral outputs.
7. To assign new modify values, enter these and then activate them with *Variable > Activate Modify Values*

Modifying or "Enable Peripheral Outputs" remains active until you deactivate it using *Variable -> Enable Peripheral Outputs,* or you press the *ESC key*.

**Note**

When you change the operating mode of the CPU from STOP to RUN or STARTUP, the Enable Peripheral Outputs is deactivated and a message appears.

## Overwriting Variables using "Force"




SIMATIC® S7



Date: 12.03.03
File: PRO1_11E.41

**SITRAIN** Training for Automation and Drives

### Function and Area of Use

With Force, you can overwrite variables with any values you like, independent of the user program. You can open only one Force Values window for a CPU.

With the S7-300™, you can only force the process image inputs and outputs; with the S7-400™ you can also force bit memories and peripherals.

### Notes on Forcing

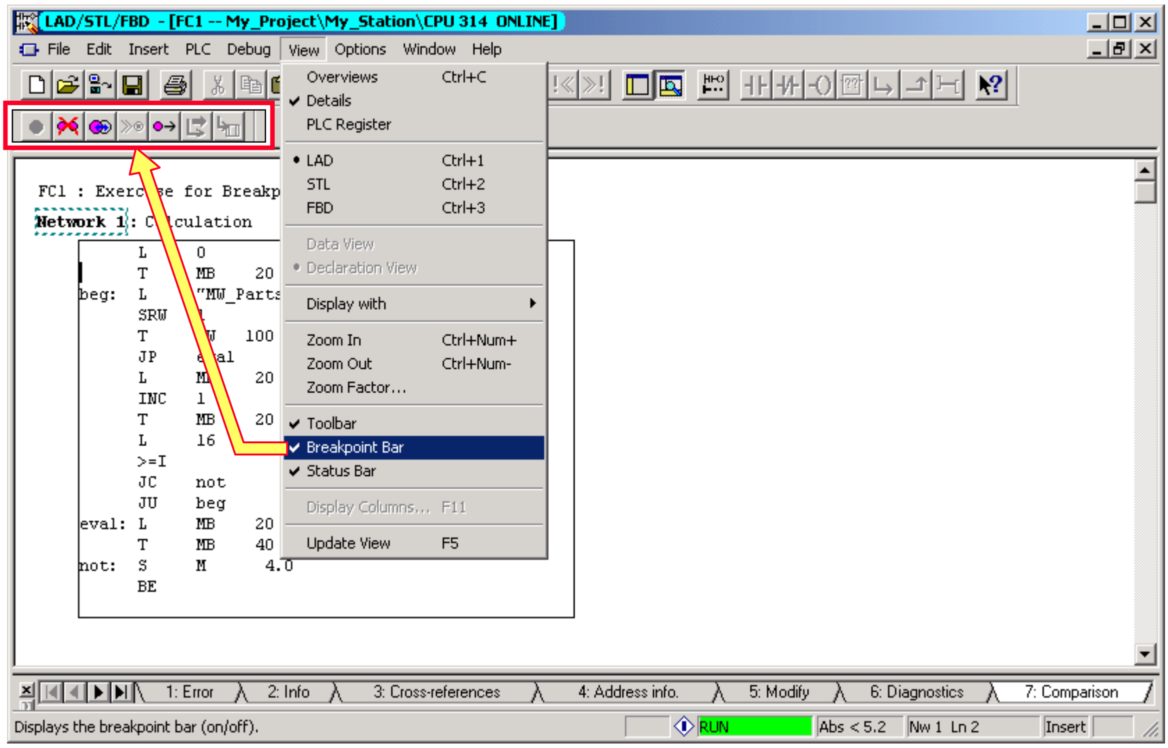
- Before your start the "Force" function, you should make sure that no one else is carrying out this function at the same time on the same CPU.
- You can only cancel a force job by selecting the *Variable -> Stop Forcing* menu option
- You cannot undo "Forcing" with the *Edit -> Undo* menu option.
- You cannot cancel the force job by closing the *Force Values* window or by exiting the "Monitor/Modify Variables" application.

### Selecting the "Force" Function

1. To start the Force Function in the SIMATIC® Manager, select the CPU to be forced.
2. Select the *PLC -> Display Force Values* menu option.

   The *Monitor/Modify Variables* tool opens the Force Values window with the addresses currently being forced and their associated force values. The status bar also shows the date and time of the current force job in the CPU. If no address in the CPU is forced, this window is empty.
3. In the "Address" column enter the variables and in the "Force Values" column enter the values you want.
4. Start forcing with the *Variable -> Force* menu option.
5. End the force job with the *Variable -> Stop Forcing* menu option.

# Testing the Program Execution using Breakpoints (Part 1)




SIMATIC® S7


Date: 12.03.03
File: PRO1_11E.42

**SITRAIN** Training for Automation and Drives

**Breakpoints**

With the help of this test function, you can test a program you see in STL in single-step mode and thus follow the sequence of the executed instructions as well as the associated register contents.

You can set several breakpoints, depending on the CPU. The number of possible breakpoints depends on the CPU used.

**Note**

In order to carry out these test functions, you must have fulfilled the following requirements:

- The "Test Operation" mode must have been assigned parameters.
- The block to be tested must be opened online.
- The LAD/FBD/STL Editor must be explicitly set to *View -> STL*.
- The block must not be protected (Know_how_protect)

**Breakpoint Functions**

You can choose the breakpoint functions in the Program Editor by selecting the "Test" menu option or through the Breakpoint Bar.

You can activate the breakpoint bar by selecting the *View -> Breakpoint Bar* menu option in the Program Editor.

**Attention**

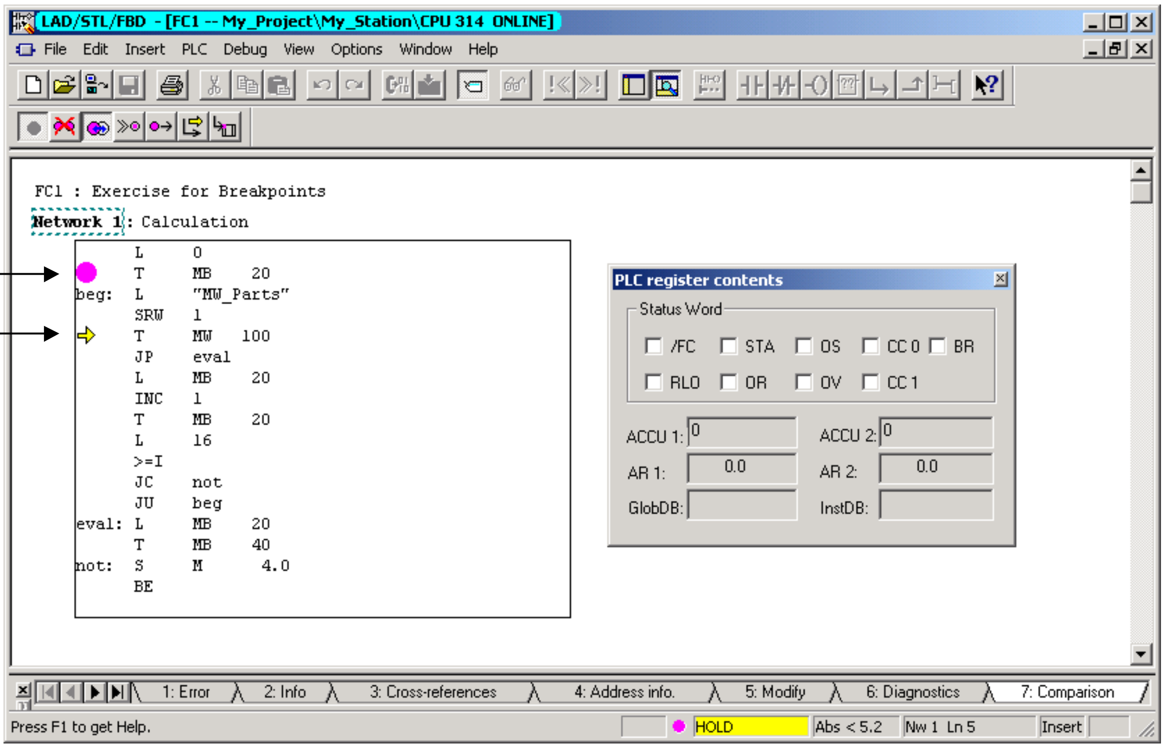
If the program execution encounters a breakpoint, the CPU switches from RUN to HOLD mode.

In this mode, the STOP LED lights up and at the same time the RUN LED flashes. The outputs are deactivated for safety reasons.

# Testing the Program Execution using Breakpoints (Part 2)

Break-
point

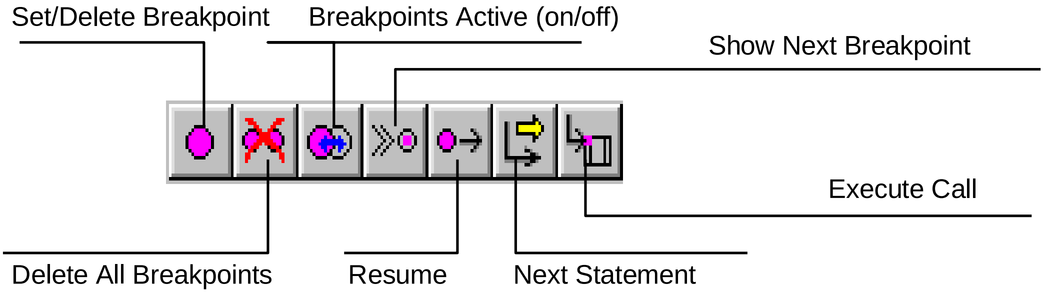
Next
state-
ment







**Breakpoint Bar**




**Set/Delete Breakpoint** With "Set/Delete Breakpoint" you determine where the program execution is to be halted. The breakpoint's statement is not executed.

**Breakpoints Active** With "Breakpoints Active" you activate all breakpoints; not only those already set but also those still to be set.

**Show Next Breakpoint** With "Show Next Breakpoint", the Editor jumps to the next selected breakpoint, without executing the program.

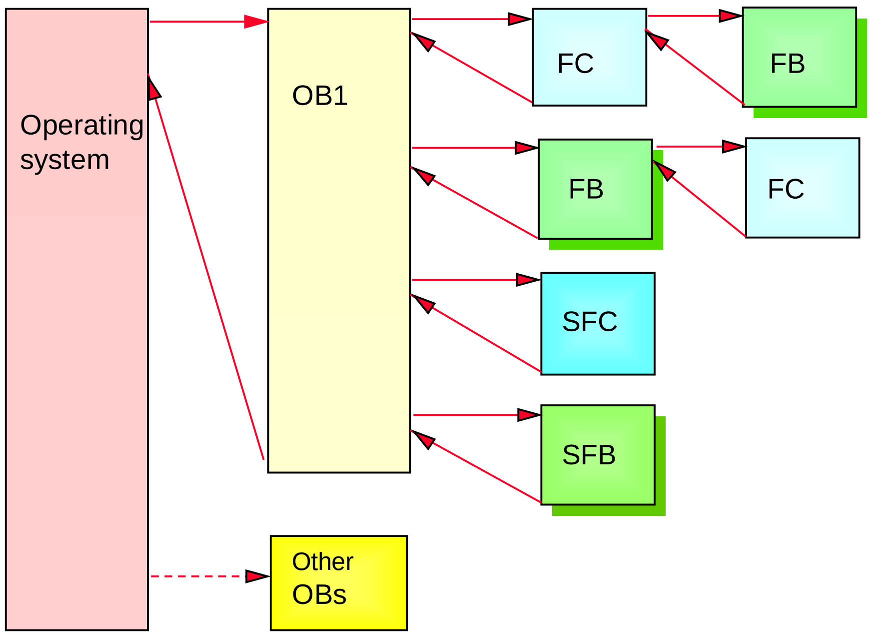
**Resume** With "Resume", the program runs until the next active breakpoint.

**Next Statement** With "Next Statement", you execute the program in single-step. If you reach a block call, you jump to the first statement after the block call with "Next Statement".

The *Execute Next Statement* and *Execute Call* menu options require a free breakpoint for the internal implementation.

**Execute Call** Here, when you reach a block call you branch into the block with "Execute Call". At the end of the block you jump back to the next statement after the block call.

## Organization Blocks




SIMATIC® S7


Date: 12.03.03
File: PRO1_12E.1

SITRAIN Training for Automation and Drives

**Contents** Page

# Objectives

**Upon completion of this chapter the participant will ...**

... know the organization blocks that are available

... understand the difference between "Complete restart", "(Warm) Restart" and "Cold restart"

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

... know the "Time-of-Day Interrupt", "Cyclic Interrupt", "Hardware Interrupt", "Time-Delay Interrupt", "Diagnostic Interrupt"

... know the error OBs for synchronous and asynchronous errors and will be able to implement them to influence the CPU response when errors occur

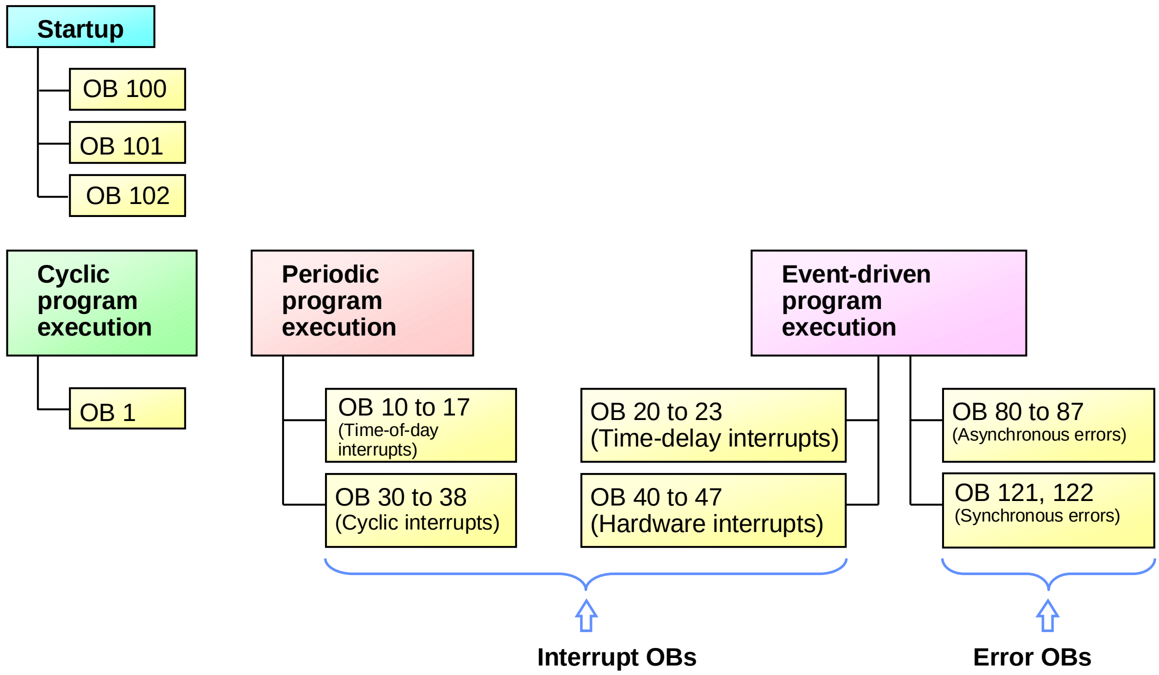
... be able to interpret the OB start information and will be able to evaluate it in the program

SIMATIC® S7


Date: 12.03.03
File: PRO1_12E.2

SITRAIN Training for Automation and Drives

# Overview of the Organization Blocks




SIMATIC® S7


Date: 12.03.03
File: PRO1_12E.3

SITRAIN Training for Automation and Drives

**Startup**

A startup program is carried out before the cyclic program execution after a power recovery, or a change of operating mode (through the CPU's mode selector or by the PG). OBs 100 to OB 102 are available for this.
In these blocks you can, for example, preset the communications connections.

**Cyclic Program Execution**

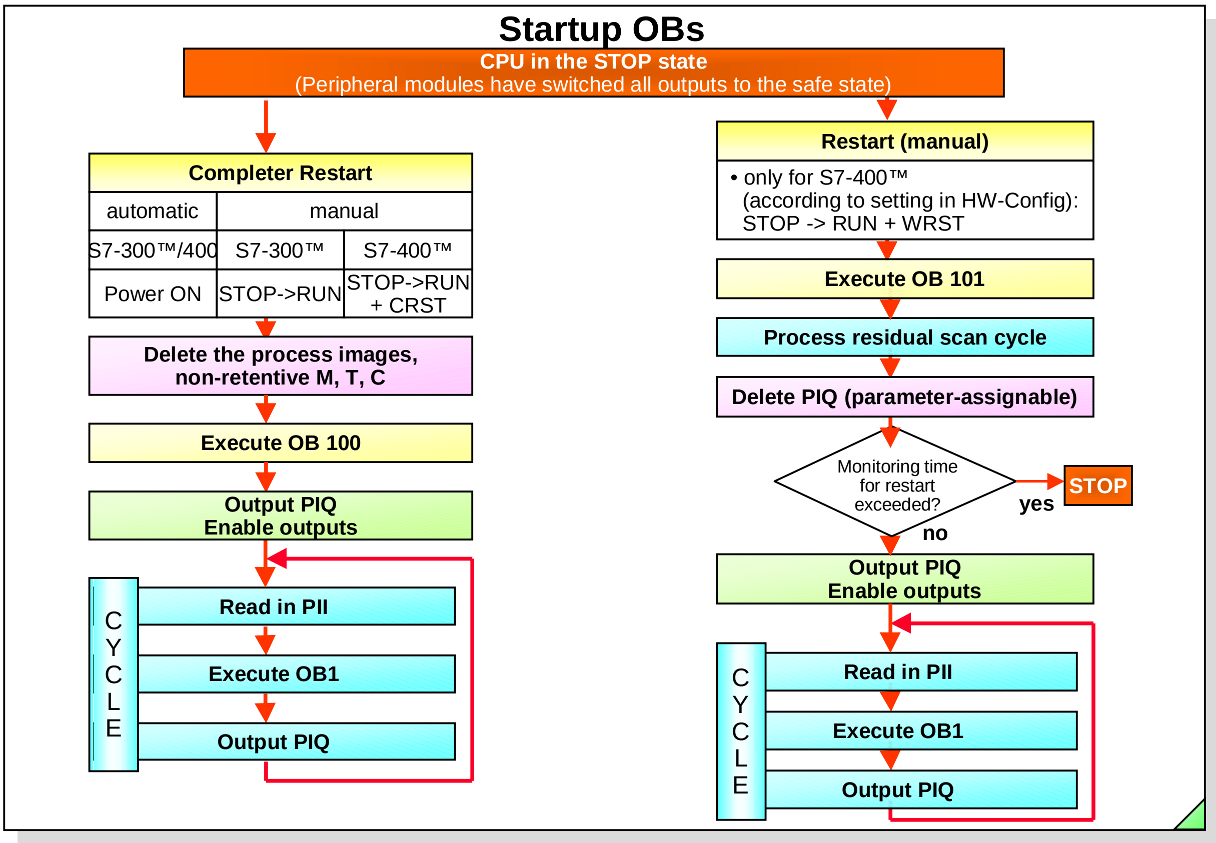
The program that is to be continuously executed is stored in the Organization Block OB 1. After the user program has been completely executed in OB 1, a new cycle begins with the updating of the process images and the processing of the first statement in OB 1. The scan cycle time and the response time of the system is a result of these operations.
The response time is the total of the executing time of the CPU's operating system and the time it takes to execute all of the user program.
The response time, that is, how quickly an output can be switched dependent on an input signal, is equal to scan cycle time x 2.

**Periodic Program Execution**

With periodic program execution, you can interrupt the cyclic program execution at fixed intervals. With cyclic interrupts, an OB 30 to OB 37 organization block is executed after a preset timing code has run out, every 100 ms for example. Control-loop blocks with their sampling interval are called, for example, in these blocks.
With time-of-day interrupts, an OB is executed at a specific time, for example every day at 17:00 hours (5:00 p.m.), to save the data.

**Event-driven Program Execution**

The hardware interrupt can be used to respond quickly to a process event. After the event occurs, the cycle is immediately interrupted and an interrupt program is carried out.
The time-delay interrupt reponds after a delayed time period to a process event.
With the error OBs you can determine how the system is to behave, for example, if the backup battery fails.




SIMATIC® S7


Date: 12.03.03
File: PRO1_12E.4

SITRAIN Training for Automation and Drives

**Complete Restart**

The complete restart type of startup deletes the process images (PII, PIQ) and all non-retentive bit memories, timers and counters. Retentive bit memories, timers and counters as well as the data blocks‘ current values are retained (only with battery backup. With the S7-300™ even without battery backup if an EPROM is used and the CPU‘s retentive behavior has been parameterized).

The program stored in OB 100 is executed once and then cyclic program execution begins.

**(Warm) Restart**

The restart type of startup retains the states of all addresses (bit memories, timers, counters, process images, the data blocks‘ current values). The program stored in OB 101 is executed once.

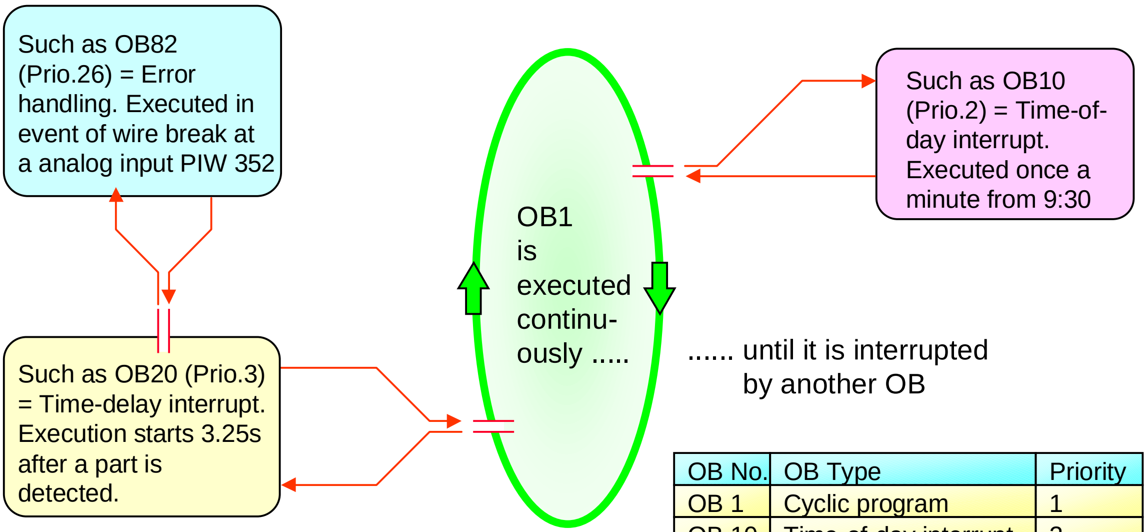
Then, program execution resumes from the point where the interruption occurred (power off, CPU STOP). After this "residual cycle" has been executed, the cyclic program execution begins.

**Cold Restart**

The CPUs 318-2 and 417-4 also have the additional cold restart type of startup. This type of startup can be set up for power recovery in the HW Config tool when the CPU is parameterized.

The only difference between a cold restart and a complete restart is that in addition to the process images, all bit memories, timers and counters (even the retentive ones!) are deleted. As well, the data blocks‘ current values are overwritten with the current values stored in the load memory or with those that were originally downloaded with the data blocks to the CPU.

## Interrupting the Cyclic Program




| 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 | 26 / 28 |

SIMATIC® S7


Date: 12.03.03
File: PRO1_12E.5

SITRAIN Training for Automation and Drives

**OB Calls**

Organization blocks (OBs) are the interface between the CPU's operating system and the user program.

The operating system calls the organization blocks exclusively. There are various start events (time-of-day interrupts, hardware interrupts - see slide) that each lead to the start of their associated organization block.

**Interrupting the Cyclic Program**

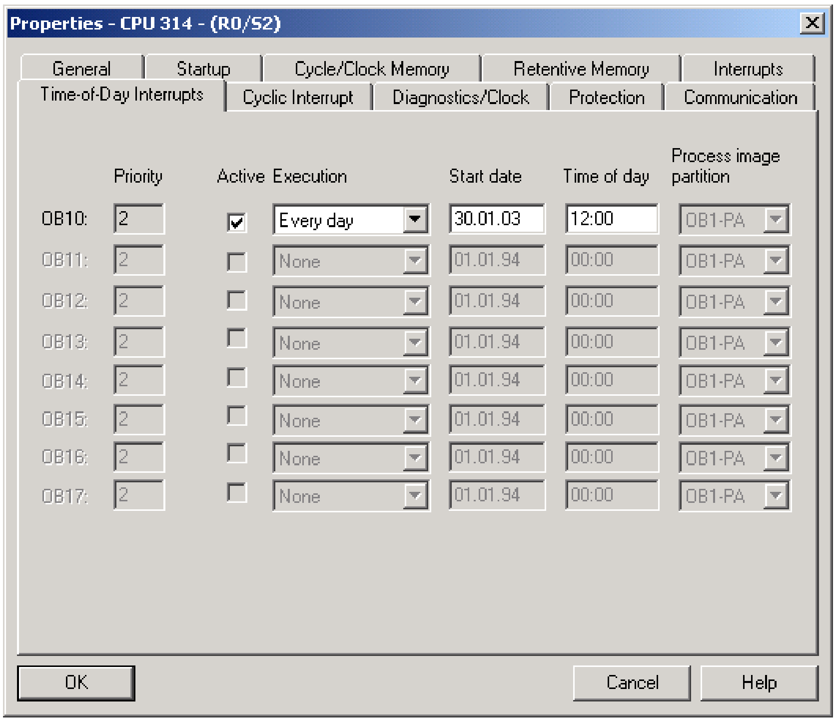
When the operating system calls another OB, it interrupts the cyclic program execution because OB1 has the lowest priority. Any other OB can therefore interrupt the main program and execute its own program. Afterwards, OB1 resumes execution at the point of interruption.
If an OB with a higher priority than the one currently being executed is called, the lower priority OB is interrupted after the current statement has been completed. The operating system then saves the entire register stack for the interrupted block. This register information is restored when the operating system resumes execution of the interrupted block.

**Priorities**

Every OB program execution can be interrupted by a higher priority event (OB) at command boundaries. Priorities are graduated from 0 to 27, whereby 0 has the lowest priority and 28 has the highest priority.

OBs of the same priority do not interrupt each other, but are started one after the other in the sequence they are recognized.

## Time-of-Day Interrupt (OB10)




SIMATIC® S7



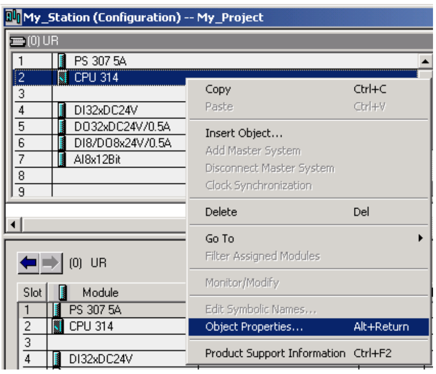
Date: 12.03.03
File: PRO1_12E.6

SITRAIN Training for Automation and Drives

**Time-of-Day Interrupts**

Time-of-day interrupts are used for executing a certain program called in OB 10 (as an example) either once only at a certain time or periodically (once a minute, hourly, daily, weekly, monthly, yearly) starting at that time.

Time-of-day interrupts are configured with the "HW Config" tool. To select when and how OB10 is to be activated choose the menu options *CPU -> Object Properties* ->-> “Time-of-Day Interrupts" tab.




**"Active"**

If you check the "Active" checkbox, the time-of-day interrupt OB is executed on every complete restart of the CPU.

**Note**

System functions, at runtime, can also control time-of-day interrupts. The following system functions are available:

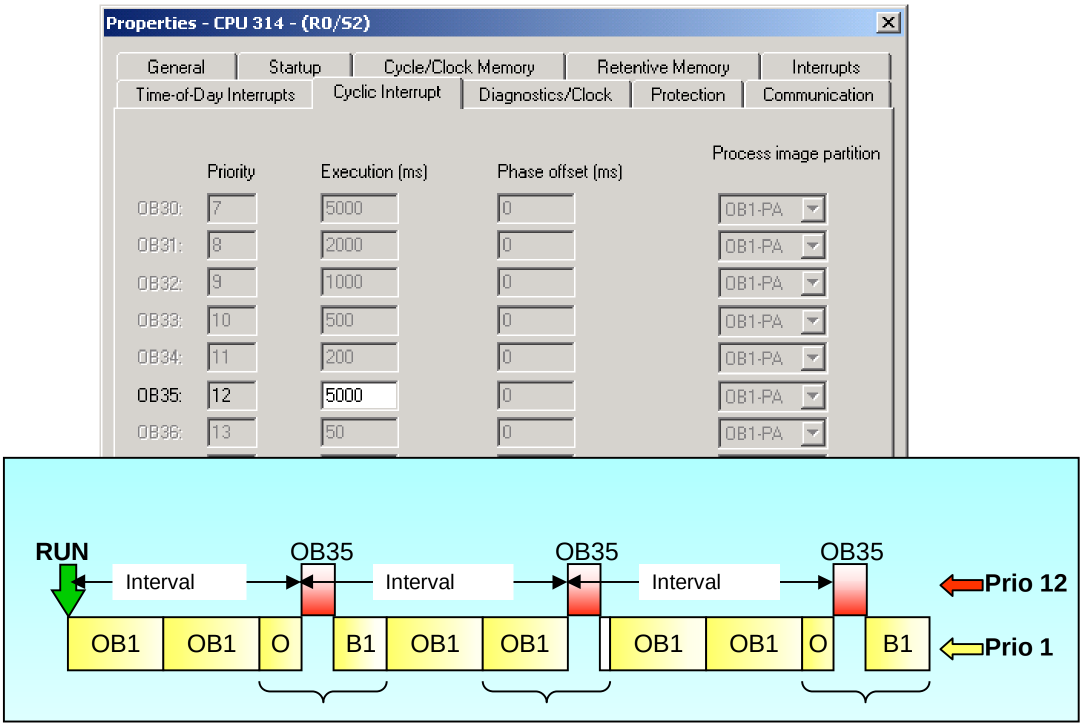
- SFC 28 "SET_TINT" Set starting date, time and period
- SFC 29 "CAN_TINT" Cancel time-of-day interrupt
- SFC 30 "ACT_TINT" Activate time-of-day interrupt
- SFC 31 "QRY_TINT" Query time-of-day interrupt.

**S7-400™**

There are up to eight different time-of-day interrupt OBs (OB 10 to 17) for the S7-400™ PLC.

SIEMENS

## Cyclic Interrupt (OB35)




SIMATIC® S7



Date: 12.03.03
File: PRO1_12E.7

SITRAIN Training for Automation and Drives

**Cyclic Interrupt** Cyclic (watchdog) interrupts are used for executing blocks at fixed intervals. The cyclic interrupt OB for the S7-300™ is OB 35.
The default call interval for OB 35 is 100ms. You can change this to a value within the permitted range of 1ms to 1 minute.

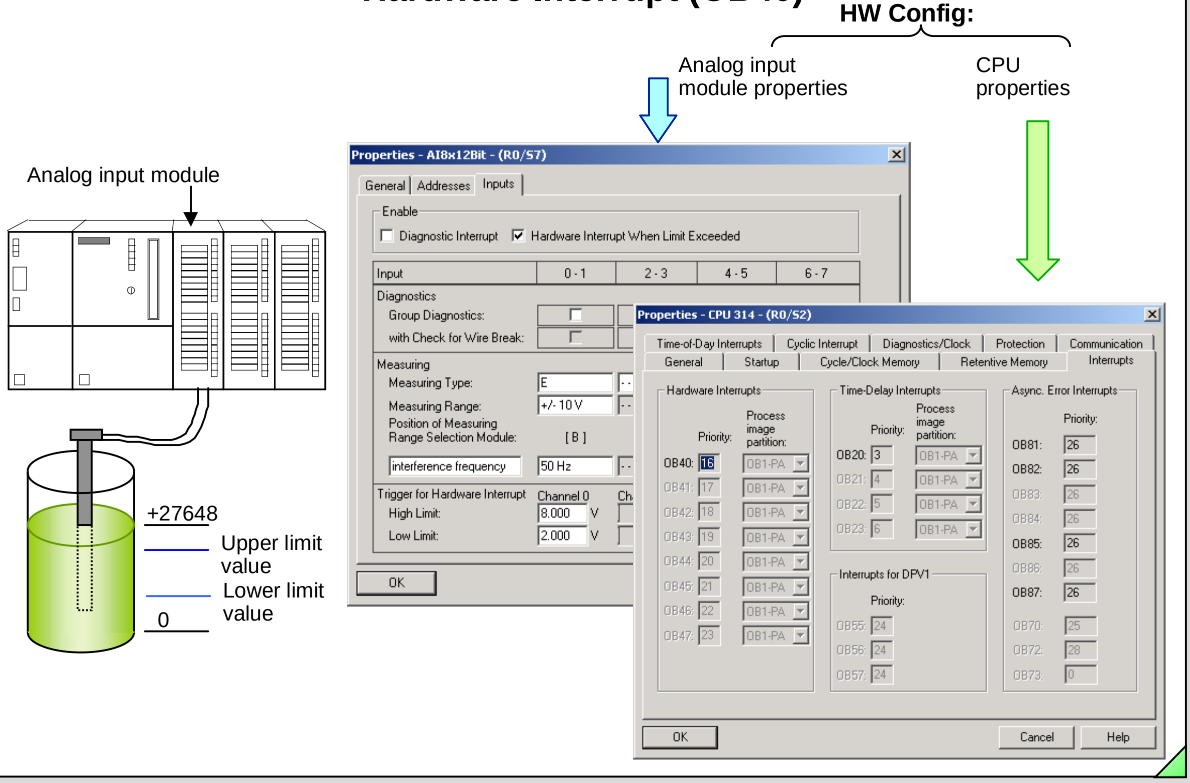
**Starting Time** When you activate a time-controlled interrupt, you specify the interval in relation to the "starting time". The starting time begins every time the CPU mode changes from STOP to RUN.

**Interval** You must make sure that the interval you specify is longer than the time required for execution. The operating system calls OB35 at the specified time. If OB35 is still active at this time, the operating system calls OB80 (cyclic interrupt error OB).

**Note** System functions at run time cannot control cyclic interrupts.

**S7-400™** There are up to nine different cyclic interrupt OBs (OB30 to 38) for the S7-400™ PLC.

## Hardware Interrupt (OB40)




SIMATIC® S7

Date: 12.03.03
File: PRO1_12E.8
**SITRAIN** Training for Automation and Drives

**Hardware Interrupt**

The program in a hardware interrupt OB (OB40) is executed as soon as a certain event occurs.

Various module-specific signals can trigger hardware interrupts:

- For parameter-assignable signal modules (DI, DO, AI, AO) you use the "HW Config" tool to specify the signal that is to trigger a hardware interrupt.
- In the case of CPs and FMs, you specify the interrupt characteristics using the configuration software for the module concerned.

**Example**

In the example above, suitable limit values have been configured for an analog input module. If the measured value exceeds the limit, OB40 is called.

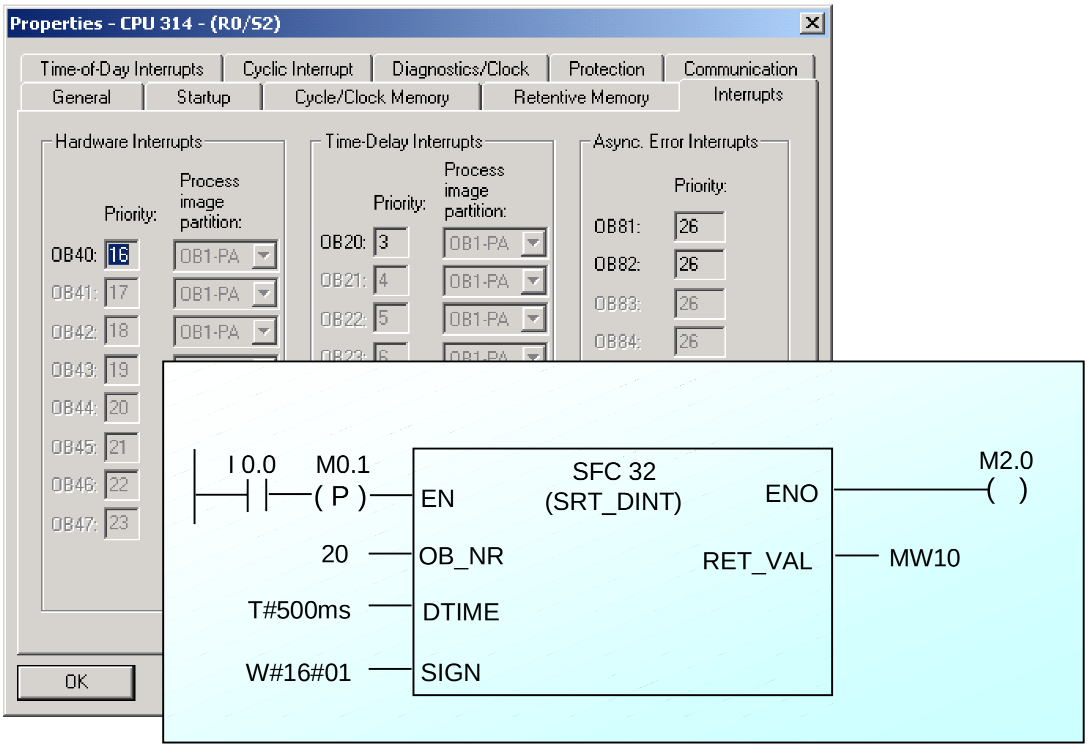
This has the same effect as including a comparison operation in OB1 which causes an FB or FC to be called when the upper limit is reached. However, if you use OB40 you don't need to write a program in another block.

You can use the program in OB40 for interrupt generation or for process control.

**S7-400™**

There are up to eight different hardware interrupt OBs (OB40 to 47) for the S7-400™ PLC.

## Time-Delay Interrupt (OB20)




SIMATIC® S7


Date: 12.03.03
File: PRO1_12E.9

**SITRAIN** Training for Automation and Drives

**Time-Delay Interrupt**

The program in a time-delay OB (OB20) is executed with a specified delay after a certain event has occurred.

OB20 <u>can only</u> be activated by calling system function SFC32 (SRT_DINT). SFC32 is also used for setting the delay time.

**SFC 32**

- OB_NR = OB number to be executed with a time delay.
- DTIME = Delay time (1 to 60000ms)
- SIGN = User-specified signal for starting the time-delay interrupt OB
- RET_VAL = Error code, if an error occurs during execution of the time-delay interrupt OB (See online help for meanings of error numbers).

**Note**

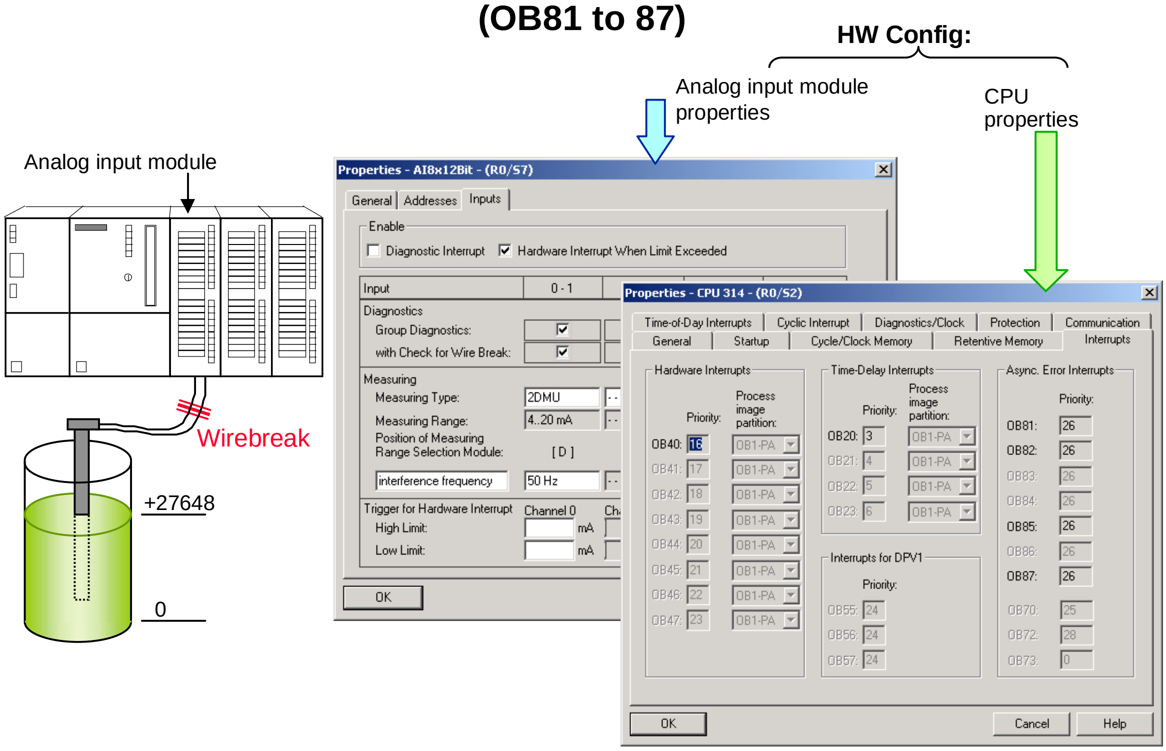
In addition to SFC32, the following SFCs are also available for dealing with time-delay interrupts:

- SFC33 (CAN_DINT) = Cancel time-delay interrupt
- SFC34 (QRY_DINT) = Query time-delay interrupt.

**S7-400™**

There are up to four different time-delay interrupt OBs (OB20 to 23) for the S7-400™ PLC.

## The Diagnostic Interrupt and Asynchronous Error Interrupt (OB81 to 87)




SIMATIC® S7


Date: 12.03.03
File: PRO1_12E.10

SITRAIN Training for Automation and Drives

**Asynchronous Errors**

Asynchronous errors are faults in the PLC functionality. They occur asynchronously to the execution of the program and cannot be traced to a particular point in the program (such as a diagnostic interrupt from a module).

**Response**

If a fault is detected in RUN mode and the relevant error OB has been programmed, the OB is called and the program in it is executed. This program could, for example, contain:

- instructions for switching on a siren
- instructions for data backup, followed by a STOP instruction
- a program for recording the frequency with which the fault occurs, without causing the CPU to go into STOP mode.

**Note**

If the error OB for a particular fault is not present, the CPU automatically goes into STOP mode.

**Example**

Asynchronous error interrupt OB82 is called in the following situations, for example:

- wirebreak on a module with diagnostic capability
- failure of the power supply to an analog input module
- measuring range of an analog input module exceeded, etc.

## Asynchronous Error OBs

| Type of error | Example | OB | Priority |
|---|---|---|---|
| Time error | Maximum scan cycle time exceeded | OB80 | 26 |
| Power supply fault | Backup battery failure | OB81 | 26 / 28 |
| 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 | |

SIMATIC® S7


Date: 12.03.03
File: PRO1_12E.11

SITRAIN Training for Automation and Drives

**Priority** The error OBs called in response to asynchronous errors are executed immediately because they have the highest priority of all interrupt and error OBs:

- Priority 26 if the error occurs while an OB with lower priority (<26) is being executed
- Priority 28 if the error occurs while a startup OB (priority 27) is being executed.

## 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 | |

SIMATIC® S7



Date: 12.03.03
File: PRO1_12E.12

**SITRAIN** Training for Automation and Drives

**Synchronous Errors**

These errors can be traced to a particular point in the program, if the error occurred during execution of a particular statement. The error OBs called in response to synchronous errors are executed as part of the program, with the same priority as the block that was being executed when the error was detected.

# System Functions for Controlling Interrupt OBs

| Organization block | | Priority in S7-300™ | SFCs for controlling OBs | Remarks |
|---|---|---|---|---|
| Function | Number | | | |
| Time-of-day interrupt | OB 10 to 17 | 2 | SFC 28 to 31 | HW Config alternative |
| Cyclic interrupt | OB 30 to 38 | 12 | none | |
| Time-delay interrupt | OB 20 to 23 | 3 | SFC32 to 34 | Mandatory |
| Hardware interrupt | OB 40 to 47 | 16 | none | |
| Diagnostic interrupt | OB 81 to 87 | 26 | none | |

SIMATIC® S7


Date: 12.03.03
File: PRO1_12E.13

**SITRAIN** Training for Automation and Drives

**OBs** You will find a complete list and a description of the error OBs in the online help: LAD/STL/FBD Editor -> *Help -> Contents -> Help on Blocks -> Help on Organization Blocks.*

**SFCs** The system functions and their uses, how to call them and assign parameters to them are discussed in an advanced programming course.

SIEMENS

## OB Start Information

| L-Byte | | | |
|---|---|---|---|
| 0 / 1 | Start event | Consecutive number | Management information |
| 2 / 3 | Priority | OB No. | |
| 4 / 5 | Data formats of L-Bytes 8, 9, 10, 11 | | Start information |
| 6 / 7 | Additional info 1 (such as start address of interrupt module) | | |
| 8 / 9 | Additional info 2 (such as interrupt status) | | |
| 10 / 11 | Additional info 3 (such as channel number) | | |
| 12 / 13 | Year | Month | Start time |
| 14 / 15 | Day | Hours | |
| 16 / 17 | Minutes | Seconds | |
| 18 / 19 | 1/10 Second, 1/100 Second | 1 /1000 Second, Weekday | |

SIMATIC® S7


Date: 12.03.03
File: PRO1_12E.14

SITRAIN Training for Automation and Drives

**Start Information** You have a uniform system start information in the local data stack when the the operating system calls the OB. This start information has a length of 20 bytes and is available after the OB starts execution.

**Access to Start Information** The STEP 7 software makes a standard declaration table available for the symbolic access to start information (example for OB 81).

Interface
TEMP
- OB81_EV_CLASS
- OB81_FLT_ID
- OB81_PRIORITY
- OB81_OB_NUMBR
- OB81_RESERVED_1
- OB81_RESERVED_2
- OB81_MDL_ADDR
- OB81_RESERVED_3
- OB81_RESERVED_4
- OB81_RESERVED_5
- OB81_RESERVED_6
- OB81_DATE_TIME

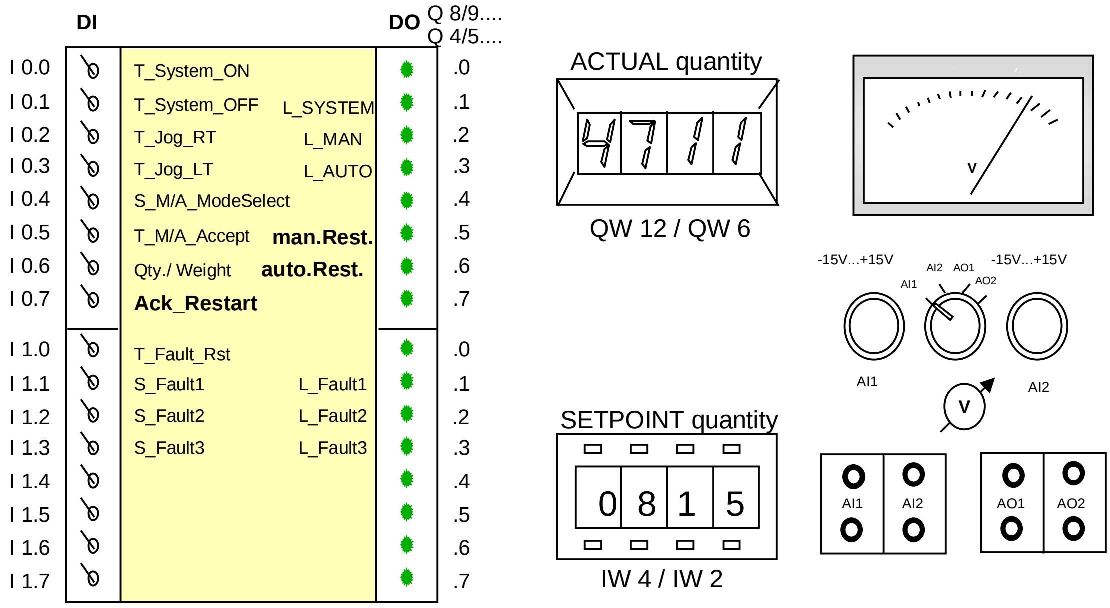
Contents Of: 'Environment\Interface\TEMP'

| Name | Data Type | Address | Comment |
|---|---|---|---|
| OB81_EV_CLASS | Byte | 0.0 | 16#39, Event class 3, Entering event state, Internal fault event |
| OB81_FLT_ID | Byte | 1.0 | 16#XX, Fault identifcation code |
| OB81_PRIORITY | Byte | 2.0 | Priority of OB Execution |
| OB81_OB_NUMBR | Byte | 3.0 | 81 (Organization block 81, OB81) |
| OB81_RESERVED_1 | Byte | 4.0 | Reserved for system |
| OB81_RESERVED_2 | Byte | 5.0 | Reserved for system |
| OB81_MDL_ADDR | Int | 6.0 | Address of bus interface module in rack with defective power supply |
| OB81_RESERVED_3 | Byte | 8.0 | Reserved for system |
| OB81_RESERVED_4 | Byte | 9.0 | Reserved for system |
| OB81_RESERVED_5 | Byte | 10.0 | Reserved for system |
| OB81_RESERVED_6 | Byte | 11.0 | Reserved for system |
| OB81_DATE_TIME | Date_And_Time | 12.0 | Date and time OB81 started |

**Note** You can change or supplement the standard declaration table.
The meanings of the variables are explained to you in the online help or in the Standard and System Functions manual.

In the example, the variable OB8_FLT_ID contains an identifier, if and which backup battery has failed.

## Exercise: Displaying the Startup Type (OB100)







**Noes:**

In the S7-300™ system, there is only the startup block OB 100 for *manual* as well as *automatic warm restart*. If different program reactions are necessary depending on the type of startup of the controller, then the corresponding *start information* has to be evaluated in OB 100.

*Start information example:*

Depending of the type of startup, the operating system stores one of the following identifiers in the *OB100_STRTUP* (BYTE) variable:

- B#16#81 = *manual warm restart*
- B#16#82 = *automatic warm restart*

*Example of the evaluation of a manual warm restart:*

```
L    OB100_STRTUP    // Load start info startup type
L    B#16#81         // Load ID for manual warm restart
==I                  // Compare for equality
=    Q 8.5           //Display manual warm restart
```

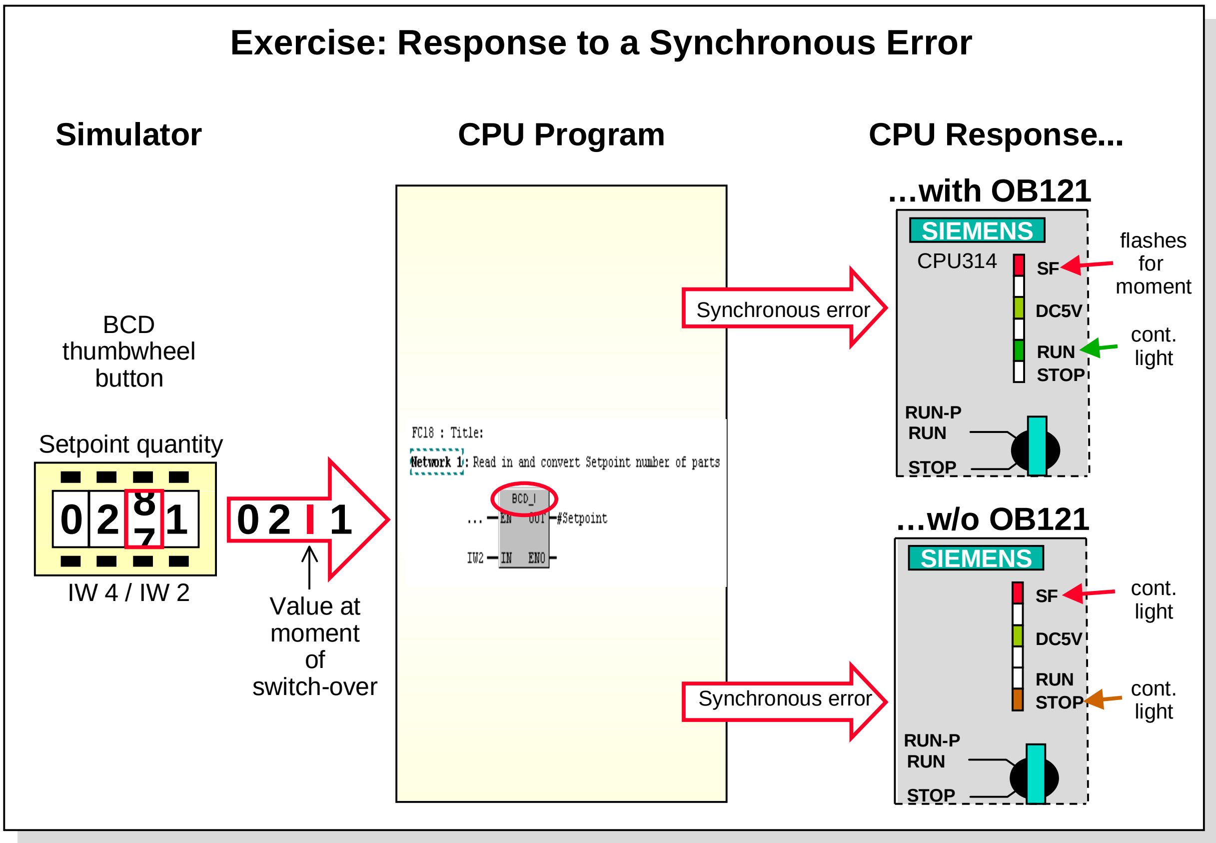
You can find more information on OB-specific start information in the Online Help.

**Task:**

- You are to program **OB 100** in such a way that a
  **manual warm restart** is displayed through the simulator LED **Q 8.5 (Q 4.5)** and an
  **automatic warm restart** is displayed through the simulator LED **Q 8.6 (Q 4.6)**
- You are to be able to acknowledge (reset) both LEDs through the simulator momentary contact switch "Ack_Rest." (I 0.7).

**What To Do:**

1. Perform a CPU memory reset, completely reload your program called "My_Program" into the CPU and continue to work with it
2. Program the startup display in OB 100 according to the task. Program the acknowledgement of the startup display in FC 15 where the operating modes are programmed.




SIMATIC® S7


Date: 12.03.03
File: PRO1_12E.16

SITRAIN Training for Automation and Drives

**Function**

In FC 18, you programmed the counting of transported parts and the comparison of the ACTUAL quantity with the SETPOINT quantity. You used the BCD thumbwheel button to enter these numbers. To prevent the CPU from going into the STOP state when you set the SETPOINT quantity, you loaded an "empty" OB121 (error OB for synchronous errors) into the CPU.

A format conversion from BCD to INT is first of all carried out in FC 18 for the necessary comparison function of the value read in with the BCD thumbwheel button. Since the thumbwheel button "rebounds" when you set the SETPOINT quantity and thus delivers non BCD digits, a "synchronous error“ results at the moment of switch-over in the format conversion from BCD to INT.

If the programmed error OB 121 has been downloaded into the CPU, the CPU remains in the RUN mode despite the synchronous error. Programming an error response (program) in OB 121 is not necessary. The error is, just the same, indicated for a short time on the red LED of the CPU with a flashing and is also entered in the diagnostic buffer.

If the programmed OB 121 has not been downloaded into the CPU, the CPU switches to the STOP mode.

**Task**

You are to check the described behavior of the CPU when a synchronous error occurs with and without a programmed error OB 121.

**What To Do**

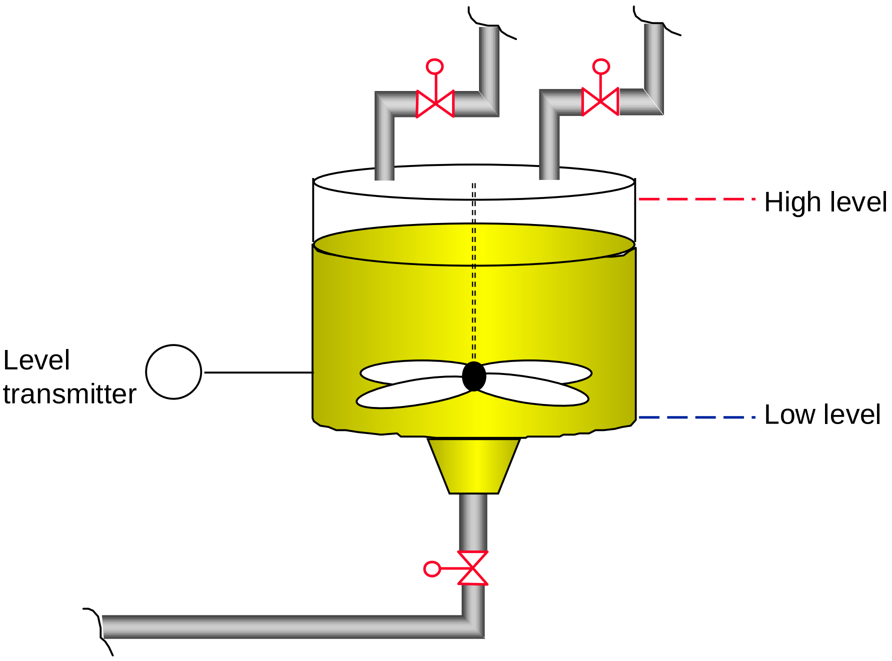
1. Delete the OB 121 online in the CPU (if it exists).
   *SIMATIC® Manager -> Online view -> Delete block*
2. Change the setting on the BCD thumbwheel button until the CPU goes into the STOP state. Read the relavent error information from the diagnostic buffer. Read the value the BCD thumbwheel button delivered the I STACK (contents of Accumulator 1) at the moment of interruption.
3. Download the "empty" OB 121 into the CPU. Once again, change the setting on the BCD thumbwheel button until an error is indicated on the CPU‘s SF-LED.

**Result**

With a programmed error OB, the CPU remains in RUN mode despite an error.

SIEMENS

# Analog Value Processing




SIMATIC® S7



Date: 12.03.03
File: PRO1_13E.1

SITRAIN Training for Automation and Drives

**Contents** Page

# Objectives

**Upon completion of this chapter the participant will ...**

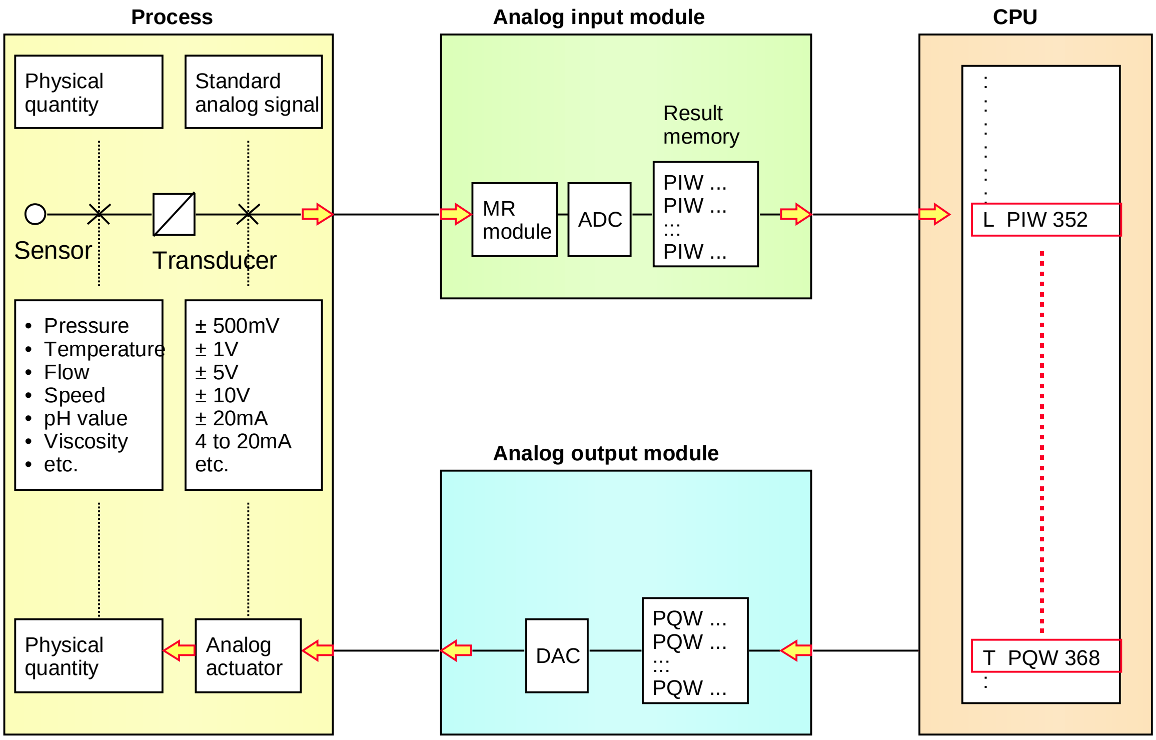
- ... know the principle of analog value processing
- ... be able to assign parameters to an analog module using the "HW Config" tool and will be able to set the measuring range module that belongs to the module to the correct position
- ... be able to address an analog module
- ... be able to interpret the resolution of a module
- ... be able to evaluate the result of an analog module in the program

SIMATIC® S7


Date: 12.03.03
File: PRO1_13E.2

SITRAIN Training for Automation and Drives

## Using Analog Modules








**Principle** In a production process, there are a variety of physical quantities (such as pressure, temperature, speed, rotational speed, pH value, and viscosity) that need to be processed in the PLC for automation purposes.

**Sensor** Measuring sensors respond to changes in the quantity to be measured by such things as linear expansion, angular ductability, and alteration of electrical conductivity.

**Transducer** Measuring transducers convert these above-mentioned changes into standard analog signals, such as: ± 500mV, ± 10V, ± 20mA, 4 to 20mA.

