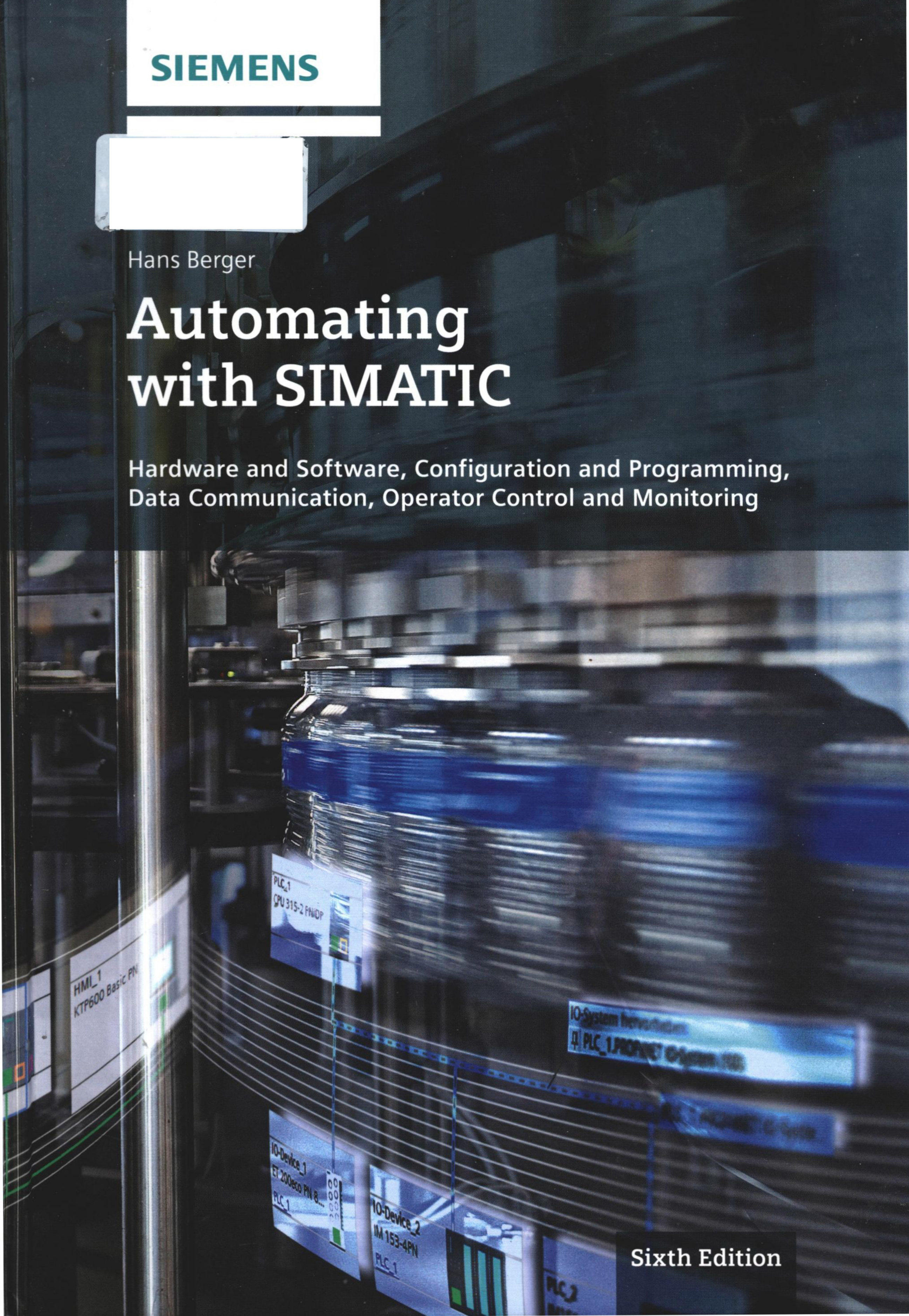
SIEMENS

Hans Berger

Automating with SIMATIC

Hardware and Software, Configuration and Programming, Data Communication, Operator Control and Monitoring



Sixth Edition

Berger Automating with SIMATIC

Automating with SIMATIC

Hardware and Software,
Configuration and Programming,
Data Communication, Operator Control
and Process Monitoring

by Hans Berger

6th revised and enlarged edition, 2016

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.

