Hans Berger

Automating with STEP 7 in STL and SCL

SIMATIC S7-300/400
Programmable Controllers

SIEMENS

Fifth Edition

Berger Automating with STEP 7 in STL and SCL

Automating with STEP 7 in STL and SCL

Programmable Controllers
SIMATIC S7-300/400

by Hans Berger

6th revised and enlarged edition, 2012

Publicis Publishing

Bibliographic information published by the Deutsche Nationalbibliothek

The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at http://dnb.d-nb.de.

This book contains one Trial DVD. **“SIMATIC STEP 7 Professional, Edition 2010 SR1, Trial License”** encompasses: SIMATIC STEP 7 V5.5 SP1, S7-GRAPH V5.3 SP7, S7-SCL V5.3 SP6, S7-PLCSIM V5.4 SP5 and can be used for trial purposes for 14 days.

This Software can only be used with the Microsoft Windows XP 32 Bit Professional Edition SP3 or Microsoft Windows 7 32/64 Bit Professional Edition SP1 or Microsoft Windows 7 32/64 Bit Ultimate Edition SP1 operating systems.

Additional information can be found in the Internet at:
http://www.siemens.com/sce/contact
http://www.siemens.com/sce/modules
http://www.siemens.com/sce/tp

The programming examples concentrate on describing the STL and SCL functions and providing SIMATIC S7 users with programming tips for solving specific tasks with this controller.

The programming examples given in the book do not pretend to be complete solutions or to be executable on future STEP 7 releases or S7-300/400 versions. Additional care must be taken in order to comply with the relevant safety regulations.

Publicis Publishing
P.O. Box 3240
91050 Erlangen
E-mail: publishing-distribution@publicis.de

Internet: www.publicis-books.de

ISBN 978-3-89578-412-5

6th edition, 2012

Editor: Siemens Aktiengesellschaft, Berlin and Munich
Publisher: Publicis Publishing, Erlangen
© 2012 by Publicis Erlangen, Zweigniederlassung der PWW GmbH

Preface

The SIMATIC automation system unites all the subsystems of an automation solution under a uniform system architecture to form a homogeneous whole from the field level right up to process control. This Totally Integrated Automation (TIA) enables integrated configuring and programming, data management and communications throughout the complete automation system.

As the basic tool for SIMATIC, STEP 7 plays an integrating role in Totally Integrated Automation. STEP 7 is used to configure and program the SIMATIC S7, SIMATIC C7 and SIMATIC WinAC automation systems. Microsoft Windows has been chosen as the operating system to take advantage of the familiar user interface of standard PCs as also used in office environments.

For block programming STEP 7 provides programming languages that comply with DIN EN 6.1131-3: STL (statement list; an Assembler-like language), LAD (ladder logic; a representation similar to relay logic diagrams), FBD (function block diagram) and the S7-SCL optional package (Structured Control Language, a Pascal-like high-level language). Several optional packages supplement these languages: S7-GRAPH (sequential control), S7-HiGraph (programming with state-transition diagrams) and CFC (connecting blocks; similar to function block diagram). The various methods of representation allow every user to select the suitable control function description. This broad adaptability in representing the control task to be solved significantly simplifies working with STEP 7.

This book describes the STL and SCL programming languages for S7-300/400. As a valuable supplement to the description of the languages, and following an introduction to the S7-300/400 automation system, it provides valuable, practice-oriented information on the basic handling of STEP 7 when configuring, networking and programming SIMATIC PLCs. The description of the “Basic Functions” of a binary control, such as logic operations or latching/unlatching functions, makes it particularly easy for first-time users or users changing from relay contactor controls to become acquainted with STEP 7. The digital functions explain how digital values are combined; for example, basic calculations, comparisons or data type conversion.

The book shows how you can control program processing (program flow) and design structured programs. In addition to the cyclically processed main program, you can also incorporate event-driven program sections as well as influence the behavior of the controller at startup and in the event of errors/faults.

One section of the book is dedicated to the description of the SCL programming language. SCL is especially suitable for programming complex algorithms or for tasks in the data management area, and it supplements STL towards higher-level programming languages. The book concludes with the description of a program for converting STEP 5 programs to STEP 7 programs, and a general overview of the system functions and the function set for STL and SCL.

The contents of this book describe Version 5.5 of the STEP 7 programming software and Version 5.3 SP5 of the S7-SCL optional package.

Nuremberg, May 2012

Hans Berger

The Contents of the Book at a Glance

The Programming Examples

The present book provides many figures representing the use of the STL and SCL programming languages. All programming examples can be downloaded from the publisher's website www.publicis-books.de. There are two libraries, one for STL examples (STL_Book) and one for SCL examples (SCL_Book). When dearchived with the Retrieve function, these libraries occupy approximately 2.9 or 1.7 MB (dependent on the PG/PC file system used).

The library STL_Book contains eight programs that are essentially illustrations of the STL method of representation. Two extensive examples show the programming of functions, function blocks and local instances (Conveyor Example) and the handling of data (Message Frame Example). All the examples exist as source files and contain symbols and comments.

The library SCL_Book contains five programs with representations of the SCL statements and