These signals are supplied to the analog input modules.

**ADC** Before these analog values can be processed in the CPU, they must be converted to digital form. The ADC (Analog-to-Digital Converter) on the analog input module handles this conversion.

The analog-to-digital conversion is performed sequentially. This means the signals are converted for each analog input channel in turn.

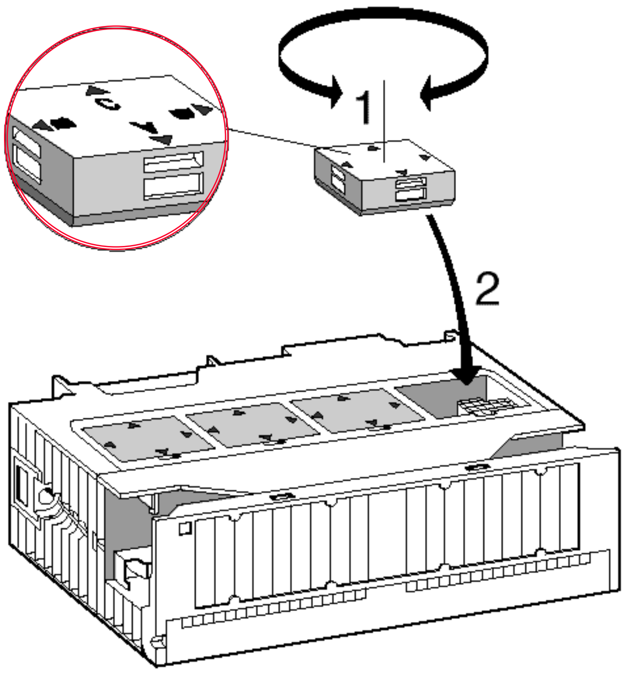
**Result Memory** The result of the conversion is stored in the result memory and remains there until it is overwritten by a new value.

You can use the "L PIW..." load instruction to read the converted analog value.

**Analog Output** The "T PQW..." transfer instruction is used to write the analog values the user program calculated to an analog output module, where a DAC (Digital-to-Analog Converter) converts them to standard analog signals.

**Analog Actuators** The analog output signals are standard signals such as ± 10V or 4 to 20mA. You can connect the analog actuators directly to the analog output modules without using converters.

## Measuring Range Module




SIMATIC® S7


Date: 12.03.03
File: PRO1_13E.4

SITRAIN Training for
Automation and Drives

**Type of Measurement** You set the type of measurement and the measuring range when you set coding keys on the measuring range module.

Special modules without coding keys have different terminals for voltage and current measurement. Thus, you set the type of measurement by wiring the appropriate terminal.

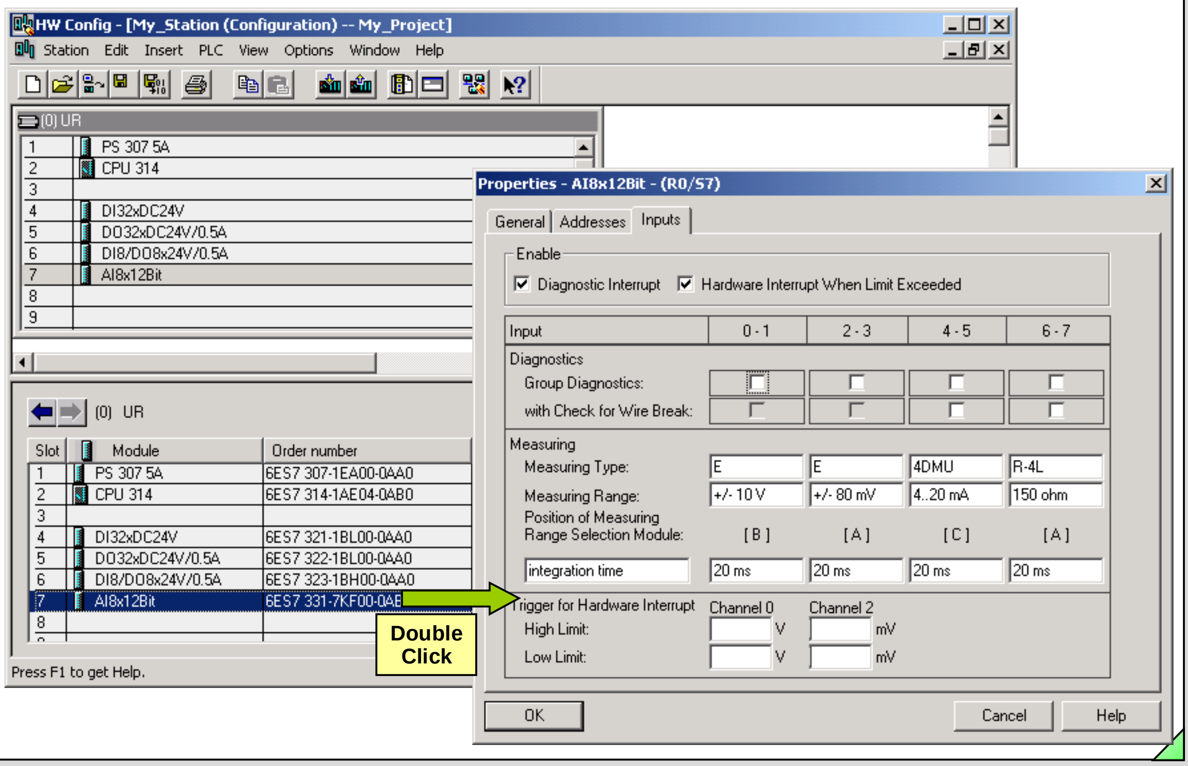
**Measuring Range Module** The measuring range modules with their coding keys are located on the left-hand side of the module. You must set them correctly before installing the module.

The possible settings are "A", "B", "C" and "D".
The settings for the various types of measurement and measuring ranges are printed on the module.

**Channel Groups** On some modules, several channels are grouped together to form a channel group. In this case, the coding key setting applies to the whole channel group.

SIEMENS

## Analog Module Parameters




SIMATIC® S7



Date: 12.03.03
File: PRO1_13E.5

SITRAIN Training for Automation and Drives

**Parameter Assignment**

The tool for assigning parameters to analog modules is HW Config. After downloading from the programming device to the S7-400/300™, the parameters are stored in the CPU. The CPU transfers these parameters to the relevant analog modules. In addition, measuring range submodules may need to be set to the required position.

In the RUN state of the CPU some of the parameters (dynamic parameters) can be changed via SFC blocks. However, after a RUN→STOP, STOP→RUN transition, the parameters created with HW Config come back into force.

**Diagnostic interrupt**

The module triggers a diagnostic interrupt on the CPU when an error occurs. Then an error information is entered in the diagnostic buffer and the CPU immediately processes the program in Diagnostic Interrupt Organization Block OB82. In this block, the user can program the necessary response to the error that occurred.

Which errors an analog module can recognize depends on the module type.

Error examples:

- Configuration/parameter assignment error
- Short circuit to ground (only for output channels)
- Wire break
- Missing load voltage L+ (not for AI 4x14 bit, Ex)

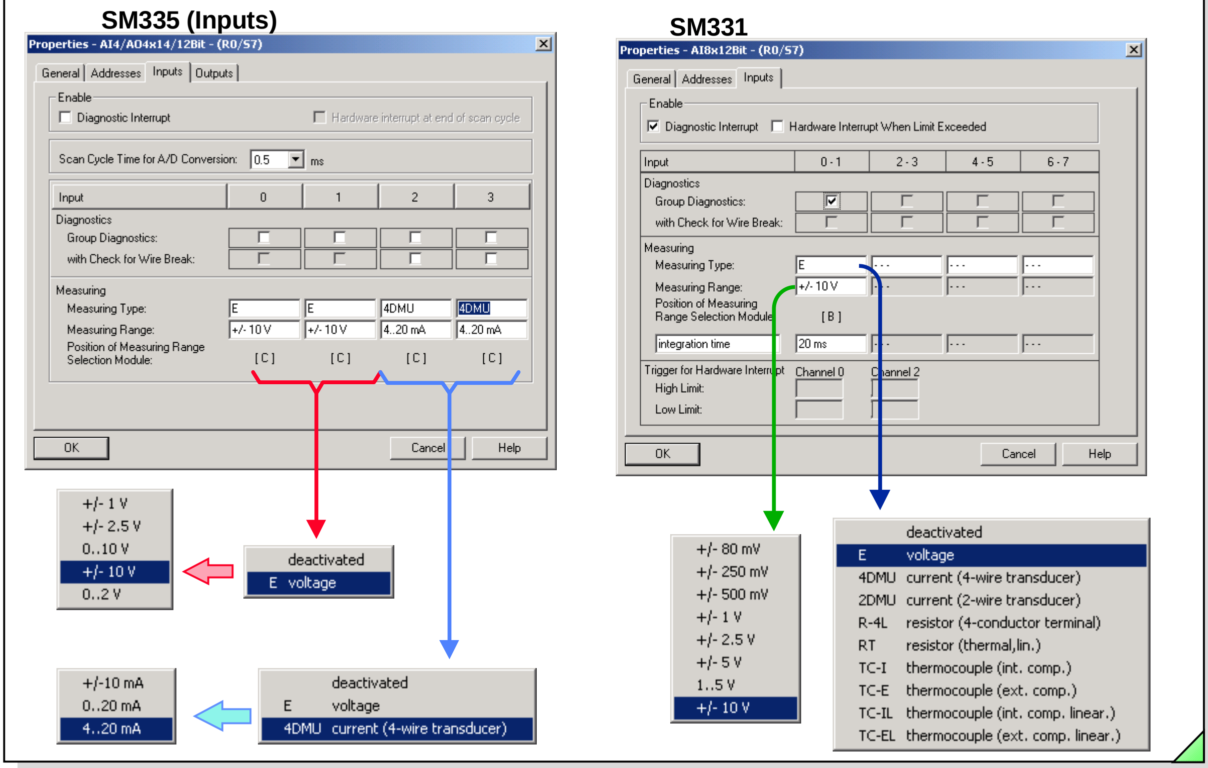
**Hardware interrupt**

Modules that have the capability to detect hardware conditions can trigger a hardware interrupt (OB40 to OB47). The module triggers a hardware interrupt when a particular event occurs (such as exceeding a voltage limit on a channel of an analog input module). Then the CPU immediately processes an interrupt program that the user stores in one of OB40 to 47, to determine the response to the event.

**Note:**

Only the first channel of a channel group can monitor the input value against the assigned limit values.
The interrupts are enabled for the whole analog module.

# Analog Input Modules




SIMATIC® S7


Date: 12.03.03
File: PRO1_13E.6

SITRAIN Training for Automation and Drives

**Scan Cycle Time**

The scan cycle time is the time it takes for the module to convert the analog signal to a digital signal and transfer the digitized values to memory. The cycle time is the sum of the conversion times of all activated analog input channels of the analog input module.
The A/D conversion time consists of a basic conversion time and additional processing times of the module for resistance measurement and wire-break monitoring. The basic conversion time depends directly on the conversion method (integrating method, successive approximation) of the analog input channel. In the case of integrating conversion methods, the integration time has a direct influence on the conversion time.

**Measuring Type**

Click the field to display and select the available measurement types (voltage, current...).

**Measuring Range**

Click the field to display and select the available effective ranges for the corresponding measurement type.

**Measuring Range Sub-Module**

Ensure that the measuring range sub-module is inserted on the module in the position shown.

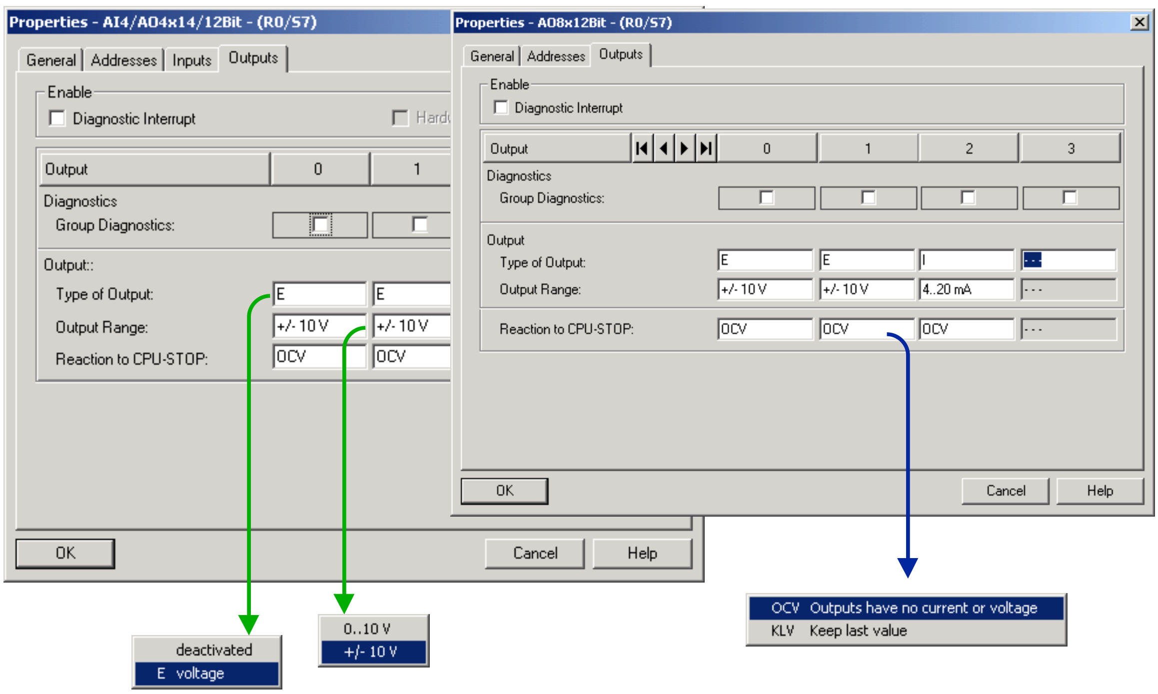
**Integration time / Interference frequency suppression**

Click the field to display and select the integration times or interference frequency suppression. The module sets the interference frequency suppression and resolution according to the selection made (that is, the integration time).

**Rule**

If a channel group is not connected, select "Deactivated." The remaining input values will then be updated in shorter time intervals.

## Analog Output Module




SIMATIC® S7


Date: 12.03.03
File: PRO1_13E.7

SITRAIN Training for Automation and Drives

**Type of Output**

Click the field to display and select the available output types (for example, current).

- Tip: To ensure that there is no voltage at the unconnected output channels, you should deactivate them (Type of Output: Deactivated) and leave them open. This also reduces the scan cycle time.

**Output Range**

Click the field to display and select the available output ranges for the selected output type.

**Reaction to CPU STOP**

Select how the outputs are to react in the case of a CPU STOP (not all settings are possible for every module):

- Switch to Substitute Value (SSV)

  - The substitute value is set to ”0” by default; that is, all outputs are switched off. You can set the substitute values for each individual output in the ”Substitute value” line. The substitute values must lie within the rated range.
- Retain Last Value (RLV)

  - If the module is to retain the last value output before the CPU enters the STOP mode.
- Outputs Without Voltage or Current (OWVC)

  - If the module is to switch off the outputs on CPU STOP (V/I = 0 V/mA).

**Warning**

Make sure that the system is always in a safe state when substitute values are output.

## Analog Value Representation and Measured Value Resolution

| Bit no. | | min. units | | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit value | | Dec. | Hex. | VZ | $2^{14}$ | $2^{13}$ | $2^{12}$ | $2^{11}$ | $2^{10}$ | $2^{9}$ | $2^{8}$ | $2^{7}$ | $2^{6}$ | $2^{5}$ | $2^{4}$ | $2^{3}$ | $2^{2}$ | $2^{1}$ | $2^{0}$ |
| Reso-lution in bits + sign | 8 | 128 | 80 | * | * | * | * | * | * | * | * | * | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 9 | 64 | 40 | * | * | * | * | * | * | * | * | * | * | 0 | 0 | 0 | 0 | 0 | 0 |
| | 10 | 32 | 20 | * | * | * | * | * | * | * | * | * | * | * | 0 | 0 | 0 | 0 | 0 |
| | 11 | 16 | 10 | * | * | * | * | * | * | * | * | * | * | * | * | 0 | 0 | 0 | 0 |
| | 12 | 8 | 8 | * | * | * | * | * | * | * | * | * | * | * | * | * | 0 | 0 | 0 |
| | 13 | 4 | 4 | * | * | * | * | * | * | * | * | * | * | * | * | * | * | 0 | 0 |
| | 14 | 2 | 2 | * | * | * | * | * | * | * | * | * | * | * | * | * | * | * | 0 |
| | 15 | 1 | 1 | * | * | * | * | * | * | * | * | * | * | * | * | * | * | * | * |

* = **0** or **1**

SIMATIC® S7


Date: 12.03.03
File: PRO1_13E.8

SITRAIN Training for Automation and Drives

**Representation**

Analog values are represented as the two's complement.
The value is positive if bit No. 15=0 and negative if bit No.15=1.

**Resolution**

If the resolution of an analog module is less than 15 bits, the analog value is written into the accumulator left-justified. The unused less significant bit positions are filled with "0"s.

**Integration Time**

The resolution is specified indirectly when you use the "HW Config" tool to select an integration time.
The following table for the SM331 illustrates the relationship between integration time, resolution and interference frequency suppression:

| Integration time (ms) | Resolution (in bits) | Interference frequency suppression (Hz) |
|---|---|---|
| 2.5 | 9 + sign bit | 400 |
| 16.6 | 12 + sign bit | 60 |
| 20 | 12 + sign bit | 50 |
| 100 | 14 + sign bit. | 10 |

**Accuracy**

Resolutions of between 8 and 15 bits are possible, depending on the type of module.

**Conversion Time**

The conversion time depends on the conversion procedure used in the module (integrating procedure or successive approximation).

The conversion times of the different modules are given in the S7-300™ manual.
Example: The SM344 has a conversion time of only 5 ms for all four input channels.

## Analog Value Representation of Different Measuring Ranges

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

SIMATIC® S7



Date: 12.03.03
File: PRO1_13E.9

SITRAIN Training for Automation and Drives

**Voltage, Current (Symmetrical)**

Encoding the symmetrical voltage or current ranges

- ± 80mV
- ± 250 mV
- ± 500 mV
- ± 1 V
- ± 2.5 V
- ± 5V
- ± 10V
- ± 3.2 mA
- ± 10 mA
- ± 20 mA

results in a rated range of -27648 to +27648.

**Voltage, Current (Asymmetrical)**

Encoding the asymmetrical voltage or current ranges

- 0 to 2 V
- 1 to 5 V
- 0 to 20 mA
- 4 to 20 mA

results in a rated range of 0 to +27648.

**Resistance**

Encoding the resistance ranges

- 0 to 150 Ohm
- 0 to 300 Ohm
- 0 to 600 Ohm

results in a rated range of 0 to +27648.

**Temperature**

Temperatures are measured with resistance thermometers or thermocouples. Encoding results in a rated range of ten times the temperature range:

| Sensor: | Temperature range: | Rated range when encoded: |
|---|---|---|
| • Pt 100 | -200 to + 850 ºC | -2000 to + 8500 |
| • Ni 100 | -60 to + 250 ºC | -600 to + 2500 |
| • Thermocouple type K | -270 to + 1372 ºC | -2700 to + 13720 |
| • Thermocouple type N | -270 to + 1300 ºC | -2700 to + 13000 |
| • Thermocouple type J | -210 to + 1200 ºC | -2100 to + 12000 |
| • Thermocouple type E | -270 to + 1000 ºC | -2700 to + 10000. |

## Analog Value Representation for the Analog Outputs

| Range | Units | Voltage, Output ranges: 0 to 10V | 1 to 5V | ± 10V | Current, Output ranges: 0 to 20mA | 4 to 20mA | ± 20mA |
|---|---|---|---|---|---|---|---|
| Overflow | >=32767 | 0 | 0 | 0 | 0 | 0 | 0 |
| Overrange | 32511 : 27649 | 11.7589 : 10.0004 | 5.8794 : 5.0002 | 11.7589 : 10.0004 | 23.515 : 20.0007 | 22.81 : 20.005 | 23.515 : 20.0007 |
| Rated range | 27648 : 0 | 10.0000 : 0 | 5.0000 : 1.0000 | 10.0000 : 0 : | 20.000 : 0 | 20.000 : 4.000 | 20.000 : 0 : |
| | : - 6912 | 0 | 0.9999 0 | : | 0 | 3.9995 0 | : |
| | - 6913 : - 27648 | | 0 | : -10.0000 | | 0 | : -20.000 |
| Underrange | - 27649 : - 32512 | | | - 10.0004 : - 11.7589 | | | - 20.007 : - 23.515 |
| Underflow | <=- 32513 | | | 0 | | | 0 |

SIMATIC® S7


Date: 12.03.03
File: PRO1_13E.10

SITRAIN Training for Automation and Drives

**Voltage, Current Symmetrical**

For symmetrical voltage or current ranges, a rated range of -27648 to +27648 is converted to:

- ± 10V
- ± 20mA.

**Voltage, Current Asymmetrical**

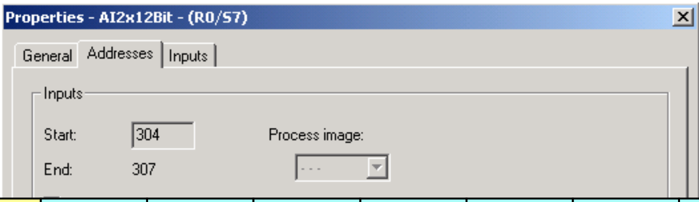
For asymmetrical voltage or current ranges, a rated range of 0 to +27648 is converted to:

- 0 to 10V
- 1 to 5V
- 0 to 20mA
- 4 to 20mA.

**Overflow**

If the value to be converted reaches the overflow range, the analog output module is disabled (0V, 0mA).

# Analog Module Addresses for the S7-300™




| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 |
| Slot | | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |

SIMATIC® S7


Date: 12.03.03
File: PRO1_13E.11

SITRAIN Training for Automation and Drives

**Address Area**

The S7-300™ has special address areas for analog inputs and analog outputs. These areas are separate from the process image input (PII) and process image output (PIQ) tables for digital modules.

Each address area extends from byte 256 to byte 767. Each analog channel occupies 2 bytes.

By default, each analog module occupies 16 bytes of access area. Like the digital Signal Modules, the slot location determines the starting byte number of the module.

**Access**

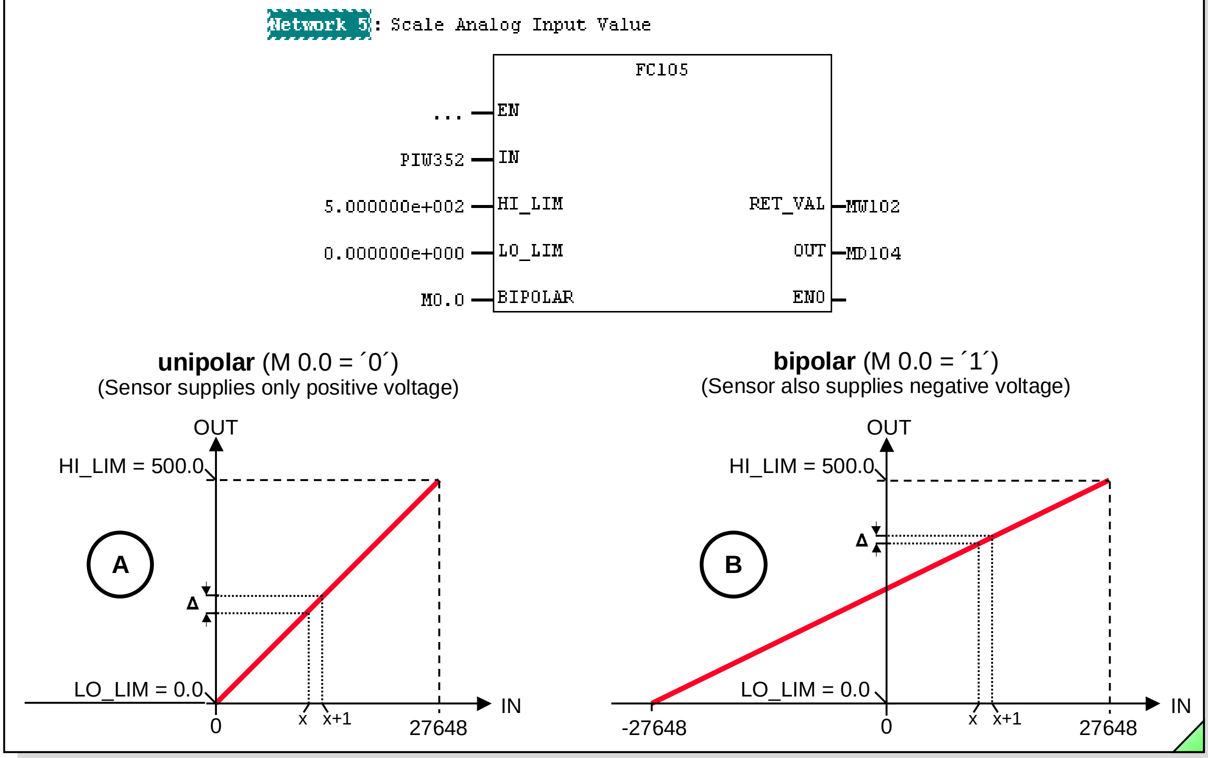
You use Load and Transfer instructions to access the analog modules.

Example: The statement "L PIW322" reads the second channel of the slot 8 module in rack 0.

**S7-400™**

On the S7-400™, the address area for the analog modules starts at byte 512.

## Scaling Analog Input Values




SIMATIC® S7


Date: 12.03.03
File: PRO1_13E.12

SITRAIN Training for Automation and Drives

**Examples** The level of a tank, whose volume is 500 liters, is to be measured in liters.

Example **A** shows the scaling when a sensor is used that supplies a measured voltage of 0V when the tank is empty and a measured voltage of +10V when the tank is full. Example **B** shows the scaling when a sensor is used that supplies a measured voltage of -10V when the tank is empty and a measured voltage of +10V when the tank is full.

**Resolution** In example **B,** the level is measured with twice the resolution or with half as much measuring tolerance **Δ**, since the volume of the tank is scaled to the greater units range of -27648 to +27648.

**Scaling** The analog module encrypts the voltage range of -10V to +10V in the value range of -27648 to +27648. The conversion of this value range to the original physical size (such as 0 l to 500 l) is called scaling.

The standard block FC 105 is used for scaling the analog value. FC 105 is supplied with the STEP 7 software in the "Standard Library" in the "TI-S7 Converting Blocks" S7 program.

**IN** The analog value at input IN can be read in from the module directly or can be passed from a data interface in INT format.

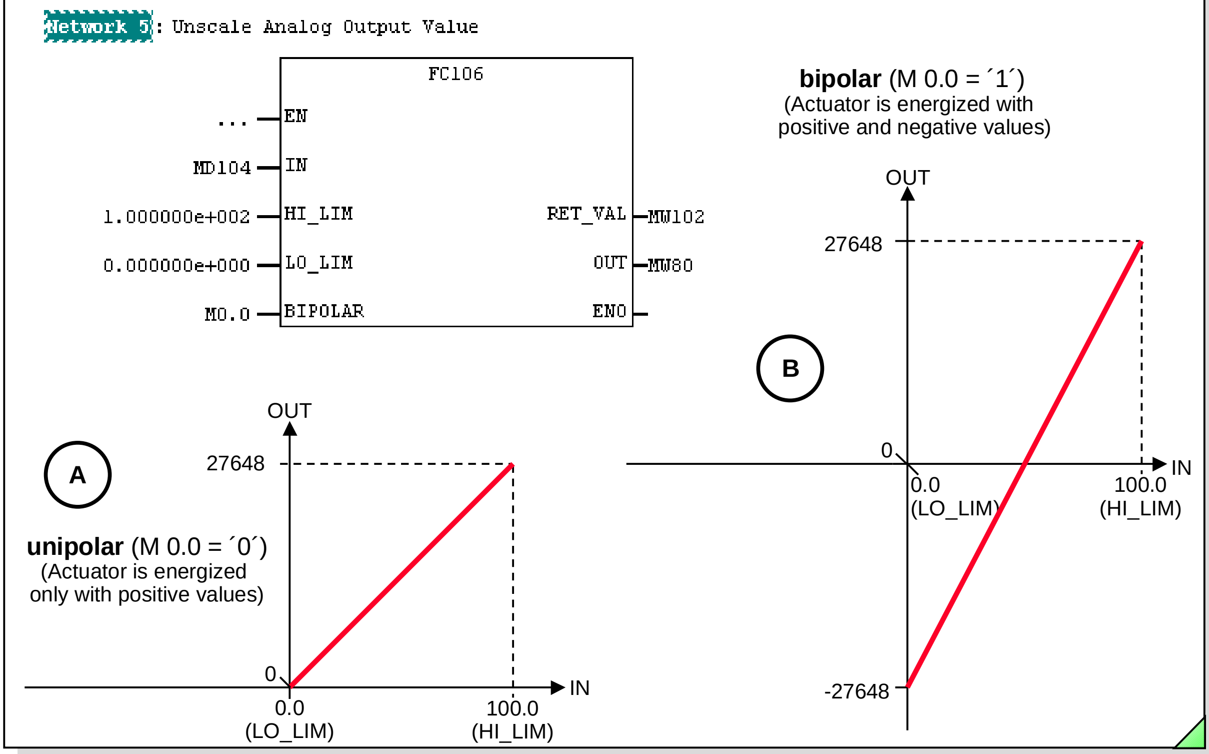
**LO_LIM, HI_LIM** Inputs LO_LIM (low limit) and HI_LIM (high limit) are used for specifying the limits of the basic physical size. In the example, a conversion to the range 0 to 500 liters is made.

**OUT** The scaled value (physical size) is stored as a real number at output OUT (LO_LIM <= **OUT** <= HI_LIM).

**BIPOLAR** At input BIPOLAR you can specify as to whether only positive or also negative values are to be converted. If an operand with the state ´0´ (unipolar) is passed to the parameter, scaling is made for the range 0 to +27648. If the operand state is ´1´ (bipolar), scaling is made for the range -27648 to +27648.

**RET_VAL** The output RET_VAL supplies the value 0 when execution is error free.

# Unscaling Analog Output Values




SIMATIC® S7


Date: 12.03.03
File: PRO1_13E.13

SITRAIN Training for Automation and Drives

**Examples**

An analog value in the range 0.0 to 100.0% that is calculated by the user program is converted (unscaled) with FC106 to the range 0 to +27648 (unipolar) or -27648 to +27648 (bipolar). When the unscaled value is output to an analog output module, this module will energize the analog actuator (such as a servo valve) with a value such as 0V to +10V (unipolar) or -10V to +10V (bipolar).

Example **A** shows the scaling when an actuator is used that is to be energized with the value 0 (0V or 0mA) when the program value is 0% and is to be energized with maximum value ( such as +10V or 20mA) when the program value is 100%.

Example **B** shows the scaling when an actuator is used that is to be energized with the minimum value (-10V or -20mA) when the program value is 0% and is to be energized with the maximum value (such as +10V or 20mA) when the program value is 100%.

**Unscaling**

A value calculated by the program - in the example shown a percentage - must be converted (unscaled) to the value range of the analog output module.

The standard block FC 106 is used for unscaling. FC 106 is supplied with the STEP 7 software in the "Standard Library" in the "TI-S7 Converting Blocks" S7 program.

**IN**

The value calculated by the program must be passed in the REAL format.

**LO_LIM, HI_LIM**

Inputs LO_LIM (low limit) and HI_LIM (high limit) specify the limits for the program value. In the example, this is the range 0.0% to 100.0%.

**OUT**

The unscaled value is output in the INT format at output OUT.

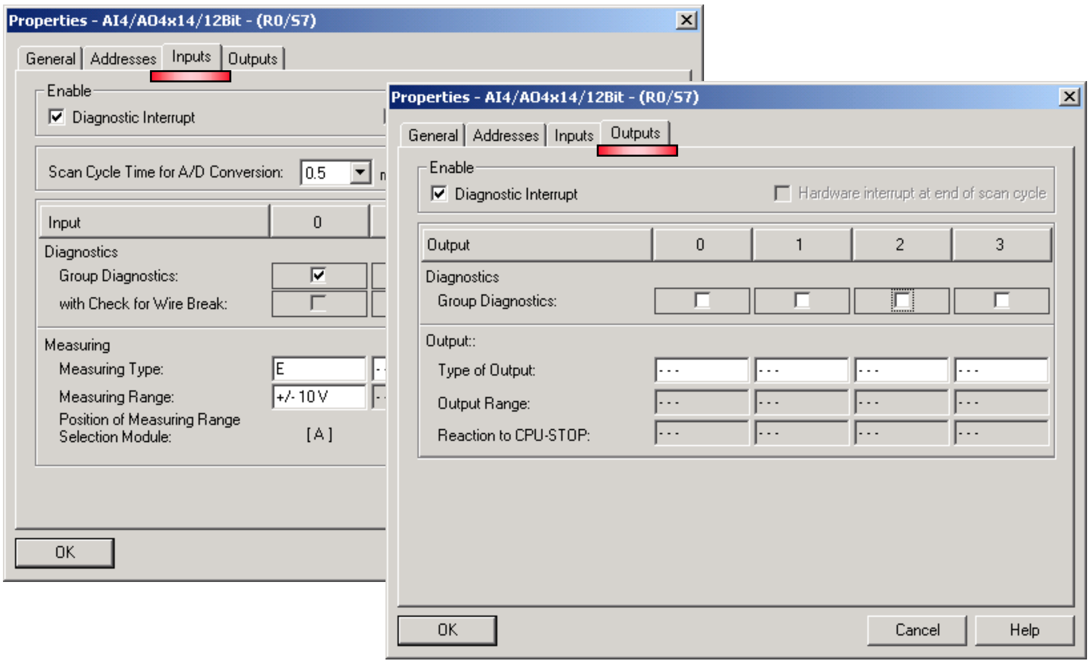
**BIPOLAR**

At input BIPOLAR you can specify as to whether only positive or also negative values are to be converted. If an operand with the state ´0´ (unipolar) is passed to the parameter, unscaling is made for the range 0 to +27648. If the operand state is ´1´ (bipolar), unscaling is made for the range -27648 to +27648.

**RET_VAL**

The output RET_VAL supplies the value 0 when execution is error free.

## Exercise: Assigning Parameters to the Analog Module SM335




SIMATIC® S7


Date: 12.03.03
File: PRO1_13E.14

SITRAIN Training for Automation and Drives

**Note**

Depending on which analog module is in your training unit, you are to do either this exercise or the one on the following page.

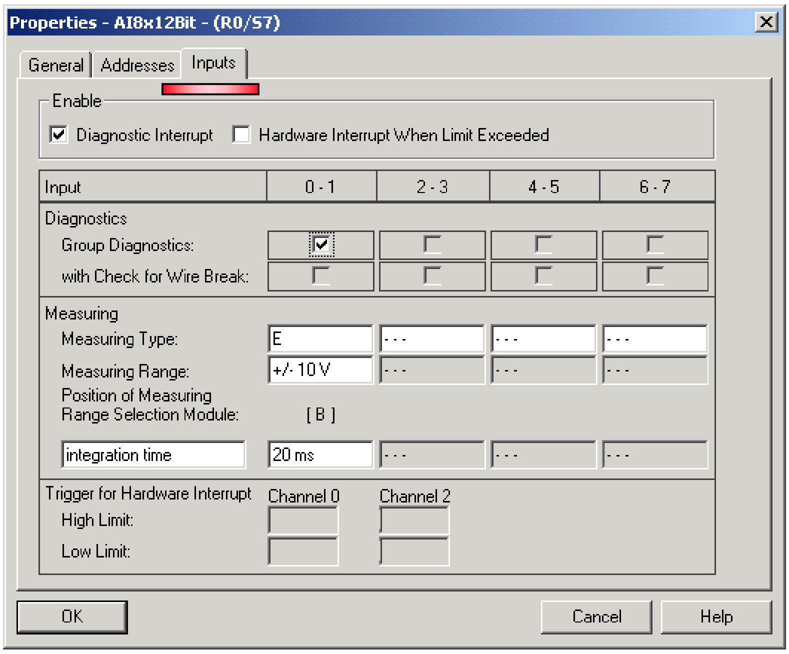
**Task**

You are to assign parameters to the analog module using the parameters shown in the slide above

**What To Do**

1. Use the HW Config tool to open your HW Station called "My_Station" .
   *SIMATIC® Manager -> Double-click the Hardware icon*
2. Start the parameter assignment dialog box for the analog module. Select the analog module and open the module‘s object properties window.
   *Edit -> Object Properties* or *Double-click the Analog Module entry*
3. Assign parameters to the module by specifying the *Measuring Type* and the *Measuring Range* for the *Inputs* as is indicated in the slide above. Activate the *Diagnostic Interrupt* ( don‘t forget the check mark(s) for the individual input channels in Group Diagnostics ! ). Since the output of the analog values is not required in the following exercise, *deactivate* all *Outputs*.
4. Save and compile the modified hardware configuration / parameter assignment.
   *HW Config -> Station -> Save and compile*
5. Download the modified system data to the CPU.
   *HW Config -> PLC -> Download...*

## Exercise: Assigning Parameters to the Analog Module SM331




SIMATIC® S7


Date: 12.03.03
File: PRO1_13E.15

**SITRAIN** Training for Automation and Drives

**Note**

Depending on which analog module is in your training unit, you are to do either this exercise or the one on the previous page.

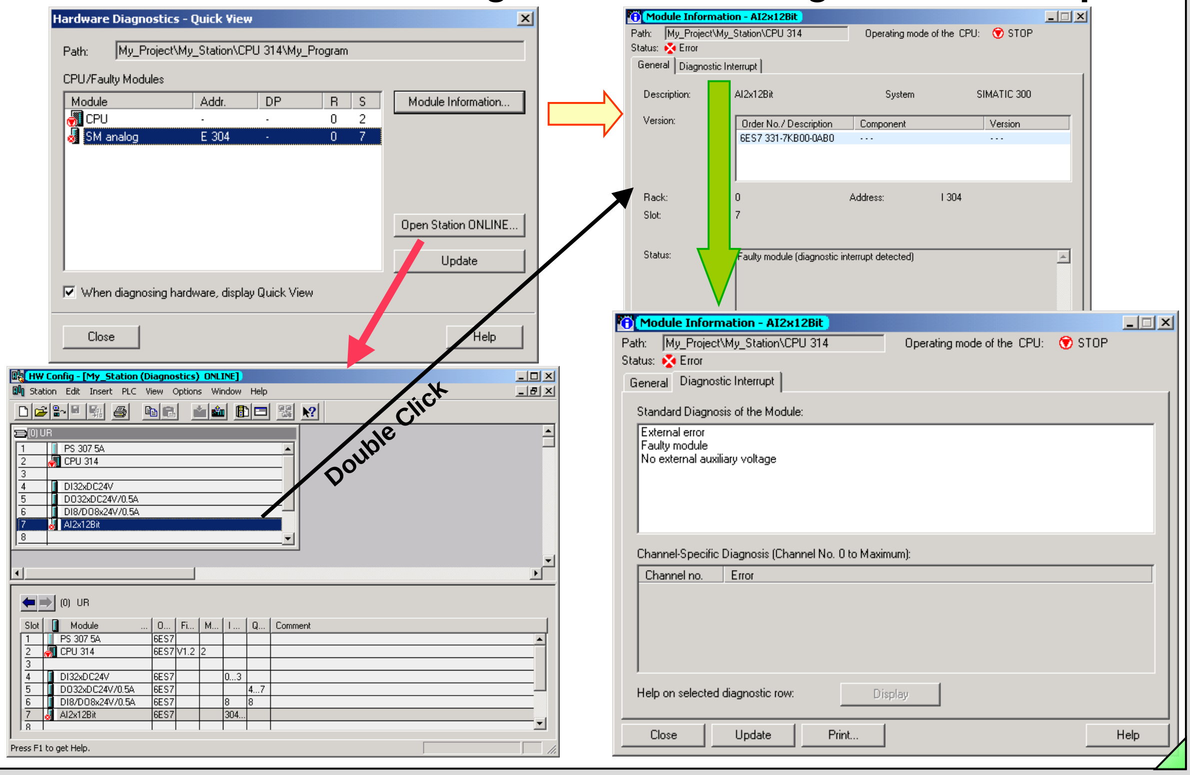
**Task**

You are to assign parameters to the analog module using the parameters shown in the slide above

**What To Do**

1. Use the HW Config tool to open your HW Station called "My_Station".
   *SIMATIC® Manager -> Double-click on the Hardware icon*
2. Start the parameter assignment dialog box for the analog module. Select the analog input module and open the object properties window.
   *Edit -> Object Properties* or *Double-click Analog Module*
3. Assign parameters to the module by specifying the *Measuring Type* and the *Measuring Range* for the *Inputs* as is indicated in the slide above. Activate the *Diagnostic Interrupt* ( don‘t forget the check mark(s) for the individual input channels in Group Diagnostics ! ).
4. Save and compile the modified hardware configuration / parameter assignment.
   *HW Config -> Station -> Save and compile*
5. Download the modified system data to the CPU.
   *HW Config -> PLC -> Download...*

# Exercise: Hardware Diagnostics with Diagnostic Interrupt




SIMATIC® S7



Date: 12.03.03
File: PRO1_13E.16

**SITRAIN** Training for Automation and Drives

**Task**

Your task is to initiate an analog input module diagnostic interrupt. You have assigned parameters to your analog module and activated the diagnostic interrupt in the previous exercise. Adjusting the simulator's analog input potentimeter will initiate the diagnostic interrupt.

After the CPU has gone into the STOP state because of the diagnostic interrupt, "troubleshoot" the "error" that occurred with the *Hardware Diagnostics* test function (see slide).

**What To Do**

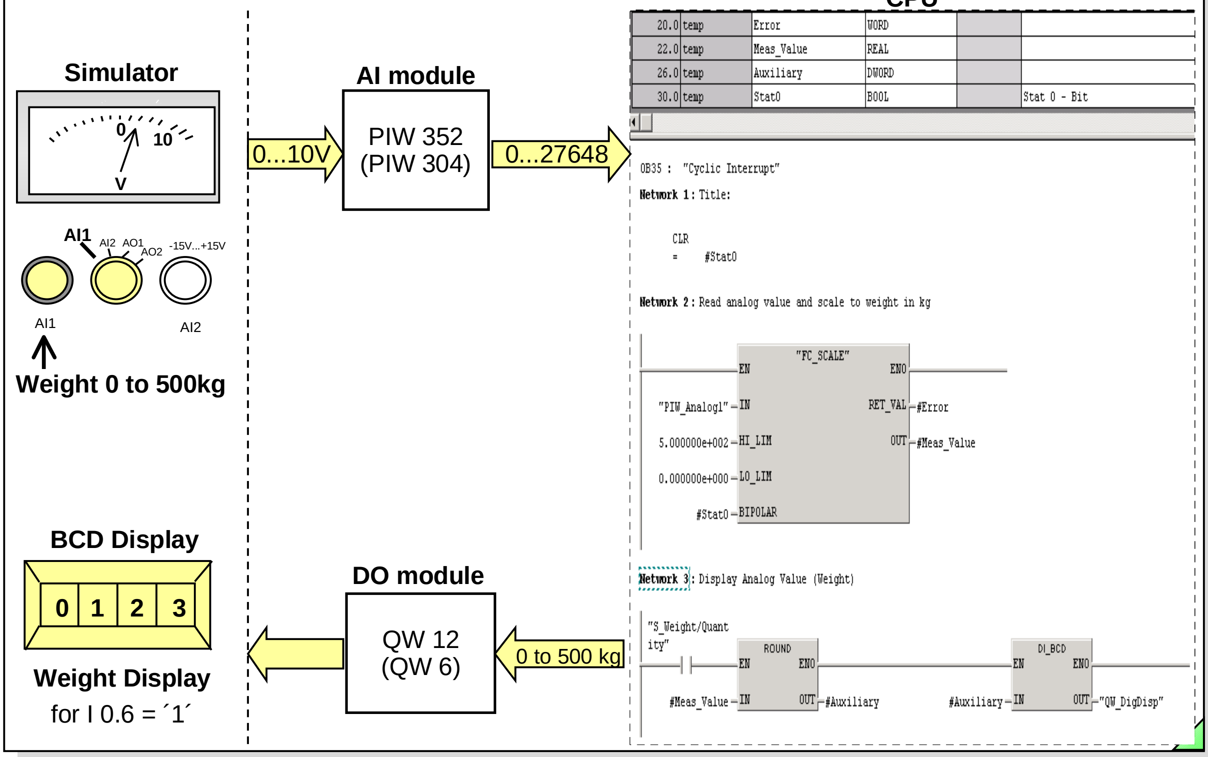
**NOTE:** Depending on the settings you made in the SIMATIC® Manager, you either start with the Hardware Diagnostics Quick View or the complete Station View.

You can change the settings with
*SIMATIC® Manager -> Options -> Customize -> View*

1) Initiate a Diagnostic Interrupt

- Adjust, on the simulator, the analog input potentimeter until a system fault occurs.
- Activate the *Diagnose hardware* test function.
  *SIMATIC® Manager -> PLC -> Hardware Dignostics*
- Double-click the CPU entry or click on the *Module Information* button to view the event entry in the diagnostic buffer.
- Double-click the analog module entry or select the analog module entry and click on the *Module Information* button to view the diagnostic data.

## Exercise: Recording and Displaying the Weight of Transported Parts






**Display Function Up Until Now**

The number of transported parts is displayed on the BCD digital display. The counting function and the display of the current quantity is programmed in FC 18.

**Task**

The parts transported in AUTO mode are to be weighed at the Conveyor End (light barrier). The current weight of 0 to 500kg can be set using the simulator potentiometer (0 to 10V).

When the simulator switch I 0.6 is switched on, the current weight (0 to 500kg) is displayed on the BCD digital display. When the switch is switched off it shows the current number of transported parts.

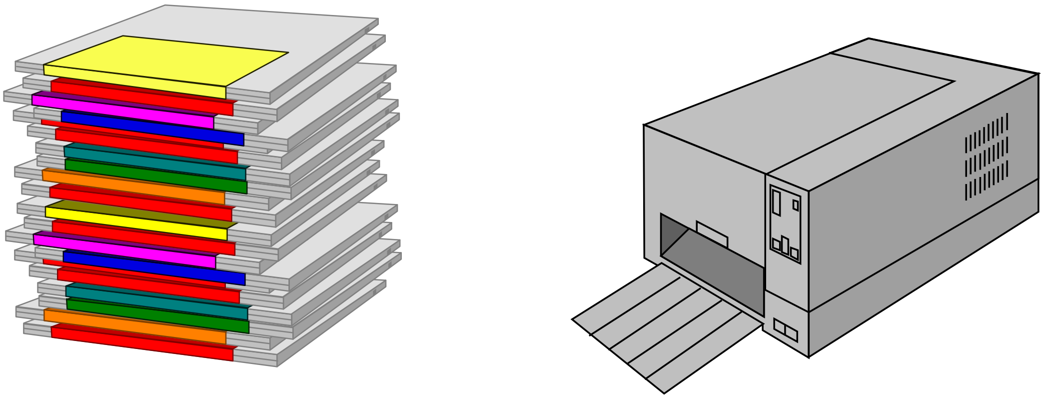
If the actual weight of the transported part is less than 100 kg., or more than 400 kg., it is considered a defective part. This part is not to be counted.

**What to Do**

1. Assign parameters to the CPU so that the OB 35 organization block (Cyclic Interrupt) is executed every 250ms.
   *HW Config -> Double-click on CPU -> Cyclic Interrupt*
2. In OB 35, program the display and the control of the actual weight (limit value monitoring using comparison functions).
   - For scaling the analog measuring result in OB 35, call the TI-S7 Converting Block FC 105, which you copied from the library into your project at the beginning of the course.
   - Program the bit memory M 35.0 as the result of the weight check. Assign the state of "1" to the bit memory, when the part's weight lies in the required range. Assign the state "0," if it is a defective part.
   - Transfer the scaled measuring result to the BCD digital display only when simulator switch I 0.6 is switched on.
3. Gate the bit memory M35.0 in FC 18 so that defective parts are not counted. Make the display of the quantity on the BCD display dependent of the simulator switch I 0.6.

SIEMENS

# Documenting, Saving, Archiving



SIMATIC® S7



Date: 12.03.03
File: PRO1_14E.1

SITRAIN Training for Automation and Drives

## Contents

**Page**

# Objectives

**Upon completion of this chapter the participant will ...**

... be familiar with the documentation possibilities for commenting on blocks and be able to apply these

... understand the "Manage Multilingual Texts" function for projects

... be able to print out documented programs

... understand the memory concept of SIMATIC S7-300™/400™ and the resulting possibilities for program modifications

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

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

... be able to archive/retrieve a project on/from a diskette

## Overview of Documentation Tools

- Network title
- Network comment
- Statement comment
- Program overview
- Cross references
- Assignment of I/Q/M/T/C
- Checklists
- Symbol table
- Configuration
- Network configuration



Date: 12.03.03
File: PRO1_14E.3

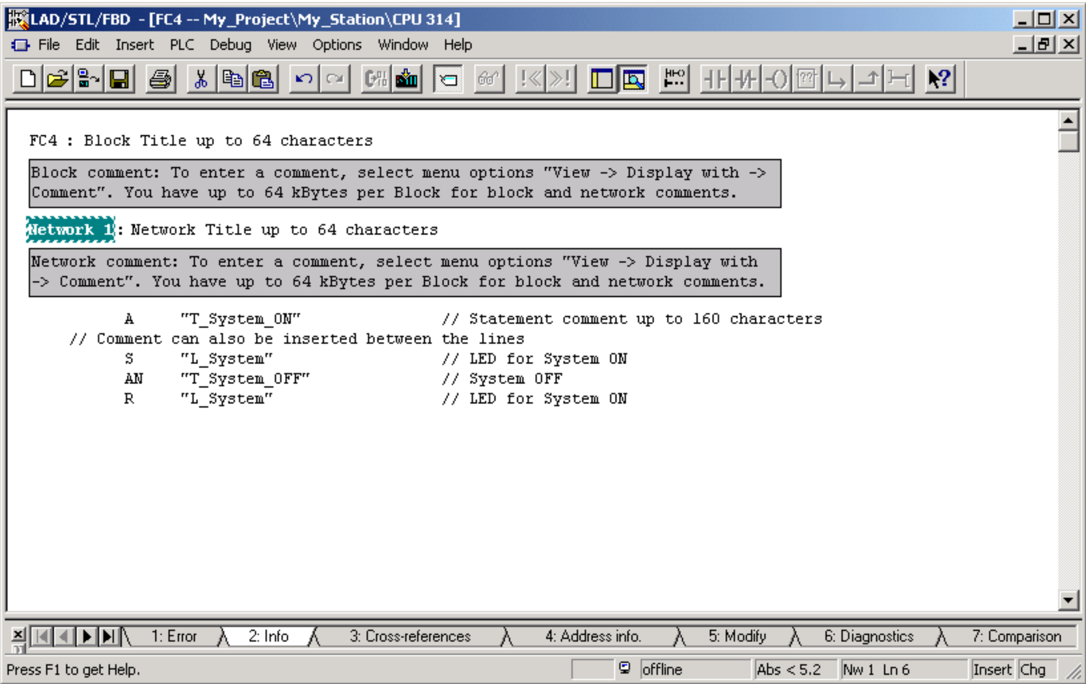


**Overview** The slide shows the different documentation tools available. All the documentation tools have a print function.

**Printer** The printer used for documentation is the one installed under Windows. If you want to use a different printer, you must set it up with the Windows Control Panel.

**DOCPRO** The DOCPRO optional software is also available for superior documentation and for writing wiring manuals.

## Block Documentation




SIMATIC® S7


Date: 12.03.03
File: PRO1_14E.4

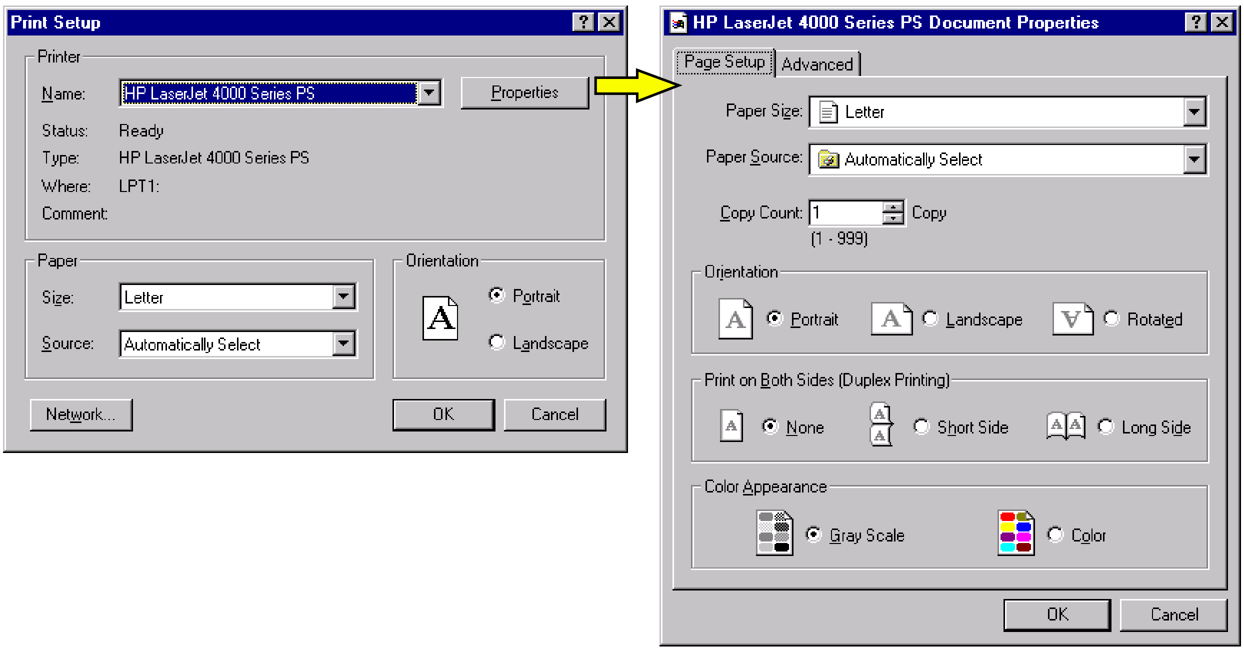
SITRAIN Training for Automation and Drives

**Block Comments** The slide above shows the different comment facilities available for a program block (OB, FC, FB).

**Printing** To start the Print function:

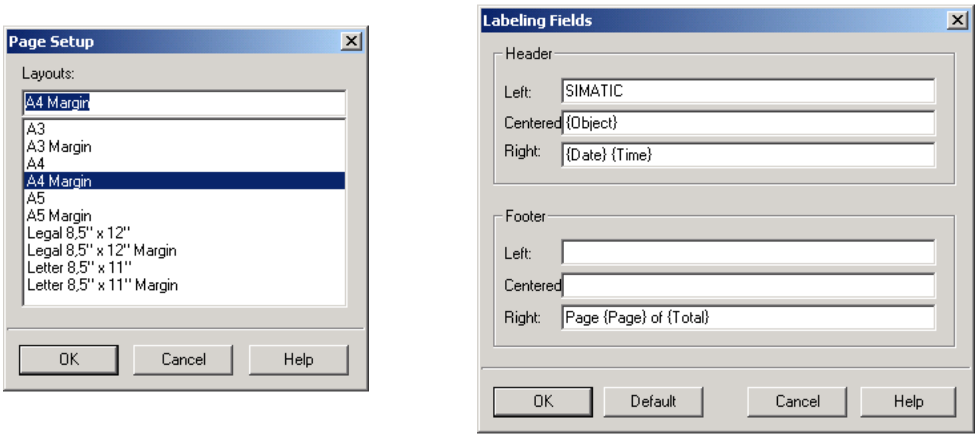
- Click the printer icon or
- Select the menu option *File --> Print*.

**Print Setup** You can change the printer settings by selecting the *File --> Print Setup...* menu option.



SIEMENS

## Page Setup




SIMATIC® S7


Date: 12.03.03
File: PRO1_14E.5

**SITRAIN** Training for Automation and Drives

**Page Setup**

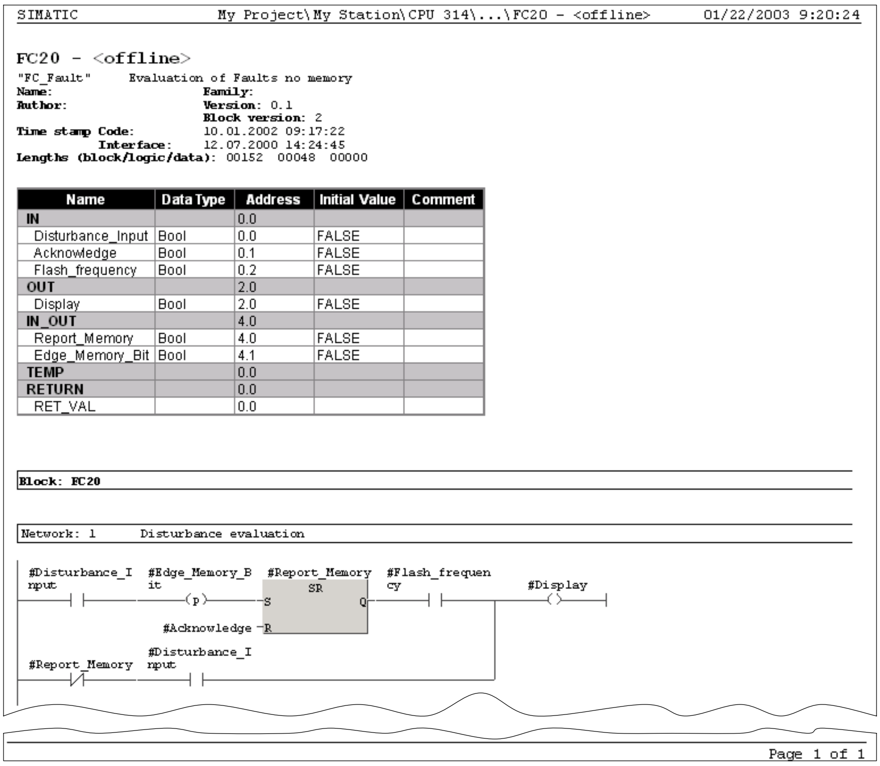
When you select the *File --> Page Setup...* menu option, a dialog box in which you can select the print format (such as A4 Margin) appears.

**Headers/Footers**

In the SIMATIC® Manager you can set the headers and footers for the documentation for an entire project with all the tools.

Select the *File -> Labeling fields...* menu option to display a dialog box for entering text for the headers and footers.
Fields for printing out the current date of the printout, the page number, and the name of the object are provided in the headers and footers (such as {Date} {Time}, Page {Page}, {Object}).

# 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

SIMATIC® S7


Date: 12.03.03
File: PRO1_14E.6

**SITRAIN** Training for Automation and Drives

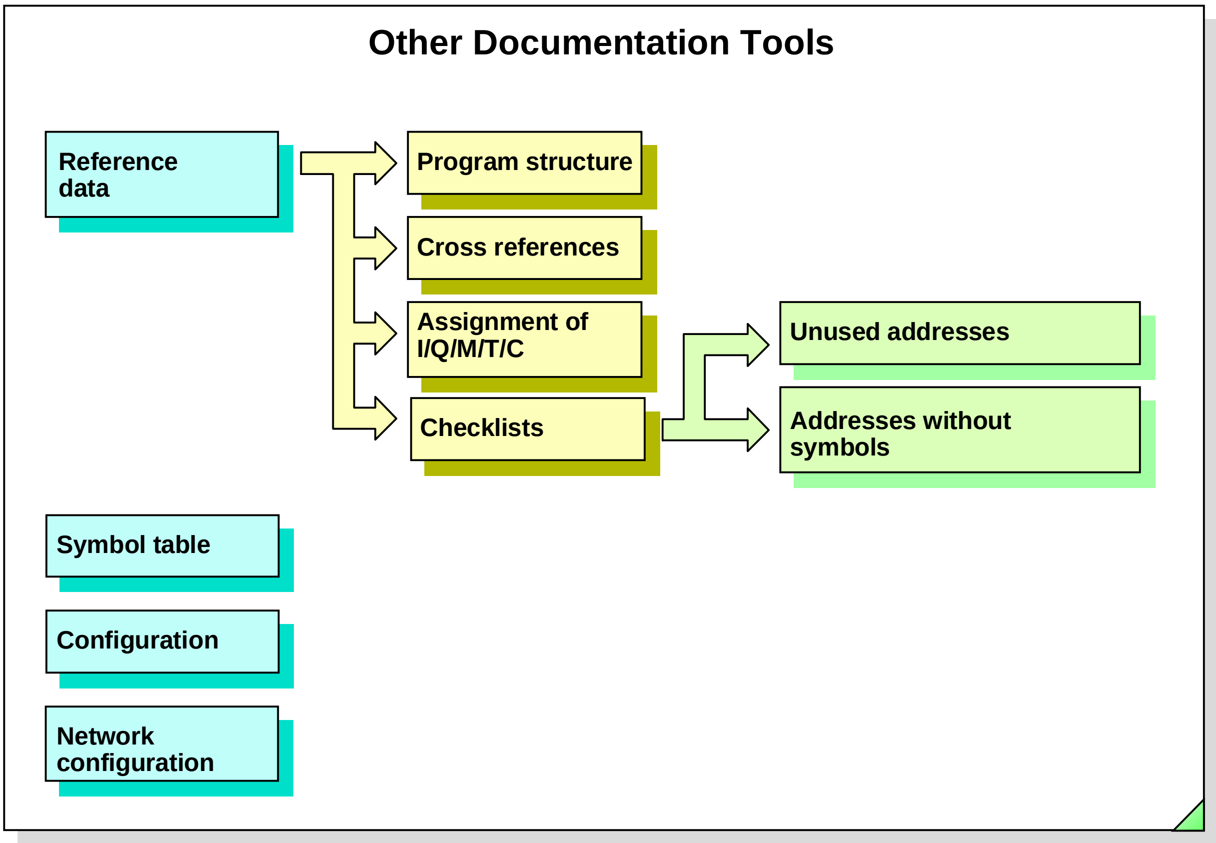
**Print Preview**

For a preview of what your printout will look like, select the *File -> Print Preview...* menu option.