Print ISBN: 978-3-89578-459-0
ePDF ISBN: 978-3-89578-943-4

6th edition, 2016

Editor: Siemens Aktiengesellschaft, Berlin and Munich
Translation: Siemens AG, Business Process Shared Services, Erlangen
Publisher: Publicis Publishing, Erlangen, Germany
© 2016 by Publicis Pixelpark Erlangen – eine Zweigniederlassung der Publicis Pixelpark GmbH

Foreword

The automation of industrial plants results in a growing demand for components which are increasingly different and more complex. Therefore a new challenge nowadays is not the further development of highly specialized devices but the optimization of their interaction.

The *Totally Integrated Automation* concept permits uniform handling of all automation components using a single system platform and tools with uniform operator interfaces. These requirements are fulfilled by the SIMATIC industrial automation system, which provides uniformity for configuration, programming, data management, and communication.

The STEP 7 engineering software is used for the complete configuration and programming of all components. Optional packages for expanding functionalities can also be introduced seamlessly in STEP 7 if they have the same operating philosophy. The SIMATIC Manager of STEP 7 V5.5 and the TIA Portal of STEP 7 V13 coordinate all tools and centrally manage any automation data. All tools have access to this central data management so that duplicate entries are avoided and coordination problems are prevented right from the start.

Integrated communication between all automation components is a prerequisite for "distributed automation". Communication mechanisms that are tuned to one another permit the smooth interaction of controllers, visualization systems, and distributed I/O without additional overhead. This puts the seminal concept of "distributed intelligence" within reach. Communication with SIMATIC is not only uniform in itself, it is also open to the outside. This means that SIMATIC applies widely-used standards such as PROFIBUS for field devices and Industrial Ethernet and TCP/IP protocol for the best possible connections to the office world and thus to the management level.

The 6th edition of this book provides an overview of the structure and principle of operation of a modern automation system with its state-of-the-art controllers and HMI devices, and describes the expanded facilities of distribution with PROFIBUS and PROFINET. Using the SIMATIC S7 programmable controllers as example, this book provides an insight into the hardware and software configuration of the controller, presents the programming level with its various languages, explains the exchange of data over networks, and describes the numerous possibilities for operator control and monitoring of the process.

Erlangen, April 2016 Hans Berger

Contents

1 Introduction

1.1 Components of the SIMATIC Automation System

The SIMATIC automation system consists of many components that are matched to each other through the concept of "Totally Integrated Automation" (TIA). Totally Integrated Automation means automation with integrated configuration, programming, data storage, and data transfer (Fig. 1.1).