Library STL_Book

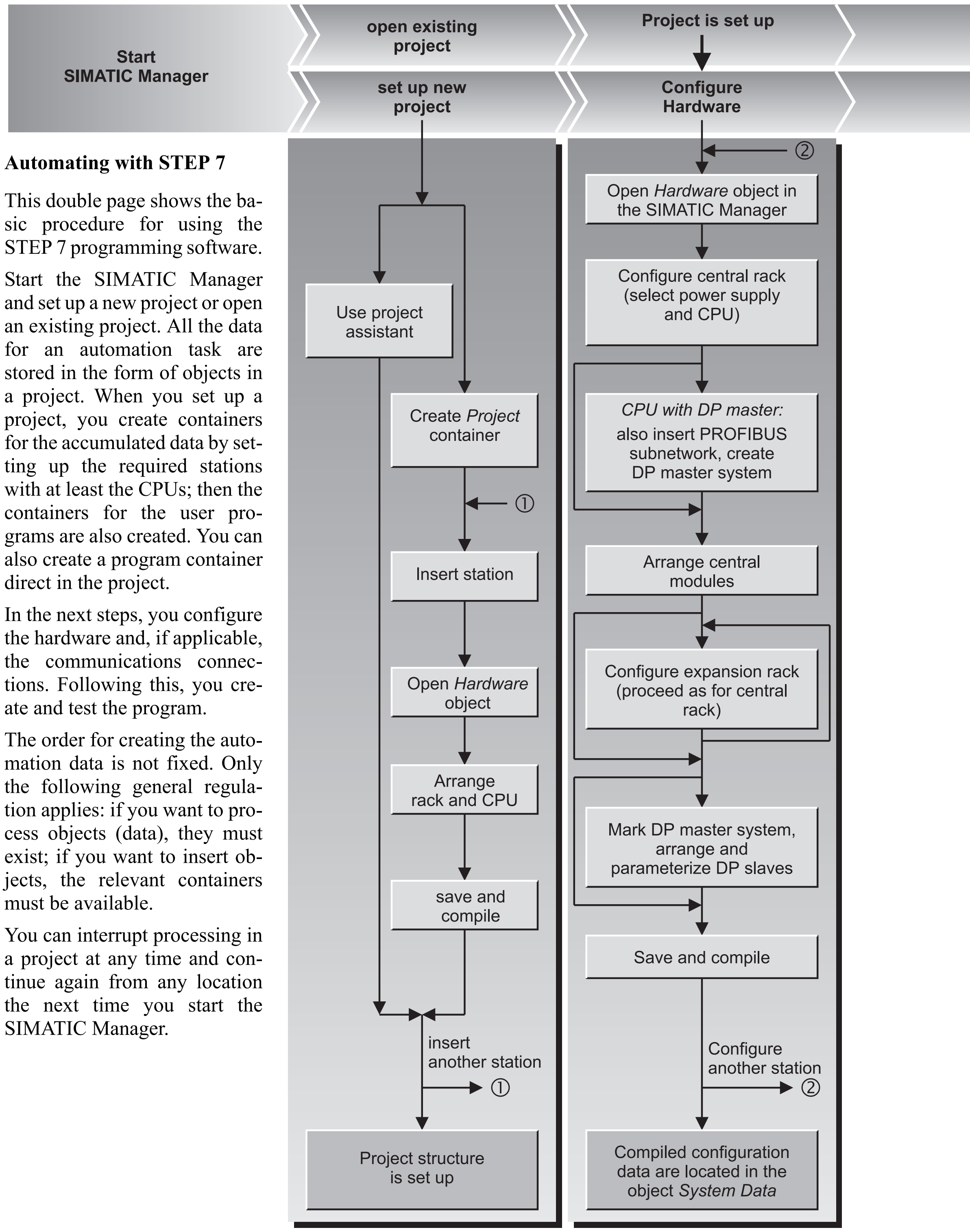
Basic Functions Examples of STL representation	**Program Processing** Examples of SFC Calls
FB 104 Chapter 4: Binary Logic Operations FB 105 Chapter 5: Memory Functions FB 106 Chapter 6: Transfer Functions FB 107 Chapter 7: Timer Functions FB 108 Chapter 8: Counter Functions	FB 120 Chapter 20: Main Program FB 121 Chapter 21: Interrupt Handling FB 122 Chapter 22: Restart Characteristics FB 123 Chapter 23: Error Handling
Digital Functions Examples of STL representation	**Variable Handling** Examples of Data Types and Variable Processing
FB 109 Chapter 9: Comparison Functions FB 110 Chapter 10: Arithmetic Functions FB 111 Chapter 11: Math Functions FB 112 Chapter 12: Conversion Functions FB 113 Chapter 13: Shift Functions FB 114 Chapter 14: Word Logic	FB 124 Chapter 24: Data Types FB 125 Chapter 25: Indirect Addressing FB 126 Chapter 26: Direct Variable Access FB 101 Elementary Data Types FB 102 Complex Data Types FB 103 Parameter Types
Program Flow Control Examples of STL representation	**Conveyor Example** Examples of Basic Functions and Local Instances
FB 115 Chapter 15: Status Bits FB 116 Chapter 16: Jump Functions FB 117 Chapter 17: Master Control Relay FB 118 Chapter 18: Block Functions FB 119 Chapter 19: Block Parameters Source File Block Programming (Chapter 3)	FC 11 Conveyor Belt Controller FC 12 Counter Control FB 20 Feed FB 21 Conveyor Belt FB 22 Parts Counter
Message Frame Example Handling Data examples	**General Examples**
UDT 51 Data Structure Header UDT 52 Data Structure Message Frame FB 51 Generate Message Frame FB 52 Save Message Frame FC 61 Clock Check FC 62 Generate Checksum FC 63 Convert Date	FC 41 Range Monitor FC 42 Limit Value Detection FC 43 Compound Interest Calculation FC 44 Double-Word-Wise Edge Evaluation FC 45 Converting S5 Floating-Point to S7 REAL FC 46 Converting S7 REAL to S5 Floating-Point FC 47 Copy Data Area (ANY Pointer)

the SCL functions. The programs “Conveyor Example” The library SCL_Book contains five programs with representations of the SCL statements and the SCL functions. The programs “Conveyor Example” and “Message Frame Example” and “Message Frame Example” show the same functions as the STL examples of the same name. The program “General Examples” contains SCL functions for processing complex data types, data storage and – for SCL programmers – a statement for programming simple STL functions for SCL programs.

To try the programs out, set up a project corresponding to your hardware configuration and then copy the program, including the symbol table from the library to the project. Now you can call the example programs, adapt them for your own purposes and test them online.