**Note**

The appearance of LAD program printouts depends on the settings made under the *Options -> Customize -> LAD/FBD* menu option in the LAD/STL/FBD editor.

Example: The setting for the length of the address field affects the number of contacts that can appear side by side in the printout and the number of characters of the symbol name that fit on a line above the contacts.




SIMATIC® S7


Date: 12.03.03
File: PRO1_14E.7

SITRAIN Training for Automation and Drives

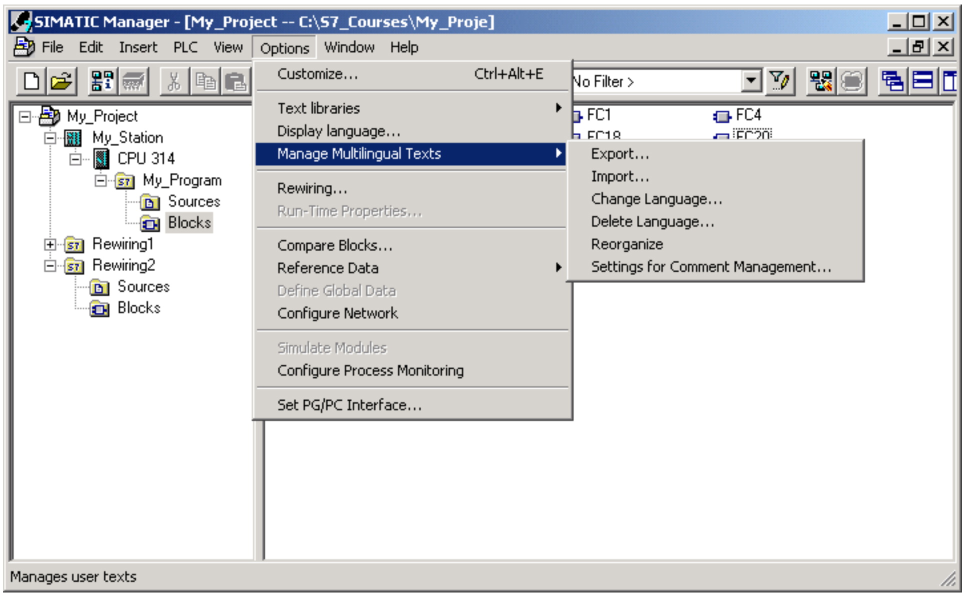
**Reference Data** — If you print out the reference data it makes troubleshooting, in particular, easier. You will find more information in the "Troubleshooting" chapter.

**Symbol Table** — The symbol table contains the association between absolute address, symbol names and symbol comments. See the "Symbols" chapter for more information.

**Configuration** — Configuration data generated with the HW Config tool. The printout is in text form. If you want a graphic printout, you can copy the graphics onto the clipboard and then paste them in another program such as Winword and print it out.

**Network Configuration** — Displays in graphic form the stations of a networked system with the relevant configuration data such as the MPI address.

## Managing Multilingual Project Documentation




SIMATIC® S7


Date: 12.03.03
File: PRO1_14E.8

SITRAIN Training for Automation and Drives

**Function**

STEP 7 lets you multilingually manage the documentation (texts and comments) created in a project. These can be exported from the project for translation purposes and then later be reimported in the language in which they were translated. The opportunity then exists to choose between different languages for the documentation.

The following types of texts can be managed multilingually.

- Block titles and block comments
- Network titles and network comments
- Line comments from STL programs
- Comments from symbol tables, variable declaration tables, user-defined data types and data blocks
- Comments, status names and so on, of blocks that were created with engineering tools such as S7-GRAPH or S7-PDIAG.

**Export**

The Export is carried out for all blocks and symbol tables that are located in the selected object folder. For every text type (see above), an export file is generated that can then be edited with EXCEL. This file contains a column with the source text in the original language and a column in which the translated text can be written.

**Import**

During Import, the translated text is accepted into the selected project. The translated text is only accepted if the original source text still exists.

**Change Language**

For Change Language, all languages can be selected that were imported into the project. The change is carried out for all selected object folders.

**Delete Language**

When you delete a language, all text in that language is deleted from the internal data management.

## Overview: Saving User Data

| | |
|---|---|
| **Uploading the program from the CPU to the PG/PC (PLC copy)** | 1. Create a new S7 program in the SIMATIC® Manager<br>2. Switch to the online view<br>3. Open the new S7 program and select the "Blocks" folder<br>4. SIMATIC® Manager -> PLC -> Upload |
| **Upload Station in the PG/PC** | • SIMATIC® Manager ->PLC -> Upload Station |
| **Load program from PG/PC 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"<br>2a. Use drag & drop to copy blocks to the "S7 Memory Card"<br>2b. With the "Blocks" folder of an S7 program highlighted, select: 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 (only S7-400™)** | 1. Select the CPU on whose Memory Card the project data are to be saved<br>2. SIMATIC® Manager -> PLC -> Save Project on Memory Card |
| **Project archiving on diskette** | 1. SIMATIC® Manager -> File -> Archive<br>2. Select project to be archived<br>3. Specify name and storage path of the archive file and start function<br>4. In the Windows Explorer, copy the archive file onto diskette |

SIMATIC® S7


Date: 12.03.03
File: PRO1_14E.9

SITRAIN Training for Automation and Drives

**Uploading a Program from CPU into PG/PC** With this function, you copy all blocks and the system data without documentation (symbol table, variable and parameter names, comments) from the CPU into the selected program folder. The selected program folder then contains a "PLC copy" with the current online program.

**Upload Station in PG/PC** With this function, you load the PLC's "actual" hardware station as a new station in the project. It is not possible to overwrite an already existing station.

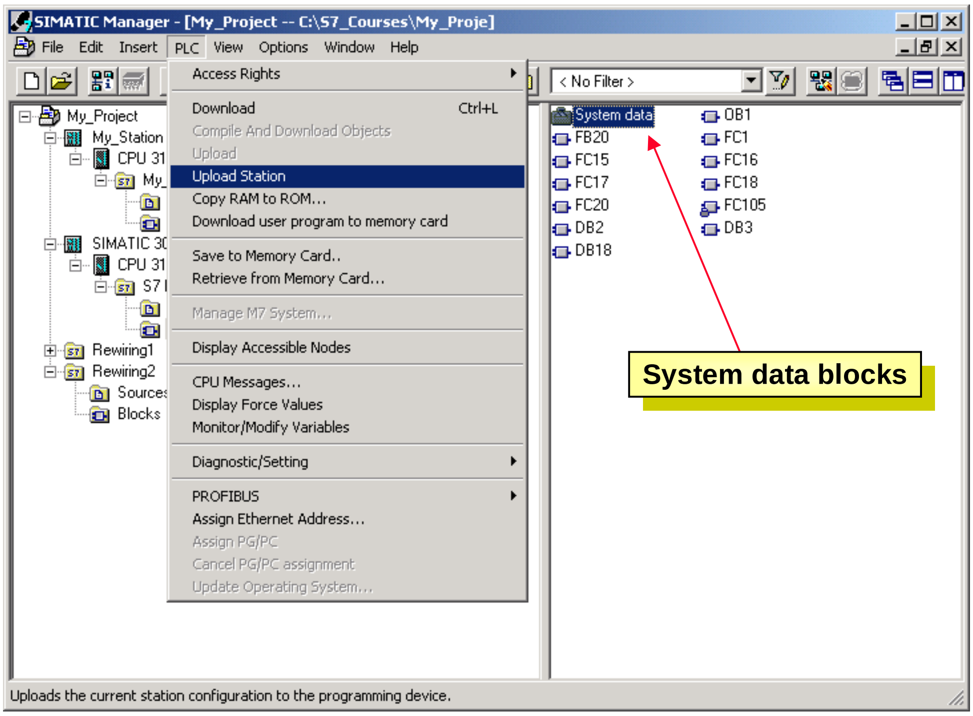
**Load Program from PG/PC to Memory Card** You can load the blocks and system data from the blocks folder of an S7 program onto a memory card. You can insert the memory card in the interface of the PG/PC or in the slot provided by the CPU if the CPU supports this service.

**Copy Program from CPU to Memory Card** If a user program is stored on a memory card, you can still make program changes online. The modified blocks are stored in the internal RAM of the CPU, while the unchanged block remains stored on the memory card. You can store the modified blocks on the memory card with the Copy RAM to ROM function.

**Archiving Project on Memory Card** You save the entire data of the project (such as user programs with all comments, symbol tables, and hardware configurations from all hardware stations) on the memory card with the *"Save Project on Memory Card"* function.

**Archiving Project on Diskette** With the *"Archive Project "* function, you save the complete data of the project (such as user programs with all comments, symbol tables, and hardware configurations from all hardware stations) in an archive file in compressed format (such as *.zip or *.arj). The archive file is much smaller than the non-archived project and you can move or copy the archive as often as you like with the Windows Explorer.

# Uploading a Program from the CPU to the PG







**Uploading a Program from CPU to PG**

When you have completed the startup phase, you should have a copy of the final version of the program on the hard disk of the PG.
The best way of doing this is to save the program with all its comments and symbols on the hard disk before starting it up on the PLC. When you make changes to the program, you should always save the modified blocks on the hard disk immediately so that you don't lose the comments and symbols.

If the program is not on your PG, you can upload the blocks from the CPU. In this case, the comments and symbols will be missing. Don't forget to upload the system data blocks because they contain configuration and communication data.

**What to Do**

To upload an entire program from the CPU to the PG, carry out the following steps:

- Create a new S7 program in the SIMATIC® Manager
- Click the "Online" icon in the toolbar
- Open the S7 program and select the "Blocks" folder (user program)
- Select the *PLC --> Upload* menu option.

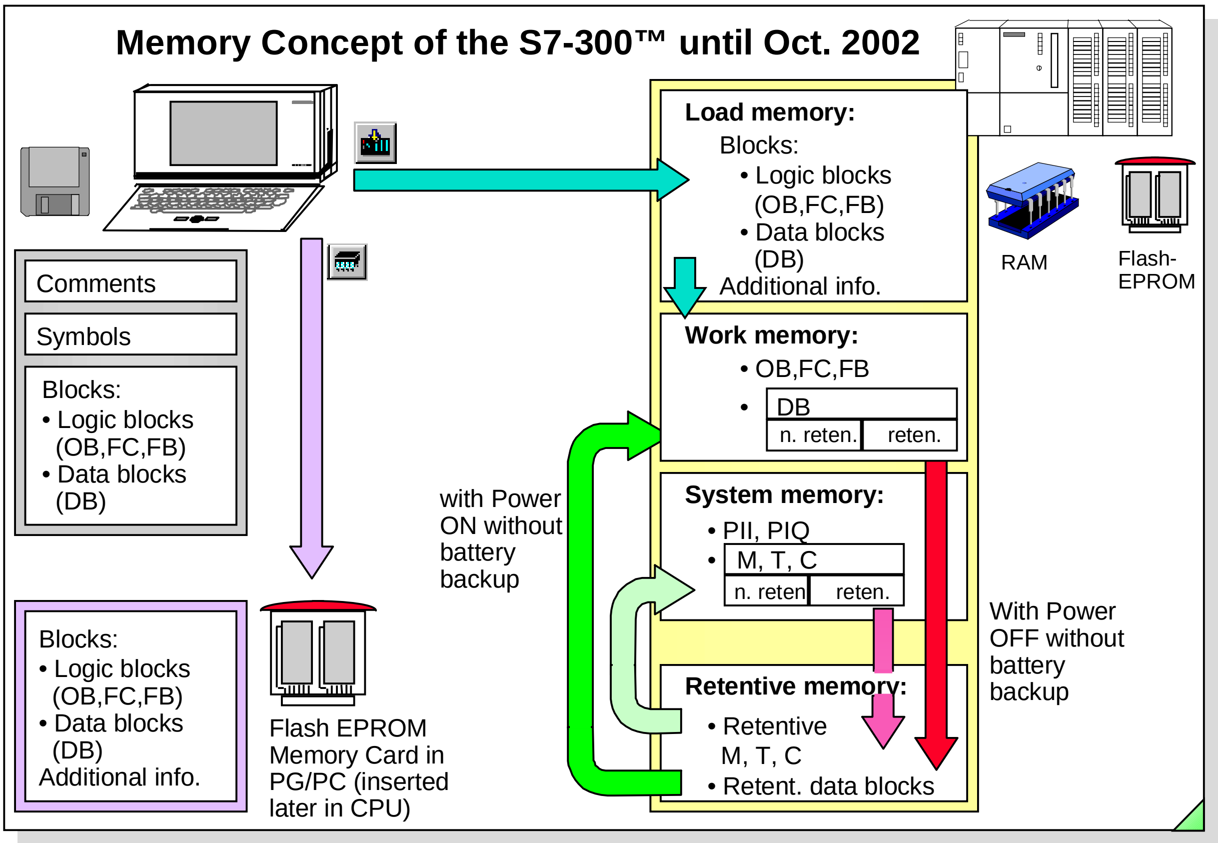
Note: The blocks are stored in the "Blocks" folder of the new S7 program on the hard disk of the PG.

**Uploading a Station**

You can also upload an entire station and its program to the PG. The advantage of this is that you can change the parameters of the hardware immediately.

What to do:

- Create a new project using the SIMATIC® Manager.
- Select the *PLC -> Upload Station* menu option.




SIMATIC® S7


Date: 12.03.03
File: PRO1_14E.11

SITRAIN Training for Automation and Drives

**Load Memory**

The load memory is the internal memory used by the CPU which contains logic blocks, data blocks, and additional information created on the programming device.

The load memory can either be a plug-in memory card (EPROM) or an integrated RAM.

**Work Memory**

The work memory (integrated RAM) contains the parts of the S7 program relevant for running your program.
The RAM work memory is integrated in the CPU and is backed up by the battery.

**System Memory**

The system memory contains the memory areas for:

| | |
|---|---|
| • Process image input and output tables | (PII, PIQ) |
| • Bit memories | (M) |
| • Timers | (T) |
| • Counters | (C) |
| • L stack | (L). |

**Retentive Memory**

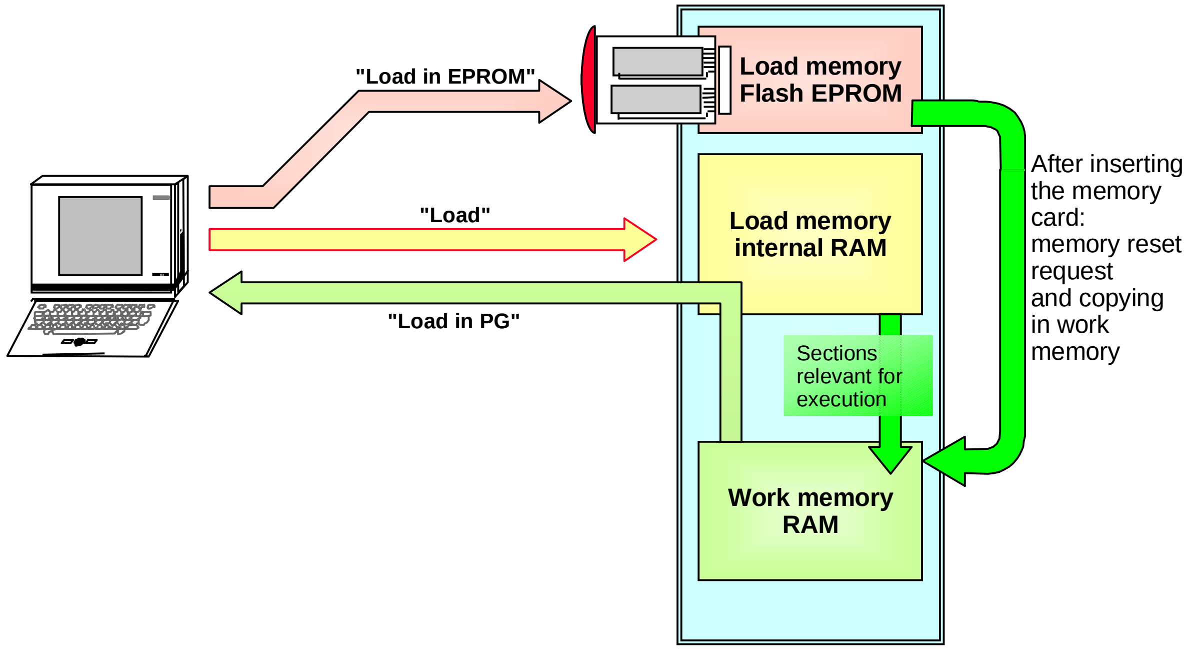
The retentive memory is a non-volatile RAM used for backing up bit memories, timers, counters and data blocks even if there is no backup battery. A Flash EPROM Memory Card must be used to do this. You specify the areas to be backed up when assigning the CPU’s object parameters.

**Inserting a Memory Card**


When you insert a memory card, the operating system requests a memory reset. (STOP LED flashes slowly). You perform the memory reset by turning the mode selector to the "MRES" position. The sections of the program relevant for execution are then transferred from the memory card (with load memory function) to the work memory.

You must leave the memory card inserted while the program is being executed.

## Loading Blocks into /out of Flash EPROM Memory Card




SIMATIC® S7


Date: 12.03.03
File: PRO1_14E.12

SITRAIN Training for Automation and Drives

**Introduction**

When you use a Flash EPROM card, it is possible to operate the CPU without battery backup. The program is stored in the Flash EPROM, making it power failure safe. You can define retentive areas in the HW Configuration.
In the S7-300™, the retentive data (timers, counters, bit memories, data areas) are stored in a retentive memory area of the CPU (non-volatile RAM).

**Insert / Remove**

When you remove or insert a memory card, the CPU requests a memory reset.
When you insert a RAM card, the user program must be reloaded from the PG.
When you insert a Flash EPROM card, its contents are copied into the work memory.

**Power Failure**

After a power failure without battery backup, the blocks are copied from the memory card into the work memory and with the S7-300™, the retentive data are supplied from the non-volatile RAM.
Data areas in DBs that were defined as retentive (only with the S7-300™), resume the state they had before power failure. Non-retentive data areas are set to the original values that are stored in the memory card.

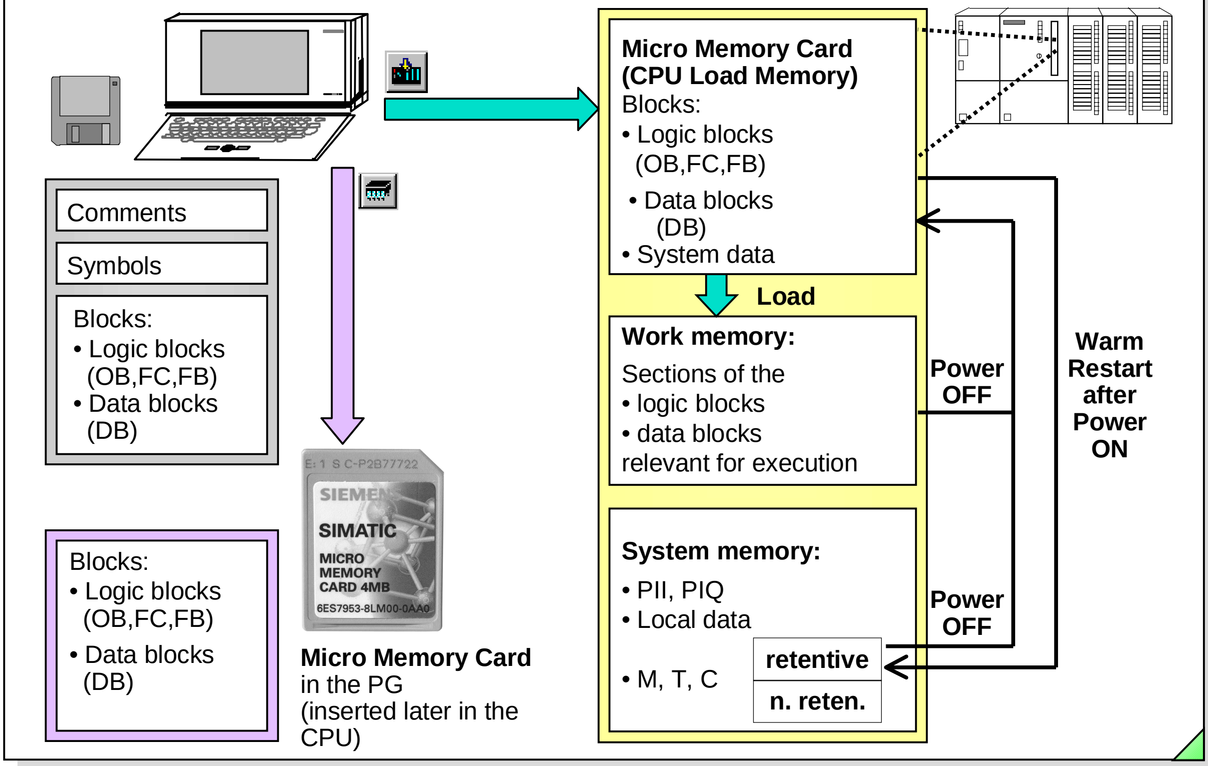
**Changing the Program**

When you make block corrections, the modified blocks are stored in the work memory. When you upload the blocks into the PG, these are retrieved from the work memory.
After a power failure (without battery), the work memory (RAM) is erased. So that the corrected blocks are available once more after a return of power, they have to be:
1. saved on the hard disk when you operate without Flash EPROM memory card,
2. saved on the hard disk or on a memory card when you operate with Flash EPROM memory card.

**Loading the Memory Card**

You either transfer the blocks onto the memory card (inserted in the PG) through the SIMATIC® Manager using drag and drop. Some CPUs allow you to write directly to the CPU using the *PLC -> Download user program to memory card* menu option.
The memory card must be erased first.
Individual blocks can be reloaded but cannot be deleted or overwritten.

## Memory Concept of the S7-300™ as of Oct. 2002




SIMATIC® S7


Date: 12.03.03
File: PRO1_14E.13

**SITRAIN** Training for Automation and Drives

**Load Memory Micro Memory Card (MMC)**

The Micro Memory Card (MMC) is used as the load memory of the CPU. It is used to store logic and data blocks as well as the system data (hardware configuration, communication connections etc.). The MMC contents are retentive.

If a block or the entire user program is downloaded into the CPU from the PG, the information is stored on the MMC. All block sections relevant for execution are automatically copied into the work memory (RAM).

**It is only possible to load a block or the user program as well as operate the CPU when the MMC is inserted!**

**A memory reset is required every time the MMC is pulled or inserted!**

**Work Memory**

The work memory (RAM) is integrated on the CPU and only contains the parts of the S7 program relevant for running your program (such as, only the current values of the data blocks, not the initial values).

**System Memory**

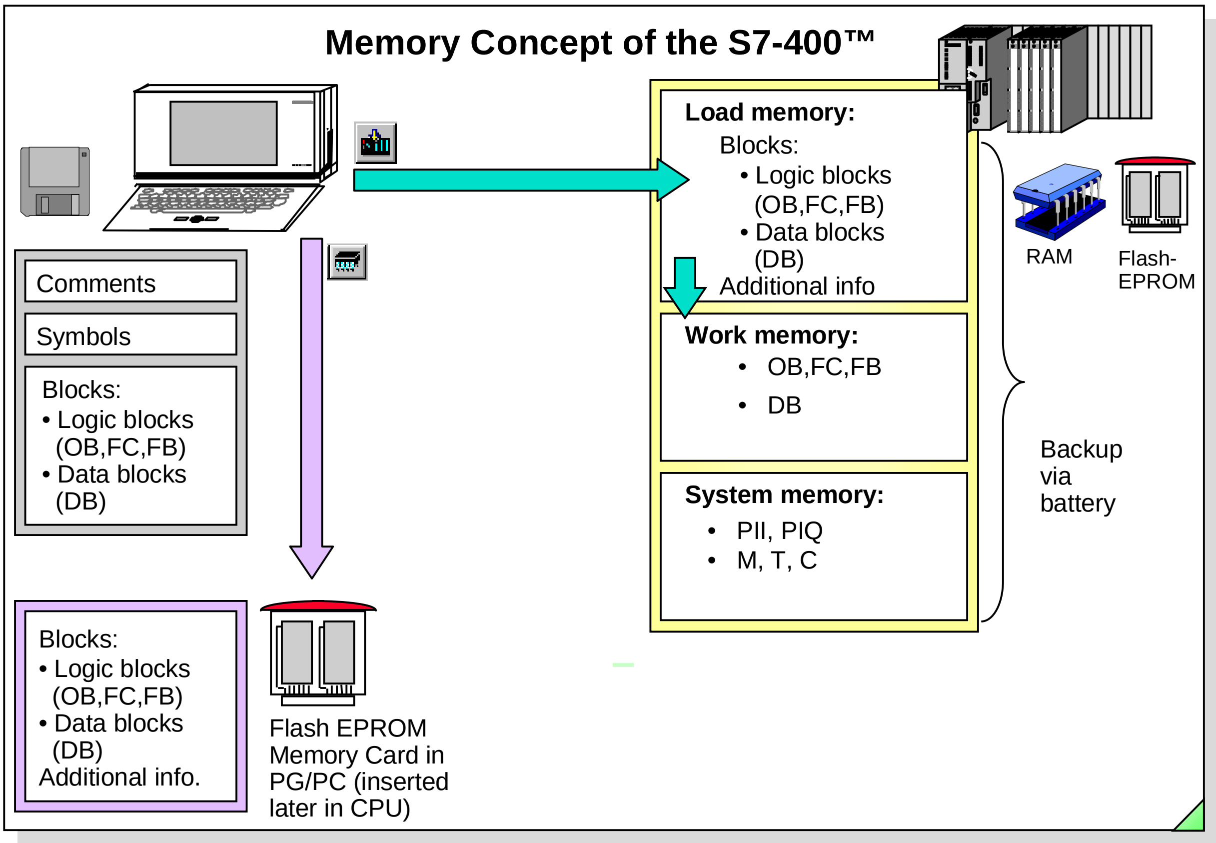
The system memory contains the memory areas for:

- Process image input and output tables (PII, PIQ)
- Bit memories (M)
- Timers (T)
- Counters (C)
- Local data (L)

**Retentive**

All data that are saved in a power failure and/or which don‘t lose their contents are considered retentive. This is all the work memory data as well as the bit memories, timers and counters declared as retentive in the hardware configuration.

Retentiveness is achieved in that the above-mentioned data are stored on the MMC in a power failure and are written back to the RAM after a warm restart when the power comes back on.




SIMATIC® S7


Date: 12.03.03
File: PRO1_14E.14

SITRAIN Training for Automation and Drives

**Load Memory**

The load memory can either be a plug-in memory card or integrated RAM. In the S7-400™, the memory card (RAM or Flash EPROM) expands the internal load memory. A memory card is always required for the S7-400™, since the internal load memory only has a limited size.

**Work Memory**

The work memory contains only the data relevant at runtime.
The RAM work memory is integrated in the CPU and is backed up by the battery.

**System Memory**

The system memory contains the memory areas for:

- Process image input and output tables (PII, PIQ)
- Bit memories (M)
- Timers (T)
- Counters (C)
- L stack (L).

**Memory Card**

When a RAM memory card is used, the system must be operated with a battey. The battery backs up the data on the memory card and any internal RAM in case of a power failure.

When a Flash EPROM memory card is used, the user program is stored in the memory card making it power failure safe. The data found in the internal RAM are backed up by the battery.

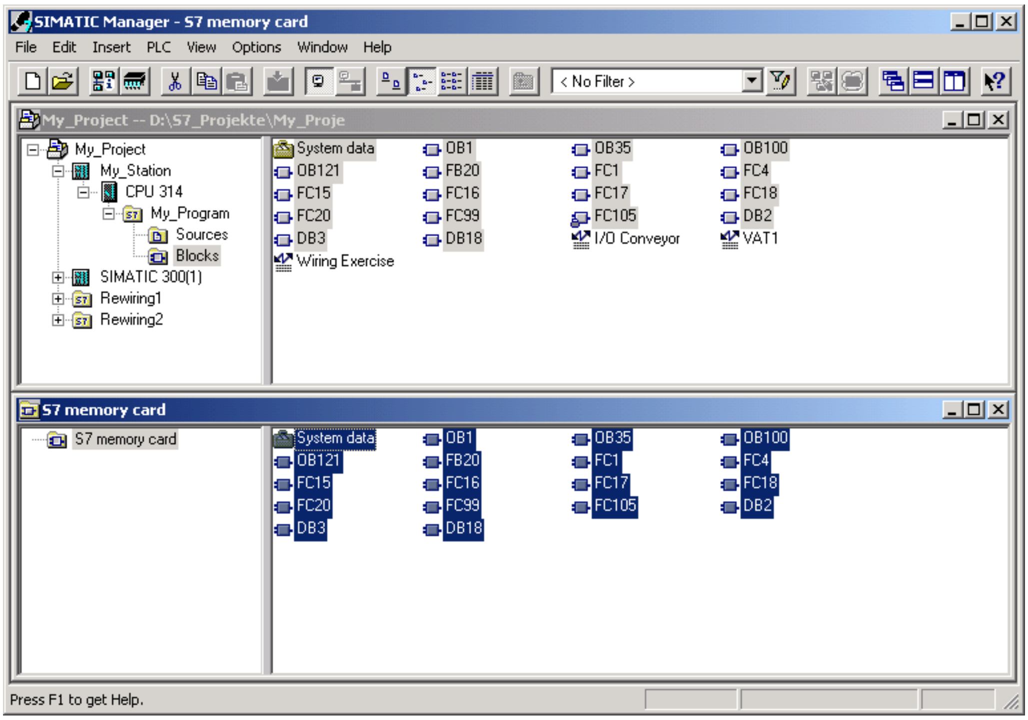
The "Restart" mode is possible only in a backed up system.

**Inserting a Memory Card**


When you insert a memory card, the operating system requests a memory reset (STOP LED flashes slowly). You perform the memory reset by turning the mode selector to the "MRES" position. The sections of the program relevant for execution are then transferred from the memory card (load memory) to the work memory.

You must leave the memory card inserted while the program is being executed.

# Copying a Program onto a Memory Card




SIMATIC® S7



Date: 12.03.03
File: PRO1_14E.15

**SITRAIN** Training for Automation and Drives

**Requirements**

The memory card driver must be installed in the STEP 7 software. If not, click the *"Start"* button and select *Simatic® -> STEP 7 -> Memory Card Parameter Assignment* and install the driver. A "Memory Card" icon will appear in the toolbar of the SIMATIC® Manager.

Remember: the memory card must be erased before you can copy your program onto it.

Select: *File -> S7 Memory Card -> Delete…*

Next: open two windows in the SIMATIC® Manager:

- One containing the user program you want to save
- The other with the memory card (*File -> S7 Memory Card -> Open*)

**Delete**

You can only completely erase the memory card. It is not possible to delete or overwrite individual blocks.

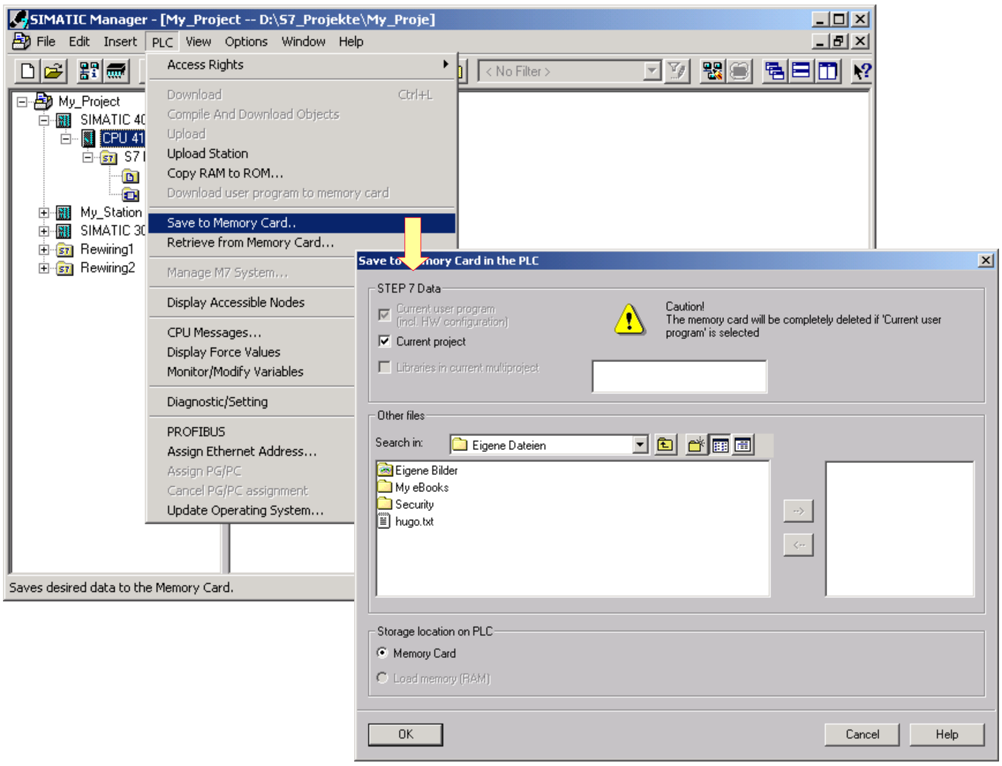
**Copying**

Select the "Blocks" folder (for all blocks) or select individual blocks from the "Blocks" folder and drag them into the Memory Card window with the mouse.

**Note**

With certain CPUs (such as CPU 416), you can also write the memory card in the CPU. To do so, use the *PLC -> Download user program to memory card* menu option.

# Saving a Project on a Memory Card




SIMATIC® S7



Date: 12.03.03
File: PRO1_14E.16

SITRAIN Training for Automation and Drives

**Function**

With the functions *"Save to Memory Card..."* and *"Retrieve from Memory Card...",* you can save and retrieve the complete data of a project (user programs with all comments, symbol tables, hardware configurations etc. from all hardware stations) on a memory card. The memory card can be located in a CPU or in the memory card programming slot of a PG or PC.

The project data are compressed before they are saved on the memory card and are extracted when retrieved. The size of the project data to be saved corresponds to the archive file size of the project. If the memory capacity of the memory card is not sufficient a message will appear indicating so.

**Project Data with / without User Program**

The project data contains - just like the archive of a project - basically all data belonging to the project and all user programs of the CPUs.
The user programs contained in the project data can not be read by the CPUs and thus cannot be executed. With the option *"Load the user program also",* the executable user program is also stored in addition to the project data. This user program is the one assigned to the CPU on which the memory card is inserted.

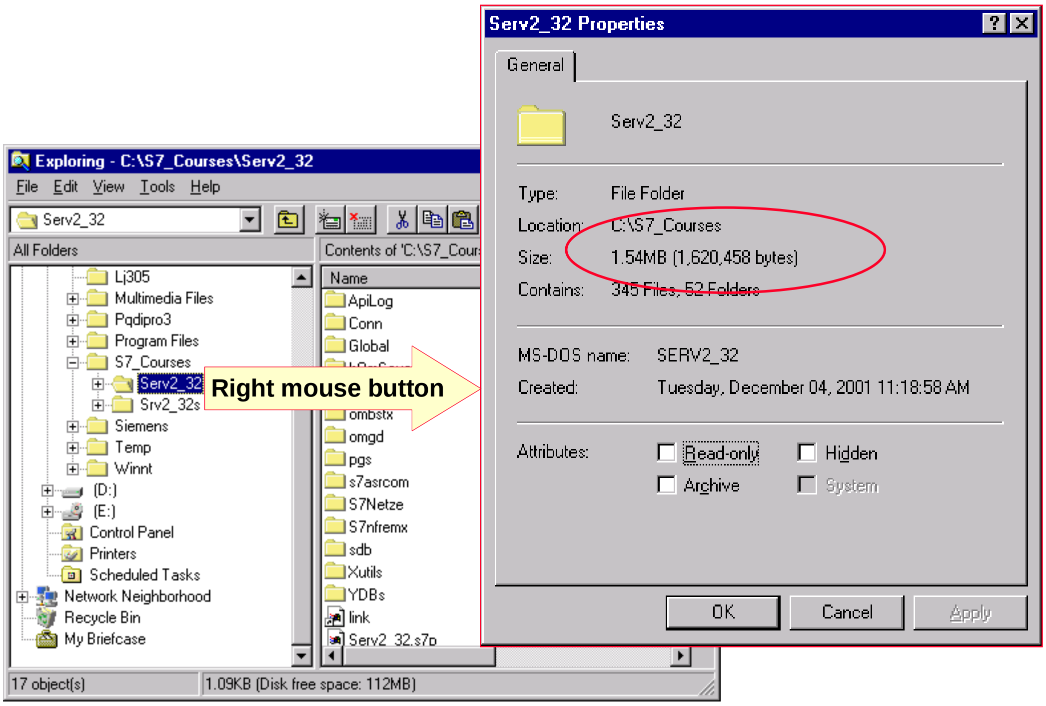
**Area of Use**

If several co-workers in the service and maintenance area have the job of maintaining the SIMATIC® S7 PLC, it is difficult to quickly provide every worker with the current project data for a service assignment. When the project data are available locally in one of the CPUs to be maintained, every worker can access the current project data and make changes, if necessary, which in turn are current and available to all other workers.

**Note**

The functions *Save to Memory Card* and *Retrieve from Memory Card* are currently only possible with the S7-400™ system. They are being developed for the S7-300™ system.

## Determining the Size of a Project




SIMATIC® S7


Date: 12.03.03
File: PRO1_14E.17

SITRAIN Training for Automation and Drives

**Introduction**

If a project needs more than 1.44 MB of memory, you can still save it on diskette by archiving (compressing) it first.

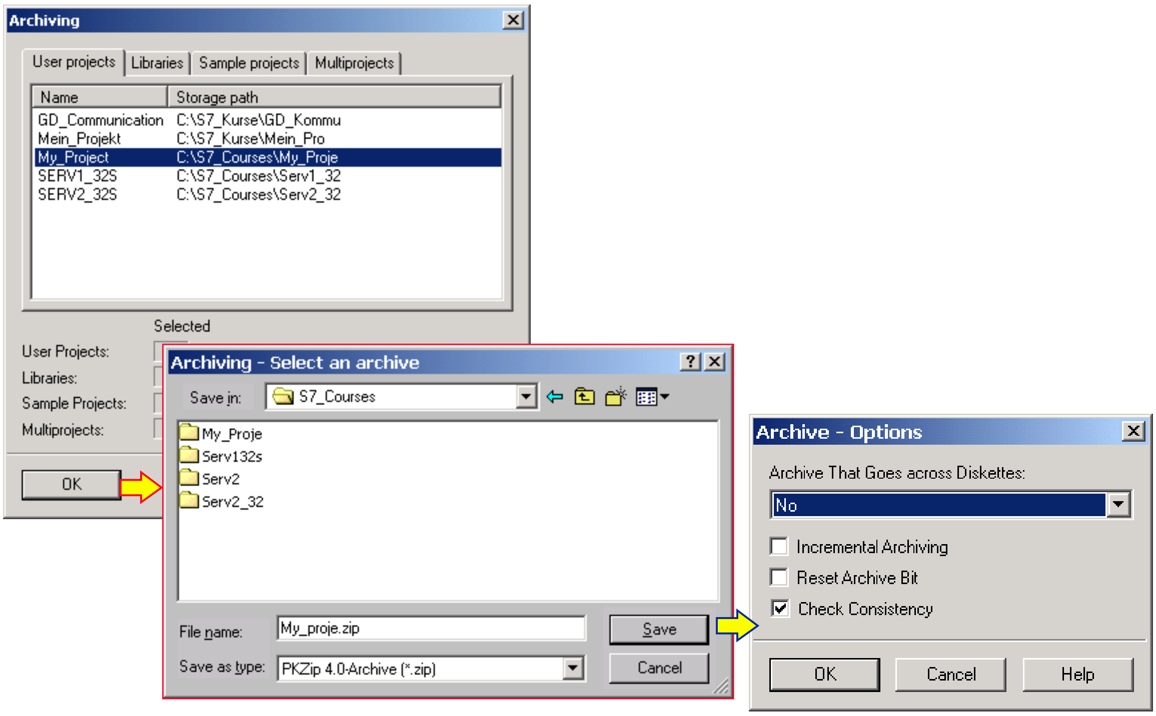
**Explorer**

You can find out the size of a project in the Explorer by:

- right clicking on the project folder and choosing "Properties" or
- selecting the project folder and then choosing the *File -> Properties* menu option.

In both cases the "Properties" window opens.

## Archiving on Diskette




SIMATIC® S7


Date: 12.03.03
File: PRO1_14E.18

**SITRAIN** Training for Automation and Drives

**Introduction**

Since the data in a project can take up a lot of memory space and might not fit onto a diskette, an archive function is provided.
This archive function compresses the data so that it only takes up approximately 1/8 of its original amount of memory. It uses the normal file compression utilities, such as PKZIP, ARJ, LHARC, RAR or WINZIP. One of these programs must first be installed on the PG/PC. If you want to use long file names for the projects, you will need PKZIP, WinZip or RAR.
The ARJ and PKZIP file compression utilities are supplied with STEP 7.

You set the path for the archive program by selecting the *Options -> Customize -> Archive* menu options in the SIMATIC® Manager.

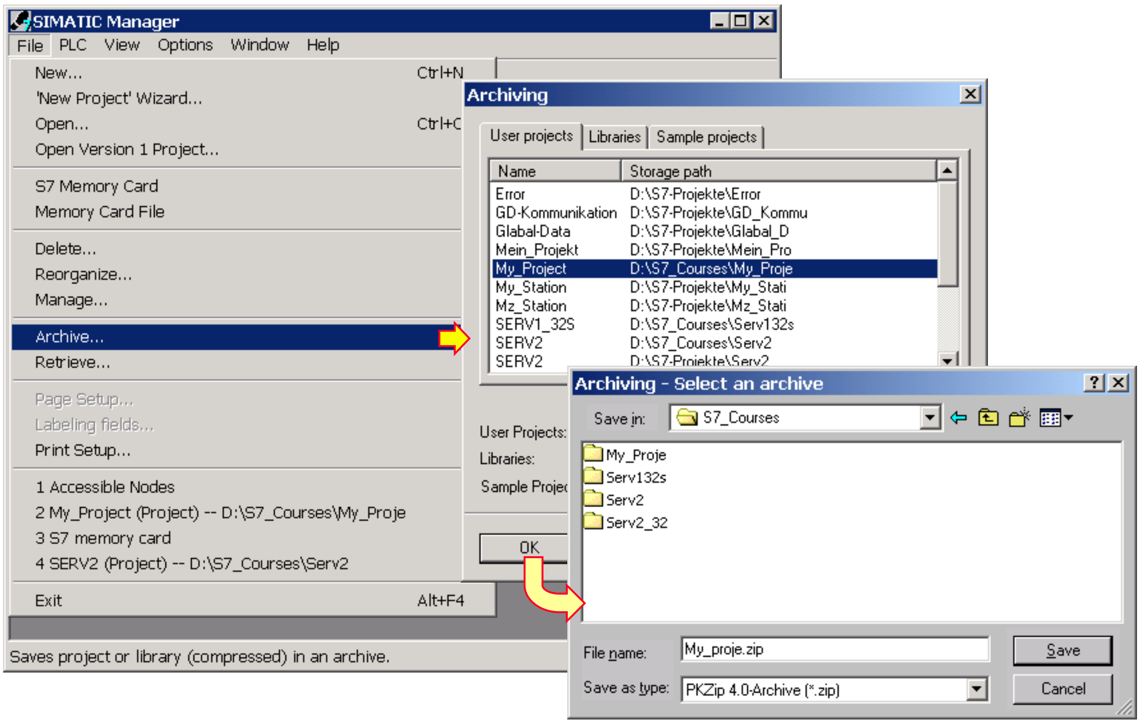
**Archiving**

- The project to be archived must be closed in the SIMATIC® Manager.
- Select the *File --> Archive...* menu option.
- Select the project to be archived in the dialog window and acknowledge "OK".
- Select the "Save in:" path and "File name" in the next dialog box and "Save".
- In the last dialog box, Archive - Options, you can choose between the following options:
  - Archive That Goes across Diskettes = Split the archive file onto several diskettes or not
  - Incremental Archiving = Only the files with the ACR attribute (STEP7 files) are archived.
  - Reset Archive Bit = Archive only the files that have been changed since the last archive.
  - Check Consistency = Compare the files to be archived (only for ARJ)

**Retrieving**

- Select the *File -> Retrieve...* menu option.
- Select the "Look in: " location where the archived file is stored and select the "File name" of the archived file and "Open".
- In the next dialog box, select the destination directory and "OK".
- Use the last dialog box to select options for overwriting and restoring the storage path.

## Exercise: Archiving a Project




SIMATIC® S7


Date: 12.03.03
File: PRO1_14E.19

SITRAIN Training for Automation and Drives

**Task**

You are to archive your project called "My_Project" so that you can then save it on a diskette.

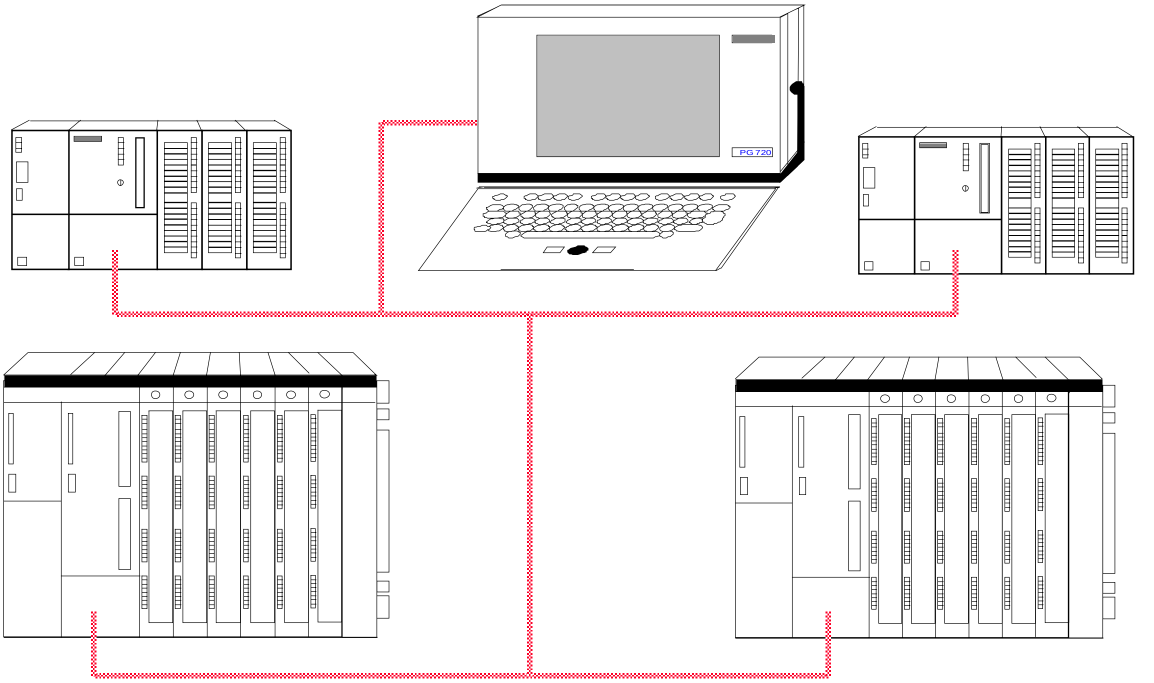
**Note**

Only closed projects can be archived. Before you start the archiving function, you must make sure that neither the SIMATIC® Manager nor any other applications (such as LAD/FBD/STL Editor, Symbol Editor, HW Config) are accessing the project to be archived.

**What to Do**

1. Close the project that you want to archive
2. Start the archive function from the SIMATIC® Manager
   *File -> Archive…* From the "User projects" tab, select the *Project* and acknowledge "OK"
3. In the following dialogs, select the project to be archived as well as the File name, the Save in and the Save as type (*.zip, *.arj, etc.) settings if the archive file
4. Using the Windows Explorer™, check the success of your archive and compare the size (memory requirement) of the original project with that of the archive created
   *Windows Explorer -> right mouse click on Archive or Project -> Properties -> Size*
5. Option:
   Copy the project archive onto a diskette

# Communication with MPI




SIMATIC® S7



Date: 12.03.03
File: PRO1_15E.1

SITRAIN Training for Automation and Drives

## Contents

Page

## Objectives

**Upon completion of this chapter the participant will ...**

... know the subnets of the SIMATIC® world

... be familiar with the S7 communication methods

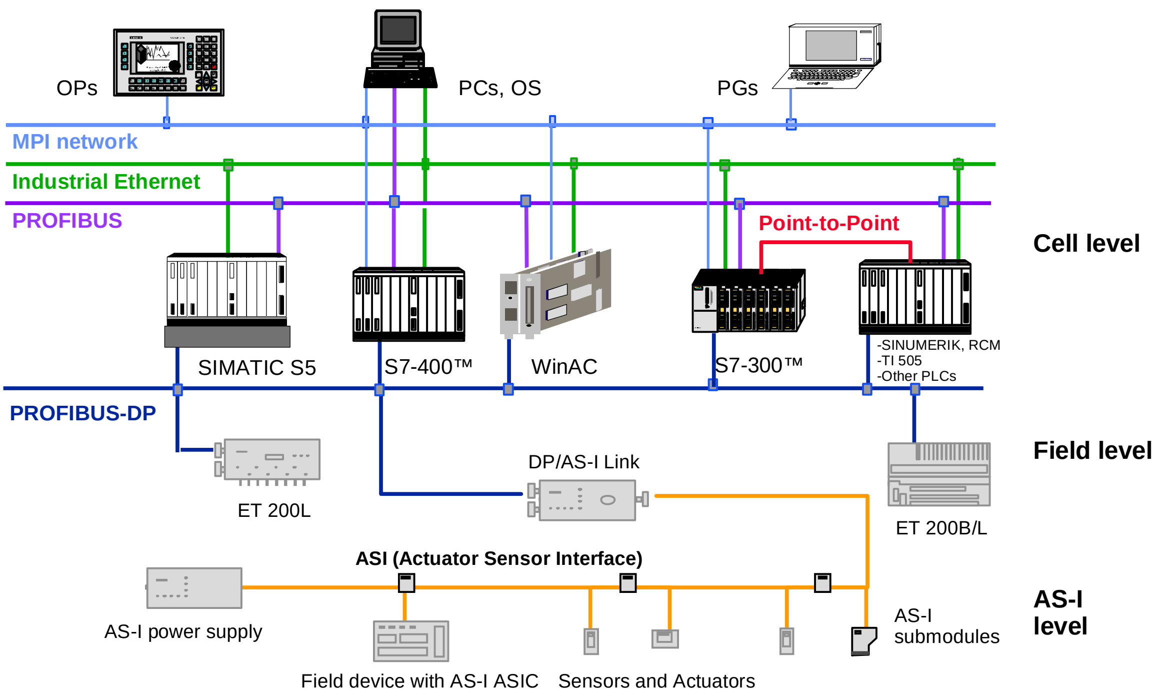
... be able to configure a global data communication

SIMATIC® S7


Date: 12.03.03
File: PRO1_15E.2

SITRAIN Training for Automation and Drives

## Subnets in SIMATIC




SIMATIC® S7


Date: 12.03.03
File: PRO1_15E.3

SITRAIN Training for Automation and Drives

**Overview**
To meet the different communication requirements at cell level (non-time-critical) and field level (time-critical) SIEMENS offers the following subnets.

**MPI**
The MPI subnet is designed for use at the cell level. MPI is the multipoint interface in SIMATIC® S7, and C7.

The MPI is basically a PG interface, that is, it is designed for the connection of PGs (for startup and testing) and OPs (human-machine interface). The MPI subnet can, however, also be used for networking a small number of CPUs.

**Industrial Ethernet**
Industrial Ethernet is the network for the plant management and cell levels in the SIMATIC® open, manufacturer-independent communication system.

Industrial Ethernet is designed for non-time-critical transmission of large quantities of data and uses gateways to provide facilities for connection to remote networks.

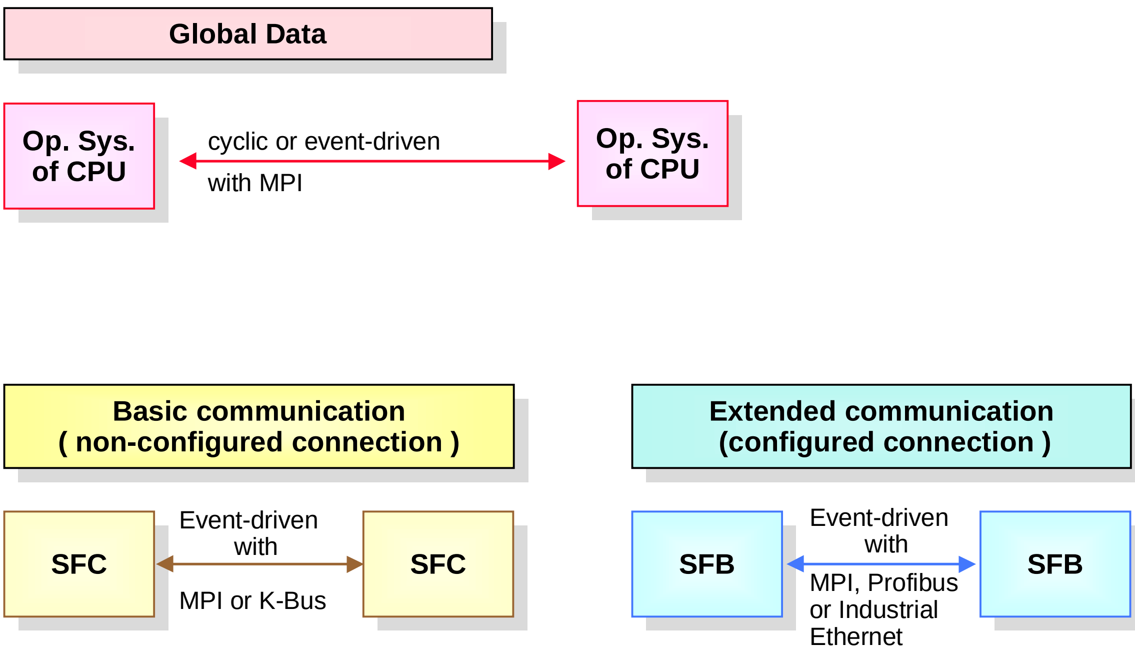
**PROFIBUS**
PROFIBUS is the network for the cell and field levels in the SIMATIC® open, manufacturer-independent communication system. There are two versions:

- PROFIBUS is for non-time-critical communication between equal, intelligent nodes at cell level.
- PROFIBUS DP is the fieldbus for time-critical, cyclic data exchange between intelligent masters and field devices.

**Point-to-Point Connection**
Point-to-point connections are primarily used for non-time-critical data exchange between two stations or for connecting devices such as OPs, printers, bar code scanners, magnetic stripe ID card readers, etc. to a station.

**AS Interface**
The Actuator-Sensor-Interface is a subnet for the lowest process level in an automation system. The AS Interface enables binary sensors and actuators to be networked.

## S7 Communication Methods






Date: 12.03.03
File: PRO1_15E.4



**Global Data**

This communication method enables data to be exchanged between CPUs cyclically with the MPI interface without programming. Data is exchanged at the scan cycle checkpoint when the process image is updated. On the S7-400™, data exchange can also be initiated using SFCs.
Global data can be inputs, outputs, bit memories, timers, counters and data block areas.
Data communication is not programmed, but configured by means of a global data table. None of the connections on the CPU need to be used for global data communication.

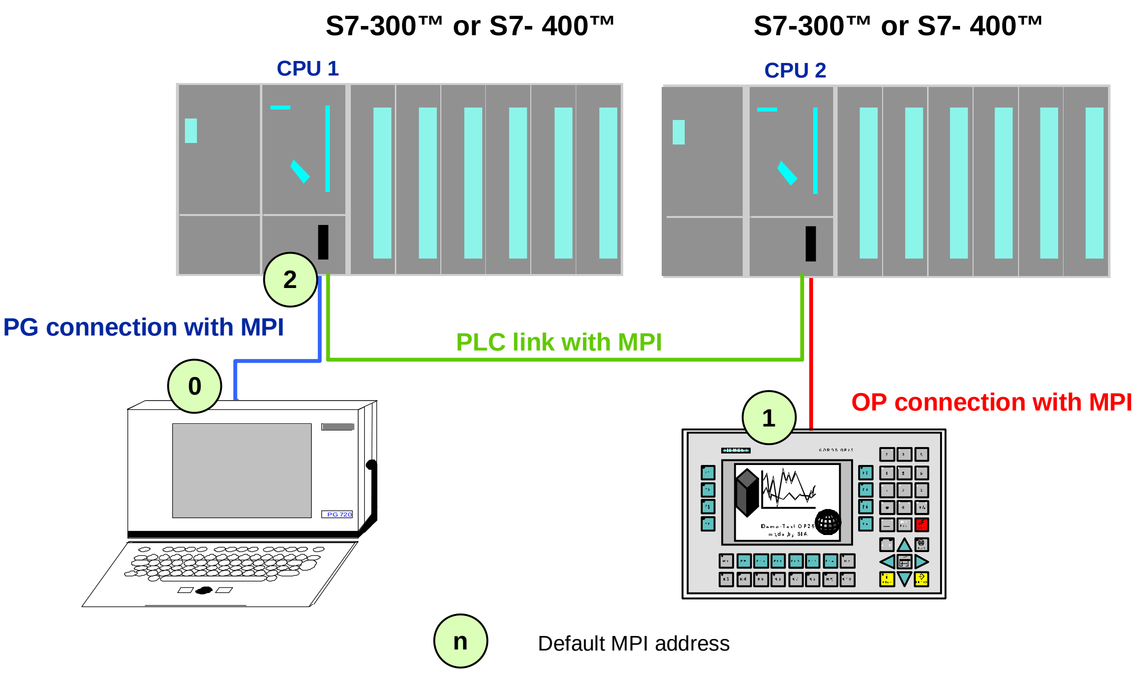
**Basic Communication**

This communication method can be used with all S7-300™/400™ CPUs for transmitting data with the MPI subnet or within a station on its K bus. System functions (SFCs), such as X_SEND at the Send end and X_RCV at the Receive end, are called in the user program.
The maximum amount of user data is 76 bytes.
When the system function is called, a connection to the communication partner is established and cleared dynamically. One free connection is required on the CPU.

**Extended Communication**

You can use this communication method with all S7-400™ CPUs. Up to 64KBytes of data can be transmitted with any subnet (MPI, Profibus, Industrial Ethernet). This transmission is done with system functions (SFBs), which also allow communication with acknowledgement. Data can also be read from or written to an S7-300™ (PUT/GET blocks).
You can not only transfer data, but also perform control functions, like Stop or Start, on the communication partner. Configured connections (connection table) are required for communication by this method. These connections are established on a complete restart of the station and usually remain in force. Free connections on the CPU are necessary for this.

## Networking with MPI




SIMATIC® S7


Date: 12.03.03
File: PRO1_15E.5

SITRAIN Training for Automation and Drives

**Introduction**

Every programming device has an MPI interface.
The MPI interface of the CPU enables all intelligent modules in a PLC to be accessed, for example, the function modules of a station.

Each MPI node needs its own MPI address (between 0 and 126, the default settings are PG=0, OP/TD=1, and CPUs=2).

In the S7-300™, the MPI bus is looped through on the K bus on a one-to-one basis. This means that every node on the K bus (FMs and CPs) in the S7-300™ rack is also an MPI node and needs to have its own MPI address.

In the S7-400™, communication frames are converted for the internal K bus (10.5 Mbps) via the MPI (187.5 Kbps). In an S7-400™ rack, only the CPU has its own MPI address. The other intelligent modules, such as FMs and CPs, do not have a separate MPI number.

**Connection Facilities**

The main advantage is that several devices can establish a communication link with the CPU at the same time.
This means, for example, that a programming device, an HMI device and a link with another PLC can be in operation at the same time.

The MPI interface also makes it possible to create a network in which a network administrator has central access with a PG to all the intelligent modules in the stations connected.

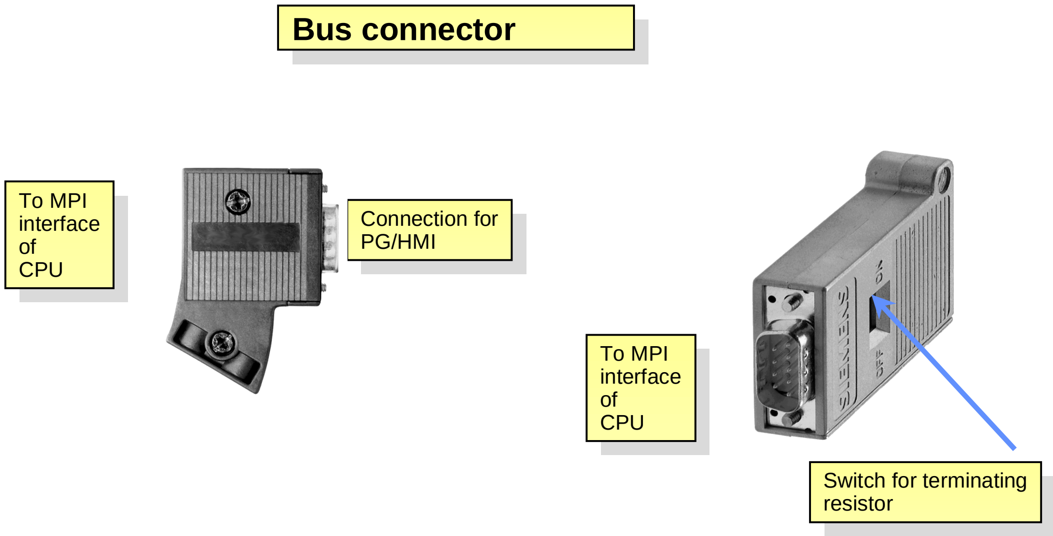
The number of channels for connection to other communication partners that can be used at the same time depends on the type of CPU. For example, the CPU 314 has four connection resources and the CPU 416 has sixty-four.