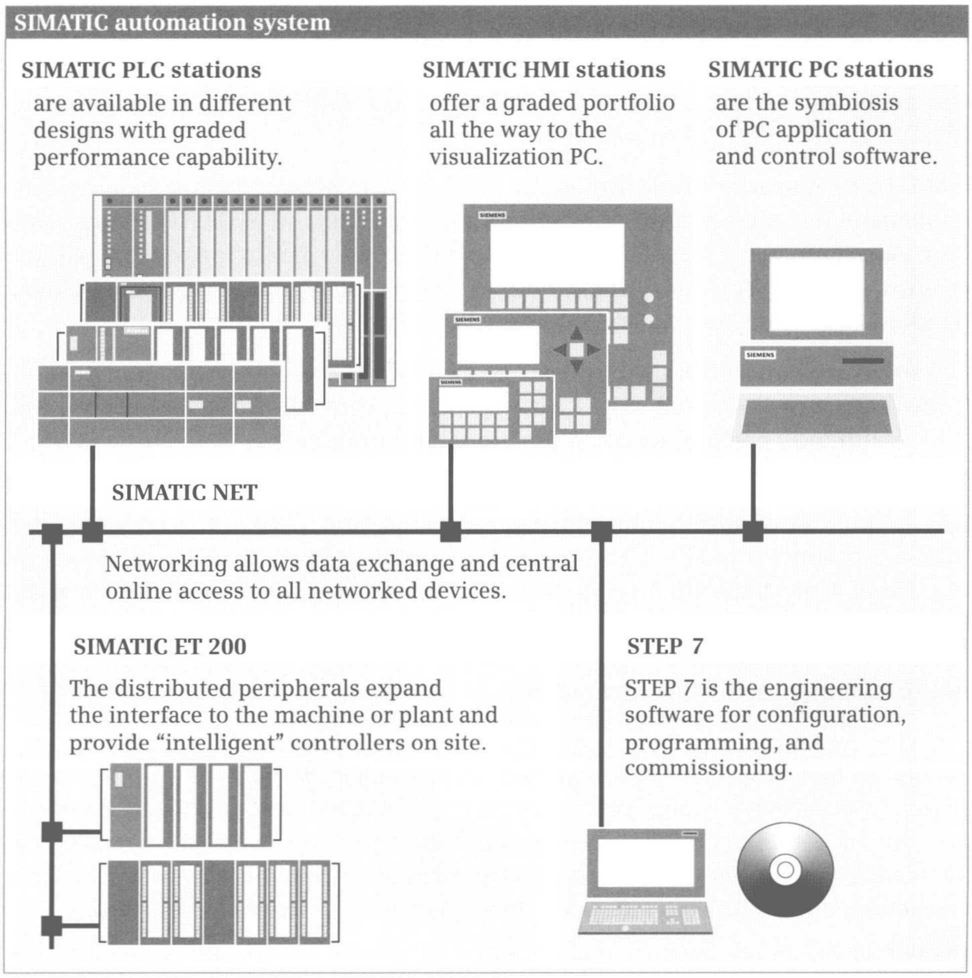

SIMATIC automation system
SIMATIC PLC stations
are available in different designs with graded performance capability.
SIMATIC HMI stations
offer a graded portfolio all the way to the visualization PC.
SIMATIC PC stations
are the symbiosis of PC application and control software.
SIMATIC NET
Networking allows data exchange and central online access to all networked devices.
SIMATIC ET 200
The distributed peripherals expand the interface to the machine or plant and provide "intelligent" controllers on site.
STEP 7
STEP 7 is the engineering software for configuration, programming, and commissioning.

Fig. 1.1 Components of the SIMATIC automation system

As programmable logic controllers (PLCs), the **SIMATIC S7 Controllers** form the basis of the automation system. SIMATIC S7-1200 Basic Controllers are the ideal choice for simply and autonomously resolving tasks in the low to medium performance range. The SIMATIC S7-1500 Advanced Controllers ensure maximum performance capability and maximum user friendliness for medium and high-end applications in machine and plant automation. The characteristics of the SIMATIC S7-300 controller are especially aimed at innovative system solutions in the manufacturing industry – e.g. in the automotive or packaging industry. Within SIMATIC, the S7-400 Controllers are designed for system solutions in factory and process automation and are especially suitable for data-intensive tasks, for example in the process industry. All controllers can be modularly expanded with signal, technology and communication modules.

SIMATIC ET 200 expands the interface between the central controller and the machine or system by I/O modules directly on-site. If autonomously functioning, "intelligent" controllers are needed at the field level, the **Distributed Controllers ET 200** CPU are used. SIMATIC ET 200 provides a multi-functional, modular and finely scalable system for the distributed automation: For solutions in the control cabinet or without a control cabinet directly at the machine and for use in hazardous areas. All of the products can be integrated into the automation via the PROFIBUS or PROFINET bus systems.

SIMATIC PC-based automation is the ideal basis to efficiently and economically implement the combination of Windows applications and SIMATIC control software. The industrial PCs of the **SIMATIC IPC** product family provide maximum performance with the latest Intel processor technology, pre-installed and enabled Windows operating system, and integrated communication interfaces.

The **software controllers** with the scope of services of the S7-1500 family of controllers are especially well-suited for the flexible control of special-purpose machines with high performance and functional requirements.