Library SCL_Book

27 Language Elements Examples of SCL Representation (Chapter 27)	**30 SCL Functions** Examples of SCL Representation (Chapter 30)
FC 271 Delimiter Example OB 1 Main Program for the Delimiter Example FB 271 Operators, Expressions, Assignments FB 272 Indirect Addressing	FB 301 Timer Functions FB 302 Counter Functions FB 303 Conversion Functions FB 304 Math Functions FB 305 Shifting and Rotating
28 Control Statements Examples of SCL Representation (Chapter 28)	**31 IEC Functions** Examples of SCL Representation (Chapter 31)
FB 281 IF Statement FB 282 CASE Statement FB 283 FOR Statement FB 284 WHILE Statement FB 285 REPEAT Statement	FB 311 Conversion Functions FB 312 Comparison Functions FB 313 String Functions FB 314 Date/Time-of-day Functions FB 315 Numerical Functions
29 SCL Block Calls Examples of SCL Representation (Chapter 29)	**General Examples**
FC 291 FC Block with Function Value FC 292 FC Block without Function Value FB 291 FB Block FB 292 Example Calls for FC and FB Blocks FC 293 FC Block for EN/ENO Example FB 293 FB Block for EN/ENO Example FB 294 Calls for EN/ENO Examples	FC 61 DT_TO_STRING FC 62 DT_TO_DATE FC 63 DT_TO_TOD FB 61 Variable Length FB 62 Checksum FB 63 Ring Buffer FB 64 FIFO Register STL Functions for SCL Programming
Conveyor Example Examples of Basic Functions and Local Instances	**Message Frame Example** Handling Data examples
FC 11 Conveyor Belt Controller FC 12 Counter Control FB 20 Feed FB 21 Conveyor Belt FB 22 Parts Counter	UDT 51 Data Structure Header UDT 52 Data Structure Message Frame FB 51 Generate Message Frame FB 52 Save Message Frame FC 61 Clock Check


Start
SIMATIC Manager
open existing project
set up new project
Project is set up
Configure Hardware
Use project assistant
Create *Project* container
①
Insert station
Open *Hardware* object
Arrange rack and CPU
save and compile
insert another station
①
Project structure is set up
②
Open *Hardware* object in the SIMATIC Manager
Configure central rack (select power supply and CPU)
CPU with DP master: also insert PROFIBUS subnetwork, create DP master system
Arrange central modules
Configure expansion rack (proceed as for central rack)
Mark DP master system, arrange and parameterize DP slaves
Save and compile
Configure another station
②
Compiled configuration data are located in the object *System Data*

Automating with STEP 7

This double page shows the basic procedure for using the STEP 7 programming software.

Start the SIMATIC Manager and set up a new project or open an existing project. All the data for an automation task are stored in the form of objects in a project. When you set up a project, you create containers for the accumulated data by setting up the required stations with at least the CPUs; then the containers for the user programs are also created. You can also create a program container direct in the project.

In the next steps, you configure the hardware and, if applicable, the communications connections. Following this, you create and test the program.

The order for creating the automation data is not fixed. Only the following general regulation applies: if you want to process objects (data), they must exist; if you want to insert objects, the relevant containers must be available.

You can interrupt processing in a project at any time and continue again from any location the next time you start the SIMATIC Manager.