**Features**

Main features of the MPI interface:

- RS 485 physics
- Transmission rate 19.2 Kbps or 187.5 Kbps or 1.5 Mbps
- Distances up to 50 m (between 2 neighboring nodes) and with two repeaters, 1100 m and 23.8 km with optical fiber and star coupler.
- Profibus components (cables, connectors)

## Connection Options to MPI








**Connectors**

Two types of connector are available for installing an MPI bus system.

The connector with PG socket shown on the left is the standard connector used for linking MPI nodes with one another, while also enabling a PG to be connected at the same time.

The connector without PG socket shown on the right is used where facilities for connecting a PG are not necessary.

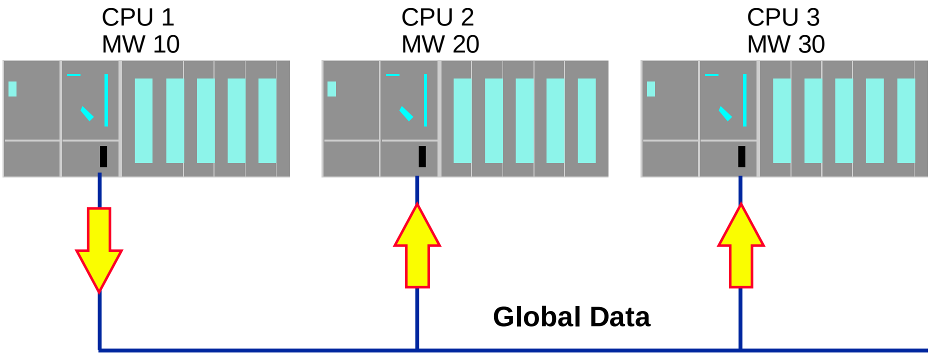
On the last bus node, you must replace the outgoing bus cable by a terminating resistor.

**Requirements**

To connect a programming device/PC to the MPI interface of the PLC, you need:

- an MPI module in the PG/PC and a connecting cable
- a PC adapter (a connecting cable with integral MPI converter, if there is no free slot in the PG/PC ). The PC adapter has the following specifications:
  - Length 5 m
  - Transmission rate up to adapter 187.5 Kbps
    Adapter to PG 19.2 or 38.4 Kbps (adjustable)

## Global Data: Overview




SIMATIC® S7


Date: 12.03.03
File: PRO1_15E.7

SITRAIN Training for Automation and Drives

**Global Network Data**

In SIMATIC® S7, global data communication allows you to establish communication between distributed PLCs without having to write a single extra line in your user program.

Communication using global data is not programmed but configured. The configuration for data exchange is stored in a table.

Global data communication can take place between up to 15 CPUs within a project. Global data communication is designed for small quantities of data which are normally transmitted cyclically.

The S7-400™ CPUs also allow program-controlled and therefore also event-driven data transfer.

**Configuring**

You configure data communication with the "Defining Global Data" tool.

First of all, you open the Global Data Table and assign the columns of the table to the CPUs that are going to exchange data.

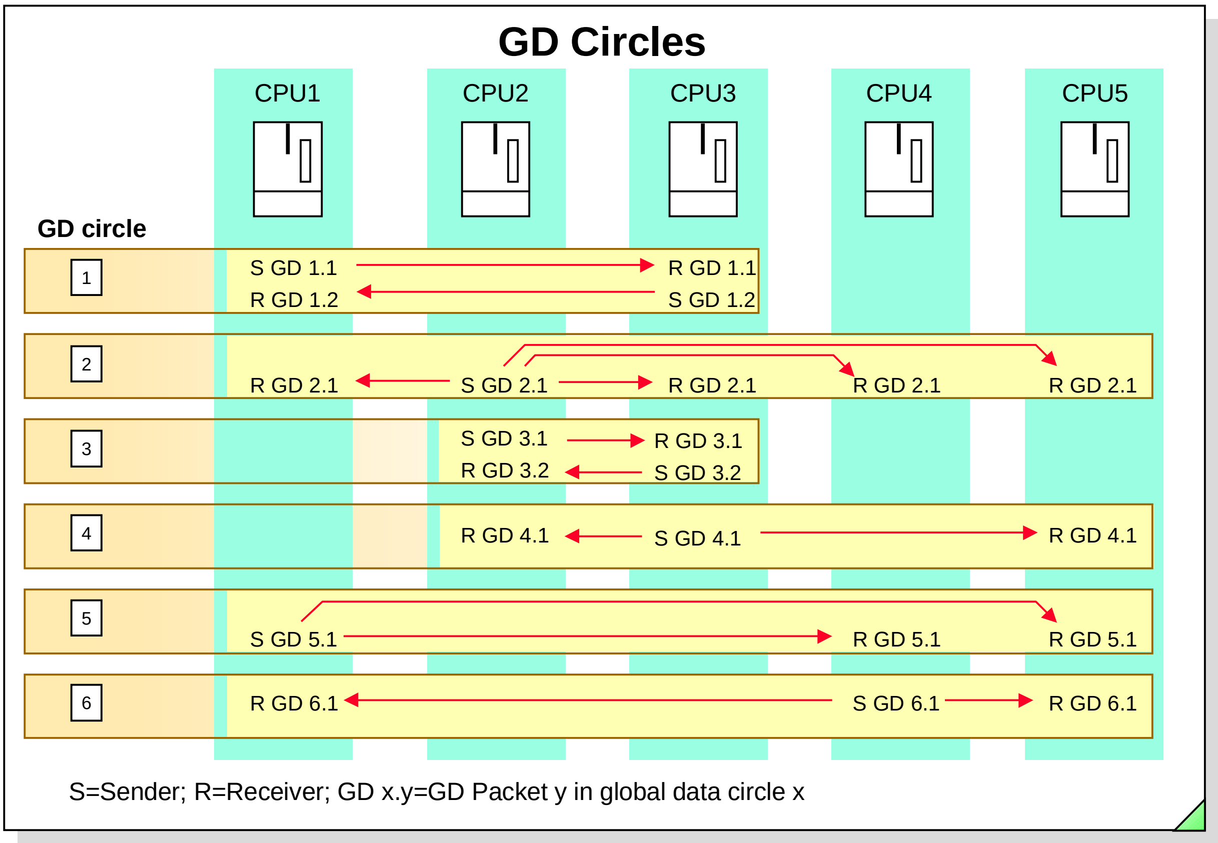
In the lines of the table you then define the variables to be exchanged. Almost all CPU address areas (apart from external inputs and outputs and temporary data) can be used as variables, such as bit memories, inputs, outputs, timers, counters and areas in data blocks.

**GD Packet**

Global data, that is, variables with the same sender/receiver, can be collected in a GD (Global Data) packet and sent together. Each GD packet is identified by a GD packet number. The variables within a packet are identified by variable numbers.

**GD Circle**

The CPUs participating in the exchange of GD packets make up a GD circle. Each GD circle is identified by a GD circle number.




SIMATIC® S7


Date: 12.03.03
File: PRO1_15E.8

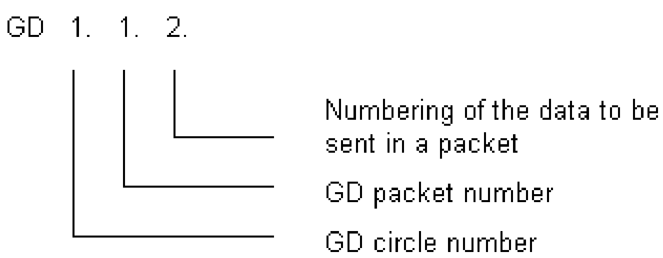
SITRAIN Training for Automation and Drives

**What is a GD Circle?** A GD circle is a fixed distribution list for GD packets. Each CPU in a global data circle can send data to the other CPUs or receive data from another CPU. Types of GD circle:

- Global data circle with more than two CPUs. One CPU is then the sender of a data packet and all the other CPUs in the GD circle are receivers.
- Global data circle with two CPUs. Each CPU can both send a data packet to the other CPU and receive a data packet from the other CPU.

**Number of GD Circles** Each CPU of an S7-300™ can be in up to four different GD circles. Up to 15 CPUs can exchange data via GD communication in one MPI network.

**Example of a GD Circle** The diagram above shows an example to illustrate the principle of communication in GD circles. Below is an example of the numbering of a GD circle.



## Global Data: Configuration Procedure

- Create hardware stations in a project
  - with the "SIMATIC® Manager"
- Create and download configuration data (MPI address) for the individual CPUs
  - with the "HW Config" tool
- Configure Global Data table
  - with the "Defining Global Data" tool

SIMATIC® S7


Date: 12.03.03
File: PRO1_15E.9

SITRAIN Training for Automation and Drives

**Creating Hardware Stations** First of all you must create the stations that you want to network in a project using the SIMATIC® Manager. When you have done this, open the HW Config tool and open the stations one after the other.

**Setting the MPI Address** When configuring the hardware, you must explicitly define the CPUs to be networked with an MPI as "Networked" and assign each of them their own MPI node address.

Save your CPU parameters on the hard disk and then download the configuration data to each CPU separately (point-to-point) ("*PLC -> Download*").

**Networking** You then link up the MPI nodes with Profibus cables. When you have done this, it should be possible to establish an online connection to all the CPUs. You can test this with the "Accessible Nodes" function in the SIMATIC® Manager.

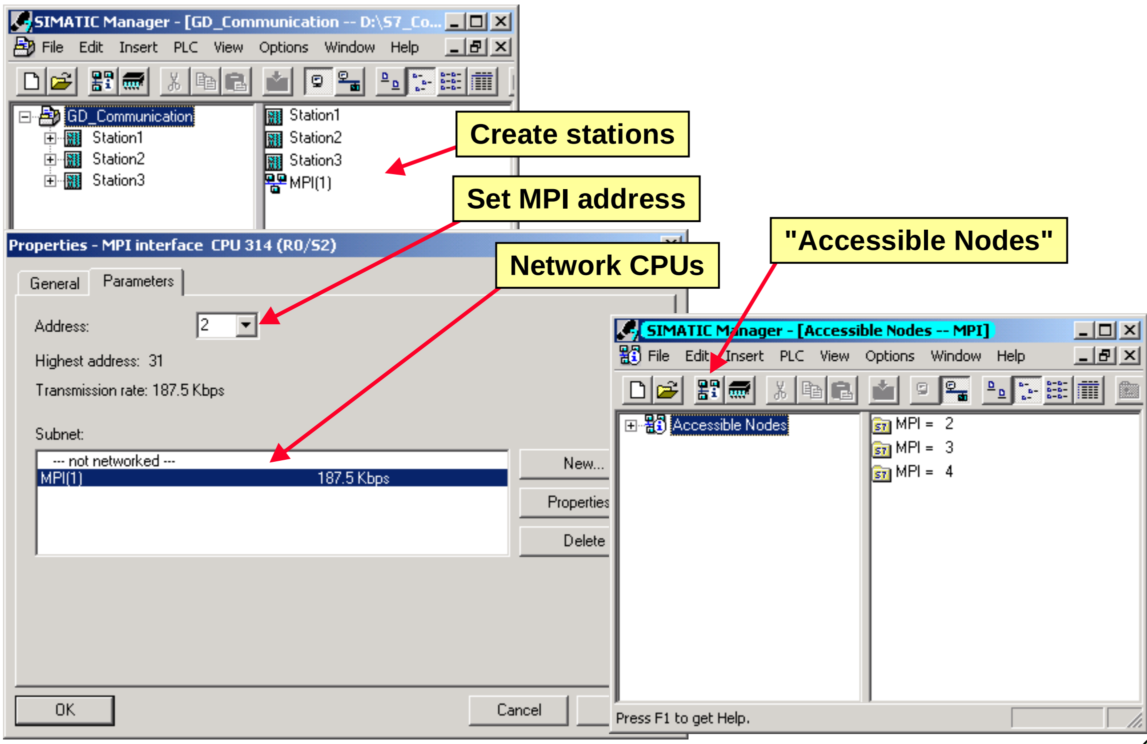
**Creating the GD Table** You use the "Defining Global Data" tool to create a global data table in which you define the data to be exchanged. You then compile the table twice and download the relevant configuration data to the CPUs.

**Volume of Data**

S7-300™: One CPU can be in up to four GD circles.
A CPU can send one packet and receive one packet maximum per GD circle.
A maximum of 22 data bytes can be transferred with one packet.

S7-400™: One CPU can be in up to 16 GD circles.
A CPU can send one packet and receive two packets maximum per GD circle.
A maximum of 54 data bytes can be transferred with one packet.

## Global Data: Configuring the Hardware




SIMATIC® S7


Date: 12.03.03
File: PRO1_15E.10

SITRAIN Training for Automation and Drives

**What To Do**

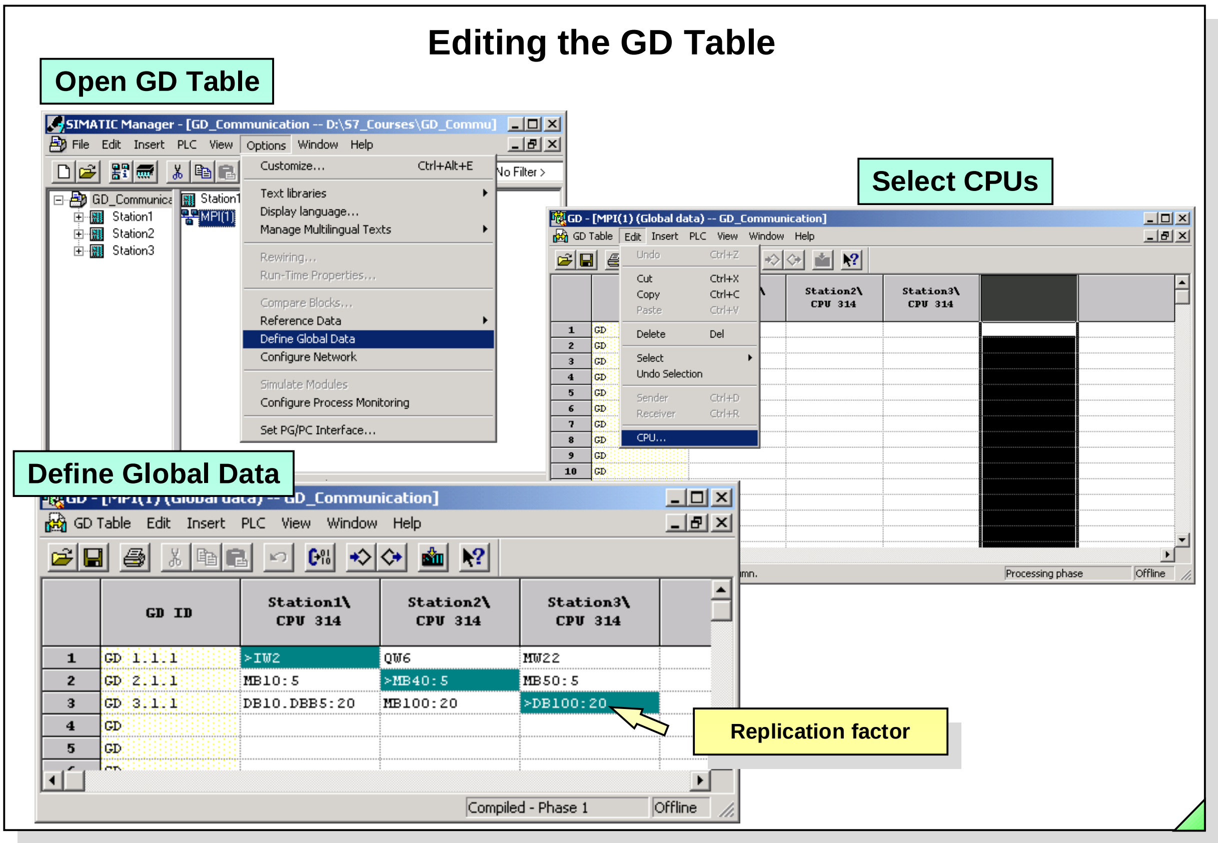
To configure the hardware for global data communication you must carry out the following steps:

1. A STEP 7 project must already have been created with the SIMATIC® Manager.
2. An MPI network object must be created in this project and assigned parameters. An MPI network object is automatically created when you create a new S7 project.
3. Configure at least two GD-capable modules in the project (such as S7 CPUs).

   When configuring the CPUs with the "HW Config" tool, explicitly define each CPU as "Networked" (see above) and assign it its own MPI address.
4. Download the configuration data you have entered to each CPU separately.
5. Physically link up the CPU modules with network cables.
6. Use the SIMATIC® Manager "Accessible Nodes" function to check that you have networked the stations correctly

**MPI Address of PG**

If several PGs are to be connected to the MPI network, then each PG must be given its own MPI address. Use the "Simatic® -> STEP 7 -> Setting the PG/PC Interface" program to set the address.




SIMATIC® S7


Date: 12.03.03
File: PRO1_15E.11

SITRAIN Training for Automation and Drives

**Overview**

The GD table is where you enter the CPUs that are going to exchange data and the address areas of the data to be exchanged.
You can also specify the scan rate and a doubleword for the status information.

**Opening the GD Table**

Open the GD table as follows:
1. Open your project and select the MPI network object.
2. Select the *Options -> Define Global Data* menu option. A new GD table is then generated, or an existing GD table is opened.

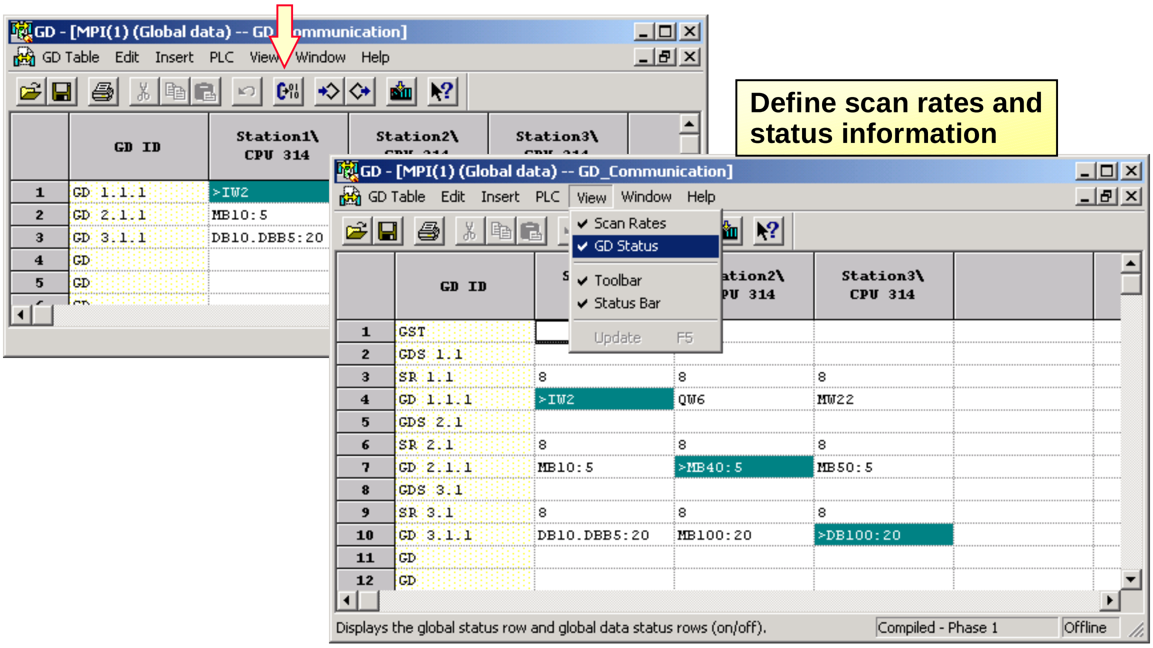
**Filling in the GD Table**

You must enter the address areas to be used in a separate column for each CPU taking part in GD communication. You do this as follows:

1. First assign each column of the table to a CPU by clicking the column header with the mouse to select it. Choose the *Edit -> CPU* menu option.
2. Select the CPU you want in the dialog box that appears and confirm with "OK".
3. Enter the global data to be transferred in the lines beneath. You can select Edit mode for the individual cells of the table with the F2 key.

   You can enter a replication factor for the variables to specify transfer of a whole section of data. In the example above: 20 bytes starting from DBB0 of DB100 (Station_3).
4. Define a sender in each line of the GD table by selecting the relevant cell. Click the icon for "Select as Sender" in the toolbar.

## Compiling the GD Table




SIMATIC® S7


Date: 12.03.03
File: PRO1_15E.12

SITRAIN Training for Automation and Drives

**Compiling the GD Table**

You can now compile configuration data from the information you have entered in the GD table. The configuration data is generated in two phases:

- Start the first compilation by selecting the *GD Table -> Compile* menu option. The first time you compile the GD table, the individual variables are put into packets and the relevant GD circles are created.

  The relevant GD circle number, packet number and variable number are displayed in the first column:

  GD 1.1.1 1st variable in the 1st packet of the 1st GD circle
  GD 1.2.1 1st variable in the 2nd packet of the 1st GD circle
  :
  GD m.3.n nth variable in the 3rd packet of the mth GD circle

- After the first compilation, that is, when the GD circles and packets have been created, you can define different scan rates or variables for storing status information for the individual packets.
- You must then start the compiler again to include the information about the scan rates and storage of the status information in the configuration data.

**Scan Rates**

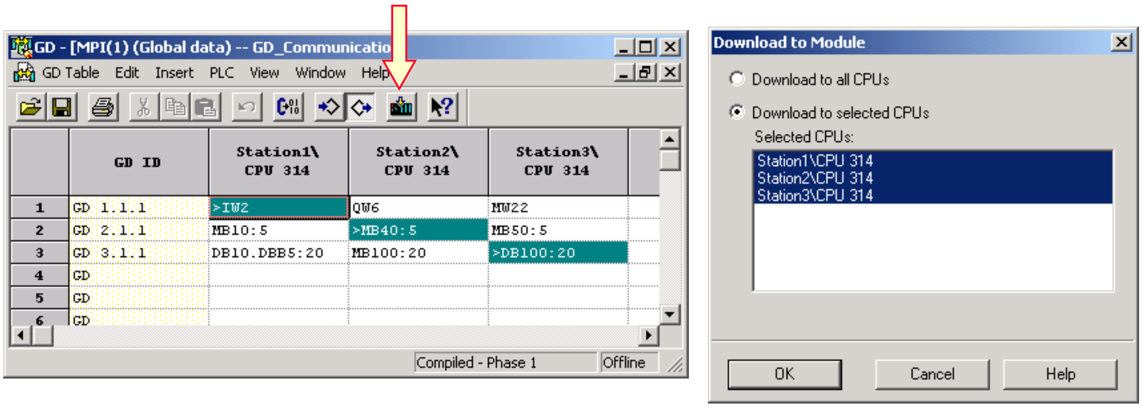
You can use the *View -> Scan Rates* menu option to select a different value (from 1 to 255 for the sender and 1 to 255 for the receiver, 0 for purely event-driven send and receive communication on the S7-400™).

**Status**

If you want to be notified whether the data has been transferred with or without errors, you can specify a doubleword for the status information for each data packet by selecting the *View -> GD Status* menu option*.* The CPU's operating system will then enter checkback information in this doubleword.

## Downloading GD Configuration Data

**Download GD configuration data**








### Downloading the GD Table

When you have compiled the configuration data for the second time, you can download it to the CPUs as follows :

1. Switch all the CPUs involved to the STOP mode.
2. Select the *PLC -> Download to Module* menu option to transfer the data.
3. When you have successfully downloaded the configuration data, switch the CPUs involved back to RUN mode.

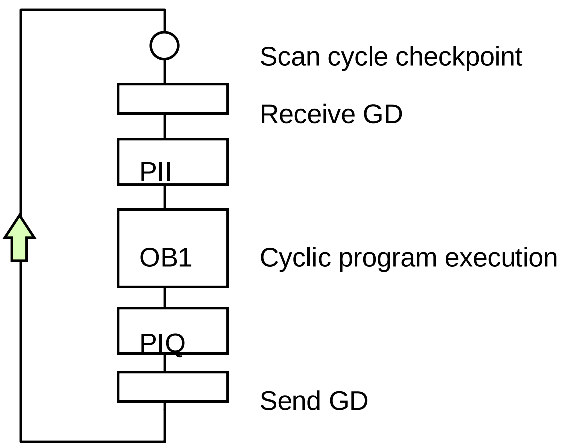
   Cyclic exchange of global data starts automatically.

### GD Exchange

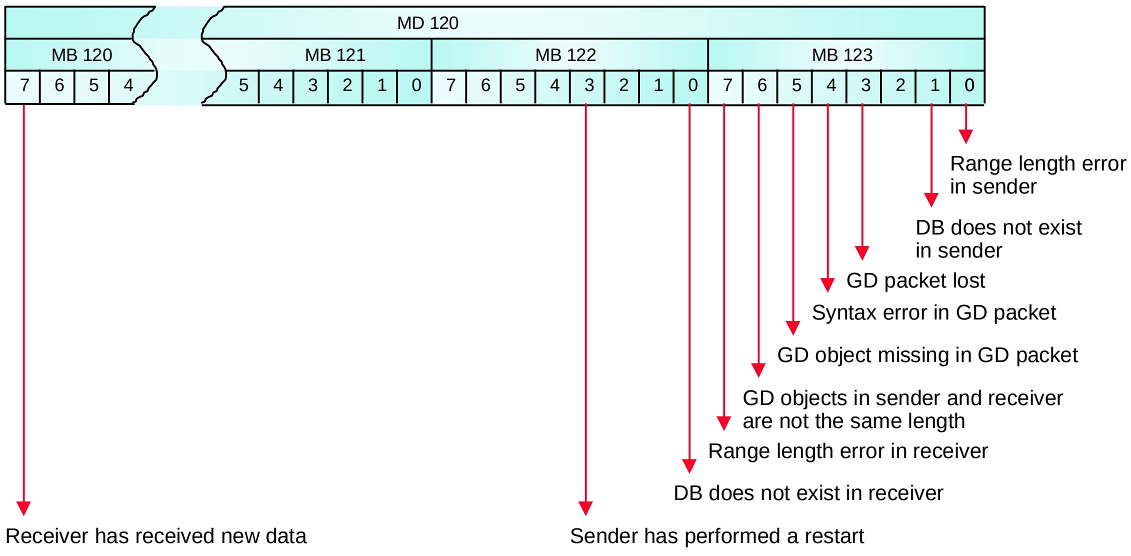
Global data is exchanged as follows:

- The sending CPU sends the global data at the end of a cycle.
- The receiving CPU transfers the data from the communication part of a CPU to the S7 address area at the beginning of a cycle.

You can specify a scan rate to set the number of scan cycles to elapse before the data is sent or received.



## Status of GD Communication




SIMATIC® S7

Date: 12.03.03
File: PRO1_15E.14
SITRAIN Training for Automation and Drives

**Status Indication**

You can specify a status doubleword for each GD packet for each CPU "involved". Status doublewords have the "GDS" identifier in the table.

**Status Doubleword Evaluation**

If you assign the status doubleword (GDS) to a CPU address (such as MD 120), you can evaluate the status in the user program or on the PG.

**Structure of the Status Doubleword**

The GD status doubleword is bit-oriented. The diagram shows the meanings of the bits if they are set. A bit remains set until the user program or a PG input resets the bit.

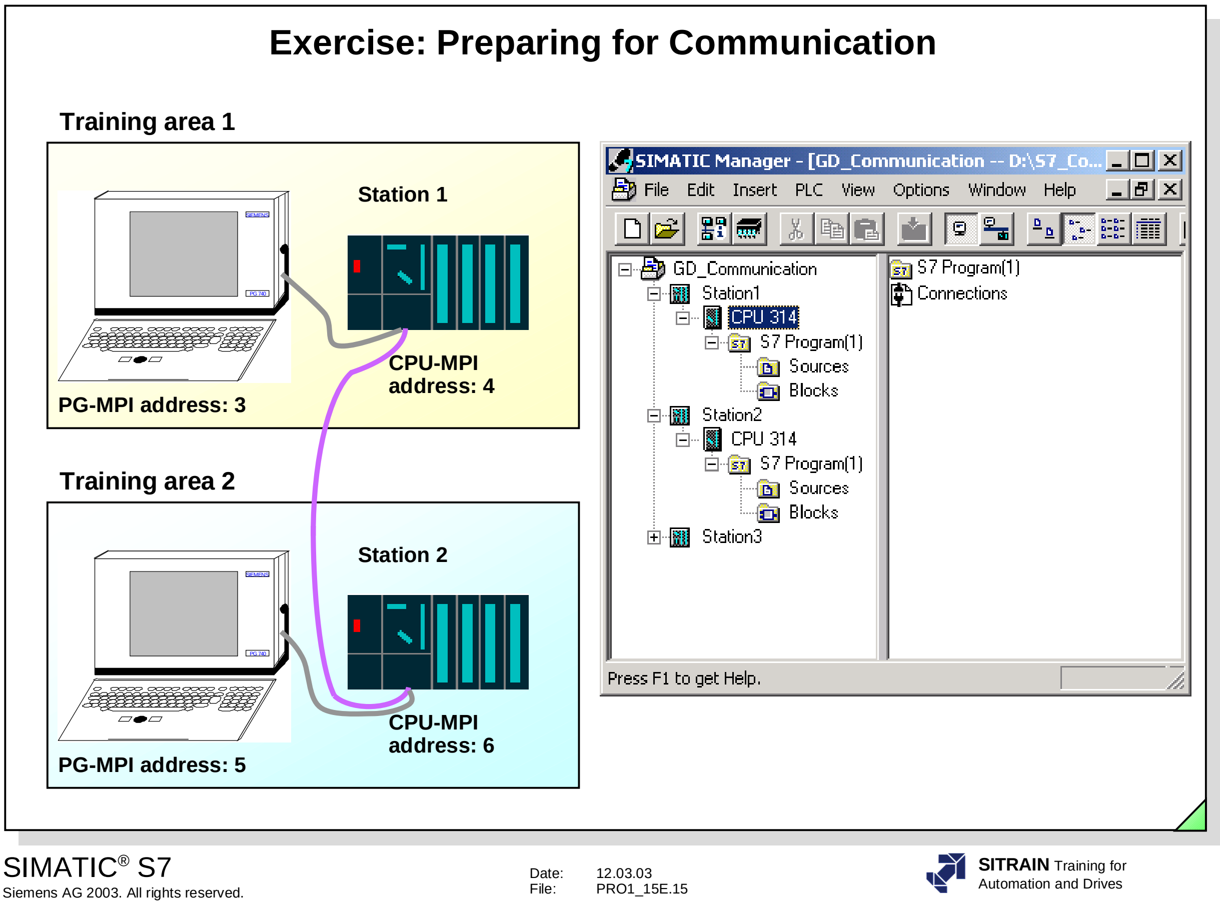
Bits that are not labeled are not used and have no meaning at present.

The GD status information requires a doubleword in memory. To make this easier to understand, MD 120 is used in the illustration.

**Group Status**

STEP 7 provides group status information (GST) for all GD packets.

This group status information, which is also stored in a doubleword with the same structure as the status doubleword (GDS), is the result obtained by OR-ing all the status words.




**Task**

To prepare for the global data communication between two stations, you have to physically network two training controllers with one another using a Profibus cable (see slide). Moreover, every training group creates the new project "GD_ Communication" with the hardware stations "Station1" and "Station2" that represent the two networked training controllers.

**Note**

Before you connect the two controllers with one another through the Profibus cable, you have to define the MPI addresses shown in the slide !!! Make the necessary decisions with the partner group.

**What To Do**

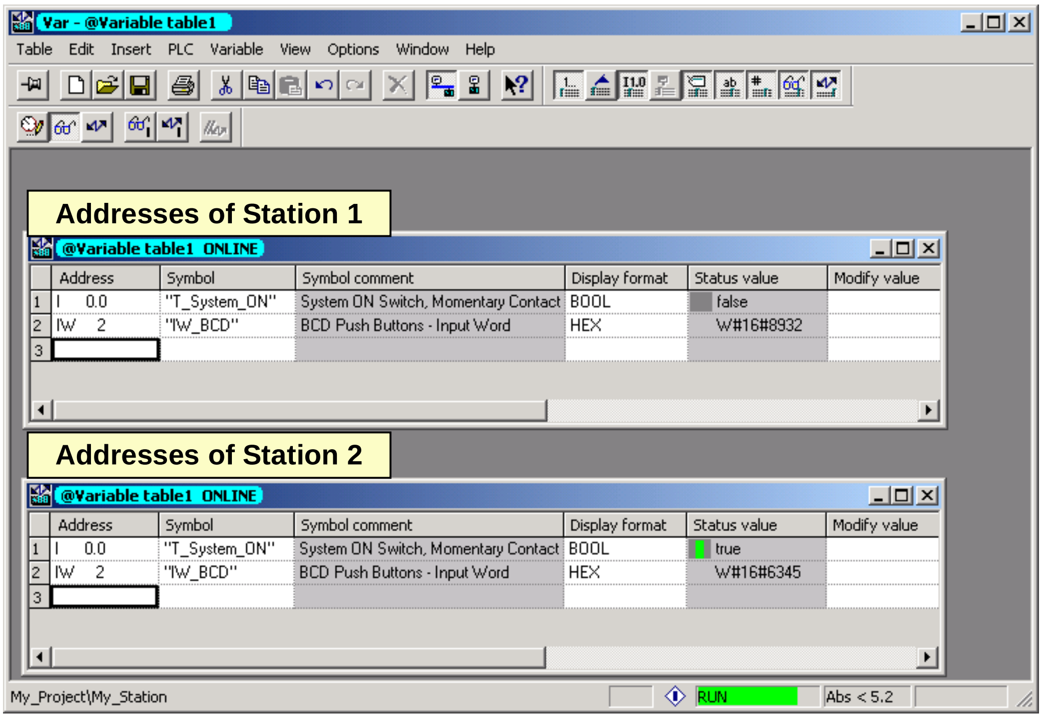
The following steps have to be carried out by each training group:

1. Perform a CPU memory reset
2. Create the new project called "GD_Communication"
3. Load your hardware station into the newly created project
   *SIMATIC® Manager -> PLC -> Upload Station*
4. Define the MPI address of the CPU and network the CPU *logically* (not physically using a cable !) with the Network MPI(1).
   *HW Config -> CPU Properties -> General -> Interface Properties*
5. Define the MPI address of your programming device.
   *SIMATIC® Manager -> Options -> Setting the PG/PC Interface...*
6. Now network the two training controllers *physically* with a Profibus cable
7. Load the hardware station of the partner group into your project
   *SIMATIC® Manager -> PLC -> Upload Station*

**Result**

Every training group has created the project "GD_Communication" in which both hardware stations "Station1" (MPI address 4) and "Station2" (MPI address 6) exists. The two stations represent the two training controllers which are logically and physically (via Profibus cable) networked with one another.

## Exercise: Monitoring the Addresses of Several Stations




SIMATIC® S7


Date: 12.03.03
File: PRO1_15E.16

**SITRAIN** Training for Automation and Drives

**Task**

You are to become familiar with monitoring variables from 2 CPUs simultaneously using the test function *Monitor/Modify Variables*.

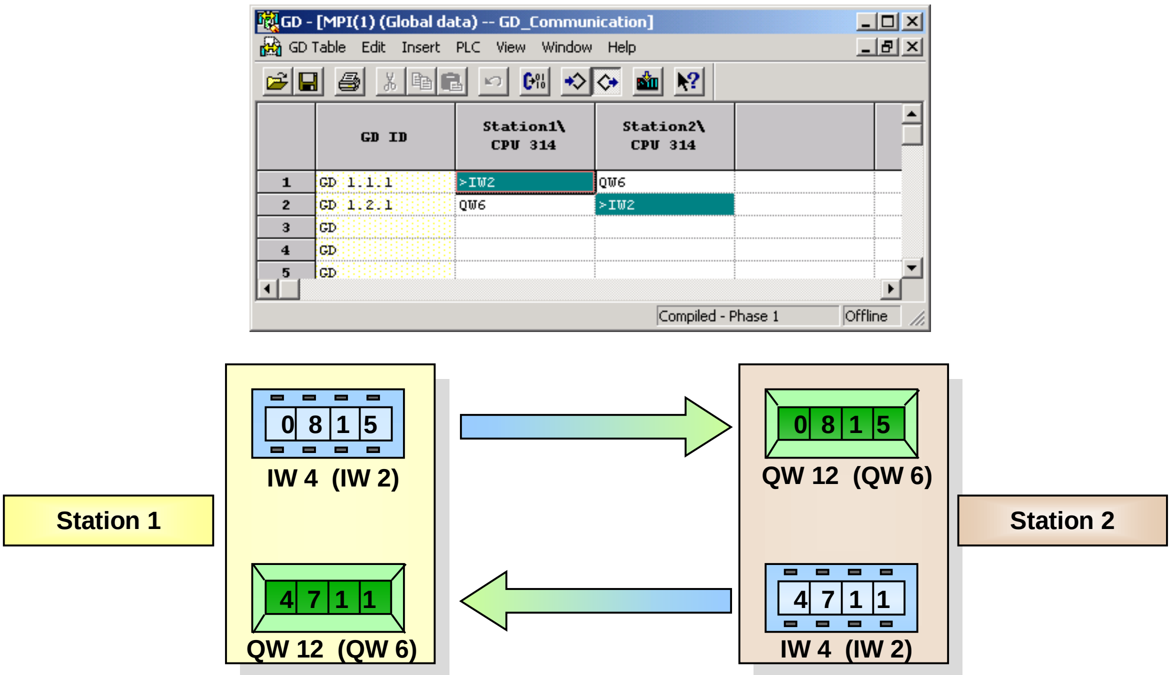
**Note**

In the previous exercise you networked the two training controllers both *physically* and *logically* with one another. For the (simultaneous) monitoring of variables using the MPI network, it would have been sufficient to define different MPI addresses and then subsequently *physically* network or insert the Profibus cable. The logical networking with the *HW Config* tool is only necessary for the global data communication that is still to follow.

**What To Do**

1. Start the "Monitor/Modify Variables" function in order to monitor the addresses of Station1 shown in the slide.
   *In the SIMATIC® Manager, select the Blocks folder of Station1 -> PLC -> Monitor/Modify Variables*
2. Monitor the addresses of Station2 shown in the slide in a new variable table, without exiting the monitoring of the addresses of Station 1.
   *Monitor/Modify Variables -> Table -> New -> Enter Address -> PLC -> Connect to -> Accessible CPU... -> in the follow-up dialog, select the S7 Program folder of Station 2 -> OK*
3. Set up the two tables (windows) one below the other as shown in the slide.
   *Monitor/Modify Variable -> Window -> Arrange -> Horizontally*

## Exercise: Global Data Communication



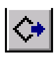




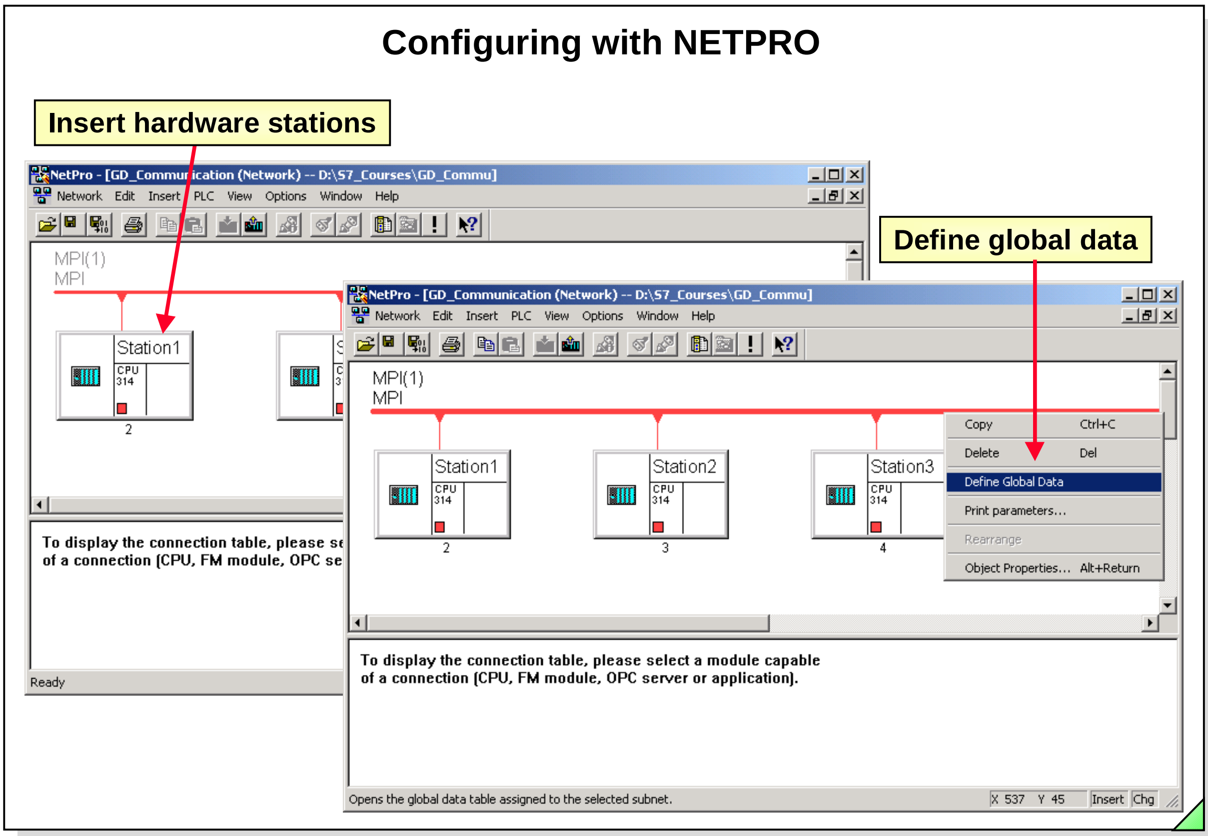
**Task** The number that is set on the BCD thumbwheel button of Station 1 is to be displayed on the BCD digital display of Station 2 and vice versa.

**Note** To implement the required function, you must merely edit, compile, save and load the global data table shown in the slide into the CPUs. Programming a user program is not necessary.

**What To Do**

1. Start the Editor for editing the global data table.
   *In the SIMATIC® Manager, select the project "GD_Communication" -> in the right window, select the "MPI(1)" object that has become visible -> Options -> Define global data*
2. Insert the CPUs taking part in the global data communication into the table.
   *Select the field in which the CPU is to be entered (see slide) -> Edit -> CPU... -> in the dialog, select the CPU*
3. In the table, enter the addresses that the CPUs are to exchange and select, in each case, the addresses that one CPU is to send as "Sender" (see slide).
   *Enter all addresses -> select the address that is to be the "Sender" -> specify addresses as "Sender" using* 
4. Compile the table
   *GD Table -> Compile...*
5. Save the table
   *GD Table -> Save*
6. Load the compiled table into <u>all</u> CPUs
   *PLC -> Download...*

## Configuring with NETPRO




SIMATIC® S7


Date: 12.03.03
File: PRO1_15E.18

SITRAIN Training for Automation and Drives

**Introduction**
Instead of the configuration method you have been using up to now, you can use the "NETPRO" tool to configure a network (MPI, Profibus or Industrial Ethernet) graphically.
This tool makes things clearer, provides you with documentation, and its tools, such as hardware configuration, are easy to call up.

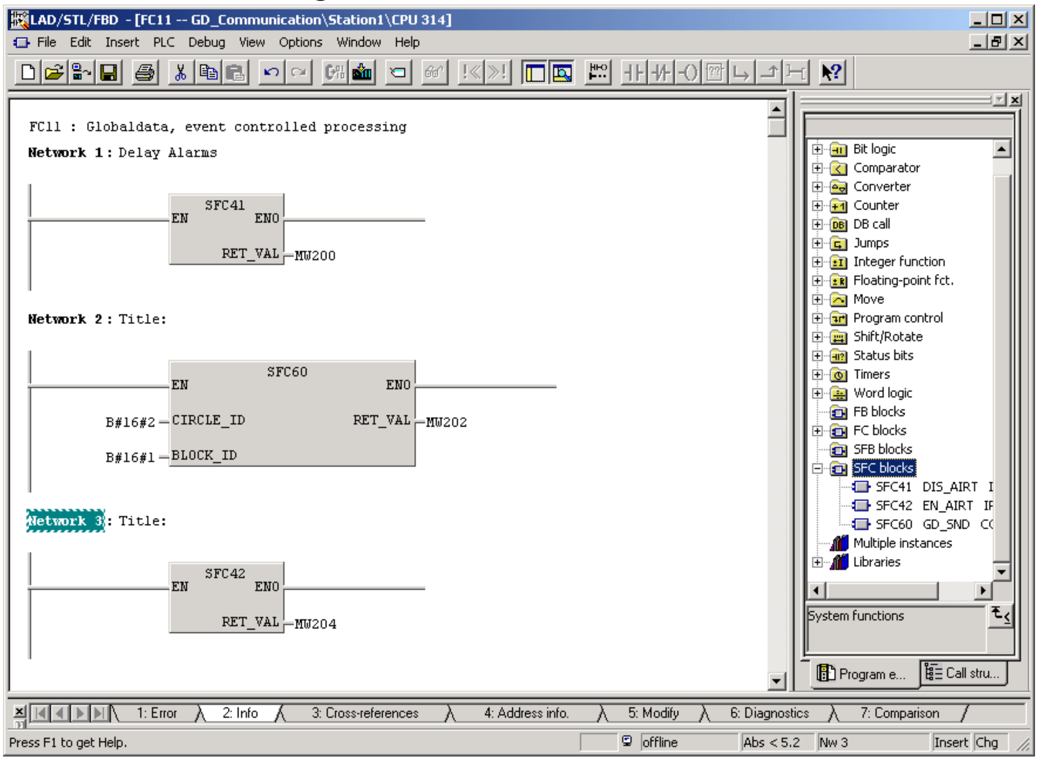
**Opening the Tool**
You open the tool by double-clicking a network icon, such as MPI, in the SIMATIC® Manager.

**Inserting Hardware Stations**
The catalog contains the components you need, such as subnets and stations, and you can insert them by drag and drop.

**Configuring Hardware**
When you have inserted the stations, you double-click to open the "Hardware Configuration" tool. You use this tool to set the MPI addresses and establish a connection to the subnet.

**Global Data**
Click the subnet, such as MPI, with the right mouse button and select the "Define Global Data" menu option. You create the global data table as before.

## Transferring Global Data with SFC 60, SFC 61




SIMATIC® S7


Date: 12.03.03
File: PRO1_15E.19

SITRAIN Training for Automation and Drives

**Introduction**

You can send and receive global data packets in a program-controlled and therefore event-driven way with SFC60 GD_SND and SFC61 GD_RCV.

The scan rate 0 must be specified in the GD table for the purely program-controlled data exchange.

You can also use the cyclic-driven and program-controlled modes either separately or combined.

**SFC60 "GD_SND"**

SFC60 collects the data of a GD packet and sends it on its configured way. SFC60 can be called anywhere in the user program.

SFC60 has the CIRCLE_ID (circle no. in which the send packet is found) and BLOCK_ID (packet no. of the packet to be sent) parameters.

**SFC61 "GD_RCV"**

SFC61 fetches the data for exactly one sent GD packet and enters it in the configured area. SFC61 can be called anywhere in the user program.

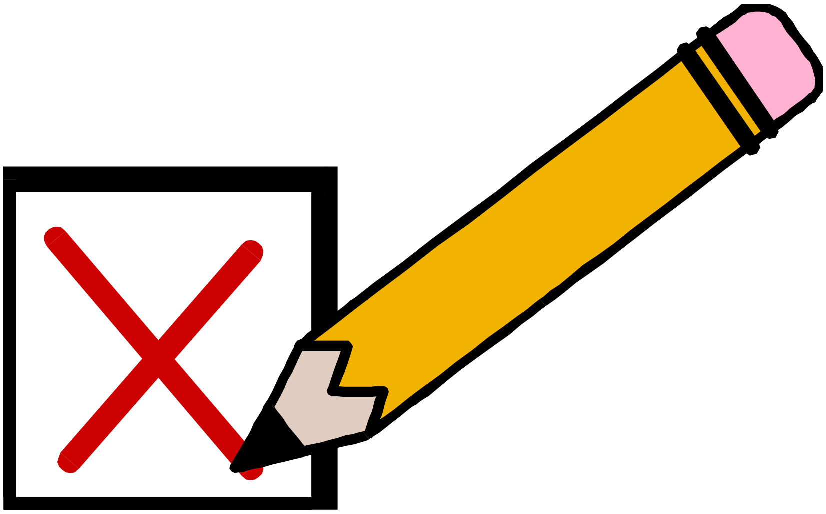
Analog to SFC60, SFC61 has the CIRCLE_ID dnd BLOCK_ID parameters. To guarantee data consistency, all interrupts must be disabled in the user program prior to the SFC60/ 61 calls.
For example:

- CALL SFC 39 //"Disable interrupt"
- CALL SFC 41 //"Delay interrupt"
- CALL SFC 60/61 //"Send/receive GD"
- CALL SFC 42 //"Enable delay"
- CALL SFC 40 //"Enable interrupts"

:
:

SIEMENS

# Solutions for the Exercises



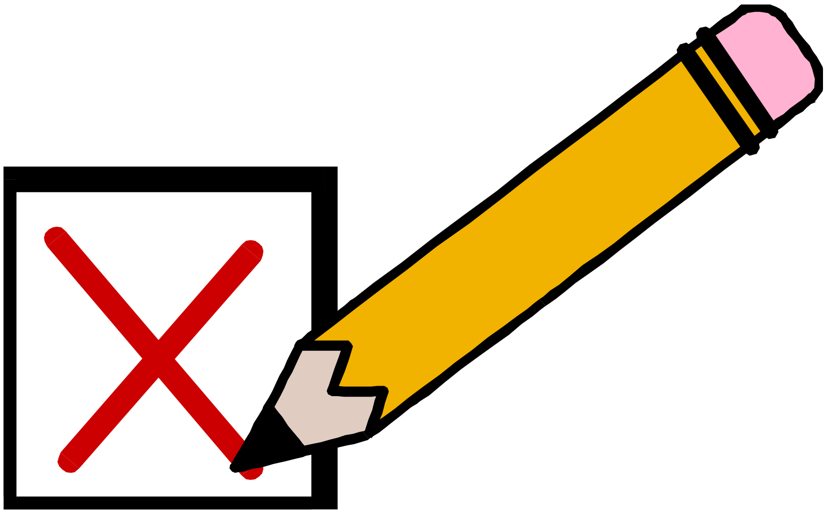
SIMATIC S7


Date: 12.03.03
File: PRO1_16E.1

SITRAIN Training for Automation and Drives

**Contents** Page

## Solutions for the Exercises



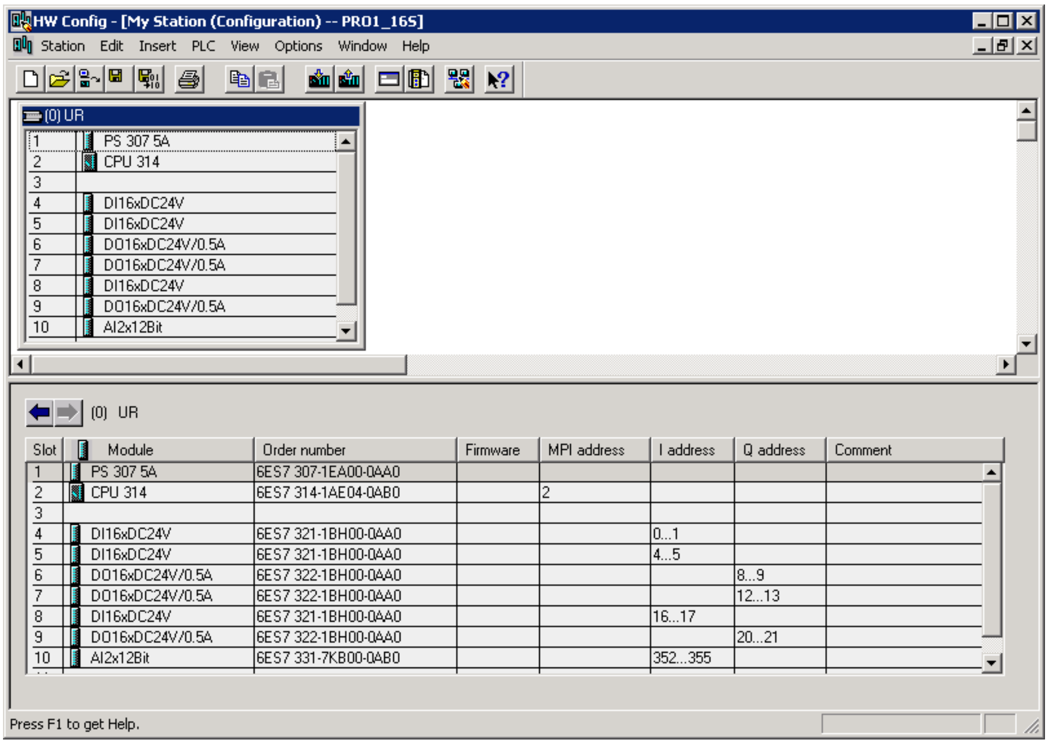
SIMATIC S7


Date: 12.03.03
File: PRO1_16E.2

SITRAIN Training for Automation and Drives

**Contents** Page

# Adapting the ACTUAL Configuration




SIMATIC S7

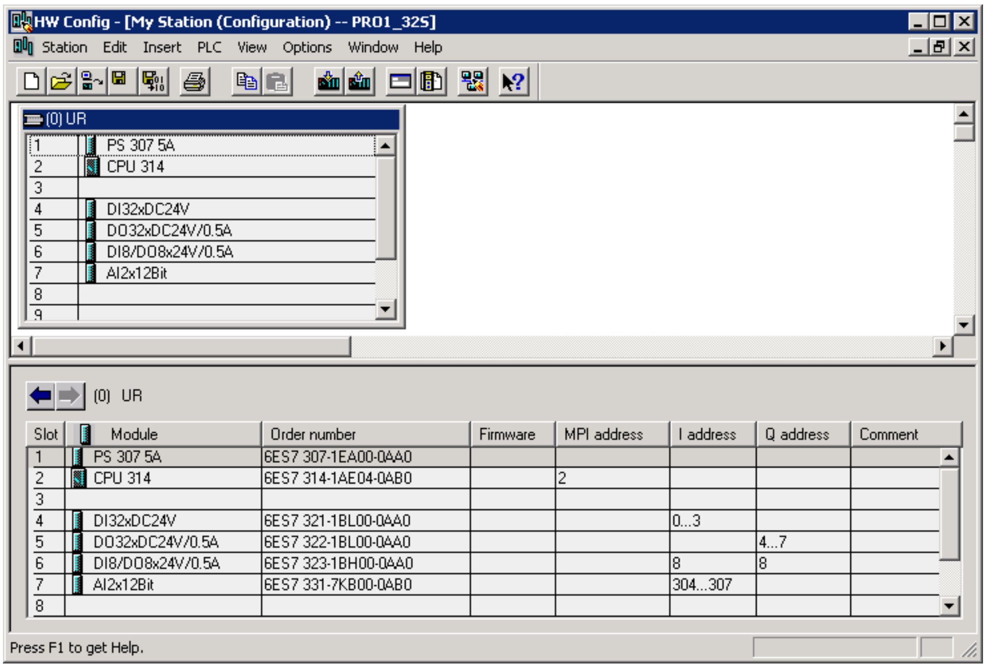


Date: 12.03.03
File: PRO1_16E.3

SITRAIN Training for Automation and Drives

**Note** The result of the Exercise is displayed in the picture above (for the S7-300 16 bit training unit).

# Adapting the ACTUAL Configuration



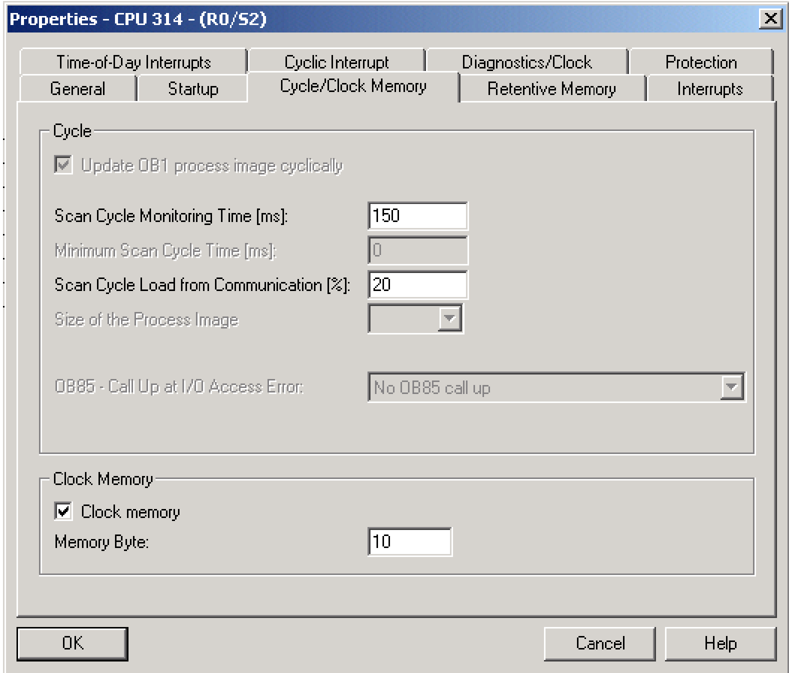

SIMATIC S7


Date: 12.03.03
File: PRO1_16E.4

**SITRAIN** Training for Automation and Drives

**Note** The result of the Exercise is displayed in the picture above (for the S7-300 32 bit training unit).

# Assign Parameters to CPU Clock Memory and Test




SIMATIC S7



Date: 12.03.03
File: PRO1_16E.5

SITRAIN Training for Automation and Drives

**Note** The result of the Exercise is displayed in the picture above.

# Symbol Table for the 16 bit Training Unit (Part 1)

| | Symbol | Address | Data type | Comment |
|---|---|---|---|---|
| 1 | C_Parts | C 18 | COUNTER | Transported Parts Counter |
| 2 | DB_Instance_Fault2 | DB 2 | FB 20 | Instance-DB for Evaluation of Fault 2 |
| 3 | DB_Instance_Fault3 | DB 3 | FB 20 | Instance-DB for Evaluation of Fault 3 |
| 4 | DB_Parts | DB 18 | DB 18 | DB with number of parts |
| 5 | FB_Faults | FB 20 | FB 20 | Evaluation of Faults with memory |
| 6 | FC_Operating_Modes | FC 15 | FC 15 | System ON/OFF, Selecting Operating Modes |
| 7 | FC_Conveyor | FC 16 | FC 16 | MAN/AUTO Control of Conveyor |
| 8 | FC_Op/Flt_Mess | FC 17 | FC 17 | Operating and Fault Messages |
| 9 | FC_Count | FC 18 | FC 18 | Count Transported Parts |
| 10 | FC_Fault | FC 20 | FC 20 | Evaluation of Faults no memory |
| 11 | FC_Scale | FC 105 | FC 105 | Scaling Values |
| 12 | T_System_ON | I 0.0 | BOOL | System ON Switch, Momentary Contact |
| 13 | T_System_OFF | I 0.1 | BOOL | System OFF Switch (N.C.), Momentary Contact |
| 14 | T_Jog_RT | I 0.2 | BOOL | Jog Conveyor Right, Momentary Contact |
| 15 | T_Jog_LT | I 0.3 | BOOL | Jog Conveyor Left, Momentary Contact |
| 16 | S_M/A_ModeSelect | I 0.4 | BOOL | Operating Mode Man=0/Auto=1 Selector Switch |
| 17 | T_M/A_Accept | I 0.5 | BOOL | Operating Mode Verification Switch |
| 18 | S_Weight/Number | I 0.6 | BOOL | Display Mode Weight=0/Number=1 Selector Switch |
| 19 | T_Conv_Rst | I 0.7 | BOOL | Conveyor Reset Switch, Momentary Contact |
| 20 | T_Fault_Rst | I 1.0 | BOOL | Fault Reset Switch, Momentary Contact |
| 21 | S_Fault1 | I 1.1 | BOOL | Fault #1 Activate Switch 0=Off/1=On |
| 22 | S_Fault2 | I 1.2 | BOOL | Fault #2 Activate Switch 0=Off/1=On |
| 23 | S_Fault3 | I 1.3 | BOOL | Fault #3 Activate Switch 0=Off/1=On |
| 24 | LB | I 16.0 | BOOL | Light Barrier at Conveyor End (N.C.) |
| 25 | T_PB1 | I 16.1 | BOOL | Push Button at Bay 1, Momentary Contact |
| 26 | T_PB2 | I 16.2 | BOOL | Push Button at Bay 2, Momentary Contact |
| 27 | T_PB3 | I 16.3 | BOOL | Push Button at Bay 3, Momentary Contact |
| 28 | T_PB4 | I 16.4 | BOOL | Push Button at Conveyor End, Momentary Contact |
| 29 | BAY1 | I 16.5 | BOOL | Proximity Sensor at Bay 1 |
| 30 | BAY2 | I 16.6 | BOOL | Proximity Sensor at Bay 2 |
| 31 | BAY3 | I 16.7 | BOOL | Proximity Sensor at Bay 3 |
| 32 | IW_BCD | IW 4 | WORD | BCD Push Buttons - Input Word |
| 33 | 2_Hz | M 10.3 | BOOL | 2 Hz Flashing Signal |
| 34 | M_LB_Edge | M 16.0 | BOOL | Cleared Light Barrier Edge Memory Location |
| 35 | M_Jog_Right | M 16.2 | BOOL | Memory bit for Jog right |
| 36 | M_Auto_right | M 16.3 | BOOL | Memory bit for Automatic Mode right |