SIMATIC HMI stands for Human Machine Interface. The SIMATIC Panels are available in numerous different performance classes and permit the efficient, machine-level control of the machine or plant. As ideal basis for HMI solutions on PCs, Panel PC systems offer a comprehensive and finely graded portfolio. Powerful HMI software indicates the state of the plant with event and fault messages, manages recipes and measured value archives, and supports plant operators with troubleshooting, servicing, and maintenance.

SIMATIC NET links all SIMATIC stations and ensures trouble-free data communication. Various bus systems with graded performance also allow third-party devices to be connected, whether field devices in the plant or process PCs connected at the control level. Data traffic can go beyond the limits of various subnets, such as the transfer of automation data such as measured values and alarms or the commissioning and troubleshooting of a central location in the network group.

STEP 7 is the engineering software which is used to configure, parameterize, and program all SIMATIC components. The "classic" version of STEP 7 with the SIMATIC Manager or the innovated version of STEP 7 inside TIA Portal are the central tools for managing automation data and related software editors in the form of a hierarchically organized project.

The main activities performed with STEP 7 are:

▷ Configuring the hardware
(arranging modules in racks and parameterizing the module properties)

▷ Configuring the communication connections
(defining communication partners and connection properties)

▷ Programming the user program
(creating the control software and testing the program)

The user program can be created in the programming languages ladder logic (LAD), function block diagram (FBD), statement list (STL), and structured control language (SCL) as “logic control”, or in the programming language GRAPH as “sequence control”

1.2 From the Automation Task to the Finished Program

If the machine to be controlled is a small one, will an S7-1200 be big enough or do you need an S7-1500? Is it better to control the plant with an S7-400 or with a pair of S7-300s? Compact central I/Os in the control cabinet or distributed I/Os in the plant?

The following is a general outline of the steps that lead from the automation task to the finished program. In individual cases, specific requirements must be met.

Choosing the hardware

There are many criteria for selecting the type of controller. For “small” controls the main criteria are the number of inputs and outputs and the size of the user program. For larger plants you need to ask yourself whether the response time is short enough, and whether the user memory is big enough for the volume of data to be managed (recipes, archives). To be able to estimate the resources you need from the requirements alone, you need a lot of experience of previous automation solutions; there is no rule of thumb.

A production machine will probably be controlled by a single station. In this case, the number of inputs/outputs, the size of the user memory and, possibly, the speed (response time) will enable you to decide between the S7-1200, S7-1500, S7-300, or the S7-400. How is the machine controlled? What HMI devices will be used?

Spatially distributed systems raise the question of what is overall the better value: the use of centralized or distributed I/O. In many cases distributed I/O not only reduces the wiring overhead needed, but also the response time and the engineering costs. This is possible due to the use of “intelligent” I/O devices with their own user program for preprocessing of signals “on site”.

Distributed automation solutions have advantages: The user programs for the different plant units are smaller with faster response times, and can often be commissioned independently of the rest of the plant. The necessary exchange of data with a “central controller” is particularly easy within the SIMATIC system using standardized bus systems.

Which programming language?

The choice of programming language depends on the task. If it mainly consists of binary signal processing, the graphical programming languages LAD (Ladder Logic) and FBD (Function Block Diagram) are ideal. For more difficult tasks requiring complex variable handling and indirect addressing, you can use the STL (Statement List) programming language, which has an assembly language format. SCL (Structured Control Language) is the best choice for people who are familiar with a high-level programming language and who mainly want to write programs for processing large quantities of data.

If an automation task consists of sequential processes, GRAPH can be used. GRAPH creates sequencers with steps and step enabling conditions that are processed sequentially. All programming languages – including GRAPH – can be used together in a user program. Every program section and “block” can be created with the suitable programming language, depending on requirements.

Creating a project

All the data for your automation solution is collected together in a “project”. You create a project using STEP 7. A project is a (software) folder in which all the data is stored in a hierarchic structure. The next level down from the project are the “stations”, which in turn contain one or more CPUs with a user program. All these objects are folders which can contain other folders or objects that represent the automation data on the screen. You use menu commands to insert new objects, open these objects, and automatically start the tool required to work with them.