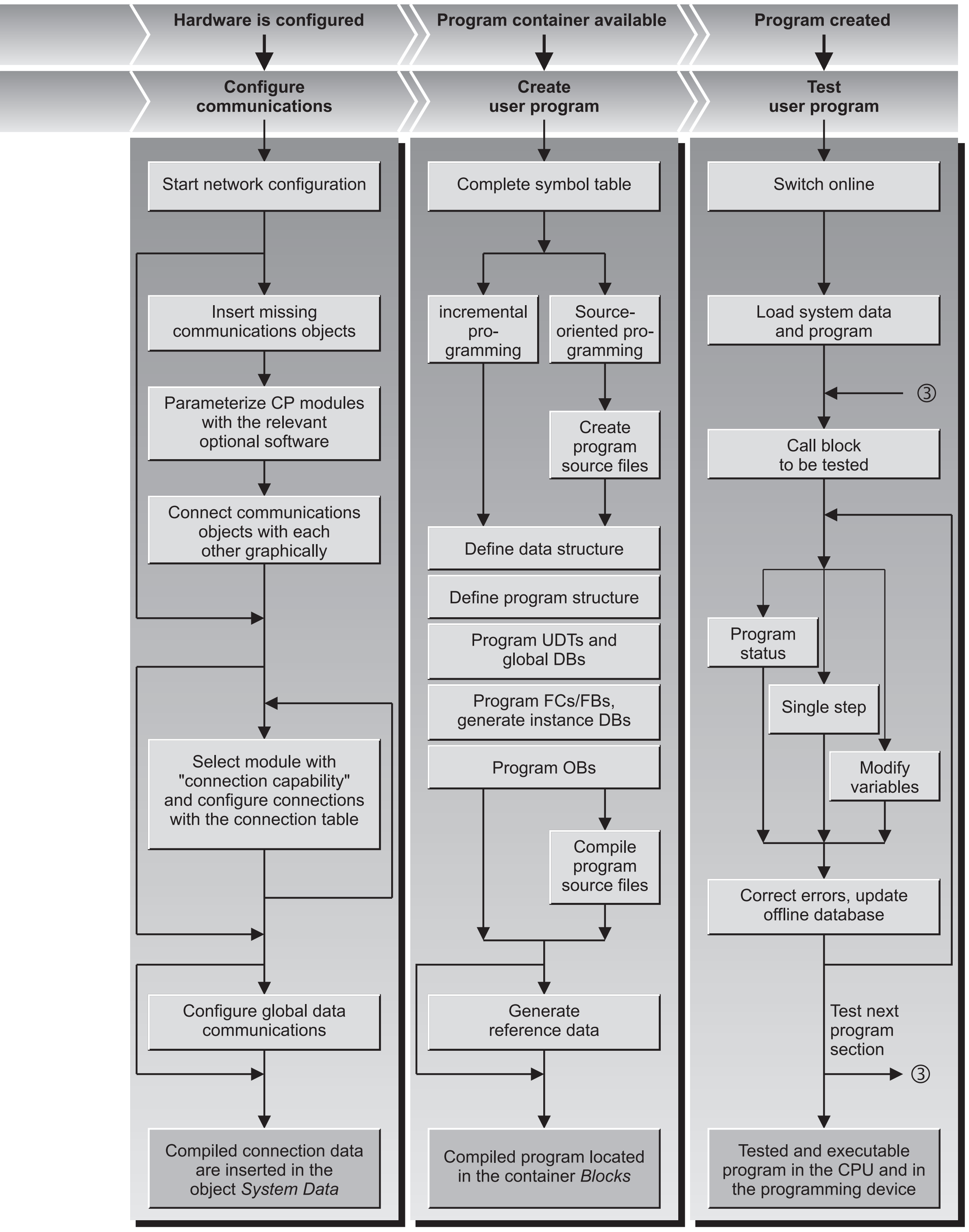

Hardware is configured
Configure communications
Start network configuration
Insert missing communications objects
Parameterize CP modules with the relevant optional software
Connect communications objects with each other graphically
Select module with "connection capability" and configure connections with the connection table
Configure global data communications
Compiled connection data are inserted in the object *System Data*
Program container available
Create user program
Complete symbol table
incremental pro-gramming
Source-oriented pro-gramming
Create program source files
Define data structure
Define program structure
Program UDTs and global DBs
Program FCs/FBs, generate instance DBs
Program OBs
Compile program source files
Generate reference data
Compiled program located in the container *Blocks*
Program created
Test user program
Switch online
Load system data and program
③
Call block to be tested
Program status
Single step
Modify variables
Correct errors, update offline database
Test next program section
③
Tested and executable program in the CPU and in the programming device

Table of Contents

Introduction

This section of the book provides an overview of the SIMATIC S7-300/400.

The **S7-300/400 programmable controller** is of modular design. The modules with which it is configured can be central (in the vicinity of the CPU) or distributed without any special settings or parameter assignments having to be made. In SIMATIC S7 systems, distributed I/O is an integral part of the system. The CPU, with its various memory areas, forms the hardware basis for processing of the user programs. A load memory contains the complete user program: the parts of the program relevant to its execution at any given time are in a work memory whose short access times are the prerequisite for fast program processing.

STEP 7 is the programming software for S7-300/400 and the automation tool is the SIMATIC Manager. The SIMATIC Manager is an application for the Windows operating systems from Microsoft and contains all functions needed to set up a project. When necessary, the SIMATIC Manager starts additional tools, for example to configure stations, initialize modules, and to write and test programs.

You formulate your automation solution in the STEP 7 programming languages. The **SIMATIC S7 program** is structured, that is to say, it consists of blocks with defined functions that are composed of networks or rungs. Different priority classes allow a graduated interruptibility of the user program currently executing. STEP 7 works with variables of various data types starting with binary variables (data type BOOL) through digital variables (e.g. data type INT or REAL for computing tasks) up to complex data types such as arrays or structures (combinations of variables of different types to form a single variable).

The first chapter contains an overview of the hardware of the S7-300/400 automation system and the second chapter contains the same overview of the STEP 7 programming software. The basis of the description is the functional scope for STEP 7 Version 5.5.

Chapter 3 “SIMATIC S7 Program” serves as an introduction to the most important elements of an S7 program and shows the programming of individual blocks in the programming languages STL and SCL. The functions and statements of STL and SCL are then described in the subsequent chapters of the book. All the descriptions are explained using brief examples.

1 SIMATIC S7-300/400 Programmable Controller

1.1 Structure of the Programmable Controller

1.1.1 Components

The SIMATIC S7-300/400 is a modular programmable controller comprising the following components:

▷ Racks;
Accommodate the modules and connect them to each other

▷ Power supply (PS);
Provides the internal supply voltages

▷ Central processing unit (CPU);
Stores and processes the user program

▷ Interface modules (IMs);
Connect the racks to one another

▷ Signal modules (SMs);
Adapt the signals from the system to the internal signal level or control actuators via digital and analog signals

▷ Function modules (FMs);
Execute complex or time-critical processes independently of the CPU

▷ Communications processors (CPs)
Establish the connection to subsidiary networks (subnets)

▷ Subnets
Connect programmable controllers to each other or to other devices

A programmable controller (or station) may consist of several racks, which are linked to one another via bus cables. The power supply, CPU and I/O modules (SMs, FMs and CPs) are plugged into the central rack. If there is not enough room in the central rack for the I/O modules or if you want some or all I/O modules to be separate from the central rack, expansion racks are available which are connected to the central rack via interface modules (Figure 1.1).

It is also possible to connect distributed I/O to a station (see Chapter 1.2.1 “PROFIBUS DP”).

The racks connect the modules with two buses: the I/O bus (or P bus) and the communication bus (or K bus). The I/O bus is designed for high-speed exchange of input and output signals, the communication bus for the exchange of large amounts of data. The communication bus connects the CPU and the programming device interface (MPI) with function modules and communications processors.

1.1.2 S7-300 Station

Centralized configuration

In an S7-300 controller, as many as 8 I/O modules can be plugged into the central rack. Should this single-tier configuration prove insufficient, you have two options for controllers equipped with a CPU 313 or a more advanced CPU:

▷ Either choose a two-tier configuration (with IM 365 up to 1 meter between racks)

▷ or choose a configuration of up to four tiers (with IM 360 and IM 361 up to 10 meters between racks)

You can operate a maximum of 8 modules in a rack. The number of modules may be limited by the maximum permissible current per rack, which is 1.2 A.

The modules are linked to one another via a backplane bus, which combines the functions of the P and K buses.