# Symbol Table for the 16 bit Training Unit (Part 2)

| | Symbol | Address | Data type | Comment |
|---|---|---|---|---|
| 37 | M_Conv_Fault | M 17.0 | BOOL | Memory Bit Fault in AUTO Mode |
| 38 | M_Fault1 | M 17.1 | BOOL | Fault #1 Memory Location |
| 39 | M_Fault1_Edge | M 17.2 | BOOL | Fault #1 Edge Detection Memory location |
| 40 | M_Fault2 | M 17.3 | BOOL | Fault #2 Memory Location |
| 41 | M_Fault2_Edge | M 17.4 | BOOL | Fault #2 Edge Detection Memory location |
| 42 | M_Count_Edge | M 18.0 | BOOL | Edge Memory Bit for Count Parts |
| 43 | M_Weight_ok | M 35.0 | BOOL | Memory bit for weight ok |
| 44 | MW_Parts | MW 20 | INT | Transported Parts Count Memory Location |
| 45 | OB_Cycle | OB 1 | OB 1 | Cyclic Program |
| 46 | OB_Cyclic_Interrupt | OB 35 | OB 35 | Time-controlled Program |
| 47 | OB_Startup | OB 100 | OB 100 | Warm restart Program |
| 48 | PIW_Analog1 | PIW 352 | INT | Analog Channel 1 |
| 49 | L_Conv_Fault | Q 8.0 | BOOL | Conveyor Fault Light |
| 50 | L_SYSTEM | Q 8.1 | BOOL | System ON Light |
| 51 | L_MAN | Q 8.2 | BOOL | Manual Mode of Operation Light |
| 52 | L_AUTO | Q 8.3 | BOOL | Automatic Mode of Operation Light |
| 53 | L_Man_Rest | Q 8.5 | BOOL | Manual Restart Light |
| 54 | L_Aut_Rest | Q 8.6 | BOOL | Automatic Restart Light |
| 55 | L_Fault1 | Q 9.1 | BOOL | Fault #1 Indicator Light |
| 56 | L_Fault2 | Q 9.2 | BOOL | Fault #2 Indicator Light |
| 57 | L_Fault3 | Q 9.3 | BOOL | Fault #3 Indicator Light |
| 58 | L_ACT=SETP | Q 20.4 | BOOL | Actual = Setpoint Number of Parts Light |
| 59 | K_RT | Q 20.5 | BOOL | Run Conveyor Right |
| 60 | K_LT | Q 20.6 | BOOL | Run Conveyor Left |
| 61 | QW_Display | QW 12 | WORD | BCD - Output Display Word |
| 62 | SD_Conv_Contr | T 17 | TIMER | Control Conveyor in Auto Mode |

# Symbol Table for the 32 bit Training Unit (Part 1)

| | Symbol | Address | Data type | Comment |
|---|---|---|---|---|
| 1 | C_Parts | C 18 | COUNTER | Transported Parts Counter |
| 2 | DB_Instance_Fault2 | DB 2 | FB 20 | Instance-DB for Evaluation of Fault 2 |
| 3 | DB_Instance_Fault3 | DB 3 | FB 20 | Instance-DB for Evaluation of Fault 3 |
| 4 | DB_Parts | DB 18 | DB 18 | DB with number of parts |
| 5 | FB_Faults | FB 20 | FB 20 | Evaluation of Faults with memory |
| 6 | FC_Operating_Modes | FC 15 | FC 15 | System ON/OFF, Selecting Operating Modes |
| 7 | FC_Conveyor | FC 16 | FC 16 | MAN/AUTO Control of Conveyor |
| 8 | FC_Op/Flt_Mess | FC 17 | FC 17 | Operating and Fault Messages |
| 9 | FC_Count | FC 18 | FC 18 | Count Transported Parts |
| 10 | FC_Fault | FC 20 | FC 20 | Evaluation of Faults no memory |
| 11 | FC_Scale | FC 105 | FC 105 | Scaling Values |
| 12 | T_System_ON | I 0.0 | BOOL | System ON Switch, Momentary Contact |
| 13 | T_System_OFF | I 0.1 | BOOL | System OFF Switch (N.C.), Momentary Contact |
| 14 | T_Jog_RT | I 0.2 | BOOL | Jog Conveyor Right, Momentary Contact |
| 15 | T_Jog_LT | I 0.3 | BOOL | Jog Conveyor Left, Momentary Contact |
| 16 | S_M/A_ModeSelect | I 0.4 | BOOL | Operating Mode Man=0/Auto=1 Selector Switch |
| 17 | T_M/A_Accept | I 0.5 | BOOL | Operating Mode Verification Switch |
| 18 | S_Weight/Number | I 0.6 | BOOL | Display Mode Weight=0/Number=1 Selector Switch |
| 19 | T_Conv_Rst | I 0.7 | BOOL | Conveyor Reset Switch, Momentary Contact |
| 20 | T_Fault_Rst | I 1.0 | BOOL | Fault Reset Switch, Momentary Contact |
| 21 | S_Fault1 | I 1.1 | BOOL | Fault #1 Activate Switch 0=Off/1=On |
| 22 | S_Fault2 | I 1.2 | BOOL | Fault #2 Activate Switch 0=Off/1=On |
| 23 | S_Fault3 | I 1.3 | BOOL | Fault #3 Activate Switch 0=Off/1=On |
| 24 | T_PB1 | I 8.1 | BOOL | Push Button at Bay 1, Momentary Contact |
| 25 | T_PB2 | I 8.2 | BOOL | Push Button at Bay 2, Momentary Contact |
| 26 | T_PB3 | I 8.3 | BOOL | Push Button at Bay 3, Momentary Contact |
| 27 | T_PB4 | I 8.4 | BOOL | Push Button at Conveyor End, Momentary Contact |
| 28 | BAY1 | I 8.5 | BOOL | Proximity Sensor at Bay 1 |
| 29 | BAY2 | I 8.6 | BOOL | Proximity Sensor at Bay 2 |
| 30 | BAY3 | I 8.7 | BOOL | Proximity Sensor at Bay 3 |
| 31 | LB | I 16.0 | BOOL | Light Barrier at Conveyor End (N.C.) |
| 32 | IW_BCD | IW 2 | WORD | BCD Push Buttons - Input Word |
| 33 | 2_Hz | M 10.3 | BOOL | 2 Hz Flashing Signal |
| 34 | M_LB_Edge | M 16.0 | BOOL | Cleared Light Barrier Edge Memory Location |
| 35 | M_Jog_Right | M 16.2 | BOOL | Memory bit for Jog right |
| 36 | M_Auto_right | M 16.3 | BOOL | Memory bit for Automatic Mode right |

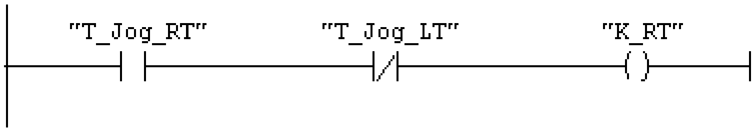
# Symbol Table for the 32 bit Training Unit (Part 2)

| | Symbol | Address | Data type | Comment |
|---|---|---|---|---|
| 37 | M_Conv_Fault | M 17.0 | BOOL | Memory Bit Fault in AUTO Mode |
| 38 | M_Fault1 | M 17.1 | BOOL | Fault #1 Memory Location |
| 39 | M_Fault1_Edge | M 17.2 | BOOL | Fault #1 Edge Detection Memory location |
| 40 | M_Fault2 | M 17.3 | BOOL | Fault #2 Memory Location |
| 41 | M_Fault2_Edge | M 17.4 | BOOL | Fault #2 Edge Detection Memory location |
| 42 | M_Count_Edge | M 18.0 | BOOL | Edge Memory Bit for Count Parts |
| 43 | M_Weight_ok | M 35.0 | BOOL | Memory bit for weight ok |
| 44 | MW_Parts | MW 20 | INT | Transported Parts Count Memory Location |
| 45 | OB_Cycle | OB 1 | OB 1 | Cyclic Program |
| 46 | OB_Cyclic_Interrupt | OB 35 | OB 35 | Time-controlled Program |
| 47 | OB_Startup | OB 100 | OB 100 | Warm restart Program |
| 48 | PIW_Analog1 | PIW 352 | INT | Analog Channel 1 |
| 49 | L_Conv_Fault | Q 4.0 | BOOL | Conveyor Fault Light |
| 50 | L_SYSTEM | Q 4.1 | BOOL | System ON Light |
| 51 | L_MAN | Q 4.2 | BOOL | Manual Mode of Operation Light |
| 52 | L_AUTO | Q 4.3 | BOOL | Automatic Mode of Operation Light |
| 53 | L_Man_Rest | Q 4.5 | BOOL | Manual Restart Light |
| 54 | L_Aut_Rest | Q 4.6 | BOOL | Automatic Restart Light |
| 55 | L_Fault1 | Q 5.1 | BOOL | Fault #1 Indicator Light |
| 56 | L_Fault2 | Q 5.2 | BOOL | Fault #2 Indicator Light |
| 57 | L_Fault3 | Q 5.3 | BOOL | Fault #3 Indicator Light |
| 58 | L_ACT=SETP | Q 8.4 | BOOL | Actual = Setpoint Number of Parts Light |
| 59 | K_RT | Q 8.5 | BOOL | Run Conveyor Right |
| 60 | K_LT | Q 8.6 | BOOL | Run Conveyor Left |
| 61 | QW_Display | QW 6 | WORD | BCD - Output Display Word |
| 62 | SD_Conv_Contr | T 17 | TIMER | Control Conveyor in Auto Mode |

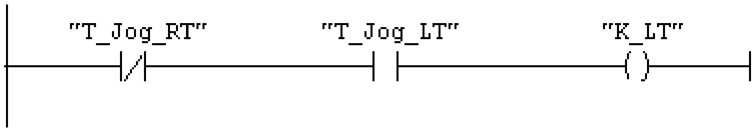
# Jog Motor (FC 16)

FC16 : Conveyor Operation

**Network 1** : Jog Conveyor RIGHT



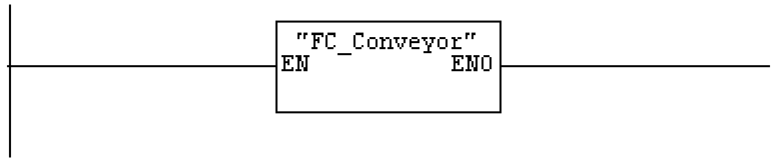

**Network 2** : Jog Conveyor LEFT



# Calling FC 16 in OB 1

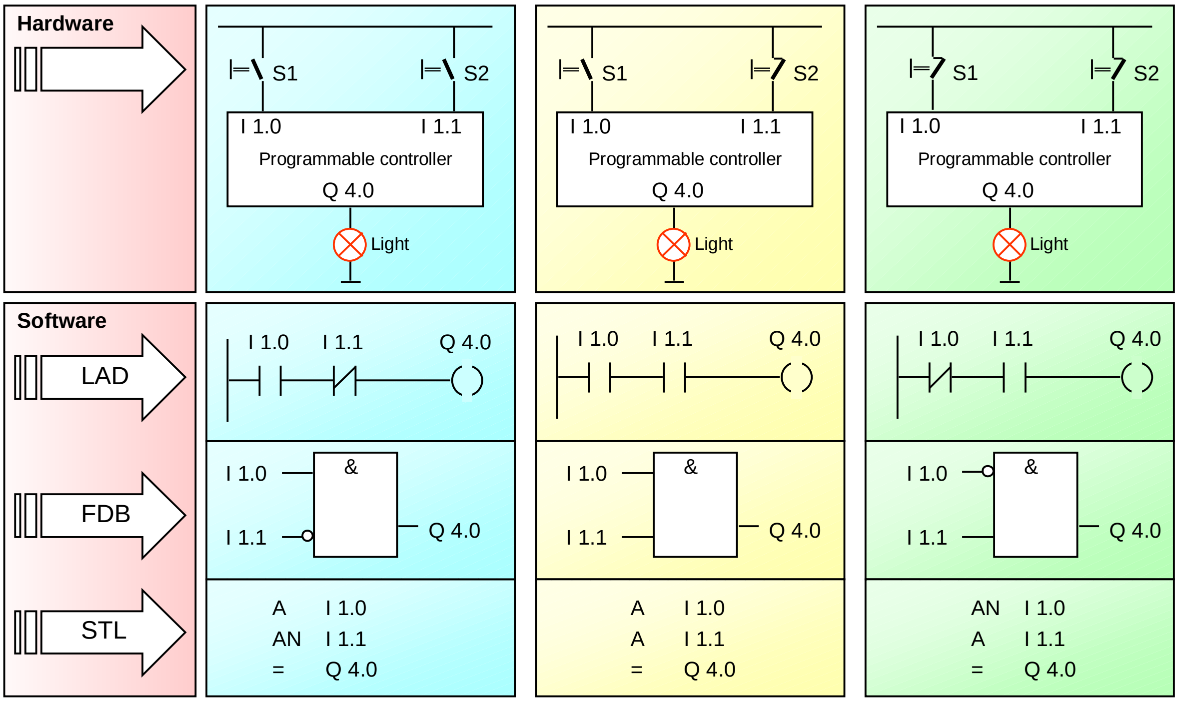
OB1 :  "Main Program Sweep (Cycle)"

**Network 1**: Title:



## Normally Open and Normally Closed Contacts

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




SIMATIC S7


Date: 12.03.03
File: PRO1_16E.12

SITRAIN Training for Automation and Drives

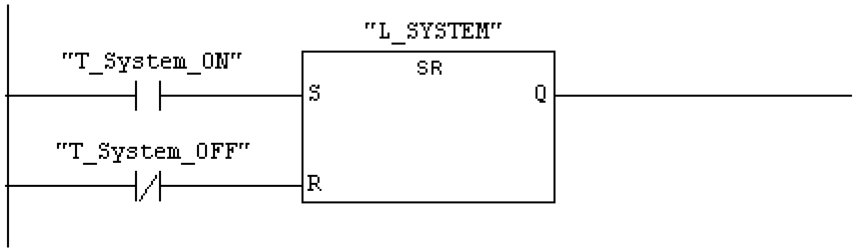
**Exercise** Complete the programs above to obtain the following functionality: When switch S1 is activated and switch S2 is not activated, the light should be ON in all three cases.

**Note !** The terms "NO contact" and "NC contact" have different meanings depending on whether they are used in the process hardware context or as symbols in the software.

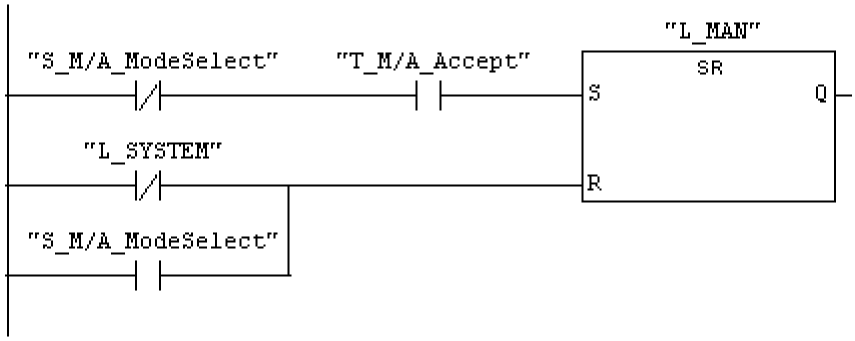
## Mode Section of the Distribution Conveyor

FC15 : Mode section

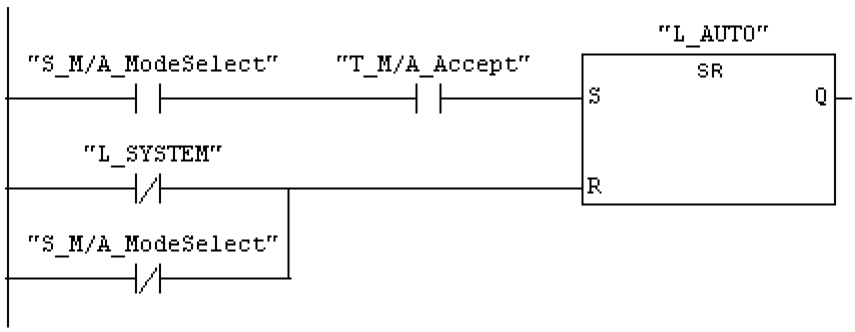
**Network 1**: System ON




**Network 2**: MANUAL Mode




**Network 3**: AUTO Mode



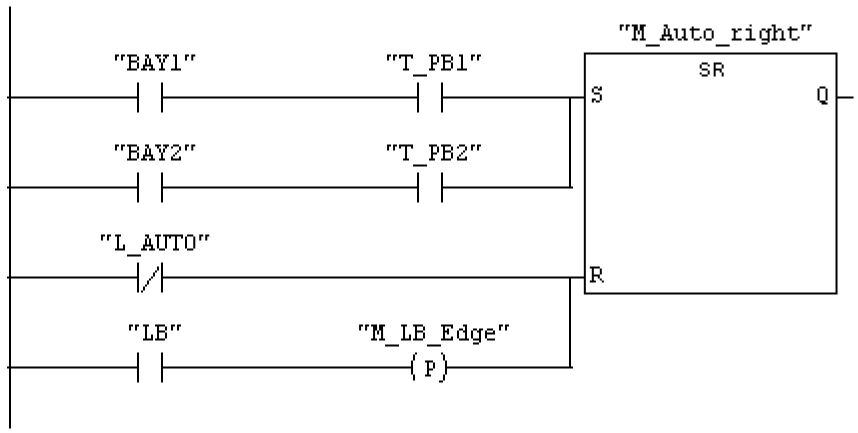
# Conveyor Movement in AUTO Mode

FC16 : Conveyor Operation

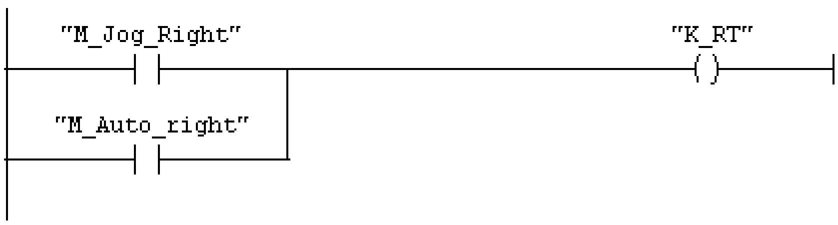
**Network 1**: Jog Conveyor RIGHT


**Network 2**: Jog Conveyor LEFT


**Network 3**: Conveyor in AUTO Mode



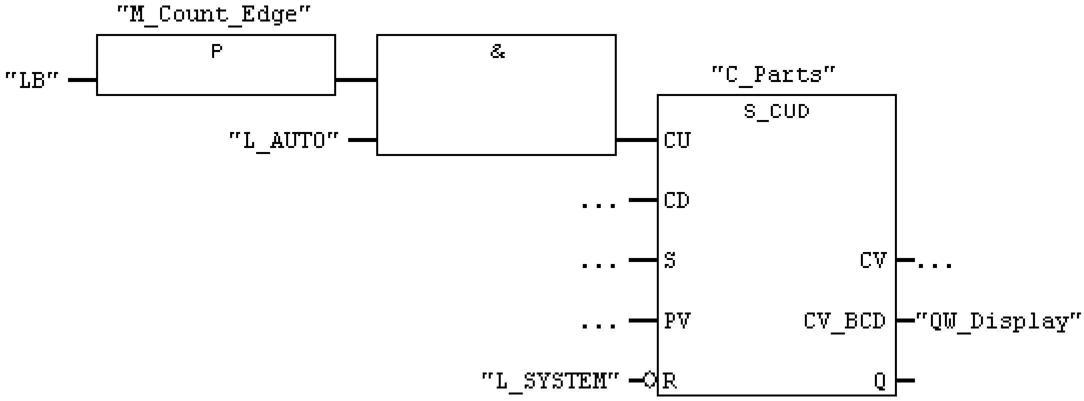

**Network 4**: Title:



## Counting the Transported Parts (FC 18, C 18)

FC18 : Title:

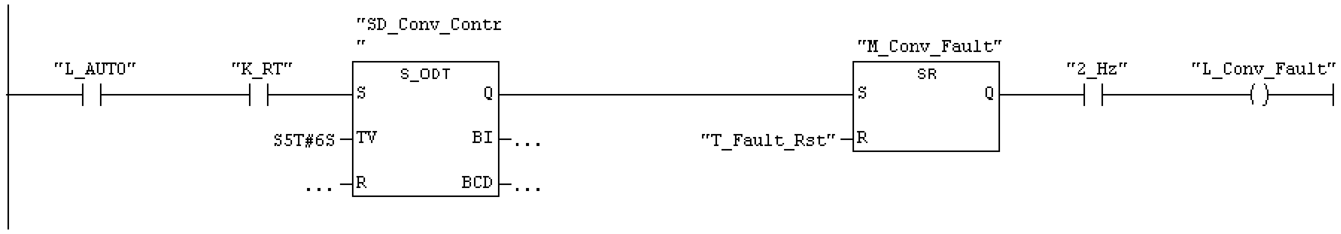
**Network 1**: Count transported parts in AUTO Mode



# Monitoring the Transport Functions (FC 17)

FC17 : Title:

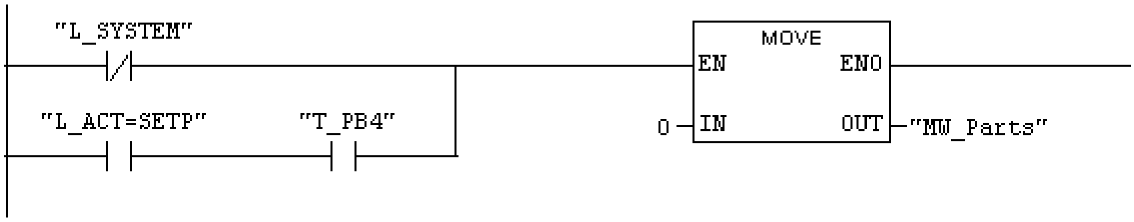
**Network 1**: Conveyor Fault: Monitoring the transport function



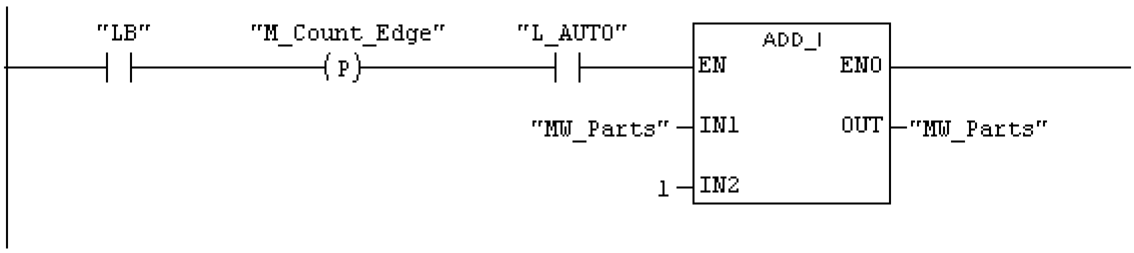
# Counting the Transported Parts (FC 18, MW 20)

FC18 : Title:

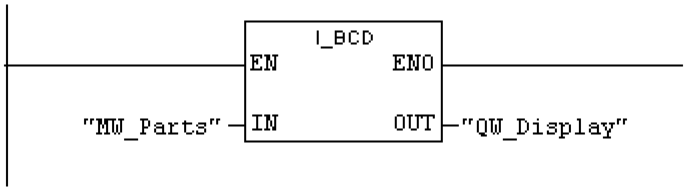
**Network 1**: Reset Counter



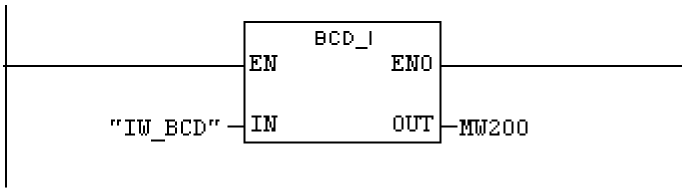

**Network 2** : Count transported parts in AUTO Mode



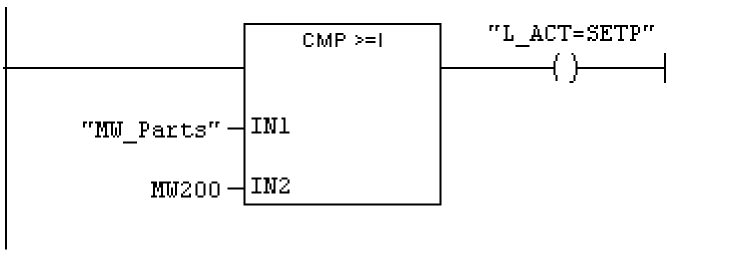

**Network 3** : Display actual number of parts




**Network 4** : Read Setpoint number of parts and convert



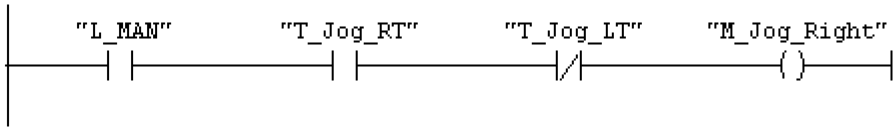

**Network 5** : SETPOINT-ACTUAL number of parts Comparison



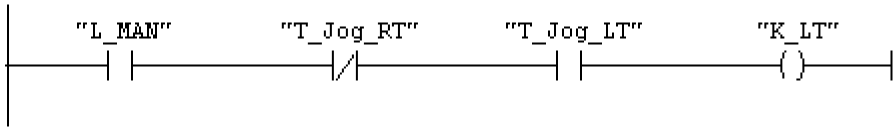
# Programming the Interlock in FC 16 (Conveyor Operation)

FC16 : Conveyor Operation

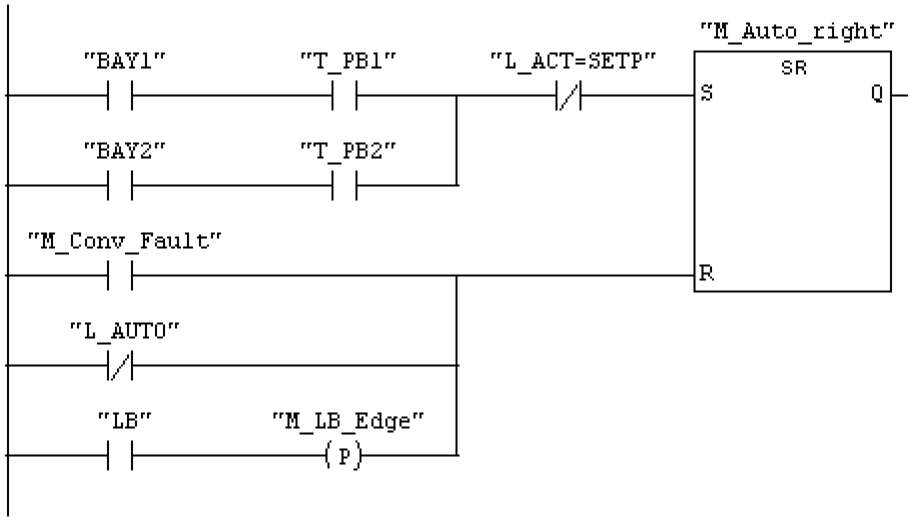
**Network 1 :** Jog Conveyor RIGHT



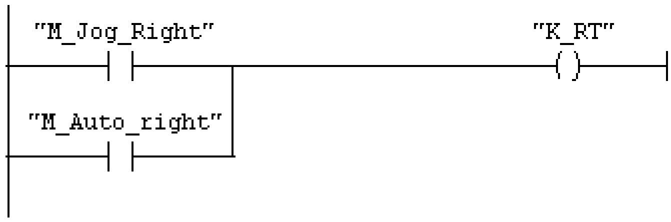

**Network 2 :** Jog Conveyor LEFT




**Network 3 :** Conveyor in AUTO Mode



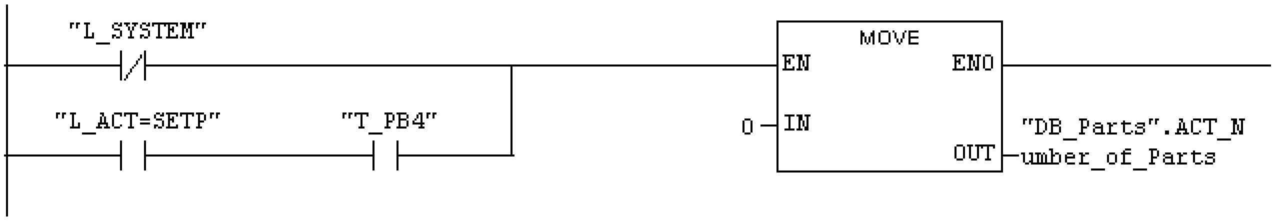

**Network 4 :** Title:



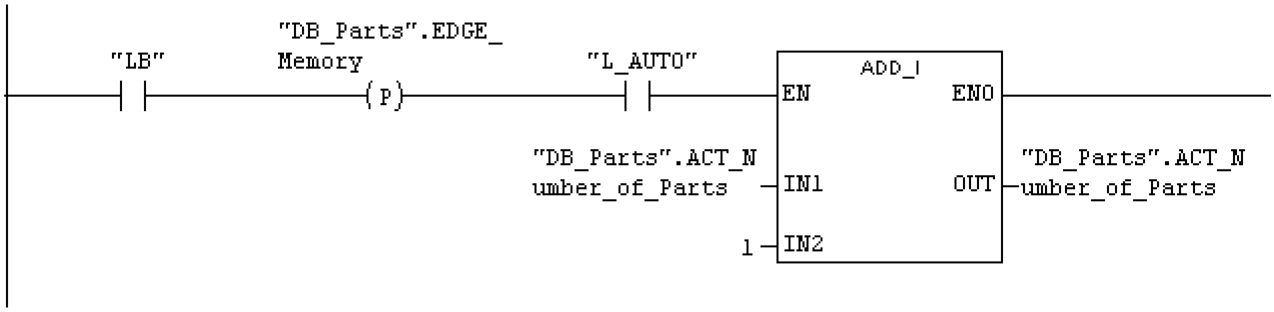
# Counting the Transported Parts (Data Word in FC 18)

FC18 : Title:

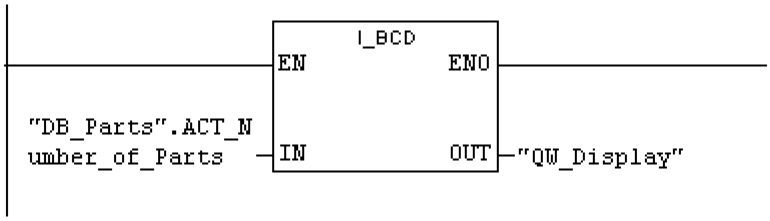
**Network 1 :** Reset Counter



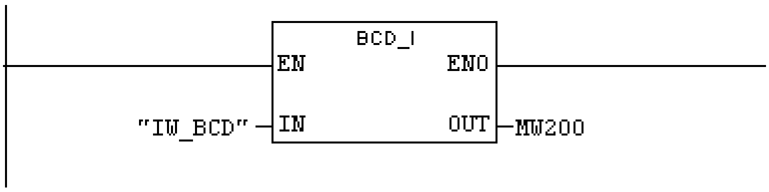

**Network 2 :** Count transported Parts in AUTO Mode



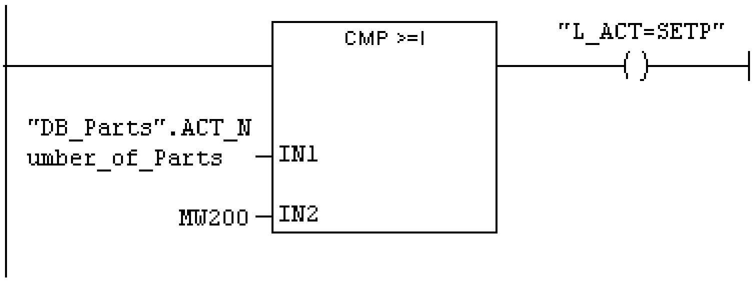

**Network 3 :** Display Actual Number of parts




**Network 4 :** Read Setpoint number of parts and convert




**Network 5 :** SETPOINT-ACTUAL number of parts Comparison



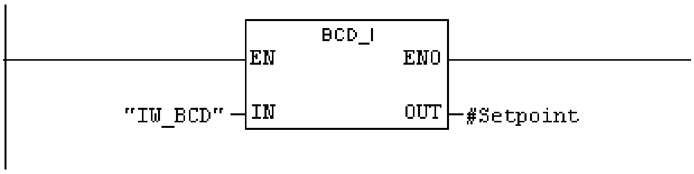
## Counting the Transported Parts (Data Word in FC 18), Data Block "DB_Parts" (DB 18)

| Address | Name | Type | Initial value | Comment |
|---|---|---|---|---|
| 0.0 | | STRUCT | | |
| +0.0 | ACT_Number_of_Parts | INT | 0 | Actual Number of transported Parts |
| +2.0 | EDGE_Memory | BOOL | FALSE | Auxiliary Memory bit edge detection |
| =4.0 | | END_STRUCT | | |

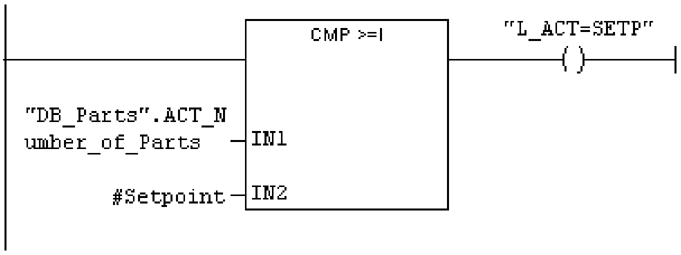
# Using temporary Variables (FC 18)

| Address | Declaration | Name | Type | Initial val | Comment |
|---|---|---|---|---|---|
| | in | | | | |
| | out | | | | |
| | in_out | | | | |
| 0.0 | temp | Setpoint | INT | | |

**Network 4 :** Read Setpoint number of parts and convert




**Network 5 :** SETPOINT-ACTUAL number of parts Comparison

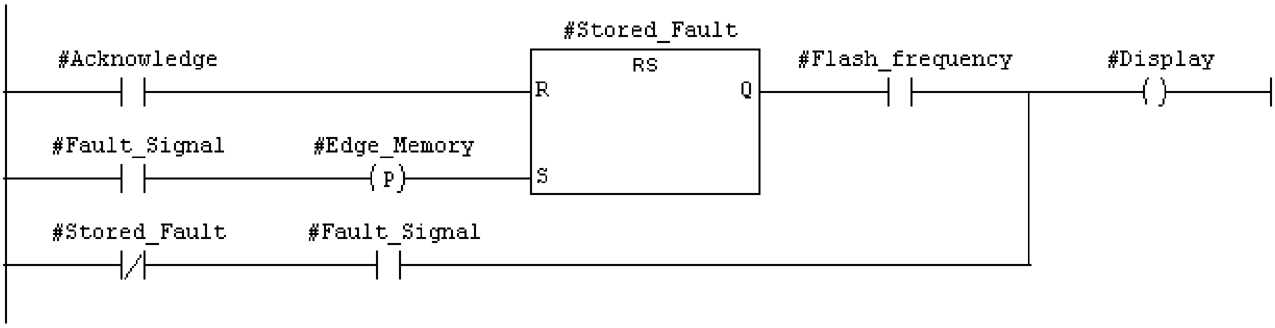


# Editing a Parameter-assignable FC (FC 20)

| Address | Declaration | Name | Type | Initial value | Comment |
|---|---|---|---|---|---|
| 0.0 | in | Fault_Signal | BOOL | | |
| 0.1 | in | Acknowledge | BOOL | | |
| 0.2 | in | Flash_frequency | BOOL | | |
| 2.0 | out | Display | BOOL | | |
| 4.0 | in_out | Stored_Fault | BOOL | | |
| 4.1 | in_out | Edge_Memory | BOOL | | |
| | temp | | | | |

FC20 : Function for Fault Signals

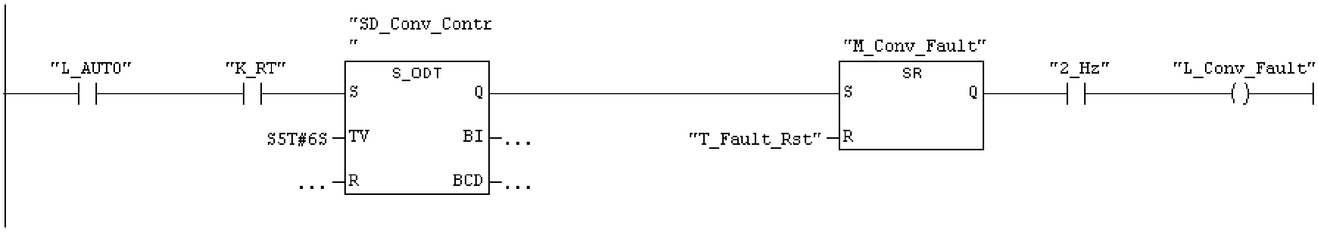
**Network 1**: Fault Logic



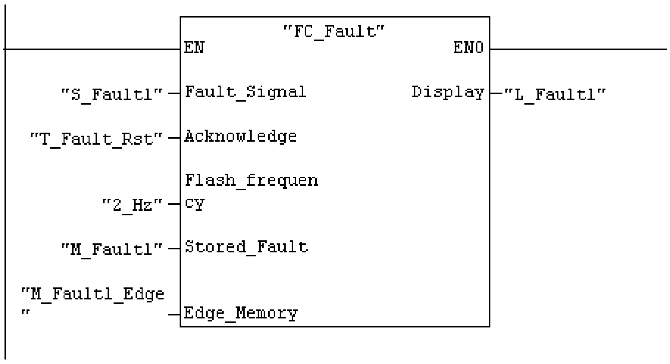
# Calling a Parameter-assignable FC (FC 20) (in FC 17)

FC17 : Title:

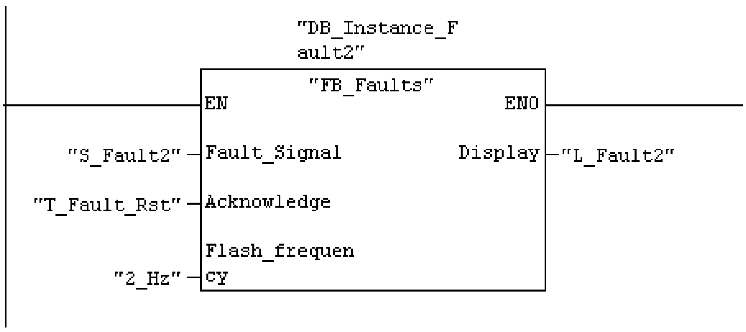
**Network 1**: Conveyor Fault: Monitoring the transport function



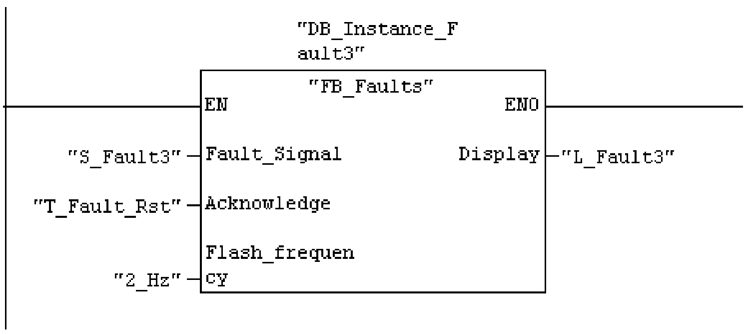

**Network 2**: Evaluation of disturbance 1




**Network 3**: Evaluation of disturbance 2




**Network 4**: Evaluation of disturbance 3

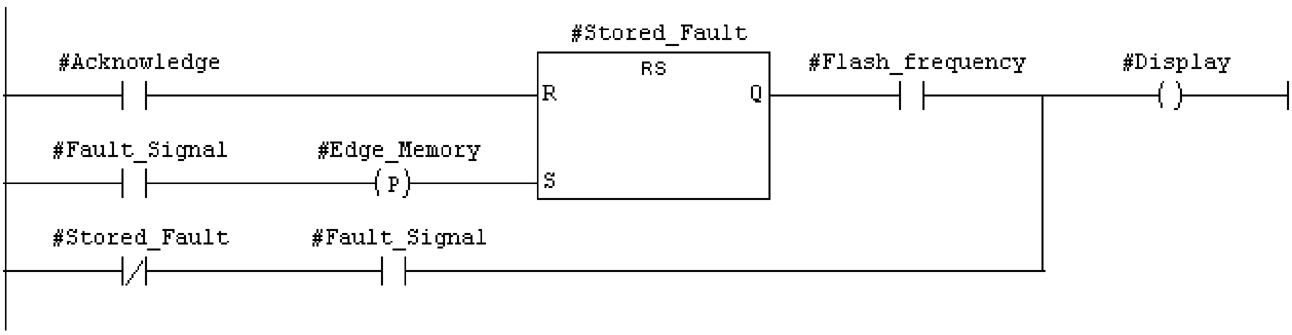


# Editing a Parameter-assignable FB (FB 20)

| Address | Declaration | Name | Type | Initial val | Comment |
|---|---|---|---|---|---|
| 0.0 | in | Fault_Signal | BOOL | FALSE | |
| 0.1 | in | Acknowledge | BOOL | FALSE | |
| 0.2 | in | Flash_frequency | BOOL | FALSE | |
| 2.0 | out | Display | BOOL | FALSE | |
| | in_out | | | | |
| 4.0 | stat | Stored_Fault | BOOL | FALSE | |
| 4.1 | stat | Edge_Memory | BOOL | FALSE | |
| | temp | | | | |

FB20 : Function for Fault Signals

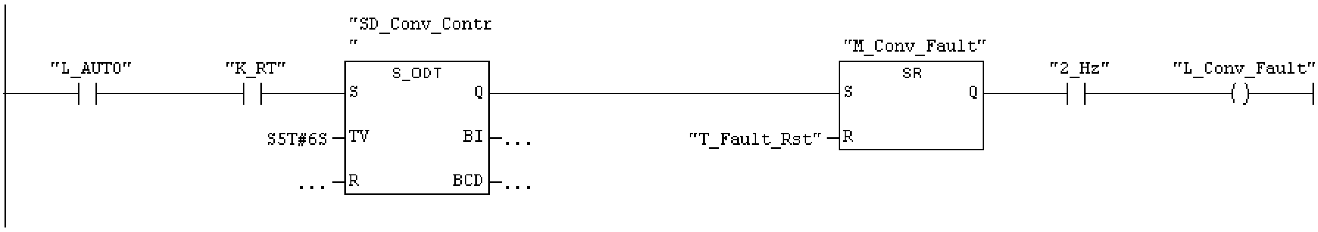
**Network 1**: Fault Logic



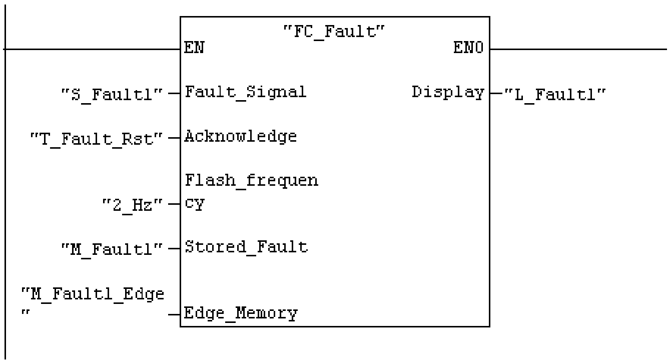
# Calling a Parameter-assignable FB (FB 20) (in FC 17)

FC17 : Title:

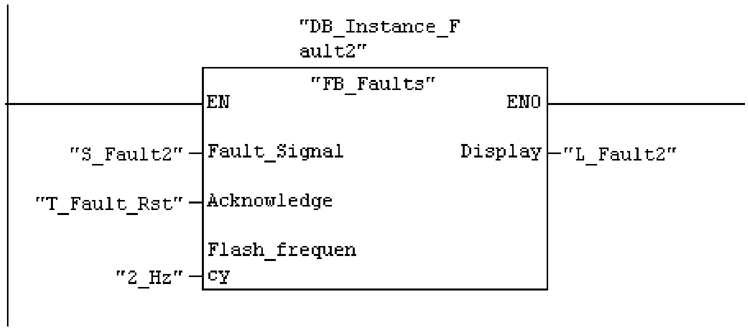
**Network 1**: Conveyor Fault: Monitoring the transport function



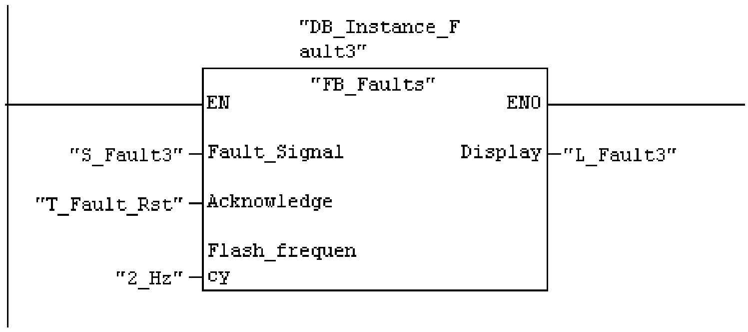

**Network 2**: Evaluation of disturbance 1




**Network 3**: Evaluation of disturbance 2




**Network 4**: Evaluation of disturbance 3



## Exercise: Recognizing Types of Variables

| Address | Decl. | Name | Type | Initial Value | Comment |
|---|---|---|---|---|---|
| 0.0 | in | Number_1 | WORD | W#16#0 | |
| 2.0 | in | Number_2 | WORD | W#16#0 | |
| 4.0 | out | Result | WORD | W#16#0 | |
| | in_out | | | | |
| 6.0 | stat | Max_value | INT | 0 | |
| 0.0 | temp | Intermediate_result | INT | | |

| Statement | TYPE OF VARIABLE | | | | | | |
|---|---|---|---|---|---|---|---|
| | Global | Local | Absolute | Symbolic | Temporary | Static | Parameter |
| L #Number_1 | | X | | X | | | X |
| L #Number_2 | | X | | X | | | X |
| T #Max_value | | X | | X | | X | |
| L #Intermediate_result | | X | | X | X | | |
| L “Number_1" | X | | | X | | | |
| T MW 40 | X | | X | | | | |
| T #Number_2 | | X | | X | | | X |





**Question** What is not correct in the statement T #Number_2 ?

*Number_2 is defined as an input parameter and thus read-only accesses are possible*

# Overview: Stop Errors and Logical Errors

**Stop Errors:**

| Error | Interrupt Point | Error Location | Incorrect -> Correct Instruction | |
|---|---|---|---|---|
| 1 | FC 18, Network 4 | FC 18, Network 4 | L PIW400 | L „IW_BCD“ |
| 2 | FC 18, Network 5 | FC 18, Network 5 | L DB18.DBW10<br>L #Setpoint<br>>=I | L DBW18.DBW0<br>L #Setpoint<br>>=I |
| 2 | FC 20, Network 1 | FC 17, Network 2 | CALL FC20<br>Flash Frequency<br>= DBX10.3 | CALL FC20<br>Flash Frequency<br>= „2_Hz“ (M10.3) |

**Logical Errors:**

| Error | Fault Function | Error Location | Incorrect -> Correct Instruction | |
|---|---|---|---|---|
| 1 | Jog Conveyor to right not possible | FC 16, Network 1 | = "K_RT“ | = "M_Jog_Right“ |
| 2 | Evaluation Disturbance3: no flash frequency | FC 17, Network 4 | CALL FB 20, DB 3<br>Flash freq.: | CALL FB 20,DB3<br>Flash freq.: „2_Hz“ |
| 3 | Record and display act. Numb.of parts not correct | FC 18, Network 2 | :<br>L #Setpoint | :<br>L 1 |

SIMATIC S7

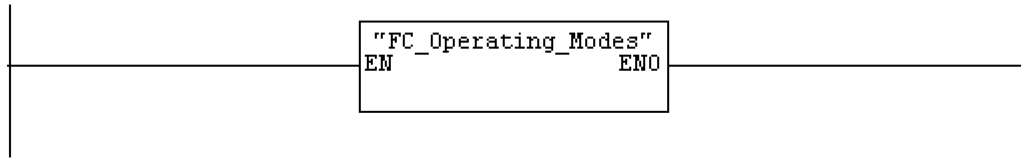

Date: 12.03.03
File: PRO1_16E.27

SITRAIN Training for Automation and Drives

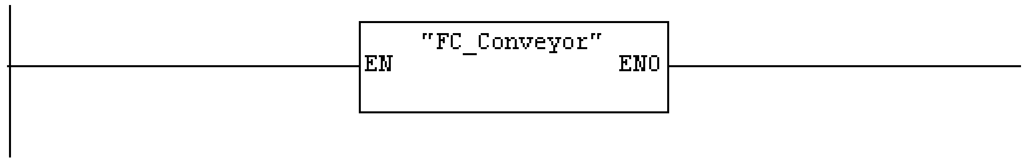
# Print out of the Error Program

OB1 :  "Main Program Sweep (Cycle)"

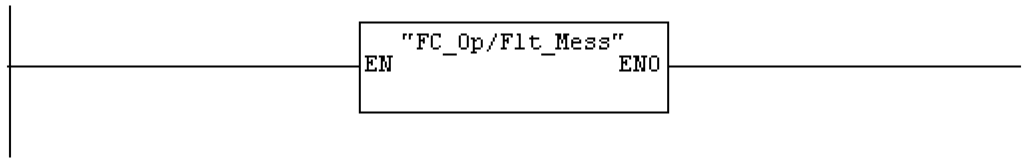
**Network 1** : Title:



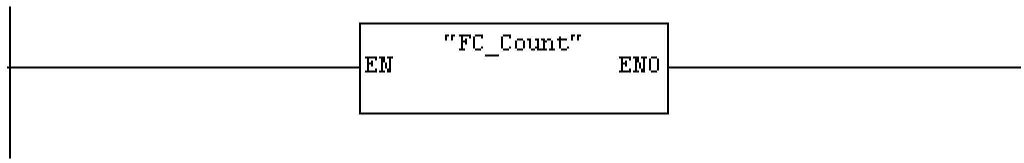

**Network 2** : Title:




**Network 3** : Title:



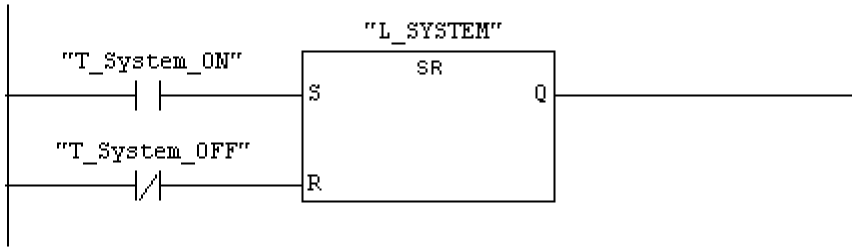

**Network 4** : Title:



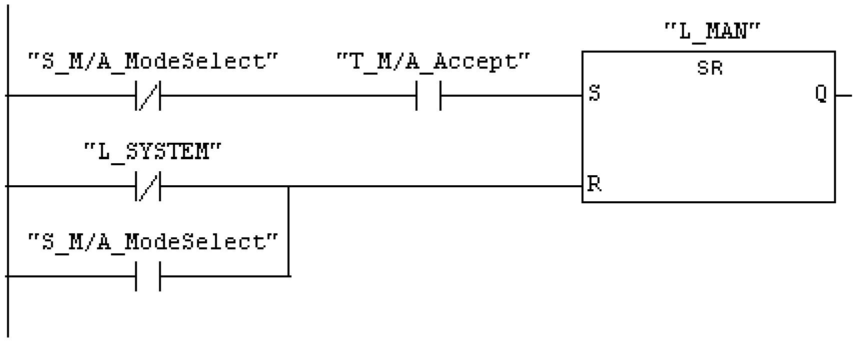
## Print out of the Error Program

FC15 : Mode section

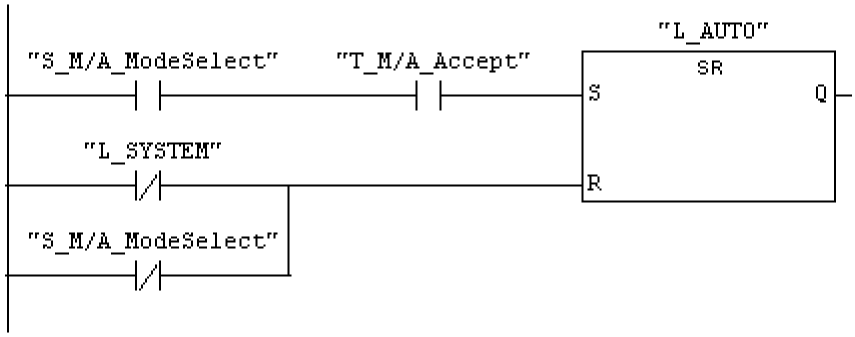
**Network 1** : System ON




**Network 2** : MANUAL Mode



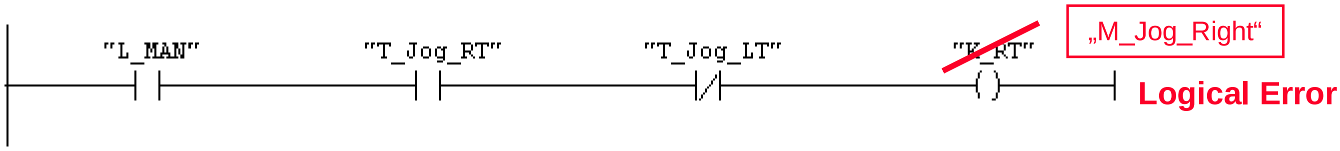

**Network 3** : AUTO Mode



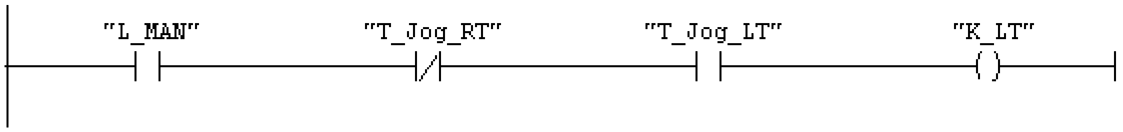
# Print out of the Error Program

FC16 : Conveyor Operation

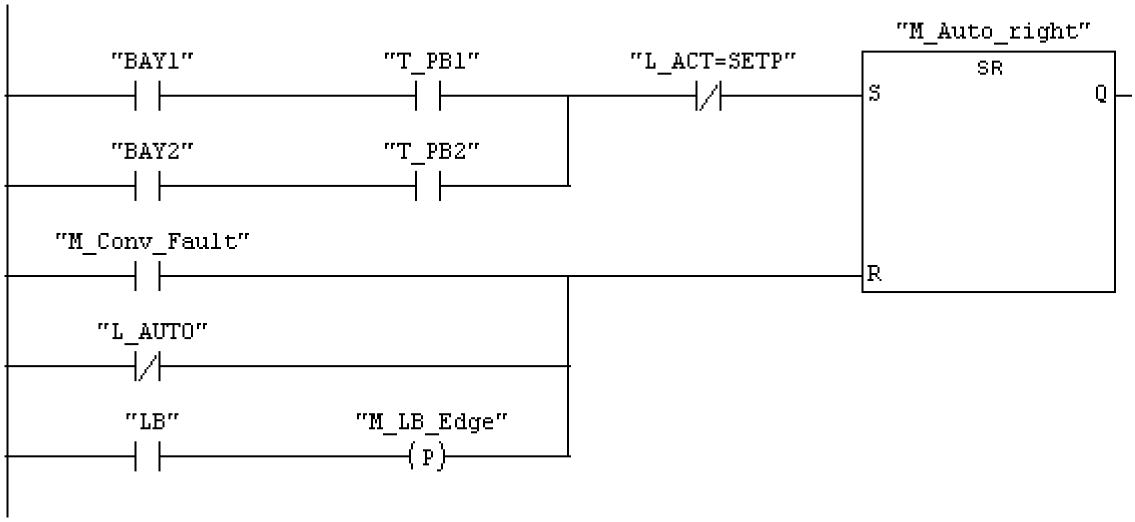
**Network 1**: Jog Conveyor RIGHT



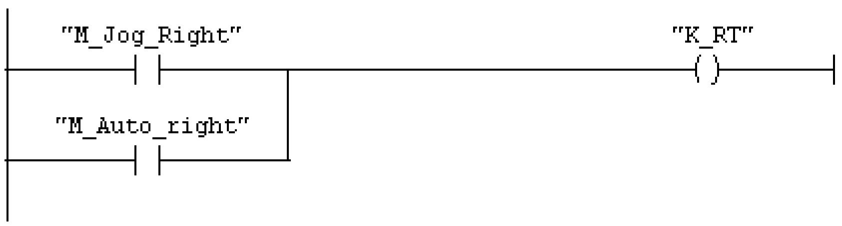

**Network 2**: Jog Conveyor LEFT




**Network 3**: Conveyor in AUTO Mode



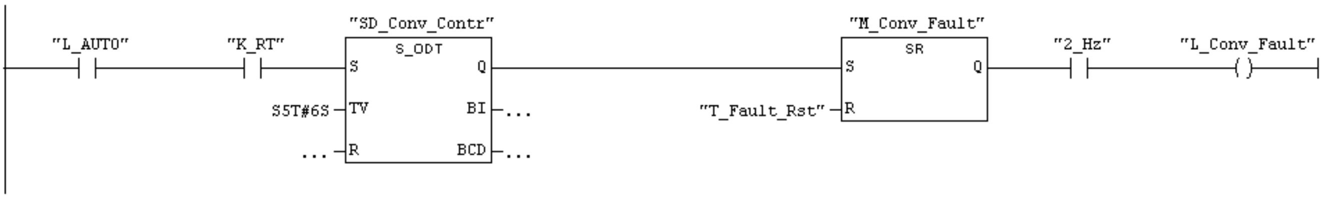

**Network 4**: Title:



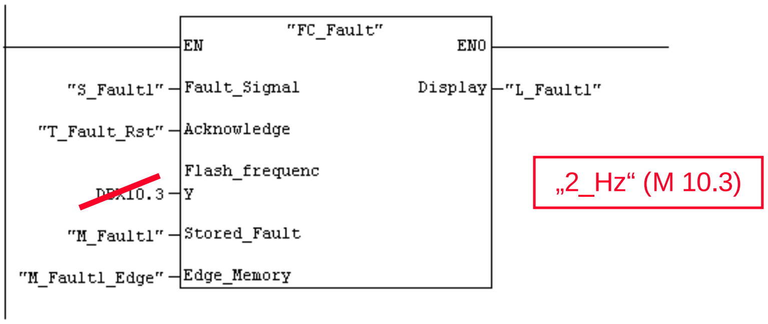
# Print out of the Error Program

FC17 : Title:

**Network 1 :** Conveyor Fault: Monitoring the transport function



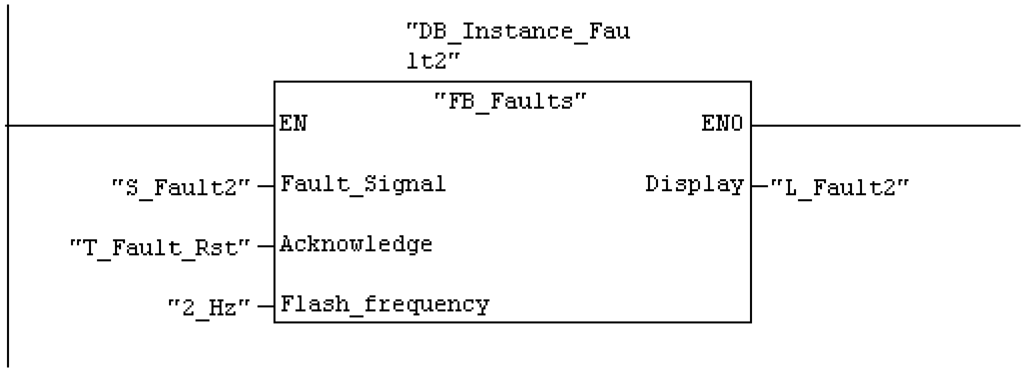

**Network 2 :** Evaluation of disturbance 1



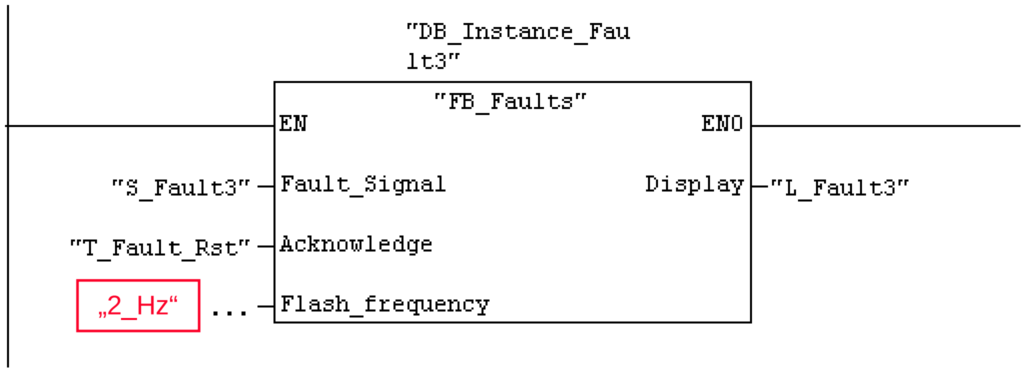

STOP Error

# Print out of the Error Program

**Network 3 :** Evaluation of disturbance 2




**Network 4 :** Evaluation of disturbance 3

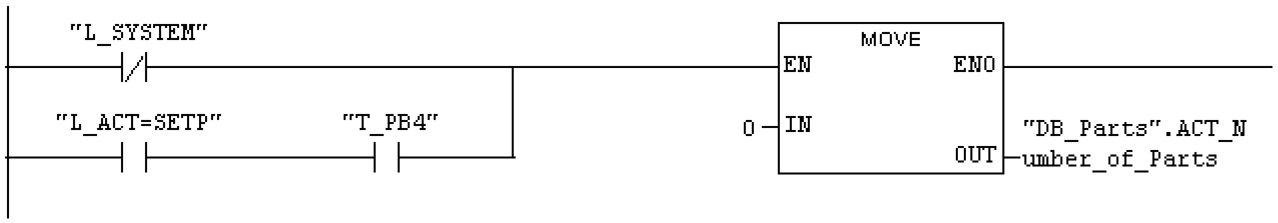



**Logical Error**

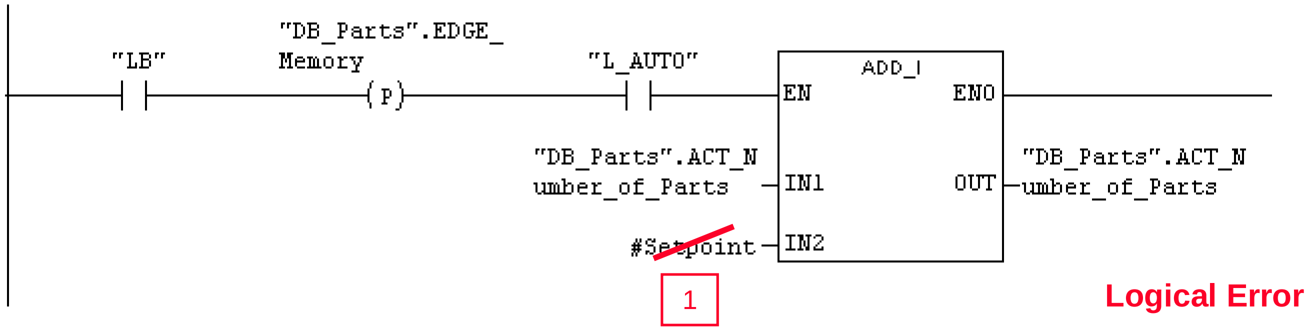
# Print out of the Error Program

FC18 : Title:

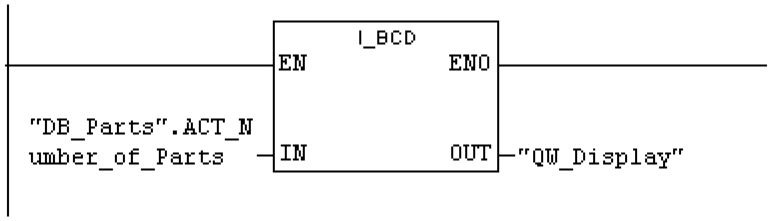
**Network 1**: Reset Counter



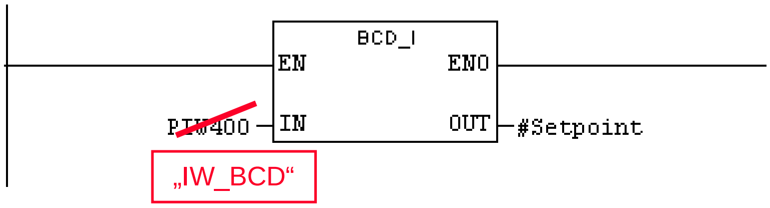

**Network 2**: Count transported Parts in AUTO Mode




**Network 3**: Display Actual Number of parts

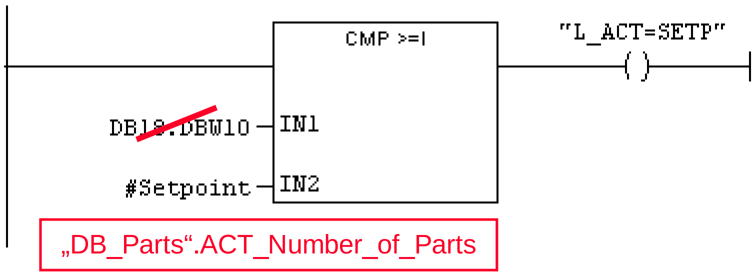



**Network 4**: Read Setpoint number of parts and convert




STOP Error

**Network 5**: SETPOINT-ACTUAL number of parts Comparison



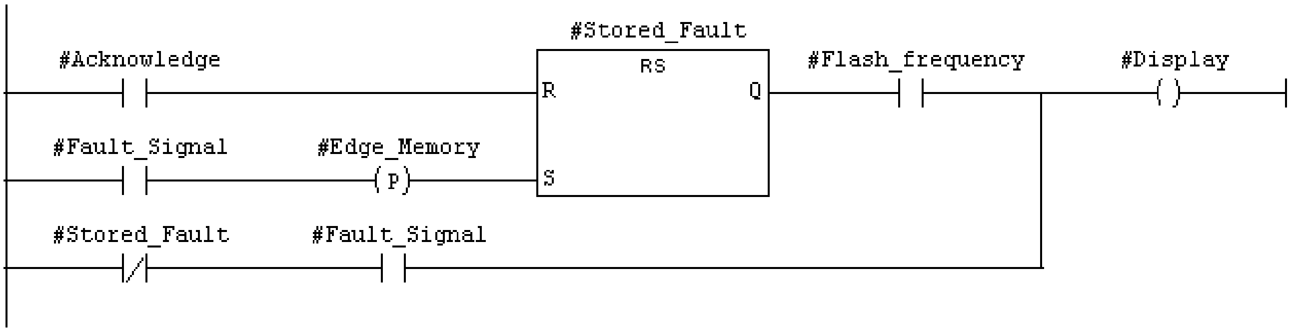

STOP Error

# Print out of the Error Program

| Address | Declaration | Name | Type | Initial value | Comment |
|---|---|---|---|---|---|
| 0.0 | in | Fault_Signal | BOOL | | |
| 0.1 | in | Acknowledge | BOOL | | |
| 0.2 | in | Flash_frequency | BOOL | | |
| 2.0 | out | Display | BOOL | | |
| 4.0 | in_out | Stored_Fault | BOOL | | |
| 4.1 | in_out | Edge_Memory | BOOL | | |
| | temp | | | | |

FC20 : Function for Fault Signals

**Network 1**: Fault Logic

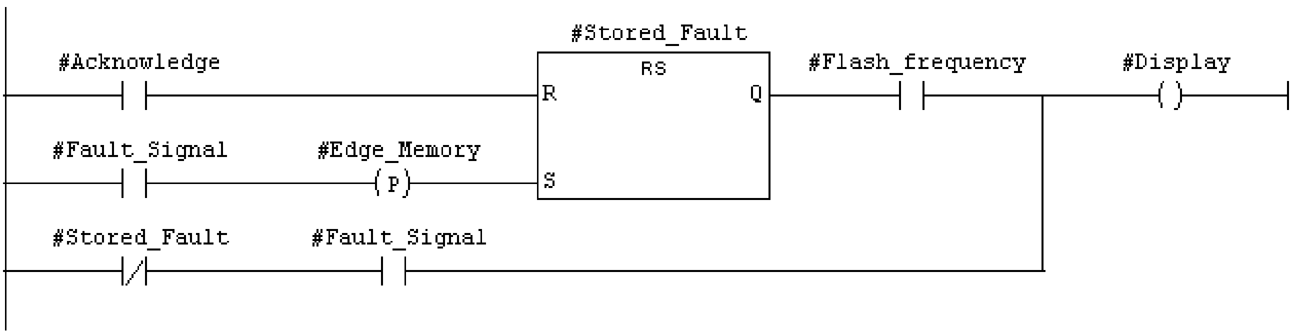


# Print out of the Error Program

| Address | Declaration | Name | Type | Initial val | Comment |
|---|---|---|---|---|---|
| 0.0 | in | Fault_Signal | BOOL | FALSE | |
| 0.1 | in | Acknowledge | BOOL | FALSE | |
| 0.2 | in | Flash_frequency | BOOL | FALSE | |
| 2.0 | out | Display | BOOL | FALSE | |
| | in_out | | | | |
| 4.0 | stat | Stored_Fault | BOOL | FALSE | |
| 4.1 | stat | Edge_Memory | BOOL | FALSE | |
| | temp | | | | |

FB20 : Function for Fault Signals

**Network 1**: Fault Logic

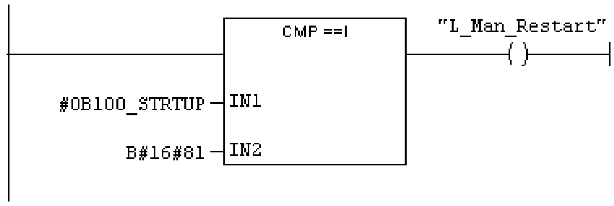


# Determing type of Startup (OB 100)

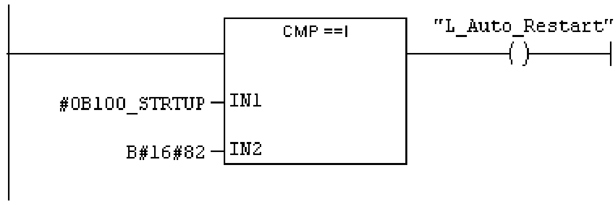
| Address | Declaration | Name | Type | Initial val | Comment |
|---|---|---|---|---|---|
| 0.0 | temp | OB100_EV_CLASS | BYTE | | 16#13, Event class 1, Entering event state, Ev |
| 1.0 | temp | OB100_STRTUP | BYTE | | 16#81/82/83/84 Method of startup |
| 2.0 | temp | OB100_PRIORITY | BYTE | | 27 (Priority of 1 is lowest) |
| 3.0 | temp | OB100_OB_NUMBR | BYTE | | 100 (Organization block 100, OB100) |
| 4.0 | temp | OB100_RESERVED_1 | BYTE | | Reserved for system |
| 5.0 | temp | OB100_RESERVED_2 | BYTE | | Reserved for system |
| 6.0 | temp | OB100_STOP | WORD | | Event that caused CPU to stop (16#4xxx) |
| 8.0 | temp | OB100_STRT_INFO | DWORD | | Information on how system started |
| 12.0 | temp | OB100_DATE_TIME | DATE_AND_TIME | | Date and time OB100 started |

OB100 : "Complete Restart"

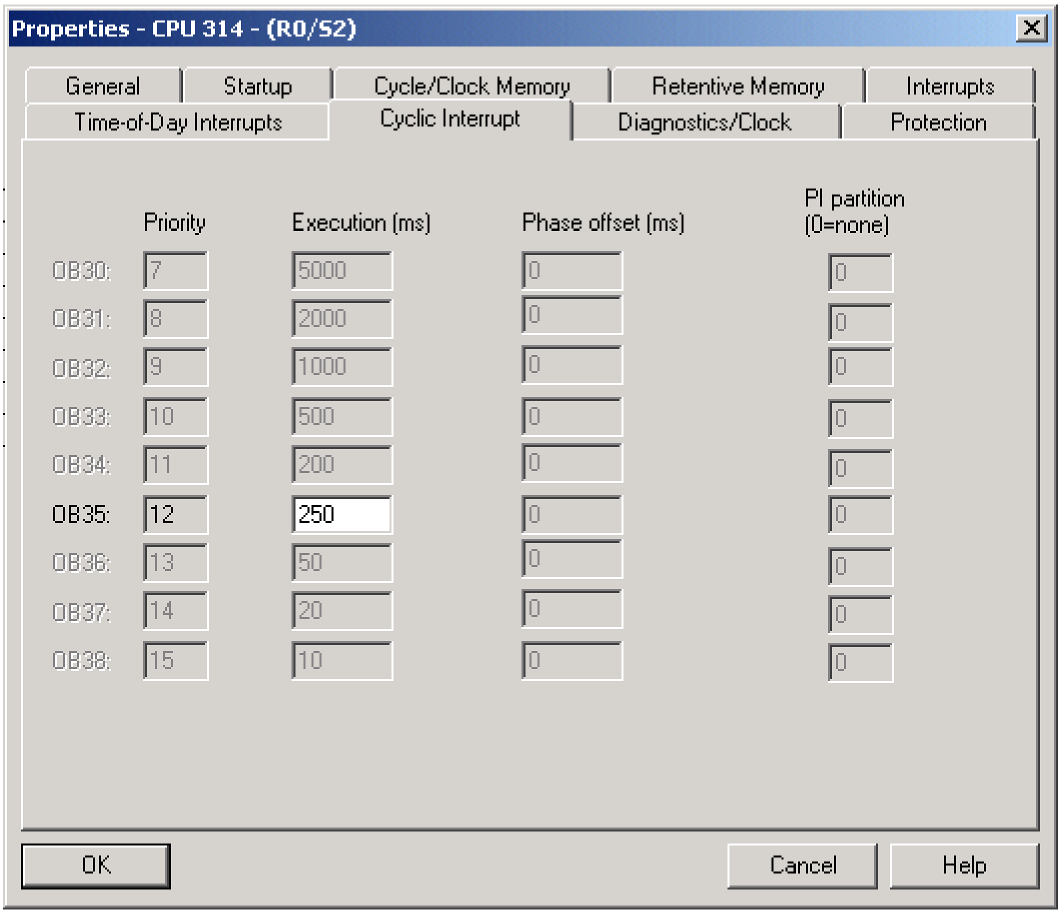
**Network 1**: Display manual restart




**Network 2**: Display automatic restart



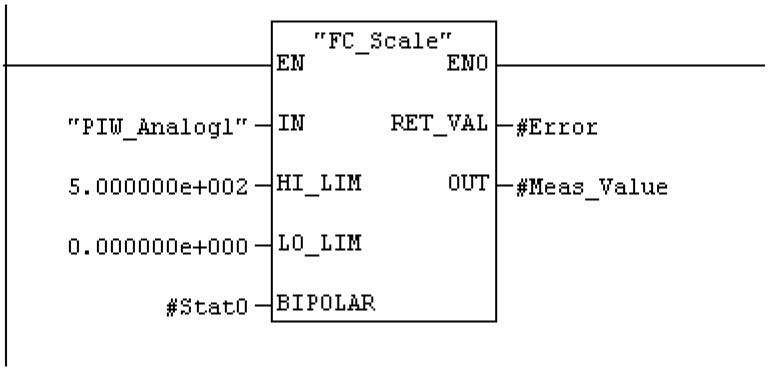
## Recording and Displaying the Weight of the Transported Parts
## CPU Properties: Cyclic Interrupt OB 35



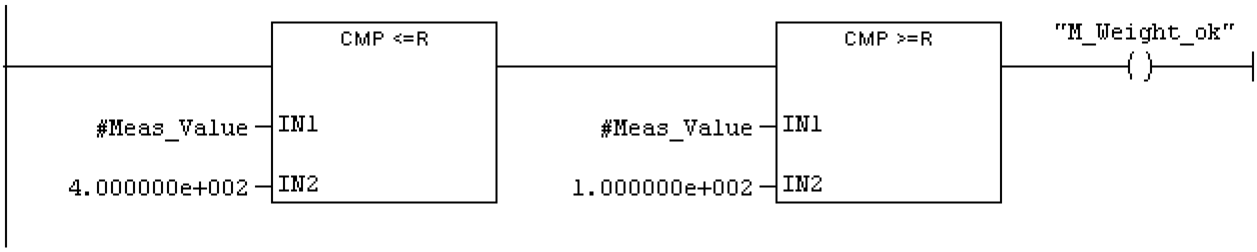
# Recording and Displaying the Weight of the Transported Parts
# Program in OB 35

```
OB35 :  "Cyclic Interrupt"
Network 1: Title:
      CLR
      =      #Stat0
```

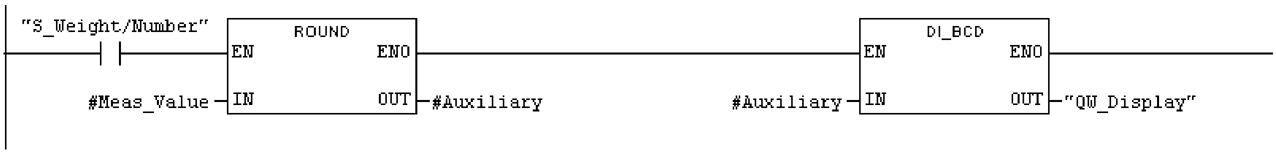
**Network 2 :** Read Analog Value and scale for Weight




**Network 3 :** Weight Control

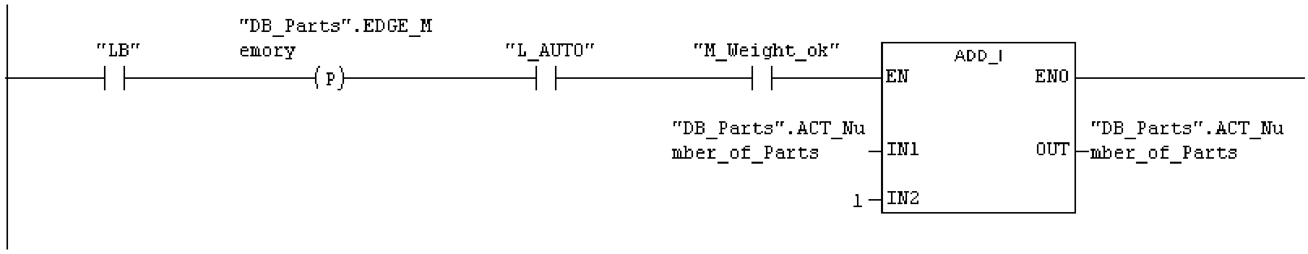



**Network 4 :** Display Analog Value (Weight)

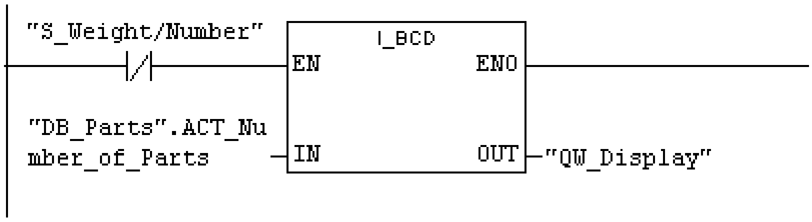


## Recording and Displaying the Weight of the Transported Parts
## Display Number of Parts in FC 18

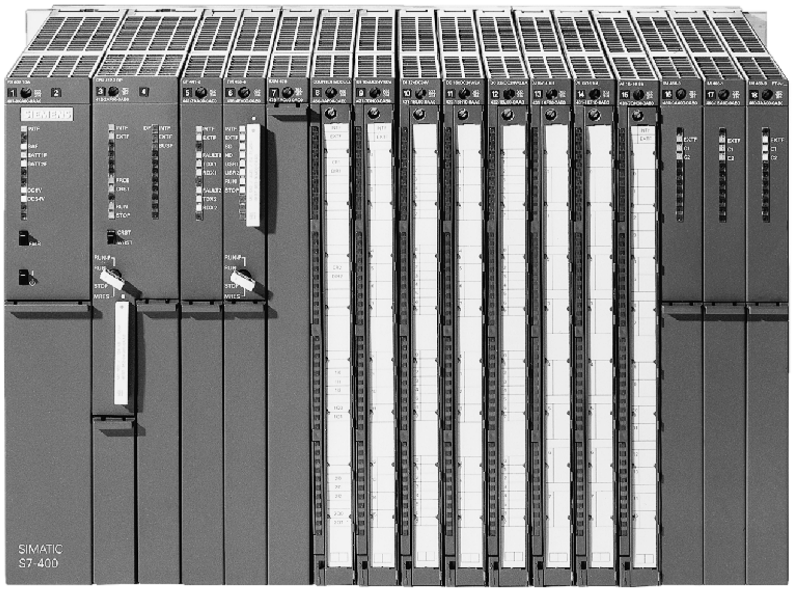
Network 2: Count transported Parts in AUTO Mode




Network 5 : Display Actual Number of parts



# Appendix: Technical Specifications and Special Features of the S7-400™




SIMATIC® S7


Date: 12.03.03
File: PRO1_17E.1

**SITRAIN** Training for Automation and Drives

**Contents** Page

## Technical Specifications of the S7-300™ CPUs (1)

| CPU | 312 IFM | 313 | 314 | 314 IFM | 315 | 315-2 DP | 316-2 DP | 318-2 DP |
|---|---|---|---|---|---|---|---|---|
| Execution time in µs | | | | | | | | |
| Bit instruction | 0.6 - 1.2 | 0.6 - 1.2 | 0.3 - 0.6 | 0.3 - 0.6 | 0.3 - 0.6 | 0.3 - 0.6 | 0.3 - 0.6 | 0.1 |
| Word instruction | 2.0 | 2.0 | 1.2 | 1.2 | 1.0 | 1.0 | 1.0 | 0.1 |
| Integer (+/-) | 3.0 | 3.0 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 | 0.1 |
| Real (+/-) | 60.0 | 60.0 | 50.0 | 50.0 | 50.0 | 50.0 | 50.0 | 0.6 |
| User memory | | | | | | | | |
| Work memory | 6 KB | 12 KB | 24 KB | 32 KB | 48 KB | 64 KB | 128 KB | 512 KB |
| Load memory integr. | 20 KB | 20 KB | 40 KB | 48 KB | 80 KB | 96 KB | 192 KB | 64 KB |
| Load memory extern | - | 4 MB | 4 MB | (4 MB) | 4 MB | 4 MB | 4 MB | 4 MB |
| Addresses | | | | | | | | |
| Bit memories | 1024 | 2048 | 2048 | 2048 | 2048 | 2048 | 2048 | 8192 |
| Clock memories | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 |
| Timers | 64 | 128 | 128 | 128 | 128 | 128 | 128 | 512 |
| Counters | 32 | 64 | 64 | 64 | 64 | 64 | 64 | 512 |
| Block types/Number | | | | | | | | |
| FBs | 32 | 128 | 128 | 128 | 192 | 192 | 256 | 1024 |
| FCs | 32 | 128 | 128 | 128 | 192 | 192 | 512 | 1024 |
| DB's | 63 | 127 | 127 | 127 | 255 | 255 | 511 | 2047 |
| Size of process image I/O in bytes | 32 each | 128 each | 128 each | 124 each | 128 each | 128 each | 128 each | 256 each (2048) |
| Maximum I/O address area in bytes | 32 each | 32 each | 768 each | 752 each | 768 each | 1024 each | 1024 each | 8192 each |
| Interfaces | MPI | MPI | MPI | MPI | MPI | MPI, DP | MPI, DP | MPI/DP, DP |

SIMATIC® S7


Date: 12.03.03
File: PRO1_17E.2

SITRAIN Training for Automation and Drives

**Introduction**

In order to be able to rate the technical specifications of the S7-400™, you can first of all see the specifications of the S7-300™. They are current as of April 2000. For the most current technical specifications, please refer to the ST 70 catalog.

# Technical Specifications of the S7-300™ CPUs (2)

| CPU | 312 IFM | 313 | 314 | 314 IFM | 315 | 315-2 DP | 316-2 DP | 318-2 DP |
|---|---|---|---|---|---|---|---|---|
| Organization blocks | OB No. | OB No. | OB No. | OB No. | OB No. | OB No. | OB No. | OB No. |
| Free cycle | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Time-of-day interrupts | - | 10 | 10 | 10 | 10 | 10 | 10 | 10,11 |
| Time-delay interrupts | - | 20 | 20 | 20 | 20 | 20 | 20 | 20,21 |
| Cyclic interrupts | - | 35 | 35 | 35 | 35 | 35 | 35 | 32,35 |
| Hardware interrupts | 40 | 40 | 40 | 40 | 40 | 40 | 40 | 40,41 |
| Background execution | - | - | - | - | - | - | - | 90 |
| Startup | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100,102 |
| Errors, asynchronous | - | 80-82, 85, 87 | 80-82, 85, 87 | 80-82, 85, 87 | 80-82, 85, 87 | 80-82, 85 87 | 80-82, 85 87 | 80-82, 85 87 |
| Errors, synchronous | - | 121,122 | 121,122 | 121,122 | 121,122 | 121,122 | 121,122 | 121,122 |
| Local data in bytes | 512 | 1536 | 1536 | 1536 | 1536 | 1536 | 1536 | 4096(8192) |
| Maximum block length | 8 KB | 8 KB | 8 KB | 8 KB | 16 KB | 16 KB | 16 KB | 64 KB |
| Block nesting depth per execution level | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 20 |
| Communication Maximum connections static/dynamic | 4/2 | 4/4 | 4/8 | 4/8 | 4/8 | 4/8 | 4/8 | 32 |
| Global data communi-cation with MPI: GD circles per CPU | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 8 |
| Send GD packets per GD circle | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Receive GD packets per GD circle | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 |
| Maximum user data size of a packet | 22 bytes | 22 bytes | 22 bytes | 22 bytes | 22 bytes | 22 bytes | 22 bytes | 54 bytes |

SIMATIC® S7


Date: 12.03.03
File: PRO1_17E.3

SITRAIN Training for Automation and Drives

# Technical Specifications of the S7-400™ CPUs (1)

| CPU | 412-1 | 412-2 | 414-2 | 414-3 | 416-2 | 416-3 | 417-4 | 417H |
|---|---|---|---|---|---|---|---|---|
| **Execution time in µs** | | | | | | | | |
| Bit instruction | 0.2 | 0.2 | 0.1 | 0.1 | 0.08 | 0.08 | 0.1 | 0.1 |
| Word instruction | 0.2 | 0.2 | 0.1 | 0.1 | 0.08 | 0.08 | 0.1 | 0.1 |
| Integer (+/-) | 0.2 | 0.2 | 0.1 | 0.1 | 0.08 | 0.08 | 0.1 | 0.1 |
| Real (+/-) | 0.6 | 0.6 | 0.6 | 0.6 | 0.48 | 0.48 | 0.6 | 0.6 |
| **User memory** | | | | | | | | |
| Work memory int. | 2x48 KB | 2x48 KB | 2x128 KB | 2x 384 KB | 2x 0.8 MB | 2x 1.6 MB | 2x2 MB | 2x2 MB |
| Load memory integr. | 256 KB | 256 KB | 256 KB | 256 KB | 256 KB | 256 KB | 256 KB | 256 KB |
| Load memory extern | 64 MB | 64 MB | 64 MB | 64 MB | 64 MB | 64 MB | 64 MB | 64 MB |
| **Addresses** | | | | | | | | |
| Byte memories | 4 K | 4 K | 8 K | 8 K | 16 K | 16 K | 16 K | 16 K |
| Clock memories | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 |
| Timers | 256 | 256 | 256 | 256 | 512 | 512 | 512 | 512 |
| Counters | 256 | 256 | 256 | 256 | 512 | 512 | 512 | 512 |
| **Block types/Number** | | | | | | | | |
| FBs | 256 | 256 | 1024 | 1024 | 2048 | 2048 | 6144 | 6144 |
| FCs | 256 | 256 | 1024 | 1024 | 2048 | 2048 | 6144 | 6144 |
| DBs | 511 | 511 | 1023 | 1023 | 4096 | 4096 | 8191 | 8191 |
| Size of process image I/O in bytes | 4 K each | 4 K each | 8 K each | 8 K each | 16 K each | 16 K each | 16 K each | 16 K each |
| Maximum I/O address area in bytes | 4 K each | 4 K each | 8K each | 8 K each | 16 K each | 16 K each | 16 K each | 16 K each |
| Interfaces | MPI/DP | MPI/DP DP | MPI/DP DP | MPI/DP 2xDP | MPI/DP DP | MPI/DP 2x DP | MPI/DP 3x DP | MPI/DP DP |

SIMATIC® S7


Date: 12.03.03
File: PRO1_17E.4

**SITRAIN** Training for Automation and Drives

**CPU Types**

CPUs are available with the appropriate execution times, sufficient work memory capacity and a suitable number of blocks for every performance range.

**Process I/O**

The logical addresses of the I/O modules are all in a linear address area of appropriate size.

The addresses of the slave stations connected to the integral DP interface are also mapped in this linear address area. This enables distributed I/Os to be accessed in the same way as central I/Os in the user program.

The address parameters for both central and distributed I/Os are assigned with STEP 7.

## Technical Specifications of the S7-400™ CPUs (2)

| CPU | 412-1 | 412-2 | 414-2 | 414-3 | 416-2 | 416-3 | 417-4 | 417H |
|---|---|---|---|---|---|---|---|---|
| Organization blocks | OB No. | OB No. | OB No. | OB No. | OB No. | OB No. | OB No. | OB No. |
| Free cycle | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Time-of-day interrups | 10,11 | 10,11 | 10-13 | 10-13 | 10-17 | 10-17 | 10-17 | 10-17 |
| Time-delay interrupts | 20,21 | 20,21 | 20-23 | 20-23 | 20-23 | 20-23 | 20-23 | 20-23 |
| Cyclic interrupts | 32,35 | 32,35 | 32-35 | 32-35 | 30-38 | 30-38 | 30-38 | 30-38 |
| Hardware interrupts | 40,41 | 40,41 | 40-43 | 40-43 | 40-47 | 40-47 | 40-47 | 40-47 |
| Multicomputing | 60 | 60 | 60 | 60 | 60 | 60 | 60 | 60 |
| Background execution | 90 | 90 | 90 | 90 | 90 | 90 | 90 | 90 |
| Startup | 100-102 | 100-101 | 100-102 | 100-102 | 100-102 | 100-102 | 100-102 | 100,102 |
| Errors, asynchronous | 80-87 | 80-87 | 80-87 | 80-87 | 80-87 | 80-87 | 80-87 | 80-87 |
| Errors, synchronous | 121,122 | 121,122 | 121,122 | 121,122 | 121,122 | 121,122 | 121,122 | 121,122 |
| Local data in bytes | 4 KB | 4 KB | 8 KB | 8 KB | 16 KB | 16 KB | 32 KB | 32 KB |
| Maximum block length | 48 KB | 64 KB | 64 KB | 64 KB | 64 KB | 64 KB | 64 KB | 64 KB |
| Block nesting depth per execution level | 16 | 16 | 16 | 16 | 16 | 16 | 16 | 24 |
| Communication Maximum connections static/dynamic | 16 | 16 | 32 | 32 | 64 | 64 | 64 | 64 |
| Global data communi-cation with MPI: GD circles per CPU | 8 | 8 | 8 | 8 | 16 | 16 | 16 | 16 |
| Send GD packets per GD circle | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Receive GD packets per GD circle | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| Maximum user data size of a packet | 54 bytes | 54 bytes | 54 bytes | 54 bytes | 54 bytes | 54 bytes | 54 bytes | 54 bytes |

SIMATIC® S7


Date: 12.03.03
File: PRO1_17E.5

**SITRAIN** Training for Automation and Drives

**Communication**

The S7-400™ offers a variety of facilities for communication:

1. Integral Multi-Point-Interface (MPI), for connection of PGs/PCs, HMI systems, M7-300/400™ systems and other S7-300/400™ systems as active nodes.
2. Integral PROFIBUS-DP interfaces on CPUs 413-2/414-2/416-2/417-4 for connection of distributed I/O stations (such as ET200) to the CPU.
3. Communication processors such as CP443, for connection to the PROFIBUS and Industrial Ethernet bus systems.
4. Communication processors such as CP441, for powerful point-to-point (PtP) communication to other S7 or S5 PLCs or PLCs from other manufacturers.

**S7 Functions**

There are two types of S7 communication functions:

S7 basic communication: These services can be used for exchanging small quantities of data (up to 76 bytes) between communication partners (S7-300/400™) with MPI or within a station (or to intelligent slaves with PROFIBUS-DP).

The necessary communication SFCs are integrated in the operating system. You don't need to configure the connections. You assign the communication resources and specify the address of the communication partner direct in the SFC call.

S7 extended communication: These services enable larger quantities of data (up to 64 KBytes) to be exchanged on any network (MPI, Profibus or Industrial Ethernet).

The necessary SFBs are integrated in the operating system of the S7-400™ (not S7-300™, S7-300™ as server only). SFBs need configured connections when called. Configured connections are established in accordance with the connection table on power up, and the relevant resources are assigned statically.

## Main Differences between the S7-400™ and the S7-300™

- Larger memory and more I/Q/M/T/C
- Addresses of input/output modules selectable
- Can connect EUs from S5 and use S5 CP/IP modules
- More system functions, such as programmed block communication
- Block size up to 64KB and twice as many DBs
- Complete restart and restart
- Setpoint/actual comparison of configuration on startup
- Modules can be removed without disconnecting the power supply
- Several part process images
- Priorities of OBs are parameter-assignable
- Several OBs for cyclic, hardware and time-of-day interrupts
- Block nesting up to 16 levels
- Size of L Stack selectable for each execution level
- 4 accumulators
- Multicomputimg

SIMATIC® S7


Date: 12.03.03
File: PRO1_17E.6

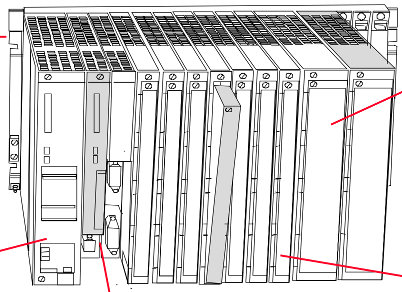
SITRAIN Training for Automation and Drives

**Differences** The main differences between the S7-400™ and the S7-300™, with which you have been working in this course, are listed above.

## S7- 400™ Components

**Racks**

- Universal rack (UR 1/2) for use as CR and ER
- Universal rack for H systems
- Extension rack (ER 1/2)
- Segmented central rack (CR2)



**Communication processors (CP)**

- Point-to-Point
  CP441-1, 1- channel
  CP441-2, 2- channel
- Networks
  Profibus
  Ethernet

**Power supplies (PS)**

- AC 120V/230V, 4A/10A/20A
- DC 24V, 4A/10A/20A

**Signal modules (SM)**

- DI/DO 32,16,8 channels
- DO wit h relay output
- AI/AO  16,8 channels
- diagnostics-capable modules

**CPUs**

CPU 412-1, 412-2
CPU 414-2, 414-3,
CPU 416-2, 416-3
CPU 417-4, 417H

**Function modules (FM)**

- Counting
- Positioning
- Closed-loop control

SIMATIC® S7


Date: 12.03.03
File: PRO1_17E.7

SITRAIN Training for Automation and Drives

**Racks**

The following racks are available for S7-400™ PLCs.

- UR1/UR2 are universal racks and can be used as either central racks or expansion racks. They have 18/9 single-width slots with P and K bus.
- ER1/ER2 are expansion racks without a K bus.
- CR2 is a segmented central rack for asymmetrical multicomputing.

**S7-CPUs**

The S7-400™ CPUs are upward compatible for all STEP 7 user programs. There are two versions: single-width and double-width with integrated DP master interface.

The integrated DP interface enables up to 125 DP slave stations to be addressed. The maximum transmission rate is 12 Mbps.

**FMs**

The FMs for positioning, closed-loop control and counting replace the S5-IP range.

**IMs**

Interface modules can be used for connecting SIMATIC® S7 and SIMATIC® S5 expansion racks to an S7-400™ central rack.

**CPs**

CP modules enable a CPU to be hooked up to the following networks:

- Industrial Ethernet (CP 443-1 and CP 444)
- PROFIBUS (CP 443-5)
- Point-to-Point network (CP441-1 and CP441-2).

Each CPU also has an MPI interface for connection to an MPI network. Up to 32 nodes can be connected to an MPI network.

## S7 - 400™ Racks

| Type of Rack | | Usable in: Central rack | Usable in: Expansion rack |
|---|---|---|---|
| UR1 / UR2 (Universal Rack) | P bus<br>K bus | Yes | Yes |
| CR2 (Central Rack) | P bus, Segment 1 / P bus, Segment 2<br>K bus | Yes | No |
| ER1 / ER2 (Expansion Rack) | P bus | No | Yes |




**UR 1 / UR 2**

UR1/UR2 can be used both as central and as expansion rack. They have a parallel peripheral bus (P bus) for the high-speed exchange of I/O signals (1.5 microsec./ byte) and the time critical access of the signal module process data.

In addition, UR1 (18 slots) / UR2 (9 slots) have a serial, powerful communication bus (K bus) for high-speed data exchange (10.5 Mbps) between K bus stations (S7/M7 CPUs, FMs, CPs, ).

By separating the P BUS and K BUS, each task is assigned its own bus system. Control and communication have their own separate "data highways". That way, the communication tasks do not slow down the control tasks.

**CR2**

The segmented CR 2 rack features an I/O bus divided into two segments with 10 and 8 slots. One CPU can be used for each segment. Both CPUs are respectively master for their P bus segment and can access only their own SMs.

Operating mode transitions are not synchronized, that is, the CPUs can be in different operating modes. Both CPUs can communicate through the continuous K bus.

**Why CR2?**

All CPUs (max. 4) have the same operating mode in symmetrical multicomputing, such as STOP. That is, the operating mode transitions are synchronized.

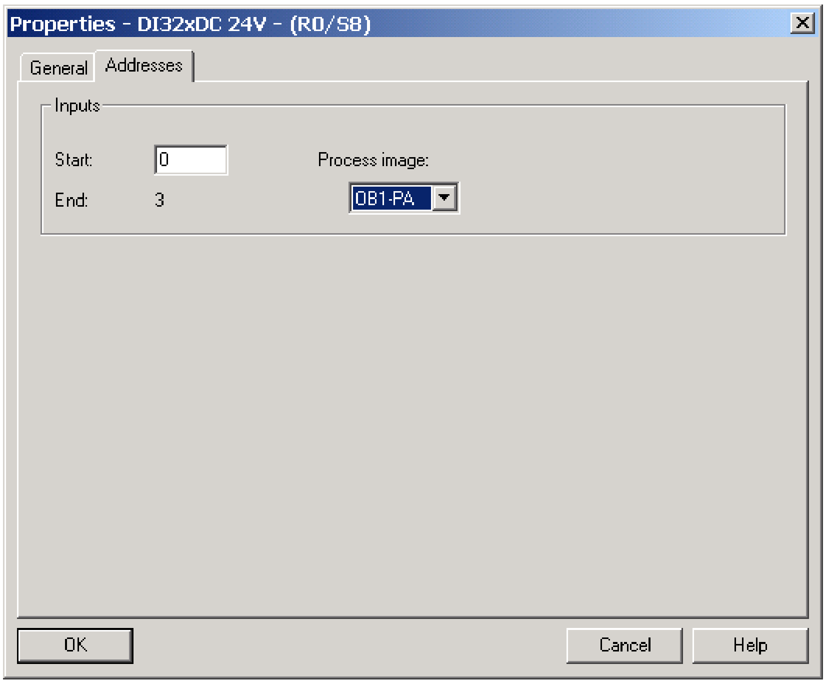
**ER 1 / ER 2**

ER1 (18 slots) / ER2 (9 slots) have no K bus, no interrupt lines, no 24 V power supply for the modules and no battery power supply.

**No Slot Rules**

**Exception:** PS on the far left and Receive IM in the ER on the far right!

## Module Parameters: Logical Addresses




SIMATIC® S7



Date: 12.03.03
File: PRO1_17E.9

SITRAIN Training for Automation and Drives

**General**

The S7-400™ has default addresses for the I/O modules. These defaults remain active until a configuration is downloaded to the CPU.
The system generates these default addresses from the geographic addresses.

**Addresses**

The default settings correspond to the slot-dependent addressing of the S7-300™.

The address depends on the slot in which the module is inserted in the rack. It is calculated as follows:

- digital starting address = [(rack number) x 18 + slot no. -1] x 4
- analog starting address = [(rack number) x 18 + slot no. -1] x 64 + 512

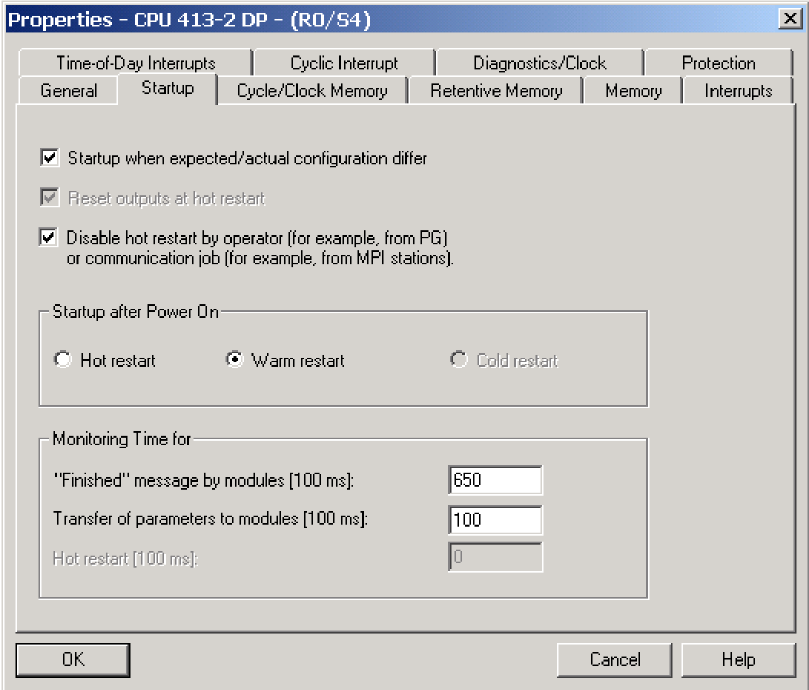
The rack number is set on the receive-IM (No. 1 to 21). The central rack always has the number 0.

Variable (slot-dependent) addresses of the I/O modules are established using the HW Config tool.

**Part Process Image**

In additon to the (full) process image (PII and PIQ), you can assign parameters for up to 8 part process images for an S7-400™ CPU (No. 1 to No. 8). You can update each part process image in the user program using SFCs. This means that you can deactivate cyclic updating of the process image and implement event-driven updating of the process image in the user program.

## CPU Parameters: Startup




SIMATIC® S7


Date: 12.03.03
File: PRO1_17E.10

SITRAIN Training for Automation and Drives

**Setpoint/Actual Difference** For specifying whether the CPU should stall start up if the actual I/O configuration differs from the setpoint (expected) configuration.

**Delete PIQ!!!** The process image output table is deleted in the first residual cycle on hot restart. Always select this if possible.

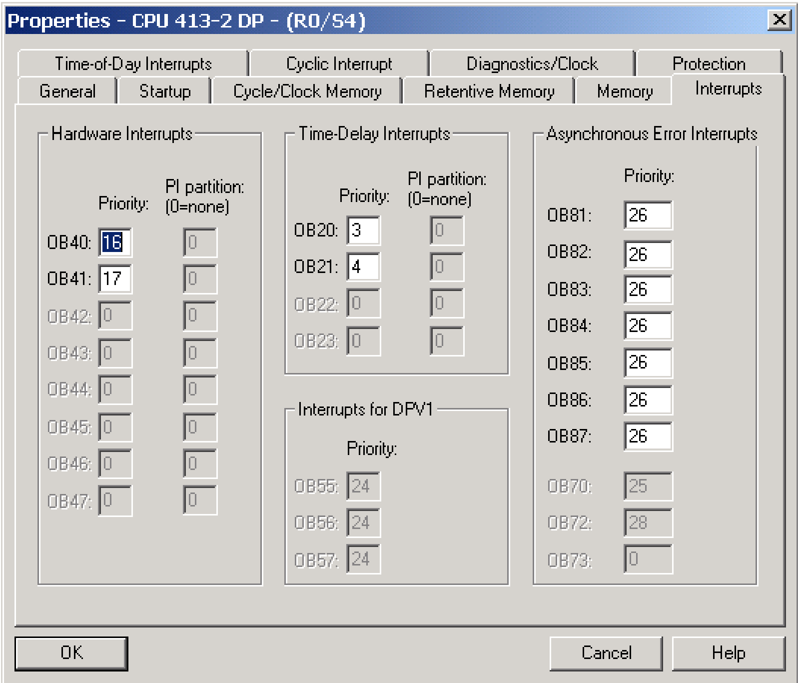
**Restarts** On Complete Restart (warm restart), the M/C/Ts are reset and the user program starts from the beginning.
On Restart (hot restart), the retentive M/C/Ts are not reset and execution of the user program resumes at the point of interruption.

**Actions** The operating system performs the following actions on startup:

- deletes stacks (CR)
- resets non-retentive bit memories, timers, counter (CR)
- resets process image output table PIQ (CR), takes action as instructed by parameter assignment (R)
- resets external output area (CR), takes action as instructed by parameter assignment (R)
- resets interrupts (CR/R) by means of OD
- updates system status list (CR/R)
- transfers configuration to modules (CR/R)

(CR= complete restart, R= restart).

## CPU Parameters: Interrupts




SIMATIC® S7


Date: 12.03.03
File: PRO1_17E.11

SITRAIN Training for Automation and Drives

**Hardware Interrupts** This parameter block is for setting the priorities of the hardware interrupt organization blocks. Permissible entries are 0 and the values from 2 to 24 (0 = deselect).
Priorities range from 1 to 24 and if two interrupts occur at the same time, the one with the higher priority is processed first.
There are 8 independent of one another hardware interrupts, each with its own organization block. You assign the interrupt OBs to the interrupt modules when assigning the I/O module parameters.

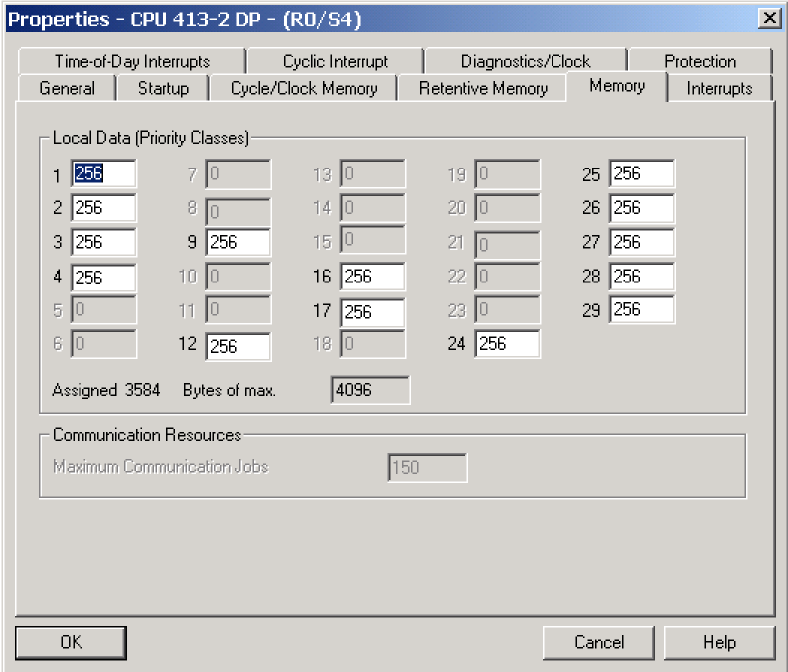
**Time-Delay Interrupts** A time-delay interrupt is a delayed one-time call of an organization block activated, for example, when a process signal is received.
In this parameter block of the Interrupts tab page, you can set the priorities of the time-delay interrupts. Permissible entries are 0 and values from 2 to 24 (0 = Deselect). Time-delay interrupts are handled by SFCs 32 to 34.

- SFC32 "SRT_DINT" = Start time-delay interrupt.
- SFC33 "CAN_DINT" = Cancel time-delay interrupt
- SFC34 "QRY_DINT" = Query status of time-delay interrupt

**Communication Interrupts (coming soon)** The arrival of communication data can be indicated by communication interrupts to enable the data received to be evaluated as quickly as possible.

- Global Data interrupt (OB50)
- SFB communication interrupt (OB51)

## CPU Parameters: Memory




SIMATIC® S7

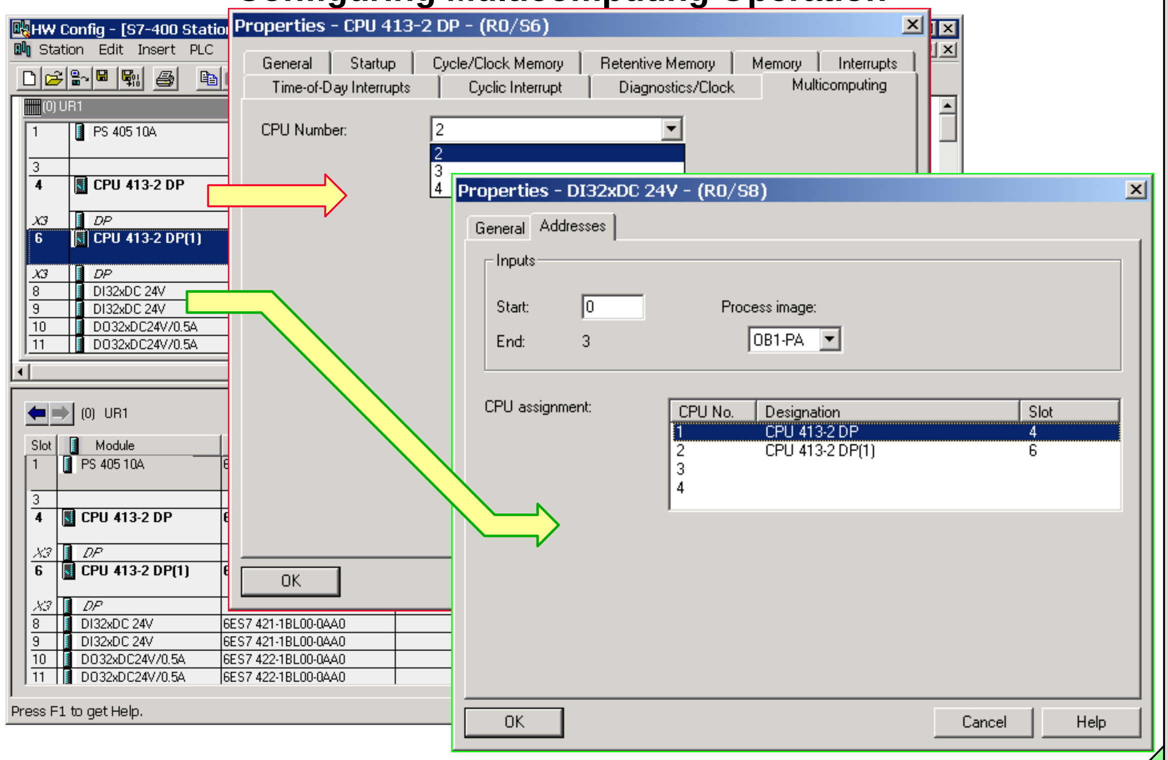

Date: 12.03.03
File: PRO1_17E.12

SITRAIN Training for Automation and Drives

**Local Data** The system reserves 256 bytes in the local data stack (default setting) for every execution level.
If the user program requires little or no local data in several levels, you can specify the local data requirements you want (scratchpad memory) per level (OB).

The maximum amount of local data depends on the type of CPU

## Configuring Multicomputing Operation







**Overview**

Multicomputing operation is the synchronous operation of several CPUs (two to four) in an S7-400™ central rack.

The CPUs startup together, if they have the same startup mode (complete restart or restart) and they also go into the STOP mode together.

**Setting Up Multicomputing**

You can set up a multicomputing operation by inserting several multicomputing-capable CPUs in a suitable rack. The infotext in the "Hardware Catalog" indicates whether a CPU is multicomputing-capable.

The CPUs participating in multicomputing, "share" a common address area, that is, the adress area of a module is always assigned to a specific CPU.

**What to Do**

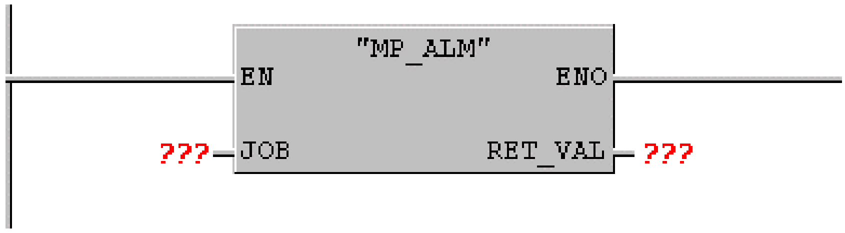
You can configure the multicomputing operation as follows:

1. Line up all the CPUs necessary for the multicomputing operation.
2. Double-click on the CPUs and adjust the CPU number in the "Multicomputing" tab.
3. To assign a module to a specific CPU, proceed as follows:
   - Arrange the modules in the rack.
   - Double-click the modules and select the "Addresses" tab.
   - In the "CPU No." field select the number of the CPU you want.
     For interrupt capable modules, the CPU assignment is displayed as the target CPU in the "Inputs" or "Outputs" tab.

You can make the modules that are assigned to a specific CPU stand out optically in the table by selecting the menu options *View -> Filter -> CPU No.x Modules*.

The parameter assignment data for a station are always downloaded into all participating CPUs; downloading into only one CPU is not possible. That way, inconsistent configurations are avoided.

## SFC 35 for Synchronization in Multicomputing Operation




| Parameter | Declaration | Data type | Memory | Description |
|---|---|---|---|---|
| JOB | INPUT | BYTE | I, Q, M, D, L, Const. | Task identifier (possible values: 1 to 15) |
| RET_VAL | OUTPUT | INT | I, Q, M, D, L | Return value (error code). |

SIMATIC® S7


Date: 12.03.03
File: PRO1_17E.14

**SITRAIN** Training for Automation and Drives

**Description**

The call of SFC 35 "MP_ALM" triggers the multicomputing interrupt. This leads to the synchronized start of OB60 on all relevant CPUs.

With single-processor operation and with operation in a segmented rack, OB 60 is only started on the CPU in which you called the SFC 35.

You can use the input parameter JOB to identify the cause for the multicomputing interrupt that you wanted. This task identifier is transferred to all relevant CPUs and you can evaluate it in OB 60.

You can call SFC 35 (MP_ALM) anywhere in your program. Since this call only makes sense in RUN mode, the multicomputing interrupt is suppressed when it is called in the STARTUP mode. A function value informs you of this.

**Error Code**

If an error occurs while the function is being executed, the return value receives an error code:

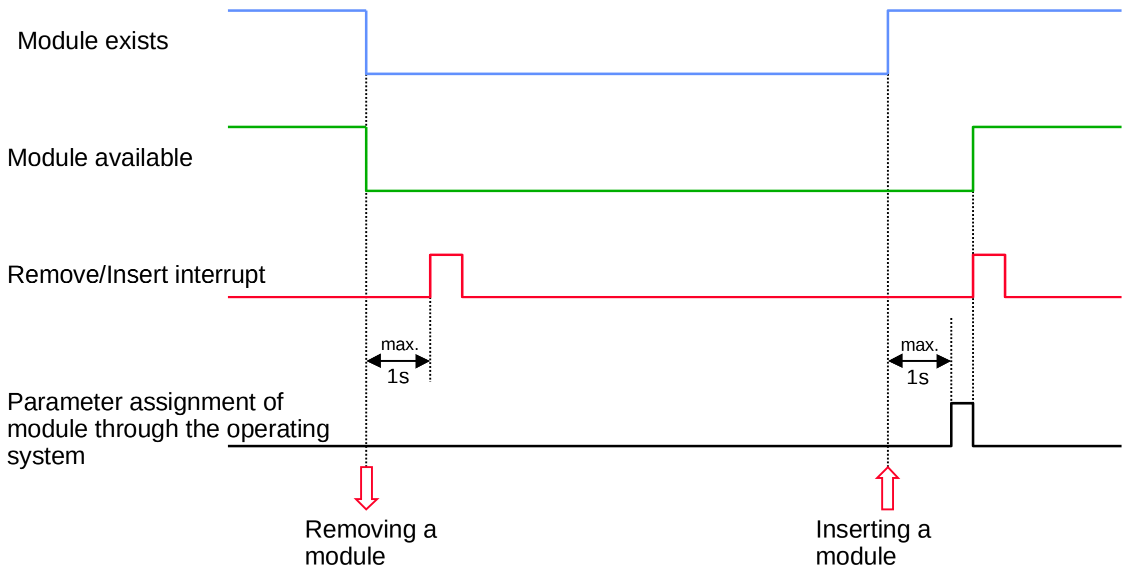
W#16#0000: No error has occurred.

W#16#8090: The input parameter JOB contains an invalid value.

W#16#80A0:The OB 60 execution of the preceeding multicomputing interrupt is not yet completed in its own or in another CPU.

W#16#80A1: Incorrect operating mode (STARTUP instead of RUN).

## Remove and Insert Interrupt




SIMATIC® S7


Date: 12.03.03
File: PRO1_17E.15

SITRAIN Training for Automation and Drives

**Remove and Insert Interrupt OB83**

In the S7-400™, it is possible to remove and to insert modules while powered up in RUN or in STOP mode. The exceptions to this are CPUs, PSs, S5 modules in adapter modules and IMs.

After removing a module in the RUN mode, you can - depending on the situation - call the following organization blocks from the CPU's operating system:

- OB 85-Process image update
- OB 122-I/O access error
- OB 83- Remove&Insert event.

You must take into consideration that OB 83 is only called after approximately 1sec., while the other OBs, as a rule, become active much sooner.

After you insert the module, it is checked by the CPU and - if no type error exists - it is assigned parameters. After a correct parameter assignment, the module is available for use.

If an error is recognized during parameter assignment, the diagnostic interrupt OB82 is automatically started.

**Start Information in OB83**

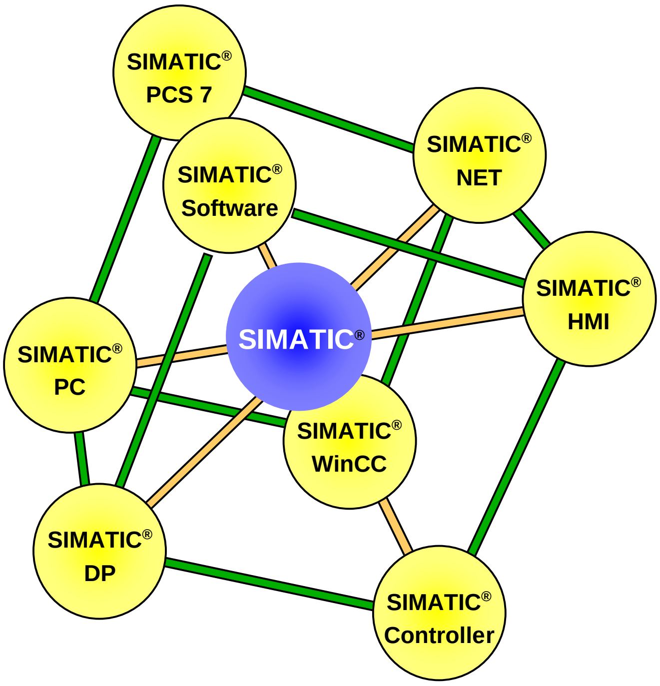
The following information exists in the local data of OB83:

- module removed or inserted
- logical address of the module
- type of module

**Replacement Value**

You can specify replacement values for the missing process signals of an input module by using a system function.

## Totally Integrated Automation

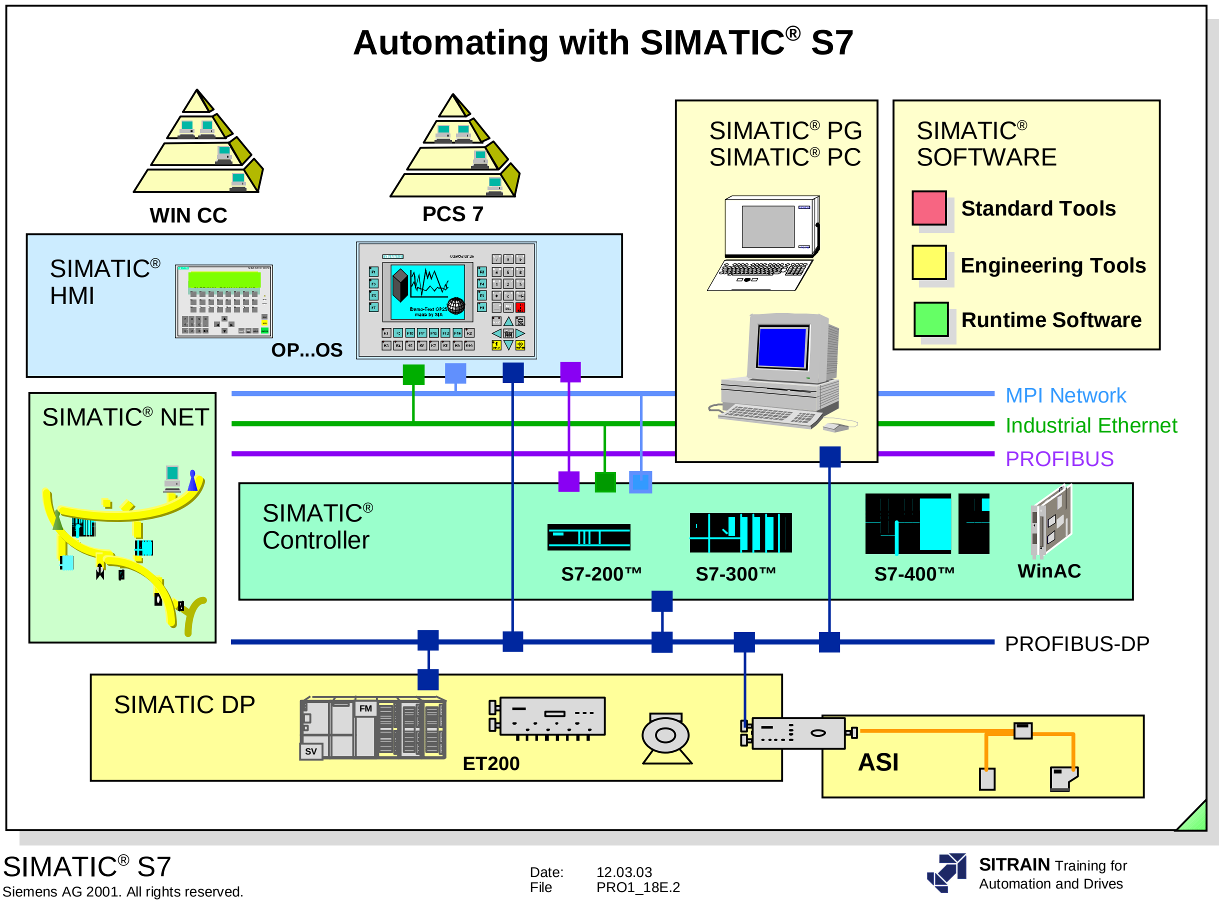



SIMATIC® S7


Date: 12.03.03
File PRO1_18E.1

SITRAIN Training for Automation and Drives

**Contents** Page




**Introduction**

In the past, the SIMATIC® product name was frequently used as a synonym for programmable logic controllers.