An example: The user program consists of blocks, which are individual program sections with limited functions. All programmed blocks are listed in the block folder. Depending on the programming language used, double-clicking on a block starts the suitable program editor, with which you can alter or expand the program in the block, guided by menus and supported by online help.

Configuring hardware

A project must contain at least one station (one device), either a programmable logic control (PLC) station, a human-machine interface (HMI) station, or a personal computer (PC) station. A PLC station is required to control a machine or plant. After the station is opened, a rack is shown onscreen, to which you can add the desired modules. To do this, drag the required modules from the hardware catalog to the relevant slot. If needed, change the default module properties to meet your requirements.

A project can contain additional stations that you configure in the same manner as the first station. The data transfer between the stations takes place via a subnet. Using network configuration, you connect the bus interfaces of the "communication modules" to the subnet and thus create the network group.

Writing, debugging and saving the user program

The user program is the totality of all instructions and declarations programmed by the user for signal processing by means of which the machine or plant to be controlled is influenced in accordance with the control task. Large, complex tasks are easier to solve if they are divided into small, manageable units, which can be programmed in "blocks" (subroutines). The division can be process-oriented or function-oriented. In the first case, each program unit corresponds to a part of the machine or plant (mixer, conveyor belt, drilling assembly). In the second case, the program is divided up according to control functions, for example signaling, communication, operating modes. In practice, mixed forms of the two structuring concepts are generally used.

In the user program, signal states and variable values are used that you should preferably address with a name (symbolic addressing). A name is assigned to a memory location in the symbol table or in the PLC tag table. You can then use the name in the program. After you have entered the user program you "compile" it so that it can process the relevant control processor. The user program is created "offline", without a connection to a controller, and is saved to the hard disk of the programming device.

You can test smaller programs, as well as individual parts of larger programs, offline with the PLCSIM simulation software and thus find and correct any possible errors before the user program is used in the machine or plant.

For commissioning, connect the programming device with the CPU, transfer the program to the CPU user memory, and test it using the STEP 7 testing functions. You can monitor and change the variable values and monitor the processing of the program by the control processor. Comprehensive diagnostic functions allow quick identification of error location and cause.

After a successful commissioning, you can document the project as a printable circuit manual and archive it as a zipped file.

1.3 How Does a Programmable Logic Controller Work?

In conventional control engineering, a control task is solved by wiring up contactors and relays individually, i.e. depending on the task. They are therefore referred to as contactor and relay controllers, and electronic controllers assembled from individual components are referred to as *hard-wired programmed* controllers. The "program" is in the wiring. *Programmable logic* controllers, on the other hand, are made up of standard components that implement the desired control function by means of a userprogram.

SIMATIC S7 is an automation system that is based on programmable logic controllers. The solution of the control task is stored in the user memory on the CPU in the form of program instructions. The control processor reads the individual instructions sequentially, interprets their content, and executes the programmed function.

The CPU module contains an additional program: the operating system. It ensures the execution of the device-internal operating functions, such as communication with the programming device and backing up data in the event of power failure. The operating system also initiates the processing of the user program, either recurring cyclically or dependent on a trigger event such as an alarm (Fig. 1.2).