Local bus segment

A special feature regarding configuration is the use of the FM 356 application module. An FM 356 is able to “split” a module's backplane bus and to take over control of the remaining modules in the split-off “local bus segment”

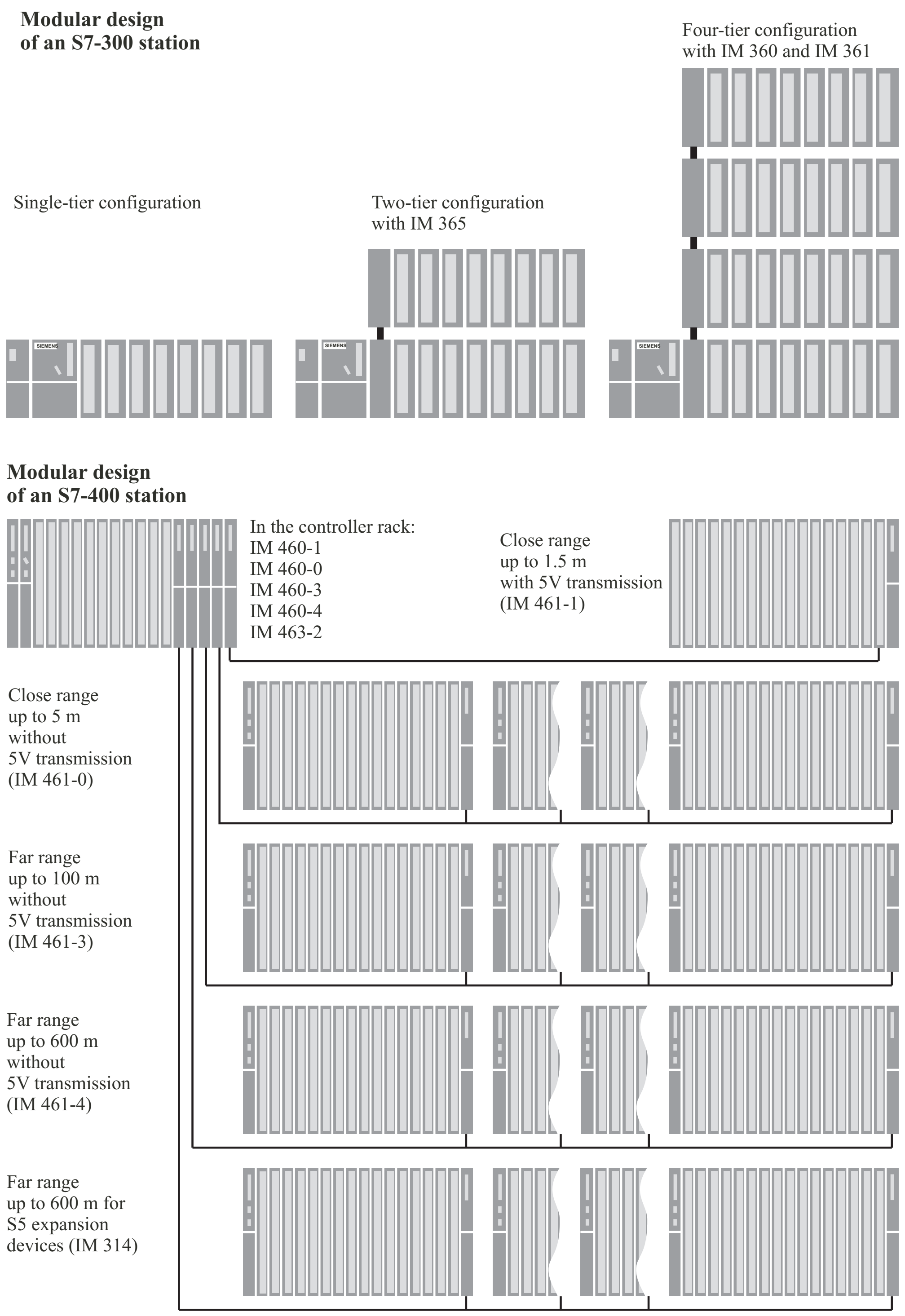


Figure 1.1 Hardware Configuration for S7-300/400

itself. The limitations mentioned above regarding the number of modules and the power consumption also apply in this case.

Standard CPUs

The standard CPUs are available in different versions with regard to memory size and processing speed. They range from the "smallest" CPU 312 for smaller applications with moderate processing speed requirements up to the CPU 319-3 PN/DP with a large program memory and fast program execution for cross-sector automation tasks. Equipped with the corresponding interfaces, some CPUs can be used as the central controller for the distributed I/O over PROFIBUS and PROFINET.

A micro memory card (MMC) is required for using the standard CPUs – as is the case with all innovated S7-300 CPUs. This memory medium opens up new application possibilities compared to the previously used memory card (see Chapter 1.1.6 "CPU Memory Areas").

The now discontinued CPU 318 can be replaced by the CPUs 317 or 319.

Compact CPUs

The 3xxC CPUs permit a compact design for mini programmable controllers. Depending on the version, they already contain:

- ▷ Integral I/O
  Digital and analog inputs/outputs
- ▷ Integral technological functions
  Counting, measurement, control, positioning
- ▷ Integral communications interfaces
  PROFIBUS DP master or slave, point-to-point coupling (PtP)

The technological functions are system blocks which use the onboard I/Os of the CPU.

Technology CPUs

The CPUs 3xxT combine open-loop control functions with simple motion control functions. The control section is designed as with a standard CPU. It is configured, parameterized and programmed using STEP 7. The technology objects and the motion control section require the S7-Technology option package that is integrated in the SIMATIC Manager following the installation.

The technology CPUs have a PROFIBUS DP interface which permits use as a DP master or DP slave. The CPUs are used for cross-sector automation tasks in series machine construction, special machine construction, and plant construction.

Fail-safe CPUs

The CPUs 3xxF are used in production plants with increased safety requirements. Corresponding PROFIBUS and PROFINET interfaces allow use of safety-related distributed I/O with the PROFIsafe bus profile (see "S7 Distributed Safety" under 1.1.5 "Safety-related SIMATIC"). Standard modules for normal applications can be used parallel to safety-related operation.