Today SIMATIC® has come to mean much more: SIMATIC® is the basic automation system for solving automation tasks in all industries. It consists of standard components in hardware and software, that offer a multitude of possibilities for customer-specific expansions.

Two factors have lead to this solution:

- the new, comprehensive SIMATIC® software, that has the optimal tool for every phase of an automation project and
- the members of the SIMATIC® automation family, that are more than just programmable logic controllers.

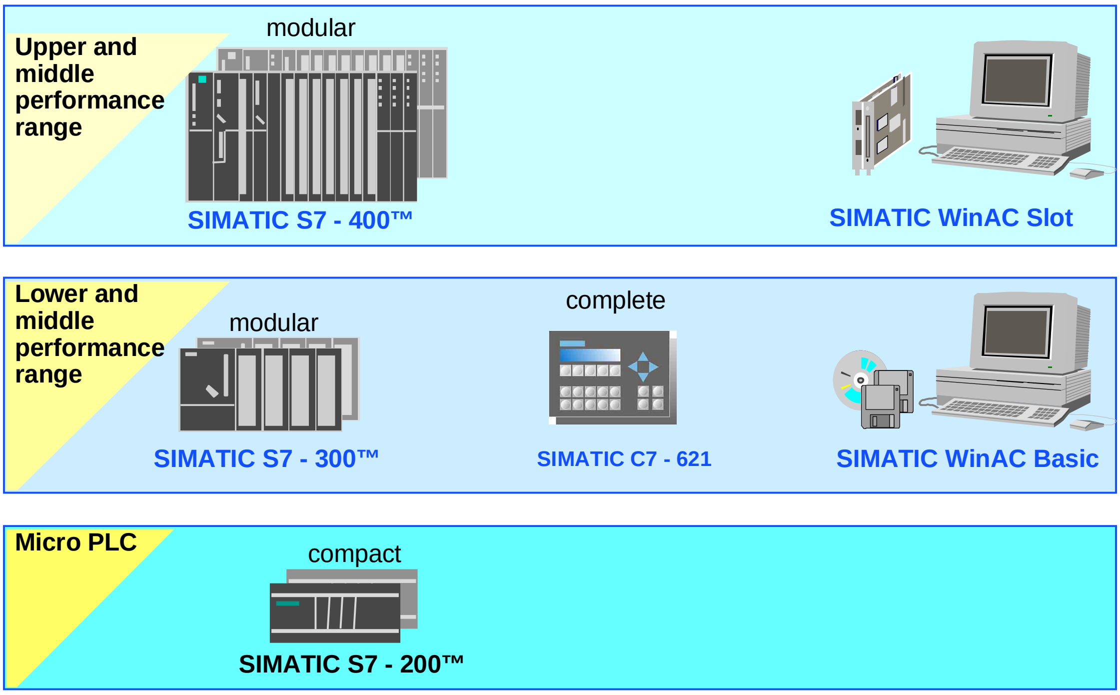
**TIA**

**T**otally **I**ntegrated **A**utomation is the new way of uniting production and process control technology.
All hardware and software components are thus united in a single system with the name **SIMATIC**®. This total integration is made possible by a threefold integraton:

- common data management (data are only entered once),
- common configuring and programming (modular software),
- common communication (simple and uniform configuration).

In the slide you can see the individual components of TIA.

## The SIMATIC® S7/C7 and WinAC Controllers




SIMATIC® S7

Date: 12.03.03
File PRO1_18E.3
SITRAIN Training for Automation and Drives

**SIMATIC S7**

The programmable logic controller family consists of the Micro PLC (S7-200™) performance range, the lower/middle performance range (S7-300™) and the middle/upper performance range (S7-400™).

**SIMATIC C7**

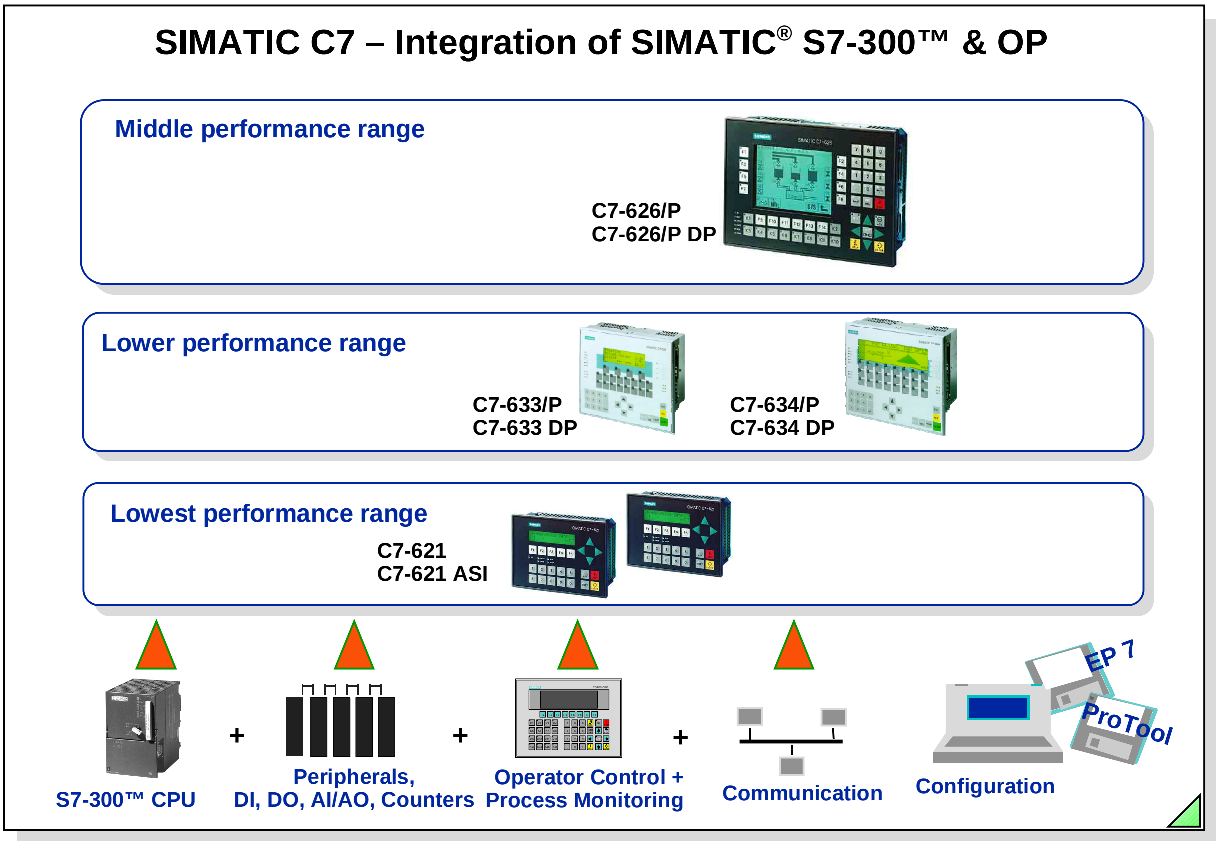
This complete system is the combination of a PLC (S7-300™) and an operator panel of the HMI operator control and process monitoring system. The integration of programmable logic controller and operator panel in one device makes complete machine controls in the smallest space and at an economical price possible.

**WinAC**

WinAC is a PC-based solution. It is used when various automation tasks (control, visualization, data processing) are to be solved with a PC.

There are 3 different products:

- WinAC Basic as software solution (PLC as Windows NT-Task),
- WinAC Slot as hardware solution (PLC as PC card)




SIMATIC® S7


Date: 12.03.03
File PRO1_18E.4

SITRAIN Training for Automation and Drives

**Overview**

The SIMATIC® C7 complete system family represents a system integration of PLC, peripherals and Operator Panel. This concept makes complete machine controls in the smallest space and at an economical price possible.

The C7- 633 DP, C7-634-DP and C7-626 DP complete systems also have an integrated PROFIBUS-DP connection.

**Complete Devices**

The new SIMATIC® C7-620 complete system now includes the six complete devices - SIMATIC® C7-623, C7-633, C7-624, C7-634 and C7-626, as well as the two space savers C7-621 and C7-621 ASI. All devices are positioned in the lower and lowest performance range of the SIMATIC® C7 complete system family.

**Customer-specific Expansions**

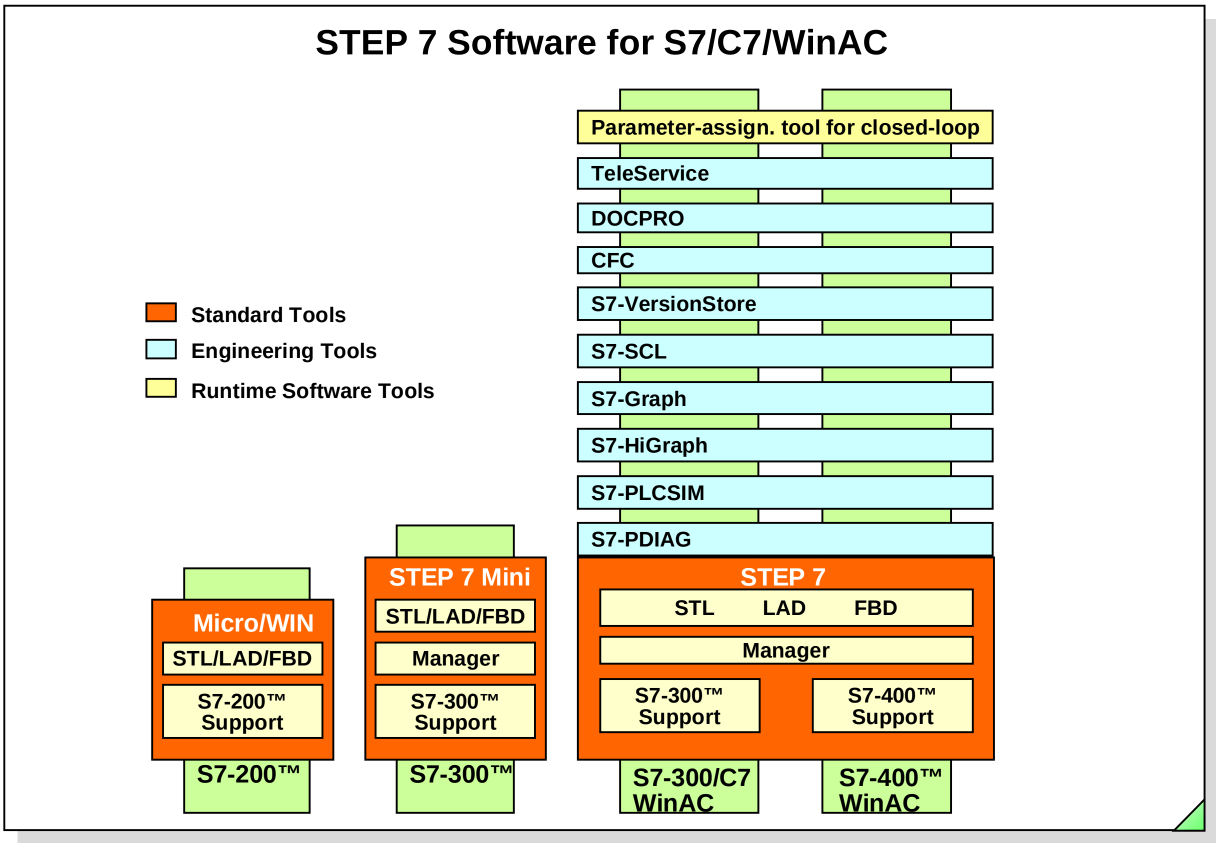
When you make an expansion with a customer-specified module, the module is installed directly on the complete device and a bus connection is established.

A customer-specified module can be connected to the SIMATIC® C7-620 when you have special requirements that cannot be covered with standard modules.

**Programming and Configuring**

STEP 7/STEP 7-Mini, the HMI configuring with ProTool/ Lite are used for programming and configuration of the system‘s hardware configuration.

The SIMATIC® C7-623/633 and the SIMATIC® C7-624/634 can be selected with ProTool/Lite. All functions that can also be configured with OP 5, OP7 and OP 15, OP17 are also supported by SIMATIC® C7-620. Special functions were also integrated in the SIMATIC® C7-620 to make working with the system easier.




SIMATIC® S7

Date: 12.03.03
File PRO1_18E.5
**SITRAIN** Training for Automation and Drives

**STEP 7 Micro/WIN** for configuration, service and commissioning of S7-200™ logic controllers

**STEP 7 Mini** for programming, service and commissioning of simple stand-alone applications of S7-300™ and C7-620.
Unlike STEP 7, there are the following restrictions:
- cannot (additionally) load option packages, for example, Engineering Tools.
- no communication configuration (CPU - CPU communication) possible.

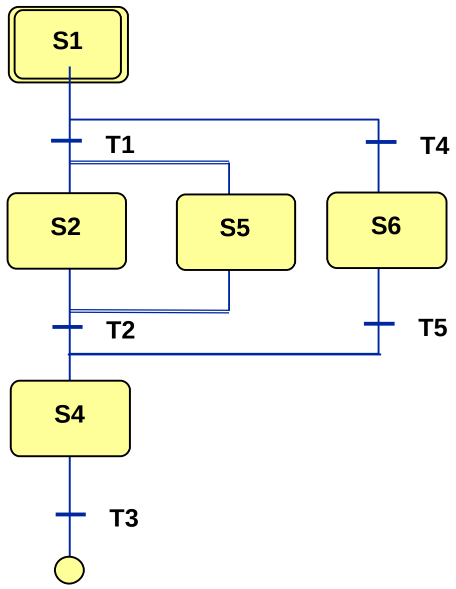
**STEP 7** Basic package for configuring and programming S7-300™/400™/WinAC logic controllers with interfaces to the option packages.

**Options** Options are software packages for program generation, debugging and commissioning:
- S7-SCL = PASCAL-similar high level language.
- S7-GRAPH = Graphic programming of sequence control systems.
- S7-HiGraph = Graphic programming of machining sequences.
- CFC = Graphic configuring and interconnection of blocks.
- S7-PLCSIM = Testing the program logic offline on the PG/PC.
- S7-PDIAG = Process diagnostics for logic controllers and sequence control systems.
- S7-VersionStore = Management of STEP 7 projects.
- TeleService = Extension of the MPI interface using the telephone network.
- HARDPRO = Configuration software for hardware.
- DOCPRO = Documentation software.

**Closed-loop Control (Engineering)** Runtime Software (standard function blocks and parameter-assignment tools) for solving closed-loop control engineering tasks.

## 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**




SIMATIC® S7


Date: 12.03.03
File PRO1_18E.6

SITRAIN Training for Automation and Drives

**S7-GRAPH**

With the S7-GRAPH programming language, you can clearly and quickly configure and program sequential sequences that you wish to control with an S7 PLC system.

The process is thus split into single steps with their own function scope. The sequence is graphically displayed and can be documented with picture and text.

The actions to be performed and the transitions, which control the conditions for switching to the next step, are determined in the individual steps. Their definitions, interlocking or monitoring are determined by a subset of the STEP 7 programming language LAD (ladder diagram).

S7-GRAPH for S7-300™/400™ is compatible with the sequence language established in the DIN EN 6.1131-3 standard.

**Functionality**

The following functions are offered:

- Several sequencers in the same S7-GRAPH function block
- Free number assignment of the steps and transitions
- Simultaneous branches and alternative branches
- Jumps (also to other sequence cascades)
- Starting/Stopping of sequence cascades as well as activating/holding of steps.

**Test Functions**

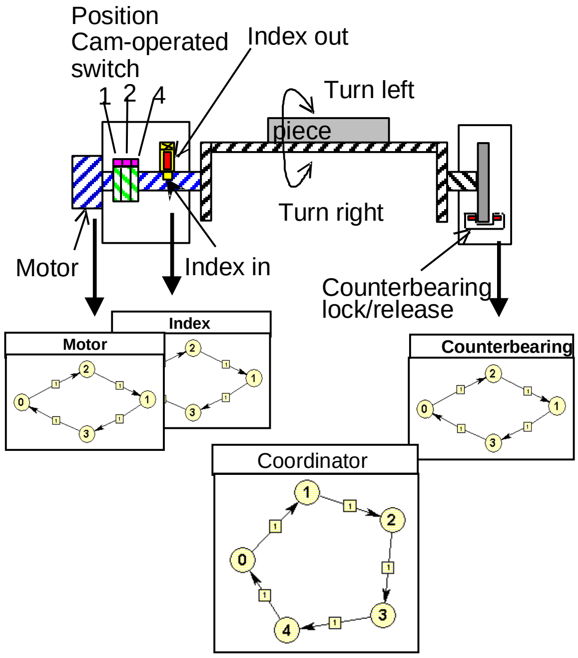
- Display of active steps or faulty steps
- Status display and Modify Variable
- Switching between the operating modes: manual, automatic and jogging mode

**User Interface**

- Overview, Single Page and Single-step display
- Graphic separation of locking controls and monitoring conditions.

## Programming using the State Diagram Method with 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




SIMATIC® S7

Date: 12.03.03
File PRO1_18E.7
SITRAIN Training for Automation and Drives

**Overview** S7-Higraph allows the asychonous processes to be described using state diagrams. The machine or system to be automated is looked upon as a combination of independent elements, the function units.

**Function Units** The function units are the smallest mechanical units of a machine or system. As a rule, a function unit is made up of mechanical and electrical basic elements. In programming, a state diagram is assigned to every function unit. In it, the functional, that is, the mechanical and electrical properties of the function unit are mapped.

**State Diagram** The state diagram describes the dynamic behaviour of a function unit. It describes the states that a function unit can have, as well as the state transitions. State diagrams can be used more than once. State diagrams that were created once for a specific function unit, can be reused in other progam locations.

**Diagram Groups and Instances** By combining parallel running state diagrams, you can describe the complete functionality of a machine or system.

**Advantages** This "object-oriented" method of S7-HiGraph is well suited for:

- the machine and system manufacturer (mechanical engineering)
- the automation specialist (electrical engineering) as common means of description
- the commissioning engineer and the maintenance specialist

The state diagram method helps to optimize the entire process for the creation of a machine or system in the sense of shorter development and turnaround time as well as less commissioning time.