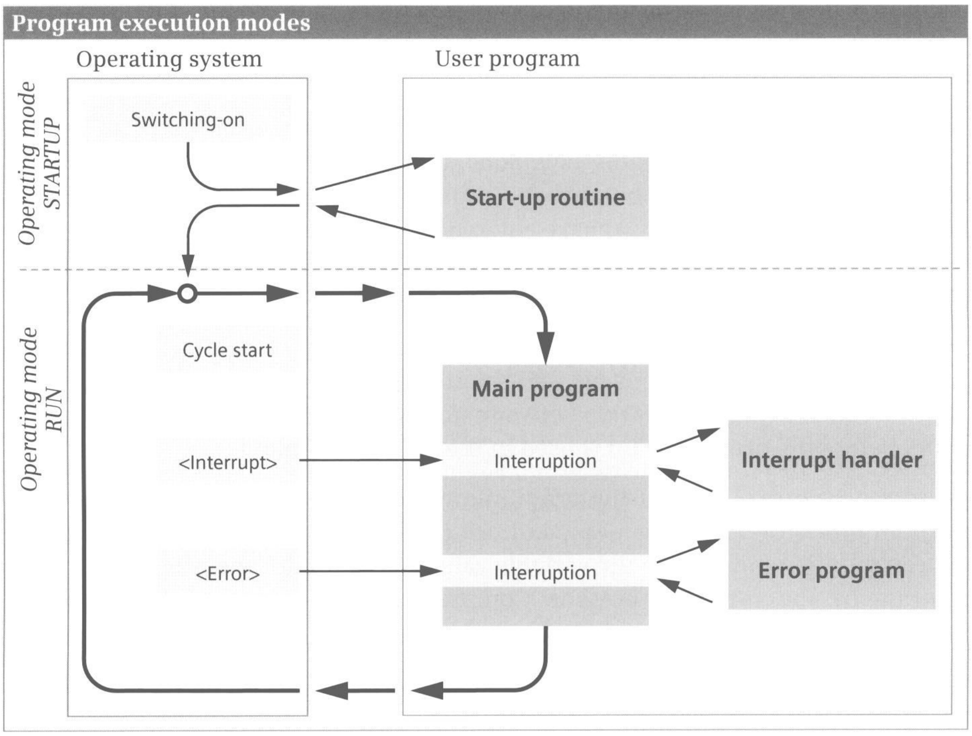


Fig. 1.2 Execution of the user program in a SIMATIC controller

Cyclic program processing

The prevalent processing type of the user program for programmable logic controllers is cyclic program processing: After the user program has been completely processed once, it is then processed again immediately from the beginning. The user program is also executed if no actions are requested "from outside", such as if the controlled machine is not running. This provides advantages when programming: For example, you program the ladder logic as if you were drawing a circuit diagram, or program the function block diagram as if you were connecting electronic components. Roughly speaking, a programmable logic controller has characteristics like those of a contactor or relay control: The many programmed operations are effective quasi simultaneously "in parallel".

After the power is switched on and the operating function test runs, the operating system starts an (optional) start-up routine once. The main program is next in the sequence. If it has been processed to the end, processing begins again immediately at the start of the program. The main program can be interrupted by alarm or error events. The operating system then starts an interrupt handler or error program. If the interruption-controlled program has been completely processed, the program processing continues from the point of interruption in the main program. A priority scheduler controls the program execution order if several interrupt events occur simultaneously.

The user program is made up of blocks. There are several block types. Organization blocks represent the interface to the operating system. After a start event occurs (power up, cycle start, alarm, error), the operating system calls the associated organization block. It contains the appropriate user program for the event. An organization block only has to be programmed if the automation solution requires it. The program in an organization block can, if needed, be structured by function blocks (blocks with static local data) and functions (blocks without static local data). Data blocks in the user memory are available to store user data.