SIPLUS

The SIPLUS product family offers modules that can be used in harsh environments. The SIPLUS components are based on standard devices which have been specially converted for the respective application, for example for an extended temperature range, increased resistance to vibration and shock, or voltage ranges differing from the standard. Please therefore note the technical data for the respective SIPLUS module. In order to carry out the configuration with STEP 7, use the equivalent type (the standard module on which it is based); this is specified, for example, on the module's nameplate.

1.1.3 S7-400 Station

Centralized configuration

The controller rack of the S7-400 is available in the versions UR1 (18 slots), UR2 (9 slots) and CR3 (4 slots). UR1 and UR2 can also be used as expansion racks. The power supply and the CPU also occupy slots in the racks, possibly even two or more per module. If necessary, the number of available slots can be increased using expansion racks: UR1 and ER1 have 18 slots each, UR2 and ER2 have 9 slots each.

Using the IM 460-1 and IM 461-1 interface modules, one expansion rack per interface can be located up to 1.5 m away from the controller rack, and the 5 V supply is also transmitted. Up to 4 expansion ranks can also be operated via IM 460-0 and 461-0 in the local range up to 5 m. For longer distances, the IM 460-3 and IM 461-3 or the IM 460-4 and 461-4 enable up to 4 expansion racks to be operated up to 100 m or 600 m away.

A maximum of 21 expansion racks can be connected to a central rack. To distinguish between racks, you set the number of the rack on the coding switch of the receiving IM.

The backplane bus consists of a parallel P bus and a serial K bus. Expansion racks ER1 and ER2 are designed for “simple” signal modules which generate no process interrupts, do not have to be supplied with 24 V voltage via the P bus, require no back-up voltage, and have no K bus connection. The K bus is in racks UR1, UR2 and CR2 either when these racks are used as central racks or expansion racks with the numbers 1 to 6.

Connecting segmented rack

A special feature is the segmented rack CR2. The rack can accommodate two CPUs with a shared power supply while keeping them functionally separate. The two CPUs can exchange data with one another via the K bus, but have completely separate P buses for their own signal modules.

Multiprocessor mode

In an S7-400, as many as four specially designed CPUs in a suitable rack UR can take part in multiprocessor mode. Each module in this station is assigned to only one CPU, both with its address and its interrupts. See Chapters 20.3.6 “Multiprocessing Mode” and 21.7 “Multiprocessor Interrupt” for more details.

Connection of SIMATIC S5 modules

The IM 463-2 interface module allows you to connect S5 expansion units (EG 183U, EG 185U, EG 186U as well as ER 701-2 and ER 701-3) to an S7-400, and also allows centralized expansion of the expansion units. An IM 314 in the S5 expansion unit handles the link. You can operate all analog and digital modules allowed in these expansion units. An S7-400 can accommodate as many as four IM 463-2 interface modules; as many as four S5 expansion units can be connected in a distributed configuration to each of an IM 463-2's interfaces.

1.1.4 Fault-Tolerant SIMATIC

For applications with high fault tolerance demands for machines and processes, there are two versions of SIMATIC S7 fault-tolerant programmable controllers with a redundant design: software redundancy and S7-400H/FH.

Software redundancy

Using SIMATIC S7-300/400 standard components, you can establish a software-based redundant system with a master station controlling the process and a standby station assuming control in the event of the master failing.

Fault tolerance through software redundancy is suitable for slow processes because transfer to the standby station can require several seconds depending on the configuration of the programmable controllers. The process signals are “frozen” during this time. The standby station then continues operation with the data last valid in the master station.

Redundancy of the input/output modules is implemented with distributed I/O (ET 200M with IM 153-2 interface module for redundant PROFIBUS DP). The software redundancy can be configured with STEP 7 Version 5.2 and higher.

Fault-tolerant SIMATIC S7-400H

The SIMATIC S7-400H is a fault-tolerant programmable controller with redundant configuration comprising two central racks, each with an H CPU and a synchronization module for data comparison via fiber optic cable. Both controllers operate in “hot standby” mode; in the event of a fault, the intact controller assumes operation alone via automatic bumpless transfer. The UR2-H rack with 2x9 slots offers the possibility for also installing a fault-tolerant system in one single rack.

The I/O can have normal availability (single-channel, single-sided configuration) or enhanced availability (single-channel, switched configuration with ET 200M). Communication takes place with a single or redundant bus.

The user program is the same as for a non-redundant device; the redundancy function is provided exclusively by the hardware that is used and is kept hidden from the user. The software package required for configuration is included in STEP 7 from V5.3. The provided standard libraries *Redundant IO* contain blocks for supporting the redundant I/O.

1.1.5 Safety-related SIMATIC

Fail-safe programmable controllers control processes in which the safe state can be achieved by direct switching-off. They are used in plants with increased safety requirements.

The safety functions are mainly located in the safety-related user program of a correspondingly designed CPU and in the failsafe input and output modules. An F-CPU complies with the safety requirements up to AK 6 in accordance with DIN V 19250/DIN V VDE 0801, up to SIL 3 in accordance with IEC 61508, and up to Category 4 in accordance with EN 954-1. Safety functions can be executed parallel to a non-safety-related user program in the same CPU.

Safety-related communication over PROFIBUS DP – also over PROFINET IO with S7 Distributed Safety – uses the PROFIsafe bus profile. This permits transmission of safety-related and non-safety-related data on a single bus cable.

Safety Integrated for the manufacturing industry

S7 Distributed Safety is a failsafe automation system for the protection of machines and personnel mainly for applications with machine controls and in the process industry.

Controllers from the SIMATIC S7-300, S7-400, and ET 200S ranges are available as F-CPUs. The safety-related I/O modules are connected to S7-400 over PROFIBUS DP or PROFINET IO using the safety-related PROFIsafe bus profile. With S7-300 and ET 200S, use of safety-related I/O modules is additionally possible in the central rack.