## 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=Structered Text))
  - Certified according to PLCopen 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
```

SIMATIC® S7


Date: 12.03.03
File PRO1_18E.8

SITRAIN Training for Automation and Drives

**Overview**

S7-SCL (Structured Control Language) is a PASCAL-similar high level text language for S7 - 300™/400™ and C7 and simplifies the programming in control technology for mathematical algorithms, data management and organization tasks.

S7-SCL has the PLC open Base Level certificate and is in accordance with the DIN EN 6.1131-3 (Structured Text) standard.

With S7-SCL, you can formulate time-saving and economical solutions for automation tasks.

**Functionality**

SCL offers the functional scope of a high level language such as:

- loops
- alternatives
- branch distributors, etc.

combined with control-specific functions such as:

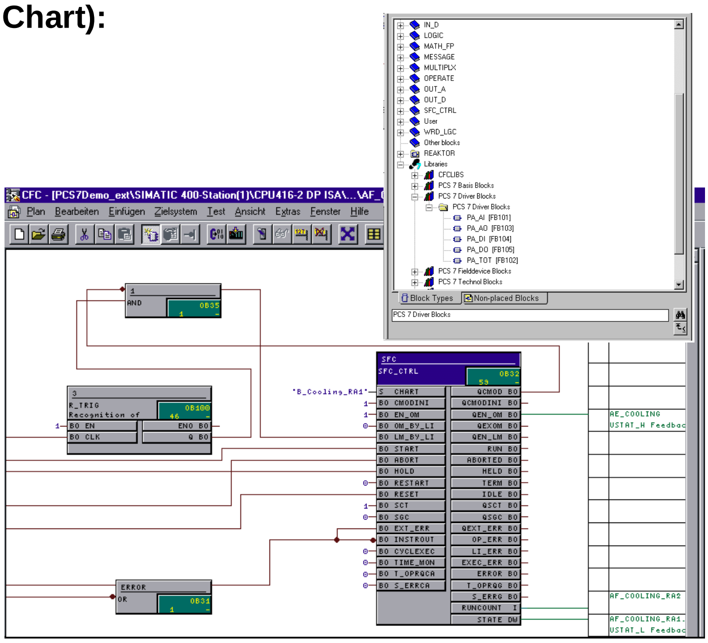
- bit accesses to the I/O, bit memories, timers, counters etc.
- access to the symbol table
- STEP7 block accesses

**Advantages of SCL**

- simple to learn programming language especially for beginners
- easy to read (understandable) programs are generated.
- simpler programming of complex algorithms and processing of complex data structures
- integral debugger for symbolic debugging of the source code (single-step, breakpoints, etc.)
- system integration in S7 languages such as STL and LAD.

## CFC for SIMATIC S7

- **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**



SIMATIC® S7


Date: 12.03.03
File PRO1_18E.9

SITRAIN Training for Automation and Drives

**Overview**

With the engineering tool CFC (Continuous Function Chart), you can create SIMATIC® S7 automation tasks by drawing a technology plan - similar to a Function Block Diagram in PLC programming.

In this graphic programming method, blocks are positioned in a type of drawing sheet and are graphically interconnected with one another. You can quickly and easily convert technological aspects into complete executable automation programs with CFC.

**Scope**

The following is supplied with CFC:

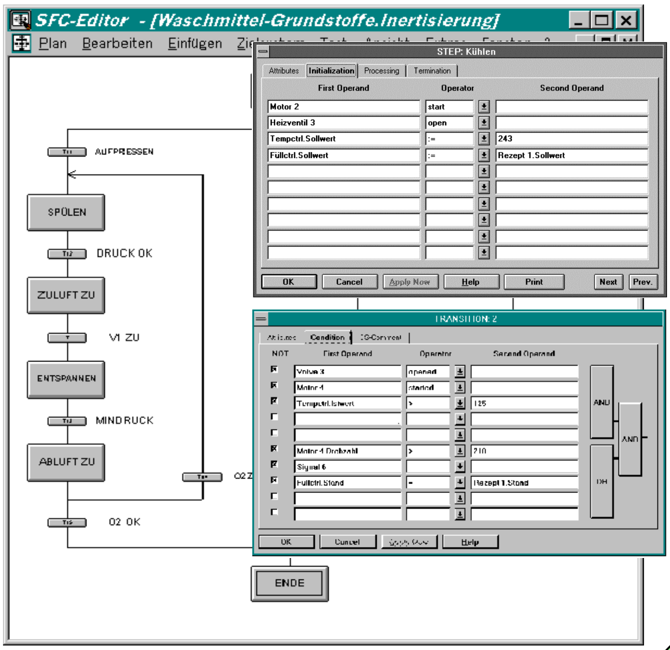
- CFC Editor
- Code Generator
- Debugger
- Standard block libraries

**Customer Benefits**

- The CFC product, as an option package, is smoothly integrated in the STEP 7 architecture with a unified Look & Feel and with common data management. CFC is easy to use, easy to learn and provides consistent data management.
- You can use CFC for simple tasks as well as for very complex tasks.
- Simple interconnection technology makes communication between blocks user-friendly to configure.
- Manual handling and management of machine resources is no longer necessary.
- User-friendly testing and debugging are supported

# Configuring Sequence Control Systems with S7- SFC

- **S7-SFC: The tool for programming sequence cascades**
  - **Designed for the require-ments of process automation**
  - **Compatible with DIN EN 6.1131-3**
  - **Steps assign values to blocks in the CFC**
  - **Transitions check the conditions for switching to the next step**
  - **Syntax check during creation**
- **Direct connection to CFC**
  - **Acceptance of values using “Drag & Drop”**
  - **Cross reference selections**
- **Visualization within WinCC**




SIMATIC® S7


Date: 12.03.03
File PRO1_18E.10

SITRAIN Training for Automation and Drives

**SFC (Sequential Function Chart)**

SFC is a sequence control system with step by step execution, that was designed especially for the demands of process control systems (process engineering, process control engineering, etc.).
The typical fields of application for sequence control systems of this type are in the areas of discrete production processes. Sequence control systems can, however, also be installed in continuous systems, for example, for startup or shutdown, working point changes as well as state changes due to disturbances etc.

With SFC, for example, product manufacturing specifications can be written as event-driven processes.

**Principle Method of Operation**

In the SFC Editor, you generate the flow chart by graphic means. The structure elements of the plan are thereby placed according to fixed rules. You do not have to worry about details such as algorithms or the allocation of machine resources, but instead can concentrate on the technological aspects of the configuration.

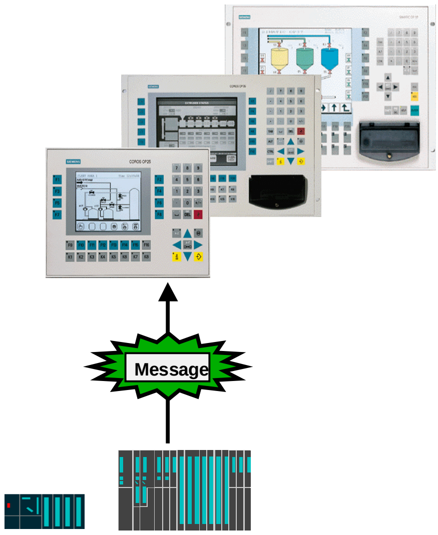
After generating the plan topology, you switch into the detail display (zoom-in configuration) and there assign parameters to the individual elements, that is, you configure the actions (steps) and the conditions (transitions).

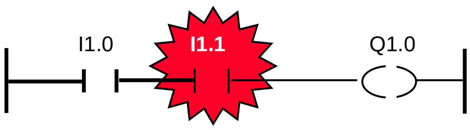
In the programming of actions, functions of the basis automation, typically generated with CFC, are controlled or selectively processed per operating change and state change.

After configuration, you generate the executable machine code through the SFC, download it into the PLC and test it with the SFC test functions.

# Process Diagnosis with S7- PDIAG

- **Process diagnosis: Detection of faults occurring <u>outside</u> 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**




SIMATIC® S7


Date: 12.03.03
File PRO1_18E.11

SITRAIN Training for Automation and Drives

**Process Diagnosis**

Diagnosis is important in the operating phase of a plant or machine. Diagnosis is usually initiated when a fault leads to standstill or malfunction of the plant or machine.

Programmable logic controllers are widely used in many areas. Field experience has proven that over 98% of faults occur in the peripherals (such as magnet valves and end switches). The distribution of fault occurrences makes it meaningful for the diagnosis to focus on process faults, since missing messages or faulty functions lead to down-times and the resulting costs.

Process diagnosis diagnoses exactly these external components (such as sensors and actuators) or sequences in the process of a plant or machine.

**S7-PDIAG**

The S7-PDIAG software package enables a uniform configuration of the process diagnosis for the SIMATIC® S7-300™/400™ controllers in the LAD, FBD and STL programming languages.

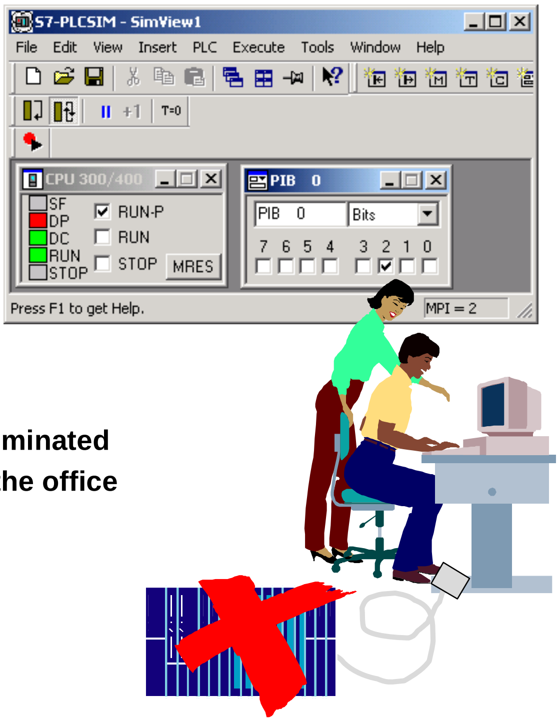
You can already define signal monitoring routines including first-up signal acquisition and criteria analysis and input the associated message texts while or after creating the user program in the LAD, FBD or STL programming languages. PDIAG automatically generates monitoring blocks which you must call in your user program.

At every call, the fault conditions are checked and in case of an error, the relevant process values are acquired and sent to the display device for the criteria analysis.

For the configuration of the operator panel, S7-PDIAG stores the process diagnosis data in a shared database. This data can then be accessed by the OP configuration software SIMATIC® ProTool with the option package ProAgent and be made available for display on the operator panel.

# 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**




SIMATIC® S7


Date: 12.03.03
File PRO1_18E.12

**SITRAIN** Training for Automation and Drives

**S7 - PLCSIM**

The SIMATIC® S7-PLCSIM engineering tool (option package) emulates a complete S7-CPU including addresses and I/O on a PG/PC.

S7-PLCSIM thus enables you to test a program offline on the PG/PC. All STEP 7 programming languages (STL, LAD, FBD, S7-Graph, S7-HiGraph, S7-SCL and CFC) can be used.

S7-PLCSIM allows you to check the functionality of user programs on the PC/PG, regardless of whether the final hardware is available or not.

**Functionality**

S7-PLCSIM offers the following functions for running a program on a simulated PLC:

- An icon in the SIMATIC® Manager‘s toolbar switches the Simulation on or off. If the simulation is turned on, every new connection is automatically made to the simulated PLC.

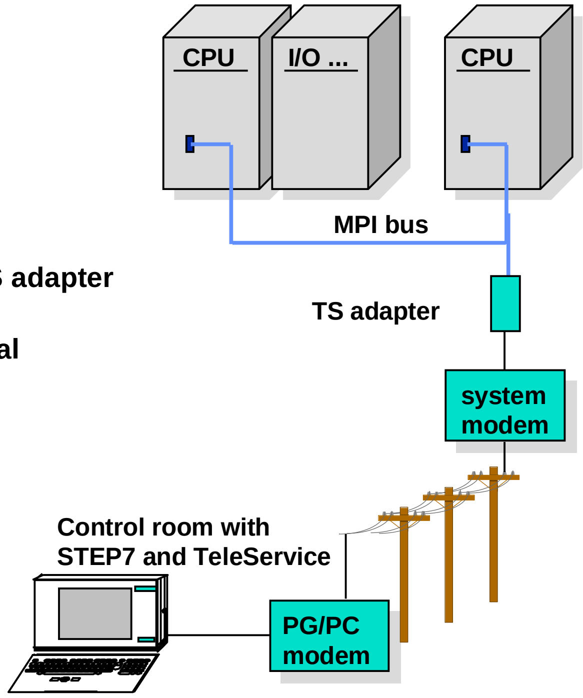
  If the simulation is turned off, then every new connection is made to the "real" PLC.
- You can create view objects that allow you to access memory areas, accumulators and tabs of the simulated CPU. You can modify and display all the data in these view objects.
- You can change the CPU‘s operating mode (STOP, RUN and RUN-P) just as with a "real" CPU. The simulation also provides a "Pause" function that allows you to halt the program execution without affecting the state of the program.

**Advantages**

With S7-PLCSIM, you can detect faults early in the development phase and eliminate them. The quality of the user programs is greatly improved and the commissioning costs are lowered.

## Remote Maintenance and Remote Diagnosis with TeleService

- TeleService: Makes an online connection to SIMATIC® S7/C7 possible
- "Extends" the MPI using telephone/radio networks
  - STEP 7 functionality
  - Market standard modems and TS adapter
  - Fault detection, fault elimination and commissioning from a central location
- Advantages:
  - Reduction of maintenance costs
  - Faster upgrading of the system




SIMATIC® S7


Date: 12.03.03
File PRO1_18E.13

SITRAIN Training for Automation and Drives

**TeleService**

With Teleservice, SIMATIC® S7 / C7 PLCs can be remotely maintained with the PG/PC using a telephone network or a radio network. All the while, you have the full functionality of STEP 7 and the Engineering Tools at your disposal.

**Configuration**

A PG/PC is connected to the PLC using standard modems available on the market. The following are supported:

- Analog modems
- External ISDN adapter/modems
- GSM technology (such as D1 network)

On the plant side, a teleservice-capable TS adapter is inserted between the standard market modem and the MPI network. All stations (nodes) are thus accessible on the MPI network with this connection.

**Procedure**

To set up teleservice operation, you must carry out the following steps:

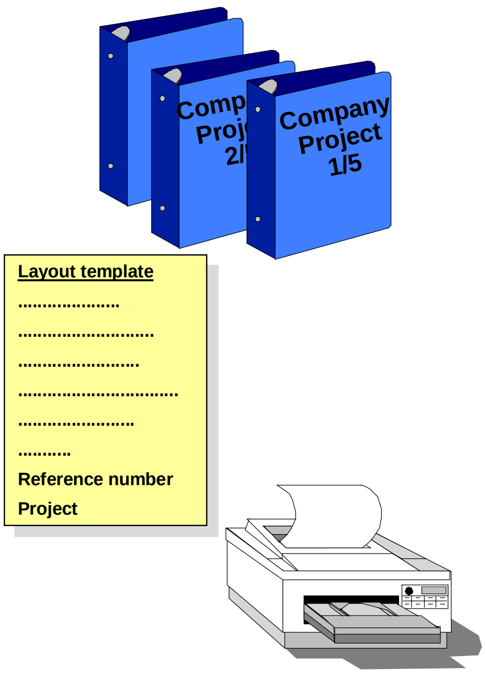
- Assigning parameters to the modem on the PG/PC side (TS adapter with default parameters for the modem on the plant side) using the teleservice package.
- Establishing a remote connection, supported by an electronic phone book, which includes system management in the form of a file system.
- Carrying out remote maintenance with the full function scope of STEP 7 and the Engineering Tools.

**Advantages**

Through the accessibility of PLCs in remote (other rooms, plants, etc.) locations you can carry out technical services such as maintenance, update services or fault analysis from a central service base cost effectively.

## Creating Plant Documentation with DOCPRO

- **DOCPRO: Creating wiring manuals for plants**
  - **Standardized layout templates, can be modified to your needs**
  - **Generates reference numbers, generates indexes**
  - **Prints the entire documentation in <u>one</u> run (such as at night)**
- **Advantage:**
  - **Convenient creation of documents**




SIMATIC® S7


Date: 12.03.03
File PRO1_18E.14

SITRAIN Training for Automation and Drives

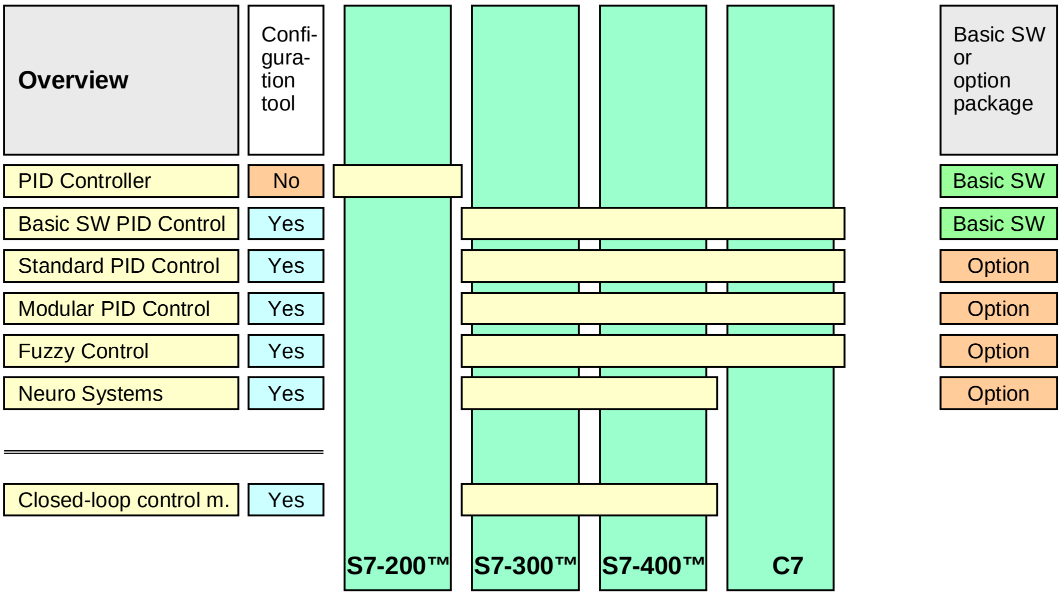
**DOCPRO** DOCPRO is a tool for creating and managing plant documentation. DOCPRO allows you to structure project data, prepare wiring manuals and print out all this information in a uniform format.

**Functionality** DOCPRO provides you with user-friendly functions for creating and managing the documentation as a wiring manual of the plant:

- Creation of wiring manuals and job lists (result of print jobs); a wiring manual is subdivided into job lists.
- Central creation, editing and managing of footer data; the individual jobs can also be assigned footers that contain information about the particular job.
- Standard layout templates supplied with the program in different formats as the starting point for your own layouts and coversheets.
- Automatic and manual assignment of reference numbers; you can assign the job's reference numbers according to your own criteria.
- Automatic creation of document indexes of the printed documentation.
- Printing of job lists and wiring manuals; the jobs of a job list are printed in the predefined sequence.
  You can take a look at the print reports and status list after printing is completed.

**Advantages** The project data of a project/plant can be clearly documented with DOCPRO. A structured (well-organized) documentation makes additional work on a project as well as service work easier and thus saves time and money.

## Runtime Software for Closed-loop Control Engineering Tasks




SIMATIC® S7


Date: 12.03.03
File PRO1_18E.15

SITRAIN Training for Automation and Drives

**Closed-loop Control Engineering** In a closed-loop control system process variables are controlled in such a way that they reach their new preset values as quickly as possible and that they maintain these in spite of the effect of disturbances.

**Basic Software PID Control** The STEP 7 basic package already contains a series of function blocks for solving simple control engineering tasks.

**Standard PID Control** This additional package contains blocks and a parameter assignment tool with integrated control setting for standard tasks such as temperature controllers, flow rate regulators, pressure regulators etc.

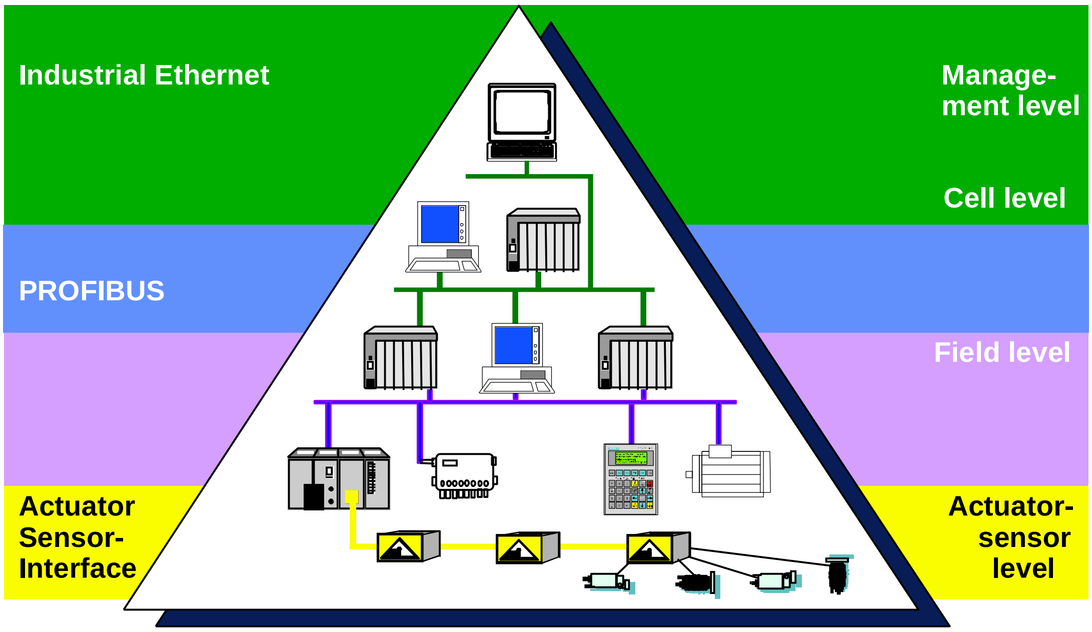
**Modular PID Control** Through the interconnection of supplied standard function blocks, you can implement just about every closed-loop control engineering structure, even in the upper performance range of process engineering.
The package contains 27 FBs and a commissioning tool.

**Fuzzy Control** Fuzzy Systems are used when the mathematic description of a process difficult or even impossible, when a process behaviour is not consistent, when non-linearities occur, but, on the other hand, experience with the process exists.

**NeuroSystems** Neuronal Systems are used with those problems, whose structure and solution are only partly known.
NeuroSystems can be used in all automation levels, from the individual closed-loop controller to the optimization of a plant.

**Closed-loop Control Modules** The closed-loop control modules FM355 (for S7-300™) and FM455 (for S7-400™) are intelligent 4 and 16 channel modules for universal closed-loop control tasks in chemical and process engineering, with rubber and plastics machinery, with heating and cooling units, in the glass, ceramic and paper industry, etc.

## Communicating with SIMATIC NET




SIMATIC® S7


Date: 12.03.03
File PRO1_18E.16

SITRAIN Training for Automation and Drives

**SIMATIC® NET**

SIMATIC® NET is the name of an entire family of networks.

- Industrial Ethernet according to IEEE 802.3 - the international standard for the networking of areas and cells
- PROFIBUS according to EN 50170 - the international standard for the field area and the cell network with a limited number of nodes
- AS-Interface - for communication with sensors and actuators.

**Industrial Ethernet**

The Industrial Ethernet network is a cell level network according the international standard IEEE 802.3 (Ethernet) and is designed for industrial use. Extensive open network solutions are possible. A high transmission rate is guaranteed with various transmission media. Industrial Ethernet is an industry standard, world-wide tested and accepted. The Industrial Ethernet network functions according to the IEEE 802.3 standardized accessing procedure CSMA/CD (Carrier sense multiple access with collision detection).

**Profibus**

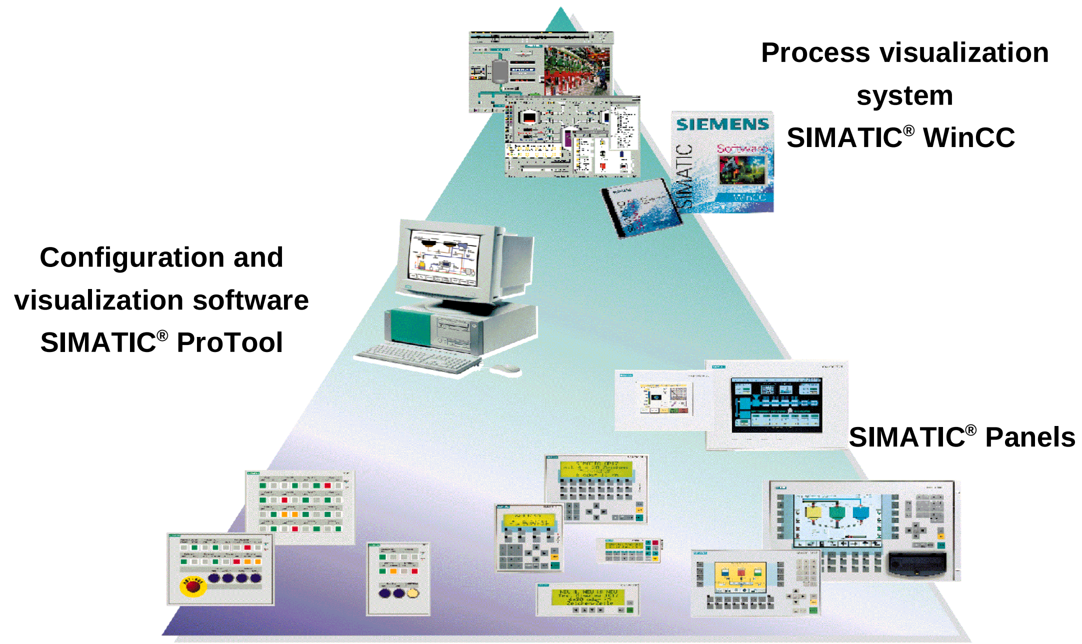
PROFIBUS is the bus system for cell networks with a limited number of nodes. It is based on the European standard EN 50170, Volume 2, PROFIBUS. Since the requirements according to EN 50170 are fulfilled, PROFIBUS guarantees openness for the connection of components from other manufacturers that conform to standards. The PROFIBUS accessing procedure functions according to the "Token Passing with subordinate Master-Slave" procedure. As a result, a distinction is made between active and passive network participants.

**AS-Interface**

The AS-Interface is a networking system for binary sensors and actuators in the field area. With AS-Interface, binary actuators and sensors become capable of communication, for which a direct field bus connection was not technically possible up until now or was not economical.

Unlike the powerful PROFIBUS, the main area of use for the AS-Interface line is the transmission of small amounts of information such as from switching positions.

## Operator Control and Process Monitoring with SIMATIC® HMI







**Overview**

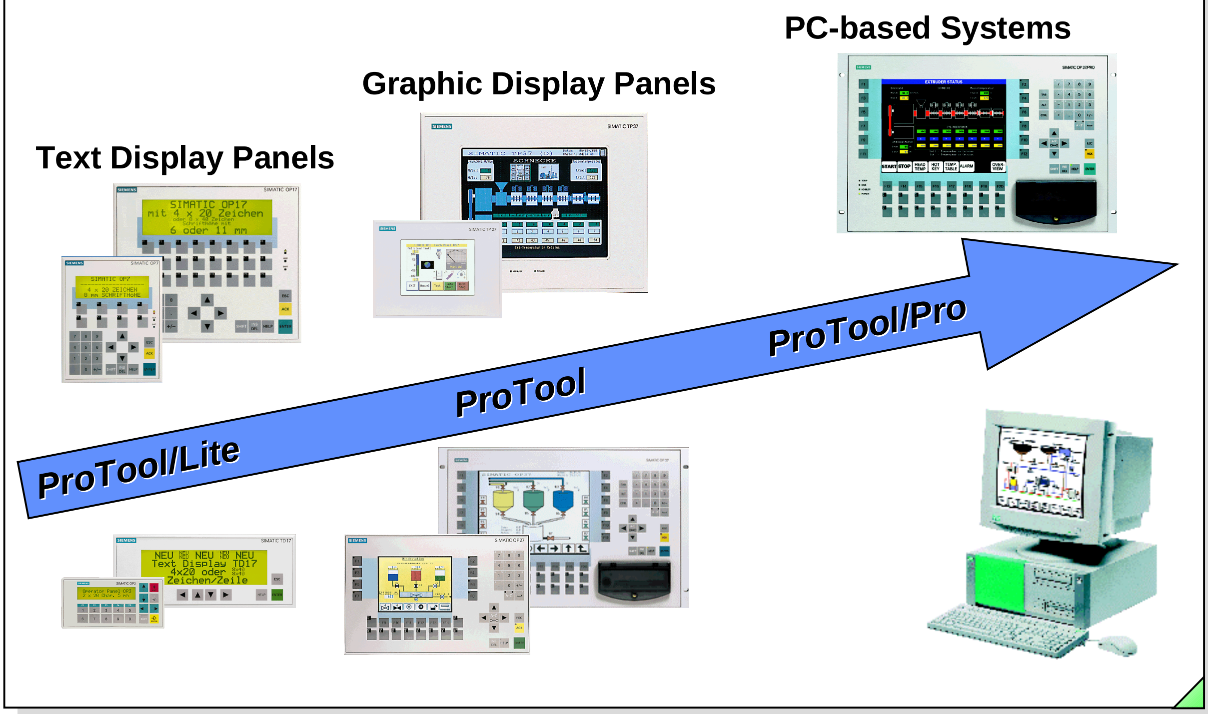
For the SIMATIC® S7, there is a field-proven HMI system for user-friendly process control and monitoring available, the SIMATIC® HMI. It ranges from the simple text display to the process visualization system.

SIMATIC® S7 and SIMATIC® HMI are completely harmonized and integrated. This simplifies the use of the human-machine interface system SIMATIC® HMI considerably.

- SIMATIC® S7 has already integrated HMI services. The HMI system requests process data from the SIMATIC® S7. Data transmission between SIMATIC® S7 and SIMATIC® HMI is carried out by the two operating systems and does not have to be taken into account in the user program.

  SIMATIC® HMI systems can be connected directly to PPI (S7-200™) and MPI or Profibus (S7-300™ and S7-400™). Operation using PROFIBUS makes process control and monitoring even over greater distances possible.
- Numerous features from the uniform database and symbols up to the same user-friendly Windows-oriented user interfaces simplify the use of HMI systems.

## Consistent Configuration with SIMATIC® ProTool




SIMATIC® S7



Date: 12.03.03
File PRO1_18E.18

**SITRAIN** Training for Automation and Drives

**ProTool**
**ProTool/Lite**

SIMATIC® ProTool and SIMATIC® ProTool/Lite are modern configuration tools for configuring SIMATIC® Text Displays, Operator Panels, Touch Panels as well as the HMI portion of the SIMATIC® C7 complete system.

While you can configure all devices with SIMATIC® ProTool, SIMATIC® ProTool/Lite, as the economical version, is restricted to the configuration of line oriented devices.

Functionally, SIMATIC® ProTool/Lite is a subset of SIMATIC® ProTool. The operator control and configuration philosophy of both tools is the same.

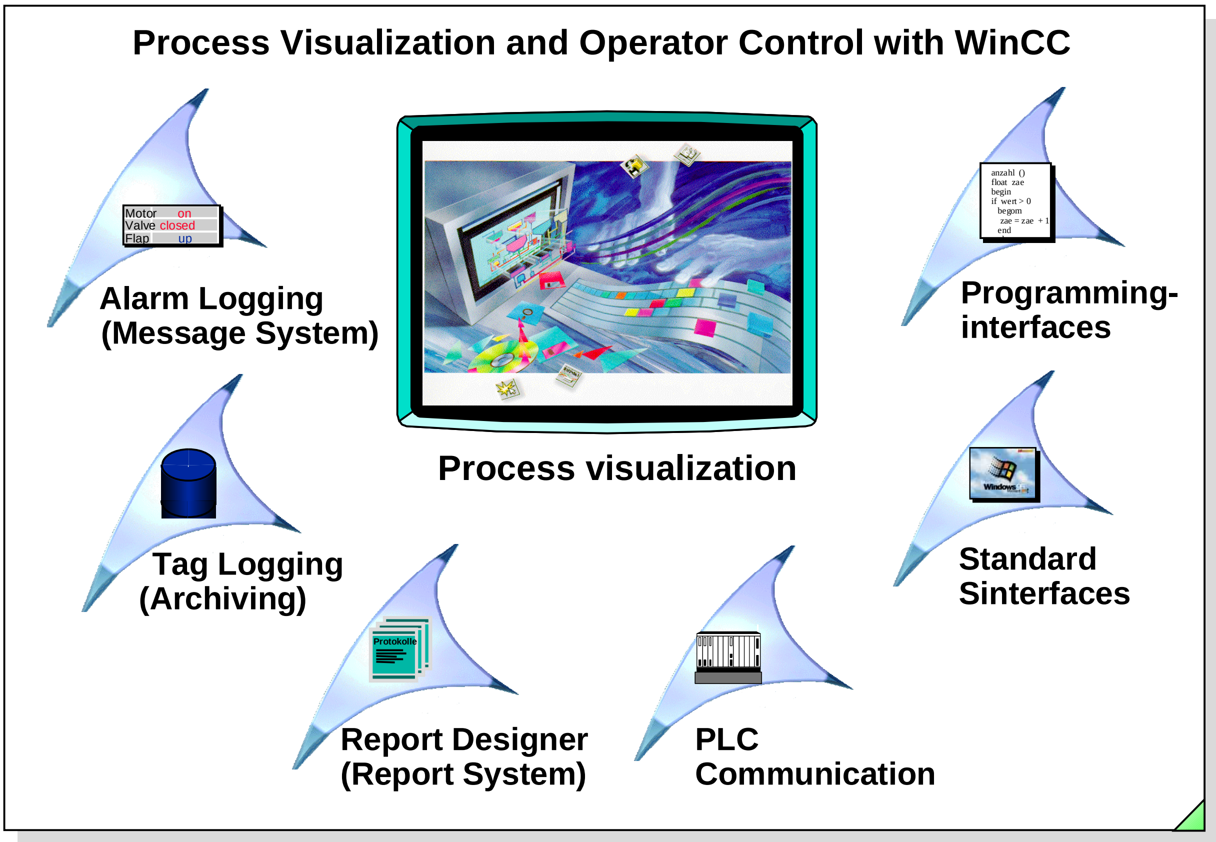
**ProTool/Pro**

SIMATIC® ProTool/Pro upwardly expands the existing product family of SIMATIC® ProTool with the Operator Panel OP37/Pro and supplements the panels with a runtime software for a standard PC.

ProTool/Pro contains the basic functionality of the graphic display units (OP27, OP37) and thus creates a visualization consistency from the existing graphic OPs up to the PC-based systems.

ProTool/Pro stands out with the following features:

- Runtime software for various platforms
  - OP37/Pro (Windows 95)
  - Standard PC (Windows 95/98ME MS and NT 4.0/2000)
- Extensive basic functionality of the graphic OP OP27, OP37
- Expanded functional scope to OP27, OP37




SIMATIC® S7


Date: 12.03.03
File PRO1_18E.19

SITRAIN Training for Automation and Drives

**WinCC**

SIMATIC® WinCC (Windows Control Center) is the open process visualization system from Siemens. It can be integrated problem free in a new or already existing PLC system.

**Function Modules**

The heart of SIMATIC® WinCC is an industry and technology independent basic system with all the important functions for operator control and monitoring, such as:

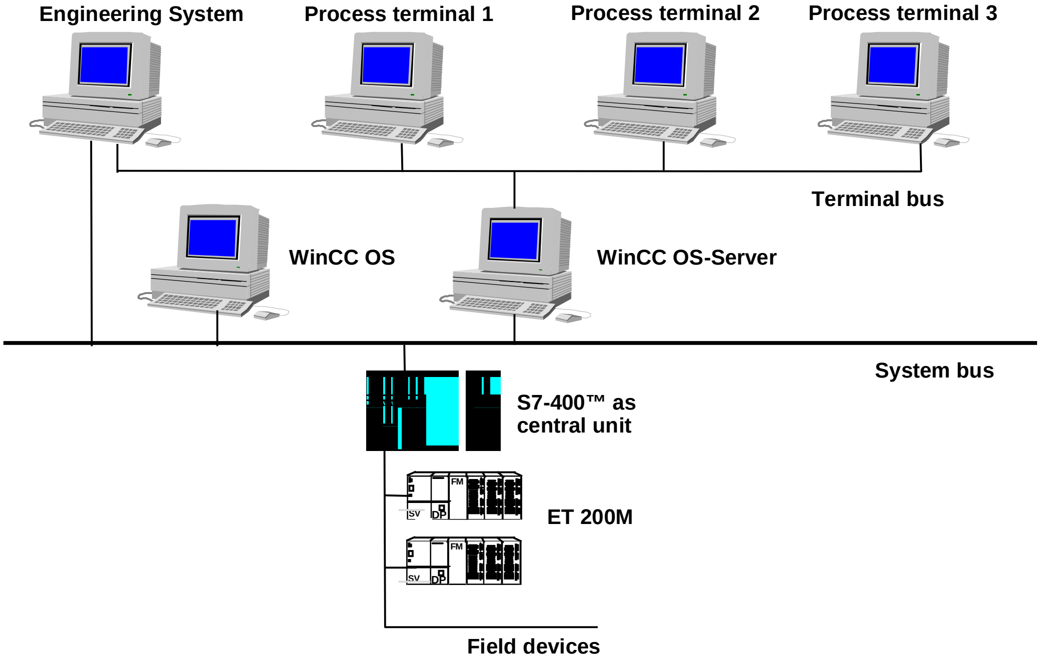
- pixel graphic display
- measured value acquisition (archiving functions, data compression, minimum and maximum values etc.)
- message display, archiving and reporting
- process communication to different PLC systems
- standard interfaces, for example, Microsoft programs
- documentation of machine and process sequences with individual reports.

**Basis of WinCC**

WinCC V5 is based on the 32-bit standard operating systems Windows NT/2000 from Mircrosoft. This platform gives WinCC the following functionality:

- use of the Windows operating equipment (such as printer and driver)
- data exchange with other Windows applications via DDE, ODBC, SQL, OLE, ActiveX and OPC.
- API programming interface
- use of hardware available in the market

## Process Automation with SIMATIC® PCS 7




SIMATIC® S7


Date: 12.03.03
File PRO1_18E.20

SITRAIN Training for Automation and Drives

**Introduction**

SIMATIC® PCS 7 represents the new control system generation in SIEMENS. It is the consistent, further development and summary of the experiences with TELEPERM M, SIMATIC® PCS and SIMATIC® S5 based systems. As a result, it is tailored to the process control system tasks in all industries.

**Engineering System**

The Engineering System can be designed as its own station in the system. It can however also be loaded as a software package in the OS components at the same time.
The Engineering System has the following components:

- STEP 7 with the SIMATIC® Manager, the central database, and with HW Config for configuring hardware and networks. It also contains the servers, that facilitate consistent configuration between PLC and OS.
- SCL (Structured Control Language) as PASCAL-similar higher level programming language for block generation
- CFC (Continuous Function Chart) for graphic configuration of the basic automation functions
- SFC (Sequential Function Chart) for graphic configuration of production sequences
- Expansion of the SIMATIC® Manager with a technological hierarchical view
- WinCC (Windows Control Center) for OS configuration
- DOCPRO for documenting configuration data
- Import–/Export wizard for bi-directional data exchange with other CAE systems

These components are supplemented by libraries that provide pre-defined blocks for PLC and OS.

## DRIVES Technology – The Muscles of TIA

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

SIMATIC® S7


Date: 12.03.03
File PRO1_18E.21

SITRAIN Training for Automation and Drives

**Overview**

You have the following drives spectrum:

- Low voltage motors are the first High Performance AC for machine and systems - the solution for the future: maintenance free, dynamic and powerful.
- SIMOVERT MASTERDRIVES frequency converters. They control the speed of AC motors extremely exact. This series was designed for world-wide use. It is suitable for all supply voltages from 230 to 690 volt and is certified world-wide.
- MICROMASTER and MICRO/MIDIMASTER Vector standard converters are frequency converters in the 120 watt to 75 kW performance range. Because of their compact form, they can be installed in the smallest space. The sensorless vector control allows it to be used in the medium performance range even for demanding applications.

  COMBIMASTER are compact units consisting of three-phase low voltage motors and frequency converters.

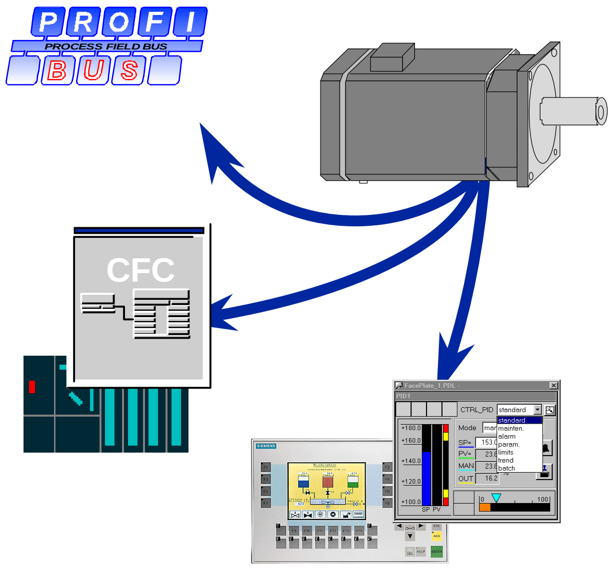
  MICROMASTER Integrated are frequency converters (IP 65) that are applied directly to three-phase low voltage motors of different manufacturers.

  MICRO/MIDIMASTER Eco are frequency converters specially designed for the requirements of the heating, ventilation and air conditioning industry.
- SIMOREG converter equipment are fully digital compact units for three-phase operation and are used for armature and field supply of variable speed DC drives. The range of rated direct current of the devices is from 15 to 2000 A and can be increased by parallel connection of SIMOREG devices. The most familiar applications include hoisting gear, ski lifts, elevators, cranes, and other reversing drives.

## DRIVES Technology – as a Component of TIA

- PROFIBUS for closed-loop control using the network
- Preconfigured software modules for SIMATIC® Controller
- Templates for SIMATIC® HMI systems

➔ Reduced time for the integration of drives in automation solutions




SIMATIC® S7


Date: 12.03.03
File PRO1_18E.22

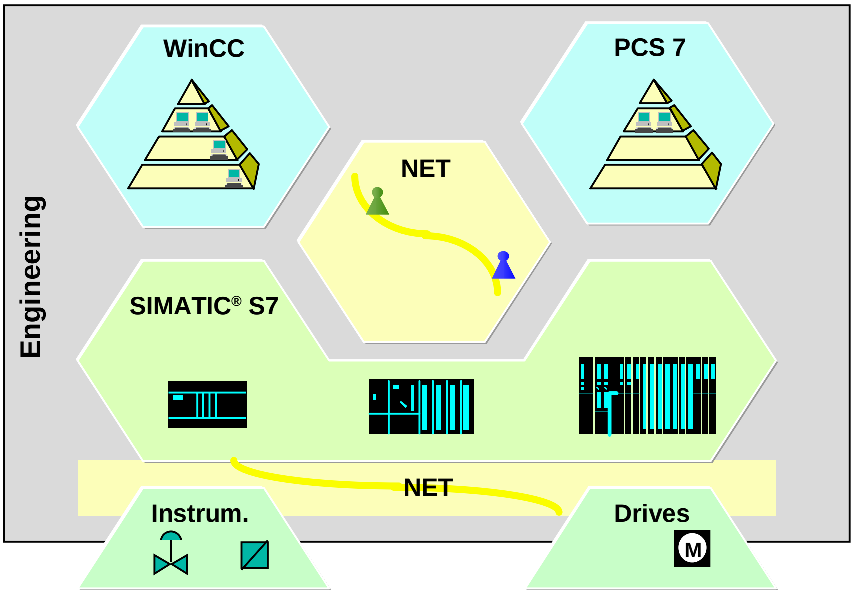
SITRAIN Training for Automation and Drives

**Software Modules** SIMATIC® DriveES supports the connection of SIMOREG, SIMOVERT, and SIMODRIVE variable speed drives to a higher-level SIMATIC® S7 controller system.

**Profibus** The connection is made using the standardized communication system PROFIBUS-DP according to the “PROFIBUS profile variable speed drives, PROFIDRIVE“ or using the “universal interface protocol“.

**Templates** Prefabricated symbols (templates) make the creation of system pictures easier.

## Summary




SIMATIC® S7


Date: 12.03.03
File PRO1_18E.23

SITRAIN Training for Automation and Drives

**Totally Integrated Automation**

The new SIMATIC® family unifies all devices and systems, that is, hardware as well as software, into a uniform, powerful system platform.
In this platform, the system borders that have existed until now, that is, the borders between computer world, PLC world and process control, that is, between operator control and monitoring and control, between central and distributed automation are overcome.

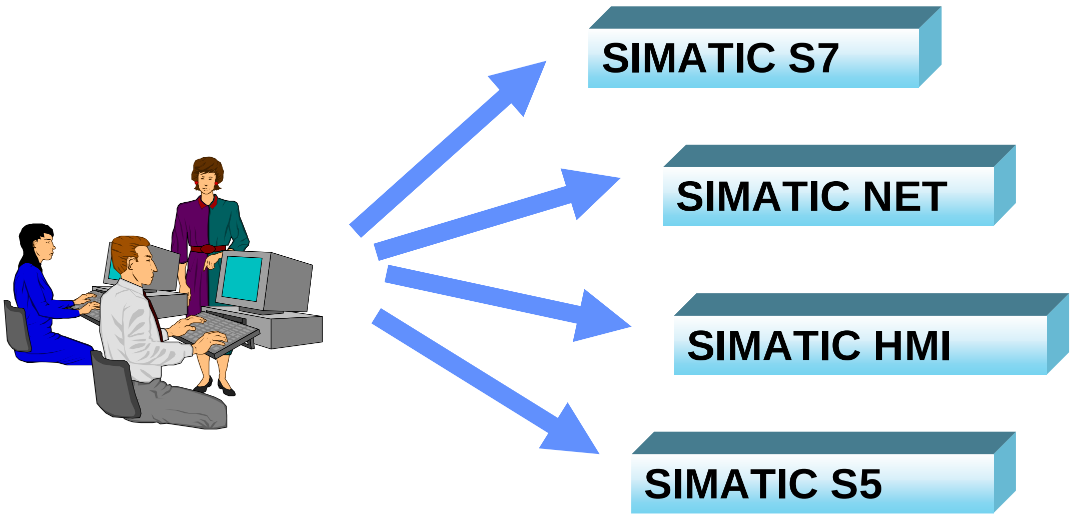
**Advantages**

This totally integrated automation offers you, among other things, the following advantages:

- A scalable hardware platform, that is, the optimal (price/performance) functionality (PLC or computer) can be chosen for the task to be solved.
- An open totally integrated automation environment, that is, an existing system can be easily extended, or existing or future automation solutions can be integrated.

  Existing investments retain their value. The transition from an existing SIMATIC®, TELEPERM or TI environment can be carried out very easily.
- Powerful software increases the productivity in the implementation of a project and thus reduces the engineering and life cycle costs. In addition, expenses for commissioning, maintenance and service are reduced.
- SIMATIC® is based on Windows standards and can thus easily use their applications (standard software) and communication mechanisms.

## What's Next ?




**and other courses on PCS7, IT, NC ....**





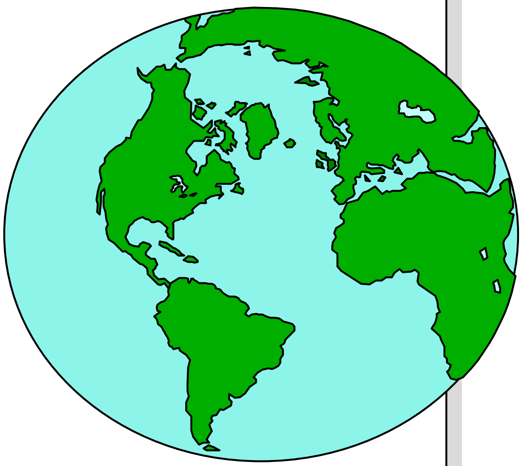
## We'd just like to say a few words...

Contents:

## Our Automation and Drives Training

- Training Courses
  - on site or in
  - 200 locations in
  - 60 countries
- Future-oriented and Topical  Training
  - first hand
  - from the market leader
- Task-oriented Training
  - individually decided with you
- Training for everybody, in all areas of Automation and Drives

Training from A&D



SIMATIC S7


Date: 12.03.03
File: PRO1_14E.2

SITRAIN Training for Automation and Drives

### What are the advantages of our SIMATIC Training for you?

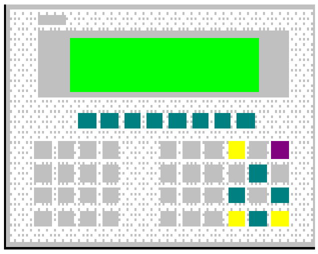
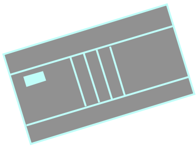
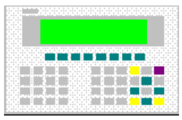
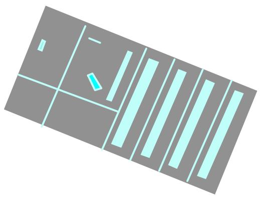
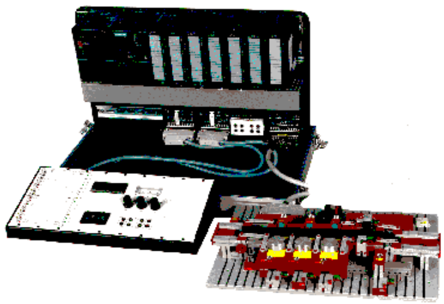
- Fast, effective acquisition of knowledge
- Saves downtimes at your plant
- Ensures quality
- Gives you motivated personnel
- Simplifies and shortens decision-making processes

| | |
|---|---|
| **Note** | The following pages present just a **sample** of our extensive **range of SIMATIC courses**.<br><br>We‘ll gladly send you information about our **entire course spectrum** at your request! |

Look us up on the Internet:
**http://www.sitrain.com**
or call our Info Line:
Tel: **01805 23 56 11**
Fax: **01805 23 56 12**

## SIMATIC Training

- SIMATIC S7
- SIMATIC HMI (ProTool/Pro, WinCC)
- SIMATIC NET (PROFIBUS, Ethernet)
- SIMATIC S5

SIMATIC S7


Date: 12.03.03
File: PRO1_14E.3

**SITRAIN** Training for Automation and Drives

You have just attended one of our courses and we hope it came up to your expectations.

Above all, we hope you will be able to use the knowledge you obtained at the course to advantage in your work.

We would like to continue to be your partner in training for your career in the future.

For this reason we have outlined some of our current courses for you on the next few pages.

## Upgrade from SIMATIC S5 to SIMATIC S7

### Installation, Service and Operating Personnel

Sound SIMATIC S5 knowledge and SIMATIC S5 programming experience

PC and Windows knowledge

SIMATIC S7

Upgrade SIMATIC S5 -> S7

ST-7UPSERV 5 days

SIMATIC S7

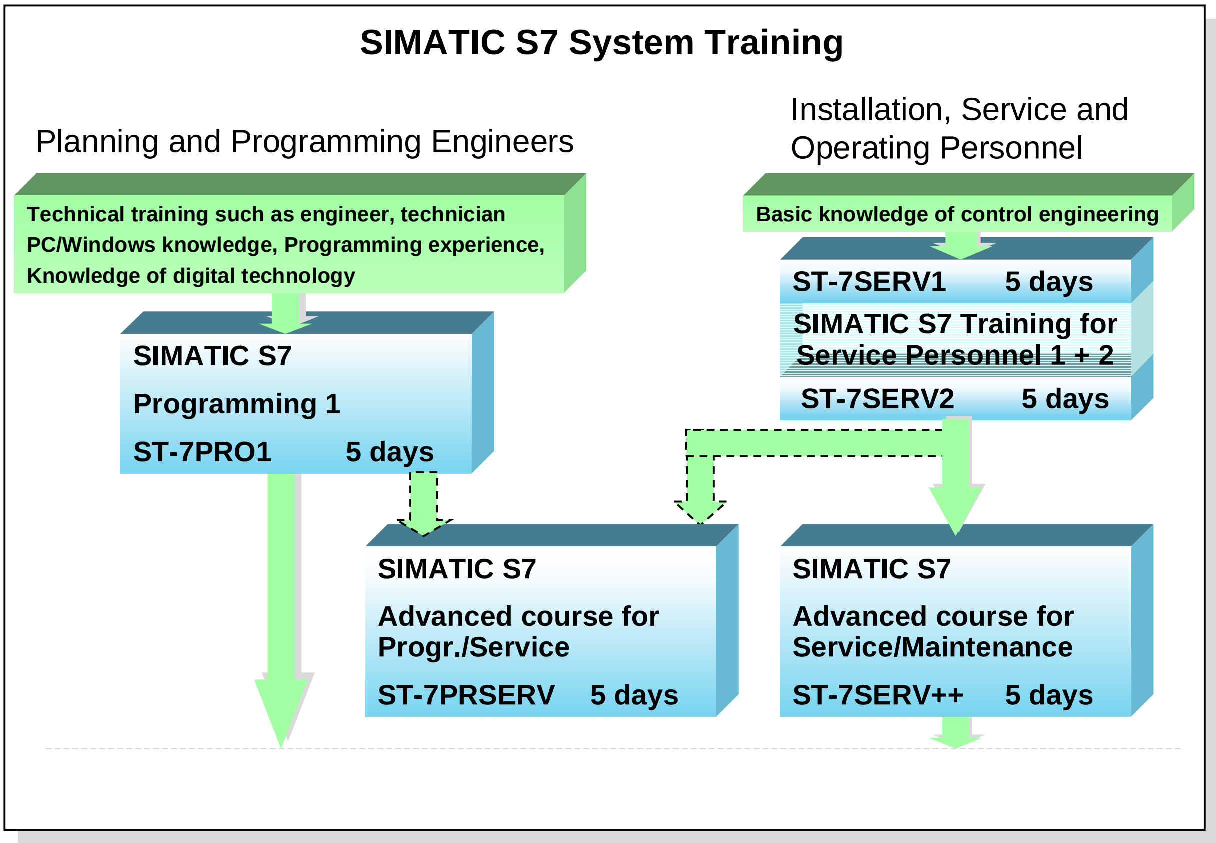

Date: 12.03.03
File: PRO1_14E.4

SITRAIN Training for Automation and Drives

**ST-7UPSERV** **Upgrade SIMATIC S5 -> S7 for Service**

Course Contents (Excerpt):

- SIMATIC S7 system overview
- Mounting and maintaining the automation system
- Configuring hardware and assigning parameters to it
- Hardware and software commissioning
- Becoming familiar with STEP 7 software for troubleshooting and program expansions and being able to use it
- Block types and symbols
- Documenting, saving and archiving




SIMATIC S7

Date: 12.03.03
File: PRO1_14E.5
**SITRAIN** Training for Automation and Drives

| | |
|---|---|
| **ST-7PRO1** | **SIMATIC S7 Programming 1** |
| Course Contents (Excerpt): | - System overview and main performance features<br>- Block types and symbols for program structuring and program creation<br>- Test tools for system information, troubleshooting and diagnostics<br>- Hardware configuration of the modules<br>- Communication via MPI interface |
| **ST-7SERV1** | **SIMATIC S7 Service Training 1 (System Handling)** |
| Course Contents (Excerpt): | - Ability to configure and install a programmable logic controller<br>- Hardware and software commissioning of the programmable logic controller<br>- Overview of the S7-300 software configuration and parameter assignment |
| **ST-7SERV2** | **SIMATIC S7 Service Training 2 (Troubleshooting)** |
| Course Contents (Excerpt): | - Using STEP 7 software for troubleshooting<br>- Detecting and eliminating software errors, that lead to the CPU Stop state<br>- Detecting and eliminating logical software errors<br>- Saving and documenting program changes that were made |
| **ST-SERV++** | **SIMATIC S7 Service/Maintenance** |
| Course Contents (Excerpt): | - Software start-up<br>- Integrating FCs and FBs in the program<br>- Error organization blocks<br>- Outputting diagnostic messages<br>- Distributed Peripherals<br>- Using ProTool/Pro for troubleshooting<br>- Troubleshooting in the MPI network |
| **ST-7PRSERV** | **SIMATIC S7 Programming/Service** |
| Course Contents (Excerpt) | - Creating a program for a conveyor belt<br>- Programming FCs and FBs<br>- Programming error organization blocks<br>- Being able to evaluate diagnostic data<br>- Distributed Peripherals, basics<br>- Using Pro/Tool Pro for troubleshooting |

# SIMATIC S7 Online Course Studies

## PLC Professional - Programming

**Planning and Programming Engineers**

**Installation, Service and Operating Personnel**

| | |
|---|---|
| **PLC Professional - Programming Basic** | |
| **ST-OKPPRO1** | **approx. 3 months** |
| **PLC Professional - Programming Advanced** | |
| **ST-OKPPRO2** | **approx. 3 months** |
| **PLC Professional - Programming Experts** | |
| **ST-OKPPRO3** | **approx. 3 months** |





**ST-OKPPRO1** **PLC Professional - Programming Basic**

Course Contents:

- Important performance features of the SIMATIC S7 system family
- Project structure and project management
- Basic programming, absolute and symbolic
- Block architecture and block editor
- Binary and digital operations
- Data storage in data blocks
- Functions and function blocks
- Organization blocks
- Debugging tools and troubleshooting procedure
- Documenting, saving and archiving programs

**ST-OKPPRO2** **PLC Professional - Programming Advanced**

Course Contents:

- SIMATIC S7 hardware configuration and simulation
- Status word and arithmetic operations
- Structured programming
- Organization blocks and analog value processing
- Variable addressing (direct and indirect)
- Troubleshooting in the user program

**ST-OKPPRO3** **PLC Professional - Programming Experts**

Course Contents:

- Multi-instances
- Creating and using complex data structures
- Accessing complex data structures using indirect addressing
- Passing on complex data between parameter-assignable blocks
- Integrated error handling and libraries
- Configuration exercises (Exam preparation)
- Exam with Certificate

## SIMATIC S7-200 Training

### Configuring/Programming and Installation/Maintenance

Basic knowledge of control engineering
PC/Windows knowledge

**SIMATIC S7**
**S7-200 Workshop**
**ST-7MICRO** **2 days**

SIMATIC S7


Date: 12.03.03
File: PRO1_14E.7

SITRAIN Training for Automation and Drives

**ST-7MICRO**

Course Contents (Excerpt):

**SIMATIC S7, S7-200 System Course**

- Becoming familiar with the performance features of the SIMATIC S7-200 PLCs and programming devices
- Expansion facilities and S7-200 addressing
- Ability to structure, write, document and start up simple programs for control tasks on SIMATIC S7-200 PLCs
- Ability to use STEP7 Micro/WIN programming tools for program creation, documentation, program test and troubleshooting

# SIMATIC WinAC

## Configuring/Programming

SIMATIC S7 knowledge equivalent to the ST-7PRO1, ST-7SERV1/2 or ST-7UPSERV courses


**SIMATIC S7**
**PC-Based Control with WinAC**
**ST-7WINAC 2 days**

**SIMATIC S7**
**Component based Automation**
**ST-7CBA 2 days**

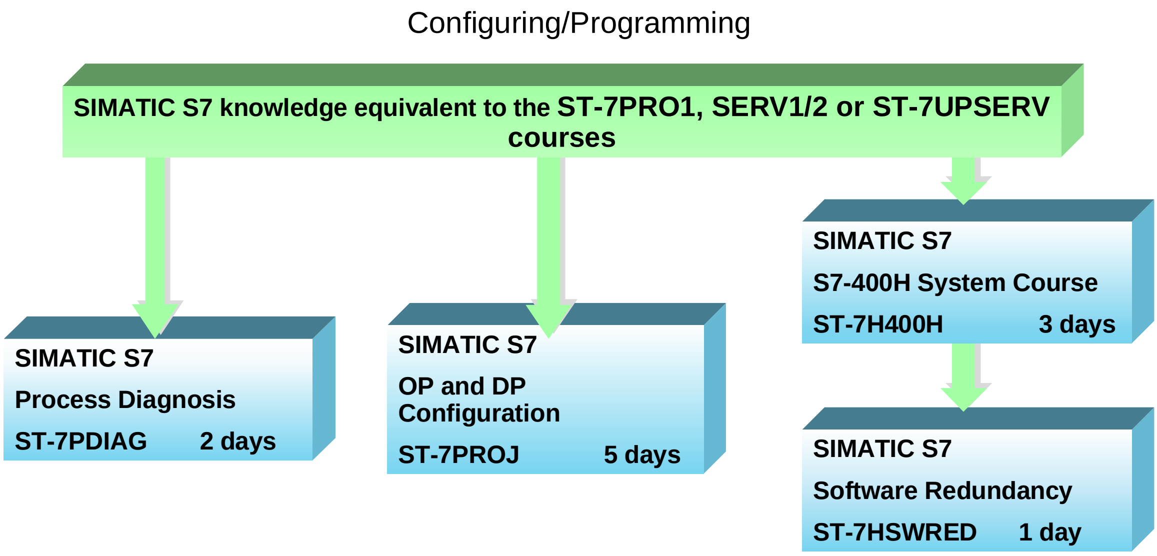
SIMATIC S7


Date: 12.03.03
File: PRO1_14E.8

SITRAIN Training for Automation and Drives

| | |
|---|---|
| **ST-7WINAC** | **SIMATIC S7, PC Based Control with WinAC** |
| Course Contents | - Introduction to PC based control with SIMATIC WinAC |
| (Excerpt): | - WinAC hardware components, properties, components |
| | - Use of Data-OCX and supplied OCXs |
| | - Assigning parameters to the MPI card |
| | - Overview of OPC / ActiveX / DCOM for WinAC |
| **ST-7CBA** | **SIMATIC S7, Component based Automation** |
| Course Contents | - Basics of structured programming |
| (Excerpt) | - Product overview of CBA components |
| | - Component creation |
| | - SIMATIC iMAP as central engineering tool |
| | - HMI link using SIMATIC OPC Server |
| | - Notes and solution approaches on component design |

# SIMATIC S7 - Additional Courses (Excerpt)



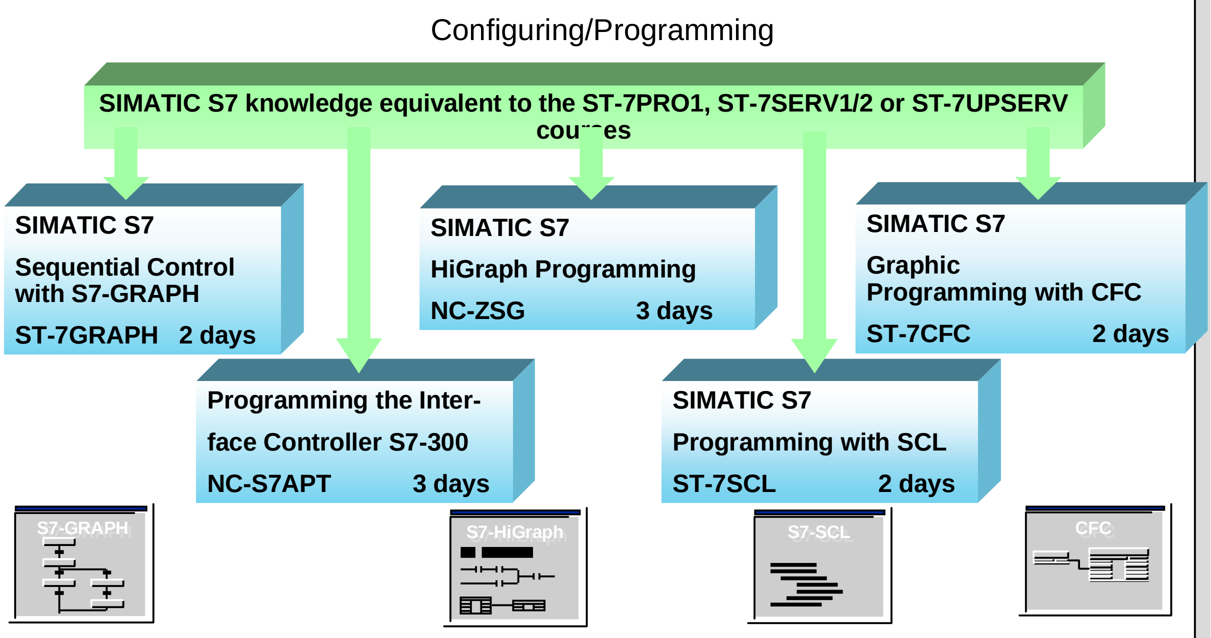

SIMATIC S7


Date: 12.03.03
File: PRO1_14E.9

SITRAIN Training for Automation and Drives

| | |
|---|---|
| **ST-7PDIAG** | **SIMATIC S7, Process Diagnosis** |
| Course Contents (Excerpt): | - Introduction to the principles of process diagnosis<br>- Becoming familiar with/configuring the types of monitoring<br>- Diagnostic messages on the Operator Panel (OP27) with ProTool and ProAgent<br>- Support of process units and motion<br>- Process diagnosis for sequential control with S7-GRAPH |
| **ST-7PROJ** | **SIMATIC S7, Projektieren OP und DP** |
| Course Contents (Excerpt): | - System overview and essential performance features of the S7 controller, Operator Panel (OP) and PROFIBUS DP<br>- Hardware configuration SIMATIC S7, OP27(17) and ET200M<br>- Introduction to configuration with ProTool<br>- Data exchange between STEP7 and a standard Windows application, such as MS Excel using DDE Manager<br>- Besonderheiten der S7-400 |
| **ST-7H400H** | **SIMATIC S7, System Course S7-400H** |
| Course Contents (Excerpt) | - Theory of redundancy basics<br>- SIMATIC S7-400 H operating system<br>- Programming and Diagnostics<br>- I/O peripheral options<br>- STEP7 special features<br>- RedConnect |
| **ST-7HSWRED** | **SIMATIC S7, Software Redundancy** |
| Course Contents (Excerpt): | - Theory of redundancy<br>- Principle of software redundancy<br>- Programming and Diagnostics<br>- IM 153-3 |

# SIMATIC S7 Option Packages

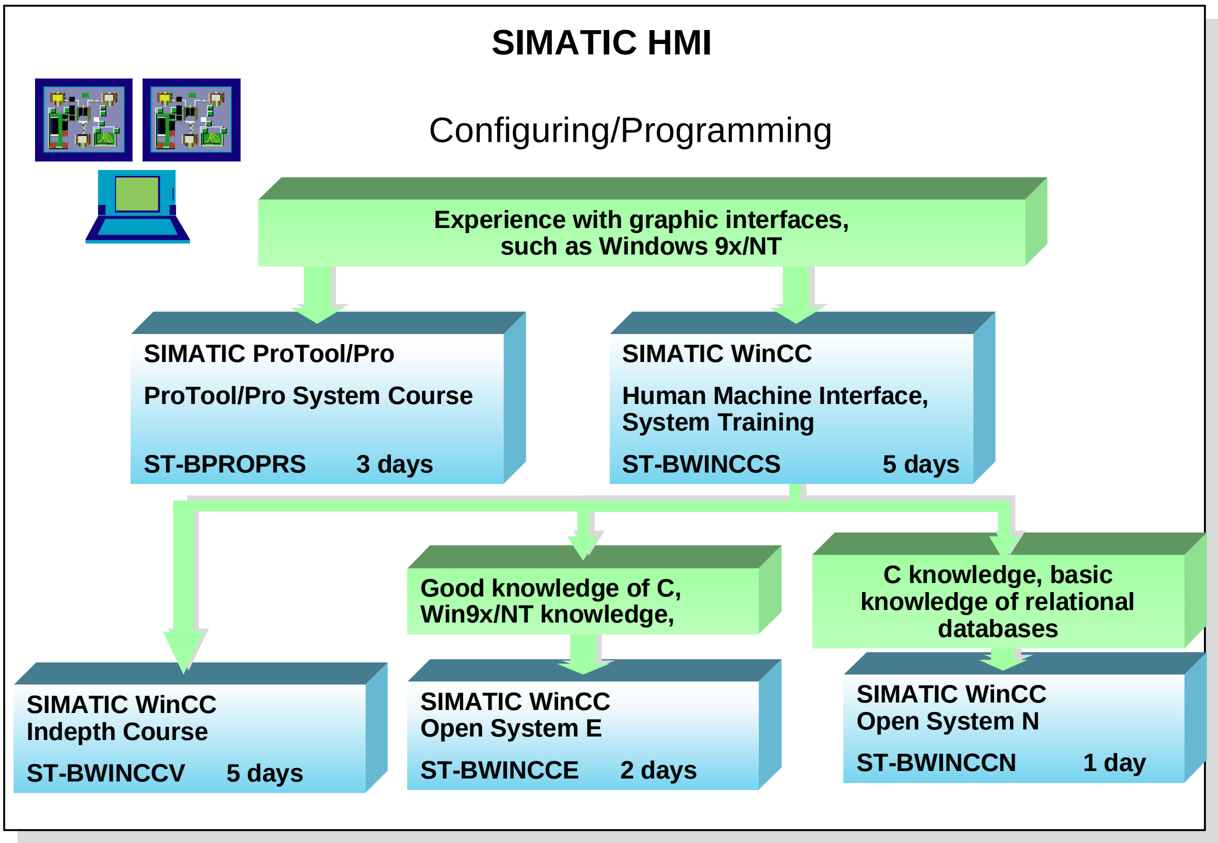



SIMATIC S7


Date: 12.03.03
File: PRO1_14E.10

SITRAIN Training for Automation and Drives

| | |
|---|---|
| **ST-7GRAPH** | **SIMATIC S7, Sequential Control with S7-GRAPH** |
| Course Contents (Excerpt): | - Design, structure and representation types of sequence controllers with S7 GRAPH<br>- Programming, documenting and starting-up sequential blocks<br>- Programming lockout and monitoring<br>- Properties of simaltaneous and alternative branches<br>- Test functions and diagnostic possibilities |
| **NC-S7APT** | **Programming Interface Controller S7-300** |
| Course Contents (Excerpt): | - Overview of the FMNC, 810D and 840D controllers<br>- Structure of the PLC - NC interface<br>- Fast data exchange between PLC and NC<br>- Communication structures<br>- Practical exercises on the individual topics |
| **NC-ZSG** | **HIGRAPH Programming SIMATIC S7** |
| Course Contents (Excerpt): | - The nature of state graphs<br>- Programming tools and their use<br>- State graphs programming language<br>- Monitoring function for program test |
| **ST-7SCL** | **SIMATIC S7, Programming with SCL** |
| Course Contents (Excerpt): | - S7-SCL Editor<br>- Data types, operations<br>- Formulating FBs, FCs, OBs, … in SCL<br>- Control structures: IF, WHILE, REPEAT, ... |
| **ST-7CFC** | **SIMATIC M7/S7, Graphic Programming with CFC** |
| Course Contents (Excerpt): | - CFC as uniform graphic configuration tool of the technologist for different targets<br>- Placing, connecting, parameter assignment and setting up sequencer properties of blocks<br>- Compiling, loading, test mode<br>- Structure and import of user-defined blocks for S7/M7 |

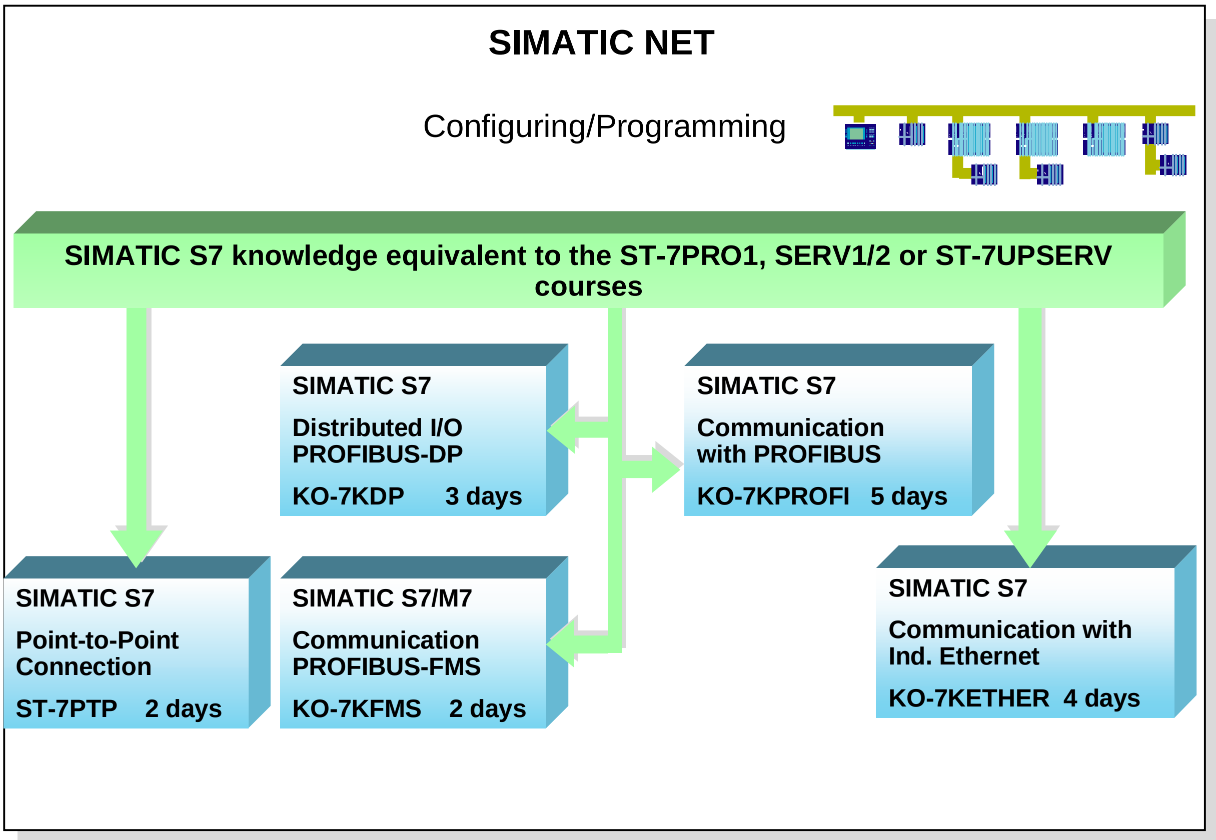



SIMATIC S7


Date: 12.03.03
File: PRO1_14E.11

**SITRAIN** Training for Automation and Drives

| | | |
|---|---|---|
| **ST-BWINCCS** | **SIMATIC WinCC, Systemkurs** |
| Course Contents (Excerpt): | - WinCC system overview<br>- Windows 95 (settings), benefits of the standard Windows interfaces*<br>- Creating a project, PLC connections, variable simulation, graphics<br>- Message display, message archiving<br>- Trend display, measured value archiving, user archives<br>- Report system*, background processing (Global Scripts)*<br>- API open user interface (benefits and structure) |
| **ST-BWINCCV** | **SIMATIC WinCC, Indepth Course** |
| Course Contents (Excerpt): | - Configuring WindowsNT Server and/or WindowsNT WS with WinCC Server<br>- Creating your own dynamic wizards<br>- Picture block configuration<br>- Database accesses, Option package: Storage<br>- Client-Server operation / Distributed Server and Multiclient<br>- OPC Server / Client<br>- Option package: WEB Server / WEB Clients |
| **ST-BWINCCE** | **SIMATIC WinCC Offenheit E** |
| Course Contents (Excerpt): | - Introduction to the WinCC system architecture (open interfaces and integration capability, databases, channel DLL, Global Scripts), introduction to Visual C++, ODK (development environment for API), WinCC-API structuring, using API functions, |
| **ST-BWINCCN** | **SIMATIC WinCC Offenheit N** |
| Course Contents (Excerpt): | - Brief introduction to the WinCC system architecture (open interfaces and integration capability, databases, channel DLL), general introduction to Global Scripts, accessing WinCC databases with Excel, OLE functions |
| **ST-BPROPRS** | **SIMATIC ProTool/Pro Systemkurs** |
| Course Contents (Excerpt) | - SIMATIC ProTool/Pro system overview<br>- Basic principles of screen layout<br>- User functions (VB-Script)<br>- Message configuration, display, archiving<br>- Trend configuration, display, measured value archiving |




SIMATIC S7


Date: 12.03.03
File: PRO1_14E.12

SITRAIN Training for Automation and Drives

**ST-7PTP** — **SIMATIC S7, Point-to-Point Connection**

Course Contents (Excerpt):

- Performance features and technical specifications of CP340 and CP441
- Creating configuration and parameter assignments of the communication processors
- Writing user programs for CP340 and CP441
- Diagnostic facilities of CP340 and CP441

**KO-7KDP** — **SIMATIC S7, PROFIBUS-DP**

Course Contents (Excerpt):

- Structure and functional principle of distributed I/O
- Planning and configuring the DP-Master in SIMATIC S7
- User programming and diagnostic facilities

**KO-7KFMS** — **SIMATIC S7, PROFIBUS-FMS**

Course Contents (Excerpt)

- How the FMS works
- NCM configuration software for PROFIBUS
- Programming FMS applications
- Diagnosis and test possibilities

**KO-7KPROFI** — **SIMATIC S7, PROFIBUS-DP/FMS**

Contents of the KO-7KDP and KO-7KFMS courses

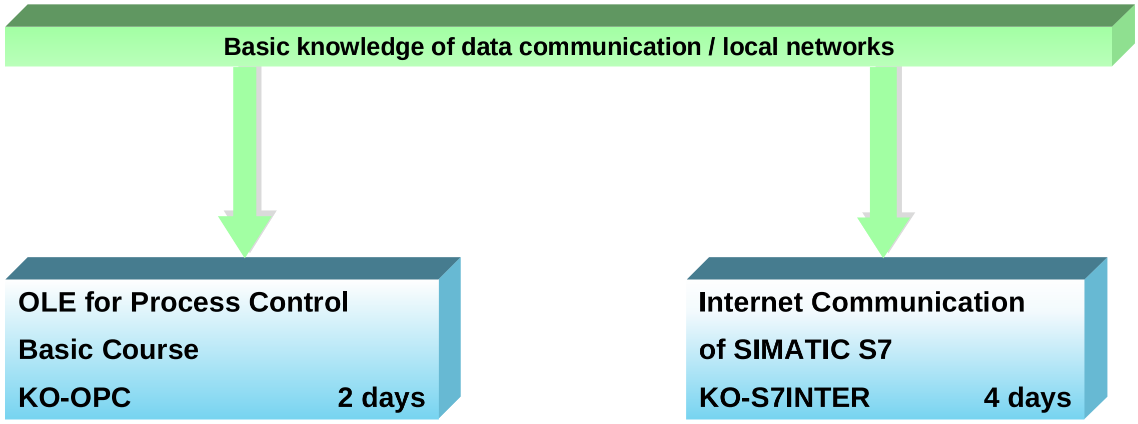
**KO-7KETHER** — **SIMATIC S7, Industrial Ethernet**

Course Contents (Excerpt):

- Mode of operation, properties and components of the Industrial Ethernet bus system
- ISO and TCP/IP protocols
- Configuration using the NCM-S7 configuration software for Industrial Ethernet
- Diagnostic functions

## SIMATIC NET - New Techniques

Planning engineer, Commissioning
engineer and User




SIMATIC S7


Date: 12.03.03
File: PRO1_14E.13

SITRAIN Training for
Automation and Drives

| | |
|---|---|
| **KO-OPC** | **OLE for Process Control, Basic Course** |
| Course Contents (Excerpt): | - OPC goals and background<br>- OLE basics (architecture of the NET software)<br>- Installation of the NET hardware and software components<br>- Implementation example based on the SIMATIC NET OPC products<br>  - S7-OPC Server<br>  - DP-OPC Server |
| **KO-S7INTER** | **Internet Communication of the SIMATIC S7** |
| Course Contents (Excerpt): | - Industrial Ethernet mode of operation and networking components<br>- IT communication processor<br>- Using the Internet in automation<br>- Introduction to network protocols<br>- The Internet and the Internet protocol familyTCP/IP<br>- e-mail and File Transfer Protocol<br>- Introduction to HTML for creating Web pages for the IT communication processor |

## Actuator-Sensor-Interface

Planning, Programming, and Commissioning engineers,
Installation, Maintenance and Service personnel

**SIMATIC S5 and/or SIMATIC S7 knowledge and basic knowledge of data communication**

**Actuator-Sensor-Interface**

**KO-ASI** **3 days**

SIMATIC S7


Date: 12.03.03
File: PRO1_14E.14

SITRAIN Training for Automation and Drives

| | |
|---|---|
| **KO-ASI** | **Actuator-Sensor-Interface** |
| Course Contents (Excerpt): | - Basics of the Actuator-Sensor-Interface (AS-Interface)<br>- Structure and configuration<br>- Introduction to the system components<br>- AS-Interface Master<br>- AS-Interface Slaves, Modules<br>- As-Interface power supplies, cables and accessories<br>- Commissioning, testing, diagnosis possibilities<br>- Addressing device<br>- PROFIBUS DP/AS-Interface transitions (DP/AS-i Link)<br>- Service and diagnosis with SCOPE for AS-Interface<br>- Practical exercises |

SIEMENS

## SIMATIC S5 - System and Supplementary Courses

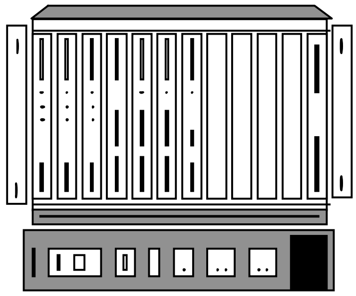
**Planning and Programming Engineers, Installation and Service Personnel**

**Basic knowledge of control engineering**

**SIMATIC S5**
**System Training Part 1**
**ST-S5SYS1 5 days**

**SIMATIC S5**
**System Training Part 2**
**ST-S5SYS2 5 days**

**SIMATIC S5**
**Service Training**
**ST-S5SERV 5 days**



**SIMATIC S5 - Supplementary Courses**

SIMATIC S7

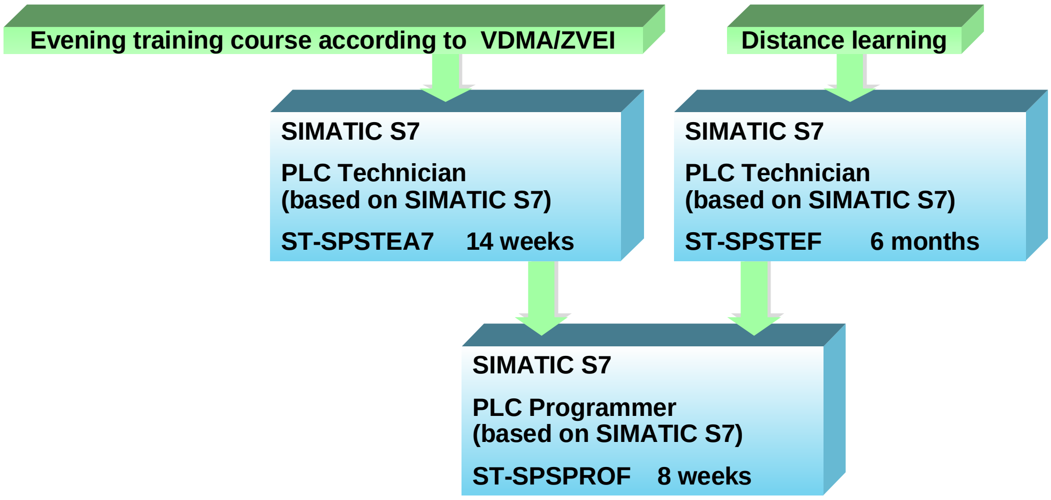

Date: 12.03.03
File: PRO1_14E.15

SITRAIN Training for Automation and Drives

| | |
|---|---|
| **System Courses** | System training<br>Service training<br>Compact training for engineers<br>Programming courses |
| **Supplementary Courses** | Project planning/configuring<br>Sequential control with GRAPH 5<br>Fail-safe, hot backup systems<br><br>Digital control engineering<br>Software-based closed-loop control<br><br>Point-to-point connections<br>L1 bus communication<br>SIMATIC S5, PROFIBUS<br>CP 5431 FMS workshop<br>S5-95/PROFIBUS workshop<br>SIMATIC S5, Industrial Ethernet |
| **Course Contents** | For full details of the contents of these courses please refer to our ITC catalog. You can obtain this catalog through your trainer or order it yourself from the Course Office direct.<br>Alternatively you can obtain information via:<br>Internet: **http://www.sitrain.com**<br>Info Line: Tel: **01805 23 56 11**<br>Fax: **01805 23 56 12** |

# PLC Technician and SIMATIC S7

**Installation, Service and Operating Personnel**




SIMATIC S7


Date: 12.03.03
File: PRO1_14E.16

**SITRAIN** Training for Automation and Drives

**Qualification for PLC Technicians**

Our structured training course for PLC technicians is based on the requirements of the VDMA/ZVEI [1)].

These requirements define what a skilled PLC technician needs to know and be able to do, regardless of the brand of PLC and from the point of view of the user.

Training can take the form of:

- **distance learning**,
- **evening courses** or
- a sequence of **daytime courses**.

Trainees can take an examination at the end of this training sequence. This normally takes one day and consists of a theory section and a practical section.

For further information, please see our ITC catalog or the special information sheet about training for PLC technicians.

[1)] Association of German Machine and Plant Manufacturers (VDMA)
Association of the Electrical and Electronics Industry (ZVEI)

**Upgrade**

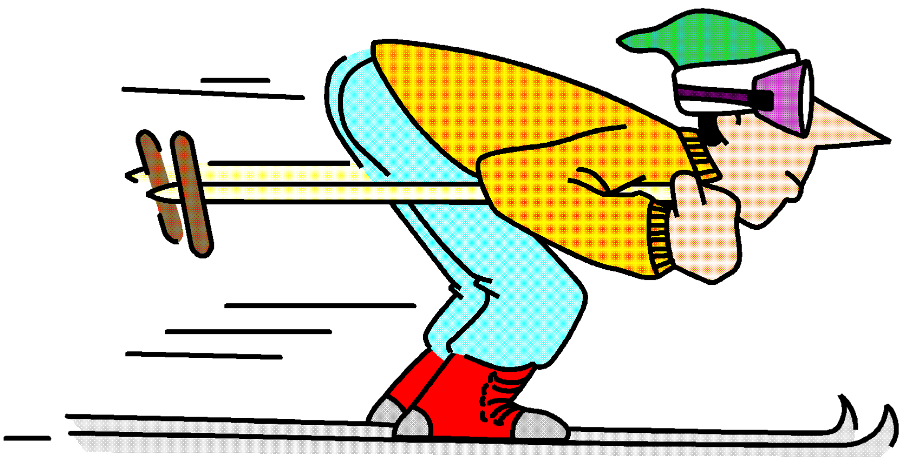
PLC technicians who have completed their training with SIMATIC S5 can obtain a further qualification in an upgrade course dealing with the differences between SIMATIC S5 and SIMATIC S7.

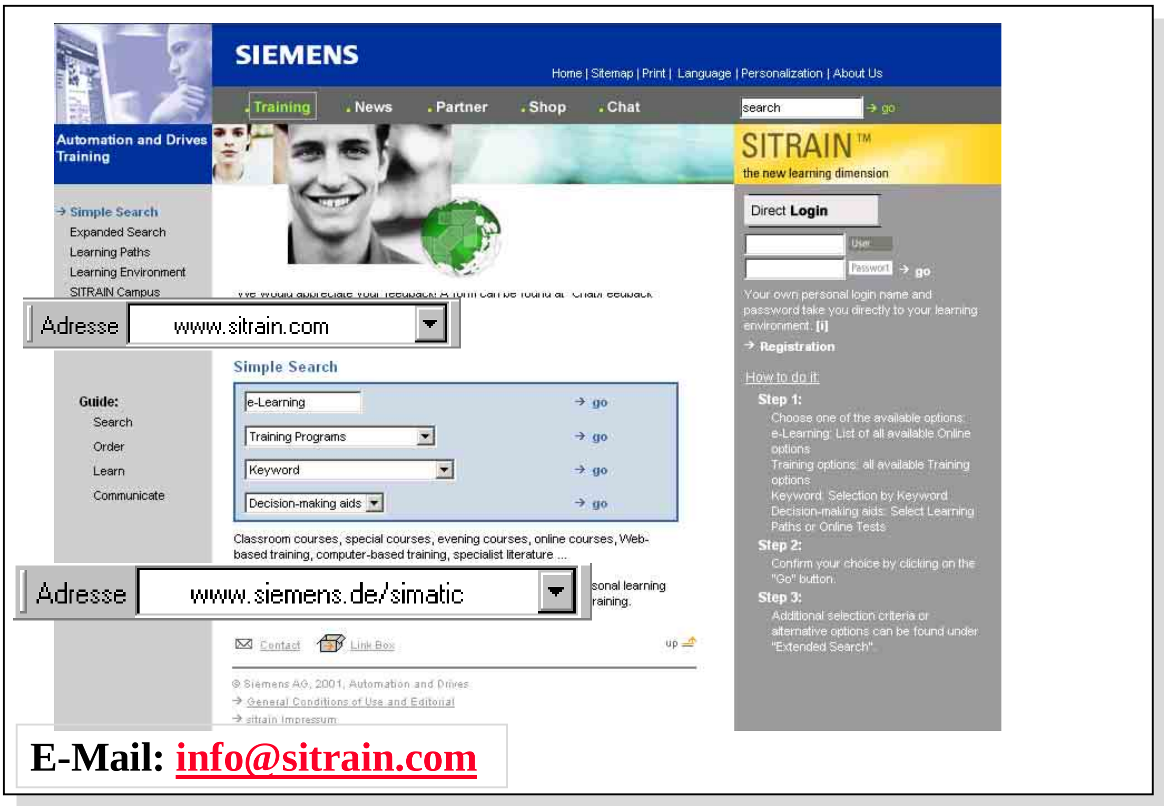

SIMATIC S7


Date: 12.03.03
File: PRO1_14E.17

**SITRAIN** Training for Automation and Drives

Now it's your turn ...





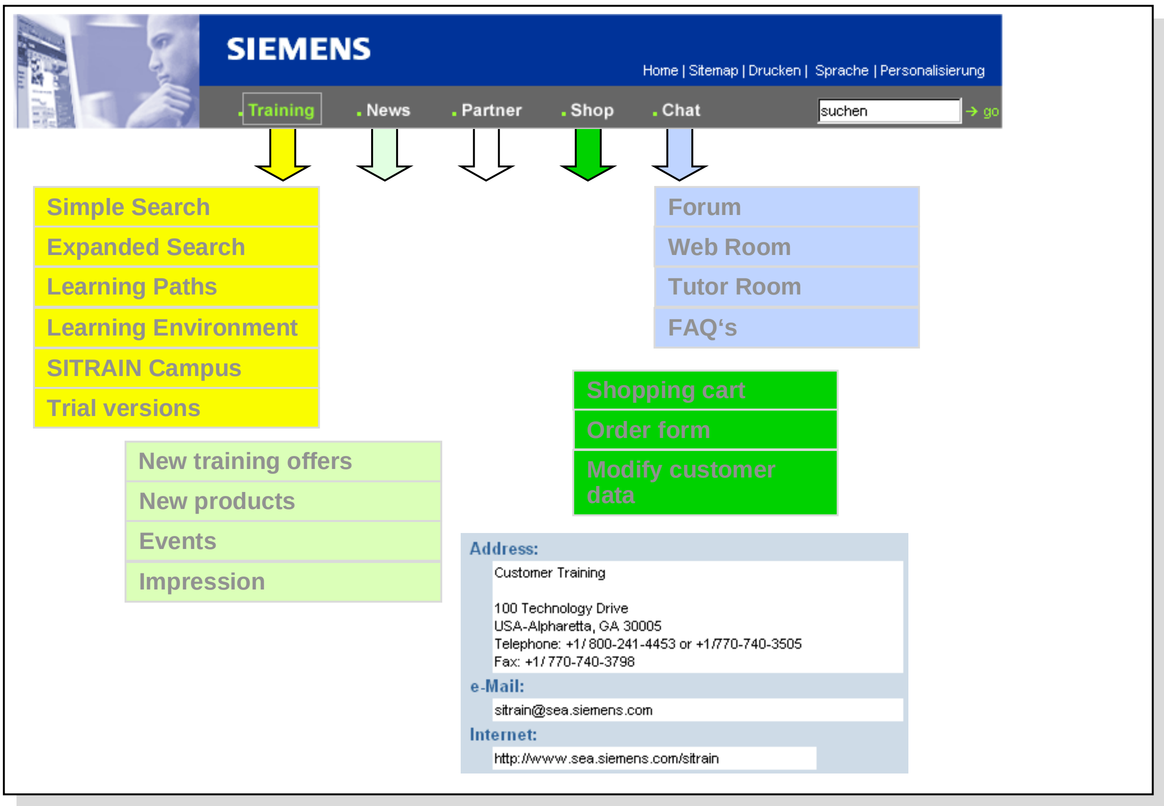

SIMATIC S7


Date: 12.03.03
File: PRO1_14E.18

SITRAIN Training for Automation and Drives

## SITRAIN™ *online* - die A&D Lernplattform im WWW

The future of training for Siemens AG Automation and Drives began at the end of November 2000. SITRAIN™online, the Internet based learning platform, was launched and offers a wide spectrum of new features and possibilities to learn "on demand" and "Just-in-time"...




SIMATIC S7

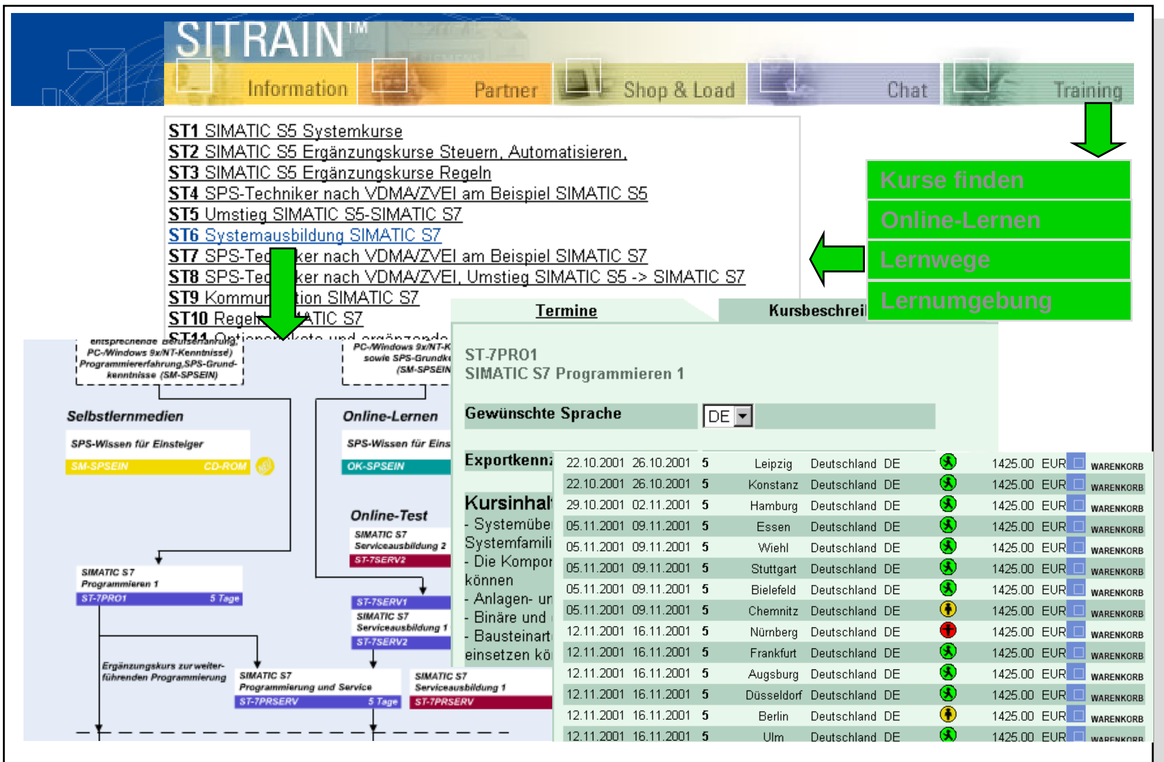
Date: 12.03.03
File: PRO1_14E.19
SITRAIN Training for Automation and Drives

**The SITRAIN™ Trademark**

The SITRAIN™ trademark is your guarantee of qualified training in over 200 locations in 60 countries worldwide. More than 70,000 course participants trust the know how of the market leader for Automation and Drives. With the start of SITRAIN™online, everyone from Argentina to Cyprus has the opportunity of planning his/her professional future from home. User-oriented online learning units let you design your future yourself.

Information modules, online modules, online courses and online course studies in the virtual classroom, synchronous and asynchronous communication areas in Chat, self-learning media, demo versions and technical manuals in the Shop, inter-active learning paths, online test modules, intelligent solution-oriented assistence programs, individualized learning environments, learning progess tests, SITRAIN™online offers innovative media as a state of the art solution. These innovative media have one goal in mind: the optimum learning success of the customer.

The most effective form of learning is surely the online course study. This type of course consists of a mix of media that can be worked on as a self study or in virtual classrooms, alone or in groups. We can offer you a complete learning path to PLC Professional Programming. This consists of online course studies for beginners, advanced and experts. It is completed with a certificate as: Siemens Certified PLC Professional. The highlight of this course study is the audio-based live tutorial on the Internet. Tutor and course participants communicate with one another at fixed intervals in a virtual classroom, work in groups, use application sharing and inter-active exercises using the STEP7 Distance Learning Software etc., just as in a "normal" classroom course. The participant receives all the necessary media, technical tools and documentation after he/she registers for the course. All you need is a Windows PC with Internet access at at least 28.8 kB/s.



A mouse click on a course in the training path gives you information about the course contents and dates.

SIMATIC S7


Date: 12.03.03
File: PRO1_14E.20

SITRAIN Training for Automation and Drives

**Learning Paths**

Do you need help with your training and continued education planning?
You haven‘t been able to make a decision?
With the interactive learning paths, we‘ll give you a little bit of help with your planning.

**Select a learning path (Germany)**

If you should still have questions or wishes, you‘ll find further information on the "Partner" page.

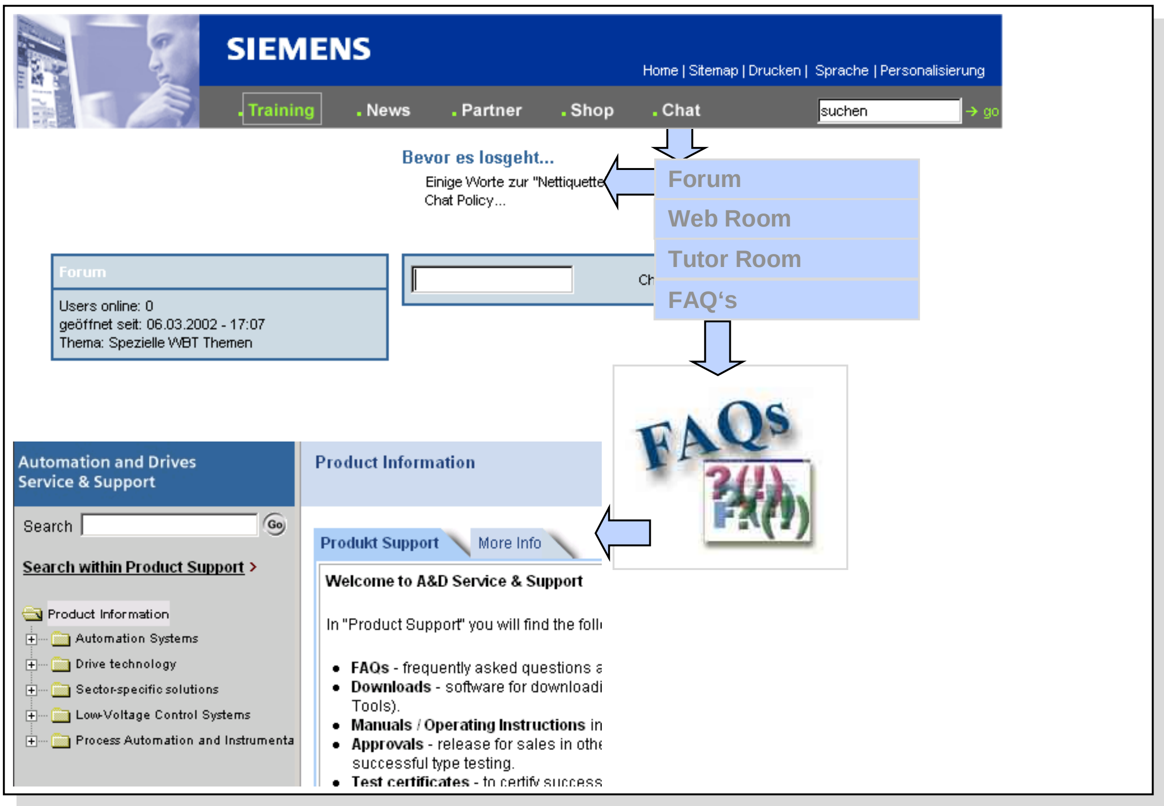
Here, please select a topic area.

**Information Technology in Automation**

**Automation Systems and Components**

**Machine tool, Positioning Control and Drives**

**Field Technology, Process, Power Station Automation and Host Systems**




SIMATIC S7


Date: 12.03.03
File: PRO1_14E.21

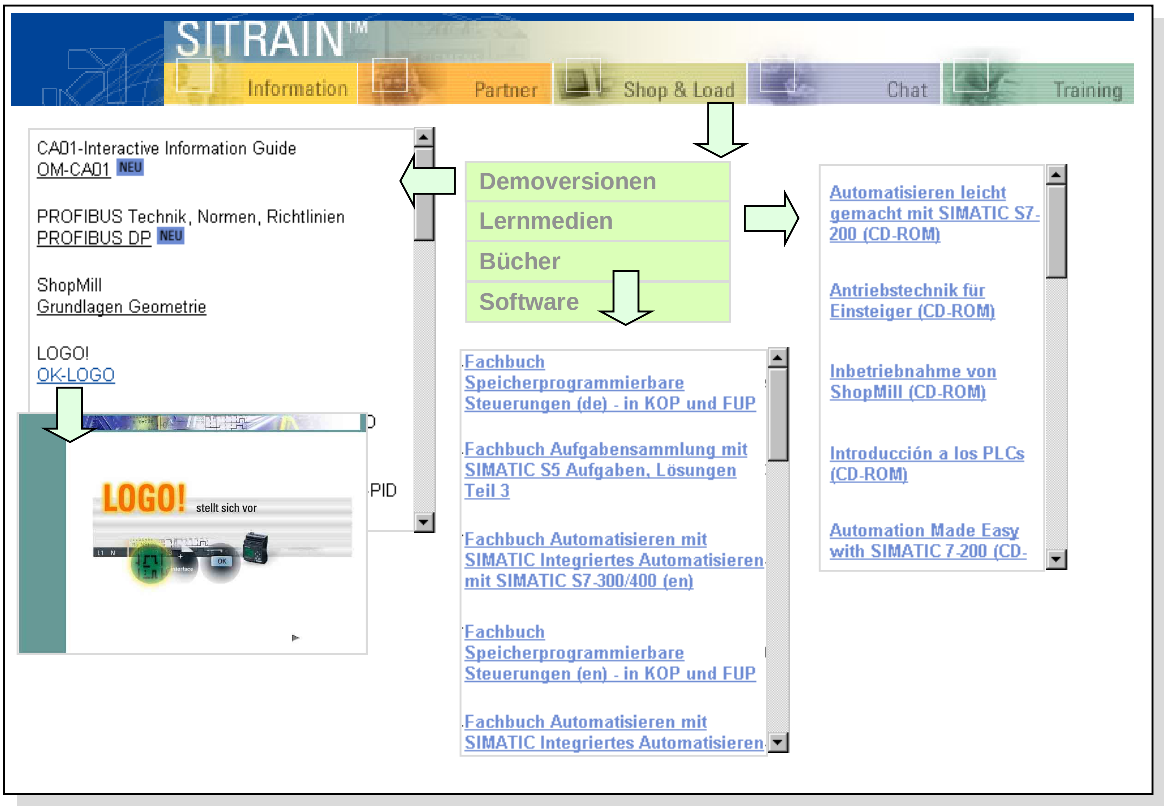
**SITRAIN** Training for Automation and Drives

**Chat**

Discussing round-the-clock and worldwide, exchanging opinions, asking questions, meeting and chatting online. SITRAIN™online - Chat, the latest forum for course participants and interested parties.

**Forum** - the open chatroom
**Web Room** - the chatroom for registered WBT users
**Tutor Room** - course-oriented chatroom
**FAQ's** - your questions, our answers

**Note:**
The chatrooms are open to all but are monitored by our chat administrator. Please be nice and follow the guidelines for behavior inside the chatroom.




SIMATIC S7


Date: 12.03.03
File: PRO1_14E.22

**SITRAIN** Training for Automation and Drives

**Shop&Load**

Are you interested in demo versions, self-learning CD-ROMs, technical literature, software tools or simulation software?

You‘ll find what you‘re looking for quickly and easily in Shop&Load. Simply make your selection from the Article Lists, place your order in the Shopping Cart, register, and pay easily with credit card or by invoice. The articles ordered will be sent to you within 14 days.