1.4 The path of a binary signal from the sensor to the program

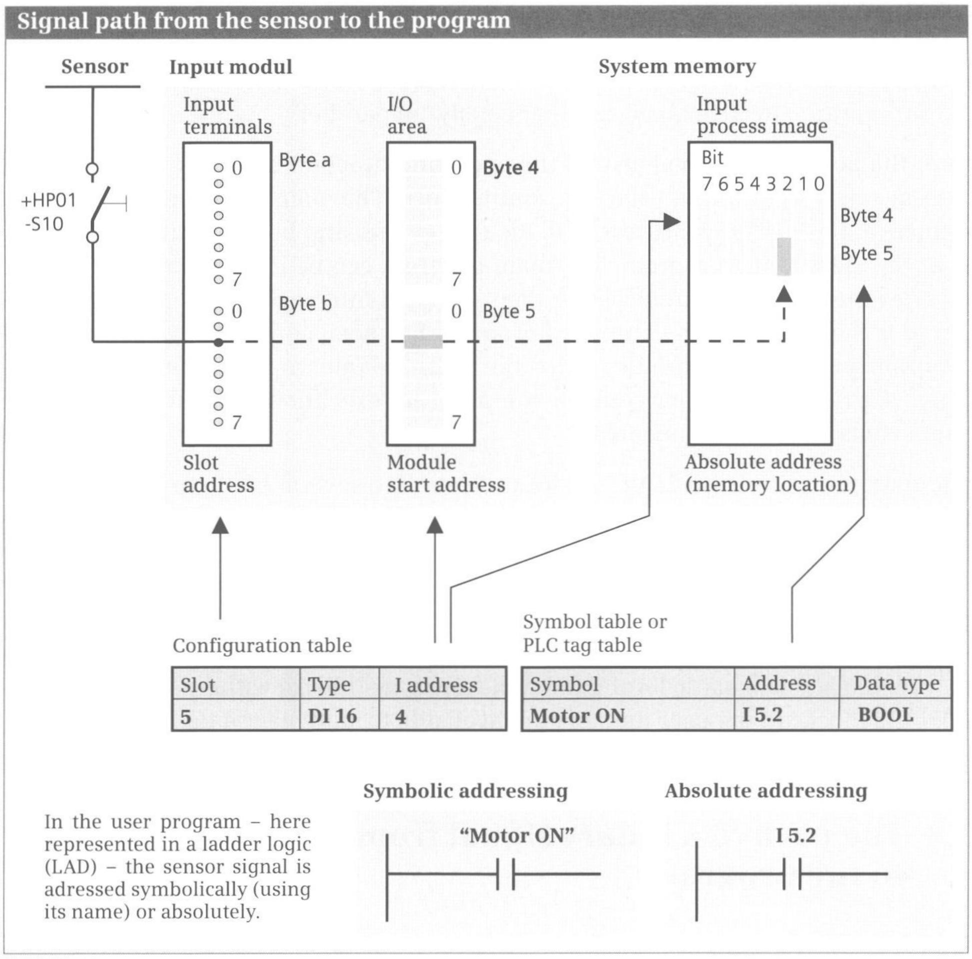
In order to do its job, the control processor in the controller needs to be connected to the machine or plant it is controlling. I/O modules that are wired to the sensors and actuators create this connection.

Connection to the programmable logic controller, module address

When wiring the machine or plant, you define which signals are connected to the programmable logic controller, and where. An input signal, e.g. the signal from pushbutton +HP01-S10 with the significance "Switch on motor", is connected to a specific terminal on an input module (Fig. 1.3).

Each module is located in a particular slot on the rack, the number of which is the slot address. In addition, each I/O module has a so-called I/O address. In the I/O address, the binary signals are aggregated into bytes (bundles of eight bits). Bytes are numbered starting from zero – even with gaps. The bit address is counted from 0 to 7 for each byte.

You determine the slot address by plugging the module into a certain place on the rack. STEP 7 assigns the I/O address consecutively, which you can change in the configuration table. In the connection diagram, the module start address is identified with "Byte a". If a module has more bytes, the byte addresses are automatical-



Configuration table

Slot	Type	I address
5	**DI 16**	**4**

Symbol table or PLC tag table

Symbol	Address	Data type
Motor ON	**I 5.2**	**BOOL**

In the user program – here represented in a ladder logic (LAD) – the sensor signal is adressed symbolically (using its name) or absolutely.

Fig. 1.3 Path of a signal from the sensor to its use in the program

ly incremented and are assigned the designations "Byte b", "Byte c", and so on. The "Motor ON" signal is connected to terminal two of the second byte (Byte b). By specifying the module address as 4, you are given the address of the signal: "Input in byte 5 to bit 2" or in short: I 5.2.

Symbolic address

The address "I 5.2" denotes the memory location and is the absolute address. It is much easier if you can address this signal in the program with a name that matches the meaning of the signal, for example "switch on motor". This is the symbolic address. You can find the assignment of absolute addresses to symbolic addresses in the symbol table or the PLC tag table. In this table, the "global" symbols are defined; these are the symbols that are valid in the entire user program. You specify the symbols that are valid for only one block ("local" symbols) when programming the block.