The hardware configuration and programming of the non-safety-related user program are carried out using the standard applications of STEP 7.

The *SIMATIC S7 Distributed Safety* option package is required to program the safety-related parts of the program. With this option package you can use the F-LAD or F-FBD programming languages to create the blocks which contain the safety-related program. Interfacing to the I/O is carried out using the process image as with the standard program. S7 Distributed Safety also includes a library with TÜV-certified safety blocks. There is an additional library available with F-blocks for press and burner controls.

The safety-related user program can be executed parallel to the standard user program. If an error is detected in the safety-related part of the program, the CPU enters the STOP state.

Safety Integrated for the process industry

S7 F/FH Systems is a failsafe automation system based on S7-400 mainly for applications in the process industry. The safety-related I/O modules are connected over PROFIBUS DP using the safety-related PROFIsafe bus profile.

An S7-400 F-CPU is provided with the safety-related control functions by application of an *S7 F Systems Runtime license*. A non-safety-related user program can be executed parallel to the safety-related plant unit.

In addition to fail-safety, the S7-400FH also provides increased availability. If a detected fault results in a STOP of the master CPU, a reaction-free switch is made to the CPU running in hot standby mode. The *S7 H Systems* option package is additionally required for operation as S7-400FH.

The hardware configuration and programming of the non-safety-related user program are carried out using the standard applications of STEP 7.

The *S7 F Systems* option package is additionally required for programming the safety-related program parts, and additionally the *CFC* option package V5.0 SP3 and higher and the *S7-SCL* option package V5.0 and higher.

The safety-related program is programmed using CFC (Continuous Function Chart). Programmed, safety-related function blocks from the supplied F-library can be called and interconnected in this manner. Alongside functions for programming safety functions, they also include fault detection and fault reaction functions. This ensures that if there are failures or errors, the F-system can be stopped in or transferred to a safe mode. If a fault is detected in the safety program, the safety-related part of the plant is switched off, whereas the remaining part can continue to operate.

Fail-safe I/O

Failsafe signal modules (F-modules or F-submodules) are required for safety operation. The fail-safety is achieved through the integrated safety functions and the corresponding wiring of sensors and actuators.

The F-modules can also be used in standard applications with enhanced diagnostics requirements. Redundant F-modules can be used with S7 F/FH systems to increase the availability both in standard and safety-related operation.

The failsafe I/O is available in various versions:

- ▷ The fail-safe signal modules in S7-300 design are used in the ET 200M distributed I/O device or – with S7-Distributed Safety – also centrally.
- ▷ Failsafe I/O modules are available for the distributed I/O devices in the designs ET 200S, ET 200pro, and ET 200eco.
- ▷ Failsafe interface modules are also available as F-CPUs for the ET 200S and ET 200pro distributed I/O devices.
- ▷ Failsafe DP standard slaves and – with S7-Distributed Safety also IO standard devices – can be used which can handle the PROFIsafe bus profile.

Failsafe CPUs and signal modules are also available in SIPLUS design.

1.1.6 CPU Memory Areas

Figure 1.2 shows the memory areas in the programming device, in the CPU, and in the signal modules which are important for your program.

The programming device contains the *off-line data*. These consist of the user program (program code and user data), the system data (e.g. hardware configuration, network and connection configuration) and further project-specific data such as e.g. the symbol table and comments.

The *online data* consists of the user program and the system data on the CPU, stored in two storage areas: the load memory and the work memory. In addition, the system memory is also present here.

Finally, the I/O modules contain memories for the signal statuses of the inputs and outputs.

The central processing units have a slot for a plug-in *memory module*. This memory module also contains the load memory, or parts thereof (see "Physical design of CPU memory", further below). The memory module is designed as memory card (S7-400-CPUs) or as micro memory card (S7-300 CPUs and derived ET 200 CPUs). A firmware update for the CPU operating system can also be performed via the memory module.

Memory card

The memory submodule for the S7-400 CPUs is the memory card (MC). There are two types of memory card: RAM cards and flash EPROM cards.

If you only want to expand the load memory, use a RAM card. A RAM card allows you to modify the entire user program online. This is necessary, for example, for larger programs when testing and during commissioning. RAM memory cards lose their contents when unplugged.

If you want to protect your user program against power failure following testing and commissioning, including configuration data and module parameters, use a flash EPROM card. In this case, load the entire program offline onto the flash EPROM card with the card plugged into the programming device. With the relevant CPUs, you can also load the program online with the memory card plugged into the CPU.

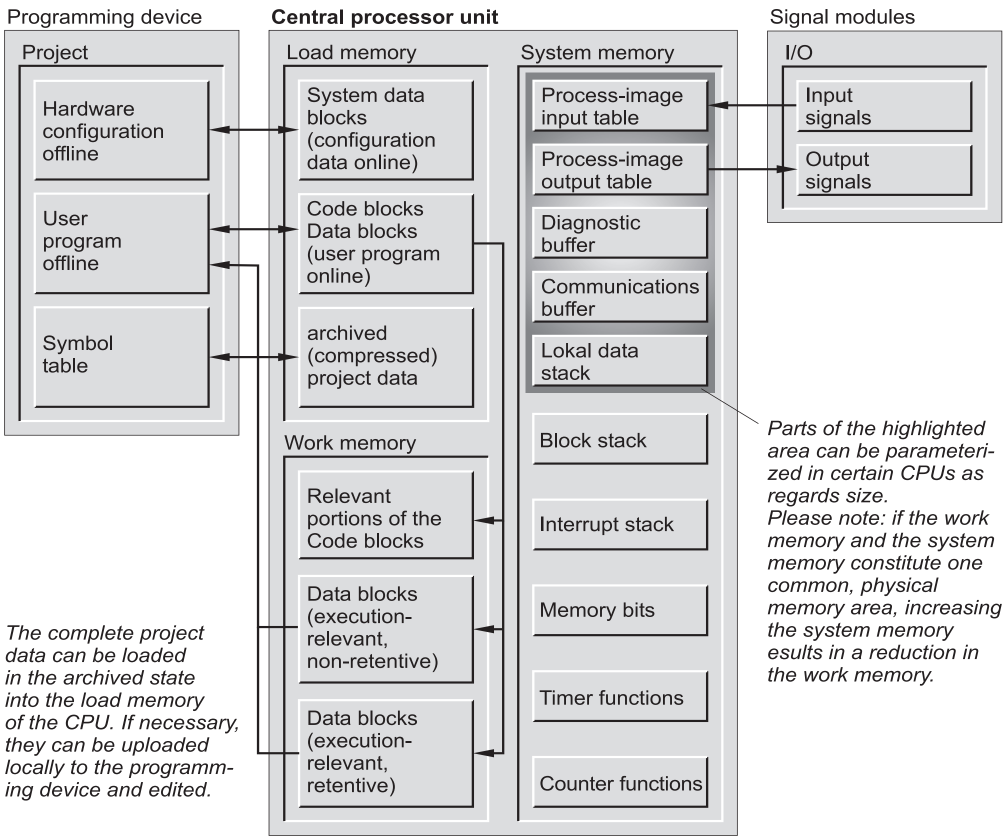

Programming device
Central processor unit
Signal modules
Project
Hardware configuration offline
User program offline
Symbol table
Load memory
System data blocks (configuration data online)
Code blocks Data blocks (user program online)
archived (compressed) project data
Work memory
Relevant portions of the Code blocks
Data blocks (execution-relevant, non-retentive)
Data blocks (execution-relevant, retentive)
System memory
Process-image input table
Process-image output table
Diagnostic buffer
Communications buffer
Lokal data stack
Block stack
Interrupt stack
Memory bits
Timer functions
Counter functions
I/O
Input signals
Output signals
Parts of the highlighted area can be parameterized in certain CPUs as regards size. Please note: if the work memory and the system memory constitute one common, physical memory area, increasing the system memory esults in a reduction in the work memory.
The complete project data can be loaded in the archived state into the load memory of the CPU. If necessary, they can be uploaded locally to the programming device and edited.

Figure 1.2 Memory areas on the CPU

Micro memory card

The memory submodule for the newer S7-300 CPUs is a micro memory card (MMC). The data on the MMC are non-volatile, but can be read, written and deleted just like with a RAM. This response permits data backup without a battery.

The MMC contains the complete load memory, so that an MMC is always required for operation. The MMC can be used as a portable storage medium for user programs or firmware updates. Using special system functions, you can read or write data blocks on the MMC from the user program, e.g. read recipes from the MMC, or create a measured-value archive on the MMC and provide it with data.

Load memory

The entire user program, including configuration data, is in the load memory (system data). From the programming device, the user program is always initially loaded into the load memory and from there into the work memory. The program in the load memory is not executed as the control program.

With a CPU 300 and a CPU ET200, the load memory is present completely on the micro memory card. Thus the contents of the load memory are retained even if the CPU is de-energized.

If the load memory with a CPU 400 consists of an integrated RAM or RAM memory card, a backup battery is required in order to keep the user program retentive. With an integrated

EEPROM or a plug-in flash EPROM memory card as the load memory, the CPU can be operated without battery backup.

From STEP 7 V5.1 and with appropriately equipped CPUs, you can save all the project data as a compressed archive file in the load memory (see Chapter 2.2.2 “Managing, Rearranging and Archiving”).

Work memory

Work memory is designed in the form of high-speed RAM fully integrated in the CPU. The operating system of the CPU copies the “execution-relevant” program code and the user data into the work memory. “Relevant” is a characteristic of the existing objects and does not mean that a particular code block will necessarily be called and executed. The “actual” control program is executed in the work memory.

Specific to the product, the work memory can be either a coherent area or divided according to program and data memories, where the latter is also divided into retentive and non-retentive parts.

When writing back the user program into the programming device, the blocks are fetched from the load memory, supplemented by the current values of the data addresses from the work memory (further information available in Chapters 2.6.4 “Loading the User Program into the CPU” and 2.6.5 “Block Handling”).

System memory

System memory contains the addresses (variables) that you access in your program. The addresses are combined into areas (address areas) containing a CPU-specific number of addresses. Addresses may be, for example, inputs used to scan the signal states of momentary-contact switches and limit switches, and outputs that you can use to control contactors and lamps.

The system memory on a CPU contains the following address areas:

▷ Inputs (I)
Inputs are an image (“process image”) of the digital input modules.

▷ Outputs (Q)
Outputs are an image (“process image”) of the digital output modules.

▷ Bit memories (M)
are information stores which are directly accessible from any point in the user program.

▷ Timers (T)
Timers are locations used to implement waiting and monitoring times.

▷ Counters (C)
Counters are software-level locations, which can be used for up and down counting.

▷ Temporary local data (L)
Locations used as dynamic intermediate buffers during block processing. The temporary local data are located in the L stack, which the CPU occupies dynamically during program execution.

The letters enclosed in parentheses represent the abbreviations to be used for the different addresses when writing programs. You may also assign a symbol to each variable and then use the symbol in place of the address identifier.

The system memory also contains buffers for communication jobs and system messages (diagnostics buffer). The size of these data buffers, as well as the size of the process input image, the process output image and the L stack, are parameterizable on certain CPUs.

Physical design of CPU memory

The physical design of the load memory differs according to the type of CPU (Figure 1.3).

A CPU 300 or CPU ET 200 does not have an integrated load memory. A micro memory card containing the load memory must always be inserted to permit operation. The load memory can be written and read like a RAM. The physical design means that the number of write operations is limited (no cyclic writing by user program). You can use the menu command COPY RAM TO ROM to transfer the current values of the data operands from the work memory to the load memory.

With a CPU 300 with firmware version V2.0.12 or later, the work memory for the user data consists of a retentive part and a non-retentive